Learning Mean-Field Control for Delayed Information Load Balancing in Large Queuing Systems

Anam Tahir*
Kai Cui*
Heinz Koeppl
{anam.tahir,kai.cui,heinz.koeppl}@tu-darmstadt.de
Department of Electrical Engineering and Information Technology
Technische Universität Darmstadt, Germany

ABSTRACT

Recent years have seen a great increase in the capacity and parallel processing power of data centers and cloud services. To fully utilize the said distributed systems, optimal load balancing for parallel queuing architectures must be realized. Existing state-of-the-art solutions fail to consider the effect of communication delays on the behaviour of very large systems with many clients. In this work, we consider a multi-agent load balancing system, with delayed information, consisting of many clients (load balancers) and many parallel queues. In order to obtain a tractable solution, we model this system as a mean-field control problem with enlarged state-action space in discrete time through exact discretization. Subsequently, we apply policy gradient reinforcement learning algorithms to find an optimal load balancing solution. Here, the discrete-time system model incorporates a synchronization delay under which the queue state information is synchronously broadcasted and updated at all clients. We then provide theoretical performance guarantees for our methodology in large systems. Finally, using experiments, we prove that our approach is not only scalable but also shows good performance when compared to the state-of-the-art power-of-d variant of the Join-the-Shortest-Queue (JSQ) and other policies in the presence of synchronization delays.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**; • **Networks** → **Network resources allocation**.

KEYWORDS

load balancing; parallel systems; mean-field control; reinforcement learning

ACM Reference Format:
Anam Tahir, Kai Cui, and Heinz Koeppl. 2022. Learning Mean-Field Control for Delayed Information Load Balancing in Large Queuing Systems. In *51st International Conference on Parallel Processing (ICPP '22), August 29-September 1, 2022, Bordeaux, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3545008.3545025

*These authors contributed equally to this research.

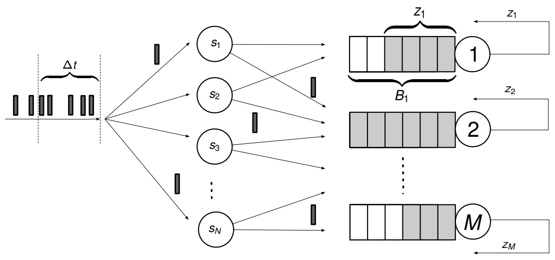

Figure 1: Our system model consists of N clients and M parallel servers. Jobs arriving in a certain time interval Δt are assigned to the clients, which consequently assign them to one of a few sampled servers based on some policy. Arrows from each client indicate the $d = 2$ servers randomly sampled by each client at the current epoch.

1 INTRODUCTION

Load balancing in large queuing systems has been of great interest in the field of parallel processing and has yielded many successful distributed algorithms such as Join-the-Shortest-Queue (JSQ), Shortest-Expected-Delay (SED) [34, 40, 41] and many others, see also [39] for a recent review. JSQ and SED have been designed for asynchronous systems with a central dispatcher (agent / client) assigning jobs (packets) to M parallel servers (queues) under the assumption that the dispatcher can obtain instantaneous, accurate and synchronized information of the queue lengths at all times. In practice, both instant information and centralized dispatching are not realistic, especially if the number of queues $M \gg 1$ is large.

To remedy this scalability issue, the power-of-d versions JSQ(d) and SED(d) of JSQ and SED [26] let the dispatcher sample only $d \leq M$ out of M servers randomly and then allocate the job to the sampled server with shortest expected processing time. However, JSQ(d) and SED(d) nonetheless assume instant and accurate information of the state of those d servers, which remains unrealistic due to both the distributed nature of the system and computational overheads introducing latency. The problem is only exacerbated in a multiple client scenario where all clients access simultaneously. Hence, to model a more realistic system, it is of importance to take communication delays Δt into account. In [25], it was shown that JSQ fails when $\Delta t > 0$ mainly due to a phenomenon known as 'herd behaviour': Multiple clients assigning jobs at the same time would consider the same subset of servers with few jobs, and thus all clients will end up assigning to the same servers. This eventually leads to higher response times and, in the case of finite queues, job drops. Though JSQ(d) ameliorates this issue somewhat since it is highly unlikely for small d and large M that many clients will randomly choose the same servers, the technique nonetheless remains suboptimal under delayed information. Indeed, as $\Delta t \to \infty$, a completely random allocation to one of the servers becomes optimal [26]. However, when the delay Δt lies between 0 and ∞, the optimal policy must lie in-between, which will be the main focus of this work.

In this paper, we shall consider a multi-agent system of N clients and M servers with $N \gg M \gg 1$ and communication delay. For scalability, each client samples d of the M servers uniformly at random using the power-of-d method. The discretized system can be understood as a delayed periodic or synchronously updating system. Most importantly, as a result of delayed information, the number of agents will make a difference as opposed to the delay-free case, since each agent may see a different subset of information. In order to scale to a great number of clients and servers, we will apply mean-field theory, analogous to fluid limits $M \to \infty$, that is used to tractably model and assess systems with many queues. Fluid limits were used to study the performance of scheduling algorithms like JSQ and JSQ(d) in terms of sojourn time and average queue length [10, 26, 29]. However, models including delayed information still remain an open problem [22], in particular in the presence of many clients. One work with similar system model and synchronization delays is given in [43], though they instead consider finitely many servers with infinite buffer sizes where the multiple clients use their local, asynchronous estimates of queue lengths to perform scheduling. This idea of using local client memory has also been proposed in [3, 38], however only for a single client.

More generally, the same tractability issue for large systems has led to the increasing popularity of general (competitive) mean-field games (MFG) [16, 19, 32] and their cooperative counterpart of mean-field control (MFC) [2, 4, 5, 9, 11], wherein a system with large numbers of interchangeable and indistinguishable agents is converted into a system where one representative agent is interacting with the distribution (mean-field) of other agents. Here, there has been great recent focus on learning-based solution algorithms for MFGs [1, 8, 14, 36] and MFC [7, 13, 28]. We will similarly apply the enlarged state-action space technique for MFCs (see e.g. [13]), its associated dynamic programming principle as well as reinforcement learning in order to find optimal load balancing policies for otherwise intractably large system. While reinforcement learning (RL) [37], so-far has found great success e.g. in games [6, 27], robotics [17] or communication and queuing networks [1, 23], in the case of multiple agents, there still remain many challenges in multi-agent reinforcement learning (MARL) such as intractability for large numbers of agents [42]. RL itself has long since been used in numerous works – though not in the context of mean-field control – to find an optimal load balancing policy. For examples, see [18, 20, 35, 41] and references therein. The combination with mean-field control allows for tractable solution of very large load balancing systems and shall be the subject of our studies. We will similarly formulate a synchronous system model with delay by assuming $N \gg M \to \infty$, which will allow us to apply reinforcement learning to the otherwise difficult to solve optimal load balancing problem. Although our model shares similarities in concept to MFC, it does not immediately fit into the framework of conventional MFC, as we not only derive the discrete-time mean-field model starting from an underlying continuous-time dynamic, but at the same time take a double limit of infinitely many queues and agents. While, existing MFC frameworks typically focus only on the limit of infinitely many agents without external dynamics of non-agent-bound (queue) states.

To summarize our contributions, (i) we consider a model not only with synchronous communication delay, but also under the limit of both many clients and many servers, stepping towards a general solution for the outstanding problem of scalable load balancing under delayed information [43]; (ii) we formulate the system as a mean-field control problem, introducing a decision hierarchy to obtain a standard Markov decision process amenable to standard solution techniques; (iii) we theoretically show the well-motivatedness of our limiting model by proving that the limiting system performance is reached with arbitrary precision in sufficiently large systems; and (iv) we apply reinforcement learning to solve the otherwise difficult-to-solve Markov decision process with continuous, high-dimensional action space, at a complexity independent of the number of clients N and servers M. We find that, as the synchronization delay increases to an intermediate value, the choice of the shortest queues or fully random assignment becomes suboptimal and is outperformed by a learned policy. This policy can either be learned offline for a given system with known parameters, or applied online to learn optimal assignments in live systems. Our claims are supported both theoretically and experimentally and ablated for the case where our formal assumption $N \gg M$ is violated, giving us a good solution for large-scale load balancing systems with many clients and servers.

2 LOAD BALANCING WITH DELAY

In this section, we will introduce the problem setting that will motivate our formulation. An overview of the considered load balancing system is given in Figure 1.

We consider N clients and M servers, where each server has its own queue with limited buffer capacity. Jobs arrive randomly according to a Markov modulated Poisson process – modelling e.g. changing load factors throughout a day – with rate $\lambda_t M$ and are divided uniformly among clients, which will allocate the jobs to servers for processing. In accordance with the power-of-d technique, clients shall randomly select d out of M queues and – according to some policy to be optimized – send their jobs to a selection of these d queues, where $d \ll M$. On the queuing side of our system model, we have M parallel and homogeneous servers in the system with service rates α. The queues are finite with a maximum buffer capacity B and the jobs in the queues are served in a first-in-first-out (FIFO) manner. Each server sends back its queue filling status, which is then used by the clients to make their decision for the next incoming jobs. The number of jobs that are currently in each queue together make up the state of the environment. Our goal is to *minimize overall job drops* under decentralized decision-making by each client, e.g. like in edge computing scenarios.

