Incentivizing Resource Pooling

Chen Chen[1], Yilun Chen[2], and Pengyu Qian[3]

[1]New York University Shanghai
[2]The School of Data Science, The Chinese University of Hong Kong, Shenzhen
[3]Questrom School of Business, Boston University
cc8029@nyu.edu, chenyilun@cuhk.edu.cn, pqian@bu.edu

First version: October 20, 2023
This version: June 23, 2025

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 in which 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. We leverage this framework to optimize the design of the mechanism and present our main results: As the number of servers increases, the proposed mechanism incentivizes *complete* resource pooling—that is, the system dynamics and performance under our mechanism match those under centralized control. Finally, we show that our mechanism achieves the first-best performance even when helping others incurs higher job processing costs and remains nearly optimal in settings with heterogeneous servers.

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.[1] 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.[2]

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

[1] 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.

[2] We assume homogeneous jobs and servers in our base model as a first step toward designing decentralized computing markets. In Section 7.2, we extend our mechanism and analysis to a 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 assuming complete information. 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—that is, 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 externally observing the congestion level of a hospital can be challenging. Therefore, designing an appropriate mechanism to provide the right incentive is essential for enhancing system efficiency and achieving the desired cooperation level.

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) serve 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 anytime to serve its jobs at the cost of one token per requested job. 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, which a smart contract can achieve. 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.

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 ϕ 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 ϕ, it constitutes an approximate Nash equilibrium when all the servers

in the original problem follow the FMFE strategy. 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 probability ϕ, which is the key element of our mechanism. We show that the optimal value of ϕ equals λ, the job arrival rate at each server.[3] 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.

Furthermore, we demonstrate via numerical results that complete resource pooling is an approximate equilibrium when the token-earning probability ϕ equals λ 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 two settings: one with higher costs for serving jobs from other queues, and another with heterogeneous servers. In the first setting, we demonstrate that with a proper choice of the token-earning probability ϕ, our mechanism continues to achieve the first-best performance and to maintain the same system dynamics as those under centralized control. In the second setting, we show that by setting the token-earning probability ϕ to be the largest utilization factor ρ_i among the servers,[4] the total number of jobs in the system is, at most, a constant multiple of the total number of jobs in the centralized setting, thereby nearly achieving complete resource pooling.

The rest of the paper is organized as follows. Section 1.2 reviews some related work. Section 2 formulates the problem. 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 can be achieved under a proper choice of the token-earning probability ϕ, 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. Moreover, in Section 6.3, we show via numerical results that complete resource pooling is an approximate equilibrium under our mechanism, even for small mar-

[3] Recall that we normalize the servers' processing rates to be one.

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

kets with only a few servers. Section 7 extends our mechanism and performance analysis to more general settings, including those with higher costs of serving jobs from other queues (Section 7.1) and another with heterogeneous servers (Section 7.2). We show that our mechanism continues to achieve the first-best performance in the former setting and nearly so in the latter setting. Section 8 concludes.

1.2 Related Literature

Power of Resource Pooling Resource pooling improves system efficiency in various applications, such as 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 works 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 expected rates of earning and spending tokens need to be equal in the long-run average. 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, such as 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, such as Anily and Haviv (2010), Anily and Haviv (2014), and Karsten et al. (2015). These works 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 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 works 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 $[a:b] = \{a, a+1, \ldots, b-1, b\}$ 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 Model Formulation

We consider a continuous-time model with N servers indexed by i. 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. Let $Q_i(t)$ denote the queue length (i.e., the number of jobs) of server i at time t. Job processing processes are modeled by a Poisson clock model, following Tsitsiklis and Xu (2013) and Spencer et al. (2014), to simplify the analysis. Specifically, each server receives a stream of "capacity units" following an independent Poisson process, with a rate normalized to one.[5] When a capacity unit arrives at server i at time t, the server must immediately take an action $Y_i(t)$ from one of the following three options $\{0, 1, \varnothing\}$ to allocate the capacity unit: $Y_i(t) = 0$, which indicates that server i will use the capacity unit to serve one of its own jobs; $Y_i(t) = 1$, which indicates that the capacity unit will be used to serve one of another server's jobs; and $Y_i(t) = \varnothing$, which indicates that the capacity unit will be wasted. The served jobs leave the system immediately.

Motivated by the applications discussed in Section 1, we consider a setting in which each server is unaware of the other servers' queue lengths or actions and does not know the exact total number of servers N. 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 have arrived by time t, respectively. The process $N_i^c(t)$ is divided into three subprocesses, $L_i(t)$, $T_i(t)$, and $I_i(t)$, such that $N_i^c(t) = L_i(t) + T_i(t) + I_i(t)$, where $L_i(t)$ counts the number of capacity units used to serve server i's own jobs, $T_i(t)$ counts the number of capacity units used to serve other servers' jobs, and $I_i(t)$ counts the number of wasted capacity units, all by

[5] Although Tsitsiklis and Xu (2013) and Spencer et al. (2014) referred to these capacity units as "service tokens," we use a different name to avoid any possible confusion with the artificial currency tokens that we will introduce later as part of our mechanism.

time t. Let w_{ij} denote the amount of time in which the j-th job that arrives at server i stays in the system.

The cost structure is defined as follows. Both serving and storing jobs incur costs. Serving a job costs $c \geq 0$, and each job waiting for processing costs one per unit of time. Servers operate in self-interest, and each server's objective is to minimize its expected time-average total cost—that is, its job processing cost plus its holding cost—over an infinite horizon, as follows:

$$\limsup_{t\to\infty} \frac{1}{t} V_i(t)\,, \qquad \text{where } V_i(t) \triangleq \mathbb{E}\left[\sum_{j\leq N_i^\lambda(t)} w_{ij} + c\cdot\big(L_i(t)+T_i(t)\big)\right].$$

We remark that the waiting time $\{w_{ij}\}$ depends on the strategies of all the servers.

Note that $T_i(t)$ is divided into $N-1$ processes $T_{ij}(t)$ $(\forall j \neq i)$ such that $T_i(t) = \sum_{j\neq i} T_{ij}(t)$, where $T_{ij}(t)$ counts the number of capacity units that arrive at server i and are used to serve a job at server j by time t. The dynamics of server i's queue length is as follows:

$$Q_i(t) = Q_i(0) + N_i^\lambda(t) - L_i(t) - \sum_{k\neq i} T_{ki}(t)\,.$$

2.1 Benchmark: Centralized Control with Full Information

We use the centralized, full-information setting as a benchmark. In this scenario, a central planner has full control over the servers and aims to minimize the expected time-average total cost of the entire system. The planner determines the action $Y_i(t)$ when a capacity unit arrives at server i at time t. Particularly, the planner decides which queue the capacity unit should serve, provided it will not be wasted. Since the servers are stochastically identical, this is equivalent to minimizing the expected time-average cost per server.

It is straightforward to show that any work-conserving policy (i.e., a policy that never wastes a capacity unit if there are still jobs in any of the queues) would minimize the objective. Under such a policy, the dynamics of the total number of jobs in the system behave as an $M/M/1$ queue, with the job arrival and processing rates being $N\lambda$ and N, respectively. Therefore, the expected total number of jobs in the stationary distribution is a constant of $\frac{\lambda}{1-\lambda}$. As a result, the time-average cost per server is as follows:

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

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.[6] This improves upon the following equation:

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

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

2.2 A Novel Mechanism for a Decentralized, Limited Information Setting

In the absence of a central planner, a strategic server may act as a free rider and lack the incentive to help others due to the presence of job processing costs. Moreover, since each server's states and actions are private information, punishing servers for free riding is impossible. 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 will show later that such a mechanism encourages server cooperation and ensures the achievement of the first-best time-average cost $V_c(N)$ in large systems.

Overview of the Mechanism In our mechanism, we expand the action set of each server by allowing it to request help for the jobs in its queue at any time, in addition to deciding how to use each incoming capacity unit. The requests and provisions of help are matched through a shared pool (to be introduced later in this section) in a first-come-first-serve (FCFS) order. To mitigate free riding, we introduce a token system. Specifically, servers possess artificial currency tokens. Requesting help requires one token, whereas a server can earn tokens by offering help.

Servers' Actions Each server makes strategic decisions about requesting help from or providing help to other servers. First, every server $i \in [N]$ can request help from other servers to serve its jobs at any time t. We let $R_i(t) \in [0 : Q_i(t)]$ denote the number of jobs for which the server requests help at time t. The mechanism enforces that requests are irrevocable. Once requested, the $R_i(t)$ jobs are relocated from server i's queue to the shared pool. The server is informed once a requested job is served. Note that the server that requests help bears the waiting cost incurred in the shared pool. On the other hand, whenever a capacity unit arrives at server i at time t, server i must make an immediate decision $Y_i(t) \in \{0, 1, \varnothing\}$ to allocate the capacity unit, as defined earlier in this section.

[6] By the "Poisson Arrivals See Time Averages (PASTA)" property, the long-run proportion of capacity units that, upon arrival, find a job in the system equals λ. Thus, the time-average job processing cost of a server equals $c\lambda$.

The Shared Pool The shared pool is a buffer that temporarily stores jobs for which servers have requested help. Whenever a server decides to use its incoming capacity unit to provide help (i.e., when $Y_i(t) = 1$), the capacity unit will be used to serve a job in the shared pool. If the shared pool contains a job, the job that has been waiting for the longest time in the pool is served. Otherwise, if the shared pool is empty, the capacity unit is wasted. Note that the shared pool can be a virtual queue rather than a physical one. The mechanism does not reveal the queue length of the shared pool to the servers.[7]

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.[8] Servers earn tokens by providing help and spend tokens to request help. The protocol for spending 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.

Based on our design, acquiring a token is independent of the success of helping others (i.e., serving a job from the shared pool). This design not only simplifies the analysis of our mechanism but also ensures that desirable equilibria are achieved, as demonstrated in our main results. Note that due to the mining process, the total number of tokens in the system is not constant. The value of the token-earning probability ϕ 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 an integer $C \in \mathbb{N}_+$ and for all $i \in [N]$ and $t \geq 0$; that is, that a server's token count is nonnegative and bounded from above by a constant C.[9]

2.3 The Stochastic Dynamic Game

The mechanism described in Section 2.2 gives rise to a complex stochastic, incomplete-information, dynamic game among servers. The perfect Bayesian equilibrium (PBE) is a standard solution

[7]However, servers can obtain partial information of the shared pool queue length in two ways. First, when a server offers help, the server knows whether the shared pool is empty because only a non-empty shared pool incurs a processing cost. Second, servers receive notifications when a job relocated to the shared pool gets served; this allows the servers to gather information about the shared pool based on the status of relocated jobs. We will formally take these into account when analyzing the equilibrium.

[8]The specific value is not essential when considering the time-average-cost objective.

[9]The introduction of C is mainly for technical reasons, and we specify its value in Section 6.

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 (i.e., history) to decisions on requesting/providing help or serving its jobs. Additionally, each server must form beliefs about the state of the system. In this section, we formally define the PBE, discuss its shortcomings, and motivate the need for an approximate equilibrium concept.

We denote the history of server i at time t as $h_i(t)$. This history includes the time when jobs and capacity units arrive at server i up to time t, the time the server requests help, and the time when jobs for which server i requested help are served. Additionally, it tracks which capacity units are used to serve its own queue, which are used to serve the shared pool, and whether these capacity units are wasted due to an empty shared pool.

The state of server i at time t includes its queue length and token amount, which is denoted by $x_i(t) \triangleq \big(Q_i(t), S_i(t)\big)$. Since servers are stochastically identical, server i interacts with the other $N-1$ servers only through the empirical distribution of their states, denoted by $\Pi_{-i}(t)$. Finally, denote the empirical distribution of all server states at time t by $\Pi(t)$, and the shared pool queue length at time t by $Q_0(t)$. The state of this stochastic game at time t is $\big(\Pi(t), Q_0(t)\big)$.

Since server i has incomplete information about the system state, it forms a belief about the system state based on its history. We denote the corresponding posterior distribution at time t as $p(\cdot|h_i(t))$. A strategy δ_i for server i is a mapping that, at time t, specifies an action based on its own state $x_i(t)$ and its belief about the system state $p(\cdot|h_i(t))$. This action determines whether to request help or how to use a capacity unit if one arrives at time t. Let $\boldsymbol{\delta}$ denote the strategy profiles of all servers. We assume that server i seeks to minimize its expected time-average cost, defined as follows:

$$\limsup_{s\to\infty} \frac{1}{s} V_i\big(s|\boldsymbol{\delta}, h_i(t)\big)\,, \quad \text{where} \quad V_i\big(s|\boldsymbol{\delta}, h_i(t)\big) \triangleq$$
$$\mathbb{E}_{(\Pi(t),Q_0(t))\sim p(\cdot|h_i(t))}\left[\mathbb{E}\left[\sum_{j=N_i^\lambda(t)}^{N_i^\lambda(t+s)} w_{ij} + c\cdot\big(L_i(t+s)-L_i(t)+T_i(t+s)-T_i(t)\big)\,\middle|\, \boldsymbol{\delta}, x_i(t), \Pi(t), Q_0(t)\right]\right].$$

Here, the inner expectation is taken with respect to the future randomness given the current state of the game, while the outer expectation is taken with respect to the server's belief $p(\cdot \mid h_i(t))$ about the system state based on its history.

The strategy profiles $\{\delta_i\}_{i\in[N]}$ and belief systems $\{p_i\}_{i\in[N]}$ form a PBE if the following holds. First, given history $h_i(t)$ and the strategies of all other servers $\{\delta_j\}_{j\neq i}$, server i's strategy should

minimize its expected time-average cost defined above. Second, the belief held by server i must 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 over history-dependent policies to determine the best responses. Since servers maintain beliefs about the states of all other servers in PBE, the dimension of the optimization problem grows very quickly as the number of servers increases, making equilibrium strategies intractable both analytically and computationally. Second, servers must maintain and update their beliefs about the current state and future evolutions of the system; such a level of sophistication imposes a stringent rationality assumption on the servers, especially when the number of servers is large. 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, which we formally introduce in the next section.

3 The Fluid Mean-Field Equilibrium (FMFE)

In this section, we introduce FMFE as an alternative equilibrium concept for the stochastic game induced by our token mechanism. This approximation relies on the following key assumptions: First, given the large number of servers in the system, each server assumes that the shared pool queue length distribution is stationary and uncorrelated over time; and second, we relax the non-negativity constraint on the token amount to a long-run break-even constraint. We define the mean-field and fluid approximations in Section 3.1, and FMFE, in Section 3.2.

3.1 Mean-Field and Fluid Approximation

A key feature of our token mechanism is that servers interact with each other only through the shared pool. When deciding whether to request help, a server needs 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, which would allow it to earn a token without incurring the cost of serving a job from the shared pool. The complexity of a server's decision problem within PBE (defined in Section 2.3) arises from the need to maintain and update beliefs about the current and future trajectories of the shared pool queue length $Q_0(t)$

given the server's past interactions with the shared pool and its belief about the states of other servers.

However, when the number of servers is large, it is reasonable to assume that the fluctuations of the servers' states average out and that their empirical distribution remains roughly constant over time. Consequently, the distribution of the shared pool's queue length also remains stationary and uncorrelated over time. 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, as stated in Assumption 3.1.

Assumption 3.1 (Mean-Field Approximation)**.** Each server $i \in [N]$ assumes that the shared pool queue length is in the stationary distribution at any time t (whose probability mass function is denoted by $\Psi(\cdot)$), independent of its observed history $h_i(t)$.

