Adv. Appl. Prob. **43,** 484–503 (2011)
Printed in Northern Ireland

EXACT MONTE CARLO SIMULATION FOR FORK-JOIN NETWORKS

HONGSHENG DAI,* *University of Brighton*

Abstract

In a fork-join network each incoming job is split into K tasks and the K tasks are simultaneously assigned to K parallel service stations for processing. For the distributions of response times and queue lengths of fork-join networks, no explicit formulae are available. Existing methods provide only analytic approximations for the response time and the queue length distributions. The accuracy of such approximations may be difficult to justify for some complicated fork-join networks. In this paper we propose a perfect simulation method based on *coupling from the past* to generate exact realisations from the equilibrium of fork-join networks. Using the simulated realisations, Monte Carlo estimates for the distributions of response times and queue lengths of fork-join networks are obtained. Comparisons of Monte Carlo estimates and theoretical approximations are also provided. The efficiency of the sampling algorithm is shown theoretically and via simulation.

Keywords: Coupling from the past; fork-join network; perfect sampling; read-once coupling from the past

2010 Mathematics Subject Classification: Primary 65C05; 65C50
Secondary 60K20; 60K25

1. Introduction

In a fork-join network jobs arrive according to a Poisson process with rate λ and each incoming job is split into K tasks at the fork station. Then the K tasks are simultaneously assigned to K parallel service stations for processing. The ith service station has s_i servers. Each server has exponential service times with rate μ_i. When all the K tasks of a job are completed, they will be joined immediately at the join station and leave the system.

Such fork-join networks are commonly used to model computer and manufacturing systems. For example, each logical computer I/O request is split into several physical I/O requests across the disk device and after all physical I/O requests are completed they are joined. Another example is that of manufacturing the assembly of a product, where the system requires several parts to be processed simultaneously at parallel work stations.

A fork-join queueing model is shown in Figure 1, where each service station has finite capacity N (the number of task-waiting places), i.e. the ith service station can hold at most $N + s_i$ tasks (including tasks in service and tasks in the queue). Other fork-join networks may have infinite capacity ($N = \infty$) or be such that the servers in each service station have different service rates. For various types of fork-join network, our main interests are the probability distributions of the response time (the time to complete a job) and the queue length (the number of jobs in the network) in equilibrium. The difficulty is that the stationary distribution of the

Received 30 April 2010; revision received 9 December 2010.
* Postal address: Department of Mathematics, School of Computing, Engineering and Mathematics, University of Brighton, Lewes Road, Brighton BN2 4GJ, UK. Email address: h.dai@brighton.ac.uk

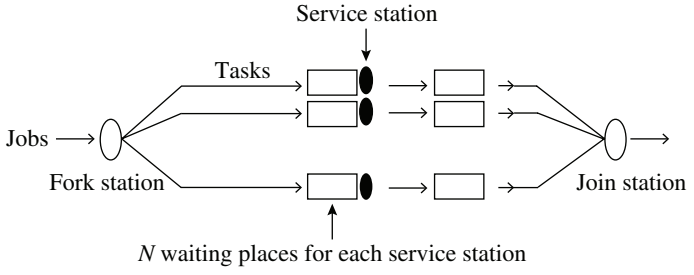

FIGURE 1: A fork-join network.

fork-join network, even if the network is as simple as that given in Figure 1, is intractable (see [9]).

Various methods for approximating or bounding the mean response times and approximating the queue length distributions for fork-join networks have been developed. Flatto and Hahn [6] and Flatto [5] derived the generating functions of the queue length distribution for a fork-join network having two service stations, each with a single server. Baccelli *et al*. [2], Balsoma *et al*. [3], and Nelson and Tantawi [10] gave bounds for the mean response time of an M/M/1 fork-join network. Ko and Serfozo [9] established a closed-form formula for approximating the distributions of the response time and queue length for certain fork-join networks in equilibrium. Raghavan and Viswanadhan [12] and Xia *et al*. [16] presented bounds for the mean response time of several types of fork-join network. Squillante *et al*. [14] considered a more general fork-join queueing system in which dynamic polices are considered for scheduling multiple tasks to maintain effective server utilization. However, most existing methods focus on analytical approximations for the equilibrium of fork-join networks. Simulation results in [9] demonstrated that the accuracy of such approximations may decrease as $s_i\mu_i$ decreases to λ.

In this paper we consider the use of Monte Carlo simulations to estimate the distributions of the response time and queue length when N is finite. This work is important since existing approximation methods are valid for only $N = \infty$. The accuracy of the Monte Carlo estimates depend on the number of simulated realisations; the more simulated realisations, the more accurate the Monte Carlo estimate. Therefore, such Monte Carlo estimates are important for fork-join networks with $\lambda \approx s_i\mu_i$ where existing analytic approximations may not be good enough. Another advantage of the Monte Carlo method is that we can obtain queue length distribution estimates for fork-join networks with multiple servers in each service station. However, analytic approximations of the queue length distribution for such fork-join networks are not available using existing methods, such as the method in [9]. In addition, we can also use the Monte Carlo distribution estimate to justify the accuracy of existing analytical approximations for fork-join networks. Finally, most existing methods focus on analytic approximations of mean response times; however, the response time distribution characteristics, such as the median and quartiles, are sometimes more important, as, for example, in our simulation scenarios where the response time distribution is highly skewed.

Traditional Monte Carlo simulation methods run the fork-join network as a continuous-time Markov chain starting from a given state. Then realisations are collected after the burn-in stage. Such methods only generate random realisations approximately from the equilibrium of the network. Thus, the estimates highly depend on the quality of the random realisations. In this paper we propose a perfect simulation method based on *coupling from the past* (CFTP),

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

introduced in [11], to generate exact realisations from the equilibrium of fork-join networks given in Figure 1. The CFTP algorithm developed in this paper is very efficient (a polynomial-time algorithm), i.e. the mean coalescence time of the algorithm is bounded by a polynomial function of KN. Based on the simulated realisations we provide Monte Carlo estimates for the distributions of the response time and queue length. Comparisons of the Monte Carlo estimates based on perfect sampling and approximation results in [9] demonstrate that the proposed perfect sampling method works very well.

This paper is organized as follows. In Section 2 we introduce the model notation and an algorithm to simulate a fork-join network from any given starting state. Then we develop a CFTP algorithm with bounding chains in Section 3 to simulate exactly from the equilibrium of the fork-join network. Simulation studies and estimates of the distributions of the queue length and response times are provided in Section 4. Complexity of the algorithm is discussed in Section 5. Section 6 contains a discussion.

2. Preliminaries and continuous-time Markov chain simulation

2.1. Notation

Consider the fork-join network shown in Figure 1. Jobs arrive at the system according to a Poisson process with rate λ and each incoming job is split into K tasks at the fork station. Then the K tasks are simultaneously assigned to K parallel service stations for processing. The ith station has s_i servers, each with exponential service times with mean $1/\mu_i$. When all the K tasks of a job are completed, they are joined immediately at the join station (the service time at the join station is 0) and leave the system. Service station i can hold $N_i = N + s_i$ tasks at most. Define $\boldsymbol{N} = (N_1, \ldots, N_K)$.

If the tasks of a job are not all completed, the completed tasks will wait in the corresponding buffer of the join station. We also assume that the join station has enough places to hold all completed waiting tasks.

Let $Q_i(t)$ be the number of tasks at service station i (including tasks in service and tasks waiting) at time t. We know that $Q_i(t)$ can take a value in $\{0, 1, \ldots, N_i + s_i\}$. The fork-join network can be represented by $\boldsymbol{Q}(t) = (Q_1(t), \ldots, Q_K(t))$, which is a K-dimensional continuous-time Markov chain.

From the results in [9] we know that when $N = \infty$, the transition rate for the K-dimensional continuous-time Markov chain $\boldsymbol{Q}(t)$ from a state $\boldsymbol{n} = (n_1, \ldots, n_K)$ to another state $\boldsymbol{n}'$ is given by

$$q(\boldsymbol{n}, \boldsymbol{n}') = \begin{cases} \lambda & \text{if } \boldsymbol{n}' = \boldsymbol{n} + \boldsymbol{1}, \\ \mu_i \min\{n_i, s_i\} & \text{if } \boldsymbol{n}' = \boldsymbol{n} - \boldsymbol{e}_i, \\ 0 & \text{otherwise.} \end{cases}$$

Here $\boldsymbol{e}_i$ is a K-dimensional vector with 1 in the ith component and 0s elsewhere, and $\boldsymbol{1}$ is a K-dimensional vector of all 1s. The Markov chain with $N = \infty$ is ergodic when $\lambda < \mu_i s_i$ for all values of i, $1 \le i \le K$.

Here we consider N to be a finite number, which means that each service station can only hold up to a finite number of tasks. This is reasonable in practice. If service station i holds the maximum number of tasks, $N + s_i$, the newly arrived job will not be split into K tasks since the buffer of service station i is full. Therefore, when N is finite, the transition rate for $\boldsymbol{Q}(t)$ is

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

given by

$$q(\boldsymbol{n}, \boldsymbol{n}') = \begin{cases} \lambda & \text{if } \boldsymbol{n}' = \boldsymbol{n} + \mathbf{1} \text{ and } \boldsymbol{n} + \mathbf{1} \le \boldsymbol{N}, \\ \mu_i \min\{n_i, s_i\} & \text{if } \boldsymbol{n}' = \boldsymbol{n} - \boldsymbol{e}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

2.2. Continuous-time Markov chain simulation

We now introduce an algorithm to simulate the Markov chain $\boldsymbol{Q}(t)$ with a given starting state. This algorithm will be used in the next section to develop a CFTP algorithm.

