Stabilizing Queuing Networks with Model Data-Independent Control

Qian Xie and Li Jin *Member, IEEE*

***Abstract*— Classical queuing network control strategies typically rely on accurate knowledge of model data, i.e., arrival and service rates. However, such data are not always available and may be time-variant. To address this challenge, we consider a class of model data-independent (MDI) control policies that only rely on traffic state observation and network topology. Specifically, we focus on the MDI control policies that can stabilize multi-class Markovian queuing networks under centralized and decentralized policies. Control actions include routing, sequencing, and holding. By expanding the routes and constructing piecewise-linear test functions, we derive an easy-to-use criterion to check the stability of a multi-class network under a given MDI policy. For stabilizable multi-class networks, we show that a centralized, stabilizing MDI policy exists. For stabilizable single-class networks, we further show that a decentralized, stabilizing MDI policy exists. In addition, for both settings, we construct explicit policies that attain maximal throughput and present numerical examples to illustrate the results.**

***Index Terms*— Multi-class queuing networks, Dynamic routing, Lyapunov function, Stability**

I. Introduction

Control of multi-class queuing networks has been studied in numerous contexts of transportation, logistics, and communication systems [1]–[5]. Most existing analysis and design approaches rely on full knowledge of model data, i.e., arrival and service rates, to ensure stability and/or optimality [6]. However, in many practical settings, such data may be unavailable or hard to estimate, and may be varying over time. Such challenges motivate the idea of *model data-independent (MDI)* control policies. MDI control policies select control actions, including routing, sequencing, and/or holding, according to state observation and network topology but independent of arrival/service rates. Such policies are easy to implement and, if appropriately designed, can resist modeling error or non-stationary environment. However, the stability of general open multi-class queuing networks with centralized or decentralized MDI control policies has not been well studied.

This work is in part supported by US NSF Award CMMI-1949710, C2SMART University Transportation Center, NYU Tandon School of Engineering, SJTU UM Joint Institute, and J. Wu & J. Sun Endowment Fund.

Q. Xie is with the School of Operations Research and Information Engineering, Cornell University and with the Tandon School of Engineering, New York University, USA. L. Jin is with the UM Joint Institute and the Department of Automation, Shanghai Jiao Tong University, China and with the Tandon School of Engineering, New York University, USA (emails: qianxie@nyu.edu, li.jin@sjtu.edu.cn).

In this paper, we consider the stability of multi-class queuing networks with throughput-maximizing MDI control policies. Particularly, we focus on acyclic open queuing networks with Poisson arrivals and exponential service times. Jobs (customers) are classified according to their origin-destination (OD) information. Service rates are independent of job classes. A network is stabilizable if there exists a control policy that ensures positive Harris recurrence of the queuing process, whether the network is open-loop or closed-loop, centralized or decentralized [7]. By standard results on Jackson networks, stabilizability is equivalent to the existence of a (typically model data-dependent) stabilizing Bernoulli routing policy [8]. We assume that the class-specific arrival rates and the server-specific service rates are unknown to the controller. The main results are as follows:

1) An easy-to-use criterion to check the stability of a multi-class network under a given MDI control policy (Proposition 1).
2) For a multi-class network, a stabilizing centralized MDI control policy exists if and only if the network is stabilizable (Theorem 1).
3) For a single-class network, a stabilizing decentralized MDI control policy exists if and only if the network is stabilizable (Theorem 2).

Previous works on stability of queuing networks are typically based on full knowledge of model data [1], [9]–[14]. So far, the best-studied MDI control policy is the join-the-shortest-queue (JSQ) routing policy for parallel queues [15]–[22] or simple networks [23], which requires only the queue lengths and does not rely on model data [24]. When and only when the network is stabilizable, i.e., the demand is less than capacity (service rate), the JSQ policy guarantees the stability of parallel queues/simple networks [23], [25] and the optimality of homogeneous servers [16]. However, JSQ routing does not guarantee stability of more complex networks [24]. MDI routing for general networks has been numerically evaluated [26], but no structural results are available. Most studies on MDI routing for general networks are not aimed for stability [27]–[29]. In addition, decentralized dynamic routing has been considered for single origin-destination networks [30], [31] but not in MDI settings.

To design stabilizing MDI control policies, we first develop a stability criterion (Proposition 1) based on route expansion for queuing networks and explicit construction of a piecewise-linear test function. The expanded network is essentially a parallel connection of all routes from the set of origins to the

set of destinations. With this expansion, we use insights on the behavior of parallel queues and of tandem queues to construct the test function and derive the stability criterion. The test function can be used to obtain a smooth Lyapunov function verifying a negative drift condition. The piecewise-linear test function technique was proposed by Down and Meyn [7]; however, their implementation relies on linear programming formulations to determine parameters of the test function, which depends on model data. We extend this technique to the MDI setting using explicitly constructed test functions.

Based on the stability criterion, we design control policies in centralized and decentralized settings. First, for multi-class networks, we present a stabilizing centralized MDI control policy requiring dynamic routing and preemptive sequencing named JSR policy (Theorem 1). The control policy is obtained by minimizing the mean drift of the piecewise-linear test function, and the mean drift is guaranteed to be negative if and only if the network is stabilizable. The JSR policy, which is centralized and MDI, maximizes throughput among all control policies. Compared with other centralized policies, it does not require knowledge of model data, and compared with other MDI policies (e.g., JSQ), it guarantees stability for any stabilizable networks. Second, for single-class networks, we present a decentralized routing and holding policy that guarantees stability (Theorem 2). Such policies can also maximize the throughput since the stabilizability of the network implies that the throughput can be as large as close to the capacity. The results are closely related to the theory on the classical JSQ routing policy [24] and the decentralized max-pressure control policy [32].

The rest of this paper is organized as follows. Section II defines the multi-class queuing network model. Section III presents the stability criterion based on route expansion and piecewise-linear test function. Section IV and Section V consider the control design problem in centralized and decentralized settings respectively. Section VI gives concluding remarks.

II. Multi-class queuing network

Consider an acyclic network of queuing servers with infinite buffer spaces. Let $\mathcal{N}$ be the set of *servers*. Each server n has an exponential *service rate* $\bar{\mu}_n$. The network has a set $\mathcal{S}$ of *origins* and a set $\mathcal{T}$ of *destinations*. Jobs are classified according to their origins and destinations. That is, we can use an origin-destination (OD) pair $(S,T) \in \mathcal{C}$ to denote a *job class*, or simply *class*. For notational convenience, classes (OD pairs) are indexed by $c = (S_c, T_c)$. Jobs of class c arrive at S_c according to a Poisson process of rate $\lambda_c \geq 0$. We assume that service rates are independent of job class.

The *topology* of the network is characterized by *routes* between origins and destinations. We use $|r|$ to denote the number of servers on route r. Let $\mathcal{R}_c$ be the set of routes between S_c and T_c, and define $\mathcal{R} = \bigcup_{c\in\mathcal{C}} \mathcal{R}_c$. Below is an example network to illustrate the notations.

