Maximally Stabilizing Task Release Control Policy for a Dynamical Queue

Ketan Savla Emilio Frazzoli

***Abstract*— In this paper, we consider the following stability problem for a novel dynamical queue. Independent and identical tasks arrive for a queue at a deterministic rate. The server spends deterministic state-dependent times to service these tasks, where the server state is governed by its utilization history through a simple dynamical model. Inspired by empirical laws for human performance as a function of mental arousal, we let the service time be related to the server state by a continuous convex function. We consider a task release control architecture which regulates task entry into service. The objective in this paper is to design such task release control policies that can stabilize the dynamical queue for the maximum possible arrival rate, where the queue is said to be stable if the number of tasks awaiting service does not grow unbounded over time. First, we prove an upper bound on the maximum stabilizable arrival rate for any task release control policy by postulating a notion of one-task equilibrium for the dynamical queue and exploiting its optimality. Then, we propose a simple threshold policy that allocates a task to the server only if its state is below a certain fixed value. We prove that this task release control policy ensures stability of the queue for the maximum possible arrival rate.**

I. Introduction

In this paper, we study the stability problem for a novel dynamical queue, whose service times are dependent on the state of the server. The evolution of the server state, and hence the service times rendered by it, are governed by its utilization history. Identical and independent tasks arrive at a deterministic rate and need to be serviced by the server in the order in which they arrive. We consider a task release control architecture that schedules the beginning of service of each task after its arrival. In this paper, we design such a task release control policy that ensures stability of the queue for the maximum possible arrival rate, where the queue is said to be stable if the number of tasks awaiting service do not grow unbounded over time.

Queueing systems, which is a framework to study systems with waiting lines, is used to model several scenarios in commerce, health-care, engineering domains, etc. Some of the quantities of interest in queueing systems are maximum sustainable workload, average waiting time, etc. An extensive treatment on queueing systems can be found in several texts, e.g., see [1], [2]. The queueing system considered in this paper falls in the category of queueing systems with state dependent parameters, e.g., see [3]. In particular, we consider a queueing system with state-dependent service times. Such systems are useful models for many practical situations, especially when the server corresponds to a human operator in a broad range of settings including, for example, human operators supervising Unmanned Aerial Vehicles, and personnel on job floor in a typical production system. The model for state-dependent service times in this paper is inspired by a well known empirical law from psychology – the Yerkes-Dodson law [4], which states that the human performance increases with mental arousal up to a point and decreases thereafter. Our model in this paper is in the same spirit as the one in [5] where the authors consider a state-dependent queueing system whose service rate is first increasing and then decreasing as a function of the amount of outstanding work. However, our model differs in the sense that the service times are function of the utilization history rather than the outstanding amount of work. A similar model has also been reported in human factors literature, e.g., see [6].

The control architecture considered in this paper falls under the category of *task release control* which has been typically used in production planning, e.g., see [7], [8], to control the release of jobs to a production system in order to deal with machine failures, input fluctuations and variations in operator workload. The task release control architecture is different than an *admission control* architecture, e.g., see [9], [10], [5], where the objective is, given a measure of the *quality of service* to be optimized, to determine criteria on the basis of which to accept or reject incoming tasks. In the setting of this paper, no task is dropped and the task release controller simply acts like a switch regulating access to the server and hence effectively determines the schedule for the beginning of service of each task after its arrival.

The contributions of the paper are threefold. First, we propose a novel dynamical queue, whose server characteristics are inspired by empirical laws relating human performance to mental arousal. Second, we provide an upper bound on the arrival rate under which the queue is stable under any task release control policy. Third, we propose a simple threshold policy that matches this bound, thereby also giving the stability condition for this queue.

Due to space limitations, we omit or only sketch the proofs at a few places. Further details can be found in [11].

II. Problem Formulation

Consider the following single-server queue model. Tasks arrive periodically, at rate λ, i.e., a new task arrives every $1/\lambda$ time units. The tasks are identical and independent of each other and need to serviced in the order of their arrival. We next state the dynamical model for server that determines the service times for each task.

The authors are with the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. {ksavla,frazzoli}@mit.edu.

A. Server Model

Let $x(t)$ be the server state at time t. Let $b : \mathbb{R} \to \{0, 1\}$ be an indicator function such that $b(t)$ is 1 if the server is busy at time t and 0 otherwise.

The evolution of $x(t)$ is governed by a simple first order model:

$$\dot{x}(t) = \frac{b(t) - x(t)}{\tau}, \qquad x(0) = x_0, \tag{1}$$

where τ is a time constant that determines the extent to which past utilization affects the current state of the server, and $x_0 \in [0, 1]$ is the initial condition. Note that the flow described by Equation (1) is such that, for any $\tau > 0$, $x_0 \in [0, 1]$ implies that $x(t) \in [0, 1]$ for all $t \geq 0$.

The service times are related to the state $x(t)$ through a map $\mathcal{S} : [0, 1] \to \mathbb{R}_+$. If a task is allocated to the server at state x, then the service time rendered by the server on that task is $\mathcal{S}(x)$. Since the controller cannot interfere the server while it is servicing a task, the only way in which it can control the server state is by scheduling the beginning of service of tasks after their arrival. Such controllers are called task release controllers and will be formally characterized later on. In this paper we assume that:

$$\mathcal{S}(x) \text{ is positive valued, continuous and convex.}$$

Let $\mathcal{S}_{\min} := \min\{\mathcal{S}(x) \mid x \in [0, 1]\}$, and $\mathcal{S}_{\max} := \max\{\mathcal{S}(0), \mathcal{S}(1)\}$.

