Maximally Stabilizing Task Release Control Policy for a Dynamical Queue

Ketan Savla and Emilio Frazzoli

Abstract—**In this technical note, we introduce a model of dynamical queue, in which the service time depends on the server utilization history. The proposed queueing model is motivated by widely accepted empirical laws describing human performance as a function of mental arousal. The objective of this technical note is to design task release control policies that can stabilize the queue for the maximum possible arrival rate, assuming deterministic arrivals. First, we prove an upper bound on the maximum possible stabilizable arrival rate for any task release control policy. Then, we propose a simple threshold policy that releases a task to the server only if its state is below a certain fixed value. Finally, we prove that this task release control policy ensures stability of the queue for the maximum possible arrival rate.**

Index Terms—**Queueing analysis, nonlinear dynamical systems, stability.**

I. INTRODUCTION

In this technical note, we introduce a novel model for a dynamical queue, in which service times depend on the utilization history of the server. In other words, we consider the server as a dynamical system, and model the service time as a function of its state. Given this model, we consider the case in which new tasks arrive at a deterministic rate, and propose a task release control architecture that schedules the beginning of service of each task after its arrival. We propose a simple threshold policy that releases a task to the server only if its state is below a certain fixed value. The proposed task release control policy is proven to ensure 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 does not grow unbounded over time.

Queueing theory is a framework to study systems with waiting lines. An extensive treatment of queueing systems can be found in several texts, e.g., see [2], [3]. The queueing system considered in this technical note falls in the category of queueing systems with state-dependent parameters, e.g., see [4]. 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 job shop personnel in a typical production system. The model for state-dependent service times in this technical note is inspired by a well known empirical law from psychology—the Yerkes-Dodson law [5]—which states that human performance increases with mental arousal up to a point and decreases thereafter. Our model in this technical note is in the same spirit as the one in [6], 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,

The authors are with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: ksavla@mit.edu; frazzoli@mit.edu).

our model differs in the sense that the service times are related to the utilization history rather than the outstanding amount of work. A similar model has also been reported in the human factors literature, e.g., see [7].

The control architecture considered in this technical note falls under the category of *task release control*, which has been typically used in production planning to control the release of jobs to a production system in order to deal with machine failures, input fluctuations and variations in operator workload (see, e.g., [8], [9]). The task release control architecture is different from an *admission control* architecture, e.g., see [6], [10], 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 technical note, 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 technical note are threefold. First, we propose a novel dynamical queue, whose server characteristics are inspired by empirical laws relating human performance to utilization history. Second, we provide an upper bound on the maximum possible 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. Third, we propose a simple threshold policy that matches this bound, thereby also giving the stability condition for this queue.

Some preliminary results are presented without proof for space limitations. The proofs are presented in [11].

II. PROBLEM FORMULATION

Consider the following deterministic 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 be serviced in the order of their arrival. We next state the dynamical model for the server, which determines the service times for each task.

A. Server Model

Let $x(t)$ be the server state at time t, and let $b : \mathbb{R} \to \{0, 1\}$ be 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 set $[0, 1]$ is invariant under the dynamics in (1) for any $\tau > 0$ and any $b : \mathbb{R} \to \{0, 1\}$.

The service times are related to the state $x(t)$ through a map $\mathcal{S} : [0, 1] \to \mathbb{R}_{>0}$. 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 technical note we assume that: $\mathcal{S}(x)$ is positive valued, continuous and convex. Let $\mathcal{S}_{\min} := \min \{\mathcal{S}(x) \mid x \in [0, 1]\}$, and $\mathcal{S}_{\max} := \max\{\mathcal{S}(0), \mathcal{S}(1)\}$.

The solution to Equation (1) is $x(t) = e^{-t/\tau} \left(\int_0^t \frac{1}{\tau} b(s) e^{s/\tau} ds + x_0 \right)$. This implies that the server state $x(t)$ is increasing at times when the server is busy, i.e, when $b(t) = 1$, and decreasing at times when the server is not busy, i.e., when $b(t) = 0$. Note that $\mathcal{S}(x)$ is not necessarily monotonically increasing

in x, since it has been noted in the human factors literature (e.g., see [5]) 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 good performance. An experimental justification of this server model in the context of humans-in-loop systems is included in our earlier work [1], where $\mathcal{S}(x)$ for that setup was found to have a U-shaped profile.

