

# Incentivizing Resource Pooling

Chen Chen<sup>1</sup>, Yilun Chen<sup>2</sup>, and Pengyu Qian<sup>3</sup>

<sup>1</sup>New York University Shanghai

<sup>2</sup>School of Data Science, Chinese University of Hong Kong, Shenzhen

<sup>3</sup>Daniels School of Business, Purdue University

cc8029@nyu.edu, chenylun@cuhk.edu.cn, qianp@purdue.edu

October 8, 2023

## Abstract

Resource pooling improves system efficiency drastically in large stochastic systems, but its effective implementation in decentralized systems remains relatively underexplored. This paper studies how to incentivize resource pooling when agents are self-interested, and their states are private information. Our primary motivation is applications in the design of decentralized computing markets, among others. We study a standard multi-server queueing model where each server is associated with an  $M/M/1$  queue and aims to minimize its time-average job holding and processing costs. We design a simple token-based mechanism where servers can earn tokens by offering help and spend tokens to request help from other servers, all in their self-interest. The mechanism induces a complex game among servers. We employ the fluid mean-field equilibrium (FMFE) concept to analyze the system, combining mean-field approximation with fluid relaxation. This framework enables us to derive a closed-form characterization of servers' FMFE strategies. We show that these FMFE strategies approximate well the servers' rational behavior. Leveraging this framework, we design the key element of the mechanism and present our main results: As the number of servers increases, the proposed mechanism incentivizes *complete* resource pooling—i.e., the system dynamics and performance under our mechanism match those under centralized control. We also extend our mechanism to settings with heterogeneous servers, and we show that our mechanism obtains near-optimal performance.

*Subject classifications:* Resource pooling, decentralized application, fluid mean-field equilibrium, dynamic games.

# 1 Introduction

Resource pooling is a fundamental concept in operations management and lies at the heart of broad applications. For example, data centers consolidate numerous devices to process large volumes of tasks efficiently. Meanwhile, modern healthcare systems commonly embrace multi-hospital networks to share specialized medical equipment and healthcare professionals within the network. The increased efficiency from resource pooling arises from the ability to direct idle servers to process jobs from overloaded ones. This advantage is exemplified by a classic scenario in which an  $M/M/N$  queueing system significantly reduces congestion (i.e., the job total in the system) compared to a collection of  $N$  independent  $M/M/1$  queues with the same total arrival rate.

In this paper, we study a *decentralized* setting and consider the problem of incentivizing resource pooling among  $N$  strategic servers that operate in self-interest. Specifically, each server receives independent streams of jobs that arrive at a rate of  $\lambda < 1$ , and can process jobs at a rate normalized to be one. Both serving and storing jobs incur costs. Serving a job costs  $c \geq 0$ , while a job waiting for service incurs a holding cost of one per unit of time. A server’s objective is to minimize its long-run average job holding and processing costs.

Such a problem is motivated by the increasing popularity of decentralized applications (or dApps). These digital applications operate on decentralized networks, normally blockchains, without the control of a central authority. We focus on the particular dApp of decentralized computing markets, exemplified by platforms such as Golem Network, iExec, and Akash Network, among others.<sup>1</sup> These blockchain-based dApps facilitate self-interested users to exchange computing resources. Specifically, users can rent out unused computing resources or request resources from others, all in their self-interest and guided by the market’s design.<sup>2</sup>

In the absence of a central planner and when servers are strategic, it remains uncertain whether the same level of resource pooling can be attained. Particularly, a server may act as a free rider anticipating that it will be helped later, and an idle server may lack the incentive to help others due to the presence of job processing costs. Moreover, the decentralized applications we consider present an additional challenge to the mechanism design, which is *limited information*. If the servers had complete information about one another (particularly the ability to monitor the actions and job counts of all other servers), the mechanism would be straightforward because servers can punish

---

<sup>1</sup>The market capitalizations of Golem Network, iExec, and Akash Network are approximately 170 million, 75 million, and 310 million US dollars, respectively, in August 2023.

<sup>2</sup>We assume homogeneous jobs and servers in our base model as a first step toward designing decentralized computing markets. In Section 7, we extend our mechanism and analysis to the setting with heterogeneous servers.

deviation from resource pooling. Specifically, if a server with no pending job refuses to help others even once, that server will be forever banned from participating in the system by all the other servers, thereby losing all potential future benefits of cooperation. Therefore, no server has an incentive to deviate from resource pooling, leading to an equilibrium where complete resource pooling is attained.

However, the assumption of complete information is only realistic when there are only a few servers. In contrast, in decentralized computing markets (our motivating example), the number of participants is usually large. Furthermore, in blockchain-based applications, participants are typically anonymous, and their states and actions are obscured from others, intensifying the flaw of complete information assumption. As a result, we consider a setting where servers possess only limited information about one another, specifically: (i) servers' states and actions are hidden from others, and (ii) a server may not know the exact total of servers, except knowing that it is relatively large. Such a limited information setting renders punishing deviation unattainable; thus, an alternative mechanism must be developed. Furthermore, it is a priori unclear whether *complete* resource pooling—i.e., the same extent of resource pooling under centralized control—can be (almost) achieved in this limited setting. This raises the following natural research question:

*Can we design a simple mechanism that incentivizes (almost) complete resource pooling in a limited information setting (such as decentralized computing markets on blockchains)?*

**Remark 1.1** (Other Applications). In addition to the decentralized computing markets, many other applications also involve incentivizing resource pooling in a limited information setting. For example, consider a regional healthcare system that seeks to incentivize cooperation among hospitals, so that a hospital can outsource patients to a nearby one during periods of over-occupation. Notably, these hospitals may belong to different entities (therefore will strategize), and observing the congestion level of a hospital externally can be challenging.

## 1.1 Our Contributions

As our main contribution, we answer the above question in the affirmative. Specifically, we develop a simple token-based mechanism. Through careful design of the mechanism, we show that *complete* resource pooling can be achieved when the number of servers  $N$  is large. Remarkably, within our mechanism, it is each server's best strategy to (i) process a job from its queue if one is present and help others otherwise, and (ii) request help whenever a job arrives (and the server has tokens). Consequently, both the system dynamics and performance match those under centralized control.

In our token-based mechanism, a server can request help from others for processing its jobs at the cost of one token per requested job at any time. Meanwhile, a server can allocate its next unit of available computing power to help others and earn a token with a pre-determined probability  $\phi \in (0, 1)$ . Decisions regarding the request and provision of help are made by each server based on their self-interest. The mechanism matches servers providing help to those requesting help in a first-come-first-serve order by managing a (virtual) shared pool. Since servers interact through the shared pool, their dynamics are only weakly coupled, which simplifies the analysis of our mechanism. Further details on our mechanism are provided in Section 2.2.1.

The mechanism introduces a complex stochastic, incomplete-information, dynamic game among servers that is challenging to analyze. To address this, we adopt the equilibrium notion of fluid mean-field equilibrium (FMFE), which is tractable and provides a good approximation to the servers' strategic behaviors in our mechanism. Specifically, our approximation combines the mean-field approximation with a fluid relaxation in a similar spirit to Balseiro et al. (2015). First, we consider a mean-field approximation as a large market approximation. When the number of servers is large, there is little value in tracking the dynamics of all the other servers (even if a server can do so). Instead, it is plausible to assume that the fluctuations of servers' states average out, and as a result, the empirical distribution of the shared pool remains roughly constant over time. The mean-field approximation assumes that servers optimize only with respect to long-run average estimates of the congestion level of the shared pool. Second, we consider a fluid relaxation to handle the dynamics of servers' tokens. The fluid relaxation requires that the tokens' flow balance constraint holds only in expectation; such relaxation is tight when a server can possess many tokens, which is the case in our mechanism.

The two approximations simplify a server's problem significantly, enabling us to derive a closed-form characterization of a server's best response in FMFE. We show that the FMFE best-response strategy is a threshold policy in a server's queue length. Specifically, a server requests help only when its queue length exceeds a certain threshold, which depends on the token-earning probability  $\phi$  and the endogenous waiting time in the shared pool, and a server offers help only when its queue is empty. Moreover, we show that FMFE provides a good approximation to the rational behavior of the servers as the number of servers  $N$  grows large, justifying our approximation methodology. Specifically, for any value of  $\phi$ , it constitutes an approximate Nash equilibrium when all the servers follow the FMFE strategies. In other words, the benefit from unilaterally deviating to other strategies becomes negligible as the number of servers grows large.

Leveraging the FMFE framework, we determine the optimal value of the token-earning proba-

bility  $\phi$ , which is the key element of our mechanism. We show that the optimal choice of  $\phi$  equals  $\lambda$ , the job arrival rate at each server.<sup>3</sup> Moreover, with this value, complete resource pooling is an FMFE under our mechanism when the number of servers is large; thus, the resulting system dynamics and performance match those under centralized control.

Moreover, we demonstrate via numerical results that complete resource pooling is an approximate equilibrium when the token-earning probability  $\phi$  equals  $\lambda$  even in small markets with only a few servers, providing further practical support to our proposed mechanism. In small markets, a server may have an incentive to continually infer the congestion level of the shared pool, so as to request or offer help only when the congestion is minimal. Nevertheless, we show that the benefit of doing so is low even when there are only very few servers, and even when the deviating server can monitor the congestion level of the shared pool completely.

Finally, we extend our mechanism and performance analysis to a general setting with heterogeneous servers (but still require that jobs are stochastically identical). Our former analyses readily indicate that by setting the token-earning probability  $\phi$  to be the largest utilization factor  $\rho_i$  among servers,<sup>4</sup> the total number of jobs in the system is at most a constant times the total number of jobs in the centralized setting when the number of servers is large and under mild regularity conditions on the problem primitives. Thus, the complete resource pooling is almost obtained.

The rest of the paper is organized as follows. Section 1.2 reviews some related work. We formulate the problem in Section 2. In Section 3, we introduce the FMFE of the problem. Section 4 characterizes the FMFE strategies in closed form. Section 5 presents the main results of our token-based mechanism. Specifically, we show that complete resource pooling is achieved under a proper choice of the token-earn probability  $\phi$  and when the number of servers is large. Section 6 justifies our approximation methodology by showing that all the servers adopting the FMFE strategies forms an approximate Nash equilibrium. Furthermore, in Section 6.3, we show via numerical results that complete resource pooling is an approximate equilibrium under our mechanism, even for small markets with only a few servers. Section 7 extends our mechanism and performance analysis to a more general setting with heterogeneous servers. We conclude in Section 8.

---

<sup>3</sup>Recall that we normalize the servers' processing rates to be one.

<sup>4</sup>Specifically,  $\phi = \max_{i \in [N]} \rho_i$  with  $\rho_i = \lambda_i / \mu_i$ , where  $\lambda_i$  and  $\mu_i$  are the job arrival and job processing rates at server  $i$ .

## 1.2 Related Literature

**Power of Resource Pooling** Resource pooling improves system efficiency in various applications, e.g., inventory pooling (Eppen 1979), manufacturing flexibility (Tsitsiklis and Xu 2017, Shi et al. 2019), retail operations (Elmachtoub et al. 2015), and transportation (Balseiro et al. 2021). Tsitsiklis and Xu (2013) show that even a little resource pooling can significantly reduce congestion in large systems. These work primarily focus on centralized settings. In contrast, we design a simple mechanism to incentivize complete resource pooling in a decentralized setting.

**Decentralized Setting with Two Servers** Hu and Caldentey (2023) examine a model similar to ours but focus on two (heterogeneous) servers. In their model, jobs that cannot be served upon arrival abandon the system. They analyze the equilibrium by assuming that both servers adopt a family of trading-favors strategies. Characterizing an equilibrium with just two servers proves challenging. In contrast, we consider a scenario with many servers. We use the approximation notion of FMFE to provide a crisp characterization of servers’ strategic behaviors in our mechanism. We demonstrate that the performance of our mechanism approaches that under centralized control as the number of servers increases.

**Mean-Field Equilibrium** A number of recent papers have utilized mean-field equilibrium to study complex, large-scale operational problems (e.g., Iyer et al. 2014, Balseiro et al. 2015, Kanoria and Saban 2021, Arnosti et al. 2021). The mean-field equilibrium relaxes the informational requirements of agents, allowing them to know only the aggregate description of the system (in our case, the congestion level of the shared pool in the steady state), which makes it tractable and appealing, and aligns well with our limited information setup. As we show in Section 6 (and similarly demonstrated in the aforementioned papers), due to the averaging effects, mean-field equilibrium provides accurate approximations of servers’ strategic behaviors as the number of servers increases.

The specific fluid mean-field equilibrium (FMFE) we consider incorporates a second fluid relaxation to better handle each server’s stochastic optimization problem. This is similar to Balseiro et al. (2015), but we address very different problems. Specifically, Balseiro et al. (2015) focus on repeated auctions with budget-constrained bidders. The fluid relaxation in their FMFE allows the bidders to satisfy the budget constraint only in expectation. The resultant optimal bidding strategy takes a simple form: a bidder shades her value by a constant factor. In contrast, we investigate a decentralized queueing system. Our fluid relaxation allows servers to possess a negative number of tokens, but the in-flow and out-flow of tokens need to be equal in expectation; consequently, a

server’s best response is a threshold policy in its queue length.

**Scrip System** Several papers have examined scrip (or token) systems in a different model, e.g., Kash et al. (2007), Kash et al. (2015), Johnson et al. (2014), and Bo et al. (2018). In their model, a random player requests service in each period, and a mechanism specifies a service provider from those willing to offer help. Successful service provision occurs if the requester has at least one scrip, in which case, one scrip transfers from the requester to the provider.

Our work considers a distinct queueing model to investigate resource pooling in a decentralized setting, which requires a significantly different analysis. Beyond that, Kash et al. (2007) and Kash et al. (2015) analyze a random provider selection rule, which is suboptimal in the centralized setting. Johnson et al. (2014) and Bo et al. (2018) examine the minimum scrip selection rule (i.e., selecting the player with the least number of scrips to provide help), which is optimal in the centralized setting. However, the minimum scrip selection rule requires publicly accessible information about each player’s scrip quantity, making it hard to implement in a limited information setup as we consider.

**Further Related Work** Other researchers have investigated server cooperation from a cooperative game perspective, e.g., Anily and Haviv (2010), Anily and Haviv (2014), and Karsten et al. (2015). These work consider a cooperative game among servers and investigate how to allocate the system’s total costs among servers to sustain cooperation (i.e., finding the core of the game). The implementation still requires that servers in a group possess complete information about one another, to ensure that every server in the group makes maximum effort to serve incoming jobs.

Finally, our work complements nicely the literature on supermarket games and their variations (e.g., Xu and Hajek 2013 and Yang et al. 2019). In their problem, customers, rather than servers, are the strategic agents. Each customer selects the number of queues to sample upon arrival and joins the shortest queue from the sample to minimize the total waiting and sampling costs; this introduces a game among customers. Many of these work also adopt the mean-field equilibrium for a tractable analysis.

### 1.3 Notation and Terminology

We let  $\mathbb{N}$  denote the set of nonnegative integers and  $\mathbb{N}_+$  the set of strictly positive integers. For any two integers  $a, b \in \mathbb{N}$  with  $a \leq b$ , we let  $[a : b] = \{a, a + 1, \dots, b - 1, b\}$  denote a sequence of integers starting from  $a$  and ending with  $b$  and we denote  $[n] = [1 : n]$  for any  $n \in \mathbb{N}_+$ . For any nonnegative

real number  $x \in \mathbb{R}_+$ , we let  $\lfloor x \rfloor \in \mathbb{N}$  denote the floor of  $x$ , which is the greatest integer less than or equal to  $x$ ; and we let  $\lceil x \rceil \in \mathbb{N}$  denote the ceiling of  $x$ , which is the least integer greater than or equal to  $x$ . For any real number  $x \in \mathbb{R}$ , we let  $(x)^+ \triangleq \max\{x, 0\}$  denote the maximum of  $x$  and 0.

## 2 Problem Formulation

We consider  $N$  servers that operate in their self-interest. Jobs independently arrive at each server following a Poisson process with a rate of  $\lambda < 1$ . Each server manages a queue to store incoming jobs awaiting processing. To model the servers' job processing processes, we adopt a Poisson clock model following Tsitsiklis and Xu (2013) and Spencer et al. (2014) to simplify our analysis. Specifically, each server receives a stream of “capacity units” that follows an independent Poisson process, with a rate normalized to one.<sup>5</sup> When a capacity unit arrives at server  $i \in [N]$ , the server can either serve a job from its queue or other places by incurring a cost  $c \geq 0$ , or waste the capacity unit at zero cost. Served jobs leave the system immediately.

Both serving and storing (or holding) jobs incur costs. Serving a job costs  $c \geq 0$ , and each job waiting for processing costs one per unit of time. Each server's objective is to minimize its expected time-average total cost—job processing cost plus holding cost—over an infinite horizon. Time is assumed to be continuous.

### 2.1 System Performance under Centralized Control

We first consider the centralized setting as a benchmark. In this setting, a central planner has complete control over the servers and minimizes the total cost in the system; since servers are stochastically identical, this is equivalent to minimizing the expected time-average total cost per server. To achieve this goal, the planner would direct an idle server to process jobs from other queues (e.g., a server with the longest queue). The dynamics of the total number of jobs in the system is an  $M/M/1$  queue under the Poisson clock model, with the job arrival and processing rates being  $N\lambda$  and  $N$ , respectively. Consequently, the expected total number of jobs in the stationary distribution is a constant of  $\frac{\lambda}{1-\lambda}$ . Therefore, the time-average cost per server is

$$V_c(N) \triangleq \frac{\lambda}{(1-\lambda)N} + c\lambda,$$

---

<sup>5</sup>Although Tsitsiklis and Xu (2013) and Spencer et al. (2014) refer to these capacity units as “service tokens,” we use a different name to prevent any possible confusion with the artificial currency tokens that we will introduce later as part of our mechanism.



where  $\frac{\lambda}{(1-\lambda)N}$  is the expected number of jobs in a server’s queue and  $c\lambda$  is a server’s time-average job processing cost.<sup>6</sup> This improves upon

$$V_1 \triangleq \frac{\lambda}{1-\lambda} + c\lambda,$$

the time-average cost of a server when it operates independently; the improvement is significant, particularly when there is a large number of servers.

## 2.2 Mechanism for Decentralized Setting

In the absence of a central planner and when servers are strategic, a server may act as a free rider if it knows it will be helped later. In addition, an idle server may lack the incentive to help others due to the presence of job processing costs. Therefore, it remains uncertain whether we can design a mechanism to attain the same time-average cost  $V_c(N)$  as in the centralized setting. In this section, we develop a simple token-based mechanism, and we show later that such a mechanism encourages server cooperation and ensures the first-best time-average cost  $V_c(N)$  is achieved in large systems.

**Limited Information Setup** As motivated in Section 1, we consider a setting where each server possesses only limited information on the other servers. In particular, each server is unaware of the other servers’ states or actions, and a server does not require precise knowledge of the total number of servers  $N$ , except knowing that it is relatively large. This renders punishing deviation from resource pooling unattainable. However, as we shall demonstrate, a simple token-based mechanism achieves complete resource pooling in this restrictive information setting. We describe our token-based mechanism in Section 2.2.1 and analyze its performance in the rest of the paper.

### 2.2.1 Incentivizing Resource Pooling via Artificial Currency Tokens

In this section, we describe a token-based mechanism that incentivizes resource pooling.

**An Overview of the Mechanism** In our mechanism, each strategic server has the autonomy to make its own decisions about requesting help from or providing help to other servers, in addition to processing its jobs. The requests and provisions of help are matched through a shared pool in a first-come-first-serve (FCFS) order. To mitigate free riding and promote cooperation, we introduce

---

<sup>6</sup>By the “Poisson Arrivals See Time Averages (PASTA)” property, the long-run proportion of capacity units who, upon arrival, find a job in the system equals  $\lambda$ . Thus, the time-average job processing cost of a server equals  $c\lambda$ .

a token system. Specifically, servers possess artificial currency tokens (or scrips). Requesting help requires one token, whereas a server can earn tokens by offering help.

In the following, we let  $Q_i(t)$  denote the queue length (i.e., number of jobs) of server  $i \in [N]$  and  $Q_0(t)$  the queue length of the shared pool at time  $t$ .

**Servers' Actions** Each server makes strategic decisions about requesting help from or providing help to other servers. The decisions of a server are as follows.

First, every server  $i \in [N]$  can request help from other servers for processing its jobs at any time  $t$ . We let  $R_i(t) \in [0:Q_i(t)]$  denote the number of jobs the server requests help with at time  $t$ . The mechanism enforces that requests are irrevocable. Once requested, the  $R_i(t)$  jobs are relocated from the queue of server  $i$  to the shared pool (i.e.,  $Q_i(t)$  decreases by  $R_i(t)$ , and  $Q_0(t)$  increases by  $R_i(t)$ ). The server is informed once a requested job gets served.

On the other hand, whenever a capacity unit arrives at server  $i$  at time  $t$ , the server must make an immediate decision  $Y_i(t) \in \{0, 1, \emptyset\}$  to allocate the capacity unit. In particular,

- $Y_i(t) = 1$  indicates that the server would like to provide help to others. By doing so, the server contributes its capacity unit to the shared pool without recall. If the shared pool contains a job (i.e.,  $Q_0(t) \geq 1$ ), the job that is waiting longest in the pool gets served (hence, an FCFS rule is used), and it incurs a cost  $c$  for the server to process the job. Otherwise, if the shared pool is empty, the capacity unit is wasted, and it incurs no job processing cost to the server.
- $Y_i(t) = 0$  indicates that the server uses the capacity unit to serve a job from its queue, incurring a job processing cost of  $c$ .
- $Y_i(t) = \emptyset$  indicates that the server would be idle and waste the capacity unit with no cost.

**The Shared Pool** The mechanism does not reveal the queue length of the shared pool to servers. However, servers can partially infer the congestion level of the shared pool in two ways. First, when a server offers help, the server knows whether the shared pool is empty because a non-empty shared pool incurs a processing cost of  $c$ , whereas an empty shared pool does not. Secondly, servers receive notifications when a job relocated to the shared pool gets served; this allows them to infer information about the shared pool based on the status of relocated jobs.

**The Token System** Finally, we introduce a token system to incentivize server cooperation and mitigate free-riding. Specifically, each server is initially endowed with a certain number of tokens,<sup>7</sup> which it can use to pay for help or receive as a reward for providing help. The protocol for paying and earning tokens is as follows:

- When a server requests help (i.e.,  $R_i(t) \geq 1$ ), it costs one token per requested job.
- (The Mining Process) A server that offers help (i.e.,  $Y_i(t) = 1$ ) is rewarded with one token with a pre-determined probability  $\phi \in (0, 1)$  independent of whether the pool is empty or not.

By the design of the mechanism, acquiring a token is independent of the success of helping others (which corresponds to serving a job from the shared pool). Such a design simplifies the analysis of our mechanism, because the dynamics of servers will be independent if every server's strategy hinges only on its individual state. Note that the total number of tokens in the system is not constant by the design of the mining process. The value of the token-earning probability  $\phi$  is pivotal to our mechanism, and we determine its optimal value in Section 5.

Finally, we denote by  $S_i(t)$  the number of tokens held by server  $i$  at time  $t$ . We impose that  $0 \leq S_i(t) \leq C$  for some integer  $C \in \mathbb{N}_+$  and for all  $i \in [N]$  and  $t \geq 0$ , i.e., a server's token count is nonnegative and bounded from above by a constant  $C$ . We specify the value of  $C$  in Section 6.

**System Dynamics and Server's Problem** For each server  $i \in [N]$ , we let  $N_i^\lambda(t)$  and  $N_i^c(t)$  denote the number of jobs and capacity units that arrive by the end of time  $t$ , and  $\tau_{ij}^c$  the arrival time of the  $j$ -th capacity unit. Additionally, we let  $N_i^R(t)$  denote the number of jobs server  $i$  requests help for by the end of time  $t$ , and  $\tau_{ij}^R$  and  $w_{ij}$  the time that server  $i$  requests help for the  $j$ -th job and the waiting time of the job in the shared pool before being served. The evolution of the queue lengths and token counts can be expressed as follows:

$$\begin{aligned}
Q_i(t) &= Q_i(0) + N_i^\lambda(t) - N_i^R(t) - \sum_{j \leq N_i^c(t)} \mathbb{1}[Y_i(\tau_{ij}^c) = 0], \forall i \in [N], t \geq 0, \\
Q_0(t) &= Q_0(0) + \sum_{i \in [N]} N_i^R(t) - \sum_{i \in [N]} \sum_{j \leq N_i^c(t)} \mathbb{1}[Y_i(\tau_{ij}^c) = 1] \cdot \mathbb{1}[Q_0(\tau_{ij}^c) \geq 1], \forall t \geq 0, \\
S_i(t) &= S_i(0) - N_i^R(t) + \sum_{j \leq N_i^c(t)} \zeta_{ij} \cdot \mathbb{1}[Y_i(\tau_{ij}^c) = 1], \forall i \in [N], t \geq 0, \\
Q_i(t) &\geq 0, \forall i \in [0 : N], t \geq 0, \quad 0 \leq S_i(t) \leq C, \forall i \in [N], t \geq 0,
\end{aligned}$$

---

<sup>7</sup>The specific value is not essential as one considers the time-average-cost objective.

where  $\zeta_{ij}$  are independent binary variables with a mean value of  $\phi$ , which indicate if server  $i$  is awarded a token if it offers help at time  $\tau_{ij}^c$ .

Finally, we can express the cumulative cost of server  $i$  until time  $t$  as follows,

$$\begin{aligned} V_i(t) = & \int_0^t Q_i(s) ds + \sum_{j \leq N_i^R(t)} \min \left\{ w_{ij}, t - \tau_{ij}^R \right\} \\ & + c \sum_{j \leq N_i^c(t)} \left\{ \mathbb{1}[Y_i(\tau_{ij}^c) = 0] + \mathbb{1}[Y_i(\tau_{ij}^c) = 1] \cdot \mathbb{1}[Q_0(\tau_{ij}^c) \geq 1] \right\} \end{aligned}$$

where the first two terms represent the waiting costs for jobs in the queue and those relocated to the shared pool, and the third term accounts for the cost of processing jobs from the queue and the shared pool. Each server's objective is to minimize its time-average cost  $\limsup_{t \rightarrow \infty} \mathbb{E}[V_i(t)]/t$  in the dynamic game induced by the mechanism. We remark that waiting times  $\{w_{ij}\}$  in the shared pool depend on the strategies of all servers.

### 3 Equilibrium of Stochastic Game

The mechanism described in Section 2.2.1 gives rise to a complex stochastic, incomplete-information, dynamic game among servers. The perfect Bayesian equilibrium (PBE) is a widely used solution concept for dynamic games with incomplete information. In essence, each server is associated with a strategy and a belief system in PBE. A strategy is a function that maps a server's current state and past observations to decisions on requesting/providing help or serving its jobs. Additionally, each server must form beliefs about the states of the shared pool and all other servers. A set of strategies and belief systems forms a PBE if the following holds. First, given a server's beliefs and the strategies of all other servers, the server's strategy should minimize its expected continuation cost. Secondly, the beliefs held by each server must be consistent with the servers' equilibrium strategies and satisfy Bayes' rule whenever possible.

Although PBE has been a standard solution concept for stochastic games, multiple works in the literature (Weintraub et al. 2008, Iyer et al. 2014, Balseiro et al. 2015, among others) have highlighted significant obstacles with the PBE approach, mainly when dealing with many players. First, servers solve a dynamic programming problem to determine the best responses. Since servers maintain beliefs about the state of all other servers in PBE, the dimension of the optimization problem grows very quickly as the number of servers increases, making equilibrium strategies quickly intractable both analytically and computationally. Secondly, as the number of servers increases,

it becomes hard to believe that servers can, in fact, maintain beliefs on the states of all other servers, as such a level of sophistication imposes a stringent rationality assumption on the servers. Given the intricacy of the PBE approach, we instead consider an approximate equilibrium concept, termed fluid mean-field equilibrium (FMFE), to facilitate the analysis. The FMFE combines the widely adopted mean-field approximation with a fluid relaxation, as we formally introduce below.

### 3.1 Mean-Field and Fluid Approximation

When deciding whether to request help, a server would like to estimate the waiting time in the shared pool if it relocates a job there. Meanwhile, a server considering whether to offer help would take into account the probability that the shared pool is empty, in which case the server has an opportunity to earn a token without incurring a cost for serving a job from the shared pool. However, when the number of servers is large, it is reasonable to assume that the fluctuations of servers' states average out, and their empirical distribution remains roughly constant over time. Consequently, the distribution of the shared pool's queue length also remains constant. Furthermore, with a large number of servers, the impact of an individual server on the shared pool becomes negligible. These factors motivate us to consider an approximation methodology named mean-field approximation, in which each of the  $N$  servers optimizes its payoff by assuming that the state of the shared pool is fixed at its long-run average. Specifically, this simplifies a server's problem in two ways:

1. The expected waiting time of a job in the shared pool becomes a constant, which we denote by  $w \geq 0$ . The value of  $w$  will be determined endogenously by an equilibrium.
2. The probability that the shared pool is non-empty also becomes a constant, whose value is precisely  $\phi$ , the probability of earning a token when offering help, as we explain in Remark 3.1.

**Remark 3.1** (Probability that Shared Pool is Non-Empty when Offering Help). The long-run average probability that a capacity unit contributing to the shared pool serves a job (which corresponds to the pool being non-empty at that moment) equals  $\phi$ . To see this, note that it is in each server's best interest to equate the rate of earning and spending tokens. A token only holds value when used, and earning tokens can incur costs; therefore, there is no benefit in accumulating more tokens than necessary. Moreover, by our mechanism, the rate of spending tokens equals the rate of requesting help, and the rate of earning tokens is  $\phi$  times the rate of offering help. Therefore, at each server, the rate of requesting help is  $\phi$  times the rate of offering help. As a result, the rate at which jobs relocate to the shared pool (which equals the total requesting help rates across servers) is  $\phi$  times

the rate at which capacity units join the shared pool (which equals the total offering help rates across servers). Consequently, the probability that a capacity unit contributed to the shared pool serves a job is precisely  $\phi$  in the long-run average.

**Simplification from Mean-Field Approximation** Since servers optimize their own objectives holding the state of the shared pool constant in the mean-field problem, their optimal strategies depend only on their individual state—i.e., a server’s queue length and the number of tokens a server possesses; we call such servers oblivious and their strategies oblivious strategies (following Weintraub et al. 2008). Furthermore, since the state of the shared pool is treated as constant, it is without loss of optimality to assume that a server requests help only when a job arrives, as we demonstrate in Lemma 3.1.

