

Chapter 2

Anticipative Control Design in Internet-Like Networks

Abstract This chapter presents the analysis and the design of anticipative controllers for packet-based NCS. The remote controller uses a model of the plant and a basis controller to compute a sequence of future control actions to compensate the effect of delays and packet dropouts. This sequence is stored into the actuator buffer and is applied synchronously at each sampling time. Two middleware layers between the process and the network, and between the controller and the network are designed to hide the elements which do not belong to a conventional control loop. First, a scheme in which the sensor sends the measurements periodically is presented, and an event-based approach is proposed afterwards for a more efficient usage of the network bandwidth. The system results to be GUUB when some constraints are imposed to the network delay. Different extensions such as disturbance estimators, output measurement and LTI anticipative controllers are discussed, preserving the GUUB property of the system. Finally, a centralized remote controller for a multi-loop architecture is presented.

2.1 Introduction

While conventional control loops are designed to work with circuit-switching networks, where dedicated communication channels provide almost constant bit rate and delay, networks such as the Internet are based on packets, carrying larger amount of information at less predictable rates.

One aspect inherent to packet-based networks is transmission overhead. Packets can be split into the header and the payload, which may be filled with useless information to reach the minimum packet size. As a consequence, transmitting a few bits per packet has essentially the same bandwidth cost as transmitting hundreds of them. Thus, rather than sending individual values, finite-length signal predictions can be transmitted. This is the motivation of the so called *packet-based control* [GT04, ZLR09] or *receding horizon control* [QSG07].

To achieve this, a common approach is to use model-based control to emulate future states of the plant and, therefore predictions for the control signal. The idea of combining packet-based control and Model Predictive Control (MPC) was first

introduce in [Bem98] in the context of teleoperation. Since then, other authors have exploited the principle of MPC in packet-based NCS [KJA06, QSG07, MJVR08, VF09, ICMS11].

The influence of the model uncertainty of model-based NCS was studied in [MA04]. In [CB08], the constraints imposed by communication protocols on state measurement access are addressed. This work is extended to nonlinear systems in [GCB12].

Among the alternatives studied to prevent the computational effort required by MPC, the anticipative controller estimates the future state of the system based on a model that considers delays [NH06]. Anticipative controllers and the use of actuator buffers have been proposed for packet-based NCS in [ESDCM07, GSD11] for different network architectures.

Whereas these approaches results in a more efficient usage of the network bandwidth and possible enlargement of transmission intervals, few publications have combined receding horizon control and event triggering. In [ESDCM07], a transmission protocol named as Input Difference Transmission Scheme (IDTS), that calculates a new control sequence at every time step but only transmits to the actuator when the difference between the new sequence and that in the buffer has exceeded a threshold. In [GDJ+11] the sensor sends measurements to the controller if the difference between the predicted state by a model, which is sent with the predictions of the control signal, and the measured state crosses a given level. More recently, a model-based periodic event-triggered control is exploited to reduce the number of transmissions [HD13], where two frameworks are proposed, perturbed linear and piecewise linear systems, to achieved global exponential stability and ℓ_2 gain performance.

The outline of this chapter is as follows. The original contributions are given in Sect. 2.2. Section 2.3 states the assumptions that are taken in this chapter. The guidelines of the design of middleware layers are given in Sects. 2.4 and 2.5, which are adapted to an event-triggered scheme in Sect. 2.6. The stability of the system is studied in Sects. 2.7–2.10 present different extensions such as disturbance estimation, output measurement, and the centralized control of N loops. Finally, conclusions end the chapter.

2.2 Contributions of This Chapter

In this chapter, a middleware approach is proposed for networked control systems. Two adaptation layers made up the novel design. The first layer is in between the process and the network and the second one serves as an interface between the network and the controller. The use of event-triggering is incorporated in the design in order to reduce the transmission frequency. The controller generates sequences of future control actions and states of the plant and sends them to the process, where the corresponding middleware layer decides which element is used at each sampling time.

One of the novel contributions is that the proposed design is aware of a more efficient usage of the bandwidth but also of facing delays and packet losses without assuming clock synchronization of the elements in the control loop, in contrast to other works in the literature such as [NH06, QSG07, QSG08, ZLR09, HD13]. Moreover, the model-based controller alleviates the additional delays caused by the computational time required by MPC, and so the proposed approach seems especially adequate in processes with fast dynamics. Also, the theoretical analysis ensures the stability of the system if the network delay is upper bounded.

Another contribution of this chapter is the design of event-triggering for output measurements that combines the two existing approaches in the literature: estimation of the state by an observer or a filter, and the use of a different controller to full state feedback. The goal is to overcome the limited computation of the sensor and the actuator and the lack of synchronization between the controller and the process. LTI controllers and a Luenberger observer are combined to preserve the stability of the system when full measurement cannot be assumed.

The disturbance estimator proposed in [LL10] is adapted to the remote controller scheme and improved in the sense that the matrix A does not require to be invertible and the model uncertainty can be also partially compensated.

Finally, another original contribution is the centralized anticipative-controller design when decentralized control cannot be implemented due to computation constraints in the elements of the control loop. The effectiveness of the centralized approach is analyzed and we show that the same performance than for periodic implementations and a single plant can be achieved with this approach if the number of processes is not large.

2.3 Assumptions

In the sequel of this chapter the following assumptions hold:

- *System architecture*: There is a single control loop with a remote controller, i.e., Fig. 1.4c. We assume that the sensor and the actuator have a very limited computation capacity and the controller is the element which makes the computation effort. The actuator processes the incoming packets and store the data into a buffer. The sensor measures the state at each sampling time and is able to compare it to a reference value and, in case, to trigger an event. The remote controller also has a buffer to store the incoming measurements.
- *System dynamics*: We consider linear plants and a sampling period denoted by T_s , that meets Nyquist criteria. Thus, the system dynamics is given by

$$x(k+1) = A_d x(k) + B_d u(k) + w(k), \quad x(0) = x_0 \quad (2.1)$$

$$y(k) = Cx(k) + v(k), \quad (2.2)$$

where $x(k) \in \mathbb{R}^n$ is the state, $u(k) \in \mathbb{R}^m$ is the control signal, $w(k) \in \mathbb{R}^n$ is the disturbance, and $v(k) \in \mathbb{R}^r$ is the measurement noise, both of which are bounded. Matrices (A_d, B_d) are obtained from a continuous model (A, B) for the sampling period T_s

$$A_d = e^{AT_s} \quad (2.3)$$

$$B_d = \int_0^{T_s} e^{As} B ds. \quad (2.4)$$

The pair (A, B) is controllable.

- *Measures of time*: All the instances or intervals of time, such as delays, the occurrence of events, etc. are given as integer numbers $k \in \mathbb{N}$, so that the measurements in units of time are t_k which are multiple of the sampling period, i.e., $t_k = kT_s$.
- *Controller*: The basis controller is assumed to be state feedback if the full state is measurable, and LTI if only the output is measured. However, the framework can be extended to other controllers that stabilize the plant in a network-free system configuration.
- *Clocks synchronization*: The elements at the plant side (sensor, actuator and event detector¹) are ruled by the same clock and hence, clock synchronization is assumed for them. By contrast, the remote controller clock is not synchronized with any of the other elements in the control loop and works asynchronously. When a new measurement is received by the controller, it computes a new sequence of future control values.
- *Network*: We consider Internet-like networks. Hence, network protocols such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) can be used, and so the size of the packets payload is around 500 bytes [Eva98]. It is assumed that there are not powered-batteries devices so that energy consumption is not affected by the packet length. Moreover, packets can experience delays or be lost during transmissions across the network.

Example 2.1 A simple chronogram is shown in Fig. 2.1 to illustrate the phenomenon of delays and packet losses. The system is sampled at discrete time instances $k, k + 1, \dots$. The transmission of measurements from the sensor to the controller can be delayed by a quantity denoted as τ_{sc} . Information sent from the controller to the actuator can also suffer from delay τ_{ca} . The Round Trip Time (RTT) denotes the number of sampling times that takes data to go from the process to the controller and back to the process, i.e.,

$$\text{RTT} = \min\{l : l \in \mathbb{N}, \tau_{sc} + \tau_{ca} < lT_s\}.$$

¹The event detector can be a software or hardware component to determine the time instances of the occurrence of an event.

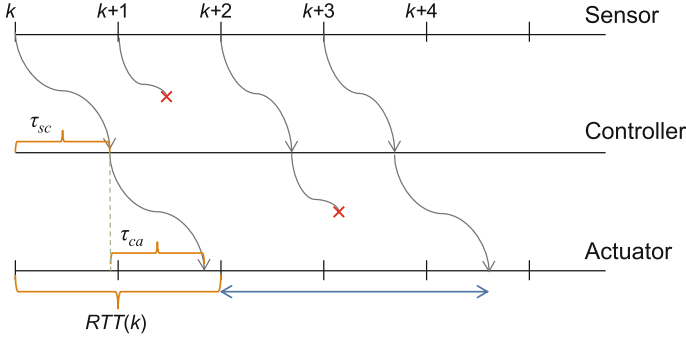


Fig. 2.1 Examples of delays and data dropouts

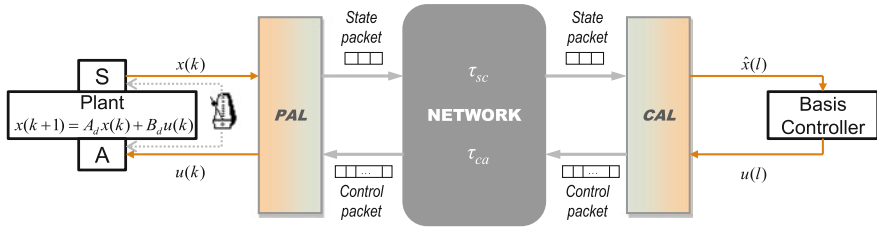


Fig. 2.2 Proposed architecture for packet-based NCS

Data can also be dropped as depicted in Fig. 2.1 between $k + 1$ and $k + 3$. As a consequence, the actuator does not received updated control inputs in the interval marked in blue.

The proposed architecture is shown in Fig. 2.2. On the left side, the linear plant is sampled according to a sampling period T_s at discrete instances of time k . On the right hand side, the basis controller computes control signals for incoming states (real measurements or estimations based on previous measurements) denoted by $\hat{x}(l)$, where in general $l \neq k$. The two intermediate elements between the plant and the network and the network and the basis controller, respectively, are two *middleware layers*. The concept of middleware for NCS is described in [GBK09] and it is used here to separate all the elements in the design from the classical components of a control loop.

We next briefly describe these two layers:

- The Controller Adaptation Layer (CAL) receives and processes the state packets sent from the plant side. Its main tasks are to estimate the future states of the plant and to interact with the basis controller to compute the sequences of future control actions. Hence, the main element of this layer is the model of the plant.
- The Process Adaptation Layer (PAL) receives the control packets and decides which control input is applied at each sampling time. Also, in an event-based

approach, it decides when to transmit the measurements to the controller through the network.

The *state packets* basically contain measurements taken from the plant. The structure of the *control packets* is described later in the chapter but the main element is the sequence of future control actions.

2.4 The Controller Adaptation Layer (CAL)

This section describes how the CAL works. The tasks carried out by this layer include the processing of packets, the update of a parameter that we denote by the Round Trip Time (RTT), the interaction with the basis controller to generate future control actions (anticipative controller), and the sending of the control packets.

Consider the discrete-time plant (2.1) and (2.2). Assume that the discrete-time model used by the anticipative controller is given by

$$\hat{x}(k+1) = \hat{A}_d \hat{x}(k) + \hat{B}_d u(k), \quad (2.5)$$

where $\hat{x}(k) \in \mathbb{R}^n$ is the estimated state. We assume that in general $A_d \neq \hat{A}_d$, $B_d \neq \hat{B}_d$ and we denote the model uncertainty as $\bar{A}_d = \hat{A}_d - A_d$, $\bar{B}_d = \hat{B}_d - B_d$.

