HAL
open science

# A unifying framework for redundancy models: product form and impact of independence assumption

Urtzi Ayesta, Tejas Bodas, Ina Maria Maaike Verloop

## To cite this version:

Urtzi Ayesta, Tejas Bodas, Ina Maria Maaike Verloop. A unifying framework for redundancy models: product form and impact of independence assumption. 2018. hal-01713937v1

**HAL Id: hal-01713937**

**https://hal.science/hal-01713937v1**

Preprint submitted on 21 Feb 2018 (v1), last revised 22 May 2018 (v2)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A unifying framework for redundancy models: product form and impact of independence assumption

Urtzi Ayesta[1,3,4,5], Tejas Bodas *[,2,3] and Ina Maria Verloop[1,3]

[1]CNRS, IRIT, 2 rue C. Camichel, 31071 Toulouse, France
[2]CNRS, LAAS, 7 avenue du colonel Roche, 31400 Toulouse, France
[3]Univ. de Toulouse, LAAS, INP, 31400 Toulouse, France
[4]IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain
[5]UPV/EHU, University of the Basque Country, 20018 Donostia, Spain

## Abstract

In this paper, we present a unifying analysis for redundancy systems with cancel-on-start (*c.o.s.*) and cancel-on-complete (*c.o.c.*) with exponentially distributed service requirements. With *c.o.s.* (*c.o.c.*) all redundant copies are removed as soon as one of the copies starts (completes) service. As a consequence, *c.o.s.* does not waste any computing resources, as opposed to *c.o.c.*.

We show that the *c.o.s.* model is equivalent to a queueing system with multi-type jobs and servers, which was analyzed in [1], and show that *c.o.c.* (under the assumption of i.i.d. copies) can be analyzed by a generalization of [1] where state-dependent departure rates are permitted. This allows us to show that the stationary distribution for both the *c.o.c.* and *c.o.s.* models have a product form. We give a detailed first-time analysis for *c.o.s* and derive a closed form expression for important metrics like mean number of jobs in the system, and probability of waiting. Comparing the performance of *c.o.s.* with that of *c.o.c.* gives the unexpected result (since *c.o.s.* does not waste any resources), that *c.o.s.* is worse in terms of mean number of jobs. The latter illustrates that the i.i.d. assumption (together with exponentially distributed requirements) might lead to conclusions that are qualitatively different from that observed in practice. We also show that the *c.o.s.* model is equivalent to Join-Shortest-Work queue with redundancy (JSW($d$)). In the latter, an incoming job is dispatched to the server with smallest workload among $d$ randomly chosen ones. Thus, all our results apply mutatis-mutandis to JSW($d$).

## 1 Introduction

Using redundancy to minimize latency in parallel server systems has become very popular in recent years [2–7]. While there are several variants of a redundancy-based system, the general notion of redundancy is to create multiple copies of the same job that will be sent to a subset of servers. By allowing for redundant copies, the aim is to minimize the system latency by exploiting the variability in the queue lengths of the different queues. Several recent works, both empirically [2, 3, 8, 9] and theoretically [4–6, 10–12], have provided indications that redundancy can help in reducing the response time of a system.

Broadly speaking, depending on when replicas are deleted, we can consider two classes of redundancy systems: cancel-on-start (*c.o.s*) and cancel-on-completion (*c.o.c*). In redundancy systems with *c.o.c*, once one of the copies has completed service, the other copies are deleted and the job is said to have received service. On the other hand, in redundancy systems with *c.o.s*, copies are deleted as soon as one copy starts being served. From a practical point of view, if servers have similar computing capacity the *c.o.s* system is preferable: both configurations require the same amount of signaling among servers, but the *c.o.s* system does not waste any computation resources. In some concrete applications, for instance in the context of organ transplants where people list themselves in multiple waiting-lists across various states, *c.o.s.* is clearly closer to reality than *c.o.c.*. In spite of this, most of the recent literature focuses on systems with *c.o.c*, and *c.o.s.* has remained largely elusive to exact analysis.

In a recent series of papers, Gardner et al. [6,7] have provided a thorough analysis of redundancy systems with *c.o.c* in a system with $K$ servers each with their own queue. In the redundancy-$d$ model of [7], redundant copies of an arriving job are sent to $d \leq K$ homogeneous servers chosen uniformly at random. Under the additional assumptions that service times are exponentially distributed and that the redundant copies are independent and identically distributed (we call this the *independence assumption*), Gardner et al. have shown that the steady-state distribution has a product form. Further, they derive the mean response time in closed form and it is shown that the stability region, i.e., the arrival rate for which steady state exists, does not reduce. The latter

*Corresponding author: tejaspbodas@gmail.com.
Research partially supported by the French "Agence Nationale de la Recherche (ANR)" through the project ANR-15-CE25-0004 (ANR JCJC RACON)

seems counter-intuitive, because having multiple copies of the same job should imply more work for the servers and as a consequence a reduction of the stability region. The reason that this does not happen is due to the *independence assumption* made in [7]: the exponential assumption together with i.i.d. copies creates the situation where the effective departure rate of a job is $d$ times faster than that in a single server. Our objective in this paper is to further assess the impact of this *independence assumption* on the performance.

In this paper, we provide, to the best of our knowledge, the first analysis on a redundancy-$d$ system with *c.o.s* for $1 \leq d \leq K$. We assume exponentially distributed service times, but copies do not need to be i.i.d. We adopt as a benchmark the same underlying multi-server topology as the one of [7]. This model is very convenient, everything is symmetric and homogeneous and this permits to isolate the impact of the redundancy scheme. We first show that *c.o.s* is equivalent to a system with a single central queue and multi-type jobs and multiple servers, as analyzed in Visschers et. al. [1]. This allows us to conclude that the steady-state distribution of *c.o.s* is of product-form. In addition, we obtain an expression for the generating function for the number of waiting jobs in the system and for the mean number of jobs in the system.

We then show that redundancy-$d$ with *c.o.s* is equivalent to Join-Shortest-Work queue with redundancy (JSW($d$)). In the latter, an incoming job is dispatched to the server with smallest workload among $d$ randomly chosen ones. The redundancy model hence represents a method of implementing JSW($d$) without requiring to know the residual work in each of the queues. In addition, performance measures obtained for the *c.o.s* model carry over to JSW($d$).

We then extend and adapt the model of Visschers et al. [1] in such a way that, using a new state descriptor (different from [1]), the redundancy systems with *c.o.c* and i.i.d. copies can be analyzed.

Even though *c.o.s* does not waste any computation resources, a comparison of the two redundancy-$d$ systems yields the surprising result that for any value of the redundancy parameter, the performance of *c.o.c* is better than that of *c.o.s*. The reason for this is the independence assumption in the analysis of *c.o.c*, which together with exponentially distributed jobs makes *c.o.c.* artificially efficient. We comment more on this in Example 1 of Section 3 and Section 8.

We believe that our results open an important avenue for research, as it establishes the link between redundancy models and the central queue model as analyzed in [1]. A summary of our main contribution is given below:

1. We obtain a unifying approach to derive the stationary distribution for both *c.o.c* and *c.o.s* model, which is of product form (Proposition 1 and Proposition 6.).

2. We provide the first exact analysis for the *c.o.s* model and derive the generating function for the number of waiting jobs and its mean (in Proposition 2). A key interpretation from the generating function is the fact that the number of waiting jobs is a mixture of sums of geometric random variables.

3. By allowing redundant copies that cancel-on start, we achieve the performance of JSW($d$) without knowing the job sizes and the workload in the servers (Proposition 1).

4. We also give the first exact analysis for the performance under JSW($d$) (Proposition 3).

5. In Proposition 5, we generalize the modeling framework of [1] and use this generalization to provide an alternative analysis of redundancy systems with *c.o.c.*

6. Numerically we observe that the impact of the modeling assumption of independent copies is non-negligible.

The rest of the paper is organized as follows. In the next section, we look at some of the related work and discuss the model and preliminaries in Section 3. We analyze the *c.o.s.* model in Section 4, its equivalence to JSW($d$) in Section 5 and consider some asymptotic regimes in Section 6. We give an alternate product form for the *c.o.c.* model in Section 7 followed by some numerical results in Section 8.

## 2 Related work

The main motivation to investigate redundancy models comes from empirical evidence suggesting that redundancy can help improve the performance in real-world applications. For example Vulimiri et al. [3] illustrate the advantages of redundancy in a DNS query network where a host computer can query multiple DNS servers simultaneously to resolve a name. Deal et al. [9] note that Google's big table services use redundancy in order to improve latency.

Most of the literature deals with the *c.o.c* model with exponentially distributed service times and the *independence assumption*. Gardner et al. [6,7] have provided a comprehensive analysis of redundancy *c.o.c* queueing models. In [6] the authors consider a class-based model where redundant copies of an arriving job type are dispatched to a type-specific subset of servers, and show that the steady state distribution has a product form. In [7], the previous result is applied to analyze a multi-server model with homogeneous servers where incoming jobs are dispatched to randomly selected $d$ servers. Bonald et al. [13,14] have shown that the *c.o.c.* model under the *independence assumption* and the balance fairness allocation yields the same steady-state distribution. An important result in the *c.o.c.* model with exponential service times and i.i.d. redundant copies is that the necessary condition for stability, i.e. that the arrival rate per server is smaller than the service rate, is also sufficient, see both [13] and [7]. An important observation made in [13] is that the *c.o.c* model is a special case of the Order Independent queue [15], which provides

a direct path to derive the steady-state distribution. However we note that the departure rates from a state under the *c.o.s* model fail to satisfy the order independence condition, which in turn implies the necessity of a different approach from that used in [13] to analyze *c.o.s*. We give more details in Section 7.

A more general redundancy model is the so-called $(n, k, r)$ proposed and studied by Joshi et al. [16]. In this model there are $n$ servers, $r$ redundant copies and a job is said to be served if $k$ copies are served. For this model, in [16], the authors analyze the impact of the service time distribution on the performance. Shah et al. [5] have showed that under i.i.d memoryless service the average response time reduces as $r$ increases, and that $r = n$ minimizes the average latency among all possible redundancy policies.

To the best of our knowledge, the only results available for the *c.o.s* model have appeared in [16, Section 4] and [12, Section 9]. In both cases, the analysis is restricted to the particular case in which redundancy copies are sent to all the servers.

Regarding the independence assumption, Gardner et al. [12] propose a more accurate model in which the service time of a redundant copy is decoupled into two components, one related to the inherent job size of the task, and the other related to the server's slowdown. As written in the abstract of [12] "The independent runtime model is unrealistic, and has led to theoretical results which can be at odds with computer systems implementation results." One of our main contributions is to provide concrete examples of this statement, and to provide an exact analysis for a redundancy model that does not need the independence assumption.

Redundancy models have also been studied in other application domains. For example an analysis of redundant models in the context of storage systems is provided in [10], and Ananthanarayanan et al. [2,8] have studied the benefits of redundancy in straggler mitigation in data centers and cloud computing.

# 3 Model, Preliminaries and Notation

In this section, we will first describe the redundancy-$d$ model with *c.o.s.* and *c.o.c.*, give some preliminaries, as well as discuss the impact of the independence assumption by looking at the case $d = K$. This is followed by a brief description of the multi-type job and multi-server model of Visschers et al. [1] which forms the basis for our analysis of the two redundancy models.

## 3.1 Redundancy-$d$ with *c.o.s.* and *c.o.c.*

The redundancy-$d$ model consists of $K$ homogeneous servers each with a first-in first-out (FIFO) queue. The service rate of each server is denoted by $\mu$. Jobs arrive according to a Poisson process with rate $\lambda$ and have an exponentially distributed service requirement with unit mean. An arriving job chooses $d$ of $K$ servers uniformly at random and sends $d$ copies of the same job to these *feasible* servers.

Under *cancel-on-start (c.o.s.)*, once any of the copies is taken for service, the remaining copies are canceled immediately. Further, on arrival of a job, if more than one of its $d$ *feasible* servers are idle, then the job is served immediately at one of the idle *feasible* servers based on a predefined *assignment rule* and the remaining copies are canceled. For this *c.o.s.* model, a natural *assignment rule* is the *uniform assignment rule*, that is, choose an idle *feasible* server uniformly at random.

Under *cancel-on-complete (c.o.c.)*, when one of the $d$ copies of a job completes its service (hence the name cancel-on-complete), the other copies are removed. In addition, when referring to *c.o.c.*, we assume that the service requirements of the copies of the same job are independent. We have referred to this assumption as *the independence assumption*.

Before giving the preliminaries, we first illustrate the impact of the independence assumption for the particular case of $d = K$.

**Example 1.** ***Case*** $d = K$***: impact of independence assumption:*** *When redundant copies are independent, the c.o.c. model is equivalent to a single-server queue with arrival rate $\lambda$ and service rate $\mu K$.*[1] *In particular, this means that the mean number of jobs in the system is $\rho/(1-\rho)$ where $\rho := \frac{\lambda}{K\mu}$. However, if instead we assume that all copies have* exactly the same service requirement*, for $d = K$, the c.o.c. model reduces to $K$ completely synchronized single-server queues, each having an arrival rate of $\lambda$ and a service rate of $\mu$.*[2] *The stability condition is now $\lambda < \mu$, i.e., $\rho < \frac{1}{K}$, and thus considerably smaller than the stability condition under the independence assumption ($\rho < 1$). In addition, without the independence assumption, the mean number of jobs increases significantly to $\rho K/(1 - K\rho)$.*

*Turning now to the c.o.s. model, when $d = K$, with and without the independence assumption, this model is equivalent to an $M/M/K$ queue (see Section 9 for details). We can thus conclude that with independent copies, the c.o.c. model significantly outperforms the c.o.s. model. However, when this assumption is removed, the performance of c.o.c. degrades severely, as the stability condition is more stringent than for c.o.s. This illustrates well the unfair advantage due to the independence assumption.*

Recall that redundancy models were created to improve the performance by sending copies to $d$ servers simultaneously. However, for *c.o.c.* and without the *independence assumption*,

[1] This can be seen as follows: starting from an empty queue, a first job will have copies in all $K$ servers. A copy will hence finish at rate $\mu K$, after which a new job starts being served in all servers.