We will assume that our system operates synchronously and broadcasts updates of sampled queue states to dispatchers only once every fixed time interval. Thus, in the following we will model our system at discrete decision epochs $\{0, \Delta t, 2 \cdot \Delta t, \dots\}$ for some synchronization delay $\Delta t > 0$, after each of which the clients will sample d new queues and keep this selection of d queues for the entire duration of that decision epoch. Not only will this allow us to incorporate communication delays, but it will also lead to significantly less sampling of server states by the clients, as each client is only required to sample d servers in every decision epoch. Another advantage of this approach is that the resulting discretized Markov decision process will allow us to apply powerful and well-established reinforcement learning algorithms, which to this date have been extensively developed for discrete-time models.

2.1 Mathematical model

Notation. *Let $\mathcal{S}$ be a discrete space equipped with the discrete topology. Define by $\mathcal{P}(\mathcal{S})$ the space of (Borel) probability measures on $\mathcal{S}$, equipped with the l_1-norm $\|\mu - \nu\|_1 = \sum_{s \in \mathcal{S}} |\mu(s) - \nu(s)|$. To keep notation simple, we denote the probability mass function of $\nu \in \mathcal{P}(S)$ by $\nu(\cdot)$. In the following, we denote random variables of the finite system with superscript N, M, of the infinite-agent version with superscript M and of the limiting mean-field system without superscript.*

Formally, the N-agent M-queue system could be considered a multi-agent Markov Decision Process (MMDP) for $N, M \in \mathbb{N}$, i.e. the cooperative and fully observable case. See e.g. [30] for a review of possible multi-agent problem formulations. In principle, one could even consider competitive or partially observed cases. However, the resulting limiting mean-field systems will be significantly more complex and thus remain outside of our scope. Instead, we will in the following consider a decentralized control setting where agents, due to the symmetry of our model, shall act depending on the current distribution of queue states.

Define $\mathcal{Z} \coloneqq \{0, \dots, B\}$ as the finite queue state space, i.e. each server can contain at most B jobs in its queue. The agent state space shall be denoted as $\mathcal{X} \coloneqq \{1, \dots, M\}^d$, i.e. a selection of d random queues. Although we could disallow repeated queue selections, it will make no difference in sufficiently large systems and adds unnecessary notational complexity. Finally, each agent can choose as an action its choice of one of d randomly sampled accessible queues, i.e. the action space is defined as the d possible queue choices $\mathcal{U} \coloneqq \{1, \dots, d\}$. At any decision epoch $t = 0, 1, \dots$, the states and actions of agents $i = 1, \dots, N$, are random variables denoted by $x_t^{N,M,i} \equiv (x_{t,1}^{N,M,i}, \dots, x_{t,d}^{N,M,i}) \in \mathcal{X}$ and $u_t^{N,M,i} \in \mathcal{U}$, and similarly the state of each queue $j = 1, \dots, M$ is denoted by $z_t^{N,M,j} \in \mathcal{Z}$ with $z_0^{N,M,j} \sim \nu_0 \in \mathcal{P}(\mathcal{Z})$ from some initial distribution ν_0. Additionally, $\lambda_t^{N,M} > 0$ – the arrival rate parameter – will be modulated as an independent discrete-time Markov chain with state space Λ, i.e.

$$\lambda_{t+1}^{N,M} \sim P_\lambda(\lambda_t^{N,M}) \tag{1}$$

for some arbitrary transition kernel P_λ.

Due to symmetry of the problem, for sufficiently many agents, the information about each specific queue's state becomes irrelevant to the problem. Thus, we assume some common, shared policy of the form $\pi_t \colon \mathcal{P}(\mathcal{Z}) \times \mathcal{Z}^d \times \Lambda \to \mathcal{P}(\mathcal{U})$ for all agents, acting on the current $\mathcal{P}(\mathcal{Z})$-valued random empirical queue state distribution

$$\mathbb{H}_t^{N,M} \coloneqq \frac{1}{M} \sum_{j=1}^{M} \delta_{z_t^{N,M,j}} \tag{2}$$

with Dirac measure δ, the sampled queue states, and the current arrival rate. In practice, we may also drop dependence on the current arrival rate and empirical distribution, or estimate e.g. the empirical queue state distribution by sampling a subset of random queues, though both will complicate the theoretical analysis of the limiting MFC problem, as it would not be possible to formulate the limiting system as a standard, fully-observed Markov decision process.

The dynamics for each agent i are thus given by

$$x_t^{N,M,i} \sim \otimes_{k=1}^{d} \mathrm{Unif}(\{1, \dots, M\}), \tag{3}$$

$$u_t^{N,M,i} \sim \pi_t \left(\mathbb{H}_t^{N,M}, (z_t^{N,M,x_{t,1}^i}, \dots, z_t^{N,M,x_{t,d}^i}), \lambda_t^{N,M} \right), \tag{4}$$

i.e. at each decision epoch, the agents decide to which of their d randomly sampled, accessible queues they decide to send their jobs to. For simplicity of exposition, this choice of destination is deterministic, though in our experiments we shall allow randomization for each packet. As a result, starting with $z_0^{N,M,j} \sim \nu_0 \in \mathcal{P}(\mathcal{Z})$ for each queue j and some initial queue state distribution ν_0, for any queue j, the next queue state $z_{t+1}^{N,M,j}$ is obtained from the previous state $z_t^{N,M,j}$ by simulating a $\mathcal{Z}$-valued continuous-time Markov chain for Δt time units, beginning with $z_t^{N,M,j}$ and decrementing or incrementing by 1 at departure rate $\alpha > 0$ and arrival rate

$$\lambda_t^{N,M,j} = M \lambda_t^{N,M} \cdot \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{d} \mathbf{1}_{x_{t,k}^{N,M,i} = j} \mathbf{1}_{u_t^{N,M,i} = k} \tag{5}$$

respectively, ignoring jumps above B or below 0. Any arrivals beyond B are counted in the average number of dropped packets

$$D_t^{N,M} = \frac{1}{M} \sum_{j=1}^{M} D_t^{N,M,j} \tag{6}$$

per queue j during each decision epoch t, which will constitute our objective through the discounted infinite-horizon objective

$$J^{N,M}(\pi) = \mathbb{E}\left[-\sum_{t=0}^{\infty} \gamma^t D_t^{N,M} \right] \tag{7}$$

to be maximized with discount factor $\gamma \in (0, 1)$.

Note that we can rewrite (5) as

$$\lambda_t^{N,M,j} = M \lambda_t^{N,M} \int_{\mathcal{X} \times \mathcal{U}} \sum_{k=1}^{d} \mathbf{1}_{x_k = j} \mathbf{1}_{u = k} \, \mathbb{G}_t^{N,M}(\mathrm{d}x, \mathrm{d}u) \tag{8}$$

with the $\mathcal{P}(\mathcal{X} \times \mathcal{U})$-valued empirical agent state-action distribution

$$\mathbb{G}_t^{N,M} \coloneqq \frac{1}{N} \sum_{i=1}^{N} \delta_{x_t^{N,M,i}, u_t^{N,M,i}}. \tag{9}$$

Intuitively speaking, when $N \gg M \gg 1$, this empirical distribution becomes deterministic and we need not track each queue state, but only their distribution. Similarly, only the overall distribution of all agent choices will matter, leading to the prospective limiting mean-field model derived in the sequel.

2.2 Infinite-agent limit

In the infinite-agent limit where $N \to \infty$, we obtain a limiting control problem with random external states (queue states). Consider the evolution of the $\mathcal{P}(\mathcal{Z})$-valued empirical queue state distribution

$$\mathbb{H}_t^M \coloneqq \frac{1}{M} \sum_{j=1}^{M} \delta_{z_t^{M,j}} \tag{10}$$

as $N \to \infty$. Conditional on the queue states and arrival rate, $(x_t^{M,i}, u_t^{M,i})_{i=1,\dots,N}$ are i.i.d. Therefore, it will be sufficient to consider only the statistics of a representative agent. By the law of large numbers, we obtain the deterministic agent state distribution

$$\tilde{\mu}_t \coloneqq \otimes_{k=1}^d \mathrm{Unif}(\{1,\dots,M\}) \in \mathcal{P}(\mathcal{X}) \tag{11}$$

of agents by (3). The $\mathcal{P}(\mathcal{X} \times \mathcal{U})$-valued agent state distribution

$$\mathbb{G}_t^M \coloneqq \mathcal{G}_t^M(\tilde{\mu}_t, h_t) \tag{12}$$

thus depends on $h_t \coloneqq \pi_t(\mathbb{H}_t^M, \cdot, \lambda_t^M)$, where we define

$$\mathcal{G}_t^M(\tilde{\mu}, h)(x,u) \coloneqq \tilde{\mu}(x) h(u \mid (z_t^{M,x_1}, \dots, z_t^{M,x_d})). \tag{13}$$

We observe that this state-action distribution is sufficient for characterizing system behaviour: Conditional on fixed λ_t^M and $\{z_t^{M,1}, \dots, z_t^{M,M}\}$, the arrival rate in (5) becomes

$$\lambda_t^{M,j} = M\lambda_t^M \mathbb{E}\left[\sum_{k=1}^{d} \mathbf{1}_{x_{t,k}^{M,1}=j} \mathbf{1}_{u_t^{M,1}=k}\right] \tag{14}$$

$$= M\lambda_t^M \int_{\mathcal{X}\times\mathcal{U}} \sum_{k=1}^{d} \mathbf{1}_{x_k=j} \mathbf{1}_{u=k}\, \mathbb{G}_t^M(\mathrm{d}x, \mathrm{d}u) \tag{15}$$

by the law of large numbers, similar to (8). In other words, the empirical agent state-action distribution $\mathbb{G}_t^{N,M}$ is replaced by the limiting distribution $\mathbb{G}_t^M$.