Future states of the plant are estimated by this model in two different steps after the reception of a new state packet that we describe next.

2.4.1 Packets Processing

The processing of the incoming packets is depicted in Fig. 2.3. The packet payload is extracted and interpreted according to a given structure. Specifically, the payload of the state packets includes the following information:

- The measured state $x(k)$.
- The plant local time k .
- A time stamp TS_u of the controller local time. TS_u allows to identify the control sequence $\mathbf{U}_h$, $h < k$, which was being applied at the time of the measurement of $x(k)$.
- An index i_u . If the computed control sequences have a size of Q elements, i_u is the number of element of the sequence $\mathbf{U}_h$ which was being applied at the time of the measurement of $x(k)$.

A second type of packets is also received. Every time a control packet is received by the plant and before processing it, a small time-stamped packet is sent back to the CAL which uses this time stamp to update the value of the RTT and its minimum value denoted by τ_{min} . $\tau_{min} \in \mathbb{N}$ gives the fastest transmission from the controller to

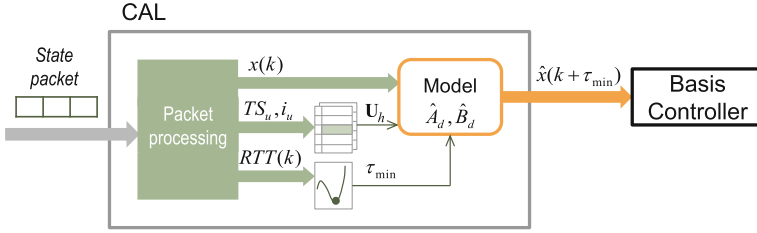


Fig. 2.3 Packets processing by the CAL layer

the plant and the other way back:

$$\tau_{min} = \min\{RTT(k), \forall k \in \mathbb{N}\}.$$

The first action is to check that k is a subsequent time to previous processed packets. If this condition is fulfilled, the state of the plant at time $k + \tau_{min}$ is estimated by the model using the aforementioned information, and $\hat{x}(k + \tau_{min})$ is taken as the initial state to compute the next control sequence.

2.4.2 Control Sequence Computation

Definition 2.1 The control sequence $\mathbf{U}_k$ is a set of Q future control values that are calculated based on the system model (2.5) for the state packet containing $x(k)$ and received by the controller after the transmission through the network from the sensor to the controller.

Once $\hat{x}(k + \tau_{min})$ is estimated based on the information received, the control input for this state is computed. Let us first assume a state feedback control law, so that

$$u(k + \tau_{min}) = K\hat{x}(k + \tau_{min})$$

is the first element of the control sequence. Thus, the state estimation for the next sampling time is

$$\hat{x}(k + \tau_{min} + 1) = \hat{A}_d\hat{x}(k + \tau_{min}) + \hat{B}_du(k + \tau_{min}) = (\hat{A}_d + \hat{B}_dK)\hat{x}(k + \tau_{min}).$$

The model and the basis controller interact $Q - 1$ times to generate the control sequence $\mathbf{U}_k$ of size Q . In general, the $j + 1$ element of $\mathbf{U}_k$ can be written as

$$\mathbf{U}_k(j + 1) = u(k + \tau_{min} + j) = K(\hat{A}_d + \hat{B}_dK)^j\hat{x}(k + \tau_{min}), \quad 0 \leq j \leq Q - 1.$$

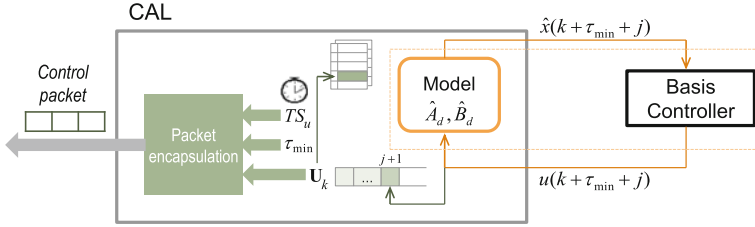


Fig. 2.4 Control sequence computation by the CAL layer

This process is depicted in Fig. 2.4. The value $u(k + \tau_{min} + j)$ is computed based on the estimation of $\hat{x}(k + \tau_{min} + j)$, it is used to estimate the state at the next sampling time $\hat{x}(k + \tau_{min} + j + 1)$, and it is saved as the $j + 1$ element of $\mathbf{U}_k$. When this process is completed, the control sequence is saved in a look-up table indexed by a time stamp, and the packet is encapsulated. The time stamp of the controller local time TS_u and the value of τ_{min} are also included in the packet.

2.5 The Process Adaptation Layer (PAL)

On the plant side, the PAL layer determines the control signal to apply. The packets received from the controller between two consecutive sampling times are enqueued (a priori, more than one packet can arrive). As they can arrive out of order, they are time-stamped to distinguish which control sequence was calculated last. The latest computed control sequence is stored in a buffer, and the rest of the packets are discarded because they contain obsolete values calculated with prior states of the plant. Thus, there is a queue for the incoming packets and one buffer which contains the current control sequence that is being handled.

The previous section pointed out that the first element of the control sequence for the sampling time k is calculated based on an estimated state $\hat{x}(k + \tau_{min})$. However, the time between the measurement of the state $x(k)$ and the reception of the control sequence $\mathbf{U}_k$ will generally be greater than τ_{min} . Let us denote this elapsed time as $\tau(k)$. The value of $\tau(k)$ is measured by subtracting k from the value of the local clock and it is compared to τ_{min} . The difference reveals how many sampling times have passed, or how many elements of $\mathbf{U}_k$ should be discarded because they are obsolete values. This value is denoted as i_0 ($i_0 = \tau(k) - \tau_{min}$). The first i_0 elements of $\mathbf{U}_k$ are then discarded, and the $i_0 + 1$ element is the first element to apply. Figure 2.5 depicts these actions taken by the PAL.

As the incoming packets queue is checked at each sampling time, if a new packet does not arrive the next element of the control sequence stored in the buffer is applied. Thus, the received control sequence is applied synchronously at each sampling time until a new one is received. In general, at any time we have:

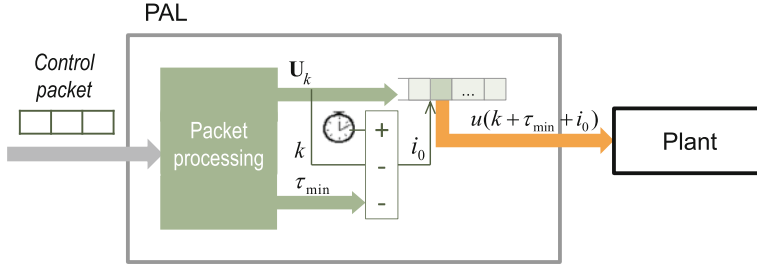


Fig. 2.5 Packets processing by the PAL layer

$$u = \begin{cases} U_k(i_0 + 1) & \text{if } U_k \text{ received in the last sampling period} \\ U_k(i_0 + 1 + j) & \text{if (no newer packet received) AND } (i_0 + 1 + j < Q) \\ U_k(Q) \text{ OR } 0 & \text{otherwise,} \end{cases} \quad (2.6)$$

where $j \in \mathbb{N}^+$ denotes an index that is incremented at each sampling time if a new packet is not received.

The last case of (2.6) shows that either zero or the previous control value is applied when the last element of the control sequence $U_k(Q)$ is reached. This choice depends on the process dynamics.

Example 2.2 Assume that the sampling period for a given plant is $T_s = 5$ ms and that, at a given instance of time, the value of τ_{min} is of two sampling periods, i.e., 10 ms. Thus, for a measurement $x(k)$ received by the controller, the value of $\hat{x}(k+2)$ is estimated and the control sequence U_k computed as described before. Assume that when U_k is received at the PAL, the local time is $k+5$, that is, between the measurement of $x(k)$ and the reception of the control sequence, five sampling periods have passed (25 ms). Hence, the first three elements of U_k are discarded and $U_k(4)$ is applied since $i_0 = (k+5) - k - \tau_{min} = k+5 - k - 2 = 3$.

The PAL is also in charge of sending the state packets to the controller. When the transmission is periodic, this takes place at each discrete time k . However, if the information is transmitted in an event-based fashion, an event detector has to be included in the scheme. We next present the changes in both the CAL and PAL layers to support the event-based policies.

2.6 Event-Based Anticipative Control

In event-based policies, an event is detected when a certain condition is violated. Thus, let us define the assumptions that we have taken in the design:

- **Error:** The error $e(k)$ is defined as the difference between the estimated state $\hat{x}(k)$ and the current measurement state $x(k)$ at the sampling time k . That is

$$e(k) = \hat{x}(k) - x(k). \quad (2.7)$$

- *Trigger function*: Let us denote the trigger function as $f(e(k))$. It detects the occurrence of an event when its value crosses zero from negative to positive. Thus, the trigger condition is $f(e(k)) \geq 0$. For instance, if we want to bound the error by a constant threshold, the trigger function turns to be

$$f(e(k)) = \|e(k)\| - c, \quad (2.8)$$

where $\|\cdot\|$ is the euclidian norm and c is the constant threshold.

Furthermore, we denote by $k_i, i \in \mathbb{N}$ the discrete time instances k at which an event is detected.

It is assumed that the constant c is chosen and the process is sampled fast enough so that the event detection is precise and $\|e(k_i)\| \approx c$. Note that being strict, the equality can be ensured just in continuous time.

- *Event detector*: The event detector is the software element which monitors the trigger condition. This element processes the measurements acquired by the sensor and, when an event is detected, the measurements are sent to the controller.

2.6.1 CAL Design for Event-Based Control

According to the assumptions above, an event is detected when the norm of the difference between the state predicted by the model and the actual state crosses a certain threshold (see (2.8)). Thus, the predictions of the model must be available at each sampling time at the plant side. Since this information is computed by the controller, it has to be transmitted through the network and included in the control packets.

Definition 2.2 The predicted states sequence $\hat{\mathbf{X}}_k$ is a set of Q future plant states predicted by the model (2.5). The j th element of $\hat{\mathbf{X}}_k$ corresponds to the state given by the model (2.5) after applying the j th element of the control sequence $\mathbf{U}_k$, i.e.,

$$\hat{\mathbf{X}}_k(j) = (\hat{A}_d + \hat{B}_d K)^j \hat{x}(k + \tau_{min}).$$

Furthermore, since measurements are only transmitted to the controller at $k_i, i \in \mathbb{N}$, i.e., when an event occurs, the predicted states and control sequences can be denoted as $\hat{\mathbf{X}}_{k_i}$ and $\mathbf{U}_{k_i}$, respectively.

Figure 2.6 illustrates the new design of the CAL. At each iteration between the model and the controller, a new element is added to both $\hat{\mathbf{X}}_{k_i}$ and $\mathbf{U}_{k_i}$. When the computation is completed, the information is encapsulated as a new control packet.

Remark 2.1 Note that the length of the control sequence $\mathbf{U}_{k_i}$ has to be cut down to include $\hat{\mathbf{X}}_{k_i}$ in the control packet. Specifically, if we denote by Q_P the length of $\mathbf{U}_k$

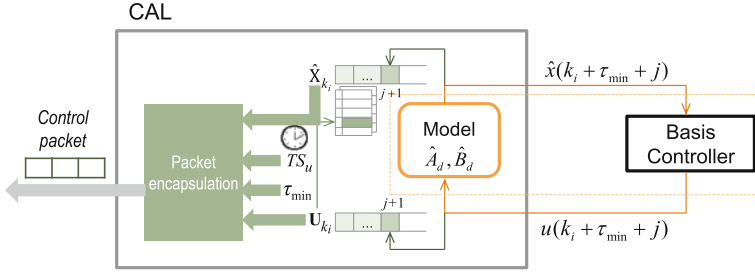


Fig. 2.6 Control and state sequences computation by the CAL layer in the event-based design

when measurements are sent to the controller periodically, and by Q_{EB} the length of $\mathbf{U}_{k_i}$ in the event-based design, it holds that

$$Q_{EB} = \frac{m}{n + m} Q_P,$$

where n and m are the state and the control input dimensions, respectively.