[2] A job will be served simultaneously in all $K$ servers. Since the service requirement is identical, all copies finish after an exponential time with mean $\mu$.

implementing full redundancy ($d = K$) gives much *worse* performance than when no redundancy is done ($d = 1$). Note that $d = 1$ is Bernoulli routing, hence, gives rise to $K$ independent $M/M/1$ queues with rates $\lambda/K$ and $\mu$, while $d = K$ results in $K$ synchronized single-servers with rates $\lambda$ and $\mu$. This inefficiency comes from the fact that under *c.o.c.* without the *independence assumption*, all servers work on copies that have *the exact same size*. Under *c.o.s.*, such inefficiency does not arise. This therefore shows the importance to study *c.o.s.* when one assumes identical copies.

### 3.1.1 Preliminaries

We define the total load as $\rho = \frac{\lambda}{K\mu}$ and assume $\rho < 1$. For both *c.o.s.* and for *c.o.c.*, $\rho < 1$ is also the stability condition, that is, the condition under which a stationary distribution exists. This was proved in [6] for *c.o.c.* For *c.o.s.* it follows as well: by symmetry, each server serves a fraction $1/K$ of the jobs. Since the service rate of a server is $\mu$, each server is stable if $\lambda/K < \mu$, i.e., $\rho < 1$.

*Mean number of busy servers:* Let $p(i)$ denote the probability that there are $i$ servers busy. Using the rate balance principle, we have (both for *c.o.s.* and for *c.o.c.*)

$$\lambda = \mu \sum_{i=0}^{K} ip(i). \tag{1}$$

Hence, the mean number of busy servers in the redundancy-$d$ system is $\sum_{i=0}^{K} ip(i) = \frac{\lambda}{\mu} = \rho K$.

*Probability of an idle server:* We now identify the probability of an arbitrary server (say server 1) being idle. Let $P_1$ denote the probability that server 1 is idle. Given $i$ busy servers, let $q_i$ denote the probability that server 1 is idle. We have,

$$\begin{aligned} P_1 &= \sum_{i=0}^{K} p(i) q_i \\ &= \sum_{i=0}^{K} p(i) \frac{\binom{K-1}{i}}{\binom{K}{i}} \\ &= 1 - \sum_{i=0}^{K} p(i) \frac{i}{K} \\ &= 1 - \rho, \quad \text{from Eq. (1).} \end{aligned}$$

### 3.1.2 Central queue architecture

An important point to note is that both these redundancy-$d$ models have a *central queue architecture* with multiple servers. While a job in the traditional representation is identified by its $d$ copies in the $d$ feasible servers, the same job is represented only once in the central queue (albeit with a type label to indicate its $d$ *feasible* server). The jobs in the *central queue* are ordered in a FIFO manner (and hence requires the knowledge of the sequence in which the arrivals came). An important interpretation with the *central queue architecture* is that the servers on completion of service will select the next feasible job from the FIFO central queue (skipping over infeasible jobs). Here a *feasible job* for a server are those jobs that have the server to be *feasible*. Note that in *c.o.c.*, multiple servers can serve a single job in the central queue while in *c.o.s.*, each server can serve only one job. See Fig. 1 for the representation of a central queue in terms of *c.o.c.* and *c.o.s.* models.

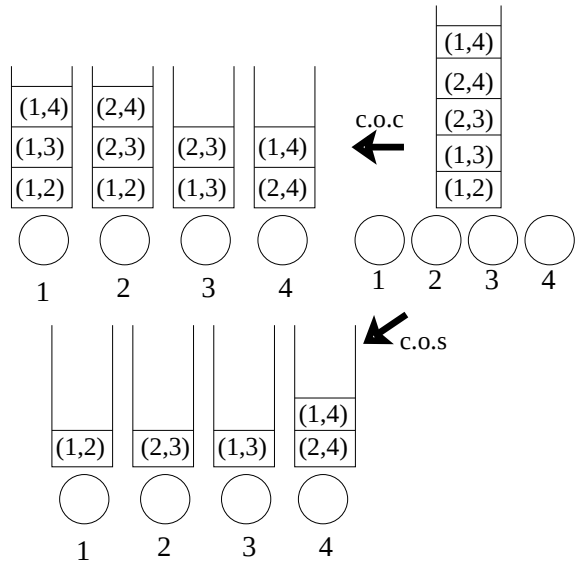


Figure 1: A representation of redundancy-2 *c.o.s.* and *c.o.c.* system and their associated central queue. Jobs are denoted by their set of feasible servers and such jobs are arranged in the central queue in a FIFO manner. The choice of server in *c.o.s.* is by the *uniform assignment rule*.

It is this central queue architecture for the two redundancy models, that enables us to offer a unifying framework for their analysis. This unifying analysis is based on the multi-type job and server model of Visschers et al [1] which also has a *central queue architecture*. We use the model of [1] to analyze redundancy-$d$ with *c.o.s.* in detail in Section 4 and Section 6. In Section 7, we give an alternative analysis for redundancy-$d$ with *c.o.c.* model that is based on an extension, together with an adaptation of the state descriptor, of Visschers et al. [1]. Thus, our extension of [1] provides a comprehensive analysis for redundancy systems with *c.o.s.* and *c.o.c.*.

### 3.2 Multi-type job and server model of [1]

In this section we present the multi-type job and multi-type server model and the results as presented in Visschers et al. [1]. This will form the basis in order to prove the steady-state probabilities for our redundancy models.

Visschers et al. [1] consider a *central queue* system with $K$ possibly heterogeneous servers and multi-type jobs. Let $\mathcal{C}$ denote the set of job types and $\mathcal{M} = \{m_1, \ldots, m_K\}$ denote the set of servers respectively. Let $\mu_i$ denote the service rate of server $m_i$. For $a \in \mathcal{C}$, $S_a$ denotes the set of *feasible* servers that can serve jobs of type $a$. Similarly, associated with a server $M \in \mathcal{M}$ is a set of job types it can serve, denoted henceforth by $C(M)$. The jobs of type $a \in \mathcal{C}$ arrive according to a Poisson process with rate $\lambda_a$ and the total arrival rate $\lambda = \sum_{a \in \mathcal{C}} \lambda_a$. All jobs have an exponential service requirement with unit mean and the service rate of server $m_i \in \mathcal{M}$ is denoted by $\mu_i$ for $1 \leq i \leq K$. Each arriving job waits in a *central queue* and is served in a first-in first-out (FIFO) basis. Each job can only be served by one of its *feasible* server. If an arriving job finds that more than one of its *feasible* servers are idle, then an *assignment rule* decides which of the idle *feasible* server to be assigned.

#### State space representation

An appropriate Markovian descriptor used in [1] to describe the system is of the type $(n_i, M_i, n_{i-1}, M_{i-1}, \ldots, n_1, M_1)$ which denotes states with $i$ busy servers (denoted by $M_1, \ldots, M_i$) and $n_j$ waiting jobs between servers $M_j$ and $M_{j+1}$ for $1 \leq j \leq i-1$. In this state space representation, jobs and active servers are arranged in a FIFO basis from right to left. Therefore all $n_j$ jobs have arrived before $n_k$ jobs where $1 \leq j < k \leq i$. Note that a job is waiting in the central queue only if all of its *feasible* servers are busy (serving jobs that came before it). Therefore $n_1$ denotes the number of those jobs (who have arrived before $n_j$ jobs where $j > 1$) that have to wait since they have server $M_1$ as their only *feasible* server which happens to be busy. Let us denote the set of such job types by $\mathcal{U}(\{M_1\})$ where clearly $\mathcal{U}(\{M_1\}) = \{a \in \mathcal{C} : S_a = M_1\}$. Similarly, $n_j$ represent jobs (that arrived after $n_{j-1}$ but before $n_{j+1}$ jobs) that have to wait because their feasible servers are busy. The feasible servers for these $n_j$ jobs must clearly be a subset of the active servers $\{M_1, M_2, \ldots, M_j\}$ ahead of it (otherwise the job would not have been waiting if any of its feasible server was idle). Therefore for $1 \leq j \leq i$, we have $\mathcal{U}(\{M_1, M_2, \ldots, M_j\}) := \{a \in \mathcal{C} : S_a \subseteq \{M_1, \ldots M_j\}\}$ as the set of possible job types for $n_j$ waiting jobs in the representation. (See [1] for more details.)

Denote the state space of the Markov chain by $\mathcal{S}$ and let any generic state $s \in \mathcal{S}$ be of the type $s = (n_i, M_i, \ldots, n_1, M_1)$. Define $\lambda_M(\{M_1, \ldots, M_i\})$ as the activation rate of server $M$, when the set of active servers is $\{M_1, \ldots, M_i\}$. In state $s = (n_i, M_i, \ldots, n_1, M_1)$, this is the transition rate from state $s$ to state $(0, M, s)$. This activation rate $\lambda_M(\{M_1, \ldots, M_i\})$ depends on the *assignment rule* defined for the model, which determines to which idle feasible machine (if any) a job is routed. By considering *assignment rules* that satisfy the following *assignment condition*, Visschers et al. [1] obtain a product form stationary distribution.

**Assignment Condition 1.** *For $i = 1 \ldots K$, and for every subset $\{M_i, \ldots, M_1\}$ of $\mathcal{M}$ of size $i$, the following holds:*

$$\prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}) = \prod_{j=1}^{i} \lambda_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\}) \quad (2)$$

*for every permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots M_i$.*

We now recall the following theorem from [1] that provides a product form stationary distribution.

**Theorem 1.** *For the multi-type job and server model with an assignment rule satisfying the assignment condition, the steady state probability for all states $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$, is given by*

$$\pi(s) = {\alpha_i}^{n_i} \ldots {\alpha_1}^{n_1} \frac{\Pi_\lambda(\{M_1, \ldots, M_i\})}{\Pi_\mu(M_i, \ldots, M_1)} \pi(0) \quad (3)$$

$$\begin{aligned}
\textit{where} \quad \Pi_\lambda(\{M_1, \ldots, M_i\}) &= \prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}), \\
\Pi_\mu(M_i, \ldots, M_1) &= \prod_{j=1}^{i} \mu_{\{M_1, \ldots M_j\}}, \\
\alpha_j &= \frac{\lambda_{\mathcal{U}(\{M_1, \ldots, M_j\})}}{\mu_{\{M_1, \ldots M_j\}}}, \\
\mu_{\{M_1 \ldots M_i\}} &= \sum_{j=1}^{i} \mu_{M_j} \ \textit{and} \\
\lambda_{\mathcal{U}(\{M_1, M_2, \ldots, M_j\})} &= \sum_{a \in \mathcal{U}(\{M_1, M_2, \ldots, M_j\})} \lambda_a.
\end{aligned}$$

## 4 Redundancy-$d$ with *c.o.s.*: An exact analysis

In this section, we provide the first exact analysis for the redundancy-$d$ model with *c.o.s.*. For this model, we will say that all jobs that choose the same $d$ servers are said to be of the same type. In all, there are $\binom{K}{d}$ job types and the arrival rate of any job type is $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$. In order to analyze the redundancy-$d$ model with *c.o.s.*, we use the central queue architecture (see Section 3.1.2) that allows us to view our model as a multi-type multi-server model [1]. To see how, note that under *c.o.s.*, once one copy starts being served, all other copies are canceled. Hence, each time a server becomes

available, it chooses a *feasible* job from the central queue in a FIFO manner (recall that a job is *feasible* for a server if this server is among the $d$ randomly chosen servers for this job). See also Figure 1. Further, again because of *c.o.s.*, a job will be served by only one server and hence, if an arriving job finds multiple *feasible* servers as idle, then the choice of a server is made based on an *assignment rule*. But this is exactly the model setting for the multi-type and multi-server model defined in Visschers et al. [1] and hence we are in a position to analyze the *c.o.s.* model using Theorem 1.

For the *c.o.s.* model, we use the Markovian descriptor $(n_i, M_i, n_{i-1}, M_{i-1}, \ldots, n_1, M_1)$ to represent any state $s \in \mathcal{S}$ (following [1], see Section 3.2). Recall that active servers and jobs are arranged in a FIFO basis from right to left. Further, $M_j$ in the representation denotes that server $M_j$ is currently serving a job. For a given state $s$, $n_j$ $(j \geq d)$ represents jobs that have their $d$ feasible servers to be a subset of $\{M_j, \ldots, M_1\}$. In fact, from the discussion in Section 3.2, the type of such jobs belongs to the set $\mathcal{U}(\{M_1, M_2, \ldots, M_j\})$. An important point to note is that since each job type has $d$ feasible servers, it can never happen that there are jobs in the central queue waiting to be served and that there are less than $d$ busy servers in the system. Therefore, $n^1 = \ldots = n^{d-1} = 0$.

Recall that to apply Theorem 1, we need to consider an *assignment rule* that satisfies the *assignment condition*. We consider the *uniform assignment rule*: if an arriving job finds $\alpha$ of its $d$ $(\alpha \leq d)$ servers busy, then the probability of choosing each idle server for service is $\frac{1}{d-\alpha}$. For the *uniform assignment rule*, we prove that it satisfies the *assignment condition*. The proof is in the Appendix.

**Lemma 1.** *The uniform assignment rule satisfies the assignment condition given by Eq. (2).*

## 4.1 Product form stationary distribution

In this section we provide the steady state distribution for the Markovian descriptor $s = (n_i, M_i, \ldots, n_1, M_1)$, which turns out to have a product form. We obtain this by applying Theorem 1 to the *c.o.s.* model. (See the Appendix for the proof.)

