# DISTRIBUTED RANDOM LOAD BALANCING

by

Chunpu Wang

B.Eng., Harbin Institute of Technology, P. R. China, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Electrical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

May 2017

The undersigned certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis entitled:

Distributed Random Load Balancing

submitted by Chunpu Wang in partial fulfilment of the requirements of the degree of

Master of Applied Science.

Dr. Chen Feng, School of Engineering

**Supervisor, Professor** (please print name and faculty/school above the line)

Dr. Yang Cao, School of Engineering

**Supervisory Committee Member, Professor** (please print name and faculty/school in the line above)

**Supervisory Committee Member, Professor** (please print name and faculty/school in the line above)

Dr. Heinz Bauschke, Faculty of Arts and Science/Mathematics

**University Examiner, Professor** (please print name and faculty/school in the line above)

**External Examiner, Professor** (please print name and university in the line above)

31/May/2017

(Date Submitted to Grad Studies)

**Additional Committee Members include:**

Dr. Julian Cheng, School of Engineering

(please print name and faculty/school in the line above)

(please print name and faculty/school in the line above)

# Abstract

Low latency is highly desirable for cloud services spanning thousands of servers. With the rapid development of cloud market, the size of server farms grows fast. Hence, stringent timing requirements are needed for task scheduling in a large-scale server farm. Conventionally, the Join-the-Shortest-Queue (JSQ) algorithm, which directs an arriving task to the least loaded server, is adopted in scheduling. Despite its excellent delay performance, JSQ is throughput-limited, and thus doesn't scale well with the number of servers. There are two distributed algorithms proposed as "approximations" of the idealized JSQ. The first one is the Power-of-$d$-choices (Po$d$) algorithm, which selects $d$ servers at random and routes a task to the least loaded server of the $d$ servers. Despite its scalability, Po$d$ suffers from long tail response times. The second one is the distributed Join-the-Idle-Queue (JIQ) [1], which take advantages idle servers for task scheduling.

In this thesis, we are interested in exploring Po$d$ and JIQ further. First, a hybrid scheduling strategy called Po$d$-Helper is proposed. It consists of a Po$d$ scheduler and a throughput-limited helper. Hybrid scheduling takes the best of both worlds, enjoying scalability and low tail response times. In particular, hybrid scheduling has bounded maximum queue size in the large-system regime, which is in sharp contrast to the Po$d$ scheduling whose maximum queue size is unbounded. Second, we conduct an in-depth analysis for distributed Join-the-Idle-Queue (JIQ), a promising new approximation of an idealized task-scheduling algorithm. In particular, we derive semi-closed form expressions for the delay performance of distributed JIQ. Third, we propose a new variant of distributed JIQ that offers clear advantages over alternative algorithms for large systems.

# Preface

This thesis is based on the research work conducted in the School of Engineering at The University of British Columbia, Okanagan Campus, under the supervision of Dr. Chen Feng and Dr. Julian Cheng. It contains both published and submitted works.

Chapter 2 of this thesis has been published in the *International Conference on Computing, Networking and Communications (ICNC)*. The research idea, problem definition, mean-field analysis, and numerical simulations are the results of my work.

Chapters 3 of this thesis have been submitted to the *IEEE Journal on Selected Areas in Communications*. I am the principle contributor for this work. Dr. Feng helped me with the problem definition and theoretical analysis of the manuscripts.

The reuse of all the materials in this thesis is authorized by the co-authors of corresponding publication. A list of my papers published at The University of British Columbia, Okanagan Campus is provided below.

## Conference Paper Accepted

1. Chunpu Wang, Chen Feng and Julian Cheng, "Randomized load balancing with a helper," *International Conference on Computing, Networking and Communications (ICNC)*, Santa Clara, CA, 2017, pp. 518-524.

## Journal Paper Submitted

1. Chunpu Wang, Chen Feng and Julian Cheng, "Distributed Join-the-Idle-Queue for Low Latency Cloud Services," submitted to *IEEE Journal on Selected Areas in Communications*.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

| Acronyms | Definitions |
|---|---|
| FIFO | First-In-First-Out |
| JSQ | Join-the-Shortest-Queue algorithm |
| JIQ | Join-the-Idle-Queue algorithm |
| Po$d$ | Power-of-$d$-choices algorithm |

# Acknowledgements

First of all, I am deeply grateful to my thesis supervisor Dr. Chen Feng and Dr. Julian Cheng for their guidance and help. It is very lucky to receive constant encouragement and advice from both of them.

Second, I would like to thank Dr. Heinz Bauschke for his willingness to serve as my external examiner. I would also like to thank Dr. Yang Cao for his willingness to serve on the committee. I really appreciate their valuable time and constructive comments on my thesis.

I also have many thanks to my friends Yanjie Dong, Changle Zhu, Haobing Chu, Renming Qi, Yonghui Lv, Shou Wang, Junyuan Leng, Shou Liu, Junchi Bin, Huan Liu, Fang Shi at the University of British Columbia for their encouragement and friendship.

Finally, I would like to specially thank my parents and my wife for their constant support and love over my graduate studies.

# Chapter 1

# Introduction

## 1.1 Motivation and Contributions

Nowadays, the market of cloud computing, such as online applications, is growing fast. Take Facebook as an example. According to the Social Skinny, Facebook users posted 510 comments, 293,000 statuses and 136,000 photos for every minute in 2016. Daily usage of online applications is so frequent that huge amount of tasks are created every second. Hence, more servers are required to cope with such massive incoming tasks. Even as of June 2014, Facebook was running at roughly 180,000 servers in its datacenters. Similarly, Amazon Web Services had around 1.3 million servers in 2016. Such huge amount of servers consist of a large system.

In many cloud computing applications, such as web search and big-data processing, a scheduler assigns arriving tasks to appropriate servers. The objective of the scheduler is to minimize the response times, which are crucial for user experience. For instance, an extra delay of 500 ms in response time could result in a 1.2% loss of users and revenue, according to Amazon and Google [2]. Hence, low latency is highly desirable for online services spanning thousands of servers. For example, Google search typically returns the query results within a few hundreds of milliseconds. The demand for fast response time, which significantly impacts user experience and service-provider revenue, is translated into stringent timing requirements for task scheduling in a large server farm.

Ideally, to minimize response times, a scheduler can track the queue lengths of all the servers and select the least-loaded server (i.e., the server with the shortest queue) whenever there is a task arrival. This scheduling strategy—often referred as the Join-the-Shortest-Queue (JSQ) algorithm—is proven to be delay optimal in certain traffic regimes (see [3] and references therein). However, JSQ scheduling is throughput-limited and doesn't scale well with the number

of servers. This is because JSQ needs to track the status of all the servers and thus cannot make scheduling decisions at high rate for a system with large number of servers, i.e., large system.

To alleviate this problem, the Power-of-$d$-Choices (Po$d$) algorithm ($d \geq 2$) has been proposed as an "approximation" of the idealized JSQ. Instead of tracking the global information, Po$d$ only probes $d$ servers uniformly at random upon a task arrival and selects the least loaded one for the new task. Although Po$d$ achieves reasonably good average response time [4], its tail response time still remains high for large systems [5, 6] and its probing operation incurs additional delay. This is undesirable for low-latency big-data-processing jobs (such as interactive Map/Reduce jobs), where each job (which often consists of a number of tasks) is sensitive to tail response time (since a job cannot complete until its last task finishes) [5, 6].

Recently, distributed Join-the-Idle-Queue (JIQ) [1] has emerged as a promising new approximation of JSQ. JIQ employs a number of distributed schedulers, each maintaining an I-queue that stores a list of idle servers. When a new task arrives at the system, it randomly visits a scheduler asking to join an idle server in its I-queue. Compared to JSQ, each scheduler in JIQ only maintains local information and is scalable to large systems. Compared to Po$d$, each scheduler in JIQ avoids the probing operation and assigns a new task to an idle server directly as long as its I-queue is non-empty. Due to its clear advantages, JIQ has begun to attract research attention from both industry and academia [7–10]. Despite these significant research achievements made recently, distributed JIQ is not yet well understood from a theoretical perspective. For example, there is no closed-form expression yet that *exactly* characterizes the delay performance of distributed JIQ [7][1]. Also, there seems no theoretical guarantee that the distributed JIQ (or any of its variants) is strictly better than Po$d$.

In this thesis, we are interested in exploring the distributed Po$d$ and distributed JIQ algorithms. First, we propose a *hybrid* scheduling algorithm called Po$d$-Helper that consists of a Po$d$ scheduler and a throughput-limited helper. Specifically, the helper constantly monitors the status of all the servers, and "steals" tasks from the most-loaded server (i.e., the server with the

[1] In [1], the authors provided closed-form expression that approximately characterizes the delay performance of distributed JIQ based on some simplifying assumptions. Although their expression is insightful, it is not very accurate for our system model as explained in Section 3.3.

longest queue) at a certain (limited) rate. Intuitively, this hybrid strategy can effectively reduce tail response times, because it essentially prevents the longest queue from growing. We then study this hybrid scheduling using a mean-field analysis. In particular, we show that, under some mild condition on the helper rate, the maximum queue size with the hybrid scheduling is bounded in the large system regime. This means that tail response times of our scheduler are indeed low.

Second, we take a further step in understanding the performance of distributed JIQ, applying a mean-field analysis to derive semi-closed form expressions of the stationary tail distribution and the expected response time for distributed JIQ. Our expressions contain a parameter $\hat{p}_0 \in (0, 1)$ that can be efficiently calculated by a binary search. We show that, in the large-system limit, the tail probability $\hat{s}_i$ of a server having at least $i$ tasks is given by $\hat{s}_i = \hat{p}_0^{i-1}\lambda^i$, where $\lambda$ is the normalized arrival rate. We also show that the expected task response time is $1+\hat{p}_0 \sum_{i=1}^{\infty} \hat{s}_i$. These two expressions allow us to compare JIQ and Po$d$ directly and suggest that JIQ is not always better than Po$d$.

Third, we propose a new variant of JIQ called JIQ-Po$d$ that strictly outperforms Po$d$. JIQ-Po$d$ enjoys the best of both worlds. Similar to JIQ, a scheduler with a non-empty I-queue in JIQ-Po$d$ assigns a new task to an idle server on its I-queue. Similar to Po$d$, a scheduler with an empty I-queue in JIQ-Po$d$ probes $d$ servers and selects the least loaded one. Intuitively, JIQ-Po$d$ improves upon JIQ in that it makes schedulers with empty I-queues "smarter"; it improves upon Po$d$ in that schedulers with non-empty I-queues can assign new tasks directly to idle servers without the probing operation. Using the mean-field analysis, we are able to quantify the improvements of JIQ-Po$d$ over JIQ and Po$d$ in the large-system limit.

## 1.2 Literature Review

The Po$d$ algorithm and its variants have been used in today's cloud systems as an alternative to the ideal JSQ algorithm. One such variant is called batch sampling, which works quite well when task arrivals occur in batches [5]. In particular, batch sampling is able to significantly reduce tail response times compared to the original Po$d$ algorithm. In fact, batch sampling leads to bounded maximum queue size, as shown in a recent study [6]. Another variant is

called batch filling [6], which also exploits batch arrivals and achieves better delay performance than batch sampling. Unlike these two variants, our hybrid scheduling doesn't rely on batch arrivals; instead, it explores a new dimension (i.e., the use of a helper) to reduce tail response times.

The use of hybrid scheduling has begun to receive increasing attention from both academia and industry. This line of work includes Hawk [11] and Mercury [12]. For instance, long jobs in Hawk are scheduled with a centralized scheduler, and short ones are scheduled in a fully distributed way [11]. To the best of the authors' knowledge, most of these works are based on simulations or prototype implementations. Our work differs from those works in that we seek for a theoretical understanding of the benefits of hybrid scheduling. In addition, our hybrid scheduling operates differently from Hawk and Mercury.

Our Po$d$-Helper is partially inspired by the seminal work of Tsitsiklis and Xu [13] on the power of resource pooling. They found a strong qualitative impact of even a small degree of centralization. In their system model, a fraction of $p$ of the total available service rate is deployed as a central server, while the remaining fraction $1-p$ is allocated to $n$ local servers. In other words, there is only a helper in the system to coordinate workloads among independent servers. In contrast, our hybrid strategy combines a helper with a Po$d$ scheduler. Such a combination leads to better delay performance, as we will see in Chapter 2.3.

The JIQ algorithm was originally proposed in a seminal work [1] in 2011. The authors assumed that all servers in I-queues are idle as a simplification of their performance analysis. As pointed out in [1], this assumption is violated when an idle server receives a random arrival. When studying JIQ in our work, we do not make such assumption. Instead, we introduce delete request messages (as explained later) to ensure that all servers in I-queue are idle. To simplify notation, we just use JIQ to denote the distributed JIQ with delete request messages, while using JIQ-original to denote the JIQ proposed in [1].

Recently, Mitzenmacher studied the distributed JIQ algorithm through a fluid limit approach [7]. He proposed an innovative classification of the states of servers and derived families of differential equations that describe the JIQ system in the large-system limit. Due to the high complexity of those differential equations, there is no expression of the equilibrium in a

convenient form in terms of $\lambda$ [7]. Our work is inspired by Mitzenmacher's fluid limit approach. By introducing delete request messages, we are able to simplify the differential equations and obtain semi-closed form expressions for distributed JIQ. Based on the insights from our analysis, we propose and analyzed a new variant of JIQ that outperforms both JIQ and Po*d* in all conditions.

In a series of papers [8–10], Stolyar studied a centralized JIQ algorithm where there is only one scheduler (or a fixed number of schedulers) in the system through mean-field analysis. It shows that centralized JIQ approaches the performance of JSQ in the large-systems limit. This means that a centralized scheduler only needs to track idle servers instead of all the servers.

## 1.3 Mathematical Background

The mean-field analysis is used to study the process of large and complex stochastic models. Generally, in the mean-field analysis, we derive deterministic differential equations to describe the system with infinite size. Here is a simple example [14]. In an epidemic model, we have the total population $N$, the number of infected people $N_i$, the number of susceptible people $N_s$ and the number of immune people $N_m$. Suppose the chance for a susceptible person becoming an infected person is proportional to the fraction of the infected population with fixed parameter $a$, the chance for an infected person becoming an immune person is proportional to a fixed parameter $b$. Then the transition rates are as following:

$$(N_s, N_i, N_m) \to (N_s - 1, N_i + 1, N_m) : q_1 = a\frac{N_i}{N}N_s = aN\frac{N_i}{N}\frac{N_s}{N}, \tag{1.1a}$$

$$(N_s, N_i, N_m) \to (N_s, N_i - 1, N_m + 1) : q_2 = bN_i = bN\frac{N_i}{N}. \tag{1.1b}$$

Let $n_i$ be the fraction of the infected population, $n_s$ be the fraction of the susceptible population, and $n_m$ be the fraction of the immune population. We then have the following differential equations:

$$\frac{dn_s}{dt} = -an_in_s, \tag{1.2a}$$

$$\frac{dn_i}{dt} = an_in_s - bn_i, \tag{1.2b}$$

$$\frac{dn_m}{dt} = bn_i. \tag{1.2c}$$

Intuitively speaking, the expectation of the system is close to the solution of above differential equations as the law of large numbers will take effect when the size of the system goes to infinity. Based on those differential equations, we can study the stationary distribution of the infinite system when $t \to \infty$.

In fact, the simple example of an epidemic model is a typical density dependent jump Markov process. The concept of the density dependent jump Markov processes goes as follows.

*Definition 1.1:* For $n \geq 1$, the continuous-time Markov process $\{\mathbf{X}_n(t), t \geq 0\}$ is a density dependent Markov process on the $d$-dimensional lattice $\mathbf{Z}^d$, such that:

1. There are only finite possible transactions $l$ of $\mathbf{X}(t)_n$, that $l \subseteq \mathbf{Z}^d$;
2. The transition rates are $q_{x,x+l}^{(n)} = n\beta_l(n^{-1}x)$, where $\beta_l$ are nonnegative continuous functions.

Then, we can measure the error between finite and infinite density dependent jump Markov systems. A well-known theorem, called Kurtz's Theorem, helps with the approximation of the finite system when the system size $n \to \infty$. Kurtz's Theorem states that the deterministic process is the limit of the process when appropriate differential equations are chosen.

*Theorem 1.2:* Consider there is a sequence of density dependent Markov processes $\{\mathbf{X}_n(t)\}$, which satisfies that:

1. The Lipschitz condition $|F(x) - F(y)| \leq K|x - y|$, where $F(x) = \sum_l l\beta_l(x)$ and $K$ is a certain constant;
2. For $x_0$, we have $x_0 = \lim_{n\to\infty} X_n(0)$.

Set $X(t)$ to be a deterministic process $X(t) = x_0 + \int_0^t F(X(u))du$. Then

$$\lim_{n\to\infty} \sup_{u\leq t} |X_n(u) - X(u)| = 0, \text{ almost surely.} \tag{1.3}$$

Kurtz's Theorem originally only applies to the finite dimensional system (i.e., $d$ is a finite

number). In the work of the "power of two choices" [4], the author extended it to include some infinite dimensional systems.

## 1.4 Organization of the Thesis

Our thesis consists of four chapters, which are organized as follows.

In Chapter 2, we introduce our system model of Pod-Helper. Through mean-field analysis, we show that it has a bounded maximum queue size in the condition of the large-system regime. The simulation result indicates such nice property also holds when system size is limited.

In Chapter 3, we introduce our system model of JIQ and JIQ-Po$d$. Then we perform the mean-field analysis for both JIQ and JIQ-Po$d$, obtaining semi-closed form expressions of the stationary distributions of JIQ and JIQ-Po$d$. Additionally, extensive simulations are conducted to validate our analysis.

In Chapter 4, we conclude the whole thesis and list some possible future work related to our Po$d$ and JIQ variants.

# Chapter 2

# Analysis of Pod-Helper Algorithm

In this chapter, we study the Po*d*-Helper algorithm through mean-field analysis. We mainly focus on the large-systems limit (i.e., the number of servers tends to $\infty$), since today's datacenter often consists of tens of thousands of servers. Our main results state that, in the large-systems regime, the maximum steady-state queue size of the hybrid algorithm is bounded and the average steady-state response time is strictly smaller than that of the Po*d* algorithm. We then use mean-field analysis to cover two proof paths. Also, we conduct simulation to verify the correctness of our theoretical analysis.

## 2.1 System Model and Main Results

### 2.1.1 System Model

Consider a cluster with $N$ identical servers together with a Po*d* scheduler ($d \geq 2$) and a throughput-limited helper, as illustrated in Fig. 2.1. Time is assumed to be continuous.

1. *Servers*: Each server maintains a first-in first-out (FIFO) queue which stores the tasks to be processed. The queue length (i.e., the number of tasks) for server $i$ at time $t$ is denoted by $Q_i(t)$ for $i \in \{1, 2, \ldots, N\}$. To simplify analyze, we assume that the task service times at each server are independent and exponentially distributed with mean 1. Or equivalently, the task processing at each server can be modelled as a "Poisson clock" with rate 1 (i.e., the times between two clock ticks are independent and exponentially distributed with mean 1). These Poisson processes are independent of each other. When the clock for server $i$ ticks at time $t$, if $Q_i(t) > 0$, server $i$ completes the task at the head of the queue and removes it from the queue immediately; otherwise, server $i$ does nothing and this clock tick is wasted. This equivalent interpretation facilitates our analysis.

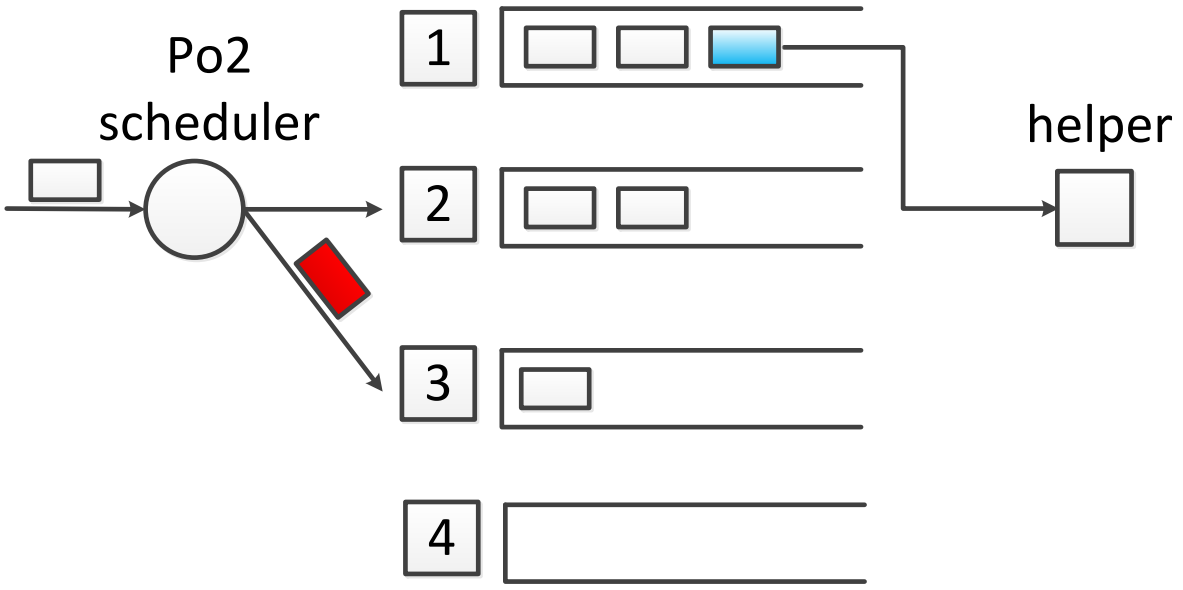


Figure 2.1: Hybrid scheduling with a Po2-scheduler and a helper for a cluster with 4 servers.