Example 2.3 Let us consider that the network protocol is UDP. The size of the payload of UDP packets is 508 bytes [Eva98] and a float value only consumes 4 bytes. Thus, if $m = 1$ we can compute the value of Q_P as

$$Q_P = \frac{508 \text{ bytes} - \mathcal{S}[TS_u] - \mathcal{S}[\tau_{min}]}{4 \text{ bytes}} = \frac{508 - 4 - 4}{4} = 125,$$

where $\mathcal{S}$ is the operator *size of*. We assume that all the elements of the control packets are float. Therefore, the number of future control sent in an control packet is 125.

However, an event-based design cuts down this value to $Q_{EB} = \frac{125}{n+1}$. For example if $n = 4$, then $Q_{EB} = 25$.

2.6.2 PAL Design for Event-Based Control

How the PAL works has been presented in Sect. 2.5. In an event-based scheme, packets are processed in a similar way, since the described mechanism is asynchronous. The main difference is that the predicted states sequence $\hat{\mathbf{X}}_{k_i}$ is also received. Due to the correspondence between the elements of $\hat{\mathbf{X}}_{k_i}$ and $\mathbf{U}_{k_i}$, the algorithm described in Sect. 2.5 is also applied to $\hat{\mathbf{X}}_{k_i}$. For instance, if the first i_0 elements of $\mathbf{U}_{k_i}$ are discarded because they are obsolete values, for the same reason the first i_0 elements of $\hat{\mathbf{X}}_{k_i}$ are also discarded.

However, the detection of events has to be included in the PAL design. The code of this new module of the PAL is given by Algorithm 2.1. The control and state predictions sequences are received as inputs as well as the computed index i_0 . The

error and the trigger function are initialized to default values (lines 2–3). The state of the plant is measured at each sampling time, and the error and trigger functions are computed (lines 7–9). The event condition is checked at each sampling time (line 10). In case an event is triggered, the module delivers $x(k_i)$ and the index value as outputs.

Note that an event is detected when either $f(e(k))$ crosses zero or $i_0 + j$ equals $\bar{Q}$, where $\bar{Q}$ is $\bar{Q} = Q - \tau_{max}$, and τ_{max} is the upper bound on the RTT whose value will be derived in the stability analysis. This constraint is imposed to prevent that the last element of the control sequence is reached without receiving a new control packet.

Input: $U_{k_i}, \hat{X}_{k_i}, i_0$ with $k_i < k$
Output: $x(k_{i+1}), i_0 + j$

```

1:  $j := 0$ 
2:  $e(k) := \mathbf{0}$ 
3:  $f(e(k)) := -1$ 
4: while  $i_0 + j < \bar{Q}$  and  $f(e(k)) < 0$  do
5:    $j := j + 1$ 
6:   Apply  $U_{k_i}(i_0 + j)$ 
7:   Measure  $x(k)$ 
8:    $\hat{x}(k) := \hat{X}_{k_i}(i_0 + j)$ 
9:    $e(k) := \hat{x}(k) - x(k)$ 
10:  Compute  $f(e(k))$ 
11: end while
```

Algorithm 2.1: PAL event-detection algorithm

Remark 2.2 We assumed that the computational power at the plant side is very limited. Note that the instructions of Algorithm 2.1 are very simple. The maximum complexity is in the computation of $f(e(k))$. We have preserved this notation for the sake of generality, but in practice we will consider trigger functions of the form (2.8). Thus, the computation is reduced to compare the error to a constant threshold.

2.7 Stability Analysis

The event-based policy (2.8) allows to reduce the communication in the control loop, but the price to pay is that asymptotic stability is no guaranteed, but the *Globally Ultimately Uniformly Boundedness* of the state can be proved.

Definition 2.3 The system (2.1) and (2.2) is Globally Ultimately Uniformly Bounded (GUUB) if for all $x(0) \in \mathbb{R}^n$ there exists a positive constant a and a time k^* such that $\|x(k)\| \leq a, \forall k \geq k^*$.

Let us first assume that disturbances are equal to zero and full state measurements are available. Then it follows that

$$x(k+1) = A_d x(k) + B_d K \hat{x}(k), \quad (2.9)$$

since the anticipative control defines the control law as the feedback of the predicted state for any sampling time k . Equation (2.9) can be rewritten in terms of the error (2.7) as

$$x(k+1) = (A_d + B_d K)x(k) + B_d K e(k). \quad (2.10)$$

Note that in the PAL layer an event is triggered whenever $\|e(k)\| \geq c$. However, the error will increase until a new control sequence is received. The next assumption establishes a bound on the maximum elapsed time between the detection of an event and the reception of a more recent control packet (RTT).

Assumption 2.1 The elapsed time between the event detection and, therefore, the transmission of a state packet to the controller, and the reception of a more recent control packet (RTT) is bounded by an upper bound denoted by τ_{max} . Moreover, this upper bound is always smaller than the minimum inter-event time:

$$\tau_{max} < k_{i+1} - k_i, \forall i \in \mathbb{N}.$$

Remark 2.3 In the elapsed time between the occurrence of an event and the reception of a new control sequence, packets can be dropped or experience delay. Hence, a flow control protocol (e.g. acknowledgments) to detect packet losses and transmission of a new measurement may be required in the event-based approach. For simplicity, let us denote the cited interval as $\tau(k_i)$ or simply as τ .

Assumption 2.1 constrains the model uncertainty and/or the maximum allowable number of sampling periods the system (2.2) can run in open loop (without receiving new control sequences from the remote controller). The derivation of these constraints will be given later in the section. First, the following result to bound the error at any time k is given as a consequence of Assumption 2.1.

Proposition 2.1 *If Assumption 2.1 holds, the error defined as (2.7) is bounded by*

$$\|e(k)\| \leq 2c. \quad (2.11)$$

Proof From Assumption 2.1 it follows that

$$\|e(k_i + \tau) - e(k_i)\| < c, \quad \forall \tau \leq \tau_{max},$$

since no event is detected in this interval.

According to the assumptions of Sect. 2.6, the error at the event detection is $\|e(k_i)\| \approx c$. Thus, assuming that this approximation is exact

$$\|e(k_i + \tau)\| \leq \|e(k_i)\| + \|e(k_i + \tau) - e(k_i)\| \leq 2c,$$

which concludes the proof. $\square$

Remark 2.4 Assumption 2.1 has allowed to establish a bound on the error $e(k)$, for all k . Alternatively, the upper bound on the RTT could be set to an arbitrary integer number of minimum inter-event times:

$$\tau_{max} < \nu(k_{i+1} - k_i), \quad \nu \in \mathbb{N}. \quad (2.12)$$

Thus, an equivalent result to Proposition 2.1 would be derived:

$$\|e(k)\| \leq (\nu + 1)c.$$

Note, however, that if the error was increased, the performance would degrade.

Let us denote $A_{dK} = A_d + B_d K$ to simplify the notation. Because A_{dK} is assumed to be Hurwitz, there exists a $P = P^T > 0$ such that

$$A_{dK}^T P A_{dK} - P = -Q, \quad (2.13)$$

where $Q = Q^T > 0$. And let us define the following Lyapunov function:

$$V(x) = x^T P x. \quad (2.14)$$

The main result of the section is presented next. The error $e(k)$ can be interpreted as an external perturbation due to the mismatch between the real dynamics of the process and the model, and the network imperfections.

Theorem 2.1 *If Assumption 2.1 holds, the state of the system (2.10) when the remote controller runs according to the model (2.5) and the event detector is defined by (2.8), is GUUB with bound*

$$\|x\| \leq \sqrt{\frac{\lambda_{max}(P)}{\lambda_{min}(P)}} (\sigma \|A_{dK}\| + \|B_d K\|) 2c, \quad (2.15)$$

where

$$\sigma = \frac{\|K^T B_d^T P A_{dK}\| + \sqrt{\|K^T B_d^T P A_{dK}\|^2 + \lambda_{min}(Q) \|K^T B_d^T P B_d K\|}}{\lambda_{min}(Q)}, \quad (2.16)$$

$\lambda_{min}(P)$ and $\lambda_{max}(P)$ are the minimum and maximum eigenvalues of P , respectively, and $\lambda_{min}(Q)$ the minimum eigenvalue of Q .

Proof The forward difference of the Lyapunov function (2.14) for (2.10) is

$$\begin{aligned}\Delta V(k) &= x^T(k+1)Px(k+1) - x^T(k)Px(k) \\ &= (A_{dK}x(k) + B_dKe(k))^T P (A_{dK}x(k) + B_dKe(k)) - x^T(k)Px(k) \\ &= -x^T(k)Qx(k) + 2e^T(k)(B_dK)^T PA_{dK}x(k) + e^T(k)(B_dK)^T PB_dKe(k),\end{aligned}$$

which is upper bounded by

$$\begin{aligned}\Delta V(k) &\leq -\lambda_{\min}(Q)\|x(k)\|^2 + 2\|(B_dK)^T PA_{dK}\|\|e(k)\|\|x(k)\| \\ &\quad + \|(B_dK)^T PB_dK\|\|e(k)\|^2.\end{aligned}\tag{2.17}$$

The right hand side of (2.17) is an algebraic second order equation in $\|x(k)\|$ such that the Lyapunov function decreases whenever

$$\|x(k)\| \geq \sigma\|e(k)\|,$$

where σ is given in (2.16).

According to Proposition 2.1, the error at any time k is bounded by $2c$. Hence, $\Delta V < 0$ in the region $\|x(k)\| > 2c\sigma$. Thus, the state decreases until it reaches this region. If we denote by k^* the time instant at which the state enters this region and according to (2.10), it follows that

$$\|x(k^* + 1)\| \leq (\sigma\|A_{dK}\| + \|B_dK\|)2c.$$

Then the state can leave the region so the Lyapunov function decreases again, and the space enclosed by the maximum of the Lyapunov function in $k^* + 1$ is an ultimate bound for the state. If the inequalities $\lambda_{\min}(P)\|x\|^2 \leq x^T Px \leq \lambda_{\max}(P)\|x\|^2$ are used, it is derived that the state $x(k)$ remains bounded by (2.15) $\forall k \geq k^*$, and this concludes the proof. $\square$

2.7.1 Analysis of the Maximum RTT and the Model Uncertainties

Assumption 2.1 has made possible to establish a bound on the error of the system and therefore the presented stability results. However, it also imposes some constraints on the maximum RTT for the network and/or the model uncertainty of the remote controller.

Assume that the last event occurred at time k_i . The error at the next sampling time is

$$\begin{aligned}
e(k_i + 1) &= \hat{x}(k_i + 1) - x(k_i + 1) = (\hat{A}_d + \hat{B}_d K) \hat{x}(k_i) - (A_d x(k_i) + B_d K \hat{x}(k_i)) \\
&= (\bar{A}_d + \bar{B}_d K) x(k_i) + (\hat{A}_d + \bar{B}_d K) e(k_i),
\end{aligned} \tag{2.18}$$

where $\bar{A}_d, \bar{B}_d K$ represent the model uncertainty.