**Proposition 1.** *The steady state distribution for any state $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$ is given by*

$$\pi(s) = \begin{cases} G^i(K,d)\frac{\pi(0)}{i!\mu^i} & \text{for } i < d \text{ and } n^1 = \ldots = n^{d-1} = 0 \\ {r_i}^{n_i} \ldots {r_d}^{n_d} G^i(K,d)\frac{\pi(0)}{i!\mu^i} & \text{for } i \geq d \text{ and } n^1 = \ldots = n^{d-1} = 0 \\ 0 & \text{elsewhere} \end{cases} \quad (4)$$

*where $\pi(0)$ is the probability of an empty system and where*

$$\begin{aligned} G^i(K,d) &= \prod_{j=1}^{i} G_j(K,d) \\ G_j(K,d) &= \frac{\lambda}{\binom{K}{d}} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha} \text{ and} \\ r_i &= \frac{\lambda\binom{i}{d}}{i\mu\binom{K}{d}}. \end{aligned}$$

□

The stationary distribution for any state $s$ can be written in an alternative form as follows. Define

$$\begin{aligned} \bar{G}_j(K,d) &= \frac{G_j(K,d)\cdot K}{\lambda} \\ &= \frac{d}{\binom{K-1}{d-1}} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha}, \end{aligned}$$

$\bar{G}^i(K,d) = \prod_{j=1}^{i} \bar{G}_j(K,d)$ and $\bar{r}_i = \frac{r_i}{\rho} = \frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}}$. The stationary distribution for any states $s$ with $n^1 = \ldots = n^{d-1} = 0$ can now be expressed as

$$\pi(s) = \begin{cases} \bar{G}^i(K,d)\frac{\rho^i \pi(0)}{i!} & \text{for } i < d \\ \bar{r}_i^{n_i} \ldots \bar{r}_d^{n_d} \bar{G}^i(K,d)\frac{\pi(0)}{i!}\rho^{\left(i+\sum_{j=d}^{i} n_j\right)} & \text{for } i \geq d. \end{cases} \quad (5)$$

Note that $i + \sum_{j=d}^{i} n_j$ represents the total number of jobs in the system. From (5), one directly concludes that the stationary distribution is a function only of $\rho, K$ and $d$.

It is interesting to point out that the stationary distribution does not depend on the identity of the servers that are active since the servers are assumed to be homogeneous. Hence the stationary probabilities for states with the same number of active servers $(i)$ and same number of waiting $(n_i)$ jobs between servers are the same.

## 4.2 Normalization constant and probability of busy servers

We denote by $p(i)$ the stationary probability that the redundancy-$d$ model with *c.o.s.* has $i$ busy servers, for $1 \leq i \leq K$. This metric would be useful in obtaining the normalizing constant $\pi(0)$, which is also the probability that the system is empty, i.e., there are no busy servers $(p(0) = \pi(0))$. Noting that

$$p(i) = \sum_{s \in \mathcal{S}_i} \pi(s) \quad (6)$$

where $\mathcal{S}_i = \{s \in \mathcal{S} : \text{exactly } i \text{ servers are busy}\}$, we have the following lemma. The proof can be found in the Appendix.

**Lemma 2.** *The probability that $i$ servers are busy is given by*

$$p(i) = \bar{p}(i)\pi(0) \tag{7}$$

*where*

$$\bar{p}(i) = \begin{cases} \binom{K}{i}\bar{G}^i(K,d)\rho^i, & \text{for } i < d \\ \left(\frac{1}{1-r_i}\right)\cdots\left(\frac{1}{1-r_d}\right)\binom{K}{i}\bar{G}^i(K,d)\rho^i, & \text{for } i \geq d. \end{cases} \tag{8}$$

*Further, $\pi(0)$ is given by*

$$\pi(0) = \left(1 + \sum_{i=1}^{K} \bar{p}(i)\right)^{-1}. \tag{9}$$

A related performance metric of interest is $P_w(j)$, the probability that an arriving job sees $j$ of its $d$ feasible servers as busy. When a job arrives, conditioned on $i$ busy servers, the probability that $j$ of its servers are already busy is $\frac{\binom{K-j}{i-j}}{\binom{K}{i}}$, $i \geq j$. Therefore we have

$$P_w(j) = \sum_{i=j}^{K} p(i)\frac{\binom{K-j}{i-j}}{\binom{K}{i}}. \tag{10}$$

In particular, the probability a job has to wait, denoted by $P_w$, is given by $P_w(d)$.

### 4.3 Distribution of number of jobs

An important performance metric for the *c.o.s.* system is the number of waiting jobs. In this section, we obtain the expression for the probability-generating function for the number of waiting jobs in the system. An inversion of this transform gives us the distribution of the number of waiting jobs.

Define $p(i,m)$ as the probability that the system has $i$ busy servers and $m$ waiting jobs in the system. When $i < d$, there are no jobs waiting in the central queue, hence, we have

$$p(i,m) = \begin{cases} p(i) & \text{for } m = 0 \\ 0 & \text{elsewhere.} \end{cases}$$

For $i \geq d$, it follows from Proposition 1 that

$$\begin{aligned} p(i,m) &= \pi(0)\binom{K}{i}\bar{G}^i(K,d)\rho^i l_i(m) \\ \text{where } l_i(m) &= \sum_{\substack{\{n_i \ldots n_d: \\ \sum_{l=d}^{i} n_l = m\}}} r_i^{n_i}\ldots r_d^{n_d}. \end{aligned}$$

Let $Q$ denote the random variable corresponding to the number of waiting jobs in the *c.o.s.* system. The probability that there are $m$ waiting jobs ($Q = m$) is given by $\hat{p}(m) = \sum_{i=1}^{K} p(i,m)$. Using the above expressions for $p(i,m)$, we can derive the distribution for the number of waiting jobs. Its proof can be found in the Appendix.

**Proposition 2.** *The P.G.F. for the number of waiting jobs is given by*

$$E(z^Q) = \sum_{i=0}^{d-1} p(i) + \sum_{i=d}^{K} p(i)\left(\prod_{j=d}^{i} Geom_{r_j}(z)\right) \tag{11}$$

*where*

$$Geom_{r_j}(z) = \frac{1-r_j}{1-r_j z}$$

*and $p(i)$ is given by Eq. (7) and (9). The expected number of waiting jobs in the system is given by*

$$E(Q) = \sum_{i=d}^{K} p(i)\left(\sum_{j=d}^{i}\frac{r_j}{1-r_j}\right) \tag{12}$$

*and the expected number of jobs in the system is given by $E(N) = E(Q) + \rho K$.*

**Remark 1.** *From the P.G.F above, one can conclude that the random variable $Q$ is a mixture of sum of geometric random variables.*

**Remark 2.** ***Special cases:*** *We now discuss two special cases for the redundancy-d model with c.o.s. namely $d = 1$ and $d = K$. The case $d = 1$ or redundancy-1 c.o.s. corresponds to the system with Bernoulli routing of jobs to the $K$ servers. This is also the case for redundancy-1 with c.o.c.. The redundancy-$K$ with c.o.s. is equivalent to an $M/M/K$ system. This is easy to see from the central queue architecture; since each job can be served at any of the $K$ servers, a server that becomes free ends up choosing the head of the queue customer for service. We verify in Appendix E that our expressions for c.o.s. for these special cases indeed coincide with the known results for these cases.*

## 5 Equivalence of JSW(d) and redundancy-d with *c.o.s.*

In this section, we will show the equivalence of the redundancy-$d$ *c.o.s.* model with that of the join the shortest work among $d$ servers (JSW($d$)) policy. This is trivial to note for the case when $d = 1$ and $d = K$. When $d = 1$, the redundancy-1 *c.o.s.* model as well as the JSW(1) model are equivalent to probabilistic routing to $K$ queues when the routing probabilities are Bernoulli ($\frac{1}{K}$). When $d = K$, redundancy-$K$ *c.o.s.* model is equivalent to an $M/M/K$ multi-server queue (see Section 9) whereas the $JSW(K)$ policy is simply the traditional JSW (join shortest work) policy. But the JSW policy is known to be equivalent to the $M/M/K$ queue [17]. While the equivalence is now apparent for $d = 1$ and $d = K$, we give a formal proof of this fact for any $d$ in this section. The reasoning is based on a sample-path argument, hence, the equivalence holds for generally distributed service requirements.

**Proposition 1.** *Assume generally distributed service requirements and heterogeneous servers. For any given sample-path realization, a given job will be served under both JSW(d) and redundancy-d with c.o.s. in the same server. As a result, the following performance measures are the same under both models:*

- *Joint probability of servers being busy or idle.*
- *Delay distribution of a job.*
- *Total number of jobs in the system.*

*Proof.* For a given realization, we couple the arrivals of jobs, the $d$ servers sampled upon arrival, and their service requirements. For redundancy-$d$ with c.o.s., a job cannot be overtaken by copies that arrive after him in any of its $d$ sampled servers (since it has copies in all $d$ FIFO-queues). It therefore follows directly that upon arrival of the job, one can determine the server in which the job will be served. In order to describe this server, we define the effective workload, $V_i^{red}(t)$ to denote the work present in server $i$ that will be served in this server. This ignores the workload due to those copies in server $i$ that will not be served in the server. As such, for redundancy-$d$ with *c.o.s.* an arriving job will be served at the server with smallest workload $V_i^{red}(t)$, among the $d$ servers.

Under JSW($d$), an arriving job is send to the server with smallest workload $W_i^{JSQ}(t)$, among the $d$ servers. Hence, in order to prove that a given job will be served in the same server under both models we are left to show that $W_i^{JSQ}(t) = V_i^{red}(t)$. This can be seen as follows. First note that the (effective) workload decreases at rate 1 whenever the process is positive. Now assume $W_i^{JSQ}(t) = V_i^{red}(t)$ is true at time $t$, and a new job arrives. (This is trivial to note for $t = 0$.) It is served in the server (among the $d$ servers) with smallest value for $W_i^{JSQ}(t)$ ($V_i^{red}(t)$), under JSW($d$) (redundancy-$d$, respectively). Since $W_i^{JSQ}(t) = V_i^{red}(t)$, it is the same server say server $m$, that has the smallest value in the two system and hence, the service requirement of this job is added both to $W_m^{JSQ}(t^+)$ and to $V_m^{red}(t^+)$. □

**Remark 3.** *With the above proposition, redundancy-d with c.o.s. can be perceived as a method of implementing JSW(d) without requiring the knowledge of the residual work in each of the queue. But such an implementation of JSW(d) would only be possible if the underlying system allows for the possibility to use extra copies of the same job. Indeed, the power of allowing even a single extra copy per job may prove to be very beneficial in the absence of workload information at the queues.*

Now from the equivalence between JSW($d$) and redundancy-$d$ with *c.o.s.*, we have the following proposition that to the best of our knowledge provides performance metrics for JSW($d$) that have not been obtained before.

**Proposition 3.** *For a JSW(d) system, the P.G.F. for the number of waiting jobs is given by*

$$E(z^Q) = \sum_{i=0}^{d-1} p(i) + \sum_{i=d}^{K} p(i) \left( \prod_{j=d}^{i} Geom_{r_j}(z) \right)$$

*where*

$$Geom_{r_j}(z) = \frac{1 - r_j}{1 - r_j z}$$

*and $p(i)$ is given by Eq. (7) and (9). The expected number of waiting jobs in JSW(d) is given by*

$$E(Q) = \sum_{i=d}^{K} p(i) \left( \sum_{j=d}^{i} \frac{r_j}{1 - r_j} \right),$$

*and the expected number of jobs in JSW(d) is given by $E(N) = E(Q) + \rho K$.* □

# 6 Asymptotic regimes for *c.o.s*

In this section, we consider the redundancy-$d$ system with cancel-on-start under limiting regimes. We first consider the heavy-traffic regime where the number of servers is fixed and the traffic intensity approaches 1. In the second regime, we scale the number of servers and the arrival rate while keeping the traffic intensity unchanged, that is, the mean-field regime.

## 6.1 Heavy-traffic regime

For the heavy-traffic analysis, we keep the number of servers, $K$, fixed and let $\frac{\lambda}{\mu} \uparrow K$, so that $\rho \uparrow 1$. When $d = 1$, we have a system of $K$ independent $M/M/1$ queues with parameters $\lambda/K$ and $\mu$. Hence, after scaling by $1 - \rho$, the total number of jobs in the system converges, as $\rho \uparrow 1$, to the sum of $K$ exponentially distributed random variables with mean 1. In the result below, we derive the distribution of the total scaled number of waiting jobs for $d \geq 2$.

**Proposition 4.** *Assume $d \geq 2$. Then*

$$\lim_{\rho \uparrow 1} p(K) = 1 \qquad \textit{and} \qquad lim_{\rho \uparrow 1} E[e^{-s(1-\rho)Q}] = \frac{1}{1+s},$$

*for any $s > 0$. In words, this implies that $(1 - \rho)Q$ converges to an exponential random variable with unity parameter, as $\rho \uparrow 1$.*

*Proof.* From Eq. (8) we have $\bar{p}(i) = O(\rho^i)$ for $i < d$, while $\bar{p}(K) = O(\rho^K/(1 - \rho))$. Since $\rho \uparrow 1$, this implies that $\lim_{\rho \uparrow 1} p(K) = 1$. Now, substituting $z = e^{-s(1-\rho)}$ in Eq. (11), together with
$\lim_{\rho \uparrow 1} p(K) = 1$ and taking the limit $\rho \to 1$ gives

$$lim_{\rho \to 1} E[e^{-s(1-\rho)Q}] = \lim_{\rho \to 1} \left( \prod_{j=d}^{K} Geom_{r_j}(z) \right).$$

For $i < K$, we have $lim_{\rho\to 1} r_i = \bar{r}_i < 1$, since $d \geq 2$. Further, $lim_{\rho\to 1} r_K = \bar{r}_K = 1$. This implies that $\lim_{\rho\to 1} Geom_{r_j}(e^{-(1-\rho)s}) = 1$ for $j \neq K$. For the case $j = K$, using L'Hopital's rule it can be seen that $\lim_{\rho\to 1} Geom_{r_K}(e^{-(1-\rho)s}) = \frac{1}{1+s}$. This completes the proof. □

The above result shows that there is a large drop in the performance when passing from $d = 1$ to $d = 2$. Further, for $d \geq 2$, the performance is independent on the parameter $d$.