2.3 Infinite-queue limit

Finally, we derive the mean-field model in the limit as $M \to \infty$, i.e. formally $N \gg M \gg 1$. The random queue states are now replaced by the queue state distribution denoted by $\nu_t \in \mathcal{P}(\mathcal{Z})$. Therefore, each agent state $x_t^i \in \mathcal{X}$ is now also replaced by the anonymous queue state $\bar{z}_t^i \in \mathcal{Z}^d$ instead of the actual queue index. The queue state distribution deterministically induces the agent state distribution

$$\mu_t \coloneqq \otimes_{k=1}^d \nu_t \in \mathcal{P}(\mathcal{Z}^d) \tag{16}$$

by assigning the d-dimensional product measure $\mu_t(\bar{z}) = \Pi_{k=1}^d \nu_t(\bar{z}_k)$ for any $\bar{z} \equiv (\bar{z}_1, \dots, \bar{z}_d) \in \mathcal{Z}^d$. For any decision rule $h_t = \pi_t(\nu_t, \cdot, \lambda_t)$, this agent state distribution induces a state-action distribution

$$\mathbb{G}_t \coloneqq \mu_t \otimes h_t \in \mathcal{P}(\mathcal{Z}^d \times \mathcal{U}). \tag{17}$$

Now consider the random amount of arriving packets $P \sim \mathrm{Pois}(M\lambda_t \Delta t)$ in a time slot Δt. Since $N \gg M$ implies $N \gg P$, the probability of any single agent receiving more than one packet is negligible. This implies that almost all packets' destination queues will be i.i.d. random variables. As a result, since packets arrive with rate $M\lambda_t$ and i.i.d. destinations, for any $z \in \mathcal{Z}$, packets will equivalently arrive with rate $M\lambda_t'(z)$ in queues with state $z \in \mathcal{Z}$ by Poisson thinning, where

$$\lambda_t'(z) = \lambda_t \int_{\mathcal{Z}^d \times \mathcal{U}} \mathbf{1}_{\bar{z}_u = z}\, \mathbb{G}_t(\mathrm{d}\bar{z}, \mathrm{d}u). \tag{18}$$

By symmetry, these packets arrive uniformly at random in any arbitrary specific fixed queue in state z. For any specific queue with state z, the probability of assigning such a packet to that queue is therefore $\frac{1}{M\nu_t(z)}$, which results in an equivalent queue packet arrival rate of

$$\lambda_t(z) \coloneqq \frac{M\lambda_t'(z)}{M\nu_t(z)} = \frac{\lambda_t'(z)}{\nu_t(z)}. \tag{19}$$

The informal derivation until now will be motivated more rigorously in Section 3 and numerically in Section 4.

2.4 Exact discretization

The final step is to formulate a discrete-time optimal control problem from the delayed, synchronous system that allows for application of standard optimal control techniques such as reinforcement learning. To discretize the mean-field system exactly at times $\{0, \Delta t, 2 \cdot \Delta t, \dots\}$, we generate the master equations for the evolution of a single queue's state over time between each of the discretization time points. The procedure is done analogously for the pre-limit systems. Consider a queue in state $z \in \mathcal{Z}$ at the beginning of a decision epoch t. Then, for any h_t, we define a $\mathcal{Z}$-valued continuous-time Markov chain y through $y(0) = z$ and formulate its Kolmogorov forward equations

$$\dot{\mathbf{P}}^z = \mathbf{Q}^z \mathbf{P}^z, \quad \mathbf{P}^z(0) = \mathbf{e}_z \tag{20}$$

for the vector of queue state probabilities $\mathbf{P}^z(\tau) \in [0,1]^{\mathcal{Z}}$ at times $\tau \in [0, \Delta t]$ with

$$P_{z'}^z(\tau) \equiv \mathbb{P}(y(\tau) = z'), \quad \forall z' \in \mathcal{Z} \tag{21}$$

and the transposed transition rate matrix $\mathbf{Q}^z \coloneqq \mathbf{Q}(\nu_t, z) \in \mathbb{R}^{\mathcal{Z}\times\mathcal{Z}}$ where $\mathbf{Q}(\nu, z)$ is defined by

$$\mathbf{Q}(\nu, z)_{i,i-1} = \lambda_t(\nu, z) \coloneqq \frac{1}{\nu(z)} \lambda_t \int_{\mathcal{Z}^d \times \mathcal{U}} \mathbf{1}_{\bar{z}_u = z}\, (\nu \otimes h_t)(\mathrm{d}\bar{z}, \mathrm{d}u) \tag{22}$$

in accordance with (16) - (19), $\mathbf{Q}(\lambda, z)_{i-1,i} = \alpha(z)$ for $i = 1, \dots, B$, $\mathbf{Q}(\lambda, z)_{i,i} = -\sum_j \mathbf{Q}(\lambda, z)_{j,i}$ for $i = 0, \dots, B$, and zero otherwise. Here, $\mathbf{e}_z$ denotes the z-unit vector.

Therefore, from the fraction $\nu_t(z)$ of queues in state $z \in \mathcal{Z}$ at time t, we will deterministically have the resulting fraction

$$\nu_{z,z'} = \nu_t(z) P_{z'}^z(\Delta t) \tag{23}$$

of queues with state $z \in \mathcal{Z}$ in resulting state $z' \in \mathcal{Z}$ at the end of the decision epoch Δt. In total, we therefore have

$$\nu_{t+1}(z') = \sum_{z\in\mathcal{Z}} \nu_{z,z'} = \sum_{z\in\mathcal{Z}} \nu_t(z) P_{z'}^z(\Delta t), \quad \forall z' \in \mathcal{Z}. \tag{24}$$

Computing the expected packet drops D_t^z per queue with state $z \in \mathcal{Z}$ is done analogously by

$$\dot{D}_t^z = \lambda_t(z) P_B^z, \quad D_t^z(0) = 0 \tag{25}$$

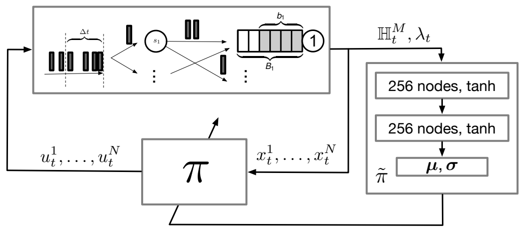

Figure 2: A schematic overview of the application of the upper-level mean-field control policy to the finite-client finite-server system. The upper-level policy $\tilde{\pi}$ returns a lower-level policy π for a given distribution of server states $\mathbb{H}_t^M$ and current arrival rate λ_t. The lower-level policy is then applied separately to each agent state x_t^i to obtain an action u_t^i.

resulting in a per-queue average packet loss of

$$D_t = \sum_{z \in \mathcal{Z}} \nu_t(z) D_t^z(\Delta t). \tag{26}$$

For exact computation of the terms in (24) - (26), observe that we have the linear matrix differential equation

$$\begin{bmatrix} \dot{\mathbf{P}}^z \\ \dot{D}_t^z \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{Q}^z & 0 \\ \lambda_t(\nu_t, z) \cdot \mathbf{e}_B^T & 0 \end{bmatrix}}_{\bar{\mathbf{Q}}^z \equiv \bar{\mathbf{Q}}(\nu_t, z)} \cdot \begin{bmatrix} \mathbf{P}^z \\ D_t^z \end{bmatrix} \tag{27}$$

where we define the extended rate matrices $\bar{\mathbf{Q}}(\nu_t, z)$ analogously to $\mathbf{Q}(\nu_t, z)$, and thus obtain exact discretization by

$$\begin{bmatrix} \mathbf{P}^z(\Delta t) \\ D_t^z(\Delta t) \end{bmatrix} = \exp(\bar{\mathbf{Q}} \Delta t) \cdot \begin{bmatrix} \mathbf{e}_z \\ 0 \end{bmatrix} \tag{28}$$

where $\exp(\cdot)$ denotes the matrix exponential.

