Event-triggered Anytime Control with Two Controllers

K.Q. Huang, Thuy V. Dang, K.V. Ling, and D.E. Quevedo

Abstract—In networked and multi-tasking environments, measurement data and processing resources may not be available at times when control calculations need to be executed. In this paper, we present an anytime algorithm which features two control policies: a coarse policy and a fine policy. The fine control policy requires more processing resources than the coarse policy. With this scheme, the network and processing resources can be used more efficiently, and performance can be improved. Specifically, for a given packet dropout rate and process availability which are independent and identically distributed (i.i.d.), the proposed two-controller scheme achieves better closed-loop performance with a lower channel utilization than alternative control formulations.

I. INTRODUCTION

In networked and embedded control systems, communication and computational resources are often shared. Thus the implicit assumption traditionally made in control design that measurement data and processing resources will always be available when a control algorithm needs to be executed may break down. Anytime algorithm is one method to design control algorithms to ensure closed-loop stability and performance in the presence of random data packets loss or processing resource unavailability.

In [1] [2], an anytime algorithm was proposed for the control of a non-linear process. A sensor communicates with a controller node through an erasure channel which introduces independent and identically distributed (i.i.d.) packet dropouts. Processor availability for control is random, unknown a priori, and at times, insufficient to calculate a control input. Tentative input sequences are calculated whenever there are excess processing resources and stored in an internal buffer for potential use in the future when processing resources become unavailable. This results in a better performance than a base-line algorithm in terms of empirical closed-loop cost, channel utilization and stochastic stability behavior.

In this paper, we go beyond the proposed approach of [2] and present an anytime algorithm which features two control policies: a coarse policy and a fine policy. With the proposed anytime algorithm, the communication and processing resources can be used more efficiently. We show that performance in terms of empirical closed-loop cost, channel utilization and the region for stochastic stability can be improved by the proposed scheme.

The fine control policy could be viewed as an improved version of the coarse policy that requires more processing resource than the coarse policy. Such ideas are widespread, e.g., in image compression applications, where iterative algorithms are used to achieve a flexible, on demand compromise between the quality of its solution and the (online) computational resource requirements [3]. In control applications, online optimisation-based algorithms, such as Model Predictive Control (MPC) [4], are candidates for anytime algorithms. For real-time applications of MPC, where processor availability is usually assumed fixed and known a priori, techniques such as early termination [5] or suboptimal MPC [6], aim at providing suboptimal solutions to reduce the online computations and thereby meet real-time requirements have been proposed.

The remainder of this paper is organized as follows: In Section 2 we state the assumptions and system model for the development of anytime control schemes. Section 3 presents the proposed two-controller anytime algorithm. Then Section 4 carries out stochastic stability analysis. Numerical simulations and results are documented in Section 5. Section 6 draws conclusions.

Notation \( \mathbb{N} = \{1, 2, \ldots\} \) represents the natural numbers, \( \mathbb{N}_0 \triangleq \mathbb{N} \cup \{0\} \), \( \mathbb{R} \) represents the real numbers, \( \mathbb{R}_{\geq 0} \triangleq [0, +\infty) \); \( \{x\}_K \) stands for \( \{x(k) : k \in K\} \), \( K \subseteq \mathbb{N}_0 \), \( |x| = \sqrt{x^T x} \) denotes the Euclidean norm of vector \( x \). A function \( \phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \) is of class \( K_{\infty} \) (\( \phi \in K_{\infty} \)) if it is continuous, zero at the origin, strictly increasing and unbounded. \( \mathbb{P}\{\Omega\} \), \( \mathbb{P}\{\Omega|\Gamma\} \) denote the probability of an event \( \Omega \), and the conditional probability of \( \Omega \) given \( \Gamma \) respectively. The expected value of a random variable \( \nu \) given \( \Gamma \) is denoted by \( \mathbb{E}\{\nu|\Gamma\} \), and \( \mathbb{E}\{\nu\} \) represents the unconditional expectation.