**Lemma 3.1.** *In the mean-field problem, it is optimal to request help only when a job arrives, and only request help for the incoming job.*

We prove Lemma 3.1 in Appendix A.1. Therefore, when a job arrives at server  $i$  at time  $t$ , the server makes a decision  $X_i(t) \in \{0, 1\}$  such that

- $X_i(t) = 1$  indicates that the server requests help for processing the job (and relocates the job to the shared pool without recall);
- $X_i(t) = 0$  indicates that the server adds the job to its queue without requesting help.

The decisions  $X_i(t)$  together with  $Y_i(t)$  defined in Section 2.2.1 constitute a strategy of server  $i$  in the mean-field problem.

We now consider a representative server  $i$  in the mean-field problem. Since the server makes decisions only when either a job or capacity unit arrives, we can consider an equivalent embedded discrete-time model of the server’s problem, where in each period, either a job or a capacity unit arrives, with probability  $\frac{\lambda}{1+\lambda}$  and  $\frac{1}{1+\lambda}$ , respectively.<sup>8</sup> With some abuse of notation, we instead let  $t$  be the index of the discrete periods. The decisions in period  $t$  are: (a) when a new job arrives, we let  $X_i(t) = 1$  if the server relocates the job to the pool, and  $X_i(t) = 0$  if the server adds the job to its queue; (b) when a capacity unit arrives, we let  $Y_i(t) = 1$  if the server offers help to the shared pool,  $Y_i(t) = 0$  if the server serves a job from its queue, and  $Y_i(t) = \emptyset$  if the server opts to be idle.

---

<sup>8</sup>The continuous-time model and its embedded discrete-time model are equivalent because, under any stationary policy, the limiting distributions of the continuous-time and discrete-time Markov chains converge to the same stationary distribution (which may depend on the initial state if the Markov chain has multiple recurrent classes).

The time-average-cost optimization problem solved by server  $i$  in the mean-field approximation, when the shared pool waiting time is  $w$ , can be expressed in (1)

$$\begin{aligned}
V(w) = \min_{\pi \in \Pi} \quad & \lim_{T \rightarrow \infty} \frac{1}{T} \cdot \mathbb{E} \left\{ \sum_{t=1}^T \left( \frac{1}{1+\lambda} \cdot Q_i(t) + \xi_{i1,t} \cdot w \cdot \mathbb{1}[X_i^\pi(t) = 1] \right. \right. \\
& \left. \left. + \xi_{i0,t} \cdot c \cdot \left( \mathbb{1}[Y_i^\pi(t) = 0] + \phi \cdot \mathbb{1}[Y_i^\pi(t) = 1] \right) \right) \right\} \\
\text{s.t.} \quad & Q_i(t+1) = Q_i(t) + \xi_{i1,t} \cdot \mathbb{1}\{X_i^\pi(t) = 0\} - \xi_{i0,t} \cdot \mathbb{1}\{Y_i^\pi(t) = 0\}, \forall t \geq 1, \\
& S_i(t+1) = S_i(t) - \xi_{i1,t} \cdot \mathbb{1}\{X_i^\pi(t) = 1\} + \zeta_{i,t} \cdot \xi_{i0,t} \cdot \mathbb{1}\{Y_i^\pi(t) = 1\}, \forall t \geq 1, \\
& Q_i(t), S_i(t) \in \mathbb{N}, S_i(t) \leq C, \forall t \geq 1.
\end{aligned} \tag{1}$$

In (1), we denote by  $\Pi$  the set of all non-anticipative policies (that is, policies that can only depend on the observed history),  $\xi_{i1,t}$  and  $\xi_{i0,t}$  independent binary variables with  $\xi_{i1,t} = 1$  ( $\xi_{i0,t} = 1$ ) if a job (a capacity unit) arrives at server  $i$  in period  $t$ , and  $\zeta_{i,t}$  independent binary variables with a mean value of  $\phi$ , which indicate that the server earns a token from offering help in period  $t$ . In the objective function,  $\frac{Q_i(t)}{1+\lambda}$  represents the holding cost for jobs in server  $i$ 's queue in period  $t$  (note that the length of a period is  $\frac{1}{1+\lambda}$  in expectation) and  $\phi$  in the last term is the probability that a capacity unit contributed to the shared pool serves a job and hence incurs a cost of  $c$ . The first and second constraints model the dynamics of the numbers of jobs and tokens, respectively, of server  $i$ . Finally, the average cost of server  $i$  in the continuous-time model equals  $(1+\lambda)V(w)$ .

**Fluid Relaxation** The mean-field problem (1) can be formulated as a two-dimensional stochastic dynamic program (DP), but is still challenging to solve. To better handle a server's problem, we introduce a second level of approximation where we allow the number of tokens to be negative and go beyond the upper bound  $C$ , and we only require that the tokens satisfy the flow balance constraint, i.e., the expected rates of earning and spending tokens are equal in the long-run average.

We provide the fluid relaxation of (1) in (2) following the same notation as in (1).

$$\begin{aligned}
V^F(w) = \min_{\pi \in \Pi} \quad & \lim_{T \rightarrow \infty} \frac{1}{T} \cdot \mathbb{E} \left\{ \sum_{t=1}^T \left( \frac{1}{1+\lambda} \cdot Q_i(t) + \xi_{i1,t} \cdot w \cdot \mathbb{1}[X_i^\pi(t) = 1] \right. \right. \\
& \left. \left. + \xi_{i0,t} \cdot c \cdot \left( \mathbb{1}[Y_i^\pi(t) = 0] + \phi \cdot \mathbb{1}[Y_i^\pi(t) = 1] \right) \right) \right\} \\
\text{s.t.} \quad & \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbb{P}[X_i^\pi(t) = 1] = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \phi \cdot \mathbb{P}[Y_i^\pi(t) = 1], \\
& Q_i(t+1) = Q_i(t) + \xi_{i1,t} \cdot \mathbb{1}\{X_i^\pi(t) = 0\} - \xi_{i0,t} \cdot \mathbb{1}\{Y_i^\pi(t) = 0\}, \forall t \geq 1, \\
& S_i(t+1) = S_i(t) - \xi_{i1,t} \cdot \mathbb{1}\{X_i^\pi(t) = 1\} + \zeta_{i,t} \cdot \xi_{i0,t} \cdot \mathbb{1}\{Y_i^\pi(t) = 1\}, \forall t \geq 1, \\
& Q_i(t) \in \mathbb{N}, S_i(t) \in \mathbb{Z}, \forall t \geq 1.
\end{aligned} \tag{2}$$

In (2), the first constraint requires that the long-run average rates of earning and spending tokens are equal. By Remark 3.1, any optimal policy of (1) satisfies this constraint; therefore, (2) is a valid relaxation to (1). Lemma 3.2 shows that (2) can be formulated as a one-dimensional DP where an optimal policy depends only on a server's queue length  $Q_i(t)$ .

**Lemma 3.2.** *The problem (2) can be formulated as a one-dimensional problem where the optimal policy depends only on the number of jobs in the queue and is independent of the number of tokens the server possesses.*

We prove Lemma 3.2 in Appendix A.2. The mean-field and fluid approximations simplify a server's problem substantially. In Section 4.1, we derive a closed-form solution to the fluid mean-field problem (2), which significantly aids our subsequent analysis. Moreover, in Section 6, we show that the optimal policies of the fluid mean-field problem provide accurate approximations of a server's strategic behavior in the original problem.

### 3.2 Fluid Mean-Field Equilibrium

We now introduce the concept of fluid mean-field equilibrium (FMFE) based on the fluid mean-field problem presented in Section 3.1. The FMFE requires a consistency check: the presumed shared pool waiting time  $w$  must arise from the optimal strategies derived from (2). Specifically, a pair  $(\pi, w)$  that contains a waiting time  $w$  and an oblivious strategy  $\pi$  that depends only on a server's queue length, constitutes an FMFE if:

1.  $\pi$  is an optimal solution to  $V^F(w)$ ;



2. The long-run average waiting time in the shared pool is  $w$  when all the  $N$  servers follow the policy  $\pi$ .

More formally, we define the mapping:

$$H(w) \triangleq \{w' \in \mathbb{R}_+ : \exists \text{ policy } \pi \text{ optimal to } V^F(w) \text{ such that if all servers employ } \pi, \\ \text{the shared pool waiting time is } w' \text{ in the long-run average}\}.$$

For  $(\pi, w)$  to be an FMFE, we must have  $w \in H(w)$ , i.e., the long-run average waiting time in the shared pool resulting from the servers' best-response strategies equals the conjectured waiting time.

## 4 Characterization of Fluid Mean-Field Equilibrium

In this section, we characterize FMFE. We first derive a closed-form expression to a server's best response in the fluid mean-field problem (2) in Section 4.1. Then, in Section 4.2, we show that under mild conditions, in all FMFE where servers cooperate, the endogenous waiting time  $w$  diminishes to zero when the number of servers  $N$  increases.

### 4.1 Server's Best Response in FMFE

In this section, we solve (2) in closed form for any shared pool waiting time  $w \geq 0$ , thus completely characterizing a server's best response in FMFE. In particular, we show that a server's best response is a threshold policy in its queue length, whereby a server requests help only when its queue length exceeds a certain threshold and offers help only when its queue is empty.

We first define a function  $m(z)$  with  $m(0) = 0$  and

$$m(z) = \frac{1}{1-\lambda} \cdot \left( z - \sum_{i=1}^z \lambda^i \right), \quad \forall z \in \mathbb{N}.$$

Proposition 4.1 provides some straightforward properties for the function  $m(z)$ .

**Proposition 4.1.** *Let  $\Delta m(z) = m(z) - m(z-1)$  be the difference of  $m(z)$  of two adjacent integers. Then, we have  $m(0) = 0$ ,  $m(1) = 1$ , and  $\Delta m(z+1) \geq \Delta m(z) \geq \Delta m(1) = 1$  for all  $z \geq 1$ .*

We now characterize the optimal policy of (2) for any  $w \geq 0$  in Lemma 4.2.

**Lemma 4.2** (Optimal Policy of (2)). *Let  $k \in \mathbb{N}$  be the unique integer that satisfies  $\phi \in (\lambda^{k+1}, \lambda^k]$ . An optimal policy of (2) is as follows.*

1. When  $w < m(k)$  (Case One):

- (a) When a job arrives, route it to the shared pool if  $q \geq k$  and add it to the queue if  $q \leq k - 2$ . Otherwise, if  $q = k - 1$ , route the job to the shared pool with probability  $p \triangleq \frac{\phi - \lambda^{k+1}}{\lambda^k - \lambda^{k+1}} \in [0, 1]$  and add it to the queue with probability  $1 - p = \frac{\lambda^k - \phi}{\lambda^k - \lambda^{k+1}}$ .
- (b) When a capacity unit arrives, serve a job from the queue if  $q \geq 1$  and offer help to the shared pool if  $q = 0$ .

2. When  $w \geq m(k)$  (Case Two): find an integer  $z \geq k$  such that  $m(z) \leq w \leq m(z + 1)$ .

- (a) When a job arrives, route it to the shared pool if  $q \geq z$  and add it to the queue if  $q \leq z - 1$ .
- (b) When a capacity unit arrives, serve a job from the queue if  $q \geq 1$ . Otherwise, if  $q = 0$ , offer help to the shared pool with probability  $\lambda^{z+1}/\phi \in [0, 1]$  (note that  $\phi \geq \lambda^{k+1} \geq \lambda^{z+1}$ ) and be idle with probability  $1 - \lambda^{z+1}/\phi$ .

Moreover, the above policy is the unique optimal policy when  $w < m(k)$  or  $w \in (m(z), m(z + 1))$  for any  $z \geq k$ . When  $w = m(z)$  for some integer  $z \geq k$ , the set of optimal policies is the mixing of the two (unique) optimal policies when  $w = m(z) - \delta$  and  $w = m(z) + \delta$  with small  $\delta$  (e.g., any  $0 < \delta < 1$ ).

According to Lemma 4.2, a server's best response is a threshold policy with respect to its queue length for any token-earning probability  $\phi$  and (endogenous) shared pool waiting time  $w$ . Particularly, a server requests help only when its queue length exceeds a certain threshold (value  $k$  in case one and value  $z \geq k$  in case two) and offers help only when its queue is empty. The threshold parameter  $k = \lfloor \frac{\ln \phi}{\ln \lambda} \rfloor$  in Lemma 4.2 depends only on the value of  $\ln \phi$  relative of  $\ln \lambda$ . When  $\phi$  is small, it is costly to earn tokens, so a server is conservative in spending tokens; consequently, the threshold parameter  $k$  is decreasing in  $\phi$ . On the other hand, when  $w$  increases, the benefit of requesting help decreases. Notably, when  $w$  surpasses  $m(k)$ , the threshold value  $z$  is larger than  $k$  and increases in  $w$ . Finally, we remark that the optimal policy is independent of the specific value of the job processing cost  $c$ . This is probably intuitive because an optimal policy serves all jobs (rather than retaining certain jobs indefinitely) regardless of the value of  $c$ .

We prove Lemma 4.2 in Appendix B. The proof relies on an analysis of the dual of problem (2) where we dualize the flow balance constraint of the tokens. We show strong duality holds and

demonstrate that the proposed policy is optimal to the dual problem with a specific dual variable; this implies that the policy is optimal to (2) and the identified dual variable is the optimal dual variable.

Lemma 4.2 shows that a server's best response remains the same as the shared pool waiting time  $w$  varies within the interval  $w \in [0, m(k)]$  or  $w \in [m(z), m(z+1)]$  for any integer  $z \geq k$ . This invariance is particularly true for small values of  $w$  (e.g.,  $w \leq m(1) = 1$ ); we describe a server's best response in this case in Corollary 4.3, which follows directly from Lemma 4.2.

**Corollary 4.3** (Optimal Policy when  $w \leq 1$ ). *Suppose that the shared pool waiting time  $w$  is no larger than one. An optimal policy of (2) is as follows. Moreover, it is the unique optimal policy for  $w < 1$ .*

1. When  $\phi < \lambda$  (Case One): let  $k \in \mathbb{N}$  be an integer that satisfies  $\phi \in [\lambda^{k+1}, \lambda^k]$ .<sup>9</sup>
  - (a) When a job arrives, route it to the shared pool if  $q \geq k$  and add it to the queue if  $q \leq k - 2$ . Otherwise, if  $q = k - 1$ , route the job to the shared pool with probability  $p \triangleq \frac{\phi - \lambda^{k+1}}{\lambda^k - \lambda^{k+1}} \in [0, 1]$  and add it to the queue with probability  $1 - p = \frac{\lambda^k - \phi}{\lambda^k - \lambda^{k+1}}$ .
  - (b) When a capacity unit arrives, serve a job from the queue if  $q \geq 1$  and offer help to the shared pool if  $q = 0$ .
2. When  $\phi \geq \lambda$  (Case Two):
  - (a) When a job arrives, route it to the shared pool.
  - (b) When a capacity unit arrives, serve a job from the queue if  $q \geq 1$ . Otherwise, if  $q = 0$ , offer help to the shared pool with probability  $\lambda/\phi \in [0, 1]$  and be idle with probability  $1 - \lambda/\phi$ .

According to Corollary 4.3, when the shared pool waiting time  $w$  is small (e.g.,  $w \leq 1$ ) and the mechanism set  $\phi$  equal to  $\lambda$ , it is each server's best response to implement complete resource pooling: i.e., (i) serving a job from its queue if one is present and helping others otherwise when a capacity unit arrives, and (ii) requesting help for all incoming jobs. Therefore, the system's dynamics are identical to those under centralized control. In Section 4.2, we show that the endogenous waiting time  $w$  in the shared pool diminishes to zero when the number of servers  $N$  increases and servers cooperate to some extent. Thus, complete resource pooling is an FMFE when there are many servers and the token-earning probability  $\phi$  equals  $\lambda$ .

---

<sup>9</sup>When  $\phi = \lambda^{z+1}$  for some integer  $z \geq 1$ , letting  $k$  be either  $z$  or  $z + 1$  yields the same policy.

## 4.2 Characterization of FMFE

In this section, we characterize FMFE using the best-response policies of servers derived in Section 4.1. First, no cooperation is an FMFE. However, as we show, under mild conditions, as long as servers cooperate, the waiting time  $w$  in the shared pool diminishes to zero as the number of servers  $N$  grows. Therefore, the waiting time  $w$  is either infinity (indicating no cooperation) or nearly zero when  $N$  is sufficiently large.

### 4.2.1 Waiting Time in Shared Pool Diminishing when Servers Cooperate

Suppose that there exists some fixed constant  $\bar{w} < \infty$  such that all servers believe that the shared pool waiting time satisfies  $w \leq \bar{w}$ . This effectively rules out the case of  $w = \infty$ , which indicates a complete absence of cooperation, and ensures servers cooperate to some extent. In Proposition 4.4 we show that under this condition, the average waiting time in the shared pool converges to zero at a rate of  $O(\frac{1}{N})$ .

**Proposition 4.4.** *Suppose that there exists a fixed constant  $\bar{w} < \infty$  such that all servers believe that the waiting time  $w$  in the shared pool is no larger than  $\bar{w}$ . Then,*

1. *The expected shared pool queue length in the stationary distribution remains uniformly bounded from above for any number of servers  $N$ , and*
2. *The waiting time  $w$  satisfies  $w \leq \frac{M_1}{N}$  for all  $N$  and some absolute constant  $M_1$  that depends only on the values of the job arrival rate  $\lambda$ , token-earning probability  $\phi$ , and upper bound  $\bar{w}$ .*

We prove Proposition 4.4 in Appendix A.3. Intuitively, as the number of servers  $N$  increases, both the job arrival rate at the shared pool (which equals the collective requesting-help rates across servers) and the job processing rate at the shared pool (which equals the collective offering-help rates across servers) grow to infinity, because all servers presume that shared pool waiting time is at most  $\bar{w}$ . On the other hand, the job arrival rate at the shared pool is always a fraction  $\phi < 1$  of the job processing rate there (see Remark 3.1). Consequently, the expected waiting time in the shared pool diminishes to zero when the number of servers  $N$  increases. We use a drift analysis to show this formally.

### 4.2.2 Existence of FMFE

According to Proposition 4.4, if all servers presume the shared pool waiting time  $w$  to be less than one and follow the best-response policy in Corollary 4.3, the induced waiting time is indeed below

one when the number of servers  $N$  is large. Therefore, all servers following the best-response policy in Corollary 4.3 forms an FMFE. We formally state this in Proposition 4.5.

**Proposition 4.5.** *Suppose that all servers follow the best-response strategy in Corollary 4.3. This constitutes an FMFE when the number of servers  $N$  is large.*

The minimum number of servers  $N$  to maintain such an FMFE in Proposition 4.5 can be specified either analytically or numerically. From Corollary 4.3, this necessitates the induced shared pool waiting time  $w$  below one when all servers follow the FMFE strategy. For instance, consider a scenario where the token-earning probability  $\phi$  equals the job arrival rate  $\lambda$ . In this case, the FMFE strategy is to implement complete resource pooling by the discussion following Corollary 4.3. Notably, when all servers follow the FMFE complete-resource-pooling strategy, the shared pool evolves as an  $M/M/1$  queue with job arrival and processing rates of  $N\lambda$  and  $N$ , respectively. Consequently, the expected number of jobs in the shared pool equals  $\frac{\lambda}{1-\lambda}$  in the stationary distribution. Applying Little’s Law, the expected waiting time in the shared pool is  $w = \frac{\lambda}{1-\lambda} \frac{1}{N\lambda} = \frac{1}{N(1-\lambda)}$ . Therefore, to ensure that  $w \leq 1$  and hence complete resource pooling is an FMFE when  $\phi = \lambda$ , the number of servers should be at least  $\lceil \frac{1}{1-\lambda} \rceil$ . Specifically, this implies a minimum of three servers when  $\lambda = 0.6$ , five servers when  $\lambda = 0.8$ , and ten servers when  $\lambda = 0.9$ . In Section 6, we demonstrate that all servers following the FMFE strategy is an approximate equilibrium in the original problem for large markets for any value of  $\phi$ , and we show via numerical results that complete resource pooling is an approximate equilibrium when  $\phi = \lambda$  even for small markets with only a few servers (e.g., with ten servers).

## 5 Optimal Value of $\phi$

In this section, we investigate the optimal value of  $\phi$ , which governs the token acquisition rate when a server “mines” tokens by offering help. Intuitively, if  $\phi$  is set too low, it is challenging to earn tokens, causing servers to be conservative in requesting help. This impedes cooperation and leads to longer queues at the servers. Conversely, if  $\phi$  is set too high, the token acquisition becomes too effortless. In this case, servers may lack the incentive to offer help, even when it has no pending job. This diminishes the overall system efficiency.

We examine the optimal value of  $\phi$  in the regime with many servers. According to Proposition 4.4, for any value of  $\phi \in (0, 1)$ , the shared pool waiting time  $w$  decreases to zero as the number of servers grows, assuming some level of server cooperation. We thus focus on the limit where  $w$

approaches zero (or nearly so) for any value of  $\phi \in (0, 1)$ . In this case, servers' optimal strategies are detailed in Corollary 4.3. Additionally, it forms an FMFE when all servers follow such a strategy and the number of servers is large by Proposition 4.5.

According to Corollary 4.3, there is a phase transition in the dynamics of servers as the token-earning probability  $\phi$  varies. To see this, let  $R(\phi)$  and  $P(\phi)$  denote the steady-state rates of requesting help and offering help, respectively, and  $q(\phi)$  the expected queue length of a server in the stationary distribution, all in the fluid mean-field problem and under the optimal strategy in Corollary 4.3. These two rates and the expected queue length  $q(\phi)$  of a server are depicted in Figure 1.

We first consider the case that  $\phi \leq \lambda$ . From Corollary 4.3 and Appendix B.2.1 we have

$$P(\phi) = \frac{1 - \lambda}{1 - \phi},$$

$$R(\phi) = \phi \cdot P(\phi) = \phi \cdot \frac{1 - \lambda}{1 - \phi},$$

and the expected queue length  $q(\phi)$  of a server is expressed in (11) in the Appendix. When  $\phi \leq \lambda$ , both the requesting and offering help rates increase with  $\phi$ , while the expected queue length of a server decreases with  $\phi$ . Intuitively, as the token-earning probability  $\phi$  increases, it becomes easier to earn tokens. Hence, a server is more willing to request help by spending tokens, which leads to a shorter queue. Meanwhile, a server offers help more to earn more tokens because the expected rates of earning and spending tokens need to be equal. When the token-earning probability  $\phi$  is strictly less than  $\lambda$ , each server has  $q(\phi) > 0$  jobs in expectation. Since the expected queue length of the shared pool is bounded by a constant that is independent of the number of servers by Proposition 4.4, the expected job total in the system scales linearly with the number of servers  $N$ , and is approximately  $q(\phi) \cdot N$  when the number of servers  $N$  is large.

Nevertheless, when the token-earning probability  $\phi$  equals the job arrival rate  $\lambda$ , by the discussion following Corollary 4.3, it is every server's best strategy to request help for all incoming jobs and, upon the arrival of a capacity unit, offer help to the shared pool as long as its queue is empty. Consequently, the rate of requesting help equals the job arrival rate  $\lambda$ , and the rate of offering help equals the arrival rate of capacity units, which is one. Additionally, this results in an empty queue at each server, i.e.,  $q(\lambda) = 0$ . In this case, all of the jobs are in the shared pool. The dynamics of the shared pool is an  $M/M/1$  queue with job arrival and processing rates of  $N\lambda$  and  $N$ , respectively. Thus, the total number of jobs in the system is  $\frac{\lambda}{1-\lambda}$  in expectation. In this case, complete resource pooling is achieved, and the system's dynamics under the mechanism are

identical to the dynamics under centralized control.

On the other hand, when  $\phi \geq \lambda$ , the steady-state rates of requesting and offering help are

$$P(\phi) = \frac{\lambda}{\phi},$$

$$R(\phi) = \phi \cdot P(\phi) = \lambda,$$

and a server's queue remains empty by Corollary 4.3. In this case, it is fairly easy to earn tokens, allowing a server to earn sufficient tokens to request help for all incoming jobs. However, the server earns just enough tokens for incoming jobs, by dedicating a fraction  $\lambda/\phi$  of capacity units to serve the shared pool, while being idle for the rest of the time. Therefore, as the token-earning probability  $\phi$  increases, the rate of requesting help remains constant (which equals the job arrival rate  $\lambda$ ) whereas the rate of offering help is  $\lambda/\phi$ , which decreases with  $\phi$ . Thus, although for any  $\phi \geq \lambda$  the waiting time in the shared pool asymptotically diminishes to zero as the number of servers increases (Proposition 4.4), the shared pool's queue length and the waiting time in the shared pool increase with the value of  $\phi$  for any problem instance. Specifically, when  $\phi \geq \lambda$ , all of the jobs are in the shared pool. Additionally, the dynamics of the shared pool is an  $M/M/1$  queue with job arrival and processing rates of  $N\lambda$  and  $N\lambda/\phi$ , respectively. Therefore, the total number of jobs in the system is  $\frac{\phi}{1-\phi}$  in expectation, which increases with  $\phi$ .

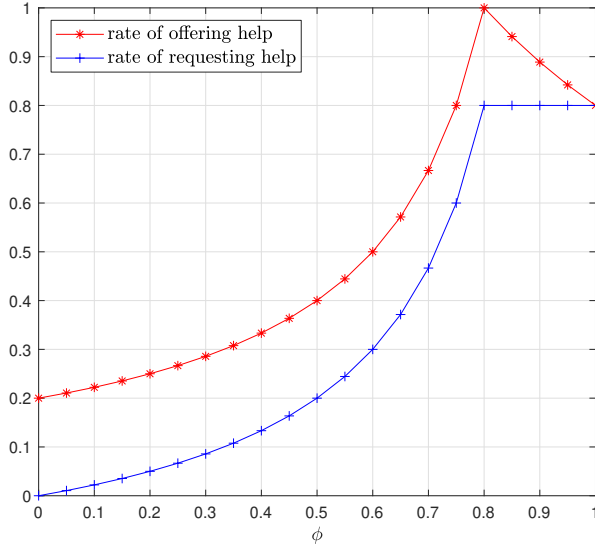
Theorem 5.1 characterizes the level of system congestion (i.e., the job total in the system) as a function of  $\phi$  based on the above discussion.

**Theorem 5.1.** *Suppose all servers follow the FMFE strategy in Corollary 4.3 in the fluid mean-field problem. This forms an FMFE when the number of servers  $N$  is large by Proposition 4.5. Let  $Q_\Sigma(\phi)$  denote the expected number of jobs in the system in the stationary distribution when the token-earning probability equals  $\phi$ . We have*

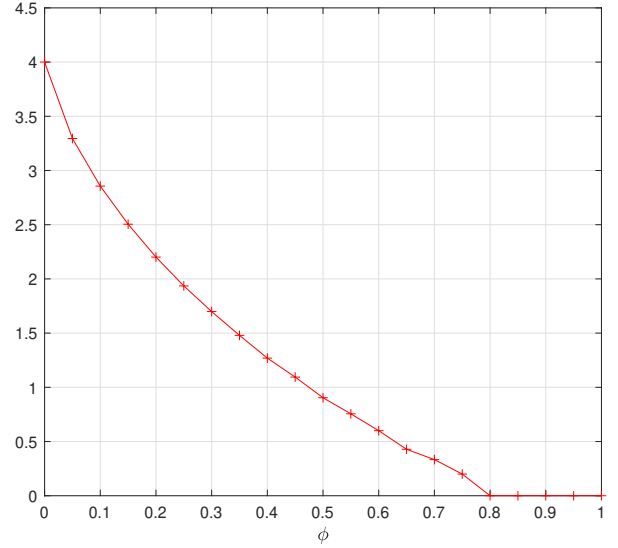
1.  $\lim_{N \rightarrow \infty} Q_\Sigma(\phi)/N = q(\phi)$  when  $\phi < \lambda$ , where  $q(\phi)$  denotes the expected queue length of a server in the stationary distribution. Notably,  $q(\phi)$  is decreasing in  $\phi$ ; the expression of  $q(\phi)$  is visualized in Figure 1b and provided in (11) in the Appendix.
2.  $Q_\Sigma(\phi) = \frac{\phi}{1-\phi}$  when  $\phi \geq \lambda$ .

According to Theorem 5.1, the optimal value of the token-earning probability  $\phi$  equals the job arrival rate  $\lambda$ , as we state formally in Theorem 5.2.

**Theorem 5.2.** *The optimal value of the token-earning probability  $\phi$  equals the job arrival rate  $\lambda$  when the number of servers  $N$  is large. In this case, complete resource pooling is an FMFE,*



(a) Steady-state rates of requesting and offering help



(b) Expected queue length of a server in the stationary distribution

**Figure 1:** Steady-state rates of requesting and offering help (Figure 1a) and expected queue length of a server in the stationary distribution (Figure 1b) in the fluid mean-field problem, when job arrival rate  $\lambda = 0.8$  and the server follows the FMFE policy in Corollary 4.3. The offering help rate is  $\frac{1-\lambda}{1-\phi}$  for  $\phi \in [0, \lambda]$  and  $\frac{\lambda}{\phi}$  for  $\phi \in [\lambda, 1]$ ; the requesting help rate is  $\phi$  times the offering help rate, i.e., is  $\phi \cdot \frac{1-\lambda}{1-\phi}$  for  $\phi \in [0, \lambda]$  and a constant  $\lambda$  for  $\phi \in [\lambda, 1]$ .

and the system's dynamics and performance under the token-based mechanism match those under centralized control.

## 6 FMFE Strategy as a Near-Optimal Best Response

In this section, we justify the use of the FMFE concept introduced in Section 3 for analyzing the mechanism. First, we rigorously show that playing an FMFE strategy when all other servers follow the FMFE strategy is a near-optimal best response in large markets with many servers, for any value of  $\phi \in (0, 1)$  (Section 6.2). Then, we illustrate via numerical results that when the token-earning probability  $\phi$  equals its optimal value  $\lambda$ , complete resource pooling is an approximate equilibrium even for small markets with only a few servers (Section 6.3).

### 6.1 FMFE Strategy in the Original Problem

The FMFE strategy in Corollary 4.3 can be implemented in the original problem (where the number of tokens  $S_i(t)$  a server  $i$  possesses at time  $t$  needs to satisfy  $0 \leq S_i(t) \leq C$ ) as well, with the only difference that a server can request help only when it has a positive number of tokens and offer help



when its token count is below the upper bound value of  $C$ . As a result, the token count is always within the range of zero to  $C$ . We denote by  $\pi^F$  the FMFE strategy described in Corollary 4.3 and  $\bar{\pi}^F$  the modified version of  $\pi^F$  that ensures feasibility in the original problem.