2.5 Upper-level decision process

We can now obtain a Markov decision process (MDP) [31] with state space $\mathcal{P}(\mathcal{Z}) \times \Lambda$ and action space $\mathcal{H} := \{h : \mathcal{Z}^d \to \mathcal{P}(\mathcal{U})\}$, since we have states (λ_t, ν_t) and actions h_t following dynamics

$$(\lambda_{t+1}, \nu_{t+1}) \sim P_\lambda(\lambda_t) \otimes \delta_{T_\nu(\nu_t, \lambda_t, h_t)} \tag{29}$$

$$h_t = \tilde{\pi}_t(\nu_t, \lambda_t) \tag{30}$$

where the transition function T_ν deterministically maps to ν_{t+1} according to (24), and the actions are given by a deterministic 'upper-level' policy $\tilde{\pi} = \{\tilde{\pi}_t\}_{t \geq 0}$, where $\tilde{\pi}_t : \mathcal{P}(\mathcal{Z}) \times \Lambda \to \mathcal{H}$. Here, the randomness of the system stems from the random packet arrival rate λ_t. Finally, by (26), the objective becomes

$$J(\tilde{\pi}) = \mathbb{E}\left[-\sum_{t=0}^{\infty} \gamma^t D_t \right]. \tag{31}$$

The application of $\tilde{\pi}$ to the N-agent, M-queue case is visualized in Figure 2, i.e. each of the agents $i = 1, \ldots, N$ first computes the decision rule $h_t = \tilde{\pi}_t(\mathbb{H}_t^M, \lambda_t)$ according to the upper-level policy, and then samples its action $u_t^i \sim h_t(x_t^i)$.

For the obtained MDPs, since the expected cost function and the dynamics are continuous in the states and actions of the MFC MDP, it is known that the typical dynamic programming principle (i.e. Bellman equation) holds, and an optimal stationary deterministic policy will exist.

Proposition 1 ([15], Theorem 4.2.3). *There exists a stationary deterministic optimal policy $\tilde{\pi}$ that maximizes $J(\tilde{\pi})$.*

To find such a deterministic policy, an exact, closed-form solution is difficult due to the complexity of the associated transition model and continuous state and action spaces. Instead, we shall in the following employ well-established reinforcement learning techniques by exploring over stochastic policies $\tilde{\pi}_t : \mathcal{P}(\mathcal{Z}) \times \Lambda \to \mathcal{P}(\mathcal{H})$, with the random decision rules $h_t \sim \tilde{\pi}_t(\nu_t, \lambda_t)$ as actions of the MFC MDP, to find the desired optimal stationary deterministic policy.

It should be noted that in this section we have presented a system which has finite capacity queues with homogeneous servers, though this model can be extended to heterogeneous servers and infinite capacity queues, which we omit for space reasons.

3 THEORETICAL ANALYSIS

Although our formulated mean-field model is intuitively a good approximation of the finite system, in this section we shall make this connection rigorous. Note that our model does not immediately fit into standard MFC frameworks introduced in [13, 28], since we perform a double limit argument and continuous-to-discrete-time modelling. To verify the mean-field model, we shall show that performance in the finite system becomes arbitrarily close to the performance in the MFC system as long as the system is sufficiently large. Quantifying the error convergence rate more precisely is left to future work. For the following theoretical analysis, we shall consider the sequence of arrival rates $(\lambda_1, \lambda_2, \ldots)$ given a priori by conditioning on them, i.e. non-random $\lambda_t^{N,M} = \lambda_t^M = \lambda_t$.

Theorem 1. *The performance of the N, M system converges to the performance of the mean-field system under any stationary deterministic policy $\hat{\pi}$ as the system size becomes sufficiently large, i.e. for any $\varepsilon > 0$ there exists $N', M'(N') \in \mathbb{N}$ such that*

$$\left| J(\hat{\pi}) - J^{N,M}(\hat{\pi}) \right| < \varepsilon$$

for all $N > N', M > M'(N')$.

Proof. We will analyze

$$\left| J(\hat{\pi}) - J^{N,M}(\hat{\pi}) \right| \leq \left| J(\hat{\pi}) - J^M(\hat{\pi}) \right| + \left| J^M(\hat{\pi}) - J^{N,M}(\hat{\pi}) \right|$$
$$\leq \sum_{t=0}^{\infty} \gamma^t \left(\left| \mathbb{E}\left[D_t - D_t^M \right] \right| + \left| \mathbb{E}\left[D_t^M - D_t^{N,M} \right] \right| \right)$$

where D_t^M denotes the random loss of packets in the infinite-agent finite-queue system.

For the first term, consider $M \to \infty$ and observe that

$$\mathbb{E}[D_t] = \mathbb{E}\left[\int \left(\exp(\bar{\mathbf{Q}}(\nu_t, z)\Delta t) \cdot \begin{bmatrix} \mathbf{e}_z \\ 0 \end{bmatrix} \right)_{B+1} \nu_t(\mathrm{d}z) \right],$$

$$\mathbb{E}\left[D_t^M \right] = \mathbb{E}\left[\frac{1}{M} \sum_j \left(\exp(\bar{\mathbf{Q}}^{M,j} \Delta t) \cdot \begin{bmatrix} \mathbf{e}_{z_t^{M,j}} \\ 0 \end{bmatrix} \right)_{B+1} \right]$$
$$= \mathbb{E}\left[\int \left(\exp(\bar{\mathbf{Q}}(\mathbb{H}_t^M, z)\Delta t) \cdot \begin{bmatrix} \mathbf{e}_z \\ 0 \end{bmatrix} \right)_{B+1} \mathbb{H}_t^M(\mathrm{d}z) \right],$$

with the rate matrices $\bar{\mathbf{Q}}^{M,j}$ of the infinite-agent finite-queue system, where the last equality follows since the rates in the M-queue case for each queue j are indeed given by

$$\begin{aligned}
\lambda_t^{M,j} &= M\lambda_t \int_{\mathcal{X}\times\mathcal{U}} \sum_{k=1}^{d} \mathbf{1}_{x_k=j\wedge u=k}\, \mathbb{G}_t^M(\mathrm{d}x, \mathrm{d}u) \\
&= \lambda_t \sum_{k=1}^{d} \sum_{x\in\mathcal{X}} \sum_{u\in\mathcal{U}} \mathbf{1}_{x_k=j\wedge u=k} \frac{1}{M^{d-1}} h_t(u \mid z_t^{M,x_1}, \ldots, z_t^{M,x_d}) \\
&= \lambda_t \sum_{k=1}^{d} \sum_{x_k\in\{1,\ldots,M\}} \sum_{x_{-k}\in\{1,\ldots,M\}^{d-1}} \\
&\qquad \sum_{u\in\mathcal{U}} \mathbf{1}_{x_k=j\wedge u=k} \frac{1}{M^{d-1}} h_t(u \mid z_t^{M,x_1}, \ldots, z_t^{M,x_d}) \\
&= \lambda_t \sum_{k=1}^{d} \sum_{x_k\in\{1,\ldots,M\}} \sum_{x_{-k}\in\{1,\ldots,M\}^{d-1}} \sum_{u\in\mathcal{U}} \sum_{\bar{z}_k\in\mathcal{Z}} \\
&\qquad \sum_{\bar{z}_{-k}\in\mathcal{Z}^{d-1}} \mathbf{1}_{x_k=j\wedge u=k} \frac{1}{M^{d-1}} h_t(u \mid (\bar{z}_k, \bar{z}_{-k}))\mathbf{1}_{\bigwedge_{i=1}^{d} z_t^{M,x_i}=\bar{z}_i} \\
&= \lambda_t \sum_{k=1}^{d} \sum_{\bar{z}_k\in\mathcal{Z}} \sum_{\bar{z}_{-k}\in\mathcal{Z}^{d-1}} \sum_{u\in\mathcal{U}} \mathbf{1}_{\bar{z}_k=z_t^{M,j}\wedge u=k} \\
&\qquad \cdot \underbrace{\frac{\sum_{x_{-k}\in\{1,\ldots,M\}^{d-1}} \mathbf{1}_{\bigwedge_{i\neq k} z_t^{M,x_i}=\bar{z}_i}}{M^{d-1}}}_{\prod_{i\neq k} \mathbb{H}_t^M(\bar{z}_i)} h(u \mid (\bar{z}_k, \bar{z}_{-k})) \\
&= \lambda_t \sum_{k=1}^{d} \sum_{\bar{z}\in\mathcal{Z}^d} \sum_{u\in\mathcal{U}} \mathbf{1}_{\bar{z}_k=z_t^{M,j}\wedge u=k} \prod_{i\neq k} \mathbb{H}_t^M(\bar{z}_i) h_t(u \mid \bar{z}) \\
&= \lambda_t \sum_{\bar{z}\in\mathcal{Z}^d} \sum_{u\in\mathcal{U}} \mathbf{1}_{\bar{z}_u=z_t^{M,j}} \prod_{i\neq u} \mathbb{H}_t^M(\bar{z}_i) h_t(u \mid \bar{z}) \\
&= \frac{\lambda_t \int_{\mathcal{Z}^d\times\mathcal{U}} \mathbf{1}_{\bar{z}_u=z_t^{M,j}} (\mathbb{H}_t^M \otimes h_t)(\mathrm{d}\bar{z}, \mathrm{d}u)}{\mathbb{H}_t^M(z_t^{M,j})} = \lambda_t(\mathbb{H}_t^M, z_t^{M,j})
\end{aligned}$$

where the indices $-k$ denote all dimensions other than k.