B. Task Release Control Policy

We now describe 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 $u(t) \in \{\mathrm{ON}, \mathrm{OFF}\}$ 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 $u = \mathrm{ON}$. 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 $\lambda, \mathcal{S}, x$, 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 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

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

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 technical note is to design a maximally stabilizing task release control policy for the dynamical queue whose server state evolves according to (1), and whose service time function $\mathcal{S}(x)$ is positive, continuous and convex.

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 ith task and let the queue length be zero at that instant. The server state upon the arrival of the $(i + 1)$th task is then obtained 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 ith 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 $x_{i+1} = (1 - (1 - x_i)e^{-\mathcal{S}(x_i)/\tau})e^{(\mathcal{S}(x_i) - 1/\lambda)/\tau} = \left(x_i - 1 + e^{\mathcal{S}(x_i)/\tau}\right)e^{-1/\lambda\tau}$. If λ and τ are such that $x_{i+1} = x_i$, then under the trivial control policy $u(t) \equiv \mathrm{ON}$, the server state at the beginning of all the tasks after and including the ith 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, the one-task equilibrium server states correspond to $x \in [0, 1]$ that satisfy $x = \left(x - 1 + e^{\mathcal{S}(x)/\tau}\right)e^{-1/\lambda\tau}$ and $\mathcal{S}(x) \leq 1/\lambda$, i.e., $\mathcal{S}(x) =$
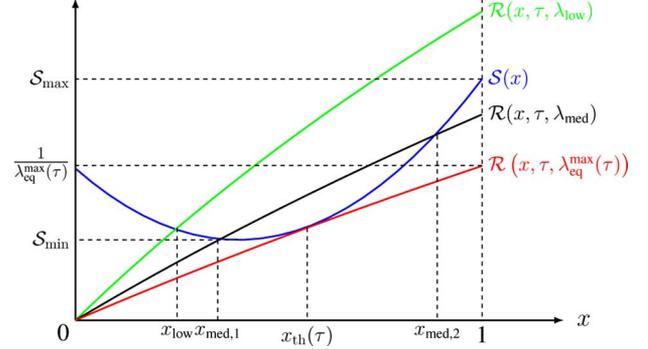
Fig. 1. Typical $\mathcal{S}(x)$ along with $\mathcal{R}(x, \tau, \lambda)$ for three values of λ: λ_{low}, λ_{med} and $\lambda_{\mathrm{eq}}^{\max}(\tau)$ in the increasing order. Also, $x_{\mathrm{eq}}(\tau, \lambda_{\mathrm{low}}) = \{x_{\mathrm{low}}\}$, $x_{\mathrm{eq}}(\tau, \lambda_{\mathrm{med}}) = \{x_{\mathrm{med},1}, x_{\mathrm{med},2}\}$ and $x_{\mathrm{eq}}(\tau, \lambda_{\mathrm{eq}}^{\max}(\tau)) = \{x_{\mathrm{th}}(\tau)\}$. Note that, since $x_{\mathrm{th}}(\tau) < 1$, $\lambda_{\mathrm{eq}}^{\max}(\tau)$ is the value of λ at which $\mathcal{R}(x, \tau, \lambda)$ is tangent to $\mathcal{S}(x)$.

$\tau \log \left(1 - (1 - e^{1/\lambda\tau})x\right)$ 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

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

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

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

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

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

Remark 3.2: Note that we did not include the constraint $\mathcal{S}(x) \leq 1/\lambda$ in the definition of $x_{\mathrm{eq}}(\tau, \lambda)$ in (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, $\mathcal{S}(x_{\mathrm{eq}}(\tau, \lambda)) = \mathcal{R}(x_{\mathrm{eq}}(\tau, \lambda), \tau, \lambda) \leq 1/\lambda$.

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

$$\lambda_{\mathrm{eq}}^{\max}(\tau) := \max \{\lambda > 0 \mid x_{\mathrm{eq}}(\tau, \lambda) \neq \emptyset\},$$
$$x_{\mathrm{th}}(\tau) := x_{\mathrm{eq}}\left(\tau, \lambda_{\mathrm{eq}}^{\max}(\tau)\right). \tag{4}$$

We now argue that the definitions in (4) 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 (3), is not-empty for all $\lambda < 1/\mathcal{S}_{\max}$. Moreover, since $\mathcal{R}(x, \tau, \lambda) \leq \mathcal{R}(1, \tau, \lambda) = 1/\lambda$ 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 (3), is empty. Hence, $\lambda_{\mathrm{eq}}^{\max}(\tau)$ and $x_{\mathrm{th}}(\tau)$ are well defined. In general, for a given $\tau > 0$ and $\lambda > 0$, $x_{\mathrm{eq}}(x, \tau)$ is not a singleton, e.g., see Fig. 1. However, due to the strict convexity of $\mathcal{S}(x) - \mathcal{R}(x, \tau, \lambda)$ in x as implied by Lemma 3.1, $x_{\mathrm{th}}(\tau)$ contains only one element. In the rest of the technical note, $x_{\mathrm{th}}(\tau)$ will denote this single element.