Note that both the policies  $\pi^F$  and  $\bar{\pi}^F$  are feasible to the fluid mean-field problem  $V^F(w)$  defined in (2). Moreover, the policy  $\pi^F$  is optimal to  $V^F(w)$  when the shared pool waiting time  $w \leq 1$  by Corollary 4.3. The two policies  $\pi^F$  and  $\bar{\pi}^F$  take different actions only when the token count is either zero or the upper bound value  $C$  in the original problem. In Lemma 6.1 we show the probability that such an event happens under policy  $\bar{\pi}^F$  decays to zero at a rate of  $(\frac{1}{C})$ . Thus, the dynamics of policy  $\bar{\pi}^F$  in the original problem converges to the dynamics of policy  $\pi^F$  in the fluid problem (where the constraint  $0 \leq S_i(t) \leq C$  is dropped) as the upper bound value  $C$  increases. Therefore, the fluid relaxation in the definition of FMFE is essentially tight for a large value of  $C$ .

**Lemma 6.1.** *Let  $\bar{S}_i(\infty)$  denote the number of tokens of server  $i$  in the stationary distribution of policy  $\bar{\pi}^F$ . We have*

$$\mathbb{P}[\bar{S}_i(\infty) = 0] + \mathbb{P}[\bar{S}_i(\infty) = C] \leq \frac{M_2}{C}$$

for some constant  $M_2$  that depends only on the values of  $\lambda$  and  $\phi$ .

We prove Lemma 6.1 in Appendix C.1. Since the dynamics of the FMFE strategy in the original problem converge to that in the fluid problem as the upper bound value  $C$  increases, the expected queue length and rates of requesting and offering help also converge, as we demonstrate in Lemma 6.2. We prove Lemma 6.2 in Appendix C.2.

**Lemma 6.2.** *Consider a representative server  $i$  that follows the FMFE strategy in both the original and fluid problems. Let  $\bar{Q}_i(\infty)$  denote the queue length of server  $i$  in the stationary distribution and  $\mathbb{P}[\bar{X}_i(\infty) = 1]$  and  $\mathbb{P}[\bar{Y}_i(\infty) = 1]$  the stationary rates of requesting and offering help, under policy  $\bar{\pi}^F$ . Analogously, let  $Q_i(\infty)$  denote the queue length of server  $i$  in the stationary distribution and  $\mathbb{P}[X_i(\infty) = 1]$  and  $\mathbb{P}[Y_i(\infty) = 1]$  the stationary rates of requesting and offering help, under policy  $\pi^F$ . We have*

1.  $\mathbb{E}[Q_i(\infty)] \leq \mathbb{E}[\bar{Q}_i(\infty)] \leq \mathbb{E}[Q_i(\infty)] + \frac{M_3}{C},$
2.  $\mathbb{P}[\bar{X}_i(\infty) = 1] \leq \mathbb{P}[X_i(\infty) = 1] \leq \mathbb{P}[\bar{X}_i(\infty) = 1] + \frac{M_4}{C},$
3.  $\mathbb{P}[\bar{Y}_i(\infty) = 1] \leq \mathbb{P}[Y_i(\infty) = 1] \leq \mathbb{P}[\bar{Y}_i(\infty) = 1] + \frac{M_5}{C},$

where  $M_3$ ,  $M_4$  and  $M_5$  are constants that depend only on the values of  $\lambda$  and  $\phi$ .

Finally, analogous to Proposition 4.4, if all servers follow the FMFE policy  $\bar{\pi}^F$  in the original problem, the long-run average waiting time in the shared pool diminishes to zero at a rate of  $O(\frac{1}{N})$  as the number of servers  $N$  increases. This is formalized in Proposition 6.3. We prove Proposition 6.3 in Appendix C.3.

**Proposition 6.3.** *Suppose all servers follow the FMFE policy  $\bar{\pi}^F$  in the original problem. Then,*

1. *The expected shared pool queue length in the stationary distribution remains uniformly bounded from above for any number of servers  $N$  and any upper bound value  $C$  for the token amount. In other words,  $\mathbb{E}[\bar{Q}_i(\infty)] \leq M_6$  for some absolute constant  $M_6$  that depends only on the values of the job arrival rate  $\lambda$  and token-earning probability  $\phi$  and is independent of the number of servers  $N$  or the upper bound value  $C$ .*
2. *The long-run average waiting time of jobs in the shared pool, denoted by  $w$ , satisfies  $w \leq \frac{M_7}{N}$  for some absolute constant  $M_7$ , which depends only on the values of the job arrival rate  $\lambda$  and token-earning probability  $\phi$  and is independent of the upper bound value  $C$  for the token amount.*

## 6.2 Asymptotic Analysis for Large Markets

In this section, we demonstrate that given any token-earning probability  $\phi \in (0, 1)$ , playing an FMFE strategy when all other servers follow the FMFE strategy is a near-optimal best response when the number of servers is large.

Specifically, assume that servers two to  $N$  follow the FMFE strategy  $\bar{\pi}^F$  in the original problem. Let  $Q^F$  denote the queue length of a server in the stationary distribution of the fluid mean-field problem when it follows the FMFE strategy  $\pi^F$ . We remark that  $c\lambda + \mathbb{E}[Q^F]$  is the optimal time-average total cost in the fluid mean-field problem when the shared pool waiting time  $w = 0$ , i.e.,  $V^F(0) = c\lambda + \mathbb{E}[Q^F]$ . We first show in Lemma 6.4 that, if server one in the original problem also follows the FMFE strategy  $\bar{\pi}^F$ , its time-average total cost surpasses  $c\lambda + \mathbb{E}[Q^F]$  by only  $O(\frac{1}{N} + \frac{1}{C})$ .

**Lemma 6.4.** *Suppose that all servers follow the FMFE strategy  $\bar{\pi}^F$  in the original problem. Then, the time-average total cost of server one is at most  $c\lambda + \mathbb{E}[Q^F] + \frac{M_3}{C} + \frac{\lambda M_7}{N}$ , where  $M_3$  and  $M_7$  are constants specified in Lemma 6.2 and Proposition 6.3 whose values depend only on  $\lambda$  and  $\phi$ .*

We prove Lemma 6.4 in Appendix C.4. Suppose server one also follows the FMFE policy  $\bar{\pi}^F$ . Intuitively, when there are many servers, the waiting time in the shared pool is nearly zero and on the order of  $O(\frac{1}{N})$  (Proposition 6.3). Additionally, when the token-amount upper bound  $C$  is large,

the expected queue length under policy  $\bar{\pi}^F$  is close to that under policy  $\pi^F$  in the fluid mean-field problem and differs by at most  $O\left(\frac{1}{C}\right)$  (Lemma 6.2). Therefore, the time-average total cost of server one deviates from the optimal value of the fluid mean-field problem, which is  $c\lambda + \mathbb{E}[Q^F]$ , by an amount of  $O\left(\frac{1}{N} + \frac{1}{C}\right)$ .

Next, we demonstrate in Lemma 6.5 that no matter what strategy server one uses, its time-average total cost is at least  $c\lambda + \mathbb{E}[Q^F] - O\left(\frac{1}{N^{1-\delta}}\right)$  for any  $\delta > 0$ .

**Lemma 6.5.** *Suppose that servers  $i \in [2 : N]$  follow the FMFE strategy  $\bar{\pi}^F$  in the original problem. The time-average total cost of server one is at least  $c\lambda + \mathbb{E}[Q^F] - \frac{M_8(\lambda, \phi, \delta)}{N^{1-\delta}}$  for any  $\delta > 0$ , regardless of the strategy server one uses, where  $M_8$  is a constant that depends only on the values of  $\lambda$ ,  $\phi$ , and  $\delta$ .*

We prove Lemma 6.5 in Appendix C.5. The proof follows two key steps:

1. We establish a relaxation to the problem of server one, where we endow server one with an additional power to empty the shared pool at the end of every interaction (i.e., requesting or offering help) with the shared pool. We show that in the relaxation, it is without loss of optimality to request help only when a job arrives, thus building a connection to the fluid mean-field problem.
2. Using a coupling argument along with a drift analysis, we show that the queue length of the shared pool transitions to the stationary distribution quickly as the number of servers grows. Moreover, in the stationary distribution, the shared pool is non-empty with a probability that differs from  $\phi$  by at most  $O\left(\frac{1}{N^{1-\delta}}\right)$ . This implies that the optimal time-average cost in the relaxation is no smaller than the optimal value of the fluid mean-field problem (which equals  $c\lambda + \mathbb{E}[Q^F]$ ) less than a small term that is  $O\left(\frac{1}{N^{1-\delta}}\right)$ .

Intuitively, as the number of servers increases, the queue length of the shared pool transitions to the steady state quickly. Therefore, it becomes more difficult to be strategic to the dynamics of the shared pool. Specifically, from Lemmas 6.4 and 6.5, if we set the token-amount upper bound to be  $C = \Omega(N)$ , the benefit from deviating from the FMFE strategy is only a negligible term that is nearly  $O\left(\frac{1}{N}\right)$ . This is the case even though server one is endowed with an unrealistic power to empty the shared pool at the end of every interaction with it.

### 6.3 Numerical Analysis for Small Markets

In Section 6.2, we rigorously justify that playing an FMFE strategy when all other servers play the FMFE strategy is a near-optimal best response when the number of servers is large. In this section, we demonstrate via numerical results that complete resource pooling is an approximate equilibrium under our mechanism even for small markets with only a few servers (e.g., with ten servers).

Recall that the FMFE concept involves two approximations: a fluid relaxation that allows the number of tokens a server possesses to be negative (and go beyond the upper bound value  $C$ ) and only requires that the expected rates of earning and spending tokens are equal in the long-run average; and a mean-field approximation as a large market approximation, motivated by the fact that the shared pool's queue length becomes roughly constant over time in the presence of many servers due to the averaging effect. The first approximation is tight when a server can possess many tokens (see Section 6.1). For this reason, we isolate the impact of the mean-field approximation for small markets and analyze a simplified fluid problem numerically.

**The Fluid Problem** We investigate the token-based mechanism when the token-earning probability  $\phi$  equals  $\lambda$  (which is optimal by Theorem 5.2). Suppose servers two to  $N$  implement complete resource pooling (which is the FMFE strategy when  $\phi = \lambda$  by Corollary 4.3), i.e., each server requests help for all incoming jobs (disregarding the nonnegativity constraint of tokens) and offers help to the shared pool upon the arrival of each capacity unit. We examine the best response of server one to the strategies of the other  $N - 1$  servers in a fluid problem, where the token count of server one can go negative and go beyond the upper bound value  $C$ , but the expected rates of earning and spending tokens need to be equal in the long-run average. In the fluid problem, server one equivalently interacts with an  $M/M/1$  queue (representing the shared pool) that has a job arrival rate of  $(N - 1)\lambda$  and an arrival rate of capacity units of  $(N - 1)$ .

**Server One's Problem** In a small market, a strategic server would like to continually infer the shared pool's queue length to request help only when the shared pool experiences low congestion. Additionally, the server is more willing to offer help when she is aware that the shared pool is empty, as this presents an opportunity to earn a token without incurring the cost of processing a job from the shared pool.<sup>10</sup>

However, the optimal strategy of server one is challenging to solve. Particularly, the server

---

<sup>10</sup>However, as we show in Section 6.2, as the number of servers increases, the shared pool's queue length converges to the steady state quickly. Therefore, it becomes more difficult to be strategic to the dynamics of the shared pool, and as a result, the benefit from deviating from the FMFE strategy becomes negligible.

can only make inferences about the shared pool based on partial information obtained from its interactions with the shared pool.<sup>11</sup> Consequently, server one’s optimal strategy is contingent upon the entire history of observations. To address this, we relax the problem of server one by providing the server additional advantages to obtain a tractable upper bound on the benefit of playing strategically. First, we enable server one to have complete information about the shared pool so that the server can make decisions based on its queue length. Second, we introduce a lower bound on the waiting time in the shared pool. Specifically, when server one requests help for a job, we let the job’s waiting time in the shared pool be  $\frac{q_0+1}{N}$ , where  $q_0$  denotes the shared pool’s queue length at the current time. Notably, since the rate at which capacity units join the shared pool is at most  $N$  and there are  $q_0$  jobs ahead of the job awaiting processing, this provides a lower bound on the expected waiting time in the shared pool.

After the two relaxations, server one’s optimal strategy depends only on two state variables—server one’s queue length and the queue length of the shared pool; this leads to a tractable optimization problem, and we formulate the problem in Appendix D. We remark that the optimal strategy can be quite different from the complete-resource-pooling FMFE strategy. For example, server one may relocate a job to the shared pool when another server has served a job from the pool, thus reducing the queue length of the shared pool. Additionally, when a job arrives, server one may instead add the job to its queue if the shared pool has high congestion at that moment.

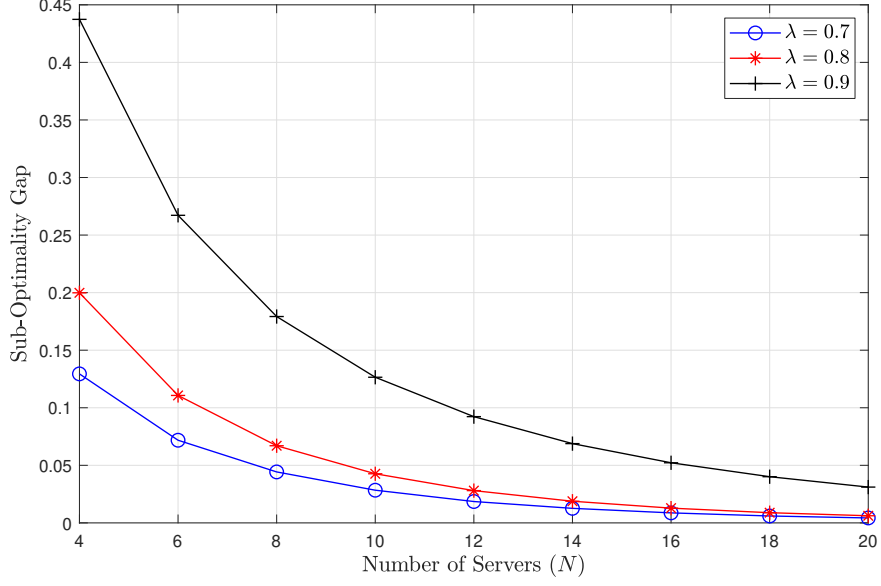
**Numerical Results** We compare the time-average total cost of server one in the fluid problem under the FMFE strategy (i.e., employing complete resource pooling) with that of the best response when the server has complete information about the shared pool. We show that the sub-optimality gap is small, even with only a few servers.

Specifically, we consider problem instances with job arrival rate  $\lambda \in \{0.7, 0.8, 0.9\}$ , number of servers  $N$  increasing linearly from 4 to 20 with step size 2, and job processing cost  $c = 1$ . For fixed values of  $\lambda$  and  $N$ , we evaluate (a) the time-average total cost  $V^{\text{CRP}}$  when server one follows complete resource pooling, which is  $\lambda \left( c + \frac{1}{N(1-\lambda)} \right)$ , where  $\frac{1}{N(1-\lambda)} = \frac{\lambda}{1-\lambda} \cdot \frac{1}{N\lambda}$  is the expected waiting time of a job in the shared pool by Little’s law, and (b) the time-average total cost  $V^{\text{OPT}}$  of an optimal strategy that has complete information about the shared pool, which can be obtained from solving a two-dimensional dynamic program in Appendix D. We evaluate the relative sub-optimality gap  $\frac{V^{\text{CRP}} - V^{\text{OPT}}}{V^{\text{OPT}}}$ , and plot the gap as a function of the number of servers in Figure 2.

From Figure 2, as the number of servers increases, the sub-optimality of playing complete

---

<sup>11</sup>Please refer to the paragraph of “The Shared Pool” in Section 2.2.1.



**Figure 2:** Complete resource pooling versus best response.

resource pooling decreases fast. In particular, the sub-optimality gap is smaller than 5% when there are more than 8 servers when  $\lambda = 0.7$  (4.43%), 10 servers when  $\lambda = 0.8$  (4.27%), and 18 servers when  $\lambda = 0.9$  (4.01%). The sub-optimality gap further drops below 1% when there are 16 servers when  $\lambda = 0.7$  (0.87%) and 18 servers when  $\lambda = 0.8$  (0.89%). We highlight here that the sub-optimality gap we evaluate in these examples is conservative in that the benchmark policy has an unrealistic informational edge because it can perfectly monitor the congestion level of the shared pool. However, within our mechanism, servers can only infer partial information about the shared pool based on the outcome of offering help and the status of relocated jobs. Hence, the ability to be strategic to the dynamics of the shared pool will be even more limited in the original problem.

In all, our numerical results above indicate that the value of continuously inferring the congestion level of the shared pool is small even in the presence of only a few servers. In other words, a given server has a limited ability to strategize and impact the market when all other servers adopt complete resource pooling.

## 7 Extension to Heterogeneous Servers

Our token-based mechanism and its performance analysis developed in the previous sections can be extended to a more general setting with heterogeneous servers (we still require that jobs are stochastically identical). We model the dynamics of each server as in Section 2. However, jobs and capacity units can arrive at different servers at different rates. Specifically, we let  $\lambda_i > 0$  denote the

job arrival rate and  $\mu_i > \lambda_i$  the arrival rate of capacity units at server  $i$ . These arrival processes are independent across servers. Additionally, we let  $\rho_i = \frac{\lambda_i}{\mu_i} < 1$  denote the utilization factor of server  $i$ . We assume the following regularity condition on the problem primitives in Assumption 7.1.

**Assumption 7.1.** For all servers  $i$ , the utilization factors are uniformly bounded from above and below by some positive constants  $\underline{\rho}, \bar{\rho} \in (0, 1)$ , i.e.,  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  for any  $i \in [N]$ . Moreover, the arrival rates are uniformly bounded from above and below by positive constants  $\underline{\lambda}, \bar{\lambda} > 0$ , i.e.,  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for any  $i \in [N]$ .

**The Centralized Setting** Since jobs are identical, in the centralized setting, the central planner would direct an idle server to process jobs from other queues to minimize the job total in the system. The dynamics of the number of jobs in the system is an  $M/M/1$  queue with a job arrival rate of  $\lambda_c \triangleq \sum_{i \in [N]} \lambda_i$  and a processing rate of  $\mu_c \triangleq \sum_{i \in [N]} \mu_i$ . As a result, the equivalent utilization factor of the system is  $\rho_c \triangleq \lambda_c / \mu_c \in [\underline{\rho}, \bar{\rho}]$ , and the expected total number of jobs in the stationary distribution is  $\frac{\rho_c}{1 - \rho_c}$ , which is between  $\frac{\underline{\rho}}{1 - \underline{\rho}}$  and  $\frac{\bar{\rho}}{1 - \bar{\rho}}$ .

**The Token-Based Mechanism** We consider the token-based mechanism defined in Section 2.2.1 with the token earning probability being  $\phi = \bar{\rho}$ , the largest utilization factor among servers.

We first characterize the FMFE. Suppose a server  $i$  presumes the waiting time in the shared pool to be small. Since  $\phi = \bar{\rho} \geq \rho_i$ , by case two of Corollary 4.3, the server's optimal strategy in the fluid mean-field problem is as follows: (i) requesting help for all incoming jobs; (ii) when a capacity unit arrives, serving a job from its queue if one is present; otherwise, if its queue is empty, offering help to the shared pool with probability  $\rho_i / \bar{\rho}$  and being idle with probability  $1 - \rho_i / \bar{\rho}$ .

Conversely, if all servers follow the strategy, the dynamics of the queue length of the shared pool is an  $M/M/1$  queue with a job arrival rate of  $\lambda_m \triangleq \sum_{i \in [N]} \lambda_i$  and a processing rate of  $\mu_m \triangleq \sum_{i \in [N]} \mu_i \cdot \rho_i / \bar{\rho} = \lambda_m / \bar{\rho}$ . Therefore, the equivalent utilization factor under the mechanism is  $\rho_m \triangleq \bar{\rho}$ . Consequently, the steady-state queue length of the shared pool is  $\frac{\bar{\rho}}{1 - \bar{\rho}}$ , and by Little's law, the steady-state waiting time in the shared pool diminishes to zero at a rate of  $O(\frac{1}{N})$  as the number of servers  $N$  increases. Therefore, if all servers follow the FMFE strategy  $\pi^F$ , it forms an FMFE when the number of servers is large.

**Comparison of System Congestion** In the above FMFE, the total number of jobs in the system is  $\frac{\bar{\rho}}{1 - \bar{\rho}}$  (because all servers have empty queues). This is at most a constant  $\frac{\bar{\rho}(1 - \underline{\rho})}{\underline{\rho}(1 - \bar{\rho})}$  times the total number of jobs in the centralized setting. Therefore, complete resource pooling is almost achieved.

In the special case where servers have the same utilization factor, complete resource pooling is attained precisely, even if servers may have varying job arrival and processing rates. We summarize these in Theorem 7.1, which extends Theorem 5.2 for the case of homogeneous servers.

**Theorem 7.1** (Extension of Theorem 5.2). *Suppose Assumption 7.1 holds, and consider the token-based mechanism with token-earning probability  $\phi = \bar{\rho}$ . If all servers follow the FMFE strategy  $\pi^F$ , this forms an FMFE when the number of servers is large. In this case, the expected number of jobs in the system is, at most, a constant factor of the number of jobs in the centralized setting. In addition, if servers have the same utilization factor, the mechanism attains the exact complete resource pooling.*

We next show that all servers following the FMFE strategy is an approximate equilibrium. We focus on the fluid problem described in Section 6.3 for simplicity. Recall that the fluid problem relaxes the constraint that the token count must be within the range of zero to the upper bound value of  $C$ . In the fluid problem, servers two to  $N$  follow the FMFE policy  $\pi^F$ ; in contrast, server one minimizes its time-average total cost subject to the flow balance constraint of the tokens (i.e., the expected rates of earning and spending tokens need to be equal). The fluid problem corresponds to the original problem where the token-amount upper bound  $C$  is infinity. In this case, the probability that servers two to  $N$  run out of tokens in the original problem is zero by Lemma 6.1; thus, the dynamics of servers two to  $N$  in the original and fluid problems are identical. As we show in Theorem 7.2, analogous to Lemmas 6.4 and 6.5, it is a near-optimal best response for server one to follow the policy  $\pi^F$  in the fluid problem. The proof of Theorem 7.2 parallels the proofs of Lemmas 6.4 and 6.5, and we provide it in Appendix C.6.

**Theorem 7.2** (Extensions of Lemmas 6.4 and 6.5). *Suppose Assumption 7.1 holds, and consider the token-based mechanism with token-earning probability  $\phi = \bar{\rho}$ . Moreover, suppose servers two to  $N$  follow the FMFE strategy  $\pi^F$ . We have the following results.*

1. (Extension of Lemma 6.4) *If server one also follows policy  $\pi^F$ , its time-average total cost is upper-bounded by  $c\lambda_1 + \frac{\bar{\lambda}\bar{\rho}}{\lambda(1-\bar{\rho}) \cdot N}$ .*
2. (Extension of Lemma 6.5) *Regardless of the strategy server one uses, its time-average total cost is lower-bounded by  $c\lambda_1 - \frac{M_9(\lambda, \bar{\rho}, \delta)}{N^{1-\delta}}$  for any  $\delta > 0$  and a constant  $M_9(\lambda, \bar{\rho}, \delta)$  that depends only on the values of  $\lambda$ ,  $\bar{\rho}$ , and  $\delta$ .*

Therefore, the benefit from deviating from the FMFE strategy  $\pi^F$  is at most  $\frac{\bar{\lambda}\bar{\rho}}{\lambda(1-\bar{\rho}) \cdot N} + \frac{M_9(\lambda, \bar{\rho}, \delta)}{N^{1-\delta}}$ , which is negligible when the number of servers increases.



**Comparison of Allocation of Job Processing Costs** The long-run average job processing cost, which equals  $c \cdot \sum_{i \in [N]} \lambda_i$ , is distributed among servers proportional to the flow of capacity units a server contributes to the system, both in the centralized setting and under our mechanism.

In the centralized setting, servers share the processing cost proportional to their job processing rates  $\mu_i$ . Therefore, a server with a higher value of  $\mu_i$  undertakes a higher processing cost, even though its demand for computing resources, which equals  $\lambda_i$ , is low. Thus, the central planner discriminates against high-processing-rate servers to minimize system congestion. As a result, a server  $i$  would like to pretend that its processing rate is  $\lambda_i/\bar{\rho}$  if it is possible.

In contrast, under our mechanism, every server  $i$  contributes its capacity units at a rate of  $\mu_i \cdot \rho_i/\bar{\rho} = \lambda_i/\bar{\rho}$ . Consequently, the job processing cost is allocated among servers proportional to  $\lambda_i$ , which aligns with servers' individual demand for computing resources. Therefore, the cost is allocated *fairly* among servers, which is a requisite given that servers cooperate in their self-interest in our mechanism.

## 8 Conclusions

We have considered the problem of incentivizing resource pooling among  $N$  strategic servers in a limited information setting, with applications in designing decentralized computing markets on blockchains, among others. Our main contribution is a simple token-based mechanism. Although the induced dynamic game is complex, we apply a notion of fluid mean-field equilibrium to simplify the analysis. We demonstrate that the mechanism incentivizes complete resource pooling when the number of servers is large. Moreover, we show via numerical results that complete resource pooling is an approximate equilibrium even for small markets with only a few servers, providing further practical support to our proposed mechanism. Finally, we extend our mechanism to a setting with heterogeneous servers and show that the system performance is close to that under centralized control.

The paper opens up many promising directions for future research. First, it is interesting to extend our mechanism further to a more general setting where both jobs and servers are heterogeneous, and the job processing time depends on the specific job-server pair. In this setting, even the centralized control benchmark can be challenging to solve as work-conserving policies are not necessarily optimal.<sup>12</sup> Nevertheless, we are optimistic that the mechanism we develop can shed light on mechanism design in this more general setting. Beyond that, we are hopeful that our mecha-

---

<sup>12</sup>This is because a planner may intentionally idle a server to await a better-fit job.

nism and analytical techniques can provide insights into the design and analysis of near-optimal mechanisms for various other decentralized applications.

## References

- Anily, S. and Haviv, M. (2010), ‘Cooperation in service systems’, *Operations Research* **58**(3), 660–673.
- Anily, S. and Haviv, M. (2014), ‘Subadditive and homogeneous of degree one games are totally balanced’, *Operations Research* **62**(4), 788–793.
- Arnosti, N., Johari, R. and Kanoria, Y. (2021), ‘Managing congestion in matching markets’, *Manufacturing & Service Operations Management* **23**(3), 620–636.
- Balseiro, S. R., Besbes, O. and Weintraub, G. Y. (2015), ‘Repeated auctions with budgets in ad exchanges: Approximations and design’, *Management Science* **61**(4), 864–884.
- Balseiro, S. R., Brown, D. B. and Chen, C. (2021), ‘Dynamic pricing of relocating resources in large networks’, *Management Science* **67**(7), 4075–4094.
- Bertsekas, D. (2017), *Dynamic programming and optimal control*, Vol. 1, 4 edn, Athena scientific Belmont, MA.
- Bertsekas, D. P., Nedic, A. and Ozdaglar, A. E. (2003), ‘Convex analysis and optimization’.
- Bo, Y., Dawande, M. and Janakiraman, G. (2018), ‘Analysis of scrip systems: On an open question in johnson et al.(2014)’, *Operations Research* **66**(3), 611–619.
- Elmachtoub, A. N., Wei, Y. and Zhou, Y. (2015), ‘Retailing with opaque products’, *Available at SSRN 2659211*.
- Eppen, G. D. (1979), ‘Note—effects of centralization on expected costs in a multi-location newsboy problem’, *Management science* **25**(5), 498–501.
- Hu, X. and Caldentey, R. (2023), ‘Trust and reciprocity in firms’ capacity sharing’, *Manufacturing & Service Operations Management* **25**(4), 1436–1450.
- Iyer, K., Johari, R. and Sundararajan, M. (2014), ‘Mean field equilibria of dynamic auctions with learning’, *Management Science* **60**(12), 2949–2970.
- Johnson, K., Simchi-Levi, D. and Sun, P. (2014), ‘Analyzing scrip systems’, *Operations Research* **62**(3), 524–534.
- Kanoria, Y. and Saban, D. (2021), ‘Facilitating the search for partners on matching platforms’, *Management Science* **67**(10), 5990–6029.
- Karsten, F., Slikker, M. and Van Houtum, G.-J. (2015), ‘Resource pooling and cost allocation among independent service providers’, *Operations Research* **63**(2), 476–488.
- Kash, I. A., Friedman, E. J. and Halpern, J. Y. (2007), Optimizing scrip systems: Efficiency, crashes, hoarders, and altruists, in ‘Proceedings of the 8th ACM conference on Electronic commerce’, pp. 305–315.
- Kash, I. A., Friedman, E. J. and Halpern, J. Y. (2015), ‘An equilibrium analysis of scrip systems’, *ACM Transactions on Economics and Computation (TEAC)* **3**(3), 1–32.

- Liu, X., Gong, K. and Ying, L. (2022), ‘Steady-state analysis of load balancing with coxian-2 distributed service times’, *Naval Research Logistics (NRL)* **69**(1), 57–75.
- Milgrom, P. and Shannon, C. (1994), ‘Monotone comparative statics’, *Econometrica: Journal of the Econometric Society* pp. 157–180.
- Sarver, T. (2022), ‘Microeconomic theory lecture notes’.
- Shi, C., Wei, Y. and Zhong, Y. (2019), ‘Process flexibility for multiperiod production systems’, *Operations Research* **67**(5), 1300–1320.
- Spencer, J., Sudan, M. and Xu, K. (2014), ‘Queuing with future information’, *The Annals of Applied Probability* **24**(5), 2091–2142.
- Topkis, D. M. (1998), *Supermodularity and complementarity*, Princeton university press.
- Tsitsiklis, J. N. and Xu, K. (2013), ‘On the power of (even a little) resource pooling’, *Stochastic Systems* **2**(1), 1–66.
- Tsitsiklis, J. N. and Xu, K. (2017), ‘Flexible queueing architectures’, *Operations Research* **65**(5), 1398–1413.
- Weintraub, G. Y., Benkard, C. L. and Van Roy, B. (2008), ‘Markov perfect industry dynamics with many firms’, *Econometrica* **76**(6), 1375–1411.
- Xu, J. and Hajek, B. (2013), ‘The supermarket game’, *Stochastic Systems* **3**(2), 405–441.
- Yang, L., Debo, L. G. and Gupta, V. (2019), ‘Search among queues under quality differentiation’, *Management Science* **65**(8), 3605–3623.

