Product Forms for FCFS Queueing Models with Arbitrary Server-Job Compatibilities: An Overview

Kristen Gardner and Rhonda Righter

September 2020

Abstract

In recent years a number of models involving different compatibilities between jobs and servers in queueing systems, or between agents and resources in matching systems, have been studied, and, under Markov assumptions and appropriate stability conditions, the stationary distributions have been shown to have product forms. We survey these results and show how, under an appropriate detailed description of the state, many existing product-form results are corollaries of similar results for the Order Independent Queue. We also discuss how to use the product form results to determine distributions for steady-state response times.

1 Introduction

Systems in which servers are flexible in the types of customers that they can serve, and customers are flexible in the servers at which they can be processed, are very common in a wide range of practical settings. In call centers, service representatives may be trained to handle different subsets of requests, or may speak different languages. A customer who speaks only Spanish can be helped by a representative who speaks only Spanish, or by a representative who speaks both Spanish and English, or by a representative who speaks both Spanish and Mandarin. In computer systems, some jobs may be able to run only on those servers that have the job's data stored locally, other jobs may require a server with a particular combination of resources, and still other jobs may be able to run on any server. In ride-sharing systems, drivers will only be assigned to users that are "nearby" in some sense.

This type of model is called a skill-based server model in the call center literature. In the scheduling literature, the compatibility constraints between job classes and servers are called eligibility constraints or processing set restrictions, and the models are typically deterministic. For matching models, compatibilities may be location based. While the language and notation used to describe these models differ across research communities, the common idea in all of the above examples is that the system consists of multiple servers and multiple classes of jobs, with a bipartite graph structure indicating which classes of jobs can be served by which servers.

The examples above, and more broadly the "flexible job/server" models that exist in the literature, vary in precisely how the bipartite matching structure is used to assign servers to jobs. We mainly consider two service models, which we call the "collaborative" and "noncollaborative" models. In the collaborative model, multiple servers can work together, with additive service rate, to process a single job. This matches the computer systems setting, in which the same (replicated) job can run on several different servers at once. In the noncollaborative model, a customer can only enter service at a single server. This matches the structure of a call center, in which a single customer cannot speak with multiple representatives at the same time. In both cases, we think of there being a single central queue for all customers. When a server becomes available, it begins working on the next compatible job in the queue, in first-come first-served (FCFS) order. In the noncollaborative case, we must also specify which server will serve an arriving job that finds multiple idle compatible servers. We will consider two policies: Assign Longest Idle Server (ALIS), which is analogous to FCFS, and Random Assignment to Idle Servers (RAIS).

An additional feature of many models of service systems with job/server compatibilities is redundancy, or job replication, i.e., the possibility of sending multiple copies of the same job to multiple servers. For

example, this is a common practice in computer systems to combat unpredictable system variability, so the hope is that the job may experience a significantly shorter response time at one of the servers. Similarly, one idea for reducing wait times on organ transplant waitlists is to allow patients to join the waitlist in multiple geographic areas at the same time. Patients are restricted in which waitlists they can join based on travel time: should an organ become available at a particular hospital, the patient must be able to travel to that hospital within a relatively short time frame to receive the transplant. Generally systems with redundancy are not modeled as a central FCFS queue as described above. Instead, each server has its own dedicated queue and an arriving job can join the queues of multiple servers. In the collaborative case, multiple copies of the same job can run on different servers at the same time, and when the first copy completes service all other copies are removed immediately from other servers or queues. This is called cancel-on-completion or late cancellation. In the noncollaborative case, all other copies of a job are removed from the system as soon as the first copy enters service. This is called cancel-on-start or early cancellation, and it is also equivalent to sending a single copy to the queue with the least work. In both cases, the cancellations occur without penalty. While the central FCFS queue and the job redundancy model describe very different system dynamics, the two views turn out to be sample-path equivalent, provided that service times are exponentially distributed and i.i.d. across jobs and servers. We will explore this relationship, as well as other model equivalences, in what follows.

Throughout most of this paper, we will make a few key assumptions: that jobs of each class arrive according to independent Poisson processes, that service times are exponentially distributed and i.i.d. across jobs and servers, and that the scheduling discipline is FCFS. Under these assumptions, we will see that the models described and motivated above, as well as related models, exhibit product-form stationary distributions. Indeed, product forms hold for several different state descriptors, each of which provides different advantages in understanding system behavior. We first consider the most detailed state descriptor, which tracks the classes of all jobs in the system. This description lends itself to a concise proof of the product form for the collaborative model, due to the Order Independence (OI) results of Berezner and Krzesinski [14, 39]. We show that for the noncollaborative model, both the job queue and the idle server queue are OI queues, resulting in a product of product forms. We extend these arguments to collaborative and noncollaborative models with abandonments. We also show that the same product-form stationary distribution holds for several related models, including new results for two-sided matching models with arrivals of both jobs and servers, and for make-to-stock inventory models with back ordering.

Following the development of the product-form stationary distributions, we turn to using these results to derive system performance metrics. We begin with class-based response time distributions. We show that if there is a job class that is compatible with all servers, that class has an exponentially distributed response time for the collaborative model; indeed, the response time for that class is the same as it would be for the M/M/1 queue in which all jobs are fully flexible. For the noncollaborative model, the queueing time for that fully flexible class is a mixture of a mass at 0 and an exponential random variable. We use this result to show response time distributions for all job classes in the collaborative model, and queueing time distributions for all classes in the noncollaborative model, when the compatibility matching has a nested structure.

Product-form distributions for an alternative, partially aggregated state descriptor have been derived in the literature; we show that these results also follow as corollaries to the detailed product forms. The partially aggregated state description allows us to derive per-class response time distributions, conditioned on the set of busy servers and the order of the jobs they are currently serving.

We briefly discuss a related queueing model in which the state description is the number of jobs of each class in the system (per-class aggregation). While this state space no longer yields a Markovian description of the system evolution, it has the same steady-state per-class mean performance measures (mean number in system, probability the system is empty) as the collaborative system. This state descriptor yields a simple, recursive approach to derive the system load and mean response time for our models when they are not nested.

We note that the product forms discussed in this paper are not the same as those obtained in the well-known Jackson and Kelly networks [35, 37]. The standard Jackson and Kelly product forms arise in networks of queues, where the state of the network can be expressed as a product of the states at each queue. In contrast, in this paper we primarily concentrate on the internal product-form structure of the steady-state distribution for a single queue. While most of our focus is on single nodes with flexible jobs and servers, in the collaborative case, these nodes are quasi-reversible under our modeling assumptions, so a network of

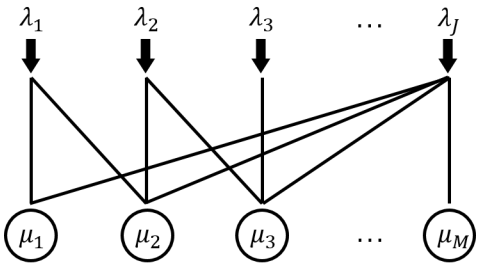


Figure 1: The system consists of J classes of jobs, M servers, and a bipartite matching structure indicating which job classes can be served by which servers.

such nodes also has a product form stationary distribution. That is, in steady state the distributions of each node will be as if they were operating independently, as is the case in Jackson and Kelly networks.

After describing our models and notation in the next section, in Section 3 we review the proof for the product-form stationary distribution for the detailed state description of the Order Independent queue, and show how this result can be extended to collaborative and noncollaborative queueing models; we also show generalizations to other related systems. In Section 4 we show how response time distributions can be obtained from the product-form results of Section 3 for the special case of nested systems. We consider partially aggregated state descriptions in Section 5; these give us partial, conditional, results for response times. We discuss methods of computing mean performance measures using a per-class aggregated state description in Section 6. We end the paper with further discussion of related work and concluding thoughts. Throughout the paper we provide pointers to the relevant literature in context.

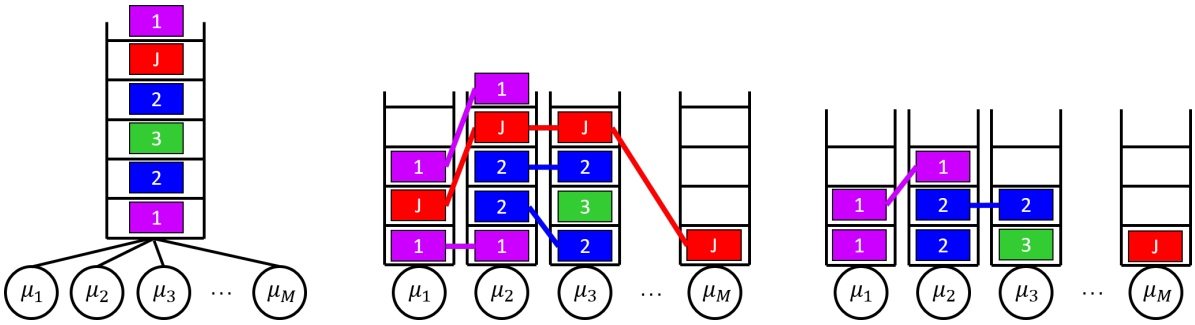
2 Model

We note at the outset that our analysis requires a heavy dose of notation that we will often reuse and abuse in the interest of readability and ease of understanding. Notation that we use throughout the paper is summarized in Table 1.

We have a set of J job classes with Poisson arrivals at rates λ_i, and a set of M parallel servers with exponential service rates μ_m, and a bipartite graph matching structure indicating which servers can serve which job classes (see Figure 1). For job class i, let $S_i = \{j : \text{server } j \text{ can serve class } i\}$, and for any set of job classes, A, let $S(A) = \bigcup_{i \in A} S_i$ be the set of servers that can serve any job class in set A. For example, for the system shown in Figure 1, $S_1 = \{1, 2\}$ and for $A = \{1, 2\}$, $S(A) = \{1, 2, 3\}$. For server j, let $C_j = \{i : \text{server } j \text{ can serve class } i\}$ be the set of job classes it can serve, and for any set of servers, B, let $C(B) = \bigcup_{j \in B} C_j$ be the set of job classes that can be served by servers in B. For example, in Figure 1, $C_3 = \{2, 3, J\}$ and for $B = \{1, 3\}$, $C(B) = \{1, 2, 3, J\}$. For a subset of job classes, A, let $\mu(A) = \sum_{m \in S(A)} \mu_m$ and $\lambda(A) = \sum_{i \in A} \lambda_i$ be the total service rate and arrival rate for job classes in A, and, abusing notation, for a subset of servers, B, let $\mu(B) = \sum_{m \in B} \mu_m$ and $\lambda(B) = \sum_{i \in C(B)} \lambda_i$ be the total service rate and arrival rate for servers in B. It will be clear from the context whether the arguments of λ and μ are job classes or servers. Finally, let $\mu = \sum_{m=1}^{M} \mu_m$ and $\lambda = \sum_{i=1}^{J} \lambda_i$ be the total system service rate and total system arrival rate respectively.

Throughout, we will assume for stability that $\lambda(A) < \mu(A)$ for all subsets of job classes A. We note that this condition is both necessary and sufficient for stability in the model described above [1, 25, 45].

We primarily consider two models of service: the collaborative model and the noncollaborative model. In the noncollaborative model, a job can only be served by a single server. When the first copy of a job enters service, all other copies are removed from the system immediately without penalty. A job that arrives to the system and finds multiple idle compatible servers begins service on one of those servers, chosen according to some assignment rule. We consider two assignment rules. Under Assign Longest Idle Server (ALIS), the arriving job begins service on the compatible server that has been idle for the longest time. Under Random Assignment to Idle Servers (RAIS), the arriving job chooses an idle server randomly; this selection must be drawn from a particular distribution, which we discuss in more detail in Section 3.3.



(a) Central FCFS queue (b) Collaborative model (c) Noncollaborative model

Figure 2: The system can be viewed, equivalently, as (a) having a single FCFS queue, or as being a distributed system in (b) the collaborative model or (c) the noncollaborative model.

In the collaborative model, a job may be in service at multiple servers at the same time. When the first copy of a job completes service, all other copies are removed from the system immediately without penalty. A job that is in service at a set of servers B receives combined service rate $\mu(B)$, hence the job experiences an exponential service time with rate $\mu(B)$. Unlike in the noncollaborative case, no assignment rule is needed for an arriving job that finds multiple idle compatible servers; such a job simply enters service on all of the idle compatible servers. Note that in the collaborative case (but not in the noncollaborative case), we can assume without loss of generality that the set of job classes a server can serve is unique to that server, i.e., $C_i \neq C_j$ for $i \neq j$. This follows because of the FCFS and collaborative assumptions; if $C_i = C_j$ for $i \neq j$, then servers i and j will always be serving the same (oldest compatible) job, so they can be considered to be a single server with rate $\mu_i + \mu_j$. In the noncollaborative model we allow multiple servers that are identical in their service rates and their sets of compatible job classes.

There are two equivalent ways of viewing the system dynamics. In the first, shown in Figure 2(a), all arriving jobs join a single FCFS queue. When a server j becomes available, it begins working on the first job in the central queue that has class $i \in C_j$. In the collaborative model, the "queue" contains all jobs in the system, including those currently in service, so that a newly available server may begin working on a job that is already in service at some other server. In the noncollaborative model, the queue contains only those jobs that are not in service. The second system view is that of a distributed system, in which each server has its own queue and works on the jobs in its queue in FCFS order. Here, an arriving job of class-i joins the queue at all servers in S_i. In the collaborative model (Figure 2(b)), multiple copies of the job may be in service at different servers. For example, in Figure 2(b) the class-1 job shown at the head of the queue at both servers 1 and 2 is in service at both servers. In the noncollaborative model (Figure 2(c)), only one copy of a job can be in service. In our example, the class-1 job shown at the head of the queue at server 1 is in service at server 1, and its other copy has been removed from the queue at server 2. Another equivalent model in the noncollaborative case is to assume, again, that each server has its own dedicated queue, and that each arriving class-i job is routed to the server in S_i with the least work (i.e., Join-the-Shortest-Work among compatible servers) [7, 10, 11].

Throughout the remainder of this paper, we will rely primarily on the central-queue view of the system when developing our state descriptors. We introduce here the notation used in the state descriptors. This notation captures a great deal of information about the system, and each state descriptor uses a slightly different subset of this information to capture different aspects of the system dynamics. We elaborate further on the specific state descriptors in the sections that follow. Let $\vec{c}_n = (c_1, \ldots, c_n)$ denote the classes of all jobs in the central queue, where c_i is the class of the ith job in the queue in order of arrival (so c_1 is the class of the oldest job, and c_n is the class of the most recent arrival). As noted above, for the collaborative model the "queue" refers to all jobs in the system, including those in service, whereas for the noncollaborative model the "queue" refers to only those jobs that are not in service. Let $\vec{b}_l = (b_1, \ldots, b_l)$ be the vector of busy servers in the arrival order of the jobs that they are serving (so b_1 is serving the oldest job in the system, and b_l is serving the most recent arrival among the jobs in service). We use $\vec{z}_m$ to denote, in the noncollaborative model, an interleaving of $\vec{c}_n$ and $\vec{b}_l$ ordered by job arrival time, where the state tracks the job class for

Notation	Definition
J	Number of job classes
M	Number of servers
λ_i	Arrival rate of class-i jobs
$\lambda = \sum_i \lambda_i$	Total system arrival rate
μ_j	Service rate at server j
$\mu = \sum_j \mu_j$	Total system service rate
S_i	The set of servers that can serve class-i jobs
$S(A)$	The set of servers that can serve any job class in subset A
C_j	The set of job classes that can be served by server j
$C(B)$	The set of job classes that can be served by any server in subset B
$\vec{c}_n$	Classes of all jobs in the queue, in order of arrival
$\vec{b}_l$	Busy servers, in the order in which they became busy
$\vec{s}_k$	Idle servers, in the order in which they became idle
$\vec{d}_l$	Classes of jobs in service, in the order in which they entered service
$\vec{n}_l$	Number of jobs not in service in between consecutive jobs in service
$\vec{x}_J$	Number of jobs of each class in the system
$\vec{z}_m$	Interleaving of jobs in the queue and busy servers, in order of job arrival times

Table 1: Summary of notation. Top section: system notation. Bottom section: notation used in state descriptors.

positions corresponding to jobs in the queue, and it tracks the busy server for positions corresponding to jobs in service. Let $\vec{s}_k = (s_1, \ldots, s_k)$ be a vector of idle servers in the order in which they became idle, where $l + k = M$. We use $\vec{d}_l = (d_1, \ldots, d_l)$ to denote the classes of the jobs currently in service, where d_i is the class of the job in service at server b_i. The vector $\vec{n}_l = (n_1, \ldots, n_l)$ denotes the number of jobs waiting to be served "between" the jobs in service. That is, n_i gives the number of jobs that arrived after the job in service at server b_i and before the job in service at server b_{i+1}. Finally, $\vec{x}_J = (x_1, \ldots, x_J)$ denotes the number of jobs of each class in the system; x_i is the number of class-i jobs in the system.

3 Detailed States and Product Forms

We first consider the most complete descriptions of the state: the detailed state descriptor tracks the classes of all jobs in the order of their arrival, denoted by $\vec{c}_n$. In the noncollaborative case, the two assigment rules that we consider (ALIS and RAIS) also require us to track some information about the servers. Under ALIS (Section 3.2), the state descriptor includes the vector $\vec{s}_k$, which tracks all idle servers in the order in which they became idle. Under RAIS (Section 3.3), the state descriptor is $\vec{z}_m$, which is an interleaving of $\vec{c}_n$ and $\vec{b}_\ell$, where $\vec{b}_l$ tracks all busy servers ordered by the arrival times of the jobs they are serving.

For both the collaborative and noncollaborative (ALIS and RAIS) models, we show that the stationary distribution for the above state descriptor exhibits a product form. We begin with the collaborative case, which is a special case of what are known as "Order Independent"(OI) queues, so named because the total service rate given to all jobs in the queue depends only on their classes, not on their order.

At the end of the section, we discuss related models that also have product-form stationary distributions.

3.1 The Collaborative Model and Order Independent Queues

The system state is $\vec{c}_n = (c_1, \ldots, c_n)$, where c_i is the class of the i'th job in the system in order of arrival, including both jobs that are in the queue and jobs that are in service (possibly at more than one server). The subscript n can take on the values $0, 1, ...$; we will generally leave this implicit. Let $\mathcal{C}$ be the set of all such states. Abusing notation, let $S(\vec{c}_n) = S(\{c_1, \ldots, c_n\})$ be the set of servers that can serve at least one

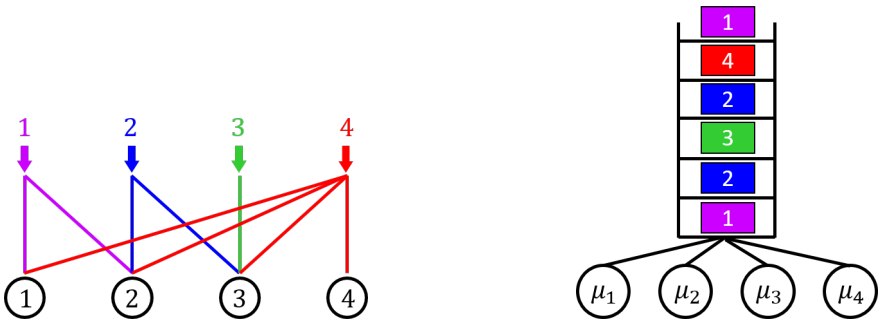


Figure 3: Left: a bipartite graph structure showing job/server compatibility constraints. Right: a system state in the collaborative model. In this example, the state is (1,2,3,2,4,1).

of the jobs in the queue, and let

$$\mu(\vec{c}_n) := \mu(\{c_1, \ldots, c_n\}) = \sum_{m \in S(\vec{c}_n)} \mu_m \tag{1}$$

be the total rate of service to jobs in the queue. Also, define $\Delta_j(\vec{c}_n)$ as the (marginal) rate of service given to the j'th job in the queue, so $\sum_{j=1}^{n} \Delta_j(\vec{c}_n) = \mu(\vec{c}_n)$, and

$$\Delta_j(\vec{c}_n) = \sum_{m \in S(\vec{c}_j) \setminus S(\vec{c}_{j-1})} \mu_m = \sum_{m \in S(\vec{c}_j)} \mu_m - \sum_{m \in S(\vec{c}_{j-1})} \mu_m = \mu(\vec{c}_j) - \mu(\vec{c}_{j-1}).$$

Figure 3 shows an example of a possible state in the collaborative model. In this example, the state is $\vec{c}_n = (1, 2, 3, 2, 4, 1)$. The class-1 job at the head of the queue is in service at both server 1 and server 2 ($\Delta_1(\vec{c}_n) = \mu_1 + \mu_2$), the class-2 job immediately behind it is in service at server 3 ($\Delta_2(\vec{c}_n) = \mu_3$), and the class-4 job is in service at server 4 ($\Delta_3(\vec{c}_n) = \Delta_4(\vec{c}_n) = 0$ and $\Delta_5(\vec{c}_n) = \mu_4$). The total rate of service given to all jobs is $\mu(\vec{c}_n) = \mu_1 + \mu_2 + \mu_3 + \mu_4$.

