Sequence-based anytime control

by Daniel E. Quevedo and Vijay Gupta

This is an author-prepared version of the article, reprinted from IEEE Transactions on Automatic Control Vol. 58, Issue 2, p. 377-390 (2013)

http://dx.doi.org/10.1109/TAC.2012.2209977

Sequence-based Anytime Control

Daniel E. Quevedo, *Member, IEEE,* and Vijay Gupta, *Member, IEEE*

Abstract

We present two related anytime algorithms for control of nonlinear systems when the processing resources available are time-varying. The basic idea is to calculate tentative control input sequences for as many time steps into the future as allowed by the available processing resources at every time step. This serves to compensate for the time steps when the processor is not available to perform any control calculations. Using a stochastic Lyapunov function based approach, we analyze the stability of the resulting closed loop system for the cases when the processor availability can be modeled as an independent and identically distributed sequence and via an underlying Markov chain. Numerical simulations indicate that the increase in performance due to the proposed algorithms can be significant.

I. Introduction

A lot of recent attention has focused on networked and embedded control (see, e.g., the special issue [1] and the references therein). One issue which plays an important role, especially in embedded systems, is that of time-varying and limited processing power. As more and more objects are equipped with micro-processors that are responsible for multiple functions such as control, communication, data fusion, system maintenance and so on, the implicit assumption traditionally made in control design about the processor being able to execute the desired control algorithm at any time will break down. Similarly, if a remote controller is in charge of many devices, multiple control tasks will compete for shared processor resources, leading to constrained availability of processing resources for the individual control loops. It is, thus, of interest to study control algorithms that can function despite limited and time-varying availability

A preliminary version of parts of this work was presented at the 49th IEEE Conference on Decision and Control, see [15].

D. Quevedo is with the School of Electrical Engineering & Computer Science, The University of Newcastle, Australia: dquevedo@ieee.org.

V. Gupta is with the Department of Electrical Engineering, University of Notre Dame: vgupta2@nd.edu.

of processing power. There is a growing number of works that deal with this issue. The impact of finite computational power has been looked at most closely for techniques such as model predictive control. McGovern and Feron [27], [28] presented bounds on computational time for achieving stability for specific optimization algorithms, if the processor has constant, but limited, computational resources. Henriksson et al [16], [17] studied the effect of not updating the control input in continuous time systems for the duration of the computational delay for optimization algorithms based on active set methods. Also related are works on event-triggered and self-triggered control systems, and online sampling, e.g., [8], [43], [45], [46], where a control input is calculated aperiodically, but on demand, depending on the plant state. In addition, we would like to mention work on scheduling of control tasks [6], [7], [41] that looks at the problem of processor queue scheduling, when control calculation is merely one of the tasks in the queue.

An alternative approach to achieve system robustness in the presence of time-varying processing resources is to develop *anytime* algorithms. The main purpose of anytime algorithms is to provide *a* solution even with limited processing resources, and to refine the solution as more resources become available. Anytime algorithms seek to make efficient use of resources and are, thus, popular in the context of real-time systems. In control, however, there are few methods available for developing anytime controllers. A notable work is that of Bhattacharya et al [4] who focused on linear systems, and presented a control algorithm that updated a different number of states depending on the available computational time. However, the available computational time was required to be known to the controller a priori. Another important work is that of Greco et al [11], who proposed switching among a pre-designed set of controllers that may require different execution times. Although the idea can be generalized to nonlinear processes, the analysis in the paper relied on Markovian jump linear system theory. In Gupta and Luo [12], an anytime algorithm for systems with multiple inputs was presented. The main idea was based on calculating the components of the control vector sequentially, and refining the process model as more processing time becomes available. Since the algorithm is based on identifying the modes of the process that require more urgent control, it is, thus, again largely limited to linear processes.

In the present work, we present two anytime control algorithms for nonlinear plants described in state-space form that are based on using extra processor availability to calculate sequences which have the potential to be implemented at the plant input at future times. This safeguards

performance at those time steps where the processor is entirely unavailable for control. Availability of processor time for control calculations determines the length of the sequences calculated and, thereby, affects the quality of the result. A distinguishing feature of the algorithms presented is that processor availability is allowed to be random, with unknown distribution. Moreover, our algorithms are one of the first that are suitable for nonlinear plants. For cases where processor availability is governed by a suitable Markov Chain, we use Lyapunov functions to establish sufficient conditions for stochastic stability of the closed loop. Numerical simulations illustrate that performance gains achieved with the algorithms proposed can be significant.

It is worth emphasizing that in the algorithms presented, the potential control values are calculated sequentially, reutilizing the already computed values for the next computation. This is computationally attractive, especially since the length of the sequence to be calculated is time-varying and not known a-priori. Thus, our approach differs significantly from the methods used in packetized predictive control, e.g., in [10], [29], [33]–[37], [44]. In the latter works calculation of control sequences requires solving optimization problems over a finite horizon of length determined by the controller itself.

The remainder of this manuscript is organized as follows: In Section II we formulate the anytime control design problem studied. Section III presents the proposed algorithms. Stochastic stability analysis is carried out in Sections IV to VII. Numerical simulations are documented in Section VIII. Section IX draws the final conclusions.

Notation: We write $\mathbb{N}$ for $\{1, 2, \ldots\}$ and $\mathbb{N}_0$ for $\mathbb{N} \cup \{0\}$. $\mathbb{R}$ represents the real numbers and $\mathbb{R}_{\geq 0} \triangleq [0, \infty)$. The $p \times p$ identity matrix is denoted via I_p, $0_{p\times q}$ is the $p \times q$ all-zeroes matrix, $0_p \triangleq 0_{p\times p}$, and $\mathbf{0}_p \triangleq 0_{p\times 1}$. The notation $\{x\}_{\mathcal{K}}$ stands for $\{x(k) : k \in \mathcal{K}\}$, where $\mathcal{K} \subseteq \mathbb{N}_0$. We adopt the conventions $\sum_{k=\ell_1}^{\ell_2} a_k = 0$ and $\prod_{k=\ell_1}^{\ell_2} a_k = 1$, if $\ell_1 > \ell_2$ and irrespective of $a_k \in \mathbb{R}$. The superscript ^{T} refers to transpose. The Euclidean norm of a vector x is denoted via $|x| = \sqrt{x^T x}$. A function $\varphi\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of *class-*$\mathscr{K}_\infty$ ($\varphi \in \mathscr{K}_\infty$), if it is continuous, zero at zero, strictly increasing, and unbounded. The probability of an event Ω is denoted by $\mathbf{Pr}\{\Omega\}$ and the conditional probability of Ω given Γ by $\mathbf{Pr}\{\Omega \,|\, \Gamma\}$. The expected value of a random variable ν given Γ is denoted by $\mathbf{E}\{\nu \,|\, \Gamma\}$ while $\mathbf{E}\{\nu\}$ refers to the unconditional expectation.

II. Problem Formulation

We consider nonlinear (and possibly unstable) plants sampled periodically with sampling interval $T_s > 0$ and described in discrete-time via:

$$x(k+1) = f(x(k), u(k), w(k)), \quad k \in \mathbb{N}_0. \tag{1}$$

In (1), $x \in \mathbb{R}^n$ is the plant state, $u \in \mathbb{R}^p$ is the plant input, and $w \in \mathbb{R}^m$ is an unmeasured disturbance. The model (1) satisfies $f(\mathbf{0}_n, \mathbf{0}_p, \mathbf{0}_m) = \mathbf{0}_n$ and the initial state, $x(0)$, is arbitrarily distributed (with possibly unbounded support).

Throughout this work, we will assume that the unperturbed plant model

$$x(k+1) = f(x(k), u(k), \mathbf{0}_m) \tag{2}$$

is globally stabilizable via state feedback. More precisely, we make the following assumption:

Assumption 1: There exist functions $\varphi_1, \varphi_2 \in \mathscr{K}_\infty$, $V\colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, $\kappa\colon \mathbb{R}^n \to \mathbb{R}^p$, and a constant $\rho \in [0, 1)$, such that for all $x \in \mathbb{R}^n$,

$$\begin{aligned} \varphi_1(|x|) \leq V(x) \leq \varphi_2(|x|) \\ V(f(x, \kappa(x), \mathbf{0}_m)) \leq \rho V(x). \end{aligned} \tag{3}$$

□

When implementing discrete-time control systems it is generally assumed that the processing resources available to the controller are such that the control law can always be evaluated within a fixed (and small) time-delay, say $\delta \in (0, T_s)$.[1] However, in practical networked and embedded systems, the processing resources (e.g., processor execution times) available for control calculations may vary and, at times, be insufficient to generate a control input within the prescribed time-delay δ. One possible remedy for this issue would to redesign the control system for a worst case by choosing larger values of δ and, possibly, T_s. Clearly, such an approach will, in general, lead to unnecessary conservativeness and associated poor performance. In the present work we adopt an anytime control paradigm to seek favorable trade-offs between processor availability and control performance.

[1]Recall that fixed delays can be easily incorporated into the model (1) by aggregating the previous plant input to the plant state, see also [32]. For ease of exposition, throughout this work, we will use the standard discrete-time notation as in (1).

Before proceeding we note that a direct implementation of the control policy used in Assumption 1, when processing resources are time varying, results in a *baseline* algorithm, which gives rise to the plant input:

$$u(k) = \begin{cases} \kappa(x(k)), & \text{if sufficient computational resources to evaluate} \\ & \kappa(x(k)) \text{ are available between } kT_s \text{ and } kT_s + \delta, \\ \mathbf{0}_p, & \text{otherwise,} \end{cases} \tag{4}$$

where the symbol $u(k)$ with $k \in \mathbb{N}_0$ denotes the plant input which is applied during the interval[2] $[kT_s+\delta, (k+1)T_s+\delta)$. Whilst the baseline algorithm (4) is intuitive and simple, it is by no means clear that it cannot be outperformed by more elaborated control formulations. In the following section, we will present two related anytime control algorithms for the plant model (1). The aim is to make more efficient use of the processing resources available for control, when compared to the baseline algorithm (4).

III. Anytime Control Through Calculation of Control Sequences

Throughout this work, we will assume that the controller needs processor time to carry out mathematical computations, such as evaluating functions. However, simple operations at a bit level, such as writing data into buffers, shifting buffer contents and setting values to zero, do not require processor time. Similarly, input-output operations, i.e., A/D and D/A conversion are triggered by external asynchronous loops with a real-time clock and do not require that the processor be available for control. As in regular discrete-time control, these external loops ensure that state measurements are available at the instants $\{kT_s\}_{k\in\mathbb{N}_0}$ and that the controller outputs are passed on to the plant actuators at times $\{kT_s + \delta\}_{k\in\mathbb{N}_0}$, where δ is fixed; see, e.g., [2]

A standing assumption is that if the processor were fully available for control, then calculating the desired plant input $u(k)$ for a given plant state $x(k)$ would be possible within the pre-allocated time-frame $t \in (kT_s, kT_s + \delta)$. Issues arise when, at times, processor availability does not permit the desired plant input to be calculated. To take care of the associated performance loss, in the present work we propose to use one of the two anytime control algorithms presented below.

[2]If sufficient computational resources are not available, then one could alternatively hold the previous control value and set $u(k) = u(k-1)$. The situation mirrors that encountered when the control input is affected by dropouts; see, e.g., [38].

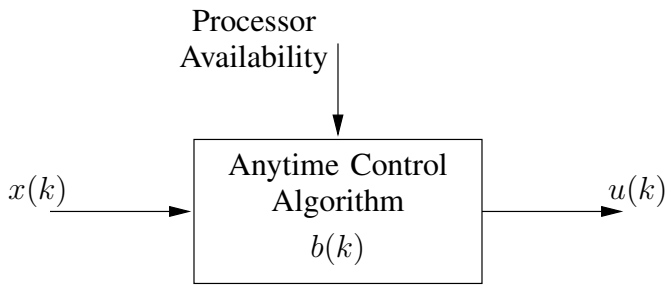

Fig. 1. Anytime control structure with internal buffer state $b(k)$.

A. Algorithm Descriptions

Both algorithms are based on the following basic idea: At time intervals when the controller is provided with more processing resources than are needed to evaluate the current control input, the algorithm calculates a sequence of tentative future plant inputs, say $\vec{u}(k)$. The sequence is stored in a local buffer and may be used when, at some future time steps, the processor availability precludes any control calculations, see Fig. 1.