Example 1: Consider the Wheatstone bridge network in Fig. 1. Two classes of jobs arrive at S_1 (resp. S_2) with $\lambda_1 > 0$ (resp. $\lambda_2 > 0$). The set of servers is $\mathcal{N} = \{1, 2, \ldots, 5\}$ and

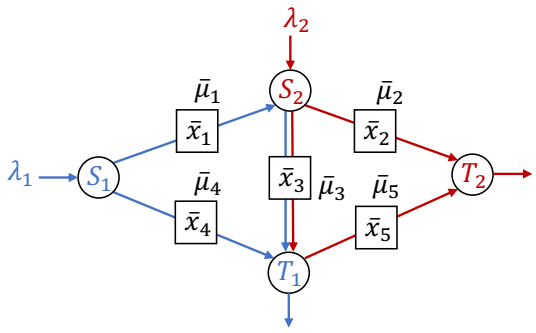


Fig. 1: A two-class queuing network.

the set of OD-specific routes are

$$\mathcal{R}_1 = \{(1,3),(4)\},\ \mathcal{R}_2 = \{(2),(3,5)\}.$$

The *state* of the network is defined as follows. Let $\bar{x} = [\bar{x}_n^c]_{n\in\mathcal{N},c\in\mathcal{C}}$ be the vector of class-specific *job numbers*, where $\bar{x}_n^c$ is the number of jobs of class c in server n, either waiting or being served. Let $\bar{\mathcal{X}}$ be the space of $\bar{x}$. We use $\bar{X}(t)$ to denote the state of the queuing process at time t.

We consider three types of *control actions*, viz. routing, sequencing, and holding. All control actions are essentially Markovian (in terms of $\bar{x}$ plus additional auxiliary states) and are applied at *transition epochs*, which include the arrival of a job at an origin or the completion of service at a server. *Routing* refers to allocating an incoming job to a server downstream to the origin or allocating a job discharged by a server to another downstream server. *Sequencing* refers to selecting a job from the waiting queue to serve. The default sequencing policy is the first-come-first-serve (FCFS) policy. For the multi-class setting, we consider the preemptive-priority that can terminate an ongoing service and start serving jobs from another class, while the job with incomplete service is sent back to the queue. *Holding* refers to holding a job that has completed its service in the server while blocking the other jobs in the queue from accessing the server.

Following [33], we say that a queuing network is *stable* if the queuing process is positive Harris recurrent; see [7], [9], [33] for details. Finally, we say that the network is *stabilizable* if a stabilizing control exists. One can check the stabilizability using the following result:

Lemma 1: An open acyclic queuing network is stabilizable if and only if there exists a vector $[\xi_r]_{r\in\mathcal{R}}$ such that

$$\begin{aligned} &\xi_r \geq 0, \quad \forall r \in \mathcal{R}, \\ &\lambda_c = \sum_{r\in\mathcal{R}_c} \xi_r, \quad \forall c \in \mathcal{C}, \\ &\sum_{r\in\mathcal{R}:n\in r} \xi_r < \bar{\mu}_n, \quad \forall n \in \mathcal{N}. \end{aligned}$$

The proof and implementation are straightforward.

III. Stability criterion

In this section, we derive a stability criterion for multi-class networks under given control policies. The techniques that we use include the route expansion of the original network and the explicit construction of a piecewise-linear test function based

on the network topology. In Section III-A, we construct an expanded network based on the original network. In Section III-B, we apply a piecewise-linear test function to the expanded network to obtain a stability criterion (Proposition 1) for both the expanded and the original networks.

A. Route expansion

For the convenience of constructing test function, we first introduce the route expansion. *Route expansion* refers to the construction of an *expanded network* based on the topology of *original network* (defined in Section II). The high-level idea is to decompose the network into routes, and the specific procedures are:

1) Place all routes $\mathcal{R}$ in the original network in parallel.
2) Add two-way connections between duplicates of servers in the original network.

For example, Fig. 2 shows the expanded network constructed from the original network in Fig. 1.

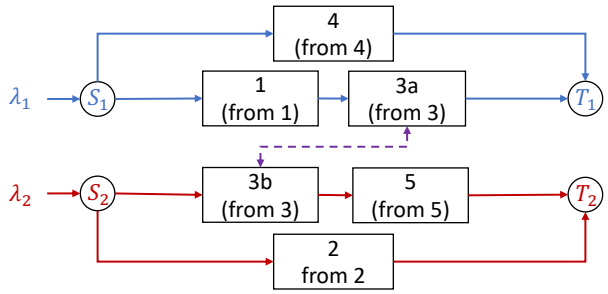


Fig. 2: Route expansion of the network in Fig. 1.

We call "servers" in the expanded network as *subservers*, since they are obtained by duplicating actual servers in the original network. Subservers are indexed by k, $c_k \in \mathcal{C}$ is the class index, $r_k \in \mathcal{R}$ is the route index, and $i_k \in \{1, 2, \ldots, |r_k|\}$ is the numbering of subserver k on route r_k. We use $k \in r$ to refer to that subserver k is on route r. Let $\mathcal{K}$ be the set of all subservers and $\mathcal{K}_c$ be the set of subservers with $c_k = c$. We use $n_k \in \mathcal{N}$ to denote the actual server that corresponds to subserver k. In addition, let k_p (resp. k_s) denote the subserver immediately upstream (resp. downstream) to subserver k.

The *state* of the expanded network is $x = \{x_k^c; k \in \mathcal{K}, c \in \mathcal{C}\}$, denoting the vector of number of class-c jobs in subserver k. Let $x_k := \sum_{c\in\mathcal{C}} x_k^c, k \in \mathcal{K}$. The expanded state space is $\mathcal{X} = \mathbb{Z}_{\geq 0}^{|\mathcal{C}|\times|\mathcal{K}|}$. Note that the states of the expanded network and the states of the original network are related by

$$\bar{x}_n^c = \sum_{k\in\mathcal{K}:n_k=n} x_k^c, \quad k \in \mathcal{K}, \tag{1}$$

for each $n \in \mathcal{N}$.

The routing policy is characterized by $\pi : \mathcal{X} \to [0,1]^{|\mathcal{C}|\times|\mathcal{K}|^2}$, where $\pi_{k,k'}^c$ is the probability that a class-c job is routed from subserver k to subserver k'.

The holding policy is characterized by $\zeta : \mathcal{X} \to \{0,1\}^{|\mathcal{K}|}$, where ζ_k specifies whether subserver k is holding ($\zeta_k(x) = 0$) or not holding ($\zeta_k(x) = 1$) when the current state is x.