Note that, $\mathcal{S}(x)$ does not necessarily have to be increasing in x since it has been noted in the human factors literature (e.g., see [4]) that, for certain cognitive tasks demanding persistence, the performance (which in our case would correspond to the inverse of $\mathcal{S}(x)$) could *increase* with the state x when x is small. This is mainly because a certain minimum level of mental arousal is required for optimal performance. Moreover, our assumption on $\mathcal{S}(x)$ being convex does not rule out the case when $\mathcal{S}(x)$ is increasing in x. An experimental justification of this server model in the context of humans-in-loop systems is included in our earlier work [12], where $\mathcal{S}(x)$ is a U-shaped curve.

B. Task Release Control Policy

We now describe the task release control policies for the dynamical queue. Without explicitly specifying its domain, a task release controller u acts like an on-off switch at the entrance of the queue. Therefore, in short, u is a task release control policy if $u(t) \in \{\text{ON}, \text{OFF}\}$ for all $t \geq 0$, and an outstanding task is assigned to the server if and only if the server is idle, i.e., when it is not servicing a task, *and* when $u = \text{ON}$. Let $\mathcal{U}$ be the set of all such task release control policies. Note that we allow $\mathcal{U}$ to be quite general in the sense that it includes control policies that are functions of λ, $\mathcal{S}$, x, etc.

C. Problem Statement

We now formally state the problem. For a given $\tau > 0$, let $n_u(t, \tau, \lambda, x_0, n_0)$ be the queue length, i.e., the number of outstanding tasks, at time t under task release control policy $u \in \mathcal{U}$ when the task arrival rate is λ and when the server state and the queue length at time $t = 0$ are x_0 and n_0 respectively. Define the maximum stabilizable arrival rate for policy u as:

$$\lambda_{\max}(\tau, u) = \sup\{\lambda | \limsup_{t \to +\infty} n_u(t, \tau, \lambda, x_0, n_0) < +\infty \quad \forall x_0 \in [0, 1], \quad \forall n_0 \in \mathbb{N}\}.$$

The maximum stabilizable arrival rate over all policies is defined as:

$$\lambda^*_{\max}(\tau) = \sup_{u \in \mathcal{U}} \lambda_{\max}(\tau, u).$$

A task release control policy u is called *maximally stabilizing* if, for any $x_0 \in [0, 1]$, $n_0 \in \mathbb{N}$, $\tau > 0$, $\limsup_{t \to +\infty} n_u(t, \tau, \lambda, x_0, n_0) < +\infty$ for all $\lambda \leq \lambda^*_{\max}(\tau)$, The objective in this paper is to design a maximally stabilizing task release control policy for the dynamical queue whose server state evolves according to Equation (1), and whose service time function $\mathcal{S}(x)$ is positive, continuous and convex.

D. The D/D/1 Queue

It is instructive to compare our setup with the standard D/D/1 queue [1], where independent and identical tasks arrive at a deterministic rate of $\lambda > 0$ and the service time for each task is constant $s > 0$. In that case, it is known that the maximum stabilizable arrival rate is $1/s$ and that the trivial policy $u(t) \equiv \text{ON}$ is maximally stabilizing. In our formulation, the service times are state-dependent and the server state is a function of its utilization profile. Therefore, a simple stability condition or a task release controller is not obvious. Note that, in the limit as $\tau \to +\infty$ in Equation (1) and/or setting $\mathcal{S}(x) \equiv c$ for some constant $c > 0$ corresponds to the standard D/D/1 queue setting.

III. Upper Bound

In this section, we prove an upper bound on $\lambda^*_{\max}(\tau)$. We do this in several steps. We start by introducing a notion of *one-task equilibrium* for the dynamical queue under consideration.

A. One-task Equilibrium

Let x_i be the server state at the beginning of service of the i-th task and let the queue length be zero at that instant. The server state upon arrival of the $(i + 1)$-th task is then evaluated by integration of (1) over the time period $[0, 1/\lambda]$, with initial condition $x_0 = x_i$. Let x'_i denote the server state when it has completed service of the i-th task. Then, $x'_i = 1 - (1 - x_i)e^{-\mathcal{S}(x_i)/\tau}$. Assuming that $\mathcal{S}(x_i) \leq 1/\lambda$, we get that $x_{i+1} = x'_i e^{-(1/\lambda - \mathcal{S}(x_i))/\tau}$, and finally

$$\begin{aligned} x_{i+1} &= (1 - (1 - x_i)e^{-\mathcal{S}(x_i)/\tau})e^{(\mathcal{S}(x_i) - 1/\lambda)/\tau} \\ &= \left(x_i - 1 + e^{\mathcal{S}(x_i)/\tau}\right) e^{-\frac{1}{\lambda\tau}}. \end{aligned}$$

If λ and τ are such that $x_{i+1} = x_i$, then under the trivial control policy $u(t) \equiv \text{ON}$, the server state at the beginning of all tasks after and including the i-th task will be x_i. We then say that the server is at *one-task equilibrium* at x_i. Therefore, for a given λ and τ, one-task equilibrium server states correspond to $x \in [0, 1]$ satisfying $x = \left(x - 1 + e^{\mathcal{S}(x)/\tau}\right) e^{-\frac{1}{\lambda\tau}}$ and $\mathcal{S}(x) \leq 1/\lambda$, i.e., when

$\mathcal{S}(x) = \tau \log\left(1-(1-e^{\frac{1}{\lambda\tau}})x\right)$ and $\mathcal{S}(x) \leq 1/\lambda$. Let us define a map $\mathcal{R} : [0,1] \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ as:

$$\mathcal{R}(x,\tau,\lambda) := \tau \log\left(1-(1-e^{\frac{1}{\lambda\tau}})x\right). \quad (2)$$

The following result establishes some key properties of $\mathcal{R}(x,\tau,\lambda)$.