In general, if a new control sequence is not received in τ sampling periods, the PAL layer continues applying control values from the same control sequence. Thus,

$$\begin{aligned}
x(k_i + \tau) &= A_d^\tau x(k_i) + \left(\sum_{j=1}^{\tau} A_d^{\tau-j} B_d K \hat{A}_{dK}^j \right) \hat{x}(k_i) \\
&= \left(A_d^\tau + \sum_{j=1}^{\tau} A_d^{\tau-j} B_d K \hat{A}_{dK}^j \right) x(k_i) + \left(\sum_{j=1}^{\tau} A_d^{\tau-j} B_d K \hat{A}_{dK}^j \right) e(k_i).
\end{aligned} \tag{2.19}$$

The error at $k + \tau$ is $e(k_i + \tau) = \hat{x}(k_i) - x(k_i)$, thus

$$e(k_i + \tau) = \hat{A}_{dK}^\tau \hat{x}(k_i) - x(k_i + \tau) = \hat{A}_{dK}^\tau e(k_i) + \hat{A}_{dK}^\tau x(k_i) - x(k_i + \tau),$$

where $x(k_i + \tau)$ is given in (2.19).

The maximum RTT can be derived imposing that

$$\|e(k_i + \tau_{max}) - e(k_i)\| < c,$$

which yields to a complicated expression which depends not only on the system and model dynamics but also on the state at the last event $x(k_i)$. It is not possible to derive an analytical solution for it, but the feasibility of the solution requires a bound for $x(k_i) \forall k_i$. Its existence is guaranteed from the results in Theorem 2.1.

However, it is possible to derive an analytical solution when the model uncertainty can be approximated to zero so that $\bar{A}_d \approx \mathbf{0}, \bar{B}_d \approx \mathbf{0}$. In this case, the evolution of $e(k)$ in (2.18) is approximated by $e(k_i + 1) \approx \hat{A}_d e(k_i)$. Thus, after τ sampling periods it is given by

$$e(k_i + \tau) \approx \hat{A}_d^\tau e(k_i) \approx A_d^\tau e(k_i). \tag{2.20}$$

Thus, according to Proposition 2.1, it holds that

$$\|e(k_i + \tau_{max}) - e(k_i)\| = \|(A_d^{\tau_{max}} - I)e(k_i)\| < c.$$

Since $\|e(k_i)\| \approx c$, an upper bound for the maximum allowable RTT will be the solution of

$$\|A_d^{\tau_{max}} - I\| < 1, \tau_{max} \in \mathbb{N}, \tag{2.21}$$

which is independent of the value of c .

Remark 2.5 According to Remark 2.4, if the condition imposed to τ_{max} was (2.12), it could be proven straightforward that (2.21) would turn into

$$\|A_d^{\tau_{max}} - I\| < \nu, \quad \nu \in \mathbb{N}.$$

Example 2.4 Assume that the scalar system

$$\dot{x}(t) = ax(t) + bu(t), \quad a, b \in \mathbb{R}, \quad (2.22)$$

is sampled with a sampling period T_s . An anticipative controller based on events is designed for this system, in which the event detector detects an event whenever the error crosses a threshold c . Assume that there is no model uncertainty in the anticipative controller. Let us compute the maximum allowable RTT for the system (2.22).

It holds that $A_d = e^{aT_s}$. Thus, according to (2.21), it holds that

$$|e^{aT_s\tau_{max}} - 1| < 1.$$

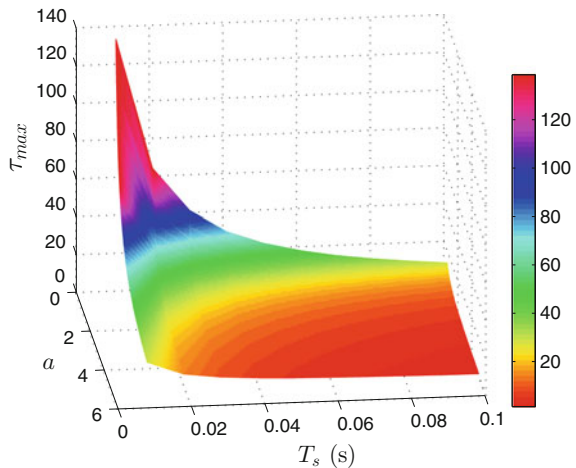
Since $a \in \mathbb{R}$, this is equivalent to $e^{aT_s\tau_{max}} < 2$. Thus,

$$\tau_{max} < \frac{\log(2)}{aT_s}. \quad (2.23)$$

Note that τ_{max} is feasible only if $a > 0$, because stable processes remain stable when there are not model uncertainties and no disturbances.

For example if $a = 1$ and $T_s = 50$ ms, $\frac{\log(2)}{aT_s} = 13.86$ and the maximum RTT is $\tau_{max} = 13$ sampling periods. In Fig. 2.7 the surface that bounds the region defined by (2.23) is depicted to illustrate the feasible range of τ_{max} as a function of a and T_s , where $a \in [0.1, 5]$ and $T_s \in [10, 100]$ ms.

Fig. 2.7 Surface defined by (2.23)



2.7.2 Analysis of the Error Bounds

The analysis has shown that the system is GUUB when Assumption 2.1 holds, and consequently, the error is upper bounded by $2c$ (see Proposition 2.1). However, one question that can be raised is what is the minimum value of the error that can be achieved with the prediction of the state at time $k + \tau_{min}$.

Under ideal network conditions, i.e., the network is reliable and the transmission delays between sensor-controller and controller-actuator are zero, the error $e(k) = \hat{x}(k) - x(k)$ is reset to zero after the occurrence of an event.

Also, if the delay τ can be measured because the architecture has a different configuration (e.g. Fig. 1.4a), the state of the plant at the time instance $k + \tau$ can be estimated, and the error is reset to zero if the model is perfect.

However, the fact that only statistics of the RTT can be known and the elements in the control loop are not synchronized, makes difficult to achieve this situation. In fact, if the RTT equals τ_{min} the error will reach its minimum value and its closure to zero will depend on the model uncertainty and the value of τ_{min} .

Thus, assume that the last event occurred at $k = k_i$. According to (2.19), the state at $k_i + \tau_{min}$ is

$$\begin{aligned} x(k_i + \tau_{min}) = & \left(A_d^{\tau_{min}} + \sum_{j=1}^{\tau_{min}} A_d^{\tau_{min}-j} B_d K \hat{A}_{dK}^j \right) x(k_i) \\ & + \left(\sum_{j=1}^{\tau_{min}} A_d^{\tau_{min}-j} B_d K \hat{A}_{dK}^j \right) e(k_i). \end{aligned}$$

While the prediction that the model gives is

$$\begin{aligned} \hat{x}(k_i + \tau_{min}) = & \left(\hat{A}_d^{\tau_{min}} + \sum_{j=1}^{\tau_{min}} \hat{A}_d^{\tau_{min}-j} \hat{B}_d K \left(\hat{A}_{dK} \right)^j \right) x(k_i) \\ & + \left(\sum_{j=1}^{\tau_{min}} \hat{A}_d^{\tau_{min}-j} \hat{B}_d K \hat{A}_{dK}^j \right) e(k_i). \end{aligned}$$

Then, it follows that the error is

$$\begin{aligned} e(k_i + \tau_{min}) = & \left(\hat{A}_d^{\tau_{min}} - A_d^{\tau_{min}} + \sum_{j=1}^{\tau_{min}} (\hat{A}_d^{\tau_{min}-j} \hat{B}_d - A_d^{\tau_{min}-j} B_d) K \left(\hat{A}_{dK} \right)^j \right) x(k_i) \\ & + \left(\sum_{j=1}^{\tau_{min}} (\hat{A}_d^{\tau_{min}-j} \hat{B}_d - A_d^{\tau_{min}-j} B_d) K \hat{A}_{dK}^j \right) e(k_i). \end{aligned} \quad (2.24)$$

Note that the right hand side of (2.24) is zero if $A_d = \hat{A}_d$ and $B_d = \hat{B}_d$, and different from zero otherwise. Moreover, it depends on the state $x(k_i)$.

Example 2.5 In order to illustrate the previous analysis, Fig. 2.8 shows the real and the estimated state of a certain plant, and the norm of the error in an interval of time, assuming that the model uncertainty is bounded by $\|\bar{A}_d\| \leq 0.1\|A_d\|$, $\|\bar{B}_d\| \leq 0.1\|B_d\|$ and $n = 2$.

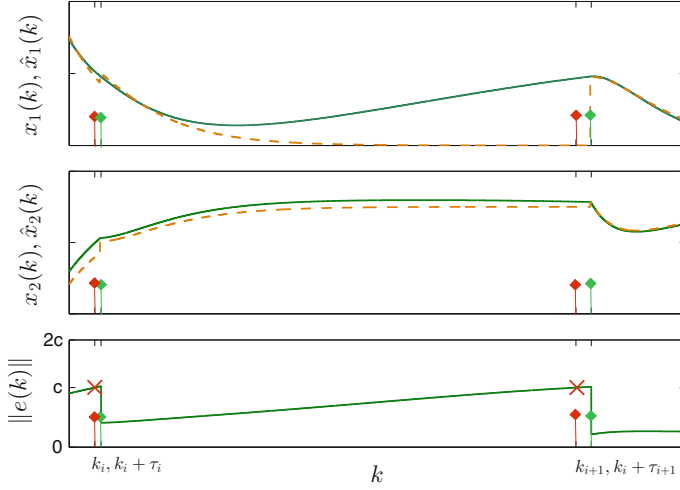


Fig. 2.8 Comparative of the state (solid line) and the model (dotted line), and the error bound. k denotes the sampling time, k_i, k_{i+1} the events occurrence, and τ_i, τ_{i+1} the delays

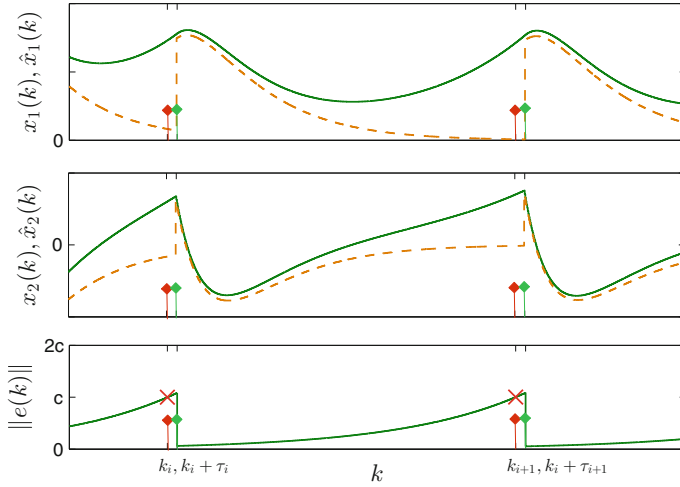


Fig. 2.9 Comparative of the state (solid line) and the model (dotted line), and the error bound. k denotes the sampling time, k_i, k_{i+1} the events occurrence, and τ_i, τ_{i+1} the delays

At time k_i an event is detected, but the next control sequence is not received at the plant time after τ_i sampling periods. Note that at $k_i + \tau_i$ the norm of the error decreases and then it increases until $\|e(k)\|$ reaches the bound c again. This time the RTT is $\tau_{i+1} > \tau_i$, as it can be observed from the figure. However, the error decreases to a value which is closer to zero than in the previous event k_i , showing the effect of $x(k_i)$ over $\|e(k)\|$ when there is a certain error in the model.

In contrast, when the model is perfect, the value that reaches the error after the reception of a new control sequence only depends on τ_i . This is illustrated on Fig. 2.9, in which $\|e(k_i + \tau_i)\| = \|e(k_{i+1} + \tau_{i+1})\|$ because $\tau_i = \tau_{i+1}$.

2.8 Disturbance Estimator

According to (2.1), the system is affected by disturbances $w(k) \in \mathbb{R}^n$. However, until now this fact has not been taken into account to predict future states of the plant according to (2.5). Disturbances can be estimated using the information given by the measurement error to improve the behavior of the anticipative control and reduce the number of events.

In [LL10], disturbances are estimated at event times assuming that they are constant between events in the proposed emulation approach, which mimics the continuous state feedback control. One constraint of the design is that the matrix A must be invertible, which excludes integrators from the dynamics of the system.