2. *Task Arrivals*: Tasks arrive to the system according to a Poisson process with rate $\lambda N$ where $\lambda < 1$.

3. *Pod Scheduler*: Upon a task arrival, the Po$d$ scheduler samples $d$ servers uniformly at random and directs the task to the least-loaded server (i.e., the server with the shortest queue)[2]. If there are multiple least-loaded servers, the scheduler chooses one uniformly at random.

4. *Helper*: The task processing at the helper is modelled as a Poisson clock with rate $\epsilon N$, where $\epsilon \ll \lambda$ (since the helper is throughput-limited). When the clock at the helper ticks at time $t$, if the system is non-empty (i.e., $\sum_{i=1}^{N} Q_i(t) > 0$), the helper "steals" the head-of-the-queue task from some server $i$, chosen uniformly at random among the most loaded-servers (i.e., the server with the longest queue), and completes the task immediately. Otherwise, the helper does nothing, and the clock tick is wasted[3].

*Remark 1:* When $\epsilon = 0$, our system model reduces to the standard setup for the Po$d$ algo-

[2]Although our system model assumes a single Po$d$ scheduler, it can be easily extended to the case of multiple Po$d$ schedulers as long as the task arrivals on these schedulers are independent Poisson processes with aggregated rate $\lambda N$.

[3]Similarly, our system model can be easily extended to the case of multiple helpers as long as the aggregate rate of their Poisson-clock processes is $\epsilon N$.

rithm. Setting $d = 1$, our system model is essentially the same as that in [13], where servers receive streams of tasks according to independent Poisson processes with a common rate $\lambda$. In this sense, our work extends [13] from $d = 1$ to an arbitrary $d$.

*Remark 2:* The helper can be implemented in a distributed fashion. For example, one can set a threshold so that any server whose queue length is larger than the threshold will constantly report its queue-length information to the helper. With a "good" choice of threshold, only a small fraction of servers will report their queue-length information, lowering the communication overhead of both servers and helper.

### 2.1.2 System State

It is straightforward to show that the queue-length process $(Q_1(t), Q_2(t), \ldots, Q_N(t))$ forms a Markov process for any fixed $N$, because all events (including task arrivals and task departures) are generated according to independent Poisson processes, and the change of queue lengths only depends on the current queue lengths.

Now, define

$$S_k^N(t) \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left(Q_i(t) \geq k\right), \ k \geq 0.$$

Here, $\mathbf{1}(\cdot)$ denotes the indicator function. Clearly, $S_k^N(t)$ represents the fraction of queues with at least $k$ tasks, which can be interpreted as the tail probability of a typical server having at least $k$ tasks. Also, $\sum_{k=1}^{\infty} S_k^N(t)$ is equal to the average queue length at time $t$. For these reasons, we will use the process $\left\{S_k^N(t)\right\}_{k=0}^{\infty}$ to represent the system state at time $t$. In fact, it is easy to show that $\left\{S_k^N(t)\right\}_{k=0}^{\infty}$ is a Markov process.

### 2.1.3 Main Results

We require several new definitions in order to formally state our main results. Consider the following iterations:

$$s_k = \lambda s_{k-1}^d - \epsilon \text{ for all } k \geq 1, \text{ given that } s_0 = 1. \tag{2.1}$$

Let $k^*(\lambda, \epsilon, d)$ be the smallest integer such that $s_{k^*(\lambda,\epsilon,d)} > 0$ and $s_{k^*(\lambda,\epsilon,d)+1} \leq 0$. Note that such $k^*(\lambda, \epsilon, d)$ always exists under our assumption of $\lambda < 1$ and $0 < \epsilon \ll \lambda$. Note also that, by construction, we have $1 = s_0 > s_1 > \cdots > s_{k^*(\lambda,\epsilon,d)} > 0$.

Define a new sequence $\{\tilde{s}_k\}_{k=0}^{\infty}$ as

$$\tilde{s}_k = \begin{cases} s_k, & \text{if } 0 \leq k \leq k^*(\lambda, \epsilon, d), \\ 0, & \text{if } k > k^*(\lambda, \epsilon, d). \end{cases} \tag{2.2}$$

Clearly, we have $1 = \tilde{s}_0 > \tilde{s}_1 > \cdots > \tilde{s}_{k^*(\lambda,\epsilon,d)} > 0$ and $\tilde{s}_k = 0$ for all $k > k^*(\lambda, \epsilon, d)$. As we will soon see, $\tilde{s}_k$ corresponds to the fraction of queues with at least $k$ tasks in the steady state for large systems. For convenience, let $\tilde{\mathbf{s}}$ denote the sequence $\{\tilde{s}_k\}_{k=0}^{\infty}$. That is,

$$\tilde{\mathbf{s}} \triangleq (\tilde{s}_0, \tilde{s}_1, \tilde{s}_2, \cdots).$$

Now, we are ready to introduce our main theorem. Let $\mathbf{S}^N(t)$ denote the Markov process $\left\{S_k^N(t)\right\}_{k=0}^{\infty}$. That is,

$$\mathbf{S}^N(t) \triangleq \left(S_0^N(t), S_1^N(t), S_2^N(t), \cdots\right).$$

If the process $\mathbf{S}^N(t)$ is ergodic, it has a unique steady-state distribution and we denote the steady-state distribution of $\mathbf{S}^N(t)$ by $\lim_{t\to\infty} \mathbf{S}^N(t)$.

*Theorem 2.1:* The Markov process $\mathbf{S}^N(t)$ is ergodic (and thus has a unique steady-state distribution), and

$$\lim_{N\to\infty} \lim_{t\to\infty} \mathbf{S}^N(t) = \tilde{\mathbf{s}}, \text{ in distribution,}$$

$$\lim_{t\to\infty} \lim_{N\to\infty} \mathbf{S}^N(t) = \tilde{\mathbf{s}}, \text{ in distribution.}$$

Theorem 2.1 states that, for all $N$, the Markov process $\mathbf{S}^N(t)$ has a unique steady-state distribution, and that the steady-state distribution concentrates on the sequence $\tilde{\mathbf{s}}$, as $N \to \infty$. In addition, it says that the process $\mathbf{S}^N(t)$ converges to a deterministic fluid limit as $N \to \infty$, and this fluid limit converges to a unique fixed point.

Since the system is fully symmetric in the steady state, Theorem 2.1 implies that the steady-

Table 2.1: Maximum queue length with $d = 2$

| $\lambda$ \ $\epsilon$ | $10^{-4}$ | 0.001 | 0.01 | 0.03 | 0.06 |
|---|---|---|---|---|---|
| 0.3 | 4 | 3 | 3 | 2 | 2 |
| 0.5 | 4 | 4 | 3 | 3 | 3 |
| 0.9 | 7 | 7 | 6 | 5 | 4 |
| 0.95 | 8 | 8 | 7 | 6 | 5 |
| 0.999 | 14 | 12 | 9 | 7 | 6 |

state distribution of the queue length of an arbitrary server in the large-system limit is given by

$$\pi_k = \tilde{s}_k - \tilde{s}_{k+1}.$$

In particular, we have $\pi_0 = 1 + \epsilon - \lambda$ and $\pi_k = 0$ for all $k > k^*(\lambda, \epsilon, d)$. That is, in the steady state, the fraction of idle servers doesn't depend on $d$, and the maximum queue size is bounded by $k^*(\lambda, \epsilon, d)$.

*Corollary 2.2:* The maximum steady-state queue size in the large-system limit under hybrid scheduling is bounded by $k^*(\lambda, \epsilon, d)$.

As a comparison, the steady-state distribution of the queue length of a server in the large-systems limit under the Po$d$ algorithm is given by

$$\pi_k = \lambda^{\frac{d^k-1}{d-1}} - \lambda^{\frac{d^{k+1}-1}{d-1}}. \tag{2.3}$$

Clearly, the maximum queue size is unbounded, since $\pi_k > 0$ for all $k$. Table 2.1 gives examples of the values for $k^*(\lambda, \epsilon, d)$ across a wide range of parameters. It appears that $k^*(\lambda, \epsilon, d)$ is rather small even when $\epsilon$ is as small as $10^{-3}$. This desirable property also holds for other values of $d$.

Next, we proceed to the expected queue length.

*Corollary 2.3:* The expected steady-state queue length in the large-systems limit under hybrid scheduling is

$$\tilde{s}_1 + \tilde{s}_2 + \cdots + \tilde{s}_{k^*(\lambda,\epsilon,d)} \tag{2.4}$$

which is bounded by

$$\sum_{k=1}^{k^*(\lambda,\epsilon,d)} \lambda^{\frac{d^k-1}{d-1}} - k^*(\lambda,\epsilon,d)\epsilon. \tag{2.5}$$

*Proof.* First, note that

$$\sum_{k=1}^{k^*(\lambda,\epsilon,d)} k\pi_k = \sum_{k=1}^{k^*(\lambda,\epsilon,d)} s_k.$$

Hence, the expected steady-state queue length is indeed given by (2.4).

Second, we will establish the upper bound (2.5). To this end, we consider a new sequence $s'_i = \lambda(s'_{i-1})^d$, with $s'_0 = 1$. In fact, this new sequence corresponds to the steady-state tail probabilities for the Po$d$ algorithm, and we have

$$s'_i = \lambda^{\frac{d^i-1}{d-1}}. \tag{2.6}$$

We can show that

$$s_i \leq s'_i - \epsilon \tag{2.7}$$

for all $i = 1, \ldots, k^*(\lambda,\epsilon,d)$. To see this, note that $s_1 = \lambda - \epsilon$ and $s'_1 = \lambda$. Hence, $s_1 \leq s'_1 - \epsilon$. Moreover, if $s_i \leq s'_i$, then we have

$$s_{i+1} = \lambda(s_i)^d - \epsilon \leq \lambda(s'_i)^d - \epsilon = s'_{i+1} - \epsilon.$$

Finally, combining (2.6) and (2.7), we prove the upper bound (2.5). □

As a comparison, the expected queue length in the large-systems limit under the Po$d$ algorithm is $\sum_{k=1}^{\infty} \lambda^{\frac{d^k-1}{d-1}}$ (which is strictly larger than the expected queue length under hybrid scheduling). Hence, hybrid scheduling indeed achieves shorter average response time than the Po$d$ algorithm in the large-system limit.

Table 2.2 compares the average queue length for Po2 ($\epsilon = 0$) and hybrid scheduling ($\epsilon > 0$). In particular, when $\epsilon = 0.05$, hybrid scheduling offers significant advantages over Po2, especially in the heavy-traffic regime (i.e., $\lambda \to 1$).

Table 2.2: Average queue length with $d = 2$

| $\lambda$ \ $\epsilon$ | 0 | 0.001 | 0.01 | 0.05 |
|---|---|---|---|---|
| 0.5 | 1.6328 | 1.6291 | 1.6001 | 1.5013 |
| 0.9 | 3.3527 | 3.3359 | 3.2006 | 2.7422 |
| 0.95 | 4.2139 | 4.1825 | 3.9320 | 3.1967 |
| 0.999 | 9.6430 | 8.6389 | 6.1667 | 3.9801 |

## 2.2 Mean-Field Analysis

In this section, we prove Theorem 2.1 by using the mean-field analysis. First, we will derive a (deterministic) fluid limit $\mathbf{s}(t)$ for the Markov process $\mathbf{S}^N(t)$ and show that $\mathbf{s}(t)$ converges to a unique fixed point $\tilde{\mathbf{s}}$ given by (2.1) and (2.2). Second, we will prove that the Markov process $\mathbf{S}^N(t)$ converges to the fluid limit $\mathbf{s}(t)$ as $N \to \infty$. These two steps prove that $\lim_{t\to\infty} \lim_{N\to\infty} \mathbf{S}^N(t) = \tilde{\mathbf{s}}$ in distribution. Finally, we will show that the steady-state distribution of the process $\mathbf{S}^N(t)$ concentrates on $\tilde{\mathbf{s}}$, proving $\lim_{N\to\infty} \lim_{t\to\infty} \mathbf{S}^N(t) = \tilde{\mathbf{s}}$ in distribution.

### 2.2.1 Fluid limit $\mathbf{s}(t)$ and its properties

First of all, we need to derive the fluid limit $\mathbf{s}(t)$ for the Markov process $\mathbf{S}^N(t)$. To this end, consider a dynamical system specified by the following rules:

(i) For $t = 0$,

$$1 = s_0(0) \geq s_1(0) \geq \cdots \geq s_m(0) > 0$$

and

$$s_k(0) = 0 \text{ for all } k > m.$$

That is, $s_m(0)$ is the "last" non-zero element of $\mathbf{s}(0)$.

(ii) For all $t \geq 0$,

$$s_0(t) = 1 \text{ and } 1 \geq s_k(t) \geq s_{k+1}(t) \geq 0$$

for all $k \geq 0$.

(iii) For almost all $t \geq 0$,

$$\dot{s}_k(t) = \lambda \left( s_{k-1}^d(t) - s_k^d(t) \right) - (s_k(t) - s_{k+1}(t)) - h_k(\mathbf{s}(t))$$

for all $k \geq 1$, where

$$h_k(\mathbf{s}(t)) = \begin{cases} 0, & s_{k-1}(t) = 0, s_k(t) = 0, \\ \min\{\lambda s_{k-1}^d(t), \epsilon\}, & s_{k-1}(t) > 0, s_k(t) = 0, \\ \max\{\epsilon - \lambda s_k^d(t), 0\}, & s_k(t) > 0, s_{k+1}(t) = 0, \\ 0, & s_k(t) > 0, s_{k+1}(t) > 0. \end{cases}$$

**Interpretation of the dynamical system.**

Conditions (i) and (ii) correspond to initial and boundary conditions, respectively. In fact, Condition (ii) is not necessary, since it is implied by Conditions (i) and (iii). (Here, we keep Condition (ii) to simplify the domain of the functions $\{h_k(\cdot)\}$.)

To understand Condition (iii), let us consider a finite system with $N$ servers. As we will soon see, when $N \to \infty$, the behaviour of the finite system can be "approximated" by the (deterministic) dynamical system.

Consider the Markov process $\mathbf{S}^N(t)$ whose current state at time $t$ is given by $\hat{\mathbf{s}} \triangleq (\hat{s}_0, \hat{s}_1, \cdots)$. Then, there exists some integer $m_t$ such that

$$1 = \hat{s}_0 \geq \hat{s}_1 \geq \cdots \geq \hat{s}_{m_t} > 0 \tag{2.8}$$

and $\hat{s}_k = 0$ for all $k > m_t$. Let

$$\mathbf{e}_k \triangleq (\underbrace{0, \cdots, 0}_{k \text{ times}}, 1, 0, \cdots).$$

Clearly, the next state of $\mathbf{S}^N(t)$ must be one of the following:

(a) $\hat{\mathbf{s}} + \frac{1}{N}\mathbf{e}_k$ for some $k$ where $1 \leq k \leq m_t + 1$;

(b) $\hat{\mathbf{s}} - \frac{1}{N}\mathbf{e}_k$ for some $k$ where $1 \leq k \leq m_t$.

Note that the transition rate from $\hat{\mathbf{s}}$ to $\hat{\mathbf{s}} + \frac{1}{N}\mathbf{e}_{m_t+1}$ is

$$\lambda N \hat{s}_{m_t}^d - \min\{\lambda N \hat{s}_{m_t}^d, \epsilon N\} \tag{2.9}$$

where the first term $\lambda N \hat{s}^d_{m_t}$ corresponds to the event that a task arrives at a server with $m_t$ tasks, and the second term $\min\{\lambda N \hat{s}^d_{m_t}, \epsilon N\}$ corresponds to the event that the helper prevents a server from having $m_t + 1$ tasks (by stealing a task if necessary). More specifically, a task arrives at a server with $m_t$ tasks *if and only if* a task arrives to the system (with rate $\lambda N$) and the Po$d$ scheduler selects $d$ servers all having $m_t$ tasks (with probability $\hat{s}^d_{m_t}$). This explains the first term. On the other hand, the helper needs to allocate a total rate of $\min\{\lambda N \hat{s}^d_{m_t}, \epsilon N\}$ in order to prevent a server from having $m_t + 1$ tasks. This explains the second term. Now, we take the limit as $N \to \infty$ and multiply the transition rate with the increment $\frac{1}{N}$, obtaining the term $\lambda \hat{s}^d_{m_t} - \min\{\lambda \hat{s}^d_{m_t}, \epsilon\}$, which explains the differential equation

$$\dot{s}_{m_t+1}(t) = \lambda s^d_{m_t}(t) - \min\{\lambda s^d_{m_t}(t), \epsilon\} \tag{2.10}$$

in the dynamical system when $k = m_t + 1$.

Using a similar argument, we can show that the transition rate from $\hat{\mathbf{s}}$ to $\hat{\mathbf{s}} + \frac{1}{N}\mathbf{e}_{m_t}$ is

$$\lambda N \left( \hat{s}^d_{m_t-1} - \hat{s}^d_{m_t} \right) \tag{2.11}$$

and the transition rate from $\hat{\mathbf{s}}$ to $\hat{\mathbf{s}} - \frac{1}{N}\mathbf{e}_{m_t}$ is

$$N \hat{s}_{m_t} + \max\{\epsilon N - \lambda N \hat{s}^d_{m_t}, 0\}. \tag{2.12}$$

Combining these two rates and taking the limit as $N \to \infty$, we obtain the term $\lambda \left( \hat{s}^d_{m_t-1} - \hat{s}^d_{m_t} \right) - \hat{s}_{m_t} - \max\{\epsilon - \lambda \hat{s}^d_{m_t}, 0\}$, which explains the differential equation

$$\dot{s}_{m_t}(t) = \lambda \left( s^d_{m_t-1}(t) - s^d_{m_t}(t) \right) - s_{m_t}(t) - \max\{\epsilon - \lambda s^d_{m_t}(t), 0\} \tag{2.13}$$

in the dynamical system when $k = m_t$.

Finally, when $k < m_t$, the evolution of $s_k(t)$ will not be affected by the helper, and so we have the same set of differential equations

$$\dot{s}_k(t) = \lambda \left( s^d_{k-1}(t) - s^d_k(t) \right) - (s_k(t) - s_{k+1}(t)) \tag{2.14}$$

as for the case of the Po*d* algorithm [4].

**Properties of the dynamical system.**

We characterize the behaviour of the dynamical system $\mathbf{s}(t)$. First, we can show that $\mathbf{s}(t)$ admits a unique solution. Hence, the dynamical system $\mathbf{s}(t)$ is indeed deterministic.

*Proposition 2.4:* Conditions (i), (ii), and (iii) determine a unique solution $\mathbf{s}(t)$.

*Proof.* Please see Appendix A for more details. □

Second, we can show that $\mathbf{s}(t)$ converges to a unique fixed point.

*Proposition 2.5:* The dynamic system $\mathbf{s}(t)$ has a unique fixed point $\tilde{\mathbf{s}}$ given by Equations (2.1) and (2.2).

*Proof.* The existence of a fixed point can be readily established by substituting the sequence $\tilde{\mathbf{s}}$. The uniqueness of the fixed point can be shown by adding all equations of Condition (iii) with $j \geq i$ to generate a recursive equation. □

*Theorem 2.6:* The dynamic system $\mathbf{s}(t)$ converges to the unique fixed point $\tilde{\mathbf{s}}$.

*Proof.* Please see Appendix B for more details. □

Therefore, every trajectory of the dynamical system (with initial condition (i)) converges to the fixed point $\tilde{\mathbf{s}}$.

### 2.2.2 The Markov process $\mathbf{S}^N(t)$ converges to the fluid limit $\mathbf{s}(t)$

Here, we will show that the Markov process $\mathbf{S}^N(t)$ converges to the dynamical system $\mathbf{s}(t)$. As such, $\mathbf{s}(t)$ is indeed the fluid limit of $\mathbf{S}^N(t)$.

We define a weighted $l_2$ norm $\|\cdot\|_\omega$ as

$$\|\mathbf{x}\|_\omega = \sqrt{\sum_{i=0}^{\infty} \frac{|x_i|^2}{2^i}}$$

where $\mathbf{x}$ is an infinite vector.

*Theorem 2.7:* Consider a sequence of systems $\mathbf{S}^N(t)$ indexed by $N$, the number of servers. Suppose that the (deterministic) initial conditions $\mathbf{s}^N(0) \to \mathbf{s}(0)$. Then, for any (finite) $T > 0$,

$$\lim_{N\to\infty} \Pr\left\{ \sup_{t\in[0,T]} \|\mathbf{S}^N(t) - \mathbf{s}(t)\|_\omega > \gamma \right\} = 0, \ \forall \gamma > 0.$$

*Proof.* The proof of Theorem 2.7 is in spirit similar to the proof of Theorem 6 in [13]. The main idea is to set up suitable fundamental processes, from which all other processes of interest, such as $\mathbf{S}^N(t)$, can be derived. Then the convergence of sample paths of $\mathbf{S}^N(t)$ can be transferred to the convergence of those fundamental processes, which is much easier to study.

