Chapter 1
Event-Triggered Feedback in Control, Estimation, and Optimization

Michael Lemmon

Abstract Networked control systems often send information across the communication network in a periodic manner. The selected period, however, must assure adequate system performance over a wide range of operating conditions and this conservative choice may result in significant over-provisioning of the communication network. This observation has motivated the use of *sporadic* transmission across the network's feedback channels. *Event-triggering* represents one way of generating such sporadic transmissions. In event-triggered feedback, a sensor transmits when some internal measure of the novelty in the sensor information exceeds a specified threshold. In particular, this means that when the gap between the current and the more recently transmitted sensor measurements exceeds a state-dependent threshold, then the information is transmitted across the channel. The state-dependent thresholds are chosen in a way that preserves commonly used stability concepts such as input-to-state stability or $\mathscr{L}_2$ stability. This approach for threshold selection therefore provides a systematic way of triggering transmissions that provides some guarantees on overall control system performance. While early work in event-triggering focused on control applications, this technique can also be used in distributed estimation and distributed optimization. This chapter reviews recent progress in the use of state-dependent event-triggering in embedded control, networked control systems, distributed estimation, and distributed optimization.

1.1 Introduction

Embedded and networked control systems often rely on the periodic sampling and transmission of data. This periodic data abstraction is advantageous from the design standpoint. It permits real-time system engineers and control system engineers to pursue their design objectives in relative isolation from each other. While this

Michael Lemmon
University of Notre Dame, Notre Dame, Indiana, USA, e-mail: lemmon@nd.edu

so-called *separation-of-concerns* has proven advantageous from a designer's perspective, it does not necessarily lead to cost effective implementations of the control system. By separating the concerns of the control engineer from the real-time system engineer, one forces each designer to adopt a conservative viewpoint that may lead to unnecessary over-provisioning in the system implementation and hence to higher system costs. When one applies these traditional design principles to extremely large-scale systems, then the cost of enforcing the periodic data abstraction may become prohibitive.

As a result of these scaling issues, there has been recent interest in developing *co-design* frameworks where the concerns of real-time systems and control systems engineers are treated in a unified manner. One of the first statements of the co-design problem was given by Seto et al. [66]. This work presented co-design as an optimization problem that sought to minimize a traditional quadratic integral measure of control cost subject to task schedulability constraints. Seto's problem was an off-line design approach since the optimization problem was solved prior to system deployment. Since that time, a number of other co-design approaches have been suggested. A number of promising methods were listed in a paper by Arzen et al. [3]. Since that time a number of research scientists have investigated the methods on this list. These methods include feedback modification of task attributes [8, 45, 9, 12] , anytime controllers [7, 21] , and event-triggered sampling[2, 70, 83] .

One approach to co-design involves adjusting task attributes through feedback. An example of this is found in the elastic scheduling method [8] of Buttazzo et al. This method uses measured task execution times to adaptively adjust task periods. Lu et al. [45] presented a feedback control approach to real-time scheduling. This idea was later applied to the scheduling of control tasks by Caccamo et al. [9] and Cervin et al. [12]. This work clearly demonstrated that feedback control principles could be used to reduce the sensitivity of real-time systems to uncertainties in control task period, jitter, and execution time. The reduction in real-time system sensitivity also leads to improved control system performance, since one no longer needs to design the real-time system for the worst-case variation in jitter and execution time.

While these early schemes used feedback about the real-time system's performance to adjust task attributes, this feedback was not directly based on the control system's measured performance. A more direct link between real-time system and control system performance will be found in recent work examining *anytime* controllers and *event-triggered* sampling. Anytime controllers are control systems that adjust their structure based on the performance of the real-time system [7, 21]. In other words, if the real-time system becomes overloaded, then the application will select a less complex (though stabilizing) controller to execute. In this way, the controller's performance is directly tied in an intelligent way to the real-time system's performance.

Event-triggered controllers, on the other hand, adapt the real-time system's task period directly in response to the application's performance [2]. Under event-triggering the control task is only executed when the application's error signal ex-

ceeds a specified threshold. Ostensibly, this error provides a measure of how valuable the current state is to the overall system's closed-loop behavior. In this way the real-time system is only used when it is essential for maintaining the system's performance. Since the system state is always changing, this approach generates a sporadic sequence of controller invocations. In general, the hope is that the average rate of this sporadic task set will be much lower than the rate of a comparable periodic task set.

There is experimental evidence to support the assertion that event-triggered feedback improves overall control system performance while reducing the real-time system's use of computational resources. Two examples are shown in figure 1.1 which shows results from [4] and [63, 65].

The left-hand plot in figure 1.1 shows a plot from [4]. This paper considers a controlled scalar diffusion process of the form,

$$dx = axdt + udt + dw,$$

where a is a real constant and w is a standard Brownian motion. The signal u is the control signal generated by a full-state controller. This control is computed in either a periodic or event-triggered manner. Under event-triggering, the control is updated whenever the state magnitude, $|x|$, exceeds a specified threshold. The performance of the system is characterized by the steady-state variance of the system state. The variance of the periodically triggered system is denoted as V_R whereas the variance of the event-triggered system is denoted as V_L. The left-hand plot in figure 1.1 plots the ratio V_R/V_L as a function of the mean sampling period, T. Note that for all choices of the system constant, a, this performance ratio is greater than one, thereby showing that the event-triggered system has better performance than periodically triggered systems operating at the same mean sampling period.

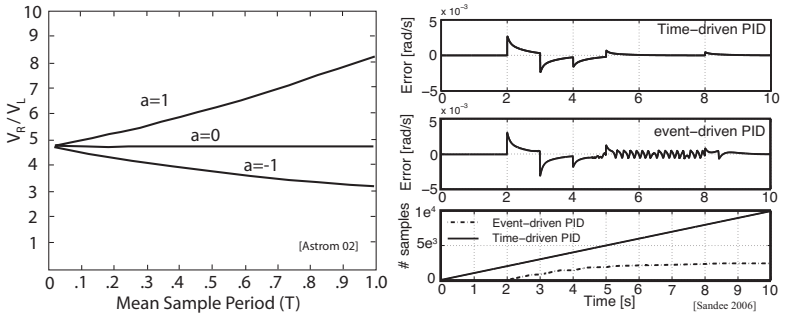


Fig. 1.1 Experimental results demonstrating that event-triggered feedback reduces a real-time system's use of computational resources while providing good overall control system performance

The right-hand side of figure 1.1 shows another example in which an event-triggered system demonstrates lower usage of computational resources. This result

is taken from [63, 24] which considers the control of a linear plant under a PID controller. This controller is discretized at a specified sampling rate and the resulting tracking error is plotted as a function of time in the top plot on the right-hand side of figure 1.1. The middle plot shows the tracking error for a comparable event-triggered implementation of the system. In this case the control is recomputed when the *gap* between the current system state and the last sampled system state exceeds a specified threshold, e_T,

$$\text{gap} = |x(t) - x(r_j)| \leq e_T = \text{threshold},$$

where r_j denotes the j^{th} consecutive time when the state was sampled. For the simulation shown in the middle plot, the threshold e_T was chosen to match the peak error of the periodically triggered system. This means that these first two plots are comparing the behavior of an event-triggered and periodically triggered system having similar performance levels. The bottom plot in the figure shows the number of samples that were generated by the periodically triggered (time-driven) and event-triggered PID control. As can be seen from this plot, the number of event-triggered samples is smaller than the time-driven control. Moreover, as the system approaches its equilibrium point, the number of samples begins to level off, thereby suggesting that as the information content within the error signal decreases, the controller needs to be invoked less often.

The left-hand example shown in figure 1.1 suggests an event-triggered system will perform better than a periodically-triggered systems with similar computational usage. The right-hand example suggests an event-triggered system will use fewer computational resources than a periodically triggered system with similar performance levels. These results, unfortunately, are only empirical in nature. The objective of this chapter is to review prior work that provides a more complete analysis of the relationship between performance and computation in event-triggered feedback systems.

Event-triggering samples the system state when some measure of the *novelty* in the state exceeds a given threshold. This approach to sampling has been around for quite awhile. Early examples of event-triggered systems may be found in relay [73] and pulse-width modulated feeback [54]. Event-triggered feedback has been used in reaction jet control of spacecraft. More recent examples have examined event-triggering in systems using motors [23, 65, 64, 24]. A comparison of the performance of event-triggered systems against periodically triggered systems may be found in [26]. Event-triggering has also appeared under a variety of other names such as interrupt-based feedback [29], Lebesgue sampling [4], asynchronous sampling [76], self-triggered feedback [75], state-triggered feedback [71], and level crossing sampling [39].

While event-triggering has been around for quite awhile it has only been in recent years [2] that researchers have made significant advances in understanding the event-triggering process. Sampled stochastic differential equations have been used to study event-triggered sampling [4]. This model has also been used to study event-triggered control [5, 27]. Optimal control and estimation in these event-triggered

stochastic systems was studied in [87] for infinite horizons. The results from [87] determine event triggers that maximize control/estimator performance subject to a soft constraint on communication usage. The resulting optimal event-triggers take the form of static thresholds. These results were extended to finite horizon event-triggered systems for control [31, 62] and estimation [30, 58, 57, 60]. These finite-horizon results optimize control/estimator performance subject to hard communication constraints. For estimation problems, the resulting optimal event-triggers are time-varying and for control problems, the event-triggers are time-varying functions of the initial system state.

Much of the aforecited work, however, focused only on scalar systems due to the computational complexity associated with solving the associated dynamic programming equations. Research scientists have recently been using state-based methods that can be more easily applied to vector systems. Making use of the emulation method [50] in sampled-data systems, recent work has identified sufficient sampling conditions that preserve closed-loop stability concepts such as input-to-state stability [70] or $\mathscr{L}_2$ stability [83]. A similar state-based approach has also been proposed in [40]. This recent work again derives state-dependent thresholds for the event-triggers. While the recent state-based methods do not explicitly constrain communication usage, experimental studies suggest that state-dependent event-triggers can be very effective in reducing an embedded system's usage of computational and communication resources.

This chapter discusses how event-triggering can be used in a wide range of networked control applications; ranging from control to estimation to optimization. In all of these application areas, event-triggering appears to greatly reduce the communication and/or computational effort required of the supporting real-time system. The remainder of this chapter is organized as follows. The chapter first reviews some mathematical preliminaries in section 1.2. Section 1.3 examines state-based event-triggering in embedded single processor control systems. The results from this section are then extended to networked control systems in section 1.4. The controllers in sections 1.3-1.4 all use full state feedback. As a first step towards developing output-feedback controls, section 1.5 examines a recent approach to event-triggered state estimation. Finally section 1.6 presents a novel application of event-triggering in distributed optimization of networked systems. Event-triggered control is still an active research area and a number of promising future research directions are discussed in section 1.7.

1.2 Mathematical Preliminaries

The event-triggers in this chapter are designed to enforce a variety of *stability concepts* found in the system science literature. This section reviews those stability concepts primarily to establish notational conventions that are followed throughout the rest of this chapter. In particular, this section reviews stability concepts such as

asymptotic stability, input-to-state stability, and $\mathscr{L}_2$ stability. Much of this material may be found in textbooks [32, 74, 37].

This chapter adopts the following notational conventions. The function x mapping elements on the real line, $\mathbb{R}$, onto elements of Euclidean n-space, $\mathbb{R}^n$, is denoted as $x : \mathbb{R} \to \mathbb{R}^n$. Let $x(t)$ denote the value that this function takes at time $t \in \mathbb{R}$ and let $\dot{x}(t) = dx(t)/dt$ denote the time derivative of x at time t. This function is said to solve an initial value problem of the form

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \tag{1.1}$$

if the above equations are satisfied for almost all $t \geq 0$. In equation (1.1), $f : \mathbb{R}^n \to \mathbb{R}^n$ is a function mapping Euclidean n-space back onto itself. To assure the above equation has unique solutions, one often requires that f be *Lipschitz continuous*. Namely, there exists a positive real constant L such that

$$\| f(x) - f(y) \| \leq L \| x - y \|$$

for all x and y in $\mathbb{R}^n$. The vector $x(t) \in \mathbb{R}^n$ is an element of a normed vector space where $\|x(t)\|$ denotes the usual Euclidean 2-norm. The function x is a member of a normed linear space. Important norms used for such functions are the supremum norm, $\|x\|_{\mathscr{L}_\infty} = \operatorname{ess\,sup}_t \|x(t)\|$ and the 2-norm $\|x\|_{\mathscr{L}_2} = \sqrt{\int_0^\infty \|x(\tau)\|^2 d\tau}$. Let $\mathscr{L}_2$ denote the linear space of all measurable functions with bounded 2-norms. $\mathscr{L}_\infty$ denotes the linear space of all measurable functions with bounded supremum norm. A function $\alpha : \mathbb{R} \to \mathbb{R}$ is said to be class $\mathscr{K}$ if it is continuous, strictly increasing, and $\alpha(0) = 0$. A function $\beta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is said to be of class $\mathscr{KL}$ if it is a continuous function that is class $\mathscr{K}$ with respect to the first argument and decreasing asymptotically to zero with respect to the second argument.

With these notational conventions established one can now define a variety of stability concepts. One of the best known stability concepts is *Lyapunov stability*. This concept applies to homogeneous systems characterized in equation (1.1). Given such as system, one says that a point $\overline{x} \in \mathbb{R}^n$ is an *equilibrium point* if $0 = f(\overline{x})$; in other words $\overline{x}$ represents a fixed point of the system. Without loss of generality, one can presume the equilibrium point $\overline{x} = 0$ lies at the origin.

The concept of Lyapunov stability is a property of the system's equilibrium point. In particular one says that the equilibrium point, $\overline{x} = 0$, is stable in the sense of Lyapunov if for all $\varepsilon > 0$ there exists $\delta > 0$ such that for all $t \geq 0$

$$\|x(0)\| < \delta \Rightarrow \|x(t)\| < \varepsilon.$$

Essentially, this means that the equilibrium point is Lyapunov stable if there always exists an initial condition that permits us to confine the system state within an arbitrarily small neighborhood of the equilibrium point.

A somewhat stronger (and better known) notion of Lyapunov stability is *asymptotic stability*. An equilibrium point is said to be asymptotically stable if the point is Lyapunov stable and if the state $x(t)$ asymptotically approaches the equilibrium point as t goes to infinity.

The existence of a *Lyapunov function* provides a well known sufficient condition for Lyapunov (asymptotic) stability. Consider a homogeneous system $\dot{x}(t) = f(x(t))$ with equilibrium point $\overline{x} = 0$. One says a continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is a Lyapunov function for the system if V is a positive definite function and its directional derivative, $\dot{V} = \frac{\partial V}{\partial x} f(x)$, is negative semi-definite. The existence of a Lyapunov function V is sufficient to show that the equilibrium point is stable in the sense of Lyapunov. Moreover, if one can strengthen the condition on $\dot{V}$ to be negative definite, then this suffices to establish that the equilibrium point is asymptotically stable.

While the Lyapunov stability concept has been widely used, it cannot be directly used to characterize the behavior of inhomogeneous systems whose state trajectories $x : \mathbb{R} \to \mathbb{R}^n$ satisfy the initial value problem,

$$\dot{x}(t) = f(x(t), w(t)), \quad x(0) = x_0. \tag{1.2}$$

where $w : \mathbb{R} \to \mathbb{R}^m$ is an external disturbance. In this case $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ maps the current system state, $x(t)$, and an external disturbance, $w(t)$, onto the state's time derivative. Because this system is driven by an external disturbance, one cannot usually identify a single equilibrium point for the system. Without this equilibrium point, one cannot use Lyapunov stability concepts to study the system's behavior. This observation motivates a variety of other stability concepts for such inhomogeneous systems. Two such stability concepts are input-to-state stability and $\mathscr{L}_2$ stability.

The system in equation (1.2) is input-to-state stable (ISS) if there exists a class $\mathscr{KL}$ function β and a class $\mathscr{K}$ function γ such that for any initial condition, $x(0) = x_0$, the response under any input $w \in \mathscr{L}_\infty$ satisfies

$$\|x(t)\| \leq \beta(\|x_0\|, t) + \gamma(\|w\|_{\mathscr{L}_\infty})$$

for all $t \geq 0$. An alternative and equivalent characterization of ISS is that the system response satisfies

$$\|x(t)\| \leq \max\{\beta(\|x_0\|, t), \gamma(\|w\|_{\mathscr{L}_\infty})\}$$

for all $t \geq 0$. Both definitions essentially require that the transient and steady state behaviors of the system are appropriately bounded. This view of the ISS-concept is illustrated in figure 1.2. The dashed line shows the bound due to the class $\mathscr{KL}$ function β acting on the initial transient portion of the system's response. The dotted line shows the bound due to the class $\mathscr{K}$ function γ acting on the steady-state portion of the system's response. To be ISS, the system's response must lie below the pointwise maximum of both of these comparison functions.

Input-to-state stability can also be characterized using Lyapunov-type functions. In particular, one says that a continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is an *ISS-Lyapunov* function for the system in equation (1.2) if there exist class $\mathscr{K}$ functions $\underline{\alpha}$, $\overline{\alpha}$, γ, and β such that

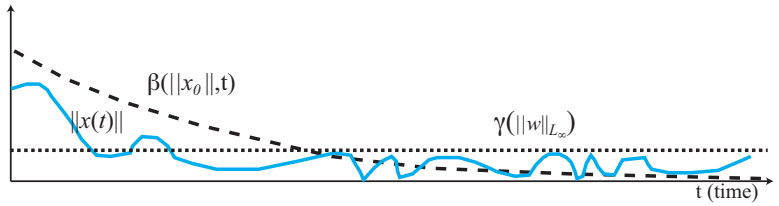


Fig. 1.2 Input-to-State Stability bounds the response's transient and steady-state behavior

$$\underline{\alpha}(\|x\|) \leq V(x) \leq \overline{\alpha}(\|x\|)$$
$$\dot{V}(x,w) \leq -\gamma(\|x\|) + \beta(\|w\|)$$

hold for $x \in \mathbb{R}^n$ and all $w \in \mathbb{R}^m$. The existence of an ISS-Lyapunov function for the system in equation (1.2) is necessary and sufficient for that system to be ISS.

$\mathscr{L}_2$ stability is another useful stability concept for inhomogeneous systems. In this case one usually thinks of the system as a mapping, $G : \mathscr{L}_2 \to \mathscr{L}_2$ between two normed linear spaces, $\mathscr{L}_2$. This means that if one is given an input $w \in \mathscr{L}_2$ then the system's output function Gw will also be a function in $\mathscr{L}_2$. The system G is finite-gain $\mathscr{L}_2$ stable (or just $\mathscr{L}_2$ stable) if there exist finite positive real constants γ and β such that

$$\|Gw\|_{\mathscr{L}_2} \leq \gamma \|w\|_{\mathscr{L}_2} + \beta. \tag{1.3}$$

The right-hand side of the above inequality represents an affine function that over-bounds the norm of the actual system's output. In particular, one can think of γ as a *gain* and β as an offset or *bias*. The so-called *induced gain* of G is then taken as the greatest lower bound on all of the possible γ's for which the above inequality holds. This *induced gain* is often denoted as $\|G\|$ and can be formally defined as

$$\|G\| = \inf \left\{ \gamma \in \mathbb{R} \ : \ \|Gw\|_{\mathscr{L}_2} \leq \gamma \|w\|_{\mathscr{L}_2} + \beta \right\}$$

for all $w \in \mathscr{L}_2$.

The induced gain provides an important way of defining a control system's performance. Many control synthesis problems can be formulated as so-called *regulator* problems in which the objective is to minimize the *gain* from the closed-loop system's uncontrolled external input to some output function. By making the induced gain of the closed-loop system sufficiently small, one provides some guarantee on the control system's performance level. The induced gain therefore becomes a direct way of characterizing overall control system performance.

When the inhomogeneous system in equation (1.2) has a special affine form then there is a useful characterization of the $\mathscr{L}_2$ induced gain. This characterization will be used later to design event-triggered systems that enforce the $\mathscr{L}_2$ stability concept. In particular, let's consider a special form of the inhomogeneous control system in which the state trajectory satisfies

$$\begin{aligned}\dot{x}(t) &= A(x(t)) + B_1(x(t))w(t) + B_2(x(t))u(t) \\ z(t) &= \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T\end{aligned} \tag{1.4}$$

where $x(0) = x_0$, $w : \mathbb{R} \to \mathbb{R}^p$ is an external $\mathscr{L}_2$ disturbance and $u : \mathbb{R} \to \mathbb{R}^m$ is a *control* signal that is generated by a controller $K : \mathbb{R}^n \to \mathbb{R}^m$. The functions $A : \mathbb{R}^n \to \mathbb{R}^n$, $B_1 : \mathbb{R}^n \to \mathbb{R}^{n\times p}$ and $B_2 : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ define how the system state, external input, and control map into the state's time derivative. The other signal $z : \mathbb{R} \to \mathbb{R}^{n+m}$ represents the system's *output* signal. The objective is to find a controller K such that the induced $\mathscr{L}_2$ gain from the external input w to the output z is less than a specified amount, γ.

The main result characterizing such a controller makes use of the so-called *Hamilton-Jacobi Inequality* (HJI). In particular, assume there exist a real constant $\gamma \geq 0$ and a positive definite continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ that satisfy the HJI,

$$\frac{\partial V}{\partial x}A(x) + \frac{1}{2}\frac{\partial V}{\partial x}\left[\frac{1}{\gamma^2}B_1(x)B_1^T(x) - B_2(x)B_2^T(x)\right]\frac{\partial V^T}{\partial x} + \frac{1}{2}x^T x \leq 0 \tag{1.5}$$

for all $x \in \mathbb{R}^n$. If one then selects the control output, u, so that

$$u = K(x) = -B_2^T(x)\frac{\partial V(x)^T}{\partial x} \tag{1.6}$$

then one can show that the closed-loop system's $\mathscr{L}_2$ gain is less than or equal to γ.

The bound on $\|G\|$ can be obtained as follows. The directional derivative of V is

$$\dot{V} = \frac{\partial V}{\partial x}A(x(t)) + \frac{\partial V}{\partial x}B_1(x(t))w(t) + \frac{\partial V}{\partial x}B_2(x(t))u(t).$$

Completing the square on the cross-term $\frac{\partial V}{\partial x}B_1(x)w$ and using the fact that $u = -B_2^T\frac{\partial V}{\partial x}^T$, yields

$$\dot{V} = \frac{\partial V}{\partial x}A - \frac{1}{2}\left\|\gamma w - \frac{1}{\gamma}B_1^T\frac{\partial V}{\partial x}^T\right\|^2 + \frac{1}{2\gamma^2}\frac{\partial V}{\partial x}B_1B_1^T\frac{\partial V}{\partial x} + \frac{1}{2}\gamma^2\|w\|^2 - \|u\|^2.$$

Making use of the Hamilton-Jacobi inequality, one can bound $\dot{V}$ as

$$\begin{aligned}\dot{V} &\leq -\frac{1}{2}\left\|\gamma w - \frac{1}{\gamma}B_1^T\frac{\partial V}{\partial x}^T\right\|^2 - \frac{1}{2}\|u\|^2 + \frac{1}{2}\gamma^2\|w\|^2 - \frac{1}{2}\|x\|^2 \\ &\leq -\frac{1}{2}(\|u\|^2 + \|x\|^2 - \gamma^2\|w\|^2).\end{aligned}$$

Since $z = \begin{bmatrix} x \\ u \end{bmatrix}$, this implies that

$$\dot{V} \leq -\frac{1}{2}\|z\|^2 + \frac{1}{2}\gamma^2\|w\|^2.$$

If one then integrates the above inequality from 0 to infinity, one can readily use the definition of the $\mathscr{L}_2$-norm to see that

$$\|z\|_{\mathscr{L}_2} \leq \gamma\|w\|_{\mathscr{L}_2} + \sqrt{2V(x(0))}.$$

This inequality is precisely what was seen in equation (1.3) which is sufficient to imply the closed-loop system is $\mathscr{L}_2$ stable with an induced gain less than or equal to γ. In other words, if one choses the control as stated in equation (1.6), then one can guarantee that the $\mathscr{L}_2$ performance level achieved by the closed-loop system is less than or equal to γ.

