María Guinaldo Losada
Francisco Rodríguez Rubio
Sebastián Dormido Bencomo *Editors*

Asynchronous Control for Networked Systems

Springer

Asynchronous Control for Networked Systems

María Guinaldo Losada
Francisco Rodríguez Rubio
Sebastián Dormido Bencomo
Editors

Asynchronous Control for Networked Systems

Springer

Editors
María Guinaldo Losada
Department of Computer Science
and Automatic Control
National Distance Education University
(UNED)
Madrid
Spain

Sebastián Dormido Bencomo
Department of Computer Science
and Automatic Control
National Distance Education University
(UNED)
Madrid
Spain

Francisco Rodríguez Rubio
Department of Systems and Automatic
Engineering
University of Seville
Seville
Spain

ISBN 978-3-319-21298-2 ISBN 978-3-319-21299-9 (eBook)
DOI 10.1007/978-3-319-21299-9

Library of Congress Control Number: 2015945570

Mathematics Subject Classification (2010): 93

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

The term ‘networked control system’ (NCS) encompasses a relatively large number of situations and problems. The feature that distinguishes a NCS from a classical control system is the presence of a communication network affecting or inside the loop. New challenges arise as a consequence. In this sense, asynchronous control and, particularly, event-based control, have received an important impulse in the last decade due to its benefits when applied to NCSs, specially on energy-aware devices. Instead of taking periodic actions as in classical control approaches, asynchronous control bases its decisions on the state of the system and, in general, reduces the amount of communication.

The book presents novel results on asynchronous control of NCSs in a concise and clear style that are supported by simulation or experimental examples and it also provides examples of application. The manuscript is written with material collected from articles written by the authors, technical reports, and lectures given to graduate students, in which the ideas have been originally presented together with the formal proofs. Emphasis is laid on the presentation of the main results and the illustration of these results by examples.

The book is mainly aimed at graduate students, Ph.D. students, and researchers in control and communication, as well as practitioners, both from the control engineering community, although it can be followed by a wide range of readers, as only basic knowledge of control theory and sampled data systems is required.

The first chapter gives an introduction to asynchronous control and NCSs, including the main research trends and introducing concepts that have been used through the book. Then, the volume has been structured in two parts. The first part accounts for centralized control schemes, whereas the second block is focused on distributed estimation and control. A summary of each chapter is given next.

Part I Asynchronous Control for Single-Loop Schemes. Centralized Solutions

Chapter 2 focuses on the study of the limit cycles that appear in a control scheme based on a PI controller with an event-based send-on-delta sampling. The processes investigated are integrator processes plus time delay and first- and second-order processes plus time delay, which are of interest because of their frequent use as models in many industrial processes. An algorithm to calculate the limit cycles properties is presented, and then the results obtained in simulations are compared with experiments performed on real plants, such as a distributed solar collector field at the Solar Platform of Almería (PSA, Spain).

Chapter 3 considers a scenario in which the sensor and controller are connected by a bidirectional network. Whenever a fresh measurement is received from the plant, the following sampling instant is decided on the controller following a self-trigger strategy. To do so, the controller includes a model of the plant to generate predictions of the evolution of the states. In order to compute the sampling times, a set of quadratic optimization problems must be solved online.

Chapter 4 presents the analysis and the design of remote controllers for packet-based NCS, following the paradigm of anticipative controllers. 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. Event-based transmission rules are proposed to save network bandwidth. Different extensions such as disturbance estimators, output measurement, and LTI anticipative controllers are discussed. Finally, the design is evaluated over experimental plants characterized by response-times closed to the network delays.

Chapter 5 is concerned with the design of mixed H_2/H_∞ controllers for networked control systems through the Lyapunov–Krasovskii approach. The main contribution of the method does not lie in the use of novel Lyapunov–Krasovskii functionals or bounding techniques, but in the optimization method that can be used for different functionals and a variety of different constraints on the delay. Furthermore, the chapter investigates an asynchronous sampling approach based on events that allow a reduction of the bandwidth usage and the energy consumption. The relation between the boundedness of the stability region and the threshold that triggers the events is studied. The robustness and performance of the proposed technique is showed by numerical simulations.

Chapter 6 presents a practical algorithm to design networked control systems able to cope with high data dropout rates. The algorithm is intended for application in packet-based networks protocols (Ethernet-like) where data packets typically content large data fields. The key concept is using such packets to transmit not only the current control signal, but predictions on a finite horizon without significantly increasing traffic load. Thus, predictive control is used together with buffered actuators and a state estimator to compensate for eventual packet dropouts. Additionally, some ideas are proposed to decrease traffic load, limiting packet size

and media access frequency. Simulation results on the control of a three-tank system are given to illustrate the effectiveness of the method.

Part II Asynchronous Control and Estimation for Large-Scale Plants. Distributed Solutions

Chapter 7 discusses different control strategies of distributed event-based control for linear interconnected systems. From the analytical point of view, two aspects are considered to compare the different existing approaches: Convergence to the equilibria and inter-event times. Later on the chapter, two extensions are presented. The first extension is based on the fact that the frequency of actuation may be high in distributed control schemes if the neighborhood of the subsystem is large, even if each agent is not transmitting so often. To deal with this problem, an error function is defined for the control input and a second set of trigger functions is proposed to deal with this problem, updating the control law when a condition is violated. The second improvement relies on the existence of smart actuators, so that continuous-time signals can be applied instead of constant piecewise signals (ZOH). A model-based control design is proposed in which each agent has knowledge of the dynamics of its neighborhood.

Chapter 8 presents a generalized framework for distributed estimation in sensor networks. A distributed event-based estimation technique based on the stabilizing properties of the predesigned observers is proposed and analyzed, showing the reduction of both energy expenditure and network traffic load due to unnecessary transmissions. The observers's structure is based on both, local Luenberger observers and consensus strategies, which take into account the information that is received from neighboring nodes. Using the same structure, actuation capabilities in the nodes are included, yielding to a control scheme based on state estimation.

Chapter 9 contributed the field of distributed estimation and control with a novel method that allows to design both the controllers and the observers at one common step. The objective is to synthesize stabilizing suboptimal controllers, in the sense that the upper bound of a given cost function is minimized. The reduction of the bandwidth usage is attained exploiting an event-based communication policy between agents. The results have been applied to an experimental plant consisting of a four coupled tank system. The efficiency of the proposed method, in terms of reduction of the traffic and tuning capabilities, is shown.

Chapter 10 extends the results of Chap. 7 for non-reliable networks. Even though event-based control has been shown adequate to reduce the communication to face the problem of reduced bandwidth, network delays and packet losses cannot be avoided. Hence, the consequences of a non-reliable channel are analyzed, and upper bounds on the delay and the number of consecutive packet losses are derived. The design of network protocols is also presented, and simulation examples are given to illustrate the theory.

Chapter 11 is an extension of Chap. 8, and focuses on the following network related issues: delays, packet dropouts and communication policy (time and event-driven). The design problem is solved via linear matrix inequalities and stability proofs are provided. The technique is of application for sensor networks and large-scale systems where centralized estimation schemes are not advisable and energy-aware implementations are of interest. Simulation examples are provided to show the performance of the proposed methodologies.

Chapter 12 deals with the formation control of networked mobile robots as an example of multi-agent systems in which the group of robots achieves a common objective (the formation) by means of distributed control laws and event-based communications. An interactive simulator to emulate this kind of setups has been developed. The distributed event-based control algorithms have also been implemented in a testbed of mobile robots, and the results are presented. A study of the energy consumption and the performance is given.

Madrid — María Guinaldo Losada
Seville — Francisco Rodríguez Rubio
Madrid — Sebastián Dormido Bencomo
May 2015

Acknowledgments

The authors would like to thank a number of people and institutions who have made this book possible. We would like to thank several individuals whose work, either independently or through collaboration, has shaped the contents of the book. Our particular mention is for Pablo Millán, Luis Orihuela, Carlos Vivas, and José Sánchez, who took an active role in the revision of the whole manuscript.

Most of the material included in the book is the result of research work funded by the Spanish Ministry of Economy and Competitiveness under grants DPI2007-61068, DPI2011-27818-C02-02, DPI2012-31303, and DPI2013-44135-R, the European Commission under grant *Feedback Design for Wireless Networked Systems. (FeedNetBack)* (FP7-ICT-2007-2, Contract number: INFSO-ICT-223866), the *Consejera de Innovación, Ciencia y Empresa de la Junta de Andalucía* under grant P09-AGR-4782, and the IV Regional Plan of Scientific Research and Technological Innovation (PRICIT) of the Autonomous Region of Madrid under Project S-0505/DPI/0391. We gratefully acknowledge these institutions for their support. Also our thank to the people of the Automation Engineering, Control, and Robotics (TEP-201) research group of the Universidad de Sevilla.

Finally, the authors thank their families for their support, patience, and understanding of family time lost during the writing of the book.

Madrid
Seville
May 2015

María Guinaldo Losada
Francisco Rodríguez Rubio
Sebastián Dormido Bencomo

Contents

Notation

The symbols are chosen according to the following conventions. Matrices are represented by capital letters, and vector and scalars by lower-case letters. The elements of a vector x or a matrix A are $x_1, \ldots, x_n$ and $a_{11}, a_{12}, \ldots, a_{nm}$, respectively. For a block matrix E, E_{ij} denotes the (i,j) block. Calligraphic letters are generally reserved to sets, like $\mathscr{V}$ or $\mathscr{G}$.

Matrix $A = \text{diag}(a_1, \ldots, a_n)$ is a diagonal matrix with diagonal entries $a_1, \ldots, a_n$. Eigenvalues of a square matrix A are denoted by λ, where $\lambda_{max}(A)$ and $\lambda_{min}(A)$ are the maximum and minimum eigenvalues, respectively.

The absolute value is denoted by $|a|$. The notations $||x||$ and $||A||$ represent the vector or matrix norm. By default, $||\cdot||$ is the 2-norm computed as

$$||x||_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$$

$$||A||_2 = \sup_{x \neq 0} \frac{||Ax||_2}{||x||_2} = \sqrt{\lambda_{max}(A^* A)},$$

where A^* is the conjugate transpose of A. We also use the supremum and 1-norms:

$$||x||_\infty = \max_{i=1,\ldots,n} |x_i| \qquad ||A||_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|$$

$$||x||_1 = \sum_{i=1}^{n} |x_i| \qquad ||A||_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|.$$

Finally, a positive definite matrix is denoted by $P > 0$ or $P \succ 0$.

The rest of the symbols can be found in the subsequence.

Indices

$(\cdot)^*$	Conjugate transpose of a matrix
$(\cdot)^{-1}$	Inverse of a matrix
$(\cdot)^T$	Transpose of a vector or matrix
$(\cdot)_0$	Initial value
$(\cdot)_b$	Broadcasted signal
$(\cdot)_c$	Referring to the controller (Signal or matrix)
$(\cdot)_m$	Referring to a model
$(\cdot)_i / (\cdot)^i$	Referring to a subsystem i
$(\cdot)_{ij} \Big/ (\cdot)^{ij}$	Referring to transmission from a subsystem i to j

Scalars

n	Dimension of the state vector
m	Dimension of the input vector
r	Dimension of the output vector
k	Discrete instant
ℓ	Counter
t	Continuous time
τ	Delay
T_s	Sampling time
T_W	Waiting time
δ	Event constant threshold
t_k	Event time
ℓ_k	Event time (discrete time)
τ_{max}	Delay bound
τ_{sc}	Delay from sensor to controller
τ_{ca}	Delay from controller to actuator
n_p	Maximum number of consecutive packet losses
p	Packet losses rate
$\kappa(A)$	Condition number of A ($\kappa(A) = \|A\|\|A^{-1}\|$)
N_a	Number of agents
N_u	Length of control sequence

Vectors

x	State vector
u	Input vector

w	Disturbance vector
y	Output vector
v	Noise vector
$\hat{x}$	Observed state
e	Observer error
ε	Event-based control error
y_{sp}	Set-point
ξ	Augmented state vector
$\mathbf{r}$	Desired inter-vehicle relative position vector
$\bar{1}$	Column vector whose components are ones
$\mathscr{U}$	Augmented control vector

Matrices

A	System matrix
B	Input matrix
C	Output matrix
D	Feedforward matrix
K	Feedback gain
A_K	System matrix of a closed-loop system ($A_K = A + BK$)
M	Observer gain
N_{ij}	Consensus gains
U_k	Control sequence
$\mathbf{0}$	Null matrix of appropriate size
I_n	$n \times n$ identity matrix

Other

$\mathscr{G}$	Graph
$\mathscr{V}$	Set of vertices and set of edges
$\mathscr{E}$	Edges
$\mathscr{N}_i$	Neighborhood of the i-th node
$\mathscr{O}$	Order of complexity
V	Lyapunov function

Contributors

Jesús Chacón Dpto. de Informática y Automática, Escuela Técnica Superior de Informática, UNED, Madrid, Spain

David Muñoz de la Peña Dpto. de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingenieros, Universidad de Sevilla, Seville, Spain

Dimos V. Dimarogonas School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

Sebastián Dormido Dpto. de Informática y Automática, Escuela Técnica Superior de Informática, UNED, Madrid, Spain

Gonzalo Farias Escuela de Ingeniería Eléctrica, Pontificia Universidad Católica de Valparaiso, Valparaiso, Chile

Carlo Fischione Access Linnaeus Center, Department of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

María Guinaldo Dpto. de Informática y Automática, Escuela Técnica Superior de Informática, UNED, Madrid, Spain

Karl H. Johansson School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

Isabel Jurado Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería, Universidad Loyola Andalucía, Seville, Spain

Daniel Lehmann School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

Pablo Millán Departamento de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería, Universidad Loyola Andalucía, Seville, Spain

Luis Orihuela Departamento de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería, Universidad Loyola Andalucía, Seville, Spain

Francisco R. Rubio Dpto. de Ingeniería de Sistemas y Automática, Escuela Superior de Ingenieros, Universidad de Sevilla, Seville, Spain

José Sánchez Dpto. de Informática y Automática, Escuela Técnica Superior de Informática, UNED, Madrid, Spain

Antonio Visioli Dipartimento di Ingegneria Meccanica e Industriale, Facoltá di Ingegneria, Universitá degli Studi di Brescia, Brescia, Italy

Carlos Vivas Dpto. de Ingeniería de Sistemas y Automática, Escuela Superior de Ingenieros, Universidad de Sevilla, Seville, Spain

Chapter 1
Introduction

María Guinaldo, Francisco R. Rubio, Sebastián Dormido, Pablo Millán, Carlos Vivas and Luis Orihuela

1.1 Historical Perspective: From Digital Control to Networked Control Systems

The idea of using digital computers for control purposes started to emerge in the 1950s. In those times, however, computers were slow and unreliable, very limited in memory and computation capabilities, and were generally restricted for use as data loggers or performing computations for managing information. As reliability improved, computers were gradually integrated, first in supervisory control operations, then as controllers themselves. In 1962, a radical breakthrough was introduced by Imperial Chemical Industries (ICI) Ltd. in the UK, installing a Ferranti Argus Computer at Burnaze Works to measure 224 variables and manipulate 129 valves

M. Guinaldo (✉) · S. Dormido
Dpto. de Informática y Automática, Escuela Técnica
Superior de Informática, UNED, Madrid, Spain
e-mail: mguinaldo@dia.uned.es

S. Dormido
e-mail: sdormido@dia.uned.es

F.R. Rubio · C. Vivas
Dpto. de Ingeniería de Sistemas y Automática,
Escuela Superior de Ingenieros, Universidad de Sevilla, Seville, Spain
e-mail: rubio@us.es

C. Vivas
e-mail: vivas@us.es

P. Millán · L. Orihuela
Dpto. de Matemáticas e Ingeniería, Escuela Técnica
Superior de Ingeniería, Universidad Loyola Andalucía, Seville, Spain
e-mail: pmillan@uloyola.es

L. Orihuela
e-mail: dorihuela@uloyola.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_1

directly. This is considered as the first time a computer was directly interfaced to and controlled a particular system, and the beginning of the era of direct digital control (DDC).

The growth of DDC was explosive since then, helped by lower costs, increasing performance, and reliability of digital technology. While the first implementations of DDC were restricted to dedicated links between controller and actuators/sensors, user needs and technological advances in communications paved way for the introduction of digital multiplexing in serial communication in the early 1970s and the first decentralized computer control systems (DCCS) in the middle and late 1970s. At this period of time, research interests shifted somehow to the new paradigm, as it is evident from the fact that IEEE and IEE conferences on distributed processing and distributed computer control systems were started.

Decentralized control systems were soon thereafter applied in integrated manufacturing and industrial applications in general. The first works treating the use of decentralized control in machinery also appeared at this time, see [63]. Excellent work dealing with some of the fundamentals of decentralized control systems was produced in the early days of decentralized processing. For example, elements for a global clock as a fundamental base for decentralized applications was put forward by [136], together with the use of datagrams for real-time applications instead of conventional positive acknowledgment and retransmission protocols. In an early work on decentralized processing by [118], the partitioning and allocation phases were also discussed. In [67, 117], the levels and degrees of decentralization were clarified. These ideas gave rise to a whole new branch of control theory whose most prominent applications came in the form of the field bus technology (e.g., FIP and PROFIBUS) and automotive buses (e.g., CAN), successfully employed for decades in the process and automation industry.

The astonishing growth of communication technologies over the past decades—reflected by available protocols, coding, and modulation algorithms and the switching/routing technologies for packet-based networks—rapidly attracted the interest of the control community. The use of a multipurpose shared network to connect decentralized control elements promised improvements in terms of more flexible architectures, reduced installation and maintenance costs, and higher reliability than traditional bus-based communication technologies. The problems associated to such a change of paradigm also proved to be challenging [188].

Networked control systems (NCSs) are decentralized systems in which the communication of the different elements of the control loop (sensors, actuators, and controllers) employs a shared digital communication network. NCSs is thus an interdisciplinary field, lying at the intersection of control and communication theories. Favored by the large number of applications and difficulties involved, in the last few years NCS has become a common issue for many control research groups all around the world (see the expert panel report on *Future Directions in Control, Dynamics, and Systems*[1]). Indeed, at least two of the technical areas of the International Federation of Automatic Control (IFAC) are devoted to this field, a new IEEE Transactions on

[1] http://www.cds.caltech.edu/murray/cdspanel.

the topic (TCNS) was launched in 2014, and there also exists an increasing number of specialized conferences and workshops, such as the IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys) or the SICE International Symposium on Control Systems.

1.2 Overview of Networked Control Systems and Asynchronous Systems

1.2.1 Emergence and Advantages of Networked Control Systems

Typically, a control system is composed of the following elements: system or plant to be controlled; sensors measuring plant outputs, and transmitting them; automatic controllers receiving plant outputs and making decisions on the control signals to be applied to the plant; and actuators receiving the inputs sent by the controller and applying these inputs to the plant. Point-to-point communication links between the different devices make it possible to implicitly consider the *perfect communication channel approach*: absence of transmission delays, information integrity and unlimited bandwidth (Fig. 1.1).

Needless to mention, the feature that distinguishes an NCS from a classical control system is the presence of a communication network affecting inside the loop (Fig. 1.2). The perfect communication channel assumption does not hold when a network mediates the connection among the different elements, at least generally speaking. Even when dedicated, standard communication networks are usually designed to preserve data integrity and do not suit the stringent real-time requirements of closed-loop control. These problems become particularly apparent when wireless or non-dedicated networks are used. A large number of systems may be using the communication channel concurrently sharing the available bandwidth.

Hence, the following questions arise: Why is it better to use this type of technology for control purposes? In which situations are these solutions more suitable?

On the one hand, there are a number of generic advantages when using digital communication networks. Namely,

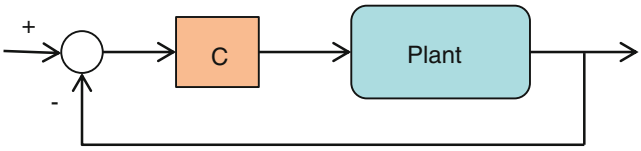

Fig. 1.1 Classic control scheme with the assumption of *perfect communication channel*

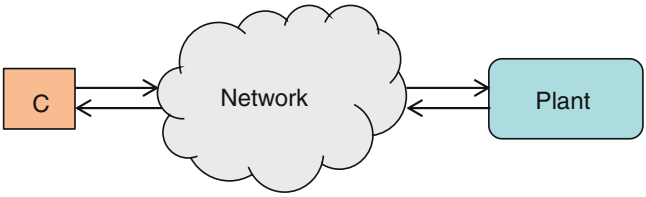

Fig. 1.2 Networked control scheme

- **Low cost** Using a point-to-point communication in large-scale systems or geographically distributed plants is generally a costly and impractical solution. Wireless or even wired networks, however, reduce the connections and the wire length. Concomitantly, the deployment and maintenance costs are shortened.
- **Reliability** In addition to the acknowledgment-retransmission mechanism of conventional communication protocols, a meshed network topology intrinsically improves reliability as dynamic routing allows to find alternative routes in the case that broken links are present. Additionally, fault detection algorithms can be easily implemented.
- **Maintenance** The reduction of wiring complexity facilitates the diagnosis and maintenance of the system.
- **Flexibility** Network structured systems offer flexible architectures, making easier the reconfiguration of the system parts and allowing a simpler addition of new devices.
- **Accessibility** Traditional centralized point-to-point control systems are no longer suitable to meet new requirements, such as modularity, control decentralization, or integrated diagnostics.

On the other hand, in a large number of practical situations the application or process advises engineers to use communication networks for control:

- **Space and weight limitation** Stringent limitations of this type need to be accomplished, for instance, in avionics (commercial aircrafts, unmanned aerial vehicles) or embedded systems in the automotive industry.
- **Coverage of considerable distances** chemical plants, large-scale factories, and automation systems.
- **Control applications where wiring is not possible** fleet of autonomous vehicles, safe driving control systems involving inter-vehicle communications, teleoperated systems, etc.

1.2.2 Communication Drawbacks

Communication through a shared network is imperfect and may be affected by some of the following problems (see Fig. 1.3):

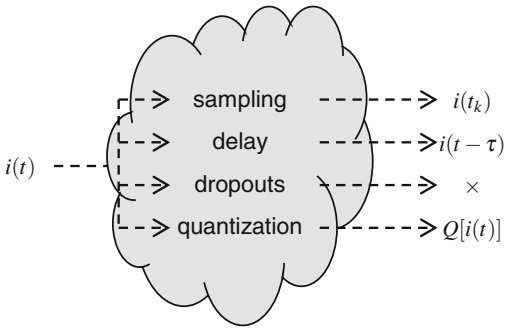

Fig. 1.3 The various problems affecting information $i(t)$ transmitted through a network

- **Sampling** In most digital networks, data are transmitted in atomic units called *packets*. These packets are sent at a finite rate, therefore continuous models must be discretized with an adequate sampling time. Since the available bandwidth is limited, sampling appears as a problem of the channel. In some network protocols, such as WiFi or Ethernet, this sampling time is not constant, as it strongly depends on the network traffic and congestion. A correct choice of the sampling periods will help to maximize the available bandwidth in those cases.
- **Delay** The overall delay between sampling and decoding at the receiver can be highly variable because both the network access delays (i.e., the time it takes for a shared network to accept data) and the transmission delays (i.e., the time during which data are in transit inside the network) depend on highly variable network conditions such as congestion and channel quality. Consequently, packets traveling through a network are received belatedly. For example, it is certainly common to receive one packet before another released earlier. Some protocols, such as TCP/IP, implement mechanisms accounting for this, but at the cost of increasing the delay. Even so, the reordering might be useless in control applications.
- **Packet dropouts** Some packets may also be lost, mainly because of the capacity of the reception buffer. If an element is receiving packets at a higher rate than it can process them, the buffer could overflow at any instant. Even, errors in physical links may cause the loss of information, as the packet must be discarded. Though some protocols guarantee data integrity through retransmission mechanisms, this is often useless in real-time control as old data packets cannot be used for control purposes. Indeed, many networked control algorithm discard and treat as losses those packets received with excessive delays.
- **Quantization** A quantizer is a function that maps a real-valued function into a piecewise constant function taking on a finite set of values. This mapping typically introduce inaccuracies inversely proportional to the cardinality of the representation alphabet. One of the basic choices in quantization is the number of discrete quantization levels to use. The fundamental tradeoff in this choice is the resulting signal quality versus the amount of data needed to represent each sample.

1.2.3 Research Trends

In the late 1990s, researchers began to identify the key distinctive issues of NCSs, driving the main research topics of the next decade.

- **Delays and packet dropouts** The *control-induced* delay, that is, the delay caused by the control scheme adopted, was first studied in the 1970s, when digital controllers were introduced to replace analog controllers. It was noticed that this kind of delay may induce by itself system instability, as was shown in a simple example in [271]. Digital controller design taking into account the computational delay has also been extensively studied, generally as extensions or applications of results developed for *time-delay systems* (TDSs) [13].
  Another source of delay is however present in NCSs, and it is caused by the transmission of the information through the network to the different components of the system. This kind of *control-induced* delay is commonly known in the literature as *network-induced delay* [242]. Network-induced delays, because of their discrete and distributed nature, are quite different from the plant delays and computational delays that have been studied in the past. However, in some cases, it is possible to use tools for linear sampled-data systems for the analysis and design of certain classes of linear NCSs [193]. Some problems also admit dealing with networked-induced delays in a similar way as traditional TDSs. This is the approach in some recent studies of time-delayed-system analysis and design [40, 122, 155, 168, 222, 280], which though not specific to NCSs, provide results that are applicable to NCSs.
  From a historical perspective, first results in the topic of network-induced delays in control systems were developed for the assessment of systems performance and design of improved communication protocols [95, 244]. Network time delays have since then been tackled in a variety of forms. In general, there are two methods to handle the networked-induced delays. One method is to design control algorithms considering the delays, such as in [159, 270]; the other is to reduce the delays by sharing a common network resource. Recently, part of the research [121, 144] on NCSs has focused on how to schedule network resources to make the network-induced delays as small as possible. These research results have also shown that network scheduling plays a subordinate, but very important role in NCSs. Other approaches tackle the problem from a robust control perspective, guaranteeing stability and performance in spite of the presence of delays. In many of these designs, the so-called maximum allowable delay bound (MADB) is established. The MADB can be defined as the maximum allowable interval from the instant when the sensor nodes sense the data from a plant, to the instant when actuators apply to the plant the corresponding control actions. For guaranteeing an NCS being stable, the sampling periods must be less than the corresponding maximum allowable delay bounds (MADBs) [38, 83, 145, 173, 268, 275, 277].
  Another significant difference between NCSs and standard digital control is the possibility that data packets may be lost while in transit through the network. For

a given sampling frequency, implementing estimation methods in an NCS would reduce the network traffic increasing the effective bandwidth of the system [25].

- **Band-Limited Channels** Any communication network can only carry a finite amount of information per unit of time. In many applications, this limitation poses significant constraints on the operation of NCSs. First incursions in the topic came from well-established results of information theory. A significant research effort has been devoted to the problem of determining the minimum bit rate that is needed to stabilize a linear system through feedback [26, 65, 105]. Recently, some progress has also been made in solving the finite-capacity stabilization problem for nonlinear systems [150, 191], derivation of stability conditions based on anytime information [223], or the study of performance limitations of feedback over finite capacity memoryless channels [161], with Bode-like extension limits of performance.
- **Stability of NCS** Unlike regular control systems, in NCSs the synchronization between different sensors, actuators, and control units is not guaranteed. Furthermore, there is no guarantee for zero delay or even constant delay in sending information from sensors to the control units and control units to the actuators. In real-time systems, particularly control systems, delays or dropped packets may be catastrophic and may cause instability in the process. Moreover, the time-varying nature of delays in NCSs may induce instability for time-varying delays in a bounded set; even when the NCS with any constant delay taken from this set is asymptotically stable [264].

  Stability under such circumstances has been investigated by a number of researchers. First results were obtained from the application of classical tools, as in [125] where a frequency-domain stability criterion, based on the small gain theorem, is proposed to investigate the stability of SISO NCS plants. A different modeling approach is used in [189, 274], where a continuous-time description, with a zero-order-hold controller, is proposed. Other relevant results regarding stability of NCSs can be found in [45, 102, 146, 151, 253, 273, 282, 283].
- **Energy aware** In all fields of engineering, energy-efficiency is becoming very important due to economical and environmental concerns. In networked control systems—especially if battery-powered devices are employed over wireless—energy-saving is key to increasing the lifespan of the system and, indirectly, reducing costs. Moreover, in some applications, the network devices can be deployed over hazardous or unreachable locations, and replacing the batteries may be expensive or impractical. This motivates current interests in developing energy-aware NCS methodologies, in particular, on protocols to reduce the average media access rate, as it is well known that wireless devices consume most energy when the radio is on.
- **Wireless Sensor Networks** A technological factor that has definitely amplified the impact of NCSs, both in industry applications and interest from academia, has been the rapid developments of wireless technologies in the past decade. Recent achievements in miniaturization, such as MEMS- and nano-technologies, have enabled the development of low-power, reduced-cost wireless devices with the capacity of establishing meshed networks in the so-called wireless sensor networks

(WSN). It is widely believed that this type of pervasive networking technology will be transparent to the user, but at the same time will allow monitoring and automation to an unprecedented scale.

- **Distributed systems** The challenge to the field is to go from the traditional view of control systems as a single process with a single controller, to recognizing control systems as a heterogeneous collection of physical and information systems with intricate interconnections and interactions. In addition to inexpensive and pervasive computation, communication, and sensing—and the corresponding increased role of information-based systems—an important trend in control is the move from low-level control to higher levels of decision making.

New possibilities and challenges arise in this context, and issues as distributed estimation and control over WSN, energy-aware NCS control, or multi-agent control are hot topics nowadays. Particularly, distributed estimation has been devised as a potentially useful strategy since the early 1990s [87], though it has found a renewed interest in the past few years with the development of WSNs. Distribution estimation techniques has been developed under different levels of imperfect channel assumptions in [128, 266, 281], and more recent unified control and estimation approaches can be found in [171, 206].

As we look forward, the opportunities for new applications that will build on advances in control expand dramatically. The advent of ubiquitous, distributed computation, communication, and sensing systems has begun to create an environment in which we have access to enormous amounts of data and the ability to process and communicate that data in ways that were unimagined 20 years ago. This will have a profound effect on military, commercial, and scientific applications, especially as software systems begin to interact with physical systems in more and more integrated ways.

1.2.4 Asynchronous Control

Traditionally, the information between sensors, actuators, and controllers is exchanged at constant rates. The sampling frequency has to guarantee the stability of the system under all possible scenarios, and this can sometimes yield a conservative choice of the sampling period. Moreover, all tasks are executed periodically and independently of the state of the plant.

In recent years, the idea of taking into account the plant state to decide when to execute the control and sampling tasks has received renewed interest. In general, in this non-conventional sampling paradigm, information is exchanged in the control loop when a certain condition depending on the state is violated. Hence, there is an adaptation to the needs of the process at any time.

However, there is no uniform terminology when referring to this concept. One can find in the literature the terms event-based control, event-triggered control, send-on-delta control, level-crossing control, self-triggered control, minimum attention

control, anytime attention control, and many more. All of them have basically the same idea, but vary in implementation. We will refer to *asynchronous control* or *asynchronous sampling* to cover all these approaches.

Despite its recent popularization, asynchronous sampling is not actually a new concept, and its origins date back to the late 1950s when it was argued that the most appropriate sampling method is to transmit data when there is a significant change in the signal [66]. Later, in the 1960s and 1970s, a heuristic method called *adaptive sampling* [60] was popularized. The objective was to reduce the number of samplings without degrading the system performance, evaluating in each interval the sampling period.

More recently, an event-based PID controller was implemented in [12] showing that the number of control updates was reduced without degrading the performance of the system. In [98], level-crossing control was applied to control the angular position of a motor with a low-resolution sensor.

The first analytical results were for first-order linear stochastic systems in [214], showing that under certain conditions the event-based control outperforms the periodic control. But the real impulse to the asynchronous control came out a few years later when many researchers realized the benefits of applying this theory to networked control systems. Section 1.4.2 will present a literature review of asynchronous control applied to NCSs as well as the main concepts used in this formalism.

1.3 Applications and Industrial Technology Over Network

Networked control systems have been finding application in a broad range of areas. Because of the attractive benefits detailed in Sect. 1.2.1, many industrial companies and institutes have shown interest in applying networks for remote industrial control purposes and factory automation [242]. The fact that many infrastructures and service systems of present-day society can naturally be described as networks of a huge number of simple interacting units increases the areas where NCSs can be applied.

For these reasons, these systems have a lot of potential applications, including environmental and pollution monitoring [113], control of water distribution networks [113], surveillance [16, 43], remote surgery [167], distributed power systems and smart grids [5, 24], mobile sensor networks [111, 198], formation control of autonomous vehicles [86, 229], haptics collaboration over the Internet [106], intelligent transportation systems [178], unmanned aerial vehicles [116] and chemical and petrochemical plants [267], just to name a few. Next, some of these NCS applications are detailed.

Wireless Sensor Networks

Built on nodes, are gaining a role of importance taking part of embedded systems. Embedded systems, by definition, interact with the physical world as sensors, actuators, and controllers that are programmed to perform specified actions. As the range

of applications grows, the demand to perform incrementally complex tasks on the nodes also increases.

In general, each node has four main parts: I/O ports connected with sensors and actuators, a radio transceiver to transmit the information, a microcontroller, and an energy source, usually a battery. Each node can monitor physical or environmental conditions such as humidity, temperature, lighting, and so on.

The advantage of WSNs with respect to traditional technologies is enormous, as deploying and maintaining a geographically distributed wired network of thousands of nodes is impractical considering the distances among nodes. WSNs themselves have several applications such as surveillance, health care, air pollution, water quality, or industrial monitoring, some of which will be commented later on. WSNs are characterized by the mobility of nodes, power consumption constraints, or node failures, all of them challenges the control design has to deal with.

Biological Systems

Renewable energy-based systems and mitigation of the greenhouse effect are two of the main concerns in the present century. Large efforts are being done around the world trying to look for clean resources and new technologies to face these issues [243]. Also, the problem of quality and quantity of water resources is a global challenge for the upcoming years. Both, an adequate amount and quality of water are essential for public health and hygiene [113].

Bioprocesses technology or biotechnology is one of the emerging areas that can highly contribute to the challenging aspects mentioned above as well as to produce high-value products. Bioprocess operations make use of microbial, animal, and plant cells and components of cells, such as enzymes, to manufacture new biotechnological products (food industry, pharmaceutical products, biofuel), destroy harmful wastes (CO_2 mitigation) [59], or obtain large quantities of water with good quality.

For example, finding suitable biofuel crops so that the oil production could replace fossil fuel usage is a trendy line of research. In this regard, microalgae are seen as the bioprocess with great potential for biofuel production in the future. Microalgal biomass can reach up to 80 % of dry weight under certain stress conditions; they can be cultivated in high area yields compared to other crops; they have high oil content in some strains, low-water consumption is required, and it is possible to produce them on arid lands [20, 196].

As far as water scarcity is concerned, water treatment and desalination plants seem a solution to provide the possibility to use water everywhere. Recycled water is most commonly used for non-potable purposes, such as agriculture, landscape, public parks, and industrial applications, among others (Fig. 1.4).

The development of new technologies has made possible the monitoring and control of such biological processes. The integration of specific sensors and actuators in motes and the adaptation of the network function to the specific requirements that this type of application impose are identified as key features. They allow the distributed monitoring and control that improve the efficiency, productivity, and optimization of these large-scale systems.

Fig. 1.4 NCS applications to agriculture

Remote Surgery

This enables the surgeon to remotely operate on the patient with the help of a medical telerobot. Theoretically, it frees the surgeon from the operation room, protects the surgeon from radiation, and provides rescue for patients in areas of difficult access [167]. Hence, the new developed technology will help to remove distance barriers from surgery.

This ability can benefit patients who would otherwise go untreated, improve the quality of care since expert surgeons can proliferate their skills more effectively, and reduce costs by avoiding unnecessary patient and surgeon journeys [27]. Yet, other obstacles such as licensing, reimbursement, liability, etc., cannot be ignored.

The first telesurgery prototypes were through wired connections [27, 167], but there are also some recent results on wireless remote surgery [158].

Smart Grids and Distributed Power Systems

We define a smart microgrid as a portion of the electrical power distribution network that connects to the transmission grid in one point and that is managed autonomously from the rest of the network [24]. The objective of transforming the current power grid into a smart grid is to provide reliable, high quality electric power in an environmentally friendly and sustainable way. To achieve this, a combination of existing and emerging technologies for energy efficiency, renewable energy integration, demand response, wide-area monitoring, and control is required (Fig. 1.5).

For instance, the so-called flexible AC transmission systems (FACTS) technology would allow to find the most efficient paths and better power production mixes and schedules. Additionally, the massive use of deployed sensors would make possible the measurement of the consumption of the end users at any time, weather data, or equipment condition. Monitoring, optimization, and control applications would

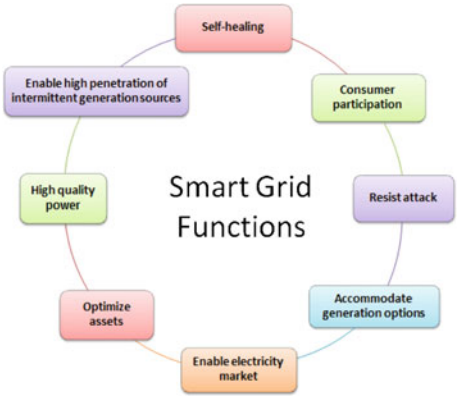

Fig. 1.5 Smart grids function diagram

increase the energy delivery efficiency and security by means of the dynamical computation of ratings and balance load and resources [227].

Intelligent Transportation Systems (ITS)

These are defined as those that utilize synergistic technologies and system engineering concepts to develop and improve transportation systems of all kinds. They provide innovative services related to different means of transport and traffic management. This will definitely achieve a *smarter* use of transport networks, making them safer and more coordinated.

Intelligent transportation technologies are based on wireless communications. The current trend is to develop new embedded system platforms that allow for more sophisticated software applications to be implemented, including model-based process control, artificial intelligence, and ubiquitous computing.

Applications of ITS are, for example, emergency vehicle notification systems, variable speed limits to control the traffic flow [263], travel time predictions [199, 221], collision avoidance systems, or dynamic traffic light sequence.

Formation Control

In many applications, a group of autonomous vehicles are required to follow a predefined trajectory while maintaining a desired spatial pattern [42]. Formation control has many applications. For example, in small satellite clustering, formation helps to reduce the fuel consumption and expand sensing capabilities. In military missions, a group of autonomous vehicles keeps a formation for exploratory purposes. Other examples include search and rescue missions, automated highway systems, detect, locate, and neutralize undersea mines by underwater vehicles, or mobile robotics [73].

Such autonomous vehicles can be coupled physically or through the control task to accomplish the specific task. Information is usually shared through a network to achieve the mission, and vehicles have only access to partial information when making decisions. Hence, new challenges arise in the control problem. For instance, communication is really weak in some scenarios, such as for underwater vehicles, where delays, reliability, and data rate constraints are very demanding (Fig. 1.6).

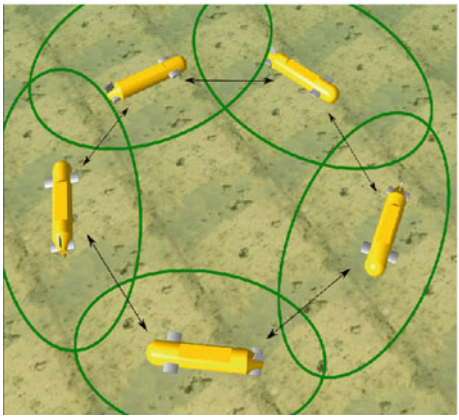

Fig. 1.6 Submarines in Formation

1.4 Networked Schemes: From Centralized to Distributed Techniques

Networked control systems are characterized by the transmission of sensor and/or control data through a shared network. Due to the finite bandwidth of the network, the flow of information is at discrete instances of time. This discontinuous flow is represented by dashed lines, whereas solid lines correspond to continuous signals. The flexibility that NCSs offer yields multiple possible architectures. In this book, we focus on the three most common configurations: centralized, decentralized, and distributed models.

1.4.1 Centralized and Decentralized Schemes

Since their inception, practically all the existing control and estimation techniques have been devised and developed for centralized schemes. In these schemes, every sensor or actuator of the plant is connected to a central agent that gathers all the data.

The advantages of centralized implementations have been widely exploited by systems engineers for decades. When a central agent collects all the available information of a system, monitoring and control tasks can potentially achieve high performances. In addition, there is a wide body of knowledge and a huge variety of techniques developed for centralized implementation, which means that the experimented practitioner can select the one that fits the system needs over a number of different possibilities.

In a centralized scheme (Fig. 1.7), the central unit receives the measurements $\{y_i(t)\}$ taken by the sensors in the plant and sends the control actions $\{u_i(t)\}$ back to the system.

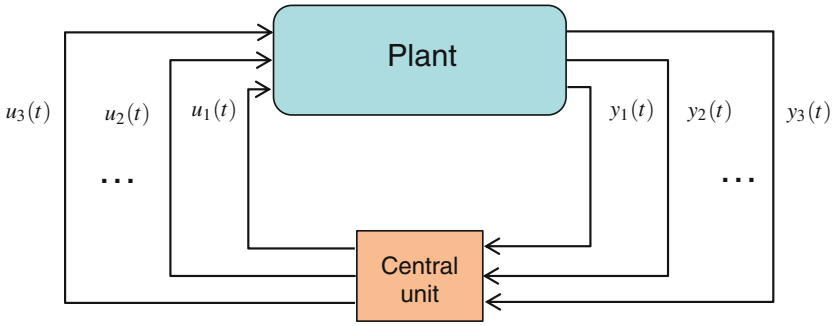

Fig. 1.7 Centralized architecture

In centralized NCS, there are different configurations depending on how the sensors (S), the actuators (A), and the controller (C) are located with respect to the network (see Fig. 1.8). Thus, the controller can be co-located with the sensor nodes (Fig. 1.8a), co-located with the actuators (Fig. 1.8b), or work as a remote controller (Fig. 1.8c):

- **Co-located with sensors** This architecture offers the advantage of providing the unaltered outputs instantaneously to, if necessary, reconstruct the state of the system. Thus, the synchronization of the controller with the sensors is a fair assumption in this case. The controller computes the control inputs that are transmitted

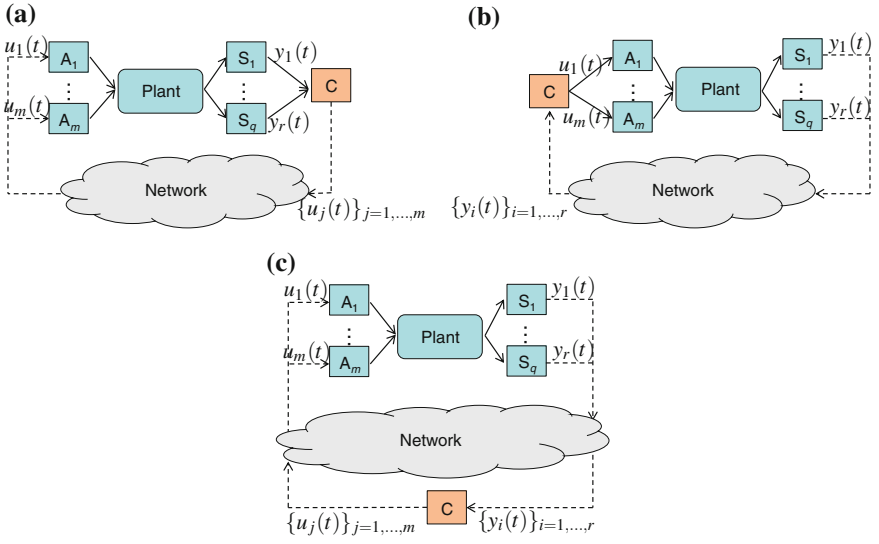

Fig. 1.8 Centralized models in NCSs

through the network at discrete instances of time (equidistant from each other or not) to the actuators, which might not have clocks' synchronization with other nodes.

- **Co-located with actuators** Information about the state of the system is transmitted from the sensor nodes to the controller through the imperfect channel. The controller will gather this information to calculate the control signals that are delivered to the actuators immediately.
- **Remote controller** This is the most general framework and the network is on both sides of the controller, which in general will not be synchronized with the other nodes in the network. Transmission of both sensor measurement and control inputs will suffer from the network imperfections.

In general, the control law is given as

$$u(t) = k(y(t)),$$

where $u(t) = (u_1(t) \dots u_m(t))^T$ and $y(t) = (y_1(t) \dots y_r(t))^T$.

Centralized architectures require to connect every device to a central node. This can be unsuitable in some applications, especially in the context of large-scale systems as, for instance, some of the applications detailed in Sect. 1.3. The implementation of centralized architectures in these kinds of systems may be challenging as important problems usually arise: technical difficulties to transmit all the system signals in real time, security issues, robustness against connection failures, high wiring costs, or excessive computational burden in the central controller.

In contrast, in decentralized schemes (Fig. 1.9), the tasks over the system are performed by a set of independent controllers suitably deployed [15, 213]. This way, each controller has access to local data and manages specific input/output channels. In decentralized architectures the computations can be carried out in parallel and the wiring costs are minimized, which also means reduced danger of breaking cables, less hassle with connectors, etc. Nonetheless, an important disadvantage of this approach is that the absence of communication between agents limits the achievable performance.

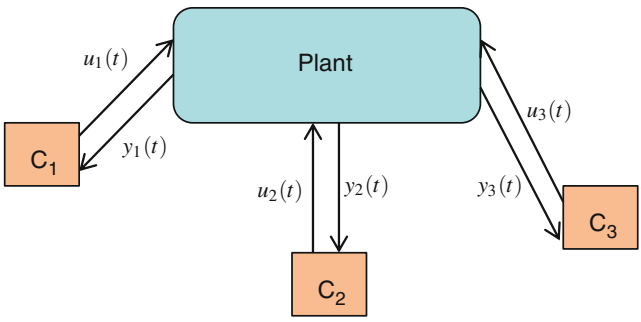

Fig. 1.9 Decentralized architecture

Each control unit C_i computes the control input $u_i(t)$ based on the local measurement $y_i(t)$. In general, the control law is

$$u_i(t) = k_i(y_i(t)).$$

1.4.2 The Middle Ground: Distributed Systems

Distributed systems are the middle ground that lies between decentralized and centralized solutions. As in decentralized architectures, in distributed systems the agents have access to local plant data. Thus, distributed architectures (Fig. 1.10) require lower levels of connectivity and less computational burden than centralized approaches.

However, as opposed to decentralized schemes, in this framework the controller nodes are endowed with communication capabilities and they can share information with a limited set of neighboring controllers (agents), which allows this approach to improve the performance. Therefore, distributed control systems (DCSs) are networked control systems where it is possible to trade-off between communication burden and control performance.

Distributed control and estimation techniques are becoming more and more popular with the development of wireless sensor networks, which has made easier the implementation of distributed control systems and has simplified deployment, migration, and decommissioning of networks, among other elements, see [225], or [2].

Nowadays, most vendors offer wireless-enabled product lines with different technologies (WSAN from ABB or OneWireless Network from Honeywell, to give a couple of examples). Although further efforts must be made to improve interoperability, computation capabilities, and connectivity of present devices, the scenario

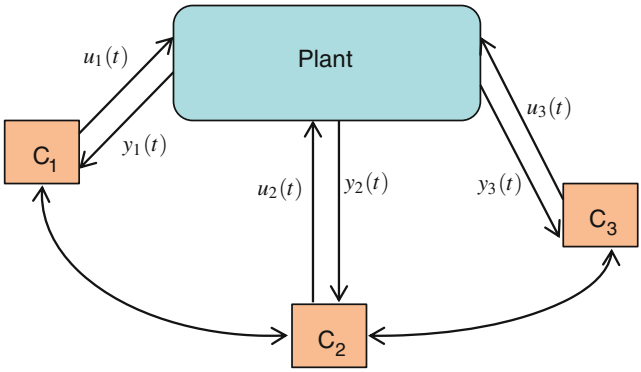

Fig. 1.10 Distributed architecture

where off-the-shelf components with the attributes required to implement sophisticated collaborative control/estimation schemes are available, is not so far in the future.

In contrast, compatibility, standardization, and integration of DCSs with other aspects of process control (human–machine interface, alarm systems, historical records, etc.) are still important issues to be resolved for a wider implementation of these systems. Besides, due to various design considerations, such as small size battery, bandwidth and cost, from the control design point of view, two types of interconnections between subsystems that compose the overall plant are distinguished. The first one is the physical interconnection, i.e., the state of a subsystem i directly drives the dynamics of another subsystem j. This fact can be used in the control design of the subsystem j to compensate this interconnection if the state of the subsystem i is available at j. The second type of interconnection is when the need for communication between the controllers comes from the fact that the system tries to achieve a common objective, such as for example, consensus. This leads to cooperative control. The usual terminology to refer to these systems in which the gathering of information from individual parts is used to control the global behavior of the networked system is *multi-agent systems*.

A scheme of a distributed NCS is depicted in Fig. 1.11. Each node i has a local controller C_i, which receives the local information $y_i(t)$ and also some but not all other information $y_j(t)$ from other subsystems (also called agents) measured at different instances of time. The agents that transmit information to i are known as its neighborhood (denoted by $\mathcal{N}_i$) and correspond to the ones that are interconnected with agent i. Hence, the control input $u_i(t)$ of the ith subsystem is

$$u_i(t) = k_i(y_i(t), \{y_j(t),\ j \in \mathcal{N}_i\}).$$

This scheme can be extended to more general frameworks. For instance, agents can exchange state estimations, different representation of sets, or other parameters.

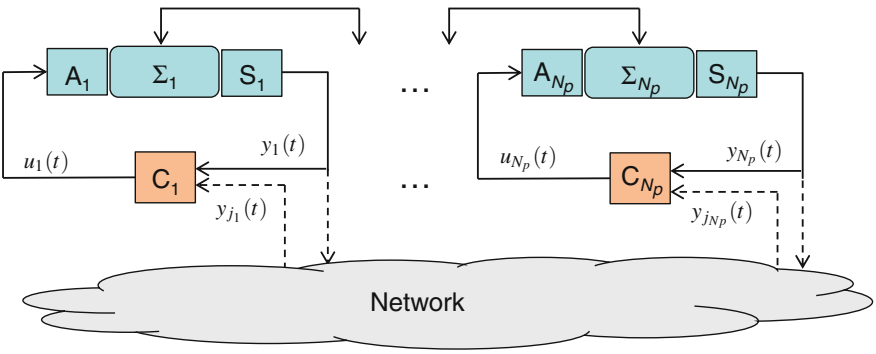

Fig. 1.11 Distributed NCS

1.5 Communication Through a Non-reliable Network

The main limitations imposed by an imperfect communication channel have been introduced in Sect. 1.2. To illustrate these concepts, let us consider the situation depicted in Fig. 1.12. There are two nodes in the network: the sender and the receiver. The first one wants to transmit some data to the other. The sender can be a controller, a sensor, or a subsystem of a distributed network, and the receiver can be an actuator, a controller, or another subsystem.

The first issue that makes different a networked system from a conventional control system is that the components are, in general, spatially distributed. As a consequence, the synchronization of the clocks of these components cannot be assumed normally, that is, measures of time are not equal. This phenomenon is illustrated in Fig. 1.13. On the left, sender and receiver have synchronized clocks. On the right, the measures of time differ from a value Δ, which is unknown by the nodes and is hard to compute. This makes difficult, for example, the measurement of delays.

The limited bandwidth that characterizes the network imposes that the amount of information transmitted per unit of time must be finite. Thus, on the one hand, analog signals must be transformed to be transmitted in a finite number of bits, which yields to quantization. The maximum amount of information that can be sent at once is given by the size of the packet, which depends on the network protocol. For instance, a packet can be divided into the control information, which provides the network needs to deliver the packet, and the user data, also known as *payload*. The size of the payload goes from 1500 bytes in Ethernet to 8 bytes in some Radio Frequency protocols used to communicate small devices.

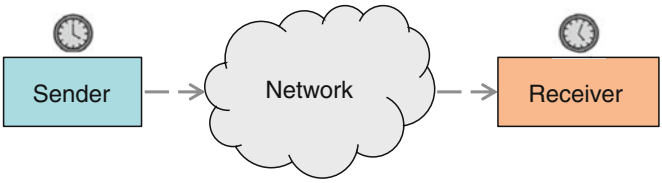

Fig. 1.12 Two nodes connected through the network

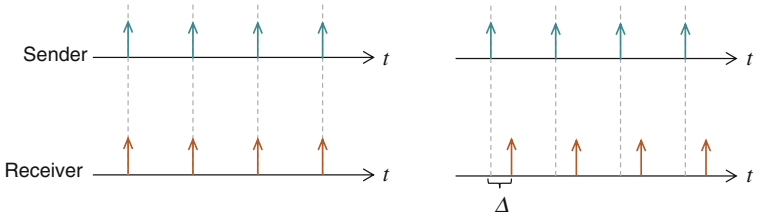

Fig. 1.13 Synchronization

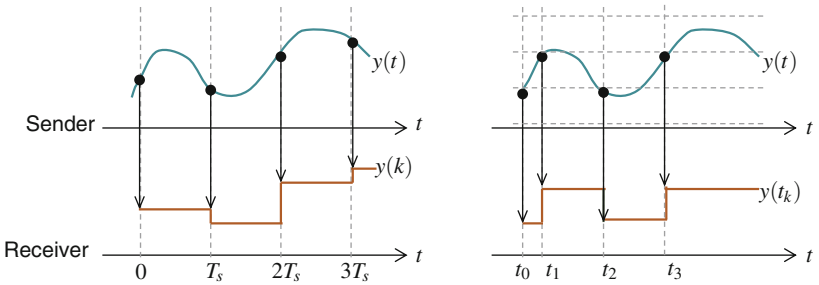

Fig. 1.14 Periodic and event-based sampling

On the other hand, the values of these signals can only be transmitted at discrete time instants. In this regard, there exist two alternatives as shown in Fig. 1.14. On the left, the measurements of the signal $y(t)$ in the sender node are sent to the receiver at equidistant instances of time given by a period T_s. Hence, the data received is $y(kT_s)$, $k \in \mathbb{N}$. For example, if we think that $y(t)$ is the output measured by a sensor, this technique corresponds to periodic or time-driven sampling, in the sense that the actions are taken based on the passing of time.

By contrast, when the transmission of data are not equidistant in time, and it is the value of the signal that matters in the decision of when to send the samples, we talk about event-driven or event-triggered sampling. Note that, for instance $t_1 - t_0 \neq t_2 - t_1$ on the right-hand side of Fig. 1.14. For a general value $k \in \mathbb{N}$, the difference between $t_{k+1} - t_k$ is called *inter-event time* and is denoted by T_k. Other authors also refer to this magnitude as broadcasting period [259].

The last concepts we want to illustrate are the network delays and the data dropouts. The reasons why these problems occur in a networked system have been discussed in Sect. 1.2. As stated there, some network protocols implement mechanisms to control the flow of packets. For instance, one common approach is to use *acknowledgment* (ACK), that is, the transmission of a small packet to confirm the reception of data. If ACK is not received after some waiting time (T_W), the sender deduces that the packet must have got lost and will try to retransmit the packet.

Let us illustrate these concepts with an example. For simplicity, assume that the sender and the receiver have synchronized clocks and periodic transmission of information as in Fig. 1.15. First, some data are sent at $t = 0$, which is received after some time τ_1 due to some delay in the transmission. Secondly, at $t = T_s$ new data are transmitted and dropped, for example, for some error in physical links. Data are retransmitted according to the protocol described above at the next sampling time. This causes the information to be finally received after some time τ_2. Hence, data dropouts and delays are related. In general, if n_p denotes the number of consecutive data dropouts and τ is the transmission delay, the effective delay is $n_p T_s + \tau$. For instance, if a control input $u(t)$ is computed by some controller node (*sender*), sent to an actuator node (*receiver*), and directly applied when received, the dynamics of the plant are in the continuous time

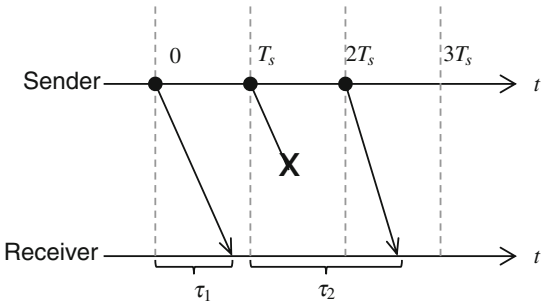

Fig. 1.15 Example of delays and data dropouts

$$\dot{x}(t) = f(x(t), u(t - (n_p T_s + \tau))),$$

or in the discrete time

$$x(k+1) = f(x(k), u(k - (n_p T_s + \tau))).$$

There usually exists an upper bound on the effective delay over which the system is unstable. Time-delay systems are by themselves an extensive research area in control theory. In this book, different strategies are proposed to deal with these kinds of problems and to compute bounds on the effective delay.

1.6 Asynchronous Control in NCSs

1.6.1 Event-Based Control Approaches in the Literature

In most implementations, an event is triggered when some error function exceeds a tolerable bound. How this error function and this bound are defined distinguishes the different approaches in the literature that are discussed next.

Deadband Control

If the error is defined as the difference between the state of the last event occurrence and the current state, and the bound is defined as a constant, an event is triggered whenever

$$\|\varepsilon(t)\| = \|x(t) - x(t_k)\| \leq \delta,$$

becomes positive, where t_k refers to the instant of the last event and t is the current instant of time. The value of δ determines, on one hand, the performance of the system and the ultimate set in which the state of the plant is confined around the equilibrium, and on the other hand, the average frequency of communication. Figure 1.16a, b depict two examples of deadband control for a first-order and a second-order system, respectively. Some works related to deadband control are [99, 226].

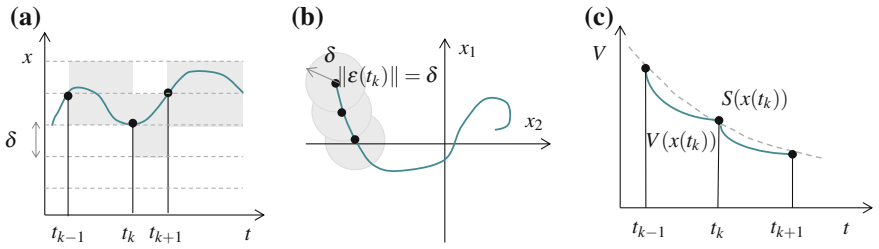

Fig. 1.16 Examples of triggering rules

Lyapunov Approaches to Asynchronous Control

Deadband control does not generally yield asymptotic stability. And so, some researchers have investigated triggering rules to fulfill this. One example is presented in [239] where the error is bounded by the state at the current time

$$\|\varepsilon(t)\| = \|x(t) - x(t_k)\| \leq \sigma \|x(t)\|.$$

This approach yields the asymptotic stability of the system but the inter-event times become shorter when the system reaches equilibrium. In [239] it is shown that a minimal inter-event time is guaranteed to exist only under suitable assumptions.

Other authors have exploited the idea of using Lyapunov methods to define the triggering rule [163]. An event is triggered when the value of the Lyapunov function of the closed-loop system for the last broadcast state reaches a certain threshold of performance $S(x, t)$ (see Fig. 1.16c):

$$V(x, t) \leq S(x, t).$$

This condition also guarantees that equilibrium is reached asymptotically.

Time-Dependent Event-Triggering

Recently, time-dependent triggering rules have been proposed to reach the equilibrium point asymptotically. In [89, 232], the trigger functions for linear interconnected systems and multi-agent systems, respectively, bound the error as

$$\|\varepsilon(t)\| \leq \delta e^{-\beta t}, \quad \delta, \beta > 0,$$

which has the aforementioned property, guaranteeing a lower bound for the inter-execution times. Note that this bound approaches to zero when $t \to \infty$, but still the Zeno behavior is avoided even in non-ideal network conditions.

Self-triggering

Sensor networks are a special case of networked control systems in which the energy consumption plays a crucial role. Thus, event-triggering approaches are convenient in

sensor networks since the number of transmissions can be decreased. However, it has been discussed [8, 10, 165] that most of the energy consumed in a sensor node comes from the task of monitoring the measured variable(s) rather than the transmission. The asynchronous control strategies discussed above require the continuous monitoring of the state. For this reason, a new approach known as self-triggered control has emerged in the recent years.

Self-triggering policies determine the next execution time t_{k+1} by a function of the last measurement of the state x_k. The sensor nodes do not monitor the process until they are woken up at time t_{k+1}, they take the measurement and transmit it, and the next execution time is computed again. The concept of self-triggering was first suggested by [251]. Self-triggered control can be regarded as a software-based emulation of event-triggered control. It has been studied for linear systems [164, 256], nonlinear systems [8, 241], and applied to sensor and actuator networks in [9, 28, 165, 240].

A general problem of this scheme is the consideration of unknown effects, such as model uncertainties or unknown exogenous disturbances. To cope with all these effects conservative results have to be derived to guarantee the stability of the self-triggered control loop which may lead to relatively short sampling intervals in practice [258].

Minimum Attention Control (MAC) and Anytime Attention Control (AAC)

Minimum attention control maximizes the time interval between executions of the control task, while guaranteeing a certain level of closed-loop performance [7, 58]. It is similar to self-triggered control in the sense that the objective is to have as few control task executions as possible but it is typically not designed using emulation-based approaches. In [58] an approach based on extended control Lyapunov functions allows to solve the problem online alleviating the computational burden as experienced in [7]. However, MAC is by far less robust against delays or disturbances than event-based control. Similar problems present the so-called 'anytime control' methods, which are an alternative way to handle limited computation and communication resources [84, 93, 94]. The AAC proposed in [7] assumes that after each execution of the control task, the control input cannot be recomputed for a certain amount of time that is specified by a scheduler, and finds a control input that maximizes the performance of the closed-loop system.

Periodic Event-Triggered Control

Periodic event-triggered control strikes a balance between periodic control and event-based control. As self-triggered control, it avoids continuous monitoring of the system outputs while preserving the reduction in resource utilization. So, instead of checking the trigger condition continuously, this is only evaluated at instances of time defined by a period T_s.

The design methods that have been proposed [97] use Lyapunov-based trigger functions and provide the tools to check stability and performance for a given control gain and a sampling period. One additional advantage is that it guarantees a minimum inter-event time of (at least) the sampling interval of the event-triggering condition.

Model-Based Event-Triggered Control

All the approaches described above consider zero-order hold at the actuator, i.e., the control input computed at event times is held constant till the next event occurrence. Although this consideration of *doing nothing* between events simplifies the analysis, it has been shown that if a precise model of the plant is available, a control input generator can emulate the continuous-time state feedback loop and under certain constraints get a better performance than a zero-order hold [154]. The idea of taking advantage of a model in NCS and working in open loop is not new and was introduced in [181, 183], though the updates from the system are periodic, not event-triggered. However, emulation approaches such as [154] require synchronization of all the elements in the control loop, and this constraint is difficult to meet in the case of remote controllers or in distributed paradigms.

Asynchronous Control and Output Measurement

The triggering rules presented previously are all based on full state measurement, although in practice the full state is not often available. If the same setups are tried to be used for output feedback controllers, the Zeno behavior might occur, as pointed out in [57].

To solve this problem, the existing approaches to output-based asynchronous controllers can be categorized as observer-based or not. To the first category belong [141, 143]. The measured state is replaced in the trigger function by the estimated state provided by the observer [141] or the filter [143]. The second direction is to use a different structure in the controller. A dynamical output-based controller is proposed in [56]. Using 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 [135], where an LTI continuous-time controller is emulated.

1.6.2 Event Definitions

We have just introduced the idea of event-based or event-triggered sampling (control). Let us formulate it in a formal way.

For simplicity, let us state full state measurement. If $x(t)$ is the state of the system and $x_b(t)$ accounts for the information available at the controller k, the error can be defined as

$$\varepsilon(t) = x_b(t) - x(t).$$

Then, the system is described as

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ u(t) &= k(x(t) + \varepsilon(t)) \end{aligned}$$

To formulate a general setup we assume that the triggering condition is given by some function $F_e(x(t), \varepsilon(t), t)$, which is jointly continuous in x and ε.

The sequence of event or *broadcasting* times t_k is determined recursively by the trigger function F_e as

$$t_{k+1} = \inf\{t : t > t_k, F_e(x, \varepsilon, t) > 0\}.$$

Most of the triggering conditions set a bound on the error function and, hence, the trigger function can be written as

$$F_e(x(t), \varepsilon(t), t) = ||\varepsilon(t)|| - \delta(x(t), t).$$

Of course, this includes the case of δ being a constant.

We say that the triggering scheme induces *Zeno behavior*, if for a given initial condition x_0 the event times t_k converge to a finite t^*. This means that $T_k = t_{k+1} - t_k$ tends to zero. This is, of course, an undesirable behavior since it requires the detection of events and transmission of data infinitely fast. Hence, the design of trigger functions F_e has to guarantee the existence of a lower bound for the inter-event time T_k.

Equivalent definitions can be given for discrete time systems. In this case, the event times are a multiple of the sampling period and, therefore, the Zeno behavior is excluded by construction.

This formalism is based on the continuous monitoring of the state $x(t)$, which requires waste of computational resources and, as a consequence, of energy. A self-triggered implementation is given by a map $F_h : \mathbb{R}^n \to \mathbb{R}$ determining the next sampling time t_{k+1} as a function of the state $x(t_k)$ at the time t_k, i.e.,

$$t_{k+1} = t_k + F_h(x(t_k)).$$

The most common implementation of F_h consists of predicting future states of the plant based on a model of the system:

$$\begin{aligned} \dot{x}_m(t) &= f(x_m(t), u(t)),\ x_m(t_k) = x(t_k) \\ u(t) &= k(x(t_k)),\ t_k < t < t_{k+1}. \end{aligned}$$

Then, the Lyapunov function at the current event time t_k, $V(x(t_k))$, and at the future times t, $V(x_m(t))$, are evaluated. The next event time t_{k+1} will be the first value of t such that

$$\Delta V(x_m(t), x(t_k)) = V(x_m((t)) - V(x(t_k))\gamma(t, t_k) \geq 0.$$

The function $\gamma(t, t_k)$ can take the value 1 and, therefore, the next sampling time will be when the computed Lyapunov function $V(x(t))$ exceeds the current value $V(x(t_k))$. An alternative that ensures the exponential decrease of the Lyapunov function is $\gamma(t, t_k) = e^{-\alpha(t-t_k)}$, $\alpha > 0$.

1.7 Stability and Performance Measurements

As in conventional systems, guaranteeing the stability of a networked system is essential. The two main approaches for verifying stability found in this book are spectral theory for linear systems and Lyapunov functions for both linear and nonlinear systems. Additionally, in order to address delays, extensions of the Lyapunov function concept in the sense of Krasovskii or Razumikhin can be used. In general, the Lyapunov–Krasovskii theory yields less conservative results, and this will be the preferred approach in this monograph.

We next introduce some concepts that will be used throughout the book.

Definition 1.1 The state of the system $x(t)$ is asymptotically stable if

$$\lim_{t\to\infty} ||x(t)|| = 0.$$

where $||\cdot||$ denotes an arbitrary matrix or vector norm.

Definition 1.2 A square matrix A is said to be Hurwitz if every eigenvalue of A has strictly negative real part, that is,

$$\mathbb{R}e[\lambda_i(A)] < 0,$$

for each eigenvalue λ_i.

It is also called the stability matrix, because then the differential equation $\dot{x}(t) = Ax(t)$ is asymptotically stable.

Analogous definitions can be given for a discrete-time system $x(k+1) = Ax(k)$. In that case, the condition over the eigenvalues is to lie inside the unit circle.

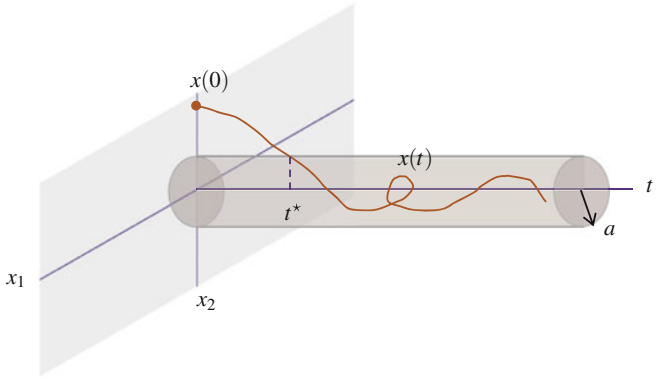

Fig. 1.17 Ultimate boundedness

For some triggering conditions in event-based control (deadband control), the asymptotic stability of the system cannot be guaranteed. A more appropriate stability definition is given by ultimate boundedness which is illustrated in Fig. 1.17 and is defined next.

Definition 1.3 The solution $x(t)$ of a continuous-time system $\dot{x}(t) = f(x(t), u(t))$ is globally uniformly ultimately bounded (GUUB) if for every $x(0) \in \mathbb{R}^n$ there exists a positive constant a and a time $t^\star$ such that the following holds:

$$x(t) \in \Omega_t \triangleq \{x : ||x|| < a\}, \forall t \geq t^*.$$

An interesting phenomenon that has been observed in some event-based control systems is the occurrence of oscillatory behaviors around the equilibrium point, and more specifically, of limit cycles. A limit cycle is defined as an isolated closed curve.

The stability of limit cycles can be defined in similar terms as for equilibrium points. For instance, if Γ is the closed orbit (limit cycle), we say that Γ is globally asymptotically stable if for any $x(0)$

$$\lim_{t\to\infty} \inf_{y\in\Gamma} ||x(t) - y|| = 0.$$

Proving global asymptotic stability is hard and most of the existing results are only for local stability [35, 82].

Limit cycles are inherent properties of nonlinear systems, and the fact that event-based control/sampled systems are nonlinear even though the dynamics of the original system is linear, is the reason why this phenomenon occurs in some types of deadband controllers. This will be studied in Chap. 2.

Part I
Asynchronous Control for Single-Loop Schemes. Centralized Solutions

This block consists of five chapters. Chapters 2 and 3 study scenarios in which the network effects can be neglected. Then, Chaps. 4–6 present different asynchronous control approaches to deal with delays and packet losses, while reducing the network traffic.

Chapter 2
Send-on-Delta PI Control

Jesús Chacón, José Sánchez and Antonio Visioli

2.1 Introduction

The work described in this chapter is focused on *event-triggered* sampling, and in particular on the level crossing or *send-on-delta* (SOD) sampling [12]. Send-on-delta control and deadband control are similar schemes, and they are normally used as synonyms, since both consist in taking a new sample when a change greater than a predefined threshold δ is detected in the signal. However, in send-on-delta sampling, it is the signal itself and not the error function $\varepsilon(t)$ (defined as the difference between the last measurement taken and the current value of the signal) what is used to trigger an event. In practice, this scheme is equivalent to introduce a nonlinearity that can be characterized as a quantization of N_l levels with hysteresis.

The behavior of a control scheme based on level crossing sampling is studied considering two possible structures, the first one is when the sampler is located after the process output (Fig. 2.1a), and the second one after the controller output (Fig. 2.1b). Each case represents a configuration of a control scheme based on wireless transmissions, and has different properties. The first case corresponds to a plant with a wireless sensor which takes measures of the process variable and sends them to the controller, physically separated from the sensor but connected to the actuator. The second case is the opposite, where the controller is directly connected to the sensor and the actuator is in other place.

J. Chacón (✉) · J. Sánchez
Dpto. de Informática y Automática, Escuela Técnica Superior de Informática,
UNED, Madrid, Spain
e-mail: jchacon@bec.uned.es

J. Sánchez
e-mail: jsanchez@dia.uned.es

A. Visioli
Dipartimento di Ingegneria Meccanica e Industriale, Facoltá di Ingegneria,
Universitá degli Studi di Brescia, Brescia, Italy
e-mail: antonio.visioli@ing.unibs.it

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_2

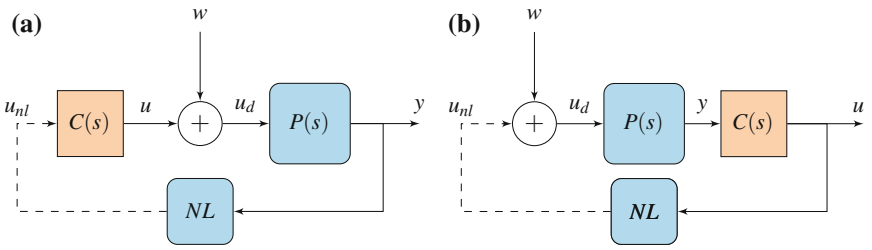

Fig. 2.1 Event-based control schemes. The block diagrams correspond to the two proposed configurations, **a** the event-based sampler at the process output and **b** at the controller output

The interest is to characterize with a systematic approach the behavior of the two event-based control structures with a set of processes such as integrator processes plus time delay process (IPTD), first-order processes plus time delay (FOPTD), and second-order processes plus time delay (SOPTD). The analysis is focused on the conditions for the existence of limit cycles, their period and amplitude, the effect of external disturbances, and the windup phenomena in the process due to the saturation of the actuators.

There are other works in the literature concerned to the study of limit cycles in event-based systems. In [216], authors analyze the existence, properties, and stability of limit cycles in relay systems. Related results can also be found in [33], where an event-based control scheme with a simple threshold detector is investigated, first in a double integrator and afterwards in the general case. Another approach can be found in [140], where the proposed structure is a model-based event-triggered PI in which the events are generated by the difference between the plant and the model implemented in the controller/sensor. The effect of actuator saturation is also investigated. In [81], a new method to perform the global stability analysis in piecewise linear systems (PLS) is proposed. The method is based on the use of quadratics Lyapunov functions in the switching surfaces. Finally, in [21, 22], authors analyze symmetric limit cycles in send-on-delta PI controllers, focusing on the stability of FOPTD processes and proposing tuning rules for this kind of controllers.

The main contributions of this chapter are the proposition of two general event-based schemes, and a new method for the analysis of the limit cycles that appear in the two presented schemes when they are applied to linear time-invariant (LTI) systems with time delay. The method has been applied to study a set of the most common industrial processes, obtaining results that have been confirmed in practice. In particular, a set of experiments carried out in the Distributed Collector Solar Field at the Solar Platform of Almería (PSA) showed the expected behavior.

2.2 The LTI and SOD Sampler Blocks

In the event-based control structure used in this work, three elements are present: the multilevel nonlinearity with hysteresis represented by the SOD sampler, the process (IPTD, FOPTD, or SOPTD), and the PI controller. In order to redefine the

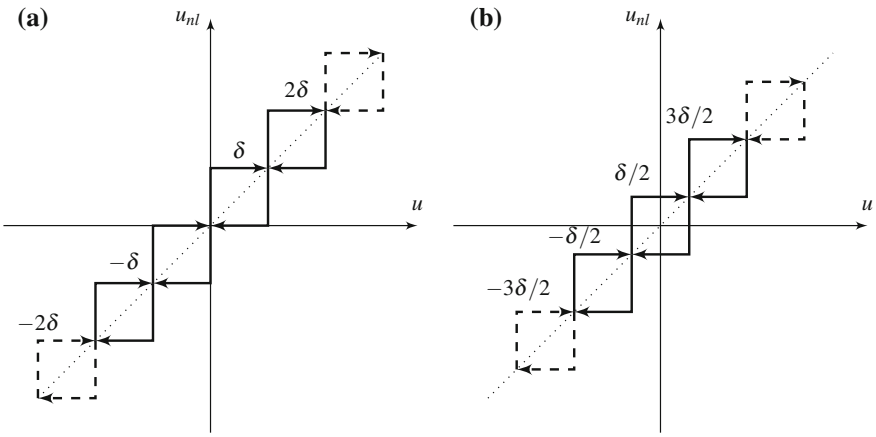

Fig. 2.2 Two cases of send-on-delta sampling with a different offset

event-based system as a PLS, first it is necessary to group the dynamics of the two linear elements in one block, that is, process and controller. Once this is done, the redefinition of the system as a PLS is simple since feedbacking the new linear block with the nonlinearity is equivalent to introduce a rule to switch between the N_l systems obtained by the combination of the dynamics of the linear block and the N_l output levels of the sampler.

Figure 2.2a shows the nonlinearity corresponding to the non-centered level crossing with saturation, and Fig. 2.2b represents a centered sampling with saturation. The dotted lines in both plots mean that the sampler has a saturation, but in general, the levels can be extended in both directions.

2.2.1 The Nonlinear Block: The SOD Sampler

The *event-triggered* control systems analyzed in this work use a level crossing sampling strategy, where, depending on the sampler location, the sensor sends information to the controller only when the sampled signal crosses certain predefined levels, or the controller sends the new values of the control action to the actuator when there is a significative change with respect to the previous value. The level crossing is considered the event that triggers the capture and the sending of a new sample.

Formally, a SOD sampler can be thought of as a block which has a continuous signal $u(t)$ as input and generates a sampled signal $u_{nl}(t)$ as output, which is a piecewise constant signal with $u_{nl}(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$. Each t_k is denoted as *event time*, and it holds $t_{k+1} = \inf\{t |\ t > t_k \wedge |u(t) - u(t_k)| \geq \delta\}$, where δ is the sampling threshold, i.e., the minimum change that triggers the taking of a new sample. The previous condition holds for all t_k, except for t_0, which is assumed to be the time instant when the block is initialized as $u_{nl}(t_0) = u(t_0)$.

In addition, this signal can be saturated due to the limitations in the sensors or in the actuators. As said before, the pattern resulting from sampling a signal with a SOD sampler can be described as a nonlinearity of N_l levels with hysteresis. It can be characterized as

$$\begin{aligned} &\text{if } u_{nl}(t^-) = (i+\alpha)\delta, \\ &u_{nl}(t) = \begin{cases} (i+\alpha+1)\delta & \text{if } u(t) \geq (i+\alpha+1)\delta \\ (i+\alpha)\delta & \text{if } u(t) \in ((i+\alpha-1)\delta, (i+\alpha+1)\delta) \\ (i+\alpha-1)\delta & \text{if } u(t) \leq (i+\alpha-1)\delta \end{cases}, \end{aligned} \tag{2.1}$$

where $\alpha \in [0, 1)$ is the offset with respect to the origin, $i \in \mathbb{Z}$ if the sampler is without saturation, and $i \in [i_{min}, i_{max}]$ when the sampler is with saturation.

Depending on the initial value, the nonlinearity introduced could have an offset with respect to the origin, $\alpha = u(t_0) - i\delta$, where $i = \lfloor u(t_0)/\delta \rfloor$. The level crossing sampling with offset is formally defined in the following paragraphs.

Definition 2.1 Let $T = \{t_0, \ldots, t_{N_l}\}$, with $t_k \in \mathbb{R}$ and $t_{k-1} < t_k$, be a set of sampling times, and $u = \{u(t_0), \ldots, u(t_{N_l})\}$ a set of samples. Thus, u is a level crossing sampling of $u(t)$ if, and only if, $|u(t_k) - u(t_{k-1})| = \delta$, $k = 0, 1, \ldots, N_l$.

Note that every sample can be expressed as $y_s(t_k) = (i_k + \alpha)\delta$, where $i_k \in \mathbb{Z}$ is the crossed level by the input signal $y(t)$, and $\alpha \in [0, 1) \subset \mathbb{R}$ is the sampling offset which depends on the initial sample, $y(t_0)$.

Definition 2.2 The order of a nonlinearity with hysteresis is defined by subtracting 1 to the number of crossing levels.

For example, a two-level nonlinearity with hysteresis and zero offset owns three crossing levels, that is, $-\delta$, 0, and δ. It must be noticed that if the nonlinearity had zero offset, the number of levels would be always even, and odd in the opposite case.

2.2.2 The Linear Blocks: The Process and the Controller

As Fig. 2.3 shows, the linear dynamics of the process and the PI controller can be joined in two different ways by placing the nonlinear block at the process output (Fig. 2.3a) or at the controller output (Fig. 2.3b). Depending on the combination, two different linear blocks G_{ps} and G_{cs} are obtained and, once the loop is closed with the sampler, two PLS are produced with different limit cycles features. The dynamics of the G_{ps} and G_{cs} blocks will be represented by the augmented state matrices obtained by combining process and controller.

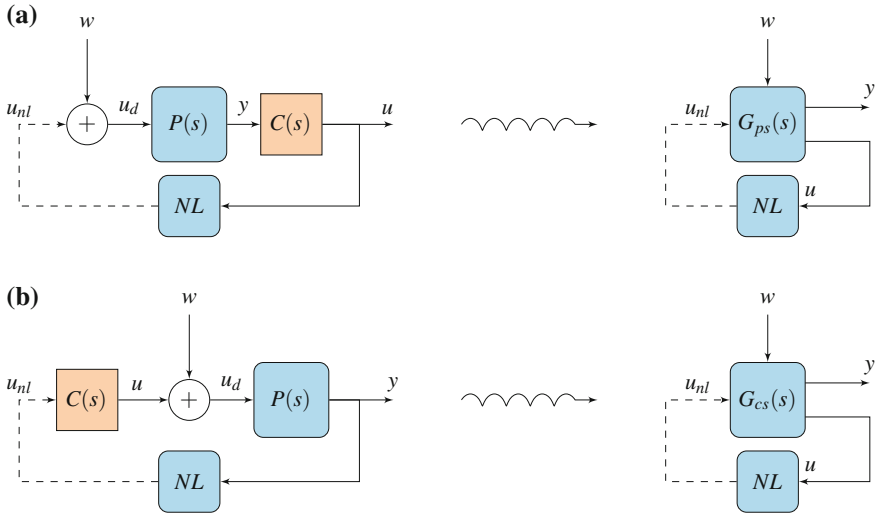

Fig. 2.3 Event-based control schemes. The block diagrams on the *left* correspond to the two proposed configurations, **a** the event-based sampler at the process output and **b** at the controller output. On the *right*, the continuous dynamics have been grouped in one block to simplify the analysis. *Solid lines* represent a continuous data flow, while *dotted lines* mean a discontinuous data flow

2.2.2.1 State-Space Representation of the Controller

Although there are different implementations of the PID algorithm in literature, all of them are essentially equivalent. In this work, the parallel form of the PID algorithm is considered (see Fig. 2.4). It can be represented by the transfer function:

$$u(s) = \left(k_p + \frac{k_i}{s} + k_d s\right) e_{sp}(s), \tag{2.2}$$

where $e_{sp} = y_{sp} - y$ is the control error.

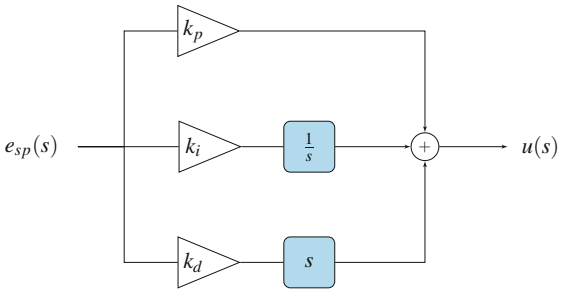

Fig. 2.4 Block diagram of the PID controller in the parallel form

The PID transfer function has a high gain for high frequencies, due to the derivative term. To avoid problems with noisy signals, in practical implementations, it is common to filter the derivative with a first-order filter. With this consideration, the resulting transfer function is

$$u(s) = \left(k_p + \frac{k_i}{s} + \frac{k_d s}{1 + \frac{k_d}{kN} s} \right) e_{sp}(s). \tag{2.3}$$

Let us start obtaining the matrices corresponding to the continuous case, that is, without considering the SOD sampler. No delays are considered now. Assume that the process $P(s)$ and the controller $C(s)$ are described by the time-invariant state-space systems:

$$\begin{aligned} P(s) &\sim \begin{cases} \dot{x}_p(t) = A_p x_p(t) + B_p(u(t) + w(t)) \\ y(t) = C_p x_p(t) \end{cases} \\ C(s) &\sim \begin{cases} \dot{x}_c(t) = A_c x_c(t) + B_c e_{sp}(t) \\ u(t) = C_c x_c(t) + D_c e_{sp}(t), \end{cases} \end{aligned} \tag{2.4}$$

where $x_p \in \mathbb{R}^n$ is the state of the process, $x_c \in \mathbb{R}^q$ is the state of the controller, and A_p is non-singular.

Combining the two parts of (2.4), the whole system is

$$\begin{aligned} \begin{pmatrix} \dot{x}_p(t) \\ \dot{x}_c(t) \end{pmatrix} &= \begin{pmatrix} A_p - B_p D_c C_p & B_p C_c \\ -B_c C_p & A_c \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} + \begin{pmatrix} B_p D_c & B_p \\ B_c & 0 \end{pmatrix} \begin{pmatrix} y_{sp} \\ w \end{pmatrix} \\ \begin{pmatrix} y(t) \\ u(t) \end{pmatrix} &= \begin{pmatrix} C_p & 0 \\ -D_c C_p & C_c \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ D_c & 0 \end{pmatrix} \begin{pmatrix} y_{sp} \\ w \end{pmatrix}. \end{aligned} \tag{2.5}$$

2.2.2.2 Sampling the Process Variable: The G_{ps} Block

The introduction of the SOD sampler in the PID control loop modifies the expressions presented in the previous section. Depending on where the sampler is placed, either the controller or the process input only changes at certain times.

To include the effect of the SOD sampler in (2.4), a new variable is introduced: u_{nl}, the nonlinear output of the sampler. At the sampling instants t_k, $u_{nl} := y(t_k)$ if the sampler is at the process variable, and $u_{nl} := u(t_k)$ in the opposite case. The expressions corresponding to the control loop with the SOD sampler are obtained introducing u_{nl} in (2.4), which yields

$$\begin{pmatrix} \dot{x}_p \\ \dot{x}_c \end{pmatrix} = \begin{pmatrix} A_p & B_p C_c \\ 0 & A_c \end{pmatrix} \begin{pmatrix} x_p \\ x_c \end{pmatrix} + \begin{pmatrix} -B_p D_c & B_p D_c & B_p \\ -B_c & B_c & 0 \end{pmatrix} \begin{pmatrix} u_{nl} \\ y_{sp} \\ w \end{pmatrix}$$

$$\begin{pmatrix} y \\ u \end{pmatrix} = \begin{pmatrix} C_p & 0 \\ 0 & C_c \end{pmatrix} \begin{pmatrix} x_p \\ x_c \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ -D_c & D_c & 0 \end{pmatrix} \begin{pmatrix} u_{nl} \\ y_{sp} \\ w \end{pmatrix}. \tag{2.6}$$

2.2.2.3 Sampling the Control Variable: The G_{cs} Block

The procedure to obtain the augmented state matrices of the G_{cs} block is similar to the previous case but taking into account that now the input to the controller $u(t)$ is the process output $y(t)$, and the input to the process is the signal resulting of adding the disturbance $w(t)$ to an input named $u_s(t)$. The resulting system $G_{cs}(s)$ is

$$\begin{aligned} \begin{pmatrix} \dot{x}_p \\ \dot{x}_c \end{pmatrix} &= \begin{pmatrix} A_p & 0 \\ -B_c C_p & A_c \end{pmatrix} \begin{pmatrix} x_p \\ x_c \end{pmatrix} + \begin{pmatrix} B_p & 0 & B_p \\ 0 & B_c & 0 \end{pmatrix} \begin{pmatrix} u_{nl} \\ y_{sp} \\ w \end{pmatrix} \\ \begin{pmatrix} y \\ u \end{pmatrix} &= \begin{pmatrix} C_p & 0 \\ -D_c C_p & C_c \end{pmatrix} \begin{pmatrix} x_p \\ x_c \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & D_c & 0 \end{pmatrix} \begin{pmatrix} u_{nl} \\ y_{sp} \\ w \end{pmatrix}. \end{aligned} \tag{2.7}$$

To simplify the analysis of the previous expressions, from now on, the set-point is assumed to be null, i.e., $y_{sp} = 0$. This is done without loss of generality, as shown in the following Proposition.

Proposition 2.1 *Consider the systems*

$$G_{ps}(s) \sim \begin{cases} \dot{x}_{ps} &= A_{ps} x_{ps} + B_{ps} u_{ps} \\ y_{ps} &= C_{ps} x_{ps} + D_{ps} u_{ps}, \end{cases} \tag{2.8}$$

where

$$A_{ps} = \begin{pmatrix} A_p & B_p C_c \\ 0 & A_c \end{pmatrix} \quad B_{ps} = \begin{pmatrix} -B_p D_c & B_p \\ -B_c & 0 \end{pmatrix}$$

$$C_{ps} = \begin{pmatrix} C_p & 0 \\ 0 & C_c \end{pmatrix} \quad D_{ps} = \begin{pmatrix} 0 & 0 \\ -D_c & 0 \end{pmatrix}, \tag{2.9}$$

and

$$G_{cs}(s) \sim \begin{cases} \dot{x}_{cs} &= A_{cs} x_{cs} + B_{cs} u_{cs} \\ y_{cs} &= C_{cs} x_{cs}, \end{cases} \tag{2.10}$$

where

$$A_{cs} = \begin{pmatrix} A_p & 0 \\ -B_c C_p & A_c \end{pmatrix} \quad B_{cs} = \begin{pmatrix} B_p & B_p \\ 0 & 0 \end{pmatrix}$$

$$C_{cs} = \begin{pmatrix} C_p & 0 \\ -D_c C_p & C_c \end{pmatrix}.$$

These systems are equivalent to (2.6) and (2.7). Therefore, the set-point can be assumed to be null without loss of generality.

Proof Consider that the sampler is placed at the process output. Note that defining $(\bar{x}_p, \bar{x}_c, \bar{u}, \bar{y}, \bar{u}_{nl}, \bar{w}) := (x_p, x_c, u, y, u_{nl} - y_{sp}, w)$, the system can be equivalently written as (2.6). Now, consider that the sampler is placed at the controller output. Defining $(\tilde{x}_p, \tilde{x}_c, \tilde{u}, \tilde{y}, \tilde{u}_{nl}, \tilde{w}) := (x_p - \frac{C_p^T}{\|C_p\|^2} y_{sp}, x_c, u, y - y_{sp}, u_{nl} - \frac{B_p^T A_p C_p^T}{\|B_p\|^2 \|C_p\|^2} y_{sp}, w)$, the system can be written equivalently as (2.7). In both cases, the introduced variables only differ from the original by a constant value. Therefore, it can be assumed without loss of generality that $y_{sp} = 0$. To make the notation more clear, from now on, the variables are written without the symbols $\tilde{}$ and $\bar{}$.

2.2.3 The P, I, PI, PD, and PID Controllers

In this section, the expressions (2.6) and (2.7) are particularized to the most frequently forms of the PID controller. Although the PI is considered in most of the examples, because it is the most extended controller, the propositions and algorithms presented in this chapter are in general form, so they can be applied to any of the enumerated cases by choosing the appropriate matrices.

2.2.3.1 Proportional Controller (P)

The proportional or P controller is a controller with a feedback based only in the current error of the process. Its associated state-space matrices are $A_c = B_c = C_c = 0$, and $D_c = k_p$. Thus,

$$\begin{aligned} A_{ps} &= \begin{pmatrix} A_p & 0 \\ 0 & 0 \end{pmatrix} \quad B_{ps} = \begin{pmatrix} -k_p B_p & B_p \\ 0 & 0 \end{pmatrix} \\ C_{ps} &= \begin{pmatrix} C_p & 0 \\ 0 & 0 \end{pmatrix} \quad D_{ps} = \begin{pmatrix} 0 & 0 \\ -k_p & 0 \end{pmatrix}, \end{aligned} \tag{2.11}$$

and

$$\begin{aligned} A_{cs} &= \begin{pmatrix} A_p & 0 \\ 0 & 0 \end{pmatrix} \qquad B_{cs} = \begin{pmatrix} B_p & B_p \\ 0 & 0 \end{pmatrix} \\ C_{cs} &= \begin{pmatrix} C_p & 0 \\ -k_p C_p & 0 \end{pmatrix}. \end{aligned} \tag{2.12}$$

2.2.3.2 Integral Controller (I)

The transfer function of the integrator or I controller is $C(s) = \frac{k_i}{s} e_{sp}(s)$, therefore one of its possible representations in the state space is given by $A_c = 0$, $B_c = 1$, $C_c = k_i$, and $D_c = 0$. Thus,

$$\begin{aligned} A_{ps} &= \begin{pmatrix} A_p & k_i B_p \\ 0 & 0 \end{pmatrix} \quad B_{ps} = \begin{pmatrix} 0 & B_p \\ -1 & 0 \end{pmatrix} \\ C_{ps} &= \begin{pmatrix} C_p & 0 \\ 0 & k_i \end{pmatrix}, \end{aligned} \tag{2.13}$$

and

$$\begin{aligned} A_{cs} &= \begin{pmatrix} A_p & 0 \\ -C_p & 0 \end{pmatrix} \quad B_{cs} = \begin{pmatrix} B_p & B_p \\ 0 & 0 \end{pmatrix} \\ C_{cs} &= \begin{pmatrix} C_p & 0 \\ 0 & k_i \end{pmatrix}. \end{aligned} \tag{2.14}$$

2.2.3.3 Proportional-Integral Controller (PI)

The PI controller state-space matrices are $A_c = 0$, $B_c = 1$, $C_c = k_i$, and $D_c = k_p$. Thus,

$$\begin{aligned} A_{ps} &= \begin{pmatrix} A_p & k_i B_p \\ 0 & 0 \end{pmatrix} \quad B_{ps} = \begin{pmatrix} -k_p B_p & B_p \\ -1 & 0 \end{pmatrix} \\ C_{ps} &= \begin{pmatrix} C_p & 0 \\ 0 & k_i \end{pmatrix} \qquad D_{ps} = \begin{pmatrix} 0 & 0 \\ -k_p & 0 \end{pmatrix}, \end{aligned} \tag{2.15}$$

and, when the sampler is at the control variable,

$$\begin{aligned} A_{cs} &= \begin{pmatrix} A_p & 0 \\ -C_p & 0 \end{pmatrix} \qquad B_{cs} = \begin{pmatrix} B_p & B_p \\ 0 & 0 \end{pmatrix} \\ C_{cs} &= \begin{pmatrix} C_p & 0 \\ -k_p C_p & k_i \end{pmatrix}. \end{aligned} \tag{2.16}$$

2.2.3.4 Proportional-Derivative Controller (PD)

The PD controller is $A_c = -N\frac{k_p}{k_d}$, $B_c = N\frac{k_p}{k_d}$, $C_c = k_d$, and $D_c = (1+N)k_p$. Thus,

$$\begin{aligned} A_{ps} &= \begin{pmatrix} A_p & k_d B_p \\ 0 & -N\frac{k_p}{k_d} \end{pmatrix} \quad B_{ps} = \begin{pmatrix} -(1+N)k_p B_p & B_p \\ -N\frac{k_p}{k_d} & 0 \end{pmatrix} \\ C_{ps} &= \begin{pmatrix} C_p & 0 \\ 0 & k_d \end{pmatrix} \qquad D_{ps} = \begin{pmatrix} 0 & 0 \\ -(1+N)k_p & 0 \end{pmatrix}, \end{aligned} \tag{2.17}$$

and, when the sampler is at the control variable, the expressions are

$$\begin{aligned} A_{cs} &= \begin{pmatrix} A_p & 0 \\ -N\frac{k_p}{k_d}C_p & A_c \end{pmatrix} \qquad B_{cs} = \begin{pmatrix} B_p & B_p \\ 0 & 0 \end{pmatrix} \\ C_{cs} &= \begin{pmatrix} C_p & 0 \\ -(1+N)k_pC_p & k_d \end{pmatrix}. \end{aligned} \tag{2.18}$$

2.2.3.5 PID with Derivative Filter

The PID controller with derivative filter has $A_c = \begin{pmatrix} 0 & 0 \\ 0 & -N\frac{k_p}{k_d} \end{pmatrix}$, $B_c = \begin{pmatrix} 1 \\ N\frac{k_p}{k_d} \end{pmatrix}$, $C_c = \begin{pmatrix} k_i & k_d \end{pmatrix}$, and $D_c = (1+N)k_p$. Thus,

$$\begin{aligned} A_{ps} &= \begin{pmatrix} A_p & B_pk_i & B_pk_d \\ 0 & 0 & 0 \\ 0 & 0 & -N\frac{k_p}{k_d} \end{pmatrix} \quad B_{ps} = \begin{pmatrix} -B_p(1+N)k_p & B_p \\ -1 & 0 \\ -N\frac{k_p}{k_d} & 0 \end{pmatrix} \\ C_{ps} &= \begin{pmatrix} C_p & 0 & 0 \\ 0 & k_i & k_d \end{pmatrix} \qquad D_{ps} = \begin{pmatrix} 0 & 0 \\ -(1+N)k_p & 0 \end{pmatrix}, \end{aligned} \tag{2.19}$$

where $A_p \in \mathbb{R}^{n\times n}$, $A_{ps} \in \mathbb{R}^{n_s\times n_s}$, $B_{ps} \in \mathbb{R}^{n_s\times 2}$, $C_{ps} \in \mathbb{R}^{2\times n_s}$, and $B_{ps} \in \mathbb{R}^{2\times 2}$, with $n_s = n+2$.

When the sampler is at the control variable, the expressions are

$$\begin{aligned} A_{cs} &= \begin{pmatrix} A_p & 0 & 0 \\ -C_p & 0 & 0 \\ -N\frac{k_p}{k_d}C_p & 0 & -N\frac{k_p}{k_d} \end{pmatrix} \qquad B_{cs} = \begin{pmatrix} B_p & B_p \\ 0 & 0 \end{pmatrix} \\ C_{cs} &= \begin{pmatrix} C_p & 0 & 0 \\ -(1+N)k_pC_p & k_i & k_d \end{pmatrix}. \end{aligned} \tag{2.20}$$

where $A_{cs} \in \mathbb{R}^{n_s\times n_s}$, $B_{cs} \in \mathbb{R}^{n_s\times 2}$, and $C_{cs} \in \mathbb{R}^{2\times n_s}$.

2.3 Defining an Event-Based System as a PLS

The results obtained in this section do not depend on where the sampler is placed. Therefore, the subindexes representing the position of the sampler, *ps* and *cs*, have been dropped to simplify the notation. For example, A is equivalent to A_{ps}, if the process variable is being sample, or to A_{cs} in the other case.

Let us consider that the input to $G(s)$ is delayed in time by τ and the loop is closed by adding the nonlinearity represented by the SOD sampler. Then,

$$G(s) \sim \begin{cases} \dot{x}(t) &= Ax(t) + Bu(t-\tau) \\ y(t) &= Cx(t) + Du(t-\tau). \end{cases} \tag{2.21}$$

Now, the resulting system is an infinite-dimensional system because of the time delay. However, it can be simplified because closing the loop with the nonlinearity makes the input signal piecewise constant. So, in the matrix u, the input $u_{nl}(t)$ is redefined as

$$u_{nl_k} = (j_k + \alpha)\delta,$$

where $j_k \in \mathbb{Z}$ is the crossed level by the input signal $u_{nl}(t)$, and $\alpha \in [0, 1) \in \mathbb{R}$ is the sampling offset which depends on the initial sample, $u_{nl}(t_0)$. Let us define the switching times as follows:

Definition 2.3 Consider a limit cycle composed of N_l switchings, and assume $T = \{t_0, \ldots, t_{N_l}\}$ are the sampling times, as in Definition 2.1. Then the switching times are defined as $t_i^* = t_i - t_{i-i}$ (see Fig. 2.5).

And the order of the limit cycle is defined as follows:

Definition 2.4 Consider a limit cycle in a symmetric nonlinearity of order N_l (N_l+1 crossing levels, as in Definition 2.2). Then, the number of switchings of the limit cycle is $2N_l$.

Then, if limit cycles of period T with switching times, $t_i^* > \tau$ where $T = t_1^* + t_2^* + \cdots + t_{2N_l}^*$, are only considered, the input $\left\{u_{nl}(t), t_i^* - \tau < t \leq t_i^*\right\}$ is

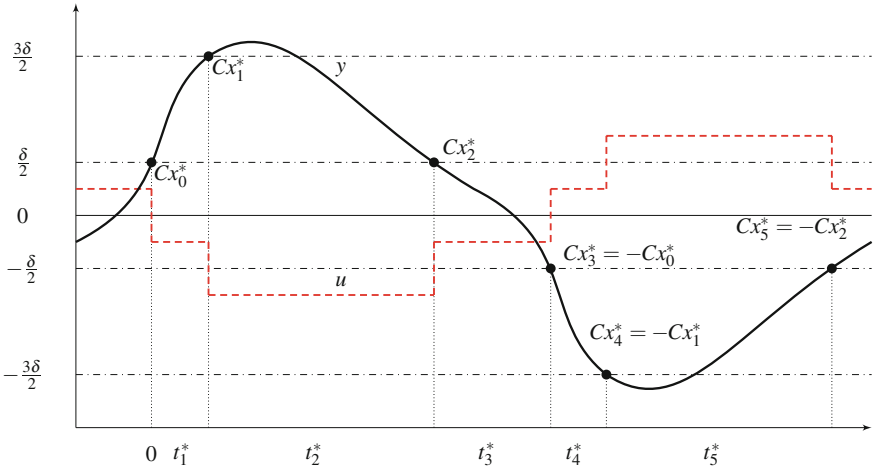

Fig. 2.5 Trajectory of a solution of a PLS system corresponding to a limit cycle in a system with a symmetric send-on-delta sampling scheme, with sampling threshold δ, and sampling offset $\alpha = 0.5$. The *solid line* represents the process output, and the *dashed line* the control input. The process output y crosses the sampling levels at times t_i, generating new updates. Each time interval is denoted by $t_i^* = t_i - t_{i-1}$. In this particular case, the control input u is constant between consecutive crossings, $|Cx_{i+1}^* - Cx_i^*| = \delta$, and each x_i^* is the system state at t_i

constant and its value is given by the feedback. In this case, the state of the system at the sampling times $t_1, t_2, \ldots, t_{2N_l}$ is uniquely given by $x(t_i)$. Taking into account this consideration, the system can be considered as a PLS defined by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_i \\ y(t) &= Cx(t) + D_i, \end{aligned} \tag{2.22}$$

where $B_i = B\left(u_{nl_i}\ w\right)^T$, $D_i = D\left(u_{nl_i}\ w\right)^T$, and $u_{nl_i} = (j_i + \alpha)\delta$, for $i = \{0, \pm 1, \ldots, \pm N_l\}$.

In this PLS, the rule to switch between the linear systems is included in the definitions of the nonlinearities. It must be noted that the switching rules have memory and the decision of which LTI to use may not depend only on the actual values of the state, but also on their past values. In the state-space system, the points in which a rule provokes the switching from the system i to the system j define a surface which is known as *switching surface* (see Fig. 2.6). These surfaces consist of hyperplanes of dimension $n - 1$, being n the order of the system, i.e., $x \in \mathbb{R}^n$,

$$S_i = \{x \,|\, Cx - u_{nl_i} = 0\} \text{ for } i = \{0, \pm 1, \ldots, \pm N_l\}.$$

It is interesting to characterize the limit cycles that can appear in the system due to the effect of the nonlinearities introduced with the event-based sampling scheme.

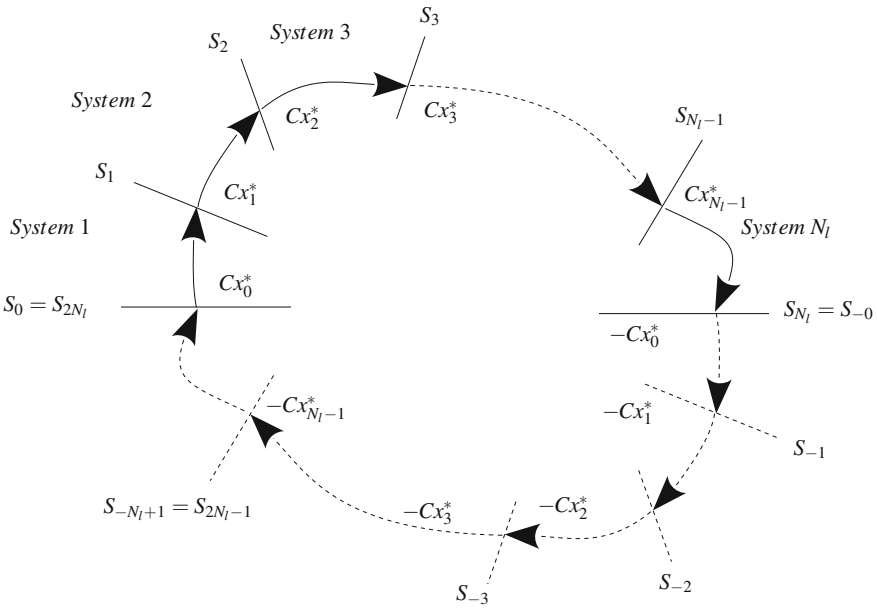

Fig. 2.6 Trajectory of a solution of a PLS system. In each region, the dynamics of the system is determined by a different LTI system defined by (2.22). The *lines* represent the switching surfaces

The study of these limit cycles can be interesting for several reasons. There is a wide range of processes that will almost surely present limit cycles with the studied control schemes, while for other processes, they can be prevented by carefully choosing the controller parameters. In any case, limit cycles mean oscillations, which, depending on the process, may be more or less problematic. For example, a high frequency of oscillation may wear out the actuators. Also, the study of limit cycles can be used for identification purposes. For example, the relay autotuning method [1] is based on the properties of the limit cycle that appears in a process subject to relay feedback (which can be considered as a particular case of the studied event-based scheme). In addition, for the cases when the limit cycles cannot be prevented, it may be important to know about the stability of these limit cycles.

In order to calculate the period and amplitude, first it is assumed that the limit cycle contains $N_l + 1$ levels, and thus it is composed of $2N_l$ switchings, where the first N_l correspond to the positive level crossings, and the rest N_l are the negative ones. Next, it is presented the set of equations that allows us to find the switching times and the values of the states at these switching times. The result stated in the following proposition is a generalization to N_l levels of the approaches in [33, 81, 216].

Proposition 2.2 *Consider the PLS in (2.22), with a nonlinearity defined by the switching surfaces* $S_i = \{x \mid Cx - (j_i + \alpha)\delta = 0\}$, *where* $\alpha \in [0, 1)$, $j_i \in \mathbb{Z}$, $i \in \{0, \pm 1, \pm N_l\}$, *and* $0 < \delta \in \mathbb{R}$. *Assume that there exists a symmetric periodic solution* γ *with* $2n$ *switching surfaces per period* $T = t_1^* + t_2^* + \cdots + t_{2N_l}^*$, *where* $t_1^*, t_2^*, \ldots, t_{2N_l}^*$ *are the switching times when the switching surfaces* $S_1, \ldots, S_{N_l}, S_{-1}, \ldots, S_{-N_l}$ *are crossed, respectively (Fig. 2.6). Define*

$$f_k(t_1^*, \ldots, t_{2N_l}^*) = C(I + e^{AT})^{-1} \left[\sum_{i=1}^{2N_l - 1} \Phi_{2N_l - 1} \cdots \Phi_{i+1} (\Phi_i - I) \Lambda_i \right] - E_k, \tag{2.23}$$

where $\Phi_i = \Phi(t_i) = \mathrm{e}^{At_i}$, *and* $\Lambda_i = A^{-1} B_i$. *Then the following conditions hold*

$$\begin{cases} f_1(t_1^*, t_2^*, \ldots, t_{2N_l}^*) = 0 \\ f_2(t_1^*, t_2^*, \ldots, t_{2N_l}^*) = 0 \\ \qquad \vdots \\ f_{2N_l}(t_1^*, t_2^*, \ldots, t_{2N_l}^*) = 0, \end{cases} \tag{2.24}$$

and

$$\begin{cases} E_i \leq y(t) = Cx_i(t) < E_{i+1} \ \textit{for} \ 0 \leq t < t_i^* \quad i = 1 \ldots N_l - 1 \\ y(t) = Cx_{N_l}(t) \geq E_{N_l+1} \quad \textit{for} \ 0 \leq t < t_{N_l}^* \\ E_i \geq y(t) = Cx_i(t) > E_{i+1} \ \textit{for} \ 0 \leq t < t_i^* \ \ i = N_l + 1 \ldots 2N_l, \end{cases} \tag{2.25}$$

where

$$E_i = (j_i + \alpha)\delta, \ \textit{and} \ x_i(t) = e^{At} x_{i-1}^* - A^{-1}(e^{At} - I) B_i.$$

Furthermore, the limit cycle can be obtained with the initial condition

$$x_0^* = (I + e^{AT})^{-1} \left[\sum_{i=1}^{2N_l-1} \Phi_{2N_l-1} \cdots \Phi_{i+1} \left(\Phi_i - I \right) \Lambda_i \right]. \tag{2.26}$$

Proof Assuming that $t_i > \tau$, where t_i is the time elapsed between the crossing of two consecutive switching surfaces, for example i and $i+1$, then the state is obtained by integrating (2.22) from $t = 0$ to $t = t_i$. It gives

$$x_{i+1} = \Phi(t_i)x_i + \Gamma_1(t_i)u_{i-1} + \Gamma_0(t_i)u_i, \tag{2.27}$$

where $\Phi(t) = \mathrm{e}^{At}$ is the state transition matrix, $\Gamma_0(t) = \int_0^{t-\tau} \Phi(s)ds$ accounts for the effect of the input, and $\Gamma_1(t) = \int_{t-\tau}^{t} \Phi(s)ds$ is a term which represents the effect of the input because of the time delay of the system.

Then, for a limit cycle involving N_l switching times, we have a system of equations described by

$$\begin{aligned} \Phi(t_1)x_1 + \Gamma_1(t_1)u_{N_l} + \Gamma_0(t_1)u_1 - x_2 &= 0 \\ &\cdots \\ \Phi(t_{N_l-1})x_{N_l-1} + \Gamma_1(t_{N_l-1})u_{N_l-2} + \Gamma_0(t_{N_l-1})u_{N_l-1} - x_{N_l} &= 0 \\ \Phi(t_{N_l})x_{N_l} + \Gamma_1(t_{N_l})u_{N_l-1} + \Gamma_0(t_{N_l})u_{N_l} - x_1 &= 0. \end{aligned} \tag{2.28}$$

Substituting recursively, we get the following expression:

$$\begin{aligned} [I - \Phi(t_{N_l}) \ldots \Phi(t_1)]x_n = {} & \Phi(t_{N_l}) \ldots \Phi(t_2)(\Gamma_1(t_1)u_{N_l} + \Gamma_0(t_1)u_1) \\ & + \Phi(t_{N_l}) \ldots \Phi(t_3)(\Gamma_1(t_2)u_1 + \Gamma_0(t_2)u_2) + \cdots \\ & + \Phi(t_{N_l})(\Gamma_1(t_{N_l-1})u_{N_l-2} + \Gamma_0(t_{N_l-1})u_{N_l-1}) \\ & + \Gamma_1(t_{N_l})u_{N_l-1} + \Gamma_0(t_{N_l})u_{N_l}. \end{aligned} \tag{2.29}$$

The previous expression can be written in compact form as

$$[I - \Phi_{N_l} \ldots \Phi_1]x_{N_l} = \sum_{i=1}^{2N_l-1} \Phi_{2N_l-1} \ldots \Phi_{i+1}[\Gamma_1(t_i)u_{i-1} + \Gamma_0(t_i)u_i], \tag{2.30}$$

and

$$x_{N_l} = [I - \Phi_{N_l} \ldots \Phi_1]^{-1} \sum_{i=1}^{2N_l-1} \left(\prod_{j=1}^{2N_l-1-i} \Phi_{2N_l-j} \right) [\Gamma_1(t_i)u_{i-1} + \Gamma_0(t_i)u_i]. \tag{2.31}$$

If we assume that the system matrix A is non-singular, then the integrals Γ_1 and Γ_0 can be explicitly computed and the state x_{N_l} can be solved, yielding the expression

of the initial state (2.26). Since we know that at the switching times the state must be at the switching surface, we can combine the previous expression with the switching conditions to get the set of equations given by (2.24). Finally, the conditions (2.25) hold because the state does not cross the switching surface in the interval between two switchings.

However, if the system matrix is assumed to be singular, neither the state nor the integrals Γ_0 and Γ_1 can be computed explicitly. In this case, the system of equations is obtained in the same way, but the computations are more involved. First the functions in (2.24) are redefined as

$$f_i(x_i^*, t_1^*, \ldots, t_{2N_l}^*) = [I - \Phi_{2N_l} \ldots \Phi_1]x_i^* - \sum_{i=1}^{2N_l-1} \Phi_{2N_l-1} \ldots \Phi_{i+1}[\Gamma_1(t_i^*)u_{i-1} + \Gamma_0(t_i^*)u_i], \tag{2.32}$$

where, in contrast to the previous case, x_i^* appear as unknowns.

Note that the system of equations given by the functions f_i is composed of N_l^2 scalar equations and $N_l^2 + N_l$ unknowns. Thus, in order to solve it, the system must be completed with N_l additional equations. These equations are obtained from the switching conditions in the sampler, which fix the values of either the process output or the control input at the event times, depending on where the sampling is placed. Then, the complete set of equations that must be solved to obtain the features of the limit cycle is

$$\begin{cases} f_1(x_1^*, t_1^*, \ldots, t_{2N_l}^*) = 0 \\ \vdots \\ f_{2N_l}(x_{2N_l}^*, t_1^*, \ldots, t_{2N_l}^*) = 0 \\ Cx_1^* - d_1 = 0 \\ \vdots \\ Cx_{2N_l}^* - d_{2N_l} = 0. \end{cases} \tag{2.33}$$

How to use this result to analyze the limit cycles is demonstrated with examples in Sects. 2.4 and 2.5.

2.3.1 Local Stability

The local stability of the limit cycles described in the previous paragraphs can be analyzed by observing the system at the switching times. The following result, which is a generalization of the approaches in [33, 81, 216], can be applied to study the behavior of the trajectories of the system in the proximities of the limit cycles.

Proposition 2.3 *Assume that there exists a limit cycle γ with k states in the system (2.22). The limit cycle is locally stable if and only if $W = W_k W_{k-1} \ldots W_1$ has all*

its eigenvalues inside the unit circle, where $W_i = (I - \frac{V_i C_i}{C_i V_i})e^{At_i^*}$, $V_i = Ax_i^* + B_i$, t_i^* *are the switching times, and* x_i^* *the state at the switching times.*

Proof Consider a trajectory with initial condition $x(0) = x_0^*$. In the time interval before the first switching, this trajectory is defined as $x(t) = \Phi(t)x_0^* + \Gamma_1(t)u_{nl_{N_l-1}} + \Gamma_0(t)u_{nl_{N_l}}$. When x reaches the switching surface at time $t_1^* + \delta_1 t_1^*$, we have

$$x(t_1^* + \delta_1 t_1^*) = \Phi(t_1^* + \delta_1 t_1^*)(x_0^* + \delta_1 x_0^*) + \Gamma_1(t_1^* + \delta_1 t_1^*)u_{nl_{N_l-1}} + \Gamma_0(t_1^* + \delta_1 t_1^*)u_{nl_{N_l}}.$$

The series expansion in $\delta_1 t_1^*$ and $\delta_1 x_0^*$ is

$$x(t_1^* + \delta_1 t_1^*) = x_1^* + v_1 \delta_1 t_1^* + \Phi(t_1^*)\delta_1 x_0^* + O(\delta_1^2),$$

where $v_1 = Ax_1^* + B_1 = A[\Phi(t_1^*)x_0^* + \Gamma_1(t_1^*)u_{nl_{N_l-1}} + \Gamma_0(t_1^*)u_{nl_{N_l}}] + Bu_{nl_{N_l}}$.

Since the solution is on the switching surface at $t_1^* + \delta_1 t_1^*$, we have

$$C_1 x(t_1^* + \delta_1 t_1^*) + d_1 = C_1 x_1^* + C_1 v_1 \delta_1 t_1^* + C_1 \Phi(t_1^*)\delta_1 x_0^* + d_1 = 0,$$

and, therefore, the following equality holds:

$$\delta_1 t_1^* = -\frac{C_1 \Phi(t_1^*)}{C_1 v_1}\delta_1 x_0^*.$$

The rest of the proof follows as in [81]. The state after the first switch is

$$x(t_1^* + \delta_1 t_1^*) = x_1^* + \left(I - \frac{v_1 C_1}{C_1 v_1}\right)\Phi(t_1^*)\delta_1 x_0 + O(\delta_1^2) = x_1^* + W_1 \delta_1 x_0^* + O(\delta_1^2). \tag{2.34}$$

Taking as initial condition $x_1^* + \delta_2 x_1^*$ and neglecting the $O(\delta^2)$ term, we have $x(t_2^* + \delta_2 t_2^*) = x_2^* + W_2 \delta_2 x_1^*$. Combining with (2.34) yields $\delta_2 x_1^* = W_1 \delta_1 x_0^*$. Replacing in the previous expression and applying successively for k eventually lead to

$$x(t_k^* + \delta_k t_k^*) = x_0^* + W_k W_{k-1} \ldots W_1 \delta_1 x_0^* + O(\delta_1^2). \tag{2.35}$$

Neglecting the $O(\delta_1^2)$ term, the dynamics of Eq. (2.35), is stable if and only if the eigenvalues of $W = W_k W_{k-1} \ldots W_1$ are inside the unit circle. This proves the proposition.

2.4 Analysis of the Limit Cycles

In the following sections, the expressions which take into account the effects of the event-based sampling at the output of the process and controller are presented. The method used is based on the grouping of all the continuous dynamics into one block

to obtain the state-space matrix (as shown in Fig. 2.3), and then to study the effect of the nonlinear feedback introduced by the sampling block.

To easily distinguish between the different types of controller and sampling, the following naming convention is used: *controller*-SOD$_n$-*process*, when the sampler is after the controller output, and *controller-process*-SOD$_n$ if the sampler is placed after the process output, where *process* corresponds to the type of process considered (IPTD, SOPDT,…) and *controller* refers to the type of controller (PI, PD, PID,…) The index n refers to the number of hysteresis bands presented in the sampler. For example, PI-SOD$_1$-IPTD denotes the system composed by an IPTD process controlled by a PI controller with the sampler placed after the controller output, and a limit cycle with one hysteresis band (i.e., a system with relay feedback).

There are two directions in which the complexity of the analysis can be increased. The first one is to consider that the order of the process is increasing, i.e., a simple integrator, a double integrator, etc., and the second one is to consider an increase in the number of hysteresis bands of the sampler.

To simplify the analysis, it is worth noting that the solutions of (2.33) are linear on δ, thus the state and control signal can be normalized dividing by δ (note that $\delta > 0$).

2.4.1 Equilibrium Points

Consider the system (2.22). The set of equilibrium points is defined as $\mathscr{X} = \{x^* | \dot{x}^* = 0\}$. An equilibrium point is one in which the derivatives of the states are null, i.e., $x^* | \dot{x}^* = 0$. Since the derivatives are null, all the trajectories which enter into it at t_0 will stay for $t > t_0$. An immediate necessary condition to have an equilibrium point is that the linear system $Ax + B_i = 0$ has at least one solution. Note that, except for the P and PD controller, it is easy to see that $det(A) = 0$, due to the integrator added by the controller, thus being possible to have a system of equations which is either indetermined or incompatible. The system does not have any solution if $rg(A) < rg(A|B_i)$, so an equilibrium point can exist only if $\exists i \in \mathbb{R} \mid rg(A) = rg(A|B_i)$.

Now assume that the output is within the k band of hysteresis, i.e., $x \in \Omega_k = \{x | \delta_k \leq y(t) = Cx(t) < \delta_{k+1}\}$ for some time interval $t_k \leq t < t_{k+1}$.

Proposition 2.4 *A necessary condition for the system to be ultimately bounded to Ω_k is that either $C[Ax + B_k] = 0$ or $C[Ax + B_{k+1}] = 0$.*

Proof Assume that $Cx(t)$ enters into Ω_k at t_0. After a time $t > \tau$, the derivative of the system is $C\dot{x}(t) = C[Ax(t) + B_i]$, for $i \in \{k, k+1\}$. Thus, if $C[Ax(t) + B_i] \neq 0$ for both $i = k$ and $i = k+1$, the process output will eventually cross the boundaries of Ω_k for some t.

Computing the equilibrium points of the PI control with SOD sampling at the process output (the set-point is assumed to be null) yields

$$\begin{pmatrix} \dot{x_p} \\ \dot{x_c} \end{pmatrix} = \begin{pmatrix} A_p & k_i B_p \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_p \\ x_c \end{pmatrix} + \begin{pmatrix} -k_p B_p & B_p \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \delta_i \\ w \end{pmatrix} = 0. \tag{2.36}$$

Thus, a necessary condition for the existence of an equilibrium point, from (2.36), is $\exists i \in \mathbb{Z} | \delta_i = 0$, because otherwise the integrator derivative is a non-null constant. Note that, if the set-point is not assumed to be null, the condition still holds with a slight modification: $\exists i \in \mathbb{Z} | \delta_i = y_{sp}$, i.e., the set-point must be an exact multiple of δ. Furthermore, since A_p is a non-singular matrix, the computation of the equilibrium point is straightforward: $x_c = \frac{-w}{k_i}$, $x_p = 0 \in \mathbb{R}^n$.

If the sampler affects to the controller output, then the equilibrium equation is

$$\begin{pmatrix} \dot{x_p} \\ \dot{x_c} \end{pmatrix} = \begin{pmatrix} A_p & 0 \\ -C_p & 0 \end{pmatrix} \begin{pmatrix} x_p \\ x_c \end{pmatrix} + \begin{pmatrix} B_p & B_p \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \delta_i \\ w \end{pmatrix} = 0, \tag{2.37}$$

and the necessary condition to have an equilibrium point is $\exists i \in \mathbb{Z} | \delta_i = (1 - C_p A_p^{-1} B_p)^{-1} w$. If this expression holds, then the equilibrium point can be computed as $x_p = \frac{C_p^T}{\|C_p\|} \delta_i$, $x_c = (1 + k_p)\delta_i$.

The rest of this section presents an algorithm to compute the period and amplitude of a limit cycle in a generic process, and then shows examples of application to several common processes.

2.4.2 Algorithm

In this section, an algorithm to obtain computationally the period of a limit cycle and the intermediate switching times is outlined. Assume that $\alpha = 0.5$ and that the system presents a limit cycle in which the condition $y(t) = Cx(t) \in ((\alpha - N_l)\delta, (\alpha + N_l - 1)\delta)$ holds. The limit cycle is assumed to have only two changes in the sign of the derivative: one at the beginning of the first semiperiod and the other at the beginning of the second semiperiod. Thus, the limit cycle must have $4N_l - 2$ switchings. Because of the symmetry, the behavior of the limit cycle can be inferred by studying only the first semiperiod, thus reducing the complexity to $2N_l - 1$ levels. The algorithm, which can be implemented either in a symbolic or in a numerical computation tool, is as follows:

- Initialization:
  1. Set N_l as the number of levels crossed within the limit cycle.
  2. Fix the values of k_p, k_i, α, δ, w, τ, and the matrices A and B.
  3. Calculate $\Phi(t) = e^{At}$, $\Gamma_0(t) = \int_0^{t-\tau} e^{As} ds$, and $\Gamma_1(t) = \int_{t-\tau}^{t} e^{As} ds$.
- To calculate the period:
  1. For i from 1 to $2N_l$ repeat steps 2–3.
  2. If $i \in (1, N_l)$, set $j_i = i - \lfloor \frac{N_l}{2} \rfloor$, else $j_i = \lfloor \frac{N_l}{2} \rfloor + N_l - i$.

3. Set $u_{nl_i} := (j_i + \alpha)\delta$, and,
   - $x_{c_i} = -\frac{u_{nl_i} + k_p x_{p_i}}{k_i}$, if sampling the controller output, or,
   - $x_{p_i} = u_{nl_i}$, if sampling the process output.
4. Set $eq_i := -x_{i+1} + \Phi(t_i)x_i + \Gamma_1(t_i)u_j + \Gamma_0(t_i)u_i = 0$.
5. Solve the system of equations given by eq_i, with the unknowns t_i, and x_{p_i} or x_{c_i}.
6. $T = \sum_{i=1}^{2N_l} t_i$.

- To calculate the amplitude:

1. Set $j_{max} = j|(x_j > x_i, \forall i \neq j)$ and $j_{min} = j|(x_j < x_i, \forall i \neq j)$.
2. Find $t_{min} = \min(\tau, t|C\dot{x}_{j_{min}}(t) = 0)$, and $t_{max} = \min(\tau, t|C\dot{x}_{j_{max}}(t) = 0)$, corresponding to the minimum and maximum values of the output.
3. Compute the amplitude of the process output, $\Delta_p = C_p[x(t_{max}) - x(t_{min})]$, and the control input, $\Delta_c = C_c[x(t_{max}) - x(t_{min})]$.

2.4.3 Examples of Analysis

To illustrate the application of Proposition 2.2 and the use of the algorithm of Sect. 2.4.2, the analysis of several systems (IPTD, SOPTD, FOPTD, and SOPTD) is detailed in the following lines. The results are summarized in Table 2.1, at the end of the section.

2.4.3.1 IPTD Process

Let us consider an IPTD process $P(s) = \frac{k}{s}e^{-\tau s}$, which can be described by its state-space model in the canonical observable form as

$$\begin{aligned} \dot{x}(t) &= -ku(t-\tau) + kw(t) \\ y(t) &= x(t) = x_p(t), \end{aligned} \tag{2.38}$$

where k is the process gain, x is the state, u is the process input, and y is the process output. First it is presented the approach for the PI-IPTD-SOD$_n$ structure, and then for the PI-SOD$_n$-IPTD.

Process variable sampling (PI-IPTD-SOD$_1$) According to (2.6), the state feedback matrix corresponding to the system controlled by a PI with SOD sampling at the process output is

$$\begin{aligned} \begin{pmatrix} \dot{x}_p(t) \\ \dot{x}_c(t) \end{pmatrix} &= \begin{pmatrix} 0 & kk_i \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} + \begin{pmatrix} kk_p & k \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_{nl_k} \\ w \end{pmatrix} \\ y(t) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix}. \end{aligned} \tag{2.39}$$

Assuming there exists a stable limit cycle with two states, then the equations that allow us to obtain the amplitude and period of the oscillations are

$$\begin{gathered} \Phi(t_1)x_1 + \Gamma_1(t_1)u_2 + \Gamma_0(t_1)u_1 - x_2 = 0 \\ \Phi(t_2)x_2 + \Gamma_1(t_2)u_1 + \Gamma_0(t_2)u_2 - x_1 = 0 \\ x_{p_1} = -u_{nl_1} = \alpha\delta \\ x_{p_2} = -u_{nl_2} = (\alpha - 1)\delta, \end{gathered} \tag{2.40}$$

where $x_i = [x_{p_i}\ x_{c_i}]^T$, and $u_i = [u_{nl_i}\ w]^T$, are the state and input, respectively.

The matrices Φ, Γ_0, and Γ_1 can be calculated as

$$\Phi(t) = \begin{pmatrix} 1 & kk_i t \\ 0 & 1 \end{pmatrix} \tag{2.41}$$

$$\Gamma_0(t) = \begin{pmatrix} kk_p t - kk_p\tau + \frac{1}{2}kk_i t^2 - kk_i t\tau + \frac{1}{2}kk_i\tau^2 & k(t-\tau) \\ t - \tau & 0 \end{pmatrix}$$

$$\Gamma_1(t) = \begin{pmatrix} kk_p\tau + kk_i t\tau - \frac{1}{2}kk_i\tau^2 & k\tau \\ \tau & 0 \end{pmatrix}. \tag{2.42}$$

Introducing (2.41) in (2.40), and solving the resulting equations, the period of the limit cycle can be obtained (see Table 2.1). Looking at the expression of the period, it can be seen that the symmetry of the limit cycle, i.e., the difference between the two semiperiods t_1 and t_2, depends on the offset of the sampler α. In particular, for $\alpha = 0.5$, the two semiperiods have the same value. When $\alpha = 0$, t_2 vanishes, which can be interpreted as this limit cycle cannot exist. In this case, either the system will reach a steady-state or enter into a limit cycle with higher number of levels. With respect to the disturbance rejection, it can be seen that w does not affect the period. This is because it is rejected by the integrator, which changes its mean value to absorb the disturbance.

With the switching times t_1 and t_2, the amplitudes of the oscillations can be computed. It is necessary to find the maximum and minimum of the output, which correspond to the times when its first derivative is null, i.e., $\dot{x}_p(t) = kk_i x_c(t) + kk_p u_{nl_k} + kw = 0$. By solving the previous expression, the value of x_c and the times when the peaks are reached can be obtained, and so for this case it can be found an analytic expression for the amplitude of the process output, A_{po}, and of the control input, A_{ci} (see Table 2.1).

Controller variable sampling (PI-SOD$_1$-IPTD) When the sampler is placed at the controller output, the description of the system in the state-space model is given by expressions (2.7). Thus, for this process, it is obtained

$$\begin{pmatrix} \dot{x}_p(t) \\ \dot{x}_c(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ k & k \end{pmatrix} \begin{pmatrix} u_{nl_k} \\ w \end{pmatrix}$$

$$y(t) = \begin{pmatrix} k_p & k_i \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix}. \tag{2.43}$$

Assuming there exists a stable limit cycle with two states, then the equations that allow to obtain the amplitude and period of the oscillations are

$$\begin{aligned} \Phi(t_1)x_1 + \Gamma_1(t_1)u_2 + \Gamma_0(t_1)u_1 - x_2 &= 0 \\ \Phi(t_2)x_2 + \Gamma_1(t_2)u_1 + \Gamma_0(t_2)u_2 - x_1 &= 0 \\ k_p x_{p_1} + k_i x_{c_1} &= u_{nl_1} = \alpha\delta \\ k_p x_{p_2} + k_i x_{c_2} &= u_{nl_2} = (\alpha - 1)\delta. \end{aligned} \tag{2.44}$$

Here, the amplitude of the limit cycle can be computed directly, since the control input is piecewise constant and the switching times and values are known. The period of the limit cycle can be obtained by solving the system of Eqs. (2.44) (see Table 2.1).

As opposed to the process sampling, here the disturbance appears in the expression of the semiperiods, thus affecting to the symmetry of the limit cycle. It is possible to have oscillations where the process is changing slowly nearly all the time and then to have an abrupt change. Since in one semiperiod the control action is more aggressive, this decreases the margin of delay that can be added to the system without reaching the next sampling level.

To obtain the expression corresponding to the amplitude, and since the maximum and minimum values of the process output are reached at times $t_1 + \tau$ and $t_2 + \tau$, integrating (2.43) yields the desired result (see Table 2.1).

2.4.3.2 DIPTD Process

It is well known in classic control theory that the double integrator process cannot be stabilized with a continuous PI controller, due to the 90 Â phase lag of each integrator. It becomes then necessary to introduce the derivative action (PD controller) to compensate this lag. In the same way, either with the PI-SOD$_n$-DIPTD or PI-SOD$_n$-DIPTD, the system will oscillate with unbounded growing amplitudes. Although it is not in the scope of this work, it should be possible to stabilize this kind of process by a SOD-PD controller. In practice, the implementation of the derivative action must be carefully studied, because it can be problematic specially when the sampler is placed after the process output, since the estimation of the derivative may be poor.

2.4.3.3 FOPTD Process

The process considered in this section is a FOPTD process $P(s) = \frac{k}{Ts+1}e^{-\tau s}$, which is described in the state-space system by the following expressions:

$$\begin{aligned} \dot{x}(t) &= -\frac{1}{T}x(t) + \frac{k}{T}u(t-\tau) + w(t) \\ y(t) &= x(t) = x_p(t), \end{aligned} \tag{2.45}$$

where k is the process gain, x is the state, u is the process input, y is the process output, and w is an external disturbance.

Process variable sampling (PI-FOPTD-SOD$_1$) The expressions corresponding to the PI-FOPTD-SOD$_1$ are as follows:

$$\begin{pmatrix} \dot{x}_p(t) \\ \dot{x}_c(t) \end{pmatrix} = \begin{pmatrix} \frac{1}{T} & \frac{kk_i}{T} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} + \begin{pmatrix} \frac{kk_p}{T} & \frac{k}{T} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_{nl_k} \\ w \end{pmatrix}$$

$$y(t) = \begin{pmatrix} k_p & k_i \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix}. \tag{2.46}$$

From (2.46), the matrices Φ, Γ_0, and Γ_1 can be obtained as

$$\begin{aligned} \Phi(t) &= \begin{pmatrix} \mathrm{e}^{\frac{t}{T}} & kk_i(\mathrm{e}^{\frac{t}{T}} - 1) \\ 0 & 1 \end{pmatrix} \\ \Gamma_0(t) &= \begin{pmatrix} -k(k_p + k_i T)(\mathrm{e}^{\frac{t-\tau}{T}} - 1) + kk_i(t-\tau) & T(\mathrm{e}^{\frac{t-\tau}{T}} - 1) \\ t - \tau & 0 \end{pmatrix} \\ \Gamma_1(t) &= \begin{pmatrix} k(k_p + k_i T)(\mathrm{e}^{\frac{t-\tau}{T}} - \mathrm{e}^{\frac{t}{T}}) - kk_i\tau & -T(\mathrm{e}^{\frac{t-\tau}{T}} - \mathrm{e}^{\frac{t}{T}}) \\ \tau & 0 \end{pmatrix}. \end{aligned} \tag{2.47}$$

As it can be seen in the previous expressions, since the system of equations obtained for the FOPTD contains terms involving exponentials, it is not possible to find analytical solutions, as opposed to the case of IPTD processes. Instead of this, the solutions have to be found numerically. However, the algorithm proposed in Sect. 2.4.2 can be applied.

Controller variable sampling (PI-SOD$_1$-FOPTD) The expressions corresponding to the PI-SOD$_1$-FOPTD are the following:

$$\begin{pmatrix} \dot{x}_p(t) \\ \dot{x}_c(t) \end{pmatrix} = \begin{pmatrix} -\frac{1}{T} & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} + \begin{pmatrix} \frac{k}{T} & \frac{k}{T} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_{nl_k} \\ w \end{pmatrix}$$

$$u(t) = \begin{pmatrix} k_p & k_i \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix} \tag{2.48}$$

$$y(t) = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_p(t) \\ x_c(t) \end{pmatrix}. \tag{2.49}$$

As in the previous case, the solutions of the equations must be found by means of numerical tools.

2.4.3.4 SOPTD Process

The process considered in this section is a SOPTD $P(s) = \frac{k}{(\tau_1 s+1)(\tau_2 s+1)}\mathrm{e}^{-\tau s}$, which is described in the state-space system by the following expressions:

$$\ddot{x}(t) = -\frac{1}{\tau_1\tau_2}x - \frac{\tau_1+\tau_2}{\tau_1\tau_2}\dot{x} + \frac{k}{\tau_1\tau_2}u(t-\tau) + w(t)$$
$$y(t) = x(t) = x_p(t), \tag{2.50}$$

where k is the process gain, τ_1 and τ_2 are the time constants, x is the state, u is the process input, y is the process output, and w is an external disturbance.

Process variable sampling (PI-SOPTD-SOD$_\text{n}$) The expressions corresponding to the PI-SOPTD-SOD$_\text{n}$ are the following ones:

$$\begin{pmatrix}\dot{x}_p(t)\\ \ddot{x}_p(t)\\ \dot{x}_c(t)\end{pmatrix} = \begin{pmatrix}0 & 1 & 0\\ -\frac{1}{\tau_1\tau_2} & -\frac{\tau_1+\tau_2}{\tau_1\tau_2} & \frac{kk_i}{\tau_1\tau_2}\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}x_p(t)\\ \dot{x}_p(t)\\ x_c(t)\end{pmatrix} + \begin{pmatrix}0 & 0\\ \frac{kk_p}{\tau_1\tau_2} & \frac{k}{\tau_1\tau_2}\\ 1 & 0\end{pmatrix}\begin{pmatrix}u_{nl_k}\\ w\end{pmatrix}$$
$$y(t) = \begin{pmatrix}1 & 0 & 0\\ k_p & 0 & k_i\end{pmatrix}\begin{pmatrix}x_p(t)\\ \dot{x}_p(t)\\ x_c(t)\end{pmatrix}. \tag{2.51}$$

As in the FOPTD case, for this system, it is not possible to find analytical solutions due to the exponentials that appear in the equations. Therefore, numerical methods must be used to find the solutions.

Controller variable sampling (PI-SOD$_\text{n}$-SOPTD) The expressions corresponding to the PI-SOD$_\text{n}$-SOPTD are the following:

$$\begin{pmatrix}\dot{x}_p(t)\\ \ddot{x}_p(t)\\ \dot{x}_c(t)\end{pmatrix} = \begin{pmatrix}0 & 1 & 0\\ -\frac{1}{\tau_1\tau_2} & -\frac{\tau_1+\tau_2}{\tau_1\tau_2} & 0\\ 1 & 0 & 0\end{pmatrix}\begin{pmatrix}x_p(t)\\ \dot{x}_p(t)\\ x_c(t)\end{pmatrix} + \begin{pmatrix}0 & 0\\ \frac{k}{\tau_1\tau_2} & \frac{k}{\tau_1\tau_2}\\ 1 & 0\end{pmatrix}\begin{pmatrix}u_{nl_k}\\ w\end{pmatrix}$$
$$y(t) = \begin{pmatrix}1 & 0 & 0\\ k_p & 0 & k_i\end{pmatrix}\begin{pmatrix}x_p(t)\\ \dot{x}_p(t)\\ x_c(t)\end{pmatrix}. \tag{2.52}$$

As in the previous case, the solutions of the equations must be found by means of numerical tools.

2.4.4 Implementation in MATLAB®

The algorithm to obtain the periods and amplitudes of the simulation examples and the models identified from experimental data has been implemented in MATLAB. The code is shown in Listing 2.1. First, the system matrices are defined, corresponding to the PI controller with sampling at the process output (lines 7–11), and with the sampling at the controller output (lines 12–16). Then, the type, order, and parameters of the sampler are stored in the variables `sampling`, `delta`, and `alpha` (lines 17–22). Finally, the set of equations is defined and solved in lines 33–38.

Table 2.1 Summary table with the limit cycle periods and amplitudes

$P(s)$	PI-process-SOD$_1$	PI-SOD$_1$-process
$\frac{k}{s}\mathrm{e}^{-\tau s}$	$T = \frac{1}{2}\frac{kk_i\tau^2-2kk_p\tau-2}{k(k_p-k_i\tau)(\alpha-1)\alpha}$ $t_1 = (1-\alpha)T, t_2 = \alpha T$ $\Delta_{po} = \frac{\delta}{2}\frac{kk_i\tau^2-2kk_p\tau-2}{k_p-k_i\tau}$ $\Delta_{co} = \frac{k_p^2+k_i}{k}\Delta_{po}$	$T = \frac{1}{2}\frac{kk_i\tau^2-2kk_p\tau-2}{k(k_p-k_i\tau)(\alpha-1+\frac{w}{\delta})(\alpha+\frac{w}{\delta})}$ $t_1 = (1-\alpha-\frac{w}{\delta})T,$ $t_2 = (\alpha+\frac{w}{\delta})T$ $\Delta_{po} = \frac{\delta}{2}\frac{kk_i\tau^2-2kk_p\tau-2}{k_p-k_i\tau}$ $\Delta_{co} = \delta$
$\frac{k}{s^2}\mathrm{e}^{-\tau s}$	Limit cycle does not exist	
$\frac{\mathrm{e}^{-\tau s}}{\tau_1 s+1}$	T, Δ_{po}, and Δ_{co} can be obtained using numerical methods	
$\frac{\mathrm{e}^{-\tau s}}{(\tau_1 s+1)(\tau_2 s+1)}$		

Δ_{po} denotes the amplitude of the process output, and Δ_{co} is the amplitude of the controller output

Listing 2.1 Limit Cycle Finder algorithm

```
%% Implementation of the Limit Cycle Finder Algorithm
clear all; clc;
numberOfProcessStates = size(Ap, 1);
numberOfProcessInputs = size(Bp, 1);
numberOfProcessOutputs = size(Cp, 1);
I = eye(numberOfProcessStates+1);
% Process Sampling
Aps = [Ap ki*Bp; zeros(1, numberOfProcessStates+1)];
Bps = [kp*Bp Bp; 1 0];
Cps = [Cp zeros(numberOfProcessOutputs, 1);];
Dps = [zeros(numberOfProcessOutputs, numberOfProcessInputs+1);
     -kp zeros(1, numberOfProcessInputs)];
% Controller Sampling
Acs = [Ap ki*Bp; zeros(1, numberOfProcessStates+1)];
Bcs = [kp*Bp Bp; 1 0];
Ccs = [Cp zeros(numberOfProcessOutputs, 1);];
Dcs = [zeros(numberOfProcessOutputs, numberOfProcessInputs+1);
     -kp zeros(1, numberOfProcessInputs)];
% Type of sampling ('process', 'controller')
sampling = 'process';
delta = 1.0;
alpha = 0.5;
% Hysteresis bands
n = 1;
% Switchings
if (alpha == 0.0)
     m = 2*n-2;
     u = [(alpha-n+1):(alpha+n-2); ones(1,m)*disturbance];
elseif (alpha == 0.5)
     m = 2*n-1;
     u = [(alpha-n+1):(alpha+n-1); ones(1,m)*disturbance];
end

tic,
for tau = 0:0.1:1
    f = @(t) slc(t, u, Aps, Bps, Cps, tau);
    Tguess = 5;
    k = Tguess*rand(1, m);
    [sol, val] = fsolve(f, k);
end
elapsedTime = toc;
```

2.5 Simulation Results

This section shows simulations which illustrate the behavior of the different combinations of control schemes and processes commented in the previous sections.

2.5.1 PI-IPTD-SOD$_n$ and PI-SOD$_n$-IPTD

Let us consider an IPTD process controlled by a PI controller with event-based sampling, and let the values of the plant parameters be $k = 1.0, \tau = 0.2$. The controller gains $k_p = 1.2, k_i = 1.0$ have been chosen to produce a limit cycle with two states, and the sampler $\alpha = 0.5, \delta = 0.1$. The system is described by the following expressions:

$$\begin{pmatrix} \dot{x}_p(t_k) \\ \dot{x}_c(t_k) \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1.0 & 0 \end{pmatrix} \begin{pmatrix} x_p(t_k) \\ x_c(t_k) \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 1.2 & 1 \end{pmatrix} \begin{pmatrix} u_{nl_k} \\ w \end{pmatrix}. \tag{2.53}$$

There exists a symmetric limit cycle with two states (see Fig. 2.7a), with period $T = 5.6093$ and amplitude $\Delta_{po} = 0.2095$, which have been computed with the algorithm presented in Sect. 2.4.2. With the chosen gains, the system converges to the limit cycle after introducing a step change in the set-point.

When the sampler is placed at the controller output, the limit cycle that appears (see Fig. 2.7c) has the same period $T = 5.6093$ but a different amplitude $\Delta_{po} = 0.1402$. While in the first case an external disturbance does not vary the properties of the limit cycle, in the second case it does as it is shown in Fig. 2.8.

Varying the parameters of the controller, it is possible to obtain limit cycles with higher number of levels. For example, increasing the integral gain of the controller to $k_i = 2.0$, and also the order of the sampler to $N_l = 2$, the system with the sampler at the process output presents the response shown in Fig. 2.7b and with the sampler at the controller output, it has the response of Fig. 2.7d. The period and amplitude of the oscillation computed are $T = 4.9745$ and $\Delta_{po} = 0.3584$, which correspond to the results obtained in the simulation.

The simulations show that the system, with the chosen parameters, converges to a limit cycle even in the presence of constant disturbances. The local stability of the limit cycles can also be proven by applying Proposition 2.3. As an example, for the two-level limit cycle, looking at the eigenvalues of the matrix $W = W_2 W_1$, where

$$W_i = \begin{pmatrix} 1 - u_{nl_i} & 0 \\ -x_{p_i} + 1.2(u_{nl_i} + w) + t_i^* & 1 \end{pmatrix}.$$

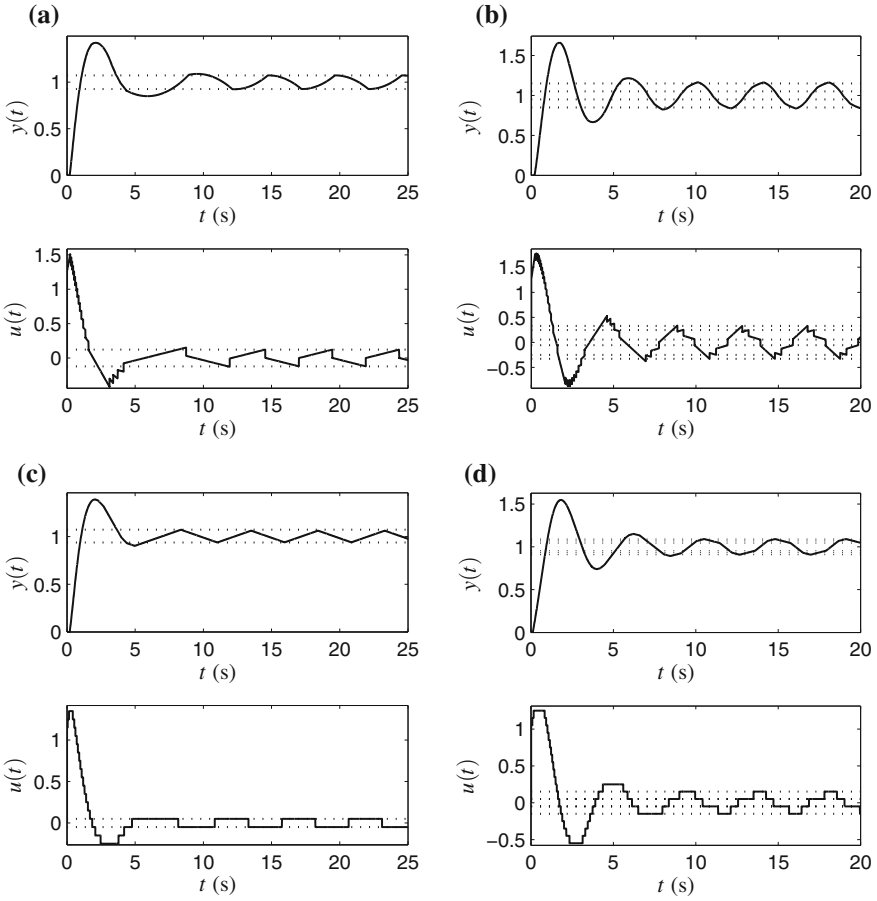

Fig. 2.7 Limit cycles in an IPTD process controlled by a PI with event-based sampling at the process output with one hysteresis band (**a**) and two (**b**), and with sampling at the controller output with one hysteresis band (**c**) and two (**d**). The *dotted lines* show the sampling levels of the process variable in (**a**), (**c**) and of the control variable in (**b**), (**d**), and the value at the switching times of the control variable in (**a**), (**c**) and of the process variable in (**b**), (**d**)

after straightforward computations, it can be shown that the eigenvalues of W are $\lambda_1 = 1, \lambda_2 = (1-\alpha)\alpha < 1$, and therefore the limit cycle is locally stable. If limit cycles with $2n$ switchings are considered, then the eigenvalues of W are $\lambda_1 = 1, \lambda_2 = \prod_{i=0}^{2N_l-1}(1-\alpha-N_l+i)$. It is easy to verify that $|\lambda_2| \leq 1$ for every α when $N_l = 1, 2$, i.e., the corresponding limit cycles are locally stable. However, for $N_l > 2$, the local stability of the limit cycles depends on the particular value of α.

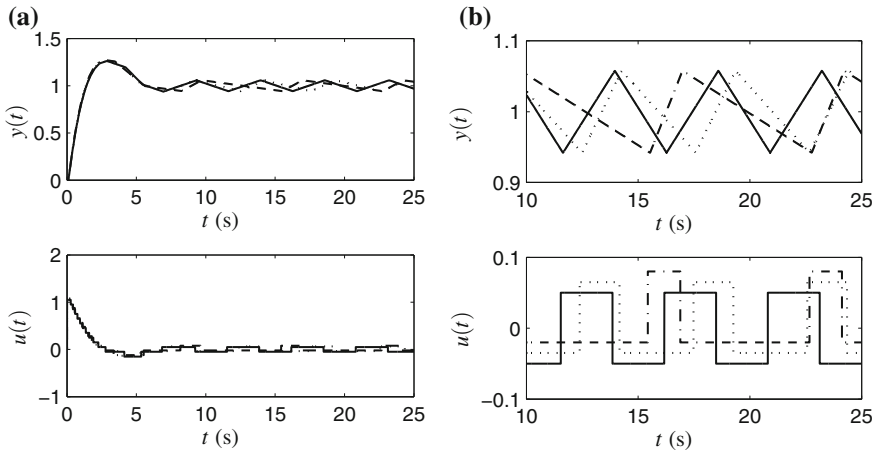

Fig. 2.8 Limit cycles in an IPTD process controlled by a PI with event-based sampling at the controller output **a** with an external disturbance $w = 0.0$ (*solid line*), $w = 0.015$ (*dotted line*), and $w = 0.03$ (*dashed-dotted line*). **b** Detail of the limit cycles

2.5.2 PI-SOPTD-SOD$_n$

Now, consider a SOPTD with parameters $k = 1.0$ (gain), $\tau_1 = 1.0$, $\tau_2 = 0.5$ (time constants), and $\tau = 0.2$ (time delay), which is controlled by a PI with event-based sampling placed at the process output, with $\alpha = 0.5$ and $\delta = 0.1$. Setting the controller gains to $k_p = 1.2$ and $k_i = 1.0$, the matrices A and B of the system are

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 2 & -2 & -3 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 2.4 & 2 \end{pmatrix}. \tag{2.54}$$

Solving the system of equations corresponding to the limit cycle with one band of hysteresis, the values of the switching times and levels are

$$\begin{aligned} t_1 \approx 3.0772, t_2 \approx 3.0772, \\ \dot{x}_{p_1} \approx -0.0517, \dot{x}_{p_2} \approx 0.0517, \\ \dot{x}_{c_1} \approx -0.0669, \dot{x}_{c_2} \approx 0.0669. \end{aligned}$$

Finally, the period is $T = t_1 + t_2 \approx 6.1545$ and the amplitude of the process output is $A \approx 0.1338$. This limit cycle, and another composed of two hysteresis bands, is plotted in Fig. 2.9.

It is interesting to note that, when the sampler is placed at the process output, the limit cycle does not vary when a constant disturbance affects the system, because it is rejected by the controller.

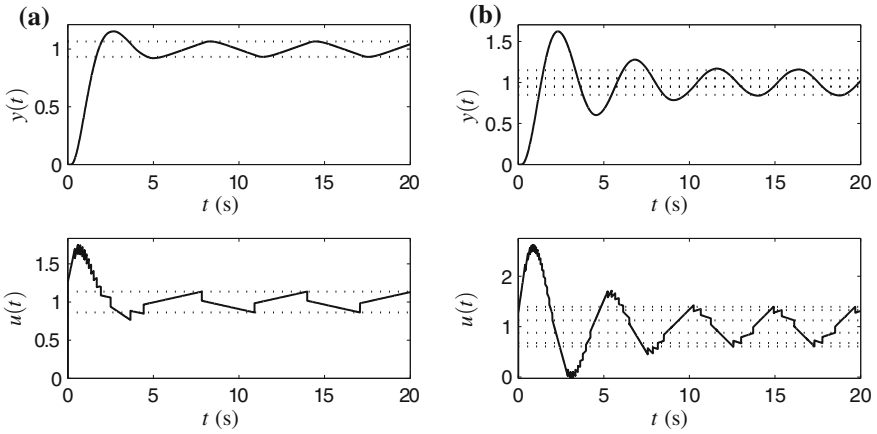

Fig. 2.9 Limit cycles in an SOPTD process controlled by a PI with event-based sampling at the controller output, involving **a** one, and **b** two hysteresis bands

The local stability of the limit cycle can be studied by looking at the eigenvalues of the matrix W:

$$W = W_2 W_1 = \begin{pmatrix} 0 & 0 & 0 \\ -0.0787 - 2.5953w & 0.0042 & 0.0021 \\ 0.0699 + 2.4253w & -0.0042 & -0.0021 \end{pmatrix}. \tag{2.55}$$

Since the eigenvalues of W are inside the unit circle ($\lambda_1 \approx 0.0021$, $\lambda_2 \approx 0.0000$, $\lambda_3 \approx 0.0000$), the limit cycle is locally stable.

2.6 Experimental Results

This section shows experimental results which clearly evidence the existence of the results derived in the previous sections in a real system. Therefore, the experiences carried out were focused on finding limit cycles in the Acurex system to compare them with that predicted by the theory and simulations. As shown in the following paragraphs, it is worth noting that even when the model used is a simplification of the process which ignores many of the complex dynamics existent in the system, the results are very close to those predicted in theory. Below, the Acurex system and the model identified from experimental data are presented, commenting some implementation issues of the controller and, finally, the obtained results are shown and interpreted.

Fig. 2.10 Acurex distributed collector system at the PSA, Spain

2.6.1 The Acurex System

The experiments have been done with an equipment, known as Acurex, built in 1981 at the PSA, Spain [228]. In this plant (Fig. 2.10), two types of collecting systems were considered, a central receiver system (CRS) and a distributed collector system (DCS) using parabolic troughs. Parabolic trough systems concentrate sunlight onto a receiver pipe which contains a heat transfer fluid (HTF) that is heated as it flows along the receiver pipe. Then, the HTF is used to produce steam that may be used for example to feed an industrial process. A survey of basic and advanced control approaches for distributed solar collector fields can be found in [29, 30]. For more information on control of solar plants see [31].

2.6.2 The Model

The plant was identified as a FOTPD, where the process input is the oil flow (l/s) in the pipes and the process output is the temperature of the oil at the collector field outlet (°C). There are unmodeled dynamics that are considered as external disturbances, such as the oil temperature at the input or the solar irradiance. However, because of the time scale of the tests performed, which is smaller than the rate of variation of these variables (under clear day conditions without clouds), the model obtained seems to be a valid representation of the process for our purposes.

The transfer function was identified from experimental data obtained from the plant in a set of step tests. The procedure followed in each test is to drive the temperature manually to the working point, and when the process has reached it to introduce a step change in the input, registering the data measured from the sensors until the process stabilizes again. The FOPTD model, obtained by applying a least-squares procedure, is

$$P(s) = -\frac{6.0715}{103.2723s + 1} \mathrm{e}^{-67.5021s}, \tag{2.56}$$

where the time constant τ_1 and time delay τ are given in seconds, and the gain k in °Cs/l. The tests were carried out with an input around 8.5 l/s, being the range of the pump from 2 to 12 l/s. Also, the time constants and delays obtained make sense from the previous published works in this field. Notice the minus sign in (2.56), which represents the inverse response of the plant, i.e., a positive change in the flow produces a negative change in the temperature.

2.6.3 Implementation

The Acurex system has a SCADA software developed in LabVIEW™. This software provides the user with an interface to execute its own controller implementation in MATLAB code. Thus, one must write a MATLAB callback function which is invoked with a configurable sampling period (it was fixed to $T_s = 15$ s). This function receives the measures from the sensors, updates the controller state and, finally, sends the new control action to the actuators.

The controller implementation can be configured to work in three modes, namely,

- **manual** the control input is set manually,
- **SOD-PI** the sampler is at the process output, and
- **PI-SOD** the sampler is at the controller output.

An excerpt of the code of the controller is shown in Listings 2.2 and 2.3.

Listing 2.2 Code of the controller with the sampler at the process output

```
% Sampling at the process output
    e = setpoint - output;
    if(~((u_prev >= umax && e > 0) || (u_prev <= umin && e < 0))
        )
        I = I + e_prev*Ts;
    end

    % event detection
    if (abs(e - e_prev) > delta)
        levels = floor(abs(e - e_prev) / delta);
        e_prev = e_prev + sign(e - e_prev)*delta*levels;
        u_prev = sat(Kp*e_prev + Ki*I, umin, umax);
        I = (u_prev - Kp *e_prev) / Ki;
    end
```

Listing 2.3 Code of the controller with the sampler at the controller output

```
% Sampling at the controller output
    e =  - output;

    % anti-windup
    if(~((u_prev >= umax && e > 0) || (u_prev <= umin && e <
        0)))
        I = I + e*Ts;
        u = Kp*e + Ki*I;
    else
        u = u_prev;
    end

    % event detection
    if (abs(u - u_prev) > delta)
        levels = floor(abs(u - u_prev) / delta);
        u_prev = sat(u_prev + sign(u - u_prev)*delta, umin,
            umax);
    end
```

2.6.3.1 Controller Sampling

The first set of tests presented was carried out with the controller in *PI-SOD* mode, with the purpose of reproducing the two-level limit cycles obtained in simulation (with $\alpha = 0.5$) and the three-level limit cycles (with $\alpha = 0.0$). The procedure followed is the same for each test, first the system temperature is moved to an operating point, and next when the process reaches a steady state, a set-point step change is introduced into the controller.

The response is shown in Fig. 2.11, where the oil temperature, the flow, and the solar irradiance during the test are plotted. It can be seen that the system goes into a limit cycle composed of two levels, which is similar to the results obtained in simulation with the FOPTD model.

Figure 2.12a, b shows the comparison of the limit cycles obtained in simulation with the model and the results obtained with the Acurex system. The results are similar both qualitatively (limit cycles with two states) and quantitatively (the period and amplitude are approximately equal).

2.6.3.2 Process Sampling

The second set of tests was carried out with the sampler placed at the process output. After verifying that, as in the previous section, the system enters into a limit cycle of two levels (Fig. 2.13), the existence of more complex limit cycles was investigated. Increasing the proportional gain, a limit cycle with eight different levels was found, which is shown in Fig. 2.14. It is remarkable that even in this case, the comparison between the experimental data and the simulated process shows that there are no significative differences in the behavior.

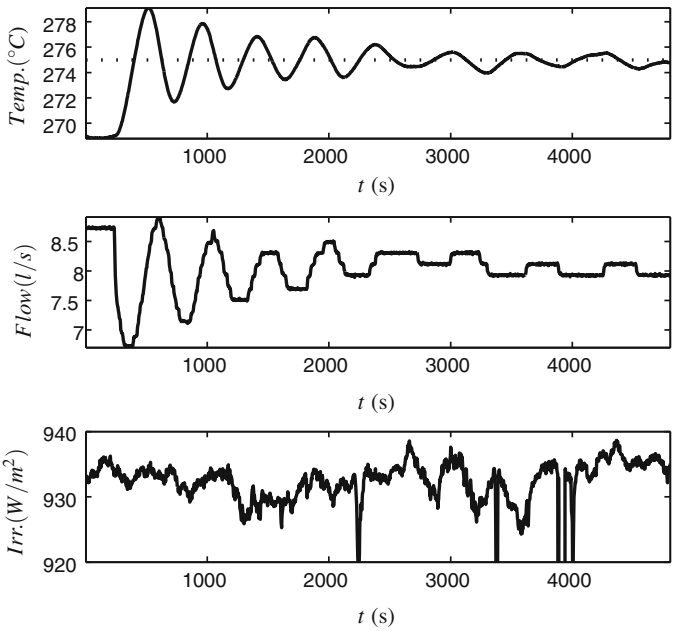

Fig. 2.11 Response of the Acurex system (*solid line*) to a step change in the set-point, with the event-based sampler placed at the controller output

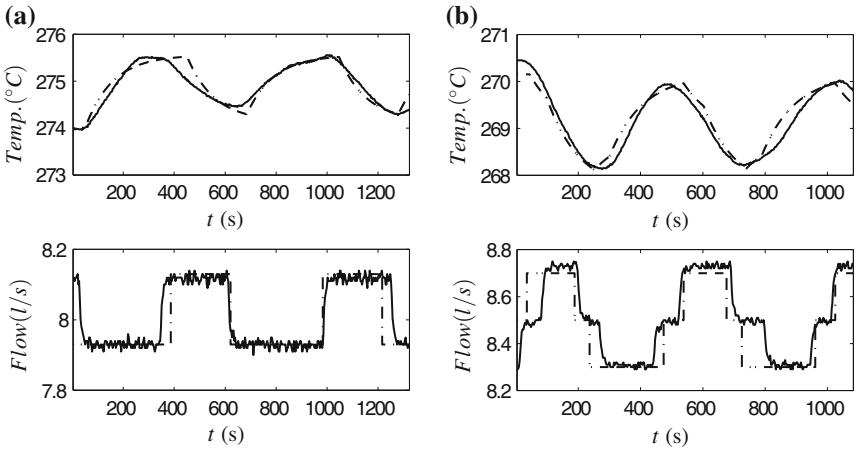

Fig. 2.12 Limit cycles in the Acurex system (*solid line*) and in the simulated model (*dashed line*) controlled by a PI with SOD sampling placed after the controller output. **a** With two levels and **b** with three levels

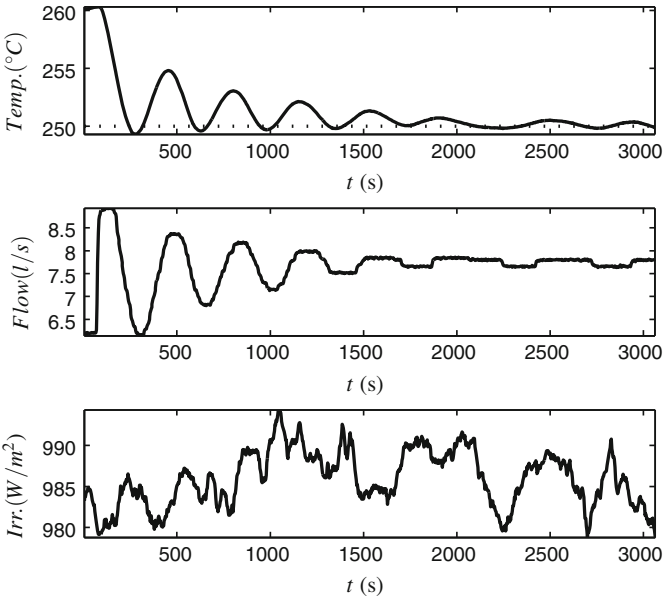

Fig. 2.13 Response of the Acurex system (*solid line*) to a step change in the set-point, with the event-based sampler placed at the process output

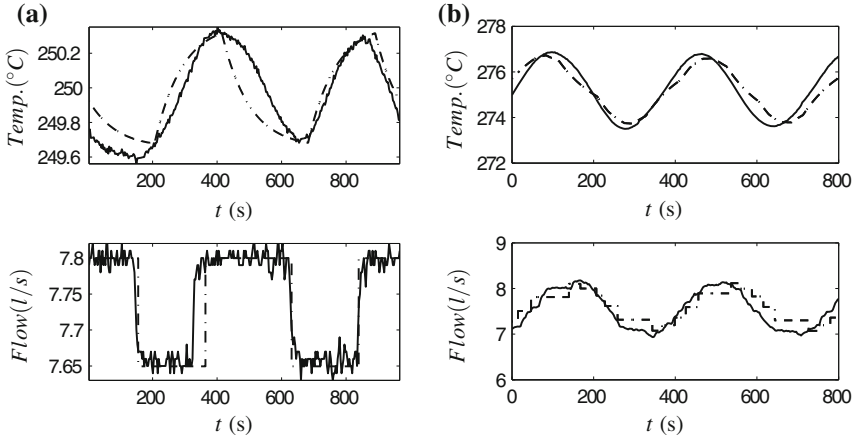

Fig. 2.14 Limit cycles in the Acurex system (*solid line*) and in the simulated model (*dashed line*) controlled by a PI with SOD sampling placed after the process output. **a** With two levels and **b** with eight levels

2.7 Conclusions

The behavior of a control system based on the use of a level crossing sampling either in the process output or in the control output has been studied. Limit cycles are of particular interest since they are associated to oscillations in processes, and therefore it is worth gaining knowledge about them in order to avoid them when possible or to assure that they are not problematic.

When trying to find properties about the limit cycles, it is common to have systems of equations involving transcendent functions and thus it is not possible, in general, to find closed-form solutions. Moreover, due to the combinatorial explosion, it can be computationally expensive to find these solutions, and it becomes harder when higher order process models are considered.

Therefore, an algorithm to analyze the properties of the limit cycles has been proposed; it allows us to introduce some knowledge in the hypothesis of the problem, so that the complexity can be reduced.

A set of simulation results illustrates the behavior of the controllers with some models frequently used in industrial context, which are the IPTD, the FOPTD, and the SOPTD. Also, this behavior has been tested and verified in the Acurex Field of the Solar Platform of Almería, Spain. The experiments performed confirm that the simulation results can be extrapolated to real cases, obviously with the divergences due to unmodeled dynamics of the process, disturbances, etc.

Chapter 3
Self-triggered Sampling Selection Based on Quadratic Programming

Luis Orihuela, Pablo Millán and David Muñoz de la Peña

3.1 Introduction

Event-based and self-triggered techniques are aimed to reduce the communication requirements of a control system by using variable sampling rates. Event-based approaches are based on triggering signal transmissions according to some conditions on the plant state or outputs [12, 46, 107, 154, 239], which requires a continuous monitoring of these signals. Self-triggered approaches try to emulate event-based ideas [6, 8, 46], but avoiding a continuous monitoring of the signals. Self-triggered methods require a model of the system and a bound on the uncertainties to decide when to trigger a signal transmission, which may result in some conservativeness in choice of the inter-sampling times. However, among other advantages, these techniques allow the communication devices to save battery by going to sleep mode during the inter-sampling periods [6, 8, 46].

Under the framework of asynchronous control for single-loop schemes, this chapter studies the problem of reducing the use of a bandwidth-limited sensor-to-controller channel in a single control loop using a self-triggered approach. The system to be controlled is modeled using a linear time invariant plant subject to bounded additive disturbances. Assuming that a stabilizing feedback controller and its corresponding Lyapunov function are available, a model-based controller is proposed in which the controller operates in open loop between two consecutive samples. The

L. Orihuela · P. Millán (✉)
Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería,
Universidad Loyola Andalucía, Seville, Spain
e-mail: pmillan@uloyola.es

L. Orihuela
e-mail: dorihuela@uloyola.es

D.M. de la Peña
Dpto. de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingenieros,
Universidad de Sevilla, Seville, Spain
e-mail: dmunoz@us.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_3

following sampling time is decided by the controller, in such a way that practical stability is guaranteed while minimizing the number of access to the shared network, that is, while maximizing the time between consecutive samples. In order to decide the following sampling time the controller must solve several on-line quadratic optimization problems (QP). This Lyapunov-based sampling policy results in a self-triggered sampling strategy as in [163]. The main difference with that work is the use of a model to minimize the access to the shared network. The problem is presented in discrete time, but the extension for continuous systems with sampled state measurements is included.

The following section presents a description of the system, controller and network. Section 3.3 describes the Lyapunov-based sampling policy. The extension for continuous plants is given in Sect. 3.4. A simulation example is shown in Sect. 3.5. Conclusions and future research proposals are summarized in Sect. 3.6.

3.2 Problem Formulation

The first part of this section presents the model for the system under consideration. Then, the model-based controller proposed in this chapter is introduced.

3.2.1 Plant Description

Consider the following discrete-time linear system given by:

$$x(k+1) = Ax(k) + Bu(k) + w(k), \tag{3.1}$$

where $x(k) \in \mathbb{R}^n$, and $u(k) \in \mathbb{R}^m$ are the state vector and control input vector, respectively. The process disturbance is $w(k) \in \mathbb{R}^n$, and satisfies $w(k) \subseteq \mathscr{W}$, where

$$\mathscr{W} = \{w \in \mathbb{R}^n : \|w(k)\|_\infty \leq \gamma,\ \gamma > 0\}. \tag{3.2}$$

It is assumed that a feedback local controller K, associated with a discrete Lyapunov function $V(x) = x^T Px$, has been designed for system (3.1) so that the control law $u(k) = Kx(k)$ ensures practical stability of the closed-loop system.

3.2.2 Model-Based Controller

Consider system (3.1) being controlled through a shared network used by several independent control loops. The network connects the sensor and the model-based controller, while the actuator is considered to have direct connection to the plant.

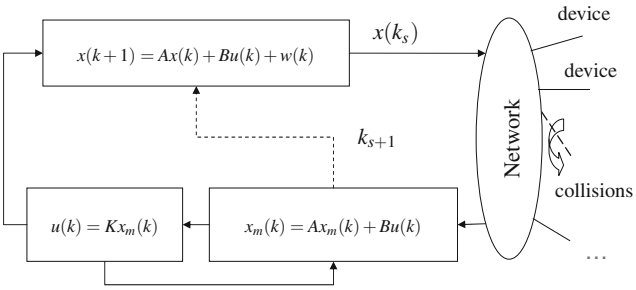

Fig. 3.1 Networked control system

The inclusion of a shared network induces collisions and packet dropouts, which may affect the dynamic behavior of the closed-loop system. This problem becomes more important as the number of devices connected to the network and the sampling frequency of such devices grow. In order to control the system while minimizing the network traffic load, we resort to a model-based controller given by the following equations:

$$x_m(k+1) = Ax_m(k) + Bu(k), \tag{3.3}$$
$$x_m(k_s) = x(k_s), \quad s = 0, 1, 2, \ldots \tag{3.4}$$
$$u(k) = Kx_m(k), \tag{3.5}$$

where k_s are the discrete-time instants at which the sensors measure the state of the plant and send it to the controller. Figure 3.1 shows an scheme of the proposed control system. The model state is updated whenever a new sample arrives. Then, the model evolves in open loop until another measurement reaches the controller. The main difference between this approach and the one of [181] is that, in the proposed control scheme, the following sampling time is decided on-line by the controller, while periodic sampling times were considered in [181]. As Fig. 3.1 shows, the controller is close to the plant, and hence, there are no communication dynamics in the feedback link.

A communication protocol between the sensors and the controller is assumed to be operating, in such a way that it is possible for the controller to decide the sampling instants. This could be performed, for instance, if the controller sends a packet to the sensors that contains the following sampling time.

Under these considerations, the closed-loop system is modeled by the following equations:

$$x(k+1) = Ax(k) + BKx_m(k) + w(k), \tag{3.6}$$
$$x_m(k+1) = (A + BK)x_m(k), \quad \forall k \in [k_s, k_{s+1}), \tag{3.7}$$
$$x_m(k_s) = x(k_s), \quad s = 1, 2, \ldots \tag{3.8}$$
$$k_{s+1} = f(x(k_s)), \tag{3.9}$$

where time k_s, with $s = 1, 2, \ldots$, are the time instants in which the controller receives the measurements form the sensor. The sampling instants k_s are computed by the controller based on the state measurements received.

In the next section, we present a method to decide the next sampling instant k_{s+1} based on the system model, the controller gain K, the Lyapunov function V, and the latest state measurement $x(k_s)$ in order to minimize the access to the network while guaranteeing closed-loop practical stability.

3.3 Lyapunov-Based Sampling Procedure

3.3.1 Main Idea and Stability Analysis

This section describes the proposed procedure to minimize the access to the network while preserving closed-loop practical stability.

In view of Eqs. (3.1), (3.3)–(3.5), the model error $\delta(k)$ can be defined as:

$$\delta(k) \triangleq x(k) - x_m(k), \tag{3.10}$$

where $\delta(k_s) = 0, \ \forall k_s$. A possible evolution of the state of the system $x(k)$, the controller $x_m(k)$, and the error $\delta(k)$ is drawn in Fig. 3.2.

The dynamics of the controller state and the model error between two consecutive sampling times can be written as follows:

$$x_m(k_s+j) = (A+BK)^j x(k_s), \ \forall j \in \mathbb{N} : \{k_s + j < k_{s+1}\} \tag{3.11}$$

$$\delta(k_s+j) = \sum_{i=1}^{j} A^{i-1} w(k_s+j-i), \ \forall j \in \mathbb{N} : \{k_s + j < k_{s+1}\}. \tag{3.12}$$

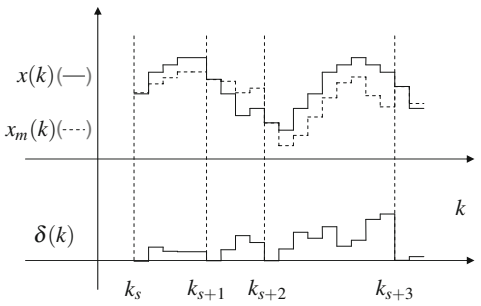

Fig. 3.2 Possible evolution of the state and the model error

From Eqs. (3.2) and (3.12), the following upper bound on the error is obtained:

$$\|\delta(k_s+j)\|_\infty < \gamma \sum_{i=1}^{j} \|A^{i-1}\|_\infty, \tag{3.13}$$

for all $w(k_s+i)$, with $i = 0, \ldots, j-1$. The forward difference of the Lyapunov function for $k \in [k_s, k_{s+1})$ yields

$$\begin{aligned} \Delta V(k_s, k_s+j) &= V(x(k_s+j)) - V(x(k_s)) &(3.14)\\ &= x^T(k_s+j)Px(k_s+j) - x^T(k_s)Px(k_s),\ j : \{k_s+j < k_{s+1}\}. &(3.15)\end{aligned}$$

Now, substituting $x(k)$ from Eq. (3.10), we obtain the following equality:

$$\begin{aligned} \Delta V(k_s, k_s+j) = {} & \delta^T(k_s+j)P\delta(k_s+j) + 2x_m^T(k_s+j)P\delta(k_s+j) \\ & + x_m^T(k_s+j)Px_m(k_s+j) - x_m^T(k_s)Px_m(k_s). \end{aligned} \tag{3.16}$$

The controller's goal is to maximize the next sampling instant k_{s+1} while guaranteeing that the forward difference is negative for all possible disturbances in order to ensure practical stability. To this end, the controller solves the following optimization problem:

$$\begin{aligned} &\max\ k_{s+1} &(3.17)\\ &\text{subject to:}\\ &\Delta V(k_s, k_s+j) \le 0,\ \forall j \in \mathbb{N} : \{k_s+j < k_{s+1}\}\\ &\forall w(k_s+i),\ i = 0, \ldots, j-1. \end{aligned}$$

The reader can see that the optimization problem looks for the maximum sampling time that satisfies that the Lyapunov function decreases in spite of the worst disturbances possible.

3.3.2 Algorithm to Select the Following Sampling Time

The optimization problem (3.17) is non-convex and in general is hard to solve. We present next an iterative algorithm that provides a conservative feasible solution based on solving a sequence of optimization problems.

Algorithm 3.1

1. Set $j = 1$.

2. Solve the problem

$$\max_{\delta} \ \Delta V(k_s, k_s + j) \tag{3.18}$$

subject to:

$$\|\delta\|_\infty < \gamma \sum_{i=1}^{j} \|A^{i-1}\|_\infty.$$

3. If $\Delta V(k_s, k_s + j) \leq 0$, increase $j = j + 1$ and go to Step 2. Otherwise, choose $k_{s+1} = k_s + j$.

Algorithm 1 increases k_{s+1} iteratively while a worst case bound on the difference between the value of the Lyapunov function of the current state and the state corresponding to the next sampling time is negative. Once this constraint does not hold, the algorithm stops. This implies that the P-norm of $V(x(k_s))$ is decreasing, with a lower bound given by the size of the uncertainty, and hence that the closed-loop system is practically stable.

3.3.3 Quadratic Programming Problem

Next, we will prove that this optimization problem can be stated as a QP problem. First of all, the standard QP problem is introduced, see [195].

Quadratic programming problem Assume ξ belongs to $\mathbb{R}^p$ space. The $p \times p$ matrix H is symmetric, and f is any $f \times 1$ vector. The QP problem is stated as

$$\min_{\xi} \ g(\xi) = \frac{1}{2}\xi^T H\xi + f^T\xi + c, \tag{3.19}$$

subject to

$$D\xi \leq b \text{ (inequality constraint).} \tag{3.20}$$

Proposition 3.1 *Problem (3.18) can be formulated as a QP if the elements of Eqs. (3.19)–(3.20) are chosen as*

$$\begin{aligned} \xi &= \delta, \\ H &= -2P, \\ f^T &= -2\left[(A + BK)^j x_m(k_s)\right]^T P, \\ c &= -\left[(A + BK)^j x_m(k_s)\right]^T P\left[(A + BK)^j x_m(k_s)\right] + x_m^T(k_s)Px_m(k_s), \end{aligned} \tag{3.21}$$

and for the inequality constraint

$$D = \begin{bmatrix} I_n \\ -I_n \end{bmatrix}, \quad b = \gamma \sum_{i=1}^{j} \|A^{i-1}\|_\infty \begin{bmatrix} \bar{1}_n \\ -\bar{1}_n \end{bmatrix}. \tag{3.22}$$

Where $\bar{1}_n$ *is a column vector of dimension n whose components are ones and* I_n *is the identity matrix with dimension n.*

Proof The proof is straightforward from Eqs. (3.16), (3.19)–(3.20). □

In view of Proposition 3.1, the controller needs to solve several QP problems in Algorithm 1 to find the next sampling time.

Remark 3.1 The minimum sampling time that can be chosen in Algorithm 1 is one. It is not possible to ensure that the Lyapunov function decreases for all k because of the presence of bounded disturbances $w(k)$, which can make $\Delta V(k, k+1)$ strictly positive in a neighborhood of the origin. However, it is worth reminding that, by assumption, the system practical stability is guaranteed for the controller K with sampling time equal to one.

It is important to remark that the QP problem that needs to be solved in order to solve (3.18) is a multi-parametric QP problem (mpQP), for which the explicit solution can be obtained, see [19]. In particular, the parameter $\theta \in R^{n+1}$ of the mpQP problem is

$$\theta(k_s, j) = \begin{bmatrix} x_m(k_s + j) \\ \sum_{i=1}^{j} \|A^{i-1}\|_\infty \end{bmatrix}.$$

This allows implementing the proposed variable sample control scheme efficiently.

3.4 Extension to Continuous Systems

The control scheme presented in the previous section can be readily extended to continuous-time systems under the following assumptions. Consider the following continuous-time linear system subject to bounded disturbances.

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t), \tag{3.23}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input vector and $w(t) \in \mathscr{W} \subset \mathbb{R}^n$ is the process disturbance where

$$\mathscr{W} = \{w \in \mathbb{R}^n : \|w(t)\|_\infty \leq \gamma,\ \gamma > 0\}. \tag{3.24}$$

We assume that there exists a linear controller $u(t) = Kx(t)$ that asymptotically stabilizes the nominal system (system (3.23) with $w(t) = 0$) with a corresponding Lyapunov function $V(x) = x^T Px$.

System (3.23) is controlled through a network with the same structure as that in Fig. 3.1, which implements the following model-based controller (which is the continuous-time version of (3.3)–(3.5)):

$$\dot{x}_c(t) = Ax_m(t) + Bu(t), \quad \forall t \in [t_k, t_{k+1}) \tag{3.25}$$

$$u(t) = Kx_m(t), \tag{3.26}$$

$$x_m(t_k) = x(t_k), \quad k = 0, 1, 2, \ldots \tag{3.27}$$

where t_k is the sampling time (equivalent to k_s) in which the sensors send new information to the controller. A Lyapunov-based control design procedure can now be followed similarly to that in Sect. 3.3. We define the model error variable $\delta(t)$ as

$$\delta(t) \triangleq x(t) - x_m(t). \tag{3.28}$$

The dynamic of the error equation now becomes

$$\begin{aligned} \dot{\delta}(t) &= \dot{x}(t) - \dot{x}_c(t) \\ &= Ax(t) + Bu(t) + w(t) - Ax_m(t) - Bu(t), \\ &= A\delta(t) + w(t), \quad \forall t \in [t_k, t_{k+1}). \end{aligned} \tag{3.29}$$

Thus, the dynamics of the controller state and the model error between two consecutive sampling times evolves as

$$x_m(t) = e^{(A+BK)(t-t_k)} x_m(t_k), \quad \forall t \in [t_k, t_{k+1}) \tag{3.30}$$

$$\begin{aligned} \delta(t) &= e^{A(t-t_k)}\delta(t_k) + \int_{t_k}^{t} e^{A(t-s)} w(s) ds \\ &= \int_{t_k}^{t} e^{A(t-s)} w(s) ds, \quad \forall t \in [t_k, t_{k+1}). \end{aligned} \tag{3.31}$$

The following proposition is needed for further developments.

Proposition 3.2 *If the dynamics of the error variable is given by (3.31), the error can be bounded as follows:*

$$\|\delta(t)\|_\infty \leq \gamma \phi(t, t_k) \tag{3.32}$$

where $\phi(t, t_k) = \frac{1}{\|A\|_\infty}(e^{\|A\|_\infty (t-t_k)} - 1)$ *and* $\|A\|_\infty$ *is the infinite norm of A.*

Proof Taking into account Eq. (3.31), the norm of the error can be bounded as follows:

$$\begin{aligned}\|\delta(t)\|_\infty &= \left\| \int_{t_k}^{t} e^{A(t-s)} w(s) ds \right\|_\infty \leq \int_{t_k}^{t} \left\| e^{A(t-s)} \right\|_\infty \|w(s)\|_\infty ds \\ &\leq \gamma \int_{t_k}^{t} e^{\|A\|_\infty (t-s)} ds = \gamma \frac{1}{\|A\|_\infty} (e^{\|A\|_\infty (t-t_k)} - 1).\end{aligned}$$

□

In what follows, the extension of the sampling procedure to continuous systems is developed. Again, the controller's goal is to maximize the next sampling instant t_{k+1}, guaranteeing in this case that the derivative of the Lyapunov function is negative for all possible disturbances. Taking the time derivative of the Lyapunov function for $t \in [t_k, t_{k+1})$ yields

$$\frac{d}{dt} V(t) = x^T(t) P \dot{x}(t) + \dot{x}^T(t) P x(t) = 2 x^T(t) P \dot{x}(t). \tag{3.33}$$

Now, substituting $x(t)$ from Eq. (3.28),

$$\begin{aligned}\dot{V}(x(t)) &= 2(\delta^T(t) + x_c^T(t)) P (\dot{\delta}(t) + \dot{x}_c(t)) \\ &= 2(\delta^T(t) + x_c^T(t)) P (A\delta(t) + w(t) + A x_m(t) + B u(t)) \\ &= \delta^T(t)(PA + A^T P)\delta(t) + 2\delta^T(t) P w(t) + 2 x_c^T(t) P w(t) \\ &+ 2\delta^T(t)(PA + A^T P + PBK) x_m(t) \\ &+ x_m^T(t) \left(P(A + BK) + (A + BK)^T P \right) x_m(t), \ \forall t \in [t_k, t_{k+1}).\end{aligned} \tag{3.34}$$

In the following algorithm, we will ensure the negative definiteness of an upper bound on the time derivative of the Lyapunov function (3.34) at time t for all possible uncertainty trajectories.

The goal of the sampling method can be written as

$$\begin{aligned} &\max \ t_{k+1} \\ &\text{subject to (3.23)}-\text{(3.27) and:} \\ &\frac{d}{dt} V(x(t)) \leq 0, \quad \forall t \in [t_k, t_{k+1}) \end{aligned} \tag{3.35}$$

This optimization problem is very difficult to solve. The parameter to be optimize, i.e., t_{k+1}, is involved in a nonlinear equation and there are an infinite number of constraints, because they must be satisfied for all $t \in [t_k, t_{k+1})$. In order to obtain the next sampling time, we propose to define $t_{k+1} = T_{min} + n\Delta$ and find the maximum n such that the time derivative of the Lyapunov function is negative at those time instants for all possible disturbances.

We present next an iterative algorithm that under mild assumptions provide an approximate solution to (3.35).

Algorithm 3.2

1. Set $t_{k+1} = t_k + T_{min}$, where T_{min} is lower bound for the following sampling time.

2. Solve the problem

$$\begin{aligned} \min_{\delta(t_{k+1}), w(t_{k+1})} \quad & -\frac{d}{dt} V(x(t_{k+1})) \\ \text{subject to:} & \\ \|w(t_{k+1})\|_\infty &\leq \gamma \\ \|\delta(t_{k+1})\|_\infty &\leq \gamma \phi(t_{k+1}, t_k) \end{aligned} \tag{3.36}$$

3. If $\dot{V}(x(t_{k+1})) \leq 0$, increase $t_{k+1} = t_{k+1} + \Delta$ and go to Step 2. Otherwise, choose t_{k+1}.

Next proposition shows that problem (3.36) can be stated as a QP.

Proposition 3.3 *Problem (3.36) can be formulated as a QP if the elements of Eqs. (3.19)–(3.20) are chosen as*

$$\begin{aligned} \xi &= \begin{bmatrix} \delta(t) \\ w(t) \end{bmatrix}, \\ H &= -2 \begin{bmatrix} PA + A^T P & P \\ P & 0 \end{bmatrix}, \\ f^T &= -2 x_m^T(t_{k+1}) \begin{bmatrix} PA + A^T P + K^T B^T P & P \end{bmatrix}, \\ c &= -x_m^T(t_{k+1}) \left(P(A + BK) + (A + BK)^T P \right) x_m(t_{k+1}), \end{aligned}$$

and for the inequality constraint

$$D = \begin{bmatrix} I_n & 0 \\ -I_n & 0 \\ 0 & I_n \\ 0 & -I_n \end{bmatrix}, \quad b = \begin{bmatrix} \gamma \phi(t_{k+1}, t_k) \bar{1}_n \\ \gamma \phi(t_{k+1}, t_k) \bar{1}_n \\ \gamma \bar{1}_n \\ \gamma \bar{1}_n \end{bmatrix}, \tag{3.37}$$

where $\bar{1}_n$ is a column vector of dimension n whose components are ones and I_n is the identity matrix with dimension n.

Proof The proof is straightforward from Eqs. (3.34), (3.19)–(3.20). □

The value of Δ must be chosen small enough in a way such that the dynamics of the controller state, and hence of the Lyapunov function, are smooth between two consecutive sampling, avoiding unexpected sign changes of the derivative of

the Lyapunov function from t_k to t_{k+1}. In general T_{min} is chosen according with the minimum sampling time of the sensors.

As in the discrete-time case, the resulting QP problem is a mpQP, and hence, an explicit solution can be obtained. In this case the parameter of the QP problem is

$$\theta(t_k, j) = \begin{bmatrix} x_m(t_k + j\Delta) \\ \phi(t_k + j\Delta, t_k) \end{bmatrix}.$$

Assuming that the sign of the time derivative of the Lyapunov function does not change between two consecutive times, Algorithm 1 provides a suboptimal solution to problem (3.35). Note that this assumption will be satisfied for a sufficiently small Δ.

The constraint on the upper bound of the time derivative of the Lyapunov function is more restrictive than the constraint on the difference of the Lyapunov function imposed in the discrete-time controller studied in the previous section. Note that, in continuous time, a constraint on the difference of the Lyapunov function does not yield a QP problem and hence is more difficult to implement in real time.

3.5 Simulation Results

In this section, we apply the previous result to a discrete and continuous unstable plants in order to show how the controller manages to reduce the traffic load maintaining the practical stability of the system.

3.5.1 Discrete System

Consider the following discrete-time LTI system:

$$x(k+1) = \begin{bmatrix} 2.72 & 2.70 \\ 0 & 2.69 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 1.7 \end{bmatrix} u(k) + w(k).$$

The initial condition for the system and the controller is $x(0)^T = \begin{bmatrix} 100 & -20 \end{bmatrix}$. The following stabilizing controller has been designed for the discrete system:

$$K = \begin{bmatrix} -1.1868 & -2.0415 \end{bmatrix}.$$

The Lyapunov function is defined by the following positive definite matrix:

$$P = 10^6 \begin{bmatrix} 1.67 & 0.67 \\ 0.67 & 0.60 \end{bmatrix}.$$

Suppose that this system is controlled using a shared network so it is interesting to reduce the number of access to the shared medium.

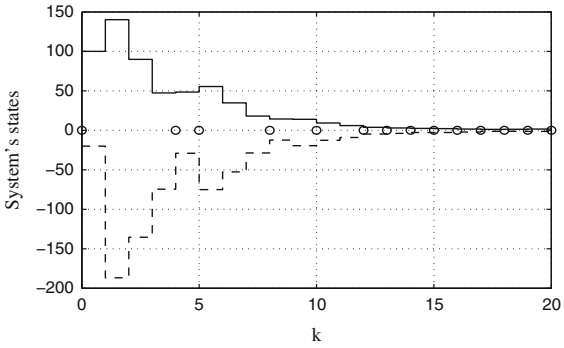

Fig. 3.3 Evolution of the system's states

Assuming that the disturbances are bounded $\|w(k)\|_\infty \leq 1$, the evolution of the system is illustrated in Fig. 3.3, where the sampling instants are indicated with circles.

In Fig. 3.4 the sampling instants obtained using the proposed method are shown. One can see that when the system is far of the equilibrium point it is possible to enlarge the sampling period still assuring asymptotic stability.

Finally, the evolution of the Lyapunov function is drawn in Fig. 3.5.

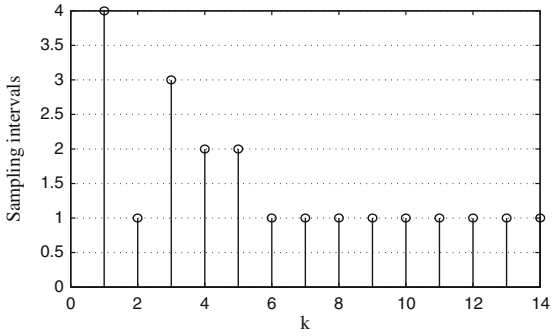

Fig. 3.4 Evolution of the sampling intervals

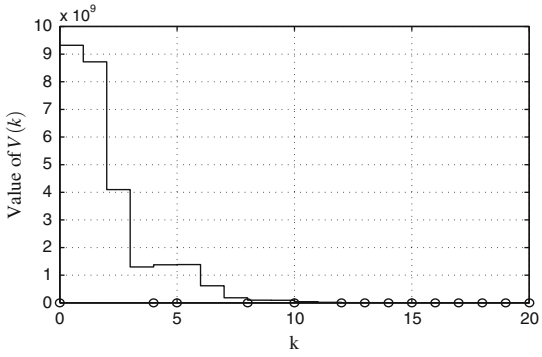

Fig. 3.5 Evolution of the Lyapunov function

3.5.2 Continuous System

Consider the following continuous LTI system:

$$\dot{x}(t) = \begin{bmatrix} 1 & 0.99 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + w(t),$$

where the disturbances $w(t)$ are supposed to verify $\|w(t)\|_\infty \leq 0.01$. The initial condition for the system and the controller is $x(0)^T = [10 \quad -5]$.

Choosing a sampling rate $T_{min} = 1$ s (which ensures the asymptotic stability of the system without disturbances), the following stabilizing controller has been designed:

$$K = \begin{bmatrix} -4.5263 & -4.4110 \end{bmatrix}.$$

The Lyapunov function is:

$$V(t) = x^T(t) \begin{bmatrix} 0.2161 & 0.1156 \\ 0.1156 & 0.1083 \end{bmatrix} x(t).$$

Assuming that the disturbances are zero, the evolution of the system $x(t)$ and the error $\delta(t)$ are shown in Fig. 3.6. As before, the asynchronous sampling times are indicated with a tiny circle. Only the first three sampling times are bigger than 1 s, since the error is quite small without disturbances.

Assume now that the disturbances are not zero. Figure 3.7 shows the response of the system with uniform disturbances. Again, the sampling rates are bigger than the basic rate only when the system is far form the equilibrium.

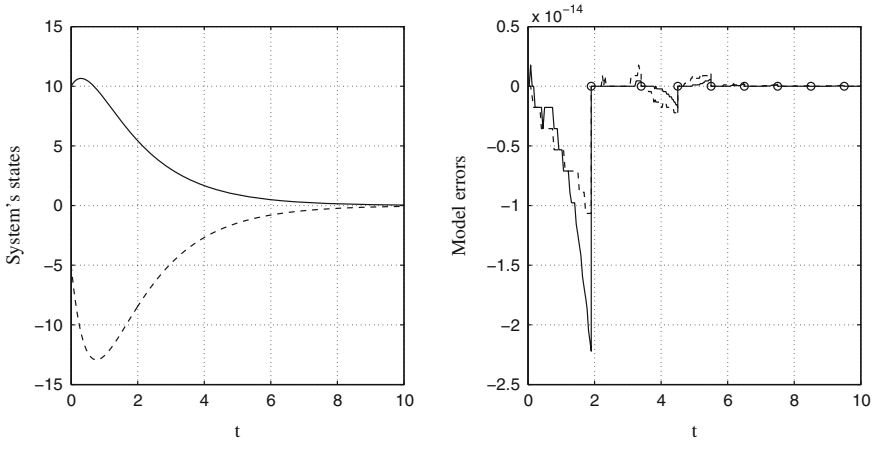

Fig. 3.6 Evolution without disturbances. *Solid* and *dashed lines* for different components

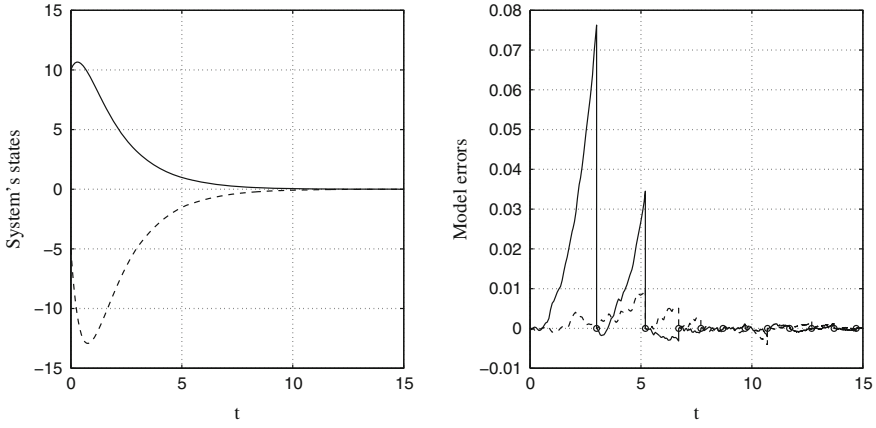

Fig. 3.7 Evolution with uniform disturbances

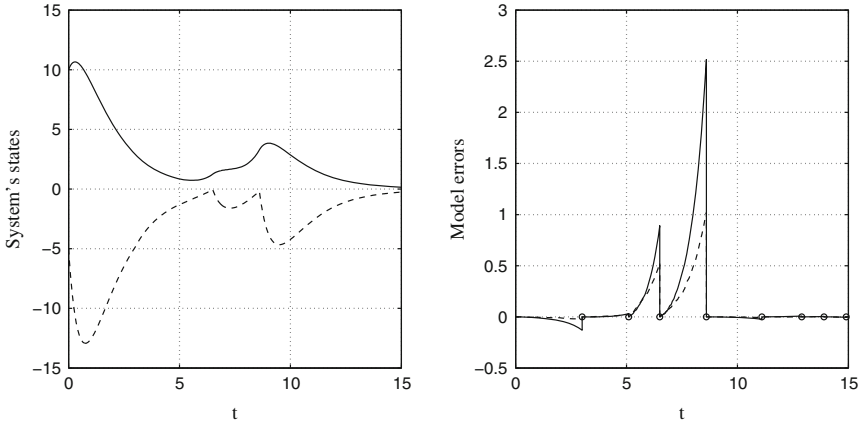

Fig. 3.8 Evolution with non-uniform disturbances

Finally, we test the method with the following description of the disturbances. At the beginning, uniform disturbances are applied. Next, the system evolves affected with large disturbances that get it out of the equilibrium. At the end, uniform disturbances are considered again. The asynchronous sampling periods for this case are shown in Fig. 3.8.

3.6 Conclusions

We have presented a novel model-based controller for networked systems, aimed at reducing the number of accesses to a shared network. It has been shown that, using predictions based on a nominal model of the system, adequate asynchronous sampling times can be found by solving several QP problems which take explicitly into account the disturbances of the model. We have considered both discrete and continuous-time controllers. The results have been demonstrated through simulation.

Chapter 4
Event-Triggered Anticipative Control over Packet-Based Networks

José Sánchez, María Guinaldo and Sebastián Dormido

4.1 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 that inherits 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* [80, 284] or *receding horizon control* [210].

To achieve this, a common approach is to use model-based control to predict 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 introduced in [18] in the context of teleoperation. Since then, other authors have exploited the principle of MPC in packet-based NCS [115, 132, 169, 210, 249].

J. Sánchez (✉) · M. Guinaldo · S. Dormido
Dpto. de Informática y Automática, Escuela Técnica Superior de Informática,
UNED, Madrid, Spain
e-mail: jsanchez@dia.uned.es

M. Guinaldo
e-mail: mguinaldo@dia.uned.es

S. Dormido
e-mail: sdormido@dia.uned.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_4

The influence of the model uncertainty in model-based NCS was studied in [182]. In [36], the constraints imposed by communication protocols on state measurement access are addressed. This work is extended to nonlinear systems in [85].

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 [190], but there is no optimization of any cost function. Anticipative controllers and the use of actuator buffers have been proposed for packet-based NCS in [68, 92] for different network architectures.

Whereas these approaches result 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 [68], a transmission protocol named as input difference transmission scheme (IDTS) 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 [92] 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 [100], where two frameworks are proposed, perturbed linear and piecewise linear systems, to achieve global exponential stability and ℓ_2 gain performance.

The proposed design in this chapter (Sect. 4.2) is not only 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 [100, 190, 210, 211, 284]. 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, as proved in Sect. 4.3.

Though the estimation of disturbances in NCSs where the controller is co-located with the actuator has been proposed in [154], Sect. 4.4 extends it to the remote controller architecture. Another contribution of this chapter is the design of event triggering for output measurements presented in Sect. 4.5. It 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.

Finally, Sect. 4.6 describes the experimental framework that is used to validate the analytical results as well as different experiments performed.

4.2 Event-Based Anticipative Control Design

4.2.1 Problem Statement

Let us consider the NCS scheme in which the controller node is neither co-located with the sensor nor with the actuator, i.e., the remote controller configuration (see Fig. 4.1). In this situation, clock synchronization between the controller and the plant is difficult to achieve in practice. Moreover, there may exist delays and data dropouts in the way from the sensor to the controller and from the controller to the actuator.

The plant dynamics is given by the discrete-time model

$$x(k+1) = Ax(k) + Bu(k) + w(k), \tag{4.1}$$

$$y(k) = Cx(k) + v(k), \tag{4.2}$$

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. The matrices A, B, and C are of the appropriate dimensions. We assume that the pairs (A, B) and (A, C) are controllable and observable, respectively. A discrete-time model implies that the measurements of time are given by multiple of a sampling period T_s, including delays, occurrence of events, etc. That is, for any instance of time t_k, there exists $k \in \mathbb{N}$ such that $t_k = kT_s$.

Example 4.1 A simple chronogram is shown in Fig. 4.2 to illustrate the phenomenon of delays and packet losses in a remote controller scheme. The system is sampled at discrete-time instances $k, k+1, \ldots$. 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.,

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

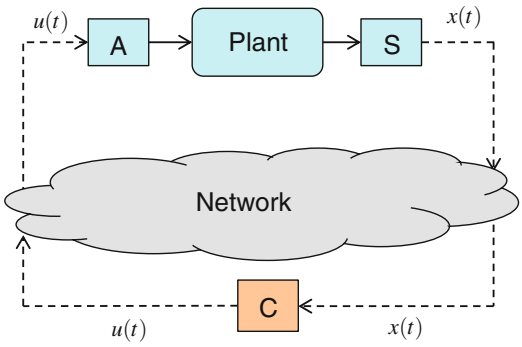

Fig. 4.1 Remote controller configuration. The sensor sends the state to the controller through the network, and the control signal is also transmitted to the actuator through the communication channel

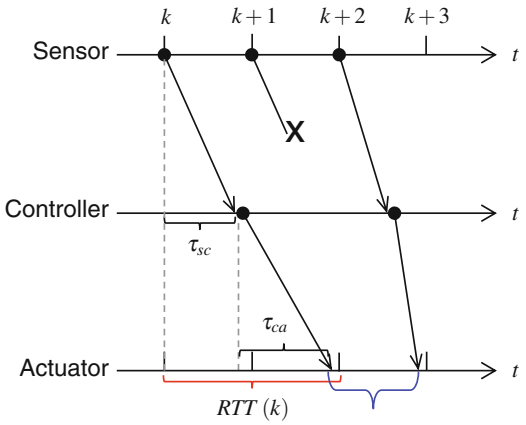

Fig. 4.2 Examples of delays and data dropouts

Data can also be dropped as depicted in Fig. 4.2 at $k+1$. As a consequence, the actuator does not receive updated control inputs in the interval marked in blue. Because the controller is not synchronized with the sensor and the actuator, the values of τ_{sc} and τ_{ca} cannot be measured. However, the communication of the closed-loop system can be characterized by the RTT that can be measured from the plant side.

The idea of the anticipative controller consists of estimating future states of the plant based on a model to deal with delays and packet losses. The model can also be used to generate predictions of the control signal. Perfect modeling is difficult to achieve in practice. Thus, we consider that the model of the plant is given by

$$\begin{aligned} x_m(k) &= A_m x_m(k) + B_m u(k) + w_m(k) \\ y_m &= C_m x_k, \end{aligned} \tag{4.3}$$

where $x_m(k), w_m(k) \in \mathbb{R}^n$ are the estimated state and disturbances, respectively. We denote the model uncertainty as $\bar{A} = A_m - A, \bar{B} = B_m - B$, and for simplicity $C = C_m$. Thus, instead of sending a single control value, the controller computes a sequence of control signals $\mathscr{U}$.

The anticipative controller works asynchronously, that is, it remains idle until it receives a new measurement. The sensor only transmits a *state packet* containing the measured state if an event is detected. The following trigger function is considered

$$F_e(\varepsilon(k)) = \|\varepsilon(k)\| - \delta, \tag{4.4}$$

where the error function is

$$\varepsilon(k) = x_m(k) - x(k). \tag{4.5}$$

The sequence of event times is determined recursively as

$$k_{\ell+1} = \inf\{k : k > k_\ell, F_e(\varepsilon) \geq 0\}, \quad \ell \in \mathbb{N}.$$

To sum up, the solution presented in the next section has the following features:

- It reduces the bandwidth usage thanks to the event-triggered transmission from the sensor to the controller. Since the controller works asynchronously, this causes a reduction in the backward channel too.
- It takes advantage of packet-based networks to send sequences instead of single values. These sequences are generated by the controller that uses a model of the plant (4.3), which might not match perfectly with the real dynamics of the plant (4.1) and (4.2).
- This information is sent to the plant, where it can be stored and used as convenience to apply the control input in the actuator.
- Delays and number of consecutive packet losses cannot be measured due to the lack of synchronization between the controller and the plant, but the RTT provides information about the quality of the channel, and this information can be used by the controller to estimate the state of the plant.

We first assume full state measurements, feedback control laws, and the non-existence of disturbances $w(k)$. Later on, in this chapter, we will deal with the problem of output measurements and disturbances, and how they can be estimated.

4.2.2 Control and Architecture Design

The proposed solution is depicted in Fig. 4.3 and has the following new components:

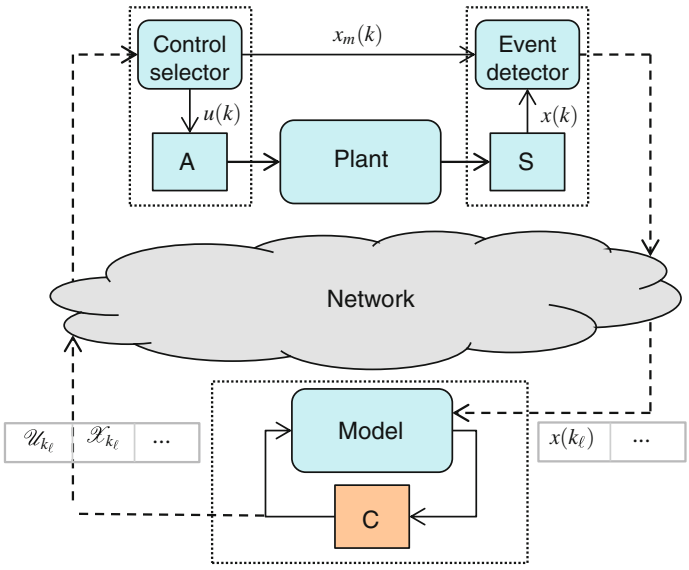

Fig. 4.3 Scheme of the proposed solution

The model-based controller performs several tasks. First, it processes the incoming packets, which include the following information:

- The measured state $x(k_\ell)$.
- The plant local time k_ℓ.
- A time stamp TS_u of the controller local time. TS_u allows to identify the control sequence $\mathscr{U}_{k_{\ell-h}}$, $h \in \mathbb{N}$ and $h \geq 1$, which was being applied at the time of the measurement of $x(k_\ell)$.
- An index i_u. If the computed control sequences have a size of N_u elements, i_u is the number of element of the sequence $\mathscr{U}_{k_{\ell-h}}$ which was being applied at the time of the measurement of $x(k_\ell)$.
- An update value of the minimum value of the RTT denoted by τ_{min}. $\tau_{min} \in \mathbb{N}$ gives the fastest transmission from the controller to the plant and the other way back:

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

With this information and the model (4.3), the first action is to estimate the state of the plant at time $k_\ell + \tau_{min}$, $x_m(k_\ell + \tau_{min})$ which is taken as the initial state to compute the next control sequence. Moreover, according to the definition of the trigger function (4.4), 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 4.1 The control sequence $\mathscr{U}_{k_\ell}$ is a set of N_u future control values that are calculated based on the system model (4.3) for the state packet containing $x(k_\ell)$ and received by the controller after the transmission through the network from the sensor to the controller.

Definition 4.2 The predicted states sequence $\mathscr{X}_{k_\ell}$ is a set of N_u future plant states predicted by the model (4.3). The jth element of $\mathscr{X}_{k_\ell}$ corresponds to the state given by the model (4.3) after applying the jth element of the control sequence $\mathscr{U}_{k_\ell}$.

Once $x_m(k_\ell + \tau_{min})$ is estimated based on the information received, the control input for this state is computed. For a state feedback control law with feedback gain K, it follows that

$$u(k_\ell + \tau_{min}) = Kx_m(k_\ell + \tau_{min})$$

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

$$x_m(k_\ell + \tau_{min} + 1) = A_m x_m(k_\ell + \tau_{min}) + B_m u(k_\ell + \tau_{min}) = (A_m + B_m K)x_m(k_\ell + \tau_{min}).$$

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

$$\mathscr{U}_{k_\ell}(j + 1) = u(k_\ell + \tau_{min} + j) = K(A_m + B_m K)^j x_m(k_\ell + \tau_{min}), \quad 0 \leq j \leq N_u - 1,$$

and the jth element of $\mathscr{X}_{k_\ell}$ is

$$\mathscr{X}_{k_\ell}(j) = (A_m + B_m K)^j x_m(k_\ell + \tau_{min}).$$

The control selector 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 and predicted states sequences are buffered, and the rest of the packets are discarded because they contain obsolete values calculated with prior states of the plant.

The previous section pointed out that the first element of the control sequence for the sampling time k_ℓ is calculated based on an estimated state $x_m(k_\ell+\tau_{min})$. However, the time between the measurement of the state $x(k_\ell)$ and the reception of the control sequence $\mathscr{U}_{k_\ell}$ will generally be greater than τ_{min}. Let us denote this elapsed time as $\tau(k_\ell)$. The value of $\tau(k_\ell)$ is measured by subtracting k_ℓ 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 $\mathscr{U}_{k_\ell}$ should be discarded because they are obsolete values. This value is denoted as i_0 ($i_0 = \tau(k_\ell) - \tau_{min}$). The first i_0 elements of $\mathscr{U}_{k_\ell}$ are then discarded, and the $i_0 + 1$ element is the first element to apply. For the same reason the first i_0 elements of $\mathscr{X}_{k_\ell}$ are also discarded.

The event detector The code executed by this component is shown in Algorithm 4.1. 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 en event is triggered, the module delivers $x(k_\ell)$ and the index value as outputs.

Algorithm 4.1 Event-detection algorithm.

Input: $\mathscr{U}_{k_\ell}$, $\mathscr{X}_{k_\ell}$, i_0 with $k_\ell < k$
Output: $x(k_{\ell+1})$, $i_0 + j$
1: $j := 0$
2: $\varepsilon(k) := \mathbf{0}$
3: $F_e(\varepsilon(k)) := -1$
4: **while** $i_0 + j < \bar{N}_u$ **and** $F_e(\varepsilon(k)) < 0$ **do**
5: $\quad j := j + 1$
6: $\quad$ Apply $\mathscr{U}_{k_\ell}(i_0 + j)$
7: $\quad$ Measure $x(k)$
8: $\quad x_m(k) := \mathscr{X}_{k_\ell}(i_0 + j)$
9: $\quad \varepsilon(k) := x_m(k) - x(k)$
10: $\quad$ Compute $F_e(\varepsilon(k))$
11: **end while**

Note that an event is detected when either $F_e(\varepsilon(k))$ crosses zero or $i_0 + j$ equals $\bar{N}_u$, where $\bar{N}_u$ is $\bar{N}_u = N_u - \tau_{max}$, 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.

4.3 Stability Analysis

The event-based policy (4.4) allows to reduce the communication in the control loop, but the price to pay is that asymptotic stability is no guaranteed, but the GUUB of the state can be proved. If disturbances are equal to zero and full state measurements are available, it follows that

$$x(k+1) = Ax(k) + BKx_m(k), \tag{4.6}$$

since the anticipative control defines the control law as the feedback of the predicted state for any sampling time k. Equation (4.6) can be rewritten in terms of the error (4.5) as

$$x(k+1) = (A + BK)x(k) + BK\varepsilon(k). \tag{4.7}$$

Note that an event is triggered whenever $\|\varepsilon(k)\| \geq \delta$. 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 4.1 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:

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

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 $\tau(k_\ell)$ or simply as τ.

Assumption 4.1 constrains the model uncertainty and/or the maximum allowable number of sampling periods the system (4.2) 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 4.1.

Proposition 4.1 *If Assumption 4.1 holds, the error defined as (4.5) is bounded by*

$$\|\varepsilon(k)\| \leq 2\delta. \tag{4.8}$$

Proof From Assumption 4.1 it follows that

$$\|\varepsilon(k_\ell + \tau) - \varepsilon(k_\ell)\| < \delta, \forall \tau \leq \tau_{max},$$

since no event is detected in this interval.

When the sampling is fast enough, the error at the event detection is $\|\varepsilon(k_\ell)\| \approx \delta$. Thus, assuming that this approximation is exact

$$\|\varepsilon(k_\ell + \tau)\| \leq \|\varepsilon(k_\ell)\| + \|\varepsilon(k_\ell + \tau) - \varepsilon(k_\ell)\| \leq 2\delta,$$

which concludes the proof.

Let us denote $A_K = A + BK$ to simplify the notation. Because A_K is assumed to be Hurwitz, there exists a $P = P^T > 0$ such that

$$A_K^T P A_K - P = -Q, \tag{4.9}$$

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

$$V(x) = x^T P x. \tag{4.10}$$

The main result of the section is presented next. The error $\varepsilon(k)$ 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 4.1 *If Assumption 4.1 holds, the state of the system (4.7) when the remote controller runs according to the model (4.3) and the event detector is defined by (4.4), is GUUB with bound*

$$\|x\| \leq \sqrt{\frac{\lambda_{max}(P)}{\lambda_{min}(P)}}(\sigma \|A_K\| + \|BK\|)2\delta, \tag{4.11}$$

where

$$\sigma = \frac{\|K^T B^T P A_K\| + \sqrt{\|K^T B^T P A_K\|^2 + \lambda_{min}(Q)\|K^T B^T P B K\|}}{\lambda_{min}(Q)}, \tag{4.12}$$

$\lambda_{min}(P)$ *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 (4.10) for (4.7) is

$$\begin{aligned}\Delta V(k) &= x^T(k+1)Px(k+1) - x^T(k)Px(k)\\ &= \big(A_K x(k) + BK\varepsilon(k)\big)^T P\big(A_K x(k) + BK\varepsilon(k)\big) - x^T(k)Px(k)\\ &= -x^T(k)Qx(k) + 2\varepsilon^T(k)(BK)^T PA_K x(k) + \varepsilon^T(k)(BK)^T PBK\varepsilon(k),\end{aligned}$$

which is upper bounded by

$$\begin{aligned}\Delta V(k) \leq &-\lambda_{min}(Q)\|x(k)\|^2 + 2\|(BK)^T PA_K\|\|\varepsilon(k)\|\|x(k)\|\\ &+ \|(BK)^T PBK\|\|\varepsilon(k)\|^2.\end{aligned} \tag{4.13}$$

The right hand side of (4.13) is an algebraic second-order equation in $\|x(k)\|$ such that the Lyapunov function decreases whenever

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

where σ is given in (4.12).

According to Proposition 4.1, the error at any time k is bounded by 2δ. Hence, $\Delta V < 0$ in the region $\|x(k)\| > 2\delta\sigma$. 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 (4.7), it follows that

$$\|x(k^*+1)\| \leq (\sigma\|A_K\| + \|BK\|)2\delta.$$

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 (4.11) $\forall k \geq k^*$, and this concludes the proof.

4.3.1 Analysis of the Maximum RTT and the Model Uncertainties

Assumption 4.1 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_ℓ. The error at the next sampling time is

$$\begin{aligned}\varepsilon(k_\ell+1) &= x_m(k_\ell+1) - x(k_\ell+1) = (A_m + B_m K)x_m(k_\ell) - \big(Ax(k_\ell) + BKx_m(k_\ell)\big)\\ &= (\bar{A} + \bar{B}K)x(k_\ell) + (A_m + \bar{B}K)\varepsilon(k_\ell),\end{aligned} \tag{4.14}$$

where $\bar{A} = A_m - A$, $\bar{B} = B_m - B$ represent the model uncertainty. Let us also define $A_{mK} = A_m + B_m K$.

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

$$\begin{aligned} x(k_\ell + \tau) &= A^\tau x(k_\ell) + \left(\sum_{j=1}^{\tau} A^{\tau - j} B K A_{mK}^j \right) x_m(k_\ell) \\ &= \left(A^\tau + \sum_{j=1}^{\tau} A^{\tau - j} B K A_{mK}^j \right) x(k_\ell) + \left(\sum_{j=1}^{\tau} A^{\tau - j} B K A_{mK}^j \right) \varepsilon(k_\ell). \end{aligned} \tag{4.15}$$

The error at $k + \tau$ is $\varepsilon(k_\ell + \tau) = x_m(k_\ell + \tau) - x(k_\ell + \tau)$, thus

$$\varepsilon(k_\ell + \tau) = A_{mK}^\tau x_m(k_\ell) - x(k_\ell + \tau) = A_{mK}^\tau \varepsilon(k_\ell) + A_{mK}^\tau x(k_\ell) - x(k_\ell + \tau),$$

where $x(k_\ell + \tau)$ is given in (4.15).

The maximum RTT can be derived imposing that

$$\|\varepsilon(k_\ell + \tau_{max}) - \varepsilon(k_\ell)\| < \delta,$$

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 $x(k_\ell)$. It is not possible to derive an analytical solution for it, but the feasibility of the solution requires a bound for $x(k_\ell)$ $\forall k_\ell$. Its existence is guaranteed from the results in Theorem 4.1.

However, it is possible to derive an analytical solution when the model uncertainty can be approximated to zero so that $\bar{A} \approx \mathbf{0}$, $\bar{B} \approx \mathbf{0}$. In this case, the evolution of $\varepsilon(k)$ in (4.14) is approximated by $\varepsilon(k_\ell + 1) \approx A_m \varepsilon(k_\ell)$. Thus, after τ sampling periods it is given by

$$\varepsilon(k_\ell + \tau) \approx A_m^\tau \varepsilon(k_\ell) \approx A^\tau \varepsilon(k_\ell). \tag{4.16}$$

Thus, according to Proposition 4.1, it holds that

$$\|\varepsilon(k_\ell + \tau_{max}) - \varepsilon(k_\ell)\| = \|(A^{\tau_{max}} - I)\varepsilon(k_\ell)\| < \delta.$$

Since $\|\varepsilon(k_\ell)\| \approx \delta$, an upper bound for the maximum allowable RTT will be the solution of

$$\|A^{\tau_{max}} - I\| < 1, \ \tau_{max} \in \mathbb{N}, \tag{4.17}$$

which is independent of the value of δ.

Example 4.2 Assume that the scalar system

$$\dot{x}(t) = ax(t) + bu(t), \quad a, b \in \mathbb{R}, \tag{4.18}$$

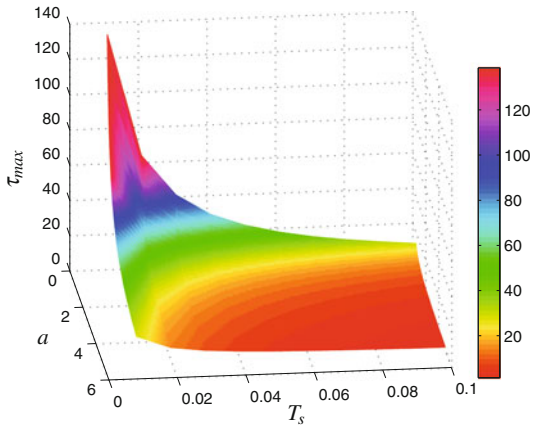

Fig. 4.4 Surface defined by (4.19)

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 δ. Assume that there is no model uncertainty in the anticipative controller. Let us compute the maximum allowable RTT for the system (4.18).

It holds that $A = e^{aT_s}$. Thus, according to (4.17), 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,

$$\tau_{max} < \frac{\log(2)}{aT_s}. \tag{4.19}$$

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 $T_s = 50\,\text{ms}$, $\frac{\log(2)}{aT_s} = 13.86$ and the maximum RTT is $\tau_{max} = 13$ sampling periods. In Fig. 4.4 the surface that bounds the region defined by (4.19) 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 $T_s \in [10, 100]\,\text{ms}$.

4.3.2 Analysis of the Error Bounds

The analysis has shown that the system is GUUB when Assumption 4.1 holds, and consequently, the error is upper bounded by 2δ (see Proposition 4.1). 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 $\varepsilon(k) = x_m(k) - x(k)$ 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. 1.9a), 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 it 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 $k = k_\ell$. According to (4.15), the state at $k_\ell + \tau_{min}$ is

$$x(k_\ell + \tau_{min}) = \left(A^{\tau_{min}} + \sum_{j=1}^{\tau_{min}} A^{\tau_{min}-j} B K A_{mK}^{j} \right) x(k_\ell) + \left(\sum_{j=1}^{\tau_{min}} A^{\tau_{min}-j} B K A_{mK}^{j} \right) \varepsilon(k_\ell).$$

While the prediction that the model gives is

$$x_m(k_\ell + \tau_{min}) = \left(A_m^{\tau_{min}} + \sum_{j=1}^{\tau_{min}} A_m^{\tau_{min}-j} B_m K \, (A_{mK})^{j} \right) x(k_\ell) + \left(\sum_{j=1}^{\tau_{min}} A_m^{\tau_{min}-j} B_m K A_{mK}^{j} \right) \varepsilon(k_\ell).$$

Then, it follows that the error is

$$\varepsilon(k_\ell + \tau_{min}) = \left(A_m^{\tau_{min}} - A^{\tau_{min}} + \sum_{j=1}^{\tau_{min}} (A_m^{\tau_{min}-j} B_m - A^{\tau_{min}-j} B) K \, (A_{mK})^{j} \right) x(k_\ell) + \left(\sum_{j=1}^{\tau_{min}} (A_m^{\tau_{min}-j} B_m - A^{\tau_{min}-j} B) K A_{mK}^{j} \right) \varepsilon(k_\ell). \tag{4.20}$$

Note that the right hand side of (4.20) is zero if $A = A_m$ and $B = B_m$, and different from zero otherwise. Moreover, it depends on the state $x(k_\ell)$.

4.4 Disturbance Estimator

According to (4.1), 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 (4.3). 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 [154], 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 state matrix A of the continuous-time model 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 the previous constraint, and considers model mismatch. The following assumptions hold henceforth:

- The system dynamics is given by (4.1) and (4.2).
- The model estimates future states of the plant according to

$$x_m(k+1) = A_m x_m(k) + B_m u(k) + w_m(k), \tag{4.21}$$

  where w_m 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_ℓ, the disturbance is estimated before computing the next control sequence $\mathscr{U}_{k_\ell}$, and held constant in the next steps.

Hence, the disturbance estimator is a new element to include in the controller side.

According to (4.1) and (4.21), the error dynamics is given by

$$\begin{aligned}\varepsilon(k+1) = x_m(k+1) - x(k+1) &= A_m x_m(k) - Ax(k) \\ &+ (B_m - B)u(k) + (w_m(k) - w(k)),\end{aligned} \tag{4.22}$$

The disturbance $w(k)$ could be calculated if the rest of the terms of (4.22) were known. However, the model mismatch is unknown. Therefore, if the approximation $\bar{A} \approx \mathbf{0}$, $\bar{B} \approx \mathbf{0}$ is taken, the value of $w(k)$ can be computed at the next sampling time $k+1$ (after measuring $x(k+1)$) as

$$\begin{aligned}w_m(k+1) &= A_m(x_m(k) - x(k)) + w_m(k) - \varepsilon(k+1) \\ &= w_m(k) + A_m\varepsilon(k) - \varepsilon(k+1).\end{aligned} \tag{4.23}$$

Let us denote h 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 $k+h$ can be approximated to $\varepsilon(k+h) \approx A_m^h \varepsilon(k)$ (see (4.16)). This approximation turns into

$$\varepsilon(k+h) = A_m^h \varepsilon(k) + \sum_{j=0}^{h-1} A_m^j (w_m(k+j) - w(k+j))$$

when disturbances are included in the model.

Because $w_m(k)$ is assumed to be held constant in this interval, the disturbance can be estimated at time $k+h$ as

$$w_m(k+h) = w_m(k) + \left(\sum_{j=0}^{h-1} A_m^j\right)^{-1} (A_m^h \varepsilon(k) - \varepsilon(k+h)). \tag{4.24}$$

Note that $\varepsilon(k)$ in (4.24), 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 plant to the controller. Thus, the *state packets* must include the following information:

- The measurement which triggered the event $x(k_\ell)$.
- A time stamp TS_u of the controller local time. TS_u allows to identify the control sequence $\mathscr{U}_{k_\ell - h}$, $h \geq 1$, which was being applied at the time of the measurement of $x(k_\ell)$.
- The index i_u which is the number of element of the sequence $\mathscr{U}_{k_\ell - h}$ which was being applied at the time of the measurement of $x(k_\ell)$.
- The error between the predicted state by the model (4.21) and the measured state after receiving $\mathscr{U}_{k_\ell - h}$. If the number of sampling periods between this instant of time and the detection of the event at time k_ℓ is n_ℓ, hence this value is $\varepsilon(k_\ell - n_\ell)$.
- The error $\varepsilon(k_\ell)$ when the event is detected.
- The number of sampling periods n_ℓ.

According to this, the code executed by the controller is illustrated in Algorithm 4.2. Note that once $w_m(k_\ell)$ is estimated, it is used in the estimation of $x_m(k_\ell + \tau_{min})$ and the computation of $\mathscr{U}_{k_\ell}$, $\mathscr{X}_{k_\ell}$. The function *getFromLookUpTable* that receives as inputs TS_u, i_u, and τ_{min} enables to identify the control inputs that are being applied in the plant.

Algorithm 4.2 Code executed by the controller for disturbance estimation.

Input: $x(k_\ell)$, TS_u, i_u, $\varepsilon(k_\ell - n_\ell)$, $\varepsilon(k_\ell)$, n_ℓ
Output: $\mathscr{U}_{k_\ell}$, $\mathscr{X}_{k_\ell}$
$w_m(k_\ell - n_\ell) :=$ getFromLookupTable(TS_u)
$w_m(k_\ell) := w_m(k_\ell - n_\ell) + (\sum_{j=0}^{n_\ell - 1} A_m^j)^{-1}(A_m^{n_\ell}\varepsilon(k_\ell - n_\ell) - \varepsilon(k_\ell))$
$[u(k_\ell) \ldots u(k_\ell + \tau_{min})] :=$ getFromLookupTable(TS_u, i_u, τ_{min})
$x_m(k) := x(k_\ell)$
for $j = 1 \rightarrow \tau_{min}$ **do**
    $x_m(k+1) := A_m x_m(k) + B_m u(k_\ell + j - 1) + w_m(k_\ell)$
    $x_m(k) := x_m(k+1)$
end for
$x_m(k_\ell + \tau_{min}) := x_m(k)$
$\mathscr{U}_{k_\ell}(1) = K x_m(k_\ell + \tau_{min})$
$\mathscr{X}_{k_\ell}(1) = (A_m + B_m K) x_m(k_\ell + \tau_{min}) + w_m(k_\ell)$
for $j = 2 \rightarrow N_u$ **do**
    $\mathscr{U}_{k_\ell}(j) = K \mathscr{X}_{k_\ell}(j-1)$
    $\mathscr{X}_{k_\ell}(j) = (A_m + B_m K)\mathscr{X}_{k_\ell}(j-1) + w_m(k_\ell)$
end for

4.4.1 Stability Analysis

Stability results can be derived when disturbances affect the system in a similar way as Theorem 4.1 if bounded disturbances are assumed:

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

In this case, it is proven that the Lyapunov function (4.10) satisfying (4.9) decreases to reach a region whose size depends on the bound of the error $\|\varepsilon(k)\|$ and the disturbances $\|w(k)\|$.

Before stating the main result of this section, let us rewrite (4.1) in terms of $\varepsilon(k)$ as

$$x(k+1) = A_K x(k) + BK\varepsilon(k) + w(k). \tag{4.25}$$

Theorem 4.2 *If Assumption 4.1 holds and the disturbances are bounded by* $\|w(k)\| \leq w_{max}$*, the state of the system (4.25) when the remote controller runs according to the model (4.21) and the event detector is defined by (4.4), is GUUB with bound*

$$\|x\| \leq \sqrt{\frac{\lambda_{max}(P)}{\lambda_{min}(P)}}\left(\|A_K\|\sigma_w + \|BK\|2\delta + w_{max}\right), \tag{4.26}$$

where

$$\sigma_w = \frac{\sigma_b + \sqrt{\sigma_b^2 + 4\sigma_a\sigma_c}}{2\sigma_a} \tag{4.27}$$

$$\sigma_a = \lambda_{min}(Q) \tag{4.28}$$

$$\sigma_b = \|(BK)^T PA_K\|2\delta + \|PA_K\|w_{max} \tag{4.29}$$

$$\sigma_c = \|(BK)^T PBK\|4\delta^2 + 4\|PBK\|w_{max}\delta + \lambda_{max}(P)w_{max}^2. \tag{4.30}$$

Proof The proof can be found in Appendix A.

Example 4.3 In this example a system modeled as a double integrator is considered, and sampled with $T_s = 5$ ms:

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

The trigger function is defined with $\delta = 0.05$. The model uncertainty is known to be $\|\bar{A}\| < 0.2\|A\|$, $\|\bar{B}\| < 0.2\|B\|$. 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 4.5 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

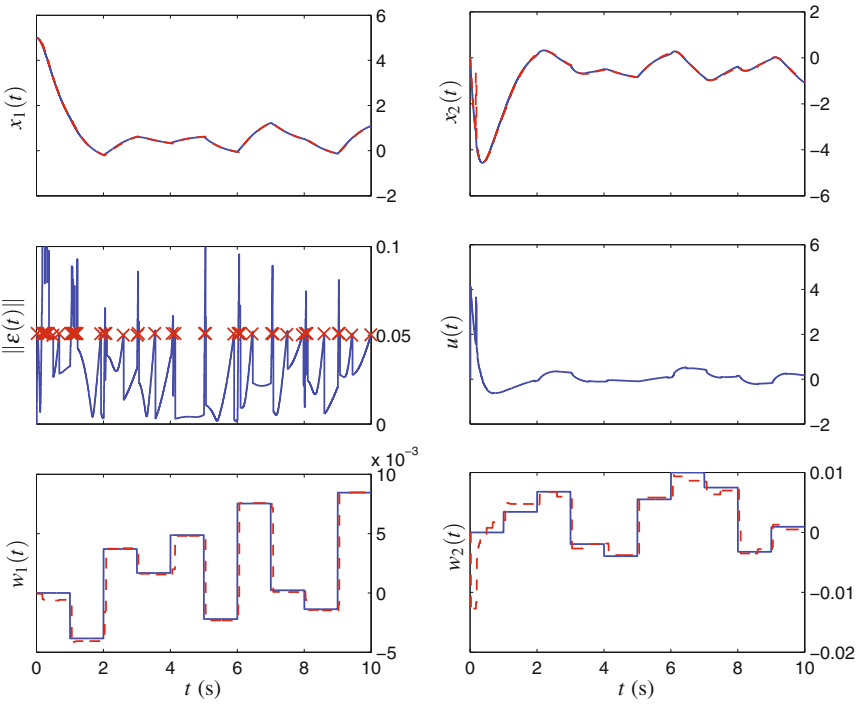

Fig. 4.5 Disturbances estimation. The estimated values are represented by the *dotted line*, and the actual values by the *solid line*

(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 δ.

4.5 Output-Based Event-Triggered Control

This section presents a method to apply 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 [101]. One may think that an intuitive solution is to redefine the error as

$$\varepsilon_y(k) = y_m(k) - y(k), \tag{4.31}$$

define a trigger function such that $\|\varepsilon_y(k_\ell)\| \approx \delta$, and extend the analysis to derive $\|\varepsilon_y(k)\| \leq 2\delta$. However, the boundedness of $\varepsilon_y(k)$ does not imply the boundedness

of $x_m(k) - x(k)$, 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 [141] an state observer is proposed. The error function is replaced by $x_m(k) - \hat{x}(k)$, where $\hat{x}(k)$ is the observed state. The analysis shows that the property of GUUB is preserved. In [143], 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 [56]. 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 [135], where a LTI continuous-time controller is emulated.

The first direction would make easier to extend the design of Sect. 4.2 and the stability results of Sect. 4.3. However, a computational cost is required in the process side to observe the state, and it has been assumed that its computational capacity is very limited.

The design proposed in this section 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 (4.31), and the trigger function for output measurement is

$$F_e(\varepsilon_y(k)) = \|\varepsilon_y(k)\| - \delta_y. \tag{4.32}$$

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

$$x_c(k+1) = A_c x_c(k) + B_c y_m(k) \tag{4.33}$$

$$u(k) = C_c x_c(k) + D_c y_m(k), \tag{4.34}$$

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 $y_m(k)$ is replaced by $y(k)$. We further assume that the pair (A, C) is observable and that a model is available and it is given by (A_m, B_m), and $C_m = C$. Finally, disturbances affecting the system (4.2) are not considered for simplicity. However, the measurement noise $v(k)$ might not zero but bounded by v_{max}.

All the above said, some of the components described in Sect. 4.2.2 have to be adapted to the output measurement scenario, as depicted in Fig. 4.6.

- **The control selector** will provide the estimated output $y_m(k)$ to the event detector, instead of $x_m(k)$, for each sampling time k.
- **The event detector** uses the trigger function (4.32) instead of (4.4). Moreover, it collects the so-called *output vector* $\overrightarrow{y}$, which is the measured outputs $y(k)$ at each

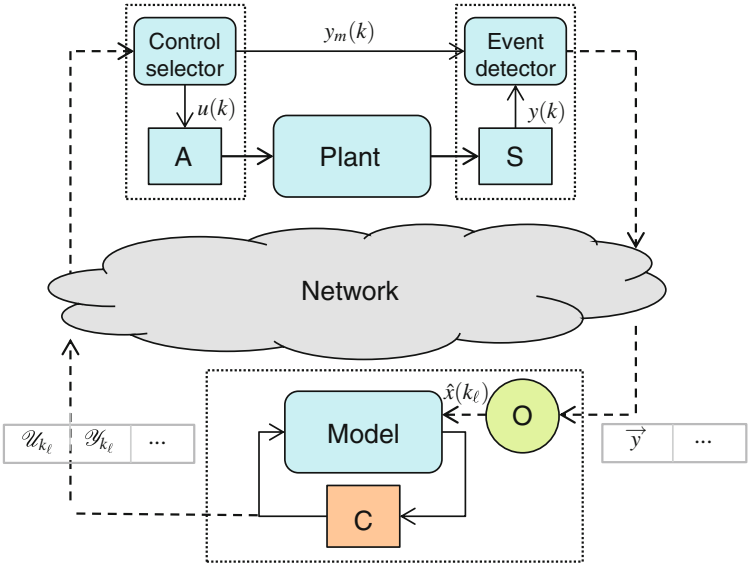

Fig. 4.6 Scheme of the proposed solution

sampling time k between the reception of a control packet and the detection of a new event. This information will be used in the controller.

- **The model-based controller** The new basis controller is given by (4.33) and (4.34). Hence, it receives from the model the predicted output of the plant $y_m(k)$, it computes its next internal state according to (4.33) and the control input as (4.34). The model needs to compute an estimation of the state of the plant $x_m(k)$ according to (4.3), but only $y_m(k) = Cx_m(k)$ is used to compute $u(k)$. Moreover, a sequence of predicted outputs is generated as $\mathscr{Y}_{k_\ell} = C\mathscr{X}_{k_\ell}$, where $\mathscr{X}_{k_\ell}$ is generated as in the case of full state measurement. However, since $x(k_\ell)$ is no longer available, it is estimated by a state observer using the information in $\overrightarrow{y}$, generating future states of the plant after that. We next describe this more in detail.

A state observer A Luenberger observer of the form

$$\hat{x}(k+1) = (A_m - MC)\hat{x}(k) + B_m u(k) + My(k), \quad \hat{x}(0) = \hat{x}_0$$
$$\hat{y}(k) = C\hat{x}(k)$$

is used to obtain the state $x(k)$, being $(A_m - MC)$ Hurwitz. Anytime a new state packet is received at the controller side, the code of Algorithm 4.3 is executed. The length of $\overrightarrow{y}$ 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 TS_u and i_u (received with the state packet as well), we can determine the control inputs applied at the actuator during this period

Algorithm 4.3 Luenberger observer state estimation.

Input: $\overrightarrow{y}$, TS_u, i_u
Output: $\hat{x}(k_\ell)$
1: $n_y := \dim(\overrightarrow{y})$
2: $[u(k_\ell - n_\ell - 1) \ldots u(k_\ell)] :=$ getFromLookupTable(TS_u, i_u, n_y)
3: **for** $j = 1 \rightarrow n_y$ **do**
4: $\quad y(k_\ell - n_y + j - 1) := \overrightarrow{y}(j)$
5: $\quad \hat{x}(k_\ell - n_y + j) := (A_m - MC)\hat{x}(k_\ell - n_y + j - 1) + B_m u(k_\ell - n_y + j - 1) + My(k_\ell - n_y + j - 1)$
6: $\quad \hat{x}(k_\ell - n_y + j - 1) := \hat{x}(k_\ell - n_y + j)$
7: **end for**

by checking them in a look-up table. Then, the Luenberger observer estimates the plant state $x(k_\ell)$ corresponding to the last output measurement $y(k_\ell)$, which is the last element of the output vector $\overrightarrow{y}$.

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

4.5.1 Stability Analysis

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

$$x(k+1) = Ax(k) + Bu(k) \tag{4.35}$$
$$y(k) = Cx(k) + v(k) \tag{4.36}$$
$$x_c(k+1) = A_c x_c(k) + B_c y_m(k) \tag{4.37}$$
$$u(k) = C_c x_c(k) + D_c y_m(k), \tag{4.38}$$

with the error defined as (4.31) and the trigger function (4.32). This can be rewritten as

$$\begin{pmatrix} x(k+1) \\ x_c(k+1) \end{pmatrix} = \begin{pmatrix} A + BD_cC & BC_c \\ B_cC & A_c \end{pmatrix} \begin{pmatrix} x(k) \\ x_c(k) \end{pmatrix} + \begin{pmatrix} BD_c \\ B_c \end{pmatrix} \big(\varepsilon_y(k) + v(k)\big).$$

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

$$A_{CL} = \begin{pmatrix} A + BD_cC & BC_c \\ B_cC & A_c \end{pmatrix}, \tag{4.39}$$

$$F = \begin{pmatrix} BD_c \\ B_c \end{pmatrix}. \tag{4.40}$$

Thus, the closed-loop system-controller dynamics is

$$\xi(k+1) = A_{CL}\xi(k) + F\varepsilon_y(k) + Fv(k). \tag{4.41}$$

Equation (4.41) 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 (4.39)) is Hurwitz, there exists a $P = P^T$ such that

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

We define the Lyapunov function

$$V(\xi) = \xi^T(k) P \xi(k). \tag{4.42}$$

The unperturbed system $\xi(k+1) = A_{CL}\xi(k)$ converges asymptotically to the origin. Nevertheless, when event triggering (4.32) is considered and measurements are affected by noise, only *GUUB* of $\xi(k)$ can be achieved.

Let us consider that Assumption 4.1 holds. The result of Proposition 4.1 can be extended to the error $\varepsilon_y(k)$ straightforward, so that

$$\|\varepsilon_y(k)\| \leq 2\delta_y, \quad \forall k.$$

The following theorem is equivalent to Theorem 4.1 but for output measurement and the proposed controller design. The error $\varepsilon_y(k)$ and the measurement noise $v(k)$ perturb the system. The error $\varepsilon_y(k)$ is a contribution of both the model uncertainties and the network imperfections, whereas $v(k)$ is inherited from the measurement itself.

Theorem 4.3 *If Assumption 4.1 holds, the augmented state $\xi(k)$ of the system-controller (4.41), when the event detector is defined by (4.32), and the measurement noise is bounded $\|v(k)\| \leq v_{max}$, is GUUB with bound*

$$\|\xi\| \leq \sqrt{\frac{\lambda_{max}(P)}{\lambda_{min}(P)}}(\sigma_\xi \|A_{CL}\| + \|F\|)(2\delta_y + v_{max}), \tag{4.43}$$

where

$$\sigma_\xi = \frac{\|F^T P A_{CL}\| + \sqrt{\|F^T P A_{CL}\|^2 + \lambda_{min}(Q)\|F^T P F\|}}{\lambda_{min}(Q)}. \tag{4.44}$$

Proof The proof can be found in Appendix A.

4.5.2 PI Anticipative Control Design

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 [215]. Most of such controllers are Proportional Integral (PI), since the derivative part is generally not used in practice [215]. 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 $e_{sp}(k)$ is defined as $e_{sp}(k) = y_{sp} - y(k)$, where y_{sp} is this reference signal.

The state-space representation of a PI controller is given in Chap. 2. In discrete time, $A_c = 1$, $B_c = -\frac{k_p T_s}{T_i}$, $C_c = 1$, and $D_c = -k_p$, where k_p is the proportional gain, T_i is the integral time, and T_s the sampling period. This allows to derive (4.41) when the basis LTI controller is PI and for set-point tracking y_{sp} (weighted by a factor b) as

$$\xi(k+1) = \begin{pmatrix} A - k_p BC & B \\ -\frac{k_p T_s}{T_i} C & 1 \end{pmatrix} \xi(k) + \begin{pmatrix} -k_p B \\ -\frac{k_p T_s}{T_i} \end{pmatrix} \big(\varepsilon_y(k) + v(k)\big) + \begin{pmatrix} k_p bB \\ \frac{k_p T_s}{T_i} \end{pmatrix} y_{sp}. \tag{4.45}$$

The output is

$$y(k) = \begin{pmatrix} C & 0 \end{pmatrix} \xi(k) + v(k),$$

and the control input

$$u(k) = \begin{pmatrix} -k_p C & 1 \end{pmatrix} \xi(k) + k_p \big(y_{sp} - \varepsilon_y(k) - v(k)\big).$$

4.5.2.1 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 (4.39) and (4.40).

A model version of (4.45) can be defined to deduce the control and the predicted output sequences. Thus,

$$\hat{\xi}(k+1) = A_{m,CL} \xi_m(k) + F_{m,b} y_{sp}, \tag{4.46}$$

where

$$A_{m,CL} = \begin{pmatrix} A_m - k_p B_m C & B_m \\ -\frac{k_p T_s}{T_i} C & 1 \end{pmatrix} \quad F_{m,b} = \begin{pmatrix} k_p b B_m \\ \frac{k_p T_s}{T_i} \end{pmatrix}. \tag{4.47}$$

Note that $x_{m,C} = x_C$, but the compact form of (4.46) simplifies the expressions. After estimating $x_m(k_\ell + \tau_{min})$ and therefore, $\xi_m(k_\ell + \tau_{min})$, the j element of the predicted

output sequence $\mathscr{Y}_{k_\ell}$, i.e., $\begin{pmatrix} C & 0 \end{pmatrix}\xi_m(k_\ell + \tau_{min} + j)$, is

$$\mathscr{Y}_{k_\ell}(j) = \begin{pmatrix} C & 0 \end{pmatrix}\left[A^j_{m,CL}\xi_m(k_\ell + \tau_{min}) + \sum_{l=0}^{j-1} A^l_{m,CL}F_{m,b}y_{sp}\right], \tag{4.48}$$

assuming that the set-point value remains constant. And the $j+1$ element of the control sequence $\mathscr{U}_{k_\ell}$, i.e., $\begin{pmatrix} -k_pC & 1 \end{pmatrix}\xi_m(k_\ell + \tau_{min} + j) + k_pby_{sp}$, is

$$\mathscr{U}_{k_\ell}(j+1) = \begin{pmatrix} -k_pC & 1 \end{pmatrix}\left[A^j_{m,CL}\xi_m(k_\ell + \tau_{min}) + \sum_{l=0}^{j-1} A^l_{m,CL}F_{m,b}y_{sp}\right] + k_pby_{sp}. \tag{4.49}$$

4.6 Experimental Results

4.6.1 Experimental Framework

In order to evaluate the proposed design, we have made use of the infrastructure of remote laboratories located in the Automatic Control Laboratory of UNED in Madrid. These laboratories are used by students to conduct their experiments remotely thanks to the web-based environment developed [247], which is based on a *client–server* architecture [133].

Traditionally, in the client–server architecture, controller and process are at the server-side (the real-time control loop) and the user gets remotely the state of the plant, modifies the parameters of the controller, and observes how the plant reacts to them (the asynchronous supervision loop) [248].

Hence, the implementation changes to adapt this environment to work using the remote anticipative controller. An scheme of the experimental framework is shown in Fig. 4.7. The event detector and control selector are hosted at the server side, which is connected to the Internet. The remote controller is at the client side, which also provide an interface for the user. The communication with the process side can be wired or wireless. We were particularly interested in testing the performance of the anticipative controller in processes with fast and/or unstable dynamics since they introduce a more challenging environment. On the one hand, small-time constant processes (10–100 ms) need lower sampling periods than the average of the measured RTT. If the system has not fast dynamics, the presence of the network could be overlooked, since the characteristic times of those processes are several orders higher than the network delay. On the other hand, the control of unstable processes has to meet hard requirements. Delays and data dropouts can easily unstabilize the control loop.

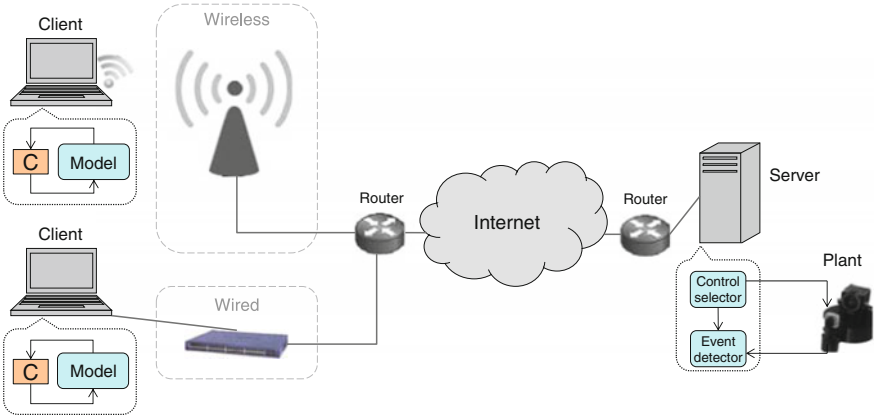

Fig. 4.7 Scheme of the experimental framework

4.6.2 Performance of Event-Triggered Control

The plant to test the event-triggered control is the flexible link shown in Fig. 4.8. The control objective is to respond to angular position commands with minimal amount of vibration and overshoot of the link. To get a complete model of a flexible link is beyond the scope of this framework. In controlling the extreme end of the link, it is sufficient to use a simplified model that will adequately describe the motion of the endpoint, that results in

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 592 & 32 & 0 \\ 0 & -947.3 & -32 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -56.2 \\ 56.2 \end{pmatrix} u, \tag{4.50}$$

where $x_1 = \theta, x_2 = \dot{\theta}, x_3 = \alpha, x_4 = \dot{\alpha}$, θ is the angle of the gear, and α is the arm deflection.

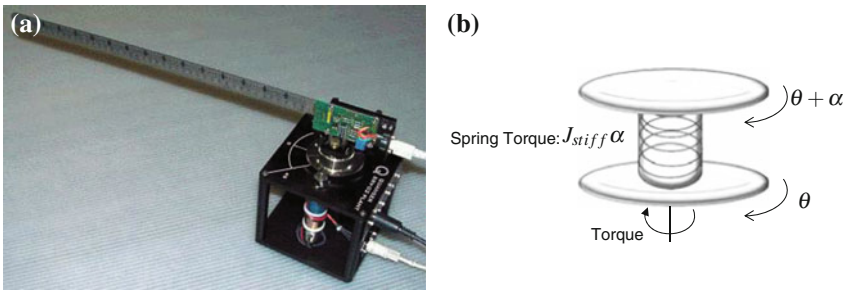

Fig. 4.8 The flexible link: **a** View of the module and **b** model

A feedback gain K is design to control the system:

$$K = (17.3205 \quad -24.7388 \quad 1.7164 \quad 0.5007), \tag{4.51}$$

that sets the poles at $\{-48.13, -35.34, -8.43 + 11.50i, -8.43 - 11.50i\}$. An anticipative controller based on the model (4.50) and the feedback gain (4.51) is designed for the flexible link. The system is sampled with $h = 10\,\text{ms}$, and the driver provided by Quanser is used to emulate full state measurement because the sensors only provides measurements of θ and α.

First, we afford the study of the influence of the parameter δ of the trigger function (4.4). The response to an angular position step command is shown in Fig. 4.9. Three values are considered: $\delta = 0.05$ (blue), $\delta = 0.1$ (red), and $\delta = 0.2$ (green). The angle of the gear x_1, the arm deflection x_2, the control input, and the events execution are depicted.

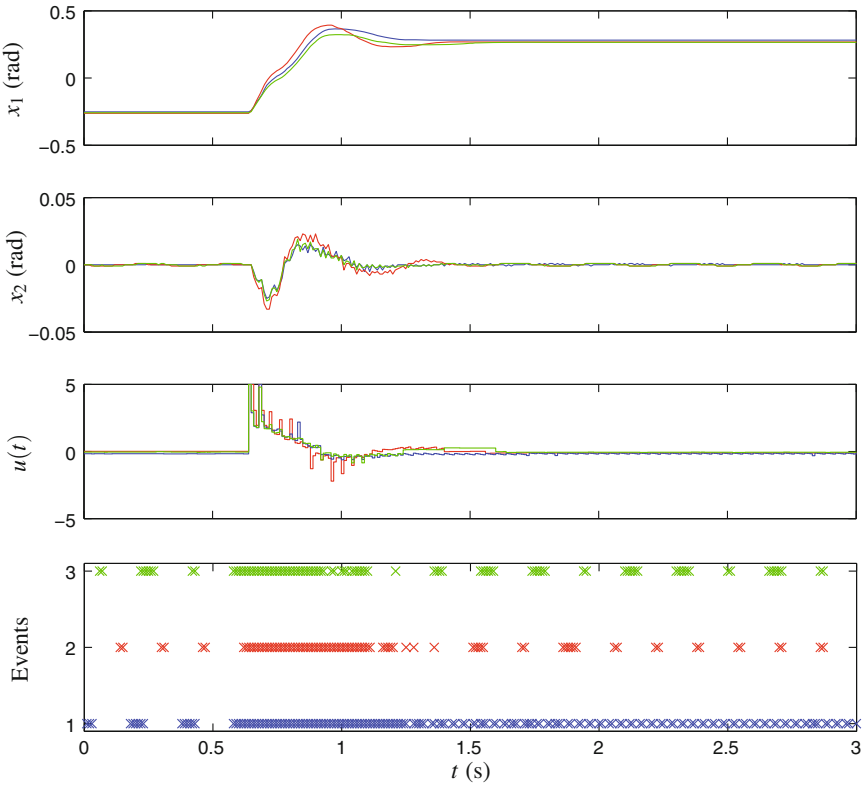

Fig. 4.9 Step response for the event-based controller with $\delta = 0.05$ (*blue*), $\delta = 0.1$ (*red*), and $\delta = 0.2$ (*green*)

Note that the performance in the three situations is similar, and the main differences are in the number of triggered events. If $\delta = 0.05$, the frequency of generation of events is high when the system reaches the final set around the equilibrium. These events are possibly caused by the noise in the measurements. Thus, this fact should be taken into account when designing the trigger function.

Another interesting phenomenon can be observed, for instance, in the second design ($\delta = 0.1$). In the interval of time (1.5, 3) s, the detection of an event usually involves an additional transmission very close to the previous one. This occurs when the RTT is larger than the sampling period. Since a new control packet has not been received, another state packet is transmitted asking for control actions. Thus, the previous transmission is considered as lost. Note that the packets are not acknowledged in the proposed protocol. Alternatively, acknowledgment of packets can be set up with a convenient *waiting time*.

Finally, it can be noticed that when the system reaches the final set around the set point, the frequency of transmission is almost constant with a transmission period around 100–140 ms, since an event is also enforced when the number of the remaining elements in the actuator buffers is below the parameter $\bar{N}_u$ (see Algorithm 4.1).

Figure 4.10 shows the system response for three different frameworks:

1. **Classical control scheme** The conventional state feedback controller is located at the process side (classical control scheme).

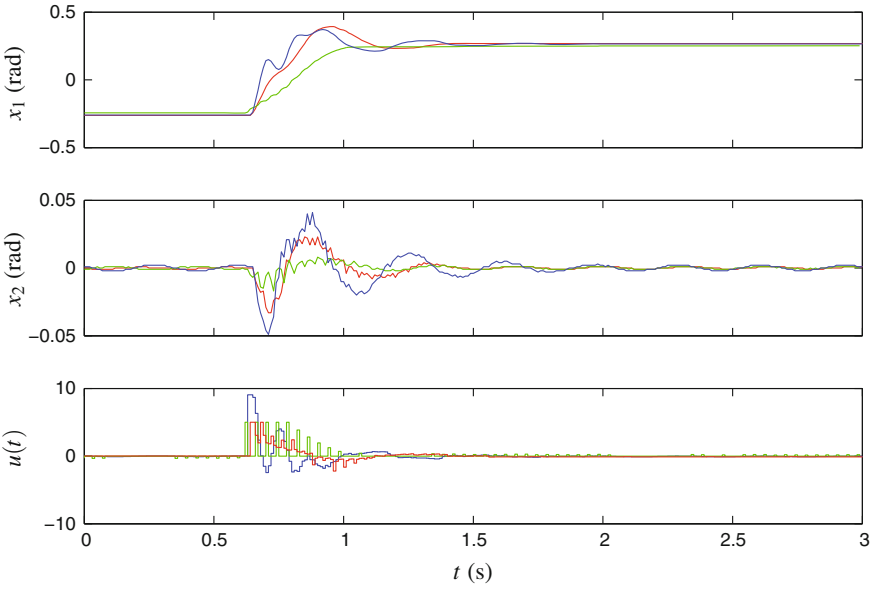

Fig. 4.10 Performance comparative of the local controller (*blue*), the remote controller with $N_u = 1$ (*green*), and the remote controller with $N_u = 20$ (*red*)

Table 4.1 Performance parameters of the three frameworks depicted in Fig. 4.10

Framework	Rise time (s)	Settling time (s)	Overshoot (%)	IAE	Events
1	0.13	0.91	40.01	0.063	300
2	0.29	0.43	0.00	0.114	182
3	0.17	0.71	46.64	0.086	65

2. **Remote state feedback** The state feedback controller receives measurements and sends control actions through the network without using a model and/or compensation of network effects with event-triggered sampling and $\delta = 0.1$.
3. **Remote state feedback controller with anticipative strategy** The anticipative controller proposed with the state feedback basis controller with sequence length $N_u = 20$ (length of control and prediction sequences), and the trigger function with $\delta = 0.1$.

The three previous frameworks have the same controller tuning and what change is only the control architecture. The performance of the three frameworks is summed up in Table 4.1. It can be noticed that in the second case (green) the response exhibits a slower response because the actuator does not receive in time a control action and applies zero, whereas the settling time and the overshoot for the state feedback controller (blue) and the proposed design (red) are similar. If the number of events from the second and third frameworks are compared, it leads to a reduction of 64 % in the number of transmissions.

The Integral Absolute Error (IAE) is computed for the three frameworks and for the first component of the output vector as

$$\text{IAE} = \int_{t_0}^{t_f} |e_{sp}(t)| dt,$$

where $e_{sp}(t) = y_{sp} - y(t)$. The IAE is increased with event-triggered due to the existence of a stationary error that varies with δ. Nevertheless, the IAE is reduced with the anticipative strategy (framework 3) respect to the second framework.

4.6.3 Response to Disturbances

The disturbance estimator is evaluated when the system is in the equilibrium and it is perturbed in the output, by pushing the flexible link in one direction first, and next to the opposite direction. Figure 4.11 shows the system response in two situations: When disturbances are estimated, and when they are not. Note that the controller exhibits much better behavior when the disturbance is estimated. Such estimation is shown in Fig. 4.12. Each plot corresponds to a component of the estimated disturbance vector $w_m(k)$. Note that the signals are piecewise constant and each update corresponds to the reception of a new state packet, i.e., the occurrence of an event.

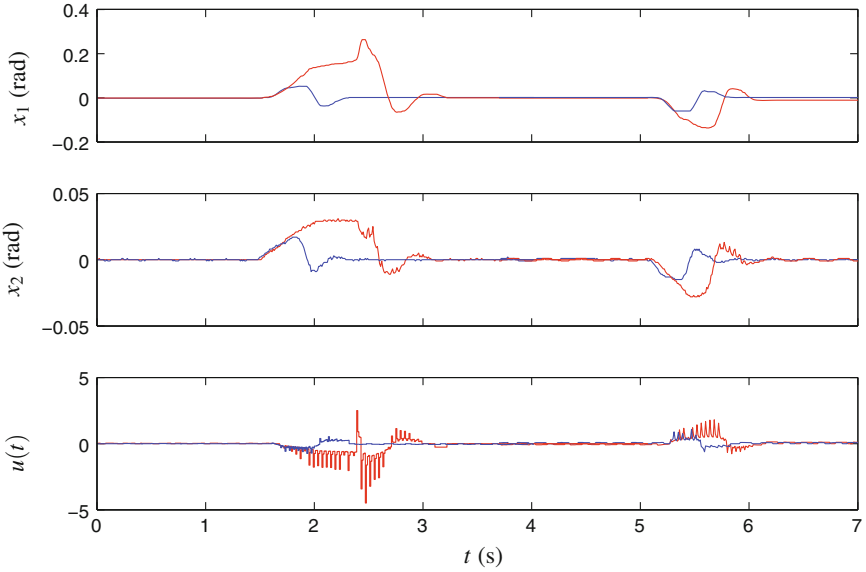

Fig. 4.11 Disturbance rejection with (*blue*) and without (*red*) disturbance estimation

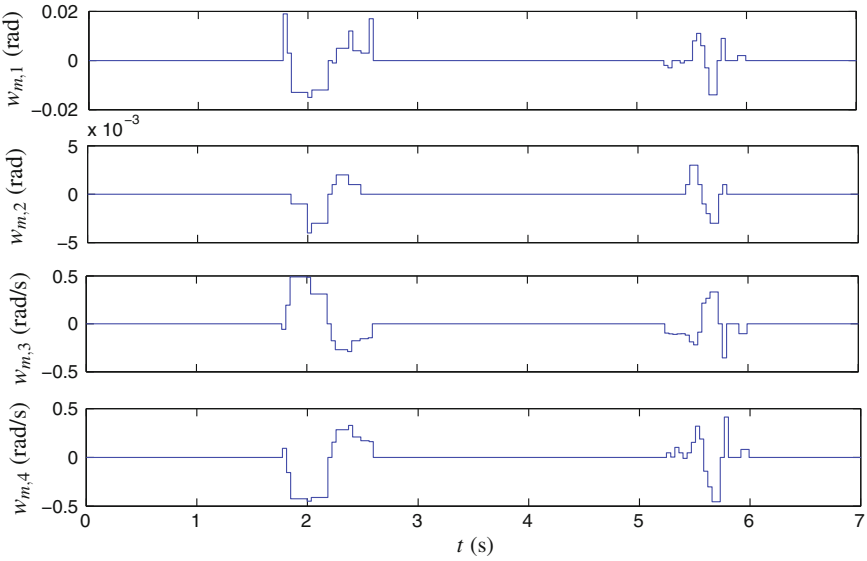

Fig. 4.12 Components of the disturbance estimation

The number of events is reduced even more and accounts for almost the half of the events without disturbances estimation.

In order to avoid the noise influence on the disturbance computation, a threshold is defined so that $w_m(k)$ is set to zero if the estimation is below this threshold. If this strategy is not taken, additional and undesired events may be generated.

4.6.4 PI Anticipative Control

To test the design for LTI anticipative controllers, a PI controller is designed for the SRV-02 gear, which is the module coupled to the flexible link described above. When used separately, one can experiment with angular position, as it has a decoder which determines the angle of the gear. The plant is modeled as a first order system with an integrator as follows

$$G(s) = \frac{-46.7}{s^2 + 33.3s}. \tag{4.52}$$

In the state-space representation, the state vector is $x^T = (x_1 \quad x_2)$, $y = x_1 = \theta$, $x_2 = \dot{\theta}$, being θ the angle of the gear. A PI controller has been design to control the angle. The parameters of this controller are

$$k_p = -2.5, \quad T_i = 1. \tag{4.53}$$

The rising time, settling time, and overshoot are 0.36 s, 2.78 s, and 16.10 %, respectively, with these parameters for a step command.

This controller is taken as the basis controller of the model-based approach. The performance of the PI anticipative controller over the SRV-02 gear is analyzed in two different situations:

1. The controller is anticipative with a length of predictions $N_u = 50$, but with periodic transmission from the sensor to the controller.
2. Such transmission is event-triggered (4.32) with $\delta_y = 0.1$ rad, and the rest of the parameters are the same.

The results for a particular experience are shown in Fig. 4.13. On the left hand side, the output is displayed when the reference is a square wave. The results highlight the benefits of the event-based communication. If the parameter δ_y is selected properly, the system response is similar to the time-based case, but the exchange of data plant-controller through the network is considerably reduced. Note, however, that there is a stationary error not compensated, defined by δ_y.

On the right hand side of Fig. 4.13 the parameter denoted *Index usage* is depicted. This parameter represents the number of times that the element *index*, *index* $= 1, \ldots, N_u$ of any control sequence $\{\mathscr{U}_k, k \in \mathbb{N}\}$ has been applied by the actuator.

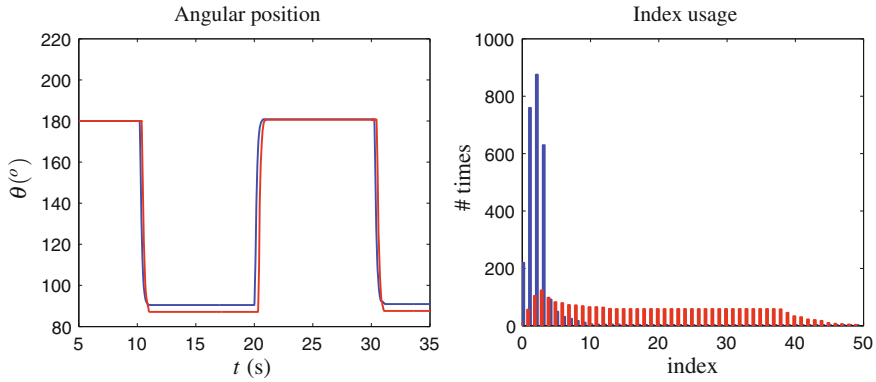

Fig. 4.13 Comparison of time-based (*blue*) and event-based (*red*) PI anticipative controllers

We remind that the first i_0 elements are discarded according to the current measurement of the RTT, and the subsequent elements are applied until a most recent computation is received.

If the time-based event-based transmission policies are compared, a more efficient usage of the received control sequences in the event-triggered approach is appreciated.

Note that the indices at which the peak values are reached are basically the same, but the index usage of the subsequent elements rapidly decreases in the periodic transmission, since they are only used in case the delay is large enough so that new data has not been received yet. However, new control packets are not requested until the error exceeds the threshold δ_y or the index reaches the bound $\bar{N}_u$ in event triggering.

4.6.5 Network: Delays and Packet Losses

An interesting property to study is the RTT that characterize this framework. In particular, when the remote controller connects to a wireless network the reliability decreases. Three samples of data taken at different times of the day for the wireless network are depicted in Fig. 4.14. It can be observed that the minimum value remains almost constant around 8 ms in the three situations. There is not a predictable profile and, apparently, there are random peaks. These sudden increments may be due to increase of network traffic or the server providing other services. The maximum values correspond to the dotted lines.

The network-induced delay and the packet dropouts are intrinsic properties to the communication channel and cannot be predetermined. Hence, studying the system under different network conditions is a difficult task a priori. However, artificial delays or packet losses can be induced from the user application.

The effect of both phenomena over the system performance is discussed next.

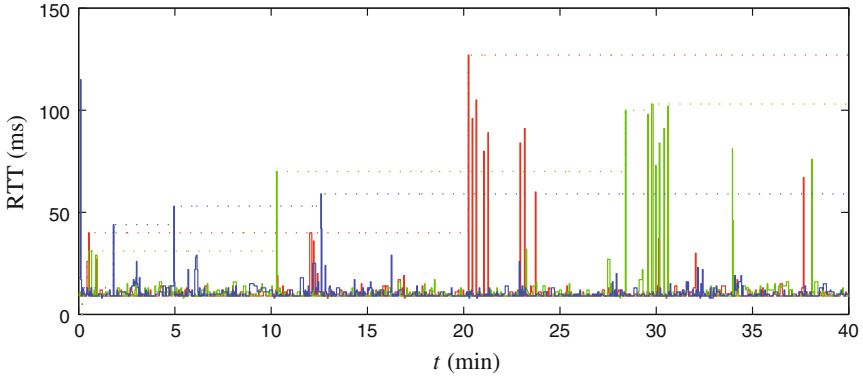

Fig. 4.14 Measured RTT in three experiments: 10 AM (*red*), 3 PM (*green*), 8 PM (*blue*)

4.6.5.1 Study of the Delay Impact

If the theoretical upper bound is computed according to (4.17) for the SRV-02 gear model, it follows that $\|A^{\tau} - I\| < 1$ is satisfied if $\tau < 23$ sampling periods, that is, 230 ms when the sampling period is 10 ms. This result holds assuming that the model is perfect, obtaining a more conservative upper bound if model uncertainties are considered.

An experiment to set the value of the average RTT to 20, 50, 100, and 230 ms has been designed by introducing artificial delays. The rise time, settling time, and overshoot have been computed for the five cases described above. The results are summarized in Table 4.2. The settling time increases with the RTT, whereas the rise time is preserved in adequate values except in the last case.

4.6.5.2 Study of the Packets Dropouts Impact

As a delay, a packet loss cannot be predetermined in the Internet, but, as in the previous case, it can be caused artificially. This allows testing the robustness of the designed approach for a set of values of probability of data dropouts.

Table 4.2 Performance parameters for different values of average RTT

RTT (ms)	Rise time (s)	Settling time (s)	Overshoot
0	0.38	0.45	0.00
20	0.32	0.63	0.00
50	0.28	1.25	2.34
100	0.37	1.20	4.67
230	1.16	2.87	2.09

Table 4.3 Performance parameters for different values of p

p	Rise time (s)	Settling time (s)	Overshoot
0.0	0.35	0.40	0.00
0.2	0.34	0.42	0.00
0.4	0.34	0.45	0.00
0.6	0.32	0.55	15.92
0.8	0.59	0.92	18.10

The chance of not losing a packet has been modeled by a Bernoulli discrete distribution with a probability of success q, so that the probability of losing a packet is $p = 1 - q$. As an example, Table 4.3 shows the parameters of the system response for different values of p. For example, a value of $p = 0.4$ means that 40 % of the packets will be lost in average. The system exhibits good behavior if $p \leq 0.6$ as the rise and settling times are almost the same in this interval, and the degradation of performance is evident if only one of each five packets are delivered. Note that a high value of p causes an increase in the overshoot, whereas this phenomenon is not so appreciated with large RTTs.

4.7 Conclusions

An anticipative controller for packet-based NCS has been presented. A model of the plant predicts future states of the plant and, with this information, generates future control actions, and the frequency of communication is reduced thanks to the event-based sampling. 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 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 proposed architecture has been implemented and evaluated in a framework in which the remote controller communicates with the process through the Internet. The remote controller has been tested over two devices, a DC motor and a flexible link, and state-feedback, and PI controllers have been taken as basis controllers. The experimental results have analyzed the influence of the architecture, the design of the trigger function, or the impact of network delays and packet dropouts. The event-based anticipative controller has been shown to be efficient against delays and packet dropouts, while reducing the need of communication.

Chapter 5
H_2/H_∞ Control for Networked Control Systems with Asynchronous Communication

Luis Orihuela and Carlos Vivas

5.1 Introduction

Due to the limited transmission capacity of the network and inherent delays in communications, data transmitted in practical NCSs cannot be guaranteed to reach the control loop devices synchronously and periodically. Therefore, to ensure closed-loop stability and achieve better performance of the considered systems, these effects should be taken into consideration.

In Chap. 4, the problem is studied resorting to model-based anticipative controllers that generate a finite-horizon sequence of future control actions. The sequence is thus stored into the actuator buffer and applied synchronously at each sampling time. A complementary approach, that is tackled in this chapter, is to attenuate the effects caused by delay disturbances designing the controller in such a way that can cope with delays and eventual packet losses to a certain extent. This is the so-called robust design approach for NCS.

The problem has been thoroughly studied in the literature ([119, 129, 184, 275] to name a few). Available design techniques typically are overly conservative, and this is the reason why many works focus in reducing conservatism for a given characterization of the communication channel. For network-induced delays, a common measurement of conservativeness is the maximum allowed delay bound (MADB), that states the maximum time span allowed for the actuator to be open-loop (without receiving control information from the controller) before the system goes unstable,

L. Orihuela (✉)
Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería,
Universidad Loyola Andalucía, Seville, Spain
e-mail: dorihuela@uloyola.es

C. Vivas
Dpto. de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingenieros,
Universidad de Sevilla, Seville, Spain
e-mail: vivas@us.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_5

see [120, 129, 275] for some examples of the use and computation of the MADB in the NCS context.

Nonetheless, the maximization of the MADB per se is not generally useful for practical NCS applications, as in addition to stability, closed-loop performance requirements are usually imposed on the system. The H_∞ control problem is able to address the issue of disturbance rejection and model uncertainty. It was first formulated [238, 279] in the early 1980s where the L_2-gain disturbance attenuation problem played a key role. The effectiveness of the controller in attenuating disturbances according to the H_∞ norm has been reported and studied for NCSs in a variety of works [37, 120, 224, 254, 259, 275].

The aforementioned results mostly solve the L_2-gain disturbance rejection problem for state-feedback controllers in NCSs. This chapter, in addition to disturbance attenuation will introduce performance control in an optimal control context for NCS. The joint optimal control and disturbance attenuation is usually referred to in the literature as the H_2/H_∞ control problem, as the H_2 part accounts for the optimization of a performance index, with an L_2-gain disturbance rejection constraint in the H_∞ component.

Some works in the literature address the mixed H_2/H_∞ problem for related frameworks, as time-delay systems [130], descriptor delayed system [276], or neutral systems with delays [38]. All these works formulate the minimization of the H_2 cost index with a given H_∞ disturbance rejection level. More precisely, to minimize the H_2 index, a more or less conservative upper bound for a cost function J is obtained, which generally depends on the initial conditions.

The method proposed in this chapter, based on the works in [170, 174], deals with the design of controllers for linear NCSs subject to L_2 bounded disturbances, where signals are influenced by time-varying delays and packet dropouts. Given a cost function J and a controlled output $z(t)$, the proposed problem can be informally stated as designing a linear controller such that:

- Stabilizes the unperturbed system ($\omega(t) \equiv 0$), as the cost function J is minimized.
- The controlled output satisfies $\|z(t)\|_2 \leq \gamma \|\omega(t)\|_2$ for any non-zero disturbance $\omega(t)$.

It is worth mentioning that the optimization of the H_2 part is conducted in this description in a different way to other similar approaches, as no knowledge of the initial condition is needed.

Once the background for the mixed H_2/H_∞ problem in NCS is introduced, the second part of the chapter is devoted to present an event-based communication policy between the sensor and the controller, in pursue of a further efficient use of the available bandwidth and energy consumption reduction.

The key feature of asynchronous schemes is its communication efficiency and energy awareness, see [8, 239]. In the case of event-based control this efficiency materializes, roughly speaking, by transmitting information only when it is significant for control purposes. The specific advantages of these strategies in system efficiency depend on the control application. For instance, in large-scale plants it is common the

use of, frequently unaccessible, wireless sensors. In such situations, increasing the battery lifetime through asynchronous communication strategies becomes crucial. Moreover, in wired networks, the communication channel is often shared by a large number of different devices and control loops. By communicating only relevant control information rather than periodically transmitting measurements or control actions, the bandwidth requirement of these control loops is reduced, which has a great impact on the network performance, reducing collisions, packet losses, and time-delays.

Asynchronous communications are introduced by appropriately selecting a LKF and remodeling the system's dynamics to account for asynchronous communications. The chapter investigates the trade-off between the frequency of communication events and control performance, obtaining an expression that relates the threshold for the sampling events with the size of the ultimate region in which the state of the system is eventually confined. As a consequence of this study, it is demonstrated for that a discrete TDS admitting a LKF remains practically stable in the presence of bounded additive disturbances.

The problem described in this chapter is precisely outlined in Sect. 5.2, with a description of the network assumptions, the system model considered and the problem to be solved. Section 5.3 provides a suboptimal solution of a mixed H_2/H_∞ control problem for a certain class of time-delayed systems with quadratic performance index. The solution is later particularized to NCS in Sect. 5.4 where an specific Lyapunov–Krasovskii functional is proposed which allows to cast the problems in the form of a set of LMIs. An event-based approach is then considered in Sect. 5.5 providing conditions for the practical stability of delayed asynchronous systems. The performance of these methods is assessed with some simulation examples in Sect. 5.6.

5.2 System Description

The system considered in this chapter is linear, but affected by external disturbances. As shown in Fig. 5.1, this system is controlled by a remote agent. The communication between plant and controlled is made through a communication network.

The dynamics of the system is given by:

$$x(k+1) = Ax(k) + Bu(k) + B_w w(k), \tag{5.1}$$

where $x(k) \in \mathbb{R}^n$ is the state, $u(k) \in \mathbb{R}^m$ the control signal and $w(k) \in \mathbb{R}^{n_w}$ external disturbances. A, B, B_w are known matrices with adequate dimensions. The initial condition for the plant is $x(k) = \phi(k)$, where $\phi(k)$ is a vector-valued function defined in $k \in [-\tau_{max}, 0]$, being τ_{max} the upper bound of the communication delay, which will be presented in the following subsection.

This section will show that, under some assumptions concerning network conditions, a control loop closed through a communication network can be modeled

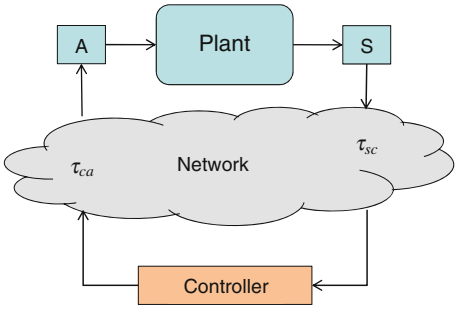

Fig. 5.1 Scheme of the networked control system

according to a piecewise discrete time-delay system. This method has been used in companion works for continuous systems, see [172, 174]. The idea underneath is called the *input-delay approach*, which was firstly proposed in [75].

5.2.1 Network Conditions

As Fig. 5.1 suggests, plant and controller are physically distributed and linked through a communication network that induces transmission delays in both, sensor to controller path (τ_{sc}) and controller-to-actuator path (τ_{ca}). The sensor samples plant states periodically at instants j_i, and sends this information through the network. Packets are affected by time-varying delay (τ_{sc}), and some packets may be eventually lost. To account for packet dropouts, it is assumed that j_i ($i = 1, 2, 3, ...$) are integers such that $\{j_1, j_2, j_3, ...\} \subseteq \{1, 2, 3, ...\}$ and $j_i < j_{i+1}$. In other words, dropouts are modeled as extended delays between two or more consecutive packets.

On the other side (see Fig. 5.1), the controller computes a new control signal whenever a new delayed state is received. Then, this control signal is released to the actuator through the controller-to-actuator path, being affected by delay (τ_{ca}) and, possibly, packet dropouts.

As described for the controller side, the actuator applies the control action as it is received. The actuator consists of a simple zero-order holder, as it maintains the value of the control action until a new packet arrives. A packet numbering protocol is assumed so the relative order of packets can be inferred at the actuator's end. Out-of-order packets are thus rejected by the actuator.

Let us define $k \in [k_i, k_{i+1} - 1]$ as the time interval where the control input applied to the system is constant, where k_i is the instant when the control action, corresponding to the plant state at j_i, is applied to the plant. Figure 5.2 illustrates a possible time scheduling, where all these elements, delays and packet dropouts, are sketched.

The state feedback control law can therefore be written as

$$u(k) = Kx(k_i - \tau_{sc}(j_i) - \tau_{ca}(j_i)), \quad k \in [k_i, k_{i+1} - 1], \tag{5.2}$$

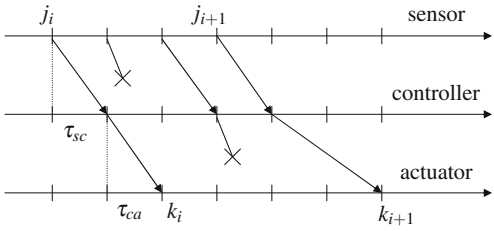

Fig. 5.2 Delays and packet dropouts affecting the communication between plant and controller

where $\tau_{sc}(j_i)$ and $\tau_{ca}(j_i)$ are the network-induced delays of the packet corresponding to the measured plant state at j_i, from sensor to controller and from controller-to-actuator, respectively. The round-trip delay $\tau_{sa}(j_i)$ can also be defined as $\tau_{sa}(j_i) \triangleq \tau_{sc}(j_i) + \tau_{ca}(j_i)$.

Thus the system (5.1) under the control law (5.2) can be rewritten as

$$x(k+1) = Ax(k) + BKx(k - \tau(k)) + B_w w(k), \ k \in [k_i, k_{i+1} - 1], \tag{5.3}$$

where $\tau(k) \triangleq k - k_i + \tau_{sa}(j_i)$ is an artificial delay that includes the effect of both network-induced delays and dropouts. It turns out that $\tau(k)$ is piecewise linear for $k \in [k_i, k_{i+1} - 1]$, as it represents the difference between the ith sampling time, j_i, and current instant k.

The following assumption characterizes the network conditions with respect to induced delays and packet dropouts. It imposes fairly standard and realistic constraints in the NCS framework.

Assumption 5.1 Three constants $\underline{\tau}_{sa}, \overline{\tau}_{sa}, n_p \geq 0$ exist such that:

- The network-induced delay from sensor to actuator $\tau_{sa}(k)$ satisfies $\underline{\tau}_{sa} \leq \tau_{sa}(k) \leq \overline{\tau}_{sa}, \forall k$.
- The maximum number of consecutive packet dropouts from sensor to actuator is bounded by n_p.

The following proposition, whose proof is straightforward, gives the numerical bounds of the artificial delay $\tau(k)$.

Proposition 5.1 *Taking into account Assumption 5.1, two constants* $\tau_{max} > \tau_{min} \geq 0$ *exist such that*

$$\tau(k) \geq \tau_{min} \triangleq \underline{\tau}_{sa}, \tag{5.4}$$

$$\tau(k) \leq \tau_{max} \triangleq n_p + \overline{\tau}_{sa}. \tag{5.5}$$

Hence, at this point, the considered NCS has been modeled as a piecewise discrete linear system affected by time-varying, but bounded delays.

5.2.2 Problem Statement

For the system (5.3) consider the following two outputs:

$$z_2(k) = C_2 x(k) + D_2 u(k), \tag{5.6}$$
$$z_\infty(k) = C_\infty x(k) + D_\infty u(k). \tag{5.7}$$

Based on the first output z_2, a cost functional is defined to evaluate the control system performance:

$$J_2 = \sum_{j=k_0}^{\infty} z_2^T(j) z_2(j). \tag{5.8}$$

Then, the problem to be solved in this chapter can be formally presented.

Definition 5.1 (*Suboptimal mixed H_2/H_∞ control problem in discrete time*) Consider the system described by (5.3). Given:

- A desired level of disturbance attenuation γ, and
- A quadratic cost function J_2 in the form (5.8),

the suboptimal mixed H_2/H_∞ control problem in discrete time consists in finding a linear controller K such that:

1. The closed-loop system is asymptotically stable for $w(k) \equiv 0$,
2. The controller minimizes the upper bound of the cost function J_2 for $w(k) \equiv 0$,
3. Under the assumption of zero initial conditions, the output $z_\infty(k)$ satisfy $\|z_\infty(k)\|_2 \leq \gamma \|w(k)\|_2$ for any non-zero disturbance $w(k) \in L_2[0, \infty)$.

Loosely speaking, the objective of the mixed H_2/H_∞ control problem will be to synthesize a linear state feedback controller such that the performance index H_2 is minimized and the closed-loop system has an H_∞ norm less or equal than γ.

Once the controller has been properly found, the chapter will show the implementation of this scheme under an asynchronous paradigm. Event-based strategies will be used to reduce the communication between controller and plant preserving the stability.

5.3 General Solution for the Suboptimal Mixed H_2/H_∞ Control Problem

In this section, a general solution for the suboptimal mixed H_2/H_∞ problem will be presented. The stability properties of the solution are based on the Lyapunov–Krasovskii theory. Next, some assumptions concerning the Lyapunov–Krasovskii functional (LKF) are introduced.

Assumption 5.2 The discrete quadratic Lyapunov–Krasovskii functional can be written as

$$V(k) = \zeta^T(k)\Psi\zeta(k), \tag{5.9}$$

where $\zeta(k) = \left[x^T(k)\ x^T(k-1)\ x^T(k-2)\ \ldots\ x^T(k-\tau_{max})\right]^T \in \mathbb{R}^{n_\zeta}$ and $\Psi \in \mathbb{R}^{n_\zeta \times n_\zeta}$ is a positive definite matrix. Additionally, the forward difference $\Delta V(k) \triangleq V(k+1) - V(k)$ can be bounded by:

$$\Delta V(k) \leq \begin{cases} \xi^T(k)\Xi(K)\xi(k), & w \equiv 0 \\ \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix}^T \Omega(K,\gamma) \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix} - z_\infty^T(k)z_\infty(k) + \gamma^2 w^T(k)w(k), & w \neq 0 \end{cases} \tag{5.10}$$

being $\xi(k) \in \mathbb{R}^{n_\xi}$ an augmented state vector which depends, among others, on the state of the system and $\Omega(K,\gamma) = \begin{bmatrix} \Xi(K) + \bar{C}_z(K) & \bar{B}_w(K) \\ * & -\gamma^2 I + \bar{D}_w(K) \end{bmatrix}$. The symmetric matrices $\Xi(K), \bar{C}_z(K) \in \mathbb{R}^{n_\xi \times n_\xi}$, $\bar{D}_w(K) \in \mathbb{R}^{n_w \times n_w}$ and the matrix $\bar{B}_w(K) \in \mathbb{R}^{n_\xi \times n_w}$ might depend, among others, on the controller K.

Assumption 5.3 The cost functional J_2 can be written in the following way:

$$J_2 = \sum_{j=k_0}^{\infty} \xi^T(j)\Phi(K)\xi(j), \tag{5.11}$$

where $\Phi(K)$ is a positive semidefinite matrix that might depend on the controller K.

Although a priori, these assumptions might seem difficult to satisfy, reviewing the functionals commonly used in the literature, as [39, 76, 131, 156], one can see that Assumptions 2.2 and 2.3 are easily fulfilled.

The following lemma proposes a general solution to the problem defined above. For a generic functional satisfying the assumptions, the H_2/H_∞ controller can be found by means of an optimization problem.

Lemma 5.1 *Let $V(k)$ be a functional satisfying Assumption 2.2 and J_2 a cost functional which can be written as detailed in Assumption 5.3 Then, the suboptimal mixed H_2/H_∞ control problem can be solved by finding a controller K such that:*

$$\min_K \quad \lambda_{max}(\Psi), \tag{5.12}$$

$$\text{subject to} \quad \Xi(K) < -\Phi(K), \tag{5.13}$$

$$\Theta(K,\gamma) < 0. \tag{5.14}$$

Proof It will be shown that a controller that solves the optimization problem (5.12) subject to (5.13) and (5.14) also satisfies all the issues of Definition 5.1.

1. For $w(k) \equiv 0$, considering (5.10), it holds that $\Delta V(k) \leq \xi^T(k)\Xi(K)\xi(k)$. From (5.13), one can easily see that Ξ is negative definite, and therefore $V(k)$ decreases, this ensuring the asymptotic stability of system.
2. For $w(k) \equiv 0$, considering (5.10), it holds that $\Delta V(k) \leq \xi^T(k)\Xi(K)\xi(k)$. From (5.13), it turns out

$$\Delta V(k) \leq \xi^T(k)\Xi(K)\xi(k) < -\xi^T(k)\Phi(K)\xi(k). \tag{5.15}$$

Calculating the summation of both sides of (5.15) from k_0 to k, it yields

$$\sum_{j=k_0}^{k} \Delta V(j) < -\sum_{j=k_0}^{k} \xi^T(j)\Phi(K)\xi(j).$$

Observe that $\sum_{j=k_0}^{k} \Delta V(j) = \sum_{j=k_0}^{k} (V(j+1) - V(j)) = V(k+1) - V(k_0)$. When $k \to \infty$, the asymptotic stability of the system implies that $V(k) \to 0$, so that,

$$-V(k_0) \leq -\frac{1}{\alpha} \sum_{j=k_0}^{\infty} \xi^T(j)\Phi(K)\xi(j).$$

The right-hand side of the previous equation is exactly $-J_2$, as Assumption 5.3 claims. Hence,

$$-V(k_0) \leq -J_2 \implies J_2 \leq V(k_0)$$

The value of $V(k_0)$ depends on the initial condition $\phi(k)$. In particular, and regarding to (5.9), it turns out

$$V(k_0) = \phi^T(k)\Psi\phi(k) \leq \lambda_{max}(\Psi)\|\phi(k)\|^2$$

Therefore, minimizing $\lambda_{max}(\Psi)$ the upper bound of the cost function J_2 is minimized regardless of the initial conditions.
3. For $w \neq 0$, taking into account Eq. (5.10) in Assumption 5.2 and condition (5.14), the term $\begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix}^T \Theta(K, \gamma) \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix}$ is negative definite. Then, under zero initial conditions,

$$\Delta V(k) \leq -z_\infty^T(k)z_\infty(k) + \gamma^2 w^T(k)w(k). \tag{5.16}$$

Computing again the summation of both sides of (5.16) one can see that

$$V(k+1) - V(k_0) \leq -\sum_{j=k_0}^{k} z_\infty^T(j)z_\infty(j) + \sum_{j=k_0}^{k} \gamma^2 w^T(j)w(j).$$

Then, letting $k \to \infty$ and taking into account that under zero initial condition $V(k_0) = 0$ and the positive definitiveness of the functional, it can be shown that

$$0 \leq -\sum_{j=k_0}^{\infty} z_\infty^T(j) z_\infty(j) + \sum_{j=k_0}^{\infty} \gamma^2 w^T(j) w(j),$$

thus $\|z_\infty(k)\|_2 \leq \gamma \|w(k)\|_2$.

Lemma 5.1 differs from other approaches available in the literature in some aspects. First of all, it presents a general result that can be applied to many sorts of Lyapunov–Krasovskii functionals. Additionally, the minimization of the upper bound of the cost index J_2 is made independent of the value initial condition $\phi(k)$. Finally, and although we will apply it to find controllers for NCS, it can also be used for time-delay systems, as it has been shown in [207].

5.4 Application to Networked Control Systems

Section 5.2 shown that the dynamics of the closed-loop NCS can be described as a piecewise discrete time-delay system. The evolution given in Eq. (5.3) was only valid for the period in which the control signal remains constant, that is, $k \in [k_i, k_{i+1} - 1]$. However, the optimization problem proposed in Lemma 5.1 can be also applied to a NCS. The underlying idea is simple. Through Lemma 5.1, it will be assured that the forward difference of the Lyapunov–Krasovskii functional is negative for the period $[k_i, k_{i+1} - 1]$. As the functional is the same for every interval, we will prove that the forward difference is negative $\forall k$. The same arguments are also valid for the other issues in Definition 5.1.

5.4.1 Lyapunov–Krasovskii Functional

For the stability of the NCS, the chapter proposes the following Lyapunov–Krasovskii functional:

$$\begin{aligned} V(k) = {} & x^T(k) P x(k) + \sum_{i=k-\tau_{max}}^{k-1} x^T(i) Z_1 x(i) + \sum_{i=k-\tau_{min}}^{k-1} x^T(i) Z_2 x(i) \\ & + \tau_{max} \sum_{j=-\tau_{max}+1}^{0} \sum_{i=k+j-1}^{k-1} \Delta x^T(i) Z_3 \Delta x(i) \\ & + (\tau_{max} - \tau_{min}) \sum_{j=-\tau_{max}+1}^{-\tau_{min}} \sum_{i=k+j-1}^{k-1} \Delta x^T(i) Z_4 \Delta x(i), \end{aligned} \tag{5.17}$$

where $\Delta x(i) = x(i+1) - x(i)$ and matrices P, Z_1, Z_2, Z_3, Z_4 are all positive definite.

It is easy to find in the literature more complex functionals to deal with time-varying delays, as in [76, 77, 96], that produce less conservative results. However, the objective of this chapter is not the reduction of the conservatism by using a novel functional or tricky mathematical manipulations, but to give a general method to design H_2/H_∞ controllers for NCS.

Proposition 5.2 *The Lyapunov–Krasovskii functional (5.17) satisfies all the conditions presented in Assumption 5.2. On the one hand, it can be written as in (5.9) by choosing*

$$\Psi = \left[\begin{array}{c|ccc|ccc} P & 0 & \dots & 0 & 0 & \dots & 0 \\ \hline 0 & Z_1+Z_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Z_1+Z_2 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 & Z_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & Z_1 \end{array}\right] + \tau_{max} \sum_{i=0}^{\tau_{max}} \left[\begin{array}{ccc} Z_3 & -Z_3 & \dots & 0 \\ -Z_3 & 2Z_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Z_3 \end{array}\right] \begin{array}{c} k \\ k-1 \\ \vdots \\ k-\tau_{max} \end{array}$$

$$+ (\tau_{max} - \tau_{min}) \sum_{i=\tau_{min}}^{\tau_{max}} \left[\begin{array}{ccccc} 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & Z_4 & -Z_4 & \dots & 0 \\ 0 & \dots & -Z_4 & 2Z_4 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & Z_4 \end{array}\right] \begin{array}{c} k \\ \vdots \\ k-\tau_{min} \\ k-\tau_{min}-1 \\ \vdots \\ k-\tau_{max} \end{array}$$

On the other hand, the augmented state vector is

$$\xi(k) = \left[x^T(k)\ x^T(k-\tau(k))\ x^T(k-\tau_{min})\ x^T(k-\tau_{max}) \right]^T$$

and the matrices in (5.10) are given by:

$$\Xi(K) = M + \tilde{A}^T P \tilde{A} + (\tilde{A} - \tilde{I})^T N (\tilde{A} - \tilde{I}),$$

$$\Omega(K,\gamma) = \left[\begin{array}{cc} M + \tilde{C}_\infty^T \tilde{C}_\infty & 0 \\ 0 & -\gamma^2 I \end{array}\right] + \left[\begin{array}{c} \tilde{A}^T \\ B_w^T \end{array}\right] P \left[\begin{array}{cc} \tilde{A} & B_w \end{array}\right] + \left[\begin{array}{c} (\tilde{A} - \tilde{I})^T \\ B_w^T \end{array}\right] N \left[\begin{array}{cc} (\tilde{A} - \tilde{I}) & B_w \end{array}\right],$$

where

$$N = \tau_{max}^2 Z_3 + (\tau_{max} - \tau_{min})^2 Z_4,$$

$$M = \left[\begin{array}{cccc} -P + Z_1 + Z_2 - Z_3 & Z_3 & 0 & 0 \\ * & -2Z_3 - 2Z_4 & Z_4 & Z_3 + Z_4 \\ * & * & -Z_2 - Z_4 & 0 \\ * & * & * & -Z_1 - Z_3 - Z_4 \end{array}\right],$$

$$\tilde{A} = \begin{bmatrix} A & BK & 0 & 0 \end{bmatrix},$$
$$\tilde{I} = \begin{bmatrix} I & 0 & 0 & 0 \end{bmatrix},$$
$$\tilde{C}_\infty = \begin{bmatrix} C_\infty & D_\infty K & 0 & 0 \end{bmatrix}.$$

The details of the proof are found in Appendix A.

5.4.2 Design Method

Using the previous Lyapunov–Krasovskii functional and Lemma 5.1, this section presents the main result of this chapter. The following theorem states the synthesis of a stabilizing controller as an optimization problem subject to nonlinear constraints. Later, two methods will be presented to sort with those nonlinear terms, in such a way that they can be replaced by a set of linear matrix inequalities.

Theorem 5.1 *Given $\tau_{max}, \tau_{min}, \gamma > 0$, if matrices $X, Y_1, Y_2, Y_3, Y_4 > 0$ and matrix W of appropriate dimensions solve the following optimization problem:*

$$\min_{X,Y_1,Y_2,Y_3,Y_4,W} \lambda_{max}(\Psi), \quad \textit{subject to:}$$

$$\begin{bmatrix} \Upsilon_1 & \Upsilon_2 & \bar{C}_2^T \\ * & -\Upsilon_3 & 0 \\ * & * & I \end{bmatrix} < 0, \tag{5.18}$$

$$\begin{bmatrix} \Upsilon_1 & 0 & \Upsilon_2 & \bar{C}_\infty^T \\ * & -\gamma^2 I & \bar{B} & 0 \\ * & * & -\Upsilon_3 & 0 \\ * & * & * & -I \end{bmatrix} < 0, \tag{5.19}$$

where

$$\Upsilon_1 = \begin{bmatrix} -X + Y_1 + Y_2 - Y_3 & Y_3 & 0 & 0 \\ * & -2Y_3 - 2Y_4 & Y_4 & Y_3 + Y_4 \\ * & * & -Y_2 - Y_4 & 0 \\ * & * & * & -Y_1 - Y_3 - Y_4 \end{bmatrix}$$
$$\Upsilon_2 = \begin{bmatrix} \bar{A} & \tau_{max}(\bar{A} - \bar{X}) & (\tau_{max} - \tau_{min})(\bar{A} - \bar{X}) \end{bmatrix}$$
$$\Upsilon_3 = diag\{X, XY_3^{-1}X, XY_4^{-1}X\}$$
$$\bar{B} = \begin{bmatrix} XB_w^T & \tau_{max} XB_w^T & (\tau_{max} - \tau_{min}) XB_w^T \end{bmatrix}$$
$$\bar{A} = \begin{bmatrix} AX & BW & 0 & 0 \end{bmatrix}^T$$
$$\bar{X} = \begin{bmatrix} X & 0 & 0 & 0 \end{bmatrix}^T$$
$$\bar{C}_2 = \begin{bmatrix} C_2 X & D_2 W & 0 & 0 \end{bmatrix}$$
$$\bar{C}_\infty = \begin{bmatrix} C_\infty X & D_\infty W & 0 & 0 \end{bmatrix}$$

then the suboptimal mixed H_2/H_∞ controller for the system (5.3) is given by $K = WX^{-1}$.

Proof The proposed Lyapunov–Krasovskii functional verifies all the issues in Assumption 5.2. Furthermore, it is straightforward to satisfy also Assumption 5.3 defining $\Phi(K) = \tilde{C}_2^T \tilde{C}_2$, with $\tilde{C}_2 = [C_2 \;\; D_2 K \;\; 0 \;\; 0]$.

Now, Lemma 5.1 is used to design the mixed H_2/H_∞ controller. From Eq. (5.13) we can obtain (5.18) applying Schur complement and pre- and post-multiplying the inequality by $diag\{P^{-1}, P^{-1}, P^{-1}, P^{-1}, I, I, I\}$. To obtain (5.19) from condition (5.14) we repeat the mathematical manipulation and pre- and post-multiplying the $diag\{P^{-1}, P^{-1}, P^{-1}, P^{-1}, I, I, I, I\}$.

Finally, introducing the definitions $X \triangleq P^{-1}$, $Y_i \triangleq XZ_iX$ $(i = 1, ..., 4)$, $W \triangleq KX$, Theorem 5.1 is proved.

Note that conditions (5.18) and (5.19) are not linear because of the terms $XY_3^{-1}X$ and $XY_4^{-1}X$ in Υ_3. In order to find a solution for the optimization problem, one can use one of the following methods to deal with those nonlinearities:

Direct constraint on $XY_i^{-1}X$ A direct method to deal with this nonlinearity consists in introducing two additional constraints to the optimization problem:

$$-XY_3^{-1}X < -\frac{1}{\mu_3}X, \;\; -XY_4^{-1}X < -\frac{1}{\mu_4}X.$$

Note that previous conditions are equivalent to $Y_i < \mu_i X$, $i = 3, 4$, which are linear. This method introduces extra conservatism, but it is computationally easy to implement. Comparing with a similar solution in [275], in which it was imposed that $Y_i = \mu_i X$, the one proposed here covers a much wider range of possible solutions in the space of positive definite matrices.

Cone complementary algorithm Similarly to [184], the cone complementary algorithm can be used to address the nonlinear matrix inequalities (5.18) and (5.19) by introducing some new matrix variables and constraints. The algorithm can be found in [64].

This method requires to solve a second optimization problem with linear constraints. Therefore, it requires much more computation capabilities. However, it does not introduce any additional conservatism in the solution.

The interested reader can find much more information of both methods in Appendix B.

5.5 Event-Based Control Implementation

This section tackles an asynchronous, event-based implementation of the controller developed in the sections above. It will be theoretically demonstrated that an asymptotically stable system affected with time-delays that admits an LKF, remains globally uniformly ultimately bounded (GUUB) in the presence of bounded disturbances. This result, together with an adequate remodeling of the system dynamics, will allow to demonstrate the stability of the H_2/H_∞ controller under an asynchronous, event-based implementation. Besides, an expression relating the size of the ultimate invariant region and the threshold triggering the communication events will be derived, which illustrates the trade-off between control performance and communication savings.

For the sake of conciseness, some simplifications will be done in this section. First, packet dropouts will not be explicitly considered, so the analysis will be carried out considering only time delays. Additionally, the analysis will be performed in the absence of exogenous disturbances $w(k)$.

5.5.1 Proposed Approach

The event-based control approach is a means to reduce the information exchange rates between the components in the network by triggering the communication only after an event has indicated that a certain relevant variable exceeds a tolerable predefined threshold. In particular, this chapter uses the deadband control approach explained in Sect. 1.6.1. From a modeling point of view, the main difference between a periodic transmission scheme and the event-driven paradigm described here is the non-uniform pattern of transmission of information.

Let $\ell_s(k)$ denotes the time instant when the sensor sent the more recent packet at current time instant k, and $\ell_a(k)$ the time instant when the sensor sent the more recent packet whose corresponding control action is available for the actuator at current time instant k.

Triggering condition Given a threshold δ, at instant k the sensor transmits the plant state to the controller if

$$\|x(\ell_s(k)) - x(k)\|_\infty \geq \delta, \quad \text{for } k > \ell_s(k). \tag{5.20}$$

Figure 5.3 illustrates the modification. As it can be seen, the sensor decides when to transmit the information. Its decision is based on the difference between the actual measurement $x(k)$ and the last measurement that it was sent $x(\ell_s(k))$. Observe that the sensor ignores $\ell_a(k)$, as the delays are time varying. The dynamics of the closed loop is now given by:

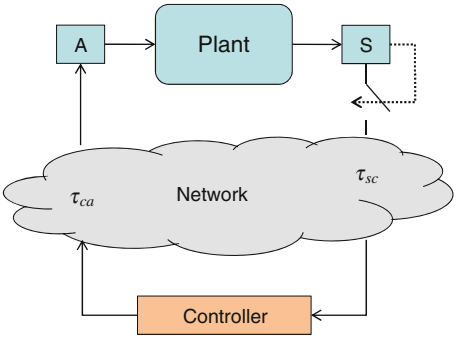

Fig. 5.3 Scheme of the asynchronous networked control system

$$x(k+1) = Ax(k) + BKx(\ell_a(k)), \tag{5.21}$$

where no external disturbances have been considered.

5.5.2 Remodeling the Node Dynamics

This section introduces the modifications needed to remodel the system dynamic equations according to the approach introduced above. Consider the round-trip communication, that is, the communication of the plant measurements from the sensor to the controller and the communication of the corresponding control signal from the controller to actuator. As Fig. 5.4 suggests, there are two possible situations with respect to the information received by the actuator:

- **Case 1** The last packet received by the actuator was sent by the sensor before $k - \tau_{max}$, so $\ell_a(k) < k - \tau_{max}$. Taking into account that the maximum delay is equal to τ_{max}, it is not possible that $\ell_s(k) \in \{\ell_a + 1, k - \tau_{max}\}$, because in that case the packet would be available at the actuator side at current time instant k. Recall that packets cannot be dropped or lost.
- **Case 2** The last packet received by the actuator was sent after $k - \tau_{max}$, so $\ell_a(k) \geq k - \tau_{max}$.

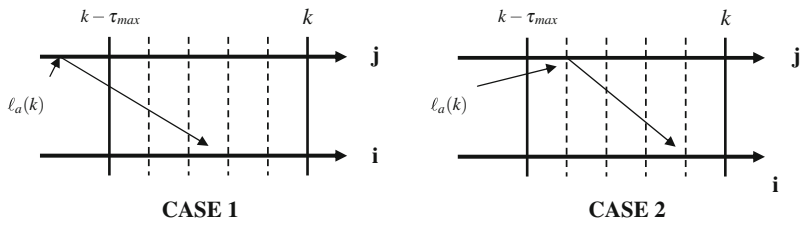

Fig. 5.4 Different cases regarding the transmission of information from node j to i

Taking into account these cases, the plant dynamics can be rewritten as

$$x(k+1) = Ax(k) + BKx(k - \mu(k)) + BK\eta(k), \tag{5.22}$$

where $\eta(k) = x(\ell_a(k)) - x(k - \mu(k))$ and $\mu(k)$ is defined as follows:

$$\mu(k) = \begin{cases} \tau_{max}, & \ell_a(k) < k - \tau_{max}, \\ k - \ell_a(k), & \ell_a(k) \geq k - \tau_{max}. \end{cases}$$

In the absence of external disturbances $w(k)$, the closed-loop dynamics in (5.22) is equivalent to the dynamics of the periodic communication case (5.3), with $\mu(k)$ playing the role of $\tau(k)$.[1] The only difference lies in the term $\eta(k)$, which can be interpreted as an external perturbation due to the discontinuous flow of information. This disturbance is reset to zero when the actuator applies to the plant a control input based on feedback belonging to the interval $\{k - \tau_{max}, k\}$.

Moreover, it is easy to see that $\|\eta(k)\|_\infty < \delta$ in both cases. In the Case 2, $\eta(k) = 0$. In the Case 1, $\eta(k) = x(\ell_a(k)) - x(k - \tau_{max})$, with $\ell_a(k) < k - \tau_{max}$. Since no packet was transmitted by the sensor in $\{\ell_a + 1, k - \tau_{max}\}$ (because, otherwise, this packet had been available in the actuator at current instant k), the triggering threshold has not been exceed before $k - \tau_{max}$, and thus $\|x(\ell_a(k)) - x(k - \tau_{max})\|_\infty < \delta$. Therefore, $\|\eta(k)\|_\infty < \delta$ holds for both cases.

5.5.3 Practical Stability for Delayed Asynchronous Systems

Next, the main result of this section is introduced. Given an H_2/H_∞ controller designed for the NCS (5.3) according to Theorem 5.1, the following result proves that, by implementing the event-based sampling policy described above, the state of the plant $x(k)$ can be ultimately bounded into an arbitrary small region that depends on the triggering threshold δ. The proof makes use of the quadratic structure of the LKF (5.17), that allows to write it as presented in Proposition 5.2.

The following theorem is developed particularizing for the LKF (5.17), although it is of general application.

Theorem 5.2 *Consider an H_2/H_∞ controller designed for the plant (5.3) by means of Theorem 5.1. Then, using an event-based communication with triggering condition (5.20), the state of the plant is GUUB with an ultimate bound given by*

$$\|x(k)\|_\infty \leq \delta \sqrt{\frac{\lambda^{\Psi}_{max}}{\lambda^{P}_{min}}} \left[(\|A\|_\infty + \|BK\|_\infty) D_1 + \|BK\|_\infty \right],$$

[1] It is straightforward to check that $\tau_{min} \leq \mu(k) \leq \tau_{max}$ for Cases 1 and 2.

where

$$D_1 = \frac{\|\Phi\|_\infty + \sqrt{\|\Phi\|_\infty^2 + \lambda_{min}^Q \|\Pi\|_\infty}}{\lambda_{min}^Q},$$
$$\Gamma = \begin{bmatrix} K^T B^T PA & K^T B^T PBK & 0 & 0 \end{bmatrix},$$
$$\Pi = K^T B^T PBK,$$

being Q any positive definite matrix such that $-Q > \Xi$, with Ξ defined in Proposition 5.2.

Proof Consider the Lyapunov–Krasovskii functional (5.17). Including the disturbances due to the asynchronous flow of information, the forward difference takes the form:

$$\Delta V(k) \leq \xi^T(k)\, \Xi\, \xi(k) + 2\eta^T(k)\Gamma\xi(k) + \eta^T(k)\Pi\eta(k).$$

From Theorem 5.1, matrix Ξ is negative definite, so there exists a positive matrix Q such that $\Xi < -Q$. Taking norms

$$\Delta V(k) \leq -\lambda_{min}^Q \|\xi(k)\|_\infty^2 + 2\|\Gamma\|_\infty \|\eta(k)\|_\infty \|\xi(k)\|_\infty + \|\Pi\|_\infty \|\eta(k)\|_\infty^2,$$

and taking into account the triggering condition (9.22), it yields

$$\Delta V(k) \leq -\lambda_{min}^Q \|\xi(k)\|_\infty^2 + 2\|\Gamma\|_\infty \delta \|\xi(k)\|_\infty + \|\Pi\|_\infty \delta^2.$$

Therefore, solving the second-order equation it can be ensured that $\Delta V(k) \leq 0$ for $\|\xi(k)\|_\infty > \delta D_1$, with $D_1 = \frac{\|\Gamma\|_\infty + \sqrt{\|\Gamma\|_\infty^2 + \lambda_{min}^Q \|\Pi\|_\infty}}{\lambda_{min}^Q}$.

For a generic vector σ and a positive scalar D, let B_D^σ denote the region of the space defined by $\{\sigma : \|\sigma\|_\infty \leq D\}$. Given that $V(k)$ is positive and decreasing for $\xi(k) \notin B_{\delta D_1}^\xi$, there exists a time instant k^* when $\xi(k^*)$ enters into the region $B_{\delta D_1}^\xi$. Since we are considering infinite norms and $x(k)$ is included in the augmented vector $\xi(k)$, it turns out that $x(k^*) \in B_{\delta D_1}^x$.

As $\xi(k^*) \in B_{\delta D_1}^\xi$ for any realization of $\mu(k) \in [\tau_{min}, \tau_{max}]$, it also holds that $\zeta(k^*) \in B_{\delta D_1}^\zeta$. Once $\xi(k^*)$ belongs to this region, the Lyapunov function is not necessarily decreasing and the augmented state may jump outside. In that case, $\xi(k^*+1) \notin B_{\delta D_1}^\xi$. Using the dynamics of the plant given in Eq. (5.22), it is possible to bound the plant state at instant k^*+1 by

$$\begin{aligned} \|x(k^*+1)\|_\infty &\leq \|A\|_\infty \|x(k^*)\|_\infty + \|BK\|_\infty \|x(k^* - \mu(k^*))\|_\infty + \|BK\|_\infty \|\eta(k^*)\|_\infty \\ &\leq [(\|A\|_\infty + \|BK\|_\infty) D_1 + \|BK\|_\infty]\, \delta. \end{aligned}$$

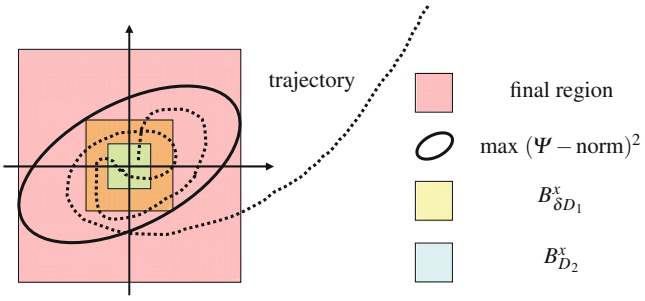

Fig. 5.5 Trajectory of plant state in a two-dimensional space

Then $x(k^*+1) \in B^x_{D_2}$, where $D_2 = [(\|A\|_\infty + \|BK\|_\infty)D_1 + \|BK\|_\infty]\,\delta$. Figure 5.5 illustrates a possible evolution of the observation error and the different regions.

In this way, $\xi(k^*+1)$ and $\zeta(k^*+1)$ may leave the regions $B^\xi_{\delta D_1}$ and $B^\zeta_{\delta D_1}$, respectively. Then, the Lyapunov–Krasovskii functional must be decreasing again, implying that $V(k) \leq \max V(k^*+1) = \max\{\zeta^T(k^*+1)\Psi\zeta(k^*+1)\}$. Therefore, $V(k) \leq \lambda^\Psi_{max} \max \|\zeta(k^*+1)\|^2_\infty \leq \lambda^\Psi_{max} D_2^2, \forall k > k^*+1$.

Finally, to get the final bound on $x(k)$ for $k > k^*+1$, note that all the terms of the Lyapunov functional involve positive definite matrices, so $x(k)^T P x(k) \leq V(k), \forall k$. Using fairy extended properties, it holds $\lambda^P_{min}\|x(k)\|^2_\infty \leq x(k)^T P x(k)$. Then, it yields $\|x(k)\|_\infty \leq \sqrt{\frac{\lambda^\Psi_{max}}{\lambda^P_{min}}}\, D_2$. This ends the proof.

Note that the final bound on $x(k)$ depends on the threshold δ that triggers the sampling. With $\delta = 0$, the event-based sampling becomes a periodic one and the asymptotic convergence will be accomplished. Furthermore, it is possible to use the theorem to find the suitable δ in order to achieve a prescribed final bound on the plant state and a trade-off between control performance and communication reduction.

Finally, it is worth mentioning that the optimization problem proposed to design the controller minimizes the maximum eigenvalue of Ψ. In other words, the optimal controller found contributes in the reduction of the final bound for the state.

5.6 Simulation Results

5.6.1 Example A

Consider the nominal discrete time system introduced in [39], in which we have included disturbances:

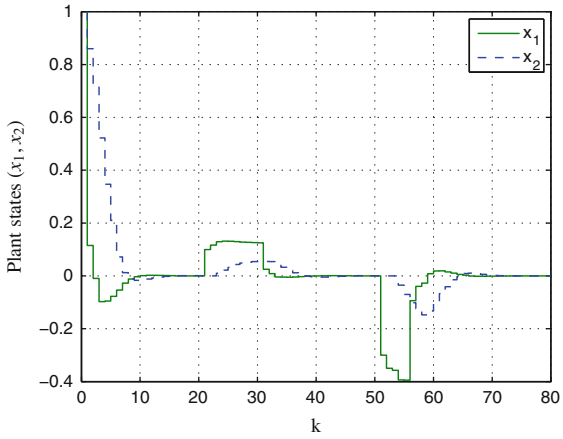

Fig. 5.6 Evolution of the state for $\tau_k \in [1, 2]$

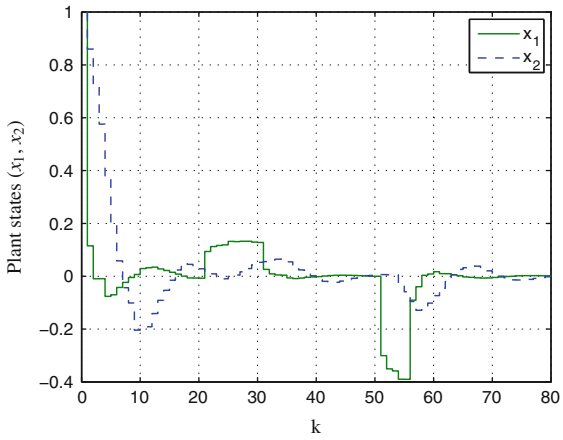

Fig. 5.7 Evolution of the state for $\tau_k \in [1, 4]$

$$x(k+1) = \begin{bmatrix} 1.01 & 0 \\ 0 & 1.2 \end{bmatrix} x(k) + \begin{bmatrix} -0.25 & 0.1 \\ 0 & 0.1 \end{bmatrix} x(k - \tau(k)) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} w(k),$$
$$z_2(k) = z_\infty(k) = x(k).$$

Let's analyze the effect of the delay in the evolution of the system. Choosing $\gamma < 1$ two controllers have been designed with Theorem 5.1 for different delay bounds:

(a) $\tau(k) \in [1, 2]$: The controller is $K = \begin{bmatrix} -0.1497 & -1.0353 \end{bmatrix}$.

(b) $\tau(k) \in [1, 4]$: The controller is $K = \begin{bmatrix} -0.0120 & -0.9411 \end{bmatrix}$.

Figures 5.6 and 5.7 show the system's response for both cases when the disturbances have been taken as:

$$w(k) = \begin{cases} w(k) = 1 & \forall k \in [20, 30]. \\ w(k) = -3 & \forall k \in [50, 55]. \\ w(k) = 0 & \text{otherwise.} \end{cases}$$

It can be observed how the controller satisfactorily drives the system to the equilibrium point, despite delays and disturbances. As expected, performance slightly degrades when the delays are bigger.

5.6.2 Example B

Consider the networked control system introduced in [265]:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 1.0 & 0.01 \\ 0.5 & 0.7 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} Kx(k - \tau(k)) + \begin{bmatrix} 0.01 \\ 0.01 \end{bmatrix} w(k), \\ z_2(k) &= \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \\ 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 0.1 \end{bmatrix} Kx(k - \tau(k)), \\ z_\infty(k) &= \begin{bmatrix} 0.1 & 0.1 \end{bmatrix} x(k). \end{aligned}$$

It is assumed that the network-induced delay are given by $\underline{\tau}_{sa} = 1$ and $\overline{\tau}_{sa} = 3$. The maximum number of consecutive data losses is bounded by $n_p = 3$.

Choosing $\gamma < 1$, Theorem 5.1 designs the controller $K = \begin{bmatrix} -0.8457 & -0.0360 \end{bmatrix}$. Next, we introduce the event-based communication policy between sensor and controller. For the same controller designed above and assuming $n_p = 0$, Fig. 5.8 shows the evolution of the system and the events for the threshold $\delta = 0.2$. Note that most of the events have been triggered in the transient. When the system is evolving near the equilibrium point, the amount of events decreases.

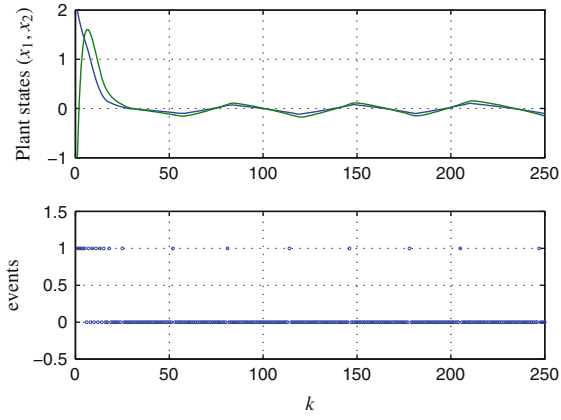

Fig. 5.8 *Up* Evolution of the state. *Down* Events triggered ($\delta = 0.2$): Each event is marked with an ∘

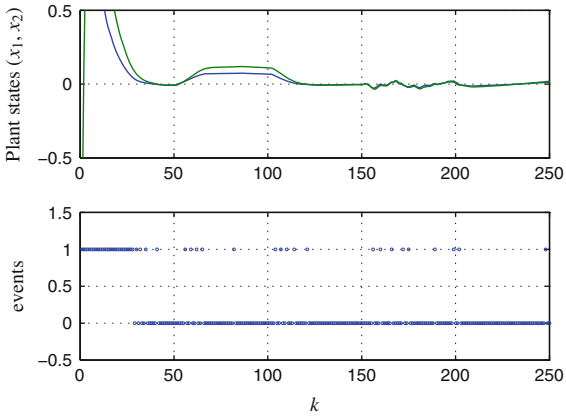

Fig. 5.9 *Up* Evolution of the state. *Down* Events triggered ($\delta = 0.02$): Each event is marked with an ∘

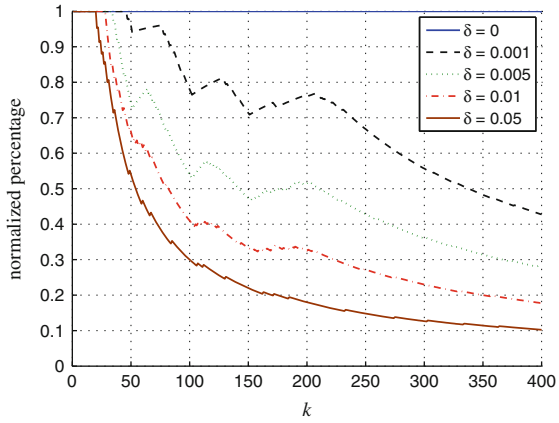

Fig. 5.10 Percentage of packets transmitted with respect to the periodic case for different thresholds δ

Assume that the following exogenous disturbances are included:

- Constant disturbances for $k \in [50, 100]$.
- Random disturbances for $k \in [150, 200]$.

Figure 5.9 illustrates this situation. In this case the threshold has been chosen as $\delta = 0.02$. As expected, the presence of disturbances triggers more events than in the unperturbed situation.

Finally, Fig. 5.10 develops a comparison between different choices of δ and the corresponding transmission rate. In case of $\delta = 0$, a 100 % of packets must be send from the sensor. Choosing $\delta = 0.05$ only a 10 % of packets must be sent through the network.

5.7 Conclusions

This chapter is devoted to the design of H_2/H_∞ controllers for discrete networked control systems. Lemma 5.1 proposes a general design method which can be used with different choices of the Lyapunov–Krasovskii functional.

Furthermore, it has been considered the implementation of an asynchronous communication policy, aiming at reducing the number of packets transmitted through the network and the energy expenditure. It has been shown that the practical stability of the system is preserved using the event-based sampling policy. Finally, Theorem 5.2 relates the size of the final region for the state and the threshold that triggers the events.

Chapter 6
Asynchronous Packetized Model Predictive Control

Isabel Jurado and Pablo Millán

6.1 Introduction

Although some of the most extended communication protocols use retransmissions to preserve the information integrity, it is well known that this is not very useful for NCSs, where the need of rigorous bounds for time delays to guarantee stability leads into the practical decision of discarding all the information that arrives later than a certain delay. This is the reason why, in protocols like Ethernet, packet dropouts must be considered.

One solution to this problem is to include more information in the packet, which leads into a lesser rate of packets sent through the network, see [217, 252, 261, 269]. The information included in these bigger packets can be a set of predictions on future control signals. With these values in a buffered actuator, the system can tolerate larger sampling times and more dropouts without compromising its stability. This idea is used, for instance, in [32, 210], using model predictive control (MPC) to compute the predicted control signals.

This chapter, based on the conference contribution [208], uses these ideas in order to improve the traffic over the network and also to reduce the energy consumption, a main objective for the particular case of wireless networks. A model predictive controller is proposed within an asynchronous control architecture that deals with disturbances and data dropouts while providing good performance. Packets are sent in an asynchronous manner to reduce the number of information exchanges. The triggering condition to send a packet is that the difference between the control values

I. Jurado (✉) · P. Millán
Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería,
Universidad Loyola Andalucía, Seville, Spain
e-mail: ijurado@uloyola.es

P. Millán
e-mail: pmillan@uloyola.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_6

at the controller side and those in the actuator buffer exceeds certain threshold. That way, information is only transmitted when it is needed. In order to deal with data dropouts in the system-to-controller link, a state estimator is included at the controller side.

The chapter is organized as follows: Section 6.2 describes the predictive control scheme. In Sect. 6.3 an example illustrating the control algorithm is presented and in Sect. 6.4 conclusions are drawn.

6.2 Networked Predictive Control Algorithm

This section describes the predictive control structure for the NCS. First of all, there is a definition of the problem and then, a description of the packetized control and buffering strategy.

6.2.1 Problem Setup

The systems under consideration are unconstrained discrete-time linear multiple-inputs systems affected by bounded disturbances:

$$x(k+1) = Ax(k) + Bu(k) + B_2 w(k) \tag{6.1}$$

with $k \in \mathbb{N}_0 \triangleq \mathbb{N} \cup \{0\}$ and

$$u(k) \in \mathbb{U} \subseteq \mathbb{R}^m, \quad x(k) \in \mathbb{X} \subseteq \mathbb{R}^n, \quad \forall k \in \mathbb{N}_0$$

disturbances, $w(k)$ are considered to be bounded as

$$w(k) \in \mathbb{W}, \quad \mathbb{W} = \{x \in \mathbb{R}^{n_w} / \|x\| < w_{max}\}. \tag{6.2}$$

The proposed control scheme is shown in Fig. 6.1. It can be seen that the plant and the controller are linked through a communication network in both ways, controller outputs to plant inputs (actuators), and plant outputs (sensors) to controller inputs. The network is supposed to be clock-driven Ethernet-like.

In this chapter, the main problems to take into account are transmission delays and packet dropouts. In particular, since round-trip communication delays are assumed small enough with respect to the sampling time, it is possible to disregard them. In case the packets arrive with a delay larger than certain threshold, or do not arrive, they are treated as packet dropouts.

It is assumed that the network is not secured either in controller to actuator or in sensor to controller links. Also, the protocol is assumed to be TCP-like, that implies

that acknowledgments are available. Therefore, at time k the controller knows the control signal applied to the plant.

The following assumptions are applied throughout all the chapters:

- Time-driven sensors sample periodically the state of the plant.
- A time-driven predictive controller computes a control sequence at each sampling time.
- A time-driven actuator buffers control signals and applies them to the plant at each sampling time.
- Network introduces packet dropouts at any point.
- Delayed packets are discarded and taken as dropouts.

With the state of the system, at every sampling time the predictive controller computes a finite horizon optimal control sequence $U_k \in (\mathbb{U})^{N_u}$, being N_u the finite prediction horizon. These control values are calculated in such a way that the following objective function is minimized:

$$V(U_k, k) = \sum_{i=k}^{k+N_u-1} \ell(x'(i), u'(i)) + F(x'(k+N_u)) \tag{6.3}$$

where $x'(\cdot)$ and $u'(\cdot)$ are predicted plant states and inputs, $\ell(\cdot)$ denotes the *stage cost*, and $F(\cdot)$ is the *terminal cost*.

6.2.2 Packetized Control and Buffering Strategy

The objective of this section is to provide the system robustness against packet dropouts. To do that, an appropriate buffering and a queuing strategy at the actuator are added to the predictive control structure.

The objective of this structure is to use the control values in the buffer as a backup in case there is a dropout, this technique is also used in [49, 261, 269]. This situation is illustrated in Fig. 6.1. In case of dropouts, the actuator uses the appropriate predicted control signal contained in the buffer.

The buffer follows the subsequent behavior: it is updated whenever a packet arrives. Therefore, the actuator uses information from the buffer, which is a prediction, if there has been a dropout in the controller–actuator link, or the actual control signal calculated at the same instant as it is required by the actuator. This corresponds to the intuitively appealing idea of *Use the most recent control sequence if available. If not, use predictions from the buffer.*

The maximum number of consecutive dropouts that the buffer can compensate for is equal to its size. Taking that into account, the reasonable thing to do is to make coincide the buffer's size with the length of the vector of control signals, that is, the prediction horizon N_u.

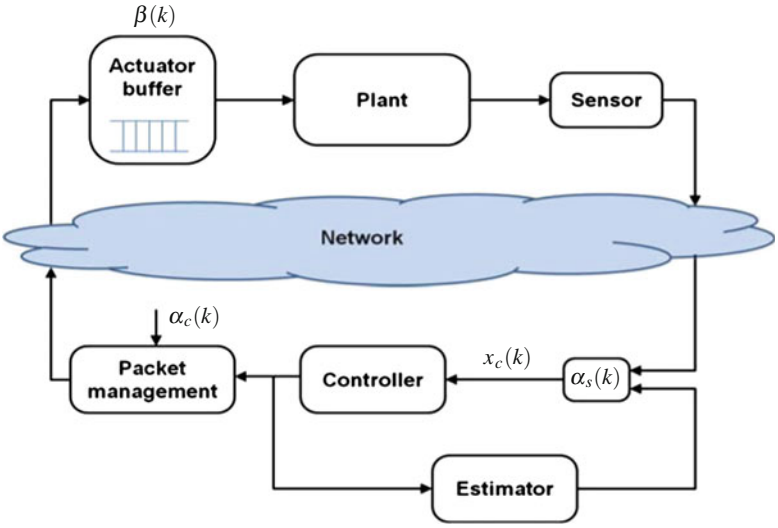

Fig. 6.1 NCS proposed scheme

The dynamics of the buffer can be represented by the following recursive rule:

$$\beta(k) = \alpha_c(k)U_k + (1 - \alpha_c(k))S\beta(k-1), \tag{6.4}$$

where $\beta(k) \in (\mathbb{U})^{N_u}$ is the state of the buffer at instant k, and matrix $S \in \mathbb{R}^{m \cdot N_u \times m \cdot N_u}$ is a shift matrix defined as the block matrix

$$S_{i,j} = \delta_{i+1,j} \cdot I_{p_1} \quad i, j = 1, \ldots, N_u,$$

with $\delta_{i,j}$ the Kronecker delta symbol.

It also needed a variable that counts the reception of data from the controller, that is $\alpha_c(k) \in \{0, 1\}$:

$$\alpha_c(k) = \begin{cases} 1 & \text{if packet } U(k) \text{ arrives to buffer at time } k \\ 0 & \text{if packet } U(k) \text{ does not arrive to buffer at time } k \end{cases}$$

Using the definition of the buffer's dynamics in (6.4), the control action that actually applied to the plant at instant k can be expressed as:

$$u(k) = \left[I_m \; 0_m \; \ldots \; 0_m \right] \beta(k)$$

This strategy implies that the controller transmits the whole control sequence of length N_u at every sampling time. Nevertheless, there are some situations that require a reduced access to the network. That is the case, for instance, of wireless sensor networks, which is considered to be one of the most energy demand processes in

networked control devices, [179]. In this kind of network, it is very important to keep sensors' batteries loaded as long as possible, and thus energy saving is a main objective. The best thing to do in this kind of systems is to design a network protocol that reduces to its minimum the use of the network. That implies the minimization of packet transmissions, sending information only when it is relevant for control purposes.

In order deal with this type of constraints, asynchronous communication is included in the control scheme. These constraints avoid the packets to be transmitted if the information that they contain do not differ in certain threshold from the one currently in the buffer. This comparison is done, thanks to the acknowledgment messages that the controller receives from the actuator, $\alpha_c(k)$. That way, the controller can know the control sequence stored in the buffer at time k, $\beta(k)$, and if this sequence is significantly different from the current computed sequence $U(k)$. If this is not the case, then $U(k)$ is discarded and it is not transmitted through the network.

It is also possible that only certain components of $U(k)$ differ enough from the corresponding values of $\beta(k)$. In this situation, only that components are sent, reducing the size of the packet to be transmitted. This *trimmed packet* only contains control actions that differ more than certain threshold from those in the buffer.

The *packet management policy* can be formalized as follows: let $\beta(k)$ be the state of the buffer and $U(k)$ the control sequence computed at instant k. Let $\beta_j(k)$ and $U_j(k)$ be, respectively, the jth component of these sequences. The size of the trimmed packet to send, $N_T(k)$, at time instant k, can be calculated as:

$$N_T(k) = N_u - \arg \min_{j \in \{1,\ldots,N_u\}} \|U_j(k) - \beta_j(k)\| > \delta. \tag{6.5}$$

That means that only the last $N_T(k)$ components of the control sequence $U(k)$ are sent to the buffer, since difference between the first $N_u - N_T(k)$ control values are below the threshold δ.

Note that the length of the sequence to send, $N_T(k)$, is a function of the time instant k, which means that it can vary each sampling time depending on the values of $U(k)$ and $\beta(k)$. Also, $N_T(k) \leq N_u \ \forall k$, since it is not possible that the length of the trimmed packet exceed the length of $U(k)$.

Using definition (6.5), a trimmed packet $U^*(k) \in (\mathbb{R})^{N_T(k)}$ can be written as:

$$U^*(k) = \left[0_{mN_T(k) \times m(N_u - N_T(k))} \ I_{mN_T(k)} \right] U(k).$$

This implies that the buffer dynamics expression in (6.4) can be modified, including trimmed packets $U^*(k)$, as:

$$\begin{aligned} \beta(k) = \alpha_c(k) \Big[\beta_1^T(k-1), \ldots, \beta_{N_u - N_T(k)}^T(k-1), (U_1^*(k))^T, \ldots, (U_{N_T(k)}^*(k))^T \Big]^T \\ + (1 - \alpha_c(k)) S \beta(k-1). \end{aligned} \tag{6.6}$$

6.2.3 State Estimator Description

As it was described before, the technique proposed in this chapter considers the possibility of losing packets between the sensor and the controller. This situation, which is not treated in most of the predictive approaches in the literature, allows us to work with more realistic networked control systems.

In order to deal with this kind of losses, we propose a model-based estimator included at the controller side that approximates plant states when the information from the sensor is not received. The estimator is designed in the following way:

$$x_m(k+1) = \alpha_s(k)x(k) + (1-\alpha_s(k))f(x_m(k), u(k)), \tag{6.7}$$

where $x_m(k) \in \mathbf{R}^n$ is the estimated plant state at instant k, and $f(x_m(k), u(k))$ is an open-loop approximation of the plant dynamics. Knowing the plant dynamics (6.1), $f(x_m(k), u(k))$ takes the form:

$$f(x_m(k), u(k)) = Ax_m(k) + B_1u(k).$$

The signal $\alpha_s(k)$, similar to $\alpha_c(k)$ in (6.4), accounts for the reception of packets from the sensor at the controller side. Therefore, $\alpha_s(k)$ takes the following values:

$$\alpha_s(k) = \begin{cases} 1 & \text{if packet } x(k) \text{ arrives to the controller at time } k \\ 0 & \text{if packet } x(k) \text{ does not arrive to the controller at time } k. \end{cases}$$

It is straightforward to see from Eq. (6.7) that the state estimation $x_m(k)$ is updated with measured values of the plant state $x(k)$ every time this information arrives to the controller. Nonetheless, if a dropout occurs, an open-loop estimation is performed using a model of the plant dynamics. This way, every time k the controller is fed with the state of the system, measured or estimated, regardless of network failures.

Therefore, the complete networked control scheme consists of the state estimator joined with the packet buffering strategy. It is also worth mentioning that the predictive controller can be designed without considering problems associated with network communications. This implies that the proposed approach can be interpreted like a network compensation technique rather than a controller design.

6.2.4 Stability Considerations

In this section, the mild requirements needed to ensure the stability of the compensation technique are presented.

Suppose that $u(k)$ is a stabilizing predictive control value for the system (6.1), computed at instant k. As it was mentioned before, the controller design does not take into account the presence of a network in the control loop. The consideration of

such dropouts makes uncertain the application of the control $u(k)$ at time instant k. The control signal actually applied to the plant, $u_c(k)$, depends on the dropouts and it is based on the predictions in the buffer and on the state estimations.

This way, the effects of the network on the system (6.1) can be expressed in the following manner:

$$\begin{aligned} x(k+1) &= Ax(k) + B_1u(k) + B_1(u_c(k) - u(k)) + B_2w(k) \\ &= Ax(k) + B_1u(k) + B_1\varepsilon_u(k) + B_2w(k), \end{aligned} \tag{6.8}$$

where $\varepsilon_u(k) = u_c(k) - u(k)$ is a disturbance term that represents the network effect on the control structure.

It also easy to see that this additional term, $\varepsilon_u(k)$, is bounded. Note that, when there is no dropout, the buffer and the estimator are reset to match the control sequence computed by the controller and the actual state measured by the sensor, and therefore, $\varepsilon_u(k) = 0$. Knowing that, by assumption, the number of consecutive packet dropouts is limited, the difference between the calculated control signal $u(k)$ and the applied one $u_c(k)$ can only increase between successful transmissions. Therefore, this implies that the value $\varepsilon_u(k)$ is bounded.

Since $\varepsilon_u(k)$ is bounded, the overall disturbance term $\Omega(k) = B_1\varepsilon_u(k) + B_2w(k)$ is also bounded.

It is known from [62], that, under certain conditions, a stabilizing controller can always be found for the unperturbed system (6.8). Also, from [48], if the following two assumptions hold:

(A1) The closed loop dynamics of the unperturbed system (6.8) is $x(k+1) = F(x(k))$, where the origin is a fixed point.

(A2) $V(x)$ is a Lyapunov function of the system Lipschitz in a neighborhood of the origin $\Lambda_r = \{x \in \mathbb{R}^n / V(x) \leq r\}$ such that

$$\begin{aligned} a \cdot \|x\|^p \leq V(x) \leq b \cdot \|x\|^p \\ V(F(x)) - V(x) \leq -c \cdot \|x\|^p \end{aligned} \tag{6.9}$$

where a, b, c are positive constants and $p > 1$.

Then, there exits a constant $\mu > 0$ such that for all disturbances

$$\Omega(k) \in B_\mu = \{\Omega(k) \in \mathbb{R}^n / \|\Omega(k)\| < \mu\},$$

the perturbed system $x(k+1) = F(x(k)) + \Omega(k)$ is asymptotically ultimately bounded $\forall x(0) \in \Lambda_r$.

Finally, the stability of the presented methodology can be ensured since conditions (A1) and (A2) are satisfied for system (6.8) taking $p = 2$, and choosing a Lyapunov function of the form $V(x) = x^T Px$, which is Lipschitz in a neighborhood of the origin Λ_r for arbitrarily large values of r.

6.3 Application Example

The proposed methodology has been tested by simulation over the three-tank system shown in Fig. 6.2.

The control problem is to track references on the water level of the third tank, being the flow rate into the first tank of our control input.

6.3.1 Modeling

The model of the system can be obtained from a mass balance:

$$\begin{aligned}\frac{dh_1}{dt} &= \frac{1}{S}\,q - \frac{1}{S}\,C_1\sqrt{h_1 - h_2} \\ \frac{dh_2}{dt} &= \frac{1}{S}\,C_1\sqrt{h_1 - h_2} - \frac{1}{S}\,C_2\sqrt{h_2 - h_3} \\ \frac{dh_3}{dt} &= \frac{1}{S}\,C_2\sqrt{h_2 - h_3} - \frac{1}{S}\,C_3\sqrt{h_3},\end{aligned} \tag{6.10}$$

where h_i represent the level of tank i.

For control purposes, this model is linearized around an equilibrium point H_1, H_2, H_3, and Q. That is:

$$\begin{aligned} h_1 &= H_1 + \Delta H_1,\ h_2 = H_2 + \Delta H_2 \\ h_3 &= H_3 + \Delta H_3,\ \ q = Q + \Delta Q,\end{aligned}$$

yielding the linear equation:

$$\Delta\dot{H} = L\,\Delta H + M\,\Delta Q, \tag{6.11}$$

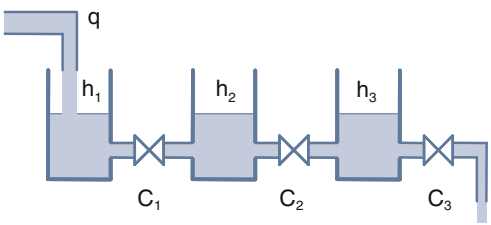

Fig. 6.2 Three-tank system

where

$$\Delta H = \begin{bmatrix} \Delta H_1 & \Delta H_2 & \Delta H_3 \end{bmatrix}^T$$

$$L = \begin{bmatrix} \frac{-C_1}{2S\sqrt{H_1 - H_2}} & \frac{C_1}{2S\sqrt{H_1 - H_2}} & 0 \\ \frac{C_1}{2S\sqrt{H_1 - H_2}} & \frac{-C_1}{2S\sqrt{H_1 - H_2}} - \frac{C_2}{2S\sqrt{H_2 - H_3}} & \frac{C_2}{2S\sqrt{H_2 - H_3}} \\ 0 & \frac{C_2}{2S\sqrt{H_2 - H_3}} & \frac{-C_2}{2S\sqrt{H_2 - H_3}} - \frac{C_3}{2S\sqrt{H_3}} \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{1}{S} & 0 & 0 \end{bmatrix}^T.$$

A discrete model is then easily obtained from (6.11) as

$$x(k+1) = Ax(k) + B_1 u(k). \tag{6.12}$$

6.3.2 Results

Applying the proposed control structure to the three-tank model, some simulations have been carried out for different network situations.

In particular the chosen parameters for the model are: $S = 0.16\,\mathrm{m}^2$, $C_1 = C_2 = 0.0256\,\frac{\mathrm{m}^3}{\mathrm{s\,m}^{1/2}}$, and $C_3 = 0.0251\,\frac{\mathrm{m}^3}{\mathrm{s\,m}^{1/2}}$, with an operation point $H_1 = 1\,\mathrm{m}$, $H_2 = 0.7\,\mathrm{m}$, $H_3 = 0.4\,\mathrm{m}$, $Q = 0.014\,\mathrm{m}^3/\mathrm{h}$.

Figure 6.3 shows a comparison of the proposed methodology to alleviate traffic in the network with the case without any traffic consideration, that is, the whole control sequence is sent through the network at every sampling time. This graphic shows the influence of the data dropout rate p and the threshold to transmit packets δ on the average of the *integral square error* (ISE).

This plot has been obtained taking a step-like sequence with period $T = 2000\,\mathrm{s}$ as reference, and a simulation time equal to 5000 s.

The reduction of packet transmissions from the controller to the actuator is shown in Fig. 6.4. This reduction is computed as the amount of information transmitted with the proposed methodology compared to the case where all the information is sent at every sampling time. It can be seen how savings above 85 % can be obtained for certain values of the threshold for communication δ. Nevertheless, it is necessary to be cautious choosing δ since, as can be seen in Fig. 6.3, excessively high values degrade the system performance faster than transmission saving. For example, from the view of Figs. 6.3 and 6.4, a reasonably good choice could be $\delta = 0.2 \cdot 10^{-4}$, as it provides a reduction of 70 % in the transmission without a significant deterioration of the system performance.

In Figs. 6.5 and 6.6, the response of the system is shown for the particular case of 30 % packet dropout probability. It can be observed how the proposed method still provides good performance even under these network conditions. However, the effect of the dropouts can be noticed since response exhibits small overshoot.

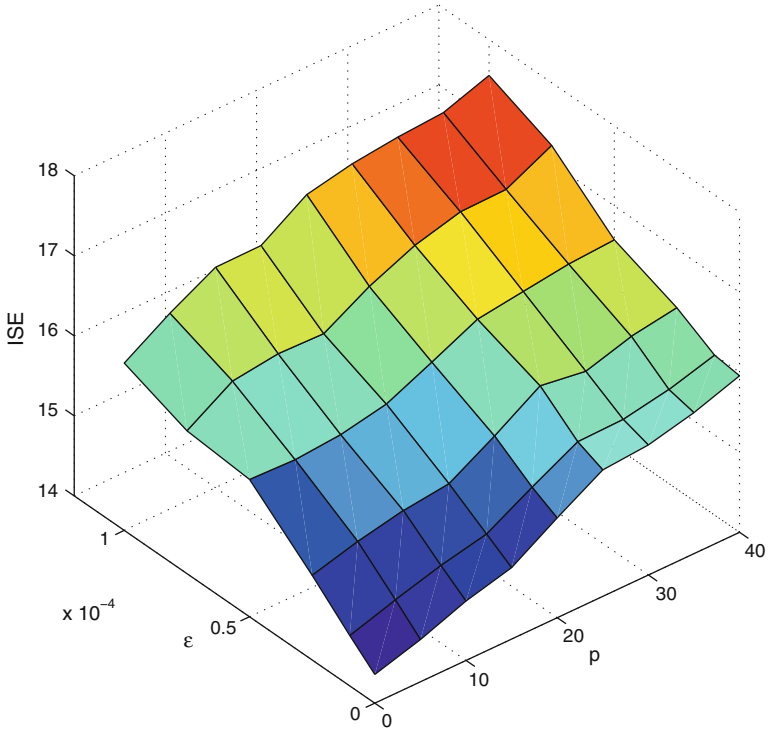

Fig. 6.3 Influence of allowed error

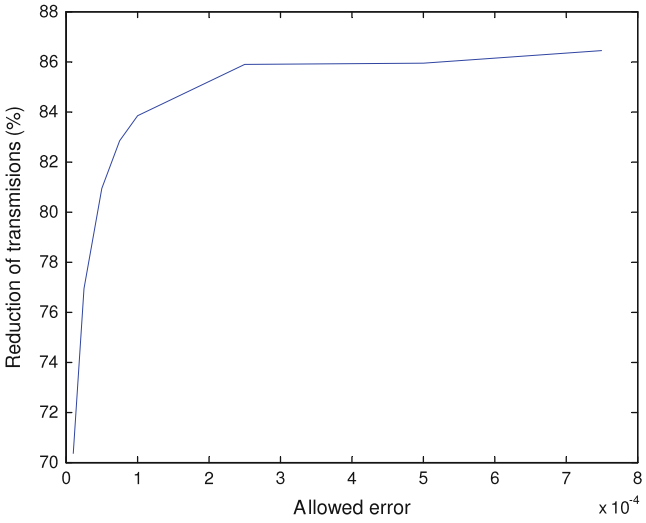

Fig. 6.4 Reduction of transmitted data

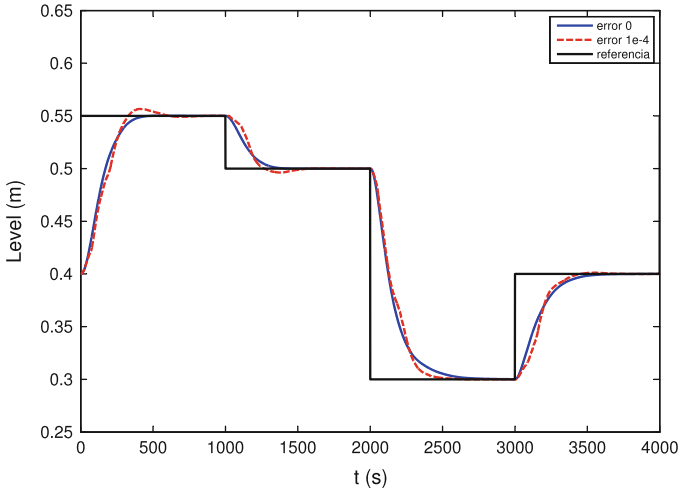

Fig. 6.5 Step response

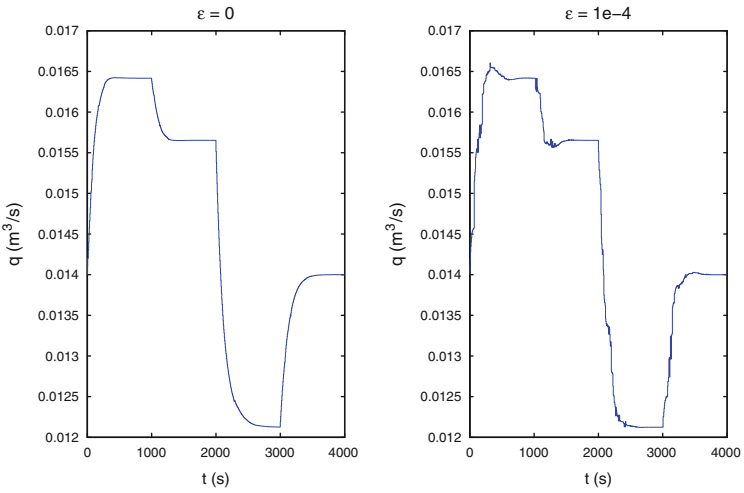

Fig. 6.6 Tracking of references

As mentioned before, the control performance gets worse as the threshold δ, or the data dropout rate p, increases. In this case, the controller admits data dropout rates above 40 %.

Finally, Fig. 6.7 shows the transmission profile for a step tracking experiment with different values of δ. As expected, at the beginning of the simulation, for the instants corresponding to the transient regime, the flow of transmission is more intense than in subsequent instants, when the system reaches the steady state. Then, the traffic over the network reduces.

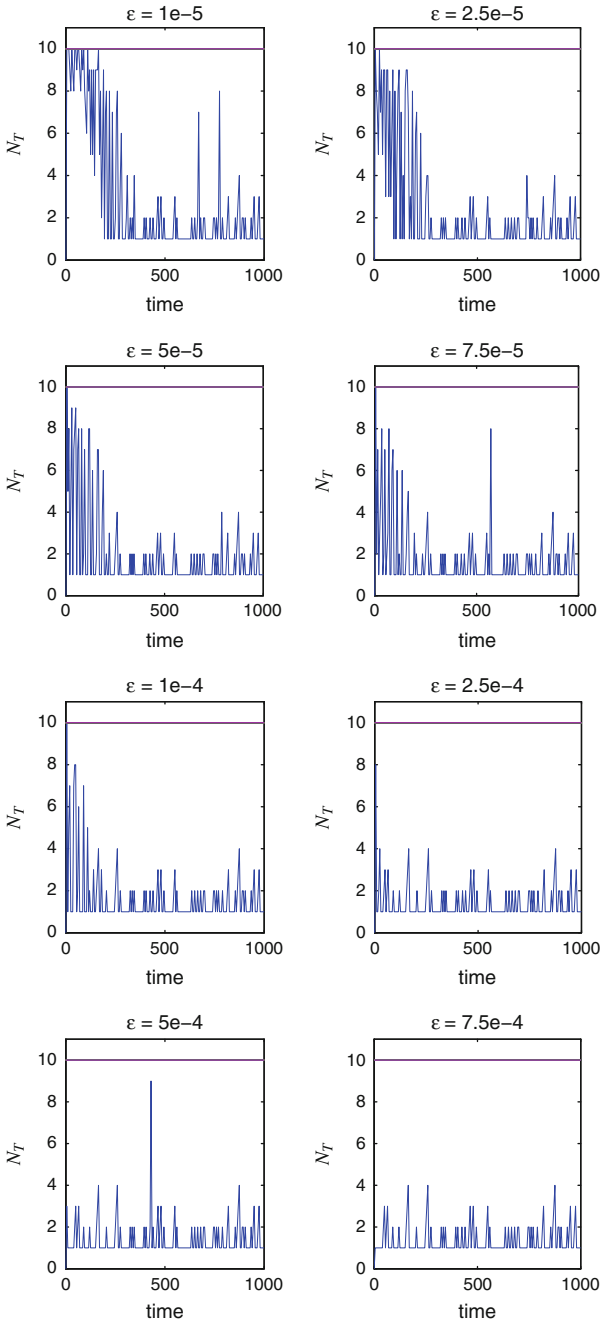

Fig. 6.7 Number of components of $U^*(k)$ sent

6.4 Conclusions

This chapter has presented an asynchronous predictive control scheme to deal with data dropouts and delays in networked control systems. The use of a MPC controller and a buffering strategy is joined with a model-based state estimator, which allows the controller to cope with packet losses and to reduce network traffic.

Finally, some simulations have been presented for data dropout rates up to 40 %. The results have shown that the proposed methodology can cope with stringent network conditions without significant performance degradation, and at the same time can provide a reduction in the network traffic load with respect to conventional predictive control architectures.

Part II
Asynchronous Control and Estimation for Large-Scale Plants. Distributed Solutions

Six chapters form this block. Chapters 7–9 present distributed solutions for asynchronous control and estimation when network effects can be neglected. Next, the work is extended to deal with delays and packet losses in Chaps. 10 and 11. Finally, Chap. 12 presents an application example of these techniques to a multi-robot system.

Chapter 7
Distributed Event-Based Control for Interconnected Linear Systems

María Guinaldo, Dimos V. Dimarogonas, Daniel Lehmann and Karl H. Johansson

7.1 Introduction

One way to study the control properties of large-scale systems is to consider that the plant is composed of interconnected systems. The motivation for this assumption is twofold. On the one hand, physical plants are made up of parts, which can be identified as different subsystems, and this structural feature can facilitate the control design. On the other hand, even if the system does not present these physical boundaries, it might be useful to decompose it into mathematical subsystems which have no obvious physical identity. These terms of physical and mathematical decomposition were first introduced by Siljak [236], and since then they have been used in the design of centralized and distributed controllers.

Practical examples of these large-scale systems are power or traffic networks, in which a centralized solution would require a very powerful network and an accurate model of all the interconnections, and moreover, it would be not robust against node failures, for example. The design of decentralized controllers for this kind of systems is a suboptimal solution since it does not take into account the interconnection between the subsystems. Hence, there is a natural interest in applying distributed

M. Guinaldo (✉)
Dpto. de Informática y Automática, Escuela Técnica Superior de Informática, UNED, Madrid, Spain
e-mail: mguinaldo@dia.uned.es

D.V. Dimarogonas, D. Lehmann and K.H. Johansson
School of Electrical Engineering, Royal Institute of Technology (KTH), Stockholm, Sweden
e-mail: dimos@kth.se

D. Lehmann
e-mail: dlehmann@kth.se

K.H. Johansson
e-mail: kallej@kth.se

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_7

control to these scenarios, and, if the communication between the local controllers is event triggered, get better usage of the network.

There are some recent contributions on distributed event-triggered control [51, 54, 88, 166, 232, 259]. The basic idea in all these contributions is that each subsystem decides when to transmit the measurements based only on local information. In the most common implementations, an event is triggered when the error of the system exceeds a tolerable bound.

This chapter discusses different control strategies of distributed event-based controls for linear interconnected systems. Part of these results are based on the contributions [88, 89, 91]. Section 7.2 provides the mathematical tools used through the chapter as well as the problem statement. Different distributed trigger functions are examined in Sect. 7.3: deadband control, Lyapunov approaches, and exponential bounds, which is the proposal of the authors to the studied problem. Other existing strategies such as, for example, small-gain approaches [51] do not prevent from Zeno behavior, and a constant threshold-like condition must be included to overcome this issue, yielding similar results to the deadband control from the analytical point of view.

The analytical results are provided in Sect. 7.4. Two aspects are analyzed: Convergence to the equilibria and inter-event times, and the results are illustrated with an example in Sect. 7.4.3. The extension to discrete-time systems is given in Sect. 7.5.

Model-based approaches has been shown to help to reduce communication in centralized schemes (see Chaps. 4 and 6). Thus, one of the first improvements presented in Sect. 7.6 consists of a distributed model-based approach combined with event-triggered communications. However, reducing the number of transmissions in the network is not the only aspect that matters in distributed systems. For instance, the frequency of the control update allows a more efficient usage of the limited resources of embedded microprocessors. Whereas in a single control loop the reduction of communication usually implies the reduction of actuator updates, this does not necessary hold in distributed systems, especially if the number of neighbors is large. Thus, the second improvement presented in Sect. 7.6 accounts for both phenomena in the design.

7.2 Background and Problem Statement

7.2.1 Matrix and Perturbations Analysis

Let $A \in \mathbb{C}^{n\times n}$ be a complex matrix, and let us define

$$\kappa(A) = \|A\|\|A^{-1}\| \quad (0 \notin \lambda(A)), \tag{7.1}$$

$$\alpha_{max}(A) = \max\{\mathbb{R}e(\lambda) : \lambda \in \lambda(A)\}, \tag{7.2}$$

The matrix exponential of A is defined as $e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$. Through this chapter, the stability of the system is proved using some hints that are summarized in this section to bound $\|e^{At}\|$.

7.2.1.1 Bounding the Matrix Exponential

In [245] various norms are discussed to bound the exponential. Three are of particular interest:

- **Log norms** If $\mu_{max}(A)$ is defined as $\mu_{max}(A) = \max\{\mu : \mu \in \lambda((A + A^*)/2)\}$, then
$$\|e^{At}\| \leq e^{\mu_{max}(A)t}.$$
An interesting corollary can be inferred from the property above. Let Y be an invertible matrix such that $A = YBY^{-1}$. It follows that
$$\|e^{At}\| = \|Ye^{Bt}Y^{-1}\| \leq \kappa(Y)e^{\mu_{max}(B)t}, \tag{7.3}$$
where $\kappa(Y)$ is defined according to (7.1).
Thus, assume that A is *diagonalizable*, i.e., there exists a matrix D, where $D = \text{diag}(\lambda_i(A))$, and a matrix V of eigenvectors, such that $A = VDV^{-1}$. From (7.3), it holds that
$$\|e^{At}\| \leq \kappa(V)e^{\mu_{max}(D)t} = \kappa(V)e^{\alpha_{max}(D)t} = \kappa(V)e^{\alpha_{max}(A)t}, \tag{7.4}$$
where $\alpha_{max}(A)$ is defined according to (7.2).
- **Jordan canonical form** Recall the Jordan decomposition theorem which states that if $A \in \mathbb{C}^{n\times n}$, then there exists an invertible matrix $X \in \mathbb{C}^{n\times n}$ such that
$$X^{-1}AX = J_{m_1}(\lambda_1) \times \cdots \times J_{m_1}(\lambda_p) \equiv J,$$
where
$$J_k \equiv J_{m_k}(\lambda_k) = \begin{pmatrix} \lambda_k & 1 & & 0 \\ 0 & \lambda_k & \ddots & \\ \vdots & & \ddots & 1 \\ 0 & 0 & \dots & \lambda_k \end{pmatrix} \in \mathbb{C}^{m_k\times m_k}, \quad k = 1, \dots, p.$$
By taking norms and defining $m = \max\{m_1, \dots, m_p\}$, it can be proved that [245]
$$\|e^{At}\| \leq m \cdot \kappa(X)e^{\alpha_{max}(A)t} \max_{0\leq r\leq m-1} \frac{t^r}{r!}. \tag{7.5}$$

Note that X may not be unique but it is assumed that it is chosen such that $\kappa(X)$ is minimized.

- **Schur decomposition bound** The Schur decomposition states that there exists a unitary $Q \in \mathbb{C}^{n\times n}$ such that

$$Q^* A Q = D + N, \tag{7.6}$$

where D is the diagonal matrix $D = \text{diag}(\lambda_i)$ and N is strictly upper triangular. The following upper bound can be obtained [245]

$$\|e^{At}\| \leq e^{\alpha_{max}(A)t} \sum_{k=0}^{n-1} \frac{\|Nt\|^2}{k!}. \tag{7.7}$$

7.2.1.2 Perturbation Bounds

The second aspect that is brought up in this section is the existing perturbation analysis on the eigenvalues and the matrix exponential, i.e., how the eigenvalues and the bound on the matrix exponential change when A is perturbed by E.

The following lemma merges classical results from [17, 44] to study the perturbation of the eigenvalues of a matrix A in two situations: when A is diagonalizable and when it is not.

Lemma 7.1 *If A is diagonalizable ($V^{-1}AV = D$), the eigenvalues $\tilde{\lambda}_i$ of $A + E$ satisfy*

$$\min_{\lambda_j \in \lambda(A)} |\tilde{\lambda}_i - \lambda_j| \leq \kappa(V)\|E\|. \tag{7.8}$$

Otherwise, Let consider the Schur decomposition (7.6). Then for $\tilde{\lambda}_i \in \lambda(A + E)$

$$\min_{\lambda_j \in \lambda(A)} |\tilde{\lambda}_i - \lambda_j| \leq max\{\theta_1, \theta_1^{1/n}\}, \tag{7.9}$$

where $\theta_1 = \|E\| \sum_{k=0}^{n-1} \|N\|^k$.

Finally, a result from semigroup theory (see [126]) states that if $\|e^{At}\| \leq ce^{\beta t}$ for some constants c and β, then

$$\|e^{(A+E)t}\| \leq ce^{(\beta + c\|E\|)t}. \tag{7.10}$$

7.2.1.3 Perturbation Analysis and Matrix Powers

In discrete-time systems, the matrix exponential is replaced by the matrix power. Thus, a bound on $(A+E)^p$ is required. We introduce the concept of *Fréchet derivative* for this purpose.

Definition 7.1 [108] Let $A, E \in \mathbb{C}^{n\times n}$. The Fréchet derivative of a matrix function f at A in the direction of E is a linear operator L_f that maps E to $L_f(A, E)$ such that

$$f(A + E) - F(A) - L_f(A, E) = \mathscr{O}(\|E\|^2),$$

for all $E \in \mathbb{C}^{n\times n}$. The Fréchet derivative may not exist, but if it does it is unique.

The following lemma characterize the Fréchet derivative of the function X^p.

Lemma 7.2 *[3] Let $A, E \in \mathbb{C}^{n\times n}$. If $L_{X^p}(A, E)$ denotes the Fréchet derivative of X^p at A in the direction of E, then*

$$L_{X^p}(A, E) = \sum_{j=0}^{p-1} A^{p-1-j} E A^j.$$

This means that the p power of $A + E$ is

$$(A + E)^p = A^p + \sum_{j=0}^{p-1} A^{p-1-j} E A^j + \mathscr{O}(\|E\|^2).$$

Then, it is a logical consequence the following

$$\|(A + E)^p\| \leq \|A^p\| + \|\sum_{j=0}^{p-1} A^{p-1-j} E A^j\| + \mathscr{O}(\|E\|^2). \tag{7.11}$$

7.2.2 Problem Statement

Consider a large-scale system that have been decomposed into N_a linear time-invariant subsystems. The dynamics of each subsystem is given by

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + \sum_{j\in\mathscr{N}_i} H_{ij} x_j(t), \quad \forall i = 1, \ldots, N_a \tag{7.12}$$

where the set of "neighbors" of the subsystem i $\mathscr{N}_i$ is the set of subsystems that directly drive agent i's dynamics, and H_{ij} is the interaction term between agent i and agent j, and $H_{ij} \neq H_{ji}$ might hold. The state x_i of the ith agent has dimension n_i, u_i is the m_i-dimensional local control signal of agent i, and A_i, B_i, and H_{ij} are matrices of appropriate dimensions.

In each node or subsystem, we can distinguish the dynamical part strictly speaking and a microprocessor in charge of monitoring the plant state and computing the control signal and the communication tasks (see Fig. 7.1).

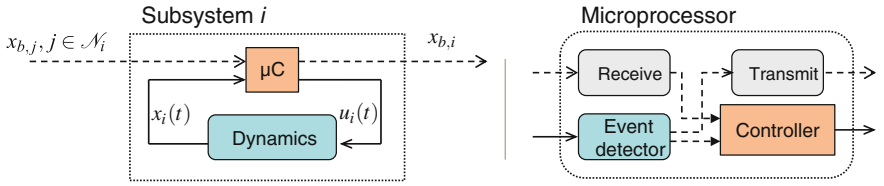

Fig. 7.1 Scheme of a node, consisting of a digital microcontroller (μC) and dynamics (*left*), and block diagram of the tasks carried out by the microprocessor

Due to the limited bandwidth, the communication between subsystems is at discrete instants of time. The dynamical coupling between subsystems makes it interesting to have access to the state of neighboring agents to include this information into the control law. Specifically, the agent i communicate with the set of agents in its neighborhood $\mathcal{N}_i$. The transmission occurs when an event is triggered. We denote by $\{t_k^i\}_{k=0}^{\infty}$ the times at which an event is detected in the agent i, where $t_k^i < t_{k+1}^i$ for all k.

The broadcast state is denoted by $x_{b,i}$. The broadcast states are used in the control law. Hence, the control signal is updated in a node, at least, when a new measurement is transmitted and/or received. In particular, the control law for each subsystem is

$$u_i(t) = K_i x_{b,i}(t) + \sum_{j \in \mathcal{N}_i} L_{ij} x_{b,j}(t), \quad \forall i = 1, \ldots, N_a \tag{7.13}$$

where K_i is the feedback gain for the nominal subsystem i. We assume that $A_i + B_i K_i$ is Hurwitz. L_{ij} is a set of decoupling gains.

Let us define the error $\varepsilon_i(t)$ between the state and the latest broadcast state as

$$\varepsilon_i(t) = x_{b,i}(t) - x_i(t) = x_i(t_k^i) - x_i(t), \quad t \in [t_k^i, t_{k+1}^i). \tag{7.14}$$

Rewriting (7.12) in terms of $\varepsilon_i(t)$ and the control law (7.13), we obtain

$$\dot{x}_i(t) = A_{K,i} x_i(t) + B_i K_i \varepsilon_i(t) + \sum_{j \in \mathcal{N}_i} \left(\Delta_{ij} x_j(t) + B_i L_{ij} \varepsilon_j(t) \right), \tag{7.15}$$

where $A_{K,i} = A_i + B_i K_i$, and $\Delta_{ij} = B_i L_{ij} + H_{ij}$ are the coupling terms. In general, $\Delta_{ij} \neq 0$ since the interconnections between the subsystems may be not well known, there might be model uncertainties or the matrix B_i does not have full rank.

We also define

$$A_K = \text{diag}(A_{K,1}, A_{K,2}, \ldots, A_{K,N_a}) \tag{7.16}$$

$$B = \text{diag}(B_1, B_2, \ldots, B_{N_a}) \tag{7.17}$$

$$K = \begin{pmatrix} K_1 & L_{12} & \cdots & L_{1N_a} \\ L_{21} & K_2 & \cdots & L_{2N_a} \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & K_{N_a} \end{pmatrix} \tag{7.18}$$

$$\Delta = \begin{pmatrix} 0 & \Delta_{12} & \cdots & \Delta_{1N_a} \\ \Delta_{21} & 0 & \cdots & \Delta_{2N_a} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{N_a1} & \Delta_{N_a2} & \cdots & 0 \end{pmatrix} \tag{7.19}$$

and the stack vectors

$$x = (x_1^T, x_2^T, \ldots, x_{N_a}^T)^T \tag{7.20}$$

$$\varepsilon = (\varepsilon_1^T, \varepsilon_2^T, \ldots, \varepsilon_{N_a}^T)^T \tag{7.21}$$

as the state and error vectors of the overall system. Note that $H_{ij}, L_{ij}, \Delta_{ij} := 0$ if $j \notin \mathcal{N}_i$. Let also be $n = \sum_{i=1}^{N} n_i$ the state and error dimension.

The dynamics of the overall system is given by

$$\dot{x}(t) = (A_K + \Delta)x(t) + BK\varepsilon(t). \tag{7.22}$$

As the broadcast states $x_{b,i}$ remain constant between consecutive events, the error dynamics in each interval is given by

$$\dot{\varepsilon}(t) = -(A_K + \Delta)x(t) - BK\varepsilon(t). \tag{7.23}$$

The above definition allows to study the stability of the overall system. These equations are valid as long as the following three time instances are simultaneous: the detection of the event, the transmission of the state $x_{b,i}$ from one node, and the reception in all neighboring nodes. When delays and packet dropouts can occur in the transmission, (7.22) and (7.23) do not generally hold. The extension to non-reliable communications is given in Chap. 10.

7.3 Event-Based Control Strategy

The design of distributed trigger functions $F_{e,i}$ to detect the occurrence of an event must satisfy the following properties:

- Guarantee the stability of the subsystem, and hence, of the overall system.
- Depend on local information of agent i only, or at most, of the neighbors, and take values in $\mathbb{R}$.

- Determine the sequence of local broadcasting times t_k^i recursively by the event-trigger function as $t_{k+1}^i = \inf\{t : t > t_k^i, F_{e,i}(t) > 0\}$.
- Ensure a lower bound for the inter-event times $T_{k,i} = t_{k+1}^i - t_k^i$.

In Chap. 1, the existing strategies for event-based control have been presented. Some of these approaches can be extended easily to distributed implementations. For instance, trigger functions for deadband control are

$$F_{e,i}(t) = \|\varepsilon_i(t)\| - \delta_i, \quad \delta_i > 0. \tag{7.24}$$

The design can be simplified by setting $\delta_i = \delta, \forall i = 1, \dots N_a$. Large values of δ allow reducing the number of events but degrades the performance. On the contrary, small values of δ give better performance but the average inter-event time decreases considerably. Moreover, this approach fails to ensure the asymptotic stability of the system, as in the case of centralized schemes.

Lyapunov-based sampling approaches to distributed event-triggering have also been studied. In this case, an event is enforced whenever

$$F_{e,i}(t) = \|\varepsilon_i(t)\| - \sigma_i \|x_i(t)\|, \quad 0 < \sigma_i < 1 \tag{7.25}$$

crosses from negative to positive. The set of parameters σ_i is determined by imposing that the Lyapunov function $V = \sum_{i=1}^{N_a} V_i(x_i)$ is locally positive definite and the time derivative of the Lyapunov-candidate-function is locally negative definite. For linear systems, the problem can be solved by solving a local LMI in each subsystem. See [259] for details. The asymptotic convergence to the equilibrium is guaranteed but a positive lower bound for the inter-event time may not be guaranteed when approaching the desired equilibria [79, 259].

In this chapter, the properties of trigger functions of the form

$$F_{e,i}(t) = \|\varepsilon_i(t)\| - \delta_{0,i} - \delta_{1,i} e^{-\beta_i t}, \quad \beta_i > 0 \tag{7.26}$$

are studied, where $\delta_{0,i}$ and $\delta_{1,i}$ cannot be zero simultaneously. To simplify the selection of parameters, we will consider that $\delta_{0,i} = \delta_0, \delta_{1,i} = \delta_1, \beta_i = \beta$, $\forall i = 1, \dots, N_a$.

Example 7.1 A trigger function (7.24) is depicted on Fig. 7.2a. The error is bounded by the constant threshold δ_0. Note that the error is reset after the occurrence of an event and that the inter-event time is always positive, since the error cannot reach the threshold again at the same time instance.
Trigger functions of the form (7.26) are represented on Fig. 7.2b. Note that the threshold decreases with time and the error is bounded by $\delta_0 + \delta_1$ at $t = 0$ and by δ_0 when $t \to \infty$. If $\delta_0 = 0$, this bound goes to zero when time increases and asymptotic stability can be achieved. Finally, Fig. 7.2c shows the error bound when events are enforced with (7.25).

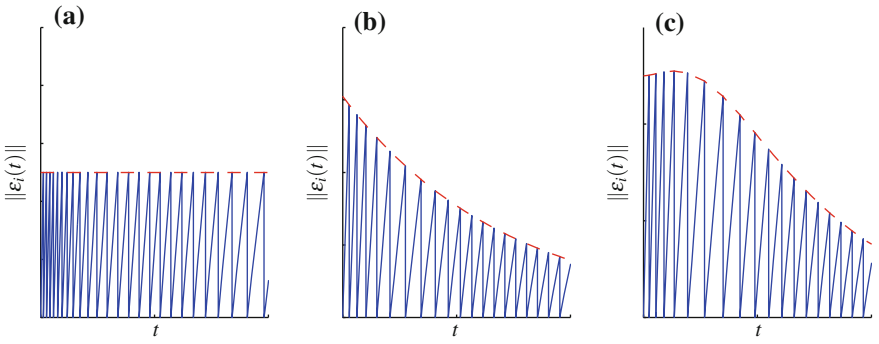

Fig. 7.2 Error function (*solid blue line*) and error bound (*dashed red line*) for trigger functions **a** (7.24), **b** (7.25), and **c** (7.26)

7.4 Performance Analysis

In this section, the stability properties of the system (7.12) are analyzed by using some of the results presented in Sect. 7.2.1. First, we briefly discuss the concepts of perfect and non-perfect decoupling that have some impact over the analytical treatment of the problem. After that the results are compared with other triggering mechanisms, and finally, this is also illustrated with a simulation example.

7.4.1 Perfect and Non-perfect Decoupling

If the decoupling gains L_{ij} can be chosen such that the *matching condition* holds, i.e., $\Delta_{ij} + B_i L_{ij} = 0$, (7.15) is transformed into

$$\dot{x}_i(t) = A_{K,i} x_i(t) + B_i K_i \varepsilon_i(t) + \sum_{j \in \mathcal{N}_i} B_i L_{ij} \varepsilon_j(t). \tag{7.27}$$

Hence, this essentially assures the perfect decoupling of the subsystems and allows to analyze their performance independently, since it holds that

$$x_i(t) = e^{A_{K,i}t} x_i(0) + \int_0^t e^{A_{K,i}(t-s)} \left(B_i K_i \varepsilon_i(s) + \sum_{j \in \mathcal{N}_i} B_i L_{ij} \varepsilon_j(s) \right) ds.$$

Then, if the error functions $\varepsilon_i(s)$, $\varepsilon_j(s)$ are bounded according to the trigger function (7.26), which are independent of the state, the convergence to the equilibrium only depends on local properties, that is, on the eigenvalues of $A_{K,i}$. Because the feedback gains K_i are designed so that $A_{K,i}$ is Hurwitz, the stability of each subsystem, and as a consequence, of the overall system, is guaranteed.

However, the perfect decoupling is a quite restrictive condition, and in many situations cannot be achieved because the interconnections between the subsystems may be not well known, there might be model uncertainties or the matrix B_i does not have full rank. Therefore, in the following, we assume that, in general, the interconnection terms $\Delta_{ij} \neq 0$.

In (7.22) Δ can be seen as a perturbation to A_K which influences the stability of the overall system. We obviously need to impose some constraints to Δ. Before doing this, the next assumption will facilitate the calculations in the following, but the extension to defective matrices is achievable as discussed later in the section.

Assumption 7.1 We assume that $A_{K,i}$, $i = 1, \ldots, N$ is diagonalizable so that there exists a matrix $D_i = \text{diag}(\lambda_k(A_{K,i}))$ and an invertible matrix of eigenvectors V_i such that $A_{K,i} = V_i D_i V_i^{-1}$.
The next lemma provides a bound for $\|\Delta\|$ that ensures that $A_K + \Delta$ is Hurwitz.

Lemma 7.3 *If $\kappa(V)\|\Delta\| < |\alpha_{max}(A_K)|$ holds, the eigenvalues $\tilde{\lambda}_i$ of $A_K + \Delta$ have negative real part.*

Proof According to the Bauer–Fike theorem (see (7.8) on p. 154), it follows that

$$\min_{\lambda_j \in \lambda(A_K)} |\tilde{\lambda}_i - \lambda_j| \leq \kappa(V)\|\Delta\|.$$

Assume that $\tilde{\lambda}_i = \tilde{\alpha}_i + i\tilde{\beta}_i$ and $\lambda_j = \alpha_j + i\beta_j$. Then, it holds that

$$|\tilde{\lambda}_i - \lambda_j| = \sqrt{(\tilde{\alpha}_i - \alpha_j)^2 + (\tilde{\beta}_i - \beta_j)^2} > |\tilde{\alpha}_i - \alpha_j|.$$

Because A_K is Hurwitz, $\alpha_j < 0, \forall j$, and according to the definition of $\alpha_{max}(A_K)$ (7.2), then it yields $|\alpha_{max}(A_K)| \leq |\alpha_j|, \forall j$. Moreover, if $\kappa(V)\|\Delta\| < |\alpha_{max}(A_K)|$, $\kappa(V)\|\Delta\|$ is also upper bounded by $|\alpha_j|, \forall j$. Thus, $\tilde{\alpha}_i$ is negative, because if it was positive

$$|\tilde{\alpha}_i - \alpha_j| = \tilde{\alpha}_i + |\alpha_j| > |\alpha_j| \geq |\alpha_{max}(A_K)| > \kappa(V)\|\Delta\|,$$

that would contradict the theorem of Bauer–Fike. Hence, $\tilde{\alpha}_i$ is negative, and this concludes the proof.

The previous result imposes a constraint over $\|\Delta\|$ to guarantee stability, and hence, an additional assumption is required.

Assumption 7.2 The coupling terms Δ_{ij} are such that $\kappa(V)\|\Delta\| < |\alpha_{max}(A_K)|$ holds.

The following theorem states that if Assumptions 7.1 and 7.2 hold, the system (7.22) with trigger functions defined as in (7.26) converges to a specified region around the equilibrium point which, without loss of generality, is assumed to be $(0, \ldots, 0)^T$. Moreover, if $\delta_0 = 0$ the convergence is asymptotical to the origin. The functions (7.26) bound the errors $\|\varepsilon_i(t)\| \leq \delta_0 + \delta_1 e^{-\beta t}$, since an event is triggered as soon as

the norm of $\varepsilon_i(t)$ crosses the threshold $\delta_0 + \delta_1 e^{-\beta t}$. The proof can be found in Appendix A.

Theorem 7.1 *Consider the closed-loop system (7.22) and trigger functions of the form (7.26), with $0 < \beta < |\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|$. Then, if Assumptions 7.1 and 7.2 hold, for all initial conditions $x(0) \in \mathbb{R}^n$, and $t > 0$, the state of the overall system is upper bounded as follows:*

$$\begin{aligned}\|x(t)\| \leq & \kappa(V)\Bigg(\frac{\|BK\|\sqrt{N_a}\delta_0}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|} + e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\Bigg(\|x(0)\|- \\ & \|BK\|\sqrt{N_a}\left(\frac{\delta_0}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|} + \frac{\delta_1}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|-\beta}\right)\Bigg) \\ & + e^{-\beta t}\frac{\|BK\|\sqrt{N_a}\delta_1}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|-\beta}\Bigg).\end{aligned} \tag{7.28}$$

Furthermore, the inter-event times are lower bounded by

$$T_{min} = \frac{\delta_0}{k_1 + k_2 + k_3}, \tag{7.29}$$

where

$$k_1 = \kappa(V)\|A_K + \Delta\|\|x(0)\| \tag{7.30}$$

$$k_2 = \|BK\|\sqrt{N_a}\delta_1\left(\frac{\kappa(V)\|A_K + \Delta\|}{|\alpha_{max}(A_K)| - \kappa(V)\|\Delta\| - \beta} + 1\right) \tag{7.31}$$

$$k_3 = \|BK\|\sqrt{N_a}\delta_0\left(\frac{\kappa(V)\|A_K + \Delta\|}{|\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|} + 1\right). \tag{7.32}$$

Remark 7.1 The results of Theorem 7.1 can be particularized to the perfect decoupling case. The state is upper bounded by

$$\begin{aligned}\|x(t)\| \leq & \kappa(V)\Bigg(\frac{\|BK\|\sqrt{N_a}\delta_0}{|\alpha_{max}(A_K)|} + e^{-|\alpha_{max}(A_K)|t}\Bigg(\|x(0)\|- \\ & \|BK\|\sqrt{N_a}\left(\frac{\delta_0}{|\alpha_{max}(A_K)|} + \frac{\delta_1}{|\alpha_{max}(A_K)|-\beta}\right)\Bigg) \\ & + e^{-\beta t}\frac{\|BK\|\sqrt{N_a}\delta_1}{|\alpha_{max}(A_K)|-\beta}\Bigg),\end{aligned}$$

and the minimum inter-event times lower bounded by

$$\frac{\delta_0}{\kappa(V)\|A_K\|\|x(0)\| + \|BK\|\sqrt{N_a}\left(\delta_1\left(\frac{\kappa(V)\|A_K\|}{|\alpha_{max}(A_K)|-\beta} + 1\right) + \delta_0\left(\frac{\kappa(V)\|A_K\|}{|\alpha_{max}(A_K)|} + 1\right)\right)}.$$

Thus, when the matching condition holds, the rate of convergence to the equilibrium is faster and the minimum inter-event times larger.

Remark 7.2 If Assumption 7.1 does not hold, the results can be extended noting that $\|e^{A_K t}\|$ can be bounded by either using the Jordan Canonical form, and hence (7.5) holds, or the Schur decomposition bound (7.7). In both cases the bound is governed by the exponential of $\alpha_{max}(A_K)$, which is negative. Thus, the stability of the system is guaranteed though the speed of convergence to the equilibria decreases. Moreover, if A_K is defective, then the restraint over Δ that guarantees that the eigenvalues of $A_K + \Delta$ have negative real part can be obtained from (7.9), enforcing $\max\{\theta_1, \theta_1^{1/n}\} < |\alpha_{max}(A_K)|$.

7.4.2 Comparison with Other Triggering Mechanisms

The results derived previously can be compared to the most frequently used event-triggered control strategies. We also particularized the results for the case $\delta_0 = 0$, which is interesting since yields asymptotic stability.

7.4.2.1 Deadband Control

In deadband control, an event is triggered whenever the state crosses some levels defined by a constant. From the analytical point of view, this is equivalent to have trigger functions (7.26) with $\delta_1 = 0$ and the error bounded by $\|\varepsilon_i(t)\| \leq \delta_0$. Thus, from Theorem 7.1 bound for the state is

$$\|x(t)\| \leq \kappa(V)\left(\frac{\|BK\|\sqrt{N_a}\delta_0}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|} + e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\left(\|x(0)\| - \|BK\|\sqrt{N_a}\frac{\delta_0}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\right)\right),$$

and a lower bound for the inter-event time is

$$T_{min} = \frac{\delta_0}{k_1 + k_3}.$$

7.4.2.2 Pure Exponential Trigger Functions

A particular case of trigger functions (7.26) is when $\delta_0 = 0$. For this situation, the state is upper bounded as

$$\|x(t)\| \leq \kappa(V)\left(e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\left(\|x(0)\| - \frac{\|BK\|\sqrt{N_a}\delta_1}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|-\beta}\right) + e^{-\beta t}\frac{\|BK\|\sqrt{N_a}\delta_1}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|-\beta}\right).$$

Note that $\|x(t)\| \to 0$ when $t \to \infty$.

The expression that provides the solution of the minimum inter-event times is not derived directly from (7.29), and is given by

$$\left(\frac{k_1}{\delta_1}e^{(\beta-|\alpha_{max}(A_K)|)t^*}+\frac{k_2}{\delta_1}\right)T=e^{-\beta T}. \tag{7.33}$$

The right-hand side of (7.33) is always positive. Moreover, for $\beta < |\alpha_{max}(A_K)|$ the left-hand side is strictly positive as well, and the term in brackets is upper bounded by $\frac{k_2+k_1}{\delta_1}$ and lower bounded by k_2/δ_1, and this yields to a positive value of T for all $t^* \geq 0$. The proof can be found in Appendix A.

7.4.2.3 Lyapunov-Based Sampling

In [257], the problem presented in this chapter is addressed with trigger functions (7.25). The asymptotic stability of the system is guaranteed if there exists positive definite matrices P_i, Q_i such that

$$A_{K,i}^T P_i + P_i A_{K,i} \leq -Q_i$$
$$W_i = \sum_{j\in\mathcal{N}_i} ||P_j \Delta_{ji}||^2 \leq \frac{\lambda_{min}(Q_i)}{8(|\mathcal{N}_i|+1)}.$$

Moreover, the parameters are $\sigma_i = \sqrt{\alpha_i/\beta_i}$ and must hold

$$0 < \alpha_i < \lambda_{min}(Q_i) - (1+|\mathcal{N}_i|)\delta - \frac{2W_i}{\delta}$$
$$\beta_i = \frac{\|P_i B_i K_i\|^2}{\delta} + \sum_{j\in\mathcal{N}_i}\frac{2\|P_j B_j L_{ij}\|^2}{\delta}$$
$$\delta < \min_i\left\{\frac{\lambda_{min}(Q_i)}{2(1+|\mathcal{N}_i|)}\left(1+\sqrt{1-\frac{8(|\mathcal{N}_i|+1)W_i}{\lambda_{min}^2(Q_i)}}\right)\right\}.$$

Note that the number of constraints are larger and, hence, the design is more complicated.

As far as the inter-execution times, there is now positive lower bound independent of the state $x(t)$ in [257]. Thus, it is unclear what happens when the system approaches the origin. However, the existence of a positive lower bound is guaranteed in [239] at least for the centralized case and linear systems.

7.4.3 Simulation Example

7.4.3.1 System Description

In order to demonstrate the effectiveness of the event-based control strategy, let us consider the system consisting of a collection of N inverted pendulums of mass m and length l coupled by springs with rate k as in Fig. 7.3. This setup will be used throughout this and Chap. 10.

The problem of coupled oscillators has numerous applications in such fields as medicine, physics, or communications [53, 237], and the inverted pendulum is a well-known control engineering problem. The inverted pendulums are physically connected by springs and we desire to design control laws to reach the equilibrium as well as to decouple the system. The state of a pendulum i is broadcast to its neighbors in the chain at discrete times given by the communication strategy.

Each subsystem can be described as follows:

$$\dot{x}_i(t) = \begin{pmatrix} 0 & 1 \\ \frac{g}{l} - \frac{a_i k}{ml^2} & 0 \end{pmatrix} x_i(t) + \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix} u_i + \sum_{j \in \mathcal{N}_i} \begin{pmatrix} 0 & 0 \\ \frac{h_{ij} k}{ml^2} & 0 \end{pmatrix} x_j(t)$$

where $x_i(t) = \begin{pmatrix} x_{i_1}(t) & x_{i_2}(t) \end{pmatrix}^T$ is the state, a_i is the number of springs connected to the ith pendulum, and $h_{ij} = 1, \forall j \in \mathcal{N}_i$ and 0 otherwise.

State-feedback gains and decoupling gains are designed so that the system is perfectly decoupled, and each decoupled subsystem poles are at -1 and -2. This yields the following control law:

$$u_i(t) = \left(-3ml^2 \; a_i k - \frac{ml^2}{4} \left(8 + \frac{4g}{l} \right) \right) x_{b,i}(t) + \sum_{j \in \mathcal{N}_i} \begin{pmatrix} -k & 0 \end{pmatrix} x_{b,j}(t)$$

where $x_{b,i}(t) = \begin{pmatrix} x_{b,i_1}(t) & x_{b,i_2}(t) \end{pmatrix}^T$. In the following, the system parameters are set to $g = 10, m = 1, l = 2,$ and $k = 5$.

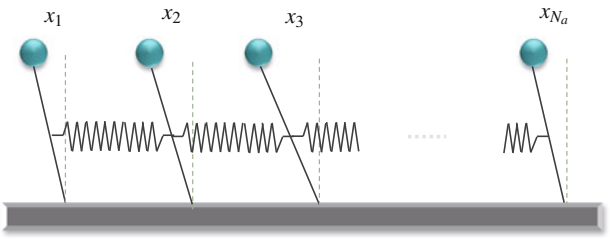

Fig. 7.3 Scheme of the network of the inverted pendulums

7.4.3.2 Performance and Comparison

The output of the system and the sequence of events for $N = 4$ and the same initial conditions than in the previous example when the trigger function is defined as in (7.26) with parameters $\delta_0 = 0.02$, $\delta_1 = 0.5$, and $\beta = 0.8$ are shown in Fig. 7.4.

The convergence of the system to a small region ($\delta_0 = 0.02$) around equilibrium is guaranteed due to the time dependency in the trigger functions. The event generation is shown in Fig. 7.4b. The system converges to zero with few events. Note that the agent that generates the highest number of events is Agent 2 (in red) and this value is 24 over a period of 15 s. Table 7.1 compares the proposed event-triggered approach to periodic control.

The bandwidth of the closed-loop subsystem is 0.8864 rad/s and the sampling period should be between (0.1772, 0.3544) s, according to [74], i.e., (42, 85) transmissions in a 15 s time, whereas the value for the minimum and maximum inter-event times are 0.1690 and 2.260, respectively. Furthermore, this comparison is even unfair with the event-based approach, since once the system is around the equilibrium point, the broadcasting periods take values around 1–2 s.

Observe also that the control signals are piecewise constant (Fig. 7.4c). They are updated if an event is triggered by the agent or its neighbors.

Table 7.2 extends this study for a larger number of agents. Several simulations were performed for different initial conditions for each value of N_a. Minimum and mean values of the inter-event times T_k^i were calculated for the set of the simulations with the same number of agents. We see that the broadcasting period remains almost constant when the number of agents increases. Thus, the amount of communication for the overall network grows linearly with N_a.

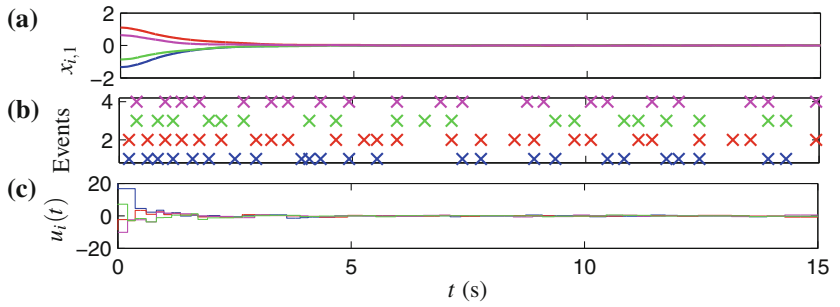

Fig. 7.4 Simulation results with trigger functions (7.26) with $\delta_0 = 0.02$, $\delta_1 = 0.5$, $\beta = 0.8$

Table 7.1 Comparison of time-triggered and event-triggered strategies

	No. updates	$\{T_k^i\}_{min}$ (s)	$\{T_k^i\}_{max}$ (s)
Time-triggered	(42, 85)	0.177	0.3544
Event-triggered	24	0.1690	2.260

Table 7.2 Inter-event times for different N

	N (s)	10	50	100	150	200
Trigger condition (7.26)	$\{T_k^i\}_{min}$	0.053	0.031	0.015	0.019	0.009
	$\{T_k^i\}_{\text{mean}}$	0.565	0.565	0.567	0.572	0.568
Trigger condition (7.24)	$\{T_k^i\}_{min}$	0.008	0.005	0.004	0.002	0.001
	$\{T_k^i\}_{\text{mean}}$	0.183	0.132	0.129	0.121	0.116
Trigger condition of [257]	$\{T_k^i\}_{\text{mean}}$	0.115	0.118	0.115	0.118	0.118

Moreover, these results are compared to other event-trigger functions: (7.24) with $\delta = 0.02$, and (7.25). For this later case, the results are taken from [257]. We see that trigger functions (7.26) can provide around five times larger broadcast periods. For example, for a number of pendulums of $N_a = 100$, trigger functions of the form (7.26) give a mean broadcasting period of 0.567, whereas trigger functions of the form (7.24) provide 0.129 and the result given in [257] is 0.115.

7.5 Extension to Discrete-Time Systems

7.5.1 System Description

The previous analysis considers that the state of the subsystems is monitored continuously. However, in practice, most of the hardware platforms only provide periodical implementations of the measurement and actuation tasks.

Hence, let us consider that each subsystem i is sampled at predefined instances of time given by a sampling period T_s. The discrete-time dynamical equation describing each subsystem is

$$x_i(\ell+1) = A_i x_i(\ell) + B_i u_i(\ell) + \sum_{j \in \mathcal{N}_i} H_{ij} x_j(\ell). \tag{7.34}$$

The control law is given by

$$u_i(\ell) = K_i x_{b,i}(\ell) + \sum_{j \in \mathcal{N}_i} L_{ij} x_{b,j}(\ell), \tag{7.35}$$

where $x_{b,i}(\ell)$ is the last-broadcast state, K_i is the feedback gain, and L_{ij} are the decoupling gains for the discrete-time subsystem i. The error is defined again as the difference between the last-broadcast state and the measured state. Thus,

$$\varepsilon_i(\ell) = x_{b,i}(\ell) - x_i(\ell), \tag{7.36}$$

and (7.34) can be rewritten in terms of the error $\varepsilon_i(\ell)$ as

$$x_i(\ell+1) = A_{K,i}x_i(\ell) + B_iK_i\varepsilon_i(\ell) + \sum_{j\in\mathcal{N}_i} \Delta_{ij}x_j(\ell) + B_iL_{ij}\varepsilon_j(\ell), \tag{7.37}$$

where $A_{K,i} = A_i + B_iK_i$ and $\Delta_{ij} = B_iL_{ij} + H_{ij}$. K_i are designed so that all the eigenvalues of $A_{K,i}$ lie inside the unit circle.

If we define the block matrices A_K, B, K, and Δ as in (7.16)–(7.19), and the stack vectors x and e as in (7.20) and (7.21), respectively, then the overall system dynamics is

$$x(\ell+1) = (A_K + \Delta)x(\ell) + BKe(\ell). \tag{7.38}$$

7.5.2 Discrete-Time Trigger Functions

Trigger functions of the form (7.26) are difficult to implement in digital platforms since they involve a decaying exponential. Therefore, for discrete-time systems, we propose the following functions

$$F_{e,i}(\varepsilon_i(\ell), \ell) = \|\varepsilon_i(\ell)\| - (\delta_0 + \delta_1\beta^{\ell}), \quad 0 < \beta < 1 \tag{7.39}$$

since they can be assimilated to (7.26) for discrete-time instances.

The instances of discrete time at which events are detected are denoted as ℓ_k^i and are defined recursively as follows:

$$\ell_{k+1}^i = \inf\{\ell > \ell_k^i, F_{e,i}(\varepsilon_i(\ell), \ell) \geq 0\}.$$

Example 7.2 Let us consider a trigger function $F_{e,i}(\varepsilon_i(t), t) = \|\varepsilon_i(t)\| - (0.01 + 0.5e^{-0.8t})$ in continuous time t, which bounds the error $\|\varepsilon_i(t)\| \leq (0.01 + 0.5e^{-0.8t})$. This bound is depicted in Fig. 7.5 (blue line). Assume that this system is sampled:

- With a sampling period $T_s = 0.1$.
- With a sampling period $T_s = 0.2$.

Trigger functions of the form (7.39) can be defined with the same values for δ_0 and δ_1 and with $\beta = e^{-0.8T_s}$. This yields values $\beta = 0.9231$ and $\beta = 0.8521$, respectively. The error bounds for both cases are shown in Fig. 7.5. Note that this bound is a piecewise constant function and changes at the sampling time instances.

7.5.3 Stability Analysis

Theorem (7.1) sums up the stability results for the continuous time system. Equivalent results can be derived for the discrete-time system (7.38).

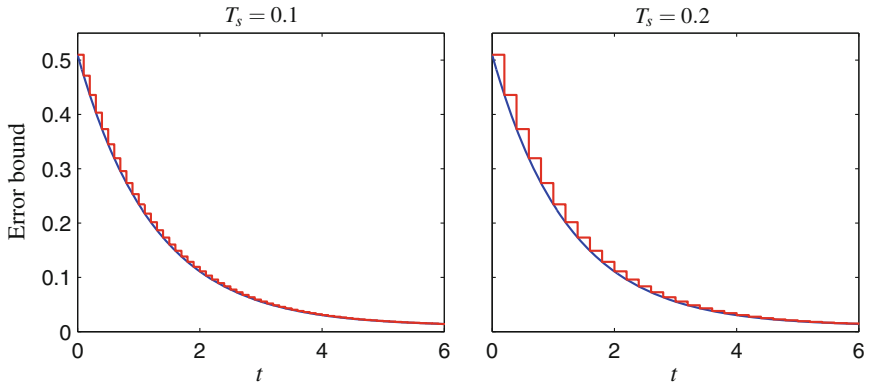

Fig. 7.5 Comparative of time-continuous (*blue*) and discrete-time (*red*) trigger functions, $T_s = 0.1$ (*left*), $T_s = 0.2$ (*right*)

However, a remark should be pointed out first. Whereas in continuous time the state is monitored continuously and this ensures that the error $\varepsilon_i(t)$ is strictly upper bounded by $\delta_0 + \delta_1 e^{-\beta t}$, in discrete-time systems it might occur that for a given ℓ, $\|\varepsilon_i(\ell)\| < \delta_0 + \delta_1 \beta^{\ell}$, but $\|\varepsilon_i(\ell+1)\| > \delta_0 + \delta_1 \beta^{\ell+1}$, so that the error reached the bound in the inter-sampling time.

In order to deal with this phenomenon, we state the following assumption.

Assumption 7.3 Fast sampling is assumed [109] so that events occur in all probability at the sampling times ℓ. Hence, $\|\varepsilon_i(\ell_k^i)\| \approx \delta_0 + \delta_1 \beta^{\ell_k^i}$ for some $\ell = \ell_k^i$.

The next theorem states that the system (7.38), when trigger functions (7.39) are used, converges to a region around the origin, which depends on δ_0.

The proof of the theorem can be found in Appendix A, being two the clues to follow the proof. First, all the eigenvalues of A_K lie inside the unit circle, so that $|\lambda_{max}(A_K)|^{\ell} < 1, \forall \ell \geq 0$ and $|\lambda_{max}(A_K)|^{\ell} \xrightarrow{\ell \to \infty} 0$, being $\lambda_{max}(A_K)$ the maximum of the eigenvalues of A_K. Second, the perturbation analysis for matrix powers, and in particular (7.11), can be applied. Before enouncing the theorem, the following assumption is required:

Assumption 7.4 A_K is diagonalizable so that $A_K = VDV^{-1}$, and the coupling terms are such that $\kappa(V)\|\Delta\| < 1 - |\lambda_{max}(A_K)|$, where $\kappa(V) = \|V\|\|V^{-1}\|$ and $\lambda_{max}(A_K)$ is the eigenvalue of A_K with the closer magnitude to 1. Furthermore, it is assumed that Δ is such that the second-order terms can be approximated to zero $\mathcal{O}(\|\Delta\|^2) \approx 0$.

Note that when $\beta \neq 0$, and additional constraint is imposed to the coupling terms. Specifically, the condition $|\lambda_{max}(A_K)| + \kappa(V)\|\Delta\| < \beta < 1$ ensures the convergence to the equilibria.

Theorem 7.2 *Consider the closed-loop system (7.38) and trigger functions of the form (7.39), where* $|\lambda_{max}(A_K)| + \kappa(V)\|\Delta\| < \beta < 1$. *If Assumptions 7.3 and 7.4 hold, then, for all initial conditions* $x(0) \in \mathbb{R}^n$ *and* $\ell > 0$, *it holds*

$$\begin{aligned}\|x(\ell)\| \leq \kappa(V)\Bigg(&\tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|}\gamma_0 + |\lambda_{max}(A_K)|^{\ell}\Big(\|x(0)\| - \tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|}\gamma_0 \\ &- \tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|}\gamma_1 + \tfrac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}\ell\left(\|x(0)\| - \tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|} - \tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|}\right)\Big) \\ &+ \beta^{\ell}\tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|}\gamma_1\Bigg),\end{aligned} \tag{7.40}$$

where

$$\gamma_0 = 1 + \frac{\kappa(V)\|\Delta\|}{1 - |\lambda_{max}(A_K)|} \tag{7.41}$$

$$\gamma_1 = 1 + \frac{\kappa(V)\|\Delta\|}{\beta - |\lambda_{max}(A_K)|}. \tag{7.42}$$

Remark 7.3 If perfect decoupling can be achieved, then $\|\Delta\| = 0$, which yields $\gamma_0, \gamma_1 = 1$. Thus, (7.40) is simplified:

$$\begin{aligned}\|x(\ell)\| \leq \kappa(V)\Bigg(&\tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|} + |\lambda_{max}(A_K)|^{\ell}\Big(\|x(0)\| - \tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|} \\ &- \tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|}\Big) + \beta^{\ell}\tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|}\Bigg).\end{aligned}$$

7.6 Improvements

The objective of this section is the proposal of some improvements to the design described previously in the chapter. First, a novel implementation is presented to reduce the number of control updates allowing a more efficient usage of the limited resources of embedded microprocessors. In the previous design, the adaption frequency of the control input may be high when the neighborhood is large even if each agent is not transmitting so often. The design is based on two sets of trigger functions. The first set decides when to transmit an update for the broadcast state and the second set checks a predefined control error at broadcasting events, updating only when this error exceeds a given threshold.

The second improvement of the discrete-event-based control (DEBC) has a different goal, which is to reduce as much as possible the communication through the network even if the load of the microprocessor is increased. We present a distributed model-based control design in which each agent has certain knowledge of the dynamics of its neighborhood. Based on this model, the subsystem estimates its state and its neighbors' continuously and computes the control law accordingly. Model uncertainty is assumed and the performance of the Sect. 7.4's and

model-based designs are compared, showing that a model-based controller allows larger inter-event times.

7.6.1 Reducing Actuation in Distributed Control Systems

This section presents a distributed control design where the goal is not only to reduce communication but also the number of control updates in each node. Note that in a single control loop the reduction of communication usually implies the reduction of actuator updates [68, 239], which does not necessary hold in distributed systems.

The control law is computed in (7.13) based on the broadcast states. Thus, $u(t)$ is a piecewise constant function. Accordingly, the control law of agent i is updated when an event is triggered by itself or any of its neighbors. This might lead to very frequent control updates if the number of neighbors was large. However, the change of the control signal $u_i(t)$ might be small due to, e.g., a weak coupling. In this situation, an update of the control signal is generally not needed.

We propose a new control law in which $u_i(t)$ is not updated at each broadcasting event, but when an additional condition is fulfilled. We consider two mechanisms driven by events. The first one is the transmission of information between nodes (*transmission events*), and the second one is the update of the control law (*control update events*). Note that the *transmission events* correspond to the considered events up to now. The description of both sets of trigger functions is given next.

7.6.1.1 Trigger Functions

Transmission Events

The occurrence of a transmission event is defined by trigger functions $F_{x,i}$ which only depend on local information of agent i and take values in $\mathbb{R}$.

The sequence of broadcasting times t_k^i are determined recursively by the event-trigger function as

$$t_{k+1}^i = \inf\{t : t > t_k^i, F_{x,i}(t) > 0\}.$$

We define the error between the current state x_i and the most recently broadcast state $x_{b,i}$ as

$$\varepsilon_{x,i}(t) = x_{b,i}(t) - x_i(t), \tag{7.43}$$

and we consider time-dependent trigger functions defined by

$$F_{x,i}(t, \varepsilon_{x,i}(t)) = \|\varepsilon_{x,i}(t)\| - \delta_{x,0} - \delta_{x,1} e^{-\beta t}, \tag{7.44}$$

with $\delta_{x,0} > 0$, $\delta_{x,1} \geq 0$, and $\alpha > 0$. An event is detected when $F_{x,i}(t, \varepsilon_{x,i}(t)) > 0$, and the error $\varepsilon_{x,i}$ is reset to zero. Note that the error remains bounded by

$$\|\varepsilon_{x,i}(t)\| \leq \delta_{x,0} + \delta_{x,1} e^{-\beta t}. \tag{7.45}$$

This type of trigger functions has been shown to decrease the number of events while maintaining a good performance of the system. The case $\delta_{x,0} = 0$ is excluded. The reason is discussed later. However, the case $\delta_{x,1} = 0$ is admitted leading to static trigger functions.

Control Update Events

Let us denote the time instants at which the control update of the agent i occurs as $\{t_\ell^i\}_{\ell=0}^{\infty}, \forall i = 1, \ldots, N_a$.

The control law is defined for the inter-event time period as

$$u_{b,i}(t) = K_i x_{b,i}(t_\ell^i) + \sum_{j \in N_i} L_{ij} x_{b,j}(t_\ell^i), t \in [t_\ell^i, t_{\ell+1}^i). \tag{7.46}$$

In order to determine the occurrence of an event, we define

$$\varepsilon_{u,i}(t) = u_{b,i}(t) - u_i(t), \tag{7.47}$$

where $u_i(t)$ is given by (7.13). The set of trigger functions is given by

$$F_{u,i}(\varepsilon_{u,i}(t)) = \|\varepsilon_{u,i}(t)\| - \delta_u, \quad \delta_u > 0. \tag{7.48}$$

The sequence of control updates is determined recursively. However, whereas the transmission events can occur at any time t because $x_i(t)$ is a continuous function, $u_i(t)$ in (7.13) is not continuous but piecewise constant and only changes its value at transmission events. This means that the events on the control update are a subsequence of the transmission events.

Denote $\bar{\mathcal{N}_i} = i \cup \mathcal{N}_i$ and $\{t_k^{\bar{\mathcal{N}_i}}\}$ the set $\{t_k^i\} \cup \{t_k^j\}$, $j \in \mathcal{N}_i$. Thus,

$$t_{\ell+1}^i = \inf\{t_k^{\bar{\mathcal{N}_i}} : t_k^{\bar{\mathcal{N}_i}} > t_\ell^i, F_{u,i}(t_k^{\bar{\mathcal{N}_i}}) > 0\}.$$

Hence, it holds that $\{t_\ell^i\} \subset \{t_k^{\bar{\mathcal{N}_i}}\}$.

Example 7.3 An example of the proposed design is given in Fig. 7.6. Assume that Agent 1 sends and receives information to/from its neighborhood through a network. At $t = t_k^2$ it receives a broadcast state $x_{b,2}$ from Agent 2. Agent 1 computes u_1 according to the new value received. For example, if Agent 2 is its unique neighbor, $u_1(t_k^2) = K_1 x_{b,1}(t_k^2) + L_{12} x_{b,2}(t_k^2) = K_1 x_{b,1}(t_{k-1}^1) + L_{12} x_{b,2}(t_k^2)$, where t_{k-1}^1 is assumed to be the last broadcasting event time for Agent 1. After computing u_1, Agent 1 checks whether the difference between this value and the current control signal applied exceeds the threshold δ_u. Since this threshold is not exceeded, it does not update $u_{b,1}$. At $t = t_k^1$, Agent 1 detects an event because εx, 1 reaches the threshold δ_x. $x_1(t_k^1)$ is broadcast through the network and u_1 is computed again. Given that $\|\varepsilon_{u,1}\| < \delta_u$, $u_{b,1}$ is not modified. Finally, a new event occurs at $t = t_{k+1}^1$ resulting in a broadcast and a control update since $\|\varepsilon_{u,1}\| \geq \delta_u$. Note that $u_{b,1}(t) = u_1(t)$.

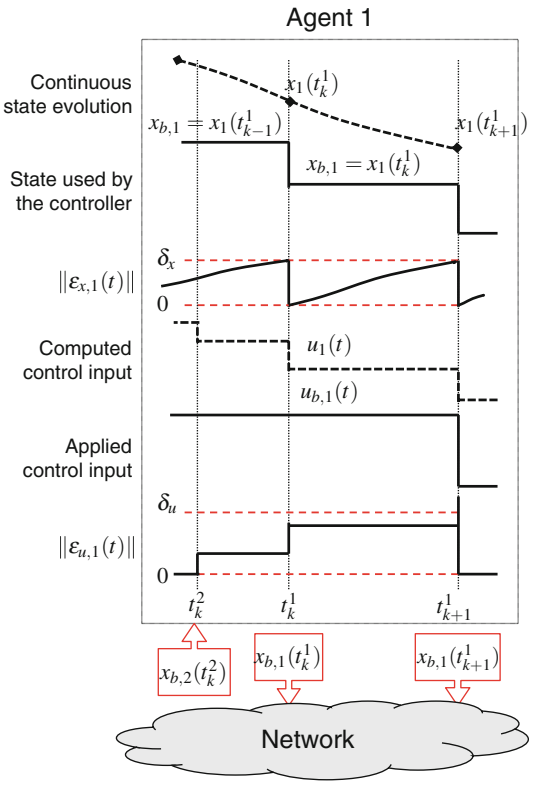

Fig. 7.6 Illustrative example of transmission and control update events between a system compound of two agents

7.6.1.2 Performance Analysis

The dynamics of the subsystems (7.12) with control law (7.46) is

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_{b,i}(t) + \sum_{j \in N_i} H_{ij} x_j(t).$$

It can be rewritten in terms of the errors $\varepsilon_{x,i}(t)$ and $\varepsilon_{u,i}(t)$ handled by the trigger functions (7.44) and (7.48). respectively, as

$$\dot{x}_i(t) = A_{K,i} x_i(t) + \sum_{j \in N_i} \Delta_{ij} x_j(t) + B_i K_i \varepsilon_{x,i}(t) + B_i \sum_{j \in N_i} L_{ij} \varepsilon_{x,j}(t) + B_i \varepsilon_{u,i}(t).$$

Let us define the stack vectors

$$\begin{aligned} \varepsilon_x^T &= \left(\varepsilon_{x,1}^T \dots \varepsilon_{x,N}^T\right) \\ \varepsilon_u^T &= \left(\varepsilon_{u,1}^T \dots \varepsilon_{u,N}^T\right), \end{aligned} \tag{7.49}$$

and consider the usual definitions for $x(t)$ and the matrices A_K, B, K, and Δ given in (7.16)–(7.19).

Accordingly, the overall system dynamics is given by

$$\dot{x}(t) = (A_K + \Delta)x(t) + BK\varepsilon_x(t) + B\varepsilon_u(t). \tag{7.50}$$

As the broadcast states $x_{b,i}$ remain constant between consecutive events, the dynamics of the state error in each interval are given by

$$\dot{\varepsilon}_x(t) = -(A_K + \Delta)x(t) - BK\varepsilon_x(t) - B\varepsilon_u(t). \tag{7.51}$$

The state error of the overall system is bounded by

$$\|\varepsilon_x(t)\| \leq \sqrt{N_a}(\delta_{x,0} + \delta_{x,1}e^{-\beta t})$$

according to (7.45). However, $\varepsilon_u(t)$ is not strictly bounded by δ_u because $u_i(t)$ is not a continuous function but piecewise constant. To find an analytical bound, we assume that that the occurrence of simultaneous transmission events in any neighborhood $\bar{\mathcal{N}_i}$ is not allowed, i.e., two neighboring nodes cannot transmit at the same instance of time. Moreover, in case that two broadcast states were received by one agent simultaneously, it could enqueue the data and do the computation of the control law sequentially. This might induce delays in the case where two nodes attempted to transmit at the same time. However, we assume that this delay is negligible in this section. The effect of delays and packet losses on event-triggered control of distributed control systems will be studied in Chap. 10.

Lemma 7.4 *The control error of the subsystem i is bounded by*

$$\|\varepsilon_{u,i}(t)\| \leq \bar{\delta}_{u,i}(t), \tag{7.52}$$

with

$$\bar{\delta}_{u,i}(t) = \delta_u + (\delta_{x,0} + \delta_{x,1}e^{-\beta t}) \cdot max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}.$$

Moreover, the control error of the overall system is bounded by

$$\|\varepsilon_u(t)\| \leq \sqrt{N_a}(\delta_u + \|\mu(K)\|_{max}(\delta_{x,0} + \delta_{x,1}e^{-\beta t})) = \bar{\delta}_u(t), \tag{7.53}$$

where

$$\mu(K) = \begin{pmatrix} \|K_1\| & \|L_{12}\| & \cdots & \|L_{1N}\| \\ \|L_{21}\| & \|K_2\| & \cdots & \|L_{2N}\| \\ \vdots & \vdots & \ddots & \vdots \\ \|L_{N1}\| & \|L_{N2}\| & \cdots & \|K_N\| \end{pmatrix}, \tag{7.54}$$

and $\|\cdot\|_{max}$ denotes the entry-wise max norm of a matrix.

Proof The proof can be found in Appendix A.

We next present the main result of the section.

Theorem 7.3 *Consider the interconnected linear system (7.50). If trigger functions (7.44) are used to broadcast the state with* $0 < \beta < |\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|$, *and trigger functions (7.48) for the control update, then, for all initial conditions* $x(0)$ *and* $t \geq 0$, *it follows that*

$$\|x(t)\| \leq \sigma_1 + (\kappa(V)\|x(0)\| - \sigma_1 - \sigma_2)e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t} + \sigma_2 e^{-\beta t}, \quad (7.55)$$

where

$$\sigma_1 = \kappa(V)\sqrt{N_a}\frac{(\|BK\| + \|B\|\|\mu(K)\|_{max})\delta_{x,0} + \|B\|\delta_u}{|\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|} \quad (7.56)$$

$$\sigma_2 = \kappa(V)\sqrt{N_a}\frac{(\|BK\| + \|B\|\|\mu(K)\|_{max})\delta_{x,1}}{|\alpha_{max}(A_K)| - \kappa(V)\|\Delta\| - \beta}. \quad (7.57)$$

Furthermore, the system does not exhibit Zeno behavior, being the lower bound for the inter-execution times

$$T_{x,min} = \frac{\delta_{x,0}}{\gamma_1 + \sqrt{N_a}(\gamma_2 + \gamma_3 + \gamma_4)}, \quad (7.58)$$

where

$$\gamma_1 = \kappa(V)\|x(0)\|\|A_K + \Delta\|$$

$$\gamma_2 = (\|BK\| + \|B\|\|\mu(K)\|_{max})\delta_{x,0}\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\right)$$

$$\gamma_3 = (\|BK\| + \|B\|\|\mu(K)\|_{max})\delta_{x,1}\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|-\alpha}\right)$$

$$\gamma_4 = \|B\|\delta_u\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\right).$$

Proof The proof can be found in Appendix A.

The previous analysis is based on two sets of trigger functions to detect transmission and control updates events. One concern that can be raised is how the values of the parameters of these trigger functions can be selected or if there is any relationship between them.

Let us first assume the case $\delta_{x,1} = 0$ yielding to static trigger functions. It follows that $\|\varepsilon_{x,i}(t)\| \leq \delta_{x,0}$ and $\|\varepsilon_{u,i}(t)\| \leq \delta_u + \delta_{x,0} \cdot \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}\ \forall t \geq 0$, according to (7.45) and (7.52), respectively.

Assume that the last control update event occurred at $t = t^*$ and denote the number of transmission events between t^* and the next broadcast as n_e. A lower

bound for n_e can be derived following the ideas of Lemma 7.4:

$$\|\varepsilon_{u,i}(t) - \varepsilon_{u,i}(t^*)\| = \|\varepsilon_{u,i}(t)\| \leq \sum_{k=1}^{n_e} \delta_{x,0} \cdot \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}$$
$$= n_e \delta_{x,0} \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}$$

and the next control update event will not be triggered before

$$\|\varepsilon_{u,i}\| = \delta_u \leq \delta_u + \delta_{x,0} \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}.$$

Thus,

$$n_e^i \geq \frac{\delta_u}{\delta_{x,0} \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}}. \tag{7.59}$$

Equation (7.59) shows the trade-off between δ_u and $\delta_{x,0}$ and gives insights on how one of these parameters should be chosen according to the other one.

Moreover, (7.59) can be translated into a relationship between the inter-execution times of the control law (7.46), denoted $T_{u,min}^i$, and the minimum broadcasting period (7.58). It holds that

$$T_{u,min}^i \geq n_e^i T_{x,min} \geq \frac{\delta_u}{(\gamma_1 + \sqrt{N_a}(\gamma_2 + \gamma_4)) \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}}.$$

Note that $\gamma_3 = 0$ because we are analyzing the case $\delta_{x,1} = 0$. Let $T_{u,min}$ be $T_{u,min} = \min\{T_{u,min}^i\}$. It yields

$$T_{u,min} \geq \frac{\delta_u}{(\gamma_1 + \sqrt{N_a}(\gamma_2 + \gamma_4)) \|\mu(K)\|_{max}}.$$

Hence, $\delta_{x,0}$ and δ_u can be chosen to meet some constraints on $T_{x,min}$ and $T_{u,min}$.

In the design of Sect. 7.6.1.1 the case $\delta_{x,0} = 0$ was excluded and the reason is given next. Assume that $\delta_{x,0} = 0$. Thus, following the steps of the previous case, $\|\varepsilon_{u,i}(t)\| \leq n_e \delta_{x,1} e^{-\beta t^*} \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}$, where n_e is the number of broadcasting events and t^* the time of the last control update event. Moreover, the next event is not triggered before $\|\varepsilon_{u,i}\|$ reaches the threshold δ_u. In this case, it holds that

$$n_e \geq \frac{\delta_u}{\delta_{x,1} e^{-\beta t^*} \max\{\|K_i\|, \|L_{ij}\| : j \in \mathcal{N}_i\}}. \tag{7.60}$$

Note that the lower bound for n_e in (7.60) goes to infinity when $t^* \to \infty$, which means that when the time values are large, many transmission events are required to trigger a new control update and may lead to small inter-event times. One possible solution is to accommodate the threshold δ_u to the decreasing bound on the state $\delta_{x,1} e^{-\beta t}$.

7.6.1.3 Simulation Example

Let us consider the system presented in Sect. 7.4.3 but with a different topology. Specifically, the mesh of inverted pendulums is depicted in Fig. 7.7. The dynamics of the subsystem change in this scheme, and three types of agents can be distinguished: the ones in the corners with two neighbors, the ones in the borders (excluding the corners) with three neighbors, and the inner pendulums with four nodes to communicate with. Moreover, movement is assumed to be in the XY plane. Hence, the dimension of the state is $n = 4$ and there are two control inputs ($m = 2$), which are the forces acting in the X and Y directions, respectively.

Figure 7.8 shows the output of the system in a 3D space for a mesh of 6×6 pendulums. The coordinates in the XY plane over time are plotted. Trigger functions with $\delta_{x,0} = 0.02$, $\delta_{x,1} = 0.5$, $\beta = 0.6$, and $\delta_u = 0.1$ are considered.

Let us focus on one particular subsystem, for example the agent (2,2) (second row, second column). The number of broadcasting events in all the neighborhood of this particular agent, which has four neighbors, is 170, while the number of control updates in the agent (2,2) is 90, so that 47 % of the transmissions do not end into a control update because the threshold δ_u is not reached.

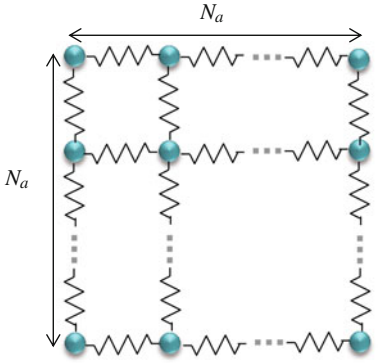

Fig. 7.7 Scheme of the coupled pendulums mesh

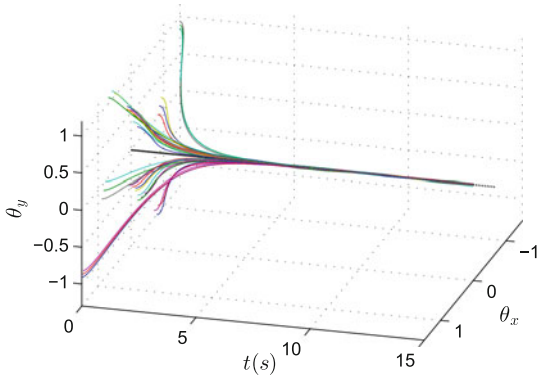

Fig. 7.8 $x_{i_1}(\theta_x)$ and $x_{i_3}(\theta_y)$ for a 6×6 mesh of inverted pendulums

If this experiment is repeated for the case in which trigger functions (7.48) are not considered, the number of broadcasting events in the neighborhood of (2,2) is 140, which is equal to the number of control updates. Thus, the proposed design with trigger functions (7.48) as expected might cause an increase of network transmissions, in this case 21 % while saving almost half of the changes on the control signal. Moreover, if we compute the average broadcasting period for the entire network as $\bar{T}_x = \frac{N_a^2 t_{sim}}{\text{No. events}}$ it yields 0.5202 s for the first case and 0.5954 s for the case without using the event-triggered control update. Hence, for the overall network the difference is not relevant. These results are extended for different values of N_a in Table 7.3. Note that the variations of the average period with the number of agents are not significant.

The influence of the parameter δ_u for given parameters $\delta_{x,0} = 0.02$, $\delta_{x,1} = 0.5$, and $\beta = 0.6$ can be analyzed and the results are illustrated in Table 7.4. For a mesh of 6×6 subsystems, the following values are computed for each value of δ_u and simulation time $t = 15$ s:

- Average number of transmissions through the network defined as $\bar{n}_x = \frac{\sum_{i=1}^{N_a^2} |\{t_k^i\}|}{N^2} |\overline{\mathcal{N}_i}|$, where $|\{t_k^i\}|$ is the cardinality of the set $\{t_k^i\}$ and $|\overline{\mathcal{N}_i}|$ is the average for the number of neighboring agents.
- Average number of control updates defined as $\bar{n}_u = \frac{\sum_{i=1}^{N_a^2} |\{t_\ell^i\}|}{N_a^2}$.

Note that the best choice of the values of δ_u, $\delta_{x,0}$ and $\delta_{x,1}$ depends on the communication and actuation costs of the implementation, and the lower bounds on the inter-event times that should be guaranteed in the system. We can say that a value $\delta_u \in [0.05, 0.1]$ would be a good option because the decrease of the control events is notable while the increase in communication events is assumable. If $\delta_u = 0.02$ all broadcasting events lead into a control update ($\bar{n}_u$ is actually larger than $\bar{n}_x$, but this is due to the error induced by the statistical treatment of the data).

Table 7.3 Average broadcasting period variations with N_a

$N_a \times N_a$	16	36	64	81	100
$\bar{T}_x$	0.5422	0.5202	0.4813	0.4676	0.4765

Table 7.4 Average transmission and control update events with c_u

δ_u	0.02	0.05	0.1	0.2
$\bar{n}_x$	86.20	83.98	95.46	181.48
$\bar{n}_u$	93.11	75.00	67.28	57.58

7.6.2 Model-Based Design

Model-based event-triggered control has been shown to reduce the amount of communication in a control loop [154]. Ideally, if the plant is stable, there are no model uncertainties or external disturbances, the control input $u(t)$ can be determined in a feedforward manner, and no communication over the feedback link is necessary [139].

The distributed approach presented in this section shows that if the model uncertainty fulfills a certain condition, the model-based approach gives larger minimum inter-event times than the zero-order hold approach of Sect. 7.4. We assume that each agent has knowledge of the dynamics of its neighborhood.

In particular, let us define the model-based control law for each agent as

$$u_i(t) = K_i x_{m,i}(t) + \sum_{j \in \mathcal{N}_i} L_{ij} x_{m,j}(t), \tag{7.61}$$

where $x_{m,i}$ now represents the state estimation of x_i given by the model $(A_{m,i}, B_{m,i})$ of each agent, and $A_{mK,i} = A_{m,i} + B_{m,i}K_i$. Let us define $A_{mK} = \text{diag}(A_{mK,1}, \ldots, A_{mK,N_a})$.

The error $\varepsilon_i(t)$ is redefined as

$$\varepsilon_i(t) = x_{m,i}(t) - x_i(t), \tag{7.62}$$

and is reset at events' occurrence. In particular, $x_{m,i}(t)$ is computed in the inter-event times as

$$x_{m,i}(t) = e^{A_{mK,i}(t-t_k^i)} x_i(t_k^i), \quad \forall t \in [t_k^i, t_{k+1}^i). \tag{7.63}$$

Note that (7.63) does not include the coupling effect since the decoupling gains L_{ij} are designed to compensate the model of the interconnections H_{ij}. Thus, if $\Delta_{ij} \neq \mathbf{0}$ it is because these interconnections are partially unknown or perfect decoupling may not be possible due to, e.g., the matrix B_i not having full rank.

Therefore, each agent i has a model of its dynamics and of its neighborhood $\mathcal{N}_i$. Based on this model, it estimates its state denoted as $x_{m,i}(t)$ to compute $u_i(t)$ in (7.61). This idea is illustrated in Fig. 7.9. Note that this is an extension of a conventional model-based controller. In the distributed approach, the controller C has $\mathcal{N}_i + 1$ inputs and one output. A block that represents the model of a subsystem is reset when a new broadcast state is received.

When the state estimation $x_{m,i}(t)$ differs a given quantity from $x_i(t)$, which depends on the trigger function, a new event is generated and the estimation is reset to the new measured state. For instance, $x_{m,i}$ might deviