Performance Analysis of Load Balancing Policies with Memory

Tim Hellemans and Benny Van Houdt
University of Antwerp
Middelheimlaan 1
Antwerp B-2020, Belgium

ABSTRACT

Joining the shortest or least loaded queue among d randomly selected queues are two fundamental load balancing policies. Under both policies the dispatcher does not maintain any information on the queue length or load of the servers. In this paper we analyze the performance of these policies when the dispatcher has some memory available to store the ids of some of the idle servers. We consider methods where the dispatcher discovers idle servers as well as methods where idle servers inform the dispatcher about their state.

We focus on large-scale systems and our analysis uses the cavity method. The main insight provided is that the performance measures obtained via the cavity method for a load balancing policy *with* memory reduce to the performance measures for the same policy *without* memory provided that the arrival rate is properly scaled. Thus, we can study the performance of load balancers with memory in the same manner as load balancers without memory. In particular this entails closed form solutions for joining the shortest or least loaded queue among d randomly selected queues with memory in case of exponential job sizes.

We present simulation results that support our belief that the approximation obtained by the cavity method becomes exact as the number of servers tends to infinity.

KEYWORDS

Memory; Power-of-d-choices; Large Scale Computer Network; Load Balancing

1 INTRODUCTION

Load balancing is often used in large-scale clusters to reduce latency. A simple algorithm, denoted by SQ(d), exists in assigning incoming jobs to a server that currently holds the least number of jobs out of d randomly selected queues. This is referred to as the *power-of-d-choices* algorithm [1, 11, 17]. Another popular algorithm which has received quite some attention recently exists in assigning an incoming job to the server which is the least loaded amongst d randomly selected queues, i.e. the server which will be first able to start working on the job receives the job. This policy is referred to as LL(d) and has been studied in e.g. [4, 7, 12, 13].

The main objective of this paper is to generalize the analysis of the SQ(d) and LL(d) policy to the case where the dispatcher has some (finite or infinite) memory available to store the ids of idle servers. These ids may be discovered by either probing servers to check whether they are idle or servers may inform the dispatcher that they became idle. We focus on the performance of large scale systems and as such make use of the cavity method introduced in [3]. The cavity method relies on the assumption that the queue length (or load) of any finite set of queues becomes independent as the number of servers tends to infinity, called the *ansatz*. While the ansatz was proven in some particular cases [4], this is very hard to do in general. Our objective is not to prove the ansatz for load balancers with memory, but to study these policies using the cavity method. To demonstrate the usefulness of our analysis we present simulation results that suggest that the cavity method captures the system behavior as the number of servers tends to infinity.

The main insight obtained in this paper is that studying a load balancing policy with memory using the cavity method, corresponds to studying the same load balancing policy without memory if we scale down the arrival rate in a proper manner (see also Theorem 5.3 and 5.8). For the LL(d) policy, we do not impose any restrictions on the job size distribution. For the SQ(d) policy, we primarily restrict our attention to exponential job sizes. Similar results may be obtained for phase type distributed job sizes (see Appendix A for more details).

As a by-product, our results allow us to study the Join-Idle-Queue policy (denoted by JIQ) with finite memory. JIQ exists in keeping track of the ids of the idle queues and assigning incoming jobs to an idle queue whenever there exists an idle server and simply assigning it to a random server otherwise. This policy has vanishing delays when the number of servers tends to infinity in case of infinite memory [5, 6, 10, 15].

The paper is structured as follows : In Section 2, the model is introduced and we shortly review previously obtained results for SQ(d) and LL(d). In Section 3 we present 4 natural methods which may be used to generate memory at the dispatcher and show how to obtain the probability that the memory is empty for each of these methods. Next we present our major analytical tool, the queue at the cavity in Section 5 and we describe how it is defined for the memory dependent LL(d) and SQ(d) policy. Our analysis is verified by means of simulation in Section 6. In Section 7 we show how our results may be used for numerical experimentation by studying one specific setting. We conclude the paper in Section 8 with a small summary and possible future work.

2 MODEL DESCRIPTION

We consider a system consisting of N identical servers (with N large). There is some central dispatcher to which jobs arrive according to a Poisson(λN) process. The dispatcher has some (finite or infinite) memory available to store ids of idle servers. When a job arrives and the dispatcher has the id(s) of some idle server(s) in its memory, the job is dispatched to a random server, the id of which is in memory. If the dispatcher's memory is empty, d servers are chosen at random and the job is either send to the server with the shortest queue (SQ(d), see Section 2.1) or to the server with the least amount of work (LL(d), see Section 2.2). Setting $d = 1$ yields the JIQ

policy where the job is simply routed arbitrarily whenever there are no idle servers known by the dispatcher. Before we proceed we provide some further details on the classic SQ(d) and LL(d) policy.

2.1 Classic SQ(d)

The SQ(d) policy was first introduced in [11, 17] for a system with Poisson(λN) arrivals and exponential job sizes (with mean $1/\mu$). Whenever a job arrives, d servers are probed at random and the incoming job is routed to the probed server with the least number of jobs in its queue. It was shown (see [11]) that in the limit as $N \to \infty$ the system behavior converges to the solution of the following set of ODEs:

$$\frac{d}{dt}u_k(t) = \lambda(u_{k-1}(t)^d - u_k(t)^d) - \mu(u_k(t) - u_{k+1}(t)),$$

where we denote by $u_k(t)$ the probability that, at time t, an arbitrary server has at least k jobs in its queue and $u_0(t) = 1$. This set of ODEs also corresponds to applying the cavity method to the SQ(d) policy. The fixed point of this set of ODEs obeys a simple recursive formula:

$$\mu u_{k+1} = \lambda u_k^d, \tag{1}$$

which has the closed form solution $u_k = \rho^{\frac{d^k-1}{d-1}}$ with $\rho = \lambda/\mu$.

2.2 Classic LL(d)

The LL(d) policy was analysed in [7] for a system with arbitrary job sizes with mean $\mathbb{E}[G]$ using the cavity method. Whenever a job arrives, d queues are probed and the job is sent to the queue which has the least amount of work left. This means that the job joins the queue at which its service can start the soonest. In practice this can be implemented through late binding, see also [13]. Let $\bar{F}(w)$ denote the equilibrium probability that an arbitrary queue has at least w work left using the cavity method. It is shown in [7] that $\bar{F}(w)$ satisfies the fixed point equation:

$$\bar{F}(w) = \rho - \lambda \int_0^w (1 - \bar{F}(u)^d)\bar{G}(w-u)\,du, \tag{2}$$

with $\rho = \lambda\mathbb{E}[G]$ and $\bar{G}(w-u)$ the probability that a job has a size greater than $w-u$. This fixed point equation can alternatively be written as the following Integro Differential Equation (IDE):

$$\bar{F}'(w) = -\lambda\left[\bar{G}(w) - \bar{F}(w)^d + \int_0^w \bar{F}(u)^d g(w-u)\,du\right],$$

with g the density function of the job size distribution. Both have the boundary condition $\bar{F}(0) = \rho$. Moreover, this equation has a closed form solution in case of exponential job sizes (with mean $1/\mu$):

$$\bar{F}(w) = (\rho + (\rho^{1-d} - \rho)e^{(d-1)w})^{\frac{1}{1-d}}. \tag{3}$$

3 DISCOVERING IDLE SERVERS

We now discuss a number of approaches for the dispatcher to discover ids of idle servers. In the first few approaches the dispatcher discovers idle servers by probing, while in the last approach the idle servers identify themselves to the dispatcher. Note that as the amount of incoming work per server per unit of time is equal to $\rho < 1$, no work is replicated, and all servers are identical, it follows that the steady state probability that a server is busy is given by ρ.[1]