As mentioned at the opening of this section, this chapter derives event-triggers that preserve input-to-state stability or $\mathscr{L}_2$ stability concepts. The preceding definitions and derivations will be used later in deriving these event-triggers. Let's now turn to see precisely how such event-triggers would be derived for both embedded control and networked control systems.

1.3 Event-Triggered Feedback in Embedded Control Systems

This section discusses the design of event-triggering schemes for embedded control systems. The main idea is to first design a continuous-time controller that guarantees a stability concept such as input-to-state stability or $\mathscr{L}_2$ stability. The section then develops an *event-triggering threshold* such that the associated sporadically triggered control system preserves this underlying stability concept. This approach is sometimes called the *emulation-based* method [50].

Sampled-Data System Model: Let's first consider how a sampled-data system might be configured. Figure 1.3 shows a block diagram for the system under study. The *plant* (G) has two types of inputs. There is an external *uncontrolled* disturbance, $w : \mathbb{R} \to \mathbb{R}^q$ and a control input, $u : \mathbb{R} \to \mathbb{R}^m$. The plant's state, $x : \mathbb{R} \to \mathbb{R}^n$, satisfies the inhomogeneous differential equation

$$\dot{x}(t) = f(x(t), u(t), w(t))$$

for $t \geq 0$ with initial condition $x(0) = x_0 \in \mathbb{R}^n$. The output of the plant is the system state, x.

Rather than working directly with the continuous-time state, x, the controller works with a sampled version of the state trajectory. In particular, let's introduce a *sampler* (S) system that is characterized by a monotone increasing sequence of *sampling instants*. This sequence of sampling instants is denoted as $\{r_j\}_{j=0}^{\infty}$ where $r_j > r_{j-1}$ for $j = 1, 2, \ldots, \infty$. The time $r_j \in \mathbb{R}$ denotes the j^{th} consecutive sampling instant. The output of the sampler is therefore a sequence of sampled states, $\{\hat{x}_j\}$,

in which $\hat{x}_j = x(r_j)$. A state-feedback controller, $K : \mathbb{R}^n \to \mathbb{R}^m$, maps the sampled state onto a control vector $\hat{u}_j \in \mathbb{R}^m$. The resulting sequence $\{\hat{u}_j\}_{j=0}^{\infty}$ of controls is then transformed into a continuous-time signal through a zero-order hold (H) without any delay. The control signal, $u \in \mathbb{R} \to \mathbb{R}^m$, used by the plant is a piecewise constant function. In particular, let's introduce a sequence of functions $\tilde{u}_j : \mathbb{R} \to \mathbb{R}^m$ that have support over the time interval $[r_j, r_{j+1})$. The value of $\tilde{u}_j$ at time $t \in \mathbb{R}$ is

$$\tilde{u}_j(t) = \begin{cases} \hat{u}_j & \text{for } t \in [r_j, r_{j+1}) \\ 0 & \text{otherwise} \end{cases}$$

for $j = 0, 1, \ldots, \infty$. With regard to this sequence the controlled input feeding the plant simply becomes $u(t) = \sum_{j=0}^{\infty} \tilde{u}_j(t)$.

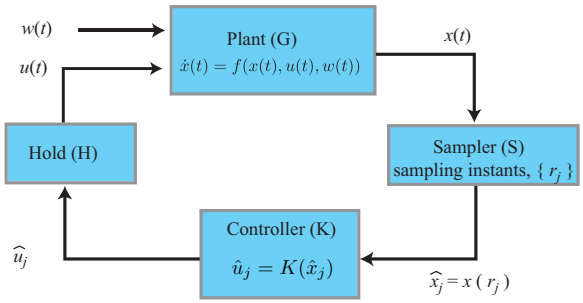


Fig. 1.3 Sampled Data Control System

ISS Event-Triggers: Under the *emulation-based* approach for developing sampled-data systems, one assumes that the controller, K, enforces a specified stability concept. In particular, let's confine our attention to input-to-state stability and let's consider the *continuously* sampled closed-loop system,

$$\dot{x}(t) = f(x(t), K(x(t) + e(t)), w(t))$$

where $e : \mathbb{R} \to \mathbb{R}^n$ and $w : \mathbb{R} \to \mathbb{R}^m$ are $\mathscr{L}_\infty$ input disturbances. Let's assume that the controller K leaves this closed-loop system ISS with respect to the two inputs w and e.

From our earlier discussion in section 1.2, the ISS assumption implies the existence of an ISS-Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ with class $\mathscr{K}$ functions $\underline{\alpha}$, $\overline{\alpha}$, γ, β_1 and β_2 such that

$$\underline{\alpha}(\|x\|) \leq V(x) \leq \overline{\alpha}(\|x\|) \tag{1.7}$$

$$\frac{\partial V}{\partial x} f(x, K(x+e), w)) \leq -\gamma(\|x\|) + \beta_1(\|e\|) + \beta_2(\|w\|). \tag{1.8}$$

The inequalities in equation (1.7) essentially require that V is positive definite. The inequality in equation (1.8) is a dissipative relation on the ISS-Lyapunov function's directional derivative.

Now let's consider the sampled-data version of this system. The sampler generates a sequence of sampling instants, $\{r_j\}_{j=0}^{\infty}$. The time r_j is referred to as the j^{th} consecutive *release time* of the system. (This term refers to the fact that in a real-time computer system, state sampling is implemented through a task that is released for execution at time r_j). The sampled states $\{\hat{x}_j\}_{j=0}^{\infty}$ form a sequence in which $\hat{x}_j = x(r_j)$. Let's define the *gap* function associated with the j^{th} sampling time as a function $e_j : [r_j, r_{j+1}) \to \mathbb{R}^n$ in which

$$e_j(t) = \hat{x}_j - x(t)$$

for $t \in [r_j, r_{j+1})$ where $j = 0, 1, \ldots, \infty$.

The sampled data system's controller uses $\hat{x}_j = e_j(t) + x(t)$, rather than $x(t)$, so the sampled-data system's state must satisfy

$$\dot{x}(t) = f(x(t), K(x(t) + e_j(t)), w(t))$$

for all $t \in [r_j, r_{j+1})$ and all $j = 0, 1, \ldots, \infty$. Under the ISS assumption, one knows that

$$\dot{V} \leq -\gamma(\|x(t)\|) + \beta_1(\|e_j(t)\|) + \beta_2(\|w(t)\|). \tag{1.9}$$

Let's assume the gap can be restricted so that for some $\sigma \in (0, 1)$

$$\beta_1(\|e_j(t)\|) \leq \sigma\gamma(\|x(t)\|) \tag{1.10}$$

for all $t \geq 0$ and all $j = 0, 1, \ldots, \infty$. Inserting equation (1.10) into equation (1.9) implies

$$\dot{V} \leq -(1 - \sigma)\gamma(\|x(t)\|) + \beta_2(\|w(t)\|).$$

In light of the characterization of input-to-state stability (Sec. 1.2) and since $0 < \sigma < 1$, it should be apparent that enforcing the constraint on the gap (equation (1.10)) leaves the sampled-data system input-to-state stable with respect to the input w.

The constraint in equation (1.10) can be viewed as a state-dependent threshold condition. In particular, one knows that at the beginning of the interval $[r_j, r_{j+1})$, that the gap $e_j(r_j) = 0$. After that, one expects the norm of the gap to increase. When the gap satisfies the inequality $\beta_1(\|e_j(t)\|) > \sigma\gamma(\|x(t)\|)$, then the system state is again *sampled* by setting $\hat{x}_j = x(t)$, thereby forcing the gap to zero again. In this way the condition in equation (1.10) can be viewed as an *event-trigger*. This event-trigger would be realized by the sampler, S. In particular, one would require the sampler to continuously monitor the inequality in equation (1.10). Upon detecting a violation of the inequality, the sampler would trigger the sampling of the system state. The resulting time history of the threshold $\gamma(\|x(t)\|)$ and the gap is shown

below in figure 1.4. As the state asymptotically approaches the origin, the threshold gets smaller. This *state-dependent* threshold idea and the above analysis underlying the ISS event-trigger was first discussed by Tabuada [70].

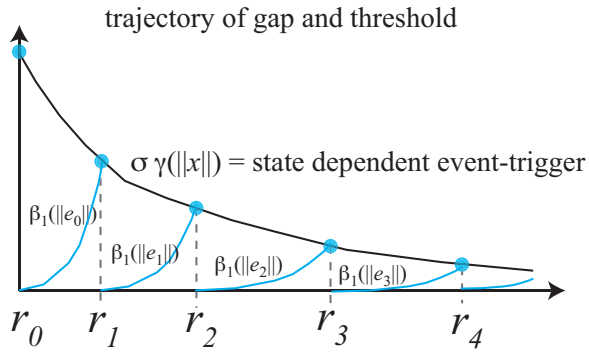


Fig. 1.4 Time history of gap and event threshold

Let's look at a simple example to see how well the ISS event-trigger works. Consider a process model (without the external disturbance w) of the form

$$\begin{aligned} \dot{x}(t) &= f(x(t)) + u(t) \\ u(t) &= -2f(\hat{x}_j) \end{aligned}$$

for $t \in [r_j, r_{j+1})$. The release times r_j are selected as the times when the *event-trigger* is violated. The ISS event-trigger is chosen to have the following form,

$$\beta_1(\|e_j(t)\|) = e_j^2(t) \leq x^2(t) = \gamma(\|x(t)\|).$$

The system function $f : \mathbb{R} \to \mathbb{R}$ is chosen so the proposed control leaves the closed-loop system input-to-state stable. In particular, let's consider three different types of system dynamics. The chosen system dynamics have sublinear, linear, and superlinear f functions of the forms,

sublinear	linear	superlinear
$f(x) = \text{sgn}(x)\sqrt{\lvert x\rvert}$	$f(x) = x$	$f(x) = x^3$

The plots below in figure 1.5 show the system response for the linear and superlinear choices for f. The top graphs plot the gap, $\beta_1(\|e(t)\|)$, and the threshold $\gamma(\|x(t)\|)$ as a function of time for both cases. This response is plotted on a logarithmic axis. For both linear and superlinear cases one sees that the gap satisfies the basic form seen earlier in figure 1.4. The bottom plots show the intersample time, $T_j = r_j - r_{j-1}$, for both cases. For the linear f, the choice of event-trigger and f yields a periodic sampling of the system state. The case with the superlinear case f shows that the intersample time gets longer as the system state approaches the equilibrium point of the unforced system.

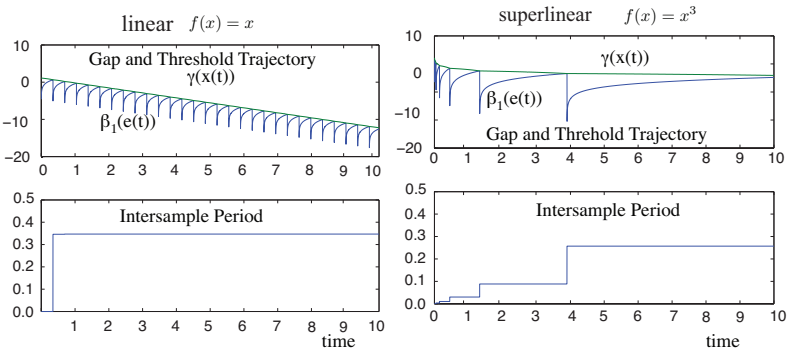


Fig. 1.5 Two examples showing gap, threshold, and intersampling time history. (left) linear function $f(x) = x$. (right) superlinear function $f(x) = x^3$

An interesting behavior is seen if f is the sublinear function $f(x) = \text{sgn}(x)\sqrt{|x|}$. The gap and intersample time histories for this case are shown in figure 1.6. In this case, the intersample times get shorter and shorter as the system approaches its equilibrium point. Asymptotically these time intervals go to zero at a finite time around 3.5 seconds into the simulation. This type of behavior is sometimes called a *Zeno* behavior. Zeno-sampling is highly undesirable in real-time control for it would require the computer to eventually sample infinitely fast.

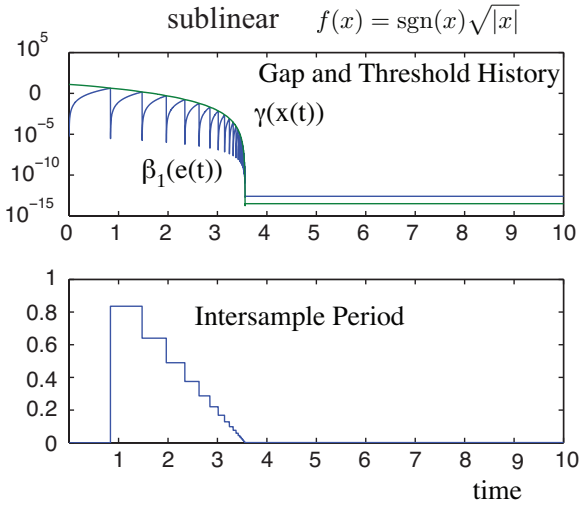


Fig. 1.6 Gap, threshold, and intersampling time history for a sublinear dynamical system where $f(x) = \text{sgn}(x)\sqrt{|x|}$.

Avoiding Zeno-sampling: To better understand when Zeno-sampling might occur, let's try to derive a lower bound on the event-triggered system's intersample time. Let's first assume that the closed-loop system is Lipschitz with respect to the state x and the gap e. In other words, there exists a positive constant L such that for all x and e in $\mathbb{R}^n$,

$$\| f(x,K(x+e))\| \leq L\|x\| + L\|e\|.$$

If the j^{th} gap function, e_j, violates the event-trigger in equation (1.10) at time r_{j+1}, then

$$\beta_1(\|e_j(r_{j+1})\|) > \sigma\gamma(\|x(r_{j+1})\|).$$

Let's assume there exists a positive constant P such that

$$P\|e_j(r_{j+1})\| \geq \gamma^{-1}\left(\frac{1}{\sigma}\beta_1(\|e_j(r_{j+1})\|)\right) \geq \|x(r_{j+1})\|.$$

It can therefore be concluded that with these constraints, the ratio of the gap and the system state must be greater than a positive constant $1/P$. In other words, the next sample occurs when

$$\frac{1}{P} \leq \frac{\|e_j(t)\|}{\|x(t)\|}. \tag{1.11}$$

This condition is a more conservative than the original event-trigger in equation (1.10). It is useful, however, because it provides an analytically tractable method of bounding the earliest time when the event-trigger can occur. As long as this earliest sampling time can be shown to be bounded away from zero, then one can assure that Zeno-sampling doesn't occur.

For any $j = 0,1,\ldots,\infty$, the trajectory for $\|e_j(t)\|/\|x(t)\|$ can be bounded through the use of differential inequalities. A direct computation of the ratio's time derivative shows that

$$\frac{d}{dt}\frac{\|e_j(t)\|}{\|x(t)\|} \leq \left(1+\frac{\|e_j(t)\|}{\|x(t)\|}\right)\frac{L\|x(t)\|+L\|e_j(t)\|}{\|x(t)\|} = L\left(1+\frac{\|e_j(t)\|}{\|x(t)\|}\right)^2.$$

This differential inequality is used in the Comparison principle [37] to obtain an upper bound on the time history of the event quotient $\|e_j(t)\|/\|x(t)\|$. This bound takes the form

$$\frac{\|e_j(t)\|}{\|x(t)\|} \leq \frac{tL}{1-tL}$$

for t beween 0 and $r_{j+1} - r_j$ where $j = 0,1,\ldots,\infty$.

So one merely needs to see when the right-hand side of the above inequality triggers the event quotient condition in equation (1.11). This occurs if the next release

time r_{j+1} satisfies

$$\frac{1}{P} \leq \frac{\|e_j(r_{j+1})\|}{\|x(r_{j+1})\|} \leq \frac{T_j L}{1 - T_j L}$$

where $T_j = r_{j+1} - r_j$ is the intersample time interval. Solving the right-hand inequality for T_j yields a lower bound of the form

$$r_{j+1} - r_j = T_j \geq \frac{1}{L + LP}.$$

Note that this is a lower bound on the intersample time. So as long as the bound is non-zero one can guarantee that the event-triggered system won't exhibit Zeno-sampling. Clearly this bound goes to zero when L is unbounded. In other words, this occurs when the system function f fails to be Lipschitz. In reviewing the sublinear example where Zeno sampling occurs, it is apparent that the sublinear function $\text{sgn}(x)\sqrt{|x|}$ is not Lipschitz. These results show that ISS event-triggering can guarantee non-Zeno sampling of the system state whenever f is Lipschitz.

The sampling generated under these conditions is sporadic rather than aperiodic. Aperiodic sampling simply means that the intersample interval T_j is not a constant. Being aperiodic, however, doesn't require that the minimum intersample interval is positive. Following notational conventions in real-time computing, the term *sporadic* is reserved for systems whose intersample intervals need not be constant and whose minimum intersample intervals are positive.

$\mathscr{L}_2$ Event-Triggers: The prior subsection derived sporadic event-triggers that preserve the input-to-state stability of the original non-sampled control system. This framework [70] places relatively few assumptions on the nature of the controller. It only requires that the controller has an ISS-Lyapunov function to ensure input-to-state stability with regard to both the external disturbance, w, and the state gap, e_j. If one makes some assumptions about the structure of the controller, it is possible to say a bit more about the robustness of the closed-loop system's stability concept with regard to non-zero delays, or what is sometimes referred to as *jitter* in the real-time systems community.

This subsection derives event-triggers that preserve the $\mathscr{L}_2$ stability of the closed-loop system. In particular, these so-called $\mathscr{L}_2$ event triggers guarantee that the closed-loop system's induced $\mathscr{L}_2$ gain is preserved (up to a user-defined scaling factor). The so-called $\mathscr{L}_2$ event-trigger was first proposed by Lemmon et al. [41] and then formally analyzed by Wang et al. [83]. Since these $\mathscr{L}_2$ event-triggers preserve the original non-sampled system's closed-loop gain, one can say that these event-triggers are *performance preserving* since the $\mathscr{L}_2$ gain is a commonly used measure of a regulator's performance.

By focusing on the $\mathscr{L}_2$ stability concept, one can use the aforecited results relating the closed-loop system $\mathscr{L}_2$ gain to a Hamilton-Jacobi inequality. To use this relationship, let's narrow our attention to systems that are affine in the external disturbance, w, and the control u. In particular, let's assume that the system state,

$x : \mathbb{R} \to \mathbb{R}^n$ satisfies the following differential equation

$$\dot{x}(t) = A(x(t)) + B_1(x(t))w(t) + B_2(x(t))u(t) \tag{1.12}$$

for $t \geq 0$ and initial condition $x(0) = x_0 \in \mathbb{R}^n$. As before $w : \mathbb{R} \to \mathbb{R}^q$ is an external disturbance that is assumed to lie in $\mathscr{L}_2$. The control signal, $u : \mathbb{R} \to \mathbb{R}^m$ is assumed to be a special control of the form

$$u(t) = -B_2^T(x(t)) \left[\frac{\partial V(x(t))}{\partial x} \right]^T = K(x(t))$$

where $V : \mathbb{R}^n \to \mathbb{R}$ is a continuously-differentiable positive definite function (sometimes called the *storage function* [37]) that satisfies the Hamilton-Jacobi inequality (1.5) for some $\gamma > 0$. For this particular control, one can show that the $\mathscr{L}_2$ gain of the system from the input w to the output $z = \begin{bmatrix} x \\ u \end{bmatrix}$ is less than or equal to γ. These results were summarized in the earlier section on mathematical preliminaries.

The event-triggered version of the above system starts by introducing a sequence of *release* or *sampling* times $\{r_j\}_{j=0}^{\infty}$ where $r_j \in \mathbb{R}$ denotes the j^{th} consecutive time when the system state has been sampled. In this case the control law uses the *sampled* state instead of the true state so that $u(t)$ becomes

$$u(t) = -B_2^T(\hat{x}_j) \left[\frac{\partial V(\hat{x}_j)}{\partial x} \right]^T = K(\hat{x}_j)$$

for $t \in [r_j, r_{j+1})$ and $j = 0, 1, \ldots, \infty$. In the above equation $\hat{x}_j \in \mathbb{R}^n$ denotes the j^{th} consecutive sampled state

$$\hat{x}_j = x(r_j).$$

As before let's introduce the *gap* between the current state $x(t)$ and the previously sampled state. The j^{th} gap function therefore is

$$e_j(t) = \hat{x}_j - x(t)$$

for $t \in [r_j, r_{j+1})$ and all $j = 0, 1, \ldots, \infty$. Assume that the controller $K : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz with respect to the gap. In other words, there exists a non-negative real constant L such that

$$\|K(x) - K(\hat{x}_j)\| = \|K(x) - K(x + e_j)\| \leq L\|e_j\|. \tag{1.13}$$

This assumption is satisfied in many applications. In particular, the assumption is valid when the controller is affine with respect to the gap signal.

Let's now examine the time rate of change of the storage function V under the sampled control law. The directional derivative of V is

$$\dot{V} = \frac{\partial V(x)}{\partial x}A(x) + \frac{\partial V(x)}{\partial x}B_1(x)w + \frac{\partial V(x)}{\partial x}B_2(x)K(\hat{x}_j).$$

Since $K(x) = -B_2^T \frac{\partial V}{\partial x}^T$, one can rewrite $\dot{V}$ as

$$\dot{V} = \frac{\partial V}{\partial x}A(x) + \frac{\partial V}{\partial x}B_1(x)w - K^T(x)K(\hat{x}_j).$$

Completing the square for the cross-term $\frac{\partial V}{\partial x}B_2(x)w$ yields,

$$\dot{V} = \frac{\partial V}{\partial x}A(x) - \frac{1}{2}\left\|\gamma w - \frac{1}{\gamma}\frac{\partial V}{\partial x}^T\right\|^2 + \frac{\gamma^2}{2}\|w\|^2 + \frac{1}{2\gamma^2}\left\|B_1^T(x)\frac{\partial V}{\partial x}^T\right\|^2 - K^T(x)K(\hat{x}_j).$$

Applying the Hamilton-Jacobi inequality bounds $\dot{V}$ as was done in section 1.2 yields

$$\begin{aligned}\dot{V} &\leq -\frac{1}{2}\left\|\gamma w - \frac{1}{\gamma}\frac{\partial V}{\partial x}^T\right\|^2 + \frac{\gamma^2}{2}\|w\|^2 - \frac{1}{2}\|x\|^2 - K^T(x)K(\hat{x}_j) \\ &\leq -\frac{1}{2}\|x\|^2 + \frac{\gamma^2}{2}\|w\|^2 - K^T(x)K(\hat{x}_j). \end{aligned} \tag{1.14}$$

The above cross-term, $K^T(x)K(\hat{x}_j)$, in equation (1.14) must still be dealt with. From the Lipschitz assumption on K in equation (1.13), one rewrites this cross-term as

$$\begin{aligned}K^T(x)K(\hat{x}_j) &= \frac{1}{2}\left\|K(x) - K(\hat{x}_j)\right\|^2 - \frac{1}{2}\|K(x)\|^2 - \frac{1}{2}\|K(\hat{x}_j)\|^2 \\ &\leq \frac{1}{2}L^2\|e_j\|^2 - \frac{1}{2}\|K(x)\|^2 - \frac{1}{2}\|K(\hat{x}_j)\|^2 \\ &\leq \frac{1}{2}L^2\|e_j\|^2 - \frac{1}{2}\|K(\hat{x}_j)\|^2\end{aligned}$$

where $e_j = \hat{x}_j - x$ is the gap. Substituting this bound for $K^T(x)K(\hat{x}_j)$ into equation (1.14) yields the following bound on the directional derivative of the storage function,

$$\begin{aligned}\dot{V} &\leq -\frac{1}{2}\|x\|^2 + \frac{\gamma^2}{2}\|w\|^2 - \frac{1}{2}\|K(\hat{x}_j)\|^2 + \frac{1}{2}L^2\|e_j\|^2 \\ &= -\frac{\beta^2}{2}\|x\|^2 + \frac{\gamma^2}{2}\|w\|^2 + \left(-\frac{1-\beta^2}{2}\|x\|^2 - \frac{1}{2}\|K(\hat{x}_j)\|^2 + \frac{1}{2}L^2\|e_j\|^2\right)\end{aligned} \tag{1.15}$$

for some user-defined parameter $\beta \in [0,1]$. Note that the above inequality will be a dissipative inequality for $\dot{V}$ provided one can guarantee that the last three terms within the parentheses are collectively negative definite. If this is the case, then

$$\dot{V} \leq -\frac{\beta^2}{2}\|x\|^2 + \frac{\gamma^2}{2}\|w\|^2$$

for all x and w. As noted in the mathematical preliminaries section, satisfaction of this inequality is sufficient to establish that the sampled-data system's induced $\mathscr{L}_2$ gain from the input w to the output x is less than or equal to γ/β. Note that the user-defined parameter β becomes a parameter that controls how close the gain of the sampled-data system will be to the original gain of the continuously-sampled system.