Basically, there are three fundamental processes.

*Definition 2.8:*

1. Overall Process, $\{Z(t)\}_{t\geq 0}$, which is defined on a probability space $(\Omega_Z, \mathcal{F}_Z, \mathbb{P}_Z)$, is a Poisson process with rate $\lambda + \epsilon + 1$, where each jump marks the time when any event happens in the system.

2. Choose Process, $\{Y(n)\}_{n\in\mathbb{Z}_+}$, which is defined on a probability space $(\Omega_V, \mathcal{F}_V, \mathbb{P}_V)$, is a discrete time process. $Y(n)$ is independent and uniformly distributed in $[0, 1]$. This process is combined with current system state to determine which type of event happens.

3. Initial conditions, $\{S^{(0,N)}\}_{N\in\mathbb{N}}$ is a sequence of random variables defined on a common probability space $(\Omega_0, \mathcal{F}_0, \mathbb{P}_0)$.

We have the product space $(\Omega, \mathcal{F}, P) \triangleq (\Omega_Z \times \Omega_Y \times \Omega_0, \mathcal{F}_Z \times \mathcal{F}_Y \times \mathcal{F}_0, \mathbb{P}_Z \times \mathbb{P}_Y \times \mathbb{P}_0)$. To simplify notation, we set

$$Z(\omega, t) \triangleq Z(\omega_Z, t), \omega \in \Omega \text{ and } \omega = (\omega_Z, \omega_Y, \omega_0).$$

As the system is scaled by $N$, we can have a normalized event process with a unit $\frac{1}{N}$ for every step that

$$Z^N(\omega, t) \triangleq \frac{1}{N} Z(\omega, Nt).$$

Similar to the description of dynamic model, $S_i^N(t)$ can be decomposed with 4 parts:

$$S_i^N(t) = S_i^N(0) + A_i^N(t) - G_i^N(t) - C_i^N(t),$$

where $S_i^N(0)$ represents initial condition of $S_i^N$, $A_i^N(t)$, $G_i^N(t)$ and $C_i^N(t)$ represent cumulative arrival, departure and helper events, which all begin with zero. Thus, we can describe the process of $S_i^N(t)$ in a fixed sample path $\omega$ by the following mapping. In the whole process, we use $t_k$ represents the time when $k$th step occurs.

1. When $Y(\omega,k) \in \frac{\lambda}{1+\lambda+\epsilon}[(S_{j+1}^N(t_{k-1}))^d, (S_j^N(t_{k-1}))^d)$, a new task arrives at a server with $j$ tasks in step $k$, then $A_{j+1}^N(t)$ will be increased by $1/N$.

2. When $Y(\omega,k) \in \frac{\lambda}{1+\lambda+\epsilon} + \frac{1}{1+\lambda+\epsilon}[S_{j+1}^N(t_{k-1}), S_j^N(t_{k-1}))$, a task departure event happens in a server with j tasks in step $k$. If $j > 1$, $G_j^N(t)$ will be increased by $1/N$.

3. When $Y(\omega,k) \in \frac{1+\lambda}{1+\lambda+\epsilon} + \frac{\epsilon}{1+\lambda+\epsilon}[0,1)$, a helper event happens in step $k$. If there are tasks in the system in step $k-1$, $C_{i^*}^N(t)$, where $i^*$ is the longest queue length, will be increased by $1/N$.

After we couple sample paths with those fundamental processes, we show that $S^N(t)$ converges to certain Lipschitz continuous trajectory. Then, we justify that the derivative of the continuous trajectory is the same as the drift of our dynamic model at every point where the derivative exists. Those steps above complete the sample path tightness. Combined with the uniqueness of $s(t)$, it is straightforward to prove the convergence of $S^N(t)$ to the unique solution of $s(t)$ in probability. This completes the proof of Theorem 2.7. Please see Appendix C for more details. □

### 2.2.3 The steady-state distribution concentrates on $\tilde{\mathbf{s}}$

Here, we validate the other half of Theorem 2.1, which states that the Markov process $\mathbf{S}^N(t)$ has a unique steady-state distribution. Moreover, the steady-state distribution concentrates on the sequence $\tilde{\mathbf{s}}$, as $N \to \infty$.

*Proposition 2.9:* For any fixed $N$, $\mathbf{S}^N(t)$ is positive recurrent and the steady state distribution of $\mathbf{S}^N(t)$ , $\pi^N$, is tight in the sense that for every $\delta > 0$, there exists $M$ that $\pi^N(\sum_{i=0}^{\infty} S_i^N \leq M) \geq 1-\delta$.

*Proof.* Please see Appendix D for more details. □

Table 2.3: Theoretical analysis of (2.2) versus Simulations when $\epsilon = 0.05$.

| $(\lambda, d)$ | $\hat{s}_k$ | Analysis | Simulation $N = 100$ | Simulation $N = 500$ |
|---|---|---|---|---|
| $(0.95, 2)$ | $\hat{s}_1$ | 0.9000 | 0.8961 | 0.8969 |
| $(0.95, 2)$ | $\hat{s}_2$ | 0.7195 | 0.7155 | 0.7144 |
| $(0.95, 2)$ | $\hat{s}_3$ | 0.4418 | 0.4422 | 0.4371 |
| $(0.95, 2)$ | $\hat{s}_4$ | 0.1354 | 0.1490 | 0.1355 |
| $(0.95, 2)$ | $\hat{s}_5$ | 0 | 0.0125 | 0.0018 |
| $(0.8, 3)$ | $\hat{s}_1$ | 0.7500 | 0.7532 | 0.7510 |
| $(0.8, 3)$ | $\hat{s}_2$ | 0.2875 | 0.2991 | 0.2953 |
| $(0.8, 3)$ | $\hat{s}_3$ | 0 | 0.0084 | 0.0016 |

*Theorem 2.10:* As $N \to \infty$, the unique steady-state distribution $\pi^N$ concentrates on the sequence $\tilde{\mathbf{s}}$, which implies

$$\lim_{N\to\infty} \pi^N = \delta_{\tilde{\mathbf{s}}}, \text{ in distribution}$$

where $\delta_{\tilde{\mathbf{s}}}$ is the Dirac measure concentrated on $\tilde{\mathbf{s}}$.

The above theorem states that the steady-state distribution $\pi^N$ concentrates on a single state $\tilde{\mathbf{s}}$, as $N \to \infty$. The proof of Theorem 2.10 is exactly the same as the proof of Theorem 7 in [13]. The key step is based on the continuous test functions that verify the limit of $\pi^N$ to be equal to the fixed point of our dynamic model.

## 2.3 Simulations

In this section, we conduct extensive simulations to evaluate the performance of hybrid scheduling. First, we investigate the accuracy of our theoretical analysis for the tail probabilities $\{\hat{s}_k\}$. It turns out that our analysis result matches simulations well across a wide range of parameters, as long as the system has a moderate size (say $N = 500$). For example, Table 2.3 compares our theoretical analysis and simulations when $\epsilon = 0.05$ under two sets of configurations. We observe that the analysis is fairly accurate when $N = 500$.

Next, we compare the tail probabilities $\{\hat{s}_k\}$ for three strategies: Po2 scheduling ($d = 2, \epsilon = 0$), hybrid scheduling ($d = 2, \epsilon = 0.05$), and the single-helper strategy ($\epsilon = 0.05$) in [13]. We set $\lambda = 0.99$ and $N = 100$. As seen from Fig. 2.2, the tail probability $\hat{s}_k$ of hybrid scheduling decays

much faster than that of Po2 and reaches 0 when $k \geq 7$, whereas the tail probability of Po2 is always positive. Also, hybrid scheduling enjoys better tail probabilities than the single-helper strategy. In other words, hybrid scheduling can effectively reduce the tail probabilities even for systems with a relatively small size. When the system size is large (say, $N = 10000$), the same trend holds and the simulation is even closer to our theoretical prediction, as illustrated in Fig. 2.3.

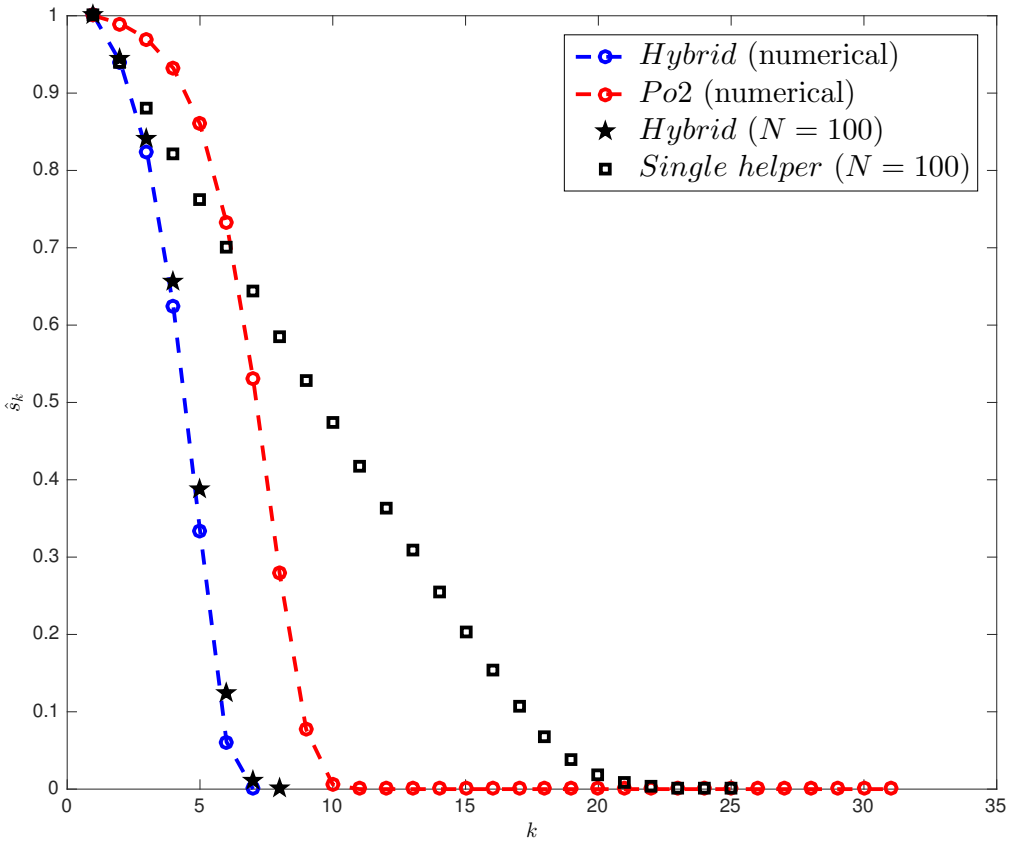


Figure 2.2: Comparison of tail probabilities for three strategies when$\lambda = 0.99$ and $N = 100$.

Then, we look at how tail probabilities $\{\hat{s}_k\}$ of hybrid scheduling evolve as the arrival rate $\lambda$ increases. We set $\epsilon = 0.05$, $d = 2$, and $N = 100$. We vary $\lambda$ from 0.5 to 0.95. As we can see from Fig. 2.4, when $\lambda \leq 0.8$, all the servers have less than four tasks in their queues. Even when $\lambda$ reaches 0.95, the fraction of servers with at least four tasks is just around 15% in our simulation.

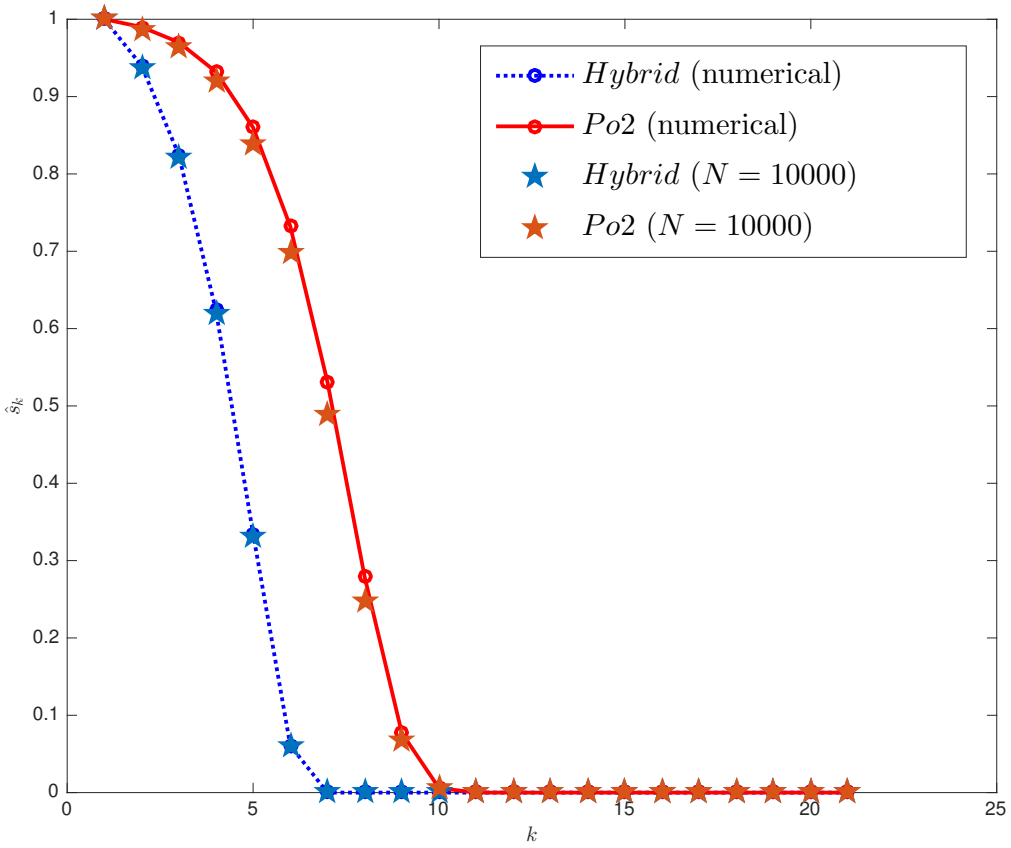


Figure 2.3: Comparison of tail probabilities for Po$d$ and hybrid strategy when $\lambda = 0.99$ and $N = 10000$.

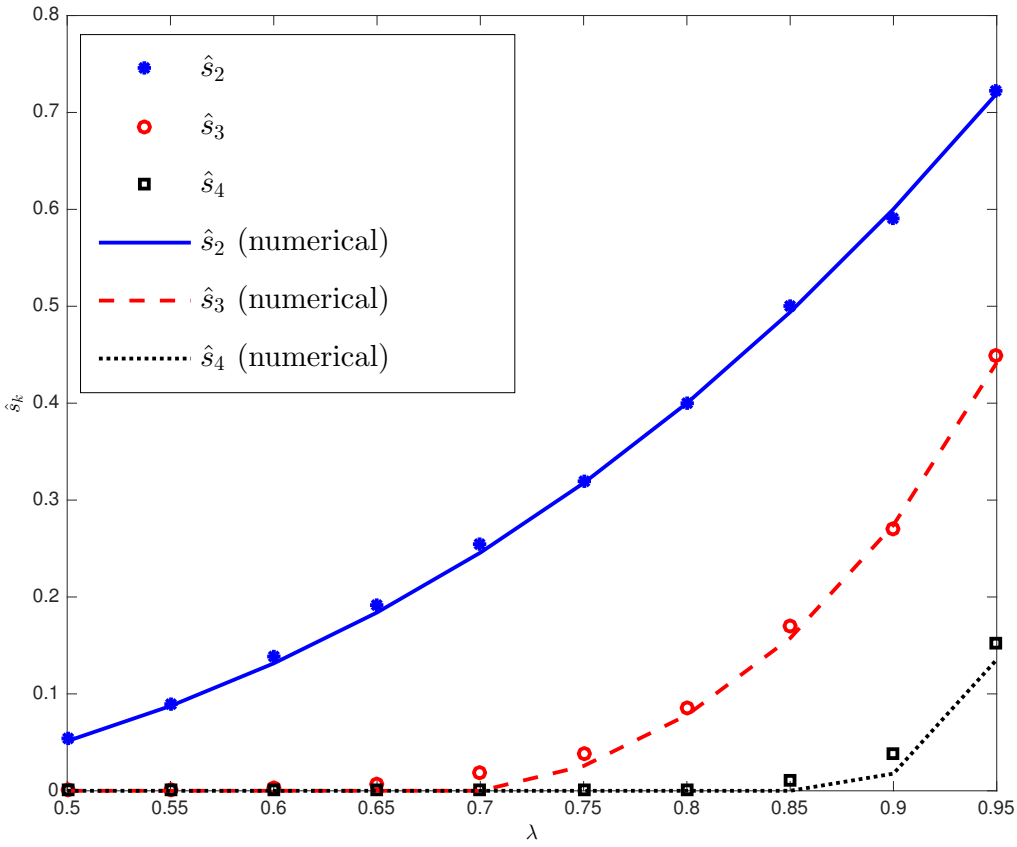


Figure 2.4: The impact of $\lambda$ on $\hat{s}_k$ with $\epsilon = 0.05$, $d = 2$ and $N = 100$.

Finally, we use some real-world trace obtained from Google's cluster data [15] to evaluate the performance of our hybrid scheduling. This data set is from a Borg cell for 7 hours period, in which there are four types of jobs. Here we focus on Type-3 jobs, which consist of 3187 jobs and 14115 tasks. In particular, we obtain task arrival times and task processing times from the trace, and use such information to replace the assumption of Poisson arrivals and exponential service-time distribution. For the following simulations, we set $\epsilon = 0.05$, $N = 200$ and $d = 2$. According to Fig. 2.5, the cumulative distribution function of job response time of hybrid algorithm implies smaller average response time than that of the Po$d$ algorithm. Intuitively, with the help of helper, it may shorten the latest task's response time in a job, which decreases the response time for this job.

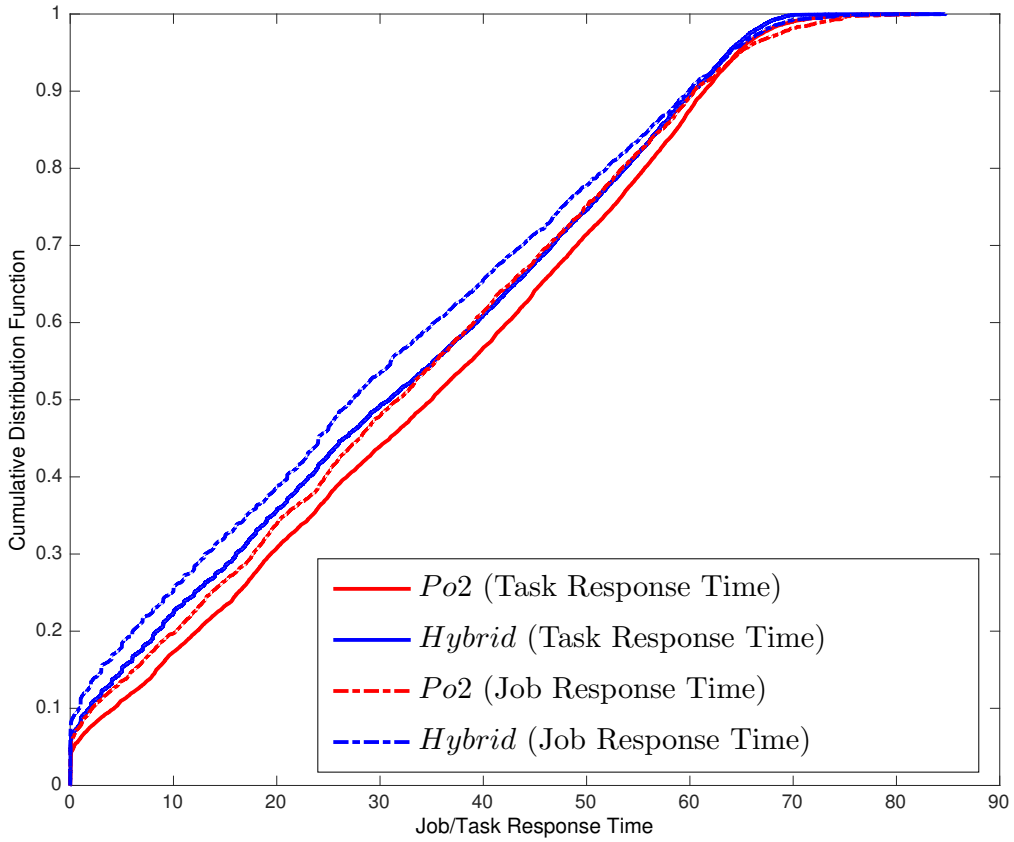


Figure 2.5: Task/job response time with $\epsilon = 0.05$, $d = 2$ and $N = 200$.

# Chapter 3

# Analysis of JIQ Algorithms

In this chapter, we take a further step in understanding the performance of distributed JIQ, applying a mean-field analysis to derive semi-closed form expressions of the stationary tail distribution and the expected response time for distributed JIQ. Then, we propose a new variant of JIQ called JIQ-Po$d$ that strictly outperforms Po$d$. JIQ-Po$d$ enjoys the best of both worlds. We also quantify the improvements of JIQ-Po$d$ over JIQ and Po$d$ in the large-system limit.