Lemma 3.1: For any $\tau > 0$ and $\lambda > 0$, the function $\mathcal{R}$ defined in Equation (2) is strictly concave in x, and $\frac{\partial}{\partial x}\mathcal{R}(x,\tau,\lambda) > 0$ for all $x \in [0,1]$.

For a given $\tau > 0$ and $\lambda > 0$, define the set of one-task equilibrium server states as:

$$x_{\rm eq}(\tau,\lambda) := \{x \in [0,1] \mid \mathcal{S}(x) = \mathcal{R}(x,\tau,\lambda)\}. \quad (3)$$

Remark 3.2: Note that we did not include the constraint $\mathcal{S}(x) \leq 1/\lambda$ in the definition of $x_{\rm eq}(\tau,\lambda)$ in Equation (3). This is because this constraint can be shown to be redundant as follows. Equation (2) and Lemma 3.1 imply that, for any $\tau > 0$ and $\lambda > 0$, $\mathcal{R}(x,\tau,\lambda)$ is strictly increasing in x and hence $\mathcal{R}(x,\tau,\lambda) \leq \mathcal{R}(1,\tau,\lambda) = 1/\lambda$ for all $x \in [0,1]$. Therefore, $\mathcal{S}(x_{\rm eq}(\tau,\lambda)) = \mathcal{R}(x_{\rm eq}(\tau,\lambda),\tau,\lambda) \leq 1/\lambda$.

We introduce a couple of more definitions. For a given $\tau > 0$, let

$$\lambda_{\rm eq}^{\rm max}(\tau) := \max\{\lambda > 0 \mid x_{\rm eq}(\tau,\lambda) \neq \emptyset\}, \quad (4)$$

$$x_{\rm th}(\tau) := x_{\rm eq}\left(\tau, \lambda_{\rm eq}^{\rm max}(\tau)\right). \quad (5)$$

We now argue that the definitions in Equations (4) and (5) are well posed. Consider the function $\mathcal{S}(x) - \mathcal{R}(x,\tau,\lambda)$. Since $\mathcal{R}(0,\tau,\lambda) = 0$ for any $\tau > 0$ and $\lambda > 0$, and $\mathcal{S}(0) > 0$, we have that $\mathcal{S}(0) - \mathcal{R}(0,\tau,\lambda) > 0$ for any $\tau > 0$ and $\lambda > 0$. Since $\mathcal{R}(1,\tau,\lambda) = 1/\lambda$, $\mathcal{S}(1) - \mathcal{R}(1,\tau,\lambda) < 0$ for all $\lambda < 1/\mathcal{S}_{\max}$. Therefore, by the continuity of $\mathcal{S}(x) - \mathcal{R}(x,\tau,\lambda)$, the set of equilibrium server states, as defined in Equation (3), is not-empty for all $\lambda < 1/\mathcal{S}_{\max}$. Moreover, $\mathcal{R}(x,\tau,\lambda) \leq \mathcal{R}(1,\tau,\lambda) = 1/\lambda$ for all $x \in [0,1]$, $\mathcal{S}(x) - \mathcal{R}(x,\tau,\lambda) \geq \mathcal{S}(x) - 1/\lambda$ for all $x \in [0,1]$. Therefore, for all $\lambda > 1/\mathcal{S}_{\min}$, the set of equilibrium states, as defined in Equation (3), is a null set. Hence, $\lambda_{\rm eq}^{\rm max}(\tau)$ and $x_{\rm th}(\tau)$ are well-defined. In general, for a given $\tau > 0$ and $\lambda > 0$, $x_{\rm eq}(x,\tau)$ is not a singleton, e.g., see Figure 1. However, due to the strict convexity of $\mathcal{S}(x) - \mathcal{R}(x,\tau,\lambda)$ in x as implied by Lemma 3.1, $x_{\rm th}(\tau)$ contains only one element. In the rest of the paper, $x_{\rm th}(\tau)$ will denote this single element.

In the rest of the paper, we will restrict our attention on those τ and $\mathcal{S}(x)$ for which $x_{\rm th}(\tau) < 1$. Loosely speaking, this is satisfied when $\mathcal{S}(x)$ is increasing on some interval in $[0,1]$ and the increasing part is *steep enough* (e.g., see Figure 1). It is reasonable to expect this assumption to be satisfied in the context of human operators whose performance deteriorates quickly at very high utilizations. Mathematically, $x_{\rm th}(\tau) < 1$ implies that $\lambda_{\rm eq}^{\rm max}(\tau)$ (which will be proven to be the maximum stabilizable arrival rate) is strictly greater than $1/\mathcal{S}(1)$, i.e., the rate at which the server is able to service tasks starting with initial condition $x_0 = 1$ and servicing tasks continuously. The implications of the case when $x_{\rm th}(\tau) = 1$ are discussed briefly at appropriate places in the paper.