The mean-field approximation simplifies a server's problem in the following two ways:

1. The expected waiting time of a job in the shared pool is a constant $w \geq 0$, which is independent of the history. The value of w will be determined endogenously by an equilibrium.
2. The probability that the shared pool is non-empty is also a constant $\Psi(0)$, which is independent of the history. Moreover, this value turns out to be ϕ, which is the probability of earning a token when offering help. We elaborate on this 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 time) equals ϕ. To see this, note that it is optimal for each server to equalize the rates of earning and spending tokens.[10] Under our mechanism, the rate of spending tokens equals the rate of requesting help, and the rate of earning tokens is ϕ times the rate of offering help. Therefore, at each server, the rate of requesting help is ϕ times the rate of offering help (as token-earning and token-spending rates are equal). Consequently, the rate at which jobs relocate to the shared pool (which equals the total requesting help rates across servers) is ϕ times the rate at which capacity units join the shared pool (which equals the total offering help rates across servers). Therefore, the probability that a capacity unit contributed to the shared pool serves a job is precisely ϕ in the long-run average.

[10]This is because a token holds value only when used, and earning tokens can incur costs; therefore, there is no benefit in accumulating more tokens than necessary.

Simplification of Mean-Field Approximation Under Assumption 3.1, a server's optimal strategy is no longer history-dependent, as the history's only purpose is to maintain a belief about the shared pool queue length. Instead, the optimal strategy depends only on a server's individual state $x_i(t) = \big(Q_i(t), S_i(t)\big)$—which includes its queue length and token amount—as well as the stationary distribution of the shared pool queue length through the waiting time w in the shared pool. We call such servers oblivious and their strategies oblivious strategies (following Weintraub et al. 2008). Furthermore, since the shared pool remains at its stationary distribution, 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 (i.e., assuming Assumption 3.1), 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. Let $X_i(t) \in \{0, 1\}$ denote the decision on whether to request help for the arriving job when a job arrives at time t, as follows:

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

The decisions $X_i(t)$ and $Y_i(t)$ defined in Section 2 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 a capacity unit arrives according to Lemma 3.1, 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.[11] With some abuse of notation, we instead let t be the index of the discrete periods. The decisions in period t are as follows: (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 $Y_i(t) = \varnothing$ if the server opts to be idle and wastes the unit. The time-average-cost optimization problem solved by server i in the mean-field approximation when the expected waiting time in the

[11] 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).

shared pool is w can be expressed in (1),

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

In (1), we denote by Π the set of all non-anticipative policies (i.e., 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 ϕ, 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 the queue of server i in period t (note that the length of a period is $\frac{1}{1+\lambda}$ in expectation), and ϕ 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 (i.e., the queue length Q_i and number of tokens S_i) 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 to go beyond the upper bound C, and we require only that the tokens satisfy the flow balance constraint, that is, that the expected rates of earning and spending tokens be equal in the long-run average. We provide the fluid relaxation of (1) in (2)

following the same notation as in (1).[12]

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

In (2), we replace the second-to-the-last constraint in (1) with the first constraint in (2), which 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 of (1). Lemma 3.2 shows that (2) can be formulated as a one-dimensional DP in which 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 stochastic dynamic programming problem with a one-dimensional state space, where an optimal stationary 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. Lemma 3.2 substantially simplifies a server's problem. 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 policy of the fluid mean-field problem provides 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). Formally, we have the following definition.

[12] In addition, we provide further theoretical and numerical justification for the fluid relaxation in Appendix E.

Definition 3.1 (Fluid Mean-Field Equilibrium)**.** Let w be the steady-state expected waiting time in the shared pool, and let π denote a stationary policy for the server's decision problem (2). We say the pair (π, w) constitutes an FMFE if:

1. The policy π is an optimal solution to $V^{\mathrm{F}}(w)$, given the steady-state expected shared pool waiting time w; and
2. The steady-state waiting time in the shared pool is w when all of the N servers follow the policy π.

Mathematically, we define the mapping:

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

The pair (π, w) is an FMFE if and only if $w \in H(w)$.

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

$$m(z) = \frac{1}{1-\lambda} \cdot \Big(z - \sum_{i=1}^{z} \lambda^i\Big), \ \forall\, z \in \mathbb{N}\,. \tag{3}$$

We can interpret $m(z)$ as a threshold for the shared pool waiting time w, above which the server

would maintain a queue longer than z. For more details, please refer to Appendix A.3. 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 $\phi \in \left(\lambda^{k+1}, \lambda^k\right]$, and denote the queue length by q. 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 $w \in \big(m(z), m(z+1)\big)$ 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 δ (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 ϕ and (endogenous) shared pool waiting time w. Specifically, a server requests help only when its queue length exceeds a certain threshold (i.e., the value of k in Case One and $z \geq k$ in Case Two) and offers help only when its queue is empty. The threshold parameter $k = \left\lfloor \frac{\ln \phi}{\ln \lambda} \right\rfloor$ in Lemma 4.2 depends only on the value of $\ln \phi$ relative of $\ln \lambda$. When ϕ is small, it is costly to earn tokens, so a server is conservative in spending tokens; consequently, the threshold parameter k decreases in ϕ. On the other hand, when w increases, the

benefit of requesting help decreases. Notably, when w surpasses $m(k)$ (Case Two), 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 $w \in \big[m(z), m(z+1)\big]$ 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 $\phi \in \big[\lambda^{k+1}, \lambda^k\big]$.*[13]
   (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 sets ϕ equal to λ, it is each server's best response to implement complete resource

[13] When $\phi = \lambda^{z+1}$ for some integer $z \geq 1$, letting k be either z or $z+1$ yields the same policy.

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 (even slightly). Thus, complete resource pooling is an FMFE when there are many servers and the token-earning probability ϕ equals λ. We defer a more comprehensive interpretation of Corollary 4.3 to Section 5.

4.2 Characterization of FMFE

In this section, we characterize FMFE using servers' best-response strategies 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 Diminishes 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 $O\big(\frac{1}{N}\big)$.

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 induced waiting time, denoted by $\tilde{w}$, satisfies $\tilde{w} \leq \frac{M_1}{N}$ for all N and some absolute constant M_1 that depends only on the values of the job arrival rate λ, token-earning probability ϕ, and upper bound $\bar{w}$.*[14]

We prove Proposition 4.4 in Appendix A.4. 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

[14] That is, $\max H(w) \leq M_1/N$ for any $w \leq \bar{w}$, where $H(\cdot)$ is the mapping defined in Definition 3.1.

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 the fraction $\phi < 1$ of the job processing rate (see Remark 3.1). Consequently, the expected waiting time in the shared pool diminishes to zero as 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 sustain 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 ϕ equals the job arrival rate λ. 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 being $N\lambda$ 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 $\phi \in (0, 1)$, 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 ϕ

In this section, we investigate the optimal value of ϕ (that minimizes the system's total cost), which governs the token acquisition rate when a server "mines" tokens by offering help. Intuitively, if ϕ 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 ϕ 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.

Theorem 5.1 characterizes the level of system congestion (i.e., the job total in the system) as a function of ϕ.

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 ϕ. We have*

1. $\lim_{N\to\infty} Q_\Sigma(\phi)/N = q(\phi)$ *when* $\phi < \lambda$, *where $q(\phi)$ denotes the expected queue length of a server in the stationary distribution. Notably, $q(\phi)$ is decreasing in ϕ; the expression of $q(\phi)$ is visualized in Figure 1b and provided in (12) in the Appendix.*
2. $Q_\Sigma(\phi) = \frac{\phi}{1-\phi}$ *when* $\phi \geq \lambda$.

Proof. 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. In addition, it forms an FMFE when all servers follow the 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 ϕ varies. To see this, let $R(\phi)$ and $P(\phi)$ denote the steady-state rates of requesting help and offering help, respectively, and $q(\phi)$ denote 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 (12) in the Appendix. When $\phi \leq \lambda$, both the requesting and offering help rates increase with ϕ, while the expected queue length of a server decreases with ϕ. Intuitively, as the token-earning probability ϕ 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 ϕ is strictly less than λ, 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.

When the token-earning probability ϕ equals the job arrival rate λ, 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 λ, 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 being $N\lambda$ 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 λ/ϕ of capacity units

to serve the shared pool, while being idle for the rest of the time. Therefore, as the token-earning probability ϕ increases, the rate of requesting help remains constant (which equals the job arrival rate λ) whereas the rate of offering help is λ/ϕ, which decreases with ϕ. 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 ϕ for any problem instance. Specifically, when $\phi \geq \lambda$, 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 $N\lambda/\phi$, respectively. Therefore, the total number of jobs in the system is $\frac{\phi}{1-\phi}$ in expectation, which increases with ϕ. □

According to Theorem 5.1, the optimal value of the token-earning probability ϕ equals the job arrival rate λ. We state this fact formally in Theorem 5.2. Theorem 5.2 directly follows Theorem 5.1; hence, we omit the proof here.

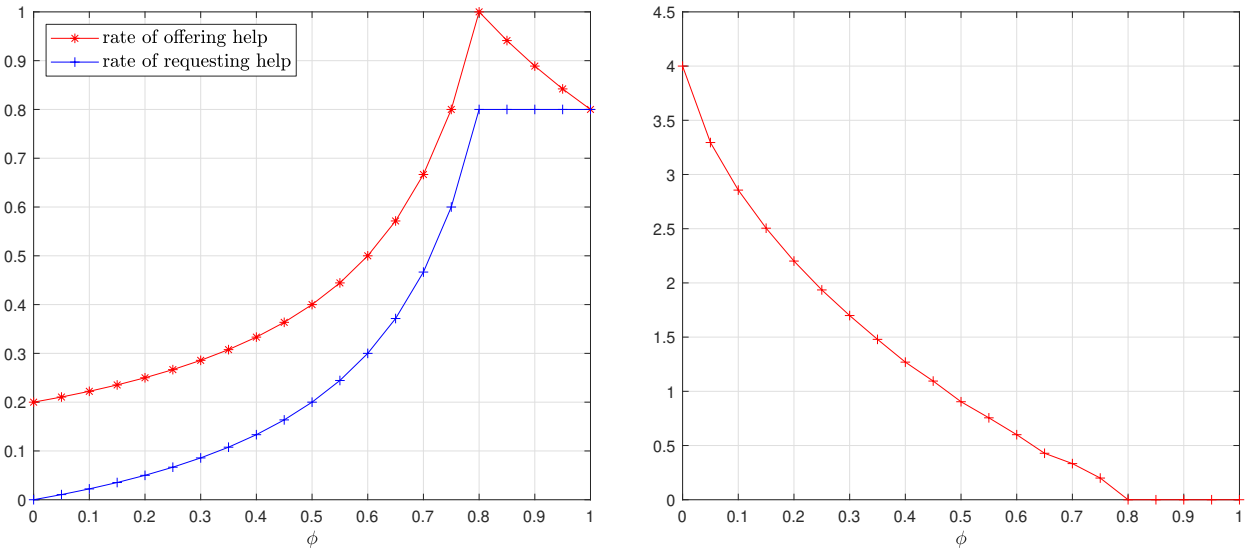
Theorem 5.2. *The optimal value of the token-earning probability ϕ equals the job arrival rate λ when the number of servers N is large. In this case, complete resource pooling is an FMFE, and the system's dynamics and performance under the token-based mechanism match those under centralized control.*

6 FMFE Strategy as a Near-Optimal Best Response

In this section, we justify the use of the FMFE notion introduced in Section 3 for analyzing our 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 ϕ equals its optimal value, which is the job arrival rate λ, 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 holds at time t must 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 only when its token count is below the upper bound value of C. As a result, the token count is always



(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 ϕ times the offering help rate, i.e., is $\phi \cdot \frac{1-\lambda}{1-\phi}$ for $\phi \in [0, \lambda]$ and a constant λ for $\phi \in [\lambda, 1]$.

within the range of zero to C. We denote by π^{F} the FMFE strategy described in Corollary 4.3 and by $\bar{\pi}^{\mathrm{F}}$ the modified version of π^{F} that ensures feasibility in the original problem.

Note that both the policies π^{F} and $\bar{\pi}^{\mathrm{F}}$ are feasible to the fluid mean-field problem $V^{\mathrm{F}}(w)$ defined in (2). Moreover, the policy π^{F} is optimal to $V^{\mathrm{F}}(w)$ when the shared pool waiting time $w \leq 1$ by Corollary 4.3. The two policies π^{F} and $\bar{\pi}^{\mathrm{F}}$ 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 $\bar{\pi}^{\mathrm{F}}$ decays to zero at a rate of $O\big(\frac{1}{C}\big)$. Thus, the dynamics of policy $\bar{\pi}^{\mathrm{F}}$ in the original problem converge to the dynamics of policy π^{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 $\bar{\pi}^{\mathrm{F}}$. We have*

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

for some constant M_2 that depends only on the values of λ and ϕ.

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 $\bar{\pi}^{\mathrm{F}}$. 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 π^{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 λ and ϕ.

Finally, analogous to Proposition 4.4, if all servers follow the FMFE policy $\bar{\pi}^{\mathrm{F}}$ in the original problem, the long-run average waiting time in the shared pool diminishes to zero at a rate of $O\big(\frac{1}{N}\big)$ 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 $\bar{\pi}^{\mathrm{F}}$ 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 λ and token-earning probability ϕ and is independent of the number of servers N or the token-amount 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 λ and token-earning probability ϕ and is independent of the token-amount upper bound value C.*

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 $\bar{\pi}^{\mathrm{F}}$ 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 π^{F}. We remark that $c\lambda+\mathbb{E}\big[Q^{\mathrm{F}}\big]$ is the optimal long-run average total cost in the fluid mean-field problem when the shared pool waiting time $w=0$; that is, $(1+\lambda)\cdot V^{\mathrm{F}}(0)=c\lambda+\mathbb{E}\big[Q^{\mathrm{F}}\big]$. We first show in Lemma 6.4 that, if server one in the original problem also follows the FMFE strategy $\bar{\pi}^{\mathrm{F}}$, its time-average total cost surpasses $c\lambda+\mathbb{E}\big[Q^{\mathrm{F}}\big]$ by only $O\left(\frac{1}{N}+\frac{1}{C}\right)$.

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

We prove Lemma 6.4 in Appendix C.4. Suppose server one also follows the FMFE policy $\bar{\pi}^{\mathrm{F}}$. Intuitively, when there are many servers, the waiting time in the shared pool is nearly zero and on the order of $O\left(\frac{1}{N}\right)$ (Proposition 6.3). Additionally, when the token-amount upper bound C is large, the expected queue length under policy $\bar{\pi}^{\mathrm{F}}$ is close to that under policy π^{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 $c\lambda+\mathbb{E}\big[Q^{\mathrm{F}}\big]$, 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 $c\lambda+\mathbb{E}\big[Q^{\mathrm{F}}\big]-O\big(\frac{1}{N^{1-\delta}}\big)$ for any $\delta>0$.

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

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, in which 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 the 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 and 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. More-

over, in the stationary distribution, the shared pool is non-empty with a probability that differs from ϕ by at most $O\big(\frac{1}{N^{1-\delta}}\big)$. This implies that the optimal time-average cost in the relaxation is at least the optimal value of the fluid mean-field problem (which equals $c\lambda+\mathbb{E}[Q^{\mathrm{F}}]$) less than a small term that is $O\big(\frac{1}{N^{1-\delta}}\big)$.

Intuitively, as the number of servers increases, the queue length of the shared pool transitions to the steady state quickly. However, the inter-arrival time of jobs or capacity units at server one does not scale with the number of servers. 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 let the token-amount upper bound be $C = \Omega\big(N\big)$, the benefit from deviating from the FMFE strategy is only a negligible term that is nearly $O\big(\frac{1}{N}\big)$. 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 distribution of 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 and Appendix E). 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 ϕ equals λ (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 it is aware that the shared pool is empty, as this presents an opportunity to earn a token without incurring the cost of serving a job from the shared pool.[15]

However, the optimal strategy of server one is challenging to solve. Particularly, server one can only make inferences about the shared pool based on the partial information obtained from its interactions with the shared pool.[16] 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 empower server one to have complete information about the shared pool so that it can make decisions based on the queue length of the shared pool. 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, thereby 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.