## 3.1 System Model and Main Results

### 3.1.1 Distributed JIQ Algorithm

Consider a system with $N$ identical servers and $M$ schedulers. Each scheduler is equipped with an I-queue that stores a list of idle servers (which will be specified later). The system evolves as follows.

- *Task arrivals*: Tasks arrive at the system according to a Poisson process of rate $\lambda N$, where $\lambda < 1$, and are sent to a scheduler uniformly at random. Thus, each scheduler observes a Poisson arrival process of rate $\lambda N/M$.
- *Schedulers with I-queues*: Each scheduler has an I-queue, which maintains a list of idle servers. Upon a task arrival, each scheduler checks its I-queue and assigns the task to a server according to the following rule: If the I-queue is non-empty, the scheduler selects an idle server uniformly at random from its I-queue. If the I-queue is empty, the scheduler selects a server uniformly at random from all the servers.
- *Servers*: Each server has an infinite buffer and processes tasks in a first-in first-out (FIFO)

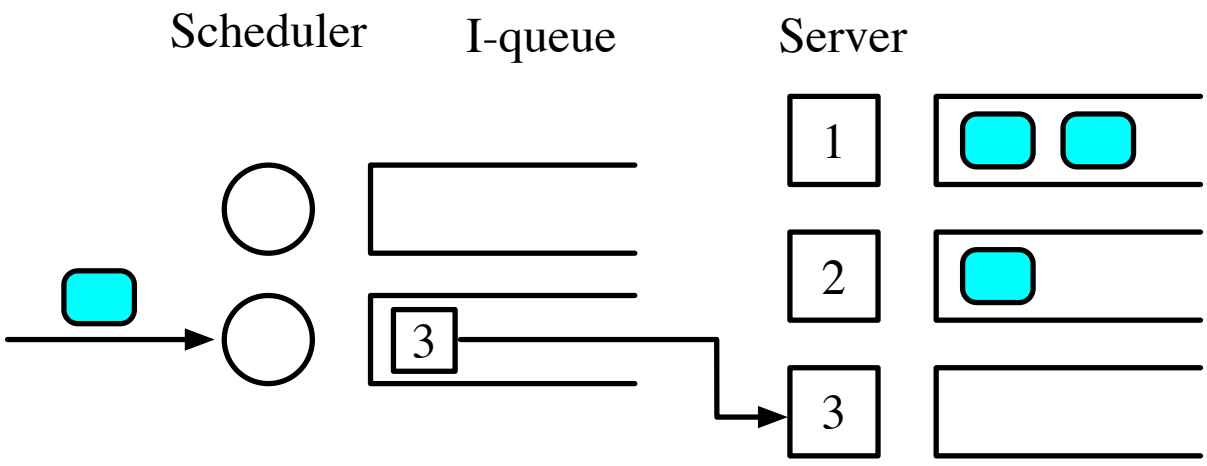


Figure 3.1: Server 3 is selected by Scheduler 2 and leaves its I-queue.

manner. The task processing times are exponentially distributed with mean 1. Whenever a server becomes idle, it joins an I-queue selected uniformly at random. Whenever an idle server becomes busy, it leaves its associated I-queue in one of the following two ways:

1. If it is selected by a scheduler with a non-empty I-queue, then it simply leaves the I-queue, as shown in Figure 3.1.
2. If it is selected by a scheduler with an empty I-queue, then it has to inform its associated I-queue by sending a "delete request" message, as shown in Figure 3.2.

We note that some distributed JIQ algorithm doesn't use the "delete request" messages (e.g., in [7]), allowing I-queues having non-idle servers. Although it reduces the communication overhead, it complicates the theoretical analysis. As we will see in Section 3.3.2, such extra communication overhead is acceptable.

### 3.1.2 Distributed JIQ-Po$d$ Algorithm

The distributed JIQ algorithm described above doesn't always outperform the Po$d$ algorithm, especially under heavy workload where most I-queues are empty. To address this issue, we propose a new variant of JIQ, namely JIQ-Po$d$, which combines the advantages of JIQ and Po$d$. It works as follows. Whenever a new task is sent to a scheduler with an empty I-queue, the scheduler probes $d$ servers uniformly at random and assigns the task to the least loaded

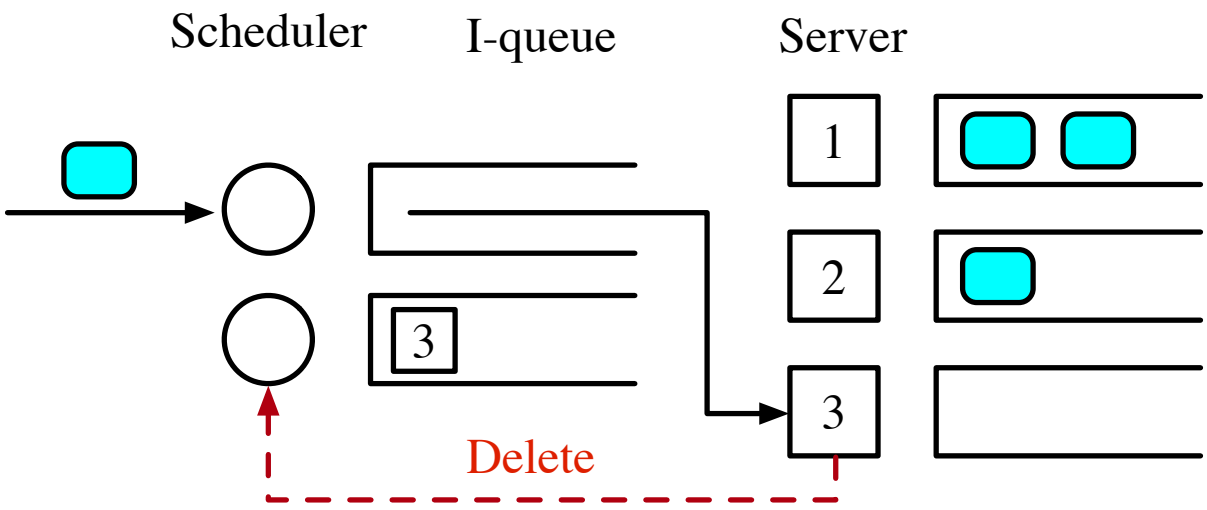


Figure 3.2: Server 3 is selected by Scheduler 1 and sends a "delete request" message to Scheduler 2.

one, as shown in Figure 3.3. That is, each scheduler with an empty I-queue is applying the Po$d$ strategy. All other steps remain the same. Clearly, when $d = 1$, our JIQ-Po$d$ algorithm reduces to the distributed JIQ algorithm.

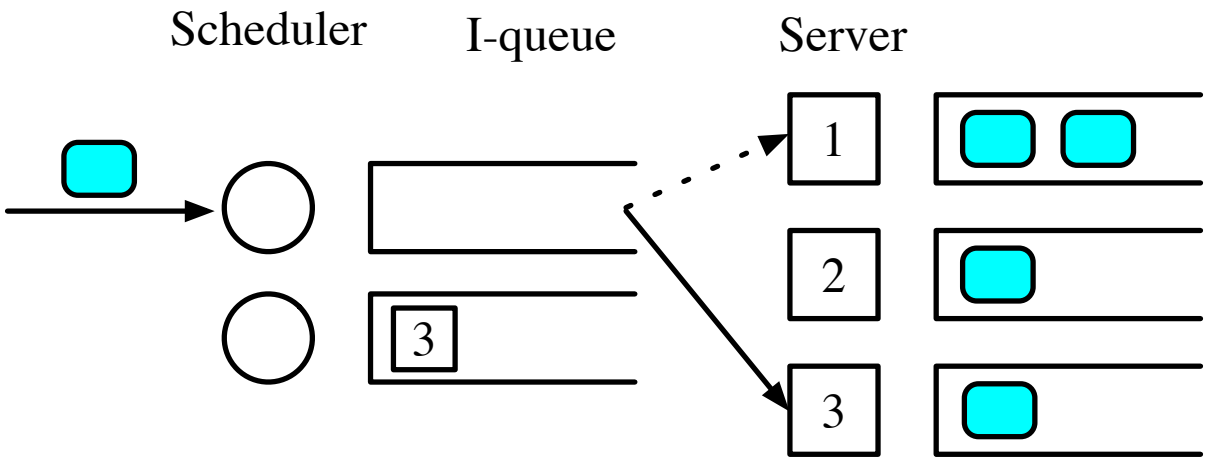


Figure 3.3: Scheduler 1 probes Server 1 and Server 3, and assigns a new task to Server 3.

### 3.1.3 Main Results

Table 3.1 presents our main results that characterize the stationary tail distribution and the expected task response time in the large-system limit (i.e., $N \to \infty$ and $M \to \infty$ with the ratio $r = N/M$ fixed), where $\hat{p}_0$ is some parameter in $(0, 1)$ (which will be specified later). The

Table 3.1: Summary of queue length distribution under JIQ, JIQ-P$o d$ and P$o d$.

| | JIQ | JIQ-P$o d$ | P$o d$ |
|---|---|---|---|
| Tail distribution of server ($\hat{s}_i$ for $i \geq 1$) | $\hat{p}_0^{i-1}\lambda^i$ | $\lambda^{\frac{d^i-1}{d-1}}\hat{p}_0^{\frac{d^{i-1}-1}{d-1}}$ | $\lambda^{\frac{d^i-1}{d-1}}$ |
| Expected task response time ($T(\lambda)$) | $1+\hat{p}_0\sum_{i=1}^{\infty}\hat{s}_i$ | $1+\hat{p}_0\sum_{i=1}^{\infty}(\hat{s}_i)^d$ | $1+\sum_{i=1}^{\infty}(\hat{s}_i)^d$ |

stationary tail distribution $\hat{s}_i$ is the fraction of servers having no less than $i$ tasks in their task queues. (Note that $\hat{s}_0$ is always equal to 1, and that the $\{\hat{s}_i\}$ are non-increasing.) The smaller $\hat{s}_i$ is, the shorter the task delay. The expected task response time $T(\lambda)$ measures the average completion time for a task in steady state.

First, we observe that JIQ-P$o d$ gives the best tail distribution $\hat{s}_i$. Compared to P$o d$, JIQ-P$o d$ has an additional factor of $\hat{p}_0^{\frac{d^{i-1}-1}{d-1}} < 1$, since $\hat{p}_0 \in (0,1)$. For instance, when $d=2$, $i=2$ and $p_0 = 0.5$, such factor equals to 0.5. Compared to JIQ, JIQ-P$o d$ has larger exponents of $\lambda$ and $\hat{p}_0$. For example, when $d=2$ and $i=3$, the exponent of $\lambda$ under JIQ-P$o d$ is $\frac{d^i-1}{d-1} = 7 > i = 3$, and the exponent of $\hat{p}_0$ under JIQ-P$o d$ is $\frac{d^{i-1}-1}{d-1} = 3 > i-1 = 2$.

Second, we observe that JIQ-P$o d$ achieves the shortest expected task response time $T(\lambda)$. Compared to P$o d$, JIQ-P$o d$ has an additional factor of $\hat{p}_0 < 1$. Compared to JIQ, JIQ-P$o d$ has larger exponents of $\hat{s}_i$. To better illustrate the advantage of JIQ-P$o d$, we provide a concrete numerical example in Figure 3.4, which shows that, when $\lambda = 0.98$, $T(\lambda)$ of JIQ-P$o d$ is only around 2.6, whereas $T(\lambda)$ of JIQ and P$o d$ are 5.3 and 4.5, respectively.

## 3.2 Mean-Field Analysis

In this section, we will use the mean-field analysis to study the stationary distributions of the queue lengths under JIQ and JIQ-P$o d$, as well as their corresponding expected task response times. The underlying assumptions behind the mean-field analysis will be validated through simulations in Sec. 3.3.1. In fact, these assumptions can be rigorously validated using proof techniques such as Kurtz's theorem, which is beyond the scope of this thesis[4].

[4]The evolution of the system state can be characterized by a density-dependent continuous-time Markov chain $\widetilde{\mathbf{Q}}^{(N)}(t)$. This allows us to apply Kurtz's theorem to rigorously validate the use of the mean-field analysis. Most of the proof steps towards this direction are standard except for the step showing the global convergence of the

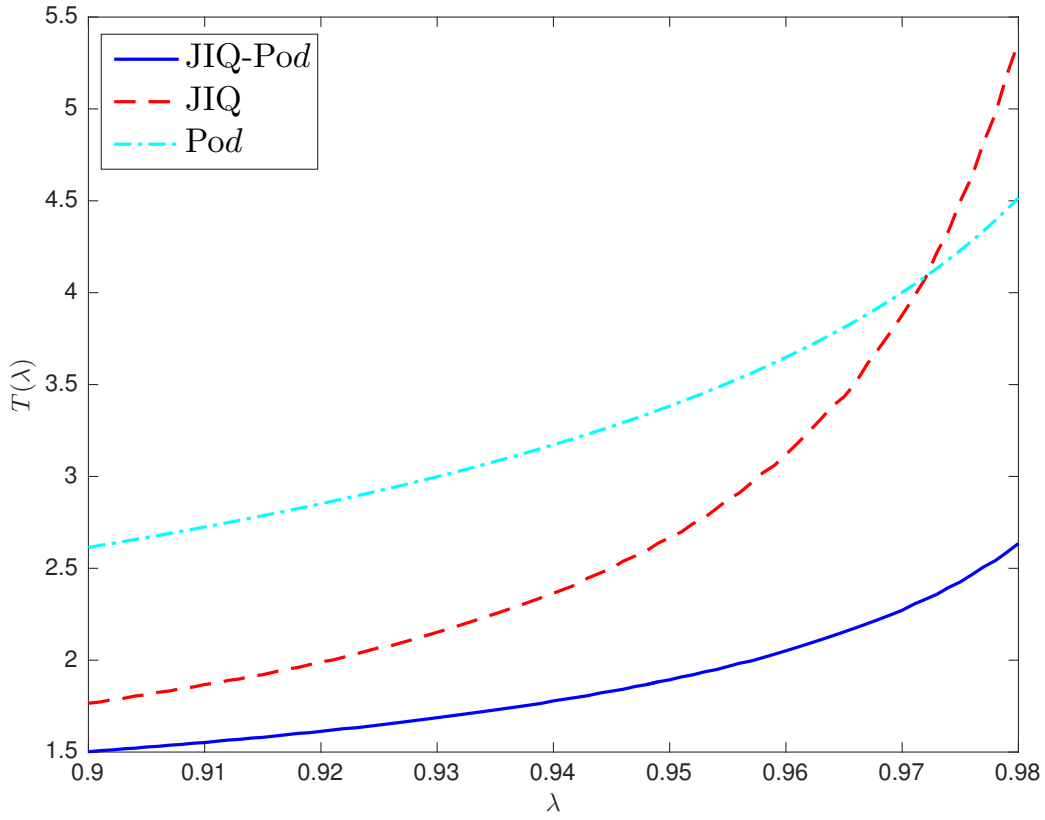


Figure 3.4: Comparison of the expected task response time among JIQ, JIQ-P$od$ and P$od$, when $r = 10$ and $d = 2$.

Let $\left(X_i^{(N)}(t), Y_i^{(N)}(t)\right)$ denote the *state* of the $i$th server at time $t$ in a system of $N$ servers and $M$ I-queues, where $X_i^{(N)}(t)$ is the queue length of the $i$th server at time $t$ and $Y_i^{(N)}(t)$ is the index of the associated I-queue. If the $i$th server doesn't belong to any I-queue at time $t$, we set $Y_i^{(N)}(t) = 0$. It is easy to check that $\left\{\left(X_i^{(N)}(t), Y_i^{(N)}(t)\right)\right\}_{i=1}^{N}$ forms an irreducible, aperiodic, continuous-time Markov chain under our system model. Moreover, the following theorem shows that this Markov chain is positive recurrent and thus has a unique stationary distribution.

*Theorem 3.1:* The Markov Chain $\left\{\left(X_i^{(N)}(t), Y_i^{(N)}(t)\right)\right\}_{i=1}^{N}$ under the JIQ algorithm is positive recurrent.

*Proof.* We first construct a potential function and then apply the Foster-Lyapunov theorem. Please see Appendix E for more details. □

In order to conduct the mean-field analysis, we need to introduce a new representation of the system state. Let $Q_i^{(N)}(t)$ denote the number of servers with $i$ tasks at time $t$. Let $Q_{(0,j)}^{(N)}(t)$ denote the number of idle servers that belong to I-queues of size $j$ at time $t$. Then, the system state at time $t$ is

$$\mathbf{Q}^{(N)}(t) = \left\{Q_{(0,1)}^{(N)}(t), Q_{(0,2)}^{(N)}(t), \cdots, Q_1^{(N)}(t), Q_2^{(N)}(t), \cdots\right\}.$$

One can verify that $\mathbf{Q}^{(N)}(t)$ also forms a continuous-time Markov chain under our system model, because the individual servers (or I-queues) of the same queue-length are *indistinguishable* for system evolution. In other words, our new Markov chain $\mathbf{Q}^{(N)}(t)$ captures the "essential" information of our original Markov chain $\left\{\left(X_i^{(N)}(t), Y_i^{(N)}(t)\right)\right\}_{i=1}^{N}$. In particular, if our original Markov chain has a unique stationary distribution, so does our new Markov chain.

For convenience, we also introduce a normalized version of $\mathbf{Q}^{(N)}(t)$ defined as

$$\widetilde{\mathbf{Q}}^{(N)}(t) = \frac{1}{N}\mathbf{Q}^{(N)}(t).$$

Note that $\tilde{Q}_{(0,j)}^{(N)}(t)$ is the fraction of servers that belong to I-queues of size $j$ at time $t$, and $\tilde{Q}_i^{(N)}(t)$ is the fraction of servers with $i$ tasks at time $t$. Clearly, $\widetilde{\mathbf{Q}}^{(N)}(t)$ is also positive

underlying ordinary differential equations.

recurrent and has a unique stationary distribution. In addition, we can show that $\widetilde{\mathbf{Q}}^{(N)}(t)$ is density dependent.

The mean-field analysis proceeds as follows. We assume that the $N$ servers are in the steady state. We also assume that the states of these servers are identically and independently distributed (i.i.d.). This i.i.d. assumption will be validated later through simulations. We now consider the state evolution of one server in the system under the i.i.d. assumption. Note that the possible server states are from the set

$$\{(0,1),(0,2),\ldots,1,2,\ldots\}$$

where the state-$(0,j)$ means that the server is idle and belongs to an I-queue of size $j$, and the state-$i$ means that the server has $i$ tasks in its queue. Let

$$\mathbf{q} = \left\{q_{(0,1)}, q_{(0,2)}, \cdots, q_1, q_2, \cdots\right\}$$

denote the stationary distribution of the server state. (Note that $\mathbf{q}$ is unique because $\widetilde{\mathbf{Q}}^{(N)}(t)$ is positive recurrent.) Then, by the Strong Law of Large Numbers, $q_{0,j}$ can be interpreted as the fraction of servers belonging to an I-queue of size $j$, and $q_i$ can be interpreted as the fraction of servers with $i$ tasks in the large-system limit. This means that the stationary distribution of $\widetilde{\mathbf{Q}}^{(N)}(t)$ "concentrates" on $\mathbf{q}$ as $N \to \infty$.

We are now ready to derive the stationary distributions under JIQ and JIQ-P$o d$ in the large-system limit.

### 3.2.1 The Stationary Distribution Under JIQ

In this subsection, we will derive the stationary distribution of one server in the system under JIQ. The i.i.d. assumption described above allows us to obtain the transition rates for the state evolution of the server. In fact, this assumption holds asymptotically in the large-system limit. In other words, the stationary distribution derived here is sufficiently accurate for large-scale systems.

To derive the transition rates, we need some additional notations. Let $p_i$ be the fraction of

I-queues of size $i$ in the large-system limit. Then we have

$$p_i = \begin{cases} \frac{rq_{(0,i)}}{i}, & \text{if } i \geq 1, \\ 1 - \sum_{j=1}^{\infty} \frac{rq_{(0,j)}}{j}, & \text{if } i = 0 \end{cases}$$

where the first equation follows from the fact that the number of servers in state-$(0, i)$ is equal to $i$ times the number of I-queues of size $i$, and the second equation follows from the fact that $\sum_{j=0}^{\infty} p_j = 1$.

We now derive the transition rates for the state evolution of a single server as follows:

- $r_{i,i-1} = 1$ for $i \geq 2$.

  The processing rate of a task is exponentially distributed with mean 1.

- $r_{i-1,i} = \lambda p_0$ for $i \geq 2$.

  The task arrival rate is $\lambda N$, the probability of joining an empty I-queue is $p_0$, and the probability of selecting the target server over all servers is $\frac{1}{N}$.

- $r_{1,(0,j)} = p_{j-1}$ for $j \geq 1$.

  The processing rate of a task is exponentially distributed with mean 1, and the probability of joining an I-queue of size $j - 1$ is $p_{j-1}$.

- $r_{(0,j),1} = \lambda(p_0 + \frac{r}{j})$ for $j \geq 1$.