In this section we present a disturbance estimator for the remote anticipative controller which does not require A to be invertible and considers model mismatch. The following assumptions hold henceforth:

- The system dynamics is given by (2.1) and (2.2).
- The model of the CAL layer estimates future states of the plant according to

$$\hat{x}(k+1) = \hat{A}_d \hat{x}(k) + \hat{B}_d u(k) + \hat{w}(k), \quad (2.25)$$

where $\hat{w}(k)$ is the estimated disturbance at time k .

- The state $x(k)$ is measurable.
- When a state packet is received with a measurement taken at time k , the disturbance is estimated before computing the next control sequence U_k , and held constant in the next steps.

Hence, the disturbance estimator is a new element to include in the CAL layer.

According to (2.1) and (2.25), the error dynamics is given by

$$\begin{aligned} e(k+1) &= \hat{x}(k+1) - x(k+1) \\ &= \hat{A}_d \hat{x}(k) - A_d x(k) + (\hat{B}_d - B_d)u(k) + (\hat{w}(k) - w(k)), \end{aligned} \quad (2.26)$$

where $u(k)$ is given by (2.6). The disturbance $w(k)$ could be calculated if the rest of the terms of (2.26) were known. However, the model mismatch is unknown.

Therefore, if the approximation $\bar{A}_d \approx \mathbf{0}$, $\bar{B}_d \approx \mathbf{0}$ is taken, the value of $w(k)$ can be computed at the next sampling time $k + 1$ (after measuring $x(k + 1)$) as

$$\hat{w}(k + 1) = \hat{A}_d(\hat{x}(k) - x(k)) + \hat{w}(k) - e(k + 1) = \hat{w}(k) + \hat{A}_d e(k) - e(k + 1). \quad (2.27)$$

Let us denote q the number of sampling periods between the reception of the last control sequence and the detection of an event. In absence of disturbances, the error at $k + q$ can be approximated to $e(k + q) \approx \hat{A}_d^q e(k)$ (see (2.20)). This approximation turns into

$$e(k + q) = \hat{A}_d^q e(k) + \sum_{j=0}^{q-1} \hat{A}_d^j (\hat{w}(k + j) - w(k + j))$$

when disturbances are included in the model.

Because $\hat{w}(k)$ is assumed to be held constant in this interval, the disturbance can be estimated at time $k + q$ as

$$\hat{w}(k + q) = \hat{w}(k) + \left(\sum_{j=0}^{q-1} \hat{A}_d^j \right)^{-1} (\hat{A}_d^q e(k) - e(k + q)). \quad (2.28)$$

Example 2.6 Consider that the system is a double integrator:

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot u(t).$$

The system is sampled with every 50 ms. Thus, it follows that

$$\hat{A}_d = \begin{pmatrix} 1 & 0.05 \\ 0 & 1 \end{pmatrix}.$$

If $(\sum_{j=0}^{q-1} \hat{A}_d^j)^{-1}$ is computed for different values of q , for instance, $q = 5, 10$ and 50 , we get

$$q = 5 \rightarrow \begin{pmatrix} 0.2 & -0.02 \\ 0 & 0.2 \end{pmatrix}, \quad q = 10 \rightarrow \begin{pmatrix} 0.1 & -0.0225 \\ 0 & 0.1 \end{pmatrix}, \quad q = 50 \rightarrow \begin{pmatrix} 0.02 & -0.0245 \\ 0 & 0.02 \end{pmatrix}.$$

Note that A is not invertible, but $\sum_{j=0}^{q-1} \hat{A}_d^j$ is, allowing to estimate $w(k)$. It is also interesting to remark that the diagonal elements of the resulting matrix decreases with q . This makes sense with the transmission policy, since q takes large values when no event is detected, meaning that the estimation of the disturbance is good. The term that gives the correction in (2.28) is weighted by $(\sum_{j=0}^{q-1} \hat{A}_d^j)^{-1}$. Thus, the larger the q , the correction the smaller.

Note that $e(k)$ in (2.28), which denotes the error between the estimated state and the measured state at the time of the reception of the control sequence is in general

non-zero. This information as well as the error at the time of the detection of the event must be known. This implies that both values have to be transmitted from the PAL to the CAL. Thus, the *state packets* must include the following information:

- The measurement which triggered the event $x(k_i)$.
- A time stamp TS_u of the controller local time. TS_u allows to identify the control sequence $\mathbf{U}_{k_{i-1}}$, $k_{i-1} < k_i$, which was being applied at the time of the measurement of $x(k_i)$.
- The index i_u which is the number of element of the sequence $\mathbf{U}_{k_{i-1}}$ which was being applied at the time of the measurement of $x(k_i)$.
- The error between the predicted state by the model (2.25) and the measured state after receiving $U_{k_{i-1}}$. If the number of sampling periods between this instant of time and the detection of the event at time k_i is q_i , hence this value is $e(k_i - q_i)$.
- The error $e(k_i)$ when the event is detected.
- The number of sampling periods q_i .

According to this, the code executed by the CAL is illustrated in Algorithm 2.2. Note that once $\hat{w}(k_i)$ is estimated, it is used in the estimation of $\hat{x}(k_i + \tau_{min})$ and the computation of $\mathbf{U}_{k_i}$, $\hat{\mathbf{X}}_{k_i}$.

```

Input:  $x(k_i), TS_u, i_u, e(k_i - q_i), e(k_i), q_i$ 
Output:  $\mathbf{U}_{k_i}, \hat{\mathbf{X}}_{k_i}$ 
1:  $\hat{w}(k_i - q_i) := \text{getFromLookupTable}(TS_u)$ 
2:  $\hat{w}(k_i) := \hat{w}(k_i - q_i) + (\sum_{j=0}^{q_i-1} \hat{A}_d^j)^{-1} (\hat{A}_d^{q_i} e(k_i - q_i) - e(k_i))$ 
3:  $[u(k_i) \dots u(k_i + \tau_{min})] := \text{getFromLookupTable}(TS_u, i_u, \tau_{min})$ 
4:  $\hat{x}(k) := x(k_i)$ 
5: for  $j = 1 \rightarrow \tau_{min}$  do
6:    $\hat{x}(k+1) := \hat{A}_d \hat{x}(k) + \hat{B}_d u(k_i + j - 1) + \hat{w}(k_i)$ 
7:    $\hat{x}(k) := \hat{x}(k+1)$ 
8: end for
9:  $\hat{x}(k_i + \tau_{min}) := \hat{x}(k)$ 
10:  $\mathbf{U}_{k_i}(1) = K \hat{x}(k_i + \tau_{min})$ 
11:  $\hat{\mathbf{X}}_{k_i}(1) = (\hat{A}_d + \hat{B}_d K) \hat{x}(k_i + \tau_{min}) + \hat{w}(k_i)$ 
12: for  $j = 2 \rightarrow Q$  do
13:    $\mathbf{U}_{k_i}(j) = K \hat{\mathbf{X}}_{k_i}(j-1)$ 
14:    $\hat{\mathbf{X}}_{k_i}(j) = (\hat{A}_d + \hat{B}_d K) \hat{\mathbf{X}}_{k_i}(j-1) + \hat{w}(k_i)$ 
15: end for

```

Algorithm 2.2: Code executed in the CAL for disturbance estimation

Remark 2.6 Note that we have explicitly considered state feedback control for the sake of clarity, but this algorithm can be easily extended to other control laws.

Remark 2.7 Note that the error $e(k)$ is the result of the effect of disturbances $w(k)$, the model uncertainty $\hat{A}_d$, $\hat{B}_d$, and the network imperfections. The estimation $\hat{w}(k)$ partially compensates these effects, although their contribution to the error cannot be separated with the proposed approach.

2.8.1 Stability Analysis

Stability results can be derived when disturbances affect the system in a similar way as Theorem 2.1 if bounded disturbances are assumed:

$$\|w(k)\| \leq w_{\max}.$$

In this case, it is proven that the Lyapunov function (2.14) satisfying (2.13) decreases to reach a region whose size depends on the bound of the error $\|e(k)\|$ and the disturbances $\|w(k)\|$.

Before stating the main result of this section, let us rewrite (2.1) in terms of $e(k)$ as

$$x(k+1) = A_{dK}x(k) + B_d K e(k) + w(k). \quad (2.29)$$

Theorem 2.2 *If Assumption 2.1 holds and the disturbances are bounded by $\|w(k)\| \leq w_{\max}$, the state of the system (2.29) when the remote controller runs according to the model (2.25) and the event detector is defined by (2.8), is GUUB with bound*

$$\|x\| \leq \sqrt{\frac{\lambda_{\max}(P)}{\lambda_{\min}(P)}} (\|A_{dK}\| \delta_x^w + \|B_d K\| 2c + w_{\max}), \quad (2.30)$$

where

$$\delta_x^w = \frac{\delta_b + \sqrt{\delta_b^2 + 4\delta_a\delta_c}}{2\delta_a} \quad (2.31)$$

$$\delta_a = \lambda_{\min}(Q) \quad (2.32)$$

$$\delta_b = \|(B_d K)^T P A_{dK}\| 2c + \|P A_{dK}\| w_{\max} \quad (2.33)$$

$$\delta_c = \|(B_d K)^T P B_d K\| 4c^2 + 4\|P B_d K\| w_{\max} c + \lambda_{\max}(P) w_{\max}^2. \quad (2.34)$$

Proof The proof can be found in the Appendix 8 on page xxx. $\square$

Example 2.7 In this example a system modeled as a double integrator is considered, and sampled with $T_s = 5$ ms:

$$\hat{x}(k+1) = \begin{pmatrix} 1 & 0.005 \\ 0 & 1 \end{pmatrix} \hat{x}(k) + \begin{pmatrix} -0.0001 \\ -0.0380 \end{pmatrix} u(k).$$

The trigger function is defined with $c = 0.05$. The model uncertainty is known to be $\|\bar{A}_d\| < 0.2\|A_d\|$, $\|\bar{B}_d\| < 0.2\|B_d\|$. Disturbances affecting the system are bounded by 0.01, and change the value every second to a new random value in $[-0.01, 0.01]$.

Figure 2.10 shows the state of the system (solid line), the prediction given by the model (dashed line), the norm of the error, the control input and the real and estimated disturbances. Note that the major number of the events occur for small values of time

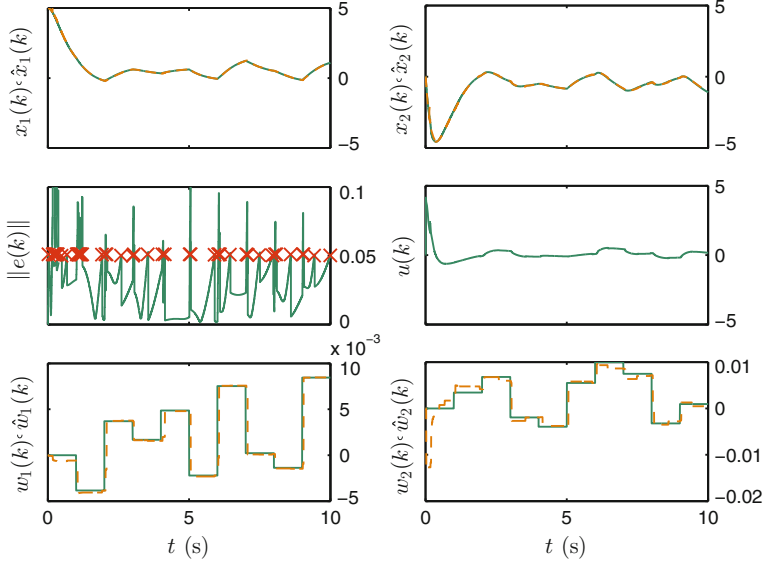


Fig. 2.10 Disturbances estimation. The estimated values are represented by the *dotted line*, and the actual values by the *solid line*

(when the state of the system is far from the equilibrium), and when the value of the disturbance changes. The difference between the real and the estimated states is not well appreciated due to the scale and the small value of c .