In summary, if the last three terms on the right-hand side of equation (1.15) are negative, then the sampled-data system is $\mathscr{L}_2$ stable with a gain less than γ/β. This inequality is satisfied for all times, t, if one can guarantee

$$L^2\|e_j(t)\|^2 \leq (1-\beta^2)\|x(t)\|^2 + \|K(\hat{x}_j)\|^2. \tag{1.16}$$

The left-hand side of this inequality is simply the size of the system gap, e_j. The right-hand side of the inequality is a state-dependent threshold that is very similar to the ISS event threshold derived earlier. In other words equation (1.16) is an $\mathscr{L}_2$ preserving event-trigger. The event-trigger is used in the same way the ISS event-trigger was used. Namely, the sampler, S, monitors the gap against the threshold on the right-hand side. When the inequality is violated (or about to be violated), then the state is sampled and the next release time r_{j+1} is generated.

Note that the event-trigger depends on the user-defined parameter β. This parameter controls how close the event-triggered system's gain will be to γ, the gain of the original continuous-time system. If β is close to one then the sampled system achieves the original gain of γ. As β gets smaller, the gain of the system increases, thereby reducing the event-triggered system's $\mathscr{L}_2$ performance. In other words, the smallest thresholds and hence the most frequent sampling occurs when β is close to one. As β gets smaller, the intersampling periods will get longer at the cost of a higher closed-loop system gain. This is a tradeoff between the system gain and how frequently the state must be sampled.

The following example illustrates the use of the $\mathscr{L}_2$ event trigger on a variation of the superlinear system examined in figure 1.5. In this case let's consider the controlled system characterized by the following equations,

$$\begin{aligned}\dot{x}(t) &= x^3(t) + u(t) + w(t)\\ u(t) &= -\alpha\hat{x}_j^3 - \hat{x}_j\end{aligned}$$

for $t \in [r_j, r_{j+1})$ and all $j = 0, 1, \ldots, \infty$. The sequence of release times $\{r_j\}_{j=0}^{\infty}$ is generated by the violation of the $\mathscr{L}_2$ event-trigger in equation (1.16). Choose a special type of $\mathscr{L}_2$ disturbance of the form,

$$w(t) = e^{-2t}v(t)$$

for $t \geq 0$ and where v is white noise process. The left-hand side of figure 1.7 plots the gap and threshold time histories for this system (top plot) and the intersample time (bottom plot), T_j, that was generated. As can be seen the sampling period is initially very small (about 0.1 sec) at the beginning of the simulation when the disturbance

is largest. As the disturbance decays, the sampling period stabilizes to a relatively long period (0.4 sec) that is four times longer than the initial sampling period.

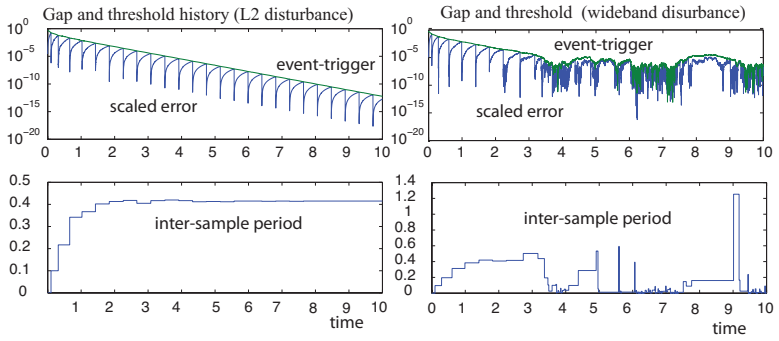


Fig. 1.7 Gap, threshold, and intersample time history using an $\mathscr{L}_2$ event-trigger. (left) $\mathscr{L}_2$ noise case. (right) wideband noise case.

The right-hand side of figure 1.7 shows the gap, threshold, and intersample time histories for the same $\mathscr{L}_2$ event-triggered system in which the disturbance is no longer guaranteed to go to zero as our system approaches the equilibrium point. In this case let the disturbance be

$$w(t) = (e^{-2t} + 0.1)v(t)$$

where $v(t)$ is again a white noise process. In this case, the disturbance amplitude does not go to zero as time goes to infinity. In particular, this $w(t)$ is referred to as a wide band disturbance. The top plot shows the gap and threshold time histories. The bottom plot shows the resulting intersample times. When the system state is far from the equilibrium point, the system's response is similar to the earlier $\mathscr{L}_2$ disturbance case. As the system state approaches the equilibrium point, however, the periodic nature of the intersampling time disappears with sampling times that can become arbitrarily short. In this case, therefore, event-triggering only yields aperiodic, rather than sporadic, sampling of the system state.

This type of behavior is common in both the $\mathscr{L}_2$ event-triggered and ISS event-triggered systems. It essentially results because the state-dependent threshold gets very small as the system state approaches the origin. With such a small threshold, the introduction of noise into the disturbance makes the system's sampling-events trigger much more often. This example therefore shows that state-dependent event-triggered system may be sensitive to wide-band disturbances. One way to address this sensitivity is to place a lower bound on the event-triggering threshold of the form (in the $\mathscr{L}_2$ event-trigger case)

$$L^2\|e_j(t)\|^2 \leq \max\left\{\overline{T}, (1-\beta^2)\|x(t)\|^2 + \|K(\hat{x}_j)\|^2\right\}.$$

Assuming that $\|e_j(t)\|$ and $\|x(t)\|$ have bounded rates of growth, this modified event-trigger prevents the sampling period from being arbitrarily close to zero and can therefore assure the sporadic nature of event-triggered sampling.

Impact of Delays: The preceding analysis for the $\mathscr{L}_2$ and ISS event-triggers assumed that both the system state and control update are generated at the same time. In other words, the control signal, based on the system state sampled at time r_j, is applied to the plant at the same time. This means that our prior analysis ignored delays. Real-life implementations of such systems will always exhibit some delay due to the amount of time it takes to compute the control signal. It would be highly desirable to show that the performance of the event-triggered system (as measured by the closed-loop system's $\mathscr{L}_2$ gain) is preserved under such delays. The following analysis from [83] shows how robust $\mathscr{L}_2$ performance will be under event-triggering with delays.

Before starting the analysis, let's discuss the modeling of event-triggered sampling with delays. In particular, one now needs to consider two sequences of times. The sequence of release times $\{r_j\}_{j=0}^{\infty}$ is defined as before. Release time r_j represents the time when 1) the state was sampled and 2) the control task was released for execution by the central processing unit (CPU). The other sequence of interest is the sequence of *finishing times*, $\{f_j\}_{j=0}^{\infty}$. The time $f_j \in \mathbb{R}$ denotes the time when the control signal computed by the control task is actually used by the plant. This time also marks the finish of the control job that had been released at time r_j. In general one make the *small delay* assumption which states that $r_j \leq f_j < r_{j+1}$ for all $j = 0, 1, \ldots, \infty$. In other words, the sample taken at time r_j is used at a time, f_j, which always occurs before the next invocation of the control task.

Figure 1.8 shows the timing relationships assumed in this analysis. The figure is a timeline in which the black rectangles indicate when the control task job is being executed. With regard to this diagram, let's define two measures of real-time system performance. The first measure is the task period $T_j = r_{j+1} - r_j$. This is the interval of time between any two consecutive invocations of the control task. As in the case of the ISS event-trigger, there is great interest in obtaining lower bounds on T_j, thereby identifying the smallest sampling period required by the real-time computer. The other measure of interest is the *delay*, D_j, of the j^{th} job. This is the time between the finishing time and release time, i.e. $D_j = f_j - r_j$. The control task *deadline* $\overline{D}$ is a real-time constraint that one might place on these delays. In particular, a real-time system that is functioning properly will have all delays less than the deadline ($D_j \leq \overline{D}$ for all $j = 0, 1, \ldots, \infty$). The choice of the deadline is an important constraint. In our case, the deadline is chosen to ensure the $\mathscr{L}_2$ performance of the control application. In particular, this means that our analysis would like to derive upper bounds on the maximally allowable delay (MAD) that any task can tolerate before losing our guarantee on $\mathscr{L}_2$ performance. This upper bound then becomes the deadline quality-of-service (QoS) constraint on the real time system.

To obtain tight bounds on the maximally allowable delays (MAD) and intersample intervals, let's confine our attention to linear time-invariant control systems of the form

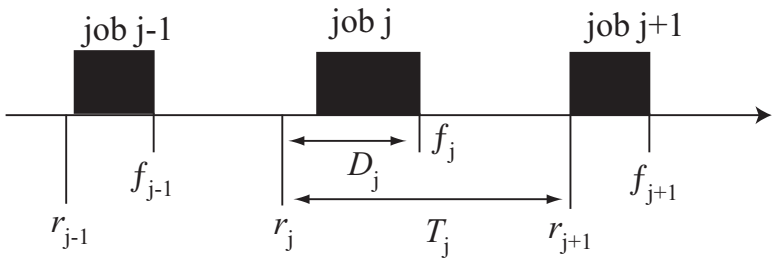


Fig. 1.8 Definition of Timing Relationships used in Studying the Real-time Implementations of $\mathscr{L}_2$ event-triggered control.

$$\dot{x}(t) = Ax(t) + w(t) + Bu(t)$$
$$u(t) = -B^T P\hat{x}_j = K(\hat{x}_j)$$

for all $t \in [f_j, f_{j+1})$ and $j = 0, 1, \ldots, \infty$. A and B are are suitably dimensioned real matrices. In terms of the earlier system model considered in equation 1.12, we let $B_1 = I$ and $B_2 = B$. This is simply done for notational convenience. The sampled state, $\hat{x}_j = x(r_j)$, is the state that occurs at the j^{th} consecutive release time. Note that the above equation holds between *finishing* times, rather than between release times. This is in accordance with the fact that control signals can only be changed after the control task's job has finished executing.

By confining our attention to linear systems one can use a storage function of the form $V(x) = x^T Px$ where P is a real valued n by n matrix. With this choice of V the Hamilton-Jacobi inequality reduces to an algebraic Riccati inequality where P is a symmetric positive definite matrix that for some real $\gamma > 0$ satisfies the Riccati inequality

$$A^T P + PA - P\left(BB^T - \gamma^{-2} I\right) P + I \leq 0.$$

With this choice of control, the induced $\mathscr{L}_2$ gain of the continuously sampled closed-loop system is guaranteed to be less than or equal to γ.

The $\mathscr{L}_2$ event-trigger is derived in the same way it was for the nonlinear system. The difference is that now in establishing the dissipative inequality, one makes use of the algebraic Riccati inequality rather than the Hamilton-Jacobi inequality. The resulting $\mathscr{L}_2$ event-trigger (derived in [83]) is

$$e_j^T(t) M e_j(t) < \delta x^T(r_j) N x(r_j)$$

where

$$M = (1-\beta^2) I + PBB^T P$$
$$N = \frac{1}{2}(1-\beta^2) I + PBB^T P$$

and β and δ are user supplied constants between 0 and 1. Note that the earlier $\mathscr{L}_2$ triggering threshold was a function of $x(t)$ and $x(r_j)$. The preceding threshold for the LTI case is only a function of $x(r_j)$. This means that the above threshold is *weaker* than our earlier result (in other words it would cause the system to sample sooner). This particular form of the $\mathscr{L}_2$ threshold was used in [83] because it rendered the analysis of delays more tractable. With this weaker threshold it was possible to obtain specific bounds on the acceptable delays and minimum periods that could be tolerated by the event-triggered system. As mentioned above, the bounds on delays are useful because they serve as deadlines for the real-time computer implementing the event-triggered control system. The bounds on period are useful in verifying whether the system can exhibit Zeno-sampling.

Let's first examine the problem of obtaining a lower bound on the sampling period under the assumption of no delays. For notational convenience, rewrite the earlier $\mathscr{L}_2$ event-trigger in terms of a *normalized* gap function,

$$z_j(t) = \sqrt{M}e_j(t)$$

so that the triggering inequality takes the form

$$\|z_j(t)\| < \sqrt{x^T(r_j)Nx(r_j)} = \rho(x(r_j))$$

where the function $\rho : \mathbb{R}^n \to \mathbb{R}$ is defined in the above equation. Figure 1.9 shows the time history of the gap functions when there are no delays; in other words the controller job's finishing time equals the release time. As was done in the earlier analysis regarding sampling periods for ISS event-triggers, let's examine the normalized gap function's rate of growth over the interval $[r_j, r_{j+1})$.

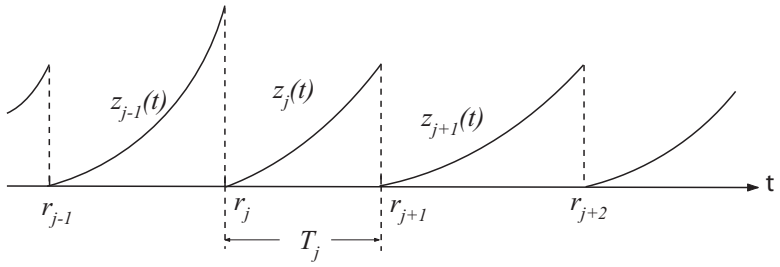


Fig. 1.9 Time history of normalized gap $\|z_j(t)\|$ when there are no delays ($r_j = f_j$).

The analysis starts by bounding the time derivative of $\|z_j(t)\|$,

$$\begin{aligned}\frac{d}{dt}\|z_j(t)\| &\leq \left\|\sqrt{M}\dot{e}_j(t)\right\| = \left\|\sqrt{M}\left(Ax(t) - BB^TPx(r_j) + w(t)\right)\right\| \\ &\leq \left\|\sqrt{M}Ae_j(t)\right\| + \left\|\sqrt{M}(A - BB^TP)x(r_j)\right\| + \left\|\sqrt{M}\right\|\|w(t)\|\end{aligned}$$

for $t \in [r_j, r_{j+1})$. Let's assume the disturbance is bounded by the norm of the system state. In other words, let's assume there exists a positive real constant W such that $\|w(t)\| \leq W\|x(t)\|$. In this case the preceding upper bound on $\frac{d}{dt}\|z_j(t)\|$ may be simplified to the form,

$$\frac{d}{dt}\|z_j(t)\| \leq \alpha\|z_j(t)\| + \mu_0(x(r_j)) \tag{1.17}$$

where α is a real constant such that

$$\alpha = \left\|\sqrt{M}A\sqrt{M^{-1}}\right\| + W\left\|\sqrt{M}\right\|\left\|\sqrt{M^{-1}}\right\|$$

and $\mu_0 : \mathbb{R}^n \to \mathbb{R}$ is a function such that

$$\mu_0(x(r_j)) = \left\|\sqrt{M}(A - BB^T P)x(r_j)\right\| + W\left\|\sqrt{M}\right\|\|x(r_j)\|.$$

The differential inequality in equation (1.17) bounds the rate of growth of the normalized gap function over the interval between two consecutive release times, r_j and r_{j+1}. Since the state is sampled at time r_j, the gap is zero at that time. So the initial condition for the differential inequality is $\|z_j(r_j)\| = 0$. One can therefore use the Comparison principle to show that for all $t \in [r_j, r_{j+1})$ that

$$\|z_j(t)\| \leq \frac{\mu_0(x(r_j))}{\alpha}\left(e^{\alpha(t-r_j)} - 1\right).$$

This is an upper bound on the normalized gap between two consecutive release times. Clearly, the next release r_{j+1} must occur before the right-hand side of the above inequality violates the $\mathscr{L}_2$ event threshold. In other words, the following inequality must hold

$$\frac{\mu_0(x(r_j))}{\alpha}(e^{\alpha T_j} - 1) \geq \rho(x(r_j)).$$

This inequality can be solved for the sampling period $T_j = r_{j+1} - r_j$ to obtain

$$T_j \geq \frac{1}{\alpha}\ln\left(1 + \alpha\frac{\rho(x(r_j))}{\mu_0(x(r_j))}\right)$$

when the next release occurs. The above inequality represents a lower bound on the intersample time intervals generated by $\mathscr{L}_2$ event-triggers. It can be shown [83] that this bound is always bounded away from zero, so as in the ISS event-trigger case Zeno-sampling does not occur. A major reason for this lies in the requirement that the external disturbance has a norm that goes to zero as the system state approaches its equilibrium. This is a particularly strong assumption that can be justified if the source of the disturbance arises from modeling uncertainty. In general, however, if this assumption does not hold, then $\mathscr{L}_2$ event-triggers can lead to Zeno-sampling as was seen in figure 1.7. One can avoid these undesirable behaviors by imposing some

additional constraints on the event-triggers. This particular approach was discussed in more detail in [84].

The usefulness of the prior analysis is limited by the no task delay assumption. Let's now examine how this assumption might be relaxed. In the case of delays, the normalized gap's evolution changes as shown in figure 1.10. With non-zero delays, the individual gap functions overlap as shown in the figure. This means that one should partition the time interval $[r_j, f_{j+1})$ into two subintervals $[r_j, f_j)$ and $[f_j, f_{j+1})$. Over the first subinterval, the system state evolves according to the differential equation

$$\dot{x}(t) = Ax(t) - BB^T Px(r_{j-1}) + w(t)$$

in which the state used in the controller is the state at sample time r_{j-1}. After the j^{th} control job finishes, the control is updated with the most recent sampled state. This means that over the time interval $[f_j, f_{j+1})$, the system state evolves according to the differential equation

$$\dot{x}(t) = Ax(t) - BB^T Px(r_j) + w(t).$$

In a manner similar to what was done in the no-delay case, differential inequalities can be used to bound $\|z_j(t)\|$ for all $t \in [r_j, f_{j+1})$.

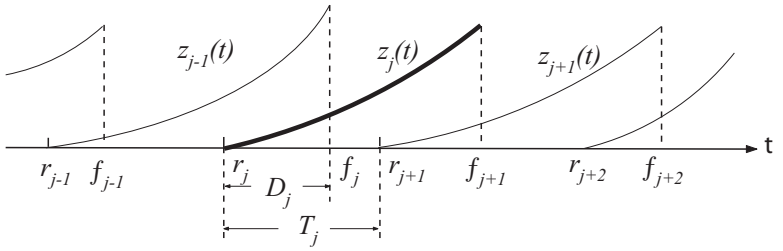


Fig. 1.10 Time history of normalized gap $\|z_j(t)\|$ when there are task delays ($r_j < f_j$).

The analysis of the non-zero delay case is done by viewing the event threshold $\rho(x(r_j))$ as a *budget* that is allocated to the normalized gap. In particular we partition this budget between the two subintervals $[r_j, f_j)$ and $[f_j, f_{j+1})$. Let's require that over the first subinterval $[r_j, f_j)$ is constrained so the normalized gap doesn't get bigger than $\varepsilon\rho(x(r_j))$ where ε is a user-defined constant between 0 and 1. One can again use differential inequalities to show that the normalized gap is bounded as

$$\|z_j(t)\| \leq \frac{\mu_1(x(r_j), x(r_{j-1}))}{\alpha}\left(e^{\alpha(t-r_j)} - 1\right) = \Phi(x(r_j), x(r_{j-1}); t - r_j) \quad (1.18)$$

for all $t \in [r_j, f_j)$ and where the function $\mu_1 : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\mu_1(x(r_j), x(r_{j-1})) = \left\| \sqrt{M}(Ax(r_j) - BB^T Px(r_{j-1})) \right\| + W\|\sqrt{M}\|\|x(r_j)\|.$$

The right-hand side of the above inequality (1.18) represents the solution to the differential equation

$$\frac{d}{dt}\|z_j\| = \alpha\|z_j(t)\| + \mu_1(x(r_j), x(r_{j-1}))$$

where α is a real constant. The solution to this differential equation is characterized by the function $\Phi : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$. This function returns the normalized gap $z_j(t)$ at time t as a function of the system states $x(r_j)$ and $x(r_{j-1})$. The dependence of Φ on the system state at times r_j and r_{j-1} is a consequence of the fact that the differential equations governing the evolution of the system state are different over time intervals $[r_j, f_j)$ and $[f_j, f_{j+1})$.

Note that the duration of the first subinterval, $[r_j, f_j)$ is the delay, D_j. To ensure the normalized gap gets no larger than $\varepsilon\rho(x(r_j))$ over this first subinterval, equation (1.18) implies that

$$\frac{\mu_1(x(r_j), x(r_{j-1}))}{\alpha}\left(e^{\alpha D_j} - 1\right) \leq \varepsilon\rho(x(r_j)).$$

Solving for D_j yields

$$0 \leq D_j \leq \frac{1}{\alpha}\ln\left(1 + \varepsilon\alpha\frac{\rho(x(r_j))}{\mu_1(x(r_j), x(r_{j-1}))}\right).$$

The above equation represents an upper bound on the *delay* that ensures the gap at the end of the first subinterval is less than the allocated budget of $\varepsilon\rho(x(r_j))$.

The analysis is completed by examining the behavior of the gap over the second subinterval $[f_j, f_{j+1})$. At the beginning of this interval,

$$\|z_j(f_j)\| \leq \Phi(x(r_j), x(r_{j-1}); D_j) \leq \varepsilon\rho(x(r_j)) \leq \rho(x(r_j)). \tag{1.19}$$

The system state over the interval $[f_j, f_{j+1})$ satisfies the differential equation

$$\dot{x}(t) = Ax(t) - BB^T Px(r_j) + w(t).$$

Using an argument similar to that employed in analyzing the gap over the first subinterval, one can show that

$$\frac{d}{dt}\|z_j(t)\| \leq \alpha\|z_j(t)\| + \mu_0(x(r_j)).$$

Solutions to this differential inequality over the interval $t \in [f_j, f_{j+1}]$ are bounded above by solutions to the associated differential equality

$$\frac{d}{dt}\|\tilde{z}_j(t)\| = \alpha\|\tilde{z}_j(t)\| + \mu_0(x(r_j))$$

with the initial condition $\tilde{z}_j(f_j) = z(f_j)$. This bounding solution, $\tilde{z}_j(t)$, is used to predict when the release time should be selected to ensure the $\mathscr{L}_2$ stability of the event-triggered system.