  The task arrival rate is $\lambda N$. There are two events leading to this state change because of a newly arrival task. The first event is that the new task is routed to an empty I-queue and then directed to the target server. The probability of this event is $p_0 \cdot \frac{1}{N}$. The second event is that the new task is routed to the I-queue associated with the target server and then directed to the target server. The probability of this event is $\frac{1}{M} \cdot \frac{1}{j} = \frac{r}{N} \cdot \frac{1}{j}$. Hence, the transition rate is $\lambda N \left( p_0 \cdot \frac{1}{N} + \frac{r}{N} \cdot \frac{1}{j} \right)$.

- $r_{(0,j-1),(0,j)} = rq_1$ for $j \geq 2$.

  The generating rate of idle servers is $q_1 N$, and the probability of selecting the I-queue associated with the target server is $\frac{1}{M}$.

- $r_{(0,j),(0,j-1)} = \lambda(j-1)(p_0 + \frac{r}{j})$ for $j \geq 2$.

The task arrival rate is $\lambda N$. There are two events resulting in this state change. The first event is that the new task is routed to an empty I-queue and then directed to an idle server having the same I-queue as the target server. The probability of this event is $p_0 \cdot \frac{j-1}{N}$. The second event is that the new task is routed to the I-queue associated with the target server and then directed to another idle server. The probability of this event is $\frac{1}{M} \cdot \frac{j-1}{j}$. Hence, the transition rate is $\lambda N \left( p_0 \cdot \frac{j-1}{N} + \frac{r}{N} \cdot \frac{j-1}{j} \right)$.

Based on the above transition rates, one can easily write down the local balance equations as

$$\begin{cases} q_i r_{i,i-1} = q_{i-1} r_{i-1,i}, \text{ for } i \geq 2, \\ q_{(0,j)} r_{(0,j),1} = q_1 r_{1,(0,j)}, \text{ for } j \geq 1, \\ q_{(0,j)} r_{(0,j),(0,j-1)} = q_{(0,j-1)} r_{(0,j-1),(0,j)}, \text{ for } j \geq 2. \end{cases} \tag{3.1}$$

The following theorem computes the stationary distribution of the state of a single server in the large-system limit by finding a particular distribution that satisfies the local balance equations (3.1).

*Theorem 3.2:* The stationary distribution of the state of a single server under JIQ in the large-system limit is

$$\begin{cases} \hat{q}_{(0,i)} = \dfrac{r^{i-1}(1-\lambda\hat{p}_0)^i}{\prod_{j=1}^{i}(r+j\hat{p}_0)} i\hat{p}_0, \text{ for } i \geq 1, \\ \hat{q}_i = \hat{p}_0^{i-1}\lambda^i(1-\hat{p}_0\lambda), \text{ for } i \geq 1, \end{cases} \tag{3.2}$$

where $\hat{p}_0$ is the unique solution to the following equation

$$p_0 + \sum_{i=1}^{\infty} \frac{r^i(1-\lambda p_0)^i}{\prod_{j=1}^{i}(r+jp_0)} p_0 = 1 \tag{3.3}$$

over the interval $(0, 1)$.

*Proof.* We will prove Theorem 3.2 through two steps. First, we will show that (3.3) indeed has a unique solution. Second, we will show that the distribution $\hat{\mathbf{q}}$ satisfies the local balance equations (3.1).

To prove the first step, we define a function

$$f(p_0) = p_0 + \sum_{i=1}^{\infty} \frac{r^i(1-\lambda p_0)^i}{\prod_{j=1}^{i}(r+jp_0)} p_0.$$

As we will show in Appendix F, the function $f(p_0)$ is differentiable and monotonically increasing over the interval $[0,1]$. Notice that $f(0) = 0$ and $f(1) > 1$ as $\lambda < 1$. Hence, by the Intermediate Value Theorem, the equation $f(p_0) = 1$ has a unique solution over the interval $(0,1)$.

To prove the second step, we only need to verify that the distribution $\hat{\mathbf{q}}$ (constructed in (3.2)) satisfies the local balance equations (3.1). □

*Remark 3:* It is technically involved to show that $f(p_0)$ is differentiable and monotonically increasing over $(0,1)$, partly because $f(p_0)$ is the sum of an infinite series. To address this difficulty, we first show the uniform convergence of $f(p_0)$ and then bound its derivative by making a particular use of the structure of $f(p_0)$.

*Remark 4:* We observe that a truncated version of $f(p_0)$

$$F_n(p_0) = p_0 + \sum_{i=1}^{n} \frac{r^i(1-\lambda p_0)^i}{\prod_{j=1}^{i}(r+jp_0)} p_0$$

is very close to $f(p_0)$ and, in particular, is monotonically increasing, as long as $n$ is reasonably large. Hence, we can apply a simple binary search to solve the equation $F_n(p_0) = 1$ in order to numerically compute the value of $\hat{p}_0$.

*Corollary 3.3:* In the large-system limit, the stationary tail distribution under JIQ is

$$\hat{s}_i = \begin{cases} 1, \text{ for } i = 0, \\ \hat{p}_0^{i-1}\lambda^i, \text{ for } i \geq 1 \end{cases}$$

and the expected task response time under JIQ is $\frac{1}{1-\hat{p}_0\lambda}$.

*Proof.* The stationary tail distribution

$$\hat{s}_i = \sum_{j=i}^{\infty} \hat{q}_j = \sum_{j=i}^{\infty} \hat{p}_0^{j-1} \lambda^j (1 - \hat{p}_0 \lambda) = \hat{p}_0^{i-1} \lambda^i.$$

This proves the first part. To compute the expected task response time, we consider the following two cases.

1. A new task is sent to a non-empty I-queue (with probability $1 - \hat{p}_0$). The expected task response time in this case is 1.

2. A new task is sent to an empty I-queue (with probability $\hat{p}_0$). The expected task response time in this case is $\sum_{i=0}^{\infty} (i+1)\hat{q}_i$.

Combining the above two cases, the expected task response time is

$$(1 - \hat{p}_0) + \hat{p}_0 \sum_{i=0}^{\infty} (i+1)\hat{q}_i = \frac{1}{1 - \hat{p}_0 \lambda}.$$

This completes the second part. □

### 3.2.2 The Stationary Distribution Under JIQ-Po$d$

Similar to our previous analysis for JIQ, we can derive the transition rates for JIQ-Po$d$ as follows:

$$\begin{cases} r_{i,i-1} = 1, \text{ for } i \geq 2, \\ \lambda p_0 \left[ \dfrac{\left( \sum_{j=i-1}^{\infty} q_j \right)^d - \left( \sum_{j=i}^{\infty} q_j \right)^d}{q_{i-1}} \right], \text{ for } i \geq 2, \\ r_{1,(0,j)} = p_{j-1}, \text{ for } j \geq 1, \\ r_{(0,j),1} = \lambda \left[ p_0 \dfrac{1 - \left( \sum_{j=i-1}^{\infty} q_j \right)^d}{q_0} + \frac{r}{j} \right], \text{ for } j \geq 1, \\ r_{(0,j-1),(0,j)} = r q_1, \text{ for } j \geq 2. \\ r_{(0,j),(0,j-1)} = \lambda(j-1) \left[ p_0 \dfrac{1 - \left( \sum_{j=i-1}^{\infty} q_j \right)^d}{q_0} + \frac{r}{j} \right], \text{ for } j \geq 2. \end{cases}$$

The local balance equations are the same as those in Equation (3.1), based on which we can calculate the stationary distribution of the status of single server in the large-system limit.

*Theorem 3.4:* The stationary distribution of the state of a single server under JIQ-Po$d$ in the large-system limit is

$$\begin{cases} \hat{q}_{(0,i)} = \frac{r^{i-1}(1-\lambda^d \hat{p}_0)^i}{\prod_{j=1}^{i}\left[r+j\hat{p}_0(\frac{1-\lambda^d}{1-\lambda})\right]} i\hat{p}_0, \text{ for } i \geq 1, \\ \hat{q}_i = \lambda^{\frac{d^i-1}{d-1}} \hat{p}_0^{\frac{d^{i-1}-1}{d-1}}(1-\lambda^{d^i}\hat{p}_0^{d^{i-1}}), \text{ for } i \geq 1 \end{cases} \tag{3.4}$$

where $\hat{p}_0$ is the unique solution to the following equation

$$p_0 + \sum_{i=1}^{\infty} \frac{r^i(1-\lambda p_0)^i}{\prod_{j=1}^{i}(r+jp_0)} p_0 = 1 \tag{3.5}$$

over the interval $(0, 1)$.

*Proof.* The proof is omitted here as it's almost the same as Appendix F. □

*Corollary 3.5:* In the large-system limit, the stationary tail distribution under JIQ-Po$d$ is

$$\hat{s}_i = \begin{cases} 1, \text{ for } i = 0, \\ \lambda^{\frac{d^i-1}{d-1}} \hat{p}_0^{\frac{d^{i-1}-1}{d-1}}, \text{ for } i \geq 1 \end{cases}$$

and the expected task response time under the JIQ-Po$d$ algorithm is $1 + \hat{p}_0 \sum_{i=0}^{\infty} (\hat{s}_i)^d$.

*Proof.* The proof is omitted here as it's almost the same as that of Corollary 3.3. □

Under the JIQ-Po$d$ algorithm, the tail distribution of server queue length is lighter. Hence, a task is processed faster.

## 3.3 Simulations

In this section, we validate our theoretical results, measure the impact of "delete request" messages and compare various JIQ algorithms in finite-sized systems. In all of our simulations,

Table 3.2: Theoretical analysis of (3.2) and (3.4) versus Simulations for JIQ and JIQ-P$o d$ when $r = 10$ and $\lambda = 0.9$.

| | Analysis (JIQ) | $n = 500$ (JIQ) | $n = 1000$ (JIQ) | Analysis (JIQ-P$o d$) | $n = 500$ (JIQ-P$o d$) | $n = 1000$ (JIQ-P$o d$) |
|---|---|---|---|---|---|---|
| $\hat{q}_{(0,1)}$ | 0.0260 | 0.0259 | 0.0261 | 0.0267 | 0.0266 | 0.0266 |
| $\hat{q}_{(0,2)}$ | 0.0269 | 0.0266 | 0.0268 | 0.0281 | 0.0279 | 0.0278 |
| $\hat{q}_{(0,3)}$ | 0.0200 | 0.0200 | 0.0199 | 0.0206 | 0.0203 | 0.0204 |
| $\hat{q}_{(0,4)}$ | 0.0126 | 0.0128 | 0.0127 | 0.0125 | 0.0124 | 0.0125 |
| $\hat{q}_{(0,5)}$ | 0.0072 | 0.0074 | 0.0072 | 0.0067 | 0.0068 | 0.0066 |
| $\hat{q}_1$ | 0.5097 | 0.5071 | 0.5089 | 0.5571 | 0.5523 | 0.5532 |
| $\hat{q}_2$ | 0.2210 | 0.2206 | 0.2207 | 0.2931 | 0.2931 | 0.2939 |
| $\hat{q}_3$ | 0.0958 | 0.0963 | 0.0960 | 0.0487 | 0.0529 | 0.0517 |
| $\hat{q}_4$ | 0.0416 | 0.0424 | 0.0418 | 0.0010 | 0.0016 | 0.0014 |
| $\hat{q}_5$ | 0.0180 | 0.0186 | 0.0183 | 0.0000 | 0.0000 | 0.0000 |

we start from an empty system with the number of servers, $N$, set to be either 500 or 1000. The number of I-queues, $M$, is chosen as $M = \frac{N}{r}$, where $r$ will be specified later. The simulation results are based on the average of 10 runs, where each run lasts for 10,000 unit times.

### 3.3.1 Validation of the Mean-field analysis Results

In this subsection, we evaluate our analysis results for the stationary distribution and the expected task response time.

Table 3.2 compares the stationary distributions for JIQ and JIQ-P$o d$ obtained from theoretical analysis and simulation. We observe that the larger the system size, the higher the accuracy. When the server size is only 500, the maximum relative error rate under JIQ is as small as 3.3% for $\hat{q}_5$.

Figure 3.5 and Figure 3.6 show the tail distribution of JIQ and JIQ-P$o d$ obtained from theoretical analysis and simulation. we observe that the tail distribution $s_i$ increase with the growth of arrival rate. It also shows that our JIQ-P$o d$ algorithm enjoys lighter $s_i$ than the JIQ algorithm.

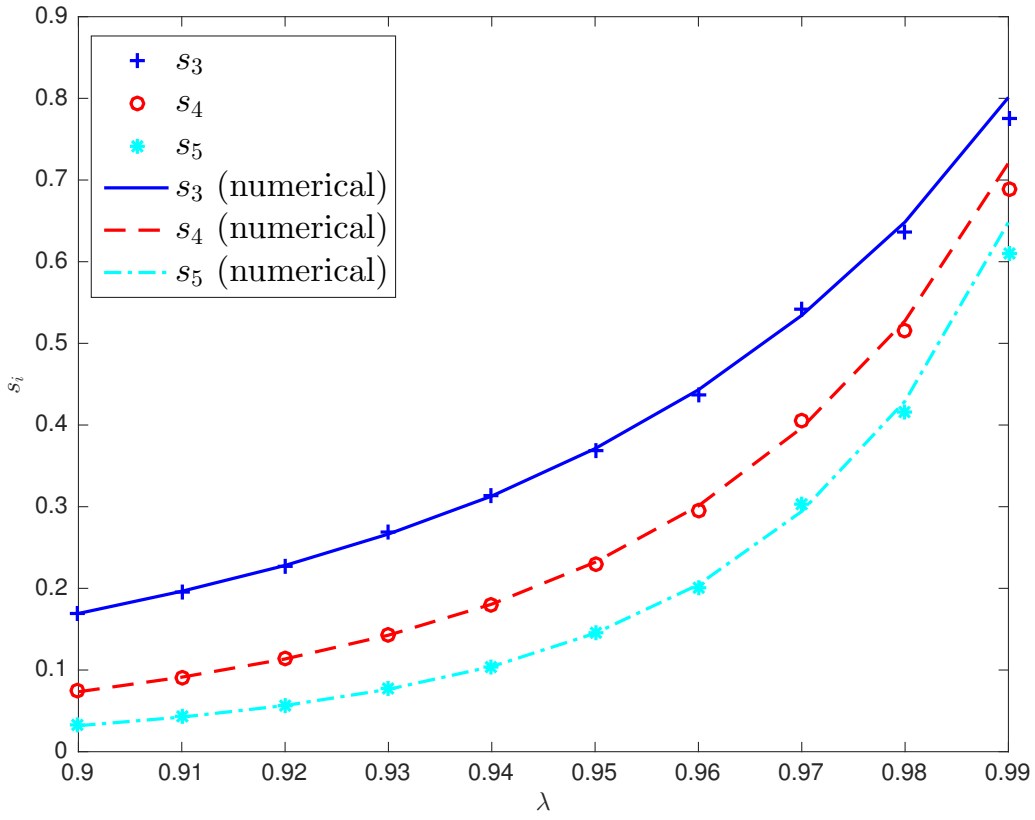


Figure 3.5: Tail distribution $s_i$ of JIQ when $r = 10$ and $\lambda$ changes from 0.9 to 0.99.

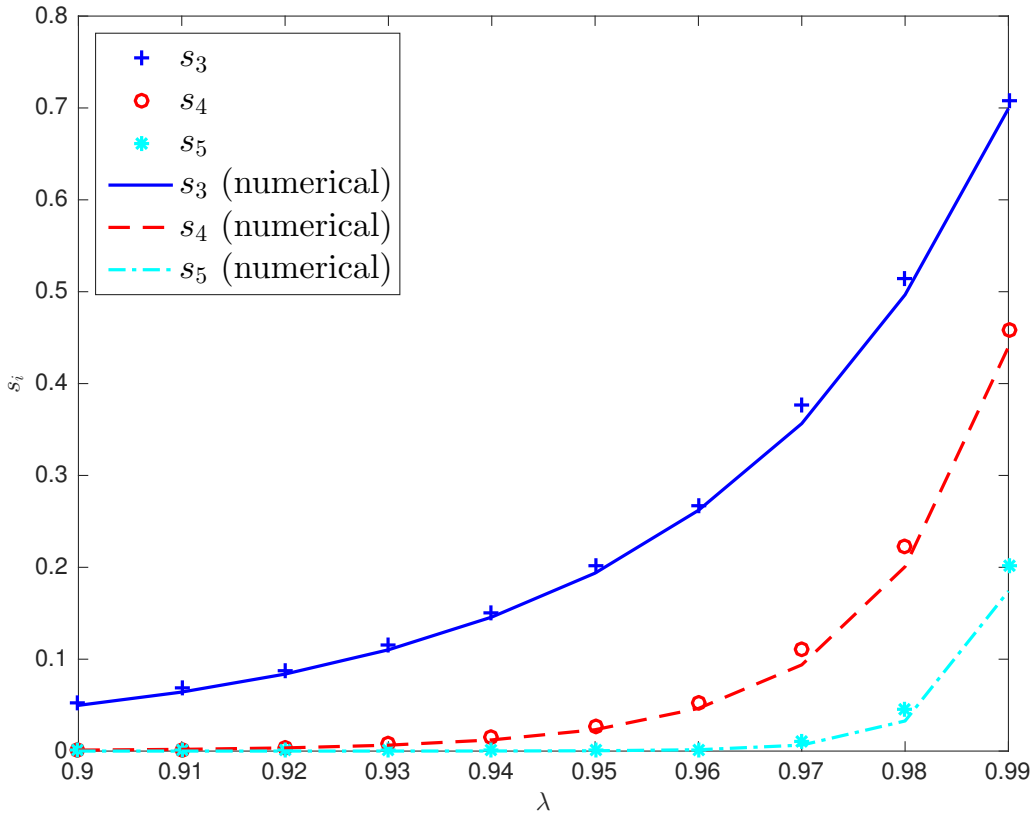


Figure 3.6: Tail distribution $s_i$ of JIQ-Po$d$ when $r = 10$ and $\lambda$ changes from 0.9 to 0.99.

Figure 3.7 shows the task response times of JIQ and JIQ-Po$d$ obtained from theoretical analysis and simulation. As we can see, when the server size is 1000, the maximum relative error is only 3.4% and the corresponding absolute error is 0.178. Hence, our theoretical predictions are fairly accurate even for systems of relatively small size. In Figure 3.7, we also compared the theoretical analysis of task response times from [1] with ours. The higher arrival rate is, the more accuracy [1] acquires.

### 3.3.2 Impact of "Delete request" Messages

Recall that in Section 3.1, we added a "delete request" strategy to the conventional JIQ algorithm (e.g., JIQ-Original). In the subsection, we explore the impact of such "delete request" strategy on our JIQ algorithm.

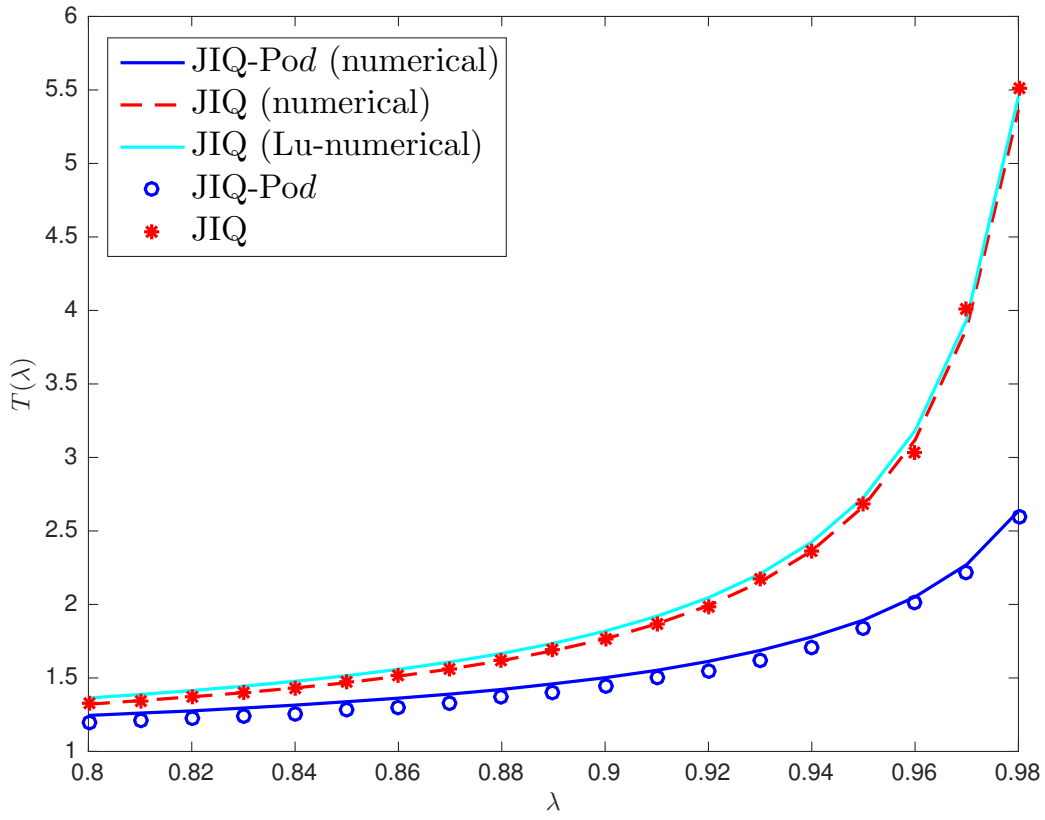


Figure 3.7: Response time of JIQ and JIQ-Pod when $N = 1000$, $r = 10$ and $\lambda$ changes from 0.9 to 0.98.