3.1 Interrupted probing (IP)

In the first approach, called *interrupted* probing (IP), the dispatcher probes d servers when its memory is empty upon a job arrival. If there are $k \geq 1$ idle servers among the d probed servers, it sends the incoming job to one of the idle servers and stores ids of the $k-1$ other servers in memory. These $k-1$ ids are then used for the subsequent $k-1$ arrivals. Thus for these $k-1$ arrivals, the dispatcher does not probe any servers. As ρ is the steady state probability that a server is busy, we can find the probability π_0 of having no ids in memory when a new job arrives by looking at the Markov chain with state space $0, \ldots, d-1$ and transition probability matrix $M(\rho)$:

$$\begin{aligned} M(\rho)_{0,0} &= \rho^d + \binom{d}{1}\rho^{d-1}(1-\rho), \\ M(\rho)_{0,\ell} &= \binom{d}{\ell+1}\rho^{d-1-\ell}(1-\rho)^{\ell+1}, \\ M(\rho)_{k,k-1} &= 1, \end{aligned}$$

and $M(\rho)_{k,\ell} = 0$ otherwise.

As only the first row is non-trivial, it is not hard to check that $\pi = (\pi_0, \ldots, \pi_{d-1})$ given by:

$$\pi_k = \pi_0\left[1 - \sum_{j=0}^{k}\binom{d}{j}\rho^{d-j}(1-\rho)^j\right],$$

for $k \geq 1$ is an invariant vector of $M(\rho)$. From the requirement $\sum_{k=0}^{d-1}\pi_k = 1$ it then follows that

$$\pi_0 = \frac{1}{\rho^d + (1-\rho)d}. \tag{4}$$

The number of probes used per arrival is clearly given by $\pi_0 d$ which equals

$$\frac{1}{1 - \rho + \frac{\rho^d}{d}}.$$

The main advantage of the IP approach is that it uses far less than d probes per arrival when ρ is not too large (see also Figure 1b).

3.2 Continuous probing (CP)

This approach is similar to the IP approach, except that whenever we use a server id from memory for a job arrival, the dispatcher still probes d random servers. The ids of the d servers that are idle are subsequently added to memory. We assume that the available memory is unlimited.

In order to determine the probability π_0 of having a server id in memory, we need to study a Markov chain on an infinite state space.

[1] For SQ(d) this is shown explicitly in the proof of Theorem 5.2, while for LL(d) this easily follows from integrating both sides of (17).

Its transition probability matrix $M(\rho)$ has the following form:

$$M(\rho)_{0,0} = \rho^d + \binom{d}{1}\rho^{d-1}(1-\rho),$$
$$M(\rho)_{0,\ell} = \binom{d}{\ell+1}\rho^{d-1-\ell}(1-\rho)^{\ell+1},$$

for $1 \le \ell \le d-1$. For $k \ge 1$, we have

$$M(\rho)_{k,k-1+\ell} = \binom{d}{\ell}\rho^{d-\ell}(1-\rho)^{\ell},$$

for $k-1 \le \ell < d+k$, and $M(\rho)_{k,\ell} = 0$ otherwise. First note that if $d > \frac{1}{1-\rho}$, this Markov chain is transient as the drift in state $k > 0$ is given by $d(1-\rho)-1$, meaning after some point in time the chain never returns to state zero and all incoming arrivals can be assigned to an idle server. When $d < \frac{1}{1-\rho}$, the chain is positive recurrent and we need to determine $\pi_0 < 1$.

The average time the memory remains empty is equal to:

$$\frac{1}{1-M(\rho)_{0,0}} = \frac{1}{1-\rho^d - d\rho^{d-1}(1-\rho)}.$$

Furthermore, when the memory becomes non-empty, the length that it remains non-empty depends on the number of server ids that are placed into memory. More specifically let $\mathbb{E}[T_{k,0}]$ denote the expected first return time to 0 given that the chain starts in state $k > 0$, then:

$$\mathbb{E}[T_{k,0}] = k\mathbb{E}[T_{1,0}],$$

and the mean time that the Markov chain stays away from state 0 given that it just made a jump from state 0 to some state $k > 0$ is given by $\mathbb{E}[X_0]\mathbb{E}[T_{1,0}]$, where $\mathbb{E}[X_0]$ is one less than the mean number of idle servers among d servers given that at least 2 are idle. It is not hard to see that

$$\mathbb{E}[X_0] = \frac{d(1-\rho)-(1-\rho^d)}{1-\rho^d - d\rho^{d-1}(1-\rho)}.$$

Further, $\mathbb{E}[T_{1,0}] = \frac{1}{1-d(1-\rho)}$ as $\mathbb{E}[T_{1,0}] = 1 + d(1-\rho)\mathbb{E}[T_{1,0}]$. Putting this together we obtain

$$\pi_0 = \frac{1-d(1-\rho)}{\rho^d}, \tag{5}$$

when $d < \frac{1}{1-\rho}$. Note that the CP approach uses d probes per arrival.

3.3 Bounded Continuous Probing (BCP)

This approach is identical to the CP approach, but with finite memory size A. Hence the transition matrix $M(\rho)$ is of size $A+1$ and its transitions are the same as in Section 3.2, except that any transition from a state $k \le A$ to a state $\ell > A$ becomes a transition to state A. In particular for $k > A-d+1$ we have:

$$M(\rho)_{k,A} = \sum_{j=A-k+1}^{d} \binom{d}{j}\rho^{d-j}(1-\rho)^j,$$

and for all other values, $M(\rho)$ coincides with the expressions given in Section 3.2. This Markov chain does not appear to have a simple closed form solution for arbitrary values of d, however for $d = 2$ one finds:

$$\pi_0 = \frac{1-\left(\frac{1-\rho}{\rho}\right)^2}{1-\left(\frac{1-\rho}{\rho}\right)^{2(A+1)}}.$$

For $d > 2$ a simple numerical scheme can be used to compute π_0. Note that this approach uses d probes per arrival unless the dispatcher sends the probes one at a time and stops probing when the memory is full.

3.4 Other probing schemes

In this section we present a result that applies to any scheme where the dispatcher discovers idle servers by probing and any idle server that is discovered is stored in memory. Thus the result only applies to BCP if the probes are sent one at a time.