For further reference, we denote the buffer states via $\{b\}_{\mathbb{N}_0}$, where

$$b(k) = \begin{bmatrix} b_1(k) \\ b_2(k) \\ \vdots \\ b_\Lambda(k) \end{bmatrix} \in \mathbb{R}^{\Lambda p}, \quad k \in \mathbb{N}_0 \tag{5}$$

for a given value $\Lambda \in \mathbb{N}$ and where each $b_j(k) \in \mathbb{R}^p$, $j \in \{1, \ldots, \Lambda\}$. We also introduce the shift matrix S and the unit vector e_1 via:

$$S \triangleq \begin{bmatrix} 0_p & I_p & 0_p & \ddots & 0_p \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0_p & \ldots & 0_p & I_p & 0_p \\ 0_p & \ldots\ldots & 0_p & I_p & \\ 0_p & \ldots\ldots\ldots & & 0_p & \end{bmatrix} \in \mathbb{R}^{\Lambda p \times \Lambda p}, \quad e_1 \triangleq \begin{bmatrix} I_p \\ 0_p \\ \vdots \\ 0_p \end{bmatrix} \in \mathbb{R}^{\Lambda p \times p}. \tag{6}$$

Algorithm A_1 is presented in Fig. 2. It can be seen that the algorithm proposed amounts to a dynamic state feedback policy with internal state variable $b(k)$. The latter provides

$$u(k) = e_1^T b(k) = b_1(k), \tag{8}$$

Step 1: At time $t = 0$,

SET $b(-1) \leftarrow \mathbf{0}_{\Lambda p}$, $k \leftarrow 0$

Step 2: IF $t \geq kT_s$, THEN

INPUT $x(k)$;

SET $\chi \leftarrow x(k)$, $j \leftarrow 1$, $b(k) \leftarrow Sb(k-1)$;

END

Step 3: WHILE "sufficient processor time is available" and $j \leq \Lambda$ and time $t < (k+1)T_s$,

SET $v \leftarrow V(\chi)$, where V is the Lyapunov function in (3);

Use v and χ to find $u_j(k)$, such that

$$V(f(\chi, u_j(k), \mathbf{0}_m)) \leq \rho v; \tag{7}$$

IF $j = 1$, THEN

OUTPUT $u_1(k)$;

SET $b(k) \leftarrow \mathbf{0}_{\Lambda p}$;

END

SET $b_j(k) \leftarrow u_j(k)$;

IF "sufficient processor time is not available" or $t \geq (k+1)T_s$, THEN

GOTO Step 5

END

SET $\chi \leftarrow f(\chi, u_j(k), \mathbf{0}_m)$, $j \leftarrow j + 1$;

END

Step 4: IF $j = 1$, THEN

OUTPUT $b_1(k)$;

END

Step 5: SET $k \leftarrow k + 1$ and GOTO Step 2;

Fig. 2. Algorithm A_1

and suggested plant inputs at future time steps. At the time steps when more processor time is available, a longer suggested trajectory of control inputs is calculated and stored in the buffer.[3] If the buffer runs out of tentative plant inputs (as calculated in Step 3), then the actuator values are set to zero. With Algorithm A_1, as soon as the processor calculates a control input $u_0(k)$, it

[3]Note that, by Assumption 1, in Step 3, one could simply set $u_j(k) \leftarrow \kappa(\chi)$.

Step 3: WHILE "sufficient processor time is available" and $j \leq \Lambda$ and time $t < (k+1)T_s$,
SET $v \leftarrow V(\chi)$, where V is the Lyapunov function in (3);
Use v and χ to find $u_j(k)$, such that $V(f(\chi, u_j(k), 0)) \leq \rho v$;
IF $j = 1$, THEN
OUTPUT $u_1(k)$;
END
SET $b_j(k) \leftarrow u_j(k)$;
IF "sufficient processor time is not available" or $t \geq (k+1)T_s$, THEN
GOTO Step 5;
END
SET $\chi \leftarrow f(\chi, u_j(k), \mathbf{0}_m)$, $j \leftarrow j + 1$;
END

Fig. 3. Step 3 of Algorithm A_2

throws away the remaining elements in the buffer, see line "$b(k) \leftarrow \mathbf{0}_{\Lambda p}$" in Step 3.

Algorithm A_2 is almost identical to the first algorithm, A_1. The only difference is that, in Step 3, the buffer contents are never re-set to zero, i.e., the line "$b(k) \leftarrow \mathbf{0}_{\Lambda p}$" is eliminated, see Fig. 3. Thus, if Algorithm A_2 is used, then buffer elements may stem from calculations carried out at different time instants. By not deleting the entire buffer, but only replacing the *appropriate* entries, when using A_2 the buffer will run out of data less often than when using A_1.

It is worth noting that neither algorithm requires prior knowledge of future processor availability for control. This opens the possibility to employ the algorithms in shared systems, where the controller task can be preempted by other computational tasks carried out by the processor, see also [5], [25], [48]. As in other anytime algorithms, there exists a compromise between resultant closed loop performance and the processor availability. Understanding this trade-off forms the bulk of this work.

B. Basic Properties

With the algorithms presented in Section III-A, extra processing time is used to calculate additional elements of the tentative plant input sequences, thus, providing higher quality results, i.e., sequences $\vec{u}(k)$ which better safeguard against performance loss at *future* time instances

where processor availability may be insufficient. To further elucidate the situation, we note that in both algorithms, during each iteration of the while-loop in Step 3, the state value $x(k)$ is used to calculate a tentative control, namely $u_j(k)$. In the sequel, we will denote by $N(k)$ the total number of iterations of the while-loop which are carried out during the interval $t \in (kT_s, (k+1)T_s)$ and note that $N(k) \in \{0, 1, \ldots, \Lambda\}$. Thus, if $N(k) \geq 1$, then the entire sequence of tentative controls is

$$\vec{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_{N(k)}(k) \end{bmatrix} \in \mathbb{R}^{N(k)\cdot p}. \tag{9}$$

If $N(k) = 0$, then the processor was not available for control, and (with either of the algorithms) the actuator values are taken as the first p elements of the shifted state $b(k) = Sb(k-1)$, see (6).

In terms of the notation introduced above and in (6), if Algorithm A_1 is used, then the buffer $b(k)$ obeys the recursion:

$$b(k) = \begin{cases} Sb(k-1), & \text{if } N(k) = 0, \\ \begin{bmatrix} \vec{u}(k) \\ \mathbf{0}_{(\Lambda - N(k))p} \end{bmatrix}, & \text{if } N(k) \geq 1. \end{cases} \tag{10}$$

On the other hand, if Algorithm A_2 is used, then we have:

$$b(k) = \begin{cases} Sb(k-1), & \text{if } N(k) = 0, \\ \begin{bmatrix} \vec{u}(k) \\ \mathbf{0}_{(\Lambda - N(k))p} \end{bmatrix} + M_{N(k)} b(k-1), & \text{if } N(k) \geq 1, \end{cases} \tag{11}$$

where

$$M_i \triangleq \big(I_{\Lambda p} - D_i\big) S, \tag{12}$$

with

$$D_i \triangleq \begin{cases} \mathrm{diag}(I_{ip}, 0_{(\Lambda - i)p}), & \text{if } i \in \{1, 2, \ldots, \Lambda - 1\} \\ I_{\Lambda p}, & \text{if } i = \Lambda. \end{cases} \tag{13}$$

In addition to studying the length of the tentative control sequences provided by the algorithms, namely $\{N\}_{\mathbb{N}_0}$, it is convenient to investigate how many values which stem from the tentative

control sequences $\{\vec{u}(k-\ell)\}$, $\ell \in \mathbb{N}_0$ are contained in the buffer state $b(k)$. We will refer to this value as the *effective buffer length* (at time k), and denote it as

$$\lambda(k) \in \{0, 1, \ldots, \Lambda\}, \quad k \in \mathbb{N}_0 \tag{14}$$

with $\lambda(-1) = 0$. It is easy to see that, if Algorithm A_1 is used, then $\{\lambda\}_{\mathbb{N}_0}$ is governed by

$$\lambda(k) = \begin{cases} N(k), & \text{if } N(k) \geq 1, \\ \max\{\lambda(k-1) - 1, 0\}, & \text{if } N(k) = 0, \end{cases} \tag{15}$$

whereas, with Algorithm A_2, we have

$$\lambda(k) = \max\{N(k), \lambda(k-1) - 1\}, \quad k \in \mathbb{N}_0. \tag{16}$$

The following example illustrates the quantities introduced above:

Example 1: Suppose that $\Lambda = 5$ and that the processor availability for control is such that $N(0) = 5$, $N(1) = 0$, $N(2) = 1$, $N(3) = 0$. If Algorithm A_1 is used, then the buffer state at times $k \in \{0, 1, 2, 3\}$ becomes:

$$b(0) = \begin{bmatrix} u_1(0) \\ u_2(0) \\ u_3(0) \\ u_4(0) \\ u_5(0) \end{bmatrix}, \; b(1) = \begin{bmatrix} u_2(0) \\ u_3(0) \\ u_4(0) \\ u_5(0) \\ \mathbf{0}_p \end{bmatrix}, \; b(2) = \begin{bmatrix} u_1(2) \\ \mathbf{0}_p \\ \mathbf{0}_p \\ \mathbf{0}_p \\ \mathbf{0}_p \end{bmatrix}, \; b(3) = \begin{bmatrix} \mathbf{0}_p \\ \mathbf{0}_p \\ \mathbf{0}_p \\ \mathbf{0}_p \\ \mathbf{0}_p \end{bmatrix},$$

which gives $\lambda(0) = 5$, $\lambda(1) = 4$, $\lambda(2) = 1$, $\lambda(3) = 0$, and plant inputs $u(0) = u_1(0)$, $u(1) = u_2(0)$, $u(2) = u_1(2)$, $u(3) = \mathbf{0}_p$. On the other hand, if Algorithm A_2 is used, then we have

$$b(0) = \begin{bmatrix} u_1(0) \\ u_2(0) \\ u_3(0) \\ u_4(0) \\ u_5(0) \end{bmatrix}, \; b(1) = \begin{bmatrix} u_2(0) \\ u_3(0) \\ u_4(0) \\ u_5(0) \\ \mathbf{0}_p \end{bmatrix}, \; b(2) = \begin{bmatrix} u_1(2) \\ u_4(0) \\ u_5(0) \\ \mathbf{0}_p \\ \mathbf{0}_p \end{bmatrix}, \; b(3) = \begin{bmatrix} u_4(0) \\ u_5(0) \\ \mathbf{0}_p \\ \mathbf{0}_p \\ \mathbf{0}_p \end{bmatrix}$$

and $\lambda(0) = 5$, $\lambda(1) = 4$, $\lambda(2) = 3$, $\lambda(3) = 2$, $u(0) = u_1(0)$, $u(1) = u_2(0)$, $u(2) = u_1(2)$, $u(3) = u_4(0)$. Note that with Algorithm A_1, $\lambda(3) = 0$ and therefore the plant input at time $k = 3$ is set to zero; with Algorithm A_2, the value calculated at time $k = 0$ is used. $\square$

IV. Stochastic Stability of Anytime Control Algorithms

Since the processor availability for control calculations is random, the plant input is random, and thus the system (1) evolves stochastically. Various stability notions for stochastic systems have been studied in the literature (e.g., [20], [22]). In the present work, we are interested in the following notion:

Definition 1 (Stochastic Stability): A dynamical system with state trajectory $\{x\}_{\mathbb{N}_0}$ is said to be stochastically stable, if

$$\sum_{k=0}^{\infty} \mathbf{E}\big\{\varphi(|x(k)|)\big\} < \infty, \tag{17}$$

for some $\varphi \in \mathscr{K}_\infty$. □

Remark 1: It follows directly from (17), that stochastic stability implies that there exists $\varphi \in \mathscr{K}_\infty$, such that:

$$\lim_{k\to\infty} \mathbf{E}\big\{\varphi(|x(k)|)\big\} = 0. \tag{18}$$

In the particular case where $\varphi(s) = s^2$, (17) reduces to $\sum_{k=0}^{\infty} \mathbf{E}\{|x(k)|^2\} < \infty$, and (18) to $\lim_{k\to\infty} \mathbf{E}\{|x(k)|^2\} = 0$; see also [9], [20]. □

A. Assumptions

Our subsequent stability analysis considers the unperturbed system (2), i.e., where $w(k) = 0$, for all $k \in \mathbb{N}_0$. For pedagogical ease, we also begin by presenting the analysis with the additional assumption that the processor availability for control is independent and identically distributed (i.i.d.). Thus, for the analysis in Sections V and VI, we make the following assumption:

Assumption 2: The process $\{N\}_{\mathbb{N}_0}$ introduced in Section III-B is i.i.d., with probability distribution

$$\mathbf{Pr}\{N(k) = l\} = p_l, \tag{19}$$

where $l \in \{0, 1, 2, \ldots, \Lambda\}$ and with $p_0 \in [0, 1)$. □

In Section VII, we will show how to extend this analysis for the case when the processor availability can be described by a Markov chain, and thus has memory.