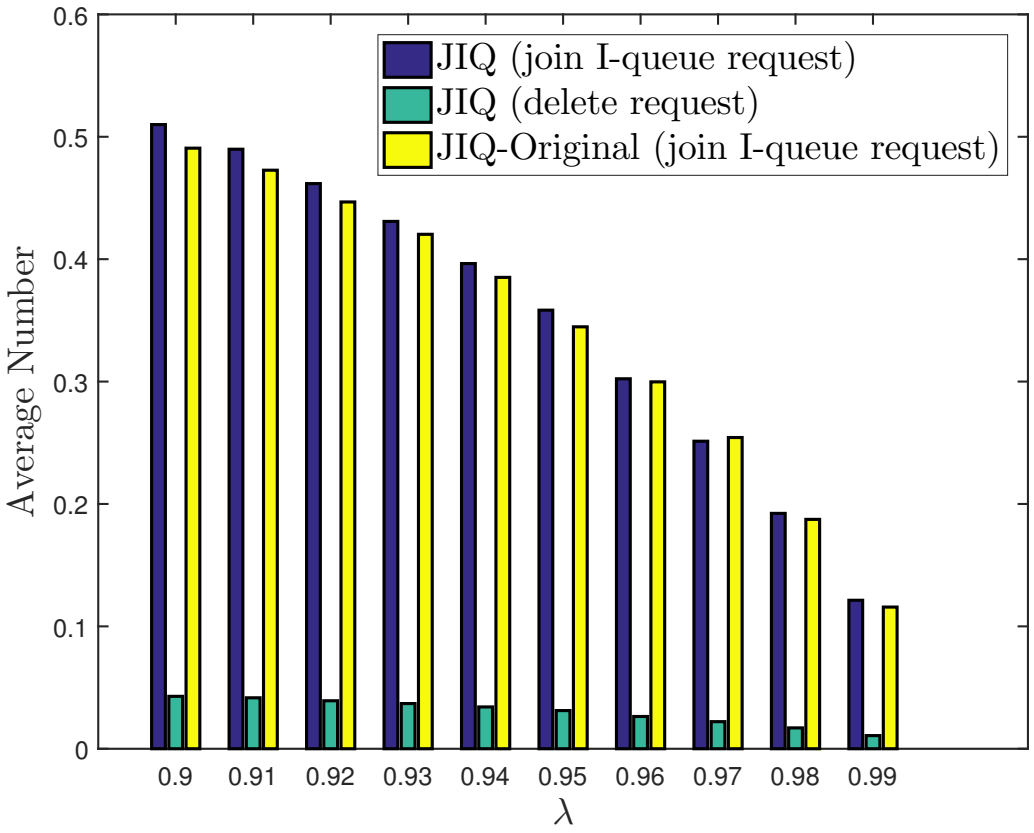


Figure 3.8: Average number of "request" messages per unit time per server of JIQ and JIQ-Original when $r = 10$ and $\lambda$ changes from 0.9 to 0.99.

First, idle servers not only send "join I-queue request", but also send "delete request" under our JIQ algorithm, which will increase the number of "request" messages for each server. Figure 3.8 studies the average number of "request" messages per unit time per server under JIQ-Original and JIQ. It turns out that the "delete request" only contributes to a small portion of the overall requests. For instance, when $\lambda = 0.9$, the number of "delete request" messages is no more than 8% of overall requests.

Second, such "delete request" strategy has little impact on the mean task response time. Figure 3.10 compares the mean task response times of different JIQ algorithms. It shows that the mean task response times of both JIQ-Original and JIQ are close to each other. To sum up, the "delete request" strategy has little impact on JIQ.

### 3.3.3 Comparison of JIQ Algorithms

Finally, we compare our JIQ-Po$d$ with two other variants, JIQ-Threshold and JIQ-SQ($d$) [1, 7].

- **JIQ-Threshold:** There is a threshold $z$ for server queue length. As long as a server has less than or equal to $z$ tasks, it will send a "join I-queue request" message to an I-queue. Thus, I-queues contain all servers with less than or equal to $z$ tasks.

- **JIQ-SQ($d$):** When an idle server needs to send a "join I-queue request" message to an I-queue, it adopts the Po$d$ algorithm to select which I-queue to report.

For comparison, we use the tail distribution $\hat{s}_i$ and the mean task response time. Figure 3.9 compares the tail distributions $\hat{s}_i$ among those three algorithms when $d = 2$ and $z = 1$. In Figure 3.9, the JIQ-Po$d$ algorithm always has the lightest tail in the heavy workload region. Figure 3.10 compares the mean task response time of different JIQ algorithms. It is shown that the mean task response time of JIQ-Po$d$ is the shortest among five alternative algorithms. Overall, our JIQ-Po$d$ algorithm achieves the best delay performance compared with other alternative JIQ algorithms.

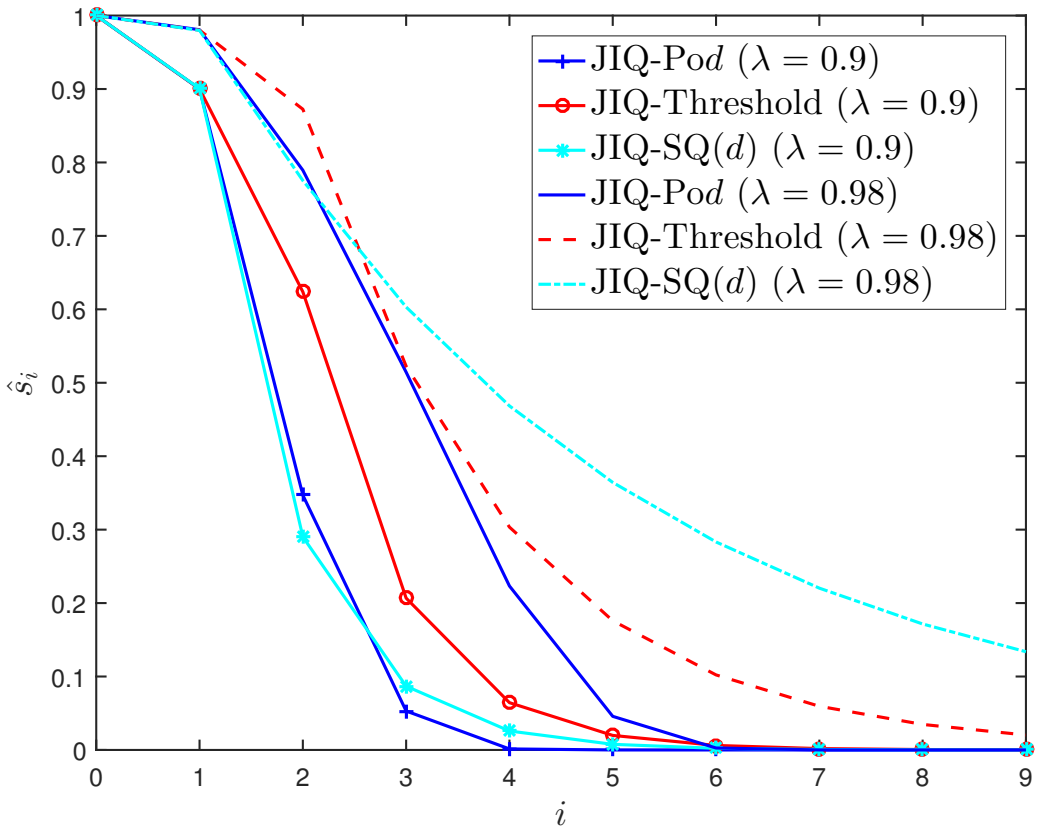


Figure 3.9: Tail distribution of three algorithms under light workload ($\lambda = 0.9$) and heavy workload ($\lambda = 0.98$).

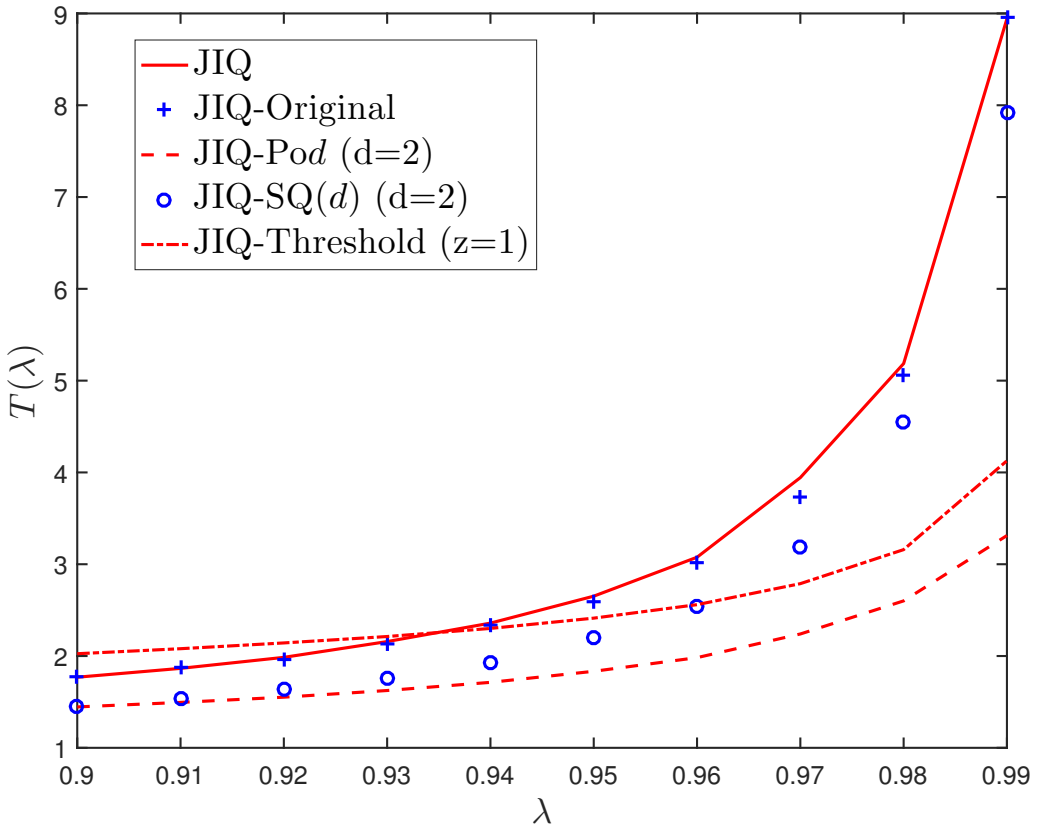


Figure 3.10: An comparison of average task response time between five algorithms.

# Chapter 4

# Conclusions

In this chapter, we summarize the contributions of the thesis and discuss the potential future works based on the current process.

## 4.1 Main Contributions

Our main contributions are as follows:

1. First, we propose a hybrid algorithm called Po$d$-Helper and analyze it by mean-field analysis. In particular, theoretical results show that, under some mild condition of the helper rate, the maximum queue size is bounded in the large-system regime. This means that tail response time of JIQ-Helper is indeed low. Moreover, extensive simulation results show that our analytical results are still valid in large, yet finite, systems.

2. Second, we analyze the distributed JIQ with mean-field analysis. We derive semi-closed form expressions of the stationary tail distribution and the expected response time for distributed JIQ. Comparing JIQ and Po$d$, we find out that JIQ is not always better than Po$d$.

3. Third, we extend the JIQ algorithm to the JIQ-Po$d$ algorithm. With the combination of the JIQ and Po$d$, the performance of the JIQ-Po$d$ is improved. With the same approach of JIQ, we get the stationary state distribution under large-scale limit in the JIQ-Po$d$ system, which is better than that of the JIQ system. Then, we are able to quantify the improvements of JIQ-Po$d$ over JIQ and Po$d$ in the large-system limit.

## 4.2 Future Work

In the thesis, we studied variants of Po*d* and JIQ. For real practice, there are some heuristic algorithms to simplify the difficulties of implementation. For example, there can be a threshold of queue length such that each distributed server won't receive more tasks if its queue length reaches the threshold. Thus, it is of great interest to conduct relevant theoretical analysis.

In this thesis, our analysis is based on the supermarket model, in which we assume the task arrival and departure event follows Poisson processes. Thus, it remains unknown for the performance of those algorithms in other stochastic conditions. Moreover, nowadays, a job consists of hundreds of tasks. Hence, the job is completed only if all the tasks of it are finished. In this condition, how to study the impact of our algorithms for jobs' completion times is meaningful.

Last but not least, to the best of the author's knowledge, hybrid algorithms still lack theoretical analyze of multi-resources scenarios, in which tasks require more than just computing resource. For example, when tasks require both CPU and memory, how should one arrange tasks in a way that minimizes the jobs' completion time? These questions are related to more complicated models and deserve further exploration.

# Bibliography


[1] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services," *Performance Evaluation*, vol. 68, no. 11, pp. 1056–1071, Nov. 2011. → pages iii, 2, 4, 39, 42

[2] E. Schurman and J. Brutlag, "The user and business impact of server delays, additional bytes, and http chunking in web search," in *O'Reilly Velocity Web Performance and Operations Conference*, 2009. → pages 1

[3] A. Eryilmaz and R. Srikant, "Asymptotically tight steady-state queue length bounds implied by drift conditions," *Queueing Systems: Theory and Applications*, vol. 72, no. 3–4, pp. 311–359, Dec. 2012. → pages 1

[4] M. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, UC Berkeley, 1996. → pages 2, 7, 17

[5] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in *Proc. of SOSP*, Farminton, Pennsylvania, Nov. 3–6, 2013, pp. 69–84. → pages 2, 3

[6] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," in *Proc. of INFOCOM*, Hong Kong, Apr. 26 – May 1, 2015, pp. 1131–1139. → pages 2, 3, 4

[7] M. Mitzenmacher, "Analyzing distributed join-idle-queue: A fluid limit approach," *arXiv preprint arXiv:1606.01833*, 2016. → pages 2, 4, 5, 26, 42

[8] A. L. Stolyar, "Pull-based load distribution in large-scale heterogeneous service systems," *Queueing Systems*, vol. 80, no. 4, pp. 341–361, 2015. → pages 5

[9] ——, "Pull-based load distribution among heterogeneous parallel servers: The case of multiple routers," *Queueing Syst. Theory Appl.*, vol. 85, no. 1-2, pp. 31–65, Feb. 2017. → pages

[10] S. Foss and A. Stolyar, "Large-scale join-idle-queue system with general service times," *ArXiv e-prints*, May 2016. → pages 2, 5

[11] P. Delgado, F. Dinu, A.-M. Kermarrec, and W. Zwaenepoel, "Hawk: Hybrid datacenter scheduling," in *Proc. of ATC*, Santa Clara, CA, USA, Jul. 8–10, 2015, pp. 499–510. → pages 4

[12] K. Karanasos, S. Rao, C. Curino, C. Douglas, K. Chaliparambil, G. M. Fumarola, S. Heddaya, R. Ramakrishnan, and S. Sakalanaga, "Mercury: Hybrid centralized and distributed scheduling in large shared clusters," in *Proc. of ATC*, Santa Clara, CA, USA, Jul. 8–10, 2015, pp. 485–497. → pages 4

[13] J. N. Tsitsiklis and K. Xu, "On the power of (even a little) resource pooling," *Stochastic Systems*, vol. 2, no. 1, pp. 1–66, 2012. → pages 4, 10, 18, 20, 50, 54

[14] H. Andersson and T. Britton, *Stochastic epidemic models and their statistical analysis*. Springer, 2012. → pages 5

[15] J. Wilkes and C. Reiss., "Traceversion1," 2015. [Online]. Available: https://github.com/google/cluster-data → pages 24

[16] K. Xu, "On the power of centralization in distributed processing," Ph.D. dissertation, MIT, 2011. → pages 54, 56

[17] W. Rudin, "Functional analysis," 1991. → pages 67

# Appendices

# Appendix A

Our first step proof is similar to that of Theorem 4 in [13]. First, we convert $\mathbf{s}(t)$ to $\mathbf{u}(t)$, where $u_j(t) \triangleq \sum_{i=j}^{\infty} s_i(t)$. Consider two different solutions of $\mathbf{u}(t)$ with different initial conditions: $\mathbf{u}^l(t)$ with initial condition $\mathbf{u}^l(0) = \mathbf{l}^0$ and $\mathbf{u}^f(t)$ with initial condition $\mathbf{u}^f(0) = \mathbf{f}^0$. By expanding and scaling the derivative of $\left\|\mathbf{u}^f - \mathbf{u}^l\right\|_\omega^2$ , we have

$$\frac{d}{dt}\left\|\mathbf{u}^f - \mathbf{u}^l\right\|_\omega^2 \leq M\left\|\mathbf{u}^f - \mathbf{u}^l\right\|_\omega^2 \tag{A.1}$$

where $M$ is a fixed number. When $\mathbf{f}^0 = \mathbf{l}^0$, according to Grönwall's inequality, we have

$$\left\|\mathbf{u}^f(t) - \mathbf{u}^l(t)\right\|_\omega^2 \leq e^{Mt}\left\|\mathbf{u}^f(0) - \mathbf{u}^l(0)\right\|_\omega^2 = 0. \tag{A.2}$$

This establishes the uniqueness of $\mathbf{u}(t)$.

When it comes to the uniqueness of $\mathbf{s}(t)$, we consider the initial condition $\mathbf{s}(0)$ and $\mathbf{u}(0)$ first. According to Condition (i), we have the finite support of $\mathbf{s}(0)$, which implies that there only exists one $\mathbf{u}(0)$ correspond to $\mathbf{s}(0)$. Hence, when $\mathbf{s}^f(0) = \mathbf{s}^l(0)$, we have $\mathbf{u}^f(0) = \mathbf{u}^l(0)$.

$$\begin{aligned}
&\left\|\mathbf{s}^f(t) - \mathbf{s}^l(t)\right\|_\omega^2 \\
=&\sum_{i=0}^{\infty}\frac{\left|(u_i^f(t) - u_{i+1}^f(t)) - (u_i^l(t) - u_{i+1}^l(t))\right|^2}{2^i} \\
\leq&\sum_{i=0}^{\infty}\frac{\left(\left|u_i^f(t) - u_i^l(t)\right| + \left|u_{i+1}^f(t) - u_{i+1}^l(t)\right|\right)^2}{2^i} \\
\leq&2\sum_{i=0}^{\infty}\frac{\left|u_i^f(t) - u_i^l(t)\right|^2 + \left|u_{i+1}^f(t) - u_{i+1}^l(t)\right|^2}{2^i} \\
\leq&4\sum_{i=0}^{\infty}\frac{\left|u_i^f(t) - u_i^l(t)\right|^2}{2^i} = 4\left\|\mathbf{u}^f(t) - \mathbf{u}^l(t)\right\|_\omega^2 = 0.
\end{aligned} \tag{A.3}$$

This completes the proof.

# Appendix B

First, we show the coordinate-wise dominance of $\mathbf{s}(t)$. Then, through the dominance, we prove the convergence of $s_i(t)$ to $\tilde{\mathbf{s}}$. Let $\mathbf{s}^a(t)$ and $\mathbf{s}^b(t)$ be two solutions at any $t > 0$ in dynamic model with initial condition that $\mathbf{s}^a(0)$ dominates $\mathbf{s}^b(0)$ (e.g., $\mathbf{s}^a(0) \succeq \mathbf{s}^b(0)$).

*Lemma B.1:* $\mathbf{s}^a(t)$ dominates $\mathbf{s}^b(t)$ (i.e., $\mathbf{s}^a(t) \succeq \mathbf{s}^b(t)$) for $t > 0$.

*Proof.* Let $\tau > 0$ be the first time that $s_i^a(\tau) = s_i^a(\tau) > 0$ for some $i > 0$, while $s_j^a(\tau) > s_j^a(\tau)$ for other coordinates.

If $\tau = \infty$, it is obvious that $\mathbf{s}^a(t) \succeq \mathbf{s}^b(t)$ for all $t > 0$. If $\tau < \infty$, we have

$$\begin{aligned}
&\frac{d(s_i^a(\tau))}{dt} - \frac{d(s_i^b(\tau))}{dt} \\
=&\{((s_{i-1}^a(\tau))^d - (s_i^a(\tau))^d) - ((s_{i-1}^b(\tau))^d - (s_i^b(\tau))^d)\} \\
&- \{(s_i^a(\tau) - s_{i+1}^a(\tau)) - (s_i^b(\tau) - s_{i+1}^b(\tau))\} \\
&- \{h_i(\mathbf{s}^a(\tau)) - h_i(\mathbf{s}^b(\tau))\} \\
=&((s_{i-1}^a(\tau))^d - (s_{i-1}^b(\tau))^d) + (s_{i+1}^a(\tau) - s_{i+1}^b(\tau)) \\
&- \{h_i(\mathbf{s}^a(\tau)) - h_i(\mathbf{s}^b(\tau))\}.
\end{aligned} \tag{B.1}$$

Next, we can discuss the three terms separately.