Therefore, as long as $\mathbb{H}_t^M \xrightarrow{d} \nu_t$ (convergence in distribution), we find $\mathbb{E}\left[D_t - D_t^M\right] \to 0$ by the continuous mapping theorem. In particular, this holds true if $\mathbb{H}_t^M \xrightarrow{p} \nu_t$, i.e. for any $\delta > 0$ as $M \to \infty$,

$$\mathbb{P}\left(\left\|\mathbb{H}_t^M - \nu_t\right\| > \delta\right) \to 0.$$

We show this by induction: At $t = 0$ the statement holds by the law of large numbers. Now assume that the statement holds for t, then for $t + 1$ we first show that for any $\varepsilon, \delta > 0$ there exists $M', \delta' > 0$ such that for all $M > M'$ we have

$$\mathbb{P}\left(\left\|\mathbb{H}_{t+1}^M - \nu_{t+1}\right\| > \delta \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) < \varepsilon.$$

Note that

$$\begin{aligned}
&\mathbb{P}\left(\left\|\mathbb{H}_{t+1}^M - \nu_{t+1}\right\| > \delta \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \\
&\leq \sum_{z\in\mathcal{Z}} \mathbb{P}\left(\left|\mathbb{H}_{t+1}^M(z) - \nu_{t+1}(z)\right| > \delta \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \\
&\leq \sum_{z\in\mathcal{Z}} \mathbb{P}\left(\left|\mathbb{H}_{t+1}^M(z) - \mathbb{E}\left[\mathbb{H}_{t+1}^M(z) \;\middle|\; \mathbb{H}_t^M\right]\right| > \frac{\delta}{2} \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \\
&\quad + \sum_{z\in\mathcal{Z}} \mathbb{P}\left(\left|\mathbb{E}\left[\mathbb{H}_{t+1}^M(z) \;\middle|\; \mathbb{H}_t^M\right] - \nu_{t+1}(z)\right| > \frac{\delta}{2} \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right)
\end{aligned}$$

and we shall bound the former term as follows: Define

$$\Delta_{z_{t+1}^{M,j}|z_t^{M,j}} f \coloneqq f(z_{t+1}^{M,j}) - \mathbb{E}\left[f(z_{t+1}^{M,j}) \;\middle|\; f(z_t^{M,j})\right]$$

and let $f \colon \mathcal{Z} \to \mathbb{R}$, then we have

$$\begin{aligned}
&\mathbb{P}\left(\left|\mathbb{H}_{t+1}^M(f) - \mathbb{E}\left[\mathbb{H}_{t+1}^M(f) \;\middle|\; \mathbb{H}_t^M\right]\right| > \frac{\delta}{2} \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \\
&= \mathbb{P}\left(\left|\frac{1}{M}\sum_{j=1}^{M} \Delta_{z_{t+1}^{M,j}|z_t^{M,j}} f\right| > \frac{\delta}{2} \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \\
&\leq \frac{4}{\delta^2}\mathbb{E}\left[\left(\frac{1}{M}\sum_{j=1}^{M}\left(\Delta_{z_{t+1}^{M,j}|z_t^{M,j}} f\right)\right)^2 \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right] \\
&= \frac{4}{\delta^2 M^2}\sum_{j=1}^{M}\mathbb{E}\left[\left(\Delta_{z_{t+1}^{M,j}|z_t^{M,j}} f\right)^2 \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right] \\
&\leq \frac{16 \max_z f(z)^2}{\delta^2 M} \to 0
\end{aligned}$$

as $M \to \infty$ by conditional independence of $(z_{t+1}^{M,1}, \ldots, z_{t+1}^{M,M})$ given $z_t^M = (z_t^{M,1}, \ldots, z_t^{M,M})$, the Chebyshev inequality and tower property. In particular, this holds for $f_z \equiv \mathbf{1}_{\{z\}}$, $z \in \mathcal{Z}$. Therefore,

$$\sum_{z\in\mathcal{Z}} \mathbb{P}\left(\left|\mathbb{H}_{t+1}^M(z) - \mathbb{E}\left[\mathbb{H}_{t+1}^M(z) \;\middle|\; \mathbb{H}_t^M\right]\right| > \frac{\delta}{2} \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \to 0$$

as $M \to \infty$. For the latter term, note that analogously

$$\begin{aligned}
&\left|\mathbb{E}\left[\mathbb{H}_{t+1}^M(f) \;\middle|\; \mathbb{H}_t^M\right] - \nu_{t+1}(f)\right| \\
&\leq \Bigg|\sum_{z\in\mathcal{Z}} f(z) \sum_{z'\in\mathcal{Z}} \left(\mathbb{H}_t^M(z) - \nu_t(z)\right) \\
&\qquad \cdot \left(\exp\left(\bar{\mathbf{Q}}(\mathbb{H}_t^M, z')\Delta t\right) \cdot \begin{bmatrix}\mathbf{e}_{z'}\\0\end{bmatrix}\right)_z\Bigg| \\
&\quad + \Bigg|\sum_{z\in\mathcal{Z}} f(z) \sum_{z'\in\mathcal{Z}} \nu_t(z) \cdot \Bigg(\exp\left(\bar{\mathbf{Q}}(\mathbb{H}_t^M, z')\Delta t\right) \cdot \begin{bmatrix}\mathbf{e}_{z'}\\0\end{bmatrix} \\
&\qquad - \exp\left(\bar{\mathbf{Q}}(\nu_t, z')\Delta t\right) \cdot \begin{bmatrix}\mathbf{e}_{z'}\\0\end{bmatrix}\Bigg)_z\Bigg|
\end{aligned}$$

and by boundedness ($\lambda_t(\nu, z) \leq d\lambda_t$) and continuity in $\mathbb{H}_t^M, \nu_t$, for any $\varepsilon > 0$ there exists $\delta' > 0$ such that $\left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'$ implies $\left|\mathbb{E}\left[\mathbb{H}_{t+1}^M(f) \mid \mathbb{H}_t^M\right] - \nu_{t+1}(f)\right| < \varepsilon$. As a result, by the law of total probability

$$\begin{aligned}
&\mathbb{P}\left(\left\|\mathbb{H}_{t+1}^M - \nu_{t+1}\right\| > \delta\right) \\
&= \mathbb{P}\left(\left\|\mathbb{H}_{t+1}^M - \nu_{t+1}\right\| > \delta \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \cdot \mathbb{P}\left(\left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) \\
&\quad + \mathbb{P}\left(\left\|\mathbb{H}_{t+1}^M - \nu_{t+1}\right\| > \delta \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| > \delta'\right) \cdot \mathbb{P}\left(\left\|\mathbb{H}_t^M - \nu_t\right\| > \delta'\right) \\
&\leq \mathbb{P}\left(\left\|\mathbb{H}_{t+1}^M - \nu_{t+1}\right\| > \delta \;\middle|\; \left\|\mathbb{H}_t^M - \nu_t\right\| \leq \delta'\right) + \mathbb{P}\left(\left\|\mathbb{H}_t^M - \nu_t\right\| > \delta'\right)
\end{aligned}$$

$\to 0$

since we can choose M', δ' according to the former analysis and the induction assumption, completing the induction step. It then follows at all times t by the continuous mapping theorem that

$$\mathbb{E}\left[D_t - D_t^M\right] \to 0.$$

For the second term, fix M and let $N \to \infty$. We find that

$$\mathbb{E}\left[D_t^M\right] = \frac{1}{M}\sum_j \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{M,j}}\\ 0\end{bmatrix}\right)_{B+1}\right],$$

$$\mathbb{E}\left[D_t^{N,M}\right] = \frac{1}{M}\sum_j \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{N,M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{N,M,j}}\\ 0\end{bmatrix}\right)_{B+1}\right]$$

where $\bar{\mathbf{Q}}^{N,M,j}$ and $\bar{\mathbf{Q}}^{M,j}$ are continuous functions of

$$\lambda_t^{N,M,j} = \lambda_t \frac{M}{N}\sum_{i=1}^{N}\sum_{k=1}^{d} \mathbf{1}_{x_{t,k}^i = j}\mathbf{1}_{u_t^i = k},$$

$$\lambda_t^{M,j} = \lambda_t M \int_{\mathcal{X}\times\mathcal{U}} \sum_{k=1}^{d} \mathbf{1}_{x_k = j \wedge u = k}\, \mathbb{G}_t^M(\mathrm{d}x, \mathrm{d}u),$$

and as $N \to \infty$, by the conditional law of large numbers [24, Theorem 3.5]

$$\lambda_t^{N,M,j} \to \lambda_t^{M,j}$$

a.s. conditional on $z_t^{N,M,j} = z_t^{M,j} = z$ for any $z \in \mathcal{Z}$. Therefore, again by the continuous mapping theorem, for all $j = 1, \dots, M$ a.s.