This result is proven in [83]. In particular, this paper states that the closed-loop system is $\mathscr{L}_2$ stable with a gain less than or equal to γ/β if the task's $(j+1)^{\text{st}}$ release time is generated by

$$r_{j+1} = f_j + \frac{1}{\alpha}\ln\left(1+\alpha\frac{\delta\rho(x(r_j)) - \Phi(x(r_j),x(r_{j-1});D_j)}{\mu_0(x(r_j)) + \alpha\Phi(x(r_j),x(r_{j-1});D_j)}\right) \tag{1.20}$$

and the delay D_{j+1} satisfies

$$D_{j+1} \leq \frac{1}{\alpha}\left(1+\varepsilon\alpha\frac{(1-\delta)\rho(x(r_j))}{\alpha\delta\rho(x(r_j)) + \mu_0(x(r_j))}\right). \tag{1.21}$$

In the above function α is the real constant defined earlier, ρ and μ_0 are the class $\mathscr{K}$ functions defined above and Φ bounds a function of the system state as it evolves over the delay time D_j.

Self-Triggered Feedback Control: The results in the prior section do more than suggest that an event-triggered system's $\mathscr{L}_2$ stability will be robust with respect to task delays. Equation (1.20) is interesting in that it computes the *future* release time given the current release and finishing times. This equation therefore provides a prediction of the next release time and it suggests that it should be possible to develop a *software* implementation of event-triggered systems. This software version of event-triggering has sometimes been referred to as *self-triggered* feedback control. Such software versions of event-triggering may be preferred in applications where the cost of adding event-detection hardware is deemed unacceptable.

The concept of self-triggered task models was original presented in [75]. Simulation results [41] suggested that self-triggering systems exhibit a robustness to delays that is consistent with what one might exhibit from event-triggered systems. In these earlier works the computation of the next release time was usually done in a heavy handed fashion that was not computationally efficient. This has changed recently with results in [83] which allow a more computationally efficient way of selecting the next release time. More recently, it has been noted that for *homogenous* systems [1], release times enforcing input-to-state stability satisfy certain scaling relationships. These relationships can be used to reduce the computation of the next release time to a table look-up. Another important aspect of these analytical bounds on acceptable delay and release times is that they can be used as quality-of-service constraints for real-time schedulers. Since it now becomes possible to predict when the next control job should be released, one can use these estimates as the period and deadline that govern how real-time scheduling services adjust task priorities. In other words, the aforecited analytical bounds provide a formal way of connecting real-time scheduling constraints to the application's (i.e. control system's) actual performance.

Event-triggering was proposed [2, 3] as a means to co-design controllers and schedulers in embedded systems. The analysis of *state-dependent* event-triggers [70] formally characterized the stability properties of event-triggering and the later analyses of intersample intervals and maximally allowable deadlines [83] provided bounds that could be used in adapting the embedded system's controller tasks. These methods raise the possibility of moving away from the traditional hard real-time task models that have dominated embedded control. While the use of event-triggering has focused on conserving the embedded system's computational resources, it is anticipated that event-triggering may be used to conserve other types of shared resources. Of particular interest are embedded sensor-actuator control systems where communication resources are highly constrained. The next section examines the application of state-dependent event-triggering to such networked control systems.

1.4 Event-Triggered Feedback in Networked Control Systems

Many of the results for event-triggered control of embedded systems can be extended to networked control systems. A networked control system or NCS is a set of controllers that coordinate their actions over a communication network. For NCS, event-triggering is used to decide when to *transmit* or *broadcast* the system state to a local controller's neighbors. Using events to trigger communication actually provides a much stronger motivation for event-triggered control. The reason for this is that in many cases, the energy or cost associated with the transmission of a bit of information is much more than the energy associated with using that bit to compute the control law. Event-triggering, therefore, provides a realistic way of reducing traffic congestion in communication networks used by NCS. The objective of this section is to show how the earlier results from event-triggered control of embedded systems can be extended to networked control systems. The section first discusses the NCS architecture under study and then it derives event-triggers assuring the NCS is ISS. As in the case of embedded systems, the NCS implementation introduces a number of so-called *network artifacts* that complicate the analysis of the idealized model. These network artifacts include delays in the transmission of information as well as dropped information packets. This section studies the impact of such network artifacts and demonstrates that event-triggered NCS stability is robust to such network artifacts in a quantifiable manner.

While there is a great deal of literature [11, 25, 50, 51, 90] examining networked control systems, there is relatively little work pertaining to event-triggered NCS. Most of the results in this section are drawn from [81] and [80]. Related work will be found in [48].

Model of Networked Control System: Let's first describe a model of a networked control system or NCS. Consider a distributed NCS consisting of N agents. Figure 1.11 provides a graphic illustration of an NCS with three agents. Each agent consists of a *physical* component and a *cyber*-component. The physical components are

interconnected as shown by the solid lines in the figure. The cyber-components are also interconnected through a communication network whose links are shown by the dashed lines in the figure.

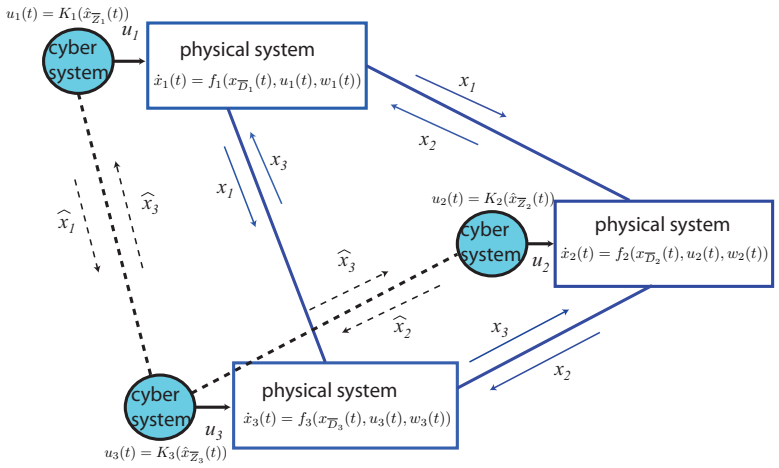


Fig. 1.11 Model of Event-Triggered Networked Control Systems

This system may be more formally characterized using graph theoretic notation. In particular, let $\mathscr{N} = \{1, 2, \ldots, N\}$ denote the set of agents. A graph $\mathscr{G}_\mathrm{p} = (\mathscr{N}, \mathscr{E}_\mathrm{p})$ represents the physical coupling between the agents. $\mathscr{N}$ denotes the vertices of the graph and $\mathscr{E}_\mathrm{p} \subset \mathscr{N} \times \mathscr{N}$ denotes the set of edges in the graph. The edge (i, j) connects node $i \in \mathscr{N}$ to node $j \in \mathscr{N}$. The edge, therefore, is an ordered pair (i, j) of nodes. The ordered pair (i, j) is in $\mathscr{E}_\mathrm{p}$ if the dynamics of agent j's physical component are directly driven by agent i's local state. The graph $\mathscr{G}_\mathrm{c} = (\mathscr{N}, \mathscr{E}_\mathrm{c})$ models the interconnections between the cyber-components of the agents. As before $\mathscr{N}$ denotes the vertices (nodes) of the graph and $\mathscr{E}_\mathrm{c} \subset \mathscr{N} \times \mathscr{N}$ represents the edges of the graph.

In this section, the graphs for the physical and cyber-interconnections need not be the same. This requires us to define a number of special neighborhoods in the graph. In particular,

- $Z_i = \{j \in \mathscr{N} \,|\, (j, i) \in \mathscr{E}_\mathrm{c}\}$ represents those agents whose cyber-components can send information to agent i's cyber-component.
- $U_i = \{j \in \mathscr{N} \,|\, (i, j) \in \mathscr{E}_\mathrm{c}\}$ denotes those agents whose cyber-components can receive information from agent i's cyber-component.
- $D_i = \{j \in \mathscr{N} \,|\, (j, i) \in \mathscr{E}_\mathrm{p}\}$ represents those agents whose physical components directly drive the dynamics of agent i's physical component.
- $S_i = \{j \in \mathscr{N} \,|\, (i, j) \in \mathscr{E}_\mathrm{p}\}$ denotes those agents whose physical components are directly driven by the physical component of agent i.

For any set $\Sigma \subset \mathscr{N}$, let $|\Sigma|$ denote the number of elements in that set and let $\overline{\Sigma} = \Sigma \cup \{i\}$.

The physical component of agent i is characterized by a *local state* $x_i : \mathbb{R} \to \mathbb{R}^n$ where x_i satisfies the differential equation

$$\begin{aligned} \dot{x}_i(t) &= f_i(x_{\overline{D}_i}(t), u_i(t), w_i(t)) \\ x_i(t_0) &= x_{i0} \end{aligned}$$

where $x_{\overline{D}_i} = \{x_j\}_{j\in\overline{D}_i}$ are the local states of agent i's neighbors that are physically connected to it. The system dynamics are characterized by the function $f_i : \mathbb{R}^{n|\overline{D}_i|} \times \mathbb{R}^m \times \mathbb{R}^\ell \to \mathbb{R}^n$ which is locally Lipschitz and satisfies $f_i(0,0,0) = 0$. $u_i : \mathbb{R} \to \mathbb{R}^m$ is a control input generated by the cyber-component of the agent and $w_i : \mathbb{R} \to \mathbb{R}^\ell$ is an external disturbance. The above characterization assumes all subsystems have the same local state dimension, n. This is done for notational convenience. The model and subsequent analysis would also apply to subsystems with local states of different dimensionalities.

The control u_i is generated by agent i's cyber-component. Since these cyber-components exchange information over a digital communication network, local states are transmitted in a discrete manner. In particular, let $\{r_j^i\}_{j=1}^\infty$ denote the sequence of broadcast release times for the i^{th} agent. So the *transmitted* state from agent i is denoted as

$$\hat{x}_i(t) = x_i(r_j^i)$$

for $t \in [r_j^i, r_{j+1}^i)$ and $j = 0, 1, \ldots, \infty$. Agent i's cyber-component uses the local state information received from all its neighbors in the set Z_i to compute the control u_i. So let $K_i : \mathbb{R}^{n|\overline{Z}_i|} \to \mathbb{R}^m$ denote the i'th agent's local controller so that

$$u_i(t) = K_i(\hat{x}_{\overline{Z}_i}(t)).$$

Following the same notational conventions as before, $\hat{x}_{\overline{Z}_i}$ denotes the broadcast states of all neighbors of agent i whose cyber-components send information directly to agent i.

ISS Event-Triggered Networked Control: Let's now derive ISS event-triggers for the NCS described above. In particular, let $e_i(t) = \hat{x}_i(t) - x_i(t)$ denote the local *gap* between agent i's current state and its last broadcast state. Assume there exist positive definite function $V : \mathbb{R}^{nN} \to \mathbb{R}$, controllers $K_i : \mathbb{R}^{|n\overline{Z}_i|} \to \mathbb{R}^m$, and class $\mathscr{K}$ functions γ_i, ψ_i, and β_i (for $i = 1, 2, \ldots, N$) such that

$$\begin{aligned} \dot{V} &= \sum_{i=1}^N \frac{\partial V}{\partial x_i} f_i(x_{\overline{D}_i}, K_i(x_{\overline{Z}_i} + e_{\overline{Z}_i}), w_i) \\ &\leq \sum_{i=1}^N \left(-\gamma_i(\|x_i\|) + \psi_i(\|e_i\|) + \beta_i(\|w_i\|)\right) \end{aligned} \tag{1.22}$$

where $e_{\overline{Z}_i}$ is the gap of all agent i's cyber-neighbors. This assumption means that V is an ISS-Lyapunov function with respect to w when the the gap $e_i(r_j^i) = 0$. In

view of our earlier discussion, this is sufficient to imply that the local controllers K_i leave the original continuously sampled version of the networked control system input-to-state stable.

So again, one selects a user-defined parameter $\sigma_i \in (0,1)$ and notes that if the local state and gap trajectories satisfy the inequality

$$-\sigma_i \gamma_i(\|x_i(t)\|) + \psi_i(\|e_i(t)\|) \leq 0 \tag{1.23}$$

for all $t \in \mathbb{R}$ and all $i = 1,2,\ldots,N$, then the bound on $\dot{V}$ becomes

$$\dot{V} \leq \sum_{i=1}^{N} \left(-(1-\sigma_i)\gamma_i(\|x_i\|) + \beta_i(\|w_i\|)\right).$$

This is a dissipative inequality that was seen earlier to be sufficient to show that the event-triggered NCS is ISS with respect to the external input w_i.

As before in our study of the embedded event-triggered controllers, the inequality in equation (1.23) can be used as the basis of a state-dependent threshold test. In particular, the i^{th} agent would check the validity of the following threshold test on the gap,

$$\psi_i(\|e_i(t)\|) \leq \sigma_i \gamma_i(\|x_i(t)\|). \tag{1.24}$$

At the broadcast time r_j^i, the local gap, $e_i = 0$. This gap then grows until $\psi_i(\|e_i(t)\|)$ exceeds the state dependent threshold $\gamma_i(\|x_i(t)\|)$. The violation of that threshold triggers agent i to broadcast its state again. Note that this is a *cooperative* broadcast mechanism in that the violation of the threshold results in an agent sharing its local state information with its neighbors. In other words, the success of such an event-triggered broadcast scheme relies on all agent's agreeing to work in the same manner.

Note that the ISS event trigger given above is only a *local* function of the agent's state. This is important, for it means each agent is able to trigger its broadcast without relying directly on its neighbors. A key part of the prior analysis is the assumption that there exists an ISS Lyapunov function that is separable in the sense specified by the bounds in equation (1.22). Such a Lyapunov function may be constructed by identifying a set of N positive definite functions $V_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1,2,\ldots,N$ with class $\mathscr{K}$ functions γ_i, η_i,ψ_i, and β_i such that

$$\begin{aligned}\frac{\partial V_i}{\partial x_i} f_i(x_{\overline{D}_i}, K_i(x_{\overline{Z}_i} + e_{\overline{Z}_i}), w_i) \leq & -\gamma_i(\|x_i\|) + \sum_{j \in D_i \cup Z_i} \eta_j(\|x_j\|) \\ & + \sum_{j \in \overline{Z}_i} \psi_j(\|e_j\|) + \beta_i(\|w_i\|).\end{aligned} \tag{1.25}$$

As a specific example, let's consider class $\mathscr{K}$ functions that are quadratic so $\gamma_i(\|x\|)$ can be expressed as $\gamma_i\|x\|^2$ and similarly for the other functions, η_i, ψ_i, and β_i. In this case, one sees that by choosing $V = \sum_{i=1}^{N} V_i$, the following inequality is obtained

$$\dot{V} \leq \sum_{i=1}^{N} \left(-\gamma_i \|x_i\|^2 + \sum_{j \in D_i \cup Z_i} \eta_j \|x_j\|^2 + \sum_{j \in \overline{Z}_i} \psi_j \|e_j\|^2 + \beta_i^2 \|w_i\|^2 \right)$$
$$= \sum_{i=1}^{N} \left(-\left(\gamma_i - |S_i \cup U_i| \eta_i\right) \|x_i\|^2 + \psi_i |\overline{U}_i| \|e_i\|^2 + \beta_i \|w_i\|^2 \right).$$

Note that this matches the conditions in equation (1.22) provided the first term on the right-hand side is negative definite. This term will be negative definite if

$$\gamma_i - |S_i \cup U_i| \eta_i > 0.$$

This condition places a restriction on the amount of coupling between physically interconnected physical systems. In particular, it says that if one can appropriately bound this physical coupling and if there exist candidate ISS-Lyapunov functions satisfying the bounds in equation (1.25), then it is always possible to construct a global V that is an ISS-Lyapunov function for the entire networked system. In this case, the associated ISS event-trigger is shown to have the form

$$\|e_i(t)\| \leq \sigma_i \sqrt{\frac{\gamma_i - |S_i \cup D_i| \eta_i}{|\overline{U}_i| \psi_i}} \|x_i(t)\| = \frac{\sigma_i}{\alpha_i} \|x_i(t)\|$$

which would ensure the $\mathscr{L}_2$ stability of the entire system.

The ability to construct V from smaller *local* candidate ISS-Lyapunov functions is important, for it allows us to distribute the design of the ISS event-triggers. This is particularly important in large-scale networked systems where agent subsystems may be added and modified in an ad hoc manner. Linear networked systems provide a particularly good example of when one can exploit this distributed strategy for constructing ISS event-triggers. For linear NCS, the parameters in the triggering conditions can be computed using linear matrix inequalities [82].

Simulation results for this approach to event-triggered broadcasting are shown in figure 1.12. This example was taken from [81]. It consists of N carts that are interconnected through soft springs. The local state of the ith cart is $x_i = \begin{bmatrix} y_i & \dot{y}_i \end{bmatrix}^T$ where y_i is the position of the ith cart with respect to the system's equilibrium point. Assuming soft spring coupling between the carts, the state equation for the ith cart can be written as

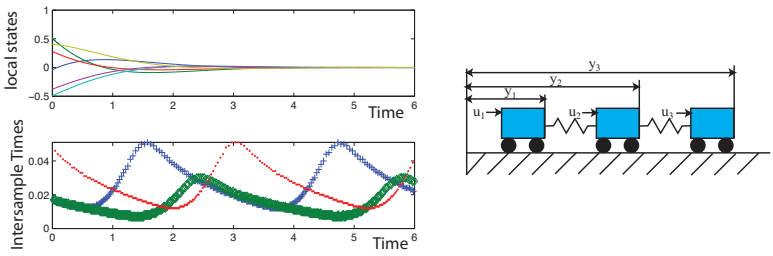
$$\dot{x}_i(t) = \frac{d}{dt} \begin{bmatrix} y_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \dot{y}_i(t) \\ u_i(t) + k_i^1 \tanh(y_{i+1}(t) - y_i(t)) + k_i^2 \tanh(y_{i-1}(t) - y_i(t)) + w_i(t) \end{bmatrix}$$

for all $t \in \mathbb{R}$ where $i = 1, 2, \ldots, N$. The parameters k_i^1 and k_i^2 denote the spring constants for the springs on the right-hand and left-hand side of the ith cart, respectively. From the cart geometry shown in figure 1.12, one can see that these spring constants satisfy $k_i^1 = k_{i+1}^2$ for $i = 1, 2, \ldots, N-1$. The left-end cart's spring constant is $k_1^2 = 0$ and the right-end cart's spring constant in $k_N^1 = 0$. The function $u_i : \mathbb{R} \to \mathbb{R}$ denotes the control applied to the cart by its local controller.

In this example the communication network's links mirror the physical interactions between the carts so that $Z_i = D_i$. The sampled state is denoted as $\hat{x}_i(t) = \left[\hat{y}_i(t) \;\; \frac{d}{dt}\hat{y}_i(t)\right]^T$ where $\hat{y}_i(t) = y_i(r_j^i)$ and $\frac{d}{dt}\hat{y}_i(t) = \dot{y}_i(r_j^i)$ for all $t \in [r_j^i, r_{j+1}^i)$ and $j = 0, 1, \ldots, \infty$. The local control is computed from these sampled measurements as

$$u_i(t) = K_i\hat{x}_i(t) - k_i^1(\tanh(\hat{y}_{i+1}(t) - \hat{y}_i(t)) - k_i^2\tanh(\hat{y}_{i-1}(t) - \hat{y}_i(t)).$$

In this case, the agents controlling the *end* cars use the ISS event-trigger $5.9\|e_i(t)\| < 0.2\|x_i(r_j^i)\|$ and the interior agents use the event-trigger $10.3\|e_i(t)\| < 0.2\|x_i(r_j^i)\|$. The results from this simulation are shown in figure 1.12.



.

Fig. 1.12 Simulation Example of Event-Triggered Networked Control System consisting of three coupled carts.

The top plot on the left-hand side of figure 1.12 plots the state trajectories for all three carts. As can be seen, this event-triggered system is asymptotically stable since all points asymptotically approach their equilibrium points at zero. The bottom plot on the left-hand side of figure 1.12 plots the intersample time intervals that were generated by the proposed event-triggers. As can be seen, these intersample time intervals vary over time in a regular manner.

Impact of Network Artifacts: The prior analysis for the ISS event-triggers in networked control systems had two important assumptions that now need to be examined in more detail. The first assumption was that there was no delay between the transmission and reception of information over the communication network. The second important assumption was that all neighbors in the set U_i receive and use the broadcast data in a synchronized manner. Both assumptions are difficult to justify in real-life wireless sensor-actuator networks. This difficulty is a direct consequence of the unreliable and time-varying nature of wireless communication. The second assumption can be dealt with by making use of a *broadcast protocol* that essentially synchronizes the transmitted data across all neighbors in U_i. The use of such a protocol, however, introduces a number of *network artifacts* such as delays and dropped messages; both of which have a significant impact on the event-triggered system's performance. The objective of this subsection is to establish bounds on acceptable transmission delays and message dropout rates, thereby showing that the stability of the event-triggered NCS is robust to such network artifacts.

Let's first describe the network broadcast protocol used to ensure that the received broadcasts are synchronized between all neighbors in U_i. Such a broadcast protocol is illustrated graphically on the left-hand side of figure 1.13. In this case, the shaded agent represents the ith broadcasting agent at time instant r_j^i. This broadcast is made to the two neighboring agents. Since this is a broadcast, both neighboring agents receive the same sampled copy of the transmitting agent's local state. Upon receiving the ith agent's message, each agent acknowledges the receipt of that message through an ACK signal. When the ith agent receives ACKs from all of the neighbors in U_i, it then broadcasts a *permission* or PERM message to those neighbors. As soon as all neighbors receive the PERM message they use the previously received data in computing their controls. The delay between initial transmission and the final receipt of the PERM messages represents the delay between sampling and actuation. As long as this delay is sufficiently small, the overall networked system should still be stable.

ACKs and PERMs are control packets that are very short in length and can therefore be delivered with a high degree of reliability. The data packets, on the other hand are relatively long and will be more subject to unreliable transmission. Even if an ACK or PERM message were lost, the impact such lost information has on the overall system's performance can be detected and used to trigger additional data broadcasts. So one would expect the system's overall performance to be robust to such faults. Just how robust this system is to such faults, however, has yet to be fully studied.

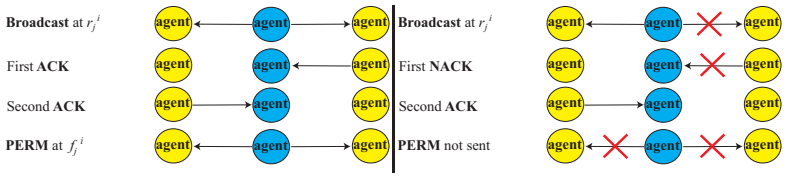


Fig. 1.13 Broadcast Protocol in Wireless NCS. (left) step-by-step description of broadcast protocol. (right) mechanism by which transmitted data is dropped.

It is relatively easy to see why the assumption that transmissions are received instantaneously is unreasonable. While the transmitted signal propagates at the speed of light, it takes time for a message to work its way though an agent's network stack. Moreover, it takes time to transmit, receive and acknowledge the ACKs of an agent's neighbors. As a result, the analysis cannot assume that messages are transmitted and received with zero delay.