### 6.2 Mean-field regime

In this subsection we consider the mean-field regime and assume that the arrival rate into the system is $\lambda = \hat{\lambda}K$ where $\hat{\lambda} < \mu$, and let $K \to \infty$. Define $\hat{\rho} = \frac{\hat{\lambda}}{\mu}$.

For JSW($d$), asymptotic independence in the mean-field limit has been proved in [18]. This, together with the equivalence between JSW($d$) and red-$d$ *c.o.s.* and the fact that a given server under red-$d$ *c.o.s.* is busy with probability $\hat{\rho}$ (see preliminaries), gives the following result for the probability $P_w(j)$ that an arriving job sees $j$ busy feasible servers.

**Proposition 2.** *Let $\lambda = K\hat{\lambda}$. We have*

$$\lim_{K\to\infty} P_w(j) = \hat{\rho}^j (1-\hat{\rho})^{d-j} \binom{d}{j}.$$

*Proof.* In [18, Theorem 2.1], JSW($d$) is considered in the mean-field regime. It is proved there that given a finite set of servers, these servers behave independent in the mean-field limit. Note that the joint distribution of servers being idle is the same for both JSW($d$) and red-$d$ c.o.s. (Proposition 1). In addition, we showed that the probability a server is busy equals $\rho$. Hence, the probability that $j$ servers are busy out of $d$, follows a binomial distribution with parameters $d$ and $\hat{\rho}$. □

## 7 Redundancy with *c.o.c.*: an alternate product from

In this section, we give an alternate analysis for the *c.o.c.* redundancy models of [6,7] by generalizing the multi-type job and multi-server model of Visschers et. al. [1].

In the *c.o.c.* model, multiple *feasible* servers can serve a job simultaneously and hence a direct application of Theorem 1 is not possible. Additionally, the service rate of a job in the central queue of the *c.o.c.* model depends on the service rate received by jobs ahead of it (in the central queue), since the job will be served simultaneously in all those servers that cannot serve jobs ahead of him. Therefore, the departure rate of a job is in fact state dependent.

In Section 7.1 we first present our generalization of [1] that allows for state dependent departure rates. This generalization shall allow us to provide a Markovian state space descriptor for the *c.o.c.* redundancy systems, whose stationary distribution is of product form, see Subsection 7.2.

### 7.1 A generalized multi-type job and server model

In the existing model of [1], in any state $s \in \mathcal{S}$, an active server $M_j \in \mathcal{M}$ has a fixed service rate $\mu_{M_j}$. Associated with the model is what we call a *standard departure set function* $\mu_{\{\cdot\}}$ ($\{\cdot\}$ is used to denote a set) which is defined as

$$\mu_{\mathcal{Y}} = \sum_{t\in\mathcal{Y}} \mu_t \text{ for } \mathcal{Y} \subseteq \mathcal{M}. \tag{13}$$

Therefore the total departure rate in state $s = (n_i, M_i, \ldots, n_1, M_1)$, is given by $\mu_{\{M_1\ldots M_i\}} = \sum_{j=1}^{i} \mu_{M_j}$. In this case, one can see that the departure rate from a server is independent of the machines that are active. We will show that it is in fact possible to generalize this and allow state dependent departure rates without affecting the nature of the results in [1]. We say that a departure set function $\mu_{\{\cdot\}}$ is a *generalized departure set function* if it satisfies the following condition.

**Condition 1.** *For $1 \leq j \leq K$ and a subset $\{M_1 \ldots M_j\}$ of $\mathcal{M}$,*

$$\mu_{\{M_1\ldots M_j\}} - \mu_{\{M_1\ldots M_{j-1}\}} \geq 0. \tag{14}$$

□

For any state $s = (n_i, M_i, \ldots, n_1, M_1)$ and a *generalized departure set function* $\mu_{\{\cdot\}}$ define

$$\hat{\mu}_{M_j}(s) := \mu_{\{M_1\ldots M_j\}} - \mu_{\{M_1\ldots M_{j-1}\}}.$$

$\hat{\mu}_{M_j}(s)$ now denotes the state dependent departure rate from server $M_j$ where $1 \leq j \leq i$. From the definition of a *generalized departure set function*, it should be clear that we are restricting to state dependent departure rates of a specific type where the departure rate from an active server depends only on the active servers ahead of it. Of course, the total departure rate in state $s = (n_i, M_i, \ldots, n_1, M_1)$ is $\mu_{\{M_1\ldots M_i\}}$ where

$$\mu_{\{M_1\ldots M_i\}} = \sum_{j=0}^{i} \hat{\mu}_{M_j}(s)$$

(noting that for $j = 1$, we have $\mu_{\{M_1\ldots M_{j-1}\}} = \mu_{\{\phi\}} = 0$.) It should be noted that the set function defined by Eq. (13) is just a special case as it satisfies the Condition 1. Also note that the *generalized departure set functions* could be such that in state $s$, $\hat{\mu}_{M_j}(s) = 0$ for some $1 \leq j \leq i$. This implies that although server $M_j$ has picked a job it can serve, the configuration of servers in front of it is such that its current service rate is zero.

We shall henceforth refer to the multi-type job and server model with a *generalized departure set function* satisfying Condition 1 by the name *generalized multi-type job and*

*server model*. This model is thus a generalization of [1] that allows a more general departure rate functions. We now have the following proposition that upholds the validity of Theorem 1 for the *generalized multi-type job and server model*.

**Proposition 5.** *Consider a generalized multi-type job and server model with an assignment rule satisfying the assignment condition (Eq. (2)). Assuming the steady state exists, the steady state probability for any state* $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$ *is given by*

$$\pi(s) = {\alpha_i}^{n_i} \ldots {\alpha_1}^{n_1} \frac{\Pi_\lambda(\{M_1, \ldots, M_i\})}{\Pi_\mu(M_i, \ldots, M_1)} \pi(0) \tag{15}$$

*where*

$$\begin{aligned} \Pi_\lambda(\{M_1, \ldots, M_i\}) &= \prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}), \\ \Pi_\mu(M_i, \ldots, M_1) &= \prod_{j=1}^{i} \mu_{\{M_1, \ldots M_j\}}, \\ \alpha_j &= \frac{\lambda_{\mathcal{U}(\{M_1, \ldots, M_j\})}}{\mu_{\{M_1, \ldots M_j\}}}. \end{aligned}$$

*Proof.* The proof of this proposition is similar to that of Theorem 2 of [1] (Theorem 1 in this paper). An outline of the proof is given in the appendix to point out to the changes due to the *generalized departure rate functions*. □

## 7.2 An alternate product form for *c.o.c.*

In Proposition 5, we have generalized the results of [1] by considering *generalized departure set functions*. This generalization can now be exploited to apply Proposition 5 to redundancy-$d$ with *c.o.c.*.

For the redundancy-$d$ *c.o.c.* model, one can view a type for a job as the choice of $d$ servers that are chosen (see preliminaries). Since this choice is uniform over the set of all servers, each type is equally likely. Recall that $\lambda_{type}$ denotes the arrival rate of each type where $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$. Also recall some earlier notation where $\mathcal{C}$ denotes the set of all job types. For any type $c \in \mathcal{C}$, let $S_c$ denote the set of $d$ *feasible* servers for that type. For each type $c \in \mathcal{C}$, we associate a label $O_c$. This label will be used to identify the least recent (oldest) type $c$ job present in the central queue.

### A new state space representation for *c.o.c.*

We propose a new state space representation $(n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ to analyze the redundancy-$d$ model of [7]. In such a state, the job at the head of line (of the central queue) is of type $c_1$. Since the central queue has FIFO ordering from right to left, this is also the oldest type $c_1$ job in state $s$ and is therefore indicated by the label $O_{c_1}$. $n_1$ denotes the number of type-$c_1$ jobs that arrived after $O_{c_1}$. These jobs were followed by a type-$c_2$ job, represented by $O_{c_2}$. Now $n_2$ denotes the subsequent arrivals that are of either type $c_1$ or $c_2$. In general, in state $s$ and for $0 \leq j \leq i$, $O_{c_j}$ indicates the position in the central queue of the oldest type-$c_j$ job and $n_j$ denotes waiting jobs of type $c \in \bigcup_{i=1}^{j} c_i$. Now in state $s$, the jobs that can receive service are the ones that are represented by the labels $(O_c)$. This is because of the FIFO scheduling in the central queue and the fact that the waiting jobs $(n_j)$ are not the oldest jobs in their type. In state $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$, the departure rate of job $O_{c_j}$ equals $\mu$ times the number of its feasible machines that are not used by the jobs $O_{c_{j-1}}, \ldots, O_{c_1}$. Hence, it is given by

$$\hat{\mu}_{c_j}(s) := \mu\left(|F_j(s)| - |F_{j-1}(s)|\right),$$

where $F_j(s) := \bigcup_{l=1}^{j} S_{c_l}$. The total departure rate in state $s$ is the sum of the departure rates at each $O_{c_j}$ for $0 \leq j \leq i$. This corresponds to the *departure rate set function* for the model and is given by

$$\mu_{\{O_{c_1}, \ldots O_{c_j}\}} = \sum_{t \in \bigcup_{l=1}^{j} S_{c_l}} \mu = \mu |F_j(s)|, \tag{16}$$

which satisfies Condition 1 (Eq. (14)).

### Redundancy-$d$ *c.o.c.* as *generalized multi-type job and server model*

With a state space representation of $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$, the *c.o.c.* model can now be viewed as a *generalized multi-type job and server model* where $O_{c_j}$ resembles a super-server of state dependent service rate $\mu\left(|F_j(s)| - |F_{j-1}(s)|\right)$. Another artifact of this representation is that each super-server $O_{c_j}$ can serve jobs only of type $c_j$ and therefore each type has a dedicated super-server that will only serve jobs of its type. From this interpretation, it should be clear that there is no need to specify any *assignment rule* because an arriving job is never faced with the prospect of being served by multiple super-servers (although he will be served by multiple servers that define the super-server). Due to this, we have $\lambda_{O_c}(\{O_{c_1}, \ldots, O_{c_j}\}) = \lambda_c$ and hence the *assignment condition* is trivially satisfied. Proposition 5 can now be used to provide a new product form distribution for this redundancy model. This distribution is different from that of [7] due to a different state space representation. At the end of this section we comment on the latter. We have the following proposition (proof in appendix).

**Proposition 6.** *For the redundancy-d c.o.c. model, the steady state distribution for state* $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ *is given by*

$$\pi(s) = \frac{\pi(0)}{i!} \prod_{j=1}^{i} \left( \frac{j \lambda_{type}}{\mu |F_j(s)|} \right)^{n_j + 1}. \tag{17}$$

The redundancy-$d$ *c.o.c.* system has already been extensively studied by Gardner et al [6, 7] and hence we do not analyze this any further. The main purpose of Proposition 6 is to illustrate the unifying framework for redundancy models via Visschers et al [1] and an adaptation of its generalization. To recall the main results from [7], the normalizing constant $\pi(0)$ in Proposition 6 is given by

$$\pi(0) = \prod_{i=d}^{K} \left(1 - \frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}}\rho\right),$$

and the mean number of customers in the system $E(N)$ is

$$E[N] = \sum_{i=d}^{K} \frac{\rho}{\frac{\binom{K-1}{d-1}}{\binom{i-1}{d-1}} - \rho}. \tag{18}$$

Since our state space representation for analyzing *c.o.c.* is different from that used in [6, 7], a direct comparison between Proposition 6 and [7, Theorem 2] is cumbersome. However one can compare certain states of the system to show that the stationary probabilities obtained from Proposition 6 and [7, Theorem 2] are indeed the same. For example, consider a state with $m$ customers, all belonging to the same type. Such a state in our representation is denoted by $(m-1, O_c)$, hence $i = 1$ and $F_1(s) = d$ in Proposition 6, so

$$\pi((m-1, O_c)) = \pi(0)\left(\frac{\lambda_{type}}{d\mu}\right)^m.$$

This is the same as the steady state probability given in [7, Theorem 2]. Similarly, consider a state with $m$ customers all from distinct types. In our representation, such a state is represented by $(O_{c_m}, \ldots, O_{c_1})$ and from Proposition 6,

$$\pi((O_{c_m}, \ldots, O_{c_1})) = \pi(0)\left(\frac{\lambda_{type}}{d\mu}\right)^m.$$

Again, this coincides with [7, Theorem 2].

# 8 Numerical Results

In this section we present numerical results to further analyze the redundancy-$d$ model considered in this paper. In Section 8.1 we numerically compare the performance between redundancy-$d$ systems with *c.o.s.* and *c.o.c.*. In Section 8.2, we provide some more numerical results for redundancy-$d$ *c.o.s.* This is followed by the numerical results for the mean-field regime in Section 8.3.

## 8.1 Comparing *c.o.c.* and *c.o.s.*, impact of independence assumption

In this section we compare the performance of both systems and in particular we assess the impact of the independence assumption (as made in *c.o.c.*) on the performance.

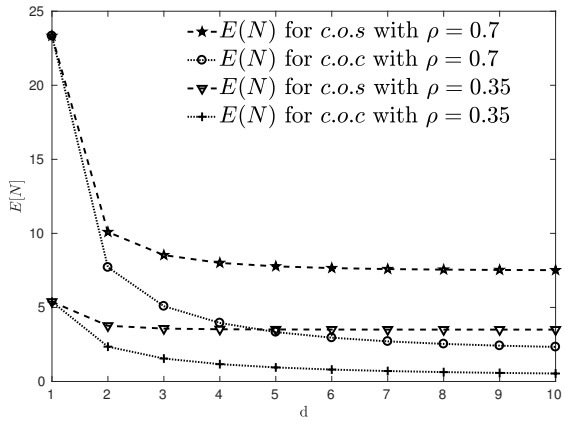


Figure 2: $E(N)$ for *c.o.c.* and *c.o.s.* for $\rho = 0.35, 0.7$ and $K = 10$.