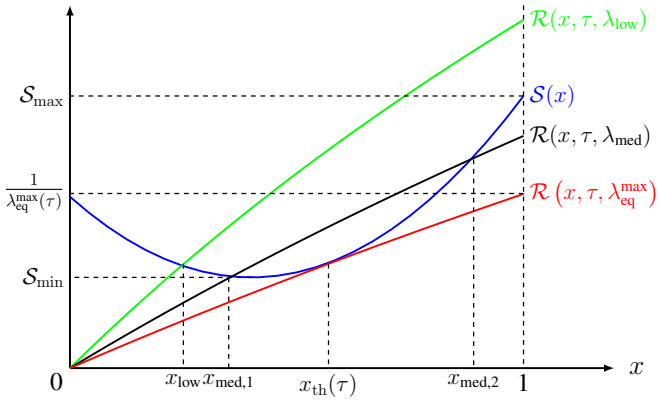

Fig. 1. A typical $\mathcal{S}(x)$ along with $\mathcal{R}(x,\tau,\lambda)$ for three values of λ: $\lambda_{\rm low}$, $\lambda_{\rm med}$ and $\lambda_{\rm eq}^{\rm max}(\tau)$ in the increasing order. Also, $x_{\rm eq}(\tau,\lambda_{\rm low}) = \{x_{\rm low}\}$, $x_{\rm eq}(\tau,\lambda_{\rm med}) = \{x_{\rm med,1}, x_{\rm med,2}\}$ and $x_{\rm eq}\left(\tau,\lambda_{\rm eq}^{\rm max}(\tau)\right) = \{x_{\rm th}(\tau)\}$. Note that, since $x_{\rm th}(\tau) < 1$, then $\lambda_{\rm eq}^{\rm max}(\tau)$ is the value of λ at which $\mathcal{R}(x,\tau,\lambda)$ is tangential to $\mathcal{S}(x)$.

The following property of $\mathcal{S}(x)$ will be used later on.

Lemma 3.3: For any $\tau > 0$, if $x_{\rm th}(\tau) < 1$, then $\frac{d}{dx}\mathcal{S}(x)|_{x=x_{\rm th}(\tau)} > 0$.

We next consider a *static* problem and establish results there that will be useful for the dynamic case.

B. The Static Problem

Consider the following *n-task static problem*: Given n tasks, what is the fastest way for the dynamical server to service these tasks starting with an initial state x and ending at final state x. We emphasize here that all the n tasks are initially enqueued and no new tasks arrive. Let $T_f(x,\tau,n,u)$ be the time required by the task release control policy $u \in \mathcal{U}$ for the n-task static problem with initial and final server state $x \in [0,1]$. The following bound on $T_f(x,\tau,n,u)$ will be critical in proving a sharp upper bound on $\lambda_{\rm max}^*(\tau)$.

Lemma 3.4: For any $x \in [0,1]$, $\tau > 0$, $n \in \mathbb{N}$ and $u \in \mathcal{U}$, we have that $T_f(x,\tau,n,u) \geq n/\lambda_{\rm eq}^{\rm max}(\tau)$.

C. Upper Bound on Stabilizable Arrival Rate

We now return to the dynamic problem, where we prove an upper bound on $\lambda_{\rm max}^*(\tau)$. Trivially, $\lambda_{\rm max}^*(\tau) \leq \frac{1}{\mathcal{S}_{\min}}$. We next establish a sharper upper bound. First, we state a useful lemma.

Lemma 3.5: For any $\tau > 0$, $x_0 \in [0,1]$, $n_0 \in \mathbb{N}$ and $\lambda > \lambda_{\rm eq}^{\rm max}(\tau)$, if $x_{\rm th}(\tau) < 1$ then there exist constants $x_L(\tau)$ and $x_U(\tau)$ satisfying $0 < x_L(\tau) < x_U(\tau) < 1$ such that for any $u \in \mathcal{U}$ under which the server states at the beginning of tasks do not lie in $[x_L(\tau), x_U(\tau)]$ infinitely often, we have that $\limsup_{t\to+\infty} n_u(t,\tau,\lambda,x_0,n_0) = +\infty$.

Proof: We first define the constants $x_L(\tau)$ and $x_U(\tau)$. For a given $\tau > 0$, let $x_{\min} := 1 - e^{-\mathcal{S}_{\min}/\tau}$ denote a lower bound on the lowest possible server state immediately after the service of a task. Note that, for any $\tau > 0$ and $\mathcal{S}_{\min} > 0$, $x_{\min} > 0$. For a given $\tau > 0$, define a map $g : [0,1] \to \mathbb{R} \cup \{+\infty\}$ as:

$$g(x) = \mathcal{S}_{\min} + \tau \log(x_{\min}/x). \quad (6)$$

Note that g is continuous, strictly decreasing with respect to x, and that $g(0) = +\infty$. Therefore, by continuity argument, there exists a $\tilde{x} > 0$ such that $g(x) > 1/\lambda_{\text{eq}}^{\max}(\tau)$ for all $x \in [0, \tilde{x})$. Define $x_{l_1}(\tau) := \min\{x_{\min}, \tilde{x}\}$. It follows from the previous arguments that $x_{l_1}(\tau) > 0$. Define the following quantities

$$\begin{aligned} x_{u_1} &= \max\{x \in [0,1] \mid \mathcal{S}(x) = 1/\lambda_{\text{eq}}^{\max}(\tau)\}, \\ x_{u_2} &= 1 - (1 - x_{l_1})e^{-2/\lambda_{\text{eq}}^{\max}(\tau)}, \\ x_{l_2} &= x_{u_2} e^{-2/\lambda_{\text{eq}}^{\max}(\tau)}, \end{aligned} \tag{7}$$

$$x_L(\tau) = \min\{x_{l_1}, x_{l_2}\}, \quad x_U(\tau) = \max\left\{\frac{1 + x_{u_1}}{2}, x_{u_2}\right\},$$

where we have dropped the dependency of x_{l_1}, x_{l_2}, x_{u_1} and x_{u_2} on τ for the sake of conciseness. From the above definitions and since $x_{\text{th}}(\tau) < 1$, it is easily seen that $x_{l_2} > 0$ and $x_{u_2} < 1$. Moreover, $x_{\text{th}}(\tau) < 1$ implies that $1/\lambda_{\text{eq}}^{\max}(\tau) < \mathcal{S}(1)$, and hence we get that $x_{u_1} < 1$. This combined with $x_{l_1} > 0$ as argued earlier, we get that $x_L(\tau) > 0$ and $x_U(\tau) < 1$. Equation (7) also implies that $x_{u_2} > x_{l_1}$ and $x_{u_2} > x_{l_2}$. Therefore, $x_L(\tau) < x_U(\tau)$.