The right-hand side of figure 1.13 shows another network artifact that can't be neglected. Wireless communication is inherently unreliable since there is a finite probability that a message will not be successfully transported across the channel. In this case, it is highly likely that a broadcast message may not be received by all neighbors in Z_i. When this occurs, ACK messages will only be sent by a subset of the agents in Z_i. Since the transmitting agent doesn't receive all of the ACKs it is

expecting, it will not send the PERM message and so the neighboring agents will not use the information that was previously transmitted to them. In this situation, the data transmitted by the i^{th} agent is actually *dropped*. An important question is whether or not event-triggered system stability is robust to such data dropouts.

Under the proposed broadcast protocol, one must therefore adopt a somewhat more complex view of the timing relations between message transmission and reception than was presented earlier. Figure 1.14 illustrates the underlying timing relationships assumed in the following analysis. As before, let's define a sequence $\{r_j^i\}_{j=0}^{\infty}$ which consists of the time instants when the i^{th} agent *releases* a message for broadcast to its neighbors. If this agent receives ACKs from all of its neighbors then the broadcast is said to be successful. One can therefore introduce a subsequence of $\{r_j^i\}_{j=0}^{\infty}$ that consists of all the *successful* broadcast times. Let $\{b_k^i\}_{k=0}^{\infty}$ denote this sequence of successful broadcasts. If a broadcast is successful, there is a finite delay associated with informing all neighbors that the broadcast was successful (i.e. the time required to execute the broadcast protocol). One can therefore define a sequence of successful *finishing* times, $\{f_k^i\}_{k=0}^{\infty}$. The time instant f_k^i denotes the time when the broadcast that was released at time instant b_k^i was given permission for use by all agents in U_i. With regard to these timing relations, the *number of dropped* broadcasts between the k^{th} and $(k+1)^{\text{st}}$ successful broadcasts is denoted as d_k^i.

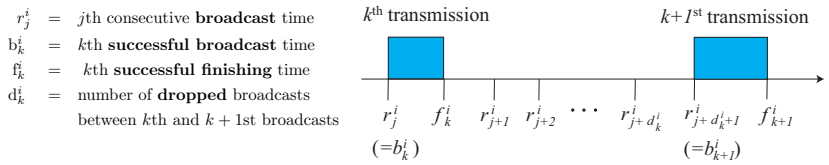


Fig. 1.14 Timing Relationships under NCS Broadcast Protocol

Analyzing the effect that such network artifacts have on the event-triggered system's performance can be done in a manner that is analogous to our earlier analysis of delays in event-triggered embedded systems. As before, one first considers a somewhat weaker version of the event-trigger in which the threshold is only a function of the last sampled state $\hat{x}_i(t)$, rather than the current local state $x_i(t)$. The original event-trigger has the form

$$\|e_i(t)\| \leq \sigma_i \sqrt{\frac{\gamma_i - |S_i \cup B_i|\eta_i}{|\overline{U}_i|\psi_i}}\|x_i(t)\| = \frac{\sigma_i}{\alpha_i}\|x_i(t)\|.$$

A sufficient condition that ensures the above inequality is satisfied would be

$$\|e_i(t)\| \leq \frac{\sigma_i}{\sigma_i + \alpha_i}\|\hat{x}_i(t)\| = c_i\|\hat{x}_i(t)\|.$$

In this weaker condition, the threshold is constant over the interval $[f_j^i, f_{j+1}^i)$. As mentioned above, this event-trigger makes it easier to analyze the impact that delays and dropouts have on the overall event-triggered system's performance.

With this simplified event-triggering condition, one analyzes the impact of dropouts and delays by allocating some portion of the threshold condition to each network artifact. Figure 1.15 shows the gap $\|e_i(t)\|$ as a function of time between a successful broadcast at time b_k^i and the finishing time of the next successful broadcast f_{k+1}^i. The gap grows over this interval of time and in order to assure the performance of the event-triggered system, one requires that this gap always remains less than $c_i\|\hat{x}_i(t)\|$. The effect of the dropouts on the gap is confined to the first part of this interval between times b_k^i and b_{k+1}^i. The effect of the delay on the gap is confined to the last part of the interval from b_{k+1}^i to f_{k+1}^i. This means that one can separate the impact of dropouts and delays between these two subintervals. One exploits this separation by requiring that the next successful broadcast at time b_{k+1}^i occur before the gap gets larger than $\delta_i c_i\|\hat{x}_i(t)\|$ where $\delta_i \in (0,1)$ is a user specified constant. Once δ_i is selected, this determines how many dropouts the system can tolerate before violating the condition.

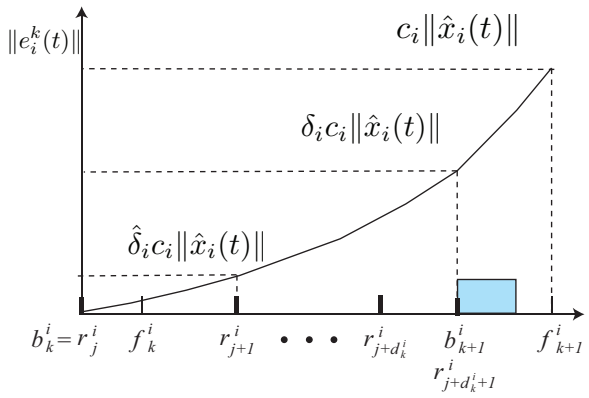


Fig. 1.15 Gap time history in the presence of network artifacts such as dropouts and delays

Is it possible to ensure the gap due to dropouts is no larger than than the allocated gap budget of $\delta_i c_i\|\hat{x}_i(t)\|$? This is done by simply triggering the event early. In particular, let's use an actual event-trigger of the form

$$\|e_i^k(t)\| \leq \hat{\delta}_i c_i\|\hat{x}_i(t)\| = \hat{\delta}_i c_i\|x_i(b_k^i)\|$$

where $e_i^k(t) = x_i(t) - x_i(b_k^i)$ for $t \in [b_k^i, f_{k+1}^i)$, $\hat{\delta}_i \in (0, \delta_i)$ and b_k^i is the k^{th} successful broadcast. As shown in figure 1.15, the use of such a smaller threshold will cause the system to trigger early, thereby providing some margin for dropouts or delays. With this threshold the next release of a transmission occurs when $\|x_i(t) - x_i(b_k^i)\| = \hat{\delta}_i c_i\|x_i(b_k^i)\|$. The transmitting agent, however, does not know if

this transmission was successful, so when it tests for the next released transmission, it uses the threshold condition

$$\|x_i(t) - x_i(r^i_{j+1})\| < \hat{\delta}_i c_i \|x_i(r^i_{j+1})\|.$$

Note that in this inequality, the sampled system state that is used is taken at time r^i_{j+1} rather than $r^i_j = b^i_k$. So if one now looks at the total gap $\|e^k_i(t)\|$ that occurs for times after r^i_{j+1}, then one can write this as

$$\begin{aligned}\|e^k_i(t)\| &= \|x_i(t) - x_i(b^i_k)\| \\ &\leq \|x_i(t) - x_i(r^i_{j+1})\| + \|x_i(r^i_{j+1}) - x_i(b^i_k)\|.\end{aligned}$$

Each of the two terms can be bounded using the event-triggering conditions to obtain

$$\begin{aligned}\|e^k_i(t)\| &\leq \hat{\delta}_i c_i \|x_i(r^i_{j+1})\| + \hat{\delta}_i c_i \|x_i(b^i_k)\| \\ &\leq \hat{\delta}_i c_i \left(\|x_i(b^i_k)\| + \hat{\delta}_i c_i \|x_i(b^i_k)\| \right) + \hat{\delta}_i c_i \|x_i(b^i_k)\| \\ &= \left((1 + \hat{\delta}_i c_i)^2 - 1 \right) \|x_i(b^i_k)\|.\end{aligned}$$

The last relationship shows that under event-trigger $\|x_i(t) - x_i(r^i_j)\| < \hat{\delta}_i c_i \|x_i(r^i_j)\|$, that the first release r^i_{j+1} occurs when the gap reaches $\hat{\delta}_i c_i \|x_i(b^i_k)\|$. If that first release is unsuccessful, then the second release occurs at r^i_{j+2} when the gap equals $\left((1 + \hat{\delta}_i c_i)^2 - 1 \right) \|x_i(b^i_k)\|$. In a similar way one can show that if additional releases are unsuccessful then

$$\|e^k_i(t)\| \leq \left((1 + \hat{\delta}_i c_i)^{d^i_k + 1} - 1 \right) \|x_i(b^i_k)\|$$

for all t where d^i_k is the number of consecutive unsuccessful releases (i.e. dropped transmissions) between times b^i_k and b^i_{k+1}. In order to assure the stability of this system let's require that the right-hand side of the above inequality be less than $\delta c_i \|x_i(b^i_k)\|$ or rather that

$$\left((1 + \hat{\delta}_i c_i)^{d^i_k + 1} - 1 \right) \|x_i(b^i_k)\| \leq \delta_i c_i \|x_i(b^i_k)\|.$$

Solving this inequality for d^i_k determines an upper bound on the maximum number of successive dropouts that can be tolerated to assure overall system stability. This bound is called the *maximally allowed number of successive dropouts* (MANSD) and the bound is

$$d^i_k \leq \text{MANSD} = \left\lfloor \log_{1+\hat{\delta}_i c_i}(1 + \delta_i c_i) \right\rfloor - 1.$$

This bound represents the maximum number of dropouts that our system can accept.

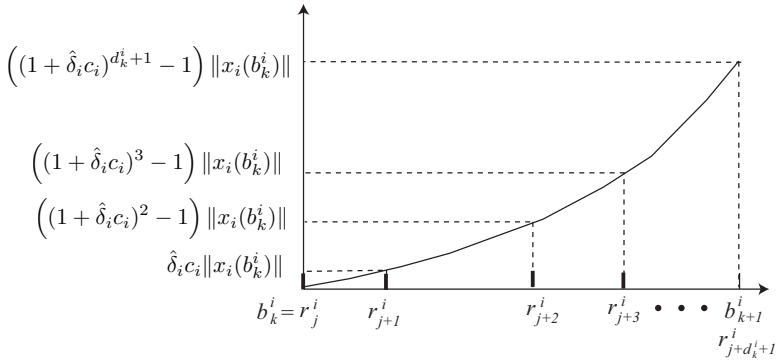


Fig. 1.16 Gap history in the presence of multiple dropouts

Delays only impact the gap in the subinterval between b_{k+1}^i and the finishing time f_{k+1}^i. This case must ensure that the gap does not get larger than the event threshold $c_i\|x_i(b_k^i)\|$. Bounding the allowable length of the interval $[b_{k+1}^i, f_{k+1}^i)$ is done by bounding the gap's rate of growth of the gap by a constant

$$\frac{d}{dt}\|e_i^k(t)\| \leq p_i.$$

This assumption is reasonable if the gap can be shown to evolve over compact sets. Given this rate of growth, p_i, the bound on the admissible *delay* between broadcast and reception will be

$$f_{k+1}^i - b_{k+1}^i \leq \frac{(1-\delta_i)c_i}{p_i}\|x_i(b_k^i)\| = \text{upper bound on } \textit{deadline}.$$

This expression represents the admissible *deadline* by which a network transmission must be received to assure overall system stability.

This section has shown that state-dependent event-triggering can greatly reduce the usage of communication resources in networked control systems. A potential weakness in the existing results is their reliance on state-feedback controllers. How one might extend these formalisms to output feedback controllers is still an open question. One way to begin addressing this question is to first examine the use of event-triggering in state estimation. The following section reviews recent results in this direction.

1.5 Event-Triggered Estimation

This section examines a simple problem involving the use of event-triggering in state estimation. In this case, let's assume that a sensor is observing a discrete-time

process over a finite horizon and computing a *local* estimate of the process state. The problem involves determining when this local estimate should be transmitted to a *remote* observer so that the remote observer's mean square estimation error is minimized subject to a constraint on the transmission rate between the local sensor and remote observer. These event-triggers are therefore referred to as MMSE (minimum mean square error) event-triggers. Problems of this sort are relevant to estimation over wireless sensor networks [91].

Early work on this problem focused on characterizing the impact that intermittently received observations had on the performance of the estimator [68, 47]. A solution to the MMSE event-triggering problem was presented by Imer et al. [30] and by Rabi et al. [59, 61, 57]. Rabi viewed the transmission problem as an optimal stopping problem [35], whereas Imer made use of dynamic programming concepts. This approach can also be applied to control systems [31]. An alternative approach to event-triggered estimation will be found in [67]. This section uses dynamic programming to rederive the results from Rabi's earlier work [59].

It should be noted that MMSE triggers differ in a significant manner from the earlier stability-based triggers derived in sections 1.3 and 1.4. The prior stability-based triggers preserved some desired stability concept such as input-to-state or $\mathscr{L}_2$ stability. The MMSE event-triggers, however, actually *optimize* the estimator's performance subject to a *constraint on transmission frequency*. Recall that one motivation for considering event-triggered systems is that experimental evidence suggests that event-triggering can greatly reduce communication and computational effort while maintaining overall system performance. None of the prior stability-based event-triggers, however, actually show why this should be the case. The MMSE event-triggers suggested in Imer's and Rabi's work, however, explicitly optimize overall estimator system performance subject to a constraint on communication effort. In this way, MMSE event-triggers may shed more light on why event-triggered systems appear to be more efficient in their use of limited computational and communication resources.

Remote Estimation Problem: The event-triggering problem considered in [61] assumes that a sensor is observing a scalar linear discrete-time process over a finite horizon of length $M+1$. The process state $x : [0,1,2,\ldots,M] \to \mathbb{R}$ satisfies the difference equation

$$x_{k+1} = ax_k + w_k$$

for $k \in [0,1,\ldots,M-1]$ where a is a real constant, $w : [0,1,\ldots,M-1] \to \mathbb{R}$ is a sample path for a zero mean white Gaussian noise process with variance Q. The initial state, $x_0 \in \mathbb{R}$, is chosen from a Gaussian distribution with mean μ_0 and variance Π_0. The sensor generates a measurement $y : [0,1,\ldots,M] \to \mathbb{R}$ that is a corrupted version of the process state. The sensor measurement at time k is

$$y_k = x_k + v_k$$

for any integer k between 0 and M where $v : [0,1,\ldots,M] \to \mathbb{R}$ is a sample path of a zero mean white Gaussian noise process with variance R that is uncorrelated with the process noise, w. The process and sensor blocks are shown on the left-hand side of figure 1.17. In this figure the output of the sensor feeds into a transmission subsystem that decides when to transmit information to a remote observer.

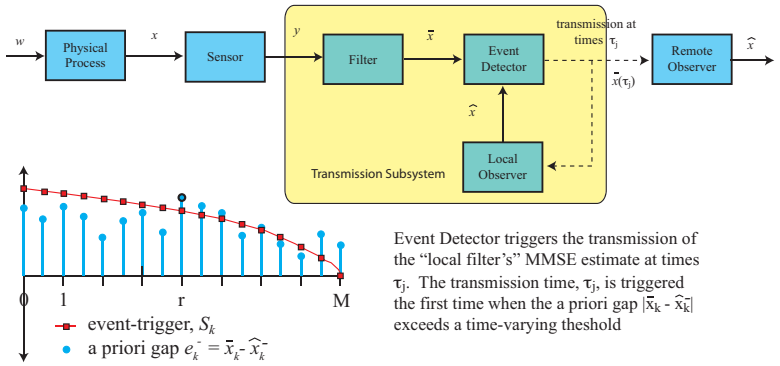


Fig. 1.17 Remote Estimation Problem

This transmission subsystem consists of three components; an *event detector*, a *filter*, and a *local observer*. The event detector decides when to transmit information to the remote observer. It is assumed that the detector is only allowed to transmit at B distinct time instants where B is an integer between 0 to $M+1$, inclusive. The particular transmission times form a sequence $\{\tau_\ell\}_{\ell=1}^{B}$ where $\tau_\ell \in [0,1,\ldots,M]$ denotes the time when the ℓ^{th} consecutive transmission was made. The decision to transmit is based on estimates that are generated by the *filter* and the *local observer* shown in figure 1.17.

The *filter* and *local observer* shown in figure 1.17 generate state estimates that the *event-detector* uses to make its transmission decision. Let $\mathscr{Y}_k = \{y_0, y_1, \ldots, y_k\}$ denote the measurement information available at time k. The *filter* generates a state estimate $\overline{x} : [0,1,\ldots,M] \to \mathbb{R}$ that minimizes the mean square estimation error (MSEE), $E\left[(x_k - \overline{x}_k)^2 \,|\, \mathscr{Y}_k\right]$, at each time step conditioned on all of the sensor information received up to and including time k. These estimates can be computed using a Kalman filter. For the scalar process under study these filter equations are

$$\begin{aligned}
\overline{x}_k &= E\left[x_k \,|\, \mathscr{Y}_k\right] = a\overline{x}_{k-1} + L_k\left(y_k - a\overline{x}_{k-1}\right) \\
\overline{P}_k &= E\left[(x_k - \overline{x}_k)^2 \,|\, \mathscr{Y}_k\right] \\
&= a^2\overline{P}_{k-1} + Q - L_k^2(a^2\overline{P}_{k-1} + Q + R)
\end{aligned}$$

where $\overline{x}_0 = \frac{\Pi_0}{\Pi_0+R} y_0 + \frac{R}{\Pi_0+R}\mu_0$, $\overline{P}_0 = \frac{\Pi_0 R}{\Pi_0+R}$ and $L_k = \frac{a^2\overline{P}_{k-1}+Q}{a^2\overline{P}_{k-1}+Q+R}$.

The event detector uses the *filter's* state estimate, $\overline{x}_k$ at time k, and another estimate generated by the *local observer* to decide when to transmit the filtered state $\overline{x}_k$

to the *remote observer*. Given a set of transmission times $\{\tau_\ell\}_{\ell=1}^{B}$, let $\overline{\mathscr{X}}_k$ denote the information received at the remote observer by time k. In particular, this information set is $\overline{\mathscr{X}}_k = \left\{\overline{x}_{\tau_1}, \overline{x}_{\tau_2}, \ldots, \overline{x}_{\tau_{\ell(k)}}\right\}$ where $\ell(k) = \max\{\ell : \tau_\ell \leq k\}$. The remote observer generates an a posteriori estimate $\hat{x} : [0,1,\ldots,M] \to \mathbb{R}$ of the process state that minimizes the MSEE, $E\left[(x_k - \hat{x}_k)^2 \,|\, \overline{\mathscr{X}}_k\right]$, at time k conditioned on the information received at the remote observer up to and including time k. The a priori estimate at the remote observer, $\hat{x}^- : [0,1,\ldots,M] \to \mathbb{R}$, minimizes the $E\left[(x_k - \hat{x}_k)^2 \,|\, \overline{\mathscr{X}}_{k-1}\right]$, the MSEE at time k conditioned on the information received up to and including time $k-1$. Due to the scalar nature of the process, these estimates take the form,

$$\begin{aligned}
\hat{x}_k^- &= E\left[x_k \,|\, \overline{\mathscr{X}}_{k-1}\right] = a\hat{x}_{k-1} \\
\hat{x}_k &= E\left[x_k \,|\, \overline{\mathscr{X}}_k\right] = \begin{cases} \hat{x}_k^- & \text{don't transmit at time step } k \\ \overline{x}_k & \text{transmit at time step } k \end{cases}
\end{aligned}$$

where $\hat{x}_0^- = \mu_0$.

The event-detection strategy that is used to select the transmission times, τ_ℓ, is based on observing the *gap*, $e_k^- = \overline{x}_k - \hat{x}_k^-$ between the filter's estimate $\overline{x}$ and the remote observer's a priori estimate, $\hat{x}_k^-$. In the following it will be convenient to adopt the following notational conventions,

$$\begin{aligned}
\hat{e}_k &= x_k - \hat{x}_k \quad \text{estimation error at step } k, \\
\overline{e}_k &= x_k - \overline{x}_k \quad \text{filtered state error at step } k, \\
e_k^- &= \overline{x}_k - \hat{x}_k^- \quad \text{a priori gap at step } k, \\
e_k &= \overline{x}_k - \hat{x}_k \quad \text{a posteriori gap at step } k.
\end{aligned}$$

Note that even through the gap is a function of the remote observer's estimate, this signal will be available to the sensor. The sensor has access to this information because the sensor has access to all of the information, $\overline{\mathscr{X}}_k$, that it sent to the remote observer. As a result, the sensor can use another *local estimator* to construct a copy of $\hat{x}$ that can be used locally by the event-detector to compute the gap, e_k^-. This local estimator is shown as part of the transmission subsystem shown in figure 1.17.

The event detector's decision to transmit is triggered when the estimate's gap, e_k^-, leaves a time-varying trigger set, S_k^b, where $k \in [0,1,\ldots,M]$ is the current time and b is the number of transmissions remaining at step k. In general, the trigger sets can be cast as threshold conditions on the estimate's gap. This is shown graphically in the lower left-hand side of figure 1.17. The event-triggers are marked with the squares. The actual gap e_k^- is shown by the solid bullets. Note that the event-triggers are time-varying and equal zero at the end of the time horizon M. Sampling is triggered the first time the gap violates the threshold as shown in figure 1.17. For a given time k, there can be at most $\min\{B, M+1-k\}$ transmissions remaining. The state of the event detector at given time r will be a function of the current a priori gap, e_r^-, and the number of remaining transmissions, T_r. This a priori information at the detector is denoted as the ordered pair, $I_r^- = (e_r^-, T_r)$. In a similar way, the a posteriori information at the detection is denoted as $I_r = (e_r, T_{r+1})$.

Backward Recursion for Value Function: One can use a backward recursion similar to that found in stochastic dynamic programming [20] to determine the triggering sets. Towards this end let's introduce two collections of triggering sets that will be used later. These collections are,

$$\mathscr{S}_r^b(k) = \left\{ S_k^{\max\{0,b-k+r\}}, \ldots, S_k^{\min\{b,M+1-k\}} \right\}$$
$$\mathscr{S}_r^b = \left\{ \mathscr{S}_r^b(r), \ldots, \mathscr{S}_r^b(M) \right\}.$$

These two sets are shown in figure 1.18 for a problem with horizon $M = 4$ and total number of transmissions, $B = 3$. This figure shows the indices for the time steps r and the number of remaining transmissions, b. The collection $\mathscr{S}_1^2$ consists of those indices enclosed within the dotted line. That set is composed of four other collections; $\mathscr{S}_1^2(1)$, $\mathscr{S}_1^2(2)$, $\mathscr{S}_1^2(3)$, and $\mathscr{S}_1^2(4)$. Each of these subcollections is shown as a column of indices enclosed within the rectangles in the figure.

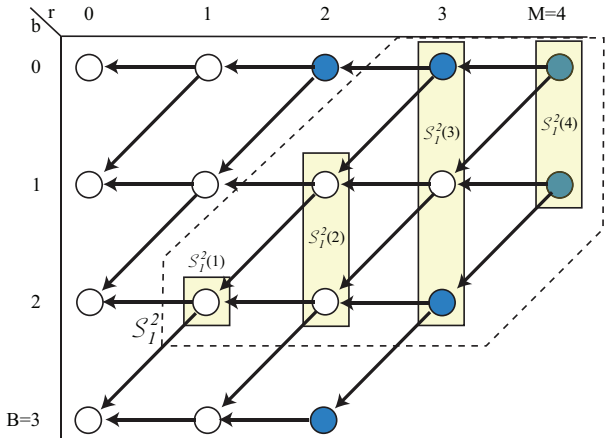


Fig. 1.18 Trigger set collection $\mathscr{S}_1^2 = \bigcup_{k=1}^M \mathscr{S}_1^2(k)$ for $B = 3$ and $M = 4$.