In the rest of the technical note, we will restrict our attention on those τ and $\mathcal{S}(x)$ for which $x_{\mathrm{th}}(\tau) < 1$. 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 Fig. 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. The implications of the case when $x_{\mathrm{th}}(\tau) = 1$ are discussed briefly at appropriate places.

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

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

Remark 3.4: The proof of Lemma 3.3 implies that, if $x_{\mathrm{th}}(\tau) < 1$ then $\partial/\partial x\left(\mathcal{S}(x) - \mathcal{R}\left(x, \tau, \lambda_{\mathrm{eq}}^{\max}(\tau)\right)\right) > 0$ for all $x \in (x_{\mathrm{th}}(\tau), 1]$ and hence $\mathcal{S}(x) - \mathcal{R}\left(x, \tau, \lambda_{\mathrm{eq}}^{\max}(\tau)\right) > 0$ for all $x \in (x_{\mathrm{th}}(\tau), 1]$. In particular, $\mathcal{S}(1) > \mathcal{R}\left(1, \tau, \lambda_{\mathrm{eq}}^{\max}(\tau)\right) = 1/\lambda_{\mathrm{eq}}^{\max}(\tau)$, i.e., $\lambda_{\mathrm{eq}}^{\max}(\tau)$ (which will be proven to be the maximum stabilizable arrival rate) is strictly greater than $1/\mathcal{S}(1)$, which is the rate at which the server is able to service tasks starting with the initial condition $x_0 = 1$ and servicing tasks continuously thereafter.

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 $x \in [0, 1]$. We first provide a result that relates the time for the one-task static problem to $\lambda_{\mathrm{eq}}^{\max}(\tau)$.

Lemma 3.5: For any $x \in [0, 1]$, $\tau > 0$ and $u \in \mathcal{U}$, we have that $T_f(x, \tau, 1, u) \geq 1/\lambda_{\mathrm{eq}}^{\max}(\tau)$.

Proof: First consider the policy $\tilde{u}$ that assigns the task to the server right away. In this case, $T_f(x, \tau, 1, \tilde{u})$ is the sum of $\mathcal{S}(x)$ and the idle time to allow the server state to return to x. From the definition of one-task equilibrium server states, it follows that $T_f(x, \tau, 1, \tilde{u})$ is such that

$$x \in x_{\mathrm{eq}}\left(\tau, \frac{1}{T_f(x, \tau, 1, \tilde{u})}\right). \tag{5}$$

In other words, $T_f(x, \tau, 1, \tilde{u})$ is the inverse of the arrival rate such that if the server starts at state x with zero queue length, then the server will be able to service a task and get back to state x exactly at the instant when the next task arrives. Now, consider a policy u_{x-} that waits for some initial time until the server state reaches state x^- before assigning the task to the server. By definition, $\tilde{u} = u_x$. In this case, also referring to Fig. 2, $T_f(x, \tau, 1, u_{x-})$ is the sum of initial idle time t^- for the server to reach state x^-, the service time $\mathcal{S}(x^-)$ and the idle time t^+ for the server state to return to x. Note that only those u_{x-} are feasible for which the server state after service time $\mathcal{S}(x^-)$ is not less than x. From the rearrangement argument illustrated in Fig. 2, it can be inferred that, for any such u_{x-}, $T_f(x, \tau, 1, u_{x-}) = T_f(x^-, \tau, 1, \tilde{u})$. Therefore, $\max_{x \in [0,1]} \sup_{\mathrm{feasible}\, u_{x-}} 1/T_f(x, \tau, 1, u_{x-}) = \max_{x-\in[0,1]} 1/T_f(x^-, \tau, 1, \tilde{u}) = \max_{x \in [0,1]} 1/T_f(x, \tau, 1, \tilde{u})$, i.e., it is sufficient to consider only the $\tilde{u}$ policy for the lemma. From (5), it follows that:

$$\max_{x \in [0,1]} \frac{1}{T_f(x, \tau, 1, \tilde{u})} = \max\left\{\lambda > 0 \,|\, \exists x \in [0, 1] \text{ s.t. } x \in x_{\mathrm{eq}}(\tau, \lambda)\right\}.$$

The result follows from (4). ∎

The following bound on $T_f(x, \tau, n, u)$ will be critical in proving a sharp upper bound on $\lambda_{\max}^*(\tau)$.