II. SYSTEM MODEL AND ASSUMPTIONS

Consider a discrete-time non-linear (and possible open-loop unstable) plant being controlled across a communication network that stochastically erases data transmitted (Fig. 1), sampled periodically with sampling interval \( T_s > 0 \):

\[
x(k+1) = f(x(k), u(k)) \quad , \quad k \in \mathbb{N}_0
\]

where \( x \in \mathbb{R}^n \) is the plant state, and \( u \in \mathbb{R}^p \) is the plant input. The transmission between sensor and controller node...
is through a delay-free link with dropouts indicated by the ternary process \( \{ \beta \} \): \( \beta(k) \in \{0, 1, 2\} \) where
\[
\begin{align*}
\beta(k) &= \begin{cases} 
0 & \text{if } x(k) \text{ is received with error} \\
1 & \text{if } x(k) \text{ is received error-free} \\
2 & \text{if the sensor did not transmit (i.e. } |x(k)| < d) 
\end{cases}
\end{align*}
\]

We adopt the following assumptions, also made in \cite{2}:

**Assumption 1** (Stabilizability): There exist \( V : \mathbb{R}^n \to \mathbb{R}_{\geq 0} \), \( \varphi_1, \varphi_2 \in \mathcal{K}_\infty \), \( \kappa : \mathbb{R}^n \to \mathbb{U} \), \( \mathbb{U} \subset \mathbb{R} \), and a constant \( \rho \in [0, 1) \), such that
\[
\varphi_1(|x|) \leq V(x) \leq \varphi_2(|x|) \quad \forall x \in \mathbb{R}^n
\]
\[
V(f(x, \kappa(x))) \leq \rho V(x) \quad \forall x \notin \mathcal{B}_d
\]
where \( \mathcal{B}_d \triangleq \{ x \in \mathbb{R}^n : |x| < d \} \).

**Assumption 2** (Processor availability): The processor availability for control at different time-instants is independent and identically distributed (i.i.d.). Thus, we denote by \( N(k) \in \{0, 1, 2, \ldots, \Phi \} \), how many unit processing resources are available at time \( k \) (i.e. during the interval \( \left[ kT_s, (k+1)T_s \right) \)). The process \( \{ N \} \) has conditional probability distribution:
\[
\Pr \{ N(k) = j \} | \beta(k) = 1 \} = p_j, \quad j \in \{0, 1, 2, \ldots, \Phi \}
\]
where \( p_j \in [0, 1] \) are known.
For other values of \( \beta(k) \), no plant input is calculated. Thus the processing resources are considered not available regardless, i.e.
\[
\Pr \{ N(k) = 0 \} | \beta(k) \in \{0, 2\} \} = 1
\]

**Assumption 3** (Erasure channel): The transmissions are Bernoulli with packet transmission success probability:
\[
\Pr \{ \beta(k) = 1 | x > d \} = \Pr \{ \beta(k) = 1 | \beta(k) \neq 2 \} = q
\]

**Assumption 4** (Open-loop bound): With \( \rho, V \) as in (2), there exists \( \alpha > \rho \) such that:
\[
V(f(x, \theta_p)) \leq \alpha V(x), \quad \forall x \in \mathbb{R}^n
\]

### III. ANYTIME ALGORITHM WITH TWO CONTROLLERS

In this section, we present our proposed anytime algorithm featuring two control policies, denoted as the coarse policy \( \kappa_1 \) and the fine policy \( \kappa_2 \). The following assumption is made regarding \( \kappa_1 \) and \( \kappa_2 \):

**Assumption 5** (Control Policies): Policy \( \kappa_1 \) requires one unit (i.e., \( N(k) = 1 \)) while \( \kappa_2 \) requires \( \eta \) units \( (N(k) = \eta) \) of processor resource to calculate the control input. The control policy \( \kappa_2 \) is better than \( \kappa_1 \) in the sense that, \( V(f(x(k), \kappa_i(x))) \leq \rho_i V(x), \quad (i = 1, 2) \) with \( \rho_2 < \rho_1 < 1 \), see (2).

We will next present two algorithms which implement \( \kappa_1 \) and \( \kappa_2 \) using limited processing and communication resources, as described in Section II.