Denote the estimation error at the remote observer as $\hat{e}_k = x_k - \hat{x}_k$. The problem to be solved involves picking the event-triggers in collection $\mathscr{S}_0^B$ to minimize the total MSEE at the remote observer. Formally, the problem is stated as follows

$$\text{minimize: } J(\mathscr{S}_0^B) = E\left[\textstyle\sum_{k=0}^M \hat{e}_k^2\right] \tag{1.26}$$

where the expectation is taken over $\hat{e}_0, \ldots, \hat{e}_M$. The optimal collection is the set $\mathscr{S}_0^{B*}$ such that $J(\mathscr{S}_0^{B*}) \leq J(\mathscr{S}_0^B)$ over all possible $\mathscr{S}_0^B$ and the resulting optimal cost is

$$J^* = \min_{\mathscr{S}_0^B} J(\mathscr{S}_0^B).$$

The problem in equation (1.26) can be treated using results from optimal stochastic control. In our case, the control variables are the collection of triggering sets in $\mathscr{S}_0^B$, rather than some control signal. Since this is a dynamic optimization problem, it can be treated using stochastic dynamic programming. This requires a *value function* that represents the remote observer's total MSEE assuming one uses the optimal triggering sets and assuming the initial information set is $I_r^- = (\zeta, b)$ where ζ is the current value of random variable e_r^- and b is the number of remaining transmissions. In other words, the value function is

$$V(\zeta, b, r) = \min_{\mathscr{S}_r^b} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, I_r^- = (\zeta, b)\right]. \tag{1.27}$$

This is the minimal expected cost conditioned on the information I_r^- available to the event detector at time r. The optimal cost achieved is $J^* = E\left[V(e_0^-, B, 0)\right]$. As will be shown below, this value function can be computed using a backward recursion often found in dynamic programming.

To develop the backward recursion, let's first consider that the event-detector starts at some time after the initial time step 0. In particular, consider an initial state, (ζ, b, r), at time step r in which the a priori gap, e_r^- equals ζ assuming there are b transmissions remaining to be made. Note that e_r^- is a random variable whereas ζ is a specific value for this random variable. From this initial condition, the collection of admissible trigger sets can be described as

$$\mathscr{S}_r^b = \{S_r^b\} \cup \mathscr{S}_r^b(r+1) \cup \cdots \cup \mathscr{S}_r^b(M).$$

This is seen from figure 1.18. The minimization in equation (1.27) may therefore be broken apart as

$$V(\zeta, b, r) = \min_{S_r^b} \left\{ \min_{\mathscr{S}_r^b(r+1), \ldots, \mathscr{S}_r^b(M)} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, I_r^- = (\zeta, b)\right] \right\}.$$

For notational convenience we'll denote the inner minimization shown above as $G(\zeta, b, r)$. In other words,

$$G(\zeta, b, r) = \min_{\mathscr{S}_r^b(r+1), \ldots, \mathscr{S}_r^b(M)} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, I_r^- = (\zeta, b)\right].$$

The computation of $G(\zeta, b, r)$ may be decomposed into two cases. The first case is when $\zeta \in S_r^b$ (i.e. the sensor decides *not* to transmit) and the other case occurs when $\zeta \notin S_r^b$ (i.e. the sensor decides to transmit). Let's outline the computation for the first case below. In particular, when $\zeta \in S_r^b$, one sees that

$$G(\zeta, b, r) = \min_{\mathscr{S}_r^b(r+1), \ldots, \mathscr{S}_r^b(M)} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, e_r^- = \zeta \in S_r^b, T_r = b\right].$$

When no transmission takes place, the information in the conditions in the above equation hold if and only if $I_r = (e_r, T_{r+1}) = I_r^- = (e_r^-, T_r) = (\zeta, b)$. $G(\zeta, b, r)$ may therefore be rewritten as

$$G(\zeta, b, r) = \min_{\mathscr{S}_r^b(r+1), \ldots, \mathscr{S}_r^b(M)} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, I_r = (\zeta, b)\right].$$

Since $T_{r+1} = b$ means that there are b transmissions remaining at step $r+1$, one can conclude that not all of the trigger sets in $\bigcup_{k=r+1}^{M} \mathscr{S}_r^b(k)$ will impact the minimization. In particular, one can disregard the sets S_k^p where p ranges from 0 to $b-1$ and $k = r+b-p$.

This means that the minimization is really done over the set $\mathscr{S}_{r+1}^b$. Let's now compute $G(\zeta, b, r)$ as a function of the value function at $V(e_{r+1}^-, b, r+1)$. In particular, $G(\zeta, b, r)$ may be written as

$$\begin{aligned} G(\zeta, b, r) &= \min_{\mathscr{S}_{r+1}^b} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, I_r = (\zeta, b)\right] \\ &= E\left[\hat{e}_r^2 \,|\, I_r = (\zeta, b)\right] + \min_{\mathscr{S}_{r+1}^b} E\left[\sum_{k=r+1}^{M} \hat{e}_k^2 \,|\, I_r = (\zeta, b)\right] \\ &= \overline{P}_r + \zeta^2 + \min_{\mathscr{S}_{r+1}^b} E\left[\sum_{k=r+1}^{M} \hat{e}_k^2 \,|\, I_r = (\zeta, b)\right]. \end{aligned}$$

Since the information set sequence $\{I_k^-, I_k\}_{k=0}^M$ is Markov and e_{r+1}^- is independent from $\mathscr{S}_{r+1}^b$, the remaining minimization may be rewritten as

$$\begin{aligned} G(\zeta, b, r) &= \overline{P}_r + \zeta^2 + \min_{\mathscr{S}_{r+1}^b} E\left[E\left[\sum_{k=r+1}^{M} \hat{e}_k^2 \,|\, I_{r+1}^- = (e_{r+1}^-, b), I_r = (\zeta, b)\right] \,|\, I_r = (\zeta, b)\right] \\ &= \overline{P}_r + \zeta^2 + \min_{\mathscr{S}_{r+1}^b} E\left[E\left[\sum_{r+1}^{M} \hat{e}_k^2 \,|\, I_{r+1}^- = (e_{r+1}^-, b)\right] \,|\, I_r = (\zeta, b)\right] \\ &= \overline{P}_r + \zeta^2 + E\left[\min_{\mathscr{S}_{r+1}^b} E\left[\sum_{k=r+1}^{M} \hat{e}_k^2 \,|\, I_{r+1}^- = (e_{r+1}^-, b)\right] \,|\, I_r = (\zeta, b)\right] \\ &= \overline{P}_r + \zeta^2 + E\left[V(e_{r+1}^-, b, r+1) \,|\, I_r = (\zeta, b)\right]. \end{aligned}$$

The preceding argument showed that if $\zeta \in S_r^b$, then the term

$$\begin{aligned} G(\zeta, b, r) &= \min_{\mathscr{S}_r^b(r+1), \ldots, \mathscr{S}_r^b(M)} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, e_r^- = \zeta \in S_r^b, T_r = b\right] \\ &= \overline{P}_r + \zeta^2 + E\left[V(e_{r+1}^-, b, r+1) \,|\, I_r = (\zeta, b)\right]. \end{aligned}$$

A similar argument can be used for the case when $\zeta \notin S_r^b$ (i.e. the sensor decides to transmit). In this case it can be shown that

$$\begin{aligned} G(\zeta,b,r) &= \min_{\mathscr{S}_r^b(r+1),\ldots,\mathscr{S}_r^b(M)} E\left[\sum_{k=r}^{M} \hat{e}_k^2 \,|\, e_r^- = \zeta \notin S_r^b, T_r = b\right] \\ &= \overline{P}_r + E\left[V(e_{r+1}^-, b-1, r+1) \,|\, I_r = (0, b-1)\right]. \end{aligned}$$

Combining both of these results determines the following backward recursion for the value function,

$$V(\zeta,b,r) = \min_{S_r^b}\left\{V_{NT}(\zeta,b,r)\mathbf{1}_{\zeta\in S_r^b} + V_T(\zeta,b,r)\mathbf{1}_{\zeta\notin S_r^b}\right\} \tag{1.28}$$

where $\mathbf{1}_{\zeta\in\Omega}$ is the indicator function that takes a value of 1 if ζ is in the set Ω and is zero otherwise. The choice implied in the above equation is between $V_{NT}(\zeta,b,r)$ and $V_T(\zeta,b,r)$ where

$$\begin{aligned} V_T(\zeta,b,r) &= \overline{P}_r + E\left[V(e_{r+1}^-, b-1, r+1) \,|\, I_r = (0, b-1)\right] \\ &= \text{optimal cost if there is a transmission at time step } r \\ V_{NT}(\zeta,b,r) &= \overline{P}_r + \zeta^2 + E\left[V(e_{r+1}^-, b, r+1) \,|\, I_r = (\zeta, b)\right] \\ &= \text{optimal cost if there is no transmission at time step } r. \end{aligned}$$

Note that $V_T(\zeta,b,r)$ is independent of ζ. Because of the properties of the indicator function, the expression given in equation (1.28) is more naturally seen as a *choice* in which the sensor chooses between the smaller of two costs,

$$V(\zeta,b,r) = \min_{S_r^b} G(\zeta,b,r) = \min\left\{V_{NT}(\zeta,b,r), V_T(\zeta,b,r)\right\}.$$

In other words, we have a recursive expression for the optimal cost from the given state (ζ,b,r) and the sensor simply decides to transmit if the cost, V_T, is smaller than not transmitting, V_{NT}.

The computation shown in equation (1.28) is a backward recursion over two sets of indices: the time steps, r, and the number of remaining transmissions, b. In particular, the value function at index (b,r) is a function of the value function at indices $(b-1,r+1)$ and $(b,r+1)$. The functional dependencies are shown in figure 1.18. The arrows in this figure illustrate the functional dependencies implied by equation (1.28).

The initial conditions for this recursion are the value functions at the indices shaded in figure 1.18. The initial values for indices $(0,r)$ where $r \in [B, B+1, \ldots, M]$ are

$$V(\zeta,0,r) = \begin{cases} \frac{Q(M+1-r)}{1-a^2} + \left(\overline{P}_r + \zeta^2 - \frac{Q}{1-a^2}\right)\frac{1-a^{2(M+1-r)}}{1-a^2} & \text{if } |a| \neq 1 \\ \frac{Q(M+r)(M+1-r)}{2} + \left(\overline{P}_r + \zeta^2 - rQ\right)(M+1-r) & \text{if } |a| = 1 \end{cases}. \tag{1.29}$$

These initial conditions are determined by recognizing that $V(\zeta,0,r)$ is the cost assuming that no transmissions occur between steps r and M. The other set of initial conditions are marked by the indices in figure 1.18 that are located along the diagonal. In this case

$$V(\zeta,b,r)=\sum_{k=r}^{M}\overline{P}_k \tag{1.30}$$

for b ranging between 1 and B and $r=M+1-b$. This value function is the MSEE from time step r assuming there is a transmission in each of the remaining time steps.

The value function $V(\zeta,B,0)$ can be computed by recursively determining the value function in sets $\mathscr{S}_0^B(k)$ for k starting at M and ranging backward to 0. The collection

$$\mathscr{S}_0^B(k)=\left\{S_k^{\max\{0,B-k\}},\ldots,S_k^{\min\{B,M+1-k\}}\right\}$$

consists of the indices enclosed by the rectangles in figure 1.18. The value function in set $\mathscr{S}_0^B(M)$ is determined by the initial conditions described above. Using the order of computation implied by the arrows in figure 1.18, it should be apparent that the value function for all nodes in $\mathscr{S}_0^B(M-1)$ can be computed from the known values in $\mathscr{S}_0^B(M)$. In a similar way, one can see that the value function at indices in $\mathscr{S}_0^B(M-2)$ are computed from the values in $\mathscr{S}_0^B(M-1)$. One continues this computation recursively to obtain the value function in $\mathscr{S}_0^B(0)$.

Let's see what's involved in computing the value function and the trigger set S_k^b at index (b,k) which corresponds to time step k with b remaining transmissions. The trigger set, S_k^b, can be chosen to be the symmetric interval $[-\theta_k^b,\theta_k^b]$ where $\theta_k^b\in\mathbb{R}$ is a real positive number that must be computed. In particular, this leads to the MSEE event-trigger where a transmission occurs if $|e_k^-|>\theta_k^b$. If (b,k) are the initial indices shaded in figure 1.18, then the value function is given by the initial conditions in equations (1.29)-(1.30). For other indices, the value function, $V(\zeta,b,k)$, and associated threshold θ_k^b must be numerically computed using the recursion in equation (1.28).

$V(\zeta,b,k)$ is computed numerically at a number of discrete points in the real line. Recall that $V(\zeta,b,k)$ is determined as a choice between the functions $V_{NT}(\zeta,b,k)$ and $V_T(\zeta,b,k)$. These two functions satisfy

$$\begin{aligned}
V_T(\zeta,b,k) &= \overline{P}_k+V_T(\zeta,b-1,k+1)\\
&\quad+\int_{-\theta_{k+1}^{b-1}}^{\theta_{k+1}^{b-1}}\left(V_T(x,b-1,k+1)-V_{NT}(x,b-1,k+1)\right)p(x|0)dx\\
V_{NT}(\zeta,b,k) &= \overline{P}_k+\zeta^2+V_T(\zeta,b,k+1)\\
&\quad-\int_{-\theta_{k+1}^{b}}^{\theta_{k+1}^{b}}\left(V_T(x,b,k+1)-V_{NT}(x,b,k+1)\right)p(x|\zeta)dx
\end{aligned}$$

where $p(x|y)$ is the probability density of e_{k+1}^- conditioned on $e_k = y$. The value of the optimal thresholds θ_{k+1}^b and θ_{k+1}^{b-1} are needed to evaluate these two functions. These thresholds can be computed in a recursive manner.

Given θ_{k+1}^b and θ_{k+1}^{b-1}, the next optimal threshold, θ_k^b, can be found using a bisection search. In particular, the optimal threshold occurs at ζ^* when $V_T(\zeta^*, b, k) = V_{NT}(\zeta^*, b, k)$. So the threshold must satisfy $\theta_k^b = |\zeta|^*$. Once this threshold is determined through the bisection search, then one can see that for $|\zeta| > \theta_k^b$ the value function $V(\zeta, b, k) = V_T(\zeta, b, k)$ and for $|\zeta| \leq \theta_k^b$, the value function must satisfy $V(\zeta, b, k) = V_{NT}(\zeta, b, k)$. This allows one to readily evaluate $V(\zeta, b, k)$ at a number of distinct points, ζ, along the real line.

An example of this computation is provided below for the system

$$\begin{aligned} x_k &= 1.2x_{k-1} + w_k \\ y_k &= x_k + v_k. \end{aligned} \tag{1.31}$$

The mean and covariance of the initial state are 1 and 2, respectively. The covariance of the noise terms, w and v, are both 1. Fix the horizon $M = 8$ and the total number of transmissions $B = 2$. Using the algorithm mentioned above, the value function is evaluated at various values of ζ. Figure 1.19 shows the resulting value function. The solid line in the figure is the value function for various values of time k. The right-hand plot shows $V(\zeta, 1, k)$ and the left-hand plot shows $V(\zeta, 2, k)$. The threshold θ_k^b is marked by the dots in the figure. Outside of the interval defined by the dots one finds that $V(\zeta, b, k) = V_T(\zeta, b, k)$ and this is a constant because $V_T(\zeta, b, k)$ is independent of ζ. Inside the region, the value function varies as a function of ζ.

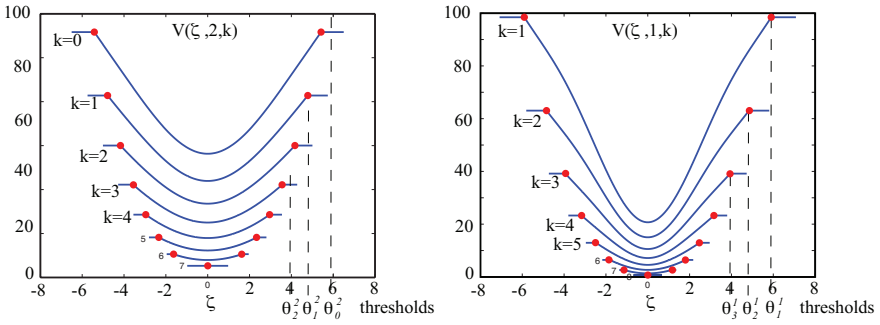


Fig. 1.19 Value functions $V(\zeta, 1, k)$ and $V(\zeta, 2, k)$ for sample system

To see how well the MSEE event-triggers perform, let's vary B from 0 to 9 and repeat the experiment $10,000$ times for both the optimal event-trigger and comparable periodic triggering of transmissions. The plot in figure 1.20 shows the MSEE for the optimal event-triggered and periodically triggered transmissions as a function of the total number of transmissions, B. The plot shows that the experimentally observed

MSEE equals the MSEE predicted by the value function. The plot also shows that the optimal event-triggered transmission strategy always generates a smaller total MSEE than comparable periodically triggered systems.

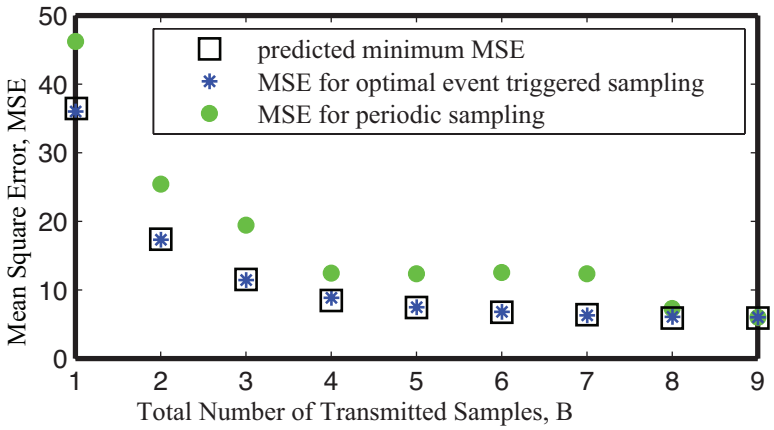


Fig. 1.20 MSE for periodic and event-triggered system

This section has studied the design of MSEE event-triggers for a simple distributed estimation problem. This problem was solved in [30, 61, 58] under a variety of assumptions. The main contribution of this section was a direct derivation of the optimal event-triggers using dynamic programming concepts as well as an explicit method for computing the optimal event-triggering thresholds. The methods in this section recover the original results in [59]. The analysis may be general enough to suggest specific ways of extending the treatment of the scalar system to state estimation of more general linear vector processes.

This section has focused on the estimation problem, but the framework used here may also be extended to control problems. For control, one simply takes the output of the remote estimator and connects it back into the plant through a controller. Real-life applications that fit into this model are found in wireless sensor-actuator networks. The associated control problem that seeks to maximize control performance subject to a communication usage constraint was solved in [87, 88] for the infinite horizon case. In general, the event-triggering thresholds solving the infinite horizon problem are constants. Finite horizon versions of this control problem were treated by [31] and [62]. In the finite horizon case, the event-triggering thresholds are time-varying functions of the initial system state. It has proven difficult, however, to apply this work to vector systems due to the computational complexity associated with solving the dynamic programming equations. Recent progress has been made in resolving the computational complexity issue for infinite horizon problems through the use of quadratic approximations for the value function [19, 18]. A related approach was used in [42] to address the complexity in the finite-horizon estimation problem.

1.6 Event-Triggered Approaches to Optimization

This section introduces an event-triggered distributed algorithm that solves network utility maximization (NUM) problems in large-scale networked systems [79, 78]. Existing distributed algorithms for the NUM problem are gradient-based schemes that converge to the optimal point provided the communication between subsystems is sufficiently frequent. Analytic bounds on the communication interval required to ensure convergence tend to be inversely proportional to certain measures of network complexity such as network diameter and connectivity. As a result, the total message passing complexity in such algorithms can be very great. The event-triggered algorithm presented in this section appears to reduce the message passing complexity by nearly two-orders of magnitude. Moreover, experimental results indicate that this complexity may be independent of network diameter and connectivity.

Related Work: Many problems in networked systems can be formulated as optimization problems. This includes estimation [56, 69, 33], source localization [56], data gathering [15, 14], routing [46], control [77], resource allocation [55, 89] in sensor networks, resource allocation in wireless communication networks [86, 16], congestion control in wired communication networks [36, 44], and optimal power dispatch [38] in electrical power grid. The consensus problem [52] can also be viewed as a distributed optimization problem where the objective function is the total state mismatch between neighboring agents. Many of these problems may be viewed as multi-agent optimization problems that can be solved by a distributed implementation of a subgradient algorithm [49]. In all of these problems, subsystems communicate with each other in order to collaboratively solve a network optimization problem.

Distributed algorithms that solve such network optimization problems include the center-free distributed algorithms [28], distributed asynchronous gradient-based algorithms [72] and distributed subgradient methods [49]. These early algorithms suggest that if the communication between adjacent subsystems is sufficiently frequent, then the state of the network will asymptotically converge to the optimal point. Later developments in such distributed algorithms may be found in the networking community. Most of these later algorithms focus on solving the *Network Utility Maximization* (NUM) problem. The NUM problem maximizes a global separable measure of network system performance subject to linear inequality constraints that are directly related to throughput constraints. This problem originates in congestion control for Internet traffic [36, 44]. The NUM problem, however, has a general form and many problems in other areas can be recast as a NUM problem with little or no variation. As a matter of fact, many of the aforementioned problems can be reformulated as NUM problems.

Among the existing algorithms [36, 44, 85, 53] solving the NUM problem, the dual-decomposition approach proposed by Low et al. [44] is the most widely used. Low et al. showed that their dual-decomposition algorithm was convergent for a step-size that was inversely proportional to two important measures of network size: the maximum path length $\overline{L}$ and the maximum number of neighbors $\overline{S}$. So

as these two measures get large, the step size required to ensure convergence becomes extremely small. Step size, of course, determines the number of computations required for the algorithm's convergence. Under dual-decomposition, system agents exchange information at each iteration, so that step size, γ, also determines the message passing complexity of the algorithm. Therefore if one uses the *stabilizing* step size, dual-decomposition algorithms will have a message passing complexity that quickly scales to unreasonable levels as the network diameter, $\overline{L}$, or the neighborhoood size, $\overline{S}$, increases. In particular, it was shown in [44] that the dual-decomposition is convergent if the step size satisfies

$$0 < \gamma < \gamma^* = \frac{-2\max_{(i,x_i)} \nabla^2 U_i(x_i)}{\overline{L}\overline{S}} \tag{1.32}$$

where $\overline{L}$ is the maximum number of links any user uses, $\overline{S}$ is the maximum number of users any link has, and $U_i(x_i)$ is the utility user i receives for transmitting at rate x_i. For many networked systems this type of message passing complexity may be unacceptable. This is particularly true for systems communicating over a wireless network. In such networked systems, the energy required for communication can be significantly greater than the energy required to perform computation. As a result, it would be beneficial if one can somehow separate communication and computation in these distributed algorithms. This could reduce the message passing complexity of distributed algorithms such as dual-decomposition. This section shows how event-triggering can be used to realize the separation between communication and computation in a primal algorithm solving the NUM problem.