Lemma 3.6: For any $x \in [0, 1]$, $\tau > 0$, $n \in \mathbb{N}$ and $u \in \mathcal{U}$, we have that $T_f(x, \tau, n, u) \geq n/\lambda_{\mathrm{eq}}^{\max}(\tau)$.

Proof: For a given $x \in [0, 1]$, $\tau > 0$ and $u \in \mathcal{U}$, we prove the result by induction on n. The statement holds true for $n = 1$ by Lemma 3.5. Assume that the result holds true for some $n = k$, i.e.,
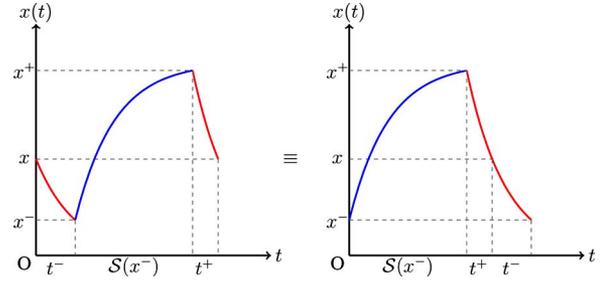

Fig. 2. Rearranging the time segments during the service cycle of a one-task static problem.

$T_f(x, \tau, k, u) \geq k/\lambda_{\mathrm{eq}}^{\max}(\tau)$. Given this, we shall prove that the statement holds true for $n = k + 1$.

Without any loss of generality, assume that u does not let the server idle before assigning the first task. This is because if u lets the server idle initially until it reaches a state, say $\tilde{x} < x$, then one can alternately consider a modified $(k + 1)$-task problem with initial and final server state $\tilde{x}$, and a modified control policy u_{mod} that does not idle the server before assigning the first task and under which the server states at the beginning of the service of tasks are the same as those under u. By following an argument similar to the one illustrated in Fig. 2, one can then see that the time for this modified $(k + 1)$-task problem, $T_f(\tilde{x}, \tau, k + 1, u_{mod})$, is the same as $T_f(x, \tau, k + 1, u)$.

For $i \in \{1, \ldots, k + 1\}$, let x_i and x_i' denote the server states at the beginning and at the end of service of task i respectively under the policy u. As argued before, we assume without loss of generality that u is such that $x_1 = x$, and hence

$$x_1' > x. \tag{6}$$

For $i \in \{1, \ldots, k + 1\}$, let $t_+(x_i, x_i') = \mathcal{S}(x_i)$ denote the time required for the server state to go from x_i to x_i' and for $i \in \{1, \ldots, k\}$, let $t_-(x_i', x_{i+1}) = \tau(\log x_i' - \log x_{i+1})$ denote the time required by the server to idle from x_i' to x_{i+1}. With these notations, one can write that

$$T_f(x, \tau, k + 1, u) = \sum_{i=1}^{k} t_+(x_i, x_i') + \sum_{i=1}^{k} t_-(x_i', x_{i+1}) + t_+(x_{k+1}, x_{k+1}') + t_-(x_{k+1}', x). \tag{7}$$

The rest of the proof is split among the following two cases. Case 1: $x_{k+1} \geq x$. We write $t_-(x_{k+1}', x)$ as

$$t_-(x_{k+1}', x) = \tau \log\left(\frac{x_{k+1}'}{x_{k+1}}\right) + \tau \log\left(\frac{x_{k+1}}{x}\right). \tag{8}$$

Therefore, from (7) and (8), we get that

$$\begin{aligned}
T_f(x, \tau, k + 1, u) &= \sum_{i=1}^{k} t_+(x_i, x_i') + \sum_{i=1}^{k-1} t_-(x_i', x_{i+1}) \\
&\quad + t_+(x_{k+1}, x_{k+1}') + \tau \log x_{k+1}' \\
&\quad - \tau \log x_{k+1} + t_-(x_k', x_{k+1}) \\
&\quad + \tau(\log x_{k+1} - \log x).
\end{aligned}$$