2.9 Output-Based Event-Triggered Control

This section presents a method to anticipative control when the state $x(k)$ cannot be measured and the only available information at each sampling time is the output $y(k)$. The extension of event-triggered control to output measurement is not trivial [HJT12]. One may think that an intuitive solution is to redefine the error as

$$e_y(k) = \hat{y}(k) - y(k), \quad (2.35)$$

define a trigger function such that $\|e_y(k_i)\| \approx c$, and extend the analysis to derive $\|e_y(k)\| \leq 2c$. However, the boundedness of $e_y(k)$ does not imply the boundedness of $\hat{x}(k) - x(k)$, which is required to proof the stability of the system when the basis controller is state feedback.

There are two directions to solve the problem in the literature. One direction is to process the measurements by an observer or a filter. For instance, in [LL11a] an state observer is proposed. The error function is replaced by $\hat{x}(k) - \tilde{x}(k)$, where $\tilde{x}(k)$ is

the observed state. The analysis shows that the property of GUUB is preserved. In [LL11c], a Kalman filter approach is presented.

The second direction is to use a different structure in the controller. A dynamical output-based controller is proposed in [DH10]. Using a mixed event-triggering mechanisms, the ultimate boundedness can be guaranteed while excluding the Zeno behavior. A level crossing sampling solution with quantization in the control signal is presented in [KB06], where a LTI continuous-time controller is emulated.

The first direction would make easier to extend the design of Sect. 2.6 and the stability results of Sect. 2.7. However, a computational cost is required in the PAL layer to observe the state, and it has been assumed that the computational capacity at the process side is very limited.

The design proposed in this thesis is a mixed solution of the two directions aforementioned. On the one hand, an observer is required to recover the state of the system in order to estimate future control values by the iteration of the plant model and the basis controller. However, since the observer needs to be implemented in the controller side, this does not allow to use the observation in the trigger functions. Thus, the error is defined as (2.35), and the trigger function for output measurement is

$$f(e_y(k)) = \|e_y(k)\| - c_y. \quad (2.36)$$

On the other hand, since only boundedness of the output error can be guaranteed, let us consider the following LTI discrete-time controller

$$x_C(k+1) = A_C x_C(k) + B_C \hat{y}(k) \quad (2.37)$$

$$u(k) = C_C x_C(k) + D_C \hat{y}(k), \quad (2.38)$$

for the basis controller. $x_C(k)$ is the state of the controller, and A_C , B_C , C_C and D_C are matrices of the appropriate dimensions. We assume that the controller is designed to render the system asymptotically stable when $\hat{y}(k)$ is replaced by $y(k)$. We further assume that the pair (A_d, C) is observable and that a model is available and it is given by $(\hat{A}_d, \hat{B}_d)$, and $\hat{C} = C$. Finally, disturbances affecting the system (2.2) are not considered for simplicity. However, the measurement noise $v(k)$ might not zero but bounded by v_{max} .

The description of how both Controller and Process Adaptation Layers can be adapted to this new scenario is given next.

2.9.1 PAL Design for Output Measurement

The tasks of the PAL can be divided into four modules (see Fig. 2.11):

- *Packet processing and encapsulation*: This module includes the packet processing (incoming packets) and packet encapsulation (outcoming packets) tasks, which are basically the same than for state measurement.

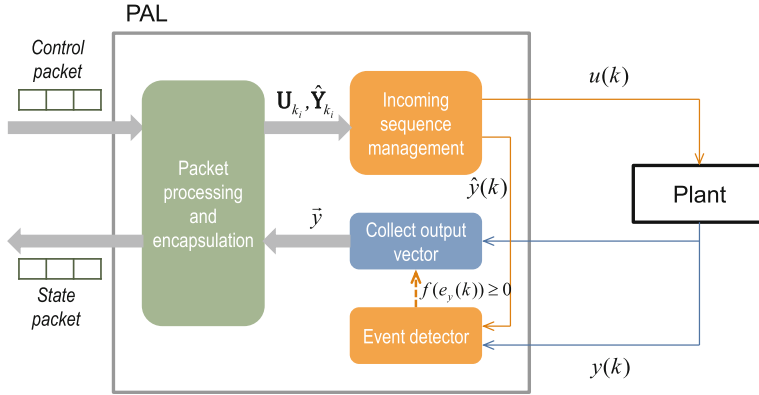


Fig. 2.11 PAL design for output measurement

- *Incoming sequence management*: This module is in charge of selecting the control input at each sampling time, as described in Sect. 2.5. Since event-triggering is supported, it also manages sequence of predictions given by the model. This sequence has been denoted as $\hat{\mathbf{X}}_{k_i}$ for state measurement. For output-based control, the controller sends $\hat{\mathbf{Y}}_{k_i}$ referring to a sequence of predicted outputs. The details of how $\hat{\mathbf{Y}}_{k_i}$ is computed can be found in the next section.
- *Event detector*: It monitors the system output at each sampling time. If the error (2.35) exceeds a certain threshold, i.e., the trigger function becomes positive, an event is generated. This is illustrated in Algorithm 2.3.
- The completely novel module in the PAL for output measurement is the one in charge of *collecting an output vector* denoted as $\vec{y}$. The measured outputs $y(k)$ at each sampling time k between the reception of a control packet and the detection of a new event are stored in $\vec{y}$ (see Algorithm 2.3). This information is used by the PAL to estimate the state of the plant via an state observer. Note that $\vec{y}$ can actually be a matrix if the system has multiple outputs. Since the inter-event time is limited by the fact that an event is triggered when the index of the control sequence reaches the value $\bar{Q}$, there is no need in imposing an additional constraint to the length of $\vec{y}$.

According to Fig. 2.11, the event detector informs when to stop collecting the output vector and then a new packet is encapsulated and sent to the controller.

2.9.2 CAL Design for Output Measurement

Three are the novelties in the CAL design respect to the ideas presented in Sects. 2.4 and 2.6.1:

Input: $\mathbf{U}_{k_i}, \hat{\mathbf{Y}}_{k_i}, i_0$ with $k_i < k$
Output: $\vec{y}, i_0 + j$

```

1:  $j := 0$ 
2:  $\vec{y} := []$ 
3:  $e_y(k) := 0$ 
4:  $f(e_y(k)) := -1$ 
5: while  $i_0 + j < \bar{Q}$  and  $f(e_y(k)) < 0$  do
6:    $j := j + 1$ 
7:   Apply  $\mathbf{U}_{k_i}(i_0 + j)$ 
8:   Measure  $y(k)$ 
9:    $\vec{y} := [\vec{y}, y(k)]$ 
10:   $\hat{y}(k) := \hat{\mathbf{Y}}_{k_i}(i_0 + j)$ 
11:   $e_y(k) := \hat{y}(k) - y(k)$ 
12:  Compute  $f(e_y(k))$ 
13: end while

```

Algorithm 2.3: PAL event-detection algorithm for output measurement

- The controller structure: The new basis controller is given by (2.37) and (2.38). Hence, it receives from the model the predicted output of the plant $\hat{y}(k)$, it computes its next internal state according to (2.37) and the control input as (2.38).
- The model needs to compute an estimation of the state of the plant $\hat{x}(k)$ according to (2.5), but only $\hat{y}(k) = C\hat{x}(k)$ is required by the controller.
- In Sect. 2.6.1, a predicted states sequence $\hat{\mathbf{X}}_{k_i}$ is generated and sent to the process. This information is not useful anymore since the state is not measurable, therefore, a predicted outputs sequence $\hat{\mathbf{Y}}_{k_i}$ is used instead. Since we have assumed that $\hat{C} = C$, it holds that $\hat{\mathbf{Y}}_{k_i} = C\hat{\mathbf{X}}_{k_i}$. Note that one advantage of this approach is that the length of $\mathbf{U}_{k_i}$ for an output-based scheme is, in general, larger than for full state measurement in event-based communications (see Remark 2.1).
- Since $x(k_i)$ is no longer available, it is estimated by a state observer using the information in $\vec{y}$, generating future states of the plant after that. We next describe this more in detail.

A Luenberger Observer of the State

A Luenberger observer of the form

$$\begin{aligned}\tilde{x}(k+1) &= (\hat{A}_d - LC)\tilde{x}(k) + \hat{B}_d u(k) + Ly(k), \quad \tilde{x}(0) = \tilde{x}_0 \\ \tilde{y}(k) &= C\tilde{x}(k)\end{aligned}$$

is used to obtain the state $x(k)$, being $(\hat{A}_d - LC)$ Hurwitz. We use the notation $\tilde{x}(k)$ rather than $\hat{x}(k)$ to differentiate it from the model predictions given by (2.5). Anytime a new state packet is received at the controller side, the code of Algorithm 2.4 is executed. The length of $\vec{y}$ is calculated first, that is, the number of sampling times between the reception of the last control sequence at the process side and the detection of the last event. Based on this information, and on $T S_u$ and i_u (received

Input: $\vec{y}, T S_u, i_u$
Output: $\tilde{x}(k_i)$

```

1:  $n := \dim(\vec{y})$ 
2:  $[u(k_i - n - 1) \dots u(k_i)] := \text{getFromLookupTable}(T S_u, i_u, n)$ 
3: for  $j = 1 \rightarrow n$  do
4:    $y(k_i - n - 1 + j) := \vec{y}(j)$ 
5:    $\tilde{x}(k_i - n + j) := (\hat{A}_d - LC)\tilde{x}(k_i - n - 1 + j) + \hat{B}_d u(k_i - n - 1 + j) + L y(k_i - n - 1 + j)$ 
6:    $\tilde{x}(k_i - n - 1 + j) := \tilde{x}(k_i - n + j)$ 
7: end for

```

Algorithm 2.4: Luenberger observer state estimation

with the state packet as well), we can determine the control inputs applied at the actuator during this period by checking them in a look-up table (see Sect. 2.4). Then, the Luenberger observer estimates the plant state $x(k_i)$ corresponding to the last output measurement $y(k_i)$, which is the last element of the output vector $\vec{y}$.

Thus, in an output-based scenario, the state $x(k_i)$ is replaced by $\tilde{x}(k_i)$ to estimate $\hat{x}(k_i + \tau_{min})$ first, and after that, to generate the control sequence $\mathbf{U}_{k_i}$.

2.9.3 Stability Analysis

To formulate the analysis, let us gather the equations that describe the dynamics of both the system and the controller

$$x(k+1) = A_d x(k) + B_d u(k) \quad (2.39)$$

$$y(k) = C x(k) + v(k) \quad (2.40)$$

$$x_c(k+1) = A_C x_c(k) + B_C \hat{y}(k) \quad (2.41)$$

$$u(k) = C_C x_c(k) + D_C \hat{y}(k), \quad (2.42)$$

with the error defined as (2.35) and the trigger function (2.36). This can be rewritten as

$$\begin{pmatrix} x(k+1) \\ x_c(k+1) \end{pmatrix} = \begin{pmatrix} A_d + B_d D_C C & B_d C_C \\ B_C C & A_C \end{pmatrix} \begin{pmatrix} x(k) \\ x_c(k) \end{pmatrix} + \begin{pmatrix} B_d D_C \\ B_C \end{pmatrix} (e_y(k) + v(k)).$$

Let us define the augmented state vector of the system by combining process and controller $\xi^T(k) = (x^T(k) \ x_c^T(k))$, and the matrices

$$A_{CL} = \begin{pmatrix} A_d + B_d D_C C & B_d C_C \\ B_C C & A_C \end{pmatrix}, \quad (2.43)$$

$$F = \begin{pmatrix} B_d D_C \\ B_C \end{pmatrix}. \quad (2.44)$$

Thus, the closed-loop system-controller dynamics is

$$\xi(k+1) = A_{CL}\xi(k) + F e_y(k) + F v(k). \quad (2.45)$$

Equation (2.45) compacts the dynamics of the system and the controller. It can be seen as a perturbed version of $\xi(k+1) = A_{CL}\xi(k)$. Hence, if we assume that the controller is designed so that A_{CL} (see (2.43)) is Hurwitz, there exist a $P = P^T$ such that

$$A_{CL}^T P A_{CL} - P = -Q, \quad Q = Q^T.$$

We define the Lyapunov function

$$V(\xi) = \xi^T(k) P \xi(k). \quad (2.46)$$

The unperturbed system $\xi(k+1) = A_{CL}\xi(k)$ converges asymptotically to the origin. Nevertheless, when event-triggering (2.36) is considered and measurements are affected by noise, only *GUUB* of $\xi(k)$ can be achieved.