Note that, for the collaborative model, the total service rate $\mu(\vec{c}_n)$ is independent of the order of the jobs in the queue, and the service rate allocated to the j'th job doesn't depend on the jobs (if any) after job j in the queue. That is, our collaborative model is a special case of an *Order Independent* queue, defined as follows.

Definition 3.1. *A queue is said to be* Order Independent *(OI) if it satisfies the following properties for all* $\vec{c}_n$*:*

(i) $\Delta_j(\vec{c}_n) = \Delta_j(\vec{c}_j)$ *for* $j \leq n$,

(ii) $\mu(\vec{c}_n)$ *is the same for any permutation of* $c_1, \ldots, c_n$,

(iii) $\mu(c) > 0$ *for any class* c.

Properties (i)-(iii) are essentially the same as those defined by Krzesinski [39], though Krzesinski's definition generalizes property (i) to also allow for an extra multiplicative service rate factor based on the number in queue. Our collaborative model can be generalized in this way to have *speed scaling*, i.e., a total service capacity of $\gamma(n)$ when there are n jobs in the system. Under this generalization, μ_m would be interpreted as the proportion of the total capacity used by server m, for $m \in S(\vec{c}_n)$. The addition of a speed-scaling factor is straightforward, but complicates the notation, so we do not include it here. Similarly, it is straightforward to include an arrival scaling (or rejection) factor, so that arrivals of class c occur according to a Poisson process with rate $r(n)\lambda_c$ when the number in queue is n, but, again, we do not include it for ease of exposition.

Property (iii) ensures irreducibility of the Markov chain. Property (ii) guarantees that the total rate of transitions out of any state $\vec{c}_n$ depends only on the set of customers in the queue and not on their order. A consequence of (i) and (ii) is that $\Delta_j(\vec{c}_j)$ does not depend on the order of the first $j-1$ jobs. As Krzesinski shows, Properties (i)-(iii) are all that are needed to show that the stationary distribution has a product form.

The proof below is essentially the same as Krzesinski's [39]; the special case for the collaborative model was shown by Gardner et al. [25].

We first recall the definition of quasi-reversibility.

Definition 3.2. *A queue is called quasi-reversible if its state at time t is independent of*

- *arrival times after time t*
- *departure times before time t.*

An equivalent definition is that the stationary distribution for the queue satisfies partial balance, i.e., for any state and any class c, the steady-state rate out of the state due to a class-c arrival equals the steady-state rate into the state due to a class-c departure, and the rate out of the state due to a departure equals the rate in due to an arrival. Theorem 3.3 shows that the OI properties are sufficient for partial balance for the product-form distribution, and therefore for quasi-reversibility of the system.

Theorem 3.3. *(Berezner, Kriel, Krzesinski [13], Krzesinski [39]) For any OI queue, including the collaborative model, the system is quasi-reversible and the stationary distribution is given by*

$$\pi^C(\vec{c}_n) = \pi^C(\emptyset)\prod_{i=1}^{n}\frac{\lambda_{c_i}}{\mu(\vec{c}_i)} = \frac{\lambda_{c_n}}{\mu(\vec{c}_n)}\pi^C(\vec{c}_{n-1}) \tag{2}$$

as long as $G := \sum_{n,\vec{c}_n\in\mathcal{C}}\prod_{i=1}^{n}\frac{\lambda_{c_i}}{\mu(\vec{c}_i)} < \infty$. Then $\pi^C(\emptyset) = 1/G$ is the probability the system is empty.

Proof. We will show that the product form of equation (2) satisfies partial balance. First note that equation (2) immediately satisfies the condition that the rate out of any state $\vec{c}_n$ due to a departure equals the rate into the state due to an arrival: $\mu(\vec{c}_n)\pi^C(\vec{c}_n) = \lambda_{c_n}\pi^C(\vec{c}_{n-1})$. Now we show that under the product-form probabilities (2), the rate out of any state $\vec{c}_n$ due to a class-c arrival equals the rate into the state due to a class-c departure, $\forall c$:

$$\begin{aligned}
\pi^C(\vec{c}_n)\lambda_c &= \sum_{j=0}^{n}\pi^C(c_1,\ldots,c_j,c,c_{j+1},\ldots,c_n)\Delta_{j+1}(c_1,\ldots,c_j,c,c_{j+1},\ldots,c_n)\\
&= \sum_{j=0}^{n}\pi^C(c_1,\ldots,c_j,c,c_{j+1},\ldots,c_n)\Delta_{j+1}(\vec{c}_j,c) \qquad \text{(Property (i))}\\
&= \frac{\lambda_{c_n}}{\mu(\vec{c}_n,c)}\sum_{j=0}^{n-1}\pi^C(c_1,\ldots,c_j,c,c_{j+1},\ldots,c_{n-1})\Delta_{j+1}(\vec{c}_j,c)\\
&\qquad + \frac{\lambda_c}{\mu(\vec{c}_n,c)}\pi^C(\vec{c}_n)\Delta_{n+1}(\vec{c}_n,c) \qquad \text{((2) and Property (ii)).}
\end{aligned}$$

We will show this by induction on n. For $n = 0$, $\pi^C(0)\lambda_c = \pi^C(c)\mu(c)$ is immediate, given property (iii). Assume partial balance holds for the product-form probabilities (2) for any $\vec{c}_{n-1}$, i.e.,

$$\pi^C(\vec{c}_{n-1})\lambda_c = \sum_{j=0}^{n-1}\pi^C(c_1,\ldots,c_j,c,c_{j+1},\ldots,c_{n-1})\Delta_{j+1}(\vec{c}_j,c).$$

Then we need to show

$$\pi^C(\vec{c}_n)\lambda_c = \frac{\lambda_{c_n}}{\mu(\vec{c}_n,c)}\sum_{j=0}^{n-1}\pi^C(c_1,\ldots,c_j,c,c_{j+1},\ldots,c_{n-1})\Delta_{j+1}(\vec{c}_j,c) + \frac{\lambda_c}{\mu(\vec{c}_n,c)}\pi^C(\vec{c}_n)\Delta_{j+1}(\vec{c}_n,c)$$

From the induction hypothesis, and the definition of $\Delta_{n+1}(\vec{c}_n, c)$, the right-hand-side is:

$$\begin{aligned}
&\frac{\lambda_{c_n}}{\mu(\vec{c}_n, c)}\pi^C(\vec{c}_{n-1})\lambda_c + \frac{\lambda_c}{\mu(\vec{c}_n, c)}\pi^C(\vec{c}_n)[\mu(\vec{c}_n, c) - \mu(\vec{c}_n)] \\
&= \frac{\lambda_{c_n}}{\mu(\vec{c}_n, c)}\pi^C(\vec{c}_{n-1})\lambda_c + \lambda_c\pi^C(\vec{c}_n) - \frac{\lambda_c}{\mu(\vec{c}_n, c)}\frac{\lambda_{c_n}}{\mu(\vec{c}_n)}\pi^C(\vec{c}_{n-1})\mu(\vec{c}_n) \\
&= \pi^C(\vec{c}_n)\lambda_c.
\end{aligned}$$

□

In the collaborative example in Figure 3, recalling that μ is the total service rate, the stationary probability of the depicted state is

$$\pi^C(\vec{c}_n) = \pi^C(\emptyset)\left(\frac{\lambda_1}{\mu_2+\mu_3}\right)\left(\frac{\lambda_2}{\mu_1+\mu_2+\mu_3}\right)\left(\frac{\lambda_3}{\mu_1+\mu_2+\mu_3}\right)\left(\frac{\lambda_2}{\mu_1+\mu_2+\mu_3}\right)\left(\frac{\lambda_4}{\mu}\right)\left(\frac{\lambda_1}{\mu}\right).$$

We reiterate that the skill-based collaborative queue is a special case of an OI queue. Other queues that are OI are the (noncollaborative) M/M/K queue with heterogeneous servers, the M/M/∞ queue, the M/M/1 queue under processor sharing, and the Multiserver Station with Concurrent Classes of Customers (MSCCC) queue [39]. The MSCCC queue is a multi-class M/M/K/FCFS queue with the restriction that at most B_c customers of class c can be in service (noncollaboratively) at the same time. The M/M/1/LCFS queue is *not* an OI queue even though it is a symmetric queue in the sense of Kelly, and is therefore quasi-reversible.

The following corollaries, generalizing the OI queue, follow immediately from Theorem 3.3.

Corollary 3.4. *The departure process from an OI queue is a Poisson process; thus a network of OI queues will have a product-form stationary distribution.*

Consider an order independent queue with abandonments, where a job of class i abandons the system after an exponential time with rate γ_i. This model also fits within the OI framework (i.e., properties (i)-(iii) are satisfied), so again has a product-form stationary distribution.

Corollary 3.5. *In an order independent queue with abandonments,*

$$\pi_A^C(\vec{c}_n) = \pi_A^C(\emptyset)\prod_{i=1}^{n}\frac{\lambda_{c_i}}{\mu(\vec{c}_i)} = \frac{\lambda_{c_n}}{\mu(\vec{c}_n)}\pi_A^C(\vec{c}_{n-1}), \tag{3}$$

where

$$\mu(\vec{c}_j) = \sum_{i=1}^{j}\gamma_{c_i} + \sum_{m\in S(\vec{c}_j)}\mu_m$$

and

$$\Delta_j(\vec{c}_n) = \mu(\vec{c}_j) - \mu(\vec{c}_{j-1}) = \gamma_{c_j} + \sum_{m\in S(\vec{c}_j)\setminus S(\vec{c}_{j-1})}\mu_m.$$

As Berezner and Krzesinski [14] show, the product form result for OI queues also extends easily to OI loss models, where, following their terminology, we use the term loss in the general sense that arriving jobs may be rejected or lost, depending on the current state. For the product-form to continue to hold, the acceptance, or truncated, region must satisfy the *truncation property*: the job acceptance/rejection decision is also order independent and rejection is more likely with more jobs. In particular, letting $\mathcal{C}_T$ comprise the states $(\vec{c}_n, c)$ in which jobs of class c are accepted when the state just before their arrival is $\vec{c}_n$, we have the following.

Definition 3.6. *A set of states $\mathcal{C}_T$ satisfies the* truncation property *if:*

(i) $\vec{c}_n \in \mathcal{C}_T \Rightarrow \mathcal{P}(\vec{c}_n) \subseteq \mathcal{C}_T$, *where $\mathcal{P}(\vec{c}_n)$ denotes the set of permutations of $\vec{c}_n$, and*

(ii) $\vec{c}_n \in \mathcal{C}_T \Rightarrow \vec{c}_{n-1} \in \mathcal{C}_T$. *That is, using part (i) of the truncation property, if a job would be accepted with a given set of jobs in the queue, it will still be accepted if any job is removed from that set.*

Letting $\vec{x} = (x_1, ..., x_J)$ be the per-class aggregated state for $\vec{c}_n$ (which is sufficient for the acceptance/rejection decision because of its OI property), the truncation property means the acceptance region for x is coordinately convex. That is, the rejection decision is a threshold decision, such that arrivals of type c are rejected if $x_c > t(x_1, ..., x_{c-1}, x_{c+1}, ...x_J)$ for some function t. Simple examples include having an upper bound on the total number of jobs, or having upper bounds on the number in each job class.

The product form, now for $\vec{c}_n \in \mathcal{C}_T$, is exactly the same, except for the normalizing constant. In other words, the stationary probability of being in a state in $\mathcal{C}_T$ for the loss model is the same as the conditional probability of being in that state in the model without losses, given that the state is in $\mathcal{C}_T$. Let $\vec{C} \in \mathcal{C}$ be the random variable representing the state of the original collaborative system, with no rejections, in steady state, i.e., $\vec{C} \sim \pi^C$. Let $\vec{C}_T \in \mathcal{C}_T$ and π_T^C be similarly defined for the model with rejections.

Corollary 3.7. *For an OI queue with job rejection, if the acceptance region $\mathcal{C}_T$ satisfies the truncation property, then*

$$P\{\vec{C}_T = \vec{c}_n\} = P\{\vec{C} = \vec{c}_n | \vec{C} \in \mathcal{C}_T\} = \pi_T^C(\vec{c}_n) = \pi_T^C(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} = \frac{\lambda_{c_n}}{\mu(\vec{c}_n)} \pi^C(\vec{c}_{n-1}) \text{ for } \vec{c}_n \in \mathcal{C}_T,$$

where $\pi_T^C(\emptyset) = \pi^C(\emptyset)/P\{\vec{C} \in \mathcal{C}_T\}$.

Proof. To see that π_T has the given product form, note that for states and transitions to states in $\mathcal{C}_T$ the same partial balance equations hold as for the original OI queue, and for transitions where some of the states are not in $\mathcal{C}_T$, the partial balance equations are easily seen to reduce to $0 = 0$ because of the truncation property. For example, if $\vec{c}_n \in \mathcal{C}_T$, but $(\vec{c}_n, c) \notin \mathcal{C}_T$, then the rate out of $\vec{c}_n$ due to a class-c arrival is 0, and the rate into $\vec{c}_n$ due to a class-c departure is also 0, because $\pi_T^C(\vec{c}_n, c) = 0$ for all permutations of $(\vec{c}_n, c)$. Also, for $\vec{c}_n \in \mathcal{C}_T$,

$$P\{\vec{C} = \vec{c}_n | \vec{C} \in \mathcal{C}_T\} = \frac{\pi^C(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)}}{\sum_j \sum_{\vec{c}_j \in \mathcal{C}_T} \pi^C(\vec{c}_j)} = G \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)}$$

where $G = \pi^C(\emptyset)/P\{\vec{C} \in \mathcal{C}_T\}$ is a normalizing constant, and, because of the form of π_T^C, $G = \pi_T^C(\emptyset)$. □

The following special cases will be useful later. Let the subscript $-A$ represent the system where all job classes in A are removed. Let the subscript $\vdash B$ represent a reduced system in which all the servers in set B are removed, as well as all job classes that are compatible with those servers, i.e., job class i is removed if $S_i \cap B \neq \emptyset$. Note that if the original system is stable, such subsystems also will be stable.

Corollary 3.8. **(i)** *For all $\vec{c}_n \in \mathcal{C}_{-A}$*

$$P\{\vec{C} = \vec{c}_n | \vec{C} \in \mathcal{C}_{-A}\} = P\{\vec{C}_{-A} = \vec{c}_n\} = \pi_{-A}^C(\vec{c}_n).$$

(ii) *For all $\vec{c}_n \in \mathcal{C}_{\vdash B}$*

$$P\{\vec{C} = \vec{c}_n | \vec{C} \in \mathcal{C}_{\vdash B}\} = P\{\vec{C}_{\vdash B} = \vec{c}_n\} = \pi_{\vdash B}^C(\vec{c}_n).$$

We mention here a recent extension of the OI queue by Comte and Dorsman [24], the "pass and swap" queue. In their model there is an undirected graph linking the classes of the OI queue, such that an edge between two classes indicates that they are "swappable." The service process satisfies the conditions of the OI queue, but now when a job completes service or is replaced, it in turn replaces a later, swappable job in the queue. A job that completes or is replaced and that finds no later swappable job leaves the system. Comte and Dorsman show that the same product form steady-state distribution holds for the pass and swap queue.

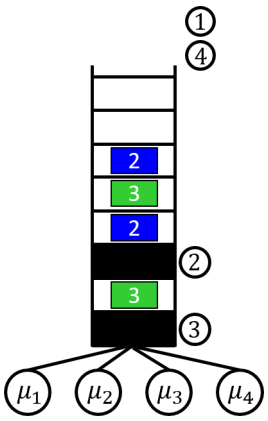


Figure 4: A system state in the noncollaborative model. In this example, the state is $(3, 2, 3, 2; 4, 1)$.

3.2 The Noncollaborative Service Model: Assign Longest Idle Server

We now turn to the noncollaborative model, in which a job is only allowed to enter service on one server and services are completed nonpreemptively. For this model we must also specify which server is used when an arriving job finds multiple idle and compatible servers; in this section we assume that this is according to Assign Longest Idle (compatible) Server (ALIS), and we will use the superscript $ALIS$ for the stationary distribution.

For the noncollaborative ALIS model we define the state as $(\vec{c}_n, \vec{s}_k)$ where c_i is the class of the i'th oldest job that is *not* receiving service, and s_i is the idle server that has been idle i'th longest (out of k that are idle). Note that unlike in the collaborative model, here the $\vec{c}_n$ vector includes *only* those jobs that are in the queue waiting for service (we will call this the job queue) and *not* jobs that are in service.

Figure 4 shows an example of a possible state in the noncollaborative ALIS model. The state here is $(3, 2, 3, 2; 4, 1)$. The class-3 job at the head of the queue is waiting to enter service on server 3, and the class-2 job immediately behind it is waiting to enter service on server 2 or server 3. Servers 4 and 1 are idle, and server 4 became idle before server 1. While our state does not explicitly record the positions of the busy servers within the job queue, we can infer that, for any job class c that appears in the job queue, all servers in $S(c)$ are serving jobs that arrived earlier than that class-c job. For example, we can tell from the state of the job queue that server 2 is serving a job that arrived earlier than the first class-2 job in the queue.

We define the set of valid states, $\mathcal{X}^{ALIS}$, as those states $(\vec{c}_n, \vec{s}_k)$ such that $s_i \notin S(\vec{c}_n)$, $i = 1, \ldots, k$. That is, $\mathcal{X}^{ALIS} = \mathcal{C}^C_{\vdash \vec{s}_k} \times \mathcal{S}$, where $\mathcal{S}$ is the set of all permutations of all subsets of $\{1, ..., M\}$, and $\mathcal{C}^C_{\vdash \vec{s}_k}$ is the set of valid states for the system queue (including jobs in service) for the reduced collaborative model with the servers in $\vec{s}_k$ removed. Defining, as before,

$$\mu(\vec{c}_n) = \sum_{m \in S(\vec{c}_n)} \mu_m$$

we now have that $\mu(\vec{c}_n)$ is the rate at which one of the first n jobs in the *job queue* leaves the queue (and enters service), and $\Delta_j(\vec{c}_n) = \Delta_j(\vec{c}_j) = \mu(\vec{c}_j) - \mu(\vec{c}_{j-1})$ is the rate at which the j'th job in the job queue leaves the queue (and enters service). The OI properties (i)-(iii) given in Definition 3.1 continue to hold for $\Delta_j(\vec{c}_n)$ and $\mu(\vec{c}_n)$. Indeed, given $\vec{s}_k$, the job queue is an OI loss queue. In state $(\vec{c}_n, \vec{s}_k)$ an arrival of class c will be rejected from the job queue (and it will remove a server from the idle-server queue) if $s_i \in S(c)$ for some $i = 1, ..., k$. The state-dependent acceptance region for the job queue, $\mathcal{S}(\vec{s}_k)$, satisfies the truncation property of the OI loss queue given in Definition 3.6.