$$\mathbb{E}\left[\exp(\bar{\mathbf{Q}}^{N,M,j}\Delta t) \;\middle|\; z_t^{N,M,j} = z\right] \to \mathbb{E}\left[\exp(\bar{\mathbf{Q}}^{M,j}\Delta t) \;\middle|\; z_t^{M,j} = z\right].$$

At the same time, $z_t^{N,M} \xrightarrow{d} z_t^M$ at all times t as $N \to \infty$ via induction: For $t = 0$ trivially $\mathcal{L}(z_t^{N,M}) = \nu_0 = \mathcal{L}(z_t^M)$. For $t+1$

$$\begin{aligned}
&\left|\mathbb{P}(z_{t+1}^{N,M} = z) - \mathbb{P}(z_{t+1}^{M} = z)\right| \\
&\leq \sum_{z'\in\mathcal{Z}} \left|\mathbb{P}(z_t^{N,M} = z') - \mathbb{P}(z_t^M = z')\right| \cdot \mathbb{P}(z_{t+1}^{N,M} = z \mid z_t^{N,M} = z') \\
&\quad + \sum_{z'\in\mathcal{Z}} \mathbb{P}(z_t^M = z') \\
&\qquad \cdot \left|\mathbb{P}(z_{t+1}^{N,M} = z \mid z_t^{N,M} = z') - \mathbb{P}(z_{t+1}^{M} = z \mid z_t^{M} = z')\right|
\end{aligned}$$

where the former tends to zero by induction assumption, while for the latter we have

$$\begin{aligned}
&\left|\mathbb{P}(z_{t+1}^{N,M} = z \mid z_t^{N,M} = z') - \mathbb{P}(z_{t+1}^{M} = z \mid z_t^{M} = z')\right| \\
&= \left|\prod_{j=1}^{M} \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{N,M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z'^j}\\ 0\end{bmatrix}\right)_{z^j} \;\middle|\; z_t^{N,M} = z'\right]\right. \\
&\qquad \left. - \prod_{j=1}^{M} \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z'^j}\\ 0\end{bmatrix}\right)_{z^j} \;\middle|\; z_t^{M} = z'\right]\right| \to 0
\end{aligned}$$

as $N \to \infty$ again as $\bar{\mathbf{Q}}^{N,M,j} \to \bar{\mathbf{Q}}^{M,j}$ conditionally a.s. for each j.

By Slutzky's theorem (on the conditional probability spaces given $z_t^{N,M,j} = z_t^{M,j} = z$), we have

$$\begin{aligned}
&\mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{N,M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{N,M,j}}\\ 0\end{bmatrix}\right)_{B+1} \;\middle|\; z_t^{N,M,j} = z\right] \\
&\quad \to \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{M,j}}\\ 0\end{bmatrix}\right)_{B+1} \;\middle|\; z_t^{M,j} = z\right]
\end{aligned}$$

for any $z \in \mathcal{Z}$, such that

$$\begin{aligned}
&\mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{N,M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{N,M,j}}\\ 0\end{bmatrix}\right)_{B+1}\right] \\
&= \sum_{z\in\mathcal{Z}} \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{N,M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{N,M,j}}\\ 0\end{bmatrix}\right)_{B+1} \;\middle|\; z_t^{N,M,j} = z\right] \\
&\qquad \cdot \mathbb{P}\left(z_t^{N,M,j} = z\right) \\
&\to \sum_{z\in\mathcal{Z}} \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{M,j}}\\ 0\end{bmatrix}\right)_{B+1} \;\middle|\; z_t^{M,j} = z\right] \cdot \mathbb{P}\left(z_t^{M,j} = z\right) \\
&= \mathbb{E}\left[\left(\exp(\bar{\mathbf{Q}}^{M,j}\Delta t)\cdot \begin{bmatrix}\mathbf{e}_{z_t^{M,j}}\\ 0\end{bmatrix}\right)_{B+1}\right]
\end{aligned}$$

which shows that $\mathbb{E}\left[D_t^{N,M}\right] \to \mathbb{E}\left[D_t^M\right]$ at all times t.

Now note that the terms $D_t, D_t^M, D_t^{N,M}$ are uniformly bounded by the maximum expected average number of lost packets by dropping all packets, given by the expectation of the Poisson-distributed number of arriving packets $\lambda_t \cdot \Delta t$. Therefore, for any $\varepsilon > 0$ we can choose T such that

$$\sum_{t=T}^{\infty} \gamma^t \left(\left|\mathbb{E}\left[D_t - D_t^M\right]\right| + \left|\mathbb{E}\left[D_t^M - D_t^{N,M}\right]\right|\right) < \frac{\varepsilon}{3}.$$

Consequently choose M sufficiently large such that

$$\left|\mathbb{E}\left[D_t - D_t^M\right]\right| < \frac{\varepsilon}{3T}, \quad \forall t \in \{0, 1, \dots, T-1\}$$

and similarly choose N sufficiently large to obtain

$$\left|\mathbb{E}\left[D_t^M - D_t^{N,M}\right]\right| < \frac{\varepsilon}{3T}, \quad \forall t \in \{0, 1, \dots, T-1\}$$

according to the prequel, such that $\left|J(\hat{\pi}) - J^{N,M}(\hat{\pi})\right| < \varepsilon$. □

Therefore, our mean-field model is well-motivated for sufficiently large systems, as we will also verify numerically.

4 EXPERIMENTS

In this section, we will begin by giving details on the experimental setup. Afterwards, we will demonstrate numerical results of applying reinforcement learning to the MFC MDP problem.

We have M homogeneous queues with exponential service rate α and N clients with Markov modulated arrival rate λ. Beginning with $\lambda_0 \sim \mathrm{Unif}(\{\lambda_h, \lambda_l\})$, at each decision epoch the arrival rate switches between high, λ_h, and low, λ_l, levels, using the transition law

$$\mathbb{P}(\lambda_{t+1} = \lambda_l \mid \lambda_t = \lambda_h) = 0.2, \tag{32}$$

$$\mathbb{P}(\lambda_{t+1} = \lambda_h \mid \lambda_t = \lambda_l) = 0.5. \tag{33}$$

Table 1: System parameters used in the experiments.

Symbol	Name	Value
Δt	Time step size	$1-10$
α	Service rate	1
(λ_h, λ_l)	Arrival rates	$(0.9, 0.6)$
N	Number of clients	$1000-1000000$
M	Number of queues	$100-1000$
d	Number of accessible queues	2
n	Monte Carlo simulations	100
B	Queue buffer size	5
ν_0	Queue starting state distribution	$[1, 0, 0, \ldots]$
D	Drop penalty per job	1
T	Training episode length	500
T_e	Evaluation episode length	$50-500$

Table 2: Hyperparameter configuration for PPO.

Symbol	Name	Value
γ	Discount factor	0.99
λ_{RL}	GAE lambda	1
β	KL coefficient	0.2
ϵ	Clip parameter	0.3
l_r	Learning rate	0.00005
B_b	Training batch size	4000
B_m	SGD Mini batch size	128
T_b	Number of epochs	30

In general, the experiments could be conducted with more levels of arrival rates and with different modulation rates estimated from a real system, though in our work we will use two arbitrarily chosen values to show the theoretical applicability of our methodology. The values for the system parameters in all of our experiments are given in Table 1.

In order to assess the performance of our MF policy, we compare it to JSQ(d) and the random policy, RND. In JSQ(d), at every decision epoch, d queues are selected out of M and jobs are allocated to the shortest one. In RND, we similarly select d queues randomly out of M and instead allocate the jobs to a random queue out of the d queues, which will be equivalent to a completely random selection out of M queues for sufficiently large $N \gg M$. In our work, we have used $d = 2$, since in [26] it has been shown that while moving from $d = 1$ to $d = 2$ shows an exponential increase in performance of JSQ(d), an additional increase to $d = 3$ does not add much in terms of achieved performance.

In order to obtain our MF policy by solving the optimal control problem, we apply proximal policy optimization (PPO) [33] using the RLlib implementation [21], a well-known and robust policy gradient reinforcement learning algorithm. The learning algorithm hyperparameters used in our experiments can be found in Table 2.