Having defined the constants $x_L(\tau)$ and $x_U(\tau)$, we now prove the statement of the lemma. For the rest of the proof, we drop the dependency of x_L and x_U on τ. Consider a u such that the maximum task index for which the server state lies in $[x_L, x_U]$ is finite, say I. Let x_i and x_i' be the server states at the beginning of service of task i and the end of service of task i respectively. Consider a service cycle of a typical task for $i > I$. We now consider four cases depending on where x_i and x_{i+1} belong, and in each case we show that the time between the beginning of successive tasks is strictly greater than $1/\lambda_{\text{eq}}^{\max}(\tau)$, thereby establishing the lemma.

- $x_i \in [0, x_L)$ and $x_{i+1} \in [0, x_L)$: The service time for task i, $\mathcal{S}(x_i)$, is lower bounded by $\mathcal{S}_{\min}$. By the definition of $x_{\min}$, $x_i' \geq x_{\min}$ and hence $x_i' \geq x_L$. Since x_{i+1} is less than x_L, the server has to idle for time $\tau \log(x_i'/x_{i+1})$, which is lower bounded by $\tau \log(x_{\min}/x_L)$. In summary, the total time between the service of successive tasks is lower bounded by $\mathcal{S}_{\min} + \tau \log(x_{\min}/x_L)$, which is equal to $g(x_L)$ from Equation (6). By the choice of x_L, $g(x_L)$ is strictly greater than $1/\lambda_{\text{eq}}^{\max}(\tau)$.
- $x_i \in (x_U, 1]$ and $x_{i+1} \in (x_U, 1]$: The convexity of $\mathcal{S}(x)$ along with Lemma 3.3 imply that $\mathcal{S}(x) > 1/\lambda_{\text{eq}}^{\max}(\tau)$ for all $x \in (x_{u_1}, 1]$. Since $x_U > x_{u_1}$ from Equation (7), we have that $\mathcal{S}(x_i) > 1/\lambda_{\text{eq}}^{\max}(\tau)$. Therefore, the time spent between successive tasks is lower bounded by $1/\lambda_{\text{eq}}^{\max}(\tau)$.
- $x_i \in [0, x_L)$ and $x_{i+1} \in (x_U, 1]$: The fact that it takes at least $2/\lambda_{\text{eq}}^{\max}(\tau)$ amount of service time on task i for the server to go from from x_i to x_{i+1} follows from the definition of x_{l_1} and x_{u_2} and their relation to x_L and x_U respectively, as stated in Equation (7). Therefore, the time spent between successive tasks is at least $2/\lambda_{\text{eq}}^{\max}(\tau)$.
- $x_i \in (x_U, 1]$ and $x_{i+1} \in [0, x_L)$: The fact that it takes at least $2/\lambda_{\text{eq}}^{\max}(\tau)$ time for the server to idle from x_i' to x_{i+1} follows from the definition of x_{l_2} and x_{u_2} and their relation to x_L and x_U respectively, as stated in Equation (7). Therefore, the time spent between successive tasks is at least $2/\lambda_{\text{eq}}^{\max}(\tau)$.

■

Theorem 3.6: For any $\tau > 0$, $x_0 \in [0,1]$, $n_0 \in \mathbb{N}$, $\lambda > \lambda_{\text{eq}}^{\max}(\tau)$ and $u \in \mathcal{U}$, if $x_{\text{th}}(\tau) < 1$ then we have that $\limsup_{t \to +\infty} n_u(t, \tau, \lambda, x_0, n_0) = +\infty$.

Proof: Lemma 3.5 implies that there exist $x_L > 0$ and $x_U < 1$ such that it suffices to consider set of task release control policies under which the server states at the beginning of service of tasks lie in $[x_L, x_U]$ infinitely often. Consider one such control policy and let the sequence of indices of tasks for which the server state at the beginning of their service belongs to $[x_L, x_U]$ be denoted as $i_1, i_2, \ldots$. Let x_i and t_i be the server state and the time respectively at the beginning of the service of the i-th task. We have that $x_{i_k} \in [x_L, x_U]$ for all $k \geq 1$. Define constants κ_1 and κ_2 as follows:

$$\kappa_1 := -\tau \log x_L, \quad \kappa_2 := -\tau \log(1 - x_U). \tag{8}$$

Note that both κ_1 and κ_2 are positive. For each $k > 1$, we now relate $t_{i_k} - t_{i_1}$ to the time for a related static problem. If $x_{i_k} \geq x_{i_1}$, then consider the $(i_k - i_1)$-task static problem with initial and final server state x_{i_1}. Then, for a control policy u' for this static problem under which the server states are $x_{i_1}, x_{i_1+1}, \ldots, x_{i_k}$, we have that $T_f(x_{i_1}, \tau, i_k - i_1, u') = t_{i_k} - t_{i_1} + \tau(\log x_{i_k} - \log x_{i_1})$. Therefore, $t_{i_k} - t_{i_1} = T_f(x_{i_1}, \tau, i_k - i_1, u') - \tau(\log x_{i_k} - \log x_{i_1}) \geq T_f(x_{i_1}, \tau, i_k - i_1, u') + \tau \log x_{i_1} \geq T_f(x_{i_1}, \tau, i_k - i_1, u') - \kappa_1$, where the last inequality follows from Equation (8). If $x_{i_k} < x_{i_1}$, then consider the $(i_k - i_1 + m)$-task static problem with initial and final server state x_{i_1} and a control policy u'' for this static problem such that: the server states at the beginning of the service of first $i_k - i_1$ tasks are $x_{i_1}, x_{i_1+1}, \ldots, x_{i_k}$ and m is the smallest number such that on servicing these m tasks without any idling after i_k-th task, one has that $x_{i_k+m} \geq x_{i_1}$. One can upper bound the time for the static problem under u'' as $T_f(x_{i_1}, \tau, m + i_k - i_1, u'') \leq t_{i_k} - t_{i_1} + \tau\left(\log(1 - x_{i_1}) - \log(1 - x_{i_k})\right) + \mathcal{S}_{\max} - \tau \log x_{i_1}$. Hence, one can write that $t_{i_k} - t_{i_1} \geq T_f(x_{i_1}, \tau, m + i_k - i_1, u'') - \mathcal{S}_{\max} + \tau \log(1 - x_{i_k}) + \tau \log x_{i_1} \geq T_f(x_{i_1}, \tau, m + i_k - i_1, u'') - \mathcal{S}_{\max} - \kappa_1 - \kappa_2$, where the last inequality follows from Equation (8).