Two subservers k and k' are *duplicating* if $n_k = n_{k'}$. Note that the service rates of duplicating subservers are coupled in the sense that for each server $n \in \mathcal{N}$, at a given time, at most one subserver k such that $n_k = n$ can be actively serving jobs, or *active*. This can be modeled as an *imaginary service rate control policy* $\mu : \mathcal{X} \to \mathbb{R}^{|\mathcal{K}|}$ such that the service rate $\mu_k(x)$ of subserver k satisfies

$$\sum_{k:n_k=n} \mu_k(x) \leq \bar{\mu}_n, \quad \forall x \in \mathcal{X}.$$

Such control policy is essentially equivalent to the class-based preemptive sequencing in the original network.

Note that $\{X(t) : t \geq 0\}$ is a Markov process, and the positive Harris recurrence refers to that there exists a unique invariant measure ν on $\mathcal{X}$ such that for every measurable set $D \subseteq \mathcal{X}$ with $\nu(D) > 0$ and for every initial condition $x \in \mathcal{X}$,

$$\Pr\{\tau_D < \infty | X(0) = x\} = 1,$$

where $\tau_D = \inf\{t \geq 0 : X(t) \in D\}$. Also, though $\{\bar{X}(t) : t \geq 0\}$ is not a Markov process, it will eventually converge to a steady state distribution.

The route expansion technique not only expands the network but also decomposes the state variables. Jobs can move along the expanded network using two transition mechanisms. One is *actual transition*, referring to moving a job from subserver k (or an origin) to its downstream subserver k_s (or a destination). The other is *imaginary transition* that moves a job from one subserver k to a duplicating subserver k' thereof, see imaginary switch in Section V. Imaginary transitions always occur instantaneously. Note that an actual transition corresponds to a transition in the original network, while an imaginary transition does not; this is also revealed in (1).

One can always map a control action in the expanded network to the original network. However, an MDI control policy may not exist on the state space of the original network; we do need an expanded state space for MDI control. In addition, we allow imaginary control actions in the expanded network, including *imaginary service rate control* and *imaginary switch*; see Section IV and Section V. Such imaginary actions only make sense in the expanded network and do not correspond to actual service rate control or switch in the original network.

B. Stability of the expanded network

After introducing the expanded network and the mathematical definition of the control policy, the main question of this paper can be expressed in a formal way as follows.

Given an expanded queuing network and a control policy $\phi = (\pi, \mu, \zeta)$, how can we tell if the control policy ϕ is stabilizing, i.e., the expanded network is stable under ϕ?

The answer of this question will be given in Proposition 1. Before that, we need to introduce the test function technique first. As opposed to linear programming-based construction in [7], we provide an explicit construction, where parameters of the test function do not rely on solving any optimization problems. The high level idea is to identify the bottlenecks and their upstream subservers. Our construction is based on the route expansion described in the previous subsection.

1) For each class $c \in \mathcal{C}$ and expanded state $x \in \mathcal{X}$, define

$$g_c(x) := \max_{\substack{K_c \subseteq \mathcal{K}_c: \\ \kappa \in K_c \Rightarrow \kappa_p \in \mathcal{K}_c}} \sum_{k \in K_c} a_k x_k,$$

where $a_k \in (0,1)$ is a parameter.

2) Define a piecewise-linear *test function*

$$V(x) := \max_{C \subseteq \mathcal{C}} \sum_{c \in C} b_c g_c(x),$$

where $b_c \in (0,1)$ is a parameter.

We call $V(x)$ the test function rather than the Lyapunov function, since strictly speaking, a smooth Lyapunov function should be developed based on the piecewise-linear test function to verify the Foster-Lyapunov stability criterion. Down and Meyn [7] showed that as long as a piecewise-linear test function can be determined, one can always smooth it to obtain a qualified C^2 Lyapunov function.

Remark 1: The test functions we proposed in this work are MDI. But generally speaking, they do not need to be MDI since it does not affect the control policies to be MDI.

Definition 1 (Dominance): Consider state $x \in \mathcal{X}$.

1) We call C^* a set of *dominant* classes if

$$C^* \in \operatorname*{argmax}_{C \subseteq \mathcal{C}} \sum_{c \in C} b_c g_c(x).$$

Each class $c \in C^*$ is a *dominant* class.

2) We call K_c^* a set of *dominant* class-c subservers if

$$K_c^* \in \operatorname*{argmax}_{\substack{K_c \subseteq \mathcal{K}_c: \\ k \in K_c \Rightarrow k_p \in \mathcal{K}_c}} \sum_{k \in K_c} a_k x_k.$$

Each subserver $k \in K_c^*$ is a *dominant* class-c subserver.

3) A route $r \in \mathcal{R}_c$ is *dominant* if it includes dominant class-c subservers, i.e. there exists dominant class-c subserver $k \in K_c^*$ such that $k \in r$.
Let R_c be the set of dominant class-c routes.

4) A subserver $b \in K_c^*$ is called a *bottleneck* if it is a dominant class-c subserver while its immediate downstream subserver $b_s \notin K_c^*$ is not.

Remark 2: A route or server is dominant if changes in its traffic state immediately affect the test function V.

A *regime* X of the piecewise-linear test function is a subset of $\mathcal{X}$ such that there exist $C^X \subseteq \mathcal{C}$, $K^X = \bigcup_{c \in C^X} K_c^X \subseteq \mathcal{K}$, and $R^X = \bigcup_{c \in C^X} R_c^X \subseteq \mathcal{R}$ where C^X, K_c^X, R_c^X are dominant for each $x \in X$, i.e., the test function is linear over X. Let $\mathscr{X}$ be the set of regimes; note that $\bigcup_{X \in \mathscr{X}} X = \mathcal{X}$.

Definition 2 (Mean velocity and drift): Consider a multi-class network with state $x \in \mathcal{X}$ under an expanded control policy $\phi = (\pi, \mu, \zeta)$.

1) The *mean velocity* at state x is a function $v : \mathcal{X} \to \mathbb{R}^{|\mathcal{K}|}$ such that for each $k \in \mathcal{K}$,

$$v_k(x) := \sum_{c \in \mathcal{C}} \lambda_c \pi^c_{S_c,k}(x) + \mu_{k_p}(x)\zeta_{k_p}(x) - \mu_k(x)\zeta_k(x).$$

where $\pi^c_{S_c,k}$ is the probability that a class-c job is routed from origin S_c to subserver k, while μ_k and ζ_k are the controlled service rate and the holding status of the subserver k respectively.

2) Given $X \in \mathscr{X}$ such that $x \in X$, the *mean drift over* X is given by

$$D^X(x) := \sum_{c \in C^X} b_c \sum_{k \in K_c^X} a_k v_k(x).$$

Remark 3: In our subsequent analysis, the mean drift $D^X(x)$ of the test function will play the role of infinitesimal generator applied to a Lyapunov function; see [7] for the connection between the test function and the Lyapunov function.

The main result of this section is as follows:

Proposition 1: Consider a multi-class network under the expanded control policy ϕ. Suppose there exist constants $M < \infty$, $\epsilon > 0$, and $a_k, b_c \in (0,1)$ ($\forall\, c \in \mathcal{C}$, $k \in K_c$), such that for each $X \in \mathscr{X}$ and each $x \in X$ where $|x| = \sum_{k \in \mathcal{K}} x_k > M$,

$$\sum_{c \in C^X} b_c \sum_{k \in K_c^X} a_k v_k(x) \le -\epsilon. \tag{2}$$

Then, the network is stable.