In Figure 3, we observe the learning curve of the applied reinforcement learning algorithm for $\Delta t = 5$ and find that the simple parameterization of the lower-level policies is indeed successful and leads to stable learning. For the demonstrated experiment, we trained in parallel (offline) on 20 cores of a commodity server CPU for approximately 35 hours, after which the optimal policy can be applied in practice, to finite systems. Here, MF-JSQ(2) and MF-RND refer to the corresponding JSQ and RND policies in the mean-field model, i.e. each applies a fixed h_t regardless of the current queue state distribution ν_t. In the case of MF-JSQ given by

$$h_t(u \mid \bar{z}) = \begin{cases} 0 & \text{if } u \notin \arg\min_{u'} \bar{z}_{u'} \\ \frac{1}{N_{\min}} & \text{else} \end{cases} \tag{34}$$

where $N_{\min}$ is the number of actions u that minimize the chosen queue's state $\bar{z}_u$. In the case of MF-RND, we similarly choose

$$h_t(u \mid \bar{z}) = \frac{1}{|\mathcal{U}|}, \quad \forall (\bar{z}, u) \in \mathcal{Z}^d \times \mathcal{U} \tag{35}$$

As expected, indicated by the horizontal lines, the JSQ(2) and random (RND) assignment policies in the mean-field case are both suboptimal for the chosen delay time of $\Delta t = 5$, and our reinforcement learning approach is capable of finding better load balancing policies after approximately 5 million simulated decision epochs. Though we have tried Dirichlet-parameterized upper-level policies to directly output simplex-valued actions in order to eliminate the need for manual normalization, we found that performance was significantly worse, hence motivating our approach.

Performance comparison on finite systems. We will now compare the performance of the evaluated load balancing algorithms on systems of finite size. For simulation of the finite-agent and finite-queue system, we simulate the continuous-time Markov processes exactly by sampling exponential waiting times for all events according to the Gillespie algorithm [12]. For an easy comparison between different Δt, we set the episode lengths T_e for evaluation to the integer nearest to $\frac{500}{\Delta t}$. Pseudocode for simulating and applying our MF policy in the finite system is given in Algorithm 1 [1].

[1]https://github.com/AnamTahir7/mfc_large_queueing_systems.git

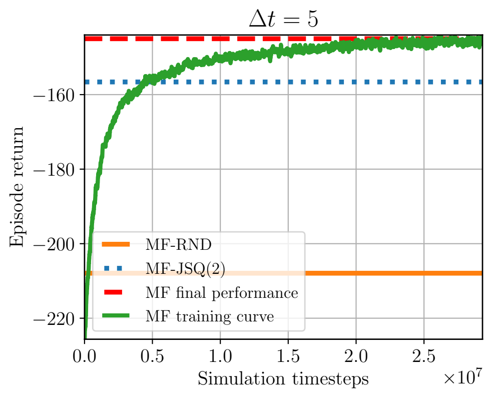

Figure 3: Training curve for the MF policy for $\Delta t = 5$ and $T_e = 500$ timesteps – i.e. the expected negative number of packet drops per episode during training – together with a comparison to the MF-JSQ(2) and MF-RND policies. The horizontal lines indicate the estimated expected returns for each policy. The red dotted line indicates the final achieved return of the learned MF policy in the mean-field MDP.

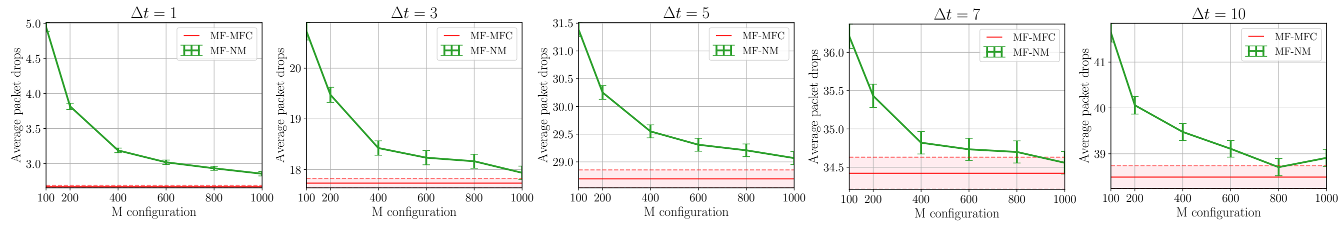

Figure 4: Comparison of the estimated expected packet drops (lower is better) of MF policies over the number of queues M in the finite system for different values of Δt, together with 95% confidence intervals depicted as shaded regions and error bars. Here, we use total running times of approximately 500 time units, and $N = M^2$ to fulfill $N \gg M$. The red dotted line indicates the equivalent achieved return of the learned MF policy in the mean-field control MDP, i.e. the limiting model as $N \gg M \to \infty$. It can be observed that as the system size $N = M^2$ increases, the performance under the MF policy (green) becomes increasingly close to the mean-field system performance (red), validating the accuracy of our mean-field formulation.

Algorithm 1 Application of MFC policy in finite system

```
1:  **Input**: System parameters from Table 1
2:  **Input**: Markovian upper-level policy $\tilde{\pi} = \{\tilde{\pi}_t\}_{t\geq 0}$
3:  Initialize $\lambda_0 \sim \mathrm{Unif}(\{\lambda_h, \lambda_l\})$.
4:  for $j = 1, \dots, M$ do
5:      Initialize queue states $z_0^j \sim \nu_0$.
6:  end for
7:  for $t = 0, 1, \dots, T_e$ do
8:      Compute empirical distribution $\mathbb{H}_t^M = \frac{1}{M}\sum_{j=1}^M \delta_{z_t^j}$.
9:      Sample decision rule $h_t \sim \tilde{\pi}_t(\mathbb{H}_t^M, \lambda_t)$.
10:     for $i = 1, \dots, N$ do
11:         Sample agent state $x_t^i \sim \otimes_{k=1}^d \mathrm{Unif}(\{1, \dots, M\})$.
12:         Compute anonymous state $\bar{z}_t^i = (z_t^{x_{t,1}^i}, \dots, z_t^{x_{t,d}^i})$.
13:         Sample agent action $u_t^i \sim h_t(\bar{z}_t^i)$.
14:     end for
15:     for $j = 1, \dots, M$ do
16:         Simulate continuous-time Markov chain $y^j$ with jump
            rates $\lambda_t^j, \alpha$ and $y^j(0) = z_t^j$ for $\Delta t$ time units.
17:         Count number of dropped packets.
18:         Set queue state $z_{t+1}^j = y^j(\Delta t)$.
19:     end for
20:     Sample $\lambda_{t+1} \sim \mathbb{P}(\lambda_{t+1} \mid \lambda_t)$.
21: end for
22: return Number of dropped packets.
```

In Figure 4, we show that the performance of the final learned MF policies over a wide range of delays Δt and system sizes (N, M). It can be seen that the overall achievable performance of our MF policy increases up to the performance achieved in the MFC MDP (red dotted line) as the system size (N, M) becomes sufficiently large ($N \gg M \gg 1$). Hence, our findings empirically validate the fact that our mean-field approximations are indeed accurate for sufficiently large system sizes.

The returns for the policies at each Δt, for the case where all experiments are run for approximately equal overall time instead of an equal number of decision epochs, are given in Figure 5. Here, we have trained a separate MF policy for each of the Δt and compared to JSQ(2) and RND. It can be seen that – as expected due to fewer updates – the overall achievable performance in the system worsens as the synchronization delay Δt of the system increases. It can be seen that MF achieves better performance than JSQ(2) starting from $\Delta t > 2$, while it outperforms RND in all cases. This stems from the fact that reinforcement learning only finds approximately optimal solutions. Nonetheless, at an intermediate level of synchronization delay beginning with $\Delta t = 3$, our learning-based methodology appears to be able to find a better policy than the optimal policies for $\Delta t \to 0$ (JSQ(2)) and $\Delta t \to \infty$ (RND). Even for small $\Delta t = 1$, our MF policy has comparable performance to the optimal JSQ(2) policy, as long as N, M are sufficiently large. As Δt keeps increasing, MF and RND are therefore expected to perform equally good in sufficiently large systems as long as we indeed have $N \gg M$.

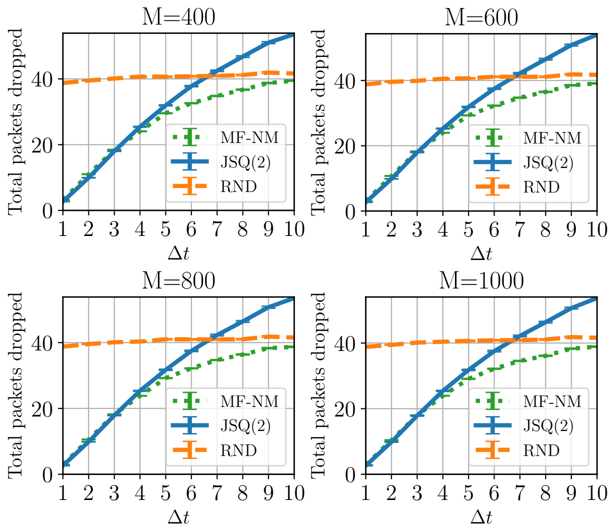

Figure 5: Comparison of the estimated expected packet drops of MF, JSQ(2), RND policies together with 95% confidence intervals for different configurations of M and $N = M^2$. We keep the total running time of each setting approximately equal to 500 time units to compare the effect of Δt. It can be observed that as Δt rises, the achievable performance by choosing emptier queues degrades.