1. As $s_{i-1}^a(t) \geq s_{i-1}^b(t)$, we have $((s_{i-1}^a(\tau))^d - (s_{i-1}^b(\tau))^d) \geq 0$.

2. As $s_{i+1}^a(t) \geq s_{i+1}^b(t)$, we have $(s_{i+1}^a(\tau) - s_{i+1}^b(\tau)) \geq 0$.

3. As $s_{i+1}^a(t) \geq s_{i+1}^b(t)$, we have $\{h_i(\mathbf{s}^a(\tau)) - h_i(\mathbf{s}^b(\tau))\} \leq 0$.

Thus, $\frac{d(s_i^a(\tau))}{dt} - \frac{d(s_i^b(\tau))}{dt} \geq 0$. Because $\mathbf{s}^a(t)$ and $\mathbf{s}^b(t)$ are continuous, we have $\mathbf{s}^a(t) \succeq \mathbf{s}^b(t)$ for $t > \tau$. This completes the proof. □

Now, let's set $\hat{\mathbf{s}}(t)$ and $\bar{\mathbf{s}}(t)$, which satisfy $\hat{\mathbf{s}}(0) \succeq \tilde{\mathbf{s}} \succeq \bar{\mathbf{s}}(0)$. According to Lemma B.1, we have $\hat{\mathbf{s}}(t) \succeq \tilde{\mathbf{s}} \succeq \bar{\mathbf{s}}(t)$.

Here we discuss the coordinate-wise convergence of $s_i(t)$. Recall that $u_j(t) \triangleq \sum_{i=j}^{\infty} s_i(t)$, we have

$$\begin{aligned} \frac{du_i}{dt} &= \lambda(u_{i-1} - u_i)^d - (u_i - u_{i+1}) - \sum_{j=i}^{\infty} h_j(\mathbf{s}) \\ &= \lambda s_{i-1}^d - s_i - \sum_{j=i}^{\infty} h_j(\mathbf{s}). \end{aligned} \tag{B.2}$$

First, let's focus on $\hat{u}_1(t)$. Recall that $\tilde{s}_0 = \hat{s}_0 = 1$, we have

$$\begin{aligned} \frac{d\hat{u}_1}{dt} &= \lambda \hat{s}_0(t)^d - \hat{s}_1(t) - \sum_{j=i}^{\infty} h_j(\hat{\mathbf{s}})(t) \\ &= \lambda - \hat{s}_1(t) - \epsilon \\ &= \tilde{s}_1 - \hat{s}_1(t). \end{aligned} \tag{B.3}$$

Here we assume that $\lim_{t\to\infty} \sup |\tilde{s}_1 - \hat{s}_1(t)| = \sigma$, where $\sigma > 0$. Since $\hat{\mathbf{s}}(t) \succeq \tilde{\mathbf{s}}$, we have $\hat{s}_1(t) \geq \tilde{s}_1$, which implies that

$$\lim_{t\to\infty} \inf(\frac{d\hat{u}_1}{dt}) = -\sigma. \tag{B.4}$$

As $\hat{s}_i(t)$ is L-Lipschitz-continuous for all $i$, we can get a infinite sequence $\{t_k\}$ with $\lim_{k\to\infty} t_k = \infty$ in which there exists certain $\tau > 0$ that

$$\frac{d\hat{u}_1}{dt} \leq -\frac{1}{2}\sigma, \quad \forall t \in [t_k - \tau, t_k + \tau], \forall k \geq 0. \tag{B.5}$$

According to the initial condition that $\hat{u}_1(0) < \infty$, we have $\lim_{t\to\infty} \hat{u}_1(t) = -\infty$ by integration. This contradicts to the fact that $\hat{u}_1(t) \geq 0, \quad \forall t > 0$. Thus, we can conclude that

$$\lim_{t\to\infty} |\tilde{s}_1 - \hat{s}_1(t)| = 0. \tag{B.6}$$

Next, let $\lim_{t\to\infty} |\tilde{s}_i - \hat{s}_i(t)| = 0$ holds for some $i > 0$. Then for $\hat{u}_{i+1}(t)$, we have

$$\frac{d\hat{u}_{i+1}}{dt} = \lambda \hat{s}_i^d(t) - \hat{s}_{i+1}(t) - \sum_{j=i+1}^{\infty} h_j(\hat{\mathbf{s}})(t)$$

$$
\begin{aligned}
=&(\lambda \hat{s}_i^d(t) - \sum_{j=i+1}^{\infty} h_j(\hat{\mathbf{s}})(t)) - \hat{s}_{i+1}(t) \\
=&(\lambda \tilde{s}_i^d - \sum_{j=i+1}^{\infty} h_j(\hat{\mathbf{s}})(t)) - \hat{s}_{i+1}(t) + \lambda(\hat{s}_i^d(t) - \tilde{s}_i^d) \\
\leq&(\tilde{s}_{i+1} - \hat{s}_{i+1}(t)) + \lambda d(\hat{s}_i(t) - \tilde{s}_i).
\end{aligned} \tag{B.7}
$$

Recall that $\lim_{t\to\infty} |\tilde{s}_i - \hat{s}_i(t)| = 0$, then we can repeat the proof procedure of $\tilde{s}_1(t)$, from which we have $\lim_{t\to\infty} |\tilde{s}_{i+1} - \hat{s}_{i+1}(t)| = 0$. Thus, we can have the weak convergence of $\hat{\mathbf{s}}(t)$, i.e.

$$
\lim_{t\to\infty} \|\tilde{\mathbf{s}} - \hat{\mathbf{s}}(t)\|_{\omega} = 0. \tag{B.8}
$$

When it comes to the convergence of $\overline{\mathbf{s}}(t)$, the proof is almost the same, we have

$$
\lim_{t\to\infty} \|\tilde{\mathbf{s}} - \overline{\mathbf{s}}(t)\|_{\omega} = 0. \tag{B.9}
$$

This completes the proof.

# Appendix C

First, we introduce a preliminary Lemma from [16].

*Lemma C.1:* Fix $T > 0$. There exists a measurable set $\mathcal{C} \subset \Omega$, such that $\Pr(\mathcal{C}) = 1$ and for all $\omega \in \mathcal{C}$,

$$\lim_{N\to\infty} \sup_{t\in[0,T]} \left| Z^N(\omega, t) - (1+\lambda)t \right| = 0$$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \mathrm{I}_{[a,b)} \left( Y(\omega, i) \right) = b - a, \forall [a, b) \subset [0, 1]$$

There are some definitions from [16], which will also be used in the following steps.

- The $N$th system is defined as $\mathbf{X}^N \triangleq (\mathbf{S}^N, \mathbf{A}^N, \mathbf{G}^N, \mathbf{C}^N)$.
- $D^{\mathbb{Z}_+}[0, T]$ denotes the space of functions from $[0, T]$ to $\mathbb{R}^{\mathbb{Z}_+}$ that are right-continuous-with-left-limit for every coordinate. We use $D[0, T]$ to represent $D^{\mathbb{Z}_+}[0, T]$ for simplicity.
- The uniform metric $d^{\mathbb{Z}_+}(\cdot, \cdot)$ is defined as $d^{Z_+}(x, y) \triangleq \sup_{t\in[0,T]} \|x(t) - y(t)\|_\omega$, where $x, y \in D^{\mathbb{Z}_+}[0, T]$. We use $d(\cdot, \cdot)$ to represent $d^{\mathbb{Z}_+}(\cdot, \cdot)$ for simplicity.

### C.0.1 Step One: Convergence to Continuous Functions

First, we show that the sample paths are "close to" some continuous functions. Here we have a similar proposition. As the proof is essentially the same as Proposition 11 [13], we only introduce the outline of this proof.

*Proposition C.2:* For all $\omega \in \mathcal{C}$, suppose there exists $s(0) = s^0$ and

$$\lim_{N\to\infty} \left\| \mathbf{S}^N(\omega, 0) - \mathbf{s}(0) \right\|_\omega = 0. \tag{C.1}$$

Then any subsequence of the sample path $\{\mathbf{X}^N(\omega, .)\}$ contains a subsequence $\{\mathbf{X}^{N_k}(\omega, .)\}$ that converges to some coordinate-wise Lipschitz-continuous function $\mathbf{x}(t) = (\mathbf{s}(t), \mathbf{a}(t), \mathbf{g}(t), \mathbf{c}(t))$,

where $\mathbf{a}(0) = \mathbf{g}(0) = \mathbf{c}(0) = 0$ and

$$|x_i(a) - x_i(b)| \leq L\,|a-b|\,, \forall a, b \in [0,T], i \in \mathbb{Z}_+, \tag{C.2}$$

where $L > 0$ is a universal constant, independent of $\omega$, $\mathbf{x}$ and $T$. Then $d^{\mathbb{Z}_+}(\mathbf{S}^{N_k}, \mathbf{s})$,$d^{\mathbb{Z}_+}(\mathbf{A}^{N_k}, \mathbf{a})$, $d^{\mathbb{Z}_+}(\mathbf{G}^{N_k}, g)$ and $d^{\mathbb{Z}_+}(\mathbf{C}^{N_k}, \mathbf{c})$ converge to 0 as $k \to \infty$.

*Proof.* The proof consists of 2 steps. First, we find the coordinate-wise limit of $X_i^N$ for all $i$. Second, through a diagonal argument approach, we construct the limit point of $\mathbf{X}^N$.

When showing the coordinate-wise limit, we first introduce some preliminary definitions and lemmas.

*Definition C.3:* Let $E_c$ be a non-empty compact subset of $D[0,T]$. A sequence of subsets $\epsilon = \{E_N\}_{N\geq 1}$, which are in space $D[0,T]$, are asymptotically close to $E_c$ if they satisfy

$$\lim_{N\to\infty} \sup_{x\in E_N} d(x, E_c) = 0$$

where $d(x, E_c) \triangleq \inf_{y\in E_c} d(x,y)$. .

Now, set $\epsilon = \{E_N\}_{N\geq 1}$ be a sequence of subsets in the space $D[0,T]$ that

$$E_N = \{x \in D[0,T] : \left|x(0) - x^0\right| \leq M_N, \text{ and } |x(a) - x(b)| \leq L\,|a-b| + \gamma_N\}$$

where $x^0$ is a constant and $M_N \to 0$, $\gamma_N \to 0$ with $N \to \infty$. Then, let a set of Lipschitz-continuous functions on $[0,T]$ with Lipschitz constant $L$ and initial values bound $M \geq 0$ be denoted by

$$E_c = \{x \in D[0,T] : |x(0)| \leq M, \text{ and } |x(a) - x(b)| \leq L\,|a-b|\}.$$

With $E_c$ and $\epsilon$ defined as above, the preliminary lemmas go as following.

*Lemma C.4:* $E_c$ is compact.

*Lemma C.5:* $\epsilon$ is asymptotically close to $E_c$.

*Lemma C.6:* Suppose there exists a $\mathbf{s}^0$ such that for all $\omega \in \mathcal{C}$, $\left\|S(0,\omega) - s^0\right\|_\omega \leq \tilde{M}_N$, for some $\tilde{M}_N \downarrow 0$. Then, for all $\omega \in \mathcal{C}$ and coordinate $i$, there exist $L > 0$ and $M_N \downarrow 0$ and $\gamma_N \downarrow 0$,

such that $X_i^N(\omega,\cdot) \in E_N$.

Lemma C.4 and Lemma C.5 are directly from Appendix A [16], while there is slightly change for Lemma C.6. Although we use $\mathbf{S}^N$ instead, for each coordinate $i$, the total number of jumps $A_i^N$, $G_i^N$ and $C_i^N$ are still dominated by $Z^N$, while $A_i^N$, $G_i^N$ and $C_i^N$ are monotonically non-decreasing. Hence, the proof of Lemma C.6 remains unchanged.

Through the lemmas above and the definition of "asymptotically close", we can show that for every coordinate $i$, there exist a subsequence of $\{\mathbf{X}^N(\omega,\cdot)\}$, denoted by $\{\mathbf{X}^{N_j}(\omega,\cdot)\}$, and a sequence of $\{y_j\}$, that

$$\lim_{j\to\infty} d(X_i^{N_j}(\omega,\cdot), y_j) = 0. \tag{C.3}$$

According to the character of $E_c$, the limit point of $\{y_j\}$ implies that a further subsequence converges to that limit point. This is the main idea in finding the coordinate-wise limit of $X_i^N$ for all $i$.

Then, the construction of the limit points of $\mathbf{X}^N$ mainly depends on the following two steps. Take $\mathbf{S}^N(\omega,\cdot)$ as an example. The first key step is the construction of the further subsequence. From sequence $\{\mathbf{S}^{N_j}(\omega,\cdot)\}$, we derive a subsequence $\{\mathbf{S}^{N_j^1}(\omega,\cdot)\}$, that $S_1^{N_j^1}(\omega,\cdot) \to s_1$ with $j \to \infty$ . Recursively, we have a subsequence $\{\mathbf{S}^{N_j^{i+1}}(\omega,\cdot)\}$, that $S_{i+1}^{N_j^{i+1}}(\omega,\cdot) \to s_{i+1}$ with $j \to \infty$, from previous subsequence. The second key step is to select suitable $N_k$ that

$$N_k = \min\left\{ N \geq N_{k-1} : \sup_{1\leq i\leq k} d(s_i, S_i^N(\omega,\cdot)) \leq \frac{1}{k} \right\},$$

where $N_1 = 1$ and $k \geq 2$. Then we can bound the $d^{\mathbb{Z}_+}(\mathbf{S}^{N_k}, \mathbf{s})$. This completes the proof. □

### C.0.2 Step Two: Convergence to the Solution of the Dynamic Model

Then, we ought to show the fluid limit of $\mathbf{S}^N(t)$ is indeed the solution of the dynamic model. We can achieve this goal through showing that the drift of the fluid limit is exactly the same as that of the dynamic model in regular points.

*Proposition C.7:* Fix $\omega \in \mathcal{C}$ and $T > 0$. Set $\mathbf{x}(t)$ be the limit point of the subsequence of $\{\mathbf{X}^N(\omega,\cdot)\}$ in Proposition C.2 and $\mathbf{x}(t)$ is coordinately differentiable at $t$. Then we can show

that for $\forall i \in \mathbb{N}$,

$$\begin{aligned} \dot{a}_i(t) &= \lambda\{(s_{i-1}(t))^d - (s_i(t))^d\}, \\ \dot{g}_i(t) &= (s_i(t) - s_{i+1}(t)), \\ \dot{c}_i(t) &= h_i(\mathbf{s}(t)), \end{aligned} \tag{C.4}$$

where $h_i(\mathbf{s}(t))$ is defined in dynamic model, with initial condition $\mathbf{s}(0) = \mathbf{s}^0$ and boundary condition $s_1(t) = 1$.

This implies that the limitation of $\mathbf{S}^N(t)$ is the solution of the dynamic model.

*Proof.* Here we will discuss those three parts one by one.

**claim 1:** $\dot{g}_i(t) = (s_i(t) - s_{i+1}(t))$. Since $g_i(t)$ is differentiable, we have

$$\dot{g}_i(t) = \lim_{\tau \to 0} \frac{g_i(t+\tau) - g_i(t)}{\tau}. \tag{C.5}$$

Recall Proposition C.2, we then have

$$g_i(t+\tau) - g_i(t) = \lim_{N \to \infty} (G_i^{N_k}(t+\tau) - G_i^{N_k}(t)). \tag{C.6}$$

According to coupling construction, $G_i^N(t)$ will increased by $\frac{1}{N}$ if an event happens and the corresponding $Y(\cdot) \in \frac{\lambda}{1+\lambda+\epsilon} + \frac{1}{1+\lambda+\epsilon}[S_{i+1}^N(t-), S_i^N(t-))$. Thus, we can have

$$G_i^{N_k}(t+\tau) - G_i^{N_k}(t) = \frac{1}{N_k} \sum_{j=N_k W^{N_k}(t)}^{N_k W^{N_k}(t+\tau)} \mathrm{I}_{I_j}(Y(j)), \tag{C.7}$$

where $I_j \triangleq \frac{\lambda}{1+\lambda+\epsilon} + \frac{1}{1+\lambda+\epsilon}[S_{i+1}^N(t-), S_i^N(t-))$ and $t_j^{N_k}$ is the time when $j$th event happens in $Z^{N_k}(\cdot)$.

*Lemma C.8:* Fix $i$ and $t$. For any sufficiently small $\tau > 0$,

$$|(g_i(t+\tau) - g_i(t)) - \tau(s_i(t) - s_{i+1}(t))| \leq 2\tau^2 L. \tag{C.8}$$

*Proof.* First, fix $\omega \in \mathcal{C}$, $i \geq 1$, $t > 0$ and $\epsilon > 0$. According to Proposition C.2, there exists a non-increasing sequence $\gamma_n \downarrow 0$ so that for all $b \in [t, t+\tau]$ and all sufficiently large $k$, we have

$S_j^{N_k}(s) \in [s_j(t) - (\tau L + \gamma_{N_k}), s_j(t) + (\tau L + \gamma_{N_k}))$, for $j \in \{i-1, i, i+1\}$. Then, we have

$$[S_{i+1}^{N_k}(b), S_i^{N_k}(b)) \supset [s_{i+1}(t) + (\tau L + \gamma_{N_k}), s_i(t) - (\tau L + \gamma_{N_k})),$$
$$[S_{i+1}^{N_k}(b), S_i^{N_k}(b)) \subset [[s_{i+1}(t) - (\tau L + \gamma_{N_k})]^+, s_i(t) + (\tau L + \gamma_{N_k})).$$

Then we define the sequence of self-valued functions $\{\eta^n(t)\}$ as

$$\eta^n(t) \triangleq \frac{\lambda}{1+\lambda+\epsilon} + \frac{1}{1+\lambda+\epsilon}[[s_{i+1}(t) - (\tau L + \gamma_{N_k})]^+, s_i(t) + (\tau L + \gamma_{N_k})). \tag{C.9}$$

Recall that $\gamma_n \downarrow 0$, we have $\eta^{n+1}(t) \subset \eta^n(t)$ and

$$\lim_{n\to\infty} \eta^n(t) = \frac{\lambda}{1+\lambda+\epsilon} + \frac{1}{1+\lambda+\epsilon}[[s_{i+1}(t) - \tau L]^+, s_i(t) + \tau L). \tag{C.10}$$

Set $1 \le l \le N_k$, we have

$$\begin{aligned}
&G_i^{N_k}(t+\tau) - G_i^{N_k}(t) \\
\le &\frac{1}{N_k} \sum_{j=N_k W^{N_k}(t)+1}^{N_k W^{N_k}(t+\tau)} \mathrm{I}_{\eta^{N_k}(t)}(Y(j)) \\
\le &\frac{1}{N_k} \sum_{j=N_k W^{N_k}(t)+1}^{N_k W^{N_k}(t+\tau)} \mathrm{I}_{\eta^{l}(t)}(Y(j)) \\
= &\frac{1}{N_k}\Big( \sum_{j=1}^{N_k W^{N_k}(t+\tau)} \mathrm{I}_{\eta^{l}(t)}(Y(j)) - \sum_{j=1}^{N_k W^{N_k}(t+\tau)} \mathrm{I}_{\eta^{l}(t)}(Y(j))\Big).
\end{aligned} \tag{C.11}$$

According to Lemma C.1, we have

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{NW^N(t)} \mathrm{I}_{[a,b)}(Y(i)) = (1+\lambda+\epsilon)t(b-a). \tag{C.12}$$

Combining (C.11) and (C.12), we can establish the connection between $g_i(\cdot)$ and $s_i(\cdot)$

$$\begin{aligned}
&g_i(t+\tau) - g_i(t) \\
= &\lim_{k\to\infty} (G_i^{N_k}(t+\tau) - G_i^{N_k}(t))
\end{aligned}$$

$$\leq (t+\tau-t)(1+\lambda+\epsilon)\frac{1}{1+\lambda+\epsilon}(s_i(t)-s_{i+1}(t)+2(\tau L+\gamma_n)) \tag{C.13}$$
$$\leq \tau(s_i(t)-s_{i+1}(t))+2\tau(\tau L+\gamma_n).$$

Taking $l \to \infty$, we have

$$(g_i(t+\tau)-g_i(t))-\tau(s_i(t)-s_{i+1}(t)) \leq 2\tau^2 L. \tag{C.14}$$

Through similar steps, if we set

$$\eta^n(t) \triangleq \frac{\lambda}{1+\lambda+\epsilon}+\frac{1}{1+\lambda+\epsilon}[s_{i+1}(t)+(\tau L+\gamma_{N_k}), s_i(t)-(\tau L+\gamma_{N_k})), \tag{C.15}$$

there will be a lower bound that

$$(g_i(t+\tau)-g_i(t))-\tau(s_i(t)-s_{i+1}(t)) \geq -2\tau^2 L. \tag{C.16}$$

This completes the proof of Lemma C.8. □

Thus we have

$$\dot{g}_i(t)=\lim_{\tau\to 0}\frac{g_i(t+\tau)-g_i(t)}{\tau}=(s_i(t)-s_{i+1}(t)). \tag{C.17}$$

**claim 2:** $\dot{a}_i(t)=\lambda\{(s_{i-1}(t))^d-(s_i(t))^d\}$. The proof of Claim 2 is similar as that of Claim 1.

**claim 3:** $\dot{c}_i(t)=h_i(\mathbf{s}(t))$. Recall the dynamic model, $h_i(\mathbf{s}(t))$ is separately discussed in four cases. Thus $\dot{c}_i(t)$ also requires separate discussion.