Assumption 3 stated below, bounds the rate of increase of the Lyapunov function V in (3), when the nominal system (2) is run in open-loop. It also imposes a (mild) restriction on the distribution of the initial plant state.

Assumption 3: There exists $\alpha \in [1, 1/p_0)$ such that

$$V(f(x, \mathbf{0}_p, \mathbf{0}_m)) \leq \alpha V(x), \quad \forall x \in \mathbb{R}^n. \tag{20}$$

The initial plant state satisfies

$$\mathbf{E}\big\{\varphi_2(|x(0)|)\big\} < \infty, \tag{21}$$

where $\varphi_2 \in \mathscr{K}_\infty$ is as in (3). □

It is worth emphasizig that the fact that Assumptions 1 and 3 are global and stated in terms of a common Lyapunov function limits the class of plants and control policies considered in our subsequent analysis. One case where (20) is satisfied is when V and f are globally Lipschitz continuous, more precisely, when there exist $\varphi_V, \varphi_f \in \mathbb{R}_{\geq 0}$ such that:

$$|V(x) - V(z)| \leq \varphi_V |x - z|$$

$$|f(x, u, w) - f(z, u, w)| \leq \varphi_f |x - z|.$$

In this case, and since $f(\mathbf{0}_n, \mathbf{0}_p, \mathbf{0}_m) = \mathbf{0}_n$, we have

$$V(f(x, \mathbf{0}_p, \mathbf{0}_m)) = |V(f(x, \mathbf{0}_p, \mathbf{0}_m)) - V(f(\mathbf{0}_n, \mathbf{0}_p, \mathbf{0}_m))| \leq \varphi_V |f(x, \mathbf{0}_p, \mathbf{0}_m)|$$

$$\leq \varphi_V \varphi_f |x| \leq \varphi_V \varphi_f \varphi_1^{-1} V(x) = \alpha V(x),$$

for $\alpha = (\varphi_V \varphi_f)/\varphi_1$ and (20) will hold provided $p_0 < \varphi_1/(\varphi_V \varphi_f)$.

Example 2: Consider an open-loop unstable constrained plant model of the form (1) with

$$f(x, u, w) = \begin{bmatrix} x_2 + u_1 \\ -\mathrm{sat}(x_1 + x_2) + u_2 \end{bmatrix} + \begin{bmatrix} \sqrt{w^2 + 5} - \sqrt{5} \\ 0 \end{bmatrix},$$

with

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \; u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \mathrm{sat}(\mu) = \begin{cases} -1, & \text{if } \mu < -1, \\ \nu & \text{if } \mu \in [-1, 1], \\ 1, & \text{if } \mu > 1, \end{cases}$$

see [21, Example 2.3] and [34]. The second component of the plant input is constrained via $|u_2(k)| \leq 0.8$, $\forall k \in \mathbb{N}_0$. If we choose $V(x) = 2|x|$ and policy $\kappa(x) = \begin{bmatrix} -x_2 & 0.8\mathrm{sat}(x_1 + x_2) \end{bmatrix}^T$, then direct calculations provide that

$$V\big(f(x, \kappa(x), \mathbf{0}_m)\big) = 0.4|\mathrm{sat}(x_1 + x_2)| \leq 0.4|x_1 + x_2|$$

$$\leq 0.8 \max\{|x_1|, |x_2|\} - \max\{|x_1|, |x_2|\} + |x| \leq |x|.$$

Thus, Assumption 1 holds with $\rho = 1/2$, and $\varphi_1(s) = \varphi_2(s) = 2s$. Furthermore, by proceeding as in [21, p.73], it can be shown that (20) holds with $\alpha = 1.618$. Thus, provided that (21) holds and $p_0 < 0.618$, Assumption 3 is also satisfied. □

The following example illustrates that, at times, it may be convenient to first find a Lyapunov function V which satisfies (20) and then seek a control policy which ensures that Assumption 1 holds.

Example 3: Consider a scalar unconstrained and unperturbed open-loop unstable non-linear plant where $f(x, u, w) = x^2 + u$, with $x, u \in \mathbb{R}$. A stabilizing control policy which satisfies Assumption 1 for $V_1(x) = |x|^2$ is given by $\kappa_1(x) = -x^2 + \rho x$, with $\rho \in [0, 1)$. However, $V_1(x) = |x|^2$ is not suitable for use in Assumption 3, since $V_1(f(x, 0, 0))/V_1(x) = x^2 \to \infty$ as $x \to \infty$.

In contrast, if we choose $V_2 \in \mathscr{K}_\infty$ as $V_2(x) = \ln(|x| + 1)$, for all $x \in \mathbb{R}$, then

$$V_2(f(x, 0, 0)) = \ln(x^2 + 1) \leq \ln(x^2 + 2|x| + 1) = 2\ln(|x| + 1) = 2V_2(x), \quad \forall x \in \mathbb{R}$$

and (20) holds with open-loop rate of growth bound constant $\alpha = 2$. The associated control policy $\kappa_2(x) = -x^2 + \exp(\rho V_2(x)) - 1$, where $\rho \in [0, 1)$, gives

$$V_2(f(x, \kappa_2(x), 0)) = \ln(|\exp(\rho V_2(x)) - 1| + 1) = \ln(\exp(\rho V_2(x))) = \rho V_2(x).$$

We conclude that if $p_0 < 1/2$ and the initial plant state is suitably distributed, then Assumptions 1 and 3 will hold. □

B. Stochastic Stability with the Baseline Algorithm

We will next present sufficient conditions under which the baseline algorithm (4) achieves stochastic stability of the closed loop system. As in (19), we denote via p_0 the probability that the controller is unable to calculate any control input. Thus, if the baseline algorithm (4) is used and Assumption 2 holds, then (in the disturbance-free case) the closed loop is characterised by:

$$\mathbf{Pr}\big\{x(k+1) = \chi^+ \,\big|\, x(k) = \chi\big\} = \begin{cases} p_0, & \text{if } \chi^+ = f(\chi, \mathbf{0}_p, \mathbf{0}_m), \\ 1 - p_0, & \text{if } \chi^+ = f(\chi, \kappa(\chi), \mathbf{0}_m). \end{cases} \tag{22}$$

It can be seen that the plant state trajectory is similar to that of a networked control system in which the controller is unable to communicate with the actuator with probability p_0 at any time

step. Stability conditions for such systems have been derived both for linear systems [13], [18] and nonlinear systems [34]. In particular, for a scalar linear plant model with a scalar input,

$$f(x,u,w) = ax + b_u u + b_w w, \quad (a, b_u, b_w) \in \mathbb{R}^3,$$

and quadratic Lyapunov function, $V(x) = x^2$, the condition $p_0|a|^2 < 1$ has been shown to be necessary and sufficient for stabilizability in [13]. Thus, the constant α needs to satisfy $\alpha \in [1, 1/p_0)$ for stability with the baseline algorithm. More generally, we have the following sufficient condition for stochastic stability when the baseline algorithm is used:

Theorem 1: Suppose that Assumptions 1 to 3 hold. If

$$p_0\alpha + (1-p_0)\rho < 1, \tag{23}$$

then (22) is stochastically stable.

Proof: First we note that, by (22), the process $\{x\}_{\mathbb{N}_0}$ is Markovian. Thus, stability can be examined by using a stochastic Lyapunov function approach; see, e.g., [22]. The law of total expectation, when applied to $\mathbf{E}\{V(x(1)) \,|\, x(0)\}$, with V as in (3), gives

$$\begin{aligned} \mathbf{E}\big\{V(x(1)) \,\big|\, x(0) = \chi\big\} &= p_0 V(f(\chi, \mathbf{0}_p, \mathbf{0}_m)) + (1-p_0) V(f(\chi, \kappa(\chi), \mathbf{0}_m)) \\ &\le p_0\alpha V(\chi) + (1-p_0)\rho V(\chi) < V(\chi), \quad \forall \chi \in \mathbb{R}^n, \end{aligned} \tag{24}$$

where we have used (3), (20) and (23). Theorem 2 of [22, Chapter 8.4.2] implies that there exists $c < \infty$ such that $\sum_{k=0}^{\infty} \mathbf{E}\big\{V(x(k)) \,\big|\, x(0) = \chi\big\} \le cV(\chi)$. Thus, by using (3) and taking expectation with respect to the distribution of $x(0)$, we obtain

$$\sum_{k=0}^{\infty} \mathbf{E}\big\{\varphi_1(|x(k)|)\big\} = \mathbf{E}\bigg\{ \sum_{k=0}^{\infty} \mathbf{E}\big\{\varphi_1(|x(k)|) \,\big|\, x(0)\big\} \bigg\} \le \mathbf{E}\{cV(x(0))\} \le c\mathbf{E}\{\varphi_2(|x(0)|)\} < \infty,$$

where the last inequality follows from (21). Since $\varphi_1 \in \mathscr{K}_\infty$, stochastic stability follows. ■

For the proposed anytime algorithms, stability analysis is more subtle than for the baseline algorithm. The main reason is that, due to buffering, the plant state $\{x\}_{\mathbb{N}_0}$ will in general not be Markovian and simple conditioning as in (24) is not possible.[4]

[4]Note that some of the results included in Section IV of [15] are incorrect.

V. Stability with Algorithm A_1

To derive sufficient conditions for stochastic stability when Algorithm A_1 is used, we will employ a technique which is roughly based on the approaches used in [23], [34]–[36], [47]. As will become apparent, randomness of the sequence length process $\{N\}_{\mathbb{N}_0}$, see (9), makes the analysis of the anytime algorithms studied significantly more involved than the analysis of the predictive networked control formulations of [34]–[36].

A. Plant model at times $k \in \mathcal{K}$

For ease of exposition, in the sequel we assume that $N(0) > 0$ and denote the time steps at which at least one control input is calculated via $\mathcal{K} = \{k_i\}_{i\in\mathbb{N}_0}$, where $k_0 = 0$ and

$$k_{i+1} = \inf\big\{k \in \mathbb{N}\colon k > k_i,\ N(k) > 0\big\}, \quad i \in \mathbb{N}_0. \tag{25}$$

It is convenient to introduce the iterated mappings with input $x(k_i)$, $k_i \in \mathcal{K}$:[5]

$$f^j(x(k_i)) \triangleq \begin{cases} x(k_i), & \text{if } j = 0, \\ f(f^{j-1}(x(k_i)), u_{j-1}(k_i), \mathbf{0}_m), & \text{if } j \in \{1, \dots, N(k_i)\} \end{cases} \tag{26}$$

and the related mappings which describe the nominal plant model when the input is set to zero:

$$f_{\mathrm{OL}}^j(x) \triangleq \begin{cases} x, & \text{if } j = 0, \\ f(f_{\mathrm{OL}}^{j-1}(x), \mathbf{0}_p, \mathbf{0}_m), & \text{if } j \geq 1. \end{cases} \tag{27}$$

We also denote the time between two consecutive elements of $\mathcal{K}$ via

$$\Delta_i \triangleq k_{i+1} - k_i, \quad \forall (k_{i+1}, k_i) \in \mathcal{K} \times \mathcal{K} \tag{28}$$

and note that, by Assumption 2, the process $\{\Delta_i\}_{i\in\mathbb{N}_0}$ is i.i.d. with geometric distribution

$$\mathbf{Pr}\{\Delta_i = j\} = (1 - p_0)p_0^{j-1}, \qquad j \in \mathbb{N}, \tag{29}$$

see [47]. In terms of the quantities introduced above, the state of the nominal plant (2) when Algorithm A_1 is used satisfies:

$$x(k_i + \ell) = \begin{cases} f^\ell(x(k_i)), & \text{if } \ell \in \big\{0, 1, \dots, \min(N(k_i), \Delta_i)\big\}, \\ f_{\mathrm{OL}}^{\ell - N(k_i)}\big(f^{N(k_i)}(x(k_i))\big), & \text{if } N(k_i) < \Delta_i \text{ and } \ell \in \{N(k_i) + 1, \dots, \Delta_i\}, \end{cases} \tag{30}$$

[5]For example, we have $f^1(x(k_i)) = f(x(k_i), u_0(k_i), \mathbf{0}_m)$ and $f^2(x(k_i)) = f(f(x(k_i), u_0(k_i), \mathbf{0}_m), u_1(k_i), \mathbf{0}_m)$. Note that, by Step 3 in the algorithm description, the values $\{u_j(k_i)\}$, $j \in \{0, 1, \dots, N(k_i) - 1\}$ are determined by $x(k_i)$.

for all $k_i \in \mathcal{K}$. It is worth emphasizing that (30) describes the plant state trajectory $\{x(k)\}$ for all $k \in \mathbb{N}_0$.