Since $t_-(x_k', x_{k+1}) + \tau(\log x_{k+1} - \log x) = \tau(\log x_k' - \log x_{k+1}) + \tau(\log x_{k+1} - \log x) = \tau(\log x_k' - \log x) = t_-(x_k', x)$, we can write that

$$\begin{aligned}
T_f(x, \tau, k + 1, u) &= \sum_{i=1}^{k} t_+(x_i, x_i') + \sum_{i=1}^{k-1} t_-(x_i', x_{i+1}) \\
&\quad + t_+(x_{k+1}, x_{k+1}') \\
&\quad + t_-(x_{k+1}', x_{k+1}) + t_-(x_k', x). \tag{9}
\end{aligned}$$

Now, consider the k-task static problem with initial and final server state x. Let u' denote the control policy under which the server states at the beginning of the tasks are $x_1, x_2, \ldots, x_k$. Also, $t_+(x_{k+1}, x'_{k+1}) + t_-(x'_{k+1}, x_{k+1}) = T_f(x_{k+1}, \tau, 1, \tilde{u})$, where $\tilde{u}$, as in the proof of Lemma 3.5, is the control policy for the one-task static problem that assigns the task to the server without any initial idling time. Combining these with (9), one gets that, $T_f(x, \tau, k+1, u) = T_f(x, \tau, k, u') + T_f(x_{k+1}, \tau, 1, \tilde{u})$. We have $T_f(x, \tau, k, u') \geq k/\lambda_{\text{eq}}^{\max}(\tau)$ by the induction argument and $T_f(x_{k+1}, \tau, 1, \tilde{u}) \geq 1/\lambda_{\text{eq}}^{\max}(\tau)$ by Lemma 3.5. Therefore, $T_f(x, \tau, k+1, u) \geq (k+1)/\lambda_{\text{eq}}^{\max}(\tau)$.

Case 2: $x_{k+1} < x$. Let

$$\tilde{i} := \max\left\{ i \in \{1, \ldots, k\} \mid x'_i > x \quad \& \quad x_{i+1} < x \right\}$$

. Since $x'_1 > x$ by (6), $\tilde{i}$ is well defined. Therefore

$$
T_f(x, \tau, k+1, u) = \sum_{i=1}^{\tilde{i}} t_+(x_i, x'_i) + \sum_{i=1}^{\tilde{i}-1} t_-(x'_i, x_{i+1})
$$
$$
+ t_-(x'_{\tilde{i}}, x_{\tilde{i}+1}) + \sum_{i=\tilde{i}+1}^{k+1} t_+(x_i, x'_i)
$$
$$
+ \sum_{i=\tilde{i}+1}^{k} t_-(x'_i, x_{i+1}) + t_-(x'_{k+1}, x). \quad (10)
$$

Splitting $t_-(x'_{\tilde{i}}, x_{\tilde{i}+1})$ as $\tau(\log x'_{\tilde{i}} - \log x) + \tau(\log x - \log x_{\tilde{i}+1})$, (10) can be written as

$$T_f(x, \tau, k+1, u) = T_f(x, \tau, \tilde{i}, u_1) + T_f(x, \tau, k+1-\tilde{i}, u_2) \quad (11)$$

where u_1 is the control policy for the $\tilde{i}$-task static problem with initial and final server state x, such that the server states at the beginning of the service of tasks are $x_1, \ldots, x_{\tilde{i}}$, and u_2 is the control policy for the $k+1-\tilde{i}$-task static problem with initial and final server state x such that the server states at the beginning of the service of the tasks are $x_{\tilde{i}+1}, \ldots, x_{k+1}$. Since both $\tilde{i}$ and $k+1-\tilde{i}$ are strictly less than $k+1$, we apply induction argument to both the terms on the right side of (11) to conclude that $T_f(x, \tau, k+1, u) \geq \tilde{i}/\lambda_{\text{eq}}^{\max}(\tau) + (k+1-\tilde{i})/\lambda_{\text{eq}}^{\max}(\tau) = k+1/\lambda_{\text{eq}}^{\max}(\tau)$. ∎

C. Upper Bound on Stabilizable Arrival Rate

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

Lemma 3.7: For any $\tau > 0$, $x_0 \in [0, 1]$, $n_0 \in \mathbb{N}$ and $\lambda > \lambda_{\text{eq}}^{\max}(\tau)$, if $x_{\text{th}}(\tau) < 1$ 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:

$$g(x) = \mathcal{S}_{\min} + \tau \log\left(\frac{x_{\min}}{x}\right). \quad (12)$$