We now consider the idle server queue. Let $\lambda(\vec{s}_j)$ be the rate of arrivals of jobs that are compatible with one of the first j (idle) servers, i.e., the rate of departures from the idle server queue when it is in state $\vec{s}_j$. For $k \geq j$, let

$$\Delta^\lambda_j(\vec{s}_k) = \lambda(\vec{s}_j) - \lambda(\vec{s}_{j-1}) = \sum_{i \in C(\vec{s}_j) \setminus C(\vec{s}_{j-1})} \lambda_i \geq 0$$

be the rate at which the j'th idle server will become busy (leave the idle server queue). Note that we have the same OI properties (i)-(iii) for $\lambda(\vec{s}_k)$ and $\Delta_j^\lambda(\vec{s}_n)$ as we had for $\mu(\vec{c}_n)$ and $\Delta_j(\vec{c}_n)$:

(i) $\Delta_j^\lambda(\vec{s}_k) = \Delta_j^\lambda(\vec{s}_j)$ for $j \leq k$,

(ii) $\lambda(\vec{s}_j)$ is the same for any permutation of $s_1, \ldots, s_j$ (order independence),

(iii) $\lambda(s) > 0$ for any server s.

That is, given $\vec{c}_n$, the idle server queue is also an OI loss queue, where we can think of servers of type s arriving according to a Poisson process at rate μ_s, but if the server is already in the queue in state $\vec{s}_k$, or if it will remain busy serving another job, i.e., if $s \in \vec{s}_k \cup S(\vec{c}_n)$, then the arrival is rejected. Hence, the acceptance region for the idle-server queue, given $\vec{c}_n$, also satisfies the truncation property.

The stationary distribution of the noncollaborative ALIS model has a "product of product forms" distribution, with a product form component for the job queue and one for the idle server queue.

Theorem 3.9. *(Adan et al. [5]) For the noncollaborative model, under FCFS for jobs and ALIS for servers, and given the stability condition, for $(\vec{c}_n, \vec{s}_k) \in \mathcal{X}$,*

$$\pi^{ALIS}(\vec{c}_n, \vec{s}_k) = \pi^{ALIS}(\emptyset, \emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)} \tag{4}$$

$$= \frac{\lambda_{c_n}}{\mu(\vec{c}_n)} \pi^{ALIS}(\vec{c}_{n-1}, \vec{s}_k) = \frac{\mu_{s_k}}{\lambda(\vec{s}_k)} \pi^{ALIS}(\vec{c}_n, \vec{s}_{k-1}), \tag{5}$$

where $\pi^{ALIS}(\emptyset, \emptyset)$ is a normalizing constant equal to the probability that all servers are busy and that there are no jobs waiting in the queue.

Proof. Note that if there is an arrival to or departure from the *job queue*, the state of the idle servers do not change, and if there is an arrival to or departure from the set of idle servers, the state of the job queue does not change. Thus, the proof that the product form with $\pi^{ALIS}(\vec{c}_n, \vec{s}_k) = \frac{\lambda_{c_n}}{\mu(\vec{c}_n)} \pi^{ALIS}(\vec{c}_{n-1}, \vec{s}_k)$ satisfies partial balance for job queue arrivals and departures, for fixed $\vec{s}_k$, is exactly the same as for our earlier proof for Theorem 3.3, using the truncation property of our acceptance region for job queue arrivals. For the idle server queue, fixing $\vec{c}_n$, the proof that the rate out of $(\vec{c}_n, \vec{s}_k)$ due to departures of idle servers equals the rate in due to arrivals of idle servers is immediate from $\pi^{ALIS}(\vec{c}_n, \vec{s}_k) = \frac{\mu_{s_k}}{\lambda(\vec{s}_k)} \pi^{ALIS}(\vec{c}_n, \vec{s}_{k-1})$. Finally, the proof that the rate of leaving state $\pi^{ALIS}(\vec{c}_n, \vec{s}_k)$ due to server s becoming idle (arriving to the idle server queue) equals the rate of entering the state due to server s becoming busy (leaving the queue), for $s \notin \vec{s}_k$, is also very similar to our earlier induction proof. Note that to transition out of state $(\vec{c}_n, \vec{s}_k)$ due to the arrival of server s to the idle server queue, s must both be busy and not have any compatible jobs in the job queue, i.e., $s \notin \vec{s}_k \cup S(\vec{c}_n)$. □

In the example shown in Figure 4, the stationary probability is

$$\pi^{ALIS}(\vec{c}_n, \vec{s}_k) = \pi^{ALIS}(\emptyset, \emptyset) \left(\frac{\lambda_3}{\mu_3}\right) \left(\frac{\lambda_2}{\mu_2 + \mu_3}\right) \left(\frac{\lambda_3}{\mu_2 + \mu_3}\right) \left(\frac{\lambda_2}{\mu_2 + \mu_3}\right) \left(\frac{\mu_4}{\lambda_4}\right) \left(\frac{\mu_1}{\lambda_1 + \lambda_4}\right).$$

3.3 The Noncollaborative Service Model: Random Assignment to Idle Servers

We next consider the noncollaborative model where, instead of using ALIS to choose an idle server among compatible servers for an arriving job, servers are chosen randomly among idle compatible servers with appropriate probabilities that depend only on the set of busy (or idle) servers [45]. The results given in [45] use the partially aggregated state space we discuss in Section 5, but, as we will show, a product-form result also holds for a detailed state descriptor similar to the one used for the collaborative model. Unlike under ALIS, under RAIS the order of the idle servers no longer matters. Instead, for this version of the model we keep track of the busy servers, $\vec{b}_l$, where there are l busy servers, and where the servers are ordered by the arrival times of the jobs they are serving. To obtain a product-form stationary distribution, we need our state to be even more detailed: we must track not only the order of the busy servers, but the positions of the busy servers within the job queue. Our detailed state description for RAIS is thus $\vec{z}_m$, where m denotes the number of jobs in the system (including both jobs in the queue and jobs in service), and z_i is associated with the i'th job in the system in order of arrival. This is similar to the state $\vec{c}_n$ state used for the collaborative

model, with one key difference: the $\vec{z}_m$ state does not track the classes of jobs that are in service, instead it tracks the servers that are serving them. That is, we let $z_i = c$ if the i'th job in the system has not started service (it is in the job queue), and $z_i = b$ if the i'th job in the system is in service on server b. Note that $\vec{z}_m$ consists of an interleaving of the states of the job queue, $\vec{c}_n$, and of the busy server queue, $\vec{b}_l$. The possible states for $\vec{z}_m$, $\mathcal{X}^{RAIS}$, are such that $\vec{b}_l \subseteq \{1, ..., M\}$, and for any position i, if $z_i = c$, all compatible servers are serving earlier arrivals, i.e., $S(c) \subseteq \vec{z}_{i-1}$, because of the FCFS service discipline.

In order to completely define the RAIS policy, we must specify the probability that an arriving job enters service at compatible idle server $s \notin \{b_1, \ldots, b_l\}$. When the set of ordered busy servers is $\vec{b}_j$, let $\lambda_s^a(\vec{b}_j)$ represent the *activation rate* of idle server $s \notin \{b_1, ..., b_j\}$ (the rate of going from state $\vec{b}_j$ to $(\vec{b}_j, s)$ for any $\vec{c}_n$). We allow the activation rates to depend only on the set of busy servers, not on their order. Indeed, as Visschers et al. showed for their aggregated state description of this model [45], in order for the stationary distribution to have a product form, we need the following stronger condition, called the *assignment condition*. Let $\Pi_\lambda(\vec{b}_l) = \prod_{j=1}^{l} \lambda_{b_j}^a(\vec{b}_{j-1})$. The assignment condition requires that the probabilities for routing to compatible idle servers be chosen so that $\Pi_\lambda(\vec{b}_l)$ depends only on the set of busy servers, not on their order (i.e., so that $\Pi_\lambda(\vec{b}_l)$ is the same for any permutation of $b_1, \ldots, b_l$). Visschers et al. show that it is always possible to choose assignment probability distributions so that the assignment condition holds [45]; the derivation involves solving a max flow problem for each subset of busy servers.

One way to interpret the assignment condition is to consider the loss system in which customers are not allowed to queue, so that the state is just $\vec{b}_l$, the set of busy servers. Then the assignment condition, along with the fact that $\mu(\vec{b}_l)$ doesn't depend on the order of busy servers, reduces to Kolmogorov's criterion for reversibility of Markov chains, namely that the product of the transition probabilities along any path from a state back to itself is the same if the states are traversed in the reverse order. For example, consider the path traversing the states $\emptyset, (u), (u, v), (v), \emptyset$, where u and v are two servers. Then the probability of traversing that path is $C\lambda_u^a(\emptyset)\lambda_v^a(u)\mu_u\mu_v$ where C is an appropriate normalizing constant, and the probability for the reverse path, in which v is activated first and finishes first, is $C\lambda_v^a(\emptyset)\lambda_u^a(v)\mu_u\mu_v$. These are the same, given the assignment condition: $\lambda_u^a(\emptyset)\lambda_v^a(u) = \lambda_v^a(\emptyset)\lambda_u^a(v)$. Indeed, Adan, Hurkens, and Weiss showed that the loss model (under the assignment condition) is reversible, has a product-form stationary distribution, and is insensitive to the service time distribution [4]. We also note that the same is true for the loss model under ALIS [3]. (Recall that the idle server queue is an OI queue.) Haji and Ross [33] further showed that if the bipartite matching structure satisfies an exchangeability assumption, the same result holds for any policy for assigning a job to a compatible server, as long as the assignment policy depends only the ordered queue of idle servers (ordered by the times at which they became idle). Indeed, under the exchangeability assumption, all orders are equally likely.

Let $\mu(\vec{z}_i) = \sum_{j=1}^{i} I(z_j)\mu_{z_j}$, where $I(z_j)$ is an indicator that is 1 if z_j corresponds to a busy server and 0 if it corresponds to a job in the job queue. Note that $\mu(\vec{z}_m)$ satisifies the conditions for order independence, with $\Delta_j(\vec{z}_m) = \Delta_j(\vec{z}_j) = \mu(\vec{z}_j) - \mu(\vec{z}_{j-1})$. Let $\lambda_i^z(\vec{z}_i) = \lambda_i^z(\vec{z}_m) = \lambda_c$ if $z_i = c$ for some job class c, and, if $z_i = b$ for some busy server b, let $\lambda_i^z(\vec{z}_i) = \lambda_i^z(\vec{z}_m) = \lambda_b^a(\vec{b}_{k(i)})$, where $k(i) = \sum_{j=1}^{i-1} I(z_j)$ is the number of busy servers in the first $i-1$ positions (the number of servers serving the first $i-1$ arrivals). Note that $\lambda_m^z(\vec{z}_m)$ is the same for any permutation of $\vec{z}_{m-1}$ regardless of whether z_m is a waiting job or a busy server, and, from the assignment condition, for any $\vec{z}_m \in \mathcal{X}^{RAIS}$, $\lambda_{m-1}^z(\vec{z}_{m-1})\lambda_m^z(\vec{z}_m) = \lambda_{m-1}^z(\vec{z}_{m-2}, z_m)\lambda_m^z(\vec{z}_{m-2}, z_m, z_{m-1}) = \lambda_{m-1}^z(\vec{z}_{m-2}, z)\lambda_m^z(\vec{z}_m)$.

Theorem 3.10. *For the noncollaborative model, under FCFS for jobs and random assignment to idle servers, and under the assignment condition and the stability condition, for $\vec{z}_m \in \mathcal{X}$,*

$$\pi^{RAIS}(\vec{z}_m) = \pi^{RAIS}(\emptyset) \prod_{i=1}^{m} \frac{\lambda_i^z(\vec{z}_i)}{\mu(\vec{z}_i)} = \frac{\lambda_m^z(\vec{z}_m)}{\mu(\vec{z}_m)} \pi^{RAIS}(\vec{z}_{m-1}),$$

where $\pi^{RAIS}(\emptyset)$ is a normalizing constant that represents the probability that the system is empty (i.e., that there are no busy servers and no jobs in the queue).

Before proving Theorem 3.10, we give a brief example of the system state and stationary probability under RAIS. Consider the example in Figure 4. The state under RAIS is $\vec{z}_m = (3_b, 3_c, 2_b, 2_c, 3_c, 2_c)$, where

we use a subscript of b or c to indicate whether the entry in $\vec{z}_m$ corresponds to a job that is waiting for service in the job queue (subscript c) or to a busy server (subscript b). Note that for jobs in the job queue the notation i_c indicates that the job in this position is class-i, whereas for busy servers the notation j_b indicates that server j is in this position (that is, we do not track the classes of jobs in service). The state $\vec{z}_m$ is an interleaving of $\vec{c}_n = (3, 2, 3, 2)$ and $\vec{b}_l = (3, 2)$. The stationary probability for this state is

$$\pi^{RAIS}(\vec{z}_m) = \pi^{RAIS}(\emptyset)\left(\frac{\lambda_3^a(\emptyset)}{\mu_3}\right)\left(\frac{\lambda_3}{\mu_3}\right)\left(\frac{\lambda_2^a(3)}{\mu_3+\mu_2}\right)\left(\frac{\lambda_2}{\mu_3+\mu_2}\right)\left(\frac{\lambda_3}{\mu_3+\mu_2}\right)\left(\frac{\lambda_2}{\mu_3+\mu_2}\right).$$

We are now ready to prove Theorem 3.10.

Proof. Fix $\vec{z}_m \in \mathcal{X}^{RAIS}$, with corresponding $\vec{b}_l$. We will show that partial balance holds in three steps:

1. The rate out of state $\vec{z}_m$ due to a service completion equals the rate into state $\vec{z}_m$ due to an arrival.
2. The rate out of state $\vec{z}_m$ due to server b becoming busy equals the rate into state $\vec{z}_m$ due to server b becoming idle.
3. The rate out of state $\vec{z}_m$ due to a class-c job arrival to the job queue equals the rate into state $\vec{z}_m$ due to a class-c departure from the job queue.

1. First suppose $z_m = b_l$, so $\vec{z}_m = (\vec{z}_{m-1}, b_l)$. Then our product form immediately satisfies $\mu(\vec{z}_m)\pi^{RAIS}(\vec{z}_m) = \lambda^a_{b_l}(\vec{b}_{l-1})\pi^{RAIS}(\vec{z}_{m-1})$, i.e., the rate of transitions out of state $\vec{z}_m$ due to a service completion equals the rate into $\vec{z}_m$ due to a new server arrival, i.e., of server b_l going from idle to busy and serving the most recently arriving job. If $z_m \neq b_l$ then it is not possible to enter state $\vec{z}_m$ with an idle server becoming busy. Now suppose $z_m = c_n$. In this case we have, for our product form, $\mu(\vec{z}_m)\pi^{RAIS}(\vec{z}_m) = \lambda_{c_n}\pi^{RAIS}(\vec{z}_{m-1})$, i.e., the rate of transitions out of state $\vec{z}_m$ due to a service completion equals the rate into $\vec{z}_m$ due to a new arrival to the job queue. Note that c_n is such that $S(c_n) \subseteq \vec{b}_l$.

2. We now show that under the product-form probabilities above, the rate out of state $\vec{z}_m$ due to the (external) arrival of any server $b \notin \vec{b}_l$ to the busy server queue equals the rate into the state due to server b's departure from the busy server queue. Note that because $\vec{z}_m \in \mathcal{X}^{RAIS}$ and $b \notin \vec{b}_l$, none of the jobs in the job queue are compatible with server b, so a job completion at server b in state $(z_1, \ldots, z_j, b, z_{j+1}, \ldots, z_m)$ will result in server b leaving the busy server queue. Using the OI properties of $\mu(\vec{z}_m)$, and that $\lambda^z_m(\vec{z}_m)$ is the same for any permutation of $\vec{z}_{m-1}$, we need to show that the given product form satisfies

$$\begin{aligned}
\pi^{RAIS}(\vec{z}_m)\lambda^z_{m+1}(\vec{z}_m, b) &= \sum_{j=0}^{m} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, \ldots, z_m)\Delta_{j+1}(\vec{z}_j, b) \\
&= \frac{\lambda^z_{m+1}(\vec{z}_{m-1}, b, z_m)}{\mu(\vec{z}_m, b)} \sum_{j=0}^{m-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, \ldots, z_{m-1})\Delta_{j+1}(\vec{z}_j, b) \\
&\quad + \frac{\lambda^z_{m+1}(\vec{z}_m, b)}{\mu(\vec{z}_m, b)}\pi^{RAIS}(\vec{z}_m)\Delta_m(\vec{z}_m, b). \qquad (6)
\end{aligned}$$

We use induction on m; the induction hypothesis is that

$$\pi^{RAIS}(\vec{z}_{m-1})\lambda^z_m(\vec{z}_{m-1}, b) = \sum_{j=0}^{m-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, \ldots, z_{m-1})\Delta_{j+1}(\vec{z}_j, b).$$

Then the RHS of (6) is

$$\begin{aligned}
&\frac{\lambda^z_{m+1}(\vec{z}_{m-1}, b, z_m)}{\mu(\vec{z}_m, b)}\pi^{RAIS}(\vec{z}_{m-1})\lambda^z_m(\vec{z}_{m-1}, b) + \frac{\lambda^z_{m+1}(\vec{z}_m, b)}{\mu(\vec{z}_m, b)}\pi^{RAIS}(\vec{z}_m)[\mu(\vec{z}_m, b) - \mu(\vec{z}_m)] \\
&= \frac{\lambda^z_{m+1}(\vec{z}_{m-1}, b, z_m)}{\mu(\vec{z}_m, b)}\pi^{RAIS}(\vec{z}_{m-1})\lambda^z_m(\vec{z}_{m-1}, b) + \lambda^z_{m+1}(\vec{z}_m, b)\pi^{RAIS}(\vec{z}_m) \\
&\quad - \frac{\lambda^z_{m+1}(\vec{z}_m, b)}{\mu(\vec{z}_m, b)}\frac{\lambda^z_m(\vec{z}_m)}{\mu(\vec{z}_m)}\pi^{RAIS}(\vec{z}_{m-1})\mu(\vec{z}_m) \\
&= \pi^{RAIS}(\vec{z}_m)\lambda^z_{m+1}(\vec{z}_m, b)
\end{aligned}$$

where $\lambda^z_m(\vec{z}_{m-1}, b)\lambda_{m+1}(\vec{z}_{m-1}, b, z_m) = \lambda^z_m(\vec{z}_m)\lambda^z_{m+1}(\vec{z}_m, b)$ from the assignment condition.