Using the continuous-time Markov chain theory, we know that the amount of time that $\boldsymbol{Q}(t)$ spends in the current state $\boldsymbol{n}$ (holding time of a state) is exponentially distributed with rate

$$r_{\boldsymbol{n}} := \begin{cases} \displaystyle\sum_{\boldsymbol{n}' \neq \boldsymbol{n}} q(\boldsymbol{n}, \boldsymbol{n}') = \lambda + \sum_i \mu_i \min\{n_i, s_i\} & \text{if } \boldsymbol{n} + \mathbf{1} \le \boldsymbol{N}, \\ \displaystyle\sum_{\boldsymbol{n}' \neq \boldsymbol{n}} q(\boldsymbol{n}, \boldsymbol{n}') = \sum_i \mu_i \min\{n_i, s_i\} & \text{if } \boldsymbol{n} + \mathbf{1} \not\le \boldsymbol{N}. \end{cases}$$

Then at the end of the holding time, $\boldsymbol{Q}(t)$ jumps to another state $\boldsymbol{n}'$ with transition probability

$$p_{\boldsymbol{n},\boldsymbol{n}'} = \begin{cases} \dfrac{\lambda}{r_{\boldsymbol{n}}} & \text{if } \boldsymbol{n}' = \boldsymbol{n} + \mathbf{1} \text{ and } \boldsymbol{n} + \mathbf{1} \le \boldsymbol{N}, \\ \dfrac{\mu_i \min\{n_i, s_i\}}{r_{\boldsymbol{n}}} & \text{if } \boldsymbol{n}' = \boldsymbol{n} - \boldsymbol{e}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Note that the holding times of each state and the transitions of $\boldsymbol{Q}(t)$ are independent.

We may simulate the continuous-time Markov chain using the following method. Suppose that $\boldsymbol{Q}(t) = \boldsymbol{n}$. Simulate the next transition time $t' = T + t$, where T is drawn from an exponential distribution with rate $r_{\boldsymbol{n}}$. This can be done by simulating $\xi_0 \sim \exp(\lambda)$ and $\xi_{ij} \sim \exp(\mu_i),\ j = 1, \ldots, s_i,\ i = 1, \ldots, K$, and setting

$$T = \min\{\xi_0, \xi_{ij},\ j = 1, \ldots, \min\{n_i, s_i\},\ i = 1, \ldots, K\}$$

if $\boldsymbol{n} + \mathbf{1} \le \boldsymbol{N}$ or setting $T = \min\{\xi_{ij},\ j = 1, \ldots, \min\{n_i, s_i\},\ i = 1, \ldots, K\}$ if $\boldsymbol{n} + \mathbf{1} \not\le \boldsymbol{N}$. This is because such T is from $\exp(r_{\boldsymbol{n}})$. Note that if $n_i = 0$ then define $\min_{j \le \min\{n_i, s_i\}} \xi_{ij} = \infty$.

At time t' the process $\boldsymbol{Q}$ will jump to $\boldsymbol{n}'$ with probability $p_{\boldsymbol{n},\boldsymbol{n}'}$. The transitions of the chain can be simulated as follows. Provided that $\boldsymbol{n} + \mathbf{1} \le \boldsymbol{N}$, if $T = \xi_0$ then let $\boldsymbol{n}' = \boldsymbol{n} + \mathbf{1}$; if $T = \min_{j \le \min\{n_i, s_i\}} \xi_{ij}$ for some i then let $\boldsymbol{n}' = \boldsymbol{n} - \boldsymbol{e}_i$. This is because $\mathrm{P}(T = \min_{j \le \min\{n_i, s_i\}} \xi_{ij}$ for some $i) = \mu_i \min\{n_i, s_i\}/r_{\boldsymbol{n}}$ and $\mathrm{P}(T = \xi_0) = \lambda/r_{\boldsymbol{n}}$, which correspond to the transition probabilities in (2). Similarly, given $\boldsymbol{n} + \mathbf{1} \not\le \boldsymbol{N}$, if $T = \min_{j \le \min\{n_i, s_i\}} \xi_{ij}$ for some i then let $\boldsymbol{n}' = \boldsymbol{n} - \boldsymbol{e}_i$. On the other hand, with such a simulation method, the transitions are independent of the holding times. So the Markov chain $\boldsymbol{Q}(t)$ is correctly simulated.

The above simulation method can be proved using the properties of exponential variables. Details can be found in [13, Chapter 5]. Although this simulation idea is easy to implement, in order to develop a CFTP method later, we use the following more complicated algorithm instead to simulate $\{\boldsymbol{Q}(t),\ t \le \mathcal{T}\}$. The algorithm outputs a sequence of times $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots\}$ which is a superset of the transition times $\boldsymbol{t} = \{t_1, t_2, \ldots\}$ of $\boldsymbol{Q}(t)$. A thinning method is used to identify the transition times $\boldsymbol{t}$ from $\boldsymbol{\tau}$.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

Algorithm 1. (*Markov chain simulation algorithm.*) Outputs: τ_l, $\boldsymbol{Q}(\tau_l)$, and t_k.

01. Set $t_0 = \tau_0 = 0, k, l = 0$, and $\boldsymbol{Q}(t_0) =$ starting state.

02. **While** $(t_k < \mathcal{T})$

03. Simulate $\xi_0^{(l)} \sim \exp(\lambda)$ and $\xi_{ij}^{(l)} \sim \exp(\mu_i)$ for $j = 1, \ldots, s_i$ and $i = 1, \ldots, K$.

Note that we use $\mathcal{U}^{(l)}$ to denote all simulated $\xi^{(l)}$s.

04. MC(τ_l, $\boldsymbol{Q}(\tau_l)$, $\mathcal{U}^{(l)}$; τ_{l+1}, $\boldsymbol{Q}(\tau_{l+1})$, I) (given in Algorithm 2 below).

05. **If** $I = 1$ **then** $t_{k+1} = \tau_{l+1}$ (the next transition time) and $k = k + 1$.

06. $l = l + 1$.

Algorithm 2. (*Markov chain updating algorithm.*) This subroutine is given by MC(τ_l, $\boldsymbol{Q}(\tau_l)$, $\mathcal{U}^{(l)}$; τ_{l+1}, $\boldsymbol{Q}(\tau_{l+1})$, I).
Inputs: τ_l, $\boldsymbol{Q}(\tau_l)$, and $\mathcal{U}^{(l)}$. Outputs: τ_{l+1}, $\boldsymbol{Q}(\tau_{l+1})$, and I.

For simplicity, denote $\boldsymbol{Q}(\tau_l)$ as $\boldsymbol{n}^{(l)} = (n_1^{(l)}, \ldots, n_K^{(l)})$. Let $T_l^{\min} = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)}, j = 1, \ldots, s_i, i = 1, \ldots, K\}$ and $T_l = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)}, j = 1, \ldots, \min\{n_i^{(l)}, s_i\}, i = 1, \ldots, K\}$.

01. $\tau_{l+1} = \tau_l + T_l^{\min}$

02. **If** $T_l = T_l^{\min}$ **then**

03. **if** $T_l = \xi_0^{(l)}$ **then**

04. $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{Q}(\tau_l) + \mathbf{1}$ and $I = 1$

05. **if** $\boldsymbol{Q}(\tau_{l+1}) \not\leq \boldsymbol{N}$ **then** $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{Q}(\tau_l)$ and $I = 0$.

06. **else if** $T_l = \xi_{ij}^{(l)}$ for some i **then**

07. $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{Q}(\tau_l) - \boldsymbol{e}_i$ and $I = 1$

08. **end if**

09. **else**

10. $I = 0$ and $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{Q}(\tau_l)$

11. **end if**

Lemma 1. *Algorithm 1 outputs* $\boldsymbol{\tau} = \{\tau_0, \tau_1, \ldots, \tau_l, \ldots\}$ *and* $\{\boldsymbol{Q}(\tau_l), l = 0, 1, \ldots\}$. *The algorithm also outputs a subsequence of* $\boldsymbol{\tau}$, *as* $\{t_0, t_1, \ldots, t_k, \ldots\}$.

The simulated continuous-time Markov process is given by $\{t_k, \boldsymbol{Q}(t_k)\}$, $k = 0, 1, \ldots$. *The time* t_k *is the kth transition time of* $\boldsymbol{Q}(t)$ *and the process* $\boldsymbol{Q}(t)$ *is constant in* $[t_k, t_{k+1})$. *The simulated process has transition rates given in (1).*

Lemma 1 is true since Algorithm 1 simulates potential holding times $T_l^{\min}$ of $\boldsymbol{Q}(t)$ from an exponential distribution with rate parameter $\lambda + \sum_i \mu_i s_i$, which is larger than the rates of exponential holding times of all states. Then a thinning method is used to check whether time τ_l is a transition time point of $\boldsymbol{Q}(t)$. See Appendix A for a detailed proof.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

Note that the updating rules in Algorithm 2 can be denoted as a deterministic function ϕ as

$$\boldsymbol{Q}(\tau_l + a) = \phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a)$$
$$= \begin{cases} \boldsymbol{Q}(\tau_l) & \text{if } 0 < a < T_l^{\min}, \\ \boldsymbol{Q}(\tau_l) + \mathbf{1} & \text{if } a = T_l^{\min},\ \boldsymbol{Q}(\tau_l) + \mathbf{1} \le \boldsymbol{N},\ \text{and } T_l^{\min} = T_l = \xi_0^{(l)} \\ & \text{(line 04 of Algorithm 2)}, \\ \boldsymbol{Q}(\tau_l) & \text{if } a = T_l^{\min},\ \boldsymbol{Q}(\tau_l) + \mathbf{1} \not\le \boldsymbol{N},\ \text{and } T_l^{\min} = T_l = \xi_0^{(l)} \\ & \text{(line 05 of Algorithm 2)}, \\ \boldsymbol{Q}(\tau_l) - \boldsymbol{e}_i & \text{if } a = T_l^{\min} \text{ and } T_l^{\min} = T_l = \xi_{ij}^{(l)} \text{ for some } i \\ & \text{(line 07 of Algorithm 2)}, \\ \boldsymbol{Q}(\tau_l) & \text{if } a = T_l^{\min} \text{ and } T_l^{\min} < T_l \text{ (line 10 of Algorithm 2)}, \end{cases} \tag{3}$$

where $\mathcal{U}^{(l)}$ is defined in Algorithm 1.