Let us consider that Assumption 2.1 holds. The result of Proposition 2.1 can be extended to the error $e_y(k)$ straightforward, so that

$$\|e_y(k)\| \leq 2c_y, \quad \forall k.$$

The following theorem is equivalent to Theorem 2.1 but for output measurement and the proposed controller design. The error $e_y(k)$ and the measurement noise $v(k)$ perturb the system. The error $e_y(k)$ is a contribution of both the model uncertainties and the network imperfections, whereas $v(k)$ is inherited from the measurement itself.

Theorem 2.3 *If Assumption 2.1 holds, the augmented state $\xi(k)$ of the system-controller (2.45), when the event detector is defined by (2.36), and the measurement noise is bounded $\|v(k)\| \leq v_{\max}$, is GUUB with bound*

$$\|\xi\| \leq \sqrt{\frac{\lambda_{\max}(P)}{\lambda_{\min}(P)}} (\sigma_\xi \|A_{CL}\| + \|F\|) (2c_y + v_{\max}), \quad (2.47)$$

where

$$\sigma_\xi = \frac{\|F^T P A_{CL}\| + \sqrt{\|F^T P A_{CL}\|^2 + \lambda_{\min}(Q) \|F^T P F\|}}{\lambda_{\min}(Q)}. \quad (2.48)$$

Proof The proof can be found in the Appendix 8 on page xxx. $\square$

Remark 2.8 A similar analysis to Sects. 2.7.1 and 2.7.2 can be done for output measurement to derive the constraints on the delay that guarantees that Assumption 2.1 holds.

For output measurement, the state of the process at the event time is not available and it is estimated via the Luenberger observer (see Sect. 2.9.2). This causes an initial error to estimate future states of the plant. Specifically, the recursive equation for the observation error $\tilde{e}(k)$ is

$$\begin{aligned}\tilde{e}(k+1) &= \tilde{x}(k+1) - x(k+1) \\ &= \hat{A}_d \tilde{x}(k) + L(y(k) - \tilde{y}(k)) + \hat{B}_d u(k) \\ &\quad - A_d x(k) - B_d u(k) \\ &= \bar{A}_d x(k) + \bar{B}_d u(k) + (\hat{A}_d - LC)\tilde{e}(k) + Lv(k).\end{aligned}\quad (2.49)$$

Note that if there are not model uncertainties and no measurement noise, the observation error converges asymptotically to zero, and only boundedness can be proved otherwise.

The observation error (2.49) can be rewritten in terms of the augmented state $\xi(k)$ if $u(k)$ is replaced by (2.38). It yields

$$\begin{aligned}\tilde{e}(k+1) &= \bar{A}_d x(k) + \bar{B}_d (C_C x_C(k) + D_C \hat{y}(k)) + (\hat{A}_d - LC)\tilde{e}(k) + Lv(k) \\ &= (\bar{A}_d + \bar{B}_d D_C C \quad \bar{B}_d C_C) \xi(k) + (\hat{A}_d - LC)\tilde{e}(k) + (L + \bar{B}_d D_C)v(k).\end{aligned}$$

Thus, the error is bounded due to the results of Theorem 2.3, the boundedness of $v(k)$, and because $(\hat{A}_d - LC)$ is Hurwitz.

2.9.4 PI Anticipative Control

The proportional-integral-derivative (PID) controller has been and is currently applied to solve many control problems. Even though many controller choices are available nowadays, PID controllers are still by far the most widely used form of feedback control. In process industry it is known that more than 90 % of the control loops are regulated by PID controllers [rH06]. Most of such controllers are Proportional Integral (PI), since the derivative part is usually not used in practice [rH06].

For this reason, we particularize the previous results for output measurement and LTI controllers to the PI controller, and we include the set-point tracking. The tracking error $\epsilon(k)$ is defined as $\epsilon(k) = y_{sp} - y(k)$, where y_{sp} is this reference signal.

State Representation of a PI Controller

A conventional continuous-time PI controller can be written as

$$u(t) = K_p \left((by_{sp} - y(t)) + \frac{1}{T_i} \int_0^t (y_{sp} - y(s)) ds \right).$$

The state of the controller x_C can be defined as

$$\dot{x}_C(t) = \frac{K_p}{T_i}(y_{sp} - y(t)), \quad x_C(0) = 0. \quad (2.50)$$

So that the control signal $u(t)$ is then

$$u(t) = x_C(t) + K_p(b y_{sp} - y(t)). \quad (2.51)$$

A discrete time formulation for (2.50) and (2.51) can be derived using the Euler method. It yields

$$\begin{aligned} x_C(k+1) &= x_C(k) - \frac{K_p}{T_i} T_s (b y_{sp} - y(k)) \\ u(k) &= x_C(k) + (b y_{sp} - y(k)). \end{aligned}$$

It follows that $A_C = 1$, $B_C = -\frac{K_p T_s}{T_i}$, $C_C = 1$, and $D_C = -K_p$. This allows to derive (2.45) when the basis LTI controller is PI and for set-point tracking y_{sp} as

$$\xi(k+1) = \begin{pmatrix} A_d - K_p B_d C & B_d \\ -\frac{K_p T_s}{T_i} C & 1 \end{pmatrix} \xi(k) + \begin{pmatrix} -K_p B_d \\ -\frac{K_p T_s}{T_i} \end{pmatrix} (e_y(k) + v(k)) + \begin{pmatrix} K_p b B_d \\ \frac{K_p T_s}{T_i} \end{pmatrix} y_{sp}. \quad (2.52)$$

The output is

$$y(k) = (C \ 0) \xi(k) + v(k),$$

and the control input

$$u(k) = (-K_p C \ 1) \xi(k) + K_p (y_{sp} - e_y(k) - v(k)).$$

Control and Predicted States Sequences Computation

The control and the predicted output sequences have not been explicitly computed in this section. We derive them next for the PI controller to include the set-point tracking, but the results also hold for any A_{CL} and F of the form (2.43) and (2.44).

A model version of (2.52) can be defined to deduce the control and the predicted output sequences. Thus,

$$\hat{\xi}(k+1) = \hat{A}_{CL} \hat{\xi}(k) + \hat{F}_b y_{sp}, \quad (2.53)$$

where

$$\hat{A}_{CL} = \begin{pmatrix} \hat{A}_d - K_p \hat{B}_d C & \hat{B}_d \\ -\frac{K_p T_s}{T_i} C & 1 \end{pmatrix} \quad \hat{F}_b = \begin{pmatrix} K_p b \hat{B}_d \\ \frac{K_p T_s}{T_i} \end{pmatrix}. \quad (2.54)$$

Note that $\hat{x}_C = x_C$, but the compact form of (2.53) simplifies the expressions. After estimating $\hat{x}(k_i + \tau_{min})$ and therefore, $\hat{\xi}(k_i + \tau_{min})$, the j element of the predicted output sequence $\hat{\mathbf{Y}}_{k_i}$, i.e., $(C \ 0)\hat{\xi}(k_i + \tau_{min} + j)$, is

$$\hat{\mathbf{Y}}_{k_i}(j) = (C \ 0) \left[\hat{A}_{CL}^j \hat{\xi}(k_i + \tau_{min}) + \sum_{l=0}^{j-1} \hat{A}_{CL}^l \hat{F}_b y_{sp} \right], \quad (2.55)$$

assuming that the set-point value remains constant. And the $j + 1$ element of the control sequence $\mathbf{U}_{k_i}$, i.e., $(-K_P C \ 1)\hat{\xi}(k_i + \tau_{min} + j) + K_P b y_{sp}$, is

$$\mathbf{U}_{k_i}(j + 1) = (-K_P C \ 1) \left[\hat{A}_{CL}^j \hat{\xi}(k_i + \tau_{min}) + \sum_{l=0}^{j-1} \hat{A}_{CL}^l \hat{F}_b y_{sp} \right] + K_P b y_{sp}. \quad (2.56)$$

2.10 Centralized Anticipative Control for N Subsystems

The proposed scheme can be extended to the control of N subsystems from a centralized controller. From the process perspective, it works exactly the same as in a single control loop scenario. Indeed, each subsystem has its PAL layer, which does not require any new module in the design. However, the controller has to handle with the income and outcome of packets from/to different plants. Moreover, the processes can be far away from each other and the communication constraints can be different in each control loop.

Thus, new elements has to be added to the CAL design to handle with these new requirements. But before describing the proposed architectures, let us enumerate the following assumptions:

- The system dynamics is given by

$$x_i(k + 1) = A_{d,i} x_i(k) + B_{d,i} u_i(k) + w_i(k), \quad x_i(0) = x_{0,i} \quad (2.57)$$

$$y_i(k) = C_i x_i(k) + v_i(k), \quad i = 1, \dots, N, \quad (2.58)$$

where $x_i(k) \in \mathbb{R}^{n_i}$ is the state of the subsystem i , $u_i(k) \in \mathbb{R}^{m_i}$ is the control signal of the subsystem i , $w_i(k) \in \mathbb{R}^{n_i}$ is the disturbance and $v_i(k) \in \mathbb{R}^{r_i}$ is the measurement noise, both of which are bounded, and $A_{d,i}$, $B_{d,i}$, C_i are matrices of appropriate dimensions. The subsystems can have different dynamics and even more, different dimensions.

- There is a model for each plant given by

$$\hat{x}_i(k + 1) = \hat{A}_{d,i} \hat{x}_i(k) + \hat{B}_{d,i} u_i(k), \quad (2.59)$$

where $\hat{x}_i(k)$ is the estimated state. The model iterates with the basic controller to generate the corresponding sequences.

- We assume full state measurement and that the basis controller runs according to state feedback

$$u_i(k) = K_i \hat{x}_i(k), \quad (2.60)$$

although the framework for output measurement and LTI controllers, such as PI, can also be applied.

- The transmission of measurements of each subsystem is event triggered. The error is defined as

$$e_i(k) = \hat{x}_i(k) - x_i(k). \quad (2.61)$$

An event is detected and, therefore, a transmission from the sensor to the controller occurs, when the trigger function of the plant i crosses zero, that is, $f_i(e_i(k)) \geq 0$.

The proposed design is shown in Fig. 2.12. There are N plants distributed across the network and a centralized controller consisting of N basis controllers, one for each plant. A single CAL is in charge of processing the incoming packets, computing the control and predicted state sequences and encapsulating the control packets. In this case, there are N sources of state packets. The CAL differentiates them thanks to the packets heading.

Once the source is identified, and the packet is processed, the CAL switches over the models to choose the one of the corresponding plant. The procedure described for a single loop to compute the control and predicted states sequences also applies to the multi-loop case. Denote them as $\mathbf{U}_k^i$ and $\hat{\mathbf{X}}_k^i$, respectively.

Note that the measurement of the minimum RTT is taken for each loop. Denote this parameter as τ_{min}^i . Moreover, there is a look-up table of computed control sequences for each subsystem. Therefore, the centralized controller must have both computational and storing capacity. Both requirements increase with the number of subsystems.