By setting $\ell = \Delta_i$ in (30), we obtain that the state in (2) when Algorithm A_1 is employed can be described *at the instants* $k_i \in \mathcal{K}$ via:

$$\mathbf{Pr}\big\{x(k_{i+1}) = \chi^+ \,\big|\, x(k_i) = \chi\big\} = \begin{cases} \mathbf{Pr}\{\Delta_i \le N(k_i)\}, & \text{if } \chi^+ = f^{\Delta_i}(\chi), \\ 1 - \mathbf{Pr}\{\Delta_i \le N(k_i)\}, & \text{if } \chi^+ = f_{\mathrm{OL}}^{\Delta_i - N(k_i)}\big(f^{N(k_i)}(\chi)\big), \end{cases} \tag{31}$$

where $\Delta_i \in \mathbb{N}$.

It is worth noting that in (31), the number of possible values for $x(k_{i+1})$ given $x(k_i)$ is countably infinite, whereas if the baseline algorithm is used, there are only two possibilities, see (22). The terms $\mathbf{Pr}\{\Delta_i \le N(k_i)\}$ can be easily evaluated as per the following lemma:

Lemma 1: Suppose that Assumption 2 holds, then

$$\mathbf{Pr}\{\Delta_i \le N(k_i)\} = \frac{1}{1-p_0}\sum_{l=1}^{\Lambda} p_l(1-p_0^l), \quad \forall k_i \in \mathcal{K}. \tag{32}$$

Proof: By (28), the random variables Δ_i and $N(k_i)$ are independent. Furthermore, the two processes $\{\Delta_i\}_{i\in\mathbb{N}_0}$ and $\{N\}_{\mathbb{N}_0}$ are i.i.d. Thus, we can condition upon $N(k_i) \ge 1$ to obtain:

$$\begin{aligned} \mathbf{Pr}\{\Delta_i \le N(k_i)\} &= \mathbf{Pr}\{\Delta_i \le N(k) \,|\, k \in \mathcal{K}\} \\ &= \sum_{l=0}^{\Lambda} \mathbf{Pr}\{N(k) = l \,|\, k \in \mathcal{K}\} \cdot \mathbf{Pr}\{\Delta_i \le N(k) \,|\, N(k) = l, k \in \mathcal{K}\} \\ &= \frac{1}{1-p_0}\sum_{l=1}^{\Lambda} p_l \cdot \mathbf{Pr}\{\Delta_i \le l\} = \sum_{l=1}^{\Lambda} p_l \sum_{j=1}^{l} p_0^{j-1}. \end{aligned}$$

■

B. Main Results

As a consequence of (31) and (32), and since $u(k_i)$ is determined by $x(k_i)$, if Algorithm A_1 is used, then the plant state $\{x(k_i)\}$, with $k_i \in \mathcal{K}$, is Markovian. Stability of the closed loop can be analyzed by using a stochastic Lyapunov function approach which, to some extent, parallels that used to prove Theorem 1. To state our result, we first give the following lemma:

Lemma 2: Consider (31) and suppose that Assumptions 2 and 3 hold. Then

$$\mathbf{E}\big\{V(x(k_{i+1})) \,\big|\, x(k_i) = \chi\big\} \le \left(\sum_{l=1}^{\Lambda} p_l \Omega_l\right) V(\chi), \quad \forall \chi \in \mathbb{R}^n,\ \forall k_i, k_{i+1} \in \mathcal{K}, \tag{33}$$

where

$$\Omega_l \triangleq \rho\frac{1-(p_0\rho)^l}{1-p_0\rho}+\alpha\frac{(p_0\rho)^l}{1-p_0\alpha}\in\mathbb{R}_{\geq 0},\quad l\in\{1,2,\ldots,\Lambda\}. \tag{34}$$

Proof: We use the total probability formula twice. First, we condition on the length of the tentative control sequence calculated during $t\in(k_iT_s,(k_i+1)T_s)$:

$$\begin{aligned}\mathbf{E}\{V(x(k_{i+1}))\,|\,x(k_i)\}&=\mathbf{E}\big\{\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i)\big\}\big\}\\&=\sum_{l=1}^{\Lambda}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i)=l\big\}\cdot\mathbf{Pr}\{N(k)=l\,|\,k\in\mathcal{K}\}\\&=\sum_{l=1}^{\Lambda}\frac{p_l}{1-p_0}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i)=l\big\}.\end{aligned} \tag{35}$$

We note that in Algorithm A_1 previously calculated control values are erased at the instant k_i and, thus, (31) holds. Consequently, the conditional expectation $\mathbf{E}\{V(x(k_{i+1}))\,|\,x(k_i),N(k_i)\}$ can be evaluated by conditioning further on Δ_i:

$$\begin{aligned}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i)\big\}&=\mathbf{E}\big\{\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i),\Delta_i\big\}\big\}\\&=\sum_{j=1}^{\infty}(1-p_0)p_0^{j-1}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i),\Delta_i=j\big\},\end{aligned} \tag{36}$$

where we have used (29). Now, using Assumption 3 and Equation (7), we obtain the bound:

$$\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i)=\chi,N(k_i)=l,\Delta_i=j\big\}\leq\begin{cases}\rho^jV(\chi), & \text{if } j\leq l,\\ \alpha^{j-l}\rho^lV(\chi), & \text{if } j>l.\end{cases}$$

Thus, (36) gives:

$$\begin{aligned}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i)=\chi,N(k_i)=l\big\}&\leq(1-p_0)\left(\sum_{j=1}^{l}p_0^{j-1}\rho^j+\sum_{j=l+1}^{\infty}p_0^{j-1}\alpha^{j-l}\rho^l\right)V(\chi)\\&=(1-p_0)\Omega_lV(\chi),\end{aligned}$$

since, by Assumption 3, we have $p_0\alpha<1$. Substitution into (35) establishes (33). ■

Despite the fact that Lemma 2 considers only the time instants $k\in\mathbb{N}_0$ where $N(k)>0$, see (25), the bound in (33) can be used to conclude about stochastic stability (for all $k\in\mathbb{N}_0$).

Theorem 2: Suppose that Assumptions 1–3 hold and define

$$\sigma\triangleq\frac{1}{1-p_0\rho}\left(\rho(1-p_0\alpha)+\frac{\alpha-\rho}{1-p_0}\sum_{l=1}^{\Lambda}p_l(p_0\rho)^l\right)\in\mathbb{R}_{\geq 0}. \tag{37}$$

If

$$p_0\alpha + (1-p_0)\sigma < 1, \tag{38}$$

then the system (30) (with state trajectory $\{x\}_{\mathbb{N}_0}$) is stochastically stable.

Proof: By Lemma 2 and since $\{x\}_{\mathcal{K}}$ is Markovian, we have that if

$$\Omega \triangleq \sum_{l=1}^{\Lambda} p_l \Omega_l < 1, \tag{39}$$

where Ω_l are defined in (34), then

$$\mathbf{E}\big\{V(x(k_{i+1})) \,\big|\, x(k_i), x(k_{i-1}), \ldots, x(k_0)\big\} \leq \Omega V(x(k_i)).$$

Since, by Assumption 1, $V\colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, we conclude that V is a stochastic Lyapunov function for (31); c.f., [22], [26].

Direct calculations yield that

$$\begin{aligned} \Omega = \sum_{l=1}^{\Lambda} p_l \Omega_l &= \frac{\rho}{1-p_0\rho} \sum_{l=1}^{\Lambda} p_l + \sum_{l=1}^{\Lambda} p_l \frac{(p_0\rho)^l(\alpha-\rho)}{(1-p_0\rho)(1-p_0\alpha)} \\ &= \frac{\rho(1-p_0)}{1-p_0\rho} + \frac{(\alpha-\rho)}{(1-p_0\rho)(1-p_0\alpha)} \sum_{l=1}^{\Lambda} p_l (p_0\rho)^l. \end{aligned} \tag{40}$$

Using equations (40) and (37), we obtain

$$\Omega = \frac{\rho(1-p_0)}{1-p_0\rho} + \frac{(1-p_0)\big(\sigma(1-p_0\rho) - \rho(1-p_0\alpha)\big)}{(1-p_0\rho)(1-p_0\alpha)} = \frac{(1-p_0)\big(\sigma - p_0\rho\sigma\big)}{(1-p_0\rho)(1-p_0\alpha)} = \frac{(1-p_0)\sigma}{(1-p_0\alpha)}.$$

Hence, (38) is equivalent to (39). As a consequence, if (38) holds, then [22, Chapter 8.4.2, Theorem 2] implies exponential stability at the instants $k_i \in \mathcal{K}$, i.e., we have:

$$\mathbf{E}\{V(x(k_i)) \,|\, x(k_0) = \chi_0\} \leq \Omega^i V(\chi_0), \qquad \forall i \in \mathbb{N}, \quad \forall \chi_0 \in \mathbb{R}^n \tag{41}$$

Now for the time instants $k \in \mathbb{N} \setminus \mathcal{K}$, i.e., at those time steps when no control input is calculated, we proceed as in the proof of Lemma 2, to obtain that

$$\begin{aligned} \mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i) \right\} &= \sum_{l=1}^{\Lambda} \frac{p_l}{1-p_0} \mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i), N(k_i) = l \right\} \\ &= \sum_{l=1}^{\Lambda} p_l \sum_{j=1}^{\infty} p_0^{j-1} \mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i), N(k_i) = l, \Delta_i = j \right\}. \end{aligned}$$

Since $\rho < 1 < \alpha$, we can bound

$$\mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i), N(k_i) = l, \Delta_i = j \right\} \leq \sum_{k=0}^{j-1} \alpha^k \mathbf{E}\big\{ V(x(k_i)) \,|\, x(k_i) = \chi \big\} = \frac{\alpha^j - 1}{\alpha - 1} V(\chi)$$

so that

$$\mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i) = \chi, N(k_i) \right\} \leq \frac{(1-p_0)}{\alpha - 1} \sum_{j=1}^{\infty} (\alpha^j - 1) p_0^{j-1} V(\chi) = \frac{1}{1 - p_0 \alpha} V(\chi),$$

in turn yielding

$$\mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i) = \chi \right\} \leq \sum_{l=1}^{\Lambda} \frac{p_l}{(1-p_0)(1-p_0\alpha)} V(\chi) = \beta V(\chi), \tag{42}$$

where $\beta \triangleq 1/(1-p_0\alpha) \in \mathbb{R}_{\geq 0}$. The expectation on the left hand side of (42) is taken with respect to the distributions of $N(k_i)$ and Δ_i. Since $\{x\}_{\mathcal{K}}$ is Markovian and $N(k_i)$ and Δ_i are independent, we can take conditional expectation $\mathbf{E}\{\,\cdot \,|\, x(k_0)\}$ on both sides of (42) to obtain:

$$\begin{aligned}
&\mathbf{E}\left\{ \mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i) \right\} \,\middle|\, x(k_0) = \chi_0 \right\} \leq \beta\, \mathbf{E}\big\{ V(x(k_i)) \,|\, x(k_0) = \chi_0 \big\} \\
\Rightarrow\ &\mathbf{E}\left\{ \mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_i), x(k_{i-1}), \ldots, x(k_0) \right\} \,\middle|\, x(k_0) = \chi_0 \right\} \leq \beta\, \Omega^i V(\chi_0) \\
\Rightarrow\ &\mathbf{E}\left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \,\middle|\, x(k_0) = \chi_0 \right\} \leq \beta\, \mathbf{E}\big\{ V(x(k_i)) \,|\, x(k_0) = \chi_0 \big\} \leq \beta\, \Omega^i V(\chi_0),
\end{aligned}$$

where we have used the bound in (41). Since we assume that $k_0 = 0$, this gives

$$\mathbf{E}\left\{ \sum_{k=0}^{k_{j+1}-1} V(x(k)) \,\middle|\, x(0) = \chi_0 \right\} \leq \beta \sum_{i=0}^{j} \Omega^i V(\chi_0) = \beta \frac{1 - \Omega^{j+1}}{1 - \Omega} V(\chi_0) \leq \frac{\beta}{1 - \Omega} V(\chi_0).$$