We first compare the mean number of customers ($E(N)$) in the system under *c.o.s.* and *c.o.c.*. The mean number of customers $E(N)$ in the *c.o.c.* system is given by Eq. (18), as derived in [7]. Consider *c.o.c.* and *c.o.s.* systems with parameters $K = 10, \rho = 0.35$ and 0.7. For varying values of $d$, Fig. 2 compares $E(N)$ for the two systems. The case $d = 1$ for both models is equivalent to Bernoulli routing to $K$ servers in which case we have $E(N) = \frac{K\rho}{1-\rho}$. The case, $d = K$ for the *c.o.c.* model coincides with an $M/M/1$ server with arrival rate of $\lambda$ and service rate of $K\mu$ and hence $E(N) = \frac{\rho}{1-\rho}$. On the other hand, $d = K$ for *c.o.s.* corresponds to an $M/M/K$ system with arrival rate $\lambda$ and $K$ servers each with a service rate 1. Naturally as expected, $E(N)$ for the $M/M/K$ system is larger than that of the $M/M/1$ system associated with $d = K$ for *c.o.c.* For $1 < d < K$, we see that $E(N)$ for the *c.o.c.* model is lower than that of the *c.o.s.* model and this is true for any value of $\rho$ (illustrated in the figure for $\rho = 0.35$ and 0.7.) What is also noticeable from the figure is that the proportion of reduction in $E(N)$ from having an extra copy ($d = 2$) in both the redundancy models depends on the load $\rho$ in the system.

To elaborate on this, in Table 1, we compare $E(N)$ for *c.o.s.* and *c.o.c.* when increasing the redundancy parameter from $d = 1$ to $d = 2$. We fix $K = 10$ and $\mu = 1$ and take varying values of $\rho$. Note from the discussion after Proposition 1 that the stationary distribution for the *c.o.s.* system is merely a function of $\rho, K$ and $d$. This is also the case for the *c.o.c.* system (see Proposition 6) and this justifies the choice of $\rho$ as a parameter for this table. Table 1 indicates that $E(N)$ for *c.o.c.* system is lower than that of the *c.o.s.* system for various values of $\rho$. We also quantify the percentage reduction in $E(N)$ when increasing the redundancy parameter from $d = 1$ to $d = 2$. A key observation from the table is the fact that for low values of $\rho$, the reduction with *c.o.s.* is smaller than that

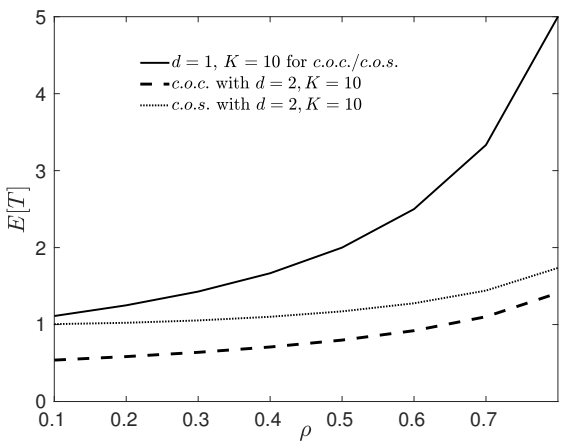


Figure 3: $E(T)$ for *c.o.c.* and *c.o.s.* with $K = 10$.

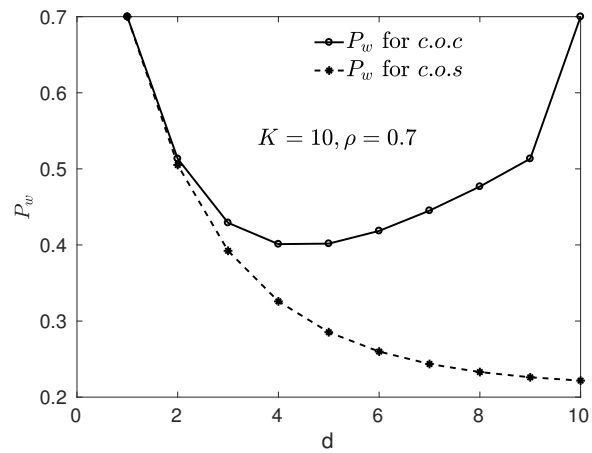


Figure 4: $P_w$ for *c.o.c.* and *c.o.s.* with $K = 10, \rho = 0.7$.

obtained with *c.o.c.*. This is due to the independence assumption: at low loads, an arriving job will likely find the system empty. Under *c.o.c.*, the job will then get an instantaneous service rate of $2\mu$, which explains the reduction of the order of 50% in the table. However, with *c.o.s.*, even if the system is empty, an arriving job will only get served in one server. However, for higher values of $\rho$, the gain obtained with *c.o.s.* becomes comparable to that obtained with *c.o.c.*. In this case, the common feature that helps improve the performance with both systems is that by sending redundant copies, they can more efficiently use the capacity of the system.

| $\rho$ | $d = 1$ $E(N)$ | % decrease in $E(N)$ for *c.o.s.* with $d = 2$ | % decrease in $E(N)$ for *c.o.c.* with $d = 2$ |
|---|---|---|---|
| 0.1 | 1.11 | 9.5 | 51.56 |
| 0.2 | 2.5 | 18.17 | 53.3 |
| 0.3 | 4.28 | 26.25 | 55.26 |
| 0.4 | 6.66 | 33.94 | 57.5 |
| 0.5 | 10 | 41.44 | 60.09 |
| 0.6 | 15 | 48.95 | 63.17 |
| 0.7 | 23.33 | 56.75 | 66.95 |
| 0.8 | 40 | 65.28 | 71.79 |
| 0.9 | 90 | 75.44 | 78.56 |

Table 1: Reduction in the mean number of jobs with *c.o.c.* and *c.o.s.* for $K = 10$, when increasing the redundancy parameter from $d = 1$ to $d = 2$.

In Figure 3 we compare the mean sojourn time. We observe that almost up to $\rho = 0.7$, the mean sojourn time with *c.o.c.* is below 1. Since the mean service time is 1 ($\mu = 1$) for this example, this is a result that will not happen in most situations in practice. On the other hand, with *c.o.s.*, the mean sojourn time is always larger than 1.

From the examples above, one is tempted to conclude the superiority of *c.o.c.* over *c.o.s.* This might seem counter-intuitive at first, since in the *c.o.c.* model redundant copies can be served simultaneously, which implies extra work for the servers. The superiority of *c.o.c.* system over *c.o.s.* is primarily because of the independence assumption as made in *c.o.c.* (see also the example in Section 3.)

In Figure 4 we plot the metric $P_w$, the probability that an arriving job has to wait with $K = 10, \rho = 0.7$ and for various values of $d$. We use Equation (5) from Gardner et. al. [7] to obtain the numerical values for $p(i)$ in *c.o.c.* The probability an arriving job has to wait under *c.o.c.* can then be calculated using Equation (10) with $j = d$. With *c.o.s.* the probability that an arriving customer has to wait is lower than that for the *c.o.c.* system. The probability to wait under *c.o.s.* decreases with $d$, which is expected since as $d$ increases, resources are more efficiently pooled together. With *c.o.c.*, for large values of $d$, the probability of waiting increases because multiple servers serve a single job and hence an arriving job is more likely to have its *feasible* servers to be busy. In spite of this, the performance of *c.o.c.* is superior in terms of sojourn time (see Figure 3), because as $d$ increases, the system gets closer to being a single super-server (see also the discussion on $d = K$ in Section 3.)

## 8.2 Performance of redundancy-$d$ with *c.o.s.*

In Figure 5 we plot the metric $p(i)$, that is, the probability that $i$ servers are busy, for parameters $K = 10$, $\rho = 0.7$ and for different values of $d$. We observe from Figure 5 that $p(K)$ increases in $d$. This is because a larger $d$ implies a better utilization of all the servers and increases the probability of all servers being busy. This is however not the case in general

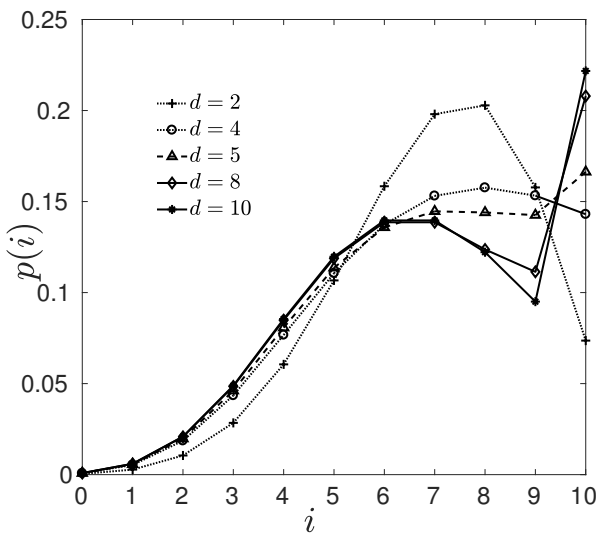


Figure 5: $p(i)$ for different $d$ when $K = 10, \rho = 0.7$.

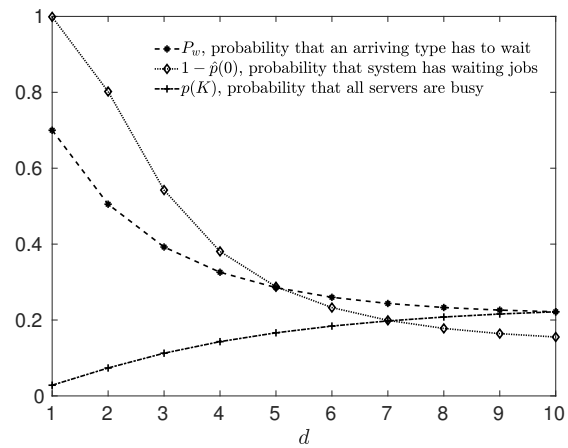


Figure 6: $P_w, 1 - \hat{p}(0)$ and $p(K)$ for $K = 10$ and $\rho = 0.7$.

for $p(i)$, for $2 \leq i < K$. In fact, $p(i)$ is increasing with $d$ for lower values of $i$, (typically $i \leq 5$) and $p(i)$ is decreasing with $d$ for higher values of $i$ ( $5 < i < 10$.) The effect reverses again for $i = K$ and we have $p(K)$ increasing in $d$. To explain this, let us consider the case $d = 10$. When $i < K$, $p(i)$ corresponds to having exactly $i$ jobs present in the system. However, $p(10)$ is the probability that the number of jobs is equal or larger than 10, that is, it corresponds to the tail of the distribution. A similar reason holds for other values of $d$.

In Figure 6, we examine the metrics $P_w, 1 - \hat{p}(0)$ and $p(K)$ for the *c.o.s.* system with parameters $K = 10$ and $\rho = 0.7$ and for various values of $d$. While $P_w$ denotes the probability that an arriving customer has to wait, $1-\hat{p}(0)$ denotes the probability that the system has waiting customers. Note that $\hat{p}(0) = \sum_{i=0}^{d-1} p(i) + \sum_{l=d}^{K} p(l, 0) = \pi(0)(1 + \sum_{i=1}^{K} f^i(K, d))$. Fig. 6 illustrates the difference in the two quantities as a function of $d$. In fact, $P_w < 1 - \hat{p}(0)$ for $d < 5$, $P_w > 1 - \hat{p}(0)$ for $d > 5$ and $P_w = 1 - \hat{p}(0)$ for $d = 5$. From the figure, we also observe that for all $d$, $p(K) \leq P_w$ (with an equality when $d = K$). A justification for this is the fact that all servers being busy implies that an arriving customer has to wait. When $d = K$, an arriving customer having to wait implies that all servers are busy ($M/M/K$ system) and hence the equality.

## 8.3 Asymptotic regimes

In this section we focus on the mean-field regime. That is, we set $\lambda = \hat{\lambda}K$, for a fixed $\hat{\lambda}$, and let $K \to \infty$. In other words, the load per server is kept constant, while taking the number of servers to infinity. In Fig. 7 we plot $P_w$ as a function of $K$ and for different values of $d$. As expected, due to the asymptotic independence of the servers, $P_w$ converges to $\hat{\rho}^d$

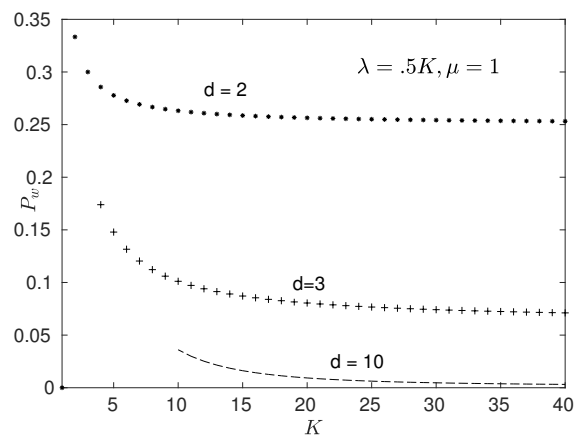


Figure 7: For $\hat{\lambda} = 0.5$ and $K \to \infty$, for a fixed $d$, we have $\lim_{K\to\infty} P_w = \hat{\rho}^d$.

where $\hat{\rho} = \frac{\hat{\lambda}}{\mu}$ (see Proposition 2).