3. Finally, we show that the rate out of $\vec{z}_m$ due to a class-c arrival to the job queue equals the rate in to state $\vec{z}_m$ due to a class-c job queue departure, for each c such that $S(c) \subseteq \vec{b}_l$. Fix c and $\vec{z}_m$ and call the class-c job whose departure causes the system to enter state $\vec{z}_m$ the tagged job. Let $\vec{z}'_{m+1}$ denote the system state just before the tagged job leaves the job queue. The transition from $\vec{z}'_{m+1}$ to $\vec{z}_m$ is triggered by a service completion at some server $b \in S(c)$. In $\vec{z}'_{m+1}$ it must be the case that b, and all other servers in $S(c)$, are serving jobs that arrived earlier than the tagged job. At the service completion on server b, the job it is working on leaves, and server b takes the position of the tagged job. Therefore, server b's position in $\vec{z}_m$, after the service completion, must be after all the other servers in $S(c)$. Call this position κ. Before the transition, in state $\vec{z}'_{m+1}$, the tagged job must be in position $\kappa + 1$, and server b must be in position $j + 1 \leq \kappa$. Thus, we need to show that

$$\pi^{RAIS}(\vec{z}_m)\lambda_c = \sum_{j=0}^{\kappa-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{\kappa-1}, c, z_{\kappa+1}, \ldots, z_m)\Delta_{j+1}(\vec{z}_j, b).$$

First suppose $z_m = b$, i.e., $\kappa = m$. Then we want to show that

$$\pi^{RAIS}(\vec{z}_{m-1}, b)\lambda_c = \sum_{j=0}^{m-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-1}, c)\Delta_{j+1}(\vec{z}_j, b). \tag{7}$$

Suppose, using induction on m, that

$$\pi^{RAIS}(\vec{z}_{m-2}, b)\lambda_c = \sum_{j=0}^{m-2} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-2}, c)\Delta_{j+1}(\vec{z}_j, b).$$

Note that for $j < m - 1$,

$$\begin{aligned}
\pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-1}, c) &= \frac{\lambda_c}{\mu(\vec{z}_{m-1}, b)} \frac{\lambda^z_{m-1}(\vec{z}_{m-1})}{\mu(\vec{z}_{m-1}, b)} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-2}) \\
&= \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-2}, c, z_{m-1}) \\
&= \frac{\lambda^z_{m-1}(\vec{z}_{m-1})}{\mu(\vec{z}_{m-1}, b)} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-2}, c).
\end{aligned}$$

Thus, the RHS of (7) is

$$\begin{aligned}
&\frac{\lambda^z_{m-1}(\vec{z}_{m-1})}{\mu(\vec{z}_{m-1}, b)} \sum_{j=0}^{m-2} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{m-2}, c)\Delta_{j+1}(\vec{z}_j, b) + \frac{\lambda_c}{\mu(\vec{z}_{m-1}, b)} \pi^{RAIS}(\vec{z}_{m-1}, b)\Delta_m(\vec{z}_{m-1}, b) \\
&= \frac{\lambda^z_{m-1}(\vec{z}_{m-1})}{\mu(\vec{z}_{m-1}, b)} \pi^{RAIS}(\vec{z}_{m-2}, b)\lambda_c + \frac{\lambda_c}{\mu(\vec{z}_{m-1}, b)} \pi^{RAIS}(\vec{z}_{m-1}, b) \left[\mu(\vec{z}_{m-1}, b) - \mu(\vec{z}_{m-1})\right] \\
&= \frac{\lambda^z_{m-1}(\vec{z}_{m-1})}{\mu(\vec{z}_{m-1}, b)} \pi^{RAIS}(\vec{z}_{m-2}, b)\lambda_c + \lambda_c \pi^{RAIS}(\vec{z}_{m-1}, b) - \frac{\lambda_c}{\mu(\vec{z}_{m-1}, b)} \frac{\lambda^z_{m-1}(\vec{z}_{m-1})}{\mu(\vec{z}_{m-1})} \pi^{RAIS}(\vec{z}_{m-2}, b)\mu(\vec{z}_{m-1}) \\
&= \lambda_c \pi^{RAIS}(\vec{z}_{m-1}, b).
\end{aligned}$$

Now suppose $\kappa < m$, so $\vec{z}_m = (z_1, ..., z_{\kappa-1}, b, z_{\kappa+1}, ..., z_m)$. Then we want to show that

$$\pi^{RAIS}(z_1, ..., z_{\kappa-1}, b, z_{\kappa+1}, ..., z_m)\lambda_c = \sum_{j=0}^{\kappa-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{\kappa-1}, c, z_{\kappa+1}, \ldots, z_m)\Delta_{j+1}(\vec{z}_j, b),$$

i.e.,

$$\prod_{i=\kappa+1}^{m} \frac{\lambda^z_i(\vec{z}_i)}{\mu(\vec{z}_i)} \pi^{RAIS}(\vec{z}_{\kappa-1}, b)\lambda_c = \prod_{i=\kappa+1}^{m} \frac{\lambda^z_i(\vec{z}_i)}{\mu(\vec{z}_i)} \sum_{j=0}^{\kappa-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{\kappa-1}, c)\Delta_{j+1}(\vec{z}_j, b).$$

From our previous argument we have

$$\pi^{RAIS}(\vec{z}_{\kappa-1}, b)\lambda_c = \sum_{j=0}^{\kappa-1} \pi^{RAIS}(z_1, \ldots, z_j, b, z_{j+1}, .., z_{\kappa-1}, c)\Delta_{j+1}(\vec{z}_j, b),$$

and the result follows. □

3.4 Relationship between Collaborative and Noncollaborative Models

The product form stationary probabilities for the collaborative model and the ALIS noncollaborative model both include the term $\prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_n)}$. Given the similarities in the stationary distributions, it is natural to ask whether the two systems also are similar in their more detailed evolution. Indeed, Adan et al. observed that when all servers are busy (i.e., the idle-server queue is empty, $\vec{s}_k = \emptyset$ under ALIS) the path-wise evolution of the state $\vec{c}_n$ (i.e., the jobs in queue) in the noncollaborative model is the same as the evolution of $\vec{c}_n$ (jobs in system) in the collaborative model [5]. We generalize this observation to relate the path-wise evolution of the two systems conditioned on the set of idle servers. Note that while the set of idle servers is fixed, we need not worry about how jobs are assigned to idle servers.

Observation 3.11. *Conditioned on the set of idle servers, $\vec{s}_k$, and while those servers remain idle, the path-wise evolution of $\vec{c}_n$ (jobs in queue) for the noncollaborative model (under either RAIS or ALIS) is the same as that of $\vec{c}_n$ (jobs in system) for the truncated collaborative model with the servers in $\vec{s}_k$ removed.*

Observation 3.11 tells us that, with coupled arrivals and service completions and the same initial $\vec{c}_n$, a service completion removes a job from the system for the collaborative model and removes the corresponding job from the job queue in the noncollaborative model. In the noncollaborative model, another job that does not appear in $\vec{c}_n$ will also leave the system (and will be replaced at the server by the job leaving the job queue). We note that the path-wise coupling still holds for general (coupled) arrival processes, not just Poisson processes.

Our path-wise equivalence between the job queue in the noncollaborative model conditioned on the set of busy servers and the system queue for the collaborative model with the idle servers removed, is somewhat analogous to the observation of Borst et al. of the equivalence between the jobs in system in a processor sharing model with the jobs in queue for a nonpreemptive random-order-of-service model [18].

The correspondence between the collaborative and noncollaborative models will be useful when we move from the stationary distribution to performance metrics such as per-class response time distributions. In Section 4, we will see that these performance metrics often are more straightforward to derive in the collaborative model. The relationship between the two models allows us to apply our results in the collaborative model to the noncollaborative model. One consequence of this relationship is the following corollary.

Corollary 3.12. *Let $\pi_s^{ALIS}(\vec{s}_k)$ be the probability that servers $s_1, ..., s_k$ are idle, in that order, in steady-state for the noncollaborative model under ALIS. Then*

$$\pi_s^{ALIS}(\vec{s}_k) = \frac{\pi^{ALIS}(\emptyset, \emptyset)}{\pi^C_{\vdash \vec{s}_k}(\emptyset)} \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)}.$$

Proof. From Theorem 3.9 and Corollary 3.8, we have

$$\begin{aligned}
\pi_s^{ALIS}(\vec{s}_k) &= \sum_{\vec{c}_n \in \mathcal{C}_{\vdash \vec{s}_k}} \pi^{ALIS}(\vec{c}_n, \vec{s}_k) = \pi^{ALIS}(\emptyset, \emptyset) \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)} \sum_{\vec{c}_n \in \mathcal{C}_{\vdash \vec{s}_k}} \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \\
&= \frac{\pi^{ALIS}(\emptyset, \emptyset)}{\pi^C_{\vdash \vec{s}_k}(\emptyset)} \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)} \sum_{\vec{c}_n \in \mathcal{C}_{\vdash \vec{s}_k}} \pi^C_{\vdash \vec{s}_k}(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \\
&= \frac{\pi^{ALIS}(\emptyset, \emptyset)}{\pi^C_{\vdash \vec{s}_k}(\emptyset)} \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)}.
\end{aligned}$$

□

3.5 Token Models

Two generalizations, combining aspects of the collaborative and noncollaborative models, have recently been introduced using the notion of "tokens" [12, 23]. In these models tokens generalize the notion of servers in the noncollaborative model. There is a bipartite compatibility matching between job classes and tokens, jobs must have tokens to enter service, and a token can be assigned to only one job at a time. Jobs of class i arrive according to a Poisson process at rate λ_i, and can be matched to tokens in set S_i. In the first token model, Ayesta et al. [12] allow jobs to wait for tokens and assume that when an arriving job sees multiple idle compatible tokens, it is assigned a token according to RAIS (or RAIT: Random Assignment to Idle Tokens). In the second token model, Comte [23] assumes a loss model, in which jobs that arrive when no compatible tokens are available are lost, and that idle tokens are assigned according to ALIS (or ALIT: Assign Longest Idle Token). We describe these models in more detail below.

3.5.1 Token Model under RAIS

In the model of Ayesta et al. [12], given the set of busy tokens $\vec{b}_l$, listed in the order of the arrival times of the jobs they are serving, and idle token s, the activation rate $\lambda_s^a(\vec{b}_l)$ (i.e., the rate at which s will be assigned to an arriving compatible job) satisfies the same assignment condition as in the noncollaborative RAIS model. The service process, given ordered busy tokens $\vec{b}_l$, is generalized from the skill-based collaborative model to the OI queue. That is, defining $\Delta_j(\vec{b}_l)$ as the (marginal) rate of service given to the job with the j'th busy token and $\mu(\vec{b}_l) = \sum_{k=1}^{m} \Delta_k(\vec{b}_l)$, the following OI conditions are assumed, as in Definition 3.1:

(i) $\Delta_k(\vec{b}_l) = \Delta_k(\vec{b}_k)$ for $j \leq l$,

(ii) $\mu(\vec{b}_l)$ is the same for any permutation of $b_1, \ldots, b_l$ (order independence),

(iii) $\mu(b) > 0$ for any busy token b.

Like Krzesinski [39], Ayesta et al. also allow the service rate to be multiplied by a factor that is a function of the total number of tokens in service. We continue to omit that factor for simplicity.

Let us define the state, as we did for the noncollaborative RAIS model, as $\vec{z}_m$ where z_i is associated with the i'th arrival in the system, $z_i = c$ if the arrival is of class c and does not yet have a token, and $z_i = b$ if it has token b. We also define $\lambda_i(\vec{z}_i)$ as we did for the noncollaborative RAIS model. Note that our proof of Theorem 3.10 did not use the particular form of $\mu(\vec{b}_l)$, only its OI properties. (In particular, we showed the result for general $\mu(\vec{z}_m)$ and $\Delta_j(\vec{z}_m)$). Hence, Theorem 3.10 also holds for the token model.

As Ayesta et al. note [12], the noncollaborative model is recovered when tokens correspond to the servers of the noncollaborative model, and the original OI queue (including the collaborative model) is recovered when each arriving job immediately obtains a token directly corresponding to its class (so there is an infinite supply of tokens, and activation rates need not be included).

3.5.2 Token Loss Model under ALIS

Comte introduced a related, multi-layered, token loss model that generalizes the noncollaborative model operating under ALIS [23]. The terminology and notation used in Comte's model are a bit different from ours; Comte refers to job "type" where we use job "class," and to token "classes" where we use "tokens". (We allow distinct tokens to have the same job class compatibilities and speeds.) In Comte's model, and in contrast to that of Ayesta et al., a job that arrives when there is no available compatible token is lost, and a job that arrives to find multiple compatible tokens takes the token that has been idle longest. Hence, in this model, the state is $(\vec{b}_l, \vec{s}_k)$ where $\vec{b}_l$ is the set of busy tokens listed in the order of the arrival times of their corresponding jobs and $\vec{s}_k$ is the set of idle tokens in the order in which they became idle. Note that, because jobs cannot wait for tokens, there is no $\vec{c}_n$ component of the state, so $\vec{b}_l$ corresponds directly to $\vec{z}_m$ of the noncollaborative RAIS model. Also, tokens alternate between being busy and idle, and therefore $\vec{b}_l$ lists the busy tokens in the order in which they became busy, i.e., their order of arrival to the busy token queue.

Instead of assuming a generic OI service process $\mu(\vec{b}_l)$ for serving tokens, as in Ayesta et al.'s model, Comte assumes a collaborative service model. That is, there is another bipartite matching layer between tokens and servers that defines the total service rate $\mu(\vec{b}_l)$ when the ordered set of busy tokens is $\vec{b}_l$, and such that $\mu(\vec{b}_l)$ satisfies the OI conditions. Note that when the idle token queue is in state $\vec{s}_k$, the rate at

which tokens leave, $\lambda(\vec{s}_k)$, is the rate at which jobs compatible with one of the idle tokens arrive, and, as in the noncollaborative ALIS model, $\lambda(\vec{s}_k)$ also satisfies the OI conditions 3.1. Because there is a finite set of tokens, we have a closed network of two OI queues, and because OI queues are quasi-reversible, the closed token (CT) network also has a product-form distribution. In particular, for $(\vec{b}_l, \vec{s}_k) \in \mathcal{X}^T$, where $\mathcal{X}^T$ is the set of states such that each token appears exactly once, i.e., $l + k$ equals the total number of tokens and $(\vec{b}_l, \vec{s}_k)$ is an arbitrary permutation of the set of tokens, we have

$$\pi^{CT}(\vec{b}_l, \vec{s}_k) = G^{CT} \prod_{i=1}^{l} \frac{1}{\mu(\vec{b}_i)} \prod_{i=1}^{k} \frac{1}{\lambda(\vec{s}_i)}$$

where G^{CT} is a normalizing constant. This result would also hold assuming a general OI process for "serving" busy tokens rather than the collaborative service model.

3.6 Discrete-time OI Queues and Matching Models

Adan et al. [5] introduced a matching model, called the directed bipartite matching (DBM) model, with a bipartite matching between servers and jobs, and in which both servers and jobs arrive according to Poisson processes, but only jobs can queue to wait for servers. Servers of type j arrive according to an independent Poisson process with rate μ_j, and the other parameters of the model are the same as for the collaborative model. The state is again $\vec{c}_n$. An arriving server matches with the first compatible job in the queue, if any, and the server, along with its job if there is one, immediately leaves. The DBM model captures important features of organ transplant waitlists, where patients wait for organs, but unmatched organs are lost, and where compatibilities are determined by biological factors such as blood types, as well as the locations of the patients and organs. As Adan et al. show, the Markov chain for this model is sample-path equivalent to that of the collaborative model; in particular, the departure rate from the queue in state $\vec{c}_n$ is $\mu(\vec{c}_n)$ as defined earlier. Therefore the matching model has the same, product-form, stationary distribution given in Theorem 3.3. The result also holds for a more general OI matching, i.e., when there are no server types, but a job will be matched to a server at rate $\mu(\vec{c}_n)$ when the state is $\vec{c}_n$ and $\mu(\vec{c}_n)$ satisfies the OI conditions (i)-(iii). The state process for the matching model is also equivalent to the queue process of a variant of the noncollaborative model in which we keep all the servers busy by assigning a server that becomes idle and that does not find a waiting compatible job a "dummy job." This might be appropriate in a call center context in which servers that would be otherwise idle handle outgoing calls or email.

If we ignore the timing between arrivals and departures in the matching model described above, we have an equivalent discrete-time model, in which at most one event (a job arrival or a server arrival/job departure) can occur in any time slot. Now λ_c is the probability of a class-c arrival and, when the state is $\vec{c}_n$, $\mu(\vec{c}_n)$ represents the probability of a job completion (or a job-server matching) in the next time slot, $\mu(\vec{c}_n) \leq 1$. Also, we need not assume a set of servers with a bipartite-matching graph, just that $\mu(\vec{c}_n)$ satisfies the OI conditions. Then the transitions of the Markov chain $\vec{c}_n$ for the discrete-time queue will be sample-path equivalent to the transitions of the embedded Markov chain for the continuous-time OI queue, and, again, the same product form will hold for the steady-state distribution. The DBM special case, with server/job compatibilities, is considered by Weiss [46]. Here a server of type s arrives with probability μ_s and matches with the earliest compatible job if there is one; the server immediately departs (along with any matching job).

The DBM model discussed in [5] does not allow unmatched servers to wait for jobs. We now show that the DBM model can be extended to include a "server queue" in which unmatched servers wait in FCFM (first-come-first-matched) order. This yields something somewhat analogous to the noncollaborative ALIS model. For stability, we must have an upper bound, K, on the server queue. Let us call this (new) model the DBM(K) model. Then the stability condition for the DBM(K) model will be the same as for the DBM =DBM(0) model, i.e., $\lambda(A) \leq \mu(A)$ for all subsets of job classes A, where $\mu(A)$ is the rate of arrivals of servers compatible with job classes in A in the continuous-time model, and is the probability of such an arrival in the discrete-time version. The state is $(\vec{c}_n, \vec{s}_k) \in \mathcal{X}^{DBM(K)}$, where $\vec{c}_n$ is the set of waiting jobs in arrival order, $\vec{s}_k$ is the set of waiting servers in arrival order, and the set of valid states, $\mathcal{X}^{DBM(K)}$, comprises those states $(\vec{c}_n, \vec{s}_k)$ such that $k \leq K$ and $s_i \notin S(\vec{c}_n)$, $i = 1, \ldots, k$. Again, both the job queue and the server queue are order independent, so the steady-state distribution will have the same form as that of the

noncollaborative ALIS model, though the latter includes a particular loss model for the idle-server queue, so its set of valid states is restricted to $\vec{s}_k$ such that each server appears in $\vec{s}_k$ at most once.

Theorem 3.13. *For the stable directed bipartite matching model with a finite buffer for servers K, DBM(K),*

$$\pi^{DBM(K)}(\vec{c}_n, \vec{s}_k) = \pi^{DBM(K)}(\emptyset, \emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)}, \forall (\vec{c}_n, \vec{s}_k) \in \mathcal{X}^{DBM(K)}.$$

By symmetry, a similar result holds if the server buffer is infinite, but the job buffer is bounded by some N. Now the stability condition is $\mu(B) \leq \lambda(B)$ for all subsets of server types B.

Note that the continous-time DBM(K) model also models a make-to-stock inventory system with a bipartite graph representing preferences of customer classes for certain types of items. Customers of class i are willing to purchase any of the items in S_i. Items of type j are produced according to a Poisson process at rate μ_j as long as the total number of items is less than the overall base-stock level K. Queueing customers represent back orders. Also, from 3.7, the result holds when we have different base-stock levels for different types of items.

Our results also extend to DBM models with abandonments and finite or infinite buffers. These models are appropriate for car sharing applications and other two-sided queues, where, for example, classes of jobs and types of servers correspond to location preferences. Suppose jobs (riders) of class i arrive (request rides) according to a Poisson process with rate λ_i, and will wait for an exponential time at rate γ_i before abandoning their request. Servers (drivers) of type j arrive according to a Poisson process at rate μ_j and will wait an exponential time at rate ν_j for a rider before leaving the platform. We assume a bipartite matching graph as defined earlier. Because of the abandonments, stability will not be an issue, even for infinite buffers. We have the following.

Theorem 3.14. *For the directed bipartite matching model with abandonments (DBMA) and finite or infinite buffers for jobs and servers,*

$$\pi^{DBMA}(\vec{c}_n, \vec{s}_k) = \pi^{DBMA}(\emptyset, \emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \prod_{j=1}^{k} \frac{\mu_{s_j}}{\lambda(\vec{s}_j)}, \forall (\vec{c}_n, \vec{s}_k) \in \mathcal{X}^{DBMA},$$

where

$$\mu(\vec{c}_j) = \sum_{i=1}^{j} \gamma_{c_i} + \sum_{m \in S(\vec{c}_j)} \mu_m, \quad \lambda(\vec{s}_j) = \sum_{i=1}^{j} \nu_{s_i} + \sum_{m \in C(\vec{s}_j)} \lambda_m,$$

and $\mathcal{X}^{DBMA}$ is the set of states $(\vec{c}_n, \vec{s}_k)$ such that $s_j \notin S(\vec{c}_n)$, $j = 1, \ldots, k$ and $c_i \notin C(\vec{s}_k)$, $i = 1, \ldots, n$.