[15] 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.

[16] Please refer to the description in footnote 7.

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 λ and N, we evaluate (a) the time-average total cost V^{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^{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 $\frac{V^{\mathrm{CRP}} - V^{\mathrm{OPT}}}{V^{\mathrm{OPT}}}$, 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 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 Extensions

In this section, we extend our mechanism and performance analysis to more general settings, including higher costs for serving jobs from other queues (Section 7.1) and heterogeneous servers (Section 7.2). We show that our mechanism continues to achieve the first-best performance in the former setting and approximately so in the latter setting, under mild regularity conditions on the

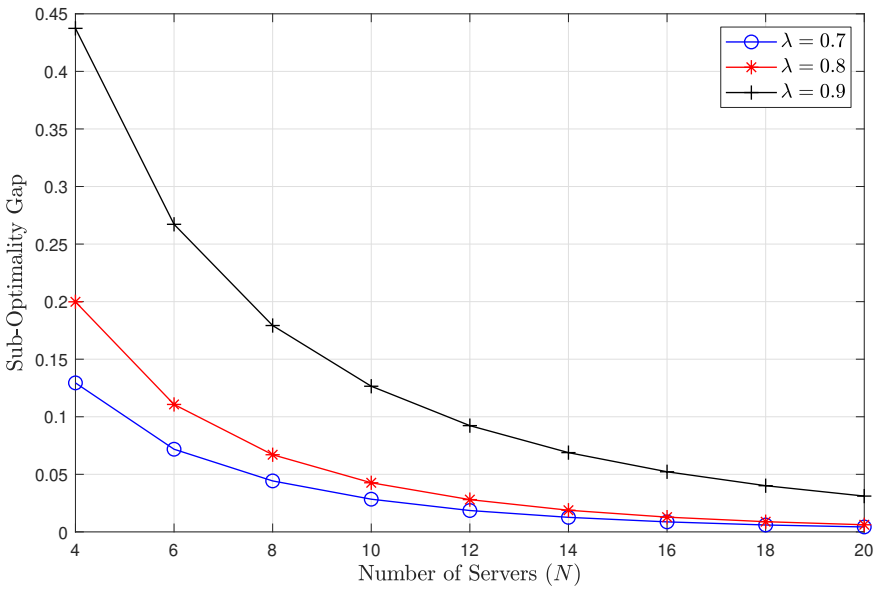


Figure 2: Complete resource pooling versus the best response.

problem primitives.

7.1 Extension to Higher Costs of Serving Jobs of Other Queues

In our base model, a server's cost of serving a job is the same whether it originates from its own queue or from other queues. In practice, one might expect jobs arriving at a particular server to prefer that server over others, and this preference can be reflected in the cost of serving the job. In this section, we consider a more general setting in which, for each server, serving a job from its queue costs $c \geq 0$, whereas serving a job from other queues costs $c' \geq c$. Let $\Delta c \triangleq c' - c \geq 0$ denote the increased cost of serving jobs from other queues. In the following, we first characterize the optimal policy in the centralized setting when the number of servers N is large. Then, we demonstrate that for any value of $\Delta c \geq 0$, with a proper choice of the token-earning probability ϕ, our mechanism continues to achieve the first-best performance and maintains the same system dynamics as those under centralized control when the number of servers is large.[17]

7.1.1 Optimal Control for Centralized Setting

In the centralized setting, a central planner has complete control over the servers and minimizes the system's total cost (i.e., the holding cost plus processing cost). Since servers are stochastically identical, it is equivalent to minimizing the expected time-average total cost per server. To achieve

[17]We thank the Associate Editor for suggesting this valuable extension.

this goal, the planner would implement a cutoff policy with a threshold $z \in \mathbb{N}$ (as justified in Appendix C.6). Specifically, when a job arrives, a server relocates the job to the shared pool when it has z jobs or more; otherwise, it adds the job to its queue to process on its own later. Additionally, when a capacity unit arrives, the server serves a job from its queue if it is not empty and, otherwise, offers help to the shared pool.

Remark 7.1. From Proposition 4.4, the expected waiting time in the shared pool, denoted by w, diminishes to zero at a rate of $w = O\left(\frac{1}{N}\right)$ for any cutoff policy. However, the inter-arrival time between a job and a capacity unit at a server does not scale with the number of servers. Consequently, in the long-run average, the proportion of jobs served by a different server among those transferred to the shared pool approaches one as the number of servers N increases.

We now identify the optimal value of $z \in \mathbb{N}$ that minimizes the system's total cost when the number of servers N is large. We assume that the waiting time in the shared pool w is zero and that serving a job from the shared pool costs $c' = c + \Delta c \geq c$, which is approximately true when N is large by Remark 7.1.

Suppose that the planner uses a cutoff policy with a threshold $z \in \mathbb{N}$. Consequently, a server's queue length q remains in the interval $[0 : z]$ in the long run, and the stationary distribution of the queue length, denoted by $\pi_i \triangleq \mathbb{P}[q = i]$, is $\pi_i = \pi_0 \cdot \lambda^i$ for $i \leq z$ with $\pi_0 = \frac{1-\lambda}{1-\lambda^{z+1}}$. Therefore, letting $v(z)$ denote the long-run average total cost of a server, we can express $v(z)$ as follows:

$$v(z) = \sum_{i=0}^{z} \pi_i \cdot i + c\lambda + \Delta c \cdot \pi_z \cdot \lambda,$$

where the first term represents the expected queue length and corresponds to the holding costs for jobs in the queue, and the last two terms correspond to the job processing costs. To elucidate the latter two terms, note that by symmetry, every server serves jobs at a rate of λ, among which a rate of $\pi_z\lambda$ jobs (equal to each server's rate of routing jobs to the shared pool) comes from the shared pool and incurs an extra processing cost Δc per job. Note that

$$v(z) - v(z-1) = \frac{\lambda^z(1-\lambda)^2}{(1-\lambda^z)(1-\lambda^{z+1})} \cdot \Big(m(z) - \Delta c\Big),$$

where $m(z)$ is the function defined in (3). Therefore, the optimal threshold z that minimizes $v(z)$, which we denote by $\bar{q} \in \mathbb{N}$, equals the largest integer z such that $m(z) \leq \Delta c$. In particular, when $\Delta c = 0$, $\bar{q} = 0$ because $m(0) = 0$; that is, servers relocate all jobs to the shared pool and pool service power to serve jobs, corresponding to the centralized control (i.e., complete resource pooling) in

Section 2.1. We summarize the above in Theorem 7.1 and prove the optimality of a cutoff policy in Appendix C.6.

Theorem 7.1. *When the number of servers N is large, the optimal policy in the centralized control is a cutoff policy with a threshold value of $\bar{q}$ equal to the largest integer z such that $m(z) \leq \Delta c$, with the function $m(z)$ defined in (3).*

7.1.2 Token-Based Mechanism for Decentralized Setting

We now analyze our token-based mechanism in the decentralized setting. First, Remark 3.1 still holds, and the definition of the fluid mean-field problem remains unchanged, except that serving a job from the shared pool now costs $c' \geq c$.[18] Lemma 7.2 characterizes a server's best response in the fluid mean-field problem, by showing that it mirrors the best response in the base model (Lemma 4.2), with $w + \Delta c$ substituting for w.

Lemma 7.2. *Lemma 4.2 continues to hold with $w + \Delta c$ replacing w.*

We prove Lemma 7.2 in Appendix C.7. Intuitively, the tokens' flow balance constraint ensures that the rate of jobs routed to the shared pool (equal to the token spending rate) matches the rate at which the server processes jobs from the shared pool (equal to the token earning rate). This implies that every job relocated to the shared pool incurs an additional processing cost Δc (compared to serving the job on its own). As a result, the cost of requesting help now comprises the shared pool waiting cost w plus the additional cost Δc.

Optimal Value of ϕ Let the token-earning probability be $\phi = \lambda^{\bar{q}+1}$. From Lemma 7.2, if $w + \Delta c < m(\bar{q}+1)$, which translates to $w < \underline{w}$ with $\underline{w} \triangleq m(\bar{q}+1) - \Delta c > 0$,[19] a server's best response is a cutoff policy with threshold $\bar{q}$, which aligns with the optimal control in the centralized setting in Theorem 7.1. Additionally, all servers following the cutoff policy with a threshold of $\bar{q}$ forms an FMFE when the number of servers N is large, because the shared pool waiting time w diminishes to zero as N increases by Proposition 4.4. We summarize these in Theorem 7.3, which extends Theorem 5.2.

[18] This is analogous to Remark 7.1. Specifically, as will become evident, the expected waiting time in the shared pool diminishes to zero as the number of servers increases by Proposition 4.4. As a result, in the long-run average, the proportion of jobs served by a different server among those transferred to the shared pool approaches one as the number of servers N increases.

[19] Recall that $\bar{q}$ represents the largest integer z such that $m(z) \leq \Delta c$. Therefore, $m(\bar{q}+1) > \Delta c$.

Theorem 7.3 (Extension of Theorem 5.2)**.** *The optimal value of the token-earning probability ϕ equals $\lambda^{\bar{q}+1}$ when the number of servers N is large. In this case, all servers following the cutoff policy with a threshold of $\bar{q}$ is an FMFE, and the system's dynamics and performance under the token-based mechanism match those under centralized control.*

Finally, we remark that Lemmas 6.4 and 6.5 continue to hold with the same proofs, except that the constants involved also depend on the cost increase Δc. In other words, playing an FMFE strategy when all other servers follow the FMFE strategy is a near-optimal best response in large markets with many servers.

7.2 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. As in the base model, we assume that a server's cost of serving a job is $c \geq 0$, which is independent of the queue from which the job originates. We further 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 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 $\phi = \bar{\rho} \geq \rho_i$, from 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 of the 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 $\rho_m \triangleq \bar{\rho}$. As a result, 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 $O\big(\frac{1}{N}\big)$ as the number of servers N increases. Therefore, if all of the servers follow the FMFE strategy π^{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 $\frac{\bar{\rho}(1-\rho)}{\rho(1-\bar{\rho})}$ 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 though servers may have varying job arrival and processing rates. We summarize these in Theorem 7.4, which extends Theorem 5.2 for the case of homogeneous servers.

Theorem 7.4 (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 π^{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 depicted 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 C of the token amount. In the fluid problem, servers two to N follow the FMFE policy π^{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 is equivalent 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 7.5, analogous to Lemmas 6.4 and 6.5, it is a near-optimal best response for server one to follow the policy π^{F} in the fluid problem. The proof of Theorem 7.5 parallels the proofs of Lemmas 6.4 and 6.5, and we provide it in Appendix C.8.

Theorem 7.5 (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 π^{F}. We have the following results.*

1. *(Extension of Lemma 6.4) If server one also follows policy π^{F}, its time-average total cost is upper-bounded by $c\lambda_1 + \frac{\bar{\lambda}\bar{\rho}}{\underline{\lambda}(1-\bar{\rho})\cdot N}$.*
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 $M_9\big(\lambda,\bar{\rho},\delta\big)$ that depends only on the values of λ, $\bar{\rho}$, and δ.*

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

Comparison of Distribution 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 individual job processing rate μ_i. Therefore, a server with a higher value of μ_i undertakes a higher processing cost, even though its demand for computing resources, which is λ_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 small if 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 λ_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 demonstrate that our mechanism achieves the first-best performance even when helping others incurs higher job processing costs and is near-optimal with heterogeneous servers.

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.[20] 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 mechanism and analytical techniques can provide insights into the design and analysis of near-optimal mechanisms for various other decentralized applications. Finally, our mechanism introduces an artificial upper bound C on the number of tokens to ensure that each server possesses a finite expected number of tokens under our mechanism.[21] As an alternative to imposing an artificial bound, the tokens could be made perishable, with an expiration rate that decreases to zero at an appropriate rate as the number of servers N grows large. We believe this approach will also ensure that a server runs out of tokens with a diminishing probability, and thus all our analyses in Section 6 remain valid. We leave a formal analysis of this modification for future work.

Acknowledgements

We thank Rene Caldentey, Ming Hu, the area editor Omar Besbes, an anonymous associate editor, and three anonymous reviewers for their valuable comments that greatly improved the paper. An extended (one-page) abstract of an earlier version of this work appeared in the proceedings of the

[20] This is because a planner may intentionally make a server wait for a better-fit job.

[21] Please refer to Section 2.2, particularly footnote 9 there.

Twenty-Fifth ACM Conference on Economics and Computation (EC'24).

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., Nedic, A. and Ozdaglar, A. (2003), *Convex analysis and optimization*, Athena Scientific.

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'. Available at `https://sites.duke.edu/toddsarver/files/2021/07/Micro-Lecture-Notes.pdf`.

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 ϕ, while incurring a cost of c (for serving a job from the shared pool) also with probability ϕ; 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 ϕ, 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 stochastically 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. Finally, 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).

Therefore, a server requests help only when a job arrives (scenario (i)). In this case, the server would request help only 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

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

where $\mathbb{P}\big(X=1\big|q\big) = \sum_{s\in\mathbb{Z}} p(q,s)\cdot\mathbb{P}(X=1|q,s)/p(q)$ denotes the marginal probability of taking the action $X=1$ when there are q jobs and a new job arrives, and $\mathbb{P}\big(Y=0\big|q\big)$ and $\mathbb{P}\big(Y=1\big|q\big)$ are defined analogously as the marginal probability of taking the actions $Y=0$ and $Y=1$ when there

are q jobs and a capacity unit arrives. Similarly, we can express the tokens' flow-balance constraint in (2) as

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

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 $\mathbb{P}\big(q,X\big) = \frac{\lambda}{1+\lambda}\cdot p(q)\cdot\mathbb{P}\big(X\big|q\big)$ and $\mathbb{P}\big(q,Y\big) = \frac{1}{1+\lambda}\cdot p(q)\cdot\mathbb{P}\big(Y\big|q\big)$ denote the joint probabilities of having q jobs and taking the action of X and Y, respectively; we can rewrite (2) as a linear program (4).

$$\begin{aligned}
V^{\mathrm{F}}(w) = \min_{p(q),\mathbb{P}(q,X),\mathbb{P}(q,Y)\geq 0}\quad & \sum_{q\in\mathbb{N}}\left\{\frac{q}{1+\lambda}\cdot p(q) + w\cdot\mathbb{P}\big(q,X=1\big) + c\cdot\Big(\mathbb{P}\big(q,Y=0\big)+\phi\cdot\mathbb{P}\big(q,Y=1\big)\Big)\right\}\\
\text{s.t.}\quad & \sum_{q\in\mathbb{N}}\mathbb{P}\big(q,X=1\big) = \sum_{q\in\mathbb{N}}\phi\cdot\mathbb{P}\big(q,Y=1\big),\\
& \mathbb{P}\big(q,X=0\big) = \mathbb{P}\big(q+1,Y=0\big),\ \forall\, q\in\mathbb{N},\\
& \mathbb{P}\big(q,X=0\big)+\mathbb{P}\big(q,X=1\big) = \frac{\lambda}{1+\lambda}\cdot p(q),\ \forall\, q\in\mathbb{N},\\
& \mathbb{P}\big(q,Y=0\big)+\mathbb{P}\big(q,Y=1\big)+\mathbb{P}\big(q,Y=\varnothing\big) = \frac{1}{1+\lambda}\cdot p(q),\ \forall\, q\in\mathbb{N},\\
& \sum_{q\in\mathbb{N}} p(q) = 1.
\end{aligned}\tag{4}$$

In (4), the first constraint represents the flow balance constraint of the tokens, i.e., expected rates of earning and spending tokens are 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 (4). We can interpret (4) as computing an optimal stationary distribution of the queue-length-control pairs to minimize a server's long-run average cost.