Thus, by letting $k_{j+1} \to \infty$, it follows that there exists $c < \infty$ such that

$$\sum_{k=0}^{\infty} \mathbf{E}\big\{ V(k) \,\big|\, x(0) = \chi_0 \big\} \leq c V(\chi_0).$$

The remainder of the proof now follows as in the proof of Theorem 1. ■

Theorem 2 establishes sufficient conditions for stochastic stability of the closed loop when processor availability is i.i.d. and Algorithm A_1 is used. The quantity introduced in (37) involves the distribution of $\{N\}_{\mathbb{N}_0}$, the contraction factor of the baseline controller κ, see (3), and the bound on the rate of increase of the plant state when left in open loop, see (20).

As a particular case, suppose that the distribution of $\{N\}_{\mathbb{N}_0}$ satisfies $p_1 = 1 - p_0$, i.e., the processor time availability is such that the Algorithm A_1 provides at most one control input. In this case, expression (37) gives that $\sigma = \rho$ and, not surprisingly, we recover the sufficient condition for stochastic stability established for the baseline algorithm (4) in (23).

More generally, if the probability that Algorithm A_1 provides more than one control value is non-zero, then Theorem 2 establishes stochastic stability for a larger class of plant models than Theorem 1. This observation follows upon noting that σ can be rewritten as:

$$\sigma = \rho - \frac{(\alpha - \rho)}{(1 - p_0)(1 - p_0\rho)} \sum_{l=1}^{\Lambda} p_l \big(\rho p_0 - (\rho p_0)^l\big).$$

Thus, if $p_{l^\star} > 0$ for some $l^\star \in \{2, 3, \ldots, \Lambda\}$, then $\sum_{l=1}^{\Lambda} p_l \big(\rho p_0 - (\rho p_0)^l\big) > 0$ and $\sigma < \rho$. This suggests that Algorithm A_1 has better stabilizing properties than the baseline algorithm.

VI. Stability with Algorithm A_2

We first note that for $\Lambda \in \{1, 2\}$, Algorithm A_2 is equivalent to Algorithm A_1. Henceforth, we focus on cases where $\Lambda > 2$. It follows directly from (11) and (16) that with Algorithm A_2 if $\Delta_i > N(k_i)$ and $\lambda(k_i - 1) > N(k_i) + 1$, then $\lambda(k_i) = \lambda(k_i - 1) - 1 > N(k_i)$ and the plant input at times $\{k_i + N(k_i), k_i + N(k_i) + 1, \ldots, k_i + \min(\lambda(k_i), \Delta_i) - 1\}$ will stem from buffer contents at time $k_i - 1$, see also Example 1. Thus, with Algorithm A_2, $\{x\}_{\mathcal{K}}$ and $\{x\}_{\mathbb{N}_0}$ are not Markovian and the analysis carried out for Algorithm A_1 does not carry over directly.

To recover a Markovian structure, consider the overall system state $\{\theta\}_{\mathbb{N}_0}$ defined via:

$$\theta(k) \triangleq \begin{bmatrix} x(k) \\ b(k-1) \end{bmatrix}. \tag{43}$$

In terms of $\theta(k)$, (11) gives that at all times where $N(k) = 0$, the plant input is given by

$$u(k) = \begin{bmatrix} 0_{p\times(n+p)} & I_p & 0_{p\times(\Lambda-2)p} \end{bmatrix} \theta(k), \quad \Lambda > 2. \tag{44}$$

Furthermore, $\{\theta\}_{\mathbb{N}_0}$ and thereby also $\{\theta\}_{\mathcal{K}}$ are Markovian processes. The mapping[6]

$$f_{\mathrm{B}}^j(\theta) \triangleq \begin{cases} \theta, & \text{if } j = 0, \\ \begin{bmatrix} f\Big(\begin{bmatrix} I_n & 0_{n\times\Lambda p} \end{bmatrix} f_{\mathrm{B}}^{j-1}(\theta), \begin{bmatrix} 0_{p\times(n+p)} & I_p & 0_{p\times(\Lambda-2)p} \end{bmatrix} f_{\mathrm{B}}^{j-1}(\theta), \mathbf{0}_m\Big) \\ S^j \begin{bmatrix} 0_{\Lambda p\times n} & I_{\Lambda p} \end{bmatrix} \theta \end{bmatrix}, & \text{if } j \geq 1 \end{cases} \tag{45}$$

[6]For example, for $j = 1$ we have $f_{\mathrm{B}}^1(\theta(k)) = \begin{bmatrix} f(x(k), b_2(k-1), \mathbf{0}_m) \\ Sb(k-1) \end{bmatrix}$, see also (5).

allows one to characterize the nominal system behaviour at times where computational resources are insufficient to calculate control inputs, so that buffered plant inputs are used. More precisely, the nominal plant state when Algorithm A_2 is used can be stated in terms of a random mapping with inputs $\{\theta\}_{\mathcal{K}}$ as follows:

$$x(k_i+\ell) = \begin{cases} f^{\ell}(x(k_i)), & \text{if } \ell \in \big\{0,1,\ldots,\min(N(k_i),\Delta_i)\big\} \\ \Big[I_n \quad 0_{n\times\Lambda p}\Big] f_{\mathrm{B}}^{\ell-N(k_i)}(\theta'), & \text{if } \Delta_i > N(k_i) \text{ and } \lambda(k_i) > N(k_i) \\ & \qquad \text{and } \ell \in \{N(k_i)+1,\ldots,\min(\lambda(k_i),\Delta_i)\} \\ f_{\mathrm{OL}}^{\ell-\lambda(k_i)}(x'), & \text{if } \Delta_i > \lambda(k_i) \text{ and } \ell \in \{\lambda(k_i)+1,\ldots,\Delta_i\}, \end{cases} \tag{46}$$

where f^{ℓ} and f_{OL}^{j} are defined in (26) and (27), respectively, $\lambda(k_i) = \max\{N(k_i), \lambda(k_i-1)-1\}$, and with

$$\begin{aligned} \theta' &\triangleq \begin{bmatrix} f^{N(k_i)}(x(k_i)) \\ S^{N(k_i)} b(k_i-1) \end{bmatrix} \\ x' &\triangleq \Big[I_n \quad 0_{n\times\Lambda p}\Big]\, f_{\mathrm{B}}^{\lambda(k_i)-N(k_i)}(\theta'). \end{aligned} \tag{47}$$

At the instants $k_i \in \mathcal{K}$, the nominal plant state in (2) when Algorithm A_2 is used can thus be described via:

$$\begin{aligned} &\mathbf{Pr}\big\{x(k_{i+1}) = \chi^+ \,\big|\, x(k_i) = \chi, b(k_i-1) = \upsilon\big\} \\ &\qquad = \begin{cases} \mathbf{Pr}\{\Delta_i \le N(k_i)\}, & \text{if } \chi^+ = f^{\Delta_i}(\chi), \\ \mathbf{Pr}\{N(k_i) < \Delta_i \le \lambda(k_i)\}, & \text{if } \chi^+ = \Big[I_n \quad 0_{n\times\Lambda p}\Big] f_{\mathrm{B}}^{\Delta_i - N(k_i)}(\vartheta), \\ \mathbf{Pr}\{\Delta_i > \lambda(k_i)\}, & \text{if } \chi^+ = f_{\mathrm{OL}}^{\Delta_i - N(k_i)}(\xi), \end{cases} \end{aligned} \tag{48}$$

where

$$\vartheta = \begin{bmatrix} f^{N(k_i)}(\chi) \\ S^{N(k_i)}\upsilon \end{bmatrix}, \quad \xi = \Big[I_n \quad 0_{n\times\Lambda p}\Big]\, f_{\mathrm{B}}^{\lambda(k_i)-N(k_i)}(\vartheta). \tag{49}$$

Note that, as shown in Lemma 1, the probabilities $\mathbf{Pr}\{\Delta_i \le N(k_i)\}$ used in (48) are i.i.d. Nevertheless, it is easy to see that

$$\begin{aligned} \mathbf{Pr}\{N(k_i) < \Delta_i \le \lambda(k_i)\} &= \frac{1}{1-p_0}\sum_{l=1}^{\Lambda} p_l \cdot \mathbf{Pr}\big\{l < \Delta_i \le \max\{l, \lambda(k_i-1)-1\}\big\} \\ &= \sum_{l=1}^{\Lambda} p_l \sum_{j=l+1}^{\max\{l,\lambda(k_i-1)-1\}} p_0^{j-1} = \frac{1}{1-p_0}\sum_{l=1}^{\Lambda} p_l\big(p_0^{l} - p_0^{\max\{l,\lambda(k_i-1)-1\}}\big), \end{aligned}$$

expression which depends upon $\lambda(k_i-1)$ and therefore on $b(k_i-1)$.

The following stochastic stability result is akin to the one developed in Section V-B for Algorithm A_1. It shows that the sufficient condition developed for Algorithm A_1 is also sufficient to guarantee stochastic stability when Algorithm A_2 is used.

Theorem 3: Suppose that Assumptions 1 to 3 hold and that Algorithm A_2 is used. If (38) is satisfied, then the closed loop system (with state trajectory $\{x\}_{\mathbb{N}_0}$) is stochastically stable. □

Proof: It follows from (48), (45) and by proceeding as in the proof of Lemma 2 that

$$\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,\theta(k_i)\big\}=\sum_{l=1}^{\Lambda}p_l\sum_{j=1}^{\infty}p_0^{j-1}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,\theta(k_i),N(k_i)=l,\Delta_i=j\big\}. \tag{50}$$

On the other hand, since $\lambda(k_i)$ is a function of $N(k_i)$ and $b(k_i-1)$, we have

$$\begin{aligned}
&\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i)=\chi,b(k_i-1)=\upsilon,N(k_i)=l,\Delta_i=j\big\}\\
&\qquad=\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i)=\chi,b(k_i-1)=\upsilon,N(k_i)=l,\Delta_i=j,\lambda(k_i)=\lambda\big\}\\
&\qquad\le\begin{cases}\rho^jV(\chi), & \text{if } j\le\lambda,\\ \alpha^{j-\lambda}\rho^\lambda V(\chi), & \text{if } j>\lambda,\end{cases}
\end{aligned}$$

where we have used the bounds in (7), (20) and where $\lambda=\max\{l,\lambda_0-1\}$ with λ_0 denoting the index of the last nonzero entry in υ, see (16). Substitution into (50) yields that

$$\begin{aligned}
\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i)=\chi,b(k_i-1)=\upsilon\big\}&\le\sum_{l=1}^{\Lambda}p_l\bigg(\sum_{j=1}^{\lambda}p_0^{j-1}\rho^j+\sum_{j=\lambda+1}^{\infty}p_0^{j-1}\alpha^{j-\lambda}\rho^\lambda\bigg)V(\chi)\\
&\le\sum_{l=1}^{\Lambda}p_l\bigg(\sum_{j=1}^{l}p_0^{j-1}\rho^j+\sum_{j=l+1}^{\infty}p_0^{j-1}\alpha^{j-l}\rho^l\bigg)V(\chi)=\Omega V(\chi),\quad\forall(\chi,\upsilon)\in\mathbb{R}^n\times\mathbb{R}^{\Lambda p}
\end{aligned}$$

where Ω is defined in (39) and where we have used the fact that $\rho<1<\alpha$ and $\lambda\ge l$.