**A. Algorithm A1: two controllers without buffer**

Algorithm A1 amounts to a direct implementation of \( \kappa_1 \) and \( \kappa_2 \) without buffer. The plant input is given as follows:
\[
u(k) = \begin{cases} 
k_2(x(k)) & N(k) \geq \eta \\
k_1(x(k)) & 0 < N(k) < \eta \\
0_p & N(k) = 0, \end{cases}
\]
where \( u(k) \) with \( k \in \mathbb{N}_0 \) denotes the plant input which is applied during the interval \( [kT_s + \delta, (k+1)T_s + \delta) \), resulting in the closed-loop system:
\[
x(k+1) = \begin{cases} 
f(x(k), \kappa_2(x(k))) & N(k) \geq \eta \\
f(x(k), \kappa_1(x(k))) & 0 < N(k) < \eta \\
f(x(k), \theta_p) & N(k) = 0.
\end{cases}
\]
The probability of the controller assignment, say \( g_j \), is a conditional distribution according to \( N(k) \) in Assumption 2:
\[
g_0 = \Pr \{ N(k) = 0 | \beta(k) = 1 \} = p_0
\]
\[
g_1 = \Pr \{ 0 < N(k) < \eta | \beta(k) = 1 \} = \sum_{j=1}^{\eta-1} p_j
\]
\[
g_2 = \Pr \{ N(k) \geq \eta | \beta(k) = 1 \} = \sum_{j=\eta}^{\Phi} p_j.
\]
In fact, \( g_2 = 1 - g_0 - g_1 \) since \( g_0 + g_1 + g_2 = 1 \).

**B. Algorithm A2: two controllers with buffer**

Similar to the idea of *Anytime Control Algorithm* in \cite{2}, we use a local buffer \( b(k) \) of size \( \Lambda \), to store the sequence of tentative future plant inputs calculated by either \( \kappa_1 \) or \( \kappa_2 \) at time \( k \). The algorithm will execute \( \kappa_2 \) whenever possible even though this might result in fewer tentative control inputs calculated had policy \( \kappa_1 \) been used. To be more specific, the control policies \( \kappa_1 \) or \( \kappa_2 \) will be executed depending on the values of \( N(k) \), as illustrated below:

1. If \( N(k) < \eta \), then there are insufficient processing time units for \( \kappa_2 \). The sequence of tentative inputs will be computed iteratively as follows:
\[
u_0,\kappa_1(k) = \kappa_1(x(k))
\]
\[
u_1,\kappa_1(k) = \kappa_1(f(x(k), u_0,\kappa_1(k)))
\]
\[
\vdots
\]
\[
u_{N(k)-1,\kappa_1(k)} = \kappa_1(\cdots)
\]
We make the simplifying assumption that \( \delta \ll T_s \), so that the nominal control design need not account for the computational time delay.
2) If \( N(k) \geq \eta \), then we can write
\[
N(k) = \tau(k)\eta + M(k)
\]
where \( \tau(k) = \lfloor \frac{N(k)}{\eta} \rfloor \), \( M(k) = N(k) \mod \eta \), i.e., we can compute a tentative control sequence by iterating \( \tau(k) \) times of \( \kappa_2 \), and \( M(k) \) times of \( \kappa_1 \), as follows:
\[
\begin{align*}
  u_0,\kappa_2(k) &= \kappa_2(x(k)) \\
  u_1,\kappa_2(k) &= \kappa_2(f(x(k), u_0,\kappa_2(k))) \\
  \vdots \\
  u_{\tau-1,\kappa_2}(k) &= \kappa_2(\cdots) \\
  u_{\tau,\kappa_1}(k) &= \kappa_1(\cdots) \\
  \vdots \\
  u_{\tau+M(k)-1,\kappa_1}(k) &= \kappa_1(\cdots)
\end{align*}
\]
In addition, the buffer is first cleared whenever new calculations are made.

In [2], a baseline and an anytime algorithms were described; we denote them as algorithms \( R_1 \) and \( R_2 \) respectively. It is easy to see that, if \( \kappa_1 = \kappa_2 \) and \( \eta = 1 \), our proposed algorithms \( A_1 \) and \( A_2 \) reduce to \( R_1 \) and \( R_2 \) respectively. In other words, algorithms \( R_1 \) and \( R_2 \) are algorithms with one controller without and with buffer respectively.