A.3 Interpretation of Function $m(z)$

To interpret the function $m(z)$ in (3), we consider a server that follows a cutoff policy with a threshold value $z\in\mathbb{N}$. Specifically, when a job arrives, the server relocates the job to the shared pool when it has z jobs or more, and otherwise adds the job to its queue. Additionally, when a capacity unit arrives, the server serves a job from its queue if it is not empty, and otherwise offers help to the shared pool with some probability. With this policy, the server's queue length q remains in the interval $[0:z]$ in the long run, and the stationary distribution of the queue length, denoted by $\pi_i \triangleq \mathbb{P}\left[q=i\right]$, is $\pi_i = \pi_0\cdot\lambda^i$ for $i\leq z$ with $\pi_0 = \frac{1-\lambda}{1-\lambda^{z+1}}$.

Note that when the tokens' flow balance constraint holds (i.e., the rates of earning and spending tokens are equal), the time-average job processing cost equals a constant $c\lambda$. This is because the rate of jobs routed to the shared pool (which equals the rate of spending tokens) equals the rate of jobs the server serves from the shared pool (which equals the rate of earning tokens). Therefore, the long-run average total cost of following a cutoff policy with a threshold value z, which we denote

by $v(z)$, is:

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

In the above, the second term represents the expected queue length and corresponds to the holding costs for jobs in the queue, and the third term corresponds to the waiting costs for jobs relocated to the shared pool, because the server requests help at a rate of $\pi_z \lambda$. Note that $v(z) \leq v(z-1)$ if and only if $w \geq m(z)$. Therefore, we can interpret $m(z)$ as a threshold for the shared pool waiting time w, above which the server would maintain a queue longer than z.

A.4 Proof of Proposition 4.4

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

A.4.1 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*

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

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

We prove Lemma A.1 in Appendix A.4.6. 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 $i \in [N]$. We then bound the tail probabilities for each of the N random variables, thereby obtaining the desired result.

A.4.2 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 π_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 ϕ 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, that is,

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

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

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

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*

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

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 $A_q, S_q \le 1$. □

A.4.3 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*

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

We prove the first inequality of Lemma A.3 in Appendix A.4.7 by using a drift analysis. The second inequality follows directly from Lemma A.2.

A.4.4 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

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

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

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

we bound the two terms separately.

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

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

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

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

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

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 $N \to \infty$. 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.4.5 Step Five: Bounding the Shared Pool Waiting Time w

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

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

for some constant M_1 that depends only on the values of λ, ϕ, 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.4.6 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 by $\tilde{\mathbf{Q}}(t) = \big(\tilde{Q}_i(t)\big)_{i\in[0:N]}$, that mimics the original queue length process $\mathbf{Q}(t)$. We couple the two queue length 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,

$$\tilde{Q}_\Sigma(t) = \sum_{i=1}^{N} Z_i(t), \tag{5}$$

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*

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

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

As a result, we have

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

where the first inequality follows from stochastic dominance, the second inequality from (5), 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 $V^{\mathrm{F}}(w)$ with $w \le \bar{w} < \infty$. Since the server presumes the shared pool waiting time w to be smaller than a fixed constant $\bar{w} < \infty$, the set of all possible optimal policies is finite when the value of w varies between zero and $\bar{w}$ by Lemma 4.2. Therefore, 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., $Z_i \ge 1$). Consequently, the dynamics of $Z_i(t)$ is an $M/M/1$ queue with a job arrival rate of λ 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,

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

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

Second, suppose Case Two in Lemma 4.2 holds with $z = 0$; i.e., $\phi > \lambda$ and $m(0) = 0 \le w \le m(1)$. In this case, the server routes all jobs to the shared pool and offers help with probability λ/ϕ when a capacity unit arrives. Therefore, the dynamics of $Z_i(t)$ is an $M/M/1$ queue with a job arrival rate of λ and a job processing rate of λ/ϕ. Thus, $\mathbb{P}[Z_i(\infty) = i] = (1-\phi)\phi^i$ for all $i \in \mathbb{N}$ and, as a result,

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

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 $z \ge 1$. 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 $\big(\tilde{Q}_i(t), \tilde{Q}_{0i}(t)\big)$ is a Markov chain and is visualized in Figure 3.

Let $\pi_{j\ell} = \mathbb{P}\big[\tilde{Q}_i(\infty) = j, \tilde{Q}_{0i}(\infty) = \ell\big]$ 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:

$$\pi_{0,\ell+1} = \frac{\phi(1-\lambda)}{1-\lambda^{z+1}} \cdot \pi_{00} \cdot \rho^{\ell},\ \forall\, \ell \ge 0,$$
$$\pi_{z\ell} = \frac{\lambda^z(1-\lambda)}{1-\lambda^{z+1}} \cdot \pi_{00} \cdot \rho^{\ell},\ \forall\, \ell \ge 0,$$

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,

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

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

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

for some constant $\alpha_1 > 0$. The inequality in Lemma A.4 holds with constants α_1 and $\alpha_2 = -\ln \rho > 0$.

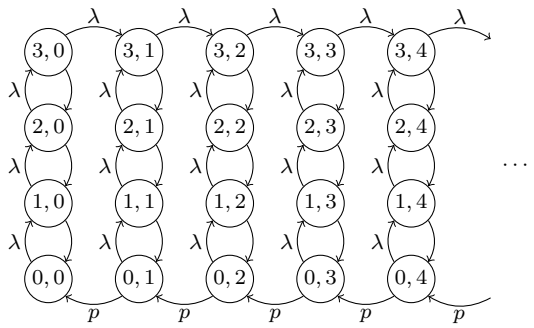


Figure 3: Markov chain of $\big(\tilde{Q}_i(t), \tilde{Q}_{0i}(t)\big)$ with $z = 3$. All unspecified transition rates take the value of one. □

A.4.7 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 $Q_0(t+1) = \big(Q_0(t) + A(t) - S(t)\big)^+$, we have

$$\begin{aligned} Q_0^2(t+1) - Q_0^2(t) &\leq \big(Q_0(t) + A(t) - S(t)\big)^2 - Q_0^2(t) \\ &= \big(A(t) - S(t)\big)^2 + 2Q_0(t)\big(A(t) - S(t)\big) \\ &\leq 2Q_0(t)\big(A(t) - S(t)\big) + 1, \end{aligned} \tag{6}$$

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

We now consider two scenarios. First, when the empirical distribution of queue lengths satisfies $\hat{\pi}(t) \in \mathcal{E}$, from (6) we have:

$$\begin{aligned} \mathbb{E}\Big[Q_0^2(t+1) - Q_0^2(t)\Big|Q_0(t), \hat{\pi}(t) \in \mathcal{E}\Big] &\leq 2\mathbb{E}\big[Q_0(t)(A(t) - S(t))|Q_0(t), \hat{\pi}(t) \in \mathcal{E}\big] + 1 \\ &\leq -(1-\phi)\underline{g}Q_0(t) + 1, \end{aligned} \tag{7}$$

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 (7) yields that for any $Q_0(t) \geq 1$,

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

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

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

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

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

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:

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

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

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

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) $\bigtriangledown L(\mathbf{x}) \leq -\gamma$ when $L(\mathbf{x}) > B$ and $\mathbf{x} \in \mathcal{E}$, and

(ii) $\bigtriangledown L(\mathbf{x}) \leq \delta$ when $L(\mathbf{x}) > B$ and $\mathbf{x} \notin \mathcal{E}$.

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

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

where $\nu_{\max} \triangleq \sup_{\mathbf{x},\mathbf{x}' \in \mathcal{X}: p(\mathbf{x},\mathbf{x}') > 0}\big|L(\mathbf{x}') - L(\mathbf{x})\big|$ 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 $Y_t = \varnothing$ 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, (4)) via a dual variable $\mu \in \mathbb{R}$. We denote the Lagrangian relaxation by $V^{\mathrm{FLR}}(\mu)$, which is provided in (8).

$$V^{\mathrm{FLR}}(\mu) = \min_{\pi \in \Pi} \sum_{q \in \mathbb{N}} \left\{ \frac{q}{1+\lambda} \cdot p^{\pi}(q) + c \cdot \mathbb{P}^{\pi}\big(q, Y=0\big) + \phi \cdot (c-\mu) \cdot \mathbb{P}^{\pi}\big(q, Y=1\big) + (w+\mu) \cdot \mathbb{P}^{\pi}\big(q, X=1\big) \right\}, \tag{8}$$

In (8), $p^{\pi}(q)$ 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 the policy π. In (8), we no longer require that the expected rates of earning and spending tokens are equal; instead, we introduce a penalty μ for spending a token and a reward $-\mu$ for earning a token.

The problem $V^{\mathrm{FLR}}(\mu)$ 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 $V^{\mathrm{FLR}}(\mu)$ and for any dual variable $\mu \in \mathbb{R}$. Therefore, without loss of generality, we can treat $V^{\mathrm{FLR}}(\mu)$ 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 (8) and attains an objective that is no larger, we have $V^{\mathrm{FLR}}(\mu) \le V^{\mathrm{F}}$ for any $\mu \in \mathbb{R}$. We formally state this weak-duality property in Lemma B.1.

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

Note that $V^{\mathrm{FLR}}(\mu)$ is a concave function of μ by (8). Therefore, we can solve a convex optimization problem

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

to obtain the tightest Lagrangian relaxation bound V^{FLR}. We let $\mu^* = \mathrm{argmax}_{\mu \in \mathbb{R}} V^{\mathrm{FLR}}(\mu)$ 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 (8) have the following relationship.*

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

3. *Let μ^* be an optimal dual variable. Then, any policy π^* that is optimal to V^{F} is optimal to $V^{\mathrm{FLR}}(\mu^*)$.*

Proof. According to Danskin's theorem (Proposition 4.5.1 in Bertsekas et al. 2003) and the fact that randomizing between two optimal policies of (8) remains optimal to (8),[22] the sub-differential (i.e., set of sub-gradients) of $V^{\mathrm{FLR}}(\mu)$ at any $\mu \in \mathbb{R}$, denoted by $\partial V^{\mathrm{FLR}}(\mu)$, can be expressed as

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

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

$$\begin{aligned} V^{\mathrm{FLR}} &= \sum_{q\in\mathbb{N}} \Big\{ \frac{q}{1+\lambda} \cdot p^{\pi^*}(q) + c \cdot \mathbb{P}^{\pi^*}\big(q, Y=0\big) + \phi \cdot (c-\mu^*) \cdot \mathbb{P}^{\pi^*}\big(q, Y=1\big) + (w+\mu^*) \cdot \mathbb{P}^{\pi^*}\big(q, X=1\big) \Big\} \\ &= \sum_{q\in\mathbb{N}} \Big\{ \frac{q}{1+\lambda} \cdot p^{\pi^*}(q) + c \cdot \Big(\mathbb{P}^{\pi^*}\big(q, Y=0\big) + \phi \cdot \mathbb{P}^{\pi^*}\big(q, Y=1\big) \Big) + w \cdot \mathbb{P}^{\pi^*}\big(q, X=1\big) \Big\} \geq V^{\mathrm{F}} \geq V^{\mathrm{FLR}}, \end{aligned} \tag{9}$$

where the first equality follows from π^* being optimal to $V^{\mathrm{FLR}}(\mu^*)$, the second equality from the fact that the flow balance constraint of the tokens holds under policy π^*, the first inequality is because policy π^* is feasible to (2), and the second inequality follows from the weak duality (Lemma B.1). Therefore, all of the inequalities in (9) are binding, which implies that strong duality holds and policy π^* is optimal to V^{F}.

Bullets 2 and 3 can be easily derived in a similar way using a chain of inequalities analogous to (9). □

Finally, we remark that the Lagrangian relaxation $V^{\mathrm{FLR}}(\mu)$ can be solved by the following Bellman equation (10)

$$\begin{aligned} v + h(q,1) &= \frac{q}{1+\lambda} + \min\big\{ h(q+1), h(q) + w + \mu \big\} \\ v + h(q,0) &= \frac{q}{1+\lambda} + \min\big\{ h(q), h(q-1) + c, h(q) + \phi(c-\mu) \big\}, \forall\, q \geq 1 \\ v + h(q,0) &= \frac{q}{1+\lambda} + \min\big\{ h(q), h(q) + \phi(c-\mu) \big\},\ q = 0. \end{aligned} \tag{10}$$

where $v = V^{\mathrm{FLR}}(\mu)$ 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.

[22] This is because the Lagrangian problem (8) can be formulated as a linear program.

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

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

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.,

$$\pi_0 \phi = \lambda\big(\pi_k + \pi_{k-1} p\big). \tag{11}$$

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

Under the policy, the expected queue length is

$$\mathbb{E}\big[q\big] = \sum_{i\ge 0} i\pi_i = \frac{1}{1-\phi}\left(\sum_{i=1}^{k} \lambda^i - k\phi\right), \tag{12}$$

the expected job processing cost is

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

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

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

where the first equality follows from (11). Thus, the expected total cost of the policy is

$$v \triangleq \frac{1}{1+\lambda} \cdot \frac{1}{1-\phi}\left(\sum_{i=1}^{k} \lambda^i - k\phi\right) + \frac{c\lambda}{1+\lambda} + \frac{1-\lambda}{1-\phi} \cdot \frac{\phi}{1+\lambda} \cdot w. \tag{13}$$

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. Therefore, the benefit of the policy relative to operating independently is

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

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

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 μ such that the policy is optimal to $V^{\mathrm{FLR}}(\mu)$, 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 μ To find an appropriate dual variable μ, 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 μ. 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 (10))

$$h(k) = h(k-1) + w + \mu. \tag{14}$$

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

$$v + h(k) = \frac{k}{1+\lambda} + \frac{\lambda}{1+\lambda}\Big(h(k) + w + \mu\Big) + \frac{1}{1+\lambda}\Big(h(k-1) + c\Big) \tag{15}$$

with v specified in (13). From (13), (14), and (15), we have

$$\mu = c + \frac{m(k)}{1-\phi} - \frac{w}{1-\phi}. \tag{16}$$

Remark B.2. From (16), 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 μ specified in (16), 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 (17) – (19).

$$v + h(0) = \frac{\lambda}{1+\lambda}h(1) + \frac{1}{1+\lambda}\big(h(0) + \phi(c-\mu)\big), \tag{17}$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda}h(q+1) + \frac{1}{1+\lambda}\big(h(q-1) + c\big),\ \forall\, q \in [1:k-1], \tag{18}$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda}\big(h(q) + w + \mu\big) + \frac{1}{1+\lambda}\big(h(q-1) + c\big),\ \forall\, q \geq k-1. \tag{19}$$

Note that (18) is the Bellman equation for $q \in [1:k-2]$, but since $h(k-1) + w + \mu = h(k)$ by (14), (18) is also valid for $q = k-1$. Analogously, (19) is the Bellman equation for $q \geq k$, but it is also valid for $q = k-1$ again by (14).