Moyal, Bušić, and Mairesse show reversibility and a product-form stationary distribution for a FCFM matching model with a General (not necessarily bipartite) Matching (GM) graph, and with sequential individual (non-paired) arrivals, under a given stability condition [41]. For this model, instead of jobs and servers we have "agents" of J different classes, with agent classes corresponding to nodes in the compatibility graph; the set of agent classes compatible with class c, $S(c)$, is its set of neighbors in the compatibility graph. The set of valid states, $\mathcal{C}^{GM}$, are those states $\vec{c}_n$ such that $c_i \notin S(\vec{c}_n)$, $i = 1, \ldots, n$. Among the arrival processes Moyal et al. consider is i.i.d. arrivals where the probability of a class-c arrival is μ_c. Given the classes of unmatched agents ordered by their arrival times, $\vec{c}_n$, let $\mu(\vec{c}_n)$ be the probability the next arrival is compatible with one of those agents. Again, $\mu(\vec{c}_n)$ satisfies the OI conditions (now in discrete time), so the stationary distribution for the GM model, assuming stability, is

$$\pi^{GM}(\vec{c}_n) = \pi^{GM}(\emptyset) \prod_{i=1}^{n} \frac{\mu_{c_i}}{\mu(\vec{c}_i)} = \frac{\mu_{c_n}}{\mu(\vec{c}_n)} \pi(\vec{c}_{n-1}) \quad \text{for } \vec{c}_n \in \mathcal{C}.$$

Caldentey, Kaplan, and Weiss [19] introduced the Infinite Bipartite Matching model in which there is an infinite sequence of jobs and servers, and where the job is type i and the server is type j independently and with respective probabilities λ_i/λ and μ_i/μ. Adan and Weiss develop a Markov chain model for the matching sequence and show it satisfies partial balance and has a product-form stationary distribution, assuming the appropriate stability condition [2]. Adan et al. show that there exists a unique FCFM (first-come first-matched) matching for this model, and that the process is reversible under an "exchange transformation" that interchanges matching servers and customers [6].

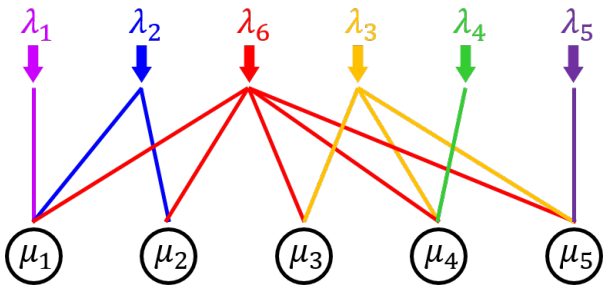


Figure 5: An example of a nested system.

4 Nested Systems and Response Time Distributions

In the previous section we developed product forms for the stationary distributions of the detailed states for variants of OI queues, but these product forms do not readily yield other important performance measures, such as response time distributions. It turns out that we will get simple, elegant results for response times in the collaborative model for a particular system structure called a nested system (see Figure 5 for an example). As noted in Observation 3.11, conditioned on the set of busy servers the noncollaborative queue state has the same sample-path evolution as the collaborative system state for a system with only the busy servers available. A consequence of this result is that our results for collaborative response times (Section 4.1) carry over to noncollaborative queueing times (Section 4.2).

Formally, a nested system is one in which, for any two job classes $i \neq j$, the sets of servers with which they are compatible, S_i and S_j, are such that $S_i \subset S_j$ or $S_j \subset S_i$ or $S_i \cap S_j = \emptyset$. This means that nested systems can be recursively defined, starting with their most flexible job class, as follows.

All nested systems have a most flexible job class, J, that is compatible with all the servers in the system, and if we remove class J from the system it decomposes into two or more nonoverlapping nested subsystems, each with its own fully flexible job class. These in turn can be decomposed by removing the fully flexible class until we get down to systems consisting of single job classes. Figure 5 shows an example of a nested system; if the fully flexible class 6 is removed, the system decomposes into one nested system consisting of servers 1 and 2 and job classes 1 and 2, and another nested system consisting of servers 3, 4, and 5 and job classes 3, 4, and 5.

We begin our response time derivations with the collaborative model, and first determine the response time of a class that is fully flexible, which, as we will see, has an exponential distribution. The derivation for the fully flexible class does not require the system to be nested, but later it will help us to develop general response times in nested systems. We note that the results for nested systems were first derived by Gardner et al. [27] using an alternative state descriptor specific to nested systems; here we provide a new derivation that follows directly from the detailed states used in Section 3.

4.1 Collaborative Model

4.1.1 Fully flexible class

In this section we will show that the fully flexible class-J jobs experience the steady-state collaborative system as if it were a multiclass M/M/1 FCFS queue with total arrival rate λ, service rate μ, and class J arrival rate λ_J. The basic intuition is that while a class-J job is in the system all servers are busy, and class-J jobs leave in the order of arrival. Note that the class-J response time in such a multiclass M/M/1 queue is stochastically the same as if they were the only jobs in an M/M/1 queue with effective service rate $\hat{\mu}_J = \mu - \sum_{i=1}^{J-1} \lambda_i = \mu - (\lambda - \lambda_J)$. That is, their response time is exponential with rate $\hat{\mu}_J - \lambda_J = \mu - \lambda$. Also observe that while there is a flexible class-J job in the system, or, more generally, while all the servers are busy, the queue "behind" the class-J job, or behind the set of jobs that make all the servers busy, operates as a multiclass M/M/1 queue. Cardinaels, Borst, and van Leeuwaarden [20] have shown that under the stability condition in heavy traffic, the collaborative system with an arbitrary bipartite compatibility matching converges to an M/M/1 multiclass queue for all classes.

We outline the argument for our result for the class-J job here, before describing the details. The response time result will follow from distributional Little's law [15] if we can show that the number of class-J jobs in steady state is the same as in an M/M/1 queue with arrival rate λ_J and service rate $\hat{\mu}_J$, (as in a multiclass M/M/1 queue) i.e., it is geometric with probability $1-\rho_J = 1-\lambda_J/\hat{\mu}_J$. We show this by first conditioning on there being at least one class-J job in the system, and showing that the conditional *total* number of jobs following the first class-J job is geometric with probability $1-\lambda/\mu$, from which it follows that the number of class-J jobs after the first class-J job is geometric with probability $1-\rho_J = \lambda_J/\hat{\mu}_J$. We then show that the probability of at least one class-J job is ρ_J, so the unconditional number of class-J jobs is also geometric with probability $1-\rho_J$. We also argue that the queue state "seen" by the first class-J job given there is one has the same distribution as the steady-state distribution of a system in which the class J job is removed. This in turn implies that the class-J effective service rate, i.e., the time from which it is the first class-J job until it leaves the system, is exponential with rate $\hat{\mu}$.

Let $\mathcal{C} = \{\vec{c}_n, n = 0, 1, ...\}$ be the set of all states $\vec{c}_n$ for the original collaborative model, and let $\vec{C}$ be a random variable representing the state of the collaborative system in steady state, i.e., $\vec{C} \sim \pi$. We use the subscript $-i$ to represent a reduced system without class i, $i = 1, ..., J$. From Corollary 3.7, we have that for $\vec{c}_n \in \mathcal{C}_{-i}$,

$$P\{\vec{C} = \vec{c}_n | \vec{C} \in \mathcal{C}_{-i}\} = \pi^C_{-i}(\vec{c}_n) = \pi^C_{-i}(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)},$$

where $\pi^C_{-i}(\emptyset) = \pi^C(\emptyset)/P\{\vec{C} \in \mathcal{C}_{-i}\}$.

Suppose there is one class, call it class J, that is fully flexible in the bipartite compatibility matching, i.e., $S_J = \{1, \ldots, M\}$. We condition on there being at least one class-J job in the system, $\vec{C} \in \mathcal{C} \backslash \mathcal{C}_{-J}$, so we know all servers will be busy. A possible state is $(\vec{c}_n, J, \vec{a}_m)$ where the first class-J job is in position $n+1$, $\vec{c}_n \in \mathcal{C}_{-J}$ represents the classes of jobs ahead of the first class-J job in order of arrival, and $\vec{a}_m \in \mathcal{C}$ represents the classes of jobs after the first class-J job in order of arrival. Then, because the denominator for all the terms corresponding to the first class-J job and the jobs after it is the total service rate μ, we have

$$\pi^C(\vec{c}_n, J, \vec{a}_m) = \pi^C(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \frac{\lambda_J}{\mu} \prod_{k=1}^{m} \frac{\lambda_{a_k}}{\mu}.$$

Let $\vec{C}_{before}$ be the conditional state of the jobs before the first class-J job, given there is such a job. Then, for $\vec{c}_n \in \mathcal{C}_{-J}$,

$$P\{\vec{C}_{before} = \vec{c}_n\} = \frac{\pi^C(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \left(\frac{\lambda_J}{\mu} \sum_{m, \vec{a}_m \in \mathcal{C}} \prod_{k=1}^{m} \frac{\lambda_{a_k}}{\mu} \right)}{\left(\frac{\lambda_J}{\mu} \sum_{m, \vec{a}_m \in \mathcal{C}} \prod_{k=1}^{m} \frac{\lambda_{a_k}}{\mu} \right) \sum_{j, \vec{c}_j \in \mathcal{C}_{-J}} \pi^C(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)}} = \frac{\pi^C(\emptyset)}{P\{\vec{C} \in \mathcal{C}_{-J}\}} \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} = \pi^C_{-J}(\vec{c}_n).$$

That is, the first class-J job "sees" the steady-state distribution for the collaborative model with class J removed. Similarly, letting $\vec{C}_{after}$ be the conditional state for the jobs after the first class-J job, given there is one, we have

$$P\{\vec{C}_{after} = \vec{a}_m\} = G \prod_{k=1}^{m} \frac{\lambda_{a_k}}{\mu},$$

where the normalizing constant is

$$G = \left(\sum_{m, \vec{a}_m \in \mathcal{C}} \prod_{k=1}^{m} \frac{\lambda_{a_k}}{\mu} \right)^{-1} = \left(\sum_{m=0}^{\infty} \prod_{k=1}^{m} \sum_{i=1}^{J} \frac{\lambda_i}{\mu} \right)^{-1} = \left(\sum_{m=0}^{\infty} \frac{\lambda}{\mu}^m \right)^{-1} = 1 - \rho$$

with $\rho = \frac{\lambda}{\mu}$. Also, given there is at least once class-J job, $\vec{C}_{before}$ and $\vec{C}_{after}$ are independent. Finally,

letting N^J be the total number of class-J jobs in the system in steady state, we have

$$\begin{aligned} P\{N^J \geq 1\} &= \frac{\lambda_J}{\mu} \sum_{j,\vec{c}_j \in \mathcal{C}_{-J}} \pi^C(\emptyset) \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \sum_{m,\vec{a}_m \in \mathcal{C}} \prod_{k=1}^{m} \frac{\lambda_{a_k}}{\mu} \\ &= \frac{\lambda_J}{\mu} P\{N^J = 0\} \frac{1}{1-\rho}. \end{aligned}$$

Solving for $P\{N^J \geq 1\} = 1 - P\{N^J = 0\}$, we obtain $P\{N^J = 0\} = \rho_J$ where $\rho_J = \frac{\lambda_J}{\mu - (\lambda - \lambda_J)}$.

Note that $P\{\vec{C}_{after} = \vec{a}_m\}$ is the same as the probability of state $\vec{a}_m$ in a multiclass M/M/1 queue with service rate μ. Let $\hat{N}$ be total number of jobs after the first class-J job in steady state. From standard results for the M/M/1 queue, we have that $\hat{N} \sim geom(1-\rho)$, where $Y \sim geom(p)$ means $P\{Y = n\} = p(1-p)^n$, $n = 0, 1, ...$ We can also obtain this result by summing the product form result above: $P\{\hat{N} = n\} = \sum_{\vec{a}_n \in \mathcal{C}} P\{\vec{C}_{after} = \vec{a}_n\}$. Each of the $\hat{N}$ jobs is independently class i with probability λ_i/λ, so $\hat{N}^J$, the number of class-J jobs after the first class-J job, is also geometrically distributed, $\hat{N}^J \sim geom(1-\rho_J)$. More generally, $\hat{N}^i$, the number of class-i jobs after the first class-J job has a geometric distribution, $\hat{N}^i \sim geom(1 - \frac{\lambda_i}{\mu - \lambda + \lambda_i})$. This is a consequence of the following simple lemma regarding Bernoulli splitting of geometric random variables, with $p = \rho = \lambda/\mu$ and $q_i = \lambda_i/\mu$; we include the proof for completeness.

Lemma 4.1. *Let $Y \sim geom(1-p)$, i.e., Y is the number of failures before the first success in i.i.d. Bernoulli trials with failure probability p. Let Y_i be the number of type-i failures before the first success in i.i.d. Bernoulli trials with success probability $1-p$ and type-i failure probability q_i, with $\sum q_i = p$, so $Y_i | Y \sim Binomial(Y, q_i/p)$. Then $Y_i \sim geom(1 - q_i/(q_i + 1 - p))$.*

Proof. When we are counting the number of type-i failures before the first success, we can ignore the other types of failures. That is, we can just look at the trials that result in either type-i failures or success. Conditioned on the trial being either a success or a type-i failure, the probability that it is a type-i failure is $q_i/(q_i + 1 - p)$. □

Because $N^J = I\{N^J > 0\}(\hat{N}^J + 1)$, and $\hat{N}^J \sim geom(1-\rho_J)$, and, as we showed above, $P\{N^J = 0\} = \rho_J$, we have the following.

Corollary 4.2. $N^J \sim geom(1-\rho_J)$.

Summarizing our observations so far, we have the following.

Theorem 4.3. *For the collaborative model with a fully flexible job class J,*

(i) *The steady-state distribution for the system conditioned on there being no class-J job is the same as that of a reduced system where there are no class-J jobs, π^C_{-J}.*

(ii) *The distribution of the state of the system ahead of the first class-J job given there is one is also π^C_{-J}.*

(iii) *The distribution of the state of the system after the first class-J job given there is one is the same as the distribution of a multiclass M/M/1 queue with arrival rate λ and service rate μ.*

(iv) *The number of class-J jobs in the system in steady state, N^J, satisfies $N^J \sim geom(1-\rho_J)$, i.e., it is the same as in an M/M/1 queue with arrival rate λ_J and service rate $\hat{\mu}_J = \mu - (\lambda - \lambda_J)$.*

Let T^i be the response time (total time in system) for a class-i job in steady state for our collaborative model, and let $T^{M/M/1}(\lambda, \mu)$ be the steady-state response time of a job in a standard M/M/1 queue with arrival rate λ and service rate μ, i.e., $T^{M/M/1}(\lambda, \mu)$ is exponentially distributed with rate $\mu - \lambda$ as long as $\lambda < \mu$. Let T^i_Q and $T^{M/M/1}_Q(\lambda, \mu)$ be similarly defined for steady-state time in queue, so $T^{M/M/1}_Q(\lambda, \mu) \sim I \cdot \text{Exp}(\mu - \lambda)$, where $I \sim Bernoulli(\lambda/\mu)$.

Corollary 4.4. *For the collaborative model with a fully flexible job class J,*

(i) $\pi^C(\emptyset) = \pi^C_{-J}(\emptyset)(1-\rho_J)$,

(ii) $T^J \sim T^{M/M/1}(\lambda_J, \hat{\mu}_J) \sim T^{M/M/1}(\lambda, \mu)$, *and* $T_Q^J \sim T_Q^{M/M/1}(\lambda_J, \hat{\mu}_J)$.

Proof. (i) From (i) and (iv) of Theorem 4.3 we have $\pi^C_{-J}(\emptyset) = \pi^C(\emptyset)/P\{N^J = 0\} = \pi^C(\emptyset)/(1-\rho_J)$.

(ii) Distributional Little's law [15] tells us that, for any λ_a and L, if the number of jobs in a queueing system is geometrically distributed with mean L, jobs arrive at rate λ_a, and jobs are served in FCFS order, then the response time is exponentially distributed with mean L/λ_a [36]. The result follows from (iv) of Theorem 4.3 with arrival rate $\lambda_a = \lambda_J$ and mean number in system $L = \frac{\rho_J}{1-\rho_J} = \frac{\lambda_J}{(\mu-(\lambda-\lambda_J))-\lambda_J} = \frac{\lambda_J}{\mu-\lambda}$. Thus, the queueing system for class-J jobs in steady state is stochastically indistinguishable from a single-class M/M/1 queue with only class-J jobs and with effective service rate $\hat{\mu}_J = \mu - (\lambda - \lambda_J)$. □

Our results for a fully flexible class in the collaborative model can be extended to general OI queues. Suppose we have an OI queue, so the service rate as a function of the ordered list of job classes, $\mu(\vec{c}_n)$, satisfies conditions (i)-(iii) of Section 3.1, and suppose there is a maximal service rate μ, such that $\mu(\vec{c}_n) \leq \mu$ for any state $\vec{c}_n$. Also suppose there is a job class J such that for any state $\vec{c}_n$ in which the first class-J job is in position k, $k \leq n$, $\mu(\vec{c}_n) = \mu(\vec{c}_k) = \mu$. Then a class-$J$ job will "block" jobs behind it in the OI queue in the same way a fully flexible job blocks jobs behind it in the skill-based collaborative queue, and Theorem 4.3 and Corollary 4.4 still hold.

4.1.2 Other classes in nested systems

Note that jobs of classes other than J will be blocked by class-J jobs in the collaborative system. Their response time therefore decomposes into an initial time to clear any class J jobs plus their response time in a subsystem in which there are no class J jobs. We use this observation to specify response time distributions for all job classes in this section.

Recall that a nested system has a fully flexible job class, J, and if class J is removed, it decomposes into two or more nonoverlapping nested subsystems. Thus, each job class i, by removing job classes j such that $S_i \subset S_j$ or $S_i \cap S_j = \emptyset$, defines a nested subsystem with servers S_i and job classes j that *require* servers $S_j \subseteq S_i$, and where class i is fully flexible. That is, for a subset S of servers, let $R(S) = \overline{C(\overline{S})} = \{1, \ldots, J\} \backslash C(\{1, \ldots, M\} \backslash S)$ be the job classes that require (i.e., that are only compatible with) servers in S. The nested subsystem defined by job class i consists of servers $k \in S_i$ and job classes $j \in R(S_i)$, i.e., the reduced system $\vdash \{1, ..., M\} \backslash S_i$. Let $\hat{\mu}_i = \mu(S_i) - \lambda(R(S_i)) + \lambda_i$ be the effective service capacity for class i in this subsystem, and let $\rho_i = \lambda_i/\hat{\mu}_i$. We will show that the overall response time for class-i jobs is the sum of the queueing times for classes j with $S_i \subset S_j$, plus the response time for class i given those classes are gone (so it is the most flexible class in its subsystem). Note that, as we observed for class J, $T^{M/M/1}(\lambda_i, \hat{\mu}_i) \sim T^{M/M/1}(\lambda(R(S_i)), \mu(S_i))$.

Theorem 4.5. *In a nested collaborative system, for any job class i,*

$$T^i \sim T^{M/M/1}(\lambda_i, \hat{\mu}_i) + \sum_{j: S_i \subset S_j} T_Q^{M/M/1}(\lambda_j, \hat{\mu}_j),$$

where all the terms are independent. Also, $\pi^C(\emptyset) = \prod_{j=1}^{J} (1-\rho_j)$.

Proof. We start with the response time result. Let class G be fully redundant in one of the subsystems obtained when class J is removed. That is, G is such that $S_j \subset S_G$ or $S_j \cap S_G = \emptyset$ for all $j \neq G, J$. We will show that $T^G \sim T_Q^J(\lambda_J, \hat{\mu}_J) + T^{M/M/1}(\lambda_G, \hat{\mu}_G)$. The result will follow by repeating the argument.