Since $\{\theta\}_{\mathcal{K}}$ is Markovian, it follows from [22, Chapter 8.4.2, Theorem 2] that

$$\mathbf{E}\big\{V(x(k_i))\,\big|\,x(k_0)=\chi_0,b(k_0-1)=\upsilon_0\big\}\le\Omega^iV(\chi_0),\quad\forall(i,\chi_0,\upsilon_0)\in\mathbb{N}\times\mathbb{R}^n\times\mathbb{R}^{\Lambda p}.$$

The remainder of the proof now follows, *mutatis mutandis*, that of Theorem 2. ■

VII. Markovian Processor State Model

So far we have assumed that the process $\{N\}_{\mathbb{N}_0}$ is i.i.d. In situations where the control loop is shared with other applications having time-varying and correlated processing demands it is likely that Assumption 2 will not be satisfied. We will next outline how the analysis presented can be

extended to encompass cases where the processor availability for control, henceforth modeled via the processor state process $\{g\}_{\mathbb{N}_0}$, is correlated.

Assumption 4: The processor state process $\{g\}_{\mathbb{N}_0}$ is an irreducible aperiodic finite Markov Chain (see, e.g., [24]) with values in the finite set $\{1, 2, \ldots, G\}$, $G \in \mathbb{N}$. Its transition matrix Q is given by

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1G} \\ q_{21} & q_{22} & & q_{2G} \\ \vdots & \vdots & \ddots & \vdots \\ q_{G1} & q_{G2} & \cdots & q_{GG} \end{bmatrix}, \tag{51}$$

where $q_{ij} = \mathbf{Pr}\{g(k+1) = j \,|\, g(k) = i\}$, $\forall i, j \in \{1, 2, \ldots, G\}$. Given any processor state $g(k) = \varsigma$, the conditional distribution of the process $\{N\}_{\mathbb{N}_0}$ satisfies

$$\mathbf{Pr}\{N(k) = l \,|\, g(k) = \varsigma\} = p_{l|\varsigma}, \quad \forall (l, \varsigma) \in \{0, 1, \ldots, \Lambda\} \times \{1, 2, \ldots, G\}, \tag{52}$$

with given probabilities $p_{l|\varsigma}$. □

For the baseline algorithm in (4) stochastic stability can be ensured as follows:

Theorem 4: Suppose that Assumptions 1, 3 and 4 hold. Define

$$\hat{p}_0 = \max_{\varsigma \in \{1,2,\ldots,G\}} p_{0|\varsigma}. \tag{53}$$

A sufficient condition for (22) to be stochastically stable is that

$$\hat{p}_0 \alpha + (1 - \hat{p}_0)\rho < 1. \tag{54}$$

Proof: First, we note that the joint process $\{(x, g)\}_{\mathbb{N}_0}$ is Markovian. Thus, by using the law of total expectation and the fact that $\alpha - \rho > 0$, we obtain

$$\mathbf{E}\big\{V(x(1)) \,\big|\, x(0) = \chi, g(0) = \varsigma\big\} \leq (p_{0|\varsigma}\alpha + (1 - p_{0|\varsigma})\rho)V(\chi) \leq (\hat{p}_0\alpha + (1 - \hat{p}_0)\rho)V(\chi), \tag{55}$$

for all $(\chi, \varsigma) \in \mathbb{R}^n \times \{1, 2, \ldots, G\}$. The remainder of the proof now follows as in the proof of Theorem 1 and is omitted for space constraints. ■

The stability results of Sections V and VI can be extended to encompass the Markovian processor model of Assumption 4. Here we only present the stability results for Algorithm A_1. The main difference from the analysis in Section V is the fact that the plant state $\{x\}_{\mathcal{K}}$ is no longer Markovian. Interestingly, the analysis can be extended by recognizing that the aggregated process $\{(x, g)\}_{\mathcal{K}}$ is Markovian.

Whilst the process $\{\Delta_i\}_{i\in\mathbb{N}_0}$ is no longer i.i.d., the conditional distributions $\mathbf{Pr}\{\Delta_i \,|\, g(k_i)\}$ can be evaluated as per the following result:

Lemma 3: Suppose that Assumption 4 holds and define

$$\mathbb{G} \triangleq \big\{\varsigma \in \{1,2,\ldots,G\} \colon p_{0|\varsigma} < 1\big\}.$$

Then[7]

$$\mathbf{Pr}\{\Delta_i = j \,|\, g(k_i) = \varsigma\} = \bar{q}_\varsigma \bar{Q}^{j-1}\bar{p}, \quad \forall (j,\varsigma) \in \mathbb{N}\times\mathbb{G}, \tag{56}$$

where

$$\bar{q}_\varsigma \triangleq \begin{bmatrix} q_{\varsigma 1} & \ldots & q_{\varsigma G}\end{bmatrix}, \quad \bar{Q} \triangleq \mathrm{diag}(p_{0|1},\ldots,p_{0|G})Q, \quad \bar{p} \triangleq \begin{bmatrix} 1-p_{0|1} & \ldots & 1-p_{0|G}\end{bmatrix}^T. \tag{57}$$

Proof: Denote

$$\vec{g} = \begin{bmatrix} g(k_i+1) & g(k_i+2) & \ldots & g(k_i+j)\end{bmatrix}, \quad \vec{\varsigma} = \begin{bmatrix} \varsigma_1 & \varsigma_2 & \ldots & \varsigma_j\end{bmatrix}$$

and $\mathcal{G} = \mathbb{G}^j$. Conditioning upon the processor state sequence $\vec{g} \in \mathcal{G}$ gives that

$$\begin{aligned}
\mathbf{Pr}\{\Delta_i = j \,|\, g(k_i) = \varsigma\} &= \sum_{\vec{\varsigma}\in\mathcal{G}} \mathbf{Pr}\big\{\Delta_i = j \,|\, g(k_i) = \varsigma, \vec{g} = \vec{\varsigma}\big\}\, \mathbf{Pr}\{\vec{g} = \vec{\varsigma} \,|\, g(k_i) = \varsigma\} \\
&= \sum_{\vec{\varsigma}\in\mathcal{G}} p_{0|\varsigma_1} p_{0|\varsigma_2} \cdots p_{0|\varsigma_{j-1}} (1-p_{0|\varsigma_j}) q_{\varsigma\varsigma_1} q_{\varsigma_1\varsigma_2} \cdots q_{\varsigma_{j-1}\varsigma_j} \\
&= \sum_{\vec{\varsigma}\in\mathcal{G}} (q_{\varsigma\varsigma_1} p_{0|\varsigma_1})(q_{\varsigma_1\varsigma_2} p_{0|\varsigma_2}) \cdots (q_{\varsigma_{j-2}\varsigma_{j-1}} p_{0|\varsigma_{j-1}})(q_{\varsigma_{j-1}\varsigma_j}(1-p_{0|\varsigma_j})),
\end{aligned}$$

which can be rewritten in compact form as in (56). ■

The state evolution at times $k_i \in \mathcal{K}$ can now be evaluated as

$$\mathbf{Pr}\big\{x(k_{i+1}) = \chi^+ \,\big|\, x(k_i) = \chi, g(k_i) = \varsigma\big\} = \begin{cases} \mathbf{Pr}\{\Delta_i \le N(k_i) \,|\, g(k_i) = \varsigma\}, \\ \qquad \text{if } \chi^+ = f^{\Delta_i}(\chi), \\ 1 - \mathbf{Pr}\{\Delta_i \le N(k_i) \,|\, g(k_i) = \varsigma\}, \\ \qquad \text{if } \chi^+ = f_{\mathrm{OL}}^{\Delta_i - N(k_i)}\big(f^{N(k_i)}(\chi)\big), \end{cases} \tag{58}$$

where $\Delta_i \in \mathbb{N}$ and $\varsigma \in \mathbb{G}$. Lemma 2 can be generalized as follows:

[7]Note that in the i.i.d. case of Assumption 2, we have $G = Q = 1$, $\mathbb{G} = \{1\}$, $\bar{Q} = p_0$ and $\bar{p} = 1 - p_0$, so that (56) reduces to (29).

Lemma 4: Consider (58) and suppose that Assumption 3 with p_0 replaced by $\hat{p}_0$, and Assumption 4 hold. Then

$$\mathbf{E}\{V(x(k_{i+1}))\,|\,x(k_i)=\chi, g(k_i)=\varsigma\} \le \Upsilon_\varsigma V(\chi)$$

for all $(\chi,\varsigma)\in\mathbb{R}^n\times\mathbb{G}$, where

$$\Upsilon_\varsigma \triangleq \bar{q}_\varsigma\big(I-\rho\bar{Q}\big)^{-1}\bigg(\rho I+\frac{(\alpha-\rho)}{1-p_{0|\varsigma}}\big(I-\alpha\bar{Q}\big)^{-1}\sum_{l=1}^{\Lambda}p_{l|\varsigma}(\rho\bar{Q})^l\bigg)\bar{p}, \tag{59}$$

with $\bar{q}_\varsigma$, $\bar{Q}$, and $\bar{p}$ as in (57).

Proof: Following as in the proof of Lemma 2, we first condition upon $N(k_i)$ to calculate, for $\varsigma\in\mathbb{G}$,

$$\mathbf{E}\big\{V(x(k_{i+1}))\,|\,x(k_i),g(k_i)=\varsigma\big\}=\sum_{l=1}^{\Lambda}\frac{p_{l|\varsigma}}{1-p_{0|\varsigma}}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i)=l,g(k_i)=\varsigma\big\} \tag{60}$$

and then condition further on Δ_i to obtain that

$$\begin{aligned}&\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i),g(k_i)=\varsigma\big\}\\&\qquad=\sum_{j=1}^{\infty}\bar{q}_\varsigma\bar{Q}^{j-1}\bar{p}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i),N(k_i),\Delta_i=j,g(k_i)=\varsigma\big\},\end{aligned}$$

where we have used (56). Equation (7) and Assumption 3 then provide the bound:

$$\begin{aligned}\mathbf{E}\big\{V(x(k_{i+1}))\,\big|\,x(k_i)=\chi,N(k_i),g(k_i)=\varsigma\big\}&\le\sum_{j=1}^{l}\bar{q}_\varsigma\bar{Q}^{j-1}\bar{p}\rho^jV(\chi)+\sum_{j=l+1}^{\infty}\bar{q}_\varsigma\bar{Q}^{j-1}\bar{p}\alpha^{j-l}\rho^lV(\chi)\\&=\bar{q}_\varsigma\bigg(\sum_{j=1}^{l}\bar{Q}^{j-1}\rho^j+\rho^l\sum_{j=l+1}^{\infty}\bar{Q}^{j-1}\alpha^{j-l}\bigg)\bar{p}V(\chi),\end{aligned} \tag{61}$$

Since Q is the transition probability of an irreducible aperiodic Markov Chain and $\rho\hat{p}_0\le\alpha\hat{p}_0<1$, the above summation is convergent. If we now substitute (61) into (60), then we obtain the bound

$$\begin{aligned}\mathbf{E}\{V(x(k_{i+1}))\,|\,x(k_i)=\chi,g(k_i)=\varsigma\}&\le\sum_{l=1}^{\Lambda}\frac{p_{l|\varsigma}\bar{q}_\varsigma}{1-p_{0|\varsigma}}\bigg(\sum_{j=1}^{l}\bar{Q}^{j-1}\rho^j+\rho^l\sum_{j=l+1}^{\infty}\bar{Q}^{j-1}\alpha^{j-l}\bigg)\bar{p}V(\chi)\\&=\frac{\bar{q}_\varsigma}{1-p_{0|\varsigma}}\sum_{l=1}^{\Lambda}p_{l|\varsigma}\Big(\rho\big(I-(\rho\bar{Q})^l\big)\big(I-\rho\bar{Q}\big)^{-1}+\alpha(\rho\bar{Q})^l\big(I-\alpha\bar{Q}\big)^{-1}\Big)\bar{p}V(\chi)\\&=\frac{\bar{q}_\varsigma}{1-p_{0|\varsigma}}\sum_{l=1}^{\Lambda}p_{l|\varsigma}\Big(\rho\big(I-\rho\bar{Q}\big)^{-1}+(\rho\bar{Q})^l\big(\alpha\big(I-\alpha\bar{Q}\big)^{-1}-\rho\big(I-\rho\bar{Q}\big)^{-1}\big)\Big)\bar{p}V(\chi)\end{aligned}$$

where we have used [3, Prop. 9.4.13]. The result now follows upon noting that $\alpha\big(I-\alpha\bar{Q}\big)^{-1}-\rho\big(I-\rho\bar{Q}\big)^{-1}=\big(I-\rho\bar{Q}\big)^{-1}\big(\alpha\big(I-\rho\bar{Q}\big)-\rho\big(I-\alpha\bar{Q}\big)\big)\big(I-\alpha\bar{Q}\big)^{-1}=(\alpha-\rho)\big(I-\rho\bar{Q}\big)^{-1}\big(I-\alpha\bar{Q}\big)^{-1}$ and some algebraic manipulations. ■

Following as in the proof of Theorem 2, one can derive the following stochastic stability result:

Theorem 5: Suppose that Assumption 1 and the hypotheses of Lemma 4 hold. If

$$\Upsilon_{\varsigma}<1,\quad \forall \varsigma\in\mathbb{G},$$

then the system (46) (with state trajectory $\{x\}_{\mathbb{N}_0}$) is stochastically stable.