Proof. Consider the test function $V(x)$. By its definition, if $x \in X$, we have

$$V(x) = \sum_{c \in C^X} b_c \sum_{k \in K_c^X} a_k x_k.$$

The mean drift is given by

$$\begin{aligned} D^X(x) &= \sum_{c \in C^X} b_c \sum_{k \in K_c^X} a_k v_k(x) \\ &\overset{(2)}{\le} -\gamma^{|C^X|-1}\epsilon \le -\gamma^{|\mathcal{C}|-1}\epsilon, \quad x : |x| > M. \end{aligned}$$

One can then apply [7, Theorem 1] and [7, Lemma 5] to obtain the stability of the network. □

As a benchmark, the approach in [7, Theorem 1] requires solving linear programs to obtain parameters of the test functions in Proposition 1, while our approach explicitly constructs the parameters (see Section IV and Section V). Moreover, the proposed control, which is independent of model data, guarantees stability if and only if the network is stabilizable (see Theorem 1 and Theorem 2), while the approach in [7] relies on knowledge of model data.

IV. Centralized control for multiple classes

In this section, we consider the "join-the-shortest-route (JSR)" policy (a joint routing and sequencing policy) for centralized control. The JSR policy is MDI and constructed based on the expanded network. We will show that it is stabilizing if and only if the network is stabilizable.

The test function is constructed as follows.

1) For each class $c \in \mathcal{C}$, each route $r \in \mathcal{R}_c$, and each expanded state $x \in \mathcal{X}$, let

$$f_r(x) := \max_{k \in r} \alpha^{i_k - 1} \sum_{j : i_j \le i_k} x_j,$$

$$g_c(x) := \max_{R_c \subseteq \mathcal{R}_c} \beta^{|R_c|-1} \sum_{r \in R_c} f_r(x),$$

where $\alpha \in (0,1), \beta \in (0,1)$ are constant parameters.

2) The piecewise-linear *test function* is given by

$$V(x) := \max_{C \subseteq \mathcal{C}} \gamma^{|C|-1} \sum_{c \in C} g_c(x),$$

where $\gamma \in (0,1)$ is a constant parameter.

Let the parameters be such that

$$\alpha = \beta \geq \frac{|\mathcal{R}|-1}{|\mathcal{R}|}, \quad \gamma \geq \frac{|\mathcal{C}|-1}{|\mathcal{C}|}, \tag{3}$$

and follow the notions of dominance accordingly (see Definition 1). Note that such MDI parameters α, β, γ always exist. The control that we consider in this subsection only depends on α, β, γ and is thus MDI. Specifically, we define the JSR policy as follows:

Definition 3 (Join-the-shortest-route (JSR) policy):

1) (Routing) At an origin S, an incoming job of class c is allocated to the route $r^* \in \mathcal{R}_c$ such that

$$r^* \in \operatorname*{argmin}_{r \in \mathcal{R}_c} f_r(x).$$

If there is only one minimum, then r^* must be a non-dominant route. Otherwise, let i_{r^*} be the index of the bottleneck on route r^*. Then, an incoming job of class c is allocated to the route $r^* \in \mathcal{R}_c$ with the largest i_{r^*}. For each class c, the index of the bottleneck on the allocated route is denoted as

$$i_c = \max_{r^* \in \mathcal{R}_c} i_{r^*}.$$

2) (Imaginary service rate control) Let $\mathcal{K}_n$ be the set of subservers corresponding to server n and let $\mathcal{B}$ be the set of bottlenecks for a given x. Then, a subserver $k \in \mathcal{K}_n$ is activated if $k \in \mathcal{B}$. If multiple subservers are in $\mathcal{K}_n \cap \mathcal{B}$, then activate the subserver k^* such that

$$k^* \in \operatorname*{argmin}_{k \in \mathcal{K}_n \cap \mathcal{B}} \{i_{c_k} + |\mathcal{R}_{c_k}|\}.$$

This is to ensure that the bottlenecks are active to discharge jobs and only one of the duplicating subservers can be active.

The main result of this section is the following:

Theorem 1 (Stability of JSR policy): The JSR policy stabilizes a multi-class network if and only if the network is stabilizable.

This theorem implies that the JSR policy is also throughput-maximizing, as long as the network is stabilizable, i.e., the demand is less than the total capacity. Note that the stabilizability can be easily checked using Lemma 1.

In the rest of this section, we apply Theorem 1 to study the stability of the Wheatstone bridge network under the JSR policy (Subsection IV-A) and then prove this theorem (Subsection IV-B).

A. Numerical Example

Consider the network in Fig. 1 and suppose that $\lambda_1 = \lambda_2 = \lambda = 1$ and $\bar{\mu}_n = \mu = 1$ for $n = 1, 2, 4, 5$ and $\bar{\mu}_3 = \frac{1}{4}$. This example is for illustrating the route expansion and the test function construction.

Note that under the above model parameters, the decentralized JSQ policy is destabilizing. To see this, $\bar{\mu}_1 = \bar{\mu}_4$ implies that on average, class-1 jobs are evenly distributed between server 1 and server 4. Thus, the average departure rate of class-1 jobs from server 1 is $\frac{1}{2}$, which exceeds the service rate of server 3. Therefore, the queue at server 3 is unstable. The main reason that the JSQ policy is destabilizing is the ignorance of downstream congestion. As $\bar{X}_3(t)$ gets large, a reasonable action is to allocate fewer class-1 jobs to server 1. However, the JSQ policy disallows such far-sighted decisions.

An alternative centralized stabilizing routing policy can be the following JSR policy:

1) A class-1 job arriving at S_1 is routed to server 1 if $\bar{X}_1^1(t) + \bar{X}_3^1(t) < \bar{X}_4^1(t)$, to server 4 if $\bar{X}_1^1(t) + \bar{X}_3^1(t) > \bar{X}_4^1(t)$, and randomly otherwise.
2) A class-2 job arriving at S_2 is routed to server 3 if $\bar{X}_3^2(t) + \bar{X}_5^2(t) < \bar{X}_2^2(t)$, to server 2 if $\bar{X}_3^2(t) + \bar{X}_5^2(t) > \bar{X}_2^2(t)$, and randomly otherwise.
3) The dominant class has a higher priority.

That is, when jobs are routed at S_1, the decision is based on not only the local state ($\bar{X}_1(t)$ and $\bar{X}_4(t)$), but also the state further downstream ($\bar{X}_3(t)$).

The expanded network is shown in Fig. 2. Each block in the figure represents a subserver. In particular, subservers 3a and 3b are decomposed from server 3; the other servers are remained. Solid arrows correspond to actual transitions in an original network, while dashed arrows correspond to imaginary transitions between duplicating subservers.