NUM Problem: The NUM problem consists of a network of N users and M links. Let $\mathscr{S} = \{1,\ldots,N\}$ denote the set of users and $\mathscr{L} = \{1,\ldots,M\}$ denote the set of links. Each user generates a flow with a specified data rate. Each flow may traverse several links (which together constitute a route) before reaching its destination. The set of links that are used by user $i \in \mathscr{S}$ will be denoted as $\mathscr{L}_i$ and the set of users that are using link $j \in \mathscr{L}$ will be denoted as $\mathscr{S}_j$. The NUM problem takes the form

$$\begin{array}{l} \text{maximize: } U(x) = \sum_{i\in\mathscr{S}} U_i(x_i) \\ \text{subject to: } Ax \le c, \quad x \ge 0 \end{array} \tag{1.33}$$

where $x = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix}^T$ and $x_i \in \mathbb{R}$ is user i's data rate. $A \in \mathbb{R}^{M\times N}$ is the routing matrix mapping users onto links and $c \in \mathbb{R}$ is a vector of link capacities. The ji^{th} component, A_{ji}, is 1 if user i's flow traverses link j and is zero otherwise. The j^{th} row of Ax represents the total data rates going through link j. This rate cannot exceed the link's capacity c_j. The cost function $U : \mathbb{R}^N \to \mathbb{R}$ is the sum of the user *utility* functions, $U_i : \mathbb{R} \to \mathbb{R}$, for $i = 1,2,\ldots,N$. These utility functions represent the reward, $U_i(x_i)$, (i.e. quality-of-service) that user i receives by transmitting at rate x_i.

A specific example of a NUM problem is shown in figure 1.21. This figure shows a linear network consisting of $M = 5$ links with $N = 3$ users. User 1's route includes links 1-4, user 2's route includes links 2-3, and user 3's route uses link 3-5. Assuming each link has a capacity limit of 1, the throughput constraint therefore becomes,

$$Ax = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = c.$$

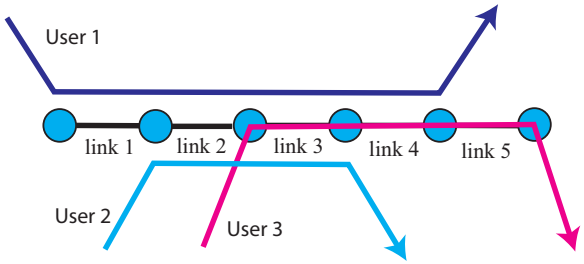


Fig. 1.21 Example of NUM Problem

The solution to the NUM problem maximizes the summed utility seen by all users in the network as a function of the users' transmission rates. These rates must clearly be non-negative. Moreover, if $U_i(x) = \alpha_i \log(x)$ where α_i is a positive constant, then it can be shown [36] that all the user rates in the optimal solution must be positive. In other words, the optimal solution does not result in certain users from being denied access to the network, thereby assuring that all users have *fair* access to network resources.

Augmented Lagrangian Algorithm: While early algorithms used methods based on the dual to the problem in equation (1.33), this section examines a primal *augmented* Lagrangian method. In particular, let's introduce a slack variable $s \in \mathbb{R}^M$ and replace the link constraints ($c_j - a_j^T x \geq 0$ for all $j \in \mathscr{L}$) by the following equality constraint

$$a_j^T x - c_j + s_j = 0, \quad s_j \geq 0, \quad \text{for all } j \in \mathscr{L}.$$

where s_j is called the *slack variable* for the jth constraint. The *augmented cost* then becomes

$$L(x,s;\lambda,w) = -\sum_{i\in\mathscr{S}} U_i(x_i) + \sum_{j\in\mathscr{L}} \lambda_j (a_j^T x - c_j + s_j) + \frac{1}{2} \sum_{j\in\mathscr{L}} \frac{1}{w_j} (a_j^T x - c_j + s_j)^2.$$

Here a penalty parameter w_j is associated with each link constraint and $w = [w_1, \ldots, w_M]$ is the vector of penalty parameters. The other variable λ_j is an estimate of the Lagrange multiplier, λ_j^*, associated with link j's constraint, $c_j - a_j^T x \geq 0$. A vector formed from these estimates is denoted as $\lambda = [\lambda_1, \ldots, \lambda_M]$. The vector $a_j^T = [A_{j1}, \cdots, A_{jN}]$ is the j^{th} row of the routing matrix A.

$L(x,s;\lambda,w)$ is a continuous function of x and s for fixed λ and w. The user rate, x^*, and the slack variable s^*, that minimizes the augmented cost satisfies the following equation

$$\min_{x\geq 0, s\geq 0} L(x,s;\lambda,w) = \min_{x\geq 0}\min_{s\geq 0} L(x,s;\lambda,w) = \min_{x\geq 0} L_p(x;\lambda,w) \tag{1.34}$$

where one defines the *augmented Lagragian function* associated with the NUM problem as

$$L_p(x;\lambda,w) = -\sum_{i\in\mathscr{S}} U_i(x_i) + \sum_{j\in\mathscr{L}} \psi_j(x;\lambda,w)$$

and where

$$\psi_j(x;\lambda,w) = \begin{cases} -\frac{1}{2}w_j\lambda_j^2 & \text{if } c_j - a_j^T x - w_j\lambda_j \geq 0 \\ \lambda_j(a_j^T x - c_j) + \frac{1}{2w_j}(a_j^T x - c_j)^2 & \text{otherwise} \end{cases}.$$

The optimization problem in equation (1.34) is then used as a starting point for developing a recursive procedure that asymptotically approaches the solution of the original NUM problem.

The original NUM problem's solution can be approached arbitrarily closely by solving an appropriately defined sequence of the optimization problems in equation (1.34). This sequence of problems involve minimizing $L_p(x;\lambda[k],w[k])$ for appropriately chosen sequences of penalty parameters, w, and multiplier estimates, λ. Let $x^*[k]$ denote the approximate minimizer for $L_p(x;\lambda[k],w[k])$. It has been shown [6] that for appropriately chosen sequences $\{w[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$, the sequence of approximate minimizers, $\{x^*[k]\}_{k=0}^{\infty}$ converges to the optimal point of the NUM problem. The appropriate choice for these sequences is that for all $j\in\mathscr{L}$

- the sequence of penalty parameters, $\{w_j[k]\}_{k=0}^{\infty}$, is monotone decreasing to zero
- and the sequence of Lagrange multiplier estimates, $\{\lambda_j[k]\}_{k=0}^{\infty}$, is a sequence where

$$\lambda_j[k+1] = \max\left\{0, \lambda_j[k] + \frac{1}{w_j[k]}\left(a_j^T x^*[k] - c_j\right)\right\}.$$

A detailed description of how the sequences $w[k]$ and $\lambda[k]$ are updated in a distributed manner will be found in [78].

A primal algorithm based on the augmented Lagrangian method was developed [78] that converges to the exact minimizer of the NUM problem. In many scenarios, however, it may suffice to obtain an approximate minimizer which can be obtained by considering the problem of minimizing $L_p(x;\lambda,w)$ for a fixed λ and w. In particular, if $\lambda_j = 0$ and w_j is sufficiently small, the minimizer of $L_p(x;\lambda,w)$ will be a good approximation to the solution of the original NUM problem. In this regard the *basic primal algorithm* can be stated as follows

1. **Initialization:** Select any initial user rate $x[0] > 0$. Set $\lambda_j = 0$ and select a sufficiently small $w_j > 0$ for all $j \in \mathscr{L}$.
2. **Recursive Loop:** Minimize $L_p(x;\lambda,w)$ by letting

$$x[k+1] = \max\left\{0, x[k] - \gamma \frac{\partial L_p}{\partial x}(x[k];\lambda,w)\right\} \tag{1.35}$$

for $k = 0,1,\ldots,\infty$.

The smaller w is the more accurate our approximate solution is. The recursion shown in step 2 tries to minimize $L_p(x;\lambda,w)$ using a gradient following method in which γ is a sufficiently small step size. The computations shown above can be easily distributed among users and links.

Event-Triggered NUM Algorithm: Implementing the aforementioned primal algorithm in a distributed manner requires communication between users and links. An event-triggered implementation of the algorithm assumes that the transmission of messages between users and links is triggered by some local error signal crossing a state-dependent threshold. The main problem is to determine a threshold condition that results in message streams ensuring asymptotic convergence to the NUM problem's approximate solution.

The minimizer of the Lagrangian $L_p(x;\lambda,w)$ is searched for using the gradient following recursion in equation (1.35). Assuming that computation is *cheap*, one realizes this gradient recursion as a continuous-time system in which

$$\begin{aligned} x_i(t) &= -\int_0^t \left(\frac{\partial L_p}{\partial x_i}(x(\tau);\lambda,w)\right)^+_{x_i(\tau)} d\tau \\ &= \int_0^t \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j\in\mathscr{L}_i} \mu_j(\tau)\right)^+_{x_i(\tau)} d\tau \end{aligned} \tag{1.36}$$

for each user $i \in \mathscr{S}$ where

$$\mu_j(t) = \max\left\{0, \lambda_j + \frac{1}{w_j}\left(\sum_{i\in\mathscr{S}_j} x_i(t) - c_j\right)\right\}. \tag{1.37}$$

The function $\mu_j : \mathbb{R} \to \mathbb{R}$ is a scalar characterizing how close the jth link constraint is to being active. The link constraint is active at time t when $\mu_j(t) = 0$. Given a function $f : \mathbb{R} \to \mathbb{R}$, its *positive projection* is defined as

$$(f(x))^+_x = \begin{cases} 0 & \text{if } x = 0 \text{ and } f(x) < 0 \\ f(x) & \text{otherwise} \end{cases}.$$

The positive projection used above guarantees that the user rate, $x_i(t)$, is always non-negative along the trajectory.

Equation (1.36) is the continuous-time version of the discrete-time update shown in equation (1.35). Note that in equation (1.36), user i computes its rate based only

on the information from itself and the information of μ_j from those links that are being used by user i. As noted above μ_j characterizes how close the j link constraint is to being active. One may think of μ_j as the j^{th} link's local *state*. From equation (1.37), link j only needs to be able to measure the total flow that goes through itself. Since all of this information is locally available, the update of the user rate in equation (1.36) can be done in a distributed manner.

In equation (1.36), the link state information is available to the user in a continuous manner. Let's consider an *event-triggered* version of equation (1.36). This is done by allowing the user to only access a *sampled* version of the link state. In particular, let's associate a sequence of *sampling* instants, $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$ with the j^{th} link. The time $T_j^L[\ell]$ denotes the instant when the j^{th} link samples its link state μ_j for the ℓ^{th} time and transmits that state to users $i \in \mathscr{S}_j$. One can see that at any time $t \in \mathbb{R}$, the sampled link state is a piecewise constant function of time in which

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell])$$

for all $\ell = 0, 1, \ldots, \infty$ and any $t \in [T_j^L[\ell], T_j^L[\ell+1])$. In this regard, the event-triggered version of equation (1.36) takes the form

$$x_i(t) = \int_0^T \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j \in \mathscr{L}_i} \hat{\mu}_j(\tau) \right)_{x_i(\tau)}^{+} d\tau$$

for all ℓ and any $t \in [T_j^L[\ell], T_j^L[\ell+1])$.

Let's now try to establish conditions on the sampling times $\{T_j^L[\ell]\}$ that ensure the gradient update shown in equation (1.35) is convergent. For notational convenience let the time derivative of the user rate, $x_i(t)$, be denoted as $z_i(t)$. Referring to $z_i(t)$ as the *user state*, one sees that z_i satisfies the equation

$$z_i(t) = \dot{x}_i(t) = \left(\frac{\partial U_i(x_i(t))}{\partial x_i} - \sum_{j \in \mathscr{L}_i} \hat{\mu}_j(t) \right)_{x_i(t)}^{+}$$

for all $i \in \mathscr{S}$. Now we take $L_p(x; \lambda, w)$ as a candidate Lyapunov function. The directional derivative of L_p is

$$\begin{aligned}
\dot{L}_p(x; \lambda, w) &= \sum_{i=1}^{M} \frac{\partial L_p}{\partial x_i} \frac{dx_i}{dt} = -\sum_{i=1}^{N} z_i \left(\frac{\partial U_i(x_i(t))}{\partial x_i} - \sum_{j=1}^{M} \mu_j A_{ji} \right) \\
&\leq -\sum_{i=1}^{N} \left(\frac{1}{2} z_i^2 - \frac{1}{2} \left(\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji} \right)^2 \right) \\
&\leq -\frac{1}{2} \sum_{i=1}^{N} z_i^2 + \frac{1}{2} \sum_{j=1}^{M} \overline{LS} (\mu_j - \hat{\mu}_j)^2.
\end{aligned}$$

To assure that $\dot{L}_p$ is negative definite, one needs to select the sampling times so that

$$\sum_{j=1}^{M} \overline{LS}(\mu_j - \hat{\mu}_j)^2 \leq \sum_{i=1}^{N} z_i^2.$$

This almost looks like one of the state-dependent event-triggers used earlier in section 1.4. Unfortunately, this trigger cannot be implemented in a distributed manner. While the left-hand side is separable over the links, the right-hand side is summed over the users. So the preceding analysis does not give rise to a distributed event triggered algorithm.

To develop local event-triggers that can be easily distributed across the network, let's consider another sequence of times $\{T_i^S[\ell]\}_{\ell=0}^{\infty}$ for each user $i \in \mathscr{S}$. The time $T_i^S[\ell]$ is the ℓ^{th} time when user i transmits its user state to all links $j \in \mathscr{L}_i$. One can therefore see that at any time $t \in \mathbb{R}$, the sampled user rate is a piecewise constant function of time satisfying

$$\hat{z}_i(t) = z_i(T_i^S[\ell])$$

for all $\ell = 0, 1, \ldots, \infty$ and any $t \in [T_i^S[\ell], T_i^S[\ell+1])$. One can now use this sampled user state in our earlier expression for $\dot{L}_p$ to show that

$$\dot{L}(x;\lambda,w) \leq -\frac{1}{2}\sum_{i=1}^{N}\left[z_i^2 - \rho\hat{z}_i^2\right] - \frac{1}{2}\sum_{j=1}^{M}\left[\rho \sum_{i\in\mathscr{S}_j} \frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j - \hat{\mu}_j)^2\right]$$

for some $\rho \in (0,1)$. The derivative, $\dot{L}_p$, is negative definite as long as

$$\begin{aligned} 0 &< \sum_{i=1}^{N}[z_i^2 - \rho\hat{z}_i^2] \\ 0 &< \sum_{j=1}^{M}\left[\rho \sum_{i\in\mathscr{S}_j} \frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j - \hat{\mu}_j)^2\right]. \end{aligned}$$

In this case, both inequalities are separable. The first one is separable over the users and the second one is separable over the links. One can therefore ensure these conditions are satisfied if

$$z_i^2 - \rho\hat{z}_i^2 > 0 \tag{1.38}$$

for each $i \in \mathscr{S}$. This condition can be enforced by requiring that the user transmit z_i at those time instants when the inequality is about to be violated. The other condition is satisfied if

$$\overline{LS}(\mu_j - \hat{\mu}_j)^2 < \rho \sum_{i\in\mathscr{S}_j} \frac{1}{\overline{L}}\hat{z}_i^2 \tag{1.39}$$

for each $j \in \mathscr{L}$. This condition can be enforced by requiring that the link transmit μ_j at those time instants when the inequality is about to be violated. The informal discussion given above therefore establishes that if user/link transmissions are generated using the event-triggers in equation (1.38) and (1.39), then $L_p(x;\lambda,w)$ indeed becomes a Lyapunov function for this system and one can ensure that this system is convergent to a neighborhood of the optimal solution of the NUM problem.

Figure 1.22 shows the event-triggered optimization algorithm in graphical from. This figure uses the system network that was introduced in figure 1.21. In this case each link in the network has an associated router which monitors the total data flowing through the link ($\sum_{i\in\mathscr{S}_j} x_i(t) - c_j$). Attached to each router is a *price agent* that updates the link state μ_j and checks the event-trigger in equation (1.39) to determine whether or not it will transmit its local link state. In a dual manner, each user that is pumping data into the network has an associated *rate agent* that updates the user state $z_i(t)$ and checks the trigger in equation (1.38) to determine when to transmit to the links. One therefore see that the algorithm has both a feedback (link to user) and feedforward path (user to link) in which the information streams are both sporadic in nature.

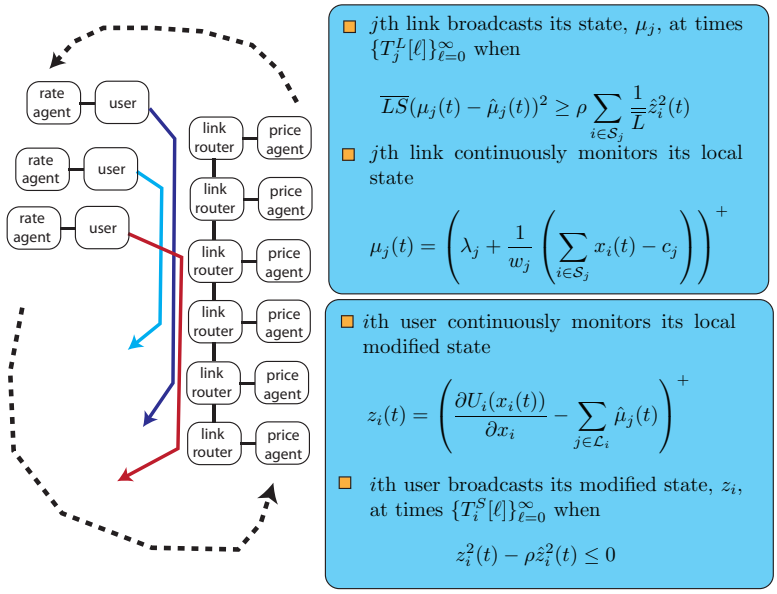


Fig. 1.22 Diagram of the event-triggered primal algorithm

Scaling of Event-Triggered Algorithm: Let's compare the number of message exchanges of the event-triggered algorithm against the dual-decomposition algorithm. Simulation results show that event-triggered algorithms reduce the number of message exchanges by up to two orders of magnitude when compared to dual-decomposition. Moreover, the event-triggered algorithm's message passing com-

plexity scales in a way that appears to be independent of network diameter or connectivity.

Denote $s \in \mathscr{U}[a,b]$ if s is a random variable uniformly distributed on $[a,b]$. Given M, N, $\overline{L}$ and $\overline{S}$, the network used for simulation is generated in the following way. One randomly generates a network with M links and N users, where $|\mathscr{S}_j| \in \mathscr{U}[1,\overline{S}]$ for $j \in \mathscr{L}$ and $|\mathscr{L}_i| \in \mathscr{U}[1,\overline{L}]$ for $i \in \mathscr{S}$. One makes sure that at least one link has $\overline{S}$ users, and at least one user uses $\overline{L}$ links. After the network is generated, a utility function $U_i(x_i) = \alpha_i \log x_i$ is assigned to each user i, where $\alpha_i \in \mathscr{U}[0.8,1.2]$. Link j is assigned capacity $c_j \in \mathscr{U}[0.8,1.2]$. Once the network is generated, dual-decomposition and the event-triggered augmented Lagrangian algorithms are simulated. The optimal rate x^* and its corresponding utility U^* are calculated using a global optimization technique.

Define the error (for all algorithms) as

$$e(k) = \left| \frac{U(x(k)) - U^*}{U^*} \right|$$

where $x(k)$ is the rate at the k^{th} iteration. $e(k)$ is the 'normalized deviation' from the optimal point at the k^{th} iteration. In all algorithms, one counts the number of iterations K for $e(k)$ to decrease to and stay in the neighborhood $\{e(k)|e(k) \le e_d\}$. In dual-decomposition, message passing from the links to the users occurs at each iteration synchronously. So K is a measure of the total number of message exchanges. In the event-triggered algorithms, events occur in a totally asynchronous way. So one adds the total number of triggered events, and divide this number by the link number M. This provides an equivalent iteration number K for the event-triggered algorithms, and is a measure of the total number of message exchanges. One should point out that since these simulations compare a primal algorithm and a dual algorithm, they run at different time scales. Iteration number is then a more appropriate measure of convergence than time [17, 34].

The default settings for the simulation are as follows: $M = 60$, $N = 150$, $\overline{L} = 8$, $\overline{S} = 15$, and $e_d = 3\%$. For all three algorithms, the initial condition is $x_i(0) \in \mathscr{U}[0.01,0.05]$ for all $i \in \mathscr{S}$. In dual-decomposition, initial price $p_j = 0$ for $j \in \mathscr{L}$, and the step size γ is calculated using equation (1.32). In the event-triggered primal algorithm, the parameters are $\rho = 0.5$, $\lambda_j = 0$, and $w_j = 0.01$ for $j \in \mathscr{L}$.

We now consider a Monte Carlo simulation where M, N, and $\overline{L}$ are fixed and $\overline{S}$ is varied from 7 to 26. For each $\overline{S}$, all algorithms were run 1500 times, and each time a random network which satisfies the above specification is generated. The mean m_K and standard deviation σ_K of K are computed for each $\overline{S}$. m_K is used as the criterion for comparing the scalability of both algorithms. The left-hand plot in figure 1.23 plots the iteration number K on a logarithmic scale as a function of $\overline{S}$ for all algorithms. The circles represent m_K for dual-decomposition and the squares correspond to the primal algorithm.

For the primal algorithm, when $\overline{S}$ increases from 7 to 26, m_K does not show noticeable increase. For the primal algorithm, m_K varies between 15.1 and 21.1. For dual-decomposition, m_K increases from 0.3856×10^3 to 5.0692×10^3. Our event-

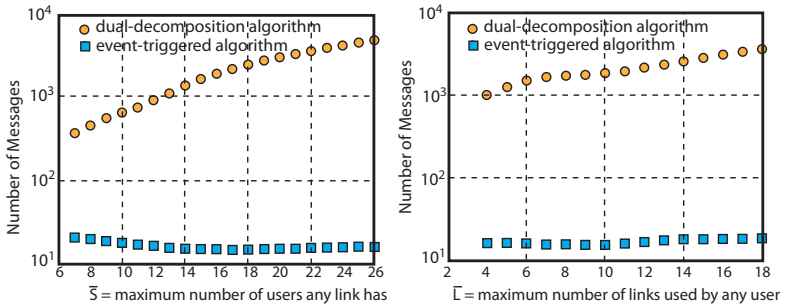


Fig. 1.23 Iteration number K as a function of $\overline{S}$ and $\overline{L}$ for all algorithms.

triggered algorithm is up to two orders of magnitude faster than dual-decomposition. These results also show that the event triggered message passing complexity scales in a manner that is independent of $\overline{S}$. This is stands in stark contrast to the dual-decomposition algorithm which scales superlinearly with respect to $\overline{S}$.

These algorithms were also simulated as a function of $\overline{L}$. In particular, $\overline{L}$ was varied from 4 to 18. The right-hand plot in figure 1.23 plots K (on a logarithmic scale) as a function of $\overline{L}$ for all algorithms. For the primal algorithm, when $\overline{L}$ increases from 4 to 18, m_K increases slowly. In particular, m_K increases from 15.0 to 18.2. For dual-decomposition, m_K increases from 0.9866×10^3 to 3.5001×10^3. The event-triggered algorithm again is up to two orders of magnitude faster than the dual-decomposition.

This section presented a primal event-triggered distributed algorithm for solving network utility maximization problems based on augmented Lagrangian methods. Simulation results suggest that event-triggering greatly reduces the message passing complexity of such distributed optimization algorithms. Optimization of networked systems therefore represents another important application of event-triggering that can be applied to a wide range of applications ranging from traffic control to power dispatch in electrical power grids.

1.7 Research Issues

No chapter of this nature is complete without a discussion of future research issues. Event-triggering represents a new paradigm for real-time feedback control, but the topics covered in this chapter only touch upon what has recently been done. As is often the case, good preliminary work presents just as many questions as it answers and this is certainly the case for event-triggered research as of the writing of this chapter. To help motivate the research issues being raised in this section, let's consider a real-time implementation of a state-dependent event-triggered control system.