The above generalizes the analysis in Section V to situations where the processor availability for control is correlated. The results of Section VI can be similarly extended.

Remark 2: It is easy to see that the i.i.d. model of Assumption 2 corresponds to the special case of the Markovian model in Assumption 4, obtained by setting $G=1$, $\mathbb{G}=\{1\}$, $Q=q_{11}=1$ and $p_l=p_{l|1}$, for all $l\in\{0,1,\ldots,\Lambda\}$. With the above parameters, (53) and (57) give that $\hat{p}_0=p_0$, $\bar{q}_{\varsigma}=q_{11}=1$, $\bar{Q}=p_0$ and $\bar{p}=1-p_0$. Thus, the term $\Upsilon_{\varsigma}=\Upsilon_1$ in (59) becomes

$$\Upsilon_1=\frac{1-p_0}{(1-p_0\rho)(1-p_0\alpha)}\left(\rho(1-p_0\alpha)+\frac{(\alpha-\rho)}{1-p_0}\sum_{l=1}^{\Lambda}p_l(p_0\rho)^{\ell}\right)=\frac{(1-p_0)\sigma}{1-p_0\alpha},$$

where σ is given in (37). Therefore, for the i.i.d. case, $\Upsilon_1<1$ if and only if (38) holds, and Theorem 5 reduces to Theorem 2. □

VIII. Numerical examples

Having established sufficient conditions for stochastic stability of the anytime control loops, we next study performance issues. For that purpose, we assume that the execution time available is i.i.d., uniformly distributed in the interval $[0,1]$. The execution time can also be viewed as the fraction of the maximum possible processor time that is available at any time step. Denote the time taken to calculate one control input by $\tau\in(0,1)$. The probability distribution of $\{N\}_{\mathbb{N}_0}$, see (19), is then given by

$$p_l=\tau,\quad \forall l\in\{0,1,\ldots,\Lambda-1\},\quad p_{\Lambda}=1-\Lambda\cdot\tau, \tag{62}$$

where $\Lambda=\lfloor 1/\tau\rfloor$ is the maximum number of control inputs that can be calculated at any time step. Throughout this section, tentative controls in (9) are obtained by evaluating κ for the corresponding predicted plant state.

To evaluate control performance, we consider the empirical cost

$$J = \frac{1}{10^5}\mathbf{E}\left\{\sum_{k=0}^{10^5-1}\left(0.2x^2(k) + 2u^2(k)\right)\right\},$$

where expectation is taken with respect to the availability of execution time as described above.

We first consider a nonlinear plant model (adapted from [31]):

$$x(k+1) = x(k) + 0.01(x^3(k) + u(k)) + w(k), \tag{63}$$

where $w(k)$ is white noise uniformly distributed in the interval $[0, 0.01]$. The baseline control policy is taken as $\kappa(x) = -x^3 - x$. It can be verified that if one chooses $V(x) = |x|$, then Assumption 1 is satisfied for $\varphi_1(s) = \varphi_2(s) = s$ and $\rho = 0.99$. Fig. 4 shows the percentage improvement in cost achieved as a function of the time taken to calculate one control input for both algorithms A_1 and A_2, as compared to the baseline algorithm (4). It can be appreciated in that figure, both algorithms proposed give a significant performance improvement, with Algorithm A_2 further outperforming Algorithm A_1.[8]

As the plant model becomes more open-loop unstable, the proposed algorithms can be expected to give higher performance gains. Figure 5 illustrates this intuitive effect for the linear model

$$x(k+1) = ax(k) + u(k) + w(k), \tag{64}$$

with system parameter $a \in [0.5, 1.5]$, and where $w(k)$ is i.i.d, Gaussian with zero mean and variance 0.1. The policy κ is taken as the associated LQR control law; $\{N\}_{\mathbb{N}_0}$ is distributed as in (62) with $\tau = 0.3$. The percentage improvement is plotted for algorithms A_1 and A_2, as compared to the baseline algorithm (4).

We finally examine the effect of artificially limiting the maximum buffer size. In particular, if the buffer size is taken as 1, then one recovers the baseline algorithm (4); as noted in Section VI, with size 2, Algorithms A_1 and A_2 are equivalent. Fig. 6 illustrates empirical results for a linear plant (64) with $a = 1.7$. The processor availability is as per (62) with $\tau = 0.23$, thus, $p_0 = p_1 = p_2 = p_3 = 0.23$ and $p_4 = 0.08$. Allowing the buffer size to be of size 4 gives the best results, although a buffer of size 3 gives almost optimal performance.

[8]A total of 1000 Monte Carlo simulations were used to generate the data. Of course, the obtained results are no more than a case-by-case analysis, and consequently one cannot conclude anything about the superiority of either algorithm in general.

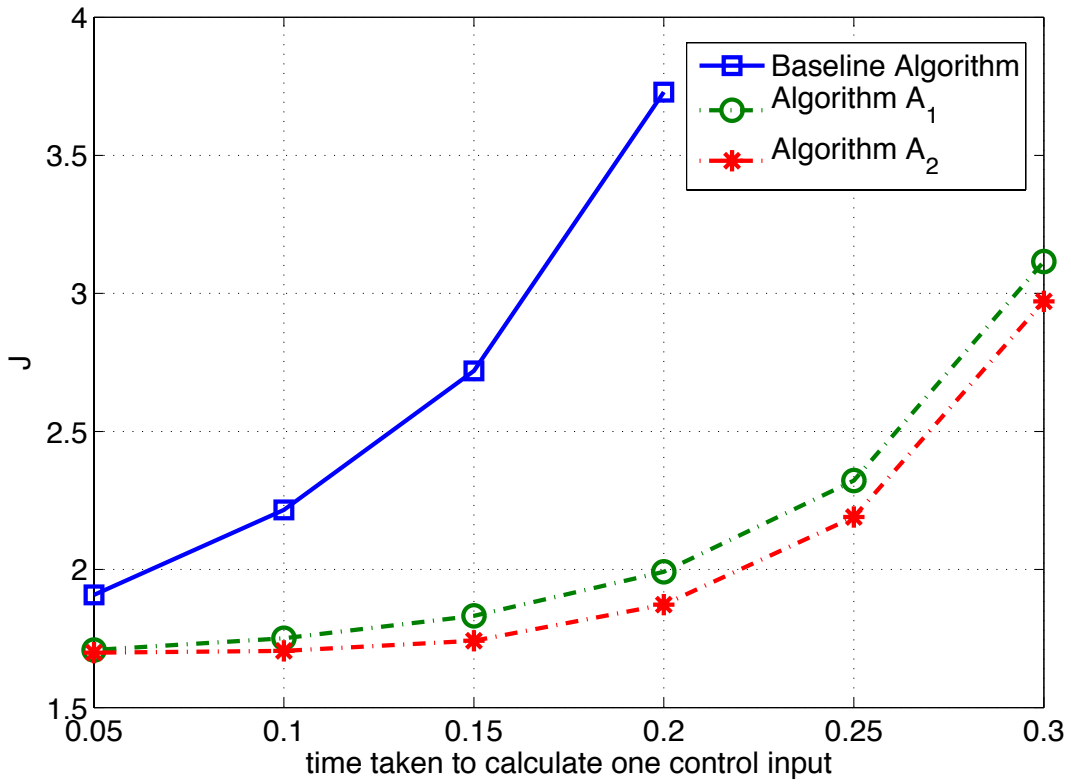

Fig. 4. Empirical cost achieved when controlling the nonlinear plant model (63) with the proposed anytime algorithms and the baseline algorithm (4), as a function of τ, the execution time required to calculate one control input, see (62).

IX. Conclusions

We proposed two related anytime control algorithms for general nonlinear processes. The algorithms use available processing resources to compute sequences of tentative control inputs. Thus, even if the processor does not provide sufficient resources at some time steps, the effect can be partially compensated for. For general non-linear systems, we established sufficient conditions for stochastic stability. Simple numerical examples indicate that the performance gains with the proposed algorithms can be significant, when compared to a simple baseline algorithm. Future work could include examining situations where system assumptions hold only locally, using the stability and performance characterizations obtained for processor scheduling, and the development of anytime algorithms for distributed systems.

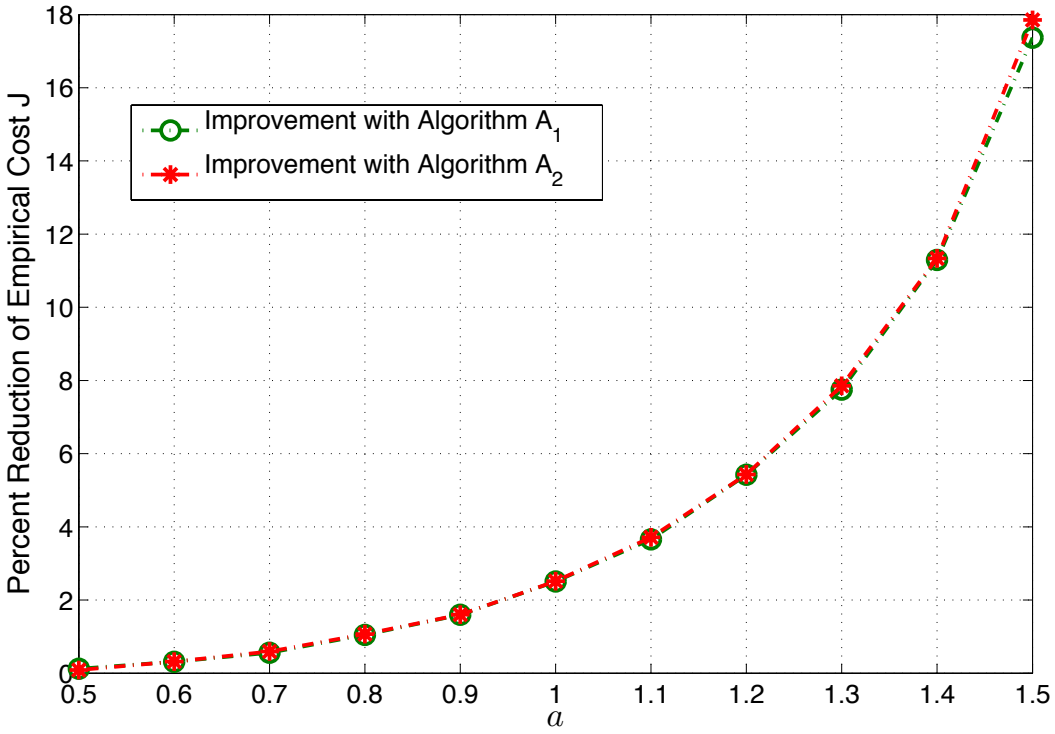

Fig. 5. Performance improvement using algorithms A_1 and A_2 when compared to the baseline algorithm (4), for the model (64).