In the expanded network, a job can move along both solid and dashed arrows. The color of an arrow shows which class can move along it: blue means class (S_1, T_1), red means (S_2, T_2), and purple means both. For ease of presentation, we label (S_1, T_1) as class 1 and (S_2, T_2) as class 2. For example, a job of class (S_1, T_1) can visit subservers 4, 1, 3a, 3b and the destination T_1.

In the expanded network, the JSR policy works as follows.

1) A class-1 job arriving at S_1 is routed to subserver 4 if $X_4(t) < X_1(t) + X_{3a}(t)$, to subserver 1 if $X_4(t) > X_1(t) + X_{3a}(t)$, and randomly otherwise.
2) A class-2 job arriving at S_2 is routed to subserver 3b if $X_{3b}(t) + X_5(t) < X_2(t)$, to subserver 2 if $X_{3b}(t) + X_5(t) > X_2(t)$, and randomly otherwise.
3) If subserver 3a is dominant while subserver 3b is non-dominant, and server 3 is serving a class-2 job, then server 3 preempts the class-2 job being served in 3b to the class-1 job in 3a, and vice versa. If both subserver 3a and subserver 3b are dominant, then server 3 gives priority to class-2 job since the index of 3b is smaller.

By Theorem 1, the network can be stabilized by the JSR policy if and only if

$$\lambda_1 < 2, \ \lambda_2 < 2, \ \lambda_1 + \lambda_2 < \frac{9}{4}.$$

We use the following parameters for the test function:

$$\alpha = \beta = \gamma = \frac{3}{4}, \ \epsilon = \left(\frac{3}{4}\right)^5.$$

One can verify that the above parameters satisfy (3) and Proposition 1 by considering the following cases:

1) Only one route is dominant. In this case, an incoming job is always allocated to a non-dominant route, leading to non-positive contribution to the mean drift:
$$D^X(x) \le -\gamma\beta\alpha\mu = -\left(\frac{3}{4}\right)^3 \le -\epsilon.$$
2) Two routes with different OD pairs are dominant. This case is analogous to the previous case:
$$D^X(x) \le -\gamma\beta\alpha\mu = -\left(\frac{3}{4}\right)^3 \le -\epsilon.$$
3) Two routes with the same OD pair or more than two routes are dominant. In such cases, the mean drift satisfies
$$D^X(x) \le \gamma\beta^3(\lambda - \mu - \alpha\mu) = -\left(\frac{3}{4}\right)^5 \le -\epsilon.$$

Consequently, the network is stable under the MDI JSR policy.

B. Proof of Theorem 1

In this subsection, we will the sufficiency and the necessity respectively, based on the connection between the sign of the mean drift and the stabilizability of the network. When analyzing the mean drift, we consider two parts: external arrivals and internal transmission. We first show that any internal transmission does not positively contribute to the mean drift and then show that any positive contribution from external arrivals can always be compensated by internal transmissions.

1) Internal transmissions: Note that under the JSR policy, every job remains on the route assigned to the job when it enters the network. Hence, internal transmissions only occur between subservers on the same route.

Given x, consider an internal transmission from subserver k to subserver j; this implicitly requires $x_k \ge 1$. The definition of dominance ensures that if j is dominant, then so is k. Hence, we need to consider the following cases:

1) If k and j are both dominant, the transmission leads to zero contribution to the mean drift $D^X(x)$ for all X such that $x \in X$.
2) If k is dominant and j is non-dominant, the transmission leads to the following contribution to the mean drift:
$$-\alpha^{i_k-1}\mu_k(x) \le 0.$$

Hence, internal transmissions never lead to positive contribution to the mean drift.

2) External arrivals: Given $x \ne 0$, consider a regime $X \in \mathscr{X}$ such that $x \in X$. For each $c \in \mathcal{C}$, the JSR policy ensures that if there exists a non-dominant route in $\mathcal{R}_c$, then an incoming job must be allocated to a non-dominant route in $\mathcal{R}_c$, leading to non-positive contribution to the mean drift. Hence, we only need to consider dominant classes c such that every route in $\mathcal{R}_c$ is dominant, i.e., $R_c^X = \mathcal{R}_c$. Recall that $C^* \subseteq \mathcal{C}$ is the set of dominant classes. The part of the mean drift associated with $c \in C^*$ satisfies
$$D_c^X(x) \le \gamma^{|C^*|-1}\beta^{|\mathcal{R}_c|-1}\Big(\alpha^{i_c-1}\lambda_c - \sum_{b\in\mathcal{B}^X:c_b=c}\alpha^{i_b-1}\mu_b(x)\Big)$$
$$:= \gamma^{|C^*|-1}\Delta_c^X(x)$$
over any regimes of the piecewise-linear test function, where i_c is given in Definition 3.

Lemma 2: When $x \ne 0$, there is no empty bottleneck, i.e.,
$$x_b \ge 1. \tag{4}$$

Proof. Since $x \ne 0$ and r_b is dominant, we have
$$\sum_{k:i_k\le i_b} x_k > 0.$$
If $i_b = 1$, then the above inequality directly implies (4).

Now consider the case that $i_b \ge 2$. Since b is a bottleneck, we have
$$\alpha^{i_b-1}\sum_{k\in r_b:i_k\le i_b} x_k \ge \alpha^{i_b-2}\sum_{k\in r_b:i_k\le i_b-1} x_k,$$
which implies
$$x_b \ge (1-\alpha)\sum_{k:i_k\le i_b} x_k > 0$$
and thus we have (4). □

Lemma 2 is to ensure that the bottlenecks are none-empty to discharge jobs and thus contribute negative terms to the drift.

Next, we show the sufficiency of Theorem 1. Based on the definition of the routing policy (see Definition 3), $\forall b \in \mathcal{B}^X$, we have $i_b \le i_c$ when the incoming job is allocated to a dominant route. Then
$$\begin{aligned}\sum_{c\in C^*}\Delta_c^X(x) &\le \sum_{c\in C^*}\beta^{|\mathcal{R}_c|-1}\alpha^{i_c-1}\Big(\lambda_c - \sum_{b\in\mathcal{B}^X:c_b=c}\mu_b(x)\Big)\\ &= \sum_{c\in C^*}\alpha^{i_c+|\mathcal{R}_c|-2}\Big(\lambda_c - \sum_{b\in\mathcal{B}^X:c_b=c}\mu_b(x)\Big)\\ &:= \sum_{c\in C^*}\alpha_c\beta_c\end{aligned}$$
Without loss of generality assume $C^* = \{1, 2, \cdots, m\}$ and
$$i_1 + |\mathcal{R}_1| \le i_2 + |\mathcal{R}_2| \le \cdots \le i_m + |\mathcal{R}_m|.$$

Then by using Abel transformation (summation by parts), the right hand side of the above inequality (abbr. RHS):
$$\text{RHS} = \sum_{i=1}^{m-1}(\alpha_i - \alpha_{i+1})\sum_{j=1}^{i}\beta_j + \alpha_m\sum_{j=1}^{m}\beta_m.$$
Based on the assumption, we have $\alpha_i \ge \alpha_{i+1}$ and
$$\begin{aligned}\sum_{j=1}^{i}\beta_j &= \sum_{j=1}^{i}\Big(\lambda_j - \sum_{b\in\mathcal{B}^X:c_b=j}\mu_b(x)\Big)\\ &= \sum_{j=1}^{i}\lambda_j - \sum_{n_b:b\in\mathcal{B}_i^X}\bar{\mu}_{n_b}\\ &= \sum_{j=1}^{i}\lambda_j - \sum_{n\in\mathcal{N}_i}\bar{\mu}_n\\ &< 0,\end{aligned}$$
where $\mathcal{B}_i$ is the set of bottlenecks in the first i classes and $\mathcal{N}_i$ is the min-cut of the original network with the first i classes. Here we use the definition of the imaginary service rate control (see Definition 3) and Lemma 1.