Recall that for $d = K$, the models $JSW$, redundancy-$d$ *c.o.s.*, and $M/M/K$ are all the same. It is known for the $M/M/K$ (and hence for $JSW$), see for example [19], that in this regime the probability of waiting and the mean waiting time vanish in the limit. The same holds for the well-known $JSQ$ dispatching policy. In recent work, van der Boor et al. [20] have shown that $JSQ(d(K))$ with $d = o(K)$ can be asymptotically optimal in the sense of giving probability of waiting and mean waiting time equal to 0 in the limit. The latter implies that $JSQ(d(K))$ yields the same asymptotic performance as $JSQ$, but with a considerably smaller amount of overhead in terms of signaling. Getting back to our model, an interesting question is then how big $d$ needs to be, in order for *c.o.s.* and $JSW(d)$, to be asymptotically optimal with respect to the waiting probability. In Figure 8 we consider the

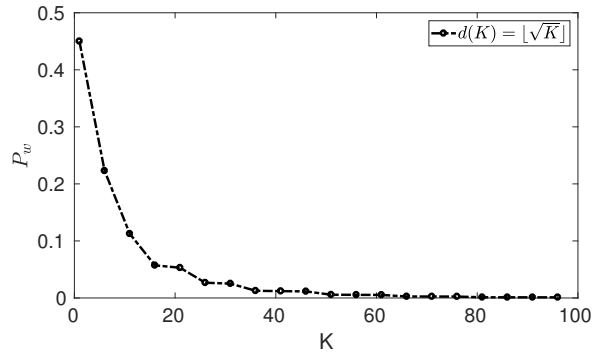


Figure 8: Asymptotic optimality of $d(K) = \lfloor\sqrt{K}\rfloor$ for $\hat{\rho} = 0.45$.

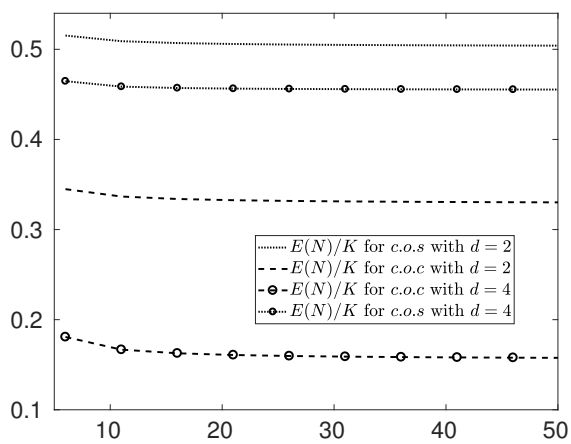


Figure 9: Asymptotics for $E(N)/K$ for *c.o.s* and *c.o.c.* system with $\hat{\lambda} = 0.45$ and various values of $d$.

case $d(K) = \lfloor\sqrt{K}\rfloor$, for $\hat{\rho} = 0.45$. This figure indicates that *c.o.s.* might also be asymptotically optimal for an appropriate scaling of the redundancy parameter. A formal proof of the latter is left for future work.

Finally, we look at the asymptotic value of $E(N)/K$ for redundancy-$d$ system with *c.o.s* and *c.o.c.* This is illustrated in Figure 9. We can derive a simple approximation for the mean sojourn time under *c.o.s.* and hence for the mean number of jobs in the system. If an incoming job does not need to wait, its sojourn time will be $1/\mu$. If it has to wait, then its sojourn time will be its waiting time plus its service time $1/\mu$. To calculate the waiting time in the latter case, we assume that there are no other jobs (with feasible servers in common) waiting in front of the job. Hence, the waiting time will be equal to the minimum of $d$ exponential service times, i.e. $1/(\mu d)$. Now, recalling that the probability of waiting in the limit is $\hat{\rho}^d$ (Proposition 2), and using Little's law, the mean number of jobs in the system, when $K$ is sufficiently large, can be approximated by:

$$E(N) \approx \hat{\lambda}K\left((1-P_w)\frac{1}{\mu} + P_w(\frac{1}{\mu d} + \frac{1}{\mu})\right) = \rho\hat{K}\left(1 + \frac{\hat{\rho}^d}{d}\right).$$

For the values of Figure 9, for $d = 2$ the approximation yields 0.496 whereas the value from the plots are 0.504, and for $d = 4$ these values are 0.4646 and 0.4553, respectively. For sufficiently large $K$, we expect the approximation to get more accurate as the load decreases and as the parameter $d$ increases, since then the probability of waiting decreases.

# 9 Concluding remarks

We have obtained a unifying framework to investigate the steady state distribution of *c.o.s.* and *c.o.c.* redundancy models. In the literature there are already several studies of *c.o.c.*, but to the best of our knowledge, we are the first to analyze *c.o.s.* for any $1 \leq d \leq K$. In addition, we have established the equivalence between *c.o.s.* and $JSW(d)$, and as a consequence our approach provides a first exact analysis of workload-based load balancing schemes.

There are several interesting research problems that stem from our work. An important extension is to investigate *c.o.s.* with heterogeneous servers and class-based models. Our approach based on the multi-type multi-server queue will still be valid, however, the routing discipline that will satisfy the assignment condition, see Equation (2), needs to be determined. In this paper we have mostly focused on the queue length and related performance metrics. Obviously, the waiting time and the sojourn time distribution are very relevant metrics as well. We plan to extend the central queue approach to study the waiting time distribution of both *c.o.s.* and *c.o.c.* models. Another research path pertains with a thorough analysis of c.o.s. model in the mean- field regime. The proof for asymptotic optimality of JSW$d(K)$ is part of future work. Last but not least, the analysis of *c.o.c.* without the independence assumption is a very relevant problem, and we plan to study whether this model can be analyzed through a similar approach as the one developed in this paper.

# References


[1] Jeremy Visschers, Ivo Adan, and Gideon Weiss, "A product form solution to a system with multi-type jobs and multi-type servers," *Queueing Systems*, vol. 70, no. 3, pp. 269–298, 2012.

[2] Ganesh Ananthanarayanan, Srikanth Kandula, Albert G Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris, "Reining in the outliers in map-reduce clusters using mantri.," in *OSDI*, 2010, vol. 10, p. 24.

[3] Ashish Vulimiri, Philip Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott

Shenker, "Low latency via redundancy," in *Proceedings of the ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 283–294.

[4] Gauri Joshi, Emina Soljanin, and Gregory Wornell, "Queues with redundancy: Latency-cost analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 2, pp. 54–56, 2015.

[5] Nihar B Shah, Kangwook Lee, and Kannan Ramchandran, "When do redundant requests reduce latency?," *IEEE Transactions on Communications*, vol. 64, no. 2, pp. 715–722, 2016.

[6] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, Esa Hyytiä, and Alan Scheller-Wolf, "Queueing with redundant requests: exact analysis," *Queueing Systems*, vol. 83, no. 3-4, pp. 227–259, 2016.

[7] Kristen Gardner, Mor Harchol-Balter, Alan Scheller-Wolf, Mark Velednitsky, and Samuel Zbarsky, "Redundancy-d: The power of d choices for redundancy," *Operations Research*, 2017.

[8] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica, "Effective straggler mitigation: Attack of the clones.," in *NSDI*, 2013, vol. 13, pp. 185–198.

[9] Jeffrey Dean and Luiz André Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[10] Kangwook Lee, Nihar B Shah, Longbo Huang, and Kannan Ramchandran, "The mds queue: Analysing the latency performance of erasure codes," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 2822–2842, 2017.

[11] Kangwook Lee, Ramtin Pedarsani, and Kannan Ramchandran, "On scheduling redundant requests with cancellation overheads," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 2, pp. 1279–1290, 2017.

[12] Kristen Gardner, Mor Harchol-Balter, Alan Scheller-Wolf, and Benny Van Houdt, "A better model for job redundancy: Decoupling server slowdown and job size," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3353–3367, 2017.

[13] Thomas Bonald and Céline Comte, "Balanced fair resource sharing in computer clusters," *Performance Evaluation*, vol. 116, pp. 70–83, 2017.

[14] Thomas Bonald, Céline Comte, and Fabien Mathieu, "Performance of balanced fairness in resource pools: A recursive approach," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 41, 2017.

[15] AE Krzesinski, "Order independent queues," *Queueing Networks*, pp. 85–120, 2011.

[16] Gauri Joshi, Emina Soljanin, and Gregory Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 2, no. 2, pp. 12, 2017.

[17] Mor Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*, Cambridge University Press, New York, NY, USA, 2013.

[18] Maury Bramson, Yi Lu, and Balaji Prabhakar, "Asymptotic independence of queues under randomized load balancing," *Queueing Systems*, vol. 71, no. 3, pp. 247–292, 2012.

[19] Shlomo Halfin and Ward Whitt, "Heavy-traffic limits for queues with many exponential servers," *Operations research*, vol. 29, no. 3, pp. 567–588, 1981.

[20] Mark van der Boor, Sem C Borst, Johan SH van Leeuwaarden, and Debankur Mukherjee, "Scalable load balancing in networked systems: Universality properties and stochastic coupling methods," *arXiv preprint arXiv:1712.08555*, 2017.

# Appendix

## A: Proof of Lemma 1

Assuming the *uniform assignment rule*, we will prove that for every subset $\{M_i, \dots, M_1\}$ of $\mathcal{M}$ of size $i$ and for every $j$ ($1 \leq j \leq i$) we have $\lambda_{M_j}(\{M_1, \dots M_{j-1}\}) = \lambda_{\bar{M}_j}(\{\bar{M}_1, \dots \bar{M}_{j-1}\})$ for every permutation $\bar{M}_1, \dots \bar{M}_i$ of $M_1, \dots M_i$. From Eq. (14), this will imply the *assignment condition*.

To characterize $\lambda_{M_j}(\{M_1, \dots, M_{j-1}\})$ (the activation rate of server $M_j$, when the set of active servers is $\{M_1, \dots, M_{j-1}\}$), we will require the number of types that can activate server $M_j$ when servers $M_1$ to $M_{j-1}$ are busy. This can be obtained by first counting the number of types that can activate server $M_j$ while having exactly $\alpha$ of its $d$ *feasible* servers already busy (and hence among $M_1$ to $M_{j-1}$) and then summing over $\alpha$.

Now let $C(m_i)$ denote the set of types which have $m_i$ as a *feasible* server. Likewise, $C(M_i)$ would denote the same for a generic server $M_i$. Let $S_c$ denote the set of $d$ *feasible* servers for type $c$ jobs. Now when the system has exactly $j-1$ busy servers denoted by $\{M_1, \dots, M_{j-1}\}$ and when $M_j$ is an idle server, define

$$C^{\alpha}_{M_j}(\{M_1, \dots, M_{j-1}\}) := \\ \{c \in C(M_j) : |S_c \cap \{M_1, \dots, M_{j-1}\}| = \alpha\}$$

as the set of types that have the idle server $M_j$ as *feasible* and have exactly $\alpha$ out of their $d$ *feasible* servers

in the set $\{M_1, \ldots, M_{j-1}\}$. Hence for any type $c \in C^{\alpha}_{M_j}(\{M_1, \ldots, M_{j-1}\})$, there are exactly $d - \alpha$ idle servers (which include server $M_j$) and due to the *uniform assignment rule*, an arriving customer of this type will activate server $M_j$ with probability $\frac{1}{d-\alpha}$. Now it can be seen that

$$|C^{\alpha}_{M_j}(\{M_1, \ldots M_{j-1}\})| = \binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}.$$

Here $\binom{j-1}{\alpha}$ is the number of ways of choosing $\alpha$ servers out of $\{M_1, \ldots M_{j-1}\}$. For a particular choice of $\alpha$ servers say $\{M^1, \ldots M^{\alpha}\}$, and given $M_j$, the number of types that have these particular $\alpha + 1$ servers as *feasible* servers is $\binom{K-j}{d-\alpha-1}$. This explains the value of $|C^{\alpha}_{M_j}(\{M_1, \ldots M_{j-1}\})|$. Note additionally that this cardinality depends only on the number of busy servers $j - 1$ and not on the label of the servers that are actually busy. This implies that for any permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots M_i$ and for $0 \le j \le i$, we have

$$|C^{\alpha}_{M_j}(\{M_1, \ldots M_{j-1}\})| = |C^{\alpha}_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\})|. \quad (19)$$

Here $\{\bar{M}_1, \ldots \bar{M}_{j-1}\}$ is the first $j$ servers from the permutation
$\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots M_i$ for $0 \le j \le i$. Now note that

$$\begin{aligned}
& \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}) \\
= & \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \sum_{c \in C^{\alpha}_{M_j}(\{M_1,\ldots M_{j-1}\})} \frac{\lambda_{type}}{d-\alpha} \\
= & \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\lambda_{type}|C^{\alpha}_{M_j}(\{M_1, \ldots M_{j-1}\})|}{d-\alpha} \\
= & \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\lambda_{type}|C^{\alpha}_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\})|}{d-\alpha} \\
= & \lambda_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\})
\end{aligned}$$

where $\{\bar{M}_1, \ldots \bar{M}_{j-1}\}$ is the first $j$ servers from the permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots M_i$ for $0 \le j \le i$. Here the range of $\alpha$ in the index of the summation is so chosen that the quantity $\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}$ is property defined. The first equality above follows from the fact the each type
$c \in C^{\alpha}_{M_j}(\{M_1, \ldots M_{j-1}\})$ activates its idle feasible server $M_j$ with rate $\frac{\lambda_{type}}{d-\alpha}$ and the total activation rate for $M_j$ is obtained by summing over all such types and then over all $\alpha$. The third equality follows from Eq. (19) and the last equality in fact follows from the first. From this, it is easy to see that for $i = 1 \ldots K$,

$$\prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}) = \prod_{j=1}^{i} \lambda_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\}) \quad (20)$$

for every permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots M_i$. This completes the proof. □

## B: Proof of Proposition 1

For a redundancy-$d$ system with *c.o.s*, first note that if less than $d$ servers are busy, then the number of waiting copies (in the traditional parallel server representation) is zero. This is because a waiting copy implies that all the other $d - 1$ copies are also waiting for service; this is possible when at least $d$ servers are busy in the system. Therefore for any *feasible* state $s = (n_i, M_i, \ldots, n_1, M_1)$ we have $n_1 = n_2 = \ldots - n_{d-1} = 0$. Since the redundancy-$d$ system with *c.o.s* can be seen as a multi-type job and server model and since the *uniform assignment rule* satisfies the *assignment condition* (Lemma 1), we can apply Proposition 1. The steady state distribution for any state $s = (n_i, M_i, \ldots, n_1, M_1)$ with $i < d$ and $n^1 = \ldots = n^{d-1} = 0$ is now given by

$$\pi(s) = \frac{\Pi_{\lambda}(\{M_1, \ldots, M_i\})}{\Pi_{\mu}(M_i, \ldots, M_1)}\pi(0)$$

and for $i \ge d$ and $n^1 = \ldots = n^{d-1} = 0$ by

$$\pi(s) = {\alpha_i}^{n_i} \ldots {\alpha_d}^{n_d} \frac{\Pi_{\lambda}(\{M_1, \ldots, M_i\})}{\Pi_{\mu}(M_i, \ldots, M_1)}\pi(0).$$