The following example illustrates our proposed algorithms \( A_1 \) and \( A_2 \) and the relationship with the anytime algorithm \( R_2 \) proposed in [2].

**Example 1.** Suppose that \( \Phi = 3, \eta = 2, \Lambda = 2 \) and that the processor availability is such that \( N(0) = 3, N(1) = 0, N(2) = 1, N(3) = 2 \).

If algorithm \( A_2 \) is used, then the buffer state at times \( k \in \{0, 1, 2, 3\} \) becomes:
\[
\begin{align*}
  b(0) &= [0, 0, 0, 0] \\
  b(1) &= [0, 1, 0, 0] \\
  b(2) &= [0, 0, 0, 1] \\
  b(3) &= [0, 0, 1, 0]
\end{align*}
\]

Note that, in the situation without buffers, for \( b_1, b_2, \kappa_2 \), suppose that Assumptions 1 to 5 hold and that:
\[
\Gamma \triangleq ((1 - q)\alpha + q(g_0\alpha + g_1\rho_1 + g_2\rho_2)) < 1
\]
where \( g_i \) are as in (6).

Using (3) and (6), it follows that:
\[
E\{V(x(1)|x(0) = \chi, \beta(0) = 1)\} \leq (g_0\alpha + g_1\rho_1 + g_2\rho_2)V(\chi)
\]
This example suggests that algorithm \( A_2 \) will outperform \( A_1 \) and \( R_2 \) by using a better control law. It also hints at the interplay between buffering and control policies, which will depend on specific processor availability scenarios.

**Remark 1:** In this paper, we adopt the strategy that the controller calculates \( \kappa_2 \) whenever possible for the reason that we implicitly assume that the buffer is unlikely to be depleted. If the external environment is such that one is likely to experience consecutive events of \( N(k) = 0 \) or \( \beta(k) = 0 \), i.e., events which will deplete the buffer, the alternative of letting the controller calculate \( \kappa_1 \) whenever possible may be a better choice.

**Remark 2:** One could also think of \( \kappa_2 \) being a refinement of \( \kappa_1 \). The algorithm simply carries out calculations if possible. If it does not have access to the processor, then calculations are not carried out. Prior knowledge of processor availability is not needed and calculations can be terminated at any time. For example, the following computational steps could be carried out: (1) calculate \( \kappa_1 \) and write in the buffer; (2) if there is still computing resource, refine \( \kappa_1 \) to \( \kappa_2 \) and overwrite \( \kappa_1 \); (3) advance the buffer pointer to the next available location; (4) go to (1).

In the next section we will establish conditions for closed-loop stochastic stability, when the two algorithms are used with i.i.d. processor and communication resources.

**IV. STABILITY ANALYSIS**

**A. Two controllers without buffer (Algorithm \( A_1 \))**

**Theorem 4.1** (Stability with Algorithm \( A_1 \)): In a two-controller scheme without buffer, suppose that Assumptions 1 to 5 hold and that:
\[
\Gamma \triangleq ((1 - q)\alpha + q(g_0\alpha + g_1\rho_1 + g_2\rho_2)) < 1
\]
where \( g_i \) are as in (6).

Then, for all \( k \in N_0, x(0) = \chi \in \mathbb{R}^n \), we have:
\[
E\{\varphi_1(|x(k)|x(0) = \chi)\} \leq \Gamma^k V(x(0)) + \frac{(\alpha - 1)\varphi_2(d)}{1 - \Gamma} < \infty
\]

**Proof:** Note that, in the situation without buffers, for i.i.d. processor and channel availabilities, \( \{x\}_{k=0} \) in (5) is Markovian. This is because the next state of \( x(k) \) depends only on the outcome \( \beta(k) \) which is judged by the current state only and not on the sequence of events that preceded it.