Proposition 3.1. *Assume all discovered idle servers are stored in memory. Then for any LL(d)/SQ(d) memory based policy, the average number of probes used per arrival is given by:*

$$\frac{1-\pi_0\rho^d}{1-\rho}. \tag{6}$$

Proof. If we think of the probes being transmitted one at a time and assigning the job as soon as an idle server is discovered, the dispatcher uses on average $\sum_{k=0}^{d-1}\rho^k$ probes for any job arrival that occurs when the memory is empty. Further, for any arrival that uses an id in memory, an average of $1/(1-\rho)$ probes was used to discover that id. Hence, the average number of probes transmitted can be written as:

$$\pi_0\frac{1-\rho^d}{1-\rho} + (1-\pi_0)\frac{1}{1-\rho}.$$

□

The above result indicates that for any such policy for which we either know the average number of probed queues (as for CP) or can express this using π_0 (as for IP), we immediately obtain π_0. As the CP policy sends d probes per arrival and the IP policy $\pi_0 d$, Proposition 3.1 yields (4) and (5) without the need to analyze a Markov chain.

3.5 Idle Server Messaging (ISM)

In this scheme the dispatcher does not probe to discover idle servers, instead a server notifies the dispatcher whenever it becomes idle. In case of infinite memory, the dispatcher knows all idle server ids at all times and the system reduces to the JIQ policy when $d = 1$. Our interest lies mostly in knowing what happens when memory is finite and the job is assigned to the shortest of d queues whenever the memory is empty when a job arrives.

If we denote A as the number of ids that can be stored in memory, we show that π_0 is given by

$$\pi_0 = \frac{1-(1-\rho^d)^{\frac{1}{A+1}}}{\rho^d}, \tag{7}$$

for SQ(d) with exponential job sizes (see Proposition 5.4) as well as for LL(d) for any job size distribution, which entails that π_0 is

insensitive to the job size distribution for LL(d) (see Proposition 5.9).

If we assume that the d probes are transmitted one at a time when memory is empty and the dispatcher stops probing as soon as an idle server is discovered. The number of probes and messages transmitted by the dispatchers and servers per job arrival can be expressed as:

$$\pi_0 \frac{1-\rho^d}{1-\rho} + (1-\pi_0\rho^d),$$

where the first term corresponds to the number of probes send per arrival by the dispatcher and the second correspond to the number of server messages per arrival (which is equal to the probability that a job is assigned to an idle server).

4 DESCRIPTION OF THE QUEUE AT THE CAVITY

Our analysis is based on the *queue at the cavity* method which was introduced in [3] to analyze load balancing systems. The key idea is to focus on the evolution of a single tagged queue, referred to as the queue at the cavity, and to assume that all other queues have the same queue length (or workload) distribution at any time t. Moreover the queue length (or workload) of any finite set of queues is assumed to be independent at any time t. We first explain the approach in a system without memory and then indicate how to adapt it to incorporate memory.

In a system without memory, the queue at the cavity experiences potential arrivals at a rate λd as this is the rate at which a tagged queue is selected as one of the d randomly selected queues. If a potential arrival occurs at time t, d i.i.d. random variables are initialized which have the same queue length (or workload) distribution as the queue at the cavity at time t. The potential arrival becomes an actual arrival if the first of these d random variables (which corresponds to the queue at the cavity) has the smallest value (where ties are broken at random). For SQ(d) with exponential job sizes with mean $1/\mu$ the queue length of the queue at the cavity decreases at a constant rate equal to μ in between potential arrivals, while for LL(d) the workload decreases linearly at rate 1 when larger than zero.

To incorporate memory into the cavity method we note that the state of the memory (that is, the number of ids that it contains) evolves at a faster time scale than the fraction of queues with a certain queue length (or workload). As such the state of the memory at time t is given by the steady state $\pi(t)$ of the discrete time Markov chain with transition matrix $M(\rho(t))$ that captures the evolution of the memory, where $\rho(t)$ is the fraction of busy servers at time t (see Section 3 for some examples with $\rho(t) = \rho$). For more details on the concept of the time-scale separation we employ, we refer the reader to [2].

Let $\pi_0(t)$, the first entry of $\pi(t)$, represent the probability that the memory is empty at time t. We modify the queue at the cavity by decreasing the potential arrival rate to the queue at the cavity to $\lambda d\pi_0(t)$, i.e. potential arrivals only occur when there is no empty queue to join in memory. These potential arrivals are then dealt with in the exact same manner as in the setting without memory. When the queue at the cavity is empty, we assume that on top of the potential arrival rate of $\lambda d\pi_0(t)$, we have an effective arrival rate of $\lambda\frac{1-\pi_0(t)}{1-\rho(t)}$. The latter arrival rate can be interpreted as follows: jobs arrive at rate λN, with probability $(1-\pi_0(t))$ such a job is assigned to a queue in memory and with probability $1/((1-\rho(t))N)$ the queue at the cavity gets the job as it is one of the $(1-\rho(t))N$ idle servers at time t.

In the next section we study the cavity process of SQ(d) and LL(d) with memory in detail. We assume job sizes have some general distribution with probability density function (pdf) g, cumulative distribution function (cdf) G and complementary cdf (ccdf) $\bar{G}$. For a random variable with cdf H we let $\mathbb{E}[H]$ denote its mean, $\mu = \frac{1}{\mathbb{E}[G]}$ the mean service rate and we have for the system load: $\rho = \lambda \cdot \mathbb{E}[G]$. Furthermore we let $\mathcal{G}$ denote a generic random variable with distribution G. We will often assume that $\mathcal{G}$ is an exponential random variable. Furthermore, for LL(d) we denote by f, F and $\bar{F}$ the pdf, cdf and ccdf of the workload distribution of the queue at the cavity in equilibrium (note that we have $\bar{F}(0) = \rho$). For SQ(d) we denote by u_k the equilibrium probability that the queue at the cavity has k or more jobs (with $u_0 = 1$ and $u_1 = \rho$).

5 ANALYSIS OF THE QUEUE AT THE CAVITY

We now analyze the queue at the cavity described in the previous section for SQ(d) and LL(d). Note that the results presented in this section apply to any of the memory schemes discussed in Section 3. To obtain results for a specific memory scheme one simply replaces π_0 by the appropriate expression. We show that the equilibrium queue length and workload distribution of SQ(d) and LL(d) with memory, respectively, have exactly the same form as in the same setting without memory if we replace λ by $\lambda\pi_0^{1/d}$ and divide by $\pi_0^{1/d}$. With respect to the response time distribution, we show that the system with memory and arrival rate λ has the same response time distribution as the system without memory and arrival rate $\lambda\pi_0^{1/d}$.

5.1 SQ(d)

We start by describing the transient behaviour of the queue at the cavity for SQ(d):

Proposition 5.1. *Consider the SQ(d) policy with memory, exponential job sizes with mean $1/\mu$ and arrival rate $\lambda < \mu$. Let $u_k(t)$ be the probability that the queue at the cavity has k or more jobs at time t, then*

$$\frac{d}{dt}u_k(t) = \lambda\pi_0(t)(u_{k-1}(t)^d - u_k(t)^d) - \mu(u_k(t) - u_{k+1}(t)), \quad (8)$$

$$\frac{d}{dt}u_1(t) = \lambda\pi_0(t)(u_0(t)^d - u_1(t)^d) + \lambda(1-\pi_0(t)) - \mu(u_1(t) - u_2(t)), \quad (9)$$

for $k \geq 2$ and $u_0(t) = 1$.

Proof. Let $\Delta > 0$ be arbitrary, we first assume that $k \geq 2$ and consider the cases in which the queue at the cavity may have k or more jobs at time $t + \Delta$. First, it may have exactly k jobs at time t and no departures occur in $[t, t+\Delta]$, this occurs with probability:

$$Q_{1,k} = (1-\mu\Delta)(u_k(t) - u_{k+1}(t)) + o(\Delta). \quad (10)$$

It may also have $k+1$ or more jobs at time t, and at most 1 departure occurs in $[t, t+\Delta]$:

$$Q_{2,k} = u_{k+1}(t) + o(\Delta). \tag{11}$$

A third possibility is that it had exactly $k-1$ jobs at time t and exactly one arrival occurs in $[t, t+\Delta]$ which joined the queue at the cavity, this occurs with probability:

$$\begin{aligned} Q_{3,k} &= \lambda d \left(\int_0^\Delta \pi_0(t+\delta) d\delta \right) (u_{k-1}(t) - u_k(t)) \\ &\quad \sum_{j=0}^{d-1} \frac{1}{j+1} \binom{d-1}{j} (u_{k-1}(t) - u_k(t))^j \cdot u_k(t)^{d-j} + o(\Delta) \\ &= \lambda \left(\int_0^\Delta \pi_0(t+\delta) d\delta \right) (u_{k-1}(t)^d - u_k(t)^d) + o(\Delta) \end{aligned} \tag{12}$$

We now obtain:

$$u_k(t+\Delta) = Q_{1,k} + Q_{2,k} + Q_{3,k},$$

subtracting $u_k(t)$ on both sides, dividing by Δ and taking the limit $\Delta \to 0$ yields (8). For (9), one needs to consider the same $Q_{1,k}, Q_{2,k}$ and $Q_{3,k}$ as for $k \geq 2$ for the case of potential arrivals. There is however an additional term for the case where the queue at the cavity is empty at time t and it experiences an arrival due to the memory induced arrival rate, this yields:

$$Q_{4,k} = \lambda \left(\int_0^\Delta \frac{1 - \pi_0(t+\delta)}{1 - u_1(t+\delta)} d\delta \right) (u_0(t) - u_1(t)) + o(\Delta),$$

one then obtains $u_1(t+\Delta) = Q_{1,k} + Q_{2,k} + Q_{3,k} + Q_{4,k}$, subtracting $u_1(t)$, dividing both sides by Δ and taking the limit $\Delta \to 0$ yields (9). Finally the last equation $u_0(t) = 1$ is trivial by the definition of $u_0(t)$. □

From the transient regime, we are able to deduce the equilibrium workload distribution:

Theorem 5.2. *Consider the SQ(d) policy with memory, exponential job sizes with mean $1/\mu$ and arrival rate $\lambda < \mu$. Let u_k be the equilibrium probability that the queue at the cavity has k or more jobs, then*

$$u_k = \rho^{\frac{d^k - 1}{d-1}} \cdot \pi_0^{\frac{d^{k-1}-1}{d-1}} = (\rho \pi_0^{1/d})^{\frac{d^k-1}{d-1}} / \pi_0^{1/d}, \tag{13}$$

for $k \geq 1$ and $\rho = \lambda/\mu$.

Proof. Taking the limit of $t \to \infty$ in (8-9) we find that the following holds:

$$\begin{aligned} 0 &= \lambda \pi_0 (u_0^d - u_1^d) + \frac{\lambda(1-\pi_0)}{1-\rho} \cdot (u_0 - u_1) - \mu \cdot (u_1 - u_2), \\ 0 &= \lambda \pi_0 (u_{k-1}^d - u_k^d) - \mu \cdot (u_k - u_{k+1}), \end{aligned}$$

for $k \geq 2$. Summing all of these equations yields $u_1 = \rho$, while taking the sum for $k \geq j$ implies that $u_j = \lambda \pi_0 u_{j-1}^d$ for $j \geq 2$. This simple recurrence relation has (13) as its unique solution. □

Comparing (13) with the solution of (1), we see that u_k is identical as in the setting without memory if we replace ρ by $\rho \pi_0^{1/d}$ and divide by $\pi_0^{1/d}$ (even for $k = 1$).

Theorem 5.3. *Let $0 < \lambda < \mu$ be arbitrary and R the response time of the SQ(d) policy with memory, exponential job sizes with mean $1/\mu$ and arrival rate λ. Further, let $\tilde{R}$ denote the response time for the same system without memory, but with arrival rate $\lambda \pi_0^{1/d}$, then $\tilde{R}$ and R have the same distribution.*

Proof. Let us denote by u_k and v_k the probability that the queue at the cavity has at least k jobs for the system with and without memory, respectively. We have $u_k = \pi_0^{-1/d} \cdot v_k$, for $k \geq 1$ and $u_0 = v_0 = 1$. Let $\bar{F}_X$ be the ccdf of X, then

$$\bar{F}_R(w) = (1-\pi_0) e^{-\mu w} + \pi_0 \sum_{k=0}^{\infty} (u_k^d - u_{k+1}^d) \sum_{n=0}^{k} \frac{w^n}{n!} e^{-\mu w},$$

as with probability $(1-\pi_0)$ the job joins an idle queue from memory (meaning the response time is simply exponential) and with probability $\pi_0(u_k^d - u_{k+1}^d)$ the job joins a queue with length k (yielding an Erlang $k+1$ response time). Exchanging the order of the sums and using $\pi_0 u_k^d = v_k^d$, for $k \geq 1$, implies that

$$\begin{aligned} \bar{F}_R(w) &= (1-\pi_0) e^{-\mu w} + \sum_{n=1}^{\infty} \frac{w^n}{n!} e^{-\mu w} v_n^d + \pi_0 e^{-\mu w} \\ &= \sum_{n=0}^{\infty} \frac{w^n}{n!} e^{-\mu w} v_n^d. \end{aligned}$$

Similarly,

$$\bar{F}_{\tilde{R}}(w) = \sum_{k=0}^{\infty} (v_k^d - v_{k+1}^d) \sum_{n=0}^{k} \frac{w^n}{n!} e^{-\mu w} = \sum_{n=0}^{\infty} \frac{w^n}{n!} e^{-\mu w} v_n^d.$$

□

Remark. *This result is generalized to phase type distributed job sizes in Appendix A.*

Using Theorem 5.2 we are able to obtain π_0 for the ISM memory scheme presented in Section 3.5.

Proposition 5.4. *For the SQ(d) policy with exponential job sizes and the ISM memory scheme presented in Section 3.5 we have*

$$\pi_0 = \frac{1 - (1-\rho^d)^{\frac{1}{A+1}}}{\rho^d}.$$

Proof. From Theorem 5.2 we have $u_1 = \rho$ and $u_2 = \pi_0 \rho^{d+1}$. For ISM the state of the memory evolves as a finite state birth-death process with birth rate $(u_1 - u_2)\mu$ (as this is the rate at which busy servers become idle) and death rate λ. Hence,

$$\pi_0 = \frac{1}{\sum_{i=0}^{A} (u_1 - u_2)^i / \rho^i} = \frac{1 - (1 - \pi_0 \rho^d)}{1 - (1 - \pi_0 \rho^d)^{A+1}}.$$

Therefore, $\rho^d = 1 - (1 - \pi_0 \rho^d)^{A+1}$, which yields the result. □

5.2 LL(d)

For LL(d), we again start by describing the transient regime (the proof is similar to the one presented in [7]).

Proposition 5.5. *The density of the cavity process associated to the memory dependent LL(d) policy satisfies the following Partial Integro Differential Equations (PIDEs):*

$$\frac{\partial f(t,w)}{\partial t} - \frac{\partial f(t,w)}{\partial w} = \lambda d\pi_0(t)\int_0^w f(t,u)\bar{F}(t,u)^{d-1}g(w-u)du + \lambda\pi_0(t)(1-\bar{F}(t,0)^d)g(w) - \lambda d\pi_0(t)f(t,w)\bar{F}(t,w)^{d-1} + \lambda(1-\pi_0(t))g(w) \quad (14)$$

$$\frac{\partial \bar{F}(t,0)}{\partial t} = -f(t,0^+) + \lambda\pi_0(t)(1-\bar{F}(t,w)^d) + \lambda(1-\pi_0(t)), \quad (15)$$

for $w > 0$, where $f(x,z^+) = \lim_{y\downarrow z} f(x,y)$.

Proof. Assume $w > 0$ and let $w > \Delta > 0$ be arbitrary. As for SQ(d), we write:

$$f(t+\Delta, w) = Q_{1,w} + Q_{2,w} + Q_{3,w}. \quad (16)$$

For $Q_{1,w}$ we consider the case where no arrivals occur in the interval $[t, t+\Delta]$: if the cavity queue at time t has a workload exactly equal to $w+\Delta$ and receives no arrivals in $[t, t+\Delta]$, it has a workload equal to w at time $t+\Delta$. Therefore we find:

$$Q_{1,w} = f(t, w+\Delta) - \lambda d\left(\int_0^\Delta \pi_0(t+\delta)f(t+\delta, w+\Delta-\delta)d\delta\right) + o(\Delta).$$

For $Q_{2,w}$ we consider the case where a single arrival occurs when the queue at the cavity is busy: in this case at some time $t+\delta, \delta \in [0,\Delta]$ an arrival of size $w+\Delta-u$ occurs, while the queue at the cavity has workload $u-\delta$ for some $u \in (\delta, w+\Delta]$. This arrival only joins the queue at the cavity if the other $d-1$ queues have a workload that exceeds $u-\delta$, hence we find:

$$Q_{2,w} = \lambda d\int_0^\Delta \pi_0(t+\delta)\int_{u=\delta}^{w+\Delta} f(t+\delta, u-\delta)\bar{F}(t+\delta, u-\delta)^{d-1} g(w+\Delta-u)dud\delta + o(\Delta).$$

Finally a single arrival may occur when the cavity queue is empty: in this case a job of size $w+\Delta-\delta$ arrives at time $t+\delta$ for some $\delta \in [0,\Delta]$. Hence,

$$Q_{3,w} = \lambda d\int_0^\Delta \pi_0(t+\delta)\frac{(1-\bar{F}(t+\delta,0)^d)}{d}g(w+\Delta-\delta)d\delta + \lambda\int_0^\Delta \frac{1-\pi_0(t+\delta)}{1-\bar{F}(t+\delta,0)}(1-\bar{F}(t+\delta,0))g(w+\Delta-\delta)d\delta + o(\Delta).$$

By subtracting $f(t, w+\Delta)$, dividing by Δ and letting Δ decrease to zero, we find (14) from (16).

We still require an equation for $F(t,0)$, the probability that the server is idle. A server may be idle at time $t+\Delta$ by remaining idle in $[t, t+\Delta]$ or by having a workload equal to $\Delta-\delta, \delta < \Delta$ at time $t+\delta$. We therefore find:

$$F(t+\Delta,0) = F(t,0) - \lambda d\int_0^\Delta \pi_0(t+\delta)\frac{(1-\bar{F}(t+\delta,0)^d)}{d}\,d\delta - \lambda\int_0^\Delta \frac{1-\pi_0(t+\delta)}{1-\bar{F}(t+\delta,0)}(1-\bar{F}(t+\delta,0))\,d\delta + \int_0^\Delta f(t+\delta, \Delta-\delta)\,d\delta + o(\Delta),$$

subtracting $F(t,0)$, dividing by Δ and letting Δ tend to zero yields (15) after multiplying both sides by (-1). □

This result readily provides us with the equilibrium workload distribution for the LL(d) policy with memory:

Theorem 5.6. *The ccdf of the equilibrium workload distribution for the cavity process associated to an LL(d) policy with memory satisfies the following IDE:*

$$\bar{F}'(w) = -\lambda\left[\bar{G}(w) + \pi_0\cdot\left(-\bar{F}(w)^d + \int_0^w \bar{F}(u)^d g(w-u)\,du\right)\right]. \quad (17)$$

with boundary condition $\bar{F}(0) = \rho$. Equivalently we have:

$$\bar{F}(w) = \rho - \lambda\int_0^w (1-\pi_0\bar{F}(u)^d)\bar{G}(w-u)\,du. \quad (18)$$

with π_0 the probability that the memory is empty.

Proof. To show this result, one first lets $t\to\infty$ in (14-15), this way we remove the $\frac{\partial f(t,w)}{\partial t}$ and $\frac{\partial \bar{F}(t,0)}{\partial t}$. One then integrates (14) once and uses (15) as a boundary condition. Using Fubini, simple integration techniques and the fact that $f(w) = -\bar{F}'(w)$ we obtain (17). The last equality (18) can be shown by integrating once more and applying Fubini's theorem. □

We can rewrite (18) as

$$\pi_0^{1/d}\bar{F}(w) = \mathbb{E}[G](\lambda\pi_0^{1/d}) - (\lambda\pi_0^{1/d})\int_0^w (1-(\pi_0^{1/d}\bar{F}(u))^d)\bar{G}(w-u)\,du.$$

Comparing this expression with (2), we note that $\bar{F}(w)$ in a system with memory is equal to the same probability in a system without memory with arrival rate $\lambda\pi_0^{1/d}$ divided by $\pi_0^{1/d}$. Due to (3) we therefore have the following corollary:

Corollary 5.7. *The equilibrium workload of the queue at the cavity of an LL(d) system with memory and exponential job sizes is given by*

$$\bar{F}(w) = (\rho\pi_0 + (\rho^{1-d} - \rho\pi_0)e^{(d-1)w})^{\frac{1}{1-d}} \quad (19)$$

We are now able to show our main result for a memory dependent LL(d) policy:

Theorem 5.8. *Let $0 < \rho = \lambda\mathbb{E}[G] < 1$ be arbitrary and R the response time of the memory dependent LL(d) policy with mean job size $\mathbb{E}[G]$ and arrival rate λ. Further, let $\tilde{R}$ denote the response time for the same system without memory, but with arrival rate $\lambda\pi_0^{1/d}$, then R and $\tilde{R}$ have the same distribution*

Proof. Let $\bar{F}(w)$ and $\bar{H}(w)$ be the ccdf of the workload for the system with and without memory, respectively. We have $\bar{F}(w)\pi_0^{1/d} = \bar{H}(w)$ which yields:

$$\bar{F}_R(w) = (1-\pi_0)\bar{G}(w) + \pi_0\left[\int_0^w \bar{F}(w-u)^d g(u)\,du + \bar{G}(w)\right] = \bar{G}(w) + \int_0^w \bar{H}(w-u)^d g(u)\,du,$$

which can easily be seen to be equal to $\bar{F}_{\tilde{R}}(w)$. □

Remark. *When $d = 1$ in Theorem 5.8, the system without memory reduces to an ordinary M/G/1 queue with arrival rate $\lambda\pi_0^{1/d}$ for which many results exist. In particular, we find from the Pollaczek-Khinchin formula that the following holds:*

$$R^*(w) = \frac{(1-\pi_0^{1/d})\rho\mathcal{G}^*(w)w}{\pi_0^{1/d}\lambda\mathcal{G}^*(w) + w - \pi_0^{1/d}\lambda} \tag{20}$$

with R^* and $\mathcal{G}^*$ the Laplace transform of R and $\mathcal{G}$, respectively. Using the ISM scheme presented in Section 3.5, this allows one to analyse the JIQ policy with finite memory by plugging $\pi_0 = \frac{1-(1-\rho)^{\frac{1}{A+1}}}{\rho}$ into (20) *(see also Proposition 5.9).*

By using the results in this section, one can easily generalise many of the results presented in [7] including an analytical proof that LL(d) outperforms SQ(d) and closed form solutions for the response time distribution, mean response time and mean workload.

Proposition 5.9. *For the LL(d) policy with the ISM memory scheme presented in Section 3.5 we have*

$$\pi_0 = \frac{1-(1-\rho^d)^{\frac{1}{A+1}}}{\rho^d}$$

for any job size distribution.

Proof. The rate at which servers send probes is equal to $f(0) = -\bar{F}'(0)$ and it follows from (17) that $f(0) = \lambda(1-\pi_0\rho^d)$. The memory state therefore evolves as a birth-death process with birth rate $\lambda(1-\pi_0\rho^d)$ and death rate λ. The remainder of the proof is therefore identical to the proof of Proposition 5.4. □

As LL(1) and SQ(1) are identical the result in Proposition 5.4 is also valid for any job size distribution when $d = 1$. In fact, it can be shown that this remains true for $d > 1$.

6 FINITE SYSTEM ACCURACY

The results presented in Section 5 all focused on the cavity process of SQ(d) and LL(d) with memory. In Table 1 we present simulation results which illustrate that the stationary mean response time in a finite stochastic system consisting of N servers converges to the mean response time obtained using the cavity method. We simulated a system with $N = 10, 20, 50, 100, 200, 500, 1000$ and 3000 servers. The arrival rate equaled λN, the runtime was set to $10^6/N$ and we used a warm-up period equal to a third of the runtime. Job sizes have mean one and are either exponential or hyperexponential with balanced means and a Squared Coefficient of Variation (SCV) equal to 2 or 3.

The following 8 arbitrarily chosen settings have been considered:
Setup 1 : LL(4), $\lambda = 0.9$, exponential job sizes and the IP memory scheme.
Setup 2 : LL(3), $\lambda = 0.8$, exponential job sizes and the CP memory scheme (meaning memory is of infinite size).
Setup 3 : LL(3), $\lambda = 0.8$, hyperexponential job sizes with SCV equal to 2 and BCP memory scheme with $A = 5$.
Setup 4 : LL(2), $\lambda = 0.85$, hyperexponential job sizes with SCV equal to 3 and the ISM memory scheme with $A = 10$.
Setups 5 through 8 are the same as 1 through 4, but using SQ(d) rather than LL(d). In all cases the mean response time appears to converge towards the response time of the cavity method. Note that in the last two setups we are considering SQ(d) with memory and hyperexponential job sizes. In this case the response time of the cavity method is simply computed as the response time in the same system without memory, but with arrival rate $\lambda\pi_0^{1/d}$. This illustrates that the results for SQ(d) are indeed not limited to exponential job sizes.

7 NUMERICAL EXAMPLE

In this section we briefly demonstrate the type of numerical results that can be obtained using our findings. This section is not intended as a detailed comparison of the different memory schemes presented in Section 3.

Figure 1 focuses on the SQ(5) policy with exponential job sizes with mean one and a memory size A of 4 (except for CP). For the BCP and ISM memory schemes the dispatcher is assumed to send its d probes one at a time (if memory is empty upon a job arrival) and stops probing as soon as an idle server is found. This is also the case for the setting without memory (labeled *No memory*). For the CP memory scheme we assume that the dispatcher has infinite memory. We plot the mean response times, the probabity of having empty memory π_0 and the average number of probes/messages used per job arrival.

In Figure 1a we see that the mean response time is nearly optimal for all schemes when the load is low (say below 0.5). For higher loads we see that the ISM scheme is the best, followed by the CP/BCP, IP and No Memory scheme. The ISM scheme is especially powerful when the load is close to one as all the other schemes use probing and probes are highly unlikely to locate an idle server. The results of CP and BCP are very close to each other, which indicates that a very small amount of memory may suffice.

In Figure 1b we depict the average number of probes that each of these memory schemes use. If we look at the results for the No Memory, CP/BCP and IP scheme, we see that the schemes that achieved a lower mean response time use more probes. In this particular case the BCP scheme may appear to be superior to CP as it has a similar response time and uses far less probes, but keep in mind that probes are transmitted one at a time by BCP, while CP can transmit the d probes at once (which is faster). Looking at both the mean response time and number of probes/messages used the ISM scheme is clearly best in this case.

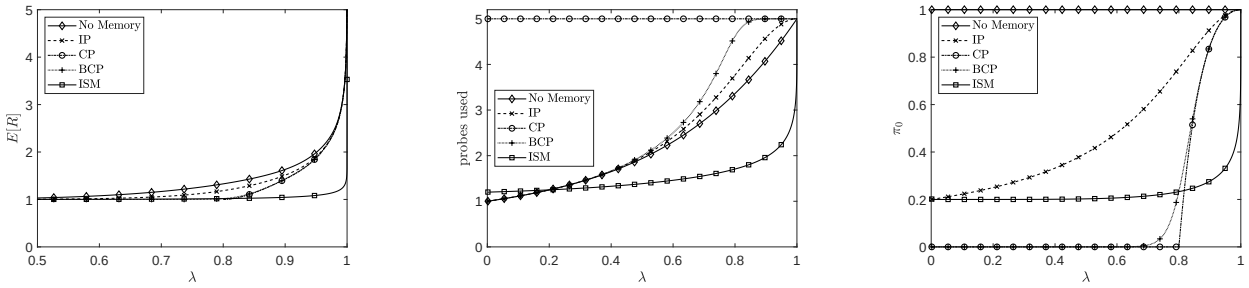
In Figure 1c we look at the probability of having an empty memory when a job arrives. For the IP scheme and a load $\lambda \approx 0$, the dispatcher almost always discovers 5 idle servers and therefore π_0 is close to $1/5$. For (B)CP we note that as long as the load is sufficiently low (that is, $5 < \frac{1}{1-\lambda}$ or equivalently $\lambda < 4/5$), we have $\pi_0 \approx 0$, but for larger λ values it sharply increases to one. When $\lambda \approx 4/5$ we also see the most significant gain in response time for (B)CP (see Figure 1a). For the ISM memory scheme, we observe that when λ is sufficiently small:

$$\pi_0 \approx \frac{1}{5} = \frac{1}{A+1} = \lim_{\rho\to 0^+} \frac{1-(1-\rho^d)^{\frac{1}{A+1}}}{\rho^d},$$

which is independent of d. Only when λ is close to one, π_0 starts a very steep climb to one.

Setup	$N = 10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$	$N = 500$	$N = 1000$	$N = 3000$	Cavity Method
1	1.8839	1.5363	1.3556	1.3059	1.2832	1.2683	1.2638	1.2574	1.2583
2	1.4533	1.3119	1.2313	1.2045	1.1926	1.1836	1.1810	1.1794	1.1787
3	1.5906	1.3860	1.2787	1.2399	1.2215	1.2110	1.2097	1.2068	1.2058
4	1.9086	1.3981	1.1643	1.1158	1.0999	1.0928	1.0921	1.0896	1.0888
5	2.3918	2.0132	1.8200	1.7733	1.7407	1.7252	1.7178	1.7146	1.7138
6	1.7583	1.5920	1.4943	1.4578	1.4404	1.4314	1.4304	1.4257	1.4256
7	2.0504	1.8040	1.6643	1.6161	1.5901	1.5753	1.5736	1.5667	1.5660
8	2.2790	1.5924	1.2950	1.2352	1.2186	1.2097	1.2096	1.2070	1.2056

Table 1: Comparison of mean response time for the finite system and the cavity method.



(a) Mean response time. **(b) Number of probes used per arrival.** **(c) Prob. of having empty memory.**

Figure 1: Performance of the different memory schemes for SQ(5) with exponential job sizes with mean one.

8 CONCLUSIONS AND FUTURE WORK

In this paper we studied the cavity process of the SQ(d) and LL(d) load balancing policies with memory. The main insight provided was that the response time distribution of the cavity process with memory is identical to the response time distribution of the cavity process of the system without memory if the arrival rate is properly set. This result holds for a large variety of memory schemes including the ones presented in Section 3. Simulation results were presented which suggest that the cavity process corresponds to the exact limit process as the number of servers tends to infinity.

The work presented in this paper can be extended in a number of manners. The results for the SQ(d) policy, which were limited to exponential job sizes, can be generalized to more general job size distributions. In case of exponential job sizes, it may be possible to prove that the cavity process is the proper limit process using the framework of [2]. It may also be possible to study load balancers with memory in the heavy traffic regime.

REFERENCES

[1] Reza Aghajani, Xingjie Li, and Kavita Ramanan. 2017. The PDE Method for the Analysis of Randomized Load Balancing Networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 38.

[2] Michel Benaim and Jean-Yves Le Boudec. 2008. A class of mean field interaction models for computer and communication systems. *Performance evaluation* 65, 11-12 (2008), 823–838.

[3] M. Bramson, Y. Lu, and B. Prabhakar. 2010. Randomized load balancing with general service time distributions. In *ACM SIGMETRICS 2010*. 275–286. https://doi.org/10.1145/1811039.1811071

[4] M. Bramson, Y. Lu, and B. Prabhakar. 2012. Asymptotic independence of queues under randomized load balancing. *Queueing Syst.* 71, 3 (2012), 247–292. https://doi.org/10.1007/s11134-012-9311-0

[5] Anton Braverman. 2018. Steady-state analysis of the Join the Shortest Queue model in the Halfin-Whitt regime. *arXiv preprint arXiv:1801.05121* (2018).

[6] Sergey Foss and Alexander L Stolyar. 2017. Large-scale join-idle-queue system with general service times. *Journal of Applied Probability* 54, 4 (2017), 995–1007.

[7] T. Hellemans and B. Van Houdt. 2018. On the Power-of-d-choices with Least Loaded Server Selection. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 2 (2018), 27.

[8] Jan Kriege and Peter Buchholz. 2014. PH and MAP fitting with aggregated traffic traces. In *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Springer, 1–15.

[9] Guy Latouche and Vaidyanathan Ramaswami. 1999. *Introduction to matrix analytic methods in stochastic modeling*. Vol. 5. Siam.

[10] Yi Lu, Qiaomin Xie, Gabriel Kliot, Alan Geller, James R Larus, and Albert Greenberg. 2011. Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. *Performance Evaluation* 68, 11 (2011), 1056–1071.

[11] M. Mitzenmacher. 2001. The Power of Two Choices in Randomized Load Balancing. *IEEE Trans. Parallel Distrib. Syst.* 12 (October 2001), 1094–1104. Issue 10.

[12] Michael Mitzenmacher. 2019. The Supermarket Model with Known and Predicted Service Times. *arXiv preprint arXiv:1905.12155* (2019).

[13] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica. 2013. Sparrow: Distributed, Low Latency Scheduling. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13)*. ACM, New York, NY, USA, 69–84. https://doi.org/10.1145/2517349.2522716

[14] A. Panchenko and A. Thümmler. 2007. Efficient Phase-type Fitting with Aggregated Traffic Traces. *Perform. Eval.* 64, 7-8 (Aug. 2007), 629–645. https://doi.org/10.1016/j.peva.2006.09.002

[15] A. L. Stolyar. 2015. Pull-based load distribution in large-scale heterogeneous service systems. *Queueing Systems* 80, 4 (2015), 341–361. https://doi.org/10.1007/s11134-015-9448-8

[16] Ignace Van Spilbeeck and Benny Van Houdt. 2015. Performance of rate-based pull and push strategies in heterogeneous networks. *Performance Evaluation* 91 (2015), 2–15.

[17] N.D. Vvedenskaya, R.L. Dobrushin, and F.I. Karpelevich. 1996. Queueing System with Selection of the Shortest of Two Queues: an Asymptotic Approach. *Problemy Peredachi Informatsii* 32 (1996), 15–27.

Appendices

A APPENDIX : PHASE TYPE JOB SIZES

Phase Type (PH) distributions consist of all distributions which have a modulating finite background Markov chain (see also [9]). They form a broad spectrum of distributions as any positive valued distribution can be approximated arbitrarily close by a PH distribution. Moreover, various fitting tools are available online for PH distributions (e.g. [8, 14]). A PH distribution with $\bar{G}(0) = 1$ is fully characterized by a stochastic vector $\alpha = (\alpha_i)_{i=1}^n$ and a subgenerator matrix $A = (a_{i,j})_{i,j=1}^n$ such that $\bar{G}(w) = \alpha e^{Aw}\mathbf{1}$, where $\mathbf{1}$ is a column vector of ones.

We find that the result found in Theorem 5.3 generalizes to the case of PH distributed job sizes.

Theorem A.1. *Let $0 < \lambda < \mu$ (with $1/\mu$ the mean of the job size distribution) be arbitrary and R the response time for a memory dependent version of the SQ(d) policy with PH distributed job sizes with parameters (α, A). Further, let $\tilde{R}$ denote the response time for the classic SQ(d) policy with the same job size distribution and arrival rate $\lambda\pi_0^{1/d}$, then R is equal to $\tilde{R}$.*

Proof. Let us denote by $u_{k,j}(t)$ resp. $v_{k,j}(t)$ the probability that, at time t, the queue at the cavity has at least k jobs and the job at the head of the queue is in phase j for the memory independent scheme resp. the memory dependent scheme. Furthermore let $u_{k,j}$ and $v_{k,j}$ denote the limit of $t \to \infty$ for these values. We first show that $u_{k,j} = \pi_0^{1/d} \cdot v_{k,j}$. Throughout we let $\nu = -A\mathbf{1}$ (with $\mathbf{1}$ a vector consisting of only ones). For $u_{k,j}$ we find by an analogous reasoning as in [16] that for $k \geq 2$:

$$\frac{d}{dt}u_{k,j}(t) = \lambda\pi_0(t)^{1/d}\frac{u_{k-1,j}(t) - u_{k,j}(t)}{u_{k-1}(t) - u_k(t)}(u_{k-1}(t)^d - u_k(t)^d) + \sum_{j'}\left(u_{k,j'}(t)A_{j',j} + u_{k+1,j'}(t)\nu_{j'}\alpha_j\right), \tag{21}$$

where $u_k(t)$ denotes $\sum_j u_{k,j}(t)$ (further on, we also use this notation for $u_k, v_k(t)$ and v_k). For $k = 1$ we find:

$$\frac{d}{dt}u_{1,j}(t) = \alpha_j\lambda\pi_0(t)^{1/d}(1 - u_1(t)^d) + \sum_{j'}\left(u_{1,j'}(t)A_{j',j} + u_{2,j'}(t)\nu_{j'}\alpha_j\right). \tag{22}$$

Taking the limit of t to infinity and multiplying by $\pi_0^{-1/d}$ we find that (21) yields for the equilibrium distribution (with $k \geq 2$):

$$0 = \pi_0\lambda\frac{(\pi_0^{-1/d}u_{k-1,j}) - (\pi_0^{-1/d}u_{k,j})}{(\pi_0^{-1/d}u_{k-1}) - (\pi_0^{-1/d}u_k)} \cdot \left((\pi_0^{-1/d}u_{k-1})^d - (\pi_0^{-1/d}u_k)^d\right) + \sum_{j'}(\pi_0^{-1/d}u_{k,j'})A_{j',j} + (\pi_0^{-1/d}u_{k+1,j'})\nu_{j'}\alpha_j. \tag{23}$$

while for $k = 1$ one may compute from (22):

$$0 = \alpha_j\lambda\left(1 - \pi_0(\pi_0^{-1/d}u_1)^d\right) + \sum_{j'}\left((\pi_0^{-1/d}u_{1,j'})A_{j',j} + (\pi_0^{-1/d}u_{2,j'})\nu_{j'}\alpha_j\right) \tag{24}$$

For $(v_{k,j}(t))$ with $k \geq 2$, we find the same ODE as (21) but with $\lambda\pi_0$ rather than $\lambda\pi_0^{1/d}$. Taking the limit $t \to \infty$ it is not hard to see that $v_{k,j}$ satisfies (23) with $\pi_0^{-1/d}u_{k,j}$ replaced by $v_{k,j}$. Furthermore for $v_{1,j}(t)$ we find (similar to Proposition 5.1):

$$\frac{d}{dt}v_{1,j}(t) = \lambda\alpha_j\pi_0(t)(1 - v_1(t)^d) + \lambda\alpha_j(1 - \pi_0(t)) + \sum_{j'}(v_{1,j'}(t)A_{j',j} + v_{2,j'}(t)\nu_{j'}\alpha_j).$$

Taking $t \to \infty$ it is not hard to see how this equation for $v_{k,j}$ reduces to (24) with $\pi_0^{-1/d}u_{k,j}$ replaced by $v_{k,j}$. This shows that we indeed have for all k and j that $u_{k,j} = \pi_0^{1/d}v_{k,j}$.

For the response time distribution we denote by $X_{k,j}$ the response time of a job that joins a queue with length k in phase j. We find for the memory dependent policy:

$$\begin{aligned}\bar{F}_R(w) &= (1 - \pi_0)\bar{G}(w) + \pi_0\Bigg((1 - v_1^d)\bar{G}(w) + \sum_{k=1}^{\infty}\sum_j \frac{v_{k,j} - v_{k+1,j}}{v_k - v_{k+1}} \cdot (v_k^d - v_{k+1}^d)\mathbb{P}\{X_{k,j} > w\}\Bigg) \\ &= (1 - (\pi_0^{1/d}v_1)^d)\bar{G}(w) + \sum_{k=1}^{\infty}\sum_j \frac{\pi_0^{1/d}v_{k,j} - \pi_0^{1/d}v_{k+1,j}}{\pi_0^{1/d}v_k - \pi_0^{1/d}v_{k+1}} \cdot \left((\pi_0^{1/d}v_k)^d - (\pi_0^{1/d}v_{k+1})^d\right)\mathbb{P}\{X_{k,j} > w\}\end{aligned}$$

One can now easily check that R and $\tilde{R}$ indeed coincide (using the fact that $u_{k\cdot j} = \pi_0^{1/d}v_{k,j}$). □