Here

$$\begin{aligned}
\Pi_{\lambda}(\{M_1, \ldots, M_i\}) &= \prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}) \\
\Pi_{\mu}(M_i, \ldots, M_1) &= \prod_{j=1}^{i} \mu_{\{M_1, \ldots M_j\}} \\
\alpha_j &= \frac{\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})}}{\mu_{\{M_1,\ldots M_j\}}} \text{ for } 1 \le j \le i.
\end{aligned}$$

For the redundancy-$d$ system with *c.o.s*, the departure rate from any state with active servers $\{M_1, \ldots M_j\}$ is given by $\mu\{M_1, \ldots M_j\} = j\mu$ and hence

$$\Pi_{\mu}(M_i, \ldots, M_1) = \mu^i i!.$$

Recall from the preliminaries that $\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})}$ denotes the arrival rate of types that can be served only at servers $\{M_1, \ldots, M_j\}$. The number of such types is $\binom{j}{d}$ and the arrival rate of each such type is $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$. Therefore

$$\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})} = \lambda \frac{\binom{j}{d}}{\binom{K}{d}}.$$

Now using the fact that

$$\begin{aligned}
& \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}) \\
= & \lambda_{type} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{|C^{\alpha}_{M_j}(\{M_1, \ldots M_{j-1}\})|}{d-\alpha} \\
= & \lambda_{type} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha},
\end{aligned}$$

we have

$$\begin{aligned}\Pi_\lambda(\{M_1,\ldots,M_i\}) &= \prod_{j=1}^{i}\lambda_{M_j}(\{M_1,\ldots M_{j-1}\})\\ &= \prod_{j=1}^{i}\lambda_{type}\sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)}\frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha}.\end{aligned}$$

Now define

$$\begin{aligned}G_j(K,d) &:= \lambda_{type}\sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)}\frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha}\\ &= \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)}\frac{\lambda\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{(d-\alpha)\binom{K}{d}}\\ G^i(K,d) &:= \prod_{j=1}^{i}G_j(K,d) \text{ and}\\ r_i &:= \left(\lambda\frac{\binom{i}{d}}{i\mu\binom{K}{d}}\right)\end{aligned}$$

Using these definitions, the statement of the theorem follows. This completes the proof. □

## C: Proof of Lemma 2

When $i<d$ it is easy to see that

$$p(i)=\binom{K}{i}i!\pi(s)$$

where $s$ is any state of the form $s=(0,M_i,0,M_{i-1},\ldots,0,M_1)$ (since the servers are homogeneous, $\pi(s)$ for all such states is the same). The $i$ busy servers can be chosen in $\binom{K}{i}$ ways and there are $i!$ ways to arrange the servers leading to the above expression.

Recall that $r_i=\rho\bar{r}_i$. Since $\rho<1$, and $\bar{r}_i=\frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}}<1$, it directly follows that $r_i<1$. The probability of having $i$ busy servers, $p(i)$, for $i\geq d$ is now given by

$$\begin{aligned}p(i) &= \binom{K}{i}i!\sum_{n_i}\ldots\sum_{n_d}\pi(n^i,M_i,\ldots,n^d,M_d,\ldots,0,M_1)\\ &= \binom{K}{i}i!\sum_{n_i}\ldots\sum_{n_d}{r_i}^{n_i}\ldots{r_d}^{n_d}G^i(K,d)\frac{\pi(0)}{i!\mu}\\ &= \binom{K}{i}\left(\frac{1}{1-r_i}\right)\ldots\left(\frac{1}{1-r_d}\right)G^i(K,d)\frac{\pi(0)}{\mu^i}.\end{aligned}$$

Therefore we have

$$p(i) = \begin{cases}\binom{K}{i}G^i(K,d)\frac{\pi(0)}{\mu^i} & \text{for } i<d\\ \binom{K}{i}\left(\frac{1}{1-r_i}\right)\ldots\left(\frac{1}{1-r_d}\right)G^i(K,d)\frac{\pi(0)}{\mu^i} & \text{for } i\geq d.\end{cases}$$

Recall that $\binom{K}{i}\frac{G^i(K,d)}{\mu^i}=\binom{K}{i}\bar{G}^i(K,d)\rho^i$ and hence $\bar{p}(i)$ given in the statement now follows. Using the fact that $\sum_{i=0}^{K}p(i)=1$ and $p(i)=\bar{p}(i)\pi(0)$, one can see that

$$\pi(0)=\left(1+\sum_{i=1}^{K}\bar{p}(i)\right)^{-1}.$$

□

## D: Proof of Proposition 2

We first analyze $l_i(m)$ and try to obtain a recursive form for it. First note that for $i=d$, we have $l_d(m)=r_d^m$ for $m\geq 0$. Now for $i>d$,

$$\begin{aligned}l_i(m) &= \sum_{\substack{\{n_i\ldots n_d:\\ \sum_{l=d}^{i}n_l=m\}}}r_i^{n_i}\ldots r_d^{n_d}\\ &= \sum_{n_i=0}^{\infty}r_i^{n_i}\sum_{\substack{\{n_{i-1}\ldots n_d:\\ \sum_{l=d}^{i-1}n_l=m-n_i\}}}r_{i-1}^{n_{i-1}}\ldots r_d^{n_d}\\ &= \sum_{n_i=0}^{m}r_i^{n_i}l_{i-1}(m-n_i)\\ &= (h_i\star l_{i-1})(m)\end{aligned}$$

where $h_i(j):=r_i^j$ and $\star$ represents the convolution operator. We will now obtain the expression for $P_i(z)$, the P.G.F. for $p(i,m)$. Let $H_i(z)$ and $L_i(z)$ be the P.G.F.'s for $h_i(\cdot)$ and $l_i(\cdot)$ respectively. When there are $i<d$ servers busy, there are no jobs waiting. Hence, we directly have $P_i(z)=p(i)$, for $i<d$.

Let

$$f^i(K,d)=\binom{K}{i}\bar{G}^i(K,d)\rho^i.$$

For $i\geq d$, we have $p(i,m)=\pi(0)f^i(K,d)l_i(m)$. Further note when $i\geq d$, we have

$$\begin{aligned}H_i(z) &= \frac{1}{1-r_iz}\\ L_{i+1}(z) &= H_{i+1}(z)L_i(z) \text{ and}\\ P_i(z) &= \pi(0)f^i(K,d)L_i(z),\end{aligned}$$

and since $l_d(m)=r_d^m$, we have $L_d(z)=H_d(z)$. We therefore have for $i\geq d$,

$$P_i(z) = \pi(0)f^i(K,d)\left(\prod_{j=d}^{i}H_j(z)\right), \tag{21}$$

where $\pi(0)$ is given by Eq. (9). Now define

$$Geom_{r_j}(z):=\frac{1-r_j}{1-r_jz}.$$

$Geom_{r_j}(z)$ denotes the P.G.F. of a *geometric* random variable $X_j$ with parameter $r_j$. ($X_j$ has the distribution $P(X_j = n) = (1-r_j)r_j^n$.). Together with Lemma 2, we then have for $i \geq d$

$$\begin{aligned} P_i(z) &= \pi(0) f^i(K,d) \left( \prod_{j=d}^{i} H_j(z) \right) \\ &= p(i) \left( \prod_{j=d}^{i} Geom_{r_j}(z) \right). \end{aligned} \tag{22}$$

The P.G.F. for the number of waiting jobs is then given by

$$\hat{P}(z) = \sum_{i=1}^{K} P_i(z) = \sum_{i=0}^{d-1} p(i) + \sum_{i=d}^{K} P_i(z).$$

The required P.G.F. in the statement of the theorem now follows from Eq. (22).

The expected number of waiting jobs is given by $\sum_{i=1}^{K} \frac{dP_i(z)}{dz}\Big|_{z=1}$. Note that $\frac{dP_i(z)}{dz} = 0$ when $i < d$. When $i \geq d$,

$$\begin{aligned} \frac{dP_i(z)}{dz}\Big|_{z=1} &= p(i) \left( \prod_{j=d}^{i} Geom_{r_j}(z) \right)\Big|_{z=1} \left( \sum_{j=d}^{i} \frac{r_j}{1-r_j} \right) \\ &= p(i) \left( \sum_{j=d}^{i} \frac{r_j}{1-r_j} \right). \end{aligned}$$

(The second equality is because $\left(\prod_{j=d}^{i} Geom_{r_j}(z)\right)\Big|_{z=1} = 1$) The expected number of waiting jobs in the system ($E(Q)$) is hence given by

$$E(Q) = \sum_{i=d}^{K} p(i) \left( \sum_{j=d}^{i} \frac{r_j}{1-r_j} \right)$$

where $p(i)$ is given by Eq. (7) and (9). Since the mean number of jobs at the servers is same as the mean number of busy servers we have $E(N) = E(Q) + K\rho$. □

## E: Special cases; $d = 1$ and $d = K$

In this appendix, we will discuss the special cases for the *c.o.s.* model corresponding to $d = 1$ and $d = K$.

When $d = 1$, both *c.o.s.* and *c.o.c.* system are equivalent to a system with Bernoulli routing to the K servers. The queue length process of each server is independent and each server represents an $M/M/1$ queue with arrival rate $\frac{\lambda}{K}$ and service rate $\mu$. It is easy to check from our analysis of Section 4 that when $d = 1$, we have $G_j(K,d) = \frac{\lambda}{K}, G^i(K,d) = \left(\frac{\lambda}{K}\right)^i$ and $r_i = \rho$. Using the notation that $\rho = \frac{\lambda}{K\mu}$ (the load per $M/M/1$ queue), the steady-state probability from Eq (4) for any state $s = (n_i, M_i, \ldots, n_1, M_1)$ for $i \geq 0$ is given by

$$\pi(s) = \pi(0) \frac{\rho^{\left(i + \sum_{j=1}^{i} n_j\right)}}{i!}.$$

$p(i)$ is given by $\bar{p}(i)\pi(0)$ where from Eq. (8) we have $\bar{p}(i) = \binom{K}{i}\left(\frac{\rho}{1-\rho}\right)^i$. From Eq. (9), the normalizing constant is obtained as

$$\begin{aligned} \pi(0) &= \left( 1 + \sum_{i=1}^{K} \binom{K}{i} \left( \frac{\rho}{1-\rho} \right)^i \right)^{-1} \\ &= \left( \left( 1 + \frac{\rho}{1-\rho} \right)^K \right)^{-1} = (1-\rho)^K. \end{aligned}$$

Since $\pi(0)$ is also the probability that the system of K independent $M/M/1$ queues is empty, this justifies $\pi(0) = (1-\rho)^K$. Using this, $p(i)$ can be simplified to $p(i) = \binom{K}{i}\rho^i(1-\rho)^{K-i}$. Now using Proposition 2, it is easy to see that the P.G.F. for the number of waiting jobs in the redundancy-1 system is given by

$$\hat{P}(z) = \left( \frac{1 + \rho - \rho z}{1 - \rho z} \right)^K (1-\rho)^K.$$

Further, the mean number of waiting jobs and the total number of jobs in the system is given by $E(Q) = \frac{K\rho^2}{1-\rho}$ and $E(N) = K\frac{\rho}{1-\rho}$.

We now analyze the **redundancy-$K$** system with *c.o.s.*. The redundancy-$K$ system is same as an $M/M/K$ system with arrival rate of $\lambda$ and $K$ servers each with rate $\mu$. For this case, substituting $d = K$ in our analysis of Section 4 gives us $G_j(K,K) = \frac{\lambda}{K-j+1}, G^i(K,K) = \frac{\lambda^i (K-i)!}{K!}$ and $r_K = \rho$. Again, by simple substitution, the probability of $i$ busy servers $p(i)$ is given by $p(i) = \pi(0)\bar{p}(i)$ where

$$\bar{p}(i) = \begin{cases} \frac{(K\rho)^i}{i!} & \text{for } i < K \\ \left(\frac{1}{1-\rho}\right) \frac{(K\rho)^K}{K!} & \text{for } i = K. \end{cases}$$

The normalizing constant $\pi(0)$ is given by

$$\pi(0) = \left( \sum_{i=0}^{K-1} \frac{(K\rho)^i}{i!} + \left( \frac{1}{1-\rho} \right) \frac{(K\rho)^K}{K!} \right)^{-1}$$

which coincides with the normalizing constant of an $M/M/K$ system (see page 260, [17]). Note that $p(K)$, the probability that all servers are busy corresponds to the blocking probability of the $M/M/K$ system. In fact, we obtain

$$p(K) = \pi(0) \left( \frac{1}{1-\rho} \right) \frac{(K\rho)^K}{K!}$$

which as expected coincides with the celebrated Erlang-$C$ formula. Based on substituting $d = K$ in Proposition 2, we see that the P.G.F. for the number of waiting jobs in the redundancy-$K$ system is given by

$$\hat{P}(z) = \sum_{i=0}^{K-1} p(i) + p(K) Geom_\rho(z).$$

Further, the mean number of waiting jobs and total number of jobs in the system is given by $E(Q) = p(K)\frac{\rho}{1-\rho}$ and $E(N) = p(K)\frac{\rho}{1-\rho} + \frac{\lambda}{\mu}$. These expressions verify with the corresponding quantities for the $M/M/K$ system (see [17]).