Similar to Theorem 1 in [2], we have:
\[
E\{V(x(1)|x(0) = \chi, \beta(0) = 0)\} \leq \alpha V(\chi)
\]
\[
E\{V(x(1)|x(0) = \chi, \beta(0) = 2)\} \leq \alpha V(\chi) \leq \alpha \varphi_2(d)
\]

For \( \beta(0) = 1 \), we have:
\[
E\{V(x(1)|x(0) = \chi, \beta(0) = 1)\} = \sum_{j \in N_0} E\{V(x(1)|x(0) = \chi, \beta(0) = 1, N(0) = j)\} \times \text{Pr}(N(0) = j|x(0) = \chi, \beta(0) = 1)
\]
Using (3) and (6), it follows that:
\[
E\{V(x(1)|x(0) = \chi, \beta(0) = 1)\} \leq (g_0\alpha + g_1\rho_1 + g_2\rho_2) V(\chi)
\]
If $x(0) \not\in B_d$, in view of Assumption 3, we have:

\[
\Pr\{\beta(0) = 0|x(0) \not\in B_d\} = 1 - q \\
\Pr\{\beta(0) = 1|x(0) \not\in B_d\} = q
\]

so that

\[
E\{V(x(1)|x(0) \not\in B_d)\} = E\{V(x(1)|x(0) \not\in B_d, \beta(0) = 0)\}(1 - q) + E\{V(x(1)|x(0) \not\in B_d, \beta(0) = 1)\}q
\]

\[
\leq ((1 - q)\alpha + q(g_0\alpha + g_1\rho_1 + g_2\rho_2))V(\chi) = \Gamma V(\chi)
\]

Direct calculations give that

\[
\Gamma = \alpha - q(1 - g_0\alpha + q(g_1\rho_1 + g_2\rho_2)) = \alpha - q(g_1 + g_2)\alpha + q(g_1\rho_1 + g_2\rho_2)
\]

Hence, $\alpha - \Gamma > 0$ since $\rho_2 < \rho_1 < \alpha$. In addition, $V(\chi) < \varphi_2(d)$ for all $\chi \in B_d$ and we have $(\forall \chi \in B_d)$:

\[
(\alpha - \Gamma)V(\chi) < (\alpha - \Gamma)\varphi_2(d)
\]

\[
\Rightarrow \alpha V(\chi) < \Gamma V(\chi) + (\alpha - \Gamma)\varphi_2(d)
\]

Combining with

\[
E\{V(x(1)|x(0) \not\in B_d)\} \leq \Gamma V(\chi)
\]

and

\[
E\{V(x(1)|x(0) \in B_d)\} \leq \alpha V(\chi)
\]

we have

\[
E\{V(x(1)|x(0) = \chi\} \leq \alpha V(\chi) < \Gamma V(\chi) + (\alpha - \Gamma)\varphi_2(d)
\]

Using the Markov property as in [7], we obtain (8) if (7) holds.

**B. Two controllers with buffer (Algorithm $A_2$)**

If Assumption 5 holds, for ease of exposition we introduce the effective buffer length (at time $k$) as $\lambda(k) \in \{0, 1, \ldots, \Lambda\}, k \in \mathbb{N}_0$, where:

\[
\lambda(k) = \begin{cases} 
\tau(k) + M(k) & \text{if } N(k) \geq \eta \\
N(k) & \text{if } 1 \leq N(k) < \eta \\
\max\{0, \lambda(k - 1) - 1\} & \text{if } N(k) = 0 \text{ and } \\
\beta(k) \in \{0, 1\} & \text{if } \beta(k) = 2. 
\end{cases}
\]

We assume that the initial effective buffer length $\lambda(0) = 0$ and denote the time steps where $\lambda(k) = 0$ via $K = \{k_i\}_{i \in \mathbb{N}_0}$, where $k_0 = 0$ and

\[
k_{i+1} = \inf\{k \in \mathbb{N}: k > k_i, \lambda(k) = 0\}, i \in \mathbb{N}_0
\]

Thus, we describe the amount of time steps between consecutive elements of $K = \{k_i\}_{i \in \mathbb{N}_0}$ via the process $\Delta_i, i \in \mathbb{N}_0$, where $\Delta_i = k_{i+1} - k_i$. It is easy to see that:

\[
\beta(k_i + l) \in \{0, 1\}, \forall l \in \{1, 2, \ldots, \Delta_i - 1\}, \forall i \in \mathbb{N}_0
\]

**Theorem 4.2 (Stability with Algorithm $A_2$):** Suppose Assumptions 1 to 5 hold and define:

\[
\Psi \triangleq \alpha \sum_{j \in \mathbb{N}} \sum_{m=0}^{j} \rho_1^m \rho_2^{-m-1} \Pr\{\Delta_i = j, r_i = m|\beta(k_i + 1) \neq 2\}
\]

where $r_i$ is the number of the time $\kappa_i$ is being used between consecutive elements of $K$, i.e. between $k_i$ and $k_{i+1}$.