1. $\dot{c}_i(t)=0, s_{i-1}(t)=0, s_i(t)=0$. Here, we have

$$\dot{c}_i(t)=\dot{a}_i(t)-\dot{g}_i(t)-\dot{s}_i(t). \tag{C.18}$$

   According to claim 1 and 2, we have $\dot{a}_i(t)=0$ and $\dot{g}_i(t)=0$. As $s_i(t)\geq 0$ always holds, if $\dot{s}_i(t)$ exists, we can have $\dot{s}_i(t)=0$. Thus, we can have $\dot{c}_i(t)=0$.

2. $\dot{c}_i(t)=0, s_i(t)>0, s_{i+1}(t)>0$. There exists some small enough $\tau$ , such that $S_{i+1}^{N_k}(b)>0$

for all $b \in [t, t+\tau]$. According to the mapping for the sample path, we have $C_i^{N_k}(t+\tau) - C_i^{N_k}(t) = 0$, which implies that

$$c_i(t+\tau) - c_i(t) = \lim_{N\to\infty}(C_i^{N_k}(t+\tau) - C_i^{N_k}(t)) = 0. \tag{C.19}$$

Thus, we can show that $\dot{c}_i(t) = 0$.

3. $\dot{c}_i(t) = \min\{\lambda s_{i-1}^d, \epsilon\}, s_{i-1}(t) > 0, s_i(t) = 0$. First, we consider $\lambda s_{i-1}^d \leq \epsilon$. According to Claim 1 and 2, we have $\dot{a}_i(t) = \lambda s_{i-1}^d$ and $\dot{g}_i(t) = 0$. Through (C.18) and the fact that $s_i(t) \geq 0$, we have $\dot{s}_i(t) = 0$. Thus, $\dot{c}_i(t) = \lambda s_{i-1}^d$. Second, we consider $\lambda s_{i-1}^d > \epsilon$. If $\dot{c}_i(t)$ exists, we will have $\dot{c}_i(t) > \epsilon$, which is impossible. Thus, the second case is not differentiable.

4. $\dot{c}_i(t) = \epsilon - \min\{\lambda s_{i-1}^d, \epsilon\}, s_i(t) > 0, s_{i+1}(t) = 0$. In the finite $N$-server system, we have

$$\sum_{i=1}^{\infty} C_i^{N_k}(t+\tau) - \sum_{i=1}^{\infty} C_i^{N_k}(t) = \frac{1}{N_k} \sum_{j=N_k W^{N_k}(t)}^{N_k W^{N_k}(t+\tau)} \mathrm{I}_{I_c}(Y(j)), \tag{C.20}$$

where $I_c = [\frac{1+\lambda}{1+\lambda+\epsilon}, 1)$.

According to the same argument in the proof of Lemma C.8, we have $\sum_{i=1}^{\infty} \dot{c}_i(t) = \epsilon$. Hence, we have

$$\dot{c}_i(t) = \epsilon - \sum_{j=1}^{i-1} \dot{c}_j(t) - \sum_{j=i+1}^{\infty} \dot{c}_j(t) = \epsilon - \min\{\lambda s_{i-1}^d, \epsilon\}. \tag{C.21}$$

Finally, we check boundary condition $s_0(t) = 1$. It is straightforward that boundary condition holds as the fact that $S_0^N = 1$. This completes the proof. □

# Appendix D

We will use the Lyapunov function to show positive recurrence of $\mathbf{S}^N(t)$. First, we consider a $N$-server system. Set the average arrival rate $\lambda < 1$, the processing rate of each server to be 1. The state of each server can be represented by a set $\{q_i(t)\}_{i=1}^N$, in which $q_i(t)$ is the queue length of server $i$. Let $\{q_i(n)\}_{i=1}^N$ be the embedded Markov chain. This implies that $\{q_i(n)\}_{i=1}^N = \{q_i(t_n)\}_{i=1}^N$, where $t_n$ is the time in which $n$th event happened. Here, we use embedded Markov chain in the analysis of the drift of the Lyapunov function.

In $M/M/N$ queues case, $q_i(n+1) = q_i(n) + a_i(n) - d_i(n)$, where $a_i(n)$ denotes task arrival at step $n$ and $d_i(n)$ denotes task departure. From $n$ step to $n+1$ step, there is only one task arrive or departure signal generated, we have

$$\begin{aligned} &\Pr[\sum_{i=1}^{N}(a_i(n)+d_i(n))=1]=1,\\ &\Pr[a_i(n)=1]=\tfrac{1}{N}\tfrac{\lambda}{1+\lambda},\\ &\Pr[d_i(n)=1]=\tfrac{1}{N}\tfrac{1}{1+\lambda}. \end{aligned} \tag{D.1}$$

Considering a Lyapunov function $V(n) = \sum_{i=1}^{\mathrm{N}} q_i^2(n)$, the drift of $V(n)$ goes as

$$\begin{aligned} &E[V(n+1)-V(n)\,|q(n)=\{q_i\}_{i=1}^N]\\ =&E[\sum_{i=1}^{\mathrm{N}}\left((q_i(n)+a_i(n)-d_i(n))^+\right)^2-\sum_{i=1}^{\mathrm{N}}q_i^2(n)]\\ \le&E[\sum_{i=1}^{\mathrm{N}}\left(q_i(n)+a_i(n)-d_i(n)\right)^2-\sum_{i=1}^{\mathrm{N}}q_i^2(n)]\\ =&E[\sum_{i=1}^{\mathrm{N}}\left(a_i(n)-d_i(n)\right)^2+\sum_{i=1}^{\mathrm{N}}2q_i(n)(a_i(n)-d_i(n))]\\ =&1+2\sum_{i=1}^{\mathrm{N}}a_i(n)q_i(n)-2\sum_{i=1}^{\mathrm{N}}d_i(n)q_i(n) \end{aligned} \tag{D.2}$$

$$
\begin{aligned}
=&1+\frac{1}{N}\frac{2\lambda}{1+\lambda}\sum_{i=1}^{\mathrm{N}}q_i-\frac{1}{N}\frac{2}{1+\lambda}\sum_{i=1}^{\mathrm{N}}q_i\\
=&1+\frac{1}{N}\frac{2(\lambda-1)}{1+\lambda}\sum_{i=1}^{\mathrm{N}}q_i.
\end{aligned}
$$

Thus, for any $\theta>0$, we have $\sum\limits_{i=1}^{\mathrm{N}}q_i>\frac{(1+\theta)(1+\lambda)N}{2(1-\lambda)}$. According to Foster Lyapunov theorem, it is straightforward to show that $\{q_i(t)\}_{i=1}^{N}$ of $M/M/N$ is positive recurrent.

Similarly, in *Pod* case, set the same average arrival rate and processing rate.

$$
\begin{aligned}
&\Pr[\sum_{i=1}^{N}(a_i(n)+d_i(n))=1]=1,\\
&\Pr[d_i(n)=1]=\tfrac{1}{N}\tfrac{1}{1+\lambda}.
\end{aligned}
\tag{D.3}
$$

Recall *Pod* algorithm, if $q_i(n)>q_j(n)$, then $\Pr[a_i(n)=1]<\Pr[a_j(n)=1]$. Using the same Lyapunov function, we have

$$
E[V(n+1)-V(n)\,\big|q(n)=\{q_i\}_{i=1}^{N}]\le 1+\frac{1}{N}\frac{2(\lambda-1)}{1+\lambda}\sum_{i=1}^{\mathrm{N}}q_i. \tag{D.4}
$$

Hence, we can prove that $\{q_i(t)\}_{i=1}^{N}$ of *Pod* is positive recurrent.

Finally, we show the positive recurrence of $\{q_i(t)\}_{i=1}^{N}$ using external strategy. We can describe the $N$-server system as follows. From $n$th step to $(n+1)$th step, with probability $\frac{1+\lambda}{1+\lambda+\epsilon}$ it will behave the same as *Pod*, while a helper may change the queue length of the maximum queue and the minimum queue with probability $\frac{\epsilon}{1+\lambda+\epsilon}$.

If the helper works in step $n$ and there exists $q_i>0$, then we have

$$
E[V(n+1)-V(n)\,\big|q(n)=\{q_i\}_{i=1}^{N}]=(q_{\max}-1)^2-{q_{\max}}^2<0. \tag{D.5}
$$

Thus, the drift of Lyapunov function goes as

$$
E[V(n+1)-V(n)\,\big|q(n)=\{q_i\}_{i=1}^{N}]\le\tfrac{1+\lambda}{1+\lambda+\epsilon}(1+\tfrac{1}{N}\tfrac{2(\lambda-1)}{1+\lambda}\sum_{i=1}^{\mathrm{N}}q_i) \tag{D.6}
$$

This also gives us a bound on average queue length, which is

$$E[V(n+1) - V(n) \,|q(n) = \{q_i\}_{i=1}^N] \leq \frac{1+\lambda}{1+\lambda+\epsilon}(1 + \frac{1}{N}\frac{2(\lambda-1)}{1+\lambda}E(\sum_{i=1}^{\mathrm{N}} q_i)). \tag{D.7}$$

Thus, for any $\theta > 0$,

$$\begin{array}{l} 1 + \frac{1}{N}\frac{2(\lambda-1)}{1+\lambda}\sum\limits_{i=1}^{\mathrm{N}} q_i < -\frac{1+\lambda+}{1+\lambda}\theta, \\ \sum\limits_{i=1}^{\mathrm{N}} q_i > \frac{(1+\frac{1+\lambda+}{1+\lambda}\theta)(1+\lambda)N}{2(1-\lambda)}. \end{array} \tag{D.8}$$

Thus, we prove the positive recurrence of $\{q_i(t)\}_{i=1}^N$. This implies the positive recurrence of $\mathbf{S}^N(t)$. When it comes to the tightness of $\pi^N$, it is straightforward to be shown by standard stochastic dominance. This completes the proof.

# Appendix E

Set the potential equation $V(\mathbf{z})$ as

$$V(\mathbf{z}) = \sum_{i=1}^{N} z_{i,1}^2 + \sum_{i=1}^{N} z_{i,2}^2 \tag{E.1}$$

where $\mathbf{z} = \left\{ \left( z_{i,1}^{(N)}(t), z_{i,2}^{(N)}(t) \right) \right\}_{i=1}^{N}$.

Let $q_{\mathbf{z},\mathbf{w}}$ be the transition rate from system state $\mathbf{z}$ to $\mathbf{w}$. The system state changes only when there is a task-arrival or a task-departure event happens. We consider the Lyapunov drift as follows:

$$\begin{aligned} &\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[ V(\mathbf{w}) - V(\mathbf{z}) \right] \\ &= \sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[ \sum_{i=1}^{N} (w_{i,1}^2 - z_{i,1}^2) + \sum_{i=1}^{N} (w_{i,2}^2 - z_{i,2}^2) \right]. \end{aligned} \tag{E.2}$$

By the Foster-Lyapunov theorem, we only need to show that for any fixed $N$ and $M$,

$$\begin{aligned} &\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[ V(\mathbf{w}) - V(\mathbf{z}) \right] \\ &\leq 2(\lambda - 1) \sum_{i=1}^{N} z_{i,1} + (1 + M^2 + \lambda) N \end{aligned} \tag{E.3}$$

This is because:

1. If $\sum_{i=1}^{N} z_{i,1} > \frac{(1+M^2+\lambda)N}{2(1-\lambda)}$, we have

$$\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[ V(\mathbf{w}) - V(\mathbf{z}) \right] < 0.$$

2. If $\sum_{i=1}^{N} z_{i,1} \leq \frac{(1+M^2+\lambda)N}{2(1-\lambda)}$, we have

$$\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[V(\mathbf{w}) - V(\mathbf{z})\right] < \infty.$$

In terms of $z_{i,2}$, it increases from 0 to a positive number in $[1, M]$ when the $i$th server becomes idle. In other cases, $z_{i,2}$ remains unchanged or decreases to 0. As $z_{i,2} \in [0, M]$ and the processing rate for each server is 1, we obtain

$$\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[\sum_{i=1}^{N} (w_{i,2}^2 - z_{i,2}^2)\right] \leq N(M^2 - 0^2) = NM^2. \tag{E.4}$$

In terms of $z_{i,1}$, it increases by 1 when a task arrives to the $i$th server; it decrease by 1 when a task departures the $i$th server. Let $p_0^{(N)}$ be the fraction of empty I-queues. Recall the evolution of a JIQ system, when a new task arrives, we obtain

$$\Pr\{\text{meet a non-empty I-queue}\} = 1 - p_0^{(N)}$$

and

$$\Pr\{\text{meet an empty I-queue}\} = p_0^{(N)}.$$

For task-arrival events, we have

$$\begin{aligned}
&\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}}^{(\text{arrival})} \left[\sum_{i=1}^{N} (w_{i,1}^2 - z_{i,1}^2)\right] \\
&\leq \lambda N p_0^{(N)} \sum_{i=1}^{N} \tfrac{(z_{i,1}-1)^2 - z_{i,1}^2}{N} + \lambda N (1 - p_0^{(N)})(1^2 - 0^2) \\
&\leq 2\lambda \sum_{i=1}^{N} z_{i,1} + \lambda N.
\end{aligned}$$

For task-departure events, recall that the server processing rate is 1, we have

$$\begin{aligned}
&\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}}^{(\text{departure})} \left[\sum_{i=1}^{N} (w_{i,1}^2 - z_{i,1}^2)\right] \\
&\leq \sum_{i=1}^{N} \left[(z_{i,1}+1)^2 - z_{i,1}^2\right] \\
&= -2\sum_{i=1}^{N} z_{i,1} + N.
\end{aligned}$$

Thus, we have

$$\sum_{\mathbf{w}\neq\mathbf{z}} q_{\mathbf{z},\mathbf{w}} \left[\sum_{i=1}^{N} (w_{i,1}^2 - z_{i,1}^2)\right] \leq 2(\lambda-1)\sum_{i=1}^{N} z_{i,1} + (1+\lambda)N. \tag{E.5}$$

Finally, we sum up (E.4) and (E.5) to have (E.3). This completes the proof.

# Appendix F

We first show $f(p_0)$ is differentiable.

*Proof.* First, we prove the uniform convergence of $f(p_0)$ through the Weierstrass M-test [17].

We construct a series of functions $H_n(p_0)$ and $h_i(p_0)$ as

$$H_n(p_0) = \sum_{i=0}^{n} h_i(p_0), n \geq 0$$

and

$$h_i(p_0) = (\frac{r - r\lambda p_0}{r + p_0})^i p_0, i \geq 0.$$

Recall from (3.2), we construct another function of $p_0$, $g_i(p_0)$ as

$$g_i(p_0) = \begin{cases} p_0, i = 0, \\ \frac{r^i(1-\lambda p_0)^i}{\prod_{j=1}^{i}(r+jp_0)} p_0, i \geq 1. \end{cases} \tag{F.1}$$

Hence, we have $f(p_0) = \sum_{i=0}^{\infty} g_i(p_0)$. Compare $g_i$ with $h_i$, we obtain that $h_i(p_0) \geq g_i \geq 0$.

Besides, for $H_n(p_0)$, we have

$$H_n(p_0) = \sum_{i=0}^{n} \left( \frac{r - r\lambda p_0}{r + p_0} \right)^i p_0$$

which is exactly the sum of a geometric sequence with a common ratio in $(0, 1)$. Thus, this summation converges. Therefore, $f(p_0)$ is uniformly convergent.

In order to show that

$$f\,(p_0) = \sum_{i=0}^{\infty} \frac{dg_i}{dp_0}, \tag{F.2}$$

we need to show the uniform convergence of $\sum_{i=0}^{\infty} \frac{dg_i}{dp_0}$ further.

According to , we have the derivative of $p_i$.

$$\frac{dg_i}{dp_0} = \begin{cases} 1, i = 0, \\ -\frac{r(r\lambda+i)}{(r+ip_0)^2} g_{i-1} + \frac{r(1-\lambda p_0)}{(r+ip_0)} \frac{dg_{i-1}}{dp_0}, i \geq 1. \end{cases} \tag{F.3}$$

For each $i \geq 1$, we have

$$\left|\frac{dg_i}{dp_0}\right| \leq \frac{r(r\lambda+i)}{(r+ip_0)^2} g_{i-1} + \frac{r(1-\lambda p_0)}{(r+ip_0)} \left|\frac{dg_{i-1}}{dp_0}\right|. \tag{F.4}$$

Hence, the summation of $p_i$ can be scaled as the following.

$$\sum_{i=0}^{\infty} \left|\frac{dg_i}{dp_0}\right| \leq \sum_{i=1}^{\infty} \frac{r(r\lambda+i)}{(r+ip_0)^2} g_{i-1} + \sum_{i=1}^{\infty} \frac{r(1-\lambda p_0)}{(r+ip_0)} \left|\frac{dg_{i-1}}{dp_0}\right| + 1$$

$$\sum_{i=0}^{\infty} \left|\frac{dg_i}{dp_0}\right| \leq \sum_{i=1}^{\infty} \frac{r(r\lambda+i)}{(r+ip_0)^2} g_{i-1} + \sum_{i=0}^{\infty} \frac{r(1-\lambda p_0)}{(r+p_0)} \left|\frac{dg_i}{dp_0}\right| + 1$$

$$\frac{p_0(1+r\lambda)}{r+p_0} \sum_{i=0}^{\infty} \left|\frac{dg_i}{dp_0}\right| \leq \sum_{i=1}^{\infty} \frac{r(r\lambda+i)}{(r+ip_0)^2} g_{i-1} + 1$$

$$\frac{p_0(1+r\lambda)}{r+p_0} \sum_{i=0}^{\infty} \left|\frac{dg_i}{dp_0}\right| \leq \frac{1}{p_0} \sum_{i=1}^{\infty} g_{i-1} + 1$$

$$\sum_{i=0}^{\infty} \left|\frac{dg_i}{dp_0}\right| \leq \frac{r+p_0}{p_0^2(1+r\lambda)} \sum_{i=0}^{\infty} g_i + \frac{r+p_0}{p_0(1+r\lambda)}$$

Recall that $\sum_{i=0}^{\infty} g_i < \infty$, we can conclude that for any $p_0 \in (0,1)$,

$$\sum_{i=0}^{\infty} \left|\frac{dg_i}{dp_0}\right| < \infty$$

Hence, $\sum_{i=0}^{\infty} \frac{dg_i}{dp_0}$ is uniformly convergent. Thus, $f(p_0)$ is differentiable in $(0,1)$ and (F.2) stands.

□

Next step shows that $f(p_0)$ increases monotonically with $p_0 \in [0,1]$.

*Proof.* To be concise, our target is

$$\sum_{i=0}^{\infty} \frac{dg_i}{dp_0} > 0.$$

According to (F.1), we have the following recursive function

$$g_i = \frac{r(1-\lambda p_0)}{(r+ip_0)} g_{i-1}, \text{ for } i \geq 1. \tag{F.5}$$

Since $\lambda < 1$ and $0 \leq g_i \leq 1$, we have

$$\begin{cases} -\dfrac{r(r\lambda+i)}{(r+ip_0)^2} < 0 \\ \dfrac{r(1-\lambda p_0)}{(r+ip_0)} > 0 \\ g_{i-1} \geq 0. \end{cases}$$

According to (F.3), if $\frac{dg_{i-1}}{dp_0} < 0$, then $\frac{dg_i}{dp_0} < 0$. This implies that when $\frac{dg_k}{dt} < 0$, then $\frac{dg_i}{dt} < 0$ for all $i \geq k$.

Also, through deformation of (F.5), we obtain

$$(r+ip_0)g_i = r(1-\lambda p_0)g_{i-1}$$

$$\sum_{i=1}^{\infty} (r+ip_0)g_i = \sum_{i=1}^{\infty} r(1-\lambda p_0)g_{i-1}$$

$$r\sum_{i=1}^{\infty} g_i + p_0 \sum_{i=1}^{\infty} i g_i = r\sum_{i=0}^{\infty} g_i - r\lambda p_0 \sum_{i=0}^{\infty} g_i.$$

Thus, we have

$$\sum_{i=0}^{\infty} i g_i = r - r\lambda \sum_{i=0}^{\infty} g_i \tag{F.6}$$

and

$$\sum_{i=0}^{\infty} i \frac{dg_i}{dp_0} = -r\lambda \sum_{i=0}^{\infty} \frac{dg_i}{dp_0}. \tag{F.7}$$

Let $k$ be the number that when $0 \leq i \leq k$, $\frac{dg_i}{dp_0} \geq 0$, when $i \geq k+1$, $\frac{dg_i}{dp_0} < 0$ .

1. If $k = \infty$, then $\sum_{i=0}^{\infty} \frac{dg_i}{dp_0} > 0$.

2. If $k < \infty$, Then, we have

$$\sum_{i=0}^{\infty} k\frac{dg_i}{dp_0} > \sum_{i=0}^{\infty} i\frac{dg_i}{dp_0} = -r\lambda\sum_{i=0}^{\infty}\frac{dg_i}{dp_0}$$

$$(k + r\lambda)\sum_{i=0}^{\infty}\frac{dg_i}{dp_0} > 0$$

$$\sum_{i=0}^{\infty}\frac{dg_i}{dp_0} > 0.$$

Thus, we conclude that $f(p_0)$ increases monotonically with $p_0 \in [0, 1]$.

This completes the whole proof. □