## A Proofs for Sections 3 and 4

### A.1 Proof of Lemma 3.1

When a server offers help in the mean-field problem, the server receives a token with a probability of  $\phi$ , while incurring a cost of  $c$  (for processing a job from the shared pool) also with probability  $\phi$ ; these two events are statistically independent. We can consider an equivalent setting where the server, when providing help, obtains a token and confronts the cost  $c$  simultaneously with probability  $\phi$ , while experiencing no cost and no token acquisition with probability  $1 - \phi$  (i.e., the cost and token acquisition are perfectly correlated). These two settings are equivalent because they share the same transition probabilities and expected payoffs per period. We now prove Lemma 3.1 in the equivalent setting.

First, we remark that a server requests help either (i) upon the arrival of a job, or (ii) after the server has taken an action when a capacity unit arrives. To see this, suppose that the server instead requests help at time  $t$ , even though neither a job nor a capacity unit arrives at that moment. Let  $t' < t$  be the time when the most recent job or capacity unit arrived. The server is better off requesting help at time  $t'$  instead of waiting till  $t$ , because waiting till  $t$  does not provide additional information about the shared pool, but incurs a higher holding cost.

Second, we show that a server will not request help when a capacity unit arrives (scenario (ii)). To see this, suppose a capacity unit arrives. First, if the server opts to be idle and waste the unit, there is no change in the queue length or the number of tokens the server owns. Therefore, the server would not request help (otherwise, it would do so before the capacity unit arrives). Second, if the server intends to serve a job from its queue, the queue becomes shorter, and the benefit of requesting help decreases. Since the server does not request help before the capacity unit arrives, it will not do so after the arrival. Lastly, suppose it is optimal to offer help to the shared pool. In this case, the server could expend a token to request help immediately if it obtains a token at a cost of  $c$ . However, it is suboptimal because the server could be better off serving a job from its queue right away (rather than offering help and then requesting help upon earning a token), so the job gets served immediately (rather than waiting to be served in the shared pool). Consequently, it is better to serve a job from the queue with probability  $\phi$  and be idle and waste the capacity unit with probability  $1 - \phi$ , instead of offering help; a contradiction to the optimality of offering help.

Therefore, a server requests help only when a job arrives (scenario (i)). In this case, the server would only request help for the incoming job. This is because if the server would like to request help for a second job, the server would ideally do so before the arrival of the new job; however, this does not hold true.

### A.2 Proof of Lemma 3.2

We consider a representative server. Let  $p(q, s)$  denote the long-run average joint probability of having  $q$  jobs and  $s$  tokens, and  $p(q) = \sum_{s \in \mathbb{Z}} p(q, s)$  the marginal probability of having  $q$  jobs. The objective function of (2) can be expressed as

$$\begin{aligned} & \sum_{q \in \mathbb{N}} \sum_{s \in \mathbb{Z}} p(q, s) \cdot \left\{ \frac{q}{1 + \lambda} + w \cdot \frac{\lambda}{1 + \lambda} \cdot \mathbb{P}(X = 1 | q, s) + c \cdot \frac{1}{1 + \lambda} \cdot [\mathbb{P}(Y = 0 | q, s) + \phi \cdot \mathbb{P}(Y = 1 | q, s)] \right\} \\ &= \sum_{q \in \mathbb{N}} p(q) \cdot \left\{ \frac{q}{1 + \lambda} + w \cdot \frac{\lambda}{1 + \lambda} \cdot \mathbb{P}(X = 1 | q) + c \cdot \frac{1}{1 + \lambda} \cdot [\mathbb{P}(Y = 0 | q) + \phi \cdot \mathbb{P}(Y = 1 | q)] \right\}, \end{aligned}$$

where  $\mathbb{P}(X = 1|q) = \sum_{s \in \mathbb{Z}} p(q, s) \cdot \mathbb{P}(X = 1|q, s)/p(q)$  denotes the marginal probability of taking the action  $X = 0$  when there are  $q$  jobs and a new job arrives, and  $\mathbb{P}(Y = 0|q)$  and  $\mathbb{P}(Y = 1|q)$  are defined analogously as the marginal probability of taking the action  $Y = 0$  or  $Y = 1$  when there are  $q$  jobs and a capacity unit arrives. Similarly, we can express the tokens' flow-balance constraint in (2) as

$$\frac{\lambda}{1+\lambda} \sum_{q \in \mathbb{N}} p(q) \cdot \mathbb{P}(X = 1|q) = \frac{\phi}{1+\lambda} \sum_{q \in \mathbb{N}} p(q) \cdot \mathbb{P}(Y = 1|q).$$

Since both the objective function and constraint of (2) are independent of the number of tokens  $s$ , the optimal policy of (2) is independent of  $s$  as well.

Finally, let  $\mathbb{P}(q, X) = \frac{\lambda}{1+\lambda} \cdot p(q) \cdot \mathbb{P}(X|q)$  and  $\mathbb{P}(q, Y) = \frac{1}{1+\lambda} \cdot p(q) \cdot \mathbb{P}(Y|q)$  denote the joint probabilities of having  $q$  jobs and taking the action of  $X$  or  $Y$ , respectively; we can rewrite (2) as a linear program (3).

$$\begin{aligned} V^F(w) = & \min_{p(q), \mathbb{P}(q, X), \mathbb{P}(q, Y) \geq 0} \sum_{q \in \mathbb{N}} \left\{ \frac{q}{1+\lambda} \cdot p(q) + w \cdot \mathbb{P}(q, X = 1) + c \cdot \left( \mathbb{P}(q, Y = 0) + \phi \cdot \mathbb{P}(q, Y = 1) \right) \right\} \\ \text{s.t.} \quad & \sum_{q \in \mathbb{N}} \mathbb{P}(q, X = 1) = \sum_{q \in \mathbb{N}} \phi \cdot \mathbb{P}(q, Y = 1), \\ & \mathbb{P}(q, X = 0) = \mathbb{P}(q+1, Y = 0), \forall q \in \mathbb{N}, \\ & \mathbb{P}(q, X = 0) + \mathbb{P}(q, X = 1) = \frac{\lambda}{1+\lambda} \cdot p(q), \forall q \in \mathbb{N}, \\ & \mathbb{P}(q, Y = 0) + \mathbb{P}(q, Y = 1) + \mathbb{P}(q, Y = \emptyset) = \frac{1}{1+\lambda} \cdot p(q), \forall q \in \mathbb{N}, \\ & \sum_{q \in \mathbb{N}} p(q) = 1. \end{aligned} \tag{3}$$

In (3), the first constraint represents the flow balance constraint of the tokens, i.e., expected rates of earning and spending tokens need to be equal in the long-run average. Once we fix a policy, the dynamics of the queue length  $q$  follow a birth-and-death chain process; hence, the stationary distribution is reversible, as indicated by the second constraint in (3). We can interpret (3) as computing an optimal stationary distribution of the queue-length-control pairs to minimize a server's time-average cost.

### A.3 Proof of Proposition 4.4

In the proof, we let  $Q_i(t)$  denote the queue length of server  $i$  for any  $i \in [N]$  and  $Q_0(t)$  the queue length of the shared pool in time  $t$ . Additionally, let the vector  $\mathbf{Q}(t) = (Q_i(t))_{i \in [0:N]}$  denote the concatenation of the queue lengths. The queue length vector  $\mathbf{Q}(t)$  is a Markov chain. Finally, let  $Q_0(\infty)$  denote the shared pool's queue length in the stationary distribution.

#### A.3.1 Step One: A Coarse Bound on the Tail Probability of $Q_0(\infty)$

We first provide a coarse bound on the tail probability of the shared pool queue length  $Q_0(\infty)$  in Lemma A.1.

**Lemma A.1.** *For any  $m \in \mathbb{N}$ , we have*

$$\mathbb{P}\left(Q_0(\infty) \geq mN\right) \leq \alpha_1 N \cdot \exp\left(-\alpha_2 \cdot m\right)$$

for some constants  $\alpha_1, \alpha_2 > 0$  that depend only on the values of  $\lambda$ ,  $\phi$ , and  $\bar{w}$ .

We prove Lemma A.1 in Appendix A.3.6. In the proof, we show that  $Q_0(\infty)$  is first-order stochastically dominated by the sum of  $N$  independent random variables, each representing the number of jobs that belong to a server  $i$ . We then bound the tail probabilities for each of the  $N$  random variables, thereby obtaining the desired result.

### A.3.2 Step Two: A High-Probability Event

We will provide a refined bound on the tail probability of the shared pool queue length  $Q_0(\infty)$  in Step Three. We first consider a high-probability event as a preliminary step to accomplish this.

Note that by Remark B.6, the steady-state rates of requesting and offering help are decreasing in the shared pool waiting time  $w$ . We let  $\underline{g} > 0$  denote the lower bound on the steady-state rate of offering help, which corresponds to the case of  $w = \bar{w}$ .

Let  $\pi_q$  denote the stationary probability that a server has  $q$  jobs in its queue, and  $A_q$  and  $S_q$  the probabilities that a server requests and offers help, respectively, when it has  $q$  jobs. By Remark 3.1, the steady-state rate of requesting help is  $\phi$  times the steady-state rate of offering help. Since the steady-state rate of offering help is at least  $\underline{g}$ , we have

$$\sum_q \pi_q A_q - \sum_q \pi_q S_q \leq -(1 - \phi)\underline{g}.$$

We now bound the deviation of the empirical distribution of the states of servers from their stationary distribution. Specifically, let  $\hat{\pi}$  denote the empirical distribution of the queue lengths of the  $N$  servers, i.e.,

$$\hat{\pi}_q(t) \triangleq \frac{1}{N} \sum_{i=1}^N \mathbb{1}\{Q_i(t) = q\}.$$

Consider the high-probability set  $\mathcal{E}$  as follows.

$$\mathcal{E} \triangleq \left\{ \hat{\pi} : \sum_q \hat{\pi}_q A_q - \sum_q \hat{\pi}_q S_q \leq -\frac{1}{2}(1 - \phi)\underline{g} \right\}.$$

We bound the stationary probability of the exception event  $\{\hat{\pi} \notin \mathcal{E}\}$  in Lemma A.2, which shows that the probability diminishes to zero exponentially fast.

**Lemma A.2.** *Let  $\mathbb{P}_\infty(\cdot)$  denote the stationary probability of the queue length process  $\mathbf{Q}(t)$ . For any number of servers  $N$  we have*

$$\mathbb{P}_\infty\left(\hat{\pi} \notin \mathcal{E}\right) \leq \exp\left(-\frac{1}{8}(1 - \phi)^2 \underline{g}^2 N\right).$$

*Proof.* Since every server uses an oblivious strategy (i.e., a strategy that depends only on a server's individual state), the dynamics of the  $N$  servers are independent by the design of mechanism. The above inequality follows directly from Hoeffding's inequality and the fact that  $A_q, S_q \leq 1$ .  $\square$

### A.3.3 Step Three: A Refined Bound on the Tail Probability of $Q_0(\infty)$

We now provide a refined bound on the tail probability of the shared pool queue length  $Q_0(\infty)$  in Lemma A.3. Intuitively, conditional on the high-probability event  $\{\hat{\pi} \in \mathcal{E}\}$ , the shared pool queue length tends to have a short tail, because servers route more capacity units than jobs to the shared pool (i.e.,  $Q_0(t)$  has a negative drift). On the other hand, the exception event  $\{\hat{\pi} \notin \mathcal{E}\}$  is rare by Lemma A.2.

**Lemma A.3.** *for any  $m \in \mathbb{N}$ , we have*

$$\begin{aligned} \mathbb{P}\left(Q_0(\infty) > \frac{2}{(1-\phi)\underline{g}} + 2m\right) &\leq \left(\frac{1}{1 + \frac{1}{4}(1-\phi)\underline{g}}\right)^{m+1} + \left(\frac{4}{(1-\phi)\underline{g}} + 1\right) \mathbb{P}(\hat{\pi} \notin \mathcal{E}) \\ &\leq \left(\frac{4}{4 + (1-\phi)\underline{g}}\right)^{m+1} + \left(\frac{4}{(1-\phi)\underline{g}} + 1\right) \exp\left(-\frac{1}{8}(1-\phi)^2 \underline{g}^2 N\right). \end{aligned}$$

We prove the first inequality of Lemma A.3 in Appendix A.3.7 by using a drift analysis. The second inequality follows directly from Lemma A.2.

### A.3.4 Step Four: Uniform Bound on the Expected Shared Pool Queue Length

We now apply Lemmas A.1 and A.3 to bound the expected shared pool queue length in the stationary distribution from above; specifically, we show that

$$\mathbb{E}[Q_0(\infty)] \leq \frac{10}{(1-\phi)\underline{g}} + o(1)$$

where  $o(1)$  denotes a term that diminishes to zero as the number of servers  $N$  grows to infinity. This implies that the expected shared pool queue length is uniformly bounded from above for any number of servers  $N$ . To see this, since

$$\mathbb{E}[Q_0(\infty)] = \sum_{n=0}^{\infty} \mathbb{P}(Q_0(\infty) > n) \leq \sum_{n=0}^{N^2} \mathbb{P}(Q_0(\infty) > n) + \sum_{n=N^2}^{\infty} \mathbb{P}(Q_0(\infty) > n),$$

we bound the two terms separately.

We first use Lemma A.1 to bound the second term. Specifically, we have

$$\begin{aligned} \sum_{n=N^2}^{\infty} \mathbb{P}(Q_0(\infty) > n) &= \sum_{m=0}^{\infty} \sum_{u=0}^{N-1} \mathbb{P}(Q_0(\infty) > N^2 + mN + u) \\ &\leq N \sum_{m=0}^{\infty} \mathbb{P}(Q_0(\infty) > N^2 + mN) \\ &\leq \alpha_1 \cdot N^2 \sum_{m=0}^{\infty} \exp\left(-\alpha_2 \cdot (N + m)\right) \\ &= \frac{\alpha_1}{1 - e^{-\alpha_2}} N^2 \exp(-\alpha_2 N), \end{aligned}$$

where the second inequality follows from Lemma A.1. This term diminishes to zero exponentially fast as  $N \rightarrow \infty$ .

We next bound the first term by using Lemma A.3. Specifically, we have

$$\begin{aligned}
\sum_{n=0}^{N^2} \mathbb{P}(Q_0(\infty) > n) &\leq \frac{2}{(1-\phi)\underline{g}} + \sum_{n=0}^{N^2/2} \left\{ \mathbb{P}\left(Q_0(\infty) > \frac{2}{(1-\phi)\underline{g}} + 2n\right) + \mathbb{P}\left(Q_0(\infty) > \frac{2}{(1-\phi)\underline{g}} + 2n+1\right) \right\} \\
&\leq \frac{2}{(1-\phi)\underline{g}} + 2 \sum_{n=0}^{N^2/2} \mathbb{P}\left(Q_0(\infty) > \frac{2}{(1-\phi)\underline{g}} + 2n\right) \\
&\leq \frac{2}{(1-\phi)\underline{g}} + 2 \sum_{n=0}^{N^2/2} \left( \left( \frac{4}{4 + (1-\phi)\underline{g}} \right)^{n+1} + \left( \frac{4}{(1-\phi)\underline{g}} + 1 \right) \exp\left(-\frac{1}{8}(1-\phi)^2 \underline{g}^2 N\right) \right) \\
&\leq \frac{2}{(1-\phi)\underline{g}} + 2 \sum_{n=0}^{\infty} \left( \frac{4}{4 + (1-\phi)\underline{g}} \right)^{n+1} + (N^2 + 2) \left( \frac{4}{(1-\phi)\underline{g}} + 1 \right) \exp\left(-\frac{1}{8}(1-\phi)^2 \underline{g}^2 N\right)
\end{aligned}$$

where the third inequality follows from Lemma A.3. The third term in the right-hand side of the last inequality diminishes to zero exponentially fast as  $N \rightarrow \infty$ . The remaining terms are:

$$\frac{2}{(1-\phi)\underline{g}} + 2 \sum_{n=0}^{\infty} \left( \frac{4}{4 + (1-\phi)\underline{g}} \right)^{n+1} = \frac{10}{(1-\phi)\underline{g}}.$$

### A.3.5 Step Five: Bounding the Shared Pool Waiting Time $w$

Since the steady-state rate of requesting help is at least  $\phi \cdot \underline{g}$  for any server, by Little's law we have

$$w \leq \frac{\mathbb{E}[Q_0(\infty)]}{\phi \underline{g} \cdot N} \leq \frac{M_1}{N}$$

for some constant  $M_1$  that depends only on the values of  $\lambda$ ,  $\phi$ , and  $\bar{w}$ ; the second inequality follows from the fact that the expected queue length in the shared pool is uniformly bounded from above by a constant by Step Four.

### A.3.6 Proof of Lemma A.1

In the proof, we construct a stochastic process that has a stationary distribution that first-order stochastically dominates the shared pool queue length  $Q_0(\infty)$ .

Specifically, we consider an auxiliary process, denoted as  $\tilde{\mathbf{Q}}(t) = (\tilde{Q}_i(t))_{i \in [0:N]}$ , that mimics the original queue length process  $\mathbf{Q}(t)$ . We couple the two processes together so that they have identical arrival sequences of jobs and capacity units at the servers. Each server  $i \in [N]$  in the auxiliary system takes the same actions of server  $i$  in the original system. The only difference between the two systems is that: in the auxiliary system, when server  $i \in [N]$  offers help, the provided capacity unit can only be used to serve a job in the shared pool that came from server  $i$  (if no such a job exists in the shared pool, the capacity unit is wasted). As a result, we have  $\tilde{Q}_0(t) \geq Q_0(t)$  at any time  $t \geq 0$ . Furthermore, let  $\tilde{Q}_\Sigma(t) \triangleq \sum_{i=0}^N \tilde{Q}_i(t) \geq \tilde{Q}_0(t)$  denote the total number of jobs in the auxiliary system. We have

$$\tilde{Q}_\Sigma(\infty) \succeq_1 Q_0(\infty),$$

where  $\succeq_1$  indicates first-order stochastic dominance.

Let  $Z_i(t)$  denote the number of jobs of server  $i$  in the auxiliary system at time  $t$ ; i.e.,  $Z_i$  is the



sum of the queue length of server  $i$  and the number of jobs in the shared pool that come from server  $i$ . Since there is no interaction between any two servers in the auxiliary system,  $Z_i(t)$  are independent across servers. Additionally, since each job must belong to one server,

$$\tilde{Q}_\Sigma(t) = \sum_{i=1}^N Z_i(t), \quad (4)$$

i.e.,  $\tilde{Q}_\Sigma(t)$  is the sum of the  $N$  independent random variables  $Z_i(t)$ . In Lemma A.4 we bound the tail probability of the job count  $Z_i(\infty)$  in the stationary distribution for any server  $i$ .

**Lemma A.4.** *Let  $Z_i(\infty)$  denote the number of jobs that belong to server  $i$  in the stationary distribution. For any server  $i \in [N]$  and integer  $m \in \mathbb{N}$  we have*

$$\mathbb{P}(Z_i(\infty) \geq m) \leq \alpha_1 \cdot \exp(-\alpha_2 \cdot m),$$

where  $\alpha_1, \alpha_2 > 0$  are two positive constants that depend only on the values of  $\lambda$ ,  $\phi$ , and  $\bar{w}$ .

As a result, we have

$$\begin{aligned} \mathbb{P}(Q_0(\infty) \geq mN) &\leq \mathbb{P}(\tilde{Q}_\Sigma(\infty) \geq mN) \\ &\leq \mathbb{P}(Z_i \geq m \text{ for some } i \in [N]) \\ &\leq \sum_{i=1}^N \mathbb{P}(Z_i \geq m) \\ &\leq \alpha_1 N \cdot \exp(-\alpha_2 \cdot m), \end{aligned}$$

where the first inequality follows from stochastic dominance, the second inequality from (4), the third inequality from the union bound, and the last inequality from Lemma A.4.

*Proof of Lemma A.4.* We consider a representative server  $i$  that follows an optimal policy to a certain fluid mean-field problem  $V^F(w)$  with  $w \leq \bar{w} < \infty$ . Since the server presumes the shared pool waiting time  $w$  to be smaller than a fixed constant  $\bar{w} < \infty$ , and the set of all possible optimal policies is finite when the value of  $w$  varies between zero and  $\bar{w}$  by Lemma 4.2, it suffices to prove Lemma A.4 when the server follows any of the finite candidate optimal policies.

First, suppose Case One in Lemma 4.2 holds (i.e.,  $w < m(k)$ ). In this case, the server offers help when a capacity unit arrives and its queue is empty. Therefore, the server in the auxiliary system serves a job with probability one when a capacity unit arrives and it has an affiliated job (i.e.,  $Z_i \geq 1$ ). Consequently, the dynamics of  $Z_i(t)$  is an  $M/M/1$  queue with a job arrival rate of  $\lambda$  and a job processing rate of one. Thus,  $\mathbb{P}[Z_i(\infty) = i] = (1 - \lambda)\lambda^i$  for all  $i \in \mathbb{N}$  and, as a result,

$$\mathbb{P}(Z_i \geq m) = \sum_{i=m}^{\infty} (1 - \lambda)\lambda^i = \lambda^m = \exp(\ln \lambda \cdot m).$$

The inequality in Lemma A.4 holds with  $\alpha_1 = 1$  and  $\alpha_2 = -\ln \lambda > 0$ .

Secondly, suppose Case Two in Lemma 4.2 holds with  $z = 0$ ; i.e.,  $\phi > \lambda$  and  $m(0) = 0 \leq w \leq m(1)$ . In this case, the server routes all jobs to the shared pool and offers help with probability  $\lambda/\phi$  when a capacity unit arrives. Therefore, the dynamics of  $Z_i(t)$  is an  $M/M/1$  queue with a job

arrival rate of  $\lambda$  and a job processing rate of  $\lambda/\phi$ . Thus,  $\mathbb{P}[Z_i(\infty) = i] = (1 - \phi)\phi^i$  for all  $i \in \mathbb{N}$  and, as a result,

$$\mathbb{P}(Z_i \geq m) = \sum_{i=m}^{\infty} (1 - \phi)\phi^i = \phi^m = \exp(\ln \phi \cdot m).$$

The inequality in Lemma A.4 holds with  $\alpha_1 = 1$  and  $\alpha_2 = -\ln \phi > 0$ .

Finally, suppose that Case Two in Lemma 4.2 holds with  $z \geq 1$ . In this case, the server routes an incoming job to the shared pool when its queue length reaches  $z$  and adds the job to its queue otherwise. When a capacity unit arrives, the server serves a job from its queue if one is present and offers help to the shared pool with probability  $p \triangleq \lambda^{z+1}/\phi$  otherwise. Let  $\tilde{Q}_i(t)$  and  $\tilde{Q}_{0i}(t)$  denote the queue length of server  $i$  and number of jobs in the shared pool that come from server  $i$ , respectively, in the auxiliary system (hence,  $Z_i(t) = \tilde{Q}_i(t) + \tilde{Q}_{0i}(t)$ ). The dynamics of  $(\tilde{Q}_i(t), \tilde{Q}_{0i}(t))$  is a Markov chain and is visualized in Figure 3.

Let  $\pi_{j\ell} = \mathbb{P}[\tilde{Q}_i(\infty) = j, \tilde{Q}_{0i}(\infty) = \ell]$  denote the stationary distribution of jobs of server  $i$  in the auxiliary system. By analyzing the flow balance equations of the Markov chain (similar to the approach in Appendix C.1.3), we obtain:

$$\begin{aligned} \pi_{0,\ell+1} &= \frac{\phi(1-\lambda)}{1-\lambda^{z+1}} \cdot \pi_{00} \cdot \rho^\ell, \forall \ell \geq 0, \\ \pi_{z\ell} &= \frac{\lambda^z(1-\lambda)}{1-\lambda^{z+1}} \cdot \pi_{00} \cdot \rho^\ell, \forall \ell \geq 0, \end{aligned}$$

with

$$\rho = \frac{\phi(1-\lambda) + \lambda(1-\lambda^z)}{1-\lambda^{z+1}} \in (0, 1).$$

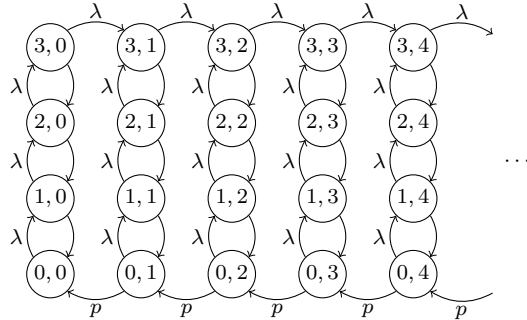
(Note that  $1 - \rho = \frac{(1-\lambda)(1-\phi)}{1-\lambda^{z+1}} > 0$ .) Therefore,

$$\pi_{j,\ell+1} = \rho \cdot \pi_{j\ell}, \forall j \in [z], \ell \geq 1,$$

i.e., the stationary probabilities diminish exponentially fast as the number of jobs in the shared pool increases. Thus,

$$\mathbb{P}(Z_i \geq m) \leq \alpha_1 \exp(\ln \rho \cdot m)$$

for some constant  $\alpha_1 > 0$ . The inequality in Lemma A.4 holds with constants  $\alpha_1$  and  $\alpha_2 = -\ln \rho > 0$ .



**Figure 3:** Markov chain of  $(\tilde{Q}_i(t), \tilde{Q}_{0i}(t))$  with  $z = 3$ . All unspecified transition rates take the value of one. □

### A.3.7 Proof of Lemma A.3

We use a drift analysis to prove Lemma A.3. To do so, we consider an equivalent discrete-time model of the system where in each period, either a job or a capacity unit arrives at some server  $i \in [N]$ . For each server  $i \in [N]$ , let binary variable  $A_i(t)$  (and  $S_i(t)$ ) indicate if server  $i$  requests help (and offers help) in period  $t$ . Moreover, let  $A(t) \triangleq \sum_{i \in [N]} A_i(t) \in \{0, 1\}$  and  $S(t) \triangleq \sum_{i \in [N]} S_i(t) \in \{0, 1\}$  indicate if there is any request or provision of help in period  $t$ . Since  $Q_0(t+1) = (Q_0(t) + A(t) - S(t))^+$ , we have

$$\begin{aligned} Q_0^2(t+1) - Q_0^2(t) &\leq (Q_0(t) + A(t) - S(t))^2 - Q_0^2(t) \\ &= (A(t) - S(t))^2 + 2Q_0(t)(A(t) - S(t)) \\ &\leq 2Q_0(t)(A(t) - S(t)) + 1, \end{aligned} \tag{5}$$

where the last inequality follows from the fact that  $|A(t) - S(t)| \leq 1$ .

We now consider two scenarios. First, when the empirical distribution of queue lengths satisfies  $\hat{\pi}(t) \in \mathcal{E}$ , from (5) we have:

$$\begin{aligned} \mathbb{E}[Q_0^2(t+1) - Q_0^2(t) | Q_0(t), \hat{\pi}(t) \in \mathcal{E}] &\leq 2\mathbb{E}[Q_0(t)(A(t) - S(t)) | Q_0(t), \hat{\pi}(t) \in \mathcal{E}] + 1 \\ &\leq -(1 - \phi)\underline{g}Q_0(t) + 1, \end{aligned} \tag{6}$$

where the second inequality follows from the definition of the event  $\mathcal{E}$ . Since  $x^2 + y^2 \geq 2xy$  for any  $x, y \in \mathbb{R}$ , for any  $Q_0(t) \geq 1$  we have

$$Q_0(t+1) - Q_0(t) \leq \frac{Q_0^2(t+1) - Q_0^2(t)}{2Q_0(t)}.$$

Taking expectations on both sides of the above and applying (6) yields that for any  $Q_0(t) \geq 1$ ,

$$\mathbb{E}[Q_0(t+1) - Q_0(t) | Q_0(t), \hat{\pi}(t) \in \mathcal{E}] \leq -\frac{1}{2}(1 - \phi)\underline{g} + \frac{1}{2Q_0(t)}.$$

As a result, when  $Q_0(t) \geq \frac{2}{(1-\phi)\underline{g}}$ ,

$$\mathbb{E}[Q_0(t+1) - Q_0(t) | Q_0(t), \hat{\pi}(t) \in \mathcal{E}] \leq -\frac{1}{4}(1 - \phi)\underline{g}.$$

On the other hand, since the queue length changes by at most one per period, we have

$$\mathbb{E}[Q_0(t+1) - Q_0(t) | Q_0(t), \hat{\pi}(t) \notin \mathcal{E}] \leq 1.$$

Applying Lemma A.5 with Lyapunov function  $L(\mathbf{Q}(t)) = Q_0(t)$  (note that Lemma A.1 implies  $\mathbb{E}[Q_0(\infty)] < \infty$ ) and constants  $\gamma = \frac{1}{4}(1 - \phi)\underline{g}$ ,  $B = \frac{2}{(1-\phi)\underline{g}}$ ,  $\delta = 1$ , and  $\nu_{\max} = \max |Q_0(t+1) - Q_0(t)| = 1$  yields the desired result:

$$\mathbb{P}\left(Q_0(\infty) > \frac{2}{(1-\phi)\underline{g}} + 2m\right) \leq \left(\frac{1}{1 + \frac{1}{4}(1-\phi)\underline{g}}\right)^{m+1} + \left(\frac{4}{(1-\phi)\underline{g}} + 1\right) \mathbb{P}(\hat{\pi} \notin \mathcal{E}).$$

**Lemma A.5** (Lemma 10 of Liu et al. 2022). *Let  $X(t) \in \mathcal{X}$  be a positive recurrent Markov chain*

with a countable state space  $\mathcal{X}$  and transition probability  $p(\mathbf{x}, \mathbf{x}') = \mathbb{P}[X(t+1) = \mathbf{x}' | X(t) = \mathbf{x}]$ , and  $X(\infty)$  denote the Markov chain in the stationary distribution. Additionally, let  $L(\cdot) : \mathcal{X} \rightarrow \mathbb{R}_+$  be a non-negative function (called Lyapunov function) with  $\mathbb{E}[L(X(\infty))] < \infty$ . Define the drift of  $L(\cdot)$  at state  $\mathbf{x} \in \mathcal{X}$  as

$$\nabla L(\mathbf{x}) = \mathbb{E}[L(X(t+1)) | X(t) = \mathbf{x}] - L(\mathbf{x}).$$

Suppose that there exists a subset  $\mathcal{E} \subseteq \mathcal{X}$  and constants  $\gamma > 0$ ,  $B \geq 0$  and  $\delta \geq 0$  such that:

- (i)  $\nabla L(\mathbf{x}) \leq -\gamma$  when  $L(\mathbf{x}) > B$  and  $\mathbf{x} \in \mathcal{E}$ ; and
- (ii)  $\nabla L(\mathbf{x}) \leq \delta$  when  $L(\mathbf{x}) > B$  and  $\mathbf{x} \notin \mathcal{E}$ .