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:

$$
x_{u_1} := \max\left\{ x \in [0, 1] \mid \mathcal{S}(x) = \frac{1}{\lambda_{\text{eq}}^{\max}}(\tau) \right\},
$$
$$
x_{u_2} := 1 - (1 - x_{l_1}) e^{-2/(\tau \lambda_{\text{eq}}^{\max}(\tau))},
$$
$$
x_{l_2} := x_{u_2} e^{-2/(\tau \lambda_{\text{eq}}^{\max}(\tau))},
$$
$$
x_L(\tau) := \min\{x_{l_1}, x_{l_2}\},
$$
$$
x_U(\tau) := \max\left\{ \frac{1 + x_{u_1}}{2}, x_{u_2} \right\} \quad (13)
$$

where we have dropped the dependency of x_{l_1}, $x_{l_{12}}$, x_{u_1} and x_{u_2} on τ for the sake of conciseness. We now infer relevant properties of the various quantities defined in (13). Remark 3.4 implies that, if $x_{\text{th}}(\tau) < 1$, then $\mathcal{S}(1) > 1/\lambda_{\text{eq}}^{\max}(\tau)$. This, combined with the fact that $\mathcal{S}_{\min} \leq 1/\lambda_{\text{eq}}^{\max}(\tau)$ and that $\mathcal{S}(x)$ is continuous, implies that x_{u_1} is well-defined and $x_{u_1} < 1$. From the definition of x_{u_2}, we have that $x_{u_2} < 1$ and also $x_{u_2} > x_{l_1}$ since $e^{-2/(\tau \lambda_{\text{eq}}^{\max}(\tau))} < 1$. This, combined with earlier conclusion after (12) that $x_{l_1} > 0$, implies that $x_{u_2} > 0$. Therefore, from the definition of x_{l_2}, we have that, $x_{l_2} > 0$ and $x_{l_2} < x_{u_2}$. Combining these facts with the definitions of $x_L(\tau)$ and $x_U(\tau)$, we have that, $0 < x_L(\tau) < x_U(\tau) < 1$.

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 (12). 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 (13), 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 (13). 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 (13). Therefore, the time spent between successive tasks is at least $2/\lambda_{\text{eq}}^{\max}(\tau)$.

∎

Theorem 3.8: 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 3.7 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 ith task. We have that $x_{i_k} \in [x_L, x_U]$ for all $k \geq 1$. Define constants c_1 and c_2 as follows:

$$c_1 := -\tau \log x_L, \quad c_2 := -\tau \log(1 - x_U). \tag{14}$$

Note that both c_1 and c_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 $T_f(x_{i_1}, \tau, i_k - i_1, u') = t_{i_k} - t_{i_1} + \tau(\log x_{i_k} - \log x_{i_1})$. Therefore, $t_{i_k} - t_{i_1} = T_f(x_{i_1}, \tau, i_k - i_1, u') - \tau(\log x_{i_k} - \log x_{i_1}) \geq T_f(x_{i_1}, \tau, i_k - i_1, u') + \tau \log x_{i_1} \geq T_f(x_{i_1}, \tau, i_k - i_1, u') - c_1$, where the last inequality follows from (14). 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_kth task, one has $x_{i_k+m} \geq x_{i_1}$. An upper bound on the time for the static problem under u'' is

$$
\begin{aligned}
T_f(x_{i_1}, \tau, m + i_k - i_1, u'') \leq\ & t_{i_k} - t_{i_1} + \tau \log(1 - x_{i_1}) \\
& - \tau \log(1 - x_{i_k}) \\
& + \mathcal{S}_{\max} - \tau \log x_{i_1}
\end{aligned} \tag{15}
$$

where $t_{i_k} - t_{i_1}$ is the time between the beginning of service of tasks i_1 and i_k, $\tau(\log(1 - x_{i_1}) - \log(1 - x_{i_k}))$ is the time required for the server state to increase from x_{i_k} to x_{i_1} under continuous usage, $\mathcal{S}_{\max}$ is the upper bound on the *overshoot* time that could happen if the server is in middle of servicing a task while crossing state x_{i_1} and $-\tau \log x_{i_1}$ is the upper bound on the time taken by the server to idle back to state x_{i_1} in case of the overshoot. Equation (15) can be rewritten as $t_{i_k} - t_{i_1} \geq T_f(x_{i_1}, \tau, m + i_k - i_1, u'') - \mathcal{S}_{\max} + \tau \log(1 - x_{i_k}) + \tau \log x_{i_1} \geq T_f(x_{i_1}, \tau, m + i_k - i_1, u'') - \mathcal{S}_{\max} - c_1 - c_2$, where the last inequality follows from (14).