From PASTA and (iv) of Theorem 4.3, an arriving (tagged) class-G job in steady state will "see" N^J class-J jobs in the system, and it will not be able to start service until all of those N^J class-J jobs have left the system. That is, the time the tagged job must wait until the system is empty of any class-J jobs it finds on arrival is the same as the queueing time for a class-J job

$$T_Q^J \sim T_Q^{M/M/1}(\lambda_J, \hat{\mu}_J).$$

Once there are no class-J jobs, from (i) of Theorem 4.3, the tagged class-G job will "see" the reduced system in steady state, with distribution π^C_{-J}. That is, it will see independent subsystems defined by the fully

flexible class in each. The subsystems that do not include class G will have no effect on our tagged job. Hence, applying Corollary 4.4 to the subsystem with G instead of J as the most flexible class, we have that the class-G response time once there are no class-J jobs is $T^G|N^J = 0 \sim T^{M/M/1}(\lambda_G, \hat{\mu}_G)$, and the overall response time result follows.

From our earlier observations, $\pi^C_{-J}(\emptyset) = \pi^C(\emptyset)/P\{\vec{C} \in \mathcal{C}_{-J}\} = \pi^C(\emptyset)/P\{N^J = 0\} = \pi^C(\emptyset)/(1-\rho_J)$, so $\pi^C(\emptyset) = (1-\rho_J)\pi^C_{-J}(\emptyset)$. If there are no class J jobs, the system decomposes into K independent subsystems, each with its own fully flexible class, G_k, $k = 1, ..., K$, so

$$\pi^C_{-J}(\emptyset) = \prod_{k=1}^{K} \pi^C_{-(J,G_k)}(\emptyset) = \prod_{k=1}^{K} (1-\rho_{G_k})\pi^C_{-J}(\emptyset).$$

Repeating the argument within each subsystem we get $\pi^C(\emptyset) = \prod_{i=1}^{J}(1-\rho_i)$. □

As an example, consider the W model in which class-i jobs can only be served by server i, $i = 1, 2$, and class-3 jobs can be served by either server. Then

$$T^3 \sim T^{M/M/1}(\lambda_3, \mu - \lambda_1 - \lambda_2) \text{ and } T^i \sim T_Q^{M/M/1}(\lambda_3, \mu - \lambda_1 - \lambda_2) + T^{M/M/1}(\lambda_i, \mu_i),\ i = 1, 2.$$

4.1.3 Matching rates and effective service times

We now consider the matching rates and effective service times for each job class i. We define the effective service time of a (tagged) class-i job, S^i_{eff}, as the time from which it first has no class-i or higher jobs ahead of it until it completes service, in steady state. The matching probability p_{ij} is the probability a class i job is served by server j in steady state. The corresponding matching rate is $\lambda_i p_{ij}$. Though, in general, matching probabilities and rates are difficult to determine (see, e.g., [2] for a discussion of the difficulties), for nested systems the problem is tractable.

Let us first consider the fully flexible class J, and its effective service time, S^J_{eff}, where $T^J = T^J_Q + S^J_{eff}$. From Corollary 4.4, $T^J \sim T^{M/M/1}(\lambda_J, \hat{\mu}_J)$ and $T^J_Q \sim T_Q^{M/M/1}(\lambda_J, \hat{\mu}_J)$, and therefore, $S_{eff} \sim \text{Exp}(\hat{\mu}_J)$, where $\hat{\mu}_J = \mu - \lambda + \lambda_J$. Another way to see this is to notice that at the time an effective service period starts, the system the tagged class-J job sees will decompose into K independent subsystems, each with its own fully flexible class, G_k, $k = 1, ..., K$, and the tagged job will join each of those subsystems as a fully flexible job for the subsystem (viewing the collaborative model as a cancel-on-completion redundancy system). From Corollary 4.4 applied to G_k in subsystem k, the response time of the fully flexible class within the subsystem will have the same distribution as the response time in the corresponding M/M/1 queue, so

$$\begin{aligned} S^J_{eff} &= \min_{k=1,...,K}\{T^{M/M/1}(\lambda_{G_k}, \mu(S_{G_k}) - (\lambda(R(S_{G_k})) - \lambda_{G_k})\} \\ &= \min_{k=1,...,K}\{\text{Exp}(\mu(S_{G_k}) - (\lambda(R(S_{G_k})))\} \sim \text{Exp}(\mu - (\lambda - \lambda_J)), \end{aligned}$$

using the fact that the minimum of exponentials is exponentially distributed with additive rate.

Now consider the matching probabilities for class J. The probability that a tagged class-J job will be served on a server in S_{G_l} is

$$\begin{aligned} P\{T^{M/M/1}(\lambda_{G_l}, \mu(S_{G_l}) - (\lambda(R(S_{G_l})) - \lambda_{G_l}) &= \min_{k=1,...,K}\{T^{M/M/1}(\lambda_{G_k}, \mu(S_{G_k}) - (\lambda(R(S_{G_k})) - \lambda_{G_k})\} \\ &= \frac{\mu(S_{G_l}) - (\lambda(R(S_{G_l}))}{\sum_k[\mu(S_{G_k}) - \lambda(R(S_{G_k}))]} = \frac{\mu(S_{G_l}) - \lambda(R(S_{G_l}))}{\mu - (\lambda - \lambda_J)}. \end{aligned}$$

The argument can be applied recursively for any class. The effective service time of class i is its effective service time in the subsytem in which class i is fully flexible, so $S^i_{eff} \sim \text{Exp}(\hat{\mu}_i)$. Similarly, letting G^i_k, $k = 1, ...K^i$, be the independent nested subsystems that result when class i and all more flexible job classes are removed, the probability that a tagged class-i job will be served on a server in $S_{G^k_l}$ is

$$\frac{\mu(S_{G^k_l}) - \lambda(R(S_{G^k_l}))}{\mu(S_i) - (\lambda R((S_i)) - \lambda_i)}.$$

Let $i_1, ..., i_l = i$ be defined as the sequence of fully flexible classes in each of the nested subsystems of class i that use server j. That is, i_1 is such that $j \in S_{i_1}$ and there is no class k such that $S_k \subset S_{i_1}$, and, for $m = 2, ..., l$, i_m is such that $S_{i_{m-1}} \subset S_{i_m}$ and $S_k \nsubseteq S_{i_m}$ for all $k \neq i_1, ..., i_{m-1}$. Then, from the argument above, we have the following.

Proposition 4.6. *In the collaborative nested system, for any class i, the effective service rate is $S^i_{eff} \sim Exp(\hat{\mu}_i)$ and the matching probability of class i to server j is*

$$p_{ij} = \frac{\mu_j}{\mu(S_{i_1})} \prod_{k=2}^{l} \frac{\mu(S_{i_{k-1}}) - \lambda(R(S_{i_{k-1}}))}{\mu(S_{i_k}) - (\lambda R((S_{i_k})) - \lambda_{i_k})}.$$

The corresponding matching rate is $\lambda_{ij} = \lambda_i p_{ij}$.

For the example of Figure 5, $S^6_{eff} \sim \text{Exp}(\mu - \lambda + \lambda_6) = \min\{\text{Exp}(\mu_1 + \mu_2 - \lambda_1 - \lambda_2), \text{Exp}(\mu_3 + \mu_4 + \mu_5 - \lambda_3 - \lambda_4 - \lambda_5)\}$, and

$$p_{61} = \frac{\mu_1}{\mu_1} \frac{\mu_1 - \lambda_1}{\mu_1 - \lambda_1 + \mu_2} \frac{\mu_1 + \mu_2 - \lambda_1 - \lambda_2}{\mu - \lambda + \lambda_6}.$$

4.2 Noncollaborative Model

Let $T^i_{Q|B}$ be the stationary time in the job queue for a class-i job in the noncollaborative model, given that the set of busy servers is $B = \{1, ..., M\}$ (i.e., all servers are busy). Then, from Observation 3.11, we know $T^i_{Q|B}$ has the same distribution as the response time for class-i jobs in the collaborative model. Therefore, from Theorem 4.5, we have

Theorem 4.7. *In a nested noncollaborative system, for any job class i, given busy servers $B = \{1, ..., M\}$,*

$$T^i_{Q|B} \sim T^{M/M/1}(\lambda_i, \hat{\mu}_i) + \sum_{j: S_i \subset S_j} T^{M/M/1}_Q(\lambda_j, \hat{\mu}_j),$$

where $\hat{\mu}_j = \mu(S_j) - \lambda(R(S_j)) + \lambda_j$ and all the terms are independent.

The result can be generalized for class i, if some servers are idle but all the servers in S_i are busy, as follows. Fix i and the set of busy servers $B \supseteq S_i$, and let y be such that $S_i \subseteq S_y \subseteq B$ and $\nexists j \neq y$ such that $S_y \subset S_j \subseteq B$. That is, class y determines a nested subsystem of busy servers in which class y is fully flexible, and there are no jobs of class j such that $S_y \subset S_j$ in the job queue. Therefore, an arriving class-i job sees a reduced system, $\vdash \{1, ..., M\} \backslash S_y$, consisting only of the servers in S_y and job classes $j \in R(S_y)$, and in which all the servers in S_y are busy. We have the following.

Corollary 4.8. *In a nested noncollaborative system, for any job class i, given the servers in $B \supseteq S_i$ are busy,*

$$T^i_{Q|B} \sim T^{M/M/1}(\lambda_i, \hat{\mu}_i) + \sum_{j: S_i \subset S_j \subseteq S_y} T^{M/M/1}_Q(\lambda_j, \hat{\mu}_j),$$

where $\hat{\mu}_j = \mu(S_j) - \lambda(R(S_j)) + \lambda_j$ and all the terms are independent.

From Proposition 4.6 we also have the following.

Corollary 4.9. *In a nested noncollaborative system, for any job class i, given the servers in $B \supseteq S_i$ are busy, the conditional response time satisfies*

$$T^i_B \sim T^{M/M/1}(\lambda_i, \hat{\mu}_i) + \sum_{j: S_i \subset S_j \subseteq S_y} T^{M/M/1}_Q(\lambda_j, \hat{\mu}_j) + I_{ik} Exp(\mu_j),$$

where I_{ik} is Bernoulli with probability equal to the matching probability for type-i jobs to server k, p_{ik}, given in Proposition 4.6, and all the terms are independent.

Finally, we can obtain unconditional response time distributions for the special case in which the service rate is the same at all servers; that is, $\mu_j = \mu/M$ for all servers $j = 1, \ldots, M$. We do this by conditioning on the set of busy servers seen by an arriving job. If an arriving class-i job finds an idle compatible server, it will immediately enter service; otherwise it must wait in the job queue before entering service. Define $y_1, y_2, ...y_l$ such that $S_i = S_{y_1} \subset S_{y_2} \subset ... \subset S_{y_l} = \{1, ..., M\}$ and such that for all $j \neq y_1, ..., y_l$, $S_j \subset S_i$ or $S_i \cap S_j = \emptyset$. Thus, classes y_i define the nested subsystems containing S_i. Let $I(S_{y_k})$ be the indicator that all the servers in S_{y_k} are busy, but not all the servers in $S_{y_{k+1}} \backslash S_{y_k}$ are busy. Then we have the following, where T^i is the (unconditional) class-i response time.

Corollary 4.10. *In a nested noncollaborative system with homogeneous service rates, for any job class i,*

$$T^i \sim Exp(\mu/M) + \sum_{k=1}^{l} I(S_{y_k}) \left(T^{M/M/1}(\lambda_i, \hat{\mu}_i) + \sum_{i=1}^{k} T_Q^{M/M/1}(\lambda_{y_i}, \hat{\mu}_{y_i}) \right),$$

where $\hat{\mu}_j = \mu |S_j|/M - \lambda(R(S_j)) + \lambda_j$ and all the terms are independent.

It is a challenge to get more complete response time results for the noncollaborative model. First, for a job that finds multiple compatible servers idle, if service rates are heterogeneous, we need to further condition on the server on which the job runs. Under RAIS this is determined probabilistically according to the assignment rule; the probabilities can be determined using the process described in [45]. Under ALIS this is determined by which server has been busy longer. Second, for a job that finds all compatible servers busy we still need to determine the server on which the job ultimately runs. This is the probability that a class-i job completes on server j in the collaborative model, i.e., the matching probability; this would then be equal to the probability that a class-J job runs on server i in the noncollaborative model.

A final challenge in the NC model is computing the probability that various subsets of servers are busy. These computations are aided by the use of Corollary 3.4 and the simple form of $\pi^C(\emptyset)$ for nested systems, but are still complicated. Though the analysis is tractable in certain small nested systems, for example, in the W model, the form of these probabilities is not particularly clean or intuitive. In larger nested systems, we believe that the probabilities needed to perform the requisite conditioning are unlikely to have a clean closed form solution, even for a symmetric nested system.

5 Partial State Aggregation and Conditional Queueing Times

Section 4 provides one approach for understanding the form of the per-class response time distributions in nested systems. In this section, we turn to a second approach that uses an alternative, partially aggregated, state description, which gives us conditional queueing times, given the busy servers in the order of the jobs they are serving, for general, possibly non-nested, systems. Like the detailed states considered in Section 3, the partially aggregated states also provide a Markov description for the model and also yield a product form stationary distribution.

5.1 Noncollaborative Model

Instead of tracking the classes of all jobs in the system, we now track the number of jobs in the queue in between jobs in service, but not their individual classes. Let l denote the number of jobs currently in service. The partially aggregated state includes the vector $\vec{n}_l = (n_1, \ldots, n_l)$, where n_i denotes the number of jobs in the queue (*not* in service) that arrived after the ith job in service and before the $(i+1)$st job in service. Under both ALIS and RAIS, we track the busy servers in the arrival order of the jobs they are serving, (but *not* the classes of the jobs in service); for the ALIS version we also track the idle servers in the order in which they became idle.

The partially aggregated state description for noncollaborative models was first introduced by Adan, Visschers, and Weiss [1, 45]. In these papers, the stationary distribution was derived directly using partial balance for the partially aggregated states. In this section we provide an alternative derivation that involves aggregating the stationary probabilities for the detailed states discussed in Section 3.

Let us first consider the noncollaborative model with the RAIS policy, in which arrivals finding multiple idle compatible servers are assigned to a server at random with appropriate probabilities that depend on the

set of busy servers. The partially aggregated state is $(\vec{b}_l, \vec{n}_l)$, where l is the number of busy servers (which in the noncollaborative model is the same as the number of jobs in service), $\vec{b}_l$ is the set of busy servers in order of the arrival times of the jobs they are serving, and n_i, $i = 1, ..., l$ is the number of jobs waiting for one of busy servers $1, \ldots, l$. Thus, server b_1 is serving the oldest job, the next n_1 jobs to have arrived are waiting for (require) server b_1, i.e., their classes are in $R(b_1)$, the $n_1 + 2^{nd}$ oldest job is being served by b_2, the next n_2 jobs are only compatible with b_1 or b_2 or both, i.e. their classes are in $R(\vec{b}_2)$, and so on. The corresponding detailed state, $\vec{z}_m$, is such that $z_1 = b_1$, $z_{n_1+2} = b_2$, etc., and $m = l + \sum_{i=1}^{l} n_i$. Thus, in the partially aggregated state $(\vec{b}_l, \vec{n}_l)$ there are l jobs in service and $\sum_{i=1}^{l} n_i$ jobs in the queue. When the set of busy servers is $\vec{b}_j$, let $\lambda_s^a(\vec{b}_j)$ represent the activation rate of idle server $s \notin \{b_1, ..., b_j\}$ (the rate of going from state $(\vec{b}_j, \vec{n}_j)$ to $((\vec{b}_j, s), (\vec{n}_j, 0))$.

With this description of the state we defer determining the class of a job until we need it. That is, we realize information about a job's class only when a server becomes available and the job under consideration is next in the queue behind the available server. At this point, we probabilistically determine whether or not the job is compatible with the server; if it is compatible, it enters service. If not, the server "skips over" the job and we have narrowed down the set of possible classes for the job, but we may have not specified its exact class.

Visschers et al. found that the above state space exhibits a product form stationary distribution, under the *assignment condition* for routing a compatible job to idle server b_j given busy servers $\vec{b}_{j-1}$: $\prod_{j=1}^{l} \lambda_{b_j}^a(\vec{b}_{j-1})$ must be the same for any permutation of $b_1, \ldots, b_l$. This is the same assignment condition that we use in Section 3 for the detailed state description.

Let $\alpha(\vec{b}_j) = \frac{\lambda(R(\vec{b}_j))}{\mu(\vec{b}_j)}$. We use the notation $\pi^{RAIS'}$ to denote the partially aggregated stationary distribution under RAIS (in contrast with π^{RAIS}, which denotes the stationary distribution for the detailed state description).

Theorem 5.1. *(Visschers et al. [45])*

$$\pi^{RAIS'}(\vec{b}_l, \vec{n}_l) = \pi^{RAIS}(\emptyset) \prod_{j=1}^{l} \frac{\lambda_{b_j}^a(\vec{b}_{j-1})}{\mu(\vec{b}_j)} \alpha(\vec{b}_j)^{n_j} = \pi^{RAIS}(\emptyset) \prod_{j=1}^{l} \alpha(\vec{b}_j)^{n_j} \prod_{j=1}^{l} \frac{\lambda_{b_j}^a(\vec{b}_{j-1})}{\mu(\vec{b}_j)}$$

$$= \pi^{RAIS}(\vec{n}_l|\vec{b}_l)\pi^{RAIS}(\vec{b}_l) = \prod_{j=1}^{l}(1 - \alpha(\vec{b}_j))\alpha(\vec{b}_j)^{n_j}\pi^{RAIS}(\vec{b}_l).$$

The proof given by Visschers et al. involves showing directly that local balance holds for the partially aggregated states. Below we give an alternative proof, which follows by summing the stationary probabilities (given in Theorem 3.10) of states $\vec{z}_m$ that are consistent with $(\vec{b}_l, \vec{n}_l)$.

Proof. We begin by recalling that $\vec{z}_m$ is an interleaving of states $\vec{c}_n$ and $\vec{b}_l$, where $m = n + l$.

That is, letting $k_i = \sum_{j=1}^{i} n_j$, we can write

$$\vec{z}_m = (b_1, c_1, \ldots, c_{n_1}, b_2, \ldots, b_j, c_{k_j+1}, \ldots, c_{k_{j-1}+n_j}, b_{j+1}, \ldots, b_l, c_{k_{l-1}+1}, \ldots, c_{k_{l-1}+n_l}).$$

Let $\mathcal{C}(\vec{b}_l, \vec{n}_l)$ denote the set of states $\vec{z_m}$ compatible with $(\vec{b}_l, \vec{n}_l)$. We then have

$$\begin{aligned}
\pi^{RAIS'}(\vec{b}_l, \vec{n}_l) &= \sum_{\vec{z}_m \in C(\vec{b}_l, \vec{n}_l)} \pi^{RAIS}(\vec{z}_m) \\
&= \pi^{RAIS}(\emptyset) \sum_{\vec{z}_m \in C(\vec{b}_l, \vec{n}_l)} \prod_{i=1}^{m} \frac{\lambda_i^z(\vec{z}_i)}{\mu(\vec{z}_i)} \\
&= \pi^{RAIS}(\emptyset) \sum_{\vec{z}_m \in C(\vec{b}_l, \vec{n}_l)} \prod_{j=1}^{l} \left(\frac{\lambda_{b_j}^a(\vec{b}_{j-1})}{\mu(\vec{b}_j)} \prod_{i=k_{j-1}+1}^{k_{j-1}+n_j} \frac{\lambda_{c_i}}{\mu(\vec{b}_j)} \right) \\
&= \pi^{RAIS}(\emptyset) \prod_{j=1}^{l} \left(\frac{\lambda_{b_j}^a(\vec{b}_{j-1})}{\mu(\vec{b}_j)} \prod_{i=k_{j-1}+1}^{k_{j-1}+n_j} \frac{\sum_{c \in R(\vec{b}_j)} \lambda_c}{\mu(\vec{b}_j)} \right) \\
&= \pi^{RAIS}(\emptyset) \prod_{j=1}^{l} \left(\frac{\lambda_{b_j}^a(\vec{b}_{j-1})}{\mu(\vec{b}_j)} \left(\frac{\lambda(R(\vec{b}_j))}{\mu(\vec{b}_j)} \right)^{n_j} \right) \\
&= \pi^{RAIS}(\emptyset) \prod_{j=1}^{l} \alpha(\vec{b}_j)^{n_j} \prod_{j=1}^{l} \frac{\lambda_{b_j^a}(\vec{b}_{j-1})}{\mu(\vec{b}_j)}.
\end{aligned}$$

□

We now turn to the ALIS policy, in which arrivals finding multiple idle compatible servers are assigned to the one that has been idle longest. Because the order of the idle servers now affects the system evolution, we now consider the aggregate state $(\vec{s}_{M-l}, \vec{b}_l, \vec{n}_l)$, where $\vec{s}_{M-l}$ is the set of $M-l$ idle servers in the order in which they became idle, and $\vec{b}_l$ and $\vec{n}_l$ are defined as in the RAIS model. Again we have a product form for the partially aggregated state description.