If Algorithm $A_2$ is used and $\Psi < 1$, then, for all $i \in \mathbb{N}$

\[
\max_{k \in \{k_i, k_{i+1}, \ldots, k_{i+1-1}\}} E\{\varphi_1(\chi)\}
\]

\[
\leq (1 + \alpha \frac{1}{1 - \rho_1})V(\chi) + \frac{\varphi_2(d)}{1 - \Psi} < \infty
\]

**Proof:** Firstly, we note that for all $k_i \in K$ and $l \in \{1, 2, \ldots, \Delta_i - 1\}$, we have,

\[
\begin{cases} 
\{u(k_i) = 0_p \\
\{u(k_i + l) \in \{\kappa_1(x(k_i + l)), \kappa_2(x(k_i + l))\}.
\end{cases}
\]

Therefore, by using (2) and (3) we get

\[
E\{V(x(k_i+1))|x(k_i) = \chi, \Delta_i = j, r_i = m\}
\leq \alpha \rho_1^m \rho_2^{-m-1} V(\chi), \forall \chi \in \mathbb{R}^n
\]

When the buffer is emptied by the trigger $\beta(k) = 2$, we have:

\[
E\{V(x(k_i+1))|x(k_i) = \chi, \beta(k_i+1) = 2\} < \varphi_2(d) \triangleq D
\]

Using the law of total expectation, the definition of conditional probability and the assumptions made, we obtain

\[
E\{V(x(k_i+1))|x(k_i) = \chi\}
\leq E\{V(x(k_i+1))|x(k_i) = \chi, \beta(k_i+1) = 2\}
\]

\[
\times \Pr\{\beta(k_i+1) = 2|x(k_i) = \chi\}
\]

\[
+ E\{V(x(k_i+1))|x(k_i) = \chi, \beta(k_i+1) \neq 2\}
\]

\[
\times \Pr\{\beta(k_i+1) \neq 2|x(k_i) = \chi\}
\]

\[
\leq \varphi_2(d) + E\{V(x(k_i+1))|x(k_i) = \chi, \beta(k_i+1) \neq 2, \Delta_i = j\}
\]

\[
\times \Pr\{\Delta_i = j|x(k_i) = \chi, \beta(k_i+1) \neq 2\}
\]

\[
= D + \sum_{j \in \mathbb{N}_0} \sum_{m=0}^{j} \Pr\{\Delta_i = j|x(k_i) = \chi, \beta(k_i+1) \neq 2\}
\]

\[
\times \Pr\{r_i = m|\Delta_i = j, x(k_i) = \chi, \beta(k_i+1) \neq 2\}
\]

\[
\leq D + \sum_{j \in \mathbb{N}_0} \alpha \rho_1^m \rho_2^{-m-1} V(\chi)
\]

\[
\times \Pr\{\Delta_i = j, r_i = m|x(k_i) = \chi, \beta(k_i+1) \neq 2\}
\]

\[
= \varphi_2(d) + \Psi V(\chi), \forall \chi \in \mathbb{R}^n
\]

with $\Psi$ defined as in (11). Note that, the definition of $K = \{k_i\}_{i \in \mathbb{N}_0}$ gives that $u(k_i) = 0_p, b(k_i) = 0_p, \lambda(k_i) = N(k_i) = 0$, so $\{x\}$ is Markovian, giving

\[
E\{V(x(k_i))|x(k_0) = \chi\} \leq \Psi V(\chi) + \frac{\varphi_2(d)}{1 - \Psi}
\]