3. Simulating from equilibrium using CFTP with bounding chains

CFTP was introduced in the landmark paper by Propp and Wilson [11], who showed how to provide perfect samples from the limiting distribution of a Markov chain. The idea is to run Markov chains starting from all states simultaneously from the past, using the same random numbers in the updating procedure, until all chains coalesce into a single chain, then keep running the coalesced chain and collect a sample at time 0.

By running all Markov chains simultaneously, CFTP may result in very heavy computational costs. The fork-join network can be described by the K-dimensional continuous-time Markov chain $\boldsymbol{Q}(t)$ with state space $\mathcal{Q}$. The number of states in $\mathcal{Q}$ is larger than K^N. It is impossible to run so many chains simultaneously. The computational cost of the algorithm can be reduced if, for the state space $\mathcal{Q}$, there is a partial order preserved by the process $\boldsymbol{Q}(t)$. This is called monotone CFTP for which we only need to run the upper chain starting from the maximum state and the lower chain starting from the minimum state (see [11] for more details). If the upper and lower chains, which bound all the other chains, coalesce then all chains coalesce. Thus, the efficiency of the CFTP algorithm can be improved significantly.

Although monotone CFTP is easy to perform, finding a partial order preserved by the Markov chains is a nontrivial task in many cases. An alternative improvement is CFTP with bounding chains or dominating processes; see [4], [7], and [8]. The advantages of bounding-chain CFTP are that the partial order preserved by the Markov chains is not required and that only several bounding chains, which bound all Markov chains, are required to run simultaneously.

In Section 3.1 we first define a partial order for $\mathcal{Q}$. Then we show that the partial order is preserved by $\boldsymbol{Q}(t)$ in some cases, but not always. Then based on this result we provide a CFTP algorithm using bounding chains in Sections 3.2 and 3.3.

3.1. A partial order

For $\boldsymbol{n}, \boldsymbol{n}' \in \mathcal{Q}$, define $\boldsymbol{n} \prec \boldsymbol{n}'$ if $n_j \le n'_j$, $j = 1, \ldots, K$, and $n_j < n'_j$ for at least one value of j. Define $\boldsymbol{n} = \boldsymbol{n}'$ if $n_j = n'_j$ for $j = 1, \ldots, K$.

The partial order is preserved by $\boldsymbol{Q}(t)$ meaning that if $\boldsymbol{Q}(\tau_l) \preceq \boldsymbol{Q}'(\tau_l)$ then $\phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) \preceq \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a)$ for all a such that $0 < a \le T_l^{\min}$. The following lemma gives the cases where the partial order is preserved and the cases where the partial order is not preserved.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

Lemma 2. *Suppose that the current time is τ_l and that $\boldsymbol{Q}(\tau_l) = \boldsymbol{n} \prec \boldsymbol{Q}'(\tau_l) = \boldsymbol{n}'$. Define*

$$T_l = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)},\ j = 1, \ldots, \min\{n_i, s_i\},\ i = 1, \ldots, K\}$$

and

$$T_l' = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)},\ j = 1, \ldots, \min\{n_i', s_i\},\ i = 1, \ldots, K\}.$$

Then we have the following results.

(a) *For all a such that $0 < a < T_l^{\min}$, we have*

$$\boldsymbol{Q}(\tau_l + a) = \phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) \preceq \boldsymbol{Q}'(\tau_l + a) = \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a).$$

Define the events $\mathbb{A}^{(l)} = \{T_l^{\min} \neq \xi_0^{(l)}\}$, $\mathbb{B}_1^{(l)} = \{(\boldsymbol{n}, \boldsymbol{n}') \colon \boldsymbol{n}' + \mathbf{1} \leq \boldsymbol{N}\}$, $\mathbb{B}_2^{(l)} = \{(\boldsymbol{n}, \boldsymbol{n}') \colon \boldsymbol{n} + \mathbf{1} \not\leq \boldsymbol{N}\}$, and $\mathbb{B}_3^{(l)} = (B_1^{(l)} \cup B_2^{(l)})^{\mathrm{c}}$. Obviously, $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l))$ must belong to $\mathbb{B}_1^{(l)} \cup \mathbb{B}_2^{(l)} \cup \mathbb{B}_3^{(l)}$.

For $a = T_l^{\min}$, the relation between $\boldsymbol{Q}(\tau_l + a)$ and $\boldsymbol{Q}'(\tau_l + a)$ must be such that one of the following cases holds.

(b) *If $T_l^{\min} \neq \xi_0^{(l)}$, i.e. $\mathbb{A}^{(l)}$ is true, we have*

$$\phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) \preceq \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a).$$

(c) *If $T_l^{\min} = \xi_0^{(l)}$, i.e. $\mathbb{A}^{(l)}$ is not true, and $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l)) \in \mathbb{B}_1^{(l)}$, then*

$$\phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) \preceq \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a).$$

(d) *If $\mathbb{A}^{(l)}$ is not true and $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l)) \in \mathbb{B}_2^{(l)}$, then*

$$\phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) \preceq \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a).$$

(e) *If $\mathbb{A}^{(l)}$ is note true and $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l)) \in \mathbb{B}_3^{(l)}$, then*

$$\phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) \preceq \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a)$$

is not guaranteed.

Proof. See Appendix B.

Note that result Lemma 2(a) implies that the partial order is always preserved by the process in (τ_l, τ_{l+1}). If $a = T_l^{\min}$ then $\tau_l + a = \tau_{l+1}$. Thus, Lemma 2(b) means that the partial order is preserved if $\mathbb{A}^{(l)}$ is true. Similarly, at time τ_{l+1} the partial order is also preserved under the conditions of Lemma 2(c) and (d). But the partial order may not be preserved at τ_{l+1} under the conditions of Lemma 2(e). Therefore, we cannot simply use a monotone CFTP algorithm, but it is possible to use a bounding-chain algorithm to deal with this problem.

3.2. Bounding chains

We run Markov chains starting from different states from time 0. Suppose that $\underline{\boldsymbol{Q}}(0) \preceq \overline{\boldsymbol{Q}}(0)$ bounds all other Markov chains. We construct two bounding chains $\underline{\boldsymbol{Q}}(t)$ and $\overline{\boldsymbol{Q}}(t)$ as follows.

Suppose that the current time is τ_l. After simulating random numbers $\mathcal{U}^{(l)}$, we update $\underline{\boldsymbol{Q}}(t)$ and $\overline{\boldsymbol{Q}}(t)$ in (τ_l, τ_{l+1}) according to Algorithm 2. We know that $\underline{\boldsymbol{Q}}(t)$ and $\overline{\boldsymbol{Q}}(t)$ will bound all Markov chains in (τ_l, τ_{l+1}) since, according to Lemma 2(a), the partial order is preserved.

At time τ_{l+1}, if one of the conditions in Lemma 2(b), (c), or (d) is satisfied, we also update $\underline{\boldsymbol{Q}}(t)$ and $\overline{\boldsymbol{Q}}(t)$ at τ_{l+1} according to Algorithm 2. We know that $\underline{\boldsymbol{Q}}(\tau_{l+1})$ and $\overline{\boldsymbol{Q}}(\tau_{l+1})$ will

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

bound all Markov chains $\boldsymbol{Q}(\tau_{l+1})$ since the partial order is preserved according to Lemma 2(b), (c), and (d).

If the condition in Lemma 2(e) is true, we let

$$\begin{aligned}\underline{\boldsymbol{Q}}(\tau_{l+1}) &= \underline{\boldsymbol{Q}}(\tau_l),\\ \overline{\boldsymbol{Q}}(\tau_{l+1}) &= (\min\{\overline{Q}_1(\tau_l)+1, N_1\}, \ldots, \min\{\overline{Q}_K(\tau_l)+1, N_K\}).\end{aligned} \tag{4}$$

Lemma 3. *The above bounding chains will bound all Markov chains, i.e.* $\underline{\boldsymbol{Q}}(t) \preceq \boldsymbol{Q}(t) \preceq \overline{\boldsymbol{Q}}(t)$ *for all* t *and any Markov chains* $\boldsymbol{Q}(t)$.

Proof. We only need to prove that if the condition in Lemma 2(e) is true then $\underline{\boldsymbol{Q}}(\tau_l) \preceq \boldsymbol{Q}(\tau_l) \preceq \overline{\boldsymbol{Q}}(\tau_l)$ implies that $\underline{\boldsymbol{Q}}(\tau_{l+1}) \preceq \boldsymbol{Q}(\tau_{l+1}) \preceq \overline{\boldsymbol{Q}}(\tau_{l+1})$.