Then, for any integer  $m \in \mathbb{N}$ , we have

$$\mathbb{P}[L(X(\infty)) > B + 2\nu_{\max}m] \leq \left(\frac{\nu_{\max}}{\nu_{\max} + \gamma}\right)^{m+1} + \left(\frac{\delta}{\gamma} + 1\right) \cdot \mathbb{P}[X(\infty) \notin \mathcal{E}],$$

where  $\nu_{\max} \triangleq \sup_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}: p(\mathbf{x}, \mathbf{x}') > 0} |L(\mathbf{x}') - L(\mathbf{x})|$  denotes the largest possible change of the function  $L(\cdot)$  in a transition.

## B Proof of Lemma 4.2: Optimal Policy of (2)

In this proof, we focus on an equivalent embedded discrete-time model of a server's problem, where in each period, either a job or a capacity unit arrives, with probability  $\frac{\lambda}{1+\lambda}$  and  $\frac{1}{1+\lambda}$ , respectively. The decisions in period  $t$  are: (a) when a job arrives, we let  $X_t = 1$  if the server relocates the job to the shared pool and  $X_t = 0$  if the server adds the job to its queue; (b) when a capacity unit arrives, we let  $Y_t = 1$  if the server offers help to the shared pool,  $Y_t = 0$  if the server serves a job from its queue, and  $Y_t = \emptyset$  if the server opts to be idle.

### B.1 Lagrangian Relaxation of (2)

We first introduce a Lagrangian relaxation to (2), which is the key to our proof. Specifically, we dualize the flow balance constraint of the tokens in (2) (or equivalently, (3)) via a dual variable  $\mu \in \mathbb{R}$ . We denote the Lagrangian relaxation by  $V^{\text{FLR}}(\mu)$ , which is provided in (7).

$$V^{\text{FLR}}(\mu) = \min_{\pi \in \Pi} \sum_{q \in \mathbb{N}} \left\{ \frac{q}{1+\lambda} \cdot p^\pi(q) + c \cdot \mathbb{P}^\pi(q, Y=0) + \phi \cdot (c-\mu) \cdot \mathbb{P}^\pi(q, Y=1) + (w+\mu) \cdot \mathbb{P}^\pi(q, X=1) \right\}, \quad (7)$$

In (7),  $p^\pi(q)$  represents the stationary distribution of the server's queue length, and  $\mathbb{P}^\pi(q, X)$  and  $\mathbb{P}^\pi(q, Y)$  the stationary probability of having  $q$  jobs in the queue and taking the action  $X$  and  $Y$ , respectively, under policy  $\pi$ . In (7), we no longer require that the expected rates of earning and spending tokens are equal; instead, we introduce a penalty  $\mu$  for spending a token and a reward  $-\mu$  for earning a token.

The problem  $V^{\text{FLR}}(\mu)$  can be formulated as a one-dimensional DP, where the queue length  $q$  is the state variable. In theory, the queue length  $q$  can take any integer; thus, the DP has a countable state space. However, as we show in Remark B.1, the queue length is bounded from above under any optimal policy of  $V^{\text{FLR}}(\mu)$  and for any dual variable  $\mu \in \mathbb{R}$ . Therefore, without loss of generality, we can treat  $V^{\text{FLR}}(\mu)$  as a finite-state DP problem. This gets rid of the intricacy related to analyzing infinite-state average-cost DP problems.

**Remark B.1** (Bounded Queue Length under an Optimal Policy). Suppose the queue length is  $q$  and a job arrives. If the server relocates the job to the shared pool, the cost is  $\mu + w$ . On the other hand, if the server adds the job to the queue, the job will wait for at least  $(q + 1)$  in expectation in continuous time before being served, thus incurring an expected holding cost of at least  $(q + 1)$ . Therefore, a server that minimizes the time-average cost will always route a job to the shared pool if  $q + 1 > \mu + w$ . Consequently, the queue length will be no longer than  $\max\{0, \lfloor \mu + w \rfloor\}$ .

Since every feasible policy to (2) is feasible to (7) and attains an objective that is no larger,  $V^{\text{FLR}}(\mu) \leq V^{\text{F}}$  for any  $\mu \in \mathbb{R}$ . We formally state this weak-duality property in Lemma B.1.

**Lemma B.1** (Weak Duality). *We have  $V^{\text{FLR}}(\mu) \leq V^{\text{F}}$  for any dual variable  $\mu \in \mathbb{R}$ .*

Note that  $V^{\text{FLR}}(\mu)$  is a concave function of  $\mu$  by (7). Therefore, we can solve a convex optimization problem

$$V^{\text{FLR}} \triangleq \max_{\mu \in \mathbb{R}} V^{\text{FLR}}(\mu) \leq V^{\text{F}}$$

to obtain the tightest Lagrangian relaxation bound  $V^{\text{FLR}}$ . We let  $\mu^* = \operatorname{argmax}_{\mu \in \mathbb{R}} V^{\text{FLR}}(\mu)$  denote an optimal Lagrangian dual variable. Lemma B.2 shows that strong duality holds.

**Lemma B.2** (Strong Duality). *The problem (2) and its Lagrangian relaxation (7) have the following relationship.*

1. *Strong duality holds, i.e.,  $V^{\text{FLR}} = V^{\text{F}}$ . Moreover, let  $\mu^*$  be an optimal dual variable; there exists an optimal policy  $\pi^*$  of  $V^{\text{FLR}}(\mu^*)$  that is feasible and optimal to  $V^{\text{F}}$ .*
2. *Suppose that a policy  $\pi$  is feasible to  $V^{\text{F}}$  (i.e., the tokens' flow balance constraint is satisfied under policy  $\pi$ ) and is optimal to  $V^{\text{FLR}}(\mu)$  with some  $\mu \in \mathbb{R}$ . Then,  $\pi$  is an optimal policy of  $V^{\text{F}}$  and  $\mu$  is an optimal dual variable.*
3. *Let  $\mu^*$  be an optimal dual variable. Then, any policy  $\pi^*$  that is optimal to  $V^{\text{F}}$  is optimal to  $V^{\text{FLR}}(\mu^*)$ .*

*Proof.* According to Danskin's Theorem (Proposition 4.5.1 in Bertsekas et al. 2003) and the fact that randomizing over two optimal policies of (7) is also optimal to (7),<sup>13</sup> the sub-differential (i.e., set of sub-gradients) of  $V^{\text{FLR}}(\mu)$  at any  $\mu \in \mathbb{R}$ , denoted by  $\partial V^{\text{FLR}}(\mu)$ , can be expressed as

$$\partial V^{\text{FLR}}(\mu) = \left\{ \sum_{q \in \mathbb{N}} \mathbb{P}^{\pi}(q, X = 1) - \phi \cdot \sum_{q \in \mathbb{N}} \mathbb{P}^{\pi}(q, Y = 1) : \pi \text{ is an optimal policy of } V^{\text{FLR}}(\mu) \right\}.$$

By the optimality condition of  $\mu^*$ , we have  $0 \in \partial V^{\text{FLR}}(\mu^*)$ . Therefore, there exists an optimal policy  $\pi^*$  of  $V^{\text{FLR}}(\mu^*)$  such that  $\sum_{q \in \mathbb{N}} \mathbb{P}^{\pi^*}(q, X = 1) = \phi \cdot \sum_{q \in \mathbb{N}} \mathbb{P}^{\pi^*}(q, Y = 1)$ ; i.e., policy  $\pi^*$  is feasible to (2). As a result,

$$\begin{aligned} V^{\text{FLR}} &= \sum_{q \in \mathbb{N}} \left\{ \frac{q}{1 + \lambda} \cdot p^{\pi^*}(q) + c \cdot \mathbb{P}^{\pi^*}(q, Y = 0) + \phi \cdot (c - \mu^*) \cdot \mathbb{P}^{\pi^*}(q, Y = 1) + (w + \mu^*) \cdot \mathbb{P}^{\pi^*}(q, X = 1) \right\} \\ &= \sum_{q \in \mathbb{N}} \left\{ \frac{q}{1 + \lambda} \cdot p^{\pi^*}(q) + c \cdot \left( \mathbb{P}^{\pi^*}(q, Y = 0) + \phi \cdot \mathbb{P}^{\pi^*}(q, Y = 1) \right) + w \cdot \mathbb{P}^{\pi^*}(q, X = 1) \right\} \geq V^{\text{F}} \geq V^{\text{FLR}}, \end{aligned} \tag{8}$$

<sup>13</sup>This is because the Lagrangian problem (7) can be formulated as a linear program.

where the first equality follows from  $\pi^*$  being optimal to  $V^{\text{FLR}}(\mu^*)$ , the second equality from the fact that the flow balance constraint of the tokens holds under policy  $\pi^*$ , the first inequality is because policy  $\pi^*$  is feasible to (2), and the second inequality follows from the weak duality (Lemma B.1). Therefore, all of the inequalities in (8) are binding, which implies that strong duality holds and policy  $\pi^*$  is optimal to  $V^{\text{F}}$ .

Bullets 2 and 3 can be easily derived in a similar way using a chain of inequalities analogous to (8).  $\square$

Finally, we remark that the Lagrangian relaxation  $V^{\text{FLR}}(\mu)$  can be solved by the following Bellman equation (9)

$$\begin{aligned} v + h(q, 1) &= \frac{q}{1 + \lambda} + \min \{h(q + 1), h(q) + w + \mu\} \\ v + h(q, 0) &= \frac{q}{1 + \lambda} + \min \{h(q), h(q - 1) + c, h(q) + \phi(c - \mu)\}, \forall q \geq 1 \\ v + h(q, 0) &= \frac{q}{1 + \lambda} + \min \{h(q), h(q) + \phi(c - \mu)\}, q = 0. \end{aligned} \quad (9)$$

where  $v = V^{\text{FLR}}(\mu)$  denotes the optimal time-average cost,  $h(q, 1)$  the differential value function when there are  $q$  jobs and a job arrives,  $h(q, 0)$  the differential value function when there are  $q$  jobs and a capacity unit arrives, and  $h(q) = \frac{\lambda}{\lambda + 1}h(q, 1) + \frac{1}{\lambda + 1}h(q, 0)$  the average differential value function.

## B.2 Case One: $w < m(k)$

Since  $0 \leq w < m(k)$  and  $m(0) = 0$ , we must have  $k \geq 1$  and hence  $\phi \leq \lambda^k \leq \lambda$  in this case.

### B.2.1 The Server's Dynamics

When a capacity unit arrives, the server offers help when the queue is empty and serves a job from the queue otherwise – hence, the server never deliberately idles. In addition, the queue length  $q$  is in the interval  $[0 : k]$  in the long run. Let  $\pi_i \triangleq \mathbb{P}[q = i]$  denote the stationary probability of having  $q$  jobs in the queue; we have

$$\pi_i = \begin{cases} \lambda^i \pi_0 & i \leq k - 1, \\ \lambda^i (1 - p) \pi_0 & i = k, \\ 0 & i \geq k + 1. \end{cases}$$

The value of  $p = \frac{\phi - \lambda^{k+1}}{\lambda^k - \lambda^{k+1}} \in [0, 1]$  ensures that the rates of earning and spending tokens are equal, i.e.,

$$\pi_0 \phi = \lambda(\pi_k + \pi_{k-1} p). \quad (10)$$

Thus, the policy is feasible to (2). Finally, we have  $\pi_0 = \frac{1 - \lambda}{1 - \phi} \in [0, 1]$  by  $\sum_{i \geq 0} \pi_i = 1$ .

Under the policy, the expected queue length is

$$\mathbb{E}[q] = \sum_{i \geq 0} i \pi_i = \frac{1}{1 - \phi} \left( \sum_{i=1}^k \lambda^i - k \phi \right), \quad (11)$$

the expected job processing cost is

$$c \cdot \frac{1}{1 + \lambda} \cdot \left[ (1 - \pi_0) + \pi_0 \cdot \phi \right] = \frac{c \lambda}{1 + \lambda},$$

and the expected waiting cost for jobs routed to the shared pool is

$$\frac{\lambda}{1+\lambda} \cdot (\pi_k + \pi_{k-1}p) \cdot w = \pi_0 \cdot \frac{\phi}{1+\lambda} \cdot w = \frac{1-\lambda}{1-\phi} \cdot \frac{\phi}{1+\lambda} \cdot w,$$

where the first equality follows from (10). Thus, the expected total cost of the policy is

$$v \triangleq \frac{1}{1+\lambda} \cdot \frac{1}{1-\phi} \left( \sum_{i=1}^k \lambda^i - k\phi \right) + \frac{c\lambda}{1+\lambda} + \frac{1-\lambda}{1-\phi} \cdot \frac{\phi}{1+\lambda} \cdot w. \quad (12)$$

Note that if the server instead operates independently, the expected total cost is  $\frac{1}{1+\lambda} \cdot \frac{\lambda}{1-\lambda} + \frac{c\lambda}{1+\lambda}$ , where  $\frac{\lambda}{1-\lambda}$  is the expected queue length of an  $M/M/1$  queue. We have

$$\frac{1}{1+\lambda} \cdot \frac{\lambda}{1-\lambda} + \frac{c\lambda}{1+\lambda} - v \geq \frac{\lambda^{1+k}}{1-\lambda^2} > 0,$$

where the first inequality follows from  $w \leq m(k)$ .

### B.2.2 Proof of Case One

We now verify that the policy is optimal to (2) when  $w < m(k)$ . To do so, since the policy is feasible to (2), if we can find a dual variable  $\mu$  such that the policy is optimal to  $V^{\text{FLR}}(\mu)$ , then  $\mu = \mu^*$  is an optimal dual variable and the policy is optimal to (2) by Lemma B.2 Bullet 2.

**Step One: Finding the Dual Variable  $\mu$**  To find an appropriate dual variable  $\mu$ , note that when  $q = k - 1$  and a job arrives, the server randomizes between relocating the job to the shared pool ( $X = 1$ ) and adding the job to the queue ( $X = 0$ ). If the policy is optimal to  $V^{\text{FLR}}(\mu)$ , then it must be indifferent between taking  $X = 0$  and  $X = 1$ , and we can use this to determine the value of  $\mu$ . Specifically, let  $h(q)$  denote the average differential value function of the policy in  $V^{\text{FLR}}(\mu)$ . The indifference between the actions  $X = 0$  and  $X = 1$  at  $q = k - 1$  implies that (please refer to (9))

$$h(k) = h(k-1) + w + \mu. \quad (13)$$

On the other hand, the Bellman equation of the policy at state  $q = k$  is

$$v + h(k) = \frac{k}{1+\lambda} + \frac{\lambda}{1+\lambda} (h(k) + w + \mu) + \frac{1}{1+\lambda} (h(k-1) + c) \quad (14)$$

with  $v$  specified in (12). From (12), (13), and (14), we have

$$\mu = c + \frac{m(k)}{1-\phi} - \frac{w}{1-\phi}. \quad (15)$$

**Remark B.2.** From (15), We have  $\mu > c$  because  $w < m(k)$ .

**Step Two: Properties of the Differential Value Function** Before we verify that the policy is optimal to the Lagrangian  $V^{\text{FLR}}(\mu)$  with  $\mu$  specified in (15), we first characterize the policy's average differential value function  $h(q)$  in the Lagrangian  $V^{\text{FLR}}(\mu)$ . These differential value functions  $h(q)$

can be derived from the policy's Bellman equations (16) – (18).

$$v + h(0) = \frac{\lambda}{1+\lambda}h(1) + \frac{1}{1+\lambda}(h(0) + \phi(c - \mu)), \quad (16)$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda}h(q+1) + \frac{1}{1+\lambda}(h(q-1) + c), \quad \forall q \in [1 : k-1], \quad (17)$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda}(h(q) + w + \mu) + \frac{1}{1+\lambda}(h(q-1) + c), \quad \forall q \geq k-1. \quad (18)$$

Note that (17) is the Bellman equation for  $q \in [1 : k-2]$ , but since  $h(k-1) + w + \mu = h(k)$  by (13), (17) is also valid for  $q = k-1$ . Analogously, (18) is the Bellman equation for  $q \geq k$ , but it is also valid for  $q = k-1$  again by (13).

We can derive the values of  $h(q)$  from (16) – (18). First, without loss of generality, we can assume  $h(0) = 0$ . Then, from (16), we can obtain  $h(1)$  as

$$h(1) = \frac{(1+\lambda)v + \phi(\mu - c)}{\lambda}. \quad (19)$$

Once we have  $h(0) = 0$  and  $h(1)$ , we can determine the values of  $h(q)$  for  $q \leq k$  from (17) recursively:

$$h(q) = \frac{(v + h(q-1))(1+\lambda) - (q-1) - (h(q-2) + c)}{\lambda}, \quad \forall q \in [2 : k]. \quad (20)$$

For example, from (20) we have

$$h(2) = \frac{(v + h(1))(1+\lambda) - 1 - c}{\lambda}. \quad (21)$$

Finally, from (18) we have

$$h(q) = h(q-1) + q + \lambda(w + \mu) + c - (1+\lambda)v, \quad \forall q \geq k-1. \quad (22)$$

(20) and (22) imply important structural properties of  $h(q)$  as we state in Lemma B.3.

**Lemma B.3.** *Let  $h(q)$  be the policy's average differential value function in the Lagrangian  $V^{\text{FLR}}(\mu)$  with  $\mu$  specified in (15), and  $\Delta h(q) = h(q) - h(q-1)$  for  $q \geq 1$  the difference of the differential values of two adjacent states. Then,  $\Delta h(q)$  is positive and strictly increasing.*

*Proof.* From (22), when  $q \geq k-1$ ,

$$\Delta h(q) = q + \lambda(w + \mu) + c - (1+\lambda)v = q + \text{const}, \quad \forall q \geq k-1, \quad (23)$$

which is strictly increasing. Hence, to prove that  $\Delta h(q)$  is strictly increasing for all  $q \geq 1$ , it suffices to show that  $\Delta h(q)$  is concave, i.e.,  $\Delta h(q+1) - \Delta h(q) \geq \Delta h(q+2) - \Delta h(q+1)$  for all  $q \geq 1$ .

We now show  $\Delta h(q)$  is concave. From (20) we have

$$\Delta h(q) = \frac{1+\lambda}{\lambda}\Delta h(q-1) - \frac{1}{\lambda}\Delta h(q-2) - \frac{1}{\lambda}, \quad \forall q \in [3 : k].$$

Hence,

$$\Delta h(q) = a\left(\frac{1}{\lambda}\right)^q + b + \frac{q}{1-\lambda}, \quad \forall q \leq k, \quad (24)$$

where  $a$  and  $b$  are two constants that depend on the initial values  $\Delta h(1) = h(1)$  and  $\Delta h(2) = h(2) -$



$h(1)$ . To see this, note that  $a_n = \frac{n}{1-\lambda}$  is a special solution to the sequence  $a_n = \frac{1+\lambda}{\lambda}a_{n-1} - \frac{1}{\lambda}a_{n-2} - \frac{1}{\lambda}$ , and  $a_n = a\left(\frac{1}{\lambda}\right)^n + b$  is the general solution to the sequence  $a_n = \frac{1+\lambda}{\lambda}a_{n-1} - \frac{1}{\lambda}a_{n-2}$  because  $\frac{1}{\lambda}$  and 1 are the roots of the quadratic equation  $x^2 = \frac{1+\lambda}{\lambda}x - \frac{1}{\lambda}$ .

Additionally, we have

$$a = \left( \Delta h(2) - \Delta h(1) - \frac{1}{1-\lambda} \right) \cdot \frac{\lambda^2}{1-\lambda} = -\frac{\lambda^{k+1}}{(1-\lambda)^2} < 0$$

where the second equality follows from (12), (15), (19), and (21). Since  $a < 0$ ,  $\Delta h(q)$  is concave for  $q \leq k$  by (24). Moreover, since  $\Delta h(q)$  is concave (actually linear) for  $q \geq k-1$  by (23),  $\Delta h(q)$  is concave for all  $q \geq 1$ . This together with the fact that  $\Delta h(q+1) - \Delta h(q) = 1 \geq 0$  for all  $q \geq k-1$  (equation (23)) implies that  $\Delta h(q)$  is strictly increasing for all  $q \geq 1$ .

Finally, since  $\Delta h(q)$  is increasing,  $\Delta h(q) \geq \Delta h(1) = h(1) > 0$  for all  $q \geq 1$ , which implies that  $h(q)$  is strictly increasing in  $q$ .  $\square$

**Step Three: Verifying Optimality** We now verify that the policy is optimal to  $V^{\text{FLR}}(\mu)$  with  $\mu$  specified in (15). It suffices to show that the policy attains the minimum in the Bellman equation (9) with  $v$  specified in (12) and  $h(q)$  being the average differential value function of the policy.

First, suppose a job arrives. Since  $\Delta h(q)$  is strictly increasing in  $q$  by Lemma B.3 and  $\Delta h(k) = \mu + w$  by (13), we have that  $\Delta h(q) < \mu + w$  for  $q \leq k-1$  and  $\Delta h(q) > \mu + w$  for  $q \geq k+1$ . Thus, it is strictly optimal to add the job to the queue when  $q \leq k-2$  and strictly optimal to relocate the job to the shared pool when  $q \geq k$ , and the server is indifferent between the two actions when  $q = k-1$ .

Next, suppose a capacity unit arrives. Since  $\mu > c$  by Remark B.2, it is strictly optimal to offer help when  $q = 0$ . When  $q \geq 1$ , since  $\Delta h(q)$  is increasing in  $q$  by Lemma B.3, to ensure that it is strictly optimal to process a job in the queue, it suffices to check that it is the case when  $q = 1$  – i.e.,  $h(1) + \phi(c - \mu) > h(0) + c = c$ . To see this, note that from (12), (15), and (19) we have

$$h(1) + \phi(c - \mu) - c = \frac{1}{\lambda} \left[ ((1+\lambda)v - c\lambda) + (1-\lambda)\phi(\mu - c) \right] > 0.$$

**Remark B.3** (Uniqueness of Optimal Policy). The policy is the *unique* optimal policy of (2). To see this, note that by Lemma B.2 Bullet 3, any optimal policy of (2) is optimal to  $V^{\text{FLR}}(\mu)$  with  $\mu$  specified in (15). On the other hand, the above discussion indicates that the proposed policy is the unique optimal policy to  $V^{\text{FLR}}(\mu)$  that satisfies the flow balance constraint of the tokens. (Note that the differential value functions in (9) are unique up to a constant by Remark B.1 and Proposition 5.5.1 of Bertsekas (2017).)

**Remark B.4.** The same proof also indicates that the policy proposed in case one is optimal to (2) when  $w = m(k)$  with  $k \geq 1$  (hence  $\phi \leq \lambda$ ). The case of  $w = m(0) = 0$  and  $k = 0$  (hence  $\phi > \lambda$ ) will be covered in case two (Appendix B.3).

### B.3 Case Two: $w \geq m(k)$

We now consider the case of  $w \geq m(k)$ . Let  $z \geq k$  be an integer that satisfies  $m(z) \leq w \leq m(z+1)$ .

### B.3.1 The Server's Dynamics

With the policy, the queue length  $q$  is in the interval  $[0 : z]$  in the long run. The stationary distribution of the queue length, denoted by  $\pi_i \triangleq \mathbb{P}[q = i]$ , satisfies

$$\pi_i = \begin{cases} \lambda^i \pi_0 & i \leq z, \\ 0 & i \geq z + 1. \end{cases}$$

Hence,  $\pi_0 = \frac{1-\lambda}{1-\lambda^{z+1}} \in [0, 1]$  by  $\sum_{i \geq 0} \pi_i = 1$ . The rates of earning and spending tokens are equal because

$$\pi_0 \cdot \frac{\phi}{1+\lambda} \cdot \frac{\lambda^{z+1}}{\phi} = \pi_z \cdot \frac{\lambda}{1+\lambda}.$$

Therefore, the policy is feasible to (2).

The expected total cost of the policy with a threshold value  $z$  is

$$v(z) = \frac{c\lambda}{1+\lambda} + \frac{1}{1+\lambda} \cdot \sum_{i=0}^z i\pi_i + \pi_z \cdot \frac{\lambda}{1+\lambda} \cdot w. \quad (25)$$

We remark that  $v(z) \leq v(z+1)$  if and only if  $w \leq m(z+1)$ . The benefit of the policy relative to operating independently is

$$\frac{c\lambda}{1+\lambda} + \frac{1}{1+\lambda} \cdot \frac{\lambda}{1-\lambda} - v(z) \geq \frac{\lambda^{2+z}}{1-\lambda^2} > 0,$$

where the first inequality follows from  $w \leq m(z+1)$ .

### B.3.2 Proof of Case Two

We now verify that the policy is optimal to (2) when  $w \geq m(k)$ . Since the policy is feasible to (2), analogous to Appendix B.2.2, if we can find a dual variable  $\mu$  such that the policy is optimal to  $V^{\text{FLR}}(\mu)$ , then  $\mu = \mu^*$  is an optimal dual variable and the policy is optimal to (2) by Lemma B.2 Bullet 2. In the following, we show that the policy is optimal to  $V^{\text{FLR}}(\mu)$  with  $\mu = c$ .

**Step One: Properties of the Differential Value Function** We first characterize the policy's average differential value function  $h(q)$  in the Lagrangian  $V^{\text{FLR}}(\mu = c)$ . These differential value functions  $h(q)$  satisfy the Bellman equations (26) – (28).

$$v + h(0) = \frac{\lambda}{1+\lambda} h(1) + \frac{1}{1+\lambda} h(0), \quad (26)$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda} h(q+1) + \frac{1}{1+\lambda} (h(q-1) + c), \quad \forall q \in [1 : z-1], \quad (27)$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda} (h(q) + w + c) + \frac{1}{1+\lambda} (h(q-1) + c), \quad \forall q \geq z, \quad (28)$$

with  $v$  given in (25).

We can derive the values of  $h(q)$  from (26) – (28). First, without loss of generality, assume  $h(0) = 0$ . Then, from (26) we have

$$h(1) = \frac{1+\lambda}{\lambda} v. \quad (29)$$

Once we have  $h(0) = 0$  and  $h(1)$ , we can determine the values of  $h(q)$  for  $q \leq z$  from (27) recursively:

$$h(q) = \frac{(v + h(q-1))(1+\lambda) - (q-1) - (h(q-2) + c)}{\lambda}, \quad \forall q \in [2 : z]. \quad (30)$$

For example, from (30) we have

$$h(2) = \frac{(v + h(1))(1+\lambda) - 1 - c}{\lambda}. \quad (31)$$

Finally, from (28) we have

$$h(q) = h(q-1) + q + \lambda w + (1+\lambda)c - (1+\lambda)v, \quad \forall q \geq z. \quad (32)$$

(30) and (32) imply the same structural properties of  $h(q)$  as in Lemma B.3, which we state in Lemma B.4.

**Lemma B.4.** *Let  $h(q)$  be the policy's average differential value function in the Lagrangian  $V^{\text{FLR}}(\mu = c)$  and  $\Delta h(q) = h(q) - h(q-1)$  for  $q \geq 1$  the difference of the differential values of two adjacent states. Then,  $\Delta h(q)$  is positive and strictly increasing.*

*Proof.* From (32), when  $q \geq z$ ,

$$\Delta h(q) = q + \lambda w + (1+\lambda)c - (1+\lambda)v = q + \text{const}, \quad \forall q \geq z, \quad (33)$$

which is strictly increasing. Thus, to prove that  $\Delta h(q)$  is strictly increasing for all  $q \geq 1$ , it suffices to show that  $\Delta h(q)$  is concave, i.e.,  $\Delta h(q+1) - \Delta h(q) \geq \Delta h(q+2) - \Delta h(q+1)$  for all  $q \geq 1$ .

We now show  $\Delta h(q)$  is concave. Note that from (30) we have

$$\Delta h(q) = \frac{1+\lambda}{\lambda} \Delta h(q-1) - \frac{1}{\lambda} \Delta h(q-2) - \frac{1}{\lambda}, \quad \forall q \in [3 : z].$$

Hence,

$$\Delta h(q) = a \left( \frac{1}{\lambda} \right)^q + b + \frac{q}{1-\lambda}, \quad \forall q \leq z, \quad (34)$$

where  $a$  and  $b$  are two constants that depend on the initial values  $\Delta h(1) = h(1)$  and  $\Delta h(2) = h(2) - h(1)$ . To see this, note that  $a_n = \frac{n}{1-\lambda}$  is a special solution to the sequence  $a_n = \frac{1+\lambda}{\lambda} a_{n-1} - \frac{1}{\lambda} a_{n-2} - \frac{1}{\lambda}$ , and  $a_n = a \left( \frac{1}{\lambda} \right)^n + b$  is the general solution to the sequence  $a_n = \frac{1+\lambda}{\lambda} a_{n-1} - \frac{1}{\lambda} a_{n-2}$  because  $\frac{1}{\lambda}$  and 1 are the roots of the quadratic equation  $x^2 = \frac{1+\lambda}{\lambda} x - \frac{1}{\lambda}$ .

Additionally, we have

$$a = \left( \Delta h(2) - \Delta h(1) - \frac{1}{1-\lambda} \right) \cdot \frac{\lambda^2}{1-\lambda} \leq -\frac{\lambda^{z+2}}{(1-\lambda)^2} < 0$$

where the first inequality follows from (25), (29), (31) and the fact that  $w \leq m(z+1)$ . Since  $a < 0$ ,  $\Delta h(q)$  is concave for  $q \leq z$  by (34). Moreover, since  $\Delta h(q)$  is concave for  $q \geq z$  by (33), and

additionally,

$$\begin{aligned}
(\Delta h(z) - \Delta h(z-1)) - (\Delta h(z+1) - \Delta h(z)) &= a \left[ \left( \frac{1}{\lambda} \right)^z - \left( \frac{1}{\lambda} \right)^{z-1} \right] + \frac{1}{1-\lambda} - 1 \\
&= \frac{\lambda(1-\lambda)}{1-\lambda^{z+1}} \cdot (w - m(z)) \\
&\geq 0,
\end{aligned}$$

$\Delta h(q)$  is concave for all  $q$ . This together with the fact that  $\Delta h(q+1) - \Delta h(q) = 1 \geq 0$  for all  $q \geq z$  (equation (33)) implies that  $\Delta h(q)$  is strictly increasing for all  $q \geq 1$ .