Combining these bounds on $t_{i_k} - t_{i_1}$ with lemma 3.4, we have that, for all $k \geq 1$,

$$t_{i_k} - t_{i_1} \geq \begin{cases} \frac{i_k - i_1}{\lambda_{\text{eq}}^{\max}(\tau)} - \kappa_1 & \text{if } x_{i_k} \geq x_{i_1}, \\ \frac{m + i_k - i_1}{\lambda_{\text{eq}}^{\max}(\tau)} - \mathcal{S}_{\max} - \kappa_1 - \kappa_2 & \text{otherwise.} \end{cases} \tag{9}$$

With $\kappa = \kappa_1 + \kappa_2 + \mathcal{S}_{\max}$, we can write Equation (9) in compact form as

$$t_{i_k} - t_{i_1} \geq \frac{i_k - i_1}{\lambda_{\text{eq}}^{\max}(\tau)} - \kappa \quad \forall k \geq 1. \tag{10}$$

For $k \geq 1$, let n_k be the queue length at the beginning of service of task i_k. Then one can write that,

$$n_k \geq n_1 + \lambda(t_{i_k} - t_{i_1}) - (i_k - i_1) \quad \forall k \geq 1.$$

Combining this with Equation (10), we get that,

$$n_k \geq n_1 - \lambda\kappa + (i_k - i_1)\left(\frac{\lambda}{\lambda_{\text{eq}}^{\max}(\tau)} - 1\right) \quad \forall k \geq 1. \quad (11)$$

From Equation (11), we get that, for $\lambda > \lambda_{\text{eq}}^{\max}(\tau)$, $\lim_{k\to+\infty} n_k = +\infty$. The theorem follows from the fact that $\limsup_{t\to+\infty} n_u(t,\tau,\lambda,x_0,n_0) \geq \lim_{k\to+\infty} n_k$. ■

Remark 3.7: (i) Theorem 3.6 establishes that, for a given $\tau > 0$, if $x_{\text{th}}(\tau) < 1$ then $\lambda_{\max}^*(\tau) \leq \lambda_{\text{eq}}^{\max}(\tau)$.

(ii) If $x_{\text{th}}(\tau) = 1$, then one can show that, for any $\epsilon > 0$, there exists no stabilizing task release control policy for arrival rate greater than $\lambda_{\text{eq}}^{\max}(\tau)+\epsilon$, i.e., $\lambda_{\max}^*(\tau) \leq \lambda_{\text{eq}}^{\max}(\tau) + \epsilon$.

In the next section, we propose a simple task release control policy and prove that it is maximally stabilizable, i.e., for any $\lambda \leq \lambda_{\text{eq}}^{\max}(\tau)$, it ensures that the dynamical queue is stable.

IV. Control Policy and Lower Bound on Stabilizable Arrival Rate

In this section, we propose a threshold policy. It can be stated as follows:

$$u_{\text{TP}}(t) = \begin{cases} \text{ON} & \text{if } x(t) \leq x_{\text{th}}(\tau), \\ \text{OFF} & \text{otherwise,} \end{cases}$$

where $x_{\text{th}}(\tau)$ is as defined in Equation (5). We now prove that this threshold policy is maximally stabilizing.

Theorem 4.1: For any $\tau > 0$, $x_0 \in [0,1]$, $n_0 \in \mathbb{N}$ and $\lambda \leq \lambda_{\text{eq}}^{\max}(\tau)$, if $x_{\text{th}}(\tau) < 1$ then we have that $\limsup_{t\to+\infty} n_{u_{\text{TP}}}(t,\tau,\lambda,x_0,n_0) < +\infty$.

Proof: Let x_i and t_i be the server state and time instants respectively at the beginning of service of the i-th task. For brevity in notation, let $n(t)$ be the queue length at time t. For any $x_0 \in [0,1]$ and $n_0 \in \mathbb{N}$, considering the possibility when $x_0 > x_{\text{th}}(\tau)$ we have that $n(t_1) = \max\{0, n_0-1, n_0 - 1 + \lfloor \lambda\tau \log(x_0/x_{\text{th}})\rfloor\}$. We now prove that $n(t_i) \leq n(t_1) + \lceil \tau(1-x_{\text{th}})\,(1/\mathcal{S}_{\max} - \lambda)\rceil + \lceil -\lambda\tau \log x_{\text{th}}\rceil$ for all i through the following two cases:

- **State 1:** $x_1 = x_{\text{th}}$. While $n(t_i) > 0$, we have that $x_{i+1} = x_{\text{th}}$ and $t_{i+1} - t_i = T_f(x_{\text{th}},\tau,1,u_{\text{TP}}) = 1/\lambda_{\text{eq}}^{\max}(\tau)$. Therefore, if $\lambda = \lambda_{\text{eq}}^{\max}(\tau)$, then the arrival rate is same as the service rate and hence $n(t_i) \equiv n(t_1)$ for all i. If $\lambda < \lambda_{\text{eq}}^{\max}(\tau)$, then the service rate is greater than the arrival rate and hence there exists an $i' \geq 1$ such that $n(t_i) < n(t_{i-1})$ for all $i \leq i'$ and $n\left(t_{i'} + 1/\lambda_{\text{eq}}^{\max}(\tau)\right) = 0$ and hence $x_{i'+1} < x_{\text{th}}$. Thereafter, we appeal to the next case by resetting $x_{i'+1}$ and $t_{i'+1}$ as x_1 and t_1 respectively. Moreover, with these notations, $n(t_1)$ will be zero.
- **State 2:** $x_1 < x_{\text{th}}$. While the queue length is non-zero, the server is never idle. The maximum amount of continuous service time required for the server state to cross x_{th} starting from any $x_1 < x_{\text{th}}$ is upper bounded by $-\tau\log(1-x_{\text{th}}) + \mathcal{S}_{\max}$. This is possibly followed by an idle time which is upper bounded by $-\tau\log x_{\text{th}}$, at the end of which the server state is x_{th}. Therefore, the maximum number of outstanding tasks when the server state reaches x_{th} is upper bounded by $n_1 + \lceil \tau\log(1-x_{\text{th}})\,(1/\mathcal{S}_{\max} - \lambda)\rceil + \lceil -\lambda\tau\log x_{\text{th}}\rceil$. Thereafter, we appeal to the earlier case by resetting $x_1 = x_{\text{th}}$ and n_1 to be the number of outstanding tasks when the server state reaches x_{th}.

In summary, when the system is in State 1, if $\lambda = \lambda_{\text{eq}}^{\max}(\tau)$, it stays there with constant queue length, else, the queue length monotonically decreases to zero at which point it enters State 2. When the system is in State 2, it stays in it for ever or eventually enters State 1 with bounded queue length. Collecting these facts, we arrive at the result. ■

Remark 4.2: (i) From Theorem 3.6 and Theorem 4.1, one can deduce that, for any $\tau > 0$, $\lambda_{\max}^*(\tau) = \lambda_{\text{eq}}^{\max}(\tau)$, and the threshold policy is a maximally stabilizing task release control policy.

(ii) In general, for a given $\lambda' \leq \lambda_{\text{eq}}^{\max}(\tau)$, the threshold policy with the threshold value set at any value in $[x_{\text{eq}}^1(\tau,\lambda'), x_{\text{eq}}^2(\tau,\lambda')]$ would ensure stability of the queue for all values of $\lambda \leq \lambda'$.

(iii) If $x_{\text{th}}(\tau) = 1$, then one can show that, given $\epsilon > 0$, there exists a $\delta(\epsilon) > 0$ such that the threshold policy with the threshold value set at $1-\delta(\epsilon)$ ensures stability of the queue for all arrival rates less than or equal to $\lambda_{\text{eq}}^{\max}(\tau) - \epsilon$.

V. Simulations

In this section we report results from some numerical experiments. We first present results for the deterministic service times. We select $\tau = 300\,s$ and $\mathcal{S}(x) = 229x^2 - 267x + 99\,s$. These values correspond to the model estimated from experimental data, as reported in [12]. For these values, $x_{\text{th}}(\tau) \approx 0.6$ and $\lambda_{\text{eq}}^{\max}(\tau) \approx 0.03\,s^{-1}$. In Figure 2, we plot the queue lengths for $x_0 = 0.9$, $n_0 = 1$ and $\lambda = 0.025$ for the two cases when $u(t) \equiv \text{ON}$ and $u(t) = u_{\text{TP}}(t)$. Figure 2 illustrates the typical unstable behavior of the dynamical queue under the trivial control policy $u(t) \equiv \text{ON}$.

We also performed simulations for stochastic service times, where the tasks arrive at a deterministic rate and service for a task is sampled from a lognormal distribution with state-dependent mean $\mathcal{S}(x)$ and a constant variance. The server state still evolves according to Equation (1). We assume problem parameters to be same as before, i.e., $\tau = 300\,s$ $\mathcal{S}(x) = 229x^2 - 267x + 99\,s$. Additionally, we set the variance of service times to be $20\,s^2$. In Figure 3, we plot queue lengths for a sample path for $x_0 = 0.4$, $n_0 = 1$ and $\lambda = 0.025$ for the two cases when $u(t) \equiv \text{ON}$ and $u(t) = u_{\text{TP}}(t)$. Figure 3 illustrates the typical unstable behavior of the dynamical queue under the trivial control policy $u(t) \equiv \text{ON}$. Moreover, the simulations suggest that the maximum arrival rate under which the queue is stable under $u_{\text{TP}}(t)$ is very close to $\lambda_{\text{eq}}^{\max}(\tau)$.