If the condition in Lemma 2(e) is true, we know that $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{Q}(\tau_l) + \mathbf{1}$ or $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{Q}(\tau_l)$. Obviously, given $\underline{\boldsymbol{Q}}(\tau_l) \preceq \boldsymbol{Q}(\tau_l) \preceq \overline{\boldsymbol{Q}}(\tau_l)$, we have

$$\underline{\boldsymbol{Q}}(\tau_l) \preceq \boldsymbol{Q}(\tau_{l+1}) \quad \text{and} \quad \boldsymbol{Q}(\tau_{l+1}) \preceq (\min\{\overline{Q}_1(\tau_l)+1, N_1\}, \ldots, \min\{\overline{Q}_K(\tau_l)+1, N_K\}).$$

So (4) guarantees that $\underline{\boldsymbol{Q}}(\tau_{l+1}) \preceq \boldsymbol{Q}(\tau_{l+1}) \preceq \overline{\boldsymbol{Q}}(\tau_{l+1})$. This completes the proof.

When $\overline{\boldsymbol{Q}}(t)$ and $\underline{\boldsymbol{Q}}(t)$ coalesce, all Markov chains coalesce. Based on this, a CFTP algorithm is given in the next subsection.

3.3. CFTP with bounding chains

Define an upper-bound chain starting from $\boldsymbol{N}$ and a lower-bound chain starting from $\mathbf{0} = (0, \ldots, 0)$. When the upper-bound chain and lower-bound chain coalesce, all Markov chains coalesce. When using CFTP, we need to re-use old random numbers. A simple variation on CFTP is the so called read-once CFTP proposed in [15], which outputs random realisations from equilibrium using just a read-once source of random numbers. Here we use the read-once CFTP algorithm, as Wilson [15] suggested that read-once CFTP has the advantage if many independent realisations are desired. According to Wilson [15], the following read-once CFTP algorithm returns realisations from the equilibrium of $\boldsymbol{Q}(t)$.

Algorithm 3. (*Read-once CFTP (NumberofRealisations).*)

```
01. State := (0, 0, ..., 0) (it can be an arbitrary state)
02. Repeat
03.     ApplyCompositeMap(State,CoalescenceFlag) (given in Algorithm 4 below)
04. Until CoalescenceFlag
05. For i = 1 to NumberofRealisations
06.     Repeat
07.         OldState = State
08.         ApplyCompositeMap(State,CoalescenceFlag)
09.     Until CoalescenceFlag
10.     Output OldState
```

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

Algorithm 4. (*ApplyCompositeMap(State,CoalescenceFlag).*) Inputs: State and CoalescenceFlag. Outputs: State and CoalescenceFlag.

01. $\overline{\boldsymbol{n}}_1 = \boldsymbol{N}, \underline{\boldsymbol{n}}_1 = \boldsymbol{0}, \overline{\boldsymbol{n}}_2 = \boldsymbol{N}, \underline{\boldsymbol{n}}_2 = \boldsymbol{0}, l = 0, \tau_l = 0$, and $\tau'_l = 0$.
02. **while** $\overline{\boldsymbol{n}}_2 \neq \underline{\boldsymbol{n}}_2$
03. Simulate $\mathcal{U}^{(l)} = \{\xi_0^{(l)} \sim \exp(\lambda),\ \xi_{ij}^{(l)} \sim \exp(\mu_i),\ j = 1, \ldots, s_i,\ i = 1, \ldots, K\}$.
04. MC(τ_l, State, $\mathcal{U}^{(l)}$; τ_{l+1}, State, I)
05. **If** $T_l^{\min} = \xi_0^{(l)}$ and $(\overline{\boldsymbol{n}}_1, \underline{\boldsymbol{n}}_1) \in \mathbb{B}_3^{(l)}$ ($\mathbb{B}_3^{(l)}$ is defined in Lemma 2) **then**
06. $\overline{\boldsymbol{n}}_1 = (\min\{\overline{n}_{1,1}+1, N_1\}, \ldots, \min\{\overline{n}_{1,K}+1, N_K\})$, $\underline{\boldsymbol{n}}_1 = \underline{\boldsymbol{n}}_1$, and $\tau_{l+1} = \tau_l + T_l^{\min}$
07. **else**
08. MC($\tau_l, \overline{\boldsymbol{n}}_1, \mathcal{U}^{(l)}$; $\tau_{l+1}, \overline{\boldsymbol{n}}_1, I$) and MC($\tau_l, \underline{\boldsymbol{n}}_1, \mathcal{U}^{(l)}$; $\tau_{l+1}, \underline{\boldsymbol{n}}_1, I$)
09. Simulate $\mathcal{U}^{(l)} = \{\xi_0^{(l)} \sim \exp(\lambda),\ \xi_{ij}^{(l)} \sim \exp(\mu_i),\ j = 1, \ldots, s_i,\ i = 1, \ldots, K\}$.
10. **If** $T_l^{\min} = \xi_0^{(l)}$ and $(\overline{\boldsymbol{n}}_2, \underline{\boldsymbol{n}}_2) \in \mathbb{B}_3^{(l)}$ **then**
11. $\overline{\boldsymbol{n}}_2 = (\min\{\overline{n}_{2,1}+1, N_1\}, \ldots, \min\{\overline{n}_{2,K}+1, N_K\})$, $\underline{\boldsymbol{n}}_2 = \underline{\boldsymbol{n}}_2$, and $\tau'_{l+1} = \tau'_l + T_l^{\min}$
12. **else**
13. MC($\tau'_l, \overline{\boldsymbol{n}}_2, \mathcal{U}^{(l)}$; $\tau'_{l+1}, \overline{\boldsymbol{n}}_2, I$) and MC($\tau'_l, \underline{\boldsymbol{n}}_2, \mathcal{U}^{(l)}$; $\tau'_{l+1}, \underline{\boldsymbol{n}}_2, I$)
14. $l = l + 1$
15. **If** $\overline{\boldsymbol{n}}_1 = \underline{\boldsymbol{n}}_1$ and $\tau_l \leq \tau'_l$ **then** CoalescenceFlag $= 1$
16. **Else then** CoalescenceFlag $= 0$

Theorem 1. *Algorithm 3 returns realisations from the equilibrium of* $\boldsymbol{Q}(t)$.

Theorem 1 follows from Lemma 3 and the fact that Algorithm 3 is the read-once CFTP algorithm in [15].

4. Simulation results

The number of jobs in the network at time t is $Q(t) = \max_{1 \leq i \leq K} Q_i(t)$. Let W_{mi} be the time from entering the system to exiting the service station of the ith task of the mth job. Then the total time the mth job spends in the network is $W_m = \max_{1 \leq i \leq K} W_{mi}$, which is the response time of the mth job. When $\boldsymbol{Q}(t)$ is at equilibrium, we denote task quantities at each service station by $\boldsymbol{Q} = (Q_1, \ldots, Q_K)$ and we define W to be the response time. We use the proposed perfect sampling methods to estimate probability distributions for $Q = \max_{1 \leq i \leq K} Q_i$ and the response time W.

In this section, all Monte Carlo simulation results are based on 100 000 realisations.

4.1. Simulation results for response times

Scenario 1. In Table 1 we present the Monte Carlo estimates of E W for the fork-join network with $K = 2$, $\lambda = 1$, $s_i = 1$, and different values of μ_i and N. From Table 1 we can see that the approximations, given in [9] with $N = \infty$, are very close to the Monte Carlo estimates using perfect simulation with a finite value of N. Obviously, the fork-join network with $N = \infty$, on average, has a longer queue at equilibrium than the fork-join network with a finite value of N.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

TABLE 1: Approximations, from [9], and estimates with $N = 300$ and $N = 100$. The estimates are based on the 100 000 realisations simulated using perfect sampling methods. The values in brackets are 100 times the standard errors of the estimates.

μ_1	μ_2	Approximation of E W with $N = \infty$	Estimates of E W with $N = 300$	Estimates of E W with $N = 100$
10.00	15.00	0.1384	0.137 945 (0.034)	0.137 299 (0.035)
10.00	20.00	0.1276	0.127 368 (0.033)	0.126 932 (0.033)
5.00	7.50	0.3057	0.303 399 (0.077)	0.301 999 (0.076)
5.00	10.00	0.2825	0.279 991 (0.076)	0.279 578 (0.074)
3.33	5.00	0.5138	0.509 327 (0.130)	0.503 866 (0.130)
3.33	6.67	0.4762	0.470 525 (0.130)	0.467 724 (0.130)
2.50	3.75	0.7822	0.766 465 (0.200)	0.760 212 (0.200)
2.50	5.00	0.7280	0.716 088 (0.200)	0.703 631 (0.200)
2.00	3.00	1.1458	1.124 944 (0.300)	1.103 505 (0.290)
2.00	4.00	1.0729	1.049 793 (0.300)	1.037 470 (0.300)

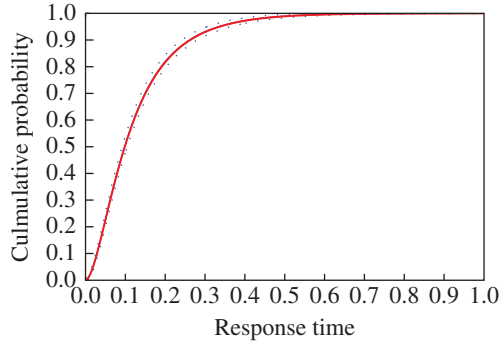

FIGURE 2: The response time distribution function with $K = 2, s_i = 1, \lambda = 1, \mu_1 = 10$, and $\mu_2 = 20$. The dotted lines are the Monte Carlo estimates ± 20 times the standard error. The solid line is the Monte Carlo estimate with $N = 300$ and the dashed line is the approximation in [9] with $N = \infty$.

Since a longer queue results in a longer response time, we expect that the fork-join network with $N = \infty$ has a slightly larger mean response time. The results in Table 1 confirm this: the mean response time decreases as N decreases.

We can also compare the Monte Carlo estimates and the approximation in [9] for $F(t) = \mathrm{P}(W \leq t)$. With $\mu_1 = 10$ and $\mu_2 = 20$, the results are shown in Figure 2, where the solid line (the Monte Carlo estimate) and the dashed line (the approximation given in [9]) are almost the same. If we zoom into the graph (see Figure 3), we can see that the approximation is slightly lower. With $\mu_1 = 2$ and $\mu_2 = 3$, the results are shown in Figures 4 and 5. We can see that the difference between the Monte Carlo estimate and the theoretical approximations becomes larger. Note that the simulated sample size 100 000 is very large and the 95% confidence interval will be almost the same as the estimate. Thus, in these graphs we output the Monte Carlo estimate ± 20 times its standard error (dotted lines).

Scenario 2. In Table 2 we present the Monte Carlo estimates of E W for the fork-join network with $K = 2, \lambda = 1, s_1 = 2, s_2 = 3$, and different values of μ_i and N. From the results we

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

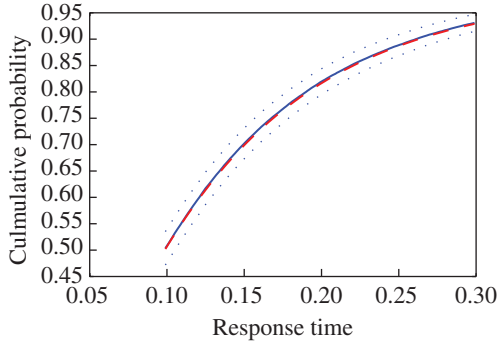

FIGURE 3: A zoomed-in view of the response time distribution function of Figure 2, with $K = 2$, $s_i = 1$, $\lambda = 1$, $\mu_1 = 10$, and $\mu_2 = 20$. The dotted lines are the Monte Carlo estimates ± 20 times the standard error. The solid line is the Monte Carlo estimate with $N = 300$ and the dashed line is the approximation in [9] with $N = \infty$.

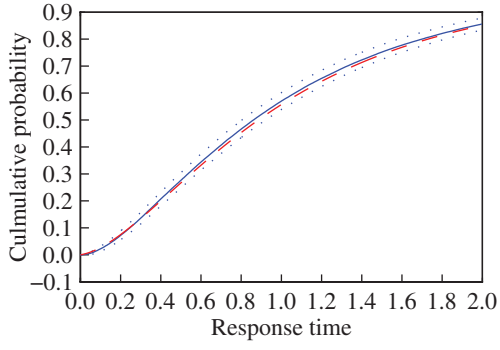

FIGURE 4: The response time distribution function with $K = 2$, $s_i = 1$, $\lambda = 1$, $\mu_1 = 2$, and $\mu_2 = 3$. The dotted lines are the Monte Carlo estimates ± 20 times the standard error. The solid line is the Monte Carlo estimate with $N = 300$ and the dashed line is the approximation in [9] with $N = \infty$.

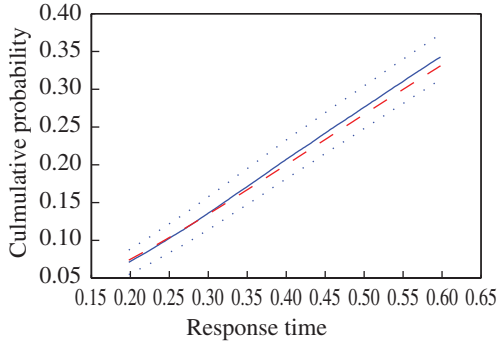

FIGURE 5: A zoomed-in view of the response time distribution function of Figure 4, with $K = 2$, $s_i = 1$, $\lambda = 1$, $\mu_1 = 2$, and $\mu_2 = 3$. The dotted lines are the Monte Carlo estimates ± 20 times standard error. The solid line is the Monte Carlo estimate with $N = 300$ and the dashed line is the approximation in [9] with $N = \infty$.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

TABLE 2: Approximations, from [9], and estimates with $N = 300$ and $N = 100$. The estimates are based on the 100 000 realisations simulated using perfect sampling methods. The values in brackets are 100 times the standard errors of the estimates.

μ_1	μ_2	Approximation of E W with $N = \infty$	Estimates of E W with $N = 300$	Estimates of E W with $N = 100$
5.00	5.00	0.2989	0.302 923 (0.071)	0.300 916 (0.071)
5.00	10.00	0.2342	0.234 578 (0.060)	0.234 795 (0.061)
5.00	15.00	0.2181	0.218 539 (0.061)	0.218 199 (0.061)
1.00	1.00	1.7208	1.756 273 (0.400)	1.746 010 (0.400)
1.00	2.00	1.4477	1.443 872 (0.370)	1.437 890 (0.370)
1.00	3.00	1.3822	1.375 673 (0.380)	1.369 130 (0.380)
0.56	0.56	9.7814	8.624 601 (2.400)	8.122 390 (2.250)
0.56	1.11	9.5285	8.310 150 (2.450)	7.807 320 (2.300)
0.56	1.67	9.4974	8.266 610 (2.470)	7.811 830 (2.290)

can see that the difference between the Monte Carlo estimate with a finite value of N and the theoretical approximation with $N = \infty$ increases as $s_i\mu_i/\lambda$ decreases. When $s_i\mu_i/\lambda$ is large (for example, $\lambda = 1$ and $\mu_i = 5$), the queue length in equilibrium is small, and $N = 300$ and $N = 100$ are large enough compared to the queue length. New jobs will not be rejected since N is very large, which is the same as the fork-join network with $N = \infty$. Therefore, in such situations, the mean response time with $N = \infty$ is almost equal to the mean response time with a large finite value of N. On the other hand, when $s_i\mu_i/\lambda$ is close to (but larger than) 1 (for example, $\lambda = 1$ and $\mu_i = 0.56$), the queue length in equilibrium may be very large, and $N = 300$ and $N = 100$ are not large enough. In such situations, new jobs may be rejected due to limited waiting space. The queue length in equilibrium with $N = \infty$ will be randomly larger than that with a finite value of N since with $N = \infty$ new jobs will never be rejected. Therefore, the mean response time with $N = \infty$ is larger than the mean response time with a finite value of N.

We plot the estimates for $F(t) = \mathrm{P}(W \le t)$ with $\mu_1 = 0.56$, $\mu_2 = 0.56$ and $\mu_1 = 5$, $\mu_2 = 15$ in Figures 6 and 7, respectively. We can see that the mean response times decreases as $s_i\mu_i/\lambda$

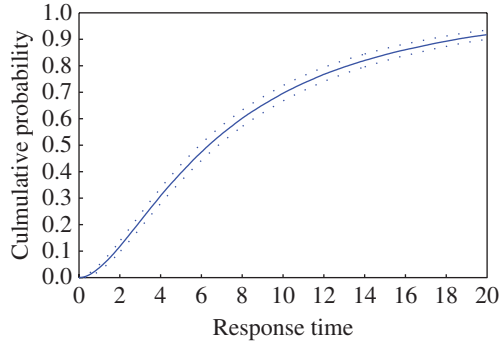

FIGURE 6: The response time distribution function with $K = 2$, $s_1 = 2$, $s_2 = 3$, $\lambda = 1$, $\mu_1 = 0.56$, and $\mu_2 = 0.56$. The dotted lines are the Monte Carlo estimates ± 20 times the standard error.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

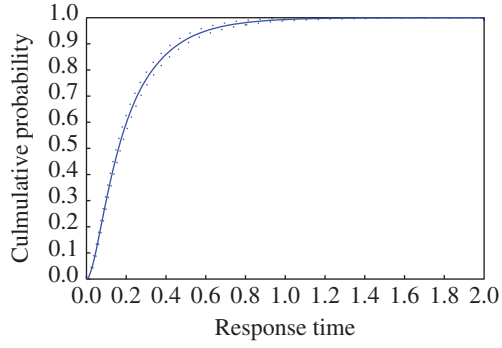

FIGURE 7: The response time distribution function with $K = 2, s_1 = 2, s_2 = 3, \lambda = 1, \mu_1 = 5$, and $\mu_2 = 15$. The dotted lines are the Monte Carlo estimates ± 20 times the standard error.

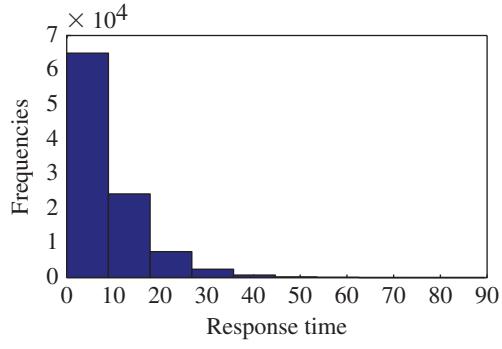

FIGURE 8: Histogram of response times, with $N = 300$, $K = 2, s_1 = 2, s_2 = 3, \lambda = 1, \mu_1 = 0.56$, and $\mu_2 = 0.56$.

increases. Under this scenario, the distribution approximation for W is not readily available by using the methods in [9].

From Figure 8 we can see that the distribution of response times is highly skewed. Therefore, the other characteristics of the response time distribution, such as the median and quartiles, will be important. Existing methods cannot give approximations for these characteristics for multi-server fork-join networks, but we can easily estimate the median or quartiles of the response time using the simulated realisations. For example, under simulation Scenario 1 with $\mu_1 = 2$, $\mu_2 = 4$, and $N = 300$, the 0.5% and 99.5% quartiles for W are 0.000 963 and 12.065 317, respectively ($\mathrm{E}\, W = 1.049\,793$); under simulation Scenario 2 with $\mu_1 = \mu_2 = 0.56$, and $N = 300$, the 0.5% and 99.5% quartiles for W are 0.004 539 and 89.327 042, respectively ($\mathrm{E}\, W = 8.624\,601$). For simplicity, the results with other values of μ_i are not shown here.

4.2. Simulation for the queue length

Now we consider the simulation for the probability distribution of Q under the following three scenarios.

Scenario 3. In Table 3 we present the Monte Carlo estimates for the probability distribution of Q for the fork-join network with $K = 2, \lambda = 1, s_i = 1$, and $\mu_1 = \mu_2 = 2$.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

TABLE 3: Estimation of $P(Q = K)$. Scenario I gives the theoretical approximation from [9], and scenarios II and III give Monte Carlo estimates based on the 100 000 realisations simulated using perfect sampling methods. In scenario II $N = 300$ and in scenario III $N = 100$.

k	Scenario		
	I	II	III
0	0.3547	0.3646	0.3721
1	0.2767	0.2763	0.2796
2	0.1680	0.1679	0.1643
3	0.0951	0.0914	0.0896
4	0.0505	0.0495	0.0458
5	0.0264	0.0245	0.0245
6	0.0135	0.0128	0.0127
7	0.0070	0.0066	0.0062
8	0.0037	0.0030	0.0026
9	0.0020	0.0020	0.0012
10	0.0010	0.0007	0.0008

TABLE 4: Estimation of $P(Q = K)$. Scenario I gives the theoretical approximation from [9], and scenarios II and III give Monte Carlo estimates based on the 100 000 realisations simulated using perfect sampling methods. In scenario II $N = 300$ and in scenario III $N = 100$.

k	Scenario		
	I	II	III
0	0.4297	0.4388	0.4511
1	0.2804	0.2829	0.2806
2	0.1488	0.1451	0.1406
3	0.0739	0.0693	0.0693
4	0.0359	0.0339	0.0308
5	0.0174	0.0149	0.0146
6	0.0085	0.0078	0.0068
7	0.0041	0.0037	0.0034
8	0.0020	0.0017	0.0017
9	0.0010	0.0010	0.0007
10	0.0005	0.0005	0.0002

Scenario 4. In Table 4 we present the Monte Carlo estimates for the probability distribution of Q for the fork-join network with $M = 2$, $\lambda = 1$, $s_i = 1$, $\mu_1 = 2$, and $\mu_2 = 3$.

From the results of both scenarios we can see that, with a finite value of N, the number of waiting jobs is randomly smaller than the number of waiting jobs with $N = \infty$, but the difference is small.

Scenario 5. In Figure 9 we plot the Monte Carlo estimates for the probability distribution of Q for the fork-join network with $N = 300$, $M = 3$, $\lambda = 1$, $s_1 = s_2 = 2$, $s_3 = 3$, $\mu_1 = \mu_2 = 1$, and $\mu_3 = 1.5$. The approximation method in [9] does not work for this scenario.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

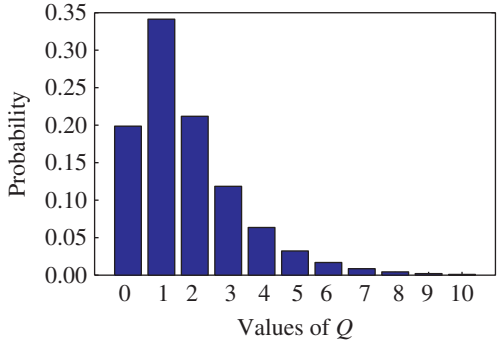

FIGURE 9: Plot of the distribution of Q, with $K = 3, s_1 = s_2 = 2, s_3 = 3, \lambda = 1, \mu_1 = \mu_2 = 1.0$, and $\mu_3 = 1.5$.

5. Computational complexity of the CFTP algorithm

The complexity of the CFTP algorithm depends on the number of simulated $\mathcal{U}^{(l)}$ until coalescence. Thus, we only need to consider $\overline{\boldsymbol{Q}}(t)$ and $\underline{\boldsymbol{Q}}(t)$ at discrete time points $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots\}$. For simplicity, let $\overline{\boldsymbol{Q}}^{(l)} = \overline{\boldsymbol{Q}}(\tau_l)$ and $\underline{\boldsymbol{Q}}^{(l)} = \underline{\boldsymbol{Q}}(\tau_l)$ for $l = 1, 2, \ldots$. The computational complexity depends on the number of steps, ν_c, needed for $\overline{\boldsymbol{Q}}^{(l)}$ and $\underline{\boldsymbol{Q}}^{(l)}$ to coalesce. In this section we will find a bound for $\mathrm{E}\,\nu_c$, given $\lambda < \mu_i s_i$ for all $i < K$.

Let $d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})$ be the distance between the upper chain and lower chain, which is defined as

$$d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}) := \sum_{i=1}^{K} |d_i^{(l)}|,$$

where $d_i^{(l)}$ is the difference between $\overline{\boldsymbol{Q}}^{(l)}$ and $\underline{\boldsymbol{Q}}^{(l)}$ at the ith component. Obviously,

$$d(\overline{\boldsymbol{Q}}^{(0)}, \underline{\boldsymbol{Q}}^{(0)}) = \sum_i N_i,$$

since $\overline{\boldsymbol{Q}}^{(0)} = (N_1, \ldots, N_K)$ and $\underline{\boldsymbol{Q}}^{(0)} = (0, \ldots, 0)$. Define $I(\cdot)$ as the indicator function. Recalling the notation for $\mathbb{A}^{(l)}$ and $\mathbb{B}_i^{(l)}$ in Lemma 2, we have

$$\begin{aligned}
&\mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l+1)}, \underline{\boldsymbol{Q}}^{(l+1)})] \\
&\quad = \mathrm{E}[\mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l+1)}, \underline{\boldsymbol{Q}}^{(l+1)}) \mid \overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}]] \\
&\quad = \sum_{j=1}^{3} \mathrm{E}[I((\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)}) \in B_i^{(l)})\, \mathrm{E}[I(\mathcal{U}^{(l)} \notin \mathbb{A}^{(l)}) d(\overline{\boldsymbol{Q}}^{(l+1)}, \underline{\boldsymbol{Q}}^{(l+1)}) \mid \overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}]] \\
&\qquad + \mathrm{E}[\mathrm{E}[I(\mathcal{U}^{(l)} \in \mathbb{A}^{(l)}) d(\overline{\boldsymbol{Q}}^{(l+1)}, \underline{\boldsymbol{Q}}^{(l+1)}) \mid \overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}]] \\
&\quad \le \mathrm{P}(\mathcal{U}^{(l)} \notin \mathbb{A}^{(l)})\, \mathrm{E}[I((\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)}) \in \mathbb{B}_1^{(l)} \cup \mathbb{B}_2^{(l)}) d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})] \\
&\qquad + \mathrm{P}(\mathcal{U}^{(l)} \notin \mathbb{A}^{(l)})\, \mathrm{E}[I((\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)}) \in \mathbb{B}_3^{(l)})[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}) + K]] \\
&\qquad + \mathrm{P}(\mathcal{U}^{(l)} \in \mathbb{A}^{(l)},\, T_l^{\min} = \overline{T}_l < \underline{T}_l)\, \mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}) - 1] \\
&\qquad + \mathrm{P}(\mathcal{U}^{(l)} \in \mathbb{A}^{(l)},\, T_l^{\min} < \overline{T}_l \le \underline{T}_l)\, \mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})]
\end{aligned}$$

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

$$
\begin{aligned}
&+ \mathrm{P}(\mathcal{U}^{(l)} \in \mathbb{A}^{(l)},\ T_l^{\min} = \overline{T}_l = \underline{T}_l)\, \mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})] \\
&= \mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})] + K\, \mathrm{P}(\mathcal{U}^{(l)} \notin \mathbb{A}^{(l)})\, \mathrm{P}((\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)}) \in \mathbb{B}_3^{(l)}) \\
&\quad - \mathrm{P}(\mathcal{U}^{(l)} \in \mathbb{A}^{(l)},\ T_l^{\min} = \overline{T}_l < \underline{T}_l), \qquad (5)
\end{aligned}
$$

where $\overline{T}_l = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)},\ j = 1, \ldots, \min\{\overline{Q}_i^{(l)}, s_i\},\ i = 1, \ldots, K\}$ and $\underline{T}_l = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)},\ j = 1, \ldots, \min\{\underline{Q}_i^{(l)}, s_i\},\ i = 1, \ldots, K\}$. In (5) the first equality sign is because $\mathcal{U}^{(l)}$ belongs to $\mathbb{A}^{(l)}$ or $(\mathbb{A}^{(l)})^{\mathrm{c}}$ and $(\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)})$ must be in $\mathbb{B}_1^{(l)} \cup \mathbb{B}_2^{(l)} \cup \mathbb{B}_3^{(l)}$. The inequality sign in (5) follows from (3) and the definitions of $\mathbb{A}^{(l)}$ and $\mathbb{B}_i^{(l)}$.

Using properties of exponential variables, it is easy to show that

$$
\mathrm{P}(\mathcal{U}^{(l)} \in \mathbb{A}^{(l)},\ T_l^{\min} = \overline{T}_l < \underline{T}_l) = \mathrm{P}(T_l^{\min} = \overline{T}_l < \underline{T}_l) \geq \frac{\min_{i \leq K} \mu_i}{\Lambda} := \Delta, \qquad (6)
$$

where $\Delta > 0$ is a constant and $\Lambda = \lambda + \sum_i s_i \mu_i$.

We also have the following lemma.

Lemma 4. *Given $\lambda < \mu_i s_i$ for all $i < K$, for a large value of l, we find that* $\mathrm{P}((\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)}) \in \mathbb{B}_3^{(l)})$ *converges to* 0 *at an exponential rate, denoted as ε_l.*

Proof. See Appendix C.