## F: Proof outline of Proposition 5

The proof outline is as follows. We first state the transition rates and the global balance equation for the *generalized multi-type job and server model*. Then, we identify the partial balance equations (as in [1]) that need to be satisfied by our guess for the stationary distribution given in the statement of the theorem. Using the *generalized departure set functions* in the equations, we show that the stationary distribution from the theorem indeed satisfies the partial balance resulting in a product-form.

### Transition rates and global balance

The transitions from an arbitrary state are of the following type.

**Arrivals:** When in state $s = (n_i, M_i, \ldots, n_1, M_1)$, the total arrival rate is $\lambda$. An arriving job either finds no *feasible* server to be idle or is picked for service by a *feasible* idle server. In case of multiple *feasible* idle server, the *assignment rule* will determine which of the *feasible* and idle server will serve this arrival. It is asumed in the statement of the theorem that the assignment rule must satisfy the assignment condition. The arrival rate of jobs that find no *feasible* server to be idle is given by $\lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}$. The arrival rate of jobs that activate a *feasible* server (according to the *assignment rule*) is $\lambda - \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}$ respectively. As in [1], we have

$$\lambda - \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})} = \sum_{M \in \mathcal{M}\setminus\{M_1,\ldots,M_i\}} \lambda_M(\{M_1,\ldots,M_i\}). \tag{23}$$

**Departures:** On departure of a job in service, either the server serving this job become idle or the server serving this job picks some waiting job. These two cases are defined by the following two events that describe transition rates associated with a departure.

- { Rate P } : the rate of transition into state $s$ due to a departure such that the server serving the departing job becomes idle.
- { Rate Q } : the rate of transition into state $s$ due to a departure such that the server serving the departing job picks a waiting job.

The global balance equations for any state that is of the form $s = (n_i, M_i, \ldots, n_1, M_1)$ are as follows. When $n_i > 0$, we have

$$\begin{aligned} \pi(s)\left(\lambda + \mu_{\{M_1,\ldots M_i\}}\right) &= \{\text{ Rate P }\} + \{\text{ Rate Q }\} \quad (24)\\ &+ \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}\\ &\times \pi(n_i - 1, M_i, \ldots, n_1, M_1). \end{aligned}$$

When $n_i = 0$, we have

$$\begin{aligned} \pi(s)\left(\lambda + \mu_{\{M_1,\ldots M_i\}}\right) &= \{\text{ Rate P }\} + \{\text{ Rate Q }\} \quad (25)\\ &+ \lambda_{M_i}(\{M_1,\ldots,M_{i-1}\})\\ &\times \pi(n_{i-1}, M_{i-1}, \ldots, n_1, M_1). \end{aligned}$$

We will now characterize Rate P and Q in detail. As in [1], we denote the state $(n_i, M_i, \ldots, l, M, n_k - l, M_k, \ldots, n_1, M_1)$ by
$insert_{kl}^M(s)$ and the state

$$(n_i, M_i, \ldots, n_{j+1}, M_{j+1}, n_j + n_{j-1} + 1, M_{j-1}, \ldots,$$
$$\ldots, M_{k+1}, l, M_j, n_k - l, M_k, \ldots, n_1, M_1)$$

by $swap_{kl}^{M_j}(s)$. Note that in state $insert_{kl}^M(s)$ (resp. $swap_{kl}^{M_j}(s)$), the departure rate of server $M$ (resp. $M_j$) in our system is given by

$$\mu_{\{M_1,\ldots,M_k,M\}} - \mu_{\{M_1,\ldots,M_k\}} (\text{resp.} \mu_{\{M_1,\ldots,M_k,M_j\}} - \mu_{\{M_1,\ldots,M_k\}}).$$

Define

$$\delta_j(M) := \frac{\lambda_{\mathcal{U}(\{M_1\ldots M_j\})}}{\lambda_{\mathcal{U}(\{M_1\ldots M_j, M\})}}, \quad j = 1\ldots i.$$

Let $p_{kl}^M(s)$ (resp. $q_{kl}^{M_j}(s)$) denote the probability of transition from state $insert_{kl}^M(s)$ (resp. $swap_{kl}^{M_j}(s)$) to state $s$ due to departure of server $M$ (resp. $M_j$). Therefore $p_{kl}^M(s)$ is the probability that after departure of server $M$, the state $insert_{kl}^M(s)$ is such that there are no waiting jobs in the system that are *feasible* to server $M$. Similarly, $q_{kl}^{M_j}(s)$ is the probability that after departure of server $M_j$, the state $swap_{kl}^{M_j}(s)$ is such that after departure of server $M_j$, the next *feasible* job for this server is the $n_{j-1} + 1^{st}$ job after server $M_{j-1}$. Therefore we have,

$$p_{kl}^M(s) = \delta_k(M)^l \delta_{k+1}(M)^{n_{k+1}} \ldots \delta_i(M)^{n_i}$$

and

$$q_{kl}^{M_j}(s) = \delta_k(M_j)^l \delta_{k+1}(M_j)^{n_{k+1}} \ldots \delta_{j-1}(M_j)^{n_{j-1}}(1 - \delta_{j-1}(M_j)).$$

The rates P and Q can be quantified as below:

$$\begin{aligned} \{\text{ Rate P }\} &= \sum_{M \in \mathcal{M}\setminus\{M_1,\ldots,M_i\}} P_M(s)\\ \{\text{ Rate Q }\} &= \sum_{j=1}^{i} Q_{M_j}(s) \end{aligned}$$

where

$$\begin{aligned} P_M(s) &= \sum_{k=1}^{i}\sum_{l=0}^{n_k} \hat{\mu}_M(insert_{kl}^M(s))p_{kl}^M(s)\pi(insert_{kl}^M(s)) \\ &+ \mu_M p_{00}^M(s)\pi(s,0,M), \\ Q_{M_j}(s) &= \sum_{k=1}^{j-1}\sum_{l=0}^{n_k} \hat{\mu}_{M_j}(swap_{kl}^{M_j}(s))q_{kl}^{M_j}(s)\pi(swap_{kl}^{M_j}(s)) \\ &+ \mu_{M_j} q_{00}^{M_j}(s)\pi(swap_{00}^{M_j}(s)). \end{aligned}$$

Note the subtle difference between our definition of $P_M(s), Q_{M_j}(s)$ given above and that in [1]. Unlike [1], we have subsumed the departure rates inside the definition of $P_M(s), Q_{M_j}(s)$.

**Partial Balance:** Now as in the proof of Theorem 2, [1], the aim is to prove that the expression for $\pi(s)$ in the statement of the theorem satisfies the following 4 partial balance equation. For $n_i > 0$

$$\pi(s)\mu_{\{M_1,\ldots M_i\}} = \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}\pi(n_i-1,M_i,\ldots,n_1,M_1) \quad (26)$$

and for $n_i = 0$,

$$\pi(s)\mu_{\{M_1,\ldots M_i\}} = \lambda_{M_i}(\{M_1,\ldots,M_{i-i}\})\pi(n_{i-1},M_{i-1},\ldots,n_1,M_1). \quad (27)$$

$$\begin{aligned} \pi(s)\lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})} &= \{\text{ Rate Q }\} \\ &= \sum_{j=1}^{i} Q_{M_j}(s) \quad (28) \\ \pi(s)\lambda_M(\{M_1,\ldots,M_i\}) &= P_M(s). \quad (29) \end{aligned}$$

Here we have made use of Eq (23) to replace $\lambda$ in the global balance equations (24) and (25). Firstly, it is easy to see that Eq. (3) satisfies Eq. (26) and (27). Hence to complete the proof, it is sufficient to prove that Eq. (3) satisfies Eq. (28) and Eq. (29). Let us first prove that Eq. (3) satisfies Eq. (29). We proceed exactly as in the proof of Theorem 2 in [1] and divide each term in the summation of $P_M(s)$ by $\pi(s)\lambda_M(\{M_1,\ldots,M_i\})$. In the corresponding part of the proof of Theorem 2 in [1], we have $\mu_M$ instead of $\hat{\mu}_M(insert_{kl}^M(s))$. Proceeding in the same way and using the fact that $\hat{\mu}_M(insert_{kl}^M(s)) = \mu_{\{M_1,\ldots M_k,M\}} - \mu_{\{M_1,\ldots M_k\}}$ it can be shown that

$$\begin{aligned} &\hat{\mu}_M(insert_{kl}^M(s))p_{kl}^M(s)\frac{\pi(insert_{kl}^M(s))}{\pi(s)\lambda_M(\{M_1,\ldots,M_i\})} \\ =\ &(1-\beta_k)(\beta_k)^l(\beta_{k+1})^{n_{k+1}+1}\ldots(\beta_i)^{n_i+1} \end{aligned}$$

where

$$\beta_j = \frac{\mu_{\{M_1,\ldots M_j\}}}{\mu_{\{M_1,\ldots M_j,M\}}} 1 \le j \le i.$$

A similar procedure for the boundary state gives us

$$\hat{\mu}_M(s,0,M)p_{1n_1}^M(s)\frac{\pi(s,0,M)}{\pi(s)\lambda_M(\{M_1,\ldots,M_i\})} = \prod_{j=1}^{i}(\beta_j)^{n_j+1}.$$

Now as in [1], this implies that

$$\begin{aligned} &\frac{P_M(s)}{\pi(s)\lambda_M(\{M_1,\ldots,M_i\})} = \\ &\left[\prod_{j=1}^{i}(\beta_j)^{n_j+1} + \sum_{k=1}^{i}\sum_{l=0}^{n_k}(1-\beta_k)(\beta_k)^l\prod_{h=k+1}^{i}(\beta_h)^{n_h+1}\right] \\ &= 1. \end{aligned}$$

To prove, that Eq. (3) satisfies Eq. (28), as in the proof of Theorem 2 in [1], divide each term in the summation of $Q_{M_j}(s)$ by $\pi(s)$. Since the departure rate $\hat{\mu}_{M_j}(swap_{kl}^{M_j}(s))$ in state $swap_{kl}^{M_j}(s)$ is given by $\mu_{\{M_1,\ldots,M_k,M_j\}} - \mu_{\{M_1,\ldots,M_k\}}$, we can follow the exact steps in the proof of Theorem 2 in [1] to obtain

$$\begin{aligned} &\hat{\mu}_{M_j}(swap_{kl}^{M_j}(s))q_{kl}^{M_j}(s)\frac{\pi(swap_{kl}^{M_j}(s))}{\pi(s)} \\ =\ &\left(\lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1},M_j\})} - \lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1}\})}\right) \\ \times\ &(1-\beta_{k,j})(\beta_{k,j})^l(\beta_{k+1,j})^{n_{k+1}+1}\ldots(\beta_{j-1,j})^{n_{j-1}+1} \end{aligned}$$

where

$$\beta_{h,j} = \frac{\mu_{\{M_1,\ldots M_h\}}}{\mu_{\{M_1,\ldots M_h,M_j\}}}, 1 \le h \le j-1.$$

For the boundary states, we can show

$$\begin{aligned} &\hat{\mu}_{M_j}(swap_{00}^{M_j}(s))q_{1,n_1}^{M_j}(s)\frac{\pi(swap_{00}^{M_j}(s))}{\pi(s)} = \\ &\left(\lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1},M_j\})} - \lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1}\})}\right)\prod_{h=1}^{j-1}(\beta_{h,j})^{n_h+1} \end{aligned}$$

and that for $1 \le j \le i$, adding terms over $k$ and $l$ we would get

$$Q_{M_j}(s) = \pi(s)\left(\lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1},M_j\})} - \lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1}\})}\right).$$

Summing over all $j$ we have

$$\pi(s)\lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})} = \sum_{j=1}^{i} Q_{M_j}(s)$$

which is same as Eq. (28). This completes the proof outline. □

## G: Proof of Proposition 6

From Section 3, recall that the steady state exists for any $\rho < 1$. Since the redundancy-$d$ *c.o.c* model can be viewed as a *generalized multi-type job and server model* with state space representation $(n_i, O_{c_i}, \ldots, n_1, O_{c_1})$, we are in position to invoke Proposition 5 for this model. Therefore the steady state distribution in state $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ is given by

$$\pi(s) = {\alpha_i}^{n_i}\ldots{\alpha_1}^{n_1}\frac{\Pi_\lambda(\{O_{c_1},\ldots,O_{c_i}\})}{\Pi_\mu(O_{c_i},\ldots,O_{c_1})}\pi(0) \quad (30)$$

where

$$\begin{aligned}
\Pi_\lambda(\{O_{c_1},\ldots,O_{c_i}\}) &= \prod_{j=1}^{i} \lambda_{O_{c_j}}(\{O_{c_1},\ldots O_{c_{j-1}}\}),\\
\Pi_\mu(O_{c_i},\ldots,O_{c_1}) &= \prod_{j=1}^{i} \mu_{\{O_{c_1},\ldots O_{c_j}\}},\\
\alpha_j &= \frac{\lambda_{\mathcal{U}(\{O_{c_1},\ldots,O_{c_j}\})}}{\mu_{\{O_{c_1},\ldots O_{c_j}\}}}.
\end{aligned}$$

Now for the redundancy-$d$ *c.o.c* model, we have $\lambda_{O_{c_j}}(\{O_{c_1},\ldots O_{c_{j-1}}\}) = \lambda_{c_j} = \lambda_{type}$. Further $\lambda_{\mathcal{U}(\{O_{c_1},\ldots,O_{c_j}\})} = \sum_{l=1}^{j} \lambda_{c_l} = j\lambda_{type}$. Therefore we have,

$$\begin{aligned}
\Pi_\lambda(\{O_{c_1},\ldots,O_{c_i}\}) &= \prod_{j=1}^{i} \lambda_{type},\\
\alpha_j &= \left(\frac{j\lambda_{type}}{\mu|F_j(s)|}\right).
\end{aligned}$$

After rearranging the terms and bit of algebra, the steady state distribution for state $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ is given by

$$\pi(s) = \frac{\pi(0)}{i!} \prod_{j=1}^{i} \left(\frac{j\lambda_{type}}{\mu|F_j(s)|}\right)^{n_j+1} \tag{31}$$

□