Since RHS < 0, we have $\sum_{c\in C^*} \Delta_c^X(x) < 0$ and thus $D_c^X(x) < 0$. Then by noting that internal transmissions lead to non-positive contributions to the mean drift, we have

$$D^X(x) \le \sum_{c\in C^*} D_c^X(x) < 0,$$

which implies stability. □

Finally, the necessity is apparent: if a network is not stabilizable, then there exists no MDI control that can stabilize the network.

V. Decentralized Control for a Single Class

For a single-class network, we can drop the class index and use x_k to denote the number of jobs in subserver k. Note that such network has a single origin and a single destination. Again we can do route expansion on such network.

We consider a decentralized MDI control policy as follows.

Definition 4 (JSQ with artificial spillback): The JSQ with artificial spillback (JSQ-AS) policy is as follows:

1) (Routing) A discharged job is routed to the shortest downstream queue, with ties randomly broken.
2) (Holding) For each subserver k, any job which has finished the service will be held if and only if $X_{s_k}(t) \ge X_k(t)$.
3) (Imaginary switch) When a dominant subserver k is inactive while its non-dominant duplicate k' is active, and both are not in the holding status, then the job in k' is moved to (and discharged from) k after service, and then routed to the downstream of k (i.e., s_k).

Note that under the holding policy, the process $\{X(t); t \ge 0\}$ admits an invariant set $\mathcal{Q} \subseteq \mathcal{X}$ given by

$$\mathcal{Q} := \{x \in \mathcal{X} : x_{s_k} \le x_k, k \in \mathcal{K}\}. \tag{5}$$

Since we consider the long-time stability of the network, it suffices to consider the states in an invariant set. The above result indicates that in the invariant set $\mathcal{Q}$, the queue size of any subserver is upper-bounded by the queue size of its immediate upstream subserver.

The JSQ-AS policy is decentralized in the sense that control actions on subserver k only depend on local traffic information: the number of jobs in duplicate subservers $\{x_{k'} : n_k = n_{k'}\}$ and that in immediate downstream subservers $\{x_{s_{k'}} : n_k = n_{k'}\}$. A key characteristic of such policies is that congestion information can propagate through the network via the forced holding: if a subserver becomes congested (i.e., x_k gets large), the congestion will propagate to the upstream subservers in a cascading manner ("artificial spillback"). Importantly, such artificial spillback does not undermine throughput like the natural spillback caused by the limited buffer size. The reason is that though congestion can propagate, the queue size in any downstream subserver is not upper-bounded. Artificial spillback is the main difference between the JSQ-AS policy and the classic JSQ policies.

Note that though the JSQ-AS policy is constructed based on the expanded network, its actions can always be converted to the ones in the original network. Importantly, the decentralized control in the expanded network must also be decentralized in the original network. Also note that the imaginary switch has no impact on the original network or the test function.

The main result of this section is as follows:

Theorem 2 (Stability of JSQ-AS policy): For the route expansion of a single-class network, the JSQ-AS policy is stabilizing if and only if

$$\lambda < \bar{\mu}^{mc}, \tag{6}$$

where $\bar{\mu}^{mc}$ is the min-cut service rate of the original network.

This theorem implies that JSQ-AS policy is also a throughput-maximizing policy since we allow any throughput that satisfies (6).

In the rest of this section, we apply Theorem 2 to study the stability of the Wheatstone bridge network under the JSQ-AS policy (Subsection V-A) and then prove this theorem (Subsection V-B).

A. Numerical Example

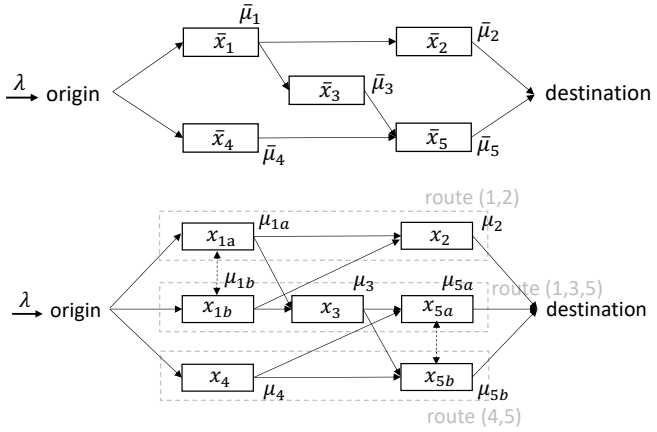


Fig. 3: A single-class queuing network and its expanded network.

Consider the original network with route expansion in Fig. 3. Again suppose that $\lambda = 1$, $\bar{\mu}_n = \frac{3}{4}$ for $n = 1, 2, \cdots, 5$. Similarly, with the above parameters, the JSQ policy is destabilizing since the queue at server 5 is unstable. However, in the decentralized setting, the control actions can only depend on the local state, say the routing decision at the origin can be based on $\bar{X}_1(t)$ and $\bar{X}_4(t)$, but not $\bar{X}_5(t)$. A remedy is to introduce the holding policy (artificial spillback) to the JSQ policy so that the downstream congestion can be relieved and the local state can somehow reflect the states further downstream.

In the expanded network, server 1 is decomposed into subserver 1a and 1b, server 5 is decomposed into subserver 5a and 5b. The states in the original network and those in the expanded network satisfy $\bar{X}_1(t) = X_{1a}(t) + X_{1b}(t)$ and $\bar{X}_5(t) = X_{5a}(t) + X_{5b}(t)$. The initial states of the expanded network can be not unique. Say the initial queue size of server 1 is 2, then the initial queue sizes of subserver 1a and 1b can be 2, 0 or 1, 1 or 0, 2 respectively. Then the states are updated based on the model and our JSQ-AS policy. For example, the routing decision at the origin is based on $X_{1a}(t)$, $X_{1b}(t)$

and $X_4(t)$ rather than $\bar{X}_1(t)$ and $\bar{X}_4(t)$; a job which has just finished the service at server 3 will be held if $X_{5a}(t) \geq X_3(t)$, once released, it will be routed to the shorter downstream queue by comparing $X_{5a}(t)$ and $X_{5b}(t)$.