There are case studies examining the performance of event-driven control based on static thresholds [64, 63]. There is, however, very little experimental work examining the implementation of the state-dependent event-triggers introduced in sections 1.3 and 1.4. Early work in this direction will be found in [10] in which a self-triggered controller is implemented in a linear analog plant using a real-time kernel. Another early implementation will be found in [13] where the performance of different scheduling protocols for event-triggered controllers on a shared network is investigated. Finally, an experimental study directly comparing the *best* periodic controller to an event-triggered controller will be found in [26].

Figure 1.24 shows results from a recent experiment implementing state-dependent event-triggered feedback-linearizing controllers for the 3 degree-of-freedom (DOF) helicopter system. The plant is a Quanser© 3DOF helicopter controlled by a pentium III PC running the S.H.a.R.K. real-time kernel [22]. In this case, a feedback-linearizing controller was designed for the system with the objective of regulating the travel rate, $\dot{\tau}$, elevation, ε, and pitch ρ of the vehicle. Event-triggered and periodically triggered implementations of this system were implemented in the S.H.a.R.K. kernel and the results from one of these experiments is shown on the right-hand side of figure 1.24.

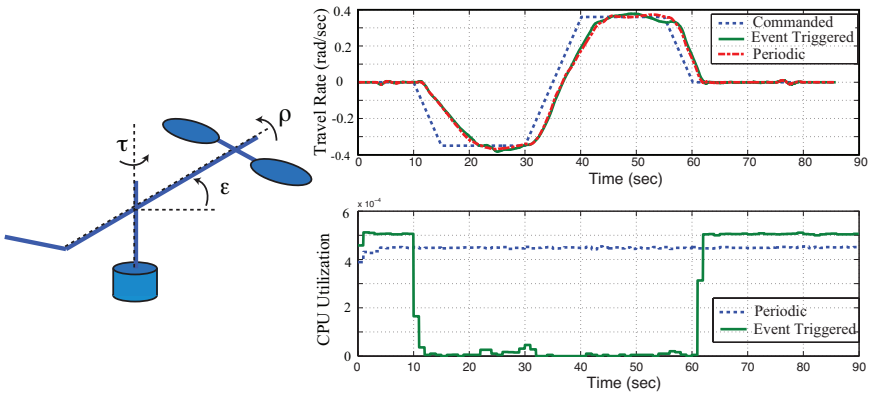


Fig. 1.24 Real-time Hardware Implementation of State-Dependent Event-Triggered System

The top plot in figure 1.24 shows the travel rate as a function of time. The commanded travel rate is shown by the solid dashed line and the other traces show the response of the event-triggered and periodically triggered controllers. What is important to note here is that the behavior is nearly identical in both cases. The bottom plot shows the normalized CPU utilization of the event-triggered and periodically triggered controllers. What one notices here is that when the vehicle is commanded to non-zero travel rates, the event-triggered task's utilization drops considerably. During those periods, however, when the commanded travel rate is near zero (i.e. the vehicle is hovering), the CPU utilization increases and actually exceeds the utilization of the periodically triggered controller.

These results actually confirm what the prior analysis in section 1.3 discovered. In particular, if one looks back at the results from [63] shown in figure 1.1, one sees that event-triggering indeed reduces the overall CPU utilization relative to comparable performing periodic controllers. For the case in [63, 24], however, a uniform event-triggering threshold is chosen so that the system demonstrates considerable chattering when the system is close to its equilibrium point. Under state-dependent event-triggering, however, this type of chattering in the system response does not appear. But because the experiment's input disturbance, w, is wideband sensor noise an excessive number of events are triggered, just as was shown earlier in figure 1.7 of section 1.3. What these results suggest is that state-dependent event-triggering can reduce the jerky behavior seen under the static thresholds used in [63]. The current theory, however, does not adequately balance that gain against the increased use of CPU resources.

With the findings from this experiment in hand, one can now identify a number of important issues that future research into event-triggered feedback must confront. Probably the most immediate is that we develop a better understanding of how to adequately *trade-off control system performance against the reduction in the use of computational or communication resources*. In particular, if one examines the ISS or $\mathscr{L}_2$ event-triggering concepts discussed in sections 1.3 and 1.4, one notes that while the analysis guarantees the preservation of some assumed stability concept, it says almost nothing about the message passing complexity. To be fair, these analyses do bound the minimum sampling period of state-dependent event-triggering. But these bounds are obtained as an afterthought, once the stability-preserving threshold has been determined. What is really needed is an analysis that treats both stability-preserving performance and communication (or computational) resource usage within the same analytical framework. To some extent, this approach was attempted in the event-triggered estimation scheme considered in section 1.5. In that case, the design of the event-trigger was posed as a minimization problem in which the transmission rate between sensor and remote observer was constrained. But that analysis is still far from being mature enough to be applied to real-life systems. The analysis constrains its attention to scalar linear systems and it is unclear how those results might be generalized to vector or nonlinear systems with real-life uncertainties.

Event-triggering samples the system state over time. The focus on constraining communication in section 1.5 can be seen as trying to identify fundamental limits on the rate at which information should be transmitted over the feedback channel. Sampling in *time*, however, is not the only way one can sample a signal. One may also sample the signal in *space*, i.e. quantization. This suggests there should be a close connection between results on minimum quantization feedback control [43] and event-triggered feedback. In particular, an important issue involves a unified approach to *quantization and sampling* in distributed control and estimation problems. Joint quantization and sampling issues were examined in [39], but a full understanding of this relationship has yet to be completed.

Another important issue concerns the development of *event-triggered output feedback controllers*. The experiment shown in figure 1.24 made use of state-

dependent event-triggers that presume full access to the state. In the experimental system, however, the sensors only directly measure the travel angle, τ, elevation angle ε, and pitch angle ρ. For the experiment a periodic task was used to estimate the actual states of the system and then a separate event-triggering task was used to invoke separate controllers for the travel, elevation, and pitch dynamics of the vehicle. This implementation, however, is still far from what one would do in practice. Since most of the computational effort is actually done in the observer task, the true reduction in CPU utilization is very modest for this experiment. To truly realize the benefits of event-triggering, one would need an event-triggered output feedback controller, in which triggering is done solely on the basis of observed sensor measurements, rather than state estimates.

To some extent, the event-triggered estimation methods discussed in section 1.5 provide a first step at developing measurement-based event-triggers. But precisely how this might be integrated into an output feedback system is unclear. One might, for instance, implement an event-triggered observer, whose states are then used to trigger the control action. But this interconnection of an event-triggered estimator and event-triggered controller has not been studied at all. It is unclear whether one can invoke some event-triggered separation principle. As soon as issues regarding observer based control are raised, one must also confront traditional observability and controllability issues. We are aware of no recent work regarding these deeper *system theoretic properties of event-triggered systems*.

Finally, let's return to the implementation questions raised in the experiment. As noted above, the task set in this experiment consists of a hybrid combination of sporadic event-triggered tasks and periodically triggered tasks that work together to realize state-dependent event-triggered controllers. In realizing such hybrid task sets there are always *implementation issues regarding scheduling and fault-tolerance* that need to be addressed. In particular, it is still unclear how best to schedule this mixture of sporadic and periodically triggered tasks to ensure the determinism so often insisted upon in *safety-critical applications*. One reason for insisting on periodically driven task sets in control, is that they provide a highly predictable behavior. When faults do occur, the impact of those faults can be readily analyzed due to the highly deterministic nature of the resulting task environment. This type of deterministic modeling does not seem to be available for the task sets currently used to support event-triggered feedback and as a result it would be highly unlikely that anyone would choose event-triggering for safety-critical applications. This need not be the case, but to establish that event-triggering is suitable for safety-critical applications one must develop a modeling framework whose predictive abilities can provide broad assurances about the fault-tolerant properties of event-triggered systems.

Event-triggered feedback represents an exciting new approach to real-time networked control systems that has the potential of more efficiently using computational and communication resources while assuring high levels of application performance. These applications can be found in control, estimation, and optimization. While the promise of event-triggering is great, there is still significant work remaining to be done. A deeper understanding of the relationship between applica-

tion performance and resource usage must be cultivated. In particular, a close examination must be made of the connection between quantized and event-triggered feedback. The current frameworks must be extended to event-triggered output controllers. This extension will require a deeper understanding of the fundamental system theoretic properties of event-triggered systems, especially as they pertain to the separation between control and estimation. Finally, we must more critically evaluate the scheduling and fault-tolerance of real-time implementations of event-triggered controllers, especially as they pertain to safety-critical applications. Much has already been done, but a great deal remains to be accomplished if event-triggering can indeed be used to build safety-critical real-time networked control systems.

1.8 Acknowledgements

The author gratefully acknowledges the partial financial support of the National Science Foundation (NSF-CNS-07-20457 and NSF-ECCS-0925229). This work grew out of discussions with P. Tabuada (UCLA) , M. Heemels (Eindhoven), A. Cervin (Lund), P. Marti (Catalunya), M. Johansson (KTH), K. Johansson (KTH), and X. Hu (Notre Dame) as well as the hard work of graduate students, X. Wang (UIUC), P. Wan, L. Li, and J. Viramontes-Perez. Finally, the author would like to thank A. Bemporad (Siena) and the European Union's WIDE project for supporting the presentation of this work at the Third WIDE Ph.D. School on Networked Control Systems.

References

1. A. Anta and P. Tabuada. Self-triggered stabilization of homogeneous control systems. In *Proceedings of the American Control Conference*, pages 4129–4134, Seattle, Washington, USA, June 11-13 2008.
2. K-E Arzen. A simple event-based PID controller. In *Procedingsof the 14th World Congress of the International Federation of Automatic Control (IFAC)*, Beijing, P.R. China, 1999.
3. K-E Arzen, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *IEEE Conference on Decision and Control*, volume 5, pages 4865–4870, Sydney, NSW, Australia, December 2000.
4. K.J. Astrom and B.M. Bernhardsson. Comparison of Riemann and Lebesgue sampling for first order stochastic systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 2, pages 2011–2016, Las Vegas, Nevada, USA, December 10-13 2002.
5. L. Bao, M. Skoglund, and K.H. Johansson. Encoder-decoder design for event-triggered feedback control over bandlimited channels. In *American Control Conference*, Minneapolis, Minnesota, USA, 2006.
6. D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.
7. R. Bhattacharya and G.J. Balas. Anytime control algorithm: model reduction approach. *Journal of Guidance, Control and Dynamics*, 27(5):767–776, 2004.
8. G. Buttazzo, G. Lipari, and L. Abeni. Elastic task model for adaptive rate control. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 286–295, 1998.

9. M. Caccamo, G. Buttazzo, and L. Sha. Elastic feedback control. In *IEEE Euromicro Conference on Real-Time Systems (ECRTS)*, 2000.
10. A. Camacho, P. Marti, M. Velasco, and E. Bini. Demo abstract: Implementation of self-triggered controllers. In *Demo Session of 15th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS09)*, San Francisco, California, USA, 2009.
11. D. Carnevale, A.R. Teel, and D. Nesic. A Lyapunov proof of improved maximum allowable transfer interval for networked control systems. *IEEE Transactions on Automatic Control*, 52:892–897, 2007.
12. A. Cervin and J. Eker. Control-scheduling codesign of real-time systems: the control server approach. *Journal of Embedded Computing*, 1(2):209–224, 2004.
13. A. Cervin and T. Henningsson. Scheduling of event-triggered controllers on a shared network. In *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.
14. W.P. Chen, J.C. Hou, L. Sha, and M. Caccamo. A distributed, energy-aware, utility-based approach for data transport in wireless sensor networks. *Proceedings of the IEEE Milcom*, 2005.
15. W.P. Chen and L. Sha. An energy-aware data-centric generic utility based approach in wireless sensor networks. *IPSN*, pages 215–224, 2004.
16. M. Chiang and J. Bell. Balancing supply and demand of bandwidth in wireless cellular networks: utility maximization over powers and rates. *Proc. IEEE INFOCOM*, 4:2800–2811, 2004.
17. M. Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.
18. R. Cogill. Event-based control using quadratic approximate value functions. In *IEEE Conference on Decision and Control*, Shanghai, China, 2009.
19. R. Cogill, S. Lall, and J.P. Hespanha. A constant factor approximation algorithm for event-based sampling. In *Proceedings of the American Control Conference*, New York City, USA, July 2007.
20. W.H. Fleming and R.W. Rishel. *Deterministic and stochastic control*. Springer Verlag, New York, Heidelberg, Berlin, 1975.
21. D. Fontanelli, L. Greco, and A. Bicchi. Anytime control algorithms for embedded real-time systems. In *Hybrid Systems: computation and control*, 2008.
22. P. Gai, L. Abeni, M. Giorgi, and G. Buttazzo. A new kernel approach for modular real-time systems development. In *Proceedings of the 13th IEEE Euromicro Conference on Real-Time Systems*, 2001.
23. W.P.M.H. Heemels, R.J.A. Gorter, A. van Zijl, P. van den Bosch, and S. Weiland. Asynchronous measurement and control: a case study on motor synchronization. *Control Engineering Practice*, 7:1467–1482, 1999.
24. W.P.M.H. Heemels, J.H. Sandee, and P.P.J. van den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590, 2008.
25. W.P.M.H. Heemels, A.R. Teel, N. van de Wouw, and D. Nesic. Networked control systems with communication constraints: tradeoffs between sampling intervals, delays and performance. submitted to the 2009 European Control Conference (ECC), 2009.
26. T. Henningsson and A. Cervin. Comparison of LTI and event-based control for a moving cart with quantized position measurements. In *European Control Conference*, Budapest Hungary, August 2009.
27. T. Henningsson, E. Johannesson, and A. Cervin. Sporadic event-based control of first-order linear stochastic systems. *Automatica*, 44(11):2890–2895, November 2008.
28. YC Ho, L. Servi, and R. Suri. A class of center-free resource allocation algorithms. In *Large Scale Systems Theory and Applications: Proceedings of the IFAC Symposium, Toulouse, France, 24-26 June 1980*, page 475. Franklin Book Co, 1981.
29. D. Hristu-Varsakelis and P.R. Kumar. Interrupt-based feedback control over shared communication medium. Technical Report TR 2003-34, University of Maryland, ISR, 2003.

30. O.C. Imer and T. Basar. Optimal estimation with limited measurements. In *Proceedings of the IEEE Conference on Decision and Control*, Seville, Spain, 2005.
31. O.C. Imer and T. Basar. To measure or to control: optimal control of LTI systems with scheduled measurements and controls. In *American Control Conference*, 2006.
32. A. Isidori. *Nonlinear Control Systems II*. Springer, 1999.
33. B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4705–4710, 2007.
34. B. Johansson, P. Soldati, and M. Johansson. Mathematical Decomposition Techniques for Distributed Cross-Layer Optimization of Data Networks. *Selected Areas in Communications, IEEE Journal on*, 24(8):1535–1547, 2006.
35. Ioannis Karatzas and Hui Wang. Utility maximization with discretionary stopping. *SIAM Journal on Control and Optimization*, 39(1):306–329, 2000.
36. F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.
37. H.K. Khalil. *Nonlinear Systems*. Prentice-Hall, 3rd edition, 2002.
38. BH Kim and R. Baldick. A comparison of distributed optimal power flow algorithms. *IEEE Transactions on Power Systems*, 15(2):599–604, 2000.
39. I. Kofman and J.H. Braslavsky. Level crossing sampling in feedback stabilization under data-rate constraints. In *IEEE Conference on Decision and Control*, San Diego, CA, USA, 2006.
40. D. Lehmann and J. Lunze. Event-based control: a state-feedback approach. In *Proceedings of the European Control Conference*, pages 1716–1721, Budapest, Hungary, 2009.
41. M. Lemmon, T. Chantem, X.S. Hu, and M. Zyskowski. On self-triggered full-information h-infinity controllers. In *Hybrid Systems: computation and control*, Pisa, Italy, July 2007.
42. L. Li and M.D. Lemmon. Optimal event triggered transmission of information in distributed state estimation problems. In *American Control Conference*, Baltimore, MD, USA, 2010.
43. D. Liberzon. On stabilization of linear systems with limited information. *IEEE Transactions on Automatic Control*, 48:304–307, 2003.
44. S.H. Low and D.E. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)*, 7(6):861–874, 1999.
45. C. Lu, J.A. Stankovic, S.H. Son, and G. Tao. Feedback control real-time scheduling: Framework, modeling and algorithms. *Real-time Systems*, 23(1-2):85–126, 2002.
46. R. Madan and S. Lall. Distributed algorithms for maximum lifetime routing in wireless sensor networks. In *IEEE GLOBECOM'04*, volume 2, 2004.
47. A. Matveev and A. Savkin. The problem of state estimation via asynchronous communication channels with irregular transmission times. *IEEE Transactions on Automatic Control*, 48(4):670–676, 2003.
48. M. Mazo and P. Tabuada. On event-triggered and self-triggered control over sensor/actuator networks. In *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.
49. A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.
50. D. Nesic and A.R. Teel. Input-output stability properties of networked control systems. *IEEE Transactions on Automatic Control*, 49(10):1650–1667, 2004.
51. D. Nesic and A.R. Teel. Input-to-state stability of networked control systems. *Automatica*, 40(12):2121–2128, December 2004.
52. R. Olfati-Saber, JA Fax, and RM Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.
53. DP Palomar and M. Chiang. Alternative Distributed Algorithms for Network Utility Maximization: Framework and Applications. *IEEE Transactions on Automatic Control*, 52(12):2254–2269, 2007.
54. E. Polak. Stability and graphical analysis of first order of pulse-width modulated sampled data regulator systems. *IRE Trans. Automatic Control*, AC-6(3):276–282, 1963.

55. Y. Qiu and P. Marbach. Bandwidth allocation in ad hoc networks: A price-based approach. In *Proceedings of IEEE INFOCOM 2003*, volume 2, pages 797–807, 2003.
56. M. Rabbat and R. Nowak. Distributed optimization in sensor networks. *Proceedings of the third international symposium on Information processing in sensor networks*, pages 20–27, 2004.
57. M. Rabi, K.H. Johansson, and M. Johansson. Optimal stopping for event-triggered sensing and actuation. In *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.
58. M. Rabi, G.V. Moustakides, and J.S. Baras. Efficient sampling for keeping track of an Ornstein-Uhlenbeck process. In *Proceedings of the Mediterranean conference on control and automation*, 2006.
59. M. Rabi, GV Moustakides, and JS Baras. Multiple sampling for estimation on a finite horizon. In *Decision and Control, 2006 45th IEEE Conference on*, pages 1351–1357, 2006.
60. M. Rabi, G.V. Moustakides, and J.S. Baras. Adaptive sampling for linear state estimation. submitted to the Siam journal on Control and Optimization, December 2008.
61. Maben Rabi. *Packet based Inference and Control*. PhD thesis, University of Maryland, 2006.
62. Maben Rabi and J.S. Baras. Level-triggered control of a scalar linear system. In *Proceedings of the 16th Mediterranean Conference on Control and Automation*, Athens, Greece, July 2007.
63. J.H. Sandee. *Event-driven Control in Theory and Practice: tradeoffs in software and control performance*. PhD thesis, Technische Universiteit Eindhoven, 2006.
64. J.H. Sandee, W.P.M.H. Heemels, and P.P.J. van den Bosch. Case studies in event-driven control. In *Hybrid Systems: computation and control*, Pisa, Italy, April 2007.
65. J.H. Sandee, P.M. Visser, and W.P.M.H. Heemels. Analysis and experimental validation of processor load for event-driven controllers. In *IEEE Conference on Control and Applications (CCA)*, pages 1879–1884, Munich, Germany, 2006.
66. D. Seto, J.P. Lehoczky, L. Sha, and K.G. Shin. On task schedulability in real-time control systems. In *IEEE Real-time Technology and Applications Symposium (RTAS)*, pages 13–21, 1996.
67. J. Sijs and M. Lasar. On event based state estimation. In *Hybrid Systems: computation and control*, volume 5469 of *Lecture Notes in Computer Science*, pages 336–350, 2009.
68. B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004.
69. A. Speranzon, C. Fischione, and K.H. Johansson. Distributed and Collaborative Estimation over Wireless Sensor Networks. *Proceedings of the IEEE Conference on Decision and Control*, pages 1025–1030, 2006.
70. P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, September 2007.
71. P. Tabuada and X. Wang. Preliminary results on state-triggered scheduling of stabilizing control tasks. In *IEEE Conference on Decision and Control*, 2006.
72. J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.
73. Y.Z. Tsypkin. *Relay Control Systems*. Cambridge University Press, 1984.
74. A. J. van der Schaft. *L2-gain and passivity techniques in nonlinear control*. Springer, 2000.
75. M. Velasco, P. Marti, and J.M. Fuertes. The self triggered task model for real-time control systems. In *Work-in-Progress Session of the 24th IEEE Real-time Systems Symposium (RTSS03)*, Cancun, Mexico, December 2003.
76. P. Voulgaris. Control of asynchronous sampled data systems. *IEEE Transactions on Automatic Control*, 39(7):1451–1455, 1994.
77. P. Wan and M. Lemmon. Distributed Flow Control using Embedded Sensor-Actuator Networks for the Reduction of Combined Sewer Overflow (CSO) Events. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 1529–1534, 2007.

78. P. Wan and M.D. Lemmon. Distributed network utility maximization using event-triggered augmented lagrangian methods. In *Proceedings of the American Control Conference*, St. Louis, MO, USA, June 2009.
79. P. Wan and M.D. Lemmon. Event-triggered distributed optimization in sensor networks. In *Information Processing in Sensor Networks (IPSN)*, San Francisco, California, USA, April 2009.
80. X. Wang and M.D. Lemmon. Decentralized event-triggered broadcasts over networked control systems. In *Hybrid Systems: computation and control*, St. Louis, Missouri, USA, April 2008.
81. X. Wang and M.D. Lemmon. Event-triggered broadcasting across distributed networked control systems. In *Proceedings of the American Control Conference*, Seattle, Washington, USA, June 2008.
82. X. Wang and M.D. Lemmon. Event-triggering in distributed networked control systems. submitted to the IEEE Transactions on Automatic Control, February 2009.
83. X. Wang and M.D. Lemmon. Self-triggered feedback control systems with finite-gain l2 stability. *IEEE Transactions on Automatic Control*, 54(3):452–467, March 2009.
84. X. Wang and M.D. Lemmon. Self-triggered feedback systems with state-independent disturbances. In *Proceedings of the American Control Conference*, St. Louis Missouri, USA, June 2009.
85. JT Wen and M. Arcak. A unifying passivity framework for network flow control. *IEEE Transactions on Automatic Control*, 49(2):162–174, 2004.
86. L. Xiao, M. Johansson, and SP Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52(7):1136–1144, 2004.
87. Y. Xu and J.P. Hespanha. Optimal communication logics in networked control systems. In *Proceedings of the IEEE Conference on Decision and Control*, volume 4, pages 3527–3532, Nassau, Bahamas, 2004.
88. Yonggang Xu and Joao P. Hespanha. Communication logic design and analysis for networked control systems. In L. Menini, L. Zaccarian, and C.T. Abdallah, editors, *Current Trends in Nonlinear Systems and Control*, Systems and Control: Foundations and Applications, pages 495–514. Birkhauser Boston, 2006.
89. Y. Xue, B. Li, and K. Nahrstedt. Optimal resource allocation in wireless ad hoc networks: a price-based approach. *IEEE Transactions on Mobile Computing*, 5(4):347–364, 2006.
90. W. Zhang, MS Branicky, and SM Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–99, 2001.
91. B. Zhu, B. Sinopoli, K. Poolla, and S. Sastry. Estimation over wireless sensor networks. In *American Control Conference*, pages 2732–2737, 2007.