We can derive the values of $h(q)$ from (17) – (19). First, without loss of generality, we can assume $h(0) = 0$. Then, from (17), we can obtain $h(1)$ as

$$h(1) = \frac{(1+\lambda)v + \phi(\mu - c)}{\lambda}. \tag{20}$$

Once we have $h(0) = 0$ and $h(1)$, we can determine the values of $h(q)$ for $q \leq k$ from (18) recursively:

$$h(q) = \frac{\big(v + h(q-1)\big)(1+\lambda) - (q-1) - \big(h(q-2) + c\big)}{\lambda},\ \forall\, q \in [2:k]. \tag{21}$$

For example, from (21) we have

$$h(2) = \frac{(v + h(1))(1+\lambda) - 1 - c}{\lambda}. \tag{22}$$

Finally, from (19) we have

$$h(q) = h(q-1) + q + \lambda(w+\mu) + c - (1+\lambda)v, \ \forall\, q \geq k-1. \tag{23}$$

(21) and (23) 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 $V^{\mathrm{FLR}}(\mu)$ with μ specified in (16), 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 (23), when $q \geq k-1$,

$$\Delta h(q) = q + \lambda(w+\mu) + c - (1+\lambda)v = q + \text{const}, \ \forall\, q \geq k-1, \tag{24}$$

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 (21) we have

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

Hence,

$$\Delta h(q) = a\Big(\frac{1}{\lambda}\Big)^q + b + \frac{q}{1-\lambda}, \ \forall\, q \leq k, \tag{25}$$

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 $a_n = a\big(\frac{1}{\lambda}\big)^n + b$ 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

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

where the second equality follows from (13), (16), (20), and (22). Since $a < 0$, $\Delta h(q)$ is concave for $q \leq k$ by (25). Moreover, since $\Delta h(q)$ is concave (actually linear) for $q \geq k-1$ by (24), $\Delta h(q)$ 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 (24)) 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. □

Step Three: Verifying Optimality We now verify that the policy is optimal to $V^{\mathrm{FLR}}(\mu)$ with μ specified in (16). It suffices to show that the policy attains the minimum in the Bellman equation (10) with v specified in (13) 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 (14), 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 (13), (16), and (20) we have

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

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 $V^{\mathrm{FLR}}(\mu)$ with μ specified in (16). On the other hand, the above discussion indicates that the proposed policy is the unique optimal policy to $V^{\mathrm{FLR}}(\mu)$ that satisfies the flow balance constraint of the tokens. (Note that the differential value functions in (10) 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 range $[0\!:\!z]$ 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

$$v(z) = \frac{c\lambda}{1 + \lambda} + \frac{1}{1 + \lambda} \cdot \sum_{i=0}^{z} i\pi_i + \pi_z \cdot \frac{\lambda}{1 + \lambda} \cdot w. \tag{26}$$

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 μ 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 (27) – (29).

$$v + h(0) = \frac{\lambda}{1+\lambda} h(1) + \frac{1}{1+\lambda} h(0), \tag{27}$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda} h(q+1) + \frac{1}{1+\lambda}\big(h(q-1) + c\big),\ \forall\, q \in [1 : z-1], \tag{28}$$

$$v + h(q) = \frac{q}{1+\lambda} + \frac{\lambda}{1+\lambda}\big(h(q) + w + c\big) + \frac{1}{1+\lambda}\big(h(q-1) + c\big),\ \forall\, q \geq z, \tag{29}$$

with v given in (26).

We can derive the values of $h(q)$ from (27) – (29). First, without loss of generality, assume $h(0) = 0$. Then, from (27) we have

$$h(1) = \frac{1+\lambda}{\lambda}\, v. \tag{30}$$

Once we have $h(0) = 0$ and $h(1)$, we can determine the values of $h(q)$ for $q \leq z$ from (28) recursively:

$$h(q) = \frac{\big(v + h(q-1)\big)(1+\lambda) - (q-1) - \big(h(q-2) + c\big)}{\lambda},\ \forall\, q \in [2 : z]. \tag{31}$$

For example, from (31) we have

$$h(2) = \frac{(v + h(1))(1+\lambda) - 1 - c}{\lambda}. \tag{32}$$

Finally, from (29) we have

$$h(q) = h(q-1) + q + \lambda w + (1+\lambda)c - (1+\lambda)v,\ \forall\, q \geq z. \tag{33}$$

(31) and (33) 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 (33), when $q \geq z$,

$$\Delta h(q) = q + \lambda w + (1+\lambda)c - (1+\lambda)v = q + \text{const},\ \forall\, q \geq z, \tag{34}$$

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 (31) we have

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

Hence,

$$\Delta h(q) = a\Big(\frac{1}{\lambda}\Big)^q + b + \frac{q}{1-\lambda},\ \forall\ q \le z, \tag{35}$$

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 $a_n = a\big(\frac{1}{\lambda}\big)^n + b$ 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

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

where the first inequality follows from (26), (30), (32) and the fact that $w \le m(z+1)$. Since $a < 0$, $\Delta h(q)$ is concave for $q \le z$ by (35). Moreover, since $\Delta h(q)$ is concave for $q \ge z$ by (34), and additionally,

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

$\Delta h(q)$ is concave for all q. This together with the fact that $\Delta h(q+1) - \Delta h(q) = 1 \ge 0$ for all $q \ge z$ (equation (34)) implies that $\Delta h(q)$ is strictly increasing for all $q \ge 1$.

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

Step Two: Verifying Optimality We now verify that the policy is optimal to $V^{\mathrm{FLR}}(\mu = c)$. It suffices to show that the policy attains the minimum in the Bellman equation (10) with the value of v in (26) 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 λ^{1+z}/ϕ ensures that the tokens' flow balance constraint holds and hence the policy is feasible to (2). When $q \ge 1$, 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., $h(1) \ge h(0) + c = c$. To see this, note that from (26) and (30), we have

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

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

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 $h(z+1) \ge h(z) + w + c$. Note that from (34), we have

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

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 (34), we have

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

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 $w \in \big(m(z), m(z+1)\big)$ for any $z \geq k$. To see this, note that by Lemma B.2 Bullet 3, any optimal policy of (2) is optimal to $V^{\mathrm{FLR}}(\mu = c)$. On the other hand, the above discussion indicates that the proposed policy is the unique optimal policy to $V^{\mathrm{FLR}}(\mu = c)$ 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 $w \in \mathbb{R}_+ \setminus \big\{m(z) : z \geq k\big\}$. From Remarks B.3 and B.5, $\pi^*(w)$ is the unique optimal policy of (2) for any $w \in \mathbb{R}_+ \setminus \big\{m(z) : z \geq k\big\}$.

When $w = m(z)$ with some integer $z \geq k$, let $\pi^*(w+)$ and $\pi^*(w-)$ be the unique optimal policy of (2) when $w \in \big(m(z), m(z+1)\big)$ and $w \in \big(m(z-1), m(z)\big)$, 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 (4), 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 in the shared pool is $w \geq 0$. From above, $\Pi^*(w) = \big\{\pi^*(w)\big\}$ for any $w \in \mathbb{R}_+ \setminus \big\{m(z) : z \geq k\big\}$ and $\Pi^*(w) \supseteq \Big\{\alpha\,\pi^*(w+) \oplus (1-\alpha)\,\pi^*(w-) : \alpha \in [0,1]\Big\}$ for any $w = m(z)$ with $z \geq k$, where $\alpha\pi_1 \oplus (1-\alpha)\pi_2$ with a probability of α and policies π_1 and π_2 denotes the randomized policy that follows policy π_1 with probability α and policy π_2 with probability $1-\alpha$. We now show that

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

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 (36),

$$V^{\mathrm{F}}(w) = \min_{r \in \left[0, \frac{\lambda}{1+\lambda}\right]} \left\{ \min_{\pi \in \Pi(r)} \frac{\mathbb{E}_\pi[q]}{1+\lambda} + rw + \frac{c\lambda}{1+\lambda} \right\} \tag{36}$$

where $r \in \left[0, \frac{\lambda}{1+\lambda}\right]$ 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 π. In (36), 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 (36) for a given value of $w \geq 0$, and $r(\pi)$ the steady-state rate of requesting help under a policy π. Note that since the objective of (36) 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 $V^{\mathrm{F}}(w')$ for any $w' \in \big(m(z-1), m(z)\big)$ and $\pi(w+)$ the unique optimal policy of $V^{\mathrm{F}}(w')$ for any $w' \in \big(m(z), m(z+1)\big)$, we have $R^*(w') = \big\{r(\pi(w-))\big\}$ for any $w' \in \big(m(z-1), m(z)\big)$ and $R^*(w') = \big\{r(\pi(w+))\big\}$ for any $w' \in \big(m(z), m(z+1)\big)$. As a result, $R^*(w) \subseteq \big[r\left(\pi(w+)\right), r\left(\pi(w-)\right)\big]$ by the property of the strong set order. On the other hand, $R^*(w) \supseteq \big[r\left(\pi(w+)\right), r\left(\pi(w-)\right)\big]$ because any mixing of $\pi(w-)$ and $\pi(w+)$ is optimal to $V^{\mathrm{F}}(w)$. Therefore, $R^*(w) = \big[r\left(\pi(w+)\right), r\left(\pi(w-)\right)\big]$ and it implies that

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

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 ϕ of the steady-state rate of offering help.)

C Proofs for Sections 6 and 7

C.1 Proof of Lemma 6.1

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

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 π^{F} and $\bar{\pi}^{\mathrm{F}}$ are visualized in Figure 4.

When $\phi \geq \lambda$, a server that follows policy $\bar{\pi}^{\mathrm{F}}$ 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 $M_2 = \frac{2-\lambda}{1-\lambda}$.

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

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

Let $k \geq 1$ denote the maximum queue length under policy π^{F}, which corresponds to when the value of ϕ satisfies $\phi \in [\lambda^{k+1}, \lambda^k)$. We first show in Lemma C.1 that the stationary probabilities

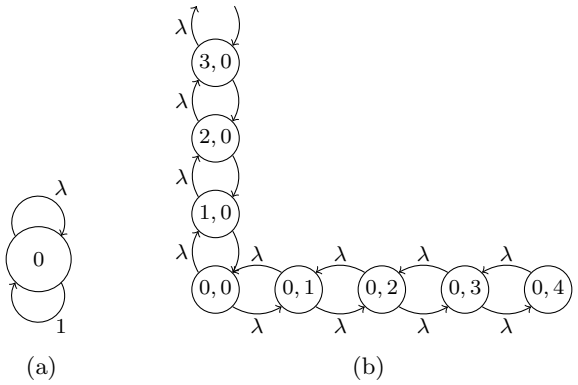


Figure 4: Markov chains when $\phi \geq \lambda$. (a) Markov chain of the queue length q under policy π^{F}. (b) Markov chain of the queue-length-token-count pair (q, s) under policy $\bar{\pi}^{\mathrm{F}}$. The maximum token count is set at $C = 4$, and all unspecified transition rates take the value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

p_{ks} for $s \in [0 : C]$, which correspond to the probabilities of states in the first row of Figure 5b, have comparable values.

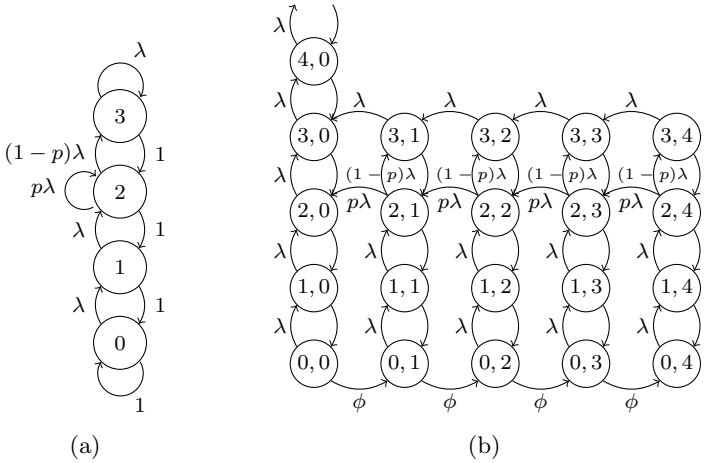


Figure 5: Markov chains when $\phi < \lambda$. Specific to this example, the token-earning probability ϕ satisfies $\phi \in \left[\lambda^{k+1}, \lambda^k\right)$ with $k = 3$, and $p = \frac{\phi - \lambda^{k+1}}{\lambda^k - \lambda^{k+1}} \in [0, 1)$. (a) Markov chain of the queue length q under policy π^{F}. (b) Markov chain of the queue-length-token-count pair (q, s) under policy $\bar{\pi}^{\mathrm{F}}$. 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., $p_{k1} \geq p_{k2} \geq \cdots \geq p_{kC}$. 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 λ and ϕ.*

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

Second, we have $p_{q0} \leq c_1(\lambda, \phi) \cdot p_{k0}$ for any index $q \leq k$ and some constant $c_1(\lambda, \phi)$ that depends only on the values of λ and ϕ. 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 follows from recursion.

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

From the above, we have

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

where the equality is due to the fact that $p_{k+i,0} = \lambda^i \cdot p_{k,0}$ for all $i \ge 1$. The inequality in Lemma 6.1 holds by taking $M_2 = \frac{1}{c}\cdot\Big(\frac{\lambda}{1-\lambda} + k\cdot\big(c_1(\lambda,\phi)+c_2(\lambda,\phi)\big)\Big)$.

C.1.3 Proof of Lemma C.1

In the proof, we express all of the stationary probabilities p_{qs} in terms of the probability p_{k0}. We first consider the case of $\phi \in (\lambda^{k+1}, \lambda^k)$ with $k \ge 3$. 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

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

Therefore, we have

$$p_{q+1,s} - p_{q,s} = \lambda\big(p_{q,s} - p_{q-1,s}\big) = \lambda^q\big(p_{1,s} - p_{0,s}\big),\ \forall\, q \in [1:k-2]. \tag{38}$$

Additionally, the general solution to the sequence in (37) is

$$p_{q,s} = a_s\lambda^q + b_s,\ \forall\, q \in [0:k-1], \tag{39}$$

where a_s and b_s are two constants (whose values are to be determined), because λ 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.

$$a_0 + b_0 = p_{00}, \tag{40}$$
$$a_0\lambda + b_0 = p_{10} = p_{00}\cdot(\lambda+\phi), \tag{41}$$
$$p_{k-1,0} = a_0\lambda^{k-1} + b_0, \tag{42}$$
$$p_{k0} = \lambda\cdot(p_{k-1,0} + p_{k1}), \tag{43}$$
$$p_{k0} + p\lambda\cdot p_{k-1,1} = p_{k-1,0} + \lambda\cdot\big(p_{k-1,0} - p_{k-2,0}\big) = p_{k-1,0} + \lambda^{k-1}\cdot(\lambda+\phi-1)\cdot p_{00}, \tag{44}$$

where (40) - (42) follow from (39) for states $(0,0)$, $(1,0)$, and $(k-1,0)$, respectively. (43) and (44) are the flow balance equations at states $(k,0)$ 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 (41) also uses the flow balance equation at state $(0,0)$, and the second equality in (44) follows from (38), (40) and (41).