Theorem 5.2. *(Adan and Weiss [1])*

$$\pi^{ALIS'}(\vec{s}_{M-l}, \vec{b}_l, \vec{n}_l) = G^{ALIS'} \prod_{j=1}^{l} \alpha(\vec{b}_j)^{n_j} \prod_{j=1}^{l} \frac{1}{\mu(\vec{b}_j)} \prod_{j=1}^{M-l} \frac{1}{\lambda(\vec{s}_j)}$$

where $G^{ALIS'}$ is a normalizing constant.

As noted by Adan and Weiss [1], aggregating the stationary distribution under ALIS over all permutations of the idle servers, $\vec{s}_k$, yields the same stationary distribution as under RAIS.

Corollary 5.3. $\sum_{\mathcal{P}(\vec{s}_{M-l})} \pi^{ALIS'}(\vec{s}_{M-l}, \vec{b}_l, \vec{n}_l) = \pi^{RAIS'}(\vec{b}_l, \vec{n}_l)$.

Corollary 5.3 tells us that the conditional stationary distribution of the time in queue, given the set of busy servers, is the same under ALIS as under RAIS. Under both policies, conditioned on $\vec{b}_l$, the number of jobs waiting in the queue between busy servers b_j and b_{j+1} is geometrically distributed with parameter $1 - \alpha(R(\vec{b}_j)) = 1 - \lambda(R(\vec{b}_j))/\mu(\vec{b}_j)$, from Theorem 5.1. Moreover, each of these jobs is of class c with probability $\lambda_c/\lambda(R(\vec{b}_j))$. Therefore, from Lemma 4.1, conditioned on $\vec{b}_l$, the number of class-i jobs waiting in the queue between busy servers b_j and b_{j+1} is geometrically distributed with parameter $1 - \frac{\lambda_i}{\mu(\vec{b}_j) - \lambda(R(\vec{b}_j)) + \lambda_i}$. Hence, from distributional Little's law, and because class-i jobs are served in order, the time a class-i job will spend in the "subsystem" behind the servers $\vec{b}_j$ is the same as the response time for a standard M/M/1 queue with arrival rate λ_i and service rate $\mu(\vec{b}_j) - \lambda(R(\vec{b}_j)) + \lambda_i$. Therefore, depending on $\vec{b}_l$, a job arriving in steady state will either start service immediately, or wait a sum of exponential times before entering service.

Theorem 5.4. *The queueing time for a class i job, given busy servers $\vec{b}_l$, is*

$$T_Q^i(\vec{b}_l) = I(i \in R(\vec{b}_l)) \sum_{j=f(i,\vec{b}_l)}^{l} T^{M/M/1}(\lambda_i, \mu(\vec{b}_j) - \lambda(R(\vec{b}_j)) + \lambda_i)$$

where $I(\cdot)$ is the indicator function, and $f(i,\vec{b}_l) = \arg\min\{j : 0 \le j \le l, i \in R(\vec{b}_j)\} = \arg\max\{j : 0 \le j \le l, b_j \in S(i)\}$ is the largest indexed busy machine that is compatible with job class i.

We note that, unlike the results for nested systems derived in Section 4, here the per-class queueing time distribution is conditioned on the ordered vector of busy servers. In general it is not straightforward to obtain closed-form expressions for the probability that a job sees a particular $\vec{b}_l$. Hence, while this form is insightful in terms of interpreting the time in queue as that in a tandem series of M/M/1 queues, the form does not permit an easy derivation of mean time in queue or other exact performance metrics.

5.2 Collaborative Model

In this section we note briefly that a similar partially aggregated state can be defined for the collaborative model. Here the partially aggregated state description is $(\vec{d}_l, \vec{n}_l)$, where $\vec{d}_l$ gives the classes of all jobs currently in service, and $\vec{n}_l$ gives the number of jobs in the queue (receiving no service) in between those jobs in service. This is similar to the $(\vec{b}_l, \vec{n}_l)$ state used for RAIS, except that now we track the classes of the job in service rather than the servers processing these jobs. We define $\mu(\vec{d}_i)$ as the total service rate given to the first i jobs that are receiving service, and $R(\vec{d}_i)$ as the classes of jobs that require one of the servers serving the jobs in $\vec{d}_i$. That is, $c \in R(\vec{d}_i)$ if $S_c \in S(\vec{d}_i)$. Note that for d_i to be in service, given $\vec{d}_{i-1}$, we must have $d_i \notin \vec{d}_{i-1}$. Let $\alpha(\vec{d}_i) = \frac{\lambda(R(\vec{d}_i))}{\mu(\vec{d}_i)}$. We use the superscript C' to denote the partially aggregated stationary distribution in the collaborative system.

Proposition 5.5. *For $l = 0, \ldots, M$, $\vec{d}_l$ such that $d_i \notin \vec{d}_{i-1}$ for $i = 2, \ldots, l$, and $n_i = 0, 1, \ldots$ for $i = 1, \ldots, l$,*

$$\pi^{C'}(\vec{d}_l, \vec{n}_l) = \pi^C(\emptyset) \prod_{j=1}^{l} \frac{\lambda_{d_j}}{\mu(\vec{d}_j)} \alpha(\vec{d}_j)^{n_j}.$$

The proof follows a similar argument to the proof of Theorem 5.1 by aggregating detailed states consistent with $(\vec{d}_l, \vec{n}_l)$; we omit the details.

As an example, consider the N model, in which class 1 jobs can only be served by server 1 and class 2 jobs can be served on either server. Then $\alpha(1) = \frac{\lambda_1}{\mu_1}$, $\alpha(2) = \alpha(1,2) = \frac{\lambda}{\mu}$, and, from Theorem 4.5, $\pi^C(\emptyset) = (1 - \frac{\lambda_1}{\mu_1})(1 - \frac{\lambda_2}{\mu - \lambda_1})$, so we have the following corollary.

Corollary 5.6. *For the N model, $\pi^{C'}(\emptyset) = (1 - \frac{\lambda_1}{\mu_1})(1 - \frac{\lambda_2}{\mu - \lambda_1})$ and*

$$\begin{aligned}
\pi^{C'}((1), n) &= \left(1 - \frac{\lambda_1}{\mu_1}\right)\left(1 - \frac{\lambda_2}{\mu - \lambda_1}\right)\left(\frac{\lambda_1}{\mu_1}\right)^{n+1} \\
\pi^{C'}((2), n) &= \left(1 - \frac{\lambda_1}{\mu_1}\right)\left(1 - \frac{\lambda_2}{\mu - \lambda_1}\right)\frac{\lambda_2}{\mu}\left(\frac{\lambda}{\mu}\right)^{n} \\
\pi^{C'}((1,2), n_1, n_2) &= \left(1 - \frac{\lambda_1}{\mu_1}\right)\left(1 - \frac{\lambda_2}{\mu - \lambda_1}\right)\left(\frac{\lambda_1}{\mu_1}\right)^{n_1+1}\frac{\lambda_2}{\mu}\left(\frac{\lambda}{\mu}\right)^{n_2}.
\end{aligned}$$

We can also aggregate for a collaborative model with abandonments, where jobs abandon after an exponential time with rate γ, as summarized below.

Proposition 5.7. *For $l = 0, \ldots, M$, $\vec{d}_l$ such that $d_i \notin \vec{d}_{i-1}$ for $i = 2, \ldots, l$, and $n_i = 0, 1, \ldots$ for $i = 1, \ldots, l$,*

$$\pi^{C'}(\vec{d}_l, \vec{n}_l) = \pi^C(\emptyset) \prod_{j=1}^{l} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) + \gamma(\sum_{i=1}^{j-1} n_i + 1)} \left(\prod_{k=1}^{n_j} \frac{\lambda(R(\vec{d}_j))}{\mu(\vec{d}_j) + \gamma(\sum_{i=1}^{j-1} n_i + k)} \right).$$

Results similar to the proposition and corollary above were derived recently for the N model by directly checking the balance equations for the aggregated states [21].

As for the noncollaborative system, we can use Proposition 5.5 and the distributional form of Little's Law to determine the distribution of $T_Q^i(\vec{d}_l)$, the conditional waiting time until a job starts service on at least one server for any class i, given $\vec{d}_l$.

Corollary 5.8.

$$T_Q^i(\vec{d_l}) = I(i \in R(\vec{d_l})) \sum_{j=f(i,\vec{d_l})}^{l} T^{M/M/1}(\lambda_i, \mu(\vec{d_j}) - \lambda(R(\vec{d_j})) + \lambda_i)$$

where $I(\cdot)$ is the indicator function, and $f(i,\vec{d_l}) = \arg\min\{j : 0 \le j \le l, i \in R(\vec{d_j})\} = \arg\max\{j : 0 \le j \le l, S(d_j) \cup S(i) \neq \emptyset$ is the largest indexed job in service that is using a server in S_i.

Corollary 5.9. $T_Q^i \sim \sum_l \sum_{\vec{d_l}} T_Q^i(\vec{d_l}) I(\vec{d_l})$.

5.3 Collaborative Model: Alternative Partial Aggregation

For the collaborative model our detailed state was $\vec{c}_n$, listing the classes of all the jobs in the system (both in service and waiting for service) in order of their arrival. In the partially aggregated states of Section 5.2, we track the classes of the jobs *in service*, as well as the number of jobs in the queue in between those jobs in service. We now consider an alternative partially aggregated state description, $(\vec{d}_m, \vec{n}_m)$, where $m \le J$ is the number of distinct classes of jobs *in the system*, $\vec{d}_m$ is redefined to list those classes in order of arrival of the first of each class, and n_i gives the number of jobs between the first job of class d_i and the first job of class d_{i+1} (or after the first job of class m for $i = m$) [20, 46]. Therefore, $n = m + \sum_{i=1}^{m} n_i$. For the example of Figure 3, with $\vec{c}_n = \vec{c}_6 = (1,2,3,2,4,1)$, the corresponding aggregated state is $(\vec{d}_m, \vec{n}_m) = (\vec{d}_4, \vec{n}_4) = (1,2,3,4;0,0,1,1)$. We call $\vec{d}_m$ the *class profile* of the state. Because of the FCFS service discipline, only jobs corresponding to the class profile $\vec{d}_m$ can be in service (though these jobs are not necessarily in service, because some of them may be blocked by earlier arrivals among $\vec{d}_m$). Also, given $\vec{d}_i$, if a job of class c is among the n_i jobs immediately after the first class-d_i job, we must have $c \in \vec{d}_i$. This aggregated state description was introduced by Weiss [46] for the corresponding directed bipartite matching (DBM) model described in Section 3.6.

The aggregated state $(\vec{d}_m, \vec{n}_m)$ still gives a Markov description of the system, but, as with the aggregated states of Section 5.2, we defer determining the class of a job until we need it. That is, we realize information about a job's class, for a job among the n_i jobs, only when one of the jobs corresponding to $\vec{d}_i$ completes. For example, if a job of class d_i completes (necessarily the first class-d_i job in the system, for this model), then we probabilistically determine whether the first of the n_i jobs following it is class d_i (with probability $\lambda_{d_i}/\lambda(\vec{d}_i)$), in which case it becomes the new "first" job of class d_i and n_i becomes $n_i - 1$. Otherwise, that job gets added to the n_{i-1} jobs after the first class-d_{i-1} job, and we check the next job (if any) of the n_i jobs after the departing class d_i job, and so on.

Let $\mathcal{C}(\vec{d}_m, \vec{n}_m)$ be the set of detailed states $\vec{c}_n$ that are consistent with $(\vec{d}_m, \vec{n}_m)$, and let $\pi^{C'}(\vec{d}_m, \vec{n}_m)$ be the stationary distribution of the aggregated state description for the collaborative model. Then we have

$$\begin{aligned}
\pi^{C'}(\vec{d}_m, \vec{n}_m) &= \sum_{\vec{c}_n \in \mathcal{C}(\vec{d}_m, \vec{n}_m)} \pi^C(\vec{c}_n) = \pi^C(\emptyset) \sum_{\vec{c}_n \in \mathcal{C}(\vec{d}_m, \vec{n}_m)} \prod_{i=1}^{n} \frac{\lambda_{c_i}}{\mu(\vec{c}_i)} \\
&= \pi^C(\emptyset) \prod_{j=1}^{m} \left[\frac{\lambda_{d_j}}{\mu(\vec{d}_j)} \left(\sum_{c \in \vec{d}_j} \frac{\lambda_c}{\mu(\vec{d}_j)} \right)^{n_i} \right] = \pi^C(\emptyset) \prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j)} \left(\frac{\lambda(\vec{d}_j)}{\mu(\vec{d}_j)} \right)^{n_j}.
\end{aligned}$$

We therefore have the following, letting $\rho(\vec{d}_j) = \lambda(\vec{d}_j)/\mu(\vec{d}_j)$.

Proposition 5.10. *(Weiss [46]) For $m = 0, \ldots, M$, $\vec{d}_m$ a permutation of a subset of $\{1, 2, ..., J\}$, and $n_i = 0, 1, \ldots$ for $i = 1, \ldots, m$,*

$$\pi^{C'}(\vec{d}_m, \vec{n}_m) = \pi^C(\emptyset) \prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j)} \rho(\vec{d}_j)^{n_j},$$

where $\pi^{C'}(\emptyset) = \pi^C(\emptyset)$.

Let $\vec{D} = (D_1, ..., D_K)$ be a random class profile in steady state, and let $\vec{N} = (N_1, ..., N_K)$ be the vector counting jobs between the jobs in the class profile in steady state, i.e., $(\vec{D}, \vec{N}) \sim \pi^{C'}(\vec{d}_m, \vec{n}_m)$. We have the following corollary.

Corollary 5.11. *Given $\vec{D} = \vec{d}_m$, $N_i \sim geom(1 - \rho(\vec{d}_i))$ and N_i and N_j are independent for $i = 1, ..., m$, $j \neq i$. Also,*

$$P(\vec{D} = \vec{d}_m) = \pi^C(\emptyset) \prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)}.$$

Therefore, $\pi^C(\emptyset) = \left[\sum_{\vec{d}_m} \prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right]^{-1}$.

Proof.

$$\begin{aligned}
\pi^{C'}(\vec{d}_m, \vec{n}_m) &= \pi^C(\emptyset) \prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j)} \rho(\vec{d}_j)^{n_j} \\
&= \pi^C(\emptyset) \prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j)(1 - \rho(\vec{d}_j))} \rho(\vec{d}_j)^{n_j} (1 - \rho(\vec{d}_j)) \\
&= \pi^C(\emptyset) \left(\prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right) \left(\prod_{j=1}^{m} \rho(\vec{d}_j)^{n_j} (1 - \rho(\vec{d}_j)) \right) \\
&= P(\vec{D} = \vec{d}_m) P(\vec{N} = \vec{n}_m | \vec{D} = \vec{d}_m).
\end{aligned}$$

□

Corollary 5.11 enables us to obtain a simple explicit form for mean performance metrics.

Corollary 5.12. *The mean number of class-i jobs in the system, $E[N^i]$, is given by*

$$E[N^i] = \sum_{\vec{d}_m : i \in \vec{d}_m} \pi^C(\emptyset) \left(\prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right) \left(1 + \sum_{j=f_d(i)}^{m} \frac{\lambda_i}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right),$$

where $f_d(i) = k$ such that $d_k = i$, i.e., $f_d(i)$ is the position of class i in $\vec{d}_m$.

Proof. Let N_j^i denote the number of class-i jobs among the n_j jobs that appear in the queue between jobs d_i and d_{i+1}. We have

$$\begin{aligned}
E[N^i] &= \sum_{m, \vec{d}_m} E[N^i | \vec{D} = \vec{d}_m] \cdot P(\vec{D} = \vec{d}_m) \\
&= \sum_{m, \vec{d}_m : i \in \vec{d}_m} \pi^C(\emptyset) \left(\prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right) \left(1 + \sum_{j=f_d(i)}^{m} E[N_j^i | \vec{D} = \vec{d}_m] \right) \\
&= \sum_{\vec{d}_m : i \in \vec{d}_m} \pi^C(\emptyset) \left(\prod_{j=1}^{m} \frac{\lambda_{d_j}}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right) \left(1 + \sum_{j=f_d(i)}^{m} \frac{\lambda_i}{\mu(\vec{d}_j) - \lambda(\vec{d}_j)} \right),
\end{aligned}$$

where $P(\vec{D} = \vec{d}_m)$ is given in Corollary 5.11 and $E[N_j^i | \vec{D} = \vec{d}_m]$ follows from Lemma 4.1. □

The per-class mean response times then follow from Little's Law. While we have yet to solve for $\pi^C(\emptyset)$, in Section 6 we will see in Proposition 6.2 that Bonald et al. [17] provide a recursive approach to computing $\pi^C(\emptyset)$.

We note that the partial aggregation results for the collaborative model generalize easily to aggregate both jobs and servers in the directed bipartite matching DBM(K) model where servers arrive and can wait in a finite buffer.

6 Per-class State Aggregation and Mean Performance Measures

In this section we consider class-based performance measures for the OI queue (and hence the collaborative model, and for the queue of the noncollaborative model given all servers busy) with general, non-nested, bipartite structures. Here it is useful to define the per-class aggregated state $x = (x_1, ..., x_N)$, where x_i is the total number of type i jobs. Let $C(x)$ be set of states of the form $\vec{c}_n$ that are consistent with x, i.e., $\vec{c}_n \in C(x)$ if and only if $x_i = \sum_{j=1}^{n} I\{c_j = i\}$ for all classes i. so $n = \sum x_i$. Abusing notation, let $\mu(x)$ be the total service rate in state x; from OI property (ii) (see Section 3), $\mu(x) = \mu(\vec{c}_n)$ for all states $\vec{c}_n \in C(x)$. Then $\pi^X(x) = \sum_{\vec{c}_n \in C(x)} \pi(\vec{c})$, where $\pi^X(x)$ is the steady-state probability of aggregate state x and, by aggregating the partial balance equations, is given by

$$\mu(x)\pi^X(x) = \sum_{i:x_i>0} \lambda_i \pi^X(x - e_i),$$

where e_i is a vector of appropriate length containing a 1 in position i and 0's elsewhere. Then $\pi^X(x)$ also has a product form. The result follows from summing the product form characterizations of $\pi^X(\vec{c})$.