VI. Conclusions

In this paper, we studied the stability problem of a dynamical queue whose service times are dependent on the state of a simple underlying dynamical system. The model for the service times is loosely inspired by the performance of a human operator in a persistent mission. We proposed

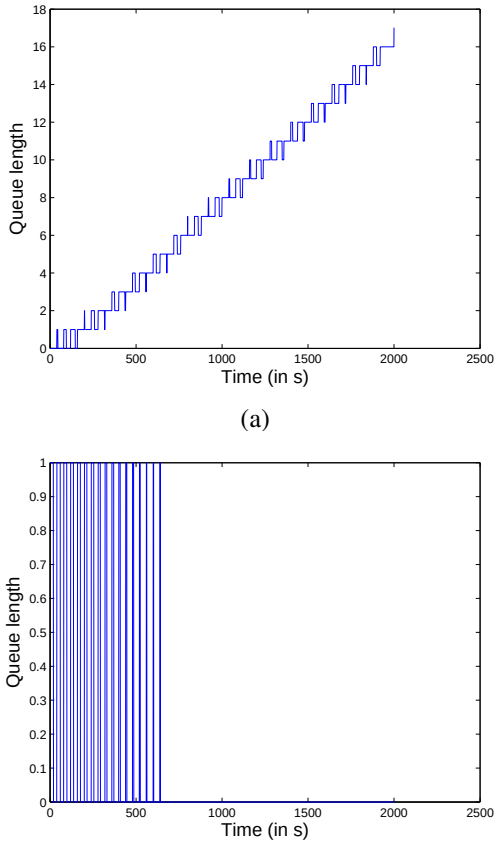

Fig. 2. Comparison of the queue length for the dynamical queue under deterministic service times and for the same problem parameters and same initial condition under (a) $u(t) \equiv \text{ON}$ and (b) $u(t) \equiv u_{\text{TP}}(t)$.

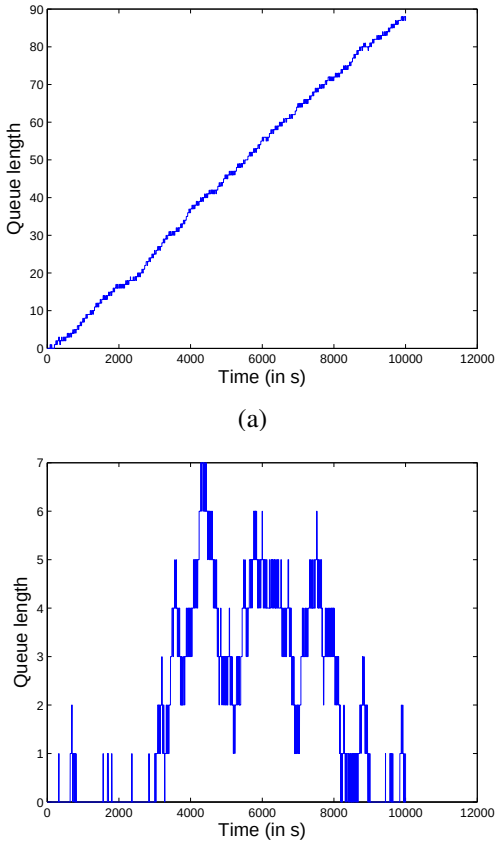

Fig. 3. Comparison of the queue length during a sample run for the dynamical queue under lognormal service times and for the same problem parameters and same initial condition under (a) $u(t) \equiv \text{ON}$ and (b) $u(t) \equiv u_{\text{TP}}(t)$.

a simple task release control policy for such a dynamical queue and proved that it ensures stability of the queue for the maximum possible arrival rate. In future, we plan to extend the analysis here to stochastic inter-arrival and service times and to general server dynamics. We also plan to design control policies for such queues that optimize other qualities of service such as average waiting time of an incoming task.

VII. Acknowledgements

This work was supported in part by the Michigan/AFRL Collaborative Center on Control Science, AFOSR grant no. FA 8650-07-2-3744. The authors thank Tom Temple for helpful discussions. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the supporting organizations.

References

[1] L. Kleinrock, *Queueing Systems I: Theory*. Wiley-Interscience, 1975.
[2] S. Asmussen, *Applied Probability and Queues*. Springer, 2003.
[3] J. H. Dshalalow, ed., *Frontiers in Queuing Models and Applications in Science and Engineering*, ch. Queueing Systems with State Dependent Parameters. CRC press, Inc., 1997.
[4] R. M. Yerkes and J. D. Dodson, "The relation of strength of stimulus to rapidity of habit-formation," *Journal of Comparative Neurology and Psychology*, vol. 18, pp. 459–482, 1908.
[5] R. Bekker and S. C. Borst, "Optimal admission control in queues with workload-dependent service rates," *Probability in the Engineering and Informational Sciences*, vol. 20, pp. 543–570, 2006.
[6] M. L. Cummings and C. E. Nehme, "Modeling the impact of workload in network centric supervisory control settings," in *2nd Annual Sustaining Performance Under Stress Symposium*, (College Park, MD), Feb. 2009.
[7] C. R. Glassey and M. G. C. Resende, "A scheduling rule for job release in semiconductor fabrication," *Operations Research Letters*, vol. 7, no. 5, pp. 213–217, 1988.
[8] J. W. M. Bertrand and H. P. G. V. Ooijen, "Workload based order release and productivity: a missing link," *Production Planning and Control*, vol. 13, no. 7, pp. 665–678, 2002.
[9] S. Stidham, "Optimal control of admission to queueing system," *IEEE Trans. Automatic Control*, vol. 30, pp. 705–713, Aug 1985.
[10] K. Y. Lin and S. M. Ross, "Optimal admission control for a single-server loss queue," *Journal of Applied Probability*, vol. 41, no. 2, pp. 535–546, 2004.
[11] K. Savla and E. Frazzoli, "Maximally stabilizing task release control policy for a dynamical queue," *IEEE Trans. on Automatic Control*, 2010. To appear, Available at `http://arxiv.org/abs/0909.3651`.
[12] K. Savla, C. Nehme, T. Temple, and E. Frazzoli, "Efficient routing of multiple vehicles for human-supervised services in a dynamic environment," in *AIAA Conf. on Guidance, Navigation, and Control*, (Honolulu, HI), 2008.