B. Proof of Theorem 2

This proof uses the connection between the stabilizability condition (6) and the sign of the mean drift. But before showing the mean drift is negative, we first present the explicit MDI piecewise-linear test function and several key lemmas that can help us analyze the mean drift.

The piecewise-linear test function is constructed as follows:

$$V(x) := \max_{\substack{K\subseteq\mathcal{K}:\\ \kappa\in K\Rightarrow p_\kappa\in\mathcal{K}}} \left\{ \frac{1+(|K|-1)\delta}{|K|} \sum_{k\in K} x_k \right\},$$

where δ can be any small value such that $0 < \delta < 1$.

The following lemmas are useful in proving Theorem 2 where we consider the regime $X \subseteq \mathcal{Q}$ containing x.

Lemma 3: A bottleneck can not be in the holding status.

Proof. Otherwise, the bottleneck must have at least one downstream subserver. By (5), $x_{s_k} \geq x_b$. Since b is a bottleneck, we have

$$\frac{1+(|K^X|-2)\delta}{|K^X|-1} \sum_{k\neq b} x_k \leq \frac{1+(|K^X|-1)\delta}{|K^X|} \sum_{k\in K^X} x_k, \quad (7)$$

which implies

$$(1-\delta) \sum_{k\in K^X} x_k \leq |K^X|[1+(|K^X|-2)\delta]x_b. \quad (8)$$

Since $x_b \leq x_{s_k}$, we have

$$(1-\delta) \sum_{k\in K^X} x_k < |K^X|(1+|K^X|\delta)x_{s_k},$$

which is equivalent to

$$\frac{1+(|K^X|-1)\delta}{|K^X|} \sum_{k\in K^X} x_k < \frac{1+|K^X|\delta}{|K^X|+1} \Big(\sum_{k\in K^X} x_k + x_{s_k} \Big),$$

contradicting with the fact that subserver b is dominant and subserver s_k is non-dominant. □

Corollary 1: Based on (8), we have $x_b > 0$, i.e., any bottleneck b must be non-empty.

This corollary and Lemma 3 ensure that all bottlenecks can discharge customers and contribute negative terms to the drift.

Lemma 4: Let k_r^1 be the first subserver on route r, then either the route with the smallest $x_{k_r^1}$ is non-dominant or every route is dominant.

Proof. If there is only one route, then that route must be dominant. Now assume there are at least two routes and route $\hat{r}$ has the smallest $x_{k_r^1}$, i.e., $\forall\ r \in \mathcal{R}$, $x_{k_{\hat{r}}^1} \leq x_{k_r^1}$. Suppose $k_{\hat{r}}^1 \in K^X$ and $\exists\ r \in \mathcal{R}$ s.t. $k_r^1 \notin K^X$. Note that by (5), $x_b \leq x_{k_{\hat{r}}^1} \leq x_{k_r^1}$, then from (7) we have

$$\frac{1+(|K^X|-1)\delta}{|K^X|} \sum_{k\in K^X} x_k < \frac{1+|K^X|\delta}{|K^X|+1} \Big(\sum_{k\in K^X} x_k + x_{k_r^1} \Big),$$

contradicting with our supposition. Therefore, either $\hat{r}$ is non-dominant or every route in $\mathcal{R}$ is dominant. □

Lemma 5: If $x \in \mathcal{X}$ makes every route $r \in \mathcal{R}$ dominant, then we have

$$\sum_{k\in\mathcal{B}^X} \mu_k(x) = \sum_{n:n=n_k,k\in\mathcal{B}^X} \bar{\mu}_n$$

Proof. Once there is an inactive bottleneck k and a non-dominant but active duplicate subserver k', the imaginary switch mechanism will move the job being served in k' to k and move one job in k to k'. This is allowed since both k and k' contain at least one job due to the fact that a job being served in k' and the bottleneck k must be non-empty. □

Similar to the proof of Theorem 1, we first analyze the internal transmissions and then the external arrivals.

1) Internal transmissions: In the proof of Theorem 1, we have already discussed the case where internal transmissions between subservers are on the same route. However, unlike the JSR policy, the JSQ-AS policy allows internal transmissions between subservers on different routes. Hence, we also need to consider the internal transmission from subserver k to subserver j where $r_k \neq r_j$.

The definition of dominance ensures that if k is non-dominant, so is s_k. According to the routing policy, $x_{s_k} \geq x_j$. Let ℓ be the first non-dominant subserver on route r_k and b be the bottleneck on route r_j. If j is non-dominant, then by (5), we have $x_\ell \geq x_{s_k} \geq x_j \geq x_b$. Now from (7) we can obtain

$$\frac{1+(|K^X|-1)\delta}{|K^X|} \sum_{k\in K^X} x_k < \frac{1+|K^X|\delta}{|K^X|+1} \Big(\sum_{k\in K^X} x_k + x_\ell \Big),$$

contradicting with the definition of dominant subservers.

Thus, it cannot be the case that k is non-dominant and j is dominant, which implies that any internal transmission does not positively contribute to the mean drift.

2) External arrivals: According to Lemma 4, if a non-dominant route exists, then the routing policy guarantees that an arriving job must be routed to the first subserver on a non-dominant route r; this leads to non-positive contribution to the mean drift. Otherwise, every route is dominant. Then for any $x \in \mathcal{Q}$ $(x \neq 0)$, the drift satisfies

$$\begin{aligned} D^X(x) &\overset{\text{Corollary 1}}{\leq} \frac{1+(|K^X|-1)\delta}{|K^X|} \Big(\lambda - \sum_{b\in\mathcal{B}^X} \mu_b(x)\zeta_b(x) \Big) \\ &\overset{(5)}{=} \frac{1+(|K^X|-1)\delta}{|K^X|} \Big(\lambda - \sum_{b\in\mathcal{B}^X} \mu_b(x) \Big) \\ &\overset{\text{Lemma 5}}{=} \frac{1+(|K^X|-1)\delta}{|K^X|} \Big(\lambda - \sum_{n:n=n_k,k\in\mathcal{B}^X} \bar{\mu}_n \Big) \\ &\overset{\text{Lemma 1}}{<} 0, \end{aligned}$$

which completes the proof. □

The JSQ-AS policy cannot be directly applied to multi-class network, because the imaginary switch mechanism may move a job to the subserver of a different class with a different destination. Although the imaginary service rate control in the JSR policy can be used for multiple classes, it needs global information such as the information of dominance and bottlenecks for the preemption, so it is not suitable for the decentralized setting. The design of a decentralized MDI control policy for multi-class network can be a future work.

VI. Concluding remarks

We study the stability of open queuing networks under a class of model data-independent control policies. In addition, we derive an easy-to-use stability criterion based on route expansion of the network and explicit piecewise-linear test functions. With the stability criterion, we generalize the classical join-the-shortest-queue policy to ensure stability and attain maximum throughput under centralized/decentralized settings. Our analysis and design can also be applied to specific network control problems with stability issues.