Finally, since  $\Delta h(q)$  is increasing,  $\Delta h(q) \geq \Delta h(1) = h(1) > 0$  for all  $q \geq 1$ , which implies that  $h(q)$  is strictly increasing in  $q$ .  $\square$

**Step Two: Verifying Optimality** We now verify that the policy is optimal to  $V^{\text{FLR}}(\mu = c)$ . It suffices to show that the policy attains the minimum in the Bellman equation (9) with the value of  $v$  in (25) and  $h(q)$  being the average differential value function of the policy.

First, suppose that a capacity unit arrives. When  $q = 0$ , since  $\mu = c$ , the server is indifferent between offering help and being idle, whereas offering help with probability  $\lambda^{1+z}/\phi$  ensures that the tokens' flow balance constraint holds and hence the policy is feasible to (2). When  $q \geq 1$ , we show that it is optimal to serve a job from the queue. Since  $\Delta h(q)$  is increasing in  $q$  by Lemma B.4, it suffices to verify that it is so when  $q = 1$  – i.e.,  $h(1) \geq h(0) + c = c$ . To see this, note that from (25) and (29), we have

$$h(1) - c = \frac{1+\lambda}{\lambda} v - c \geq \frac{1-\lambda^z}{1-\lambda} \geq 0,$$

where the first inequality follows from  $w \geq m(z)$ . Particularly, if  $w > m(z)$ , it is strictly optimal to serve a job from the queue for any  $q \geq 1$ .

Next, suppose that a job arrives. Since  $\Delta h(q)$  is increasing in  $q$  by Lemma B.4, it suffices to check that (i) when  $q = z$ , it is optimal to relocate the job to the shared pool, and (ii) when  $q = z-1$ , it is optimal to add the job to the queue. To verify (i), we show that  $h(z+1) \geq h(z) + w + c$ . Note that from (33), we have

$$h(z+1) - (h(z) + w + c) = (z+1) + \lambda c - (1+\lambda)v - (1-\lambda)w = \frac{1-\lambda}{1-\lambda^{z+1}} (m(z+1) - w) \geq 0.$$

In particular, when  $w < m(z+1)$ , it is strictly optimal to relocate the job to the shared pool. To verify (ii), we show that  $h(z-1) + w + c \geq h(z)$ . Again from (33), we have

$$h(z) - (h(z-1) + w + c) = z + \lambda c - (1+\lambda)v - (1-\lambda)w = \frac{1-\lambda}{1-\lambda^{z+1}} (m(z) - w) \leq 0.$$

In particular, when  $w > m(z)$ , it is strictly optimal to add the job to the queue.

**Remark B.5 (Uniqueness of Optimal Policy).** Analogous to Remark B.3, the policy is the *unique* optimal policy of (2) when  $w \in (m(z), m(z+1))$  for any  $z \geq k$ . To see this, note that by Lemma B.2 Bullet 3, any optimal policy of (2) is optimal to  $V^{\text{FLR}}(\mu = c)$ . On the other hand, the above discussion indicates that the proposed policy is the unique optimal policy to  $V^{\text{FLR}}(\mu = c)$  that satisfies the flow balance constraint of the tokens.

## B.4 Uniqueness of the Optimal Policy

We denote the proposed policy by  $\pi^*(w)$  when  $w \in \mathbb{R}_+ \setminus \{m(z) : z \geq k\}$ . From Remarks B.3 and B.5,  $\pi^*(w)$  is the unique optimal policy of (2) for any  $w \in \mathbb{R}_+ \setminus \{m(z) : z \geq k\}$ .

When  $w = m(z)$  with some integer  $z \geq k$ , let  $\pi^*(w+)$  and  $\pi^*(w-)$  be the unique optimal policy of (2) when  $w \in (m(z), m(z+1))$  and  $w \in (m(z-1), m(z))$ , respectively. From Remark B.4 and Appendix B.3, both  $\pi^*(w+)$  and  $\pi^*(w-)$  are optimal to (2). Moreover, since problem (2) can be formulated as a linear program (3), randomizing over the two optimal policies  $\pi^*(w+)$  and  $\pi^*(w-)$  is also optimal to (2).

Let  $\Pi^*(w)$  denote the set of optimal policies of (2) when the waiting time of a job in the shared pool is  $w \geq 0$ . From above,  $\Pi^*(w) = \{\pi^*(w)\}$  for any  $w \in \mathbb{R}_+ \setminus \{m(z) : z \geq k\}$  and  $\Pi^*(w) \supseteq \{\alpha \pi^*(w+) \oplus (1 - \alpha) \pi^*(w-) : \alpha \in [0, 1]\}$  for any  $w = m(z)$  with  $z \geq k$ , where  $\alpha \pi_1 \oplus (1 - \alpha) \pi_2$  with probability  $\alpha$  and policies  $\pi_1$  and  $\pi_2$  denotes the randomized policy that implements policy  $\pi_1$  with probability  $\alpha$  and policy  $\pi_2$  with probability  $1 - \alpha$ . We now show that

$$\Pi^*(w) = \left\{ \alpha \pi^*(w+) \oplus (1 - \alpha) \pi^*(w-) : \alpha \in [0, 1] \right\}, \forall w = m(z) \text{ and } z \geq k,$$

i.e., the set of optimal policies is the mixing of the two extreme optimal policies  $\pi^*(w+)$  and  $\pi^*(w-)$ .

To see this, note that we can re-formulate problem (2) as (35),

$$V^F(w) = \min_{r \in [0, \frac{\lambda}{1+\lambda}]} \left\{ \min_{\pi \in \Pi(r)} \frac{\mathbb{E}_\pi[q]}{1+\lambda} + rw + \frac{c\lambda}{1+\lambda} \right\} \quad (35)$$

where  $r \in [0, \frac{\lambda}{1+\lambda}]$  denotes the time-average rate of requesting help,  $\Pi(r)$  the set of policies that are feasible to (2) (i.e., the flow balance constraint of the tokens holds) while maintaining a steady-state requesting-help rate of  $r$ , and  $\mathbb{E}_\pi[q]$  the expected queue length in the stationary distribution of policy  $\pi$ . In (35), the outer problem selects an optimal requesting-help rate  $r$ . When a rate  $r$  is selected, the time-average waiting cost for jobs routed to the shared pool is a fixed constant  $rw$ . In addition, the time-average job processing cost is  $\frac{c\lambda}{1+\lambda}$  for any policy that is feasible to (2). This is because, by the flow balance constraint of the tokens, the number of jobs routed to the shared pool per unit of time (which equals the rate of spending tokens) equals the number of jobs the server serves from the shared pool per unit of time (which equals the rate of earning tokens) in the long-run average. Thus, the only component in the objective that is variable is the holding cost for jobs in the queue, and the inner problem selects an optimal policy  $\pi \in \Pi(r)$  to minimize it.

Let  $R^*(w)$  denote the set of optimal requesting-help rates in (35) for a given value of  $w \geq 0$ , and  $r(\pi)$  the steady-state rate of requesting help under a policy  $\pi$ . Note that since the objective of (35) has increasing differences in  $r$  and  $w$ , by the theory of monotone comparative statics (e.g., Milgrom and Shannon 1994, Topkis 1998, and Sarver 2022), the set  $R^*(w)$  is decreasing in  $w$  in the strong set order (Section 1.2 of Sarver 2022).

Now suppose that  $w = m(z)$  and  $z \geq k$ . Since  $\pi(w-)$  is the unique optimal policy of  $V^F(w')$  for any  $w' \in (m(z-1), m(z))$  and  $\pi(w+)$  the unique optimal policy of  $V^F(w')$  for any  $w' \in (m(z), m(z+1))$ , we have  $R^*(w') = \{r(\pi(w-))\}$  for any  $w' \in (m(z-1), m(z))$  and  $R^*(w') = \{r(\pi(w+))\}$  for any  $w' \in (m(z), m(z+1))$ . As a result,  $R^*(w) \subseteq [r(\pi(w+)), r(\pi(w-))]$  by the property of the strong set order. On the other hand,  $R^*(w) \supseteq [r(\pi(w+)), r(\pi(w-))]$  because any mixing of  $\pi(w-)$  and

$\pi(w+)$  is optimal to  $V^F(w)$ . Therefore,  $R^*(w) = [r(\pi(w+)), r(\pi(w-))]$  and it implies that

$$\Pi^*(w) = \left\{ \alpha \pi^*(w+) \oplus (1 - \alpha) \pi^*(w-) : \alpha \in [0, 1] \right\}.$$

**Remark B.6** (Requesting- and Offering-Help Rates Monotone in  $w$ ). From the above discussion, the steady-state rates of requesting and offering help under a server's best-response policy are decreasing in the shared pool waiting time  $w$  in the strong set order. (Note that the steady-state rate of requesting help is always a fraction  $\phi$  of the steady-state rate of offering help.)

## C Proof for Sections 6 and 7

### C.1 Proof of Lemma 6.1

In the proof, we let  $p_{qs} = \mathbb{P}[\bar{Q}_i(\infty) = q, \bar{S}_i(\infty) = s]$  denote the stationary distribution of the queue-length-token-count pair under policy  $\bar{\pi}^F$  and  $\pi_q = \mathbb{P}[Q_i(\infty) = q]$  the stationary distribution of the queue length under policy  $\pi^F$ . We prove that the stationary distribution  $p_{qs}$  satisfies  $\sum_{q \geq 0} (p_{q0} + p_{qC}) \leq \frac{M_2}{C}$  for some constant  $M_2$  that depends only on the values of  $\lambda$  and  $\phi$ .

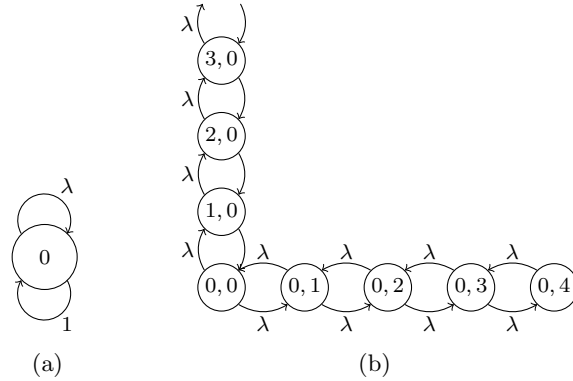
#### C.1.1 Case One: $\phi \geq \lambda$

First, suppose that  $\phi \geq \lambda$ . In this case, the Markov chains of the dynamics under policies  $\pi^F$  and  $\bar{\pi}^F$  are visualized in Figure 4.

When  $\phi \geq \lambda$ , a server that follows policy  $\bar{\pi}^F$  requests help for all incoming jobs as long as it has a token. The stationary distribution satisfies that  $p_{q0} = \lambda^q \cdot p_{00}$  for  $q \geq 1$ ,  $p_{0s} = p_{00}$  for  $s \in [1 : C]$ , and  $p_{qs} = 0$  for all the other states. Thus, from  $\sum_{q,s \geq 0} p_{qs} = 1$  we have  $p_{00} = \frac{1-\lambda}{1+C(1-\lambda)}$ . Therefore,

$$\sum_{q \geq 0} (p_{q0} + p_{qC}) = \frac{2-\lambda}{1-\lambda} \cdot p_{00} \leq \frac{2-\lambda}{1-\lambda} \cdot \frac{1}{C}.$$

The inequality in Lemma 6.1 holds by taking  $M_2 = \frac{2-\lambda}{1-\lambda}$ .

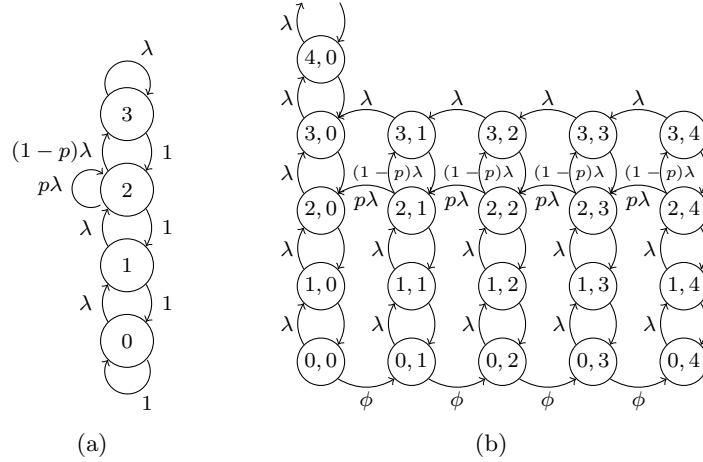


**Figure 4:** Markov chains when  $\phi \geq \lambda$ . (a) Markov chain of the queue length  $q$  under policy  $\pi^F$ . (b) Markov chain of the queue-length-token-count pair  $(q, s)$  under policy  $\bar{\pi}^F$ . The maximum token count is set at  $C = 4$ , and all unspecified transition rates take the value of one.

### C.1.2 Case Two: $\phi < \lambda$

Secondly, suppose that  $\phi < \lambda$ . The Markov chains of the dynamics under policies  $\pi^F$  and  $\bar{\pi}^F$  are visualized in Figure 5.

Let  $k \geq 1$  denote the maximum queue length under policy  $\pi^F$ , which corresponds to when the value of  $\phi$  satisfies  $\phi \in [\lambda^{k+1}, \lambda^k)$ . We first show in Lemma C.1 that the stationary probabilities  $p_{ks}$  for  $s \in [0 : C]$  (which represent the probabilities of states in the first row of Figure 5b) have comparable values.



**Figure 5:** Markov chains when  $\phi < \lambda$ . Specific to this example, the token-earning probability  $\phi$  satisfies  $\phi \in [\lambda^{k+1}, \lambda^k)$  with  $k = 3$ , and  $p = \frac{\phi - \lambda^{k+1}}{\lambda^k - \lambda^{k+1}} \in [0, 1)$ . (a) Markov chain of the queue length  $q$  under policy  $\pi^F$ . (b) Markov chain of the queue-length-token-count pair  $(q, s)$  under policy  $\bar{\pi}^F$ . The maximum token count is set at  $C = 4$ , and all unspecified transition rates take the value of one.

**Lemma C.1.** *The stationary probabilities  $p_{ks}$  are decreasing in  $s$  for  $s \geq 1$ , i.e.,  $p_{k1} \geq p_{k2} \geq \dots \geq p_{kC}$ . Moreover, the minimum probability  $p_{kC}$  satisfies  $p_{kC} \geq c \cdot p_{k0}$  for some constant  $c > 0$  that depends only on the values of  $\lambda$  and  $\phi$ .*

We prove Lemma C.1 in Appendix C.1.3. Lemma C.1 directly implies Lemma 6.1. To see this, first note that since  $\sum_{s=0}^C p_{ks} \leq 1$ , we have  $p_{k0} \leq \frac{1}{C} \cdot \frac{1}{C}$  by Lemma C.1.

Secondly, we have  $p_{q0} \leq c_1(\lambda, \phi) \cdot p_{k0}$  for all  $q \leq k$  and some constant  $c_1(\lambda, \phi)$  that depends only on the values of  $\lambda$  and  $\phi$ . This is because  $p_{q0} \leq \frac{1+\lambda}{\lambda} p_{q+1,0} \leq (\frac{1+\lambda}{\lambda})^{k-q} p_{k,0}$  for any  $q \leq k-1$ , where the first inequality follows from the flow balance equation at state  $(q+1, 0)$  and the second inequality from recursion.

Analogously, we have  $p_{qC} \leq c_2(\lambda, \phi) \cdot p_{k0}$  for all  $q \leq k$  and some constant  $c_2(\lambda, \phi)$  that depends only on the values of  $\lambda$  and  $\phi$ . To see this, first note that  $p_{kC} \leq p_{k1} \leq \frac{1+\lambda}{\lambda} p_{k0}$ , where the first inequality follows from Lemma C.1 and the second inequality from the flow balance equation at state  $(k, 0)$ . Secondly, by the flow balance equations at states  $(q, C)$  with  $q \leq k$ , we have  $p_{k-1,C} = \frac{1+\lambda}{(1-p)\lambda} p_{kC}$  and  $p_{qC} \leq \frac{1+\lambda}{\lambda} p_{q+1,C}$  for  $q \leq k-2$ .

From the above, we have

$$\begin{aligned} \sum_{q \geq 0} (p_{q0} + p_{q,C}) &= \sum_{q=0}^k (p_{q0} + p_{q,C}) + p_{k0} \cdot \frac{\lambda}{1+\lambda} \\ &\leq p_{k0} \cdot \left( \frac{\lambda}{1+\lambda} + k \cdot (c_1(\lambda, \phi) + c_2(\lambda, \phi)) \right) \end{aligned}$$

$$\leq \frac{1}{c} \cdot \left( \frac{\lambda}{1+\lambda} + k \cdot (c_1(\lambda, \phi) + c_2(\lambda, \phi)) \right) \cdot \frac{1}{C}$$

where the equality follows from the fact that  $p_{k+i,0} = \lambda^i \cdot p_{k,0}$  for all  $i \geq 1$ . The inequality in Lemma 6.1 holds by taking  $M_2 = \frac{1}{c} \cdot \left( \frac{\lambda}{1+\lambda} + k \cdot (c_1(\lambda, \phi) + c_2(\lambda, \phi)) \right)$ .

### C.1.3 Proof of Lemma C.1

The proof involves tedious algebraic computations. We express all of the stationary probabilities  $p_{qs}$  in terms of the probability  $p_{k0}$ . We first consider the case that  $\phi \in (\lambda^{k+1}, \lambda^k)$  with  $k \geq 3$ . A visualization of the Markov chain is in Figure 5b.

**Step One** For any number of tokens  $s$ , the flow balance equation at state  $(q, s)$  with  $q \in [1 : k-2]$  is

$$p_{q,s} \cdot (1 + \lambda) = \lambda \cdot p_{q-1,s} + p_{q+1,s}, \forall q \in [1 : k-2]. \quad (36)$$

Therefore, we have

$$p_{q+1,s} - p_{q,s} = \lambda(p_{q,s} - p_{q-1,s}) = \lambda^q(p_{1,s} - p_{0,s}), \forall q \in [1 : k-2]. \quad (37)$$

Additionally, the general solution to the sequence in (36) is

$$p_{q,s} = a_s \lambda^q + b_s, \forall q \in [0 : k-1], \quad (38)$$

where  $a_s$  and  $b_s$  are two constants (whose values are to be determined), because  $\lambda$  and 1 are the roots of the quadratic equation  $x^2 - (1 + \lambda)x + \lambda = 0$ .

**Step Two** We next express the probabilities  $p_{k1}$ ,  $p_{k-1,1}$  and  $p_{00}$  in terms of  $p_{k0}$  and  $p_{k-1,0}$ . In particular, we have the following equations.

$$a_0 + b_0 = p_{00}, \quad (39)$$

$$a_0 \lambda + b_0 = p_{10} = p_{00} \cdot (\lambda + \phi), \quad (40)$$

$$p_{k-1,0} = a_0 \lambda^{k-1} + b_0, \quad (41)$$

$$p_{k0} = \lambda \cdot (p_{k-1,0} + p_{k1}), \quad (42)$$

$$p_{k0} + p \lambda \cdot p_{k-1,1} = p_{k-1,0} + \lambda \cdot (p_{k-1,0} - p_{k-2,0}) = p_{k-1,0} + \lambda^{k-1} \cdot (\lambda + \phi - 1) \cdot p_{00}, \quad (43)$$

where (39) - (41) follow from (38) for states  $(0,0)$ ,  $(1,0)$ , and  $(k-1,0)$ , respectively. (42) and (43) are the flow balance equations at state  $(k,0)$  and  $(k-1,0)$ , respectively. (Note that  $p_{k+i,0} = \lambda^i \cdot p_{k,0}$  for all  $i \geq 1$ .) Additionally, the second equality in (40) also uses the flow balance equation at state  $(0,0)$ , and the second equality in (43) follows from (37), (39) and (40).

We can express the constants  $a_0$  and  $b_0$  in terms of  $p_{00}$  using (39) and (40). Then, we can express  $p_{00}$  in terms of  $p_{k-1,0}$  using (41). Lastly, we can express  $p_{k1}$  in terms of  $p_{k0}$  and  $p_{k-1,0}$  using (42) and  $p_{k-1,1}$  in terms of  $p_{k0}$  and  $p_{k-1,0}$  using (43). In the end, we have

$$\begin{pmatrix} p_{k1} \\ p_{k-1,1} \\ p_{00} \end{pmatrix} = \begin{pmatrix} \frac{1}{\lambda} & -1 \\ -\frac{1}{p\lambda} & \frac{1}{p\lambda} (1 + \lambda^{k-1}(\lambda + \phi - 1)t) \\ 0 & t \end{pmatrix} = \begin{pmatrix} p_{k0} \\ p_{k-1,0} \end{pmatrix} \quad (44)$$



with  $t = \frac{1-\lambda}{\lambda^{k-1}(1-\lambda-\phi)+\phi}$ .

**Step Three** We next recursively express the probabilities  $p_{k,i+1}$ ,  $p_{k-1,i+1}$  and  $p_{0i}$  in terms of  $p_{ki}$ ,  $p_{k-1,i}$  and  $p_{0,i-1}$  for  $i \geq 1$ . Particularly, we have the following equations.

$$p_{k-1,i} = a_i \lambda^{k-1} + b_i, \quad (45)$$

$$(a_i + b_i)(\lambda + \phi) = p_{0,i-1} \cdot \phi + (a_i \lambda + b_i), \quad (46)$$

$$p_{0i} = a_i + b_i, \quad (47)$$

$$p_{ki} \cdot (1 + \lambda) = p_{k,i+1} \cdot \lambda + p_{k-1,i} \cdot (1 - p)\lambda, \quad (48)$$

$$p_{ki} + p\lambda \cdot p_{k-1,i+1} = p_{k-1,i} + \lambda \cdot (p_{k-1,i} - p_{k-2,i}) = p_{k-1,i} + \lambda^{k-1} \cdot (\lambda - 1) \cdot a_i, \quad (49)$$

where (45) follows from (38) for state  $(k-1, i)$ , (46) from (38) for state  $(1, i)$  and the flow balance equation at state  $(0, i)$ , (47) from (38) for state  $(0, i)$ , (48) from the flow balance equation at state  $(k, i)$ , and (49) from the flow balance equation at state  $(k-1, i)$ , (37), and the fact that  $p_{1i} - p_{0i} = a_i \cdot (\lambda - 1)$  by (38).

We can express the constants  $a_i$  and  $b_i$  in terms of  $p_{0,i-1}$  and  $p_{k-1,i}$  using (45) and (46). Then, we can express  $p_{0i}$  in terms of  $p_{0,i-1}$  and  $p_{k-1,i}$  using (47). Lastly, we can express  $p_{k,i+1}$  in terms of  $p_{ki}$ ,  $p_{k-1,i}$  and  $p_{0,i-1}$  using (48) and  $p_{k-1,i+1}$  in terms of  $p_{ki}$ ,  $p_{k-1,i}$  and  $p_{0,i-1}$  using (49). In the end, we have

$$\begin{pmatrix} p_{k,i+1} \\ p_{k-1,i+1} \\ p_{0i} \end{pmatrix} = \begin{pmatrix} \frac{1+\lambda}{\lambda} & \frac{p-1}{p\lambda} (1 + \lambda^{k-1}(\lambda + \phi - 1)t) & 0 \\ -\frac{1}{p\lambda} & \frac{\lambda(1-\lambda)}{\lambda\phi + \lambda^k(1-\phi) - \lambda^{k+1}} & -\frac{1}{p\lambda} \lambda^{k-1} \phi t \\ 0 & \frac{(\lambda - \lambda^k)\phi}{\lambda\phi + \lambda^k(1-\phi) - \lambda^{k+1}} \end{pmatrix} = \begin{pmatrix} p_{ki} \\ p_{k-1,i} \\ p_{0,i-1} \end{pmatrix}, \forall i \geq 1, \quad (50)$$

again with  $t = \frac{1-\lambda}{\lambda^{k-1}(1-\lambda-\phi)+\phi}$ .

**Step Four** From (44) and (50), we can express  $p_{kC}$  and  $p_{k-1,C}$  in terms of  $p_{k0}$  and  $p_{k-1,0}$ . Then, using the flow balance equation at state  $(k, C)$ , which is

$$p_{kC} \cdot (1 + \lambda) = (1 - p)\lambda \cdot p_{k-1,C},$$

we can express  $p_{k-1,0}$  in terms of  $p_{k0}$ . With this, again using (44) and (50), we can express  $p_{ki}$  for all  $i \geq 1$  in terms of  $p_{k0}$ . This gives the following expressions: for  $i \in [1 : C]$ , we have

$$p_{ki} = p_{k0} \cdot \frac{(\lambda^k - \phi)\phi \cdot (r^C - r^{i-1})}{\lambda^k(1 - \phi)\phi \cdot r^C - \lambda(\lambda^k - \phi)(\phi - \lambda^{k+1})} \quad (51)$$

where

$$r = \frac{\phi(\lambda^{2k+2} + \lambda\phi + \lambda^k(1 - \lambda - \lambda^2 - \phi))}{\lambda(\phi - \lambda^{k+1})(\lambda\phi + \lambda^k(1 - \lambda - \phi))} \geq 1 + \frac{1}{\lambda} \geq 2.$$

Therefore, the value of  $p_{ki}$  is decreasing in  $i$ . Additionally,

$$p_{kC} \geq p_{k0} \cdot \frac{(\lambda^k - \phi)}{\lambda^k(1 - \phi)} \cdot \left(1 - \frac{1}{r}\right).$$

**Other Cases** Following the same approach, we can verify that (51) also holds for the case of  $\phi \in (\lambda^{k+1}, \lambda^k)$  with  $k = 1$  or  $2$ . Finally, for the special case of  $\phi = \lambda^{k+1}$  with some integer  $k \geq 1$ ,

we can follow the same approach and show that  $p_{ki}$  is constant across  $i \in [1 : C]$  with

$$p_{ki} = p_{k0} \cdot \frac{1 - \lambda}{1 - \lambda^{k+1}}, \forall i \in [1 : C].$$

## C.2 Proof of Lemma 6.2

For ease of notation, let  $p_{qs} = \mathbb{P}[\bar{Q}_i(\infty) = q, \bar{S}_i(\infty) = s]$  denote the stationary distribution of the queue-length-token-count pair under policy  $\bar{\pi}^F$  and  $\pi_q = \mathbb{P}[Q_i(\infty) = q]$  the stationary distribution of the queue length under policy  $\pi^F$ .

### C.2.1 Case One: $\phi \geq \lambda$

First, suppose that  $\phi \geq \lambda$ . In this case, the Markov chains of the dynamics under policies  $\pi^F$  and  $\bar{\pi}^F$  are visualized in Figure 4.

When  $\phi \geq \lambda$ , the server requests help for all incoming jobs under policy  $\pi^F$ . Therefore, we have  $\pi_0 = 0$ ,  $\mathbb{E}[Q_i(\infty)] = 0$ ,  $\mathbb{P}[X_i(\infty) = 1] = \lambda$ , and  $\mathbb{P}[Y_i(\infty) = 1] = \frac{\lambda}{\phi}$ .

On the other hand, a server that follows policy  $\bar{\pi}^F$  requests help for all incoming jobs as long as it has a token. The stationary distribution satisfies that  $p_{q0} = \lambda^q \cdot p_{00}$  for  $q \geq 1$ ,  $p_{0s} = p_{00}$  for  $s \in [1 : C]$ , and  $p_{qs} = 0$  for all the other states. Thus, from  $\sum_{q,s \geq 0} p_{qs} = 1$  we have  $p_{00} = \frac{1-\lambda}{1+C(1-\lambda)}$ . Therefore,

$$\begin{aligned} \mathbb{E}[\bar{Q}_i(\infty)] &= \sum_{q \geq 0} q \cdot p_{q0} = p_{00} \cdot \frac{\lambda}{(1-\lambda)^2} = \frac{\lambda}{1-\lambda} \cdot \frac{1}{1+C(1-\lambda)}, \\ \mathbb{P}[\bar{X}_i(\infty) = 1] &= \lambda \cdot \sum_{s=1}^C p_{0s} = \lambda \cdot \frac{C(1-\lambda)}{1+C(1-\lambda)}, \\ \mathbb{P}[\bar{Y}_i(\infty) = 1] &= \frac{\lambda}{\phi} \cdot \sum_{s=0}^{C-1} p_{0s} = \frac{\lambda}{\phi} \cdot \frac{C(1-\lambda)}{1+C(1-\lambda)}. \end{aligned}$$

Thus, Lemma 6.2 holds with proper choices of the constants  $M_3$ ,  $M_4$ , and  $M_5$ .

### C.2.2 Case Two: $\phi < \lambda$

Next, suppose that  $\phi < \lambda$ . The Markov chains of the dynamics under policies  $\pi^F$  and  $\bar{\pi}^F$  are visualized in Figure 5.

Let  $h_q \triangleq \sum_{s=0}^C p_{qs}$  denote the marginal probability of having  $q$  jobs under policy  $\bar{\pi}^F$ , for all  $q \leq k$ . By the flow balance equation for the set of states  $\{(q, s) : s \leq C\}$  for any  $q \leq k$ , we have

$$\begin{aligned} h_0 \lambda &= h_1, \\ h_1 \lambda &= h_2, \\ &\dots \\ h_{k-2} \lambda &= h_{k-1}, \\ h_k &\geq h_{k-1} \cdot (1-p) \lambda \geq (h_{k-1} - p_{k-1,0}) \cdot (1-p) \lambda \geq h_k - p_{k0}. \end{aligned} \tag{52}$$

In the last row, the first inequality follows from the fact that  $p_{k+i,0} = \lambda^i \cdot p_{k,0}$  for all  $i \geq 1$ , and the third inequality from the flow balance equation for the set of states  $\{(k, s) : 1 \leq s \leq C\}$ .