Theorem 6.1. *(Bonald and Comte [16], Krzesinski [39])The aggregate steady-state distribution satisfies*

$$\pi^X(x) = \pi^C_X(\emptyset)\Phi(x)\prod_{i=1}^{N} \lambda_i^{x_i},$$

where

$$\Phi(x) = \frac{1}{\mu(x)} \sum_{i:x_i>0} \Phi(x - e_i),\ \Phi(\emptyset) = 1,$$

and $\pi^X(\emptyset) = \pi^C(\emptyset)$ is a normalizing constant equal to the probability that the system is empty.

Note that the aggregate state description does not capture the dynamics of the OI queue and is not a Markov description for the original system. While the order of the jobs, given x, does not matter for the *total* service rate, $\mu(x)$, it does matter for the amount of service received by the j'th job in the queue, $\Delta_j(\vec{c}_j) = \Delta_j(\vec{c}_n)$. We therefore need to know $\vec{c}_n$—or at least the class of the job in service on each server—to know the rate out of the state due to a class-i departure.

As observed by Bonald and Comte, the stationary aggregate distribution $\pi^C_X(x)$ given in Theorem 6.1 is also the stationary distribution of a single-server system consisting of N job classes with Poisson arrivals at rates λ_i and state-dependent exponential service rates such that class-i jobs are served according to processor sharing at rate

$$\phi_i(x) = [\Phi(x - e_i)/\Phi(x)]I\{x_i > 0\}.$$

We call the single-server model with service rates $\phi_i(x)$ the *aggregate model*. Bonald and Comte also noted the following relationship between the aggregate model and the collaborative model:

$$\phi_i(x) = \sum_{\vec{c}_n \in C(x)} \frac{\pi^C(\vec{c}_n)}{\pi^X(x)} \mu'_i(\vec{c}_n),$$

where $\mu'_i(\vec{c}_n)$ is the service rate of the first class-i job in state $\vec{c}_n$ in the collaborative model. (Because of the FCFS service discipline, no other class-i jobs will be in service.) Note that from Theorem 6.1, $\sum_i \phi_i(x) = \mu(x)$.

The service rates $\phi_i(x)$ also satisfy the following *balance property*:

$$\phi_i(x)\phi_j(x - e_i) = \phi_j(x)\phi_i(x - e_j)$$

for $x_i, x_j > 0$. The balance property is analogous to the assignment condition required for the noncollaborative model with random assignment to idle servers to have a product-form stationary distribution. Furthermore, the balance property leads to Kolmogorov's criterion being satisfied, and therefore it leads to the system being reversible. Because the aggregate model is reversible, it is also insensitive to the job size distributions.

While the stationary distribution of the states x is the same in the aggregate model and the collaborative model, as noted above, the underlying system dynamics are very different. Bonald and Comte propose applying a round robin-like scheduling algorithm to the original collaborative model to approximate the behavior of the aggregate model [16]. Under their algorithm, server j serves the first compatible job in the queue, as in the original, collaborative, FCFS model, but, after an exponential time with rate θ_j, server j interrupts the job in service and that job is moved to the back of the queue. This is analogous to approximating processor sharing with round robin for a single server and job class. Bonald and Comte note that the aggregate model, using balanced fair processor sharing, is insensitive in that the steady-state distribution does not depend on the job size distribution.

Because the aggregate model and the collaborative model have the same steady-state distribution for the aggregate states, we can use the aggregate model to efficiently compute aggregate performance measures for the collaborative model. In particular, Bonald et al. [17] give a recursion based on successively removing servers for computing the system idle probability, $\pi^C(\emptyset) = \pi^X(\emptyset)$, as follows.

Let $\mathcal{C}$ be the set of all (detailed) states $\vec{c}_n$ for the original collaborative model. Recall that the subscript $\vdash k$ represents a reduced system without server k, i.e., in which server k as well as the job classes in C_k are removed. Let ψ_k be the probability that server k is idle in the original collaborative system. Then, from Corollary 3.8, we have that $\pi^C(\vec{c}_n|$ server k is idle$) = \pi^C_{\vdash k}(\vec{c}_n)$ for $\vec{c}_n \in \mathcal{C}_{\vdash k}$, so $\pi^C(\vec{c}_n) = \pi^C_{\vdash k}(\vec{c}_n)\psi_k$. Then $\pi^C(\emptyset) = \pi^X(\emptyset) = \pi^C_{\vdash k}(\emptyset)\psi_k$ (and hence ψ_k) can be computed recursively:

Proposition 6.2. *(Bonald et al. [17])*

$$\pi^C(\emptyset) = \pi^X(\emptyset) = (1-\rho)\frac{\mu}{\sum_{k=1}^M \frac{\mu_k}{\pi^C_{\vdash k}(\emptyset)}},$$

where $\rho = \lambda/\mu$ is the system load.

Proof. Algebra, using $\pi^C_{\vdash k}(\emptyset) = \pi^C(\emptyset)/\psi_k$, gives us that the equation above is equivalent to

$$\sum_{k=1}^M \mu_k\psi_k = \mu - \lambda.$$

This just represents two ways of computing the long-run rate of "dummy" transitions, i.e., potential service completions at idle servers. □

Recall that the collaborative model is equivalent to the directed bipartite matching model, in which servers of type k arrive according to a Poisson process at rate μ_k, and arriving servers that do not find compatible jobs (unmatched servers) immediately leave the system. Here the interpretation of ψ_k is the probability that an arriving server of type k is unmatched, and "dummy" transitions correspond to arrivals of unmatched servers. See Weiss [46] for an alternative algorithm to compute ψ_k and $\pi^C(\emptyset)$, as well as for computing the long-run matching rates of class i jobs with class k servers.

The mean number of jobs L and the mean number of class-i jobs L_i, with $L_{\vdash k}$ and $L_{i\vdash k}$ similarly defined for the reduced system without server k and its compatible job classes, can be similarly recursively calculated. Note that $L_{\vdash k}$ and $L_{i\vdash k}$ are also the conditional mean number of class-i jobs in the original system, given server k is idle. Let $\bar{S}_i = \{1,\ldots,M\}\backslash S_i$ denote the set of servers that cannot serve class-i jobs, and let $\rho_i = \lambda_i/(\mu-(\lambda-\lambda_i))$ be the mean number of class-i jobs in an M/M/1 queue with arrival rate λ_i and service rate $\mu-(\lambda-\lambda_i)$.

Proposition 6.3.

$$L_i = \frac{\lambda_i + \sum_{k\in\bar{S}_i}\mu_k\psi_k L_{i|-k}}{\mu-\lambda} = \frac{\lambda_i}{\mu-\lambda} + \sum_{k\in\bar{S}_i}\frac{\mu_k\psi_k}{\mu-\lambda}L_{i|-k}$$

$$L = \sum_{i=1}^J L_i = \frac{\lambda + \sum_{k=1}^M \mu_k\psi_k L_{\vdash k}}{\mu-\lambda} = \frac{\lambda}{\mu-\lambda} + \frac{\sum_{k=1}^M \mu_k\psi_k L_{\vdash k}}{\sum_{k=1}^M \mu_k\psi_k}.$$

Note that $\frac{\lambda_i}{\mu-\lambda}$ is the mean number of class-i jobs in an M/M/1 queue with arrival rate λ_i and service rate $\mu-(\lambda-\lambda_i)$ (the maximal service rate available to class-i jobs), as we argued in Section 4. It also represents the mean number of class-i jobs in the collaborative (not necessarily nested) model given all the servers are busy. Also, as noted above, $\mu_k\psi_k/(\mu-\lambda)$ represents the proportion of dummy transitions due to server k being idle, so the second set of terms in the above expression represent the additional expected jobs due to "wasted" service because of job/server incompatibilities. Note that servers in S_i will not be idle if there are class i jobs in the collaborative system.

Bonald et al. use the results above to obtain explicit results for special cases, such as redundancy-d, where all jobs are replicated to a randomly chosen subset of d servers, and line structures in which job classes can be ordered so that for any server k, the classes of jobs it can serve are consecutive, i.e., classes $i_1, i_1+1, \ldots, i_2$ for some $i_1 < i_2$ [17]. Nested structures are a special case of line structures so the above recursions represent an alternative method for deriving mean performance metrics to the approach given in Section 4, where, of course, mean response times follow immediately from Little's Law.

We note that though the results for this section are for the collaborative model, in light of our observation that the queue process in the noncollaborative model given all servers are busy has the same distribution as the overall process for the collaborative model, we can apply the results above to the noncollaborative case. That is, e.g., L_i as computed above will equal the expected number of class-i jobs in the queue (not receiving service) in steady state, given all the servers are busy, for the noncollaborative model.

7 Related Work

The majority of this paper has focused on surveying results related to product-form stationary distributions and derivations of performance metrics in the collaborative and noncollaborative systems, under a few key assumptions: that service times are exponentially distributed and i.i.d. (across jobs and across replicas of the same job, in the collaborative model), and that the service discipline is FCFS. There are several lines of work that relax one or more of these assumptions; such relaxations preclude product-form results, and as such fall outside the scope of this paper. In this section we provide a brief outline of some of the related work.

We begin with related work within the i.i.d. exponential model. Several papers have considered a scheduling policy that gives priority to less flexible jobs over more flexible jobs; this policy is known as "dedicated customers first" in the noncollaborative system and as "least redundant first" in the collaborative system. Such a policy has been shown to be optimal in the sense that it stochastically maximizes the departure rate, in both the noncollaborative system [7] and the collaborative system [27]. Furthermore, in the collaborative case mean response time is decreasing and convex under this policy as the proportion of jobs that are more flexible increases [28]. In a similar vein, the effect of increasing the "degree" of flexibility (i.e., the number of servers with which each job is compatible) in systems with FCFS scheduling has been studied. In both the noncollaborative [8] and collaborative [26, 38] systems, mean response time is decreasing and convex as the degree of redundancy increases. Gurvich and Whitt [30, 31, 32] consider other routing and scheduling policies for the noncollaborative model in the many-server heavy-traffic regime.

The system in which all jobs have the same degree of flexibility and are assigned a set of compatible servers uniformly at random is one special case of the system structure considered in this paper. This special case, often referred to as a "redundancy-d" system (the d indicating the degree of flexibility), has received considerable attention in the literature because the symmetric system structure makes analysis more feasible in many cases. Indeed, the redundancy-d system is another example of a system in which, under the i.i.d. exponential and FCFS assumptions of this paper, it is feasible to aggregate the product-form stationary distribution to derive performance metrics [11, 26].

Several papers have noted the lack of realism of the i.i.d. exponential assumptions, and have found that relaxing them can change the system performance significantly. For example, mean response time no longer decreases as d increases in the collaborative redundancy-d model [29] when the processing times of jobs are correlated across servers. The stability region can also be quite different when the i.i.d. exponential assumptions are relaxed [9, 34, 40, 42, 43, 44].

Finally, there has been work on optimally routing and scheduling, rather than following FCFS, for queueing models with bipartite job class/server compatibilities. See the paper by Chen, Dong and Shi in

this issue for a survey [22].

8 Conclusion

This paper presents an overview of product-form results in systems with flexible jobs and servers, in which a bipartite graph structure specifies which job classes can be served by which servers. We primarily focus on two models for service: the collaborative model, in which multiple servers can work together to serve a single job at a faster rate, and the noncollaborative model, in which each job is permitted to enter service on only one server. Both models have been studied extensively in the literature; this survey brings together the two models, as well as several other related systems, using a common language and set of notation. Our hope is that this will allow readers to draw new connections among these similar systems. Along the way, we have presented several new results that highlight the relationships between models and that show how results derived in one model can be used to obtain insights for the other.

One of the primary goals in analyzing queueing systems is to determine response time distributions; in multi-class systems such as those considered in this paper, we wish to derive per-class response time distributions. Each of the three state descriptors that we consider allows us to make partial progress towards this goal. Using the detailed state descriptor of Section 3, we can derive per-class response time distributions for the special case of nested systems. Using the partially aggregated states of Section 5, we can derive per-class queueing time distributions in general (not necessarily nested) systems, but now conditioned on the ordered set of busy servers. Using the per-class aggregated states of Section 6, we can derive unconditional per-class mean performance metrics in general systems, but this approach does not yield distributional results. Each approach has its advantages and disadvantages; we believe that a unifying analysis that provides exact unconditional per-class response time distributions is likely to be infeasible, but this remains an open question.

9 Acknowledgements

We thank Gideon Weiss, Erol Pekoz, Jan-Pieter Dorsman, Yoni Nazarathy, and an anonymous referee for their careful reading and valuable feedback.

References

[1] Adan, I. J. B. F., and G. Weiss. (2014) A skill based parallel service system under FCFS-ALIS – steady state, overloads, and abandonments. Stoch. Syst., 4(1): 250–299.

[2] Adan I. J. B. F., and G. Weiss. (2012) Exact FCFS matching rates for two infinite multi-type sequences. Oper. Res. 60(2):475–489.

[3] Adan, I. J. B. F., and G. Weiss. (2012) A loss system with skill-based servers under assign to longest idle server policy. Prob. Eng. Inf. Sci. 26: 307-321.

[4] Adan I. J. B. F., C. Hurkens, and G. Weiss (2010) A reversible Erlang loss system with multitype customers and multitype servers. Prob. in the Eng. and Inf. Sci. 24: 535-548.

[5] Adan, I. J. B. F., I. Kleiner, R. Righter, and G. Weiss. (2018) FCFS parallel service systems and matching models. Perf. Eval. 127: 253-272.

[6] Adan, I. J. B. F., A. Bušić, J. Mairesse, and G. Weiss. (2018) Reversibility and further properties of FCFS infinite bipartite matching, Math of OR, 43: 598-621.

[7] Akgun, O., R. Righter, and R. Wolff. (2013) Partial flexibility in routing and scheduling. Adv. Appl. Prob., 45: 637-691.

[8] Akgun, O., R. Righter, and R. Wolff. (2011). Understanding the marginal impact of customer flexibility. Queueing Systems 71(1-2): 5-23.

[9] Anton, E., U. Ayesta, M. Jonckheere, and I.M. Verloop (2019) On the stability of redundancy models. Preprint (https://arxiv.org/pdf/1903.04414.pdf).

[10] Ayesta, U., T. Bodas, and I.M. Verloop. (2018) On redundancy-d with cancel-on-start a.k.a Join-shortest-work (d), MAMA Workshop, SIGMETRICS.

[11] Ayesta, U., T. Bodas, and I.M. Verloop. (2018) On a unifying product form framework for redundancy models, IFIP Performance.

[12] Ayesta, U., T. Bodas, J. L. Dorsman, and I. M. Verloop. (2019) A token-based central queue with order-independent service rates. arXiv Preprint, arXiv:1902.02137.

[13] Berezner, S.A., C.F. Kriel, and A.E. Krzesinski (1995). Quasi-reversible multiclass queues with order independent departure rates, Queueing Systems, 19:345-359.

[14] Berezner, S.A. and A.E. Krzesinski (1996). Order independent loss queues, Queueing Systems, 23:331–335.

[15] Bertismas, D. and D. Nakazato (1995). The distributional Little's law and its applications, Operations Research, 43:2,298–310.

[16] Bonald, T., and Comte, C. (2017). Balanced fair resource sharing in computer clusters, Perform. Eval. 116: 70–83.

[17] Bonald, T., C. Comte, and F. Mathieu (2019). Performance of balanced fairness in resource pools: A recursive approach, ACM SIGMETRICS Perform. Eval. Rev. 46: 125-127.

[18] Borst, S.C., Boxma, O.J., Morrison, J.A., and R. Nuñez Queija (2003) The equivalence between processor sharing and service in random order. Operations Research Letters 31: 254–262.

[19] Caldentey, R., E. H. Kaplan, and G. Weiss (2009) FCFS infinite bipartite matching of servers and customers. Adv. Appl. Probab. 41(3):695-730.

[20] Cardinaels, E., S. Borst, and J.S.H. van Leeuwaarden (2020) Redundancy scheduling with locally stable compatibility graphs. Preprint. https://arxiv.org/pdf/2005.14566.pdf

[21] Castro, F., Nazerzadeh, H., and C. Yan (2020) Matching queues with reneging: a product form solution. Queueing Systems https://doi.org/10.1007/s11134-020-09662-y

[22] Chen, J., J. Dong, and P. Shi (2020) A Survey on Skill-Based Routing with Applications to Service Operations Management. Queueing Systems.

[23] Comte, C. (2019) Dynamic load balancing with tokens, Computer Communications 144: 76-88.

[24] Comte, C., Dorsman, J.-P. (2020) Pass-and-swap queues. Preprint.

[25] Gardner, K., S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytia, and A. Scheller-Wolf. (2016). Queueing with redundant requests: exact analysis. Queueing Systems 83:227-259.

[26] Gardner, K., M. Harchol-Balter, A. Scheller-Wolf, M. Velednitsky, and S. Zbarsky. (2017). Redundancy-d: The power of d choices for redundancy. Operations Research 65:4, 1078-1094.

[27] Gardner, K., M. Harchol-Balter, E. Hyytia, and R. Righter. (2017). Scheduling for efficiency and fairness in systems with redundancy. Performance Evaluation 116:1-25.

[28] Gardner, K., E. Hyytia, and R. Righter. (2019) A little redundancy goes a long way: Convexity in redundancy systems. Performance Evaluation 131:22-42.

[29] Gardner, K., M. Harchol-Balter, A. Scheller-Wolf, and B. Van Houdt. (2017) A better model for job redundancy: Decoupling server slowdown and job size. Transactions on Networking 25(6): 3353-3367.

[30] Gurvich, I., and W. Whitt (2009). Scheduling flexible servers with convex delay costs in many-server service systems. Manufacturing and Service Operations Management, 11(2): 237—253.

[31] Gurvich, I., and W. Whitt (2009). Queue-and-idleness-ratio controls in many-server service systems. Mathematics of Operations Research, 34(2): 363—396.

[32] Gurvich, I., and W. Whitt (2010). Service-level differentiation in many-server service systems via queue-ratio routing. Operations Research, 58(2): 316—328.

[33] Haji, B. and S.M. Ross (2015). A queueing loss model with heterogeneous skill based servers under idle time ordering policies. J. Appl. Prob. 52: 269-277.

[34] Hellemans, T., and B. van Houdt. (2018). Analysis of redundancy(d) with identical replicas, IFIP Performance.

[35] Jackson, J. (1957) Networks of waiting lines. Operations Research 5:516-523.

[36] Keilson, J., and L.D. Servi (1988) A distributional form of Little's Law. OR Letters 7: 223-227.

[37] Kelly, F.P. (1979). *Stochastic Networks and Reversibility*, Wiley, Chichester, UK.

[38] Kim, Y., R. Righter, and R. Wolff. (2009) Job replication on multiserver systems. Adv. Appl. Prob. 41: 546-575.

[39] Krzesinski, A.E. (2011) Order independent queues, in: R.J. Boucherie, N.M. van Dijk (Eds.), *Queueing Networks: A Fundamental Approach*, Springer, Boston, MA, USA: 85–120.

[40] Mendelson, G. (2020) A lower bound on the stability region for redundancy-d with FIFO service discipline. Preprint: https://arxiv.org/pdf/2004.14793.pdf

[41] Moyal, P., A. Bušić and J. Mairesse, (2019) A product form and a sub-additive theorem for the general stochastic matching model. Annals of Probability, submitted.

[42] Raaijmakers, Y., S. Borst, and O. Boxma.(2019) Redundancy scheduling with scaled Bernoulli service requirements. Queueing Systems 93: 67–82.

[43] Raaijmakers, Y., S. Borst, and O. Boxma.(2020) Stability of Redundancy Systems with Processor Sharing. VALUETOOLS '20: Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools: 120–127.

[44] Raaijmakers, Y. and S. Borst. (2020) Achievable Stability in Redundancy Systems. Preprint: https://arxiv.org/pdf/2008.03478.pdf.

[45] Visschers, J., Adan, I. J. B. F., and Weiss, G. (2012). A product form solution to a system with multi-type customers and multi-type servers. Queueing Systems 70: 269–298.

[46] Weiss, G. (2019). Directed FCFS infinite bipartite matching. Preprint.