We can express the constants a_0 and b_0 in terms of p_{00} using (40) and (41). Then, we can express p_{00} in terms of $p_{k-1,0}$ using (42). Lastly, we can express p_{k1} in terms of p_{k0} and $p_{k-1,0}$ using (43) and $p_{k-1,1}$ in terms of p_{k0} and $p_{k-1,0}$ using (44). In the end, we have

$$\begin{pmatrix} p_{k1} \\ p_{k-1,1} \\ p_{00} \end{pmatrix} = \begin{pmatrix} \frac{1}{\lambda} & -1 \\ -\frac{1}{p\lambda} & \frac{1}{p\lambda}\left(1+\lambda^{k-1}(\lambda+\phi-1)t\right) \\ 0 & t \end{pmatrix} = \begin{pmatrix} p_{k0} \\ p_{k-1,0} \end{pmatrix} \tag{45}$$

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 $i \geq 1$. In particular, we have the following equations.

$$p_{k-1,i} = a_i \lambda^{k-1} + b_i, \tag{46}$$
$$(a_i + b_i)(\lambda + \phi) = p_{0,i-1} \cdot \phi + (a_i \lambda + b_i), \tag{47}$$
$$p_{0i} = a_i + b_i, \tag{48}$$
$$p_{ki} \cdot (1+\lambda) = p_{k,i+1} \cdot \lambda + p_{k-1,i} \cdot (1-p)\lambda, \tag{49}$$
$$p_{ki} + p\lambda \cdot p_{k-1,i+1} = p_{k-1,i} + \lambda \cdot (p_{k-1,i} - p_{k-2,i}) = p_{k-1,i} + \lambda^{k-1} \cdot (\lambda - 1) \cdot a_i, \tag{50}$$

where (46) follows from (39) for state $(k-1,i)$, (47) from (39) for state $(1,i)$ and the flow balance equation at state $(0,i)$, (48) from (39) for state $(0,i)$, (49) from the flow balance equation at state (k,i), and (50) from the flow balance equation at state $(k-1,i)$, (38), and the fact that $p_{1i} - p_{0i} = a_i \cdot (\lambda - 1)$ by (39).

We can express the constants a_i and b_i in terms of $p_{0,i-1}$ and $p_{k-1,i}$ using (46) and (47). Then, we can express p_{0i} in terms of $p_{0,i-1}$ and $p_{k-1,i}$ using (48). Lastly, we can express $p_{k,i+1}$ in terms of p_{ki}, $p_{k-1,i}$ and $p_{0,i-1}$ using (49) and $p_{k-1,i+1}$ in terms of p_{ki}, $p_{k-1,i}$ and $p_{0,i-1}$ using (50). In the end, we have

$$\begin{pmatrix} p_{k,i+1} \\ p_{k-1,i+1} \\ p_{0i} \end{pmatrix} = \begin{pmatrix} \frac{1+\lambda}{\lambda} & p-1 & 0 \\ -\frac{1}{p\lambda} & \frac{1}{p\lambda}\left(1+\lambda^{k-1}(\lambda+\phi-1)t\right) & -\frac{1}{p\lambda}\lambda^{k-1}\phi t \\ 0 & \frac{\lambda(1-\lambda)}{\lambda\phi+\lambda^k(1-\phi)-\lambda^{k+1}} & \frac{(\lambda-\lambda^k)\phi}{\lambda\phi+\lambda^k(1-\phi)-\lambda^{k+1}} \end{pmatrix} = \begin{pmatrix} p_{ki} \\ p_{k-1,i} \\ p_{0,i-1} \end{pmatrix}, \forall\, i \geq 1, \tag{51}$$

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

Step Four From (45) and (51), 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 (45) and (51), 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

$$p_{ki} = p_{k0} \cdot \frac{(\lambda^k - \phi)\phi \cdot (r^C - r^{i-1})}{\lambda^k(1-\phi)\phi \cdot r^C - \lambda(\lambda^k - \phi)(\phi - \lambda^{k+1})} \tag{52}$$

where

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

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 (52) 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, we let $p_{qs} = \mathbb{P}\big[\bar{Q}_i(\infty) = q, \bar{S}_i(\infty) = s\big]$ denote the stationary distribution of the queue-length-token-count pair under policy $\bar{\pi}^{\mathrm{F}}$ and $\pi_q = \mathbb{P}[Q_i(\infty) = q]$ denote the stationary distribution of the queue length under policy π^{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 π^{F} and $\bar{\pi}^{\mathrm{F}}$ are visualized in Figure 4.

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

On the other hand, a server that follows policy $\bar{\pi}^{\mathrm{F}}$ 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 $s \in \big[1:C\big]$, and $p_{qs} = 0$ for all the other states. From $\sum_{q,s\geq 0} p_{qs} = 1$, we have $p_{00} = \frac{1-\lambda}{1+C(1-\lambda)}$. As a result,

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

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

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

We next suppose that $\phi < \lambda$; the Markov chains of the dynamics under policies π^{F} and $\bar{\pi}^{\mathrm{F}}$ are visualized in Figure 5.

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

$$\begin{aligned} h_0\lambda &= h_1, \\ h_1\lambda &= h_2, \\ &\cdots \\ h_{k-2}\lambda &= h_{k-1}, \\ h_k \ge h_{k-1}\cdot(1-p)\lambda &\ge (h_{k-1} - p_{k-1,0})\cdot(1-p)\lambda \ge h_k - p_{k0}, \end{aligned} \tag{53}$$

where, in the last row, the first inequality follows from the fact that $p_{k+i,0} = \lambda^i \cdot p_{k,0}$ for all $i \ge 1$, and the third inequality follows from the flow balance equation for the collection of states $\big\{(k,s) : 1 \le s \le C\big\}$.

On the other hand, the dynamics of jobs under policy π^{F} follow a birth-and-death chain process. Therefore, the stationary distribution satisfies:

$$\begin{aligned} \pi_0\lambda &= \pi_1, \\ \pi_1\lambda &= \pi_2, \\ &\cdots \\ \pi_{k-2}\lambda &= \pi_{k-1}, \\ \pi_k &= \pi_{k-1}\cdot(1-p)\lambda. \end{aligned} \tag{54}$$

From (53), (54), and the fact that $\sum_{q=1}^{k}\pi_q = 1$ and $\sum_{q=1}^{k} h_q \le 1$, we have:

$$1 \ge \frac{h_0}{\pi_0} = \frac{h_1}{\pi_1} = \cdots \frac{h_{k-1}}{\pi_{k-1}} = \frac{h_{k-1}\cdot(1-p)\lambda}{\pi_k} \ge \frac{h_k - p_{k0}}{\pi_k}. \tag{55}$$

Proof of Bullet One First, note that

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

where the third equality follows from the fact that $p_{k+i,0} = p_{k,0}\cdot\lambda^i$ for all $i \ge 0$, and the inequality follows from (55) and the facts that $\mathbb{E}[Q_i(\infty)] = \sum_{q=1}^{k}\pi_q \cdot q$ and that $p_{k0} \le \frac{M_2}{C}$ by Lemma 6.1.

Second, we have $\mathbb{E}[Q_i(\infty)] \le \mathbb{E}[\bar{Q}_i(\infty)]$. This is because the policies π^{F} and $\bar{\pi}^{\mathrm{F}}$ 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 π^{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), as we demonstrate in Lemma C.2.

Lemma C.2. *The policy π^{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 the server processes jobs from the shared pool equals the rate of earning tokens (because both equal ϕ 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. Therefore, 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 the server processes jobs equals the job arrival rate λ.

Second, by Corollary 4.3, the policy π^{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 π^{F} has the lowest expected queue length among all policies that satisfy the flow balance constraint of the tokens (because these policies share the same time-average job processing cost, which equals $c\lambda$). □

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

$$\sum_{q=0}^{k} h_q + \sum_{i=1}^{\infty} p_{k+i,0} = \sum_{q=0}^{k} h_q + p_{k,0} \cdot \frac{\lambda}{1-\lambda} = 1, \tag{56}$$

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 $h_{k-1} \cdot (1-p)\lambda \geq h_k - p_{k0}$ by (53), the equality from (56), and the second inequality from $p_{k,0} \leq \frac{M_2}{C}$ by Lemma 6.1. Therefore, from (55) we have

$$\frac{h_0}{\pi_0} = \frac{h_1}{\pi_1} = \cdots \frac{h_{k-1}}{\pi_{k-1}} = \frac{h_{k-1} \cdot (1-p)\lambda}{\pi_k} \geq r(C) \triangleq 1 - \frac{M_2}{(1-\lambda)C}. \tag{57}$$

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 $\mathbb{P}[\bar{X}_i(\infty) = 1] = \lambda(h_k - p_{k0}) + p\lambda(h_{k-1} - p_{k-1,0})$. 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 $h_k - p_{k0} \leq h_{k-1} \cdot (1-p)\lambda$ by (53) and the second inequality from (55).

On the other hand, from (57) and the fact that $h_k \geq h_{k-1} \cdot (1-p)\lambda$ by (53), we have

$$\mathbb{P}[\bar{X}_i(\infty) = 1] \geq \mathbb{P}[X_i(\infty) = 1] \cdot r(C) - \lambda \cdot p_{k0} - p\lambda \cdot p_{k-1,0}.$$

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

Proof of Bullet Three Since the policies π^{F} and $\bar{\pi}^{\mathrm{F}}$ 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.4) with small modifications.

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

We show that the expected shared pool queue length is uniformly bounded from above by an absolute constant that depends only on the values of λ and ϕ 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.4.6. 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 at time 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.

Let $S_i(t)$ denote 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 a key 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 $Q_{0i}(0) = 0$ and $S_i(0) \leq C$.

Now, suppose that the inequality holds for period $t-1$. We show that it continues to hold 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 continues to hold.

Second, 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. As a result, the inequality remains valid. Otherwise, if $Q_{0i}(t-1) = 0$, then the inequality also holds in period t because $Q_{0i}(t) = 0$ and hence $S_i(t) + Q_{0i}(t) = S_i(t) \leq C$. □

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 λ and a job processing rate of one, regardless of the value of C.

Second, 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 λ/ϕ. Hence, the service rate is at least λ/ϕ 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 λ and a job processing rate of λ/ϕ regardless of the value of C.

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

Validity of Lemma A.3 Let $g(C) > 0$ denote the steady-state rate of offering help when the server follows the FMFE policy $\bar{\pi}^{\mathrm{F}}$ 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 $\lim_{C \to \infty} g(C) = \mathbb{P}[Y_i(\infty) = 1] > 0$ by Lemma 6.2, where $\mathbb{P}[Y_i(\infty) = 1]$ is the steady-state offering help rate under the FMFE policy π^{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.4.2. Specifically, let π_{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 ϕ times the steady-state rate of offering help (this is because the expected rates of earning and spending tokens are equal under the policy $\bar{\pi}^{\mathrm{F}}$) 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.,

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

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 $O\big(\frac{1}{N}\big)$, which is uniform across the token-amount upper bound value C, following the same proof in Appendix A.4.5. 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 that all servers follow the FMFE policy $\bar{\pi}^{\mathrm{F}}$. 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. Second, the time-average holding cost for jobs in the queue, which equals $\mathbb{E}\big[\bar{Q}_i(\infty)\big]$, is at most $\mathbb{E}\big[Q^{\mathrm{F}}\big] + \frac{M_3}{C}$ 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 λw, 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). □

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, in the relaxation, the shared pool is emptied whenever server one requests or offers help. As we show in 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. We remark 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. □

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

Because servers two to N follow an oblivious strategy $\bar{\pi}^{\mathrm{F}}$, 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 τ. 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 ϕ.

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

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

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

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 τ. Clearly,

$$\tau - \bar{h}(\tau) \leq \tau - h(\tau) \tag{58}$$

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,

$$\mathbb{P}\left[\tau - h(\tau) \leq \frac{1}{1+\lambda}\frac{1}{N^{1-\delta}}\right] \leq \mathbb{P}\left[\tau - \bar{h}(\tau) \leq \frac{1}{1+\lambda}\frac{1}{N^{1-\delta}}\right] \leq \frac{1}{N^{1-\delta}} \tag{59}$$

where the first inequality follows from (58) 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 ϕ. However, every time a capacity unit arrives at a time τ, 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 τ. 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 ϕ 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 $c\lambda + \mathbb{E}[Q^{\mathrm{F}}]$) minus the time-average compensation. Specifically, let $A = \left\{\Delta(\tau) > \frac{1}{1+\lambda}\frac{1}{N^{1-\delta}}\right\}$ 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

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

where the inequality follows from the fact that $\mathbb{P}\big[A^c\big] \leq \frac{1}{N^{1-\delta}}$ by (59). 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 τ and server one offers help to the shared pool. If event $A = \left\{\Delta(\tau) > \frac{1}{1+\lambda}\frac{1}{N^{1-\delta}}\right\}$ occurs, then we have $\tau - h(\tau) > \frac{1}{1+\lambda}\frac{1}{N^{1-\delta}}$ by (58). By Lemma C.6, the difference between the expected cost of offering help in the alternative system and the relaxation is

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

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 is also smaller in the alternative system because

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

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 τ with a high probability (Lemma C.8). In addition, in the stationary distribution, the shared pool is non-empty with a probability close to ϕ (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 τ. 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 τ.

We first bound the total variation distance between the distributions of the shared pool queue lengths $Q_0(\tau)$ and $\bar{Q}_0(\tau)$ $\big(\stackrel{d}{=} \bar{Q}_0(\infty)\big)$ in the two systems at time τ 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 τ, 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*

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

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

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

$$\begin{aligned}\Big|\mathbb{P}\big[Q_0(\tau) \geq 1\big|\tau - h(\tau)\big] - \mathbb{P}[\bar{Q}_0(\infty) \geq 1]\Big| &\leq \max_A \Big|\mathbb{P}\big[Q_0(\tau) \in A\big|\tau - h(\tau)\big] - \mathbb{P}[\bar{Q}_0(\infty) \in A]\Big| \\ &= \big|\big|\mathcal{P} - \bar{\mathcal{P}}\big|\big|_{\text{TV}},\end{aligned} \tag{60}$$

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

$$\mathbb{P}\big[Q_0(\tau) \geq 1\big|\tau - h(\tau)\big] \geq \mathbb{P}[\bar{Q}_0(\infty) \geq 1] - c_1\big(\lambda, \phi, \delta\big) \cdot \exp\bigg(- \alpha_1(\lambda, \phi) \cdot N^{\delta/2}\bigg). \tag{61}$$

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

Lemma C.9. *We have*

$$\mathbb{P}[\bar{Q}_0(\infty) \geq 1] \geq \phi - \frac{c_2\big(\lambda, \phi, \delta\big)}{N^{1-\delta}} \tag{62}$$

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

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, (61) and (62) imply that there exists a constant $c\big(\lambda, \phi, \delta\big)$ 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. Therefore, 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

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

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

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

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

$$\big|\big|\mathcal{P} - \bar{\mathcal{P}}\big|\big|_{\text{TV}} \leq \mathbb{P}\Big[\bar{Q}_0(\tau) \neq Q_0(\tau)\Big] \leq \mathbb{P}\Big[T - h(\tau) \geq \tau - h(\tau)\Big], \tag{63}$$

where the first inequality is because $\bar{Q}_0(\tau) \stackrel{d}{=} \bar{Q}_0(\infty)$. In the following, we bound the probability $\mathbb{P}\big[T - h(\tau) \geq \tau - h(\tau)\big]$ 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 represent the number of arrivals of jobs or capacity units at servers $i \in [2\!:\!N]$ in the time interval $(h(\tau), \tau)$. The random variable $H \sim \text{Poisson}(\lambda_H)$ follows a Poisson distribution with a mean value of $\lambda_H \triangleq (N-1) \cdot (1+\lambda) \cdot \big(\tau - h(\tau)\big) \geq \frac{N-1}{N^{1-\delta}}$, 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:

$$\mathbb{P}\Big[H \leq N^{\delta/2}\Big] \leq c_3(\delta) \cdot \exp\left(-\frac{N^{\delta}}{2}\right) \tag{64}$$

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

Step Two: Defining a "Good" Typical Event Second, we identify a "good" event that happens with a high probability.

To do so, it is convenient to consider an equivalent discrete-time model, in which each period $\ell \in [H]$ corresponds to the ℓ-th arrival of a job or capacity unit. Specifically, suppose that the ℓ-th arrival arrives at some server at time t_ℓ, with $t_0 \triangleq h(\tau) < t_1 < \cdots < t_H < \tau$. Let $\bar{Q}_0[\ell] \triangleq \bar{Q}_0(t_\ell)$ denote the queue length of the shared pool at time t_ℓ (after the server has taken an action) in the alternative system.

For each server i, we let the binary variable $A_i[\ell]$ (and $S_i[\ell]$, respectively) represent whether server i requests help (and offers help, respectively) in period ℓ. Additionally, let $A[\ell] \triangleq \sum_{i\in[2:N]} A_i[\ell] \in \{0,1\}$ and $S[\ell] \triangleq \sum_{i\in[2:N]} S_i[\ell] \in \{0,1\}$ represent whether there is any request or provision of help in period ℓ. The dynamics of the shared pool queue length $\bar{Q}_0[\ell]$ satisfy the following:

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

Since servers two to N are in the stationary distribution at any time t_ℓ, we have $\mathbb{E}\big(A[\ell]-S[\ell]\big) = -(1-\phi)g$, where $g>0$ is the rate of offering help in the stationary distribution. Additionally, denote by $\hat{\pi}_\ell$ the empirical distribution of queue lengths and token amounts of these $N-1$ servers at time t_ℓ, that is,

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

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

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

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)

$$\mathbb{P}\Big(\mathcal{E}_\ell^c\Big) \le \exp\left(-\frac{1}{8}(1-\phi)^2 g^2 (N-1)\right). \tag{65}$$

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

$$\mathbb{P}\left(\bar{Q}_0[0] > \frac{1}{4}(1-\phi)gN^{\delta/2}\right) \le c_4(\lambda,\phi)\cdot\exp\Big(-\alpha_2(\lambda,\phi)\cdot N^{\delta/2}\Big) \tag{66}$$

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

Finally, we define a "good" event A with

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

From (64) - (66) and the union bound, we have

$$\begin{aligned}\mathbb{P}\Big(A^c\Big) &\le c_3(\delta)\cdot\exp\left(-\frac{N^\delta}{2}\right) + N^{\delta/2}\cdot\exp\left(-\frac{1}{8}(1-\phi)^2g^2(N-1)\right) + c_4(\lambda,\phi)\cdot\exp\Big(-\alpha_2(\lambda,\phi)\cdot N^{\delta/2}\Big)\\ &\le c_5(\lambda,\phi,\delta)\cdot\exp\Big(-\alpha_2(\lambda,\phi)\cdot N^{\delta/2}\Big)\end{aligned} \tag{67}$$

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

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

$$\mathbb{E}\Big(\tilde{Q}_0[\ell+1] + \frac{1}{2}(1-\phi)g(\ell+1) \,\Big|\, \tilde{Q}_0[\ell]\Big) \leq \tilde{Q}_0[\ell] + \frac{1}{2}(1-\phi)g\ell.$$

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

$$\mathbb{P}\left(\tilde{Q}_0[\ell] + \frac{1}{2}(1-\phi)g\ell - \bar{Q}_0[0] \geq \epsilon\right) \leq \exp\left(\frac{-\epsilon^2}{2\ell}\right).$$

By taking $\ell = N^{\delta/2}$ and $\epsilon = \frac{1}{4}(1-\phi)gN^{\delta/2}$, and noting that $\bar{Q}_0[0] \leq \frac{1}{4}(1-\phi)gN^{\delta/2}$ given event A, the above implies that

$$\mathbb{P}\left(\tilde{Q}_0\big[N^{\delta/2}\big] \geq 0 \Big| A\right) \leq \exp\left(-\frac{1}{32}(1-\phi)^2 g^2 N^{\delta/2}\right). \tag{68}$$

As a result,

$$\begin{aligned}
\mathbb{P}\Big[T - h(\tau) \geq \tau - h(\tau)\Big|A\Big] &\leq \mathbb{P}\Big[\bar{Q}_0[\ell] \geq 1,\, \forall\, i \leq N^{\delta/2}\Big|A\Big] \\
&= \mathbb{P}\Big[\tilde{Q}_0[\ell] \geq 1,\, \forall\, i \leq N^{\delta/2}\Big|A\Big] \\
&\leq \mathbb{P}\Big[\tilde{Q}_0\big[N^{\delta/2}\big] \geq 1\Big|A\Big] \\
&\leq \exp\left(-\frac{1}{32}(1-\phi)^2 g^2 N^{\delta/2}\right),
\end{aligned} \tag{69}$$

where the last inequality follows from (68).

Finally, from (67) and (69) we have

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

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

Lemma C.10 (Concentration Inequality for Poisson Random Variables). *Let $X \sim$ Poisson(λ) 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

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

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 $\exp(-\lambda g(u))$. □

C.5.6 Proof of Lemma C.9

We apply 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 $\bar{\pi}^{\mathrm{F}}$. 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 $\bar{\pi}^{\mathrm{F}}$. 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 $\check{Q}_0(t)$.

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

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

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

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 remains 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 ϕ, it suffices to show that the shared pool queue lengths in the two systems at time τ are close in terms of the total variation distance.

Analogous to Appendix C.5.5, 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 a job or capacity unit at server one before time τ. 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 $\big(h(\tau), \tau\big)$, the shared pool interacts only with servers two to N in both systems. Let

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

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

$$T_2 \triangleq \inf\Big\{t > h(\tau) : \check{Q}_0(t) = 0\Big\}$$

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

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

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

$$\begin{aligned}\phi - \mathbb{P}\Big[\bar{Q}_0(\infty) \geq 1\Big] =& \mathbb{P}\Big[\check{Q}_0(\tau-) \geq 1\Big] - \mathbb{P}\Big[\bar{Q}_0(\tau-) \geq 1\Big] \\ \leq& \big|\mathbb{P}\big[\check{Q}_0(\tau-) \geq 1\big] - \mathbb{P}\big[\bar{Q}_0(\tau-) \geq 1\big]\big| \\ \leq& \mathbb{P}\Big[\bar{Q}_0(\tau-) \neq \check{Q}_0(\tau-)\Big] \\ \leq& \mathbb{P}\Big[T - h(\tau) \geq \tau - h(\tau)\Big],\end{aligned} \tag{70}$$

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 (60) and (63). In the following, we bound the probability $\mathbb{P}\big[T - h(\tau) \geq \tau - h(\tau)\big]$ 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 through N in the time interval $(h(\tau), \tau)$ is not tiny with a high probability. Specifically, let H denote the number of arrivals of 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 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 $\big(h(\tau), \tau\big)$. Then, the random variable H follows a geometric distribution with a success probability of $\frac{1}{N}$; that is, $\mathbb{P}\big[H = k\big] = \big(\frac{N-1}{N}\big)^k \cdot \frac{1}{N}$ for any integer $k \in \mathbb{N}$.*

Proof. Note that 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 $(N-1) \cdot (1+\lambda) \cdot x$. Therefore, for any integer $k \in \mathbb{N}$, we have

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

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

From Lemma C.12, we have

$$\mathbb{P}\Big[H < N^\delta\Big] \leq \frac{N^\delta}{N} = \frac{1}{N^{1-\delta}}. \tag{71}$$

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

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

$$\mathbb{P}\Big(\bar{Q}_0\big(h(\tau)\big) > \frac{1}{4}(1-\phi) g N^\delta\Big) \leq c_6(\lambda, \phi) \cdot \exp\Big(-\alpha_3(\lambda, \phi) \cdot N^\delta\Big) \tag{72}$$

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

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

server one in system two. Note that

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

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 offers help only when its queue is empty and its token amount is smaller than C, and in such instances offers help at a rate 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). Therefore, for any integer $n\in\mathbb{N}$, we have

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

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

$$\mathbb{P}\Big(\check{Q}_0\big(h(\tau)\big)>\frac{1}{4}(1-\phi)gN^\delta\Big)\le c_7(\lambda,\phi)\cdot\exp\Big(-\alpha_4(\lambda,\phi)\cdot N^\delta\Big)\tag{73}$$

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

Step Three: A Drift Analysis We first define a "good" event A with

$$A\triangleq\Big\{H\ge N^\delta\Big\}\cap\Big\{\cap_{\ell\le N^\delta}\mathcal{E}_\ell\Big\}\cap\Big\{\bar{Q}_0\big(h(\tau)\big)\le\frac{1}{4}(1-\phi)gN^\delta\Big\}\cap\Big\{\check{Q}_0\big(h(\tau)\big)\le\frac{1}{4}(1-\phi)gN^\delta\Big\},$$

where event $\mathcal{E}_\ell$ is defined as in Appendix C.5.5. From (65), (71) - (73), and the union bound, we have

$$\mathbb{P}\Big(A^c\Big)\le\frac{c_8(\lambda,\phi,\delta)}{N^{1-\delta}}\tag{74}$$

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

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

$$\mathbb{P}\Big[T_1-h(\tau)\ge\tau-h(\tau)\Big|A\Big]\le\exp\left(-\frac{1}{32}(1-\phi)^2g^2N^\delta\right),\tag{75}$$

and

$$\mathbb{P}\Big[T_2-h(\tau)\ge\tau-h(\tau)\Big|A\Big]\le\exp\left(-\frac{1}{32}(1-\phi)^2g^2N^\delta\right),\tag{76}$$

which are counterparts to (69). Thus, from (74) - (76) we have

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

for some constant $c_2(\lambda, \phi, \delta)$ that depends only on the values of λ, ϕ, and δ. Combining the above with (70) yields the desired result.

C.6 Proof of Theorem 7.1

We only need to prove that an optimal policy in the centralized control is a cutoff policy when the number of servers N is large. We assume that waiting time in the shared pool is (approximately) zero, as justified by Remark 7.1 for sufficiently large N.

The central planner determines a policy that minimizes the expected time-average total cost (i.e., the holding cost plus processing cost) per server. When a capacity unit arrives at a server, the planner would command the server to serve a job from its queue if it is not empty, because every job needs to be served eventually, and it is most cost-efficient for a server to serve its own jobs. Therefore, the planner only needs to decide whether, upon the arrival of a job at a server, to let the server add the job to its queue for later processing or let other servers help serve the job (which incurs no waiting cost but a higher processing cost of $c' \geq c$). The planner can solve the following optimization problem (77) to determine an optimal policy that minimizes each server's time-average total cost:

$$\begin{aligned}\min_{q_i \geq 0} \quad & c\lambda + \Delta c \cdot \sum_{i=0}^{\infty} \big(\lambda q_i - q_{i+1}\big) + \sum_{i=1}^{\infty} i q_i \\ \text{s.t.} \quad & \lambda q_i \geq q_{i+1}, \, \forall \, i \geq 0, \\ & \sum_{i=0}^{\infty} q_i = 1.\end{aligned} \tag{77}$$

To understand (77), let q_i represent the stationary probability that a server's queue length is $i \in \mathbb{N}$. Searching for an optimal policy is equivalent to searching for an optimal stationary distribution $\{q_i\}$, which is the decision variable in (77). Let $\mathbb{P}(Y = 1|i)$ and $\mathbb{P}(Y = 0|i)$ denote the probabilities that a server requests help or adds the job to its queue, respectively, when a job arrives and there are currently i jobs in its queue. Once we fix a policy, the dynamics of the queue length follow a birth-and-death chain process.[23] Therefore, the stationary distribution is reversible, which implies $\lambda q_i \cdot \mathbb{P}(Y = 0|i) = q_{i+1}$. Consequently,

$$\lambda q_i \cdot \mathbb{P}(Y = 1|i) = \lambda q_i \cdot (1 - \mathbb{P}(Y = 0|i)) = \lambda q_i - q_{i+1} \geq 0,$$

which corresponds to the first constraint in (77) and ensures a nonnegative requesting-help rate when the queue length is i. Finally, the objective function of (77) represents the expected total cost under the stationary distribution $\{q_i\}$. Specifically, the third term represents the expected queue length and corresponds to the holding costs for jobs in the queue, and the first two terms

[23] It is without loss of optimality to focus on unichain policies where the corresponding Markov chain has a single recurrent class that includes state "0" (i.e., an empty queue).

correspond to the job processing costs. To elucidate the first two terms, note that by symmetry, every server serves jobs at a rate of λ, among which a rate of $\sum_{i=0}^{\infty}\big(\lambda q_i - q_{i+1}\big)$ jobs (equal to a server's requesting help rate) is from the other queues and incurs an extra processing cost Δc per job.

We can rewrite the objective function as

$$\begin{aligned} c\lambda + \Delta c \cdot \sum_{i=0}^{\infty}\big(\lambda q_i - q_{i+1}\big) + \sum_{i=1}^{\infty} i q_i &= c\lambda + \lambda\Delta c \cdot q_0 + \sum_{i=1}^{\infty}\big(i - (1-\lambda)\Delta c\big)\cdot q_i \\ &= c\lambda + a_0 \cdot q_0 + \sum_{i=1}^{\infty} a_i \cdot q_i, \end{aligned}$$

where $a_0 \triangleq \lambda\Delta c$ and $a_i \triangleq i - (1-\lambda)\Delta c$ for $i \geq 1$. Note that a_i is increasing for $i \geq 1$. Therefore, for a fixed q_0 and given the constraint $q_{i+1} \leq \lambda q_i$, it is optimal to set $q_i = \lambda q_{i-1} = \lambda^i q_0$ for $i \in [k]$ with an integer $k \in \mathbb{N}$ such that $q_0 \cdot \sum_{i=0}^{k}\lambda^i \leq 1 < q_0 \cdot \sum_{i=0}^{k+1}\lambda^i$. Then, we set $q_{k+1} = 1 - \sum_{i=0}^{k} q_i$ and $q_i = 0$ for all $i \geq k+2$.

Therefore, we can reformulate (77) as the problem of finding the optimal value of q_0. Moreover, the objective function is piecewise linear in q_0, where each breakpoint (or knot) corresponds to $q_0 = \frac{1-\lambda}{1-\lambda^{z+1}}$ for some integer $z \in \mathbb{N}$, $q_i = \lambda^i q_0$ for all $i \in [z]$, and $q_i = 0$ for all $i \geq z+1$. Consequently, each breakpoint corresponds to a cutoff policy with a threshold of $z \in \mathbb{N}$. Since the optimal value of q_0 will be a breakpoint (as we are minimizing a piecewise linear function), the optimal stationary distribution specifies a cutoff policy, which is optimal for centralized control.

C.7 Proof of Lemma 7.2

The proof follows the same steps as those in Appendix B with the new dual variable $\mu = c + \Delta c + \frac{m(k)}{1-\phi} - \frac{w+\Delta c}{1-\phi} > c + \Delta c$ for Case One and $\mu = c + \Delta c$ for Case Two. Note that the Bellman equation for the Lagrangian relaxation $V^{\mathrm{FLR}}(\mu)$ now becomes (see (10) for a comparison)

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

C.8 Proof of Theorem 7.5

Proof of Bullet One Suppose that all servers follow the FMFE strategy π^{F}. Then, the dynamics of the shared pool queue length 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 $w = \frac{\bar{\rho}}{1-\bar{\rho}} \cdot \frac{1}{\sum_{i\in[N]}\lambda_i}$. As a result, server one's time-average waiting cost for jobs in the shared pool is $\lambda_1 w \leq \frac{\bar{\lambda}\bar{\rho}}{\bar{\lambda}(1-\bar{\rho})\cdot N}$.

In addition, 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 at which a server routes its capacity units to the shared pool, server one's time-average job processing cost is $c\lambda_1$.