Define $M = KN$. According to Lemma 4, for a large value of M and $l > M$, we have $\varepsilon_l < [\log(l)]^{-1}$. Then, from (5), (6), and Lemma 4, we have

$$
\mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l+1)}, \underline{\boldsymbol{Q}}^{(l+1)})] \leq \mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})](1 + O(\varepsilon_l M^{-1}) - O(M^{-1})). \qquad (7)
$$

Then we have the following theorem.

Theorem 2. *The expected number of coalescence steps is such that*

$$
\mathrm{E}\, v_c \leq O(M^2 \log(M)).
$$

Proof. According to (7), for a large value of M and $l > M$, we have

$$
\begin{aligned}
\mathrm{P}(\overline{\boldsymbol{Q}}^{(l)} \neq \underline{\boldsymbol{Q}}^{(l)}) = \mathrm{P}(d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)}) \geq 1) &\leq \mathrm{E}[d(\overline{\boldsymbol{Q}}^{(l)}, \underline{\boldsymbol{Q}}^{(l)})] \\
&\leq M[1 - O([M \log(M)]^{-1})]^{l-M}.
\end{aligned}
$$

The expected number of coalescence steps is such that

$$
\begin{aligned}
\mathrm{E}\, v_c &= \sum_{l=1}^{\infty} \mathrm{P}(v_c \geq l) \\
&= \sum_{l=1}^{\infty} \mathrm{P}(\overline{\boldsymbol{Q}}^{(l)} \neq \underline{\boldsymbol{Q}}^{(l)}) \\
&\leq M + M\, O(M \log(M)) \\
&= O(M^2 \log(M)).
\end{aligned}
$$

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

TABLE 5: Running time comparisons (in seconds) with $\mu_i = 1$ and $s_i = 3$.

K	N		
	500	1000	1500
5	336	654	983
10	1147	2267	3377
15	2564	4979	7254

Therefore, the expected number of steps needed to achieve coalescence is bounded by $O(M^2 \log(M))$, which means that the complexity of the algorithm increases as a polynomial function of $M = KN$. The algorithm is efficient.

We consider the running time comparisons for different values of N and K. We choose $\lambda = 1$, $\mu_i = 1$, and $s_i = 3$ for all values of i. In Table 5 we present the running times (in seconds) for simulating 10 000 realisations on a desktop with a 2.66 GHz Intel® Core™ Duo processer. We can see that the running time increases as N and K increase. The algorithm is efficient for reasonably large values of N and K.

6. Discussion

We presented a perfect sampling method based on CFTP to draw perfect simulations from the equilibrium of a fork-join network with finite capacity and zero service time at the join station. This work is important since most existing works focus on the analytic approximation for fork-join networks with infinite capacity. The accuracy of existing approximation methods is difficult to justify in some cases. For example, the analytic approximation formula of $F(t) = \mathrm{P}(W \le t)$ in [9] depends on some constant coefficients. The accurate values of the coefficients are sometimes found by simulation experiments (details can be found in [9]). The proposed simulation method can provide a Monte Carlo estimate, whose accuracy is controlled by the number of simulated realisations. The proposed perfect sampling method brings new insight to this area.

The proposed CFTP method makes use of the memoryless property of exponential service times and simulates continuous-time Markov chains using the *thinning* idea. When service times do not follow an exponential distribution, the *thinning* idea is not valid because the memoryless property does not hold. Thus, future research work could seek perfect sampling methods for fork-join networks with general independent service times (see [1] and [17]).

Other future research work could seek perfect sampling methods for fork-join networks with infinite capacity, i.e. $N = \infty$, although the assumption of finite capacity is reasonable in practice. This problem may be solved by extending the dominated CFTP method in [8]. Future research work in this area could also seek perfect sampling methods for fork-join networks with a nonzero service time at the join station or for networks with dynamic policies for scheduling multiple tasks.

Appendix A. Proof of Lemma 1

Algorithm 1 obviously simulates the transition probability (steps 03, 05, and 06 of Algorithm 2) from (2). We only need to show that it simulates holding times correctly.

Note that Algorithm 1 outputs $\boldsymbol{\tau} = \{\tau_0, \tau_1, \ldots\}$, $\{\boldsymbol{Q}(\tau_0), \boldsymbol{Q}(\tau_1), \ldots\}$ (step 01 of Algorithm 2), and all transition times $\{t_0, t_1, \ldots\}$ (a subset of $\boldsymbol{\tau}$). Since

$$T_l^{\min} = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)},\ j = 1, \ldots, s_i,\ i = 1, \ldots, K\}$$

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

is exponentially distributed with rate $\Lambda := \lambda + \sum_{i=1}^{K} s_i \mu_i$, then, obviously, $\tau_0, \tau_1, \ldots$ are jump times of a Poisson process with rate Λ.

Now consider the thinning algorithm for simulating a Poisson process with a rate r, $r < \Lambda$. Suppose that I_k is a Bernoulli random variable independent of $T^{\min}$ and is such that $\mathrm{P}(I_l = 1) = r/\Lambda$. Then the subsequence $\{\tau_l \colon I_l = 1\}$ will be the jump times of a Poisson process with rate r. From this thinning idea we can simulate an exponential variable with mean $1/r$ as follows. Suppose that $I_l = 1$, and that we simulate $I_{l+1}, I_{l+2}, \ldots$ until we obtain the first $I_{l'} = 1$. Then the interarrival time $\tau_{l'} - \tau_l$ must come from an exponential distribution with mean $1/r$. We use this idea to simulate the holding times of the Markov chain.

Suppose that $t_k = \tau_l$ and $\boldsymbol{Q}(\tau_l) = \boldsymbol{n}^{(l)}$. First consider $\boldsymbol{n}^{(l)} + \mathbf{1} \leq \boldsymbol{N}$. Since $T_l = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)},\ j = 1, \ldots, \min\{n_i^{(l)}, s_i\},\ i = 1, \ldots, K\}$, we know that

$$\mathrm{P}(T_l^{\min} = T_l) = \frac{r_{\boldsymbol{n}^{(l)}}}{\Lambda} = \left(\lambda + \sum_i \mu_i \min\{n_i, s_i\}\right) \Big/ \Lambda.$$

Thus, we can define $I_l = 1$ if $T_l^{\min} = T_l$ and $I_l = 0$ otherwise. Algorithm 1 generates $I_{l+1}, I_{l+2}, \ldots$ until we obtain the first $I_{l'} = 1$, i.e. $T_{l'}^{\min} = T_{l'}$ and outputs the time $t_{k+1} = \tau_{l'}$ (step 05 of Algorithm 1). Therefore, $t_{k+1} - t_k$ has an exponential distribution with mean $1/r_{\boldsymbol{n}^{(l)}}$ and it is the holding times of the process at $\boldsymbol{Q}(\tau_l) = \boldsymbol{n}^{(l)}$.

If $\boldsymbol{n}^{(l)} + \mathbf{1} / \not\leq \boldsymbol{N}$, $T_l^{\min} = T_l = \xi_0^{(l)}$ implies that there is no jump at time $\tau_{l+1} = \tau_l + T_l^{\min}$ (step 05 of Algorithm 2 will reset $I = 0$). Therefore, the probability of runn- ing step 07 of Algorithm 2 is $\mathrm{P}(T_l^{\min} = T_l = \xi_{ij}^{(l)}$ for some $i) = r_{\boldsymbol{n}^{(l)}}/\Lambda = \sum_i \mu_i \min\{n_i, s_i\}/\Lambda$. With similar arguments as above, we know that the holding times are simulated correctly.

Appendix B. Proof of Lemma 2

Algorithm 1 outputs a Poisson process with rate $\Lambda = \lambda + \sum_i s_i \mu_i$ having jump times at $\boldsymbol{\tau} = \{\tau_k,\ k = 1, \ldots\}$. According to the updating rules of Algorithm 1, a simulated Markov chain can only have possible jumps at time τ_k.

We know that $\tau_{l+1} = \tau_l + T_l^{\min}$, where $T_l^{\min}$ is defined in Algorithm 2. Note that the partial order is preserved in the interval (τ_l, τ_{l+1}) since there is no jump for both $\boldsymbol{Q}(t)$ and $\boldsymbol{Q}'(t)$. Therefore, result (a) is proved.

Now consider $a = T_l^{\min}$. Since the generated variables $\xi_0^{(l)}, \xi_{ij}^{(l)}$ for $j = 1, \ldots, s_i$, and $i = 1, \ldots, K$, will be mutually unequal with probability 1, given $T_l^{\min} \neq \xi_0^{(l)}$, we know that, for some i, j, one of the following cases must be true: (i) $T_l^{\min} = T_l' = T_l = \xi_{ij}^{(l)}$, (ii) $T_l^{\min} = T_l' = \xi_{ij}^{(l)} < T_l$, or (iii) $T_l^{\min} < T_l' \leq T_l$. According to (3) if (i) is true, $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{n} - \boldsymbol{e}_i \prec \boldsymbol{Q}'(\tau_{l+1}) = \boldsymbol{n}' - \boldsymbol{e}_i$ (the partial order is preserved). If (ii) is true then $n_i' < n_i$. Therefore, according to (3), $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{n} \preceq \boldsymbol{Q}'(\tau_{l+1}) = \boldsymbol{n}' - \boldsymbol{e}_i$ (the partial order is preserved). If (iii) is true then neither $\boldsymbol{Q}(t)$ or $\boldsymbol{Q}'(t)$ has a jump at time $\tau_{l+1} = \tau_l + T_l^{\min}$. Therefore, the partial order is preserved since $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{n} \prec \boldsymbol{Q}'(\tau_{l+1}) = \boldsymbol{n}'$. Thus, result (b) is proved.

For $a = T_l^{\min}$, if $T_l^{\min} = \xi_0^{(l)}$ then $T_l^{\min} = T_l' = T_l = \xi_0^{(l)}$. On the other hand, if $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l)) \in \mathbb{B}_1^{(l)}$, i.e. $\boldsymbol{n}' + \mathbf{1} \leq \boldsymbol{N}$, we have $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{n} + \mathbf{1} \prec \boldsymbol{Q}'(\tau_{l+1}) = \boldsymbol{n}' + \mathbf{1}$ according to (3) (the partial order is preserved). Result (c) is proved.

For result (d), the conditions are $T_l^{\min} = T_l' = T_l = \xi_0^{(l)}$ and $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l)) \in \mathbb{B}_2^{(l)}$, i.e. $\boldsymbol{Q}(\tau_l) + \mathbf{1} \not\leq \boldsymbol{N}$. According to (3), we have $\boldsymbol{Q}(\tau_{l+1}) = \boldsymbol{n} \prec \boldsymbol{Q}'(\tau_{l+1}) = \boldsymbol{n}'$ (the partial order is preserved). Result (d) is proved.

For result (e), the conditions are $T_l^{\min} = T_l' = T_l = \xi_0^{(l)}$ and $(\boldsymbol{Q}(\tau_l), \boldsymbol{Q}'(\tau_l)) \in \mathbb{B}_3^{(l)}$. Then, according to (3), $\phi(\boldsymbol{Q}(\tau_l), \mathcal{U}^{(l)}, a) = \boldsymbol{n} + \mathbf{1} \preceq \phi(\boldsymbol{Q}'(\tau_l), \mathcal{U}^{(l)}, a) = \boldsymbol{n}'$ may not be true (the partial order may not be preserved). Result (e) is proved.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

Appendix C. Proof of Lemma 4

For the definition of $\mathbb{B}_3^{(l)}$ in Lemma 2 we have

$$\mathrm{P}((\underline{\boldsymbol{Q}}^{(l)}, \overline{\boldsymbol{Q}}^{(l)}) \in \mathbb{B}_3^{(l)}) \le \mathrm{P}(\overline{Q}_i^{(l)} = N_i \text{ for at least one } i) \le K \max_i \mathrm{P}(\overline{Q}_i^{(l)} = N_i).$$

Let $h = \lfloor \sqrt{l} \rfloor$, where $\lfloor \sqrt{l} \rfloor$ is the floor function. Let event $\mathbb{D}$ be the event that from time 0 to τ_l there are more than h jumps for $\overline{Q}_i(t)$.

First we show that $\mathrm{P}(\mathbb{D}^{\mathrm{c}})$ converges to 0 at an exponential rate. At each time τ_l, $\overline{Q}_i(t)$ has a jump with probability less than or equal to $(\mu_i s_i + \lambda)/\Lambda$. Therefore, we have

$$\begin{aligned}
\mathrm{P}(\mathbb{D}^{\mathrm{c}}) &= \mathrm{P}(\text{there are } k \text{ jumps at times } \tau_1, \ldots, \tau_l, \text{ with } k \le h) \\
&\le \sum_{k=1}^{h} \binom{l}{k} \left(\frac{\lambda + \mu_i s_i}{\Lambda} \right)^k \left(\frac{\lambda + \sum_{j \ne i} \mu_j s_j}{\Lambda} \right)^{l-k} \\
&\le \frac{l!}{h!\,(l-h)!} h \delta_1^l,
\end{aligned}$$

where $0 < \delta_1 = \max_i \{(\lambda + \mu_i s_i)/\Lambda, (\lambda + \sum_{j \ne i} \mu_j s_j)/\Lambda\} < 1$. Then, using Stirling's approximation for $l!/h!\,(l-h)!$, we have

$$\mathrm{P}(\mathbb{D}^{\mathrm{c}}) \le O\left(h^h \left(\frac{l}{l-h} \right)^{l-h} \right) h^{1/2} \delta_1^l = O((h \mathrm{e} \delta_1^h)^h h^{1/2}),$$

which converges to 0 at an exponential rate.

The discrete transitions for $\overline{Q}_i(t)$ are the same as the path of a simple random walk $\boldsymbol{R} = \{R_k,\ k = 0, \ldots, h\}$ starting from N_i with barriers 0 and N_i. The transition probabilities of $\boldsymbol{R}$ may be different at different states, but the positive jump probabilities p of $\boldsymbol{R}$ are such that $p \le \lambda/(\mu_i s_i + \lambda) < \frac{1}{2}$ when $\lambda < \mu_i s_i$. It is easy to show that, given $R_0 = N_i$, for a large value of k,

$$\begin{aligned}
\mathrm{P}(R_k = N_i) &\le \sum_{j=0}^{\lfloor (k+1)/2 \rfloor} \binom{k}{j} \left(\frac{\lambda}{\lambda + \mu_i s_i} \right)^{k-j} \left(\frac{\mu_i s_i}{\lambda + \mu_i s_i} \right)^j \\
&\le O(2^k k^{-1/2}) \sum_{j=0}^{k/2} \left(\frac{\lambda}{\lambda + \mu_i s_i} \right)^{k-j} \left(\frac{\mu_i s_i}{\lambda + \mu_i s_i} \right)^j \\
&= O(2^k k^{1/2}) \left(\frac{\lambda}{\lambda + \mu_i s_i} \right)^{k/2} \left(\frac{\mu_i s_i}{\lambda + \mu_i s_i} \right)^{k/2} \\
&= O\left(k^{1/2} \left[\frac{4 \lambda \mu_i s_i}{(\lambda + \mu_i s_i)^2} \right]^{k/2} \right) \\
&\le O(k^{1/2} \delta_2^{k/2}),
\end{aligned}$$

where $0 < \delta_2 = \max_i \{4 \lambda \mu_i s_i / (\lambda + \mu_i s_i)^2\} < 1$. Thus, $\mathrm{P}(R_k = N_i)$ converges to 0 at an exponential rate as $\lambda < \mu_i s_i$. Therefore,

$$\begin{aligned}
\mathrm{P}(\overline{Q}_i^{(l)} = N_i) &= \mathrm{P}(\overline{Q}_i^{(l)} = N_i, \mathbb{D}) + \mathrm{P}(\overline{Q}_i^{(l)} = N_i, \mathbb{D}^{\mathrm{c}}) \\
&\le l\, \mathrm{P}(R_h = N_i) + \mathrm{P}(\mathbb{D}^{\mathrm{c}}) \\
&= O(l^{5/4} \delta_2^{h/2}) + O((h \mathrm{e} \delta_1^h)^h h^{1/2}),
\end{aligned}$$

which converges to 0 at an exponential rate, denoted as ε_l.

https://doi.org/10.1239/aap/1308662489 Published online by Cambridge University Press

References

[1] Ayhan, H. and Seo, D.-W. (2001). Laplace transform and moments of waiting times in Poisson driven (max,+) linear systems. *Queueing Systems* **37,** 405–438.

[2] Baccelli, F., Massey, W. A. and Towsley, D. (1989). Acyclic fork-join queuing networks. *J. Assoc. Comput. Math.* **36,** 615–642.

[3] Balsoma, S., Donatiello, L. and van Dijk, N. M. (1998). Bound performance models of heterogeneous parallel processing systems. *IEEE Trans. Parallel Distributed Systems* **9,** 1041–1056.

[4] Dai, H. (2008). Perfect sampling methods for random forests. *Adv. Appl. Prob.* **40,** 897–917.

[5] Flatto, L. (1985). Two parallel queues created by arrivals with two demands. II. *SIAM J. Appl. Math.* **45,** 861–878.

[6] Flatto, L. and Hahn, S. (1984). Two parallel queues created by arrivals with two demands. I. *SIAM J. Appl. Math.* **44,** 1041–1053.

[7] Huber, M. (2004). Perfect sampling using bounding chains. *Ann. Appl. Prob.* **14,** 734–753.

[8] Kendall, W. S. and Møller, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Adv. Appl. Prob.* **32,** 844–865.

[9] Ko, S-S. and Serfozo, R. F. (2004). Response times in M/M/s fork-join networks. *Adv. Appl. Prob.* **36,** 854–871.

[10] Nelson, R. and Tantawi, A. N. (1988). Approximate analysis of fork-join synchronization in parallel queues. *IEEE Trans. Software Eng.* **37,** 739–743.

[11] Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms* **9,** 223–252.

[12] Raghavan, N. R. S. and Viswanadham, N. (2001). Generalized queueing network analysis of integrated supply chains. *Internat. J. Production Res.* **39,** 205–224.

[13] Ross, S. M. (2007). *Introduction to Probability Models*, 8th edn. Academic Press, Burlington, MA.

[14] Squillante, M. S., Zhang, Y., Sivasubramaniam, A. and Gautam, N. (2008) Generalized parallel-server fork-join queues with dynamic task scheduling. *Ann. Operat. Res.* **160,** 227–255.

[15] Wilson, D. B. (2000). How to couple from the past using a read-once source of randomness. *Random Structures Algorithms* **16,** 85–113.

[16] Xia, C. H., Liu, Z., Towsley, D. and Lelarge, M. (2007). Scalability of fork/join queueing networks with blocking. In *ACM SIGMETRICS Performance Evaluation Review*, Association for Computing Machinery, New York, pp. 133–144.

[17] Zhang, Z. (1990). Analytical results for waiting time and system size distributions in two parallel queueing systems. *SIAM J. Appl. Math.* **50,** 1176–1193.