Acknowledgments

The authors thank the valuable inputs from the anonymous reviewers and the editors. The authors also appreciate the discussion with Prof. S. Amin at MIT.

References

[1] P. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 251–260, 1995.

[2] S. P. Meyn, "Sequencing and routing in multiclass queueing networks part i: Feedback regulation," *SIAM Journal on Control and Optimization*, vol. 40, no. 3, pp. 741–776, 2001.

[3] S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli, "Dynamic vehicle routing with priority classes of stochastic demands," *SIAM Journal on Control and Optimization*, vol. 48, no. 5, pp. 3224–3245, 2010.

[4] R. Zhang, F. Rossi, and M. Pavone, "Analysis, control, and evaluation of mobility-on-demand systems: a queueing-theoretical approach," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 1, pp. 115–126, 2018.

[5] M. Wu, L. Jin, S. Amin, and P. Jaillet, "Signaling game-based misbehavior inspection in v2i-enabled highway operations," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 2728–2734.

[6] D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis, "Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance," *The Annals of Applied Probability*, pp. 43–75, 1994.

[7] D. Down and S. P. Meyn, "Piecewise linear test functions for stability and instability of queueing networks," *Queueing Systems*, vol. 27, no. 3-4, pp. 205–226, 1997.

[8] R. G. Gallager, *Stochastic processes: theory for applications*. Cambridge University Press, 2013.

[9] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *The Annals of Applied Probability*, pp. 49–77, 1995.

[10] S. Foss and N. Chernova, "On the stability of a partially accessible multi-station queue with state-dependent routing," *Queueing Systems*, vol. 29, no. 1, pp. 55–73, 1998.

[11] Y. Tang and L. Jin, "Analysis and control of dynamic flow networks subject to stochastic cyber-physical disruptions," *arXiv preprint arXiv:2004.00159*, 2020.

[12] Y. Sarikaya, T. Alpcan, and O. Ercetin, "Dynamic pricing and queue stability in wireless random access games," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 2, pp. 140–150, 2011.

[13] P. Dube and R. Jain, "Bertrand games between multi-class queues," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 8588–8593.

[14] K. Savla and E. Frazzoli, "A dynamical queue approach to intelligent task management for human operators," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 672–686, 2011.

[15] G. Foschini and J. Salz, "A basic dynamic routing problem and diffusion," *IEEE Transactions on Communications*, vol. 26, no. 3, pp. 320–327, 1978.

[16] A. Ephremides, P. Varaiya, and J. Walrand, "A simple dynamic routing problem," *IEEE transactions on Automatic Control*, vol. 25, no. 4, pp. 690–693, 1980.

[17] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich, "Queueing system with selection of the shortest of two queues: An asymptotic approach," *Problemy Peredachi Informatsii*, vol. 32, no. 1, pp. 20–34, 1996.

[18] P. Eschenfeldt and D. Gamarnik, "Join the shortest queue with many servers. the heavy-traffic asymptotics," *Mathematics of Operations Research*, vol. 43, no. 3, pp. 867–886, 2018.

[19] V. Gupta, M. H. Balter, K. Sigman, and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms," *Performance Evaluation*, vol. 64, no. 9-12, pp. 1062–1081, 2007.

[20] A. Mukhopadhyay and R. R. Mazumdar, "Analysis of randomized join-the-shortest-queue (jsq) schemes in large heterogeneous processor-sharing systems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 116–126, 2015.

[21] S. Mehdian, Z. Zhou, and N. Bambos, "Join-the-shortest-queue scheduling with delay," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 1747–1752.

[22] Y. Tang, Y. Wen, and L. Jin, "Security risk analysis of the shorter-queue routing policy for two symmetric servers," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 5090–5095.

[23] M. Bramson *et al.*, "Stability of join the shortest queue networks," *The Annals of Applied Probability*, vol. 21, no. 4, pp. 1568–1625, 2011.

[24] J. Dai, J. J. Hasenbein, and B. Kim, "Stability of join-the-shortest-queue networks," *Queueing Systems*, vol. 57, no. 4, pp. 129–145, 2007.

[25] R. D. Foley and D. R. McDonald, "Join the shortest queue: stability and exact asymptotics," *The Annals of Applied Probability*, vol. 11, no. 3, pp. 569–607, 2001.

[26] X. Ling, M.-B. Hu, R. Jiang, and Q.-S. Wu, "Global dynamic routing for scale-free networks," *Physical Review E*, vol. 81, no. 1, p. 016113, 2010.

[27] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queueing network control," in *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*. IEEE, 1994, pp. 318–322.

[28] D. Towsley, "Queuing network models with state-dependent routing," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 323–337, 1980.

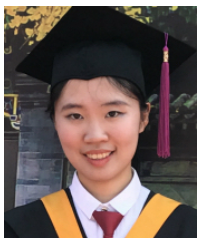
[29] F. Kelly and C. Laws, "Dynamic routing in open queueing networks: Brownian models, cut constraints and resource pooling," *Queueing systems*, vol. 13, no. 1-3, pp. 47–86, 1993.

[30] P. Sarachik and U. Ozguner, "On decentralized dynamic routing for congested traffic networks," *IEEE Transactions on Automatic Control*, vol. 27, no. 6, pp. 1233–1238, 1982.

[31] J. Gregoire, X. Qian, E. Frazzoli, A. De La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, 2014.

[32] P. Varaiya, "Max pressure control of a network of signalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 177–195, 2013.

[33] J. G. Dai and S. P. Meyn, "Stability and convergence of moments for multiclass queueing networks via fluid limit models," *IEEE Transactions on Automatic Control*, vol. 40, no. 11, pp. 1889–1904, 1995.



Qian Xie is a PhD student at the School of Operations Research and Information Engineering, Cornell University, USA. She received B.Eng. degree in Computer Science from Tsinghua University, China in 2019 and M.S. degree from New York University, USA in 2021. She was also a research assistant at NYU's C2SMART University Transportation Center (2019–2021). Her research focuses on the application of queuing theory, game theory and stochastic model to the control and optimization of queueing network systems. Applications include intelligent transportation systems, communication networks, and production lines.

Li Jin is John Wu & Jane Sun Assistant Professor (2021–present) at the UM Joint Institute and the Department of Automation, Shanghai Jiao Tong University (SJTU), China. He is also Research Assistant Professor and was Assistant Professor (2018–2020) in the Tandon School of Engineering, New York University, USA. He received his B.Eng. from SJTU in 2011, M.S. from Purdue University, USA in 2012, and Ph.D. from the Massachusetts Institute of Technology, USA in 2018. He was also a Visiting Scholar at the University of Erlangen-Nuremberg, Germany in 2016. He is interested in theory and application of control of smart and connected systems, including connected and autonomous vehicles, intelligent transportation systems, air traffic control, and smart pipeline transportation.