Proof of Bullet Two The proof mimics the proof of Lemma 6.5 (Appendix C.5). Note that since the token-earning probability ϕ 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 serving 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 serve a job). Nevertheless, server one obtains a token with probability ϕ 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.[24]

The two-dimensional DP problem can be solved by a linear program (78). In (78), $p(q_1, q_0)$ 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 (78), 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 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).

$$
\begin{aligned}
V^{\mathrm{OPT}} = \max_{\substack{p(q_1,q_0),p_2(q_1,q_0),\\ p_3(q_1,q_0,0),p_3(q_1,q_0,1),\\ p_4(q_1,q_0,0),p_4(q_1,q_0,1)}} \quad & N(1+\lambda)\cdot \sum_{q_1,q_0\geq 0} \Big\{ p(q_1,q_0)\cdot \frac{q_1}{N(1+\lambda)} + \Big(p_2(q_1,q_0)+p_3(q_1,q_0,1)\Big)\cdot \frac{q_0+1}{N} \\
& + c\cdot \Big(p_4(q_1,q_0,0)+p_4(q_1,q_0,1)\cdot \mathbb{1}[q_0\geq 1]\Big)\Big\} \\
\text{s.t.} \quad & \sum_{q_1,q_0\geq 0} \Big(p_2(q_1,q_0)+p_3(q_1,q_0,1)\Big) = \phi \cdot \sum_{q_1,q_0\geq 0} p_4(q_1,q_0,1), \qquad (78) \\
& p(q_1,q_0) = p_1\cdot p(q_1,q_0-1)\cdot \mathbb{1}[q_0\geq 1] + p_2(q_1+1,q_0) \\
& \qquad + \big(p_2\cdot p(q_1,q_0+1) - p_2(q_1,q_0+1)\big) + p_2\cdot p(q_1,q_0)\cdot \mathbb{1}[q_0=0] \\
& \qquad + p_3(q_1-1,q_0,0)\cdot \mathbb{1}[q_1\geq 1] + p_3(q_1,q_0-1,1)\cdot \mathbb{1}[q_0\geq 1] \\
& \qquad + p_4(q_1+1,q_0,0) + p_4(q_1,q_0+1,1) + p_4(q_1,q_0,1)\cdot \mathbb{1}[q_0=0] \\
& \qquad + \big(p_4\cdot p(q_1,q_0) - p_4(q_1,q_0,0) - p_4(q_1,q_0,1)\big),\ \forall\, q_1,q_0\geq 0, \\
& \sum_{q_1,q_0\geq 0} p(q_1,q_0) = 1, \\
& p_2(q_1,q_0) \leq p_2\cdot p(q_1,q_0),\ \forall\, q_1,q_0\geq 0, \\
& p_3(q_1,q_0,0)+p_3(q_1,q_0,1) = p_3\cdot p(q_1,q_0),\ \forall\, q_1,q_0\geq 0, \\
& p_4(q_1,q_0,0)+p_4(q_1,q_0,1) \leq p_4\cdot p(q_1,q_0),\ \forall\, q_1,q_0\geq 0, \\
& p_2(q_1,q_0) = 0,\ \forall\, q_1=0, q_0\geq 0, \\
& p_2(q_1,q_0) = 0,\ \forall\, q_0\geq 0, q_1=0, \\
& p_4(q_1,q_0,0) = 0,\ \forall\, q_1=0, q_0\geq 0, \\
& p(q_1,q_0), p_2(q_1,q_0), p_3(q_1,q_0,0), p_3(q_1,q_0,1)\geq 0,\ \forall\, q_1,q_0\geq 0, \\
& p_4(q_1,q_0,0), p_4(q_1,q_0,1)\geq 0,\ \forall\, q_1,q_0\geq 0.
\end{aligned}
$$

[24] By doing this, the job can get served immediately rather than waiting till being served in the shared pool.

E Justification for Fluid Relaxation in Fluid Mean-Field Approximation

The main reason that we incorporate fluid relaxation in the FMFE framework is to simplify analysis. The mean-field approximation simplifies a server's decision problem significantly because it eliminates the need to track interactions with the system to form beliefs about the system. This approximation, together with Lemma 3.1, allows us to reduce the server's decision problem to a Markov decision problem (1).

Although (1) can be formulated as a two-dimensional stochastic DP (where the queue length Q_i and token count S_i are the states), it is still challenging to solve. To better handle a server's problem, we introduce a second level of approximation, the fluid relaxation. Specifically, we allow the number of tokens a server holds to be negative and go beyond the upper bound C, and we require only the flow balance constraint of the tokens. Lemma 3.2 shows that the fluid mean-field problem can be formulated as a one-dimensional DP (2) in which an optimal policy depends only on the queue length Q_i. Moreover, a closed-form solution to (2) can be characterized, which significantly simplifies subsequent analysis.

In this section, we provide further justification for the fluid relaxation step in the FMFE framework. We provide both theoretical (Appendix E.1) and numerical (Appendix E.2) justifications.

E.1 Theoretical Justification for the Fluid Relaxation Step

We first provide a theoretical justification for fluid relaxation. Specifically, we show that the fluid relaxation problem (2) provides an accurate approximation to a server's best response strategy in the mean-field problem (1) when the token amount upper bound C is large.

To do so, suppose a server $i \in [N]$ perceives the shared pool waiting time as $w \leq 1$.[25] Recall that $V(w)$ and $V^{\mathrm{F}}(w)$ denote the optimal values of problems (1) and (2), which correspond to the discrete-time models of the mean-field problem and its fluid relaxation counterpart, respectively. Therefore, the server's optimal performance in the continuous-time versions of these problems are $(1+\lambda)V(w)$ and $(1+\lambda)V^{\mathrm{F}}(w)$, respectively. Additionally, note that $V^{\mathrm{F}}(w) \leq V(w)$, because (2) is a fluid relaxation of (1).

Following the notation from Section 6.1, let π^{F} denote the optimal policy to (2), and let $\bar{\pi}^{\mathrm{F}}$ denote the modified version of policy π^{F} that ensures the server's token count remains non-negative.[26] Let $V(\pi^{\mathrm{F}})$ and $V(\bar{\pi}^{\mathrm{F}})$ represent the performance of policies π^{F} and $\bar{\pi}^{\mathrm{F}}$, respectively, in the continuous-time model. We have that $V(\pi^{\mathrm{F}}) = (1+\lambda)V^{\mathrm{F}}(w)$, because policy π^{F} is optimal to (2), and that $(1+\lambda)V(w) \leq V(\bar{\pi}^{\mathrm{F}})$, because policy $\bar{\pi}^{\mathrm{F}}$ is feasible to (1).

The following relationship then holds (with proof provided below):

$$V(\pi^{\mathrm{F}}) = (1+\lambda)V^{\mathrm{F}}(w) \leq (1+\lambda)V(w) \leq V(\bar{\pi}^{\mathrm{F}}) \leq V(\pi^{\mathrm{F}}) + \frac{M_3}{C}\,, \tag{79}$$

where M_3 is the constant specified in Lemma 6.2 that depends only on the values of λ and ϕ. From (79), we have that $0 \leq V(w) - V^{\mathrm{F}}(w) \leq \frac{M_3}{1+\lambda} \cdot \frac{1}{C}$. Therefore, the fluid relaxation is asymptotically tight as the token amount upper bound C becomes large.

Proof of (79). We only need to prove the last inequality in (79), which we do now. Let $Q_i(\infty)$ denote the queue length of server i in the stationary distribution, and $\mathbb{P}[X_i(\infty) = 1]$ the stationary rate of requesting help, under policy π^{F}. Analogously, let $\bar{Q}_i(\infty)$ denote the queue length of server

[25] The analysis can be extended to the case of $w \leq \bar{w}$, where $\bar{w} \in \mathbb{R}_+$ is a constant, using the same approach.

[26] Specifically, under policy $\bar{\pi}^{\mathrm{F}}$, the server requests help only when it has tokens and offers help only when its token count is below the upper bound C, and follows the policy π^{F} whenever possible.

i in the stationary distribution, and $\mathbb{P}[\bar{X}_i(\infty) = 1]$ the stationary rate of requesting help, under policy $\bar{\pi}^{\mathrm{F}}$.

The time-average total cost of server i consists of three components: (i) the job processing cost, (ii) the holding cost for jobs in the queue, and (iii) the waiting cost for jobs routed to the shared pool. First, since both policies π^{F} and $\bar{\pi}^{\mathrm{F}}$ satisfy the flow balance constraint of the tokens (i.e., the expected rates of earning and spending tokens are equal), the time-average job processing costs are identical under both policies, which equal $c \cdot \lambda$ (please refer to the proof of Lemma C.2 for details). Therefore, the following holds:

$$V(\bar{\pi}^{\mathrm{F}}) = c\lambda + \bar{Q}_i(\infty) + w \cdot \mathbb{P}[\bar{X}_i(\infty) = 1],$$
$$V(\pi^{\mathrm{F}}) = c\lambda + Q_i(\infty) + w \cdot \mathbb{P}[X_i(\infty) = 1].$$

The difference in costs between the two policies is thus given by:

$$V(\bar{\pi}^{\mathrm{F}}) - V(\pi^{\mathrm{F}}) = \Big(\bar{Q}_i(\infty) - Q_i(\infty)\Big) + w \cdot \Big(\mathbb{P}[\bar{X}_i(\infty) = 1] - \mathbb{P}[X_i(\infty) = 1]\Big) \leq \frac{M_3}{C},$$

where the inequality follows from Lemma 6.2. □

E.2 Numerical Justification for the Fluid Relaxation Step

From (79), we also have that $0 \leq V(\bar{\pi}^{\mathrm{F}}) - (1+\lambda)V(w) \leq \frac{M_3}{C}$. Therefore, policy $\bar{\pi}^{\mathrm{F}}$ is asymptotically optimal to the mean-field problem (1) when the token amount upper bound C is large. Intuitively, because $\bar{\pi}^{\mathrm{F}}$ is approximately optimal to (1), it should be close to the optimal policy of (1), which is analytically intractable. In the following, we numerically solve for the optimal policy of (1) and compare it with policy $\bar{\pi}^{\mathrm{F}}$.

We consider a problem instance with a job arrival rate of $\lambda = 0.8$ and a job processing cost of $c = 1$. The token-earning probability ϕ is set to λ (which is optimal by Theorem 5.2), and we assume that the server presumes the shared pooling waiting time to be $w = 0.5$.

According to Corollary 4.3, the optimal policy π^{F} for the fluid mean-field problem (2) is to implement complete resource pooling. Specifically, with policy π^{F}, the server (i) requests help for all incoming jobs, and (ii) serves a job from its queue if one is present and offers help to the shared pool otherwise, when a capacity unit arrives. The long-run average total cost of policy π^{F} is:

$$V(\pi^{\mathrm{F}}) = (1 + \lambda)V^{\mathrm{F}}(w) = c \cdot \lambda + \lambda \cdot w = 1.2.$$

The modified policy $\bar{\pi}^{\mathrm{F}}$, which is feasible to the mean-field problem (1), mimics policy π^{F} whenever possible and can be described as follows:

1. When a job arrives, request help if the token amount is positive; otherwise, add the job to its queue.
2. When a capacity unit arrives, serve a job from its queue if one is present; otherwise, offer help to the shared pool as long as the token amount is below the upper bound C.

Finally, let $\bar{\pi}$ denote the optimal policy of (1), which adapts to both the server's queue length and the number of tokens it holds. In the following, we solve for the optimal policy $\bar{\pi}$ numerically for two values of token amount upper bound, $C = 1000$ and $C = 5000$, and compare it with policy $\bar{\pi}^{\mathrm{F}}$.

E.2.1 Numerical Results for $C = 1000$

In this case, the long-run average total costs of the optimal policy $\bar{\pi}$ (which equals $(1+\lambda)V(w)$) and policy $\bar{\pi}^{\mathrm{F}}$ are presented in Table 1(a). The optimal policy $\bar{\pi}$ can be described as follows:

1. When a job arrives, request help if the number of tokens is positive and any of the following three conditions are met:
   (a) The queue contains at least two jobs.
   (b) The queue contains one job, and the server has at least 8 tokens.
   (c) The queue is empty, and the server has at least 58 tokens.

   Otherwise, the server adds the job to its queue.
2. When a capacity unit arrives, serve a job from its queue if one is present; otherwise, offer help to the shared pool as long as the token count is below the upper bound C.

Let $Q_i \in \mathbb{N}$ and $S_i \in \mathbb{N}$ denote the queue length and token count of server i, respectively. Based on the above, policy $\bar{\pi}$ differs from policy $\bar{\pi}^{\mathrm{F}}$ only when a job arrives and the server's state $(Q_i, S_i) \in A$, where set A is defined as follows:

$$A \triangleq \Big\{ \big(Q_i, S_i\big) : \, Q_i = 0 \text{ and } 1 \le S_i \le 57 \;\text{ or }\; Q_i = 1 \text{ and } 1 \le S_i \le 7 \Big\}.$$

Finally, let $\bar{\mathbb{P}}_\infty$ denote the stationary probability under policy $\bar{\pi}$. We have:

$$\bar{\mathbb{P}}_\infty(A) = 0.0052 \ll 1.$$

Thus, the dynamics of policies $\bar{\pi}$ and $\bar{\pi}^{\mathrm{F}}$ are nearly identical.[27]

$(1+\lambda)V^{\mathrm{F}}(w)$	$(1+\lambda)V(w)$	$V(\bar{\pi}^{\mathrm{F}})$
1.2000	1.2021	1.2179

(a) $C = 1000$

$(1+\lambda)V^{\mathrm{F}}(w)$	$(1+\lambda)V(w)$	$V(\bar{\pi}^{\mathrm{F}})$
1.2000	1.2004	1.2036

(b) $C = 5000$

Table 1: Performances of different policies. Recall that $V(\pi^{\mathrm{F}}) = (1+\lambda)V^{\mathrm{F}}(w)$ and $V(\bar{\pi}) = (1+\lambda)V(w)$.

E.2.2 Numerical Results for $C = 5000$

In this case, the long-run average total costs of the optimal policy $\bar{\pi}$ and policy $\bar{\pi}^{\mathrm{F}}$ are presented in Table 1(b). The optimal policy $\bar{\pi}$ can be described as follows:

[27] We note that the dynamics of tokens under either policy $\bar{\pi}$ or $\bar{\pi}^{\mathrm{F}}$ is approximately a balanced random walk on the interval $[0 : C]$, as the expected rates of earning and spending tokens are equal.

1. When a job arrives, request help if the number of tokens is positive and any of the following three conditions are met:
   (a) The queue contains at least two jobs.
   (b) The queue contains one job, and the server has at least 7 tokens.
   (c) The queue is empty, and the server has at least 71 tokens.

   Otherwise, the server adds the job to its queue.
2. When a capacity unit arrives, serve a job from its queue if one is present; otherwise, offer help to the shared pool as long as the token count is below the upper bound C.

Let $Q_i \in \mathbb{N}$ and $S_i \in \mathbb{N}$ denote the queue length and token count of server i. Policy $\bar{\pi}$ differs from policy $\bar{\pi}^{\mathrm{F}}$ when a job arrives and the server's state $(Q_i, S_i) \in A$, where set A is defined as follows:

$$A \triangleq \Big\{ \big(Q_i, S_i\big) : \, Q_i = 0 \text{ and } 1 \le S_i \le 70 \;\text{ or }\; Q_i = 1 \text{ and } 1 \le S_i \le 6 \Big\}.$$

Let $\bar{\mathbb{P}}_\infty$ denote the stationary probability under policy $\bar{\pi}$; we have:

$$\bar{\mathbb{P}}_\infty(A) = 0.0010 \ll 1.$$

Thus, the probability that the server is in a state where the two policies differ is negligible.

From the above numerical examples, the fluid relaxation and the resulting policy $\bar{\pi}^{\mathrm{F}}$ accurately approximates a server's strategic behavior in the mean-field problem (1), especially when the token amount upper bound C is large.