Acknowledgements: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the paper. Research supported for the first author under Australian Research Council's Discovery Projects funding scheme (project number DP0988601) and in part for the second author by NSF awards 0846631 and 0834771.

REFERENCES

[1] P. Antsaklis and J. Baillieul, "Special issue on networked control systems," *Proceedings of the IEEE*, January 2007.

[2] K. J. Åström and B. Wittenmark, *Computer controlled systems. Theory and design.* Englewood Cliffs, N.J.: Prentice Hall, second ed., 1990.

[3] D. S. Bernstein, *Matrix Mathematics.* Princeton, N.J.: Princeton University Press, 2nd ed., 2009.

[4] R. Bhattacharya and G. J. Balas, "Anytime Control Algorithms: Model Reduction Approach," AIAA Journal of Guidance, Control and Dynamics, 27(5), September-October 2004.

[5] M. Caccamo, T. Baker, A. Burns, and G. Buttazzo, "Real-time scheduling for embedded systems," in *Handbook of Networked and Embedded Systems* (D. Hristu-Varsakelis and W. S. Levine, eds.), Birkhäuser, 2005.

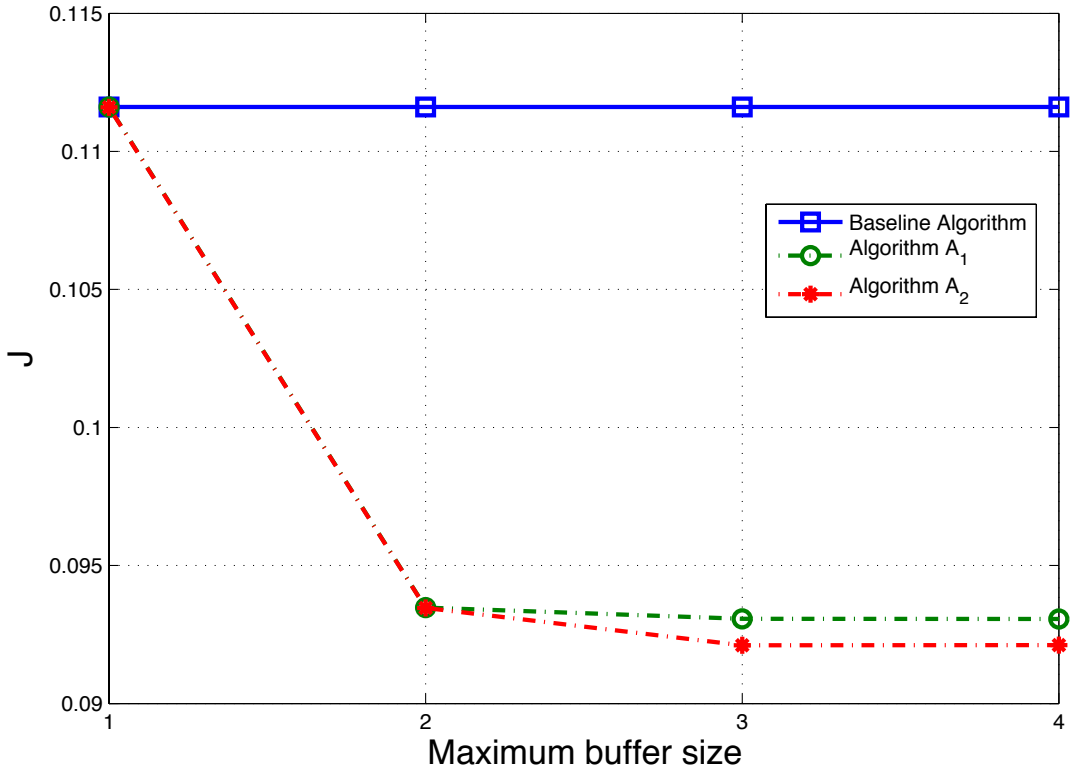

Fig. 6. Effect of limiting the maximum buffer size for the model (64) with $a = 1.7$.

[6] M. Caccamo, G. Buttazzo and L. Sha, "Handling Execution Overruns in Hard Real-time Control Systems," IEEE Transactions on Computers, 51(7), July 2002.

[7] A. Cervin, J. Eker, B. Bernhardsson and K-E. Arzen, "Feedback Feedforward Scheduling of Control Tasks", Real-Time Systems, 23(1-2), 25-53, 2002.

[8] A. Cervin, M. Velasco, P. Marti, and A. Camacho, "Optimal On-Line Sampling Period Assignment: Theory and Experiments," IEEE Transactions on Control Systems Technology, 18(5):1-9, June, 2010.

[9] Y. Fang and K. A. Loparo, "Stochastic stability of jump linear systems," IEEE Trans. Automat. Contr., vol. 47, pp. 1204–1208, July 2002.

[10] R. Findeisen and P. Varutti, "Stabilizing nonlinear predictive control over nondeterministic networks," in Int. Workshop on Assessment and Future Directions of NMPC, 2008.

[11] L. Greco, D. Fontanelli and A. Bicchi, "Almost Sure Stability of Anytime Controllers via Stochastic Scheduling", IEEE Int. Conf. on Decision and Control, 5640-5645, December 2007.

[12] V. Gupta, "On a Control Algorithm for Time-varying Processor Availability," IEEE Transactions on Automatic Control, Provisionally Accepted April 2011. See also V. Gupta, "On an Anytime Algorithm for Control," IEEE Int. Conf. on Decision and Control, December 2009.

[13] V. Gupta and N. C. Martins, "On Stability in the Presence of Analog Erasure Channels between Controller and Actuator,"

IEEE Transactions on Automatic Control, 55(1):175-179, Jan 2010.

[14] V. Gupta and D. E. Quevedo, "On a control Lyapunov function based anytime algorithm for control of nonlinear processes," 2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems, 2010.

[15] V. Gupta and D. E. Quevedo, "On anytime control of nonlinear processes through calculation of control sequences," IEEE Conf. Decis. Contr., 2010.

[16] D. Henriksson and J. Akesson, "Flexible Implementation of Model Predictive Control using Sub-optimal Solutions," Internal Report No. TFRT-7610-SE, Department of Automatic Control, Lund University, April 2004.

[17] D. Henriksson, A. Cervin, J. Akesson and K. E. Arzen, "On Dynamic Real-Time Scheduling of Model Predictive Controllers," In Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, NV, December 2002.

[18] H. Ishii, "Limitations in remote stabilization over unreliable channels without acknowledgements," Automatica, 45: 2278-2285, 2009.

[19] Y. Ji and H. J. Chizeck, "Jump linear quadratic Gaussian control: Steady state solution and testable conditions," Control Theory Adv. Technol., vol. 6, no. 3, pp. 289319, 1990.

[20] Y. Ji, H. J. Chizeck, X. Feng, and K. A. Loparo, "Stability and control of discrete-time jump linear systems," Control Theory Advanced Technology, 7(2): 247-270, 1991.

[21] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, 2nd edition, 1996.

[22] H. J. Kushner, "Introduction to Stochastic Control," Holt, Rinehart and Winston Inc., New York N.Y.

[23] H. J. Kushner and L. Tobias, "On the stability of randomly sampled systems," IEEE Trans. Automat. Contr., AC-14(4):319–324, Aug. 1969.

[24] G. F. Lawler, "Introduction to Stochastic Processes," Chapman and Hall, 2006.

[25] D. Liu, X. Hu, M.D. Lemmon, and Q. Ling, "Scheduling Tasks with Markov-Chain Constraints," 17th Euromicro Conference on Real-time Systems, July 2005.

[26] S. P. Meyn, "Ergodic theorems for discrete time stochastic systems using a stochastic Lyapunov function," SIAM Journal on Control and Optimization, vol. 27, pp. 14091439, Nov. 1989.

[27] L. K. McGovern and E. Feron, "Requirements and Hard Computational Bounds for Real-time Optimization in Safety Critical Control Systems," IEEE Conference on Decision and Control (CDC 98), 1998.

[28] L. K. McGovern and E. Feron, "Closed-loop Stability of Systems Driven by Real-Time Dynamic Optimization Algorithms," IEEE Conference on Decision and Control (CDC 99), 1999.

[29] D. Muñoz de la Peña and P. D. Christofides, "Lyapunov-based model predictive control of nonlinear systems subject to data losses," IEEE Trans. Automat. Contr., vol. 53, pp. 20762089, Sept. 2008.

[30] R. M. Murray, J. Hauser, A. Jadbabaie, M. B. Milam, N. Petit, W. B. Dunbar and R. Franz, "Online control customization via optimization-based control," chapter in "Software-Enabled Control, Information technology for dynamical systems" (eds) T. Samad , G. Balas, 149-174, Wiley-Interscience, 2003.

[31] D. Nešić, A. R. Teel and P. V. Kokotović, "Sufficient conditions for stabilization of sampled-data nonlinear systems via discrete-time approximations," Sys. Contr. Lett., 38(4-5):259-270, 1999.

[32] J. Nilsson, B. Bernhardsson and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," Automatica, vol. 34, no. 1, pp. 5764, 1998.

[33] G. Pin and T. Parisini, "Stabilization of networked control systems by nonlinear model predictive control: A set invariance approach," in Int. Workshop on Assessment and Future Directions of NMPC, 2008.

[34] D. E. Quevedo and D. Nešić, "Robust Stability of Packetized Predictive Control of Nonlinear Systems with Disturbances and Markovian Packet Dropouts," Automatica, accepted for publication.

[35] D. E. Quevedo and D. Nešić, "Input-to-state stability of packetized predictive control over unreliable networks affected by packet-dropouts," IEEE Trans. Automat. Contr., vol. 56, no. 2, pp. 370–375 Feb. 2011.

[36] D. E. Quevedo, J. Østergaard, and D. Nešić, "Packetized predictive control of stochastic systems over bit-rate limited channels with packet loss," IEEE Trans. Automat. Contr., vol. 56, pp. 2854–2868, Dec. 2011.

[37] D. E. Quevedo, E. I. Silva, and G. C. Goodwin, "Control over unreliable networks affected by packet erasures and variable transmission delays," IEEE J. Select. Areas Commun., vol. 26, pp. 672685, May 2008.

[38] L. Schenato, "To hold or to zero control inputs with lossy links?", IEEE Trans. Aut. Contr., 54(5):1093–1099, May 2009.

[39] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, S. S. Sastry, "Foundations of control and estimation over lossy networks," Proceedings of the IEEE, 95(1), pp. 163-187, Jan. 2007.

[40] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," IEEE Transactions on Automatic Control, 44(3):648-654, 1999.

[41] D. Seto, J.P. Lehoczky, L. Sha, and K.G. Shin, "On Task Schedulability in Real-Time Control System," Proc. IEEE Real-Time Systems Symp., Dec. 1996.

[42] E. D. Sontag, "Smooth stabilization implies coprime stabilization," IEEE Trans. Automat. Contr., vol. 34, pp. 435–443, Apr. 1989.

[43] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," IEEE Transactions on Automatic Control, 52(9), 1680-1685, Sept. 2007.

[44] P. L. Tang and C. W. de Silva, "Compensation for transmission delays in an Ethernet-based control network using variable-horizon predictive control," IEEE Trans. Contr. Syst. Technol., vol. 14, pp. 707 718, July 2006.

[45] M. Velsaco, P. Marti, and E. Bini, "On Lyapunov Sampling for Event-driven Controllers," In Proc. of the 48th IEEE Conference on Decision and Control (CDC09), Shanghai, China, Dec. 2009.

[46] X. Wang and M. D. Lemmon, "Self-triggered Feedback Control Systems with Finite-Gain L2 Stability," IEEE Transactions on Automatic Control, 45(3):452–457, Mar. 2009.

[47] L. Xie and L. Xie, "Stability analysis of networked sampled-data linear systems with Markovian packet losses," IEEE Trans. Automat. Contr., 54(6):1375–1381, June 2009.

[48] T. Zhou, X. Hu and E.H-M. Sha, "A probabilistic performance metric for real-time system design," 7th International Workshop on Hardware-Software Codesign (CODES) (ACM/IEEE), pp. 90-94, May 1999.