Recall that $0 < \rho_2 < \rho_1 < 1$, and with Assumptions 1 and 4, we have

\[
V(x(k_i+1)) \leq \alpha V(x(k_i)) \tag{13}
\]

\[
V(x(k+1)) \leq \rho_1 V(x(k)), \forall k \not\in K \tag{14}
\]
Then, it follows that
\[
E \left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \mid x(k_i) = \chi, \Delta_i = j \geq 2 \right\} \\
\leq \left( 1 + \alpha \sum_{l=0}^{j-2} \rho_1^l \right) V(\chi) + \varphi_2(d) \frac{1}{1 - \Psi} \\
\leq \left( 1 + \alpha \frac{1}{1 - \rho_1} \right) V(\chi) + \varphi_2(d) \frac{1}{1 - \Psi}
\]

Then, by using a similar method to that employed in [8] (Thm.5.3) we have:
\[
E \left\{ \sum_{k=k_i}^{k_{i+1}-1} V(x(k)) \mid x(k_0) = \chi \right\} \\
\leq (1 + \alpha \frac{1}{1 - \rho_1}) \Psi_i V(\chi) + \varphi_2(d) \frac{1}{1 - \Psi}, \forall i \in \mathbb{N}
\]

Using the law of total expectation and with \( \Psi < 1 \), we obtain (12).

Theorem 4.2 establishes a sufficient condition for stability when Algorithm\( A_2 \) is used and processing and communication resources are random. Due to space constraints, in this paper, we do not show how to evaluate the conditional probability in (11).

V. NUMERICAL EXAMPLES

We assume a plant model of the form (1), but with additive disturbances:
\[
x(k+1) = -x(k) + 0.1 \sin(x(k)) + u(k) + w(k) \\
x(0) = 20
\]
where the disturbance \( w(k) \) is a i.i.d., normally distributed with zero mean and unit variance. For the proposed scheme with two controllers, we adopt
\[
\begin{cases}
\kappa_1(x(k)) = x(k) - 0.1 \sin(x(k)) + \rho_1 |x(k)|, \\
\kappa_2(x(k)) = x(k) - 0.1 \sin(x(k)) + \rho_2 |x(k)|,
\end{cases}
\]
where \( \rho_1 = 0.9 \), \( \rho_2 = 0.7 \). We assume that evaluating \( \kappa_2 \) takes twice as much processor resource than \( \kappa_1 \) (i.e. \( \eta = 2 \) in (4)). We also assume that the maximum buffer length is \( \Lambda = 4 \) and maximum unit processing resources are \( \Phi = 4 \).

For comparison, we also investigated the one controller schemes \( R_1 \) and \( R_2 \) of [2] with controller
\[
\kappa(x(k)) = x(k) - 0.1 \sin(x(k)) + \rho |x(k)|, \quad \rho = 0.9
\]

In all cases, the probability of successful transmission is:
\[
q = \Pr \{ \beta(k) = 1 \mid |x(k)| \geq d \} = 0.5
\]
and the processor resource availability is assumed as
\[
\frac{1 - p_0}{p_0} = \Pr \{ N(k) \geq 1 \mid \beta(k) = 1 \} = 0.8.
\]

For \( N(k) \in \{1, 2, \cdots, 4\} \), we assume that the probabilities \( p_j = \Pr \{ N(k) = j \mid \beta(k) = 1 \} \) are equal for each \( j \in \{1, 2, \cdots, 4\} \). Thus, we have \( g_0 = 0.2, g_1 = 0.8 \) for Algorithm \( R_1 \), and \( g_0 = 0.2, g_1 = 0.2, g_2 = 0.6 \) for Algorithm \( A_1 \) according to Assumptions 2 and 5.

A. Performance

Performance is evaluated, through the closed-loop empirical cost
\[
J \triangleq \frac{1}{500} \sum_{k=201}^{700} \sum_{d=5}^{\infty} x^2(k)
\]
and the Channel Utilisation(%)

\[
\frac{\text{Total number of time steps at which } \beta(k) \neq 2}{\text{Total number of time steps}}
\]

by averaging over \( 10^4 \) realizations.

The effectiveness of algorithms \( A_1 \) and \( R_2 \) can be seen from Fig. 2. For a given threshold \( d \), which is a design parameter to trade-communication channel utilization for control performance in Assumption 1, with algorithms \( A_1 \) and \( R_2 \), the system achieves a smaller empirical closed-loop cost with lesser channel utilization when compared with algorithm \( R_1 \). Obviously, the performance gap between algorithms \( A_1 \) and \( R_2 \) depends on the parameter \( \rho_2 \).