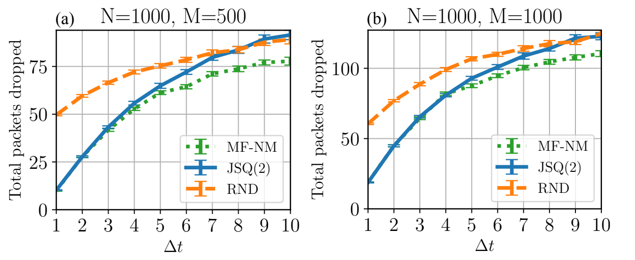

Figure 6: Comparison of the estimated expected packet drops of MF, JSQ(2), RND policies together with 95% confidence intervals for the same setting as in Figure 5, equal total running time, for the case when $M = 1000$, $N = \frac{M}{2}$ and t$N = M$. As Δt increases, the performance of our MF policy performs better than the other policies, even when $N \not\gg M$.

Finally, we perform experiments for $N \not\gg M$, i.e. we violate the formal approximation assumption used to obtain our mean-field system. Even though the assumptions made in our approximation are violated, our policy nonetheless obtains good comparative performance. As shown in Figure 6, we find that the qualitative performance differences remain the same for around 1000 agents and queues. It can also be observed that the random policy no longer obtains approximately equal performance over Δt, which is caused by the fact that the queues are increasingly sampled unequally often by an agent, and resampling resolves the resulting increased focus on a subset of queues.

5 DISCUSSION

In this work, we have proposed a mean-field-control-style formulation, with enlarged state-action space, for large-scale distributed queuing systems with synchronization delays. We have achieved this by formulating the finite-agent finite-queue system and considering $N \to \infty, M \to \infty$.

Firstly, we provide theoretical performance guarantees which show that the performance in the N, M system becomes arbitrarily close to the performance in the MFC system as long as N, M are large enough. Then, assuming a synchronous system with exact discretization of the underlying processes, we end up with an exactly discretized discrete-time Markov decision process on which we have applied reinforcement learning algorithms. As a result, we find that our learned solution can outperform the delay-free-optimal JSQ(d) policy as well as the infinite-delay-optimal random policy in the regime of intermediate delays Δt, even if $N \not\gg M$ as long as the system size N, M is sufficiently large.

An interesting future direction could be further extensions to the model such as non-exponential inter-arrival and service times, partial observability as well as explicitly modelling the case where N is not significantly larger than M. To allow for better scaling of the reinforcement learning algorithm to very large queue sizes, it may be of interest to apply further limiting, real-valued approximations of the queue states as $B \gg 1$. One straightforward extension would be to used heterogenous service rates. Finally, an implementation of the developed methods in a real world system may be of interest. We hope that our work inspires further work at the intersection of mean-field control theory and distributed queuing systems.

ACKNOWLEDGMENTS

This work has been co-funded by the German Research Foundation (DFG) as part of sub-project C3 within the Collaborative Research Center (CRC) 1053–MAKI and the LOEWE initiative (Hesse, Germany) within the emergenCITY center.

REFERENCES

[1] Vaneet Aggarwal. 2021. Machine Learning for Communications. *Entropy* 23, 7 (2021).
[2] Daniel Andersson and Boualem Djehiche. 2011. A maximum principle for SDEs of mean-field type. *Applied Mathematics & Optimization* 63, 3 (2011), 341–356.
[3] Jonatha Anselmi and Francois Dufour. 2020. Power-of-d-choices with memory: Fluid limit and optimality. *Mathematics of Operations Research* 45, 3 (2020), 862–888.
[4] Jalal Arabneydi and Aditya Mahajan. 2014. Team optimal control of coupled subsystems with mean-field sharing. In *53rd IEEE Conference on Decision and Control*. IEEE, 1669–1674.
[5] Alain Bensoussan, Jens Frehse, Phillip Yam, et al. 2013. *Mean field games and mean field type control theory*. Vol. 101. Springer.
[6] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.
[7] René Carmona, Mathieu Laurière, and Zongjun Tan. 2019. Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. *arXiv preprint arXiv:1910.12802* (2019).
[8] Kai Cui and Heinz Koeppl. 2021. Approximately solving mean field games via entropy-regularized deep reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1909–1917.
[9] Kai Cui, Anam Tahir, Mark Sinzger, and Heinz Koeppl. 2021. Discrete-Time Mean Field Control with Environment States. In *2021 60th IEEE Conference on Decision and Control (CDC)*. 5239–5246. https://doi.org/10.1109/CDC45484.2021.9683749
[10] Donald A Dawson, Jiashan Tang, and Yiqiang Q Zhao. 2005. Balancing queues by mean field interaction. *Queueing Systems* 49, 3 (2005), 335–361.
[11] Mao Fabrice Djete, Dylan Possamaï, and Xiaolu Tan. 2022. McKean–Vlasov optimal control: the dynamic programming principle. *The Annals of Probability* 50, 2 (2022), 791–833.
[12] Daniel T Gillespie. 1977. *The journal of physical chemistry* 81, 25 (1977), 2340–2361.
[13] Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. 2021. Mean-field controls with Q-learning for cooperative MARL: convergence and complexity analysis. *SIAM Journal on Mathematics of Data Science* 3, 4 (2021), 1168–1196.
[14] Xin Guo, Anran Hu, Renyuan Xu, and Junzi Zhang. 2019. Learning mean-field games. In *Advances in Neural Information Processing Systems*. 4966–4976.
[15] Onésimo Hernández-Lerma and Jean B Lasserre. 2012. *Discrete-time Markov control processes: basic optimality criteria*. Vol. 30. Springer Science & Business Media.
[16] Minyi Huang, Roland P Malhamé, Peter E Caines, et al. 2006. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems* 6, 3 (2006), 221–252.
[17] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.
[18] KR Krishnan. 1987. Joining the right queue: A Markov decision-rule. In *26th IEEE Conference on Decision and Control*, Vol. 26. IEEE, 1863–1868.
[19] Jean-Michel Lasry and Pierre-Louis Lions. 2007. Mean field games. *Japanese journal of mathematics* 2, 1 (2007), 229–260.
[20] Quan-Lin Li, Jing-Yu Ma, Rui-Na Fan, and Li Xia. 2019. An overview for Markov decision processes in queues and networks. In *International Conference of Celebrating Professor Jinhua Cao's 80th Birthday*. Springer, 44–71.
[21] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3053–3062.
[22] David Lipshutz. 2019. Open problem—load balancing using delayed information. *Stochastic Systems* 9, 3 (2019), 305–306.
[23] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. 2019. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3133–3174.
[24] Dariusz Majerek, Wioletta Nowak, and Wieslaw Zieba. 2005. Conditional strong law of large number. *Int. J. Pure Appl. Math* 20, 2 (2005), 143–156.

[25] Michael Mitzenmacher. 2000. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems* 11, 1 (2000), 6–20.

[26] Michael Mitzenmacher. 2001. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems* 12, 10 (2001), 1094–1104.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[28] Washim Uddin Mondal, Mridul Agarwal, Vaneet Aggarwal, and Satish V Ukkusuri. 2021. On the approximation of cooperative heterogeneous multi-agent reinforcement learning (marl) using mean field control (mfc). *arXiv preprint arXiv:2109.04024* (2021).

[29] Debankur Mukherjee, Sem C Borst, Johan SH Van Leeuwaarden, and Philip A Whiting. 2018. Universality of power-of-d load balancing in many-server systems. *Stochastic Systems* 8, 4 (2018), 265–292.

[30] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[31] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[32] Naci Saldi, Tamer Basar, and Maxim Raginsky. 2018. Markov–Nash Equilibria in Mean-Field Games with Discounted Cost. *SIAM Journal on Control and Optimization* 56, 6 (2018), 4256–4287.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[34] Jori Selen, Ivo Adan, Stella Kapodistria, and Johan van Leeuwaarden. 2016. Steady-state analysis of shortest expected delay routing. *Queueing Systems* 84, 3-4 (2016), 309–354.

[35] Shaler Stidham and Richard Weber. 1993. A survey of Markov decision models for control of networks of queues. *Queueing systems* 13, 1 (1993), 291–314.

[36] Jayakumar Subramanian and Aditya Mahajan. 2019. Reinforcement learning in stationary mean-field games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 251–259.

[37] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[38] Mark van der Boor, Sem Borst, and Johan van Leeuwaarden. 2019. Hyper-scalable JSQ with sparse feedback. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 1 (2019), 1–37.

[39] Mark van der Boor, Sem C Borst, Johan SH van Leeuwaarden, and Debankur Mukherjee. 2018. Scalable load balancing in networked systems: A survey of recent advances. *arXiv preprint arXiv:1806.05444* (2018).

[40] Ward Whitt. 1986. Deciding which queue to join: Some counterexamples. *Operations Research* 34, 1 (1986), 55–62.

[41] Wayne Winston. 1977. Optimality of the shortest line discipline. *Journal of applied probability* 14, 1 (1977), 181–189.

[42] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.

[43] Xingyu Zhou, Ness Shroff, and Adam Wierman. 2021. Asymptotically optimal load balancing in large-scale heterogeneous systems with multiple dispatchers. *Performance Evaluation* 145 (2021), 102146.