Combining these bounds on $t_{i_k} - t_{i_1}$ with lemma 3.6, we have that, for all $k \geq 1$

$$t_{i_k} - t_{i_1} \geq \begin{cases} \frac{i_k - i_1}{\lambda_{\text{eq}}^{\max}(\tau)} - c_1 & \text{if } x_{i_k} \geq x_{i_1}, \\ \frac{m + i_k - i_1}{\lambda_{\text{eq}}^{\max}(\tau)} - \mathcal{S}_{\max} - c_1 - c_2 & \text{otherwise.} \end{cases} \tag{16}$$

With $c = c_1 + c_2 + \mathcal{S}_{\max}$, (16) becomes

$$t_{i_k} - t_{i_1} \geq \frac{i_k - i_1}{\lambda_{\text{eq}}^{\max}(\tau)} - c \quad \forall k \geq 1. \tag{17}$$

For $k \geq 1$, let n_k be the queue length at the beginning of service of task i_k. Then one can write $n_k \geq n_1 + \lambda(t_{i_k} - t_{i_1}) - (i_k - i_1) \ \forall k \geq 1$. This with (17) gives

$$n_k \geq n_1 - \lambda c + (i_k - i_1)\left(\frac{\lambda}{\lambda_{\text{eq}}^{\max}(\tau)} - 1\right) \quad \forall k \geq 1. \tag{18}$$

From (18), 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 3.9:

i) Theorem 3.8 implies that, for a given $\tau > 0$, if $x_{\text{th}}(\tau) < 1$ then $\lambda_{\max}^*(\tau) = \lambda_{\text{eq}}^{\max}(\tau)$.

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 rates 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 THE MAXIMUM STABILIZABLE ARRIVAL RATE

Consider the following threshold policy:

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

where $x_{\text{th}}(\tau)$ is defined in (4). 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 ith 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 $n(t_i) \leq n(t_1) + \lceil(\lambda - 1/\mathcal{S}_{\max})(-\tau \log(1 - x_{\text{th}}) + \mathcal{S}_{\max})\rceil + \lceil -\lambda\tau \log x_{\text{th}} \rceil$ 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, $n(t_1) = 0$.

- **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 $-\tau \log(1 - x_{\text{th}}) + \mathcal{S}_{\max}$, where $-\tau \log(1 - x_{\text{th}})$ is the time to go from $x = 0$ to $x = x_{\text{th}}$ when the server is continuously busy, and the $\mathcal{S}_{\max}$ term accounts for the fact that the server might be in middle of servicing a task when it reaches $x = x_{\text{th}}$ and hence its server state will exceed x_{th} before it finishes the task. The maximum amount of time spent in such an *overshoot* is upper bounded by $\mathcal{S}_{\max}$. Since $1/\mathcal{S}_{\max}$ is the minimum rate at which the server will be servicing the tasks during this time, the increase in the queue length during this time is upper bounded by $\lceil(\lambda - 1/\mathcal{S}_{\max})(-\tau \log(1 - x_{\text{th}}) + \mathcal{S}_{\max})\rceil$. Under the threshold policy, the possible overshoot above x_{th} will be 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}. The increase in queue length during this time is upper bounded by $\lceil -\lambda\tau \log x_{\text{th}} \rceil$. Therefore, the maximum queue length when the server state reaches x_{th} is upper bounded by $n_1 + \lceil(\lambda - 1/\mathcal{S}_{\max})(-\tau \log(1 - x_{\text{th}}) + \mathcal{S}_{\max})\rceil + \lceil -\lambda\tau \log x_{\text{th}} \rceil$. 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 3.8 and Theorem 4.1, one can deduce that, for any $\tau > 0$, $\lambda^*_{\max}(\tau) = \lambda^{\max}_{eq}(\tau)$, and the threshold policy is a maximally stabilizing task release control policy.

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

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 $\lambda^{\max}_{eq}(\tau) - \epsilon$.

V. Conclusion

In this technical note, 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 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 using, for example, the flexibility in choosing the threshold, as noted in part (ii) of Remark 4.2.