On the other hand, the dynamics of jobs under policy  $\pi^F$  follow a birth-and-death chain process. Therefore, the stationary distribution satisfies

$$\begin{aligned}
\pi_0 \lambda &= \pi_1, \\
\pi_1 \lambda &= \pi_2, \\
&\dots \\
\pi_{k-2} \lambda &= \pi_{k-1}, \\
\pi_k &= \pi_{k-1} \cdot (1-p)\lambda.
\end{aligned} \tag{53}$$

From (52), (53), and the fact that  $\sum_{q=1}^k \pi_q = 1$  and  $\sum_{q=1}^k h_q \leq 1$ , we have

$$1 \geq \frac{h_0}{\pi_0} = \frac{h_1}{\pi_1} = \dots = \frac{h_{k-1}}{\pi_{k-1}} = \frac{h_{k-1} \cdot (1-p)\lambda}{\pi_k} \geq \frac{h_k - p_{k0}}{\pi_k}. \tag{54}$$

**Proof of Bullet One** First, note that

$$\begin{aligned}
\mathbb{E}[\bar{Q}_i(\infty)] &= \sum_{q=1}^k h_q \cdot k + \sum_{i=1}^{\infty} p_{k+i,0} \cdot (k+i) \\
&= \sum_{q=1}^{k-1} h_q \cdot k + (h_k - p_{k0}) \cdot k + \sum_{i=0}^{\infty} p_{k+i,0} \cdot (k+i) \\
&= \sum_{q=1}^{k-1} h_q \cdot k + (h_k - p_{k0}) \cdot k + p_{k,0} \cdot \frac{k + \lambda - k\lambda}{(1-\lambda)^2} \\
&\leq \mathbb{E}[Q_i(\infty)] + \frac{M_2}{C} \cdot \frac{k + \lambda - k\lambda}{(1-\lambda)^2},
\end{aligned}$$

where the third equality follows from the fact that  $p_{k+i,0} = p_{k,0} \cdot \lambda^i$  for all  $i \geq 0$ , and the inequality follows from (54) and the facts that  $\mathbb{E}[Q_i(\infty)] = \sum_{q=1}^k \pi_q \cdot q$  and that  $p_{k0} \leq \frac{M_2}{C}$  by Lemma 6.1.

Secondly, we have  $\mathbb{E}[Q_i(\infty)] \leq \mathbb{E}[\bar{Q}_i(\infty)]$ . This is because the policies  $\pi^F$  and  $\bar{\pi}^F$  both satisfy the flow balance constraint of the tokens (i.e., the expected rates of earning and spending tokens are equal) in the fluid mean-field problem. On the other hand, the policy  $\pi^F$  minimizes the expected queue length among all policies that are feasible to the fluid mean-field problem (i.e., the flow balance constraint of the tokens holds) by Lemma C.2.

**Lemma C.2.** *The policy  $\pi^F$  minimizes the expected queue length in the stationary distribution among all policies that are feasible to the fluid mean-field problem.*

*Proof.* First, all policies adhering to the flow balance constraint of the tokens have the same time-average job processing cost, which equals  $c \cdot \lambda$ . To see this, note that the rate at which jobs from the shared pool are served equals the rate of earning tokens (because both equal  $\phi$  times the rate of offering help to the shared pool). On the other hand, the rate of requesting help equals the rate of spending tokens. Consequently, by the flow balance of tokens, the rate of jobs being served from the shared pool equals the rate of routing jobs to the shared pool. This implies that the rate at which jobs are served equals the job arrival rate  $\lambda$ .

Second, by Corollary 4.3, the policy  $\pi^F$  minimizes the time-average total cost among all policies that satisfy the flow balance constraint of the tokens when the shared pool waiting time  $w \leq 1$ , particularly when  $w = 0$ . In the case of  $w = 0$ , the time-average total cost equals the time-average

job processing cost plus the time-average holding cost for jobs in the queue (which equals the expected queue length). Therefore, the policy  $\pi^F$  has the lowest expected queue length among all policies that satisfy the flow balance constraint of the tokens (because these policies shared the same time-average job processing cost, which equals  $c\lambda$ ).  $\square$

**Proof of Bullet Two** Because stationary probabilities sum up to one, we have

$$\sum_{q=0}^k h_q + \sum_{i=1}^{\infty} p_{k+i,0} = \sum_{q=0}^k h_q + p_{k,0} \cdot \frac{\lambda}{1-\lambda} = 1, \quad (55)$$

where the first equality follows from the fact that  $p_{k+i,0} = p_{k,0} \cdot \lambda^i$  for all  $i \geq 0$ . Therefore, we have

$$\sum_{q=0}^{k-1} h_q + (1-p)\lambda \cdot h_{k-1} \geq \sum_{q=0}^k h_q - p_{k,0} = 1 - \frac{p_{k,0}}{1-\lambda} \geq 1 - \frac{M_2}{(1-\lambda)C}$$

where the first inequality follows from  $h_{k-1} \cdot (1-p)\lambda \geq h_k - p_{k,0}$  by (52), the equality from (55), and the second inequality from  $p_{k,0} \leq \frac{M_2}{C}$  by Lemma 6.1. Therefore, from (54) we have

$$\frac{h_0}{\pi_0} = \frac{h_1}{\pi_1} = \dots = \frac{h_{k-1}}{\pi_{k-1}} = \frac{h_{k-1} \cdot (1-p)\lambda}{\pi_k} \geq r(C) \triangleq 1 - \frac{M_2}{(1-\lambda)C}. \quad (56)$$

The stationary rates of requesting help in the original and fluid problems can be expressed as  $\mathbb{P}[X_i(\infty) = 1] = \lambda\pi_k + p\lambda\pi_{k-1}$  and  $\mathbb{P}[\bar{X}_i(\infty) = 1] = \lambda(h_k - p_{k,0}) + p\lambda(h_{k-1} - p_{k-1,0})$ . We first show that  $\mathbb{P}[\bar{X}_i(\infty) = 1] \leq \mathbb{P}[X_i(\infty) = 1]$ . This follows because

$$\mathbb{P}[\bar{X}_i(\infty) = 1] \leq \lambda \cdot h_{k-1} \cdot (1-p)\lambda + p\lambda \cdot h_{k-1} \leq \lambda\pi_k + p\lambda\pi_{k-1} = \mathbb{P}[X_i(\infty) = 1],$$

where the first inequality follows from  $h_k - p_{k,0} \leq h_{k-1} \cdot (1-p)\lambda$  by (52) and the second inequality from (54).

On the other hand, from (56) and the fact that  $h_k \geq h_{k-1} \cdot (1-p)\lambda$  by (52), we have

$$\mathbb{P}[\bar{X}_i(\infty) = 1] \geq \mathbb{P}[X_i(\infty) = 1] \cdot r(C) - \lambda \cdot p_{k,0} - p\lambda \cdot p_{k-1,0}.$$

Thus,  $\mathbb{P}[X_i(\infty) = 1] \leq \mathbb{P}[\bar{X}_i(\infty) = 1] + \frac{M_4}{C}$  for some constant  $M_4$  by Lemma 6.1 and the expression of  $r(C)$ .

**Proof of Bullet Three** Since the policies  $\pi^F$  and  $\bar{\pi}^F$  are both feasible to the fluid mean-field problem, we have  $\mathbb{P}[\bar{X}_i(\infty) = 1] = \phi \cdot \mathbb{P}[\bar{Y}_i(\infty) = 1]$  and  $\mathbb{P}[X_i(\infty) = 1] = \phi \cdot \mathbb{P}[Y_i(\infty) = 1]$ . Therefore, Bullet Three follows directly from Bullet Two.

### C.3 Proof of Proposition 6.3

The proof is analogous to the proof of Proposition 4.4 (Appendix A.3) with small modifications.

#### C.3.1 Uniform Bound on the Expected Shared Pool Queue Length $\mathbb{E}[\bar{Q}_i(\infty)]$

We show that the expected shared pool queue length is uniformly bounded from above by a constant that depends only on the values of  $\lambda$  and  $\phi$  for any number of servers  $N$  and token-amount upper

bound value  $C$ . Following the proof of Proposition 4.4, it suffices to show that Lemmas A.1 and A.3 remain valid for the tail probability of shared pool queue length  $\bar{Q}_i(\infty)$  in the original problem.

**Validity of Lemma A.1** To verify that Lemma A.1 also holds for  $\bar{Q}_i(\infty)$ , we consider the same auxiliary system as in Appendix A.3.6. Specifically, the auxiliary system mimics the queueing process in the original problem with the only difference that, in auxiliary system, when a server  $i$  offers help, the provided capacity unit can only serve a job in the shared pool that came from server  $i$ . We again let  $Z_i(t)$  denote the number of jobs of server  $i$  in the auxiliary system in period  $t$ . To verify Lemma A.1, it suffices to confirm that Lemma A.4 remains valid for any token-amount upper bound value  $C$ , which we do now.

Denote by  $S_i(t)$  the number of tokens owned by server  $i$  and  $Q_{0i}(t)$  the number of jobs in the shared pool that come from server  $i$ , in the auxiliary system at time  $t$ . Assume, without loss of generality, that the shared pool is initially empty at time zero; hence,  $Q_{0i}(0) = 0$ . Lemma C.3 provides an important observation: for each server, the number of tokens plus the number of jobs in the shared pool cannot surpass the token-amount upper bound  $C$ ; that is,  $S_i(t) + Q_{0i}(t) \leq C$  at any time  $t \geq 0$ .

**Lemma C.3.** *We have  $S_i(t) + Q_{0i}(t) \leq C$  at any time  $t \geq 0$ .*

*Proof.* We examine the embedded discrete-time model and prove the inequality by induction. First, the inequality holds at  $t = 0$  because  $S_i(0) \leq C$  and  $Q_{0i}(0) = 0$ . Now, suppose that the inequality holds in period  $t - 1$ ; we show that it also holds in period  $t$ . Note that  $S_i(t)$  or  $Q_{0i}(t)$  can change value only when the server requests or offers help. First, suppose a job arrives, and the server requests help in period  $t$ . In this case,  $S_i(t)$  decreases by one and  $Q_{0i}(t)$  increases by one. Thus, the summation does not change, and the inequality still holds.

Secondly, suppose a capacity unit arrives, and the server offers help in period  $t$ . If  $Q_{0i}(t-1) \geq 1$ , then  $Q_{0i}(t)$  decreases by one and  $S_i(t)$  increases by at most one; thus, the inequality still holds. Otherwise, if  $Q_{0i}(t-1) = 0$ , then the inequality also holds in period  $t$  because  $S_i(t) + Q_{0i}(t) = S_i(t) \leq C$ .  $\square$

According to Lemma C.3, if the server has a job in the shared pool (i.e.,  $Q_{0i} \geq 1$ ), then  $S_i \leq C - 1$ , i.e., the token amount has not reached its upper bound.

First, suppose that  $\phi \leq \lambda$ . In this case, the server in the auxiliary system serves a job with certainty when it has an affiliated job (i.e.,  $Z_i \geq 1$ ) and a capacity unit arrives. Consequently, the dynamics of  $Z_i(t)$  is an  $M/M/1$  queue with a job arrival rate of  $\lambda$  and a job processing rate of one, regardless of the value of  $C$ .

Secondly, suppose that  $\phi > \lambda$ . In this case, when a capacity unit arrives, the server serves a job from its queue if one is present; otherwise, if there is a job in the shared pool (i.e.,  $Q_{0i} \geq 1$ ), the server serves a job from the shared pool with probability  $\lambda/\phi$ . Hence, the serving rate is at least  $\lambda/\phi$  as long as  $Z_i(t) \geq 1$ . Consequently, the dynamics of  $Z_i(t)$  is first-order stochastically dominated by an  $M/M/1$  queue with a job arrival rate of  $\lambda$  and a job processing rate of  $\lambda/\phi$  regardless of the value of  $C$ .

Thus, Lemma A.4 is still valid, and Lemma A.1 continue to hold for the original problem when servers follow the FMFE policy  $\bar{\pi}^F$ .

**Validity of Lemma A.3** Let  $g(C) > 0$  denote the steady-state rate of offering help when the server follows the FMFE policy  $\bar{\pi}^F$  and the token-amount upper bound is  $C$ , and  $\underline{g} = \min_{C \geq 1} g(C) > 0$  the lower bound on the steady-state rate of offering help. Note that  $\underline{g} > 0$  because  $g(C) > 0$  for

any value of  $C$  and  $\lim_{C \rightarrow \infty} g(C) = \mathbb{P}[Y_i(\infty) = 1] > 0$  by Lemma 6.2, where  $\mathbb{P}[Y_i(\infty) = 1]$  is the steady-state offering help rate under the FMFE policy  $\pi^F$  in the fluid mean-field problem.

We define the high-probability event  $\mathcal{E}$  in the original problem analogous to the one for the fluid mean-field problem in Appendix A.3.2. Specifically, let  $\pi_{qs}$  denote the stationary probability that a server has  $q$  jobs in the queue and  $s$  tokens at hand. Additionally, let  $A_{qs}$  and  $S_{qs}$  denote the probabilities that a server requests and offers help, respectively, when it has  $q$  jobs and  $s$  tokens. Since the steady-state rate of requesting help is  $\phi$  times the steady-state rate of offering help (this is because the expected rates of earning and spending tokens are equal under the policy  $\bar{\pi}^F$ ) and the steady-state offering help rate is at least  $\underline{g}$ , we have

$$\sum_{qs} \pi_{qs} A_{qs} - \sum_{qs} \pi_{qs} S_{qs} \leq -(1 - \phi) \underline{g}.$$

Denote by  $\hat{\pi}$  the empirical distribution of queue lengths and token amounts of the servers, i.e.,

$$\hat{\pi}_{qs}(t) \triangleq \frac{1}{N} \sum_{i=1}^N \mathbb{1}\{Q_i(t) = q, S_i(t) = s\}.$$

We define the high-probability set  $\mathcal{E}$  as follows.

$$\mathcal{E} \triangleq \left\{ \hat{\pi} : \sum_q \hat{\pi}_{qs} A_{qs} - \sum_q \hat{\pi}_{qs} S_{qs} \leq -\frac{1}{2}(1 - \phi) \underline{g} \right\}.$$

Following the same proof, Lemma A.2 is still valid. This implies that Lemma A.3 continues to hold for the original problem with the new high-probability event  $\mathcal{E}$ .

### C.3.2 Uniform Bound on the Long-Run Average Waiting Time $w$

The long-run average waiting time in the shared pool diminishes at a rate of  $O(\frac{1}{N})$ , which is uniform across the token-amount upper bound value  $C$ , following the same proof in Appendix A.3.5. This is because of Little's law and the fact that the shared pool queue length is uniformly bounded by a constant, for any number of servers  $N$  and token-amount upper bound value  $C$ .

## C.4 Proof of Lemma 6.4

Suppose all servers follow the FMFE policy  $\bar{\pi}^F$ . The time-average total cost of server one comprises three parts: (i) the job processing cost, (ii) the holding cost for jobs in its queue, and (iii) the waiting cost for jobs routed to the shared pool.

First, the time-average job processing cost is  $c\lambda$  by symmetry because servers are stochastically identical and follow the same strategy. Secondly, the time-average holding cost for jobs in the queue, which equals  $\mathbb{E}[\bar{Q}_i(\infty)]$ , is at most  $\mathbb{E}[Q^F] + \frac{M_3}{C}$  by Lemma 6.2. Finally, let  $w$  denote the long-run average waiting time in the shared pool. The time-average waiting cost for jobs in the shared pool that come from server one is at most  $\lambda w$ , which is no larger than  $\frac{\lambda M_7}{N}$  by Proposition 6.3.

## C.5 Proof of Lemma 6.5

### C.5.1 Step One: A Relaxation of Server One's Problem

The problem faced by server one is a complex, partially observable Markov decision problem because server one can only infer partial information about the shared pool through its interaction with the

shared pool. We first derive a relaxation of server one's problem. This relaxation provides a lower bound for the original problem faced by server one. Moreover, it has a close connection to the fluid mean-field problem (2) and thus is much easier to analyze.

As a first step to establishing the relaxation, we show in Lemma C.4 that, in the original problem of server one, although server one has the flexibility to request help at any time, it is without loss of optimality to assume that when a capacity unit arrives, the server does not request help until it has taken action regarding the arriving capacity unit.

**Lemma C.4.** *It is without loss of optimality to assume that in the original problem, when a capacity unit arrives, the server does not request help before it has taken action regarding the capacity unit.*

*Proof.* Suppose a capacity unit arrives at server one at time  $t$ . If server one immediately requests help, it does not collect additional information about the current state of the shared pool. Therefore, it does not enhance the decision regarding the capacity unit, which must be done now. On the contrary, if server one takes action regarding the capacity unit first, it can possibly gain immediate information about the shared pool (for example, if server one offers help without incurring a job processing cost, it indicates that the shared pool is currently empty). This additional information can help with the decision regarding requesting help. Hence, it is without loss of optimality to assume that when a capacity unit arrives, server one takes action regarding the said capacity unit first and then considers the option of requesting help (possibly after gathering more information about the shared pool).  $\square$

We now introduce a relaxation to server one's problem. The relaxation endows server one with an additional power to empty the shared pool at the end of every interaction (i.e., requesting or offering help) with the shared pool. Specifically, when server one offers help, it serves a job from the shared pool if one is present; after that, all remaining jobs in the shared pool are cleared. In addition, when server one requests help, all jobs in the shared pool, including the job the server has just routed to the shared pool, are immediately cleared; hence, the relocated job incurs no waiting cost. Finally, we impose that, when a capacity unit arrives, server one can only request help once it has taken action on the capacity unit; from Lemma C.4, this is without loss of optimality in the original problem. As a result, the shared pool is cleared whenever server one requests or offers help in the relaxation. As we show Lemma C.5, it will be futile for server one to try to acquire additional information by strategically altering the timing of requesting help. Consequently, server one requests help only when a job arrives and only requests help for the incoming job.

**Lemma C.5.** *In the relaxation, it is optimal for server one to request help only when a job arrives, and only request help for the incoming job.*

*Proof.* Note that in the relaxation, server one's assessment of the state of the shared pool at the next arrival (of either a job or capacity unit) is independent of when server one requests help. This is because the shared pool is emptied whenever the server requests help. Additionally, conditioning on server one requesting help at a given point, the time till the next arrival follows an exponential distribution with a rate parameter of  $1 + \lambda$  due to the memoryless property of the exponential distribution. Hence, server one cannot acquire more information about the shared pool by strategically timing requesting help. As a result, if server one would like to request help for a job, it is optimal to do so when it arrives to eliminate unnecessary job holding costs.  $\square$

## C.5.2 Step Two: Bounding the Shared Pool Non-Emptiness Probability

Because servers two to  $N$  follow an oblivious strategy  $\bar{\pi}^F$ , the dynamics of these  $N - 1$  servers are independent. Without loss of generality, we assume that servers two to  $N$  are in the stationary

distribution at time zero; consequently, they are in the stationary distribution at any time  $t \geq 0$ .

We now consider the scenario that a capacity unit arrives at server one at time  $\tau$ . When server one takes action regarding the capacity unit, it needs to assess the probability that the shared pool is non-empty, i.e.,  $Q_0(\tau) \geq 1$ , which corresponds to experiencing a job processing cost of  $c$  if server one offers help to the shared pool.

Let  $h(\tau) < \tau$  denote the time of the last interaction between server one and the shared pool – i.e., either requesting or offering help. The server's assessment of the probability that  $Q_0(\tau) \geq 1$  depends only on the time difference  $\tau - h(\tau)$ . This is because the shared pool is emptied at time  $h(\tau)$ , and after that, the shared pool interacts only with servers two to  $N$ .

We now bound the probability that the shared pool is non-empty conditioning on the length of the time difference  $\tau - h(\tau)$ . In particular, as we show in Lemma C.6, as long as the time difference is not very tiny, the probability that the shared pool is non-empty is close to the value of  $\phi$ .

**Lemma C.6.** *For any constants  $\Delta \geq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$  and  $\delta \in (0, 1)$ , we have*

$$\mathbb{P}[Q_0(\tau) \geq 1 | \tau - h(\tau) = \Delta] \geq \phi - \frac{c(\lambda, \phi, \delta)}{N^{1-\delta}}$$

where  $c(\lambda, \phi, \delta)$  is a constant that depends only on the values of  $\lambda$ ,  $\phi$  and  $\delta$ .

We prove Lemma C.6 in Appendix C.5.4 using a coupling method. Lastly, let  $\bar{h}(\tau) < \tau$  denote the time of the most recent arrival (of either a job or capacity unit) before time  $\tau$ . Clearly,

$$\tau - \bar{h}(\tau) \leq \tau - h(\tau) \tag{57}$$

because server one takes actions only when a job or capacity unit arrives (Lemma C.5).

The time difference  $\tau - \bar{h}(\tau)$  between two adjacent arrivals follows an exponential distribution with a rate parameter of  $1 + \lambda$ . Hence,

$$\mathbb{P}\left[\tau - h(\tau) \leq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}\right] \leq \mathbb{P}\left[\tau - \bar{h}(\tau) \leq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}\right] \leq \frac{1}{N^{1-\delta}} \tag{58}$$

where the first inequality follows from (57) and the second inequality from the cumulative distribution function of the exponential distribution and the fact that  $1 - \exp(-x) \leq x$  for any  $x \in \mathbb{R}$ .

### C.5.3 Step Three: Bounding the Optimal Value of the Relaxation

To bound the optimal value of the relaxation of server one's problem, we consider an alternative system with the only difference that, when server one offers help, it experiences a job processing cost of  $c$  with a constant probability  $\phi$ . However, every time a capacity unit arrives at a time  $\tau$ , server one is compensated based on the length of the inter-arrival time  $\Delta(\tau) \triangleq \tau - \bar{h}(\tau)$ , where  $\bar{h}(\tau)$  is the time of the most recent arrival (of either a job or capacity unit) before time  $\tau$ . Specifically, the server is compensated with  $c \cdot \phi$  if  $\Delta(\tau) \leq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$  and with  $c \cdot \frac{c(\lambda, \phi, \delta)}{N^{1-\delta}}$  otherwise, where  $c(\lambda, \phi, \delta)$  is the constant specified in Lemma C.6.

Lemma C.7 shows that server one has a lower expected time-average cost in the alternative system than in the relaxation. Thus, it suffices to bound the expected cost in the alternative system from below. First, we remark that the inter-arrival time  $\Delta(\tau)$  is beyond the control of server one. Moreover, since server one experiences a cost of  $c$  with a constant probability  $\phi$  when offering help, the optimization problem server one faces in the alternative system is precisely the mean-field problem (1) with the shared pool waiting time  $w = 0$ . Thus, the minimum time-average



total cost in the alternative system is bounded from below by the optimal value of the fluid mean-field problem (which equals  $c\lambda + \mathbb{E}[Q^F]$ ) minus the time-average compensation. Specifically, let  $A = \{\Delta(\tau) > \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}\}$  denote the event that the inter-arrival time is at least  $\frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$ . The time-average total cost in the alternative system is at least

$$\begin{aligned} & c\lambda + \mathbb{E}[Q^F] - c \cdot \left( \phi \cdot \mathbb{P}[A^c] + \frac{c(\lambda, \phi, \delta)}{N^{1-\delta}} \cdot \mathbb{P}[A] \right) \\ & \geq c\lambda + \mathbb{E}[Q^F] - c \cdot \frac{\phi + c(\lambda, \phi, \delta)}{N^{1-\delta}} \end{aligned}$$

where the inequality follows from the fact that  $\mathbb{P}[A^c] \leq \frac{1}{N^{1-\delta}}$  by (58). Finally, taking  $M_8(\lambda, \phi, \delta) = c \cdot (\phi + c(\lambda, \phi, \delta))$  completes the proof.

**Lemma C.7.** *The optimal value of the alternative system is smaller than that of the relaxation to server one's problem.*

*Proof.* It suffices to show that the time-average total cost of any policy is smaller in the alternative system than in the relaxation. This is due to the fact that the compensation in the alternative system is more than the augmented cost of offering help in the alternative system. To see this, suppose a capacity unit arrives at time  $\tau$  and server one offers help to the shared pool. If event  $A = \{\Delta(\tau) > \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}\}$  occurs, then we have  $\tau - h(\tau) > \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$  by (57). By Lemma C.6, the difference between the expected cost of offering help in the alternative system and the relaxation is

$$\left( c\phi - c \cdot \frac{c(\lambda, \phi, \delta)}{N^{1-\delta}} \right) - c \cdot \mathbb{P}[Q_0(\tau) \geq 1 | \tau - h(\tau)] \leq 0.$$

Therefore, the expected cost of offering help is smaller in the alternative system than in the relaxation. Alternatively, suppose event  $A^c$  occurs. The expected cost of offering help also is smaller in the alternative system because

$$(c\phi - c\phi) - c \cdot \mathbb{P}[Q_0(\tau) \geq 1 | \tau - h(\tau)] \leq 0. \quad \square$$

#### C.5.4 Proof of Lemma C.6

Note that the shared pool is emptied at time  $h(\tau)$ . In the proof, we show that the shared pool transitions to the stationary distribution by time  $\tau$  with a high probability (Lemma C.8). In addition, in the stationary distribution, the shared pool is non-empty with a probability close to  $\phi$  (Lemma C.9). By combining Lemmas C.8 and C.9, we obtain the desired result.

We focus on the time between  $h(\tau)$  and  $\tau$ . Within this time interval, server one has no interaction with the shared pool. Additionally, servers two to  $N$  are in the stationary distribution throughout because they are assumed so at time zero. We consider an alternative system that contains servers  $i \in [2 : N]$  and the shared pool, and we assume that the system is in the stationary distribution at time zero. Denote by  $\bar{Q}_0(t)$  the queue length of the shared pool in the alternative system at time  $t$  and by  $\bar{Q}_0(\infty)$  the queue length of the shared pool in the stationary distribution of the alternative system. Note that since the alternative system and server one do not interact, the shared pool queue length in the alternative system is in the stationary distribution in both time  $h(\tau)$  and  $\tau$ .

We first bound the total variation distance between the distributions of the shared pool queue lengths  $Q_0(\tau)$  and  $\bar{Q}_0(\tau)$  ( $\stackrel{d}{=} \bar{Q}_0(\infty)$ ) at time  $\tau$  in Lemma C.8. Lemma C.8 indicates that  $Q_0(\tau)$  is in the stationary distribution with a high probability.

**Lemma C.8.** *Let  $\mathcal{P}$  denote the distribution of the shared pool queue length  $Q_0(\tau)$  in the original system at time  $\tau$ , conditional on the value of the time difference  $\tau - h(\tau)$ , and  $\bar{\mathcal{P}}$  the stationary distribution of the shared pool queue length in the alternative system. Suppose that the time difference  $\tau - h(\tau) \geq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$ ; we have*

$$\|\mathcal{P} - \bar{\mathcal{P}}\|_{\text{TV}} \leq c_1(\lambda, \phi, \delta) \cdot \exp\left(-\alpha_1(\lambda, \phi) \cdot N^{\delta/2}\right)$$

where  $c_1(\lambda, \phi, \delta)$  is a constant that depends only on the values of  $\lambda$ ,  $\phi$  and  $\delta$ , and  $\alpha_1(\lambda, \phi)$  a constant that depends only on the values of  $\lambda$  and  $\phi$ .

We prove Lemma C.8 in Appendix C.5.5 by coupling the original and alternative systems and using a drift analysis. Since

$$\begin{aligned} \left| \mathbb{P}[Q_0(\tau) \geq 1 | \tau - h(\tau)] - \mathbb{P}[\bar{Q}_0(\infty) \geq 1] \right| &\leq \max_A \left| \mathbb{P}[Q_0(\tau) \in A | \tau - h(\tau)] - \mathbb{P}[\bar{Q}_0(\infty) \in A] \right| \\ &= \|\mathcal{P} - \bar{\mathcal{P}}\|_{\text{TV}}, \end{aligned} \quad (59)$$

Lemma C.8 immediately implies that when  $\tau - h(\tau) \geq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$ , we have

$$\mathbb{P}[Q_0(\tau) \geq 1 | \tau - h(\tau)] \geq \mathbb{P}[\bar{Q}_0(\infty) \geq 1] - c_1(\lambda, \phi, \delta) \cdot \exp\left(-\alpha_1(\lambda, \phi) \cdot N^{\delta/2}\right). \quad (60)$$

We next show that the stationary probability  $\mathbb{P}[\bar{Q}_0(\infty) \geq 1]$  is close to the token-earning probability  $\phi$  in Lemma C.9.

**Lemma C.9.** *We have*

$$\mathbb{P}[\bar{Q}_0(\infty) \geq 1] \geq \phi - \frac{c_2(\lambda, \phi, \delta)}{N^{1-\delta}} \quad (61)$$

for some constant  $c_2(\lambda, \phi, \delta)$  that depends only on the values of  $\lambda$ ,  $\phi$  and  $\delta$ .

We prove Lemma C.9 in Appendix C.5.6. The proof is analogous to the proof of Lemma C.8 where we couple the alternative system with another auxiliary system to derive the result. Finally, (60) and (61) imply that there exists a constant  $c(\lambda, \phi, \delta)$  such that Lemma C.6 holds.

### C.5.5 Proof of Lemma C.8

We use a coupling method to prove this. Specifically, we couple the original and alternative systems together so that servers two to  $N$  encounter identical arrival sequences of jobs and capacity units in both systems. As a result, servers two to  $N$  take the same actions at the same time in both systems. This, together with the fact that the shared pool in the original system is emptied at time  $h(\tau)$ , implies that

$$\bar{Q}_0(t) \geq Q_0(t), \quad \forall t \in (h(\tau), \tau),$$

Moreover, if  $\bar{Q}_0(t) = Q_0(t)$  at some time  $t \in (h(\tau), \tau)$ , they continue to be equal at any subsequent time before time  $\tau$ . Let

$$T \triangleq \inf \left\{ t > h(\tau) : \bar{Q}_0(t) = 0 \right\}$$

denote the first time that the shared pool in the alternative system becomes empty. From the above reasoning, if  $T < \tau$ , then  $\bar{Q}_0(T) = Q_0(T) = 0$  and hence  $\bar{Q}_0(\tau) = Q_0(\tau)$ . Consequently,

$$\|\mathcal{P} - \bar{\mathcal{P}}\|_{\text{TV}} \leq \mathbb{P}[\bar{Q}_0(\tau) \neq Q_0(\tau)] \leq \mathbb{P}[T - h(\tau) \geq \tau - h(\tau)], \quad (62)$$

where the first inequality is because  $\bar{Q}_0(\tau) \stackrel{d}{=} \bar{Q}_0(\infty)$ . In the following, we bound the probability  $\mathbb{P}[T - h(\tau) \geq \tau - h(\tau)]$  from above.