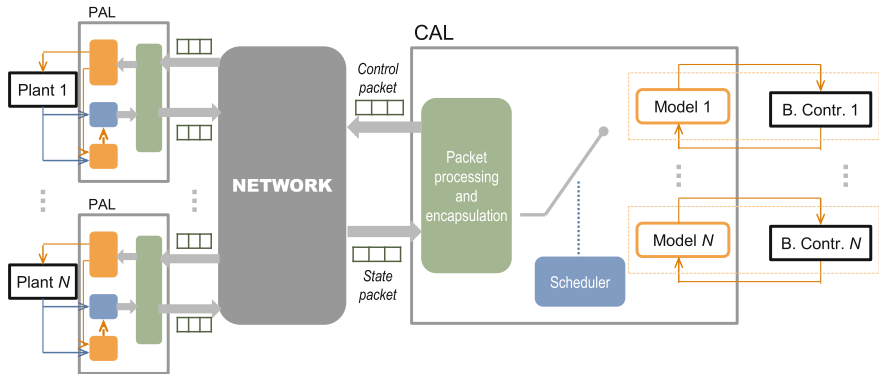


Fig. 2.12 CAL design of a centralized anticipative controller for N plants

Also the computational speed is an important issue since it directly influences the waiting delays on the packet queues. In general, the time a packet is waiting in a queue before being processed depends on this computational speed and also on the number of subsystems. This delay is added to the network of the control loop and can negatively affect the performance.

Example 2.8 Let us consider a set of N scalar systems

$$\dot{x}_i(t) = a_i x_i(t) + b_i u_i(t).$$

Each of them is sampled with a sampling period h_i . In the Example 2.4 an upper bound on the delay has been derived for the maximum RTT τ_{max} when a single system is considered so that Assumption 2.1 holds. This value is given by $\tau_{max} < \frac{\log 2}{ah}$.

When the number of controlled plants increase, the waiting delays on the packet queues in the controller also grows. If T_c^j is the computational time required to process a packet, compute the arguments of a new control packet, and encapsulate the new control packet for the process j , the worst case of the waiting time τ_W^i for another process i can be computed as

$$\tau_W^i = \sum_{\substack{j=1, \\ j \neq i}}^N T_c^j.$$

Thus, the maximum allowable delay for the channel plant i -controller turns to be

$$\tau_{max}^i < \frac{\log 2}{a_i h_i} - \tau_W^i.$$

A new element is included in the design of the CAL named as the *scheduler*. When there is more than one incoming packet, the scheduler decides which request is processed first. We describe how it works next.

2.10.1 The Scheduler

The purpose of the scheduler is to assign the priority of each packet when there is more than one packet in the queue of the incoming packets. The criteria considered in the algorithm are:

- *The dynamics of the plant:* Systems with fast or unstable dynamics are served first.
- *The quality of the communication link between the controller and the plant:* The slower connection, the higher priority.
- *The time of the last processed packet:* If a plant sent a packet because the actuator buffer was running out of data, the priority of the request increases.

Mathematically, the priority can be computed as

$$\pi^i(k) = \pi_0^i + \lambda \frac{\tau_{min}^i}{\frac{1}{N} \sum_{j=1}^N \tau_{min}^j} + \mu \frac{i_u^i}{Q^i}, \quad (2.62)$$

where i_u^i is the i_u index at the subsystem i (the number of element of the control sequence which was being applied at the time of the measurement of $x_i(k)$), and Q^i is the size of the control sequence. Note that Q^i differs from one system to another when the dimension of the states is different.

Hence, each plant has an initial priority π_0^i that is the priority of the plant in a centralized controller scenario but in absence of network. Then, the value of π_0^i is modified according to the second and third criteria by the second and third term, respectively. The parameters λ and μ in (2.62) are design parameters to adjust in order to give more or less relevance to each of the factors described above.

Example 2.9 Let us consider a system formed by four subsystems, each of one has an initial priority π_0^i and state dimension, which sets the value of Q^i , as given in Table 2.1. Let us set $\lambda = 1.5$ and $\mu = 1.2$.

The priority π^i assigned by the scheduler to each subsystem is shown in Fig. 2.13c. This π^i is computed taking into account the values of i_u^i (Fig. 2.13b) and τ_{min}^i (Fig. 2.13a), both of which change during the simulation period. The subsystem

Table 2.1 Parameters of the subsystems

No. subsystem	π_0^i	Q^i
1	1	25
2	1	25
3	2	15
4	3	15

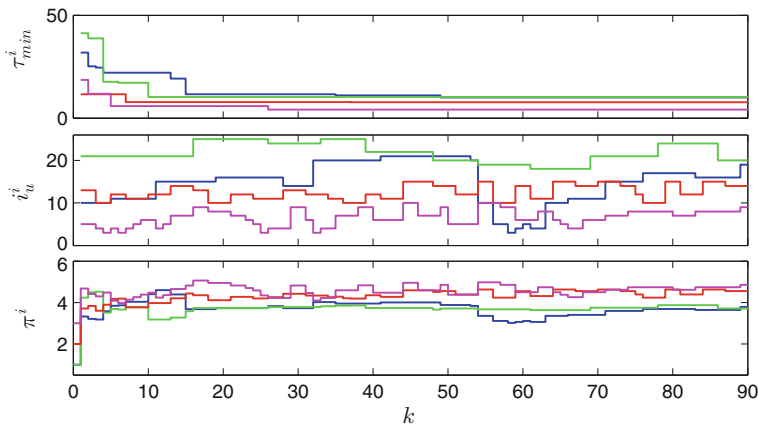


Fig. 2.13 Priority assigned by the scheduler to each of the subsystems

1 (blue line) and subsystem 2 (green line) has an initial priority of 1. The subsystem 3 (red line) has $\pi_0^3 = 2$ and, finally, $\pi_0^4 = 3$ (magenta line). Notice that, for instance, $\pi^3 > \pi^4$ in $k \in [64, 71]$ even though $\pi_0^3 < \pi_0^4$. Also, in $k \in [11, 13]$, $\pi^1 > \{\pi^2, \pi^3, \pi^4\}$ although it has the lowest initial priority. The reason is that $\tau_{min}^1 = \max\{\tau_{min}^i, i = 1, \dots, 4\}$ for this period of time.

2.11 Conclusions

An anticipative controller for packet-based NCS has been presented. The design of the middleware layers named as Controller Adaptation Layer (CAL) and Process Adaptation Layer (PAL) constitutes the main contribution to the NCS architecture.

A model of the plant predicts future states of the plant and, with this information, generates future control actions. The proposed design is improved with a disturbance estimator which allows reducing the differences between the measured and the predicted state.

The design has been extended to output measurements and LTI controllers. Also, a Luenberger observer is used in the CAL to estimate the state of the plant in the inter-event time so that future states of the plant can be predicted and, hence, future control actions can be derived.

The analysis has been particularized for PI controllers and, finally, a remote centralized controller has been presented for the N -control loops case, being the scheduler the main novelty respect to the single loop scheme.

The next chapter will present the experimental results to evaluate the proposed design.

References

- [rH06] Åström KJ, Hägglund T (2006) Advanced PID control. ISA
- [Bem98] Bemporad A (1998) Predictive control of teleoperated constrained systems with unbounded communication delays. In: Proceedings of the 37th conference of decision and control, pp 2133–2138, Tampa
- [CB08] Chaillet A, Bicchi A (2008) Delay compensation in packet switching networked controlled systems. In: Proceedings of the 47th IEEE conference on decision and control, pp 3620–3622, Cancun
- [DH10] Donkers MCF, Heemels WPMH (2010) Output-based event-triggered control with guaranteed L_∞ -gain and improved event-triggering. In: Proceedings of the IEEE conference on decision and control, pp 3246–3251, Atlanta
- [ESDCM07] Epstein M, Shi L, Di Cairano S, Murray RM (2007) Control over a network: using actuation buffers to reduce transmission frequency. In: Proceedings of the European control conference, Kos
- [Eva98] Evans WR (1998) UNIX networked programming, 2nd edn. Prentice-Hall, Upper Saddle River
- [GT04] Georgiev D, Tilbury DM (2004) Packet-based control. In: Proceedings of the American control conference, vol 1, pp 329–336

- [GBK09] Graham S, Balinga G, Kumar PR (2009) Abstractions, architecture, mechanisms and a middleware for networked control. *IEEE Trans Autom Control* 54(7):1490–1503
- [GSD11] Guinaldo M, Sánchez J, Dormido S (2011) A co-design strategy of NCS for treacherous network conditions. *IET Control Theory Appl* 5(16):1906–1915
- [GDJ+11] Guinaldo M, Dimaragonas DV, Johansson KH, Sánchez J, Dormido S (2011) Distributed event-based control for interconnected linear systems. In: *Proceedings of the 50th control and decision conference*, pp 2553–2558, Orlando
- [HD13] Heemels WPMH, Donkers MCF (2013) Model-based periodic event-triggered control for linear systems. *Automatica* 49(3):698–711
- [HJT12] Heemels WPMH, Johansson KH, Tabuada P (2012) An introduction to event-triggered and self-triggered control. In: *Proceedings of the 51st IEEE conference on decision and control*, pp 3270–3285, Maui
- [ICMS11] Irwin G, Chen J, McKernan A, Scanlon W (2011) Co-design of predictive controllers for wireless network control. *IET Control Theory Appl* 4(2):186–196
- [KJA06] Kim W-J, Ji K, Ambike A (2006) Real-time operating environment for networked control systems. *IEEE Trans Autom Sci Eng* 3(3):287–296
- [KB06] Kofman E, Braslavsky JH (2006) Level crossing sampling in feedback stabilization under data-rate constraints. In: *Proceedings of the 45th IEEE conference on decision and control*, San Diego
- [LL11a] Lehmann D, Lunze J (2011) Event-based output-feedback control. In: *Proceedings of the 19th Mediterranean conference on control and automatics*, pp 982–987, Corfu
- [LL11c] Li L, Lemmon MD (2011) Weakly coupled event triggered output feedback control in wireless networked control systems. In: *Proceedings of the Allerton conference on communication, control and computing*, University of Illinois - Urbana-Champaign
- [LL10] Lunze J, Lehmann D (2010) A state-feedback approach to event-based control. *Automatica* 46(1):211–215
- [GCB12] Luca Greco, Antoine Chaillet, Antonio Bicchi (2012) Exploiting packet size in uncertain nonlinear networked control systems. *Automatica* 48(11):2801–2811
- [MJVR08] Millán P, Jurado I, Vivas C, Rubio FR (2008) Networked predictive control of systems with data dropouts. In: *Proceedings of the 47th IEEE conference on decision and control*, pp 2704–2709
- [MA04] Montestruque LA, Antsaklis P (2004) Stability of model-based networked control systems with time-varying transmission times. *IEEE Trans Autom Control* 49(9):1562–1572
- [NH06] Naghshtabrizi P, Hespanha JP (2006) Anticipative and non-anticipative controller design for network control systems. *Netw Embed Sens Control* 331:203–218
- [QSG07] Quevedo D, Silva EI, Goodwin G (2007) Packetized predictive control over erasure channels. In: *Proceedings of the American control conference*, pp 1003–1008, Portland
- [QSG08] Quevedo DE, Silva EI, Goodwin GC (2008) Control over unreliable networks affected by packet erasures and variable transmission delays. *IEEE J Sel Areas Commun* 26(4):672–685
- [VF09] Varutti P, Findeisen R (2009) Compensating network delays and information loss by predictive control methods. In: *Proceedings of the European control conference*, Budapest
- [ZLR09] Zhao Y, Liu G, Rees D (2009) Design of a packed-based control framework for networked control systems. *IEEE Trans Control Syst Technol* 17(5):859–865

Contributions to Networked and Event-Triggered
Control of Linear Systems

Guinaldo Losada, M.

2016, XXV, 212 p. 89 illus., 36 illus. in color., Hardcover

ISBN: 978-3-319-34080-7