Acknowledgment

The authors wish to thank T. Temple for helpful discussions.

References

[1] K. Savla, C. Nehme, T. Temple, and E. Frazzoli, "Efficient routing of multiple vehicles for human-supervised services in a dynamic environment," in *Proc. AIAA Conf. Guid., Navig., Control*, Honolulu, HI, 2008, pp. 1–16.

[2] L. Kleinrock, *Queueing Systems I: Theory*. New York: Wiley-Interscience, 1975.

[3] S. Asmussen, *Applied Probability and Queues*. New York: Springer, 2003.

[4] J. H. Dshalalow, "Queueing Systems with State Dependent Parameters," in *Frontiers in Queuing Models and Applications in Science and Engineering*. Orlando, FL: CRC, 1997.

[5] R. M. Yerkes and J. D. Dodson, "The relation of strength of stimulus to rapidity of habit-formation," *J. Comparative Neur. Psychology*, vol. 18, pp. 459–482, 1908.

[6] R. Bekker and S. C. Borst, "Optimal admission control in queues with workload-dependent service rates," *Prob. Eng. Inform. Sci.*, vol. 20, pp. 543–570, 2006.

[7] M. L. Cummings and C. E. Nehme, "Modeling the impact of workload in network centric supervisory control settings," in *Proc. 2nd Annu. Sustaining Perform. Under Stress Symp.*, College Park, MD, Feb. 2009, [CD ROM].

[8] C. R. Glassey and M. G. C. Resende, "A scheduling rule for job release in semiconductor fabrication," *Oper. Res. Lett.*, vol. 7, no. 5, pp. 213–217, 1988.

[9] J. W. M. Bertrand and H. P. G. V. Ooijen, "Workload based order release and productivity: A missing link," *Prod. Planning Control*, vol. 13, no. 7, pp. 665–678, 2002.

[10] S. Stidham, "Optimal control of admission to queueing system," *IEEE Trans. Autom. Control*, vol. AC-30, no. 8, pp. 705–713, Aug. 1985.

[11] K. Savla and E. Frazzoli, Maximally Stabilizing Task Release Control Policy for a Dynamical Queue 2010 [Online]. Available: http://arxiv.org/abs/0909.3651

A General Stability Test for Switched Positive Systems Based on a Multidimensional System Analysis

Hugo Alonso and Paula Rocha

Abstract—The main goal of this technical note is to present a new sufficient condition for the stability of switched positive linear systems. This condition is general in two senses. First of all, because it applies to switching systems of order n composed of k subsystems, for any n and k. Secondly, because it applies to both discrete- and continuous-time systems. Furthermore, it is easy to verify, as it amounts to check for the stability of a certain matrix. The new condition is derived from a multidimensional system analysis, where the relation between the stability of multidimensional and switched systems is investigated. A comparison with other existing tests is also presented, showing that the one proposed here allows to infer about the stability of a system in cases where other tests fail or do not apply at all.

Index Terms—Multidimensional systems, positive linear systems, stability, switched systems.

I. Introduction

Switched systems are dynamical systems composed of several subsystems switching amongst themselves according to a decision rule. They are said to be positive if their state vector only takes nonnegative values. Switched positive systems are very common in practice and arise in important applications such as congestion control in communication networks [16] and drug delivery control in anaesthesia [12].

The stability of switched positive linear systems has been a subject of considerable research; see for instance [7], [9]–[11], [15]. The usual approach consists in finding conditions which guarantee the existence of a common quadratic Lyapunov function (CQLF), since this is sufficient to ensure the uniform asymptotic stability of the switched system [17]. It turns out that most of the stability results in the literature assume that a switched positive linear system has order $n \in \{2, 3\}$ or $k = 2$ subsystems. An exception is given in [15], where it is shown that a CQLF of diagonal form exists if the system matrices commute. However, there are obviously many examples where this sufficient condition does not hold. This motivated our search for alternative conditions applicable to any n and k. Note that several stability results exist for switched linear systems which are not necessarily positive and have an arbitrary order and number of subsystems. A recent survey can be found in [8], where for instance the relation between the stability of switched linear systems and of polytopic uncertain linear time-variant systems is described.

Here, we propose a general sufficient condition, easy to verify, for the stability of switched positive linear systems. This condition is general in two senses. First of all, because it applies to switching systems of order n composed of k subsystems, for any n and k. Secondly, because