**Step One: Bounding the Number of Arrivals from Below** We first show that the number of arrivals at servers two to  $N$  in the time interval  $(h(\tau), \tau)$  is not tiny with a high probability. Specifically, let  $H$  denote the number of arrivals of either jobs or capacity units at servers two to  $N$  within the time interval  $(h(\tau), \tau)$ . The random variable  $H \sim \text{Poisson}(\lambda_H)$  follows a Poisson distribution with a mean value of  $\lambda_H \triangleq (N-1)(1+\lambda)(\tau - h(\tau)) \geq \frac{N-1}{N^{1-\delta}}$ , where the inequality follows from the assumption that  $\tau - h(\tau) \geq \frac{1}{1+\lambda} \frac{1}{N^{1-\delta}}$ . By the concentration inequality for Poisson random variables (Lemma C.10), we have

$$\mathbb{P}[H \leq N^{\delta/2}] \leq c_3(\delta) \cdot \exp(-N^\delta) \quad (63)$$

where  $c_3(\delta)$  is a constant that depends only on the value of  $\delta$ . To see this, let  $\lambda = \lambda_H$  and  $u = 1 - N^{\delta/2}/\lambda = 1 - o(1)$  in Lemma C.10 and note that  $\lim_{u \rightarrow 1} g(u) = 1$ . The result follows from the fact that  $\lambda_H \geq \frac{N-1}{N^{1-\delta}} = \Theta(N^\delta)$ .

**Step Two: Defining a “Good” Typical Event** Secondly, we specify a “good” event that happens with a high probability. To do so, it is convenient to consider the equivalent discrete-time model, where each period corresponds to the  $\ell$ -th arrival for  $\ell \in [H]$ . Specifically, suppose that the  $\ell$ -th arrival (of either a job or capacity unit) arrives at time  $t_\ell$ , with  $t_0 \triangleq h(\tau) < t_1 < \dots < t_H < \tau$ . Let  $\bar{Q}_0[\ell] \triangleq \bar{Q}_0(t_\ell)$  denote the queue length of the shared pool at time  $t_\ell$  (after servers have taken action) in the alternative system.

For each server  $i$ , we let binary variable  $A_i[\ell]$  (and  $S_i[\ell]$ ) indicate if server  $i$  requests help (and offers help) in period  $\ell$ . Additionally, let  $A[\ell] \triangleq \sum_{i \in [2:N]} A_i[\ell] \in \{0, 1\}$  and  $S[\ell] \triangleq \sum_{i \in [2:N]} S_i[\ell] \in \{0, 1\}$  indicate if there is any request or provision of help in period  $\ell$ . We have

$$\bar{Q}_0[\ell + 1] = \left( \bar{Q}_0[\ell] + A[\ell] - S[\ell] \right)^+, \forall \ell \in [0 : H - 1].$$

Since servers two to  $N$  are in the stationary distribution at any time  $t_\ell$ ,  $\mathbb{E}(A[\ell] - S[\ell]) = -(1-\phi)g$  with  $g > 0$  being the rate of offering help in the stationary distribution. Additionally, let  $\hat{\pi}_\ell$  denote the empirical distribution of queue lengths and token amounts of servers two to  $N$  at time  $t_\ell$ , i.e.,

$$\hat{\pi}_\ell(q, s) \triangleq \frac{1}{N-1} \sum_{i=2}^N \mathbb{1}\left\{Q_i(t_\ell) = q, S_i(t_\ell) = s\right\},$$

and  $\mathcal{E}_\ell$  denote the event that the average drift in period  $\ell$  is no larger than  $-\frac{1}{2}(1-\phi)g$ , i.e.,

$$\mathcal{E}_\ell \triangleq \left\{ \hat{\pi}_\ell : \sum_q \hat{\pi}_\ell(q, s) A_{qs} - \sum_q \hat{\pi}_\ell(q, s) S_{qs} \leq -\frac{1}{2}(1-\phi)g \right\},$$

where  $A_{qs}$  and  $S_{qs}$  are the probabilities that a server requests and offers help, respectively, when it has  $q$  jobs and  $s$  tokens. Since the dynamics of servers  $i \in [2 : N]$  are independent, by Hoeffding's inequality, we have (please refer to Lemma A.2)

$$\mathbb{P}(\mathcal{E}_\ell^c) \leq \exp\left(-\frac{1}{8}(1-\phi)^2 g^2 (N-1)\right). \quad (64)$$

Finally, since  $\bar{Q}_0[0] \triangleq \bar{Q}_0(h(\tau))$  is in the stationary distribution, from Lemma A.3 we have

$$\mathbb{P}\left(\bar{Q}_0[0] > \frac{1}{4}(1-\phi)gN^{\delta/2}\right) \leq c_4(\lambda, \phi) \cdot \exp\left(-\alpha_2(\lambda, \phi) \cdot N^{\delta/2}\right) \quad (65)$$

for some constants  $c_4(\lambda, \phi)$  and  $\alpha_2(\lambda, \phi)$  that depend only on the values of  $\lambda$  and  $\phi$ .

We now define a “good” event  $A$  with

$$A \triangleq \left\{H > N^{\delta/2}\right\} \cap \left\{\cap_{\ell \leq N^{\delta/2}} \mathcal{E}_\ell\right\} \cap \left\{\bar{Q}_0[0] \leq \frac{1}{4}(1-\phi)gN^{\delta/2}\right\}.$$

From (63) - (65) and the union bound, we have

$$\begin{aligned} \mathbb{P}(A^c) &\leq c_3(\delta) \cdot \exp(-N^\delta) + N^{\delta/2} \cdot \exp\left(-\frac{1}{8}(1-\phi)^2 g^2 (N-1)\right) + c_4(\lambda, \phi) \cdot \exp\left(-\alpha_2(\lambda, \phi) \cdot N^{\delta/2}\right) \\ &\leq c_5(\lambda, \phi, \delta) \cdot \exp\left(-\alpha_2(\lambda, \phi) \cdot N^{\delta/2}\right) \end{aligned} \quad (66)$$

for some constant  $c_5(\lambda, \phi, \delta)$  that depends only on the values of  $\lambda$ ,  $\phi$ , and  $\delta$ . Thus, event  $A$  happens with a high probability.

**Step Three: A Drift Analysis** Finally, we consider a stochastic process  $\tilde{Q}_0[\ell] = \bar{Q}_0[0] + \sum_{r=1}^{\ell} (A[r] - S[r])$  with  $\ell \leq H$ . Conditioning on event  $A$ , we have

$$\mathbb{E}\left(\tilde{Q}_0[\ell+1] + \frac{1}{2}(1-\phi)g(\ell+1) \mid \tilde{Q}_0[\ell]\right) \leq \tilde{Q}_0[\ell] + \frac{1}{2}(1-\phi)g\ell.$$

Thus,  $\tilde{Q}_0[\ell] + \frac{1}{2}(1-\phi)g\ell$  a super-martingale. From Azuma's inequality we have

$$\mathbb{P}\left(Q_0[\ell] + \frac{1}{2}(1-\phi)g\ell - Q_0[0] \geq \epsilon\right) \leq \exp\left(\frac{-\epsilon^2}{2\ell}\right).$$

By taking  $\ell = N^{\delta/2}$  and  $\epsilon = \frac{1}{4}(1-\phi)gN^{\delta/2}$ , and noting that  $Q_0[0] \leq \frac{1}{4}(1-\phi)gN^{\delta/2}$  given event  $A$ , the above implies that

$$\mathbb{P}\left(Q_0[N^{\delta/2}] \geq 0 \mid A\right) \leq \exp\left(-\frac{1}{32}(1-\phi)^2 g^2 N^{\delta/2}\right). \quad (67)$$

As a result,

$$\begin{aligned}
\mathbb{P}\left[T - h(\tau) \geq \tau - h(\tau) \mid A\right] &\leq \mathbb{P}\left[\bar{Q}_0[\ell] \geq 1, \forall i \leq N^{\delta/2} \mid A\right] \\
&= \mathbb{P}\left[\tilde{Q}_0[\ell] \geq 1, \forall i \leq N^{\delta/2} \mid A\right] \\
&\leq \mathbb{P}\left[\tilde{Q}_0[N^{\delta/2}] \geq 1 \mid A\right] \\
&\leq \exp\left(-\frac{1}{32}(1-\phi)^2 g^2 N^{\delta/2}\right),
\end{aligned} \tag{68}$$

where the last inequality follows from (67).

Finally, from (66) and (68) we have

$$\|\mathcal{P} - \bar{\mathcal{P}}\|_{\text{TV}} \leq \mathbb{P}\left[T - h(\tau) \geq \tau - h(\tau)\right] \leq c_1(\lambda, \phi, \delta) \cdot \exp\left(-\alpha_1(\lambda, \phi) \cdot N^{\delta/2}\right)$$

for some constant  $c_1(\lambda, \phi, \delta)$  that depends only on the values of  $\lambda$ ,  $\phi$  and  $\delta$  and constant  $\alpha_1(\lambda, \phi)$  that depends only on the values of  $\lambda$  and  $\phi$ .

**Lemma C.10** (Concentration Inequality for Poisson Random Variables). *Let  $X \sim \text{Poisson}(\lambda)$  be a Poisson random variable with mean value  $\lambda > 0$ . We have*

$$\mathbb{P}(X \leq \lambda(1-u)) \leq \exp(-\lambda g(u))$$

for any  $0 \leq u < 1$ , where  $g(u) = u + (1-u) \ln(1-u)$ .

*Proof.* For any  $t \geq 0$ , we have

$$\begin{aligned}
\mathbb{P}(X \leq \lambda(1-u)) &= \mathbb{P}\left[\exp(-tX) \geq \exp(-t\lambda(1-u))\right] \\
&\leq \mathbb{E}\left[\exp(-tX)\right] \exp(t\lambda(1-u)) \\
&= \exp\left(\lambda(e^{-t} - 1) + t\lambda(1-u)\right)
\end{aligned}$$

where the inequality follows from Markov's inequality and the second equality from the moment generating function of the Poisson distribution. It turns out that the right-hand side of the second equality is minimized by setting  $t = -\ln(1-u)$ , in which case the right-hand side simplifies to  $\exp(-\lambda g(u))$ .  $\square$

### C.5.6 Proof of Lemma C.9

We use the same coupling method as in Appendix C.5.5 to prove the result. Specifically, we couple the alternative system defined in Appendix C.5.4 (labeled as system one in this section) with another auxiliary system (labeled as system two in this section). We first describe the two systems below.

**System One** The alternative system defined in Appendix C.5.4. The system includes servers  $i \in [2:N]$  and the shared pool. All the  $N-1$  servers follow the FMFE policy  $\bar{\pi}^F$ . We assume that the system is in the stationary distribution at time zero. We denote the queue length of the shared pool in this system by  $\bar{Q}_0(t)$ .

**System Two** The system includes servers  $i \in [N]$  and the shared pool. All the  $N$  servers follow the FMFE policy  $\bar{\pi}^F$ . We assume that the system is in the stationary distribution at time zero. We denote the queue length of the shared pool in this system by  $\tilde{Q}_0(t)$ .

Suppose server one in system two offers help at a time  $\tau$ . Let  $\tilde{Q}_0(\tau-) \triangleq \lim_{t \uparrow \tau} \tilde{Q}_0(t)$  denote the queue length of the shared pool before server one offers help. Lemma C.11 shows that the probability that  $\tilde{Q}_0(\tau-)$  is non-zero (thus, server one serves a job when offering help) is  $\phi$ .

**Lemma C.11.** *We have  $\mathbb{P}[\tilde{Q}_0(\tau-) \geq 1] = \phi$ .*

*Proof.* This is because all servers are stochastically identical and follow the same strategy  $\bar{\pi}^F$ , and a capacity unit contributed to the shared pool serves a job with probability  $\phi$  in the stationary distribution by Remark 3.1.  $\square$

On the other hand, since server one and system one do not interact, we have  $\bar{Q}_0(\tau-) \stackrel{d}{=} \bar{Q}_0(\infty)$ , i.e., the shared pool in system one is in the stationary distribution. Therefore, to show that the stationary probability  $\mathbb{P}[\bar{Q}_0(\infty) \geq 1]$  is close to the value of  $\phi$ , it suffices to show that the shared pool queue lengths in the two systems at time  $\tau$  are close in terms of the total variation distance.

We couple the two systems together so that servers  $i \in [2:N]$  receive the same arrival sequences of jobs and capacity units in both systems. As a result, servers  $i \in [2:N]$  take the same actions at the same time in both systems.

Let  $h(\tau)$  denote the time of the most recent arrival of either a job or a capacity unit at server one before time  $\tau$ . The time difference  $\Delta(\tau) = \tau - h(\tau)$  follows an exponential distribution with a rate parameter of  $1 + \lambda$ . Note that within the time interval  $(h(\tau), \tau)$ , the shared pool interacts only with servers two to  $N$  in both systems. Let

$$T_1 \triangleq \inf \left\{ t > h(\tau) : \bar{Q}_0(t) = 0 \right\}$$

denote the first time that the shared pool in system one becomes empty, and

$$T_2 \triangleq \inf \left\{ t > h(\tau) : \tilde{Q}_0(t) = 0 \right\}$$

the first time that the shared pool in system two becomes empty. Finally, let

$$T \triangleq \max \{T_1, T_2\}.$$

Since the two systems are coupled, if  $T < \tau$ , then  $\bar{Q}_0(T) = Q_0(T) = 0$  and  $\bar{Q}_0(t) = Q_0(t)$  for any  $t \in [T, \tau)$ . As a result, we have

$$\begin{aligned} \phi - \mathbb{P}[\bar{Q}_0(\infty) \geq 1] &= \mathbb{P}[\tilde{Q}_0(\tau-) \geq 1] - \mathbb{P}[\bar{Q}_0(\tau-) \geq 1] \\ &\leq \left| \mathbb{P}[\tilde{Q}_0(\tau-) \geq 1] - \mathbb{P}[\bar{Q}_0(\tau-) \geq 1] \right| \\ &\leq \mathbb{P}[\bar{Q}_0(\tau-) \neq \tilde{Q}_0(\tau-)] \\ &\leq \mathbb{P}[T - h(\tau) \geq \tau - h(\tau)], \end{aligned} \tag{69}$$

where the first inequality follows from Lemma C.11 and  $\bar{Q}_0(\tau-) \stackrel{d}{=} \bar{Q}_0(\infty)$ , and the inequalities follow from the same reasoning for (59) and (62). In the following, we bound the probability  $\mathbb{P}[T - h(\tau) \geq \tau - h(\tau)]$  from above in the same way as in Appendix C.5.5.

**Step One: Bounding the Number of Arrivals from Below** We first show that the number of arrivals at servers two to  $N$  in the time interval  $(h(\tau), \tau)$  is not tiny with a high probability. Specifically, let  $H$  denote the number of arrivals of either jobs or capacity units at servers two to  $N$  in the time interval  $(h(\tau), \tau)$ . We first show in Lemma C.12 that the random variable  $H \sim \text{Geo}(\frac{1}{N})$  follows a geometric distribution with a success probability of  $\frac{1}{N}$ .

**Lemma C.12.** *Let  $H$  denote the number of arrivals at servers two to  $N$  in the time interval  $(h(\tau), \tau)$ ;  $H$  follows a geometric distribution with a success probability of  $\frac{1}{N}$ ; i.e.,  $\mathbb{P}[H = k] = (\frac{N-1}{N})^k \frac{1}{N}$  for any integer  $k \in \mathbb{N}$ .*

*Proof.* Since the time difference  $\Delta(\tau) = \tau - h(\tau)$  follows an exponential distribution with a rate parameter of  $1 + \lambda$ , and conditioning on  $\Delta(\tau) = x$ , the number of arrivals  $H$  follows a Poisson distribution with a mean value of  $(N - 1)(1 + \lambda) \cdot x$ , for any integer  $k \in \mathbb{N}$ ,

$$\mathbb{P}[H = k] = \int_0^\infty (1 + \lambda)e^{-(1+\lambda)x} \cdot \frac{((N - 1)(1 + \lambda) \cdot x)^k \cdot e^{-(N-1)(1+\lambda) \cdot x}}{k!} = \left(\frac{N - 1}{N}\right)^k \frac{1}{N}.$$

Thus, the number of arrivals  $H$  follows a geometric distribution with a success probability of  $\frac{1}{N}$ .  $\square$

From Lemma C.12, we have

$$\mathbb{P}[H < N^\delta] \leq \frac{N^\delta}{N} = \frac{1}{N^{1-\delta}}. \quad (70)$$

**Step Two: Bounding the Shared Pool Queue Lengths from Above** Secondly, we show that with a high probability, the shared pools in both systems are not extremely long at time  $h(\tau)$ .

First, since server one and system one do not interact,  $\bar{Q}_0(h(\tau)) \stackrel{d}{=} \bar{Q}_0(\infty)$ . Therefore, from Lemma A.3 we have

$$\mathbb{P}\left(\bar{Q}_0(h(\tau)) > \frac{1}{4}(1 - \phi)gN^\delta\right) \leq c_6(\lambda, \phi) \cdot \exp\left(-\alpha_3(\lambda, \phi) \cdot N^\delta\right) \quad (71)$$

for some constants  $c_6(\lambda, \phi)$  and  $\alpha_3(\lambda, \phi)$  that depend only on the values of  $\lambda$  and  $\phi$ .

Secondly, let  $\tilde{Q}_1(\infty)$  and  $\tilde{S}_1(\infty)$  denote the number of jobs and tokens at server one in the stationary distribution of in system two and  $\mathbb{P}[\tilde{Y}_1(\infty) = 1]$  the stationary rate of offering help by server one in system two. Note that

$$\min\left\{1, \frac{\lambda}{\phi}\right\} \cdot \mathbb{P}\left(\tilde{Q}_1(\infty) = 0, \tilde{S}_1(\infty) \leq C - 1\right) = \mathbb{P}[\tilde{Y}_1(\infty) = 1] \geq g' > 0$$

for some constant  $g' > 0$  and any value  $C$  of token-amount upper bound. In the above, the equality follows from the fact that a server requests help only when its queue is empty and token amount is smaller than  $C$ , and in such instances requests help with a probability of  $\min\left\{1, \frac{\lambda}{\phi}\right\}$ ; the existence of the positive constant  $g'$  follows from Lemma 6.2 Bullet 3 (for a more detailed explanation, see the paragraph of “Validity of Lemma A.3” in Appendix C.3.1, especially the way we argue  $\underline{g} > 0$

there). Thus, for any integer  $n \in \mathbb{N}$ , we have

$$\begin{aligned}\mathbb{P}\left(\tilde{Q}_0(h(\tau)) > n\right) &= \mathbb{P}\left(\tilde{Q}_0(\infty) > n \mid \text{server one will offer help when a capacity unit arrives}\right) \\ &= \mathbb{P}\left(\tilde{Q}_0(\infty) > n \mid \tilde{Q}_1(\infty) = 0, \tilde{S}_1(\infty) \leq C - 1\right) \\ &\leq \frac{1}{g} \cdot \mathbb{P}\left(\tilde{Q}_0(\infty) > n\right)\end{aligned}$$

with constant  $g \triangleq g' / \min\left\{1, \frac{\lambda}{\phi}\right\} > 0$ . Therefore, again from Lemma A.3 we have

$$\mathbb{P}\left(\tilde{Q}_0(h(\tau)) > \frac{1}{4}(1 - \phi)gN^\delta\right) \leq c_7(\lambda, \phi) \cdot \exp\left(-\alpha_4(\lambda, \phi) \cdot N^\delta\right) \quad (72)$$

for some constants  $c_7(\lambda, \phi)$  and  $\alpha_4(\lambda, \phi)$  that depend only on the values of  $\lambda$  and  $\phi$ .

**Step Three: A Drift Analysis** We first define a “good” event  $A$  with

$$A \triangleq \left\{H \geq N^\delta\right\} \cap \left\{\cap_{\ell \leq N^\delta} \mathcal{E}_\ell\right\} \cap \left\{\tilde{Q}_0(h(\tau)) \leq \frac{1}{4}(1 - \phi)gN^\delta\right\} \cap \left\{\tilde{Q}_0(h(\tau)) \leq \frac{1}{4}(1 - \phi)gN^\delta\right\},$$

where event  $\mathcal{E}_\ell$  is defined in Appendix C.5.5. From (64), (70) - (72), and the union bound, we have

$$\mathbb{P}\left(A^c\right) \leq \frac{c_8(\lambda, \phi, \delta)}{N^{1-\delta}} \quad (73)$$

for some constant  $c_8(\lambda, \phi, \delta)$  that depends only on the values of  $\lambda$ ,  $\phi$ , and  $\delta$ .

Following the same drift analysis as in Step three of Appendix C.5.5, we have

$$\mathbb{P}\left[T_1 - h(\tau) \geq \tau - h(\tau) \mid A\right] \leq \exp\left(-\frac{1}{32}(1 - \phi)^2 g^2 N^\delta\right), \quad (74)$$

and

$$\mathbb{P}\left[T_2 - h(\tau) \geq \tau - h(\tau) \mid A\right] \leq \exp\left(-\frac{1}{32}(1 - \phi)^2 g^2 N^\delta\right), \quad (75)$$

which are counterparts to (68). Thus, from (73) - (75) we have

$$\begin{aligned}\mathbb{P}\left[T - h(\tau) \geq \tau - h(\tau)\right] &\leq \mathbb{P}(A) \cdot \mathbb{P}\left[T - h(\tau) \geq \tau - h(\tau) \mid A\right] + \mathbb{P}(A^c) \\ &\leq \mathbb{P}(A) \cdot \left(\mathbb{P}\left[T_1 - h(\tau) \geq \tau - h(\tau) \mid A\right] + \mathbb{P}\left[T_2 - h(\tau) \geq \tau - h(\tau) \mid A\right]\right) + \mathbb{P}(A^c) \\ &\leq \frac{c_2(\lambda, \phi, \delta)}{N^{1-\delta}}\end{aligned}$$

for some constant  $c_2(\lambda, \phi, \delta)$  that depends only on the values of  $\lambda$ ,  $\phi$ , and  $\delta$ . Combining the above with (69) yields the desired result.

## C.6 Proof of Theorem 7.2

**Proof of Bullet One** Suppose all servers follow the FMFE strategy  $\pi^F$ . Then, the dynamics of the queue length of the shared pool is an  $M/M/1$  queue with a utilization factor of  $\phi = \bar{\rho}$ .



Therefore, the long-run average waiting time in the shared pool, denoted by  $w$ , is  $w = \frac{\bar{\rho}}{1-\bar{\rho}} \cdot \frac{1}{\sum_{i \in [N]} \lambda_i}$ . Consequently, the time-average waiting cost for jobs in the shared pool that come from server one is  $\lambda_1 w \leq \frac{\lambda \bar{\rho}}{\lambda(1-\bar{\rho}) \cdot N}$ .

Additionally, every server  $i$  contributes its capacity units to the shared pool at a rate of  $\mu_i \cdot \rho_i / \bar{\rho} = \lambda_i / \bar{\rho}$ . Since the job processing costs are distributed among servers proportional to the rate of capacity units a server contributes to the shared pool, the time-average job processing cost of server one is  $c\lambda_1$ .

**Proof of Bullet Two** The proof mimics the proof of Lemma 6.5 (Appendix C.5). Note that because the token-earning probability  $\phi$  equals  $\bar{\rho}$ , Lemma C.9 in Appendix C.5 is simplified to be

$$\mathbb{P}[\bar{Q}_0(\infty) \geq 1] = \phi = \bar{\rho}.$$

This is because the dynamics of the shared pool's queue length  $\bar{Q}_0(\infty)$  is an  $M/M/1$  queue with a utilization factor of  $\phi = \bar{\rho}$ .

## D Problem of Server One in the Small Market Analysis

In this section, we formulate the problem of server one (i.e., the deviating server) in Section 6.3. Recall that we relax the problem by providing server one additional advantage to obtain a tractable optimization problem. First, we enable server one to have complete information about the shared pool so that server one can make decisions based on its queue length. Additionally, we bound the waiting time in the shared pool from below by  $\frac{q_0+1}{N}$ , where  $q_0$  denotes the queue length of the shared pool at the current time.

After the relaxation, server one's optimal strategy depends only on two state variables—server one's queue length (denoted by  $q_1$ ) and the queue length of the shared pool (denoted by  $q_0$ ), and can be formulated as a two-dimensional dynamic program that is easy to solve. To describe the problem, we consider an equivalent embedded discrete-time model of server one's problem, where in each period, a job or a capacity unit arrives at either server one or the shared pool.

Case One: With probability  $p_1 = \frac{(N-1)\lambda}{N(1+\lambda)}$ , a job arrives at the shared pool; consequently, the shared pool's queue length  $q_0$  increases by one. In this case, since server one does not request help before the job arrives at the pool, it will not do so after the arrival.

Case Two: With probability  $p_2 = \frac{N-1}{N(1+\lambda)}$ , a capacity unit arrives at the shared pool. If the shared pool is empty, the capacity unit is wasted. In this case, since there is no change in the queue length of either server one or the shared pool, server one would not request help (otherwise, it would do so before the capacity unit arrives). Alternatively, if the shared pool is non-empty, one job is served from the shared pool; hence, the shared pool's queue length reduces by one. In this case, server one decides whether to relocate a job from its queue (if there is any) to the shared pool by requesting help. We remark that server one would request help for at most one job, because if server one would like to request help for a second job, it would rather request help for a job before the arrival of the capacity unit; however, this does not hold true.

Case Three: With probability  $p_3 = \frac{\lambda}{N(1+\lambda)}$ , a job arrives at server one. In this case, server one needs to decide whether to add the job to its queue or relocate the job to the shared pool by requesting help. Analogously, if server one requests help, it would request help only for the incoming job. This is because if server one would like to request help for a second job, it would rather request help for a job before the new job arrives (however, this does not hold true).

Case Four: With probability  $p_4 = \frac{1}{N(1+\lambda)}$ , a capacity unit arrives at server one. In this case, server one needs to decide whether to (i) process a job from its queue (if one is present), (ii) offer help to the shared pool, or (iii) simply waste the unit and be idle. We remark that if server one offers help to the shared pool, the cost depends on the state of the shared pool. Specifically, server one experiences a cost of  $c$  from processing a job from the shared pool if the pool is non-empty and a cost of zero if the pool is empty (because server one does not process a job). Nevertheless, server one obtains a token with probability  $\phi$  regardless in both scenarios. We also remark that server one might serve a job from the shared pool and then request help immediately (because the shared pool's queue length reduces after server one serves a job). However, this is never optimal because server one can be better off serving a job from its queue instead.<sup>14</sup>

The two-dimensional DP problem can be solved by a linear program (76). In (76),  $p(q_1, q_0)$  represents the stationary probability that the system state is  $(q_1, q_0)$ —i.e., server one has  $q_1$  jobs and the shared pool has  $q_0$  jobs,  $p_2(q_1, q_0)$  the joint probability that the system state is  $(q_1, q_0)$ , a capacity unit arrives at the shared pool, and server one requests help for a job,  $p_3(q_1, q_0, 0)$  and  $p_3(q_1, q_0, 1)$  the joint probability that the system state is  $(q_1, q_0)$ , a job arrives at server one, and server one adds the job to its queue or requests help for the job, respectively, and  $p_4(q_1, q_0, 0)$  and  $p_4(q_1, q_0, 1)$  the joint probability that the system state is  $(q_1, q_0)$ , a capacity unit arrives at server one, and server one serves a job from its queue or offers help to the shared pool, respectively.

Thus, in (76), the decision variables are the stationary distribution of state-action pairs, the objective is to minimize server one's total cost in the stationary distribution, and constraints are the flow balance constraint of the tokens, i.e., the constraint that the expected rates of earning and spending tokens are equal (the first constraint), and balance equations that characterize the dynamics of the states  $(q_1, q_0)$  (the rest constraints).

$$\begin{aligned}
V^{\text{OPT}} = & \max_{\substack{p(q_1, q_0), p_2(q_1, q_0), \\ p_3(q_1, q_0, 0), p_3(q_1, q_0, 1), \\ p_4(q_1, q_0, 0), p_4(q_1, q_0, 1)}} & N(1 + \lambda) \cdot \sum_{q_1, q_0 \geq 0} \left\{ p(q_1, q_0) \cdot \frac{q_1}{N(1 + \lambda)} + \left( p_2(q_1, q_0) + p_3(q_1, q_0, 1) \right) \cdot \frac{q_0 + 1}{N} \right. \\
& \left. + c \cdot \left( p_4(q_1, q_0, 0) + p_4(q_1, q_0, 1) \cdot \mathbb{1}[q_0 \geq 1] \right) \right\} \\
\text{s.t.} & \sum_{q_1, q_0 \geq 0} \left( p_2(q_1, q_0) + p_3(q_1, q_0, 1) \right) = \phi \cdot \sum_{q_1, q_0 \geq 0} p_4(q_1, q_0, 1), \quad (76) \\
& p(q_1, q_0) = p_1 \cdot p(q_1, q_0 - 1) \cdot \mathbb{1}[q_0 \geq 1] + p_2(q_1 + 1, q_0) \\
& \quad + \left( p_2 \cdot p(q_1, q_0 + 1) - p_2(q_1, q_0 + 1) \right) + p_2 \cdot p(q_1, q_0) \cdot \mathbb{1}[q_0 = 0] \\
& \quad + p_3(q_1 - 1, q_0, 0) \cdot \mathbb{1}[q_1 \geq 1] + p_3(q_1, q_0 - 1, 1) \cdot \mathbb{1}[q_0 \geq 1] \\
& \quad + p_4(q_1 + 1, q_0, 0) + p_4(q_1, q_0 + 1, 1) + p_4(q_1, q_0, 1) \cdot \mathbb{1}[q_0 = 0] \\
& \quad + \left( p_4 \cdot p(q_1, q_0) - p_4(q_1, q_0, 0) - p_4(q_1, q_0, 1) \right), \forall q_1, q_0 \geq 0, \\
& \sum_{q_1, q_0 \geq 0} p(q_1, q_0) = 1, \\
& p_2(q_1, q_0) \leq p_2 \cdot p(q_1, q_0), \forall q_1, q_0 \geq 0, \\
& p_3(q_1, q_0, 0) + p_3(q_1, q_0, 1) = p_3 \cdot p(q_1, q_0), \forall q_1, q_0 \geq 0, \\
& p_4(q_1, q_0, 0) + p_4(q_1, q_0, 1) \leq p_4 \cdot p(q_1, q_0), \forall q_1, q_0 \geq 0, \\
& p_2(q_1, q_0) = 0, \forall q_1 = 0, q_0 \geq 0,
\end{aligned}$$

<sup>14</sup>By doing this, the job can get served immediately rather than waiting till being served in the shared pool.

$$\begin{aligned}
p_2(q_1, q_0) &= 0, \forall q_0 \geq 0, q_1 = 0, \\
p_4(q_1, q_0, 0) &= 0, \forall q_1 = 0, q_0 \geq 0, \\
p(q_1, q_0), p_2(q_1, q_0), p_3(q_1, q_0, 0), p_3(q_1, q_0, 1) &\geq 0, \forall q_1, q_0 \geq 0, \\
p_4(q_1, q_0, 0), p_4(q_1, q_0, 1) &\geq 0, \forall q_1, q_0 \geq 0.
\end{aligned}$$