Algorithm \( A_2 \) further outperforms \( A_1 \) and \( R_2 \), albeit in a limited way over \( A_1 \). This is probably because most of the slacks in the system have already been taken up with algorithm \( R_2 \).

B. Stability Regions

Now we investigate the stability boundaries of the proposed control schemes. For ease of comparison, we introduce a parameter \( \epsilon \) that represents the ratio between the shrinking factors of controller \( \kappa_1 \) and \( \kappa_2 \):
\[
\epsilon = \frac{\rho_2}{\rho_1}
\]

We see that \( \epsilon \in [0, 1] \). It can be said that the smaller the \( \epsilon \) is, the “finer” the second controller \( \kappa_2 \) is.

1) Control schemes without buffer: Fig. 3 shows the stability boundaries (\( \Gamma = 1 \) in (7) and Theorem 1 in [2]) of algorithms \( A_1 \) and \( R_1 \).

The parameters for the algorithm \( R_1 \) are \( q = 0.5 \) and \( g_0 = 0.2 \).
It can be seen from Fig. 3 that the Algorithm $A_1$ has a larger region with guaranteed stability as per our results (the left area where $\Gamma < 1$) than the $R_1$ algorithm. Note that when $\epsilon = 1$ the boundaries corresponding to algorithms $R_1$ and $A_1$ coincide.

2) Control schemes with buffer: Fig. 4 shows the stability boundaries of the proposed $A_2$ algorithm compared with the $R_2$ algorithm. For the term $\Omega$ in Algorithm $R_2$, Lemma 2 in [2] gives:

$$\Omega = \alpha (1 - q + p_0 q) (1 + \rho G^T (1 - \rho G)^{-1} e_1)$$

where $\Theta^T$, $e_1$, and $G$ are matrices in terms of $p_i$ and $q$, see [2]. The parameters for the $A_1$ algorithm are $\Phi = 4$, $p_i = 0.2$, $i \in \{0, 1, \ldots, 4\}$, and $q = 0.5$.

For the $A_2$ algorithm, with $\Phi = 4$, $p_i = 0.2$, $i \in \{0, 1, \ldots, 4\}$ and $q = 0.5$, the tentative plant inputs in $A_2$ would depend on the values of $N(k)$ in Assumption 2:

$$N(k) = 1, \quad u_0(k) = \kappa_1(x(k))$$
$$N(k) = 2, \quad u_0(k) = \kappa_2(x(k))$$
$$N(k) = 3, \quad \begin{cases} u_0(k) = \kappa_2(x(k)) \\ u_1(k) = \kappa_1(f(x(k)), u_0(k)) \\ u_0(k) = \kappa_2(x(k)) \\ u_1(k) = \kappa_2(f(x(k)), u_0(k)) \end{cases}$$
$$N(k) = 4, \quad \begin{cases} u_0(k) = \kappa_2(x(k)) \\ u_1(k) = \kappa_2(f(x(k)), u_0(k)) \end{cases}$$

In Fig. 4, we plot the stability boundaries of $A_2$ ($\Psi = 1$) for different values of the parameter $\epsilon = 1$, 0.8, 0.5, 0.05. Fig. 4 illustrates that the stability boundary of the $A_2$ algorithm depends on how good the fine control law $\kappa_2$ is. From this figure, we can see that $A_2$ can handle a more unstable plant provided $\kappa_2$ is good enough.

VI. CONCLUSIONS

We have extended the event-triggered anytime control algorithm of [2] to a scheme which features a coarse and a fine control policy. While buffering is a good technique to better utilize the excess computational resource which may become available in an a priori unknown manner, introducing a fine control policy can be a useful alternative. By using a coarse and a fine control policy (without buffering), significant performance gains can be achieved. Adding buffer in the proposed two-controller scheme could further improve performance, if there are still slacks in the system that can be exploited. Our analysis also showed that the stability region can be enlarged with the proposed two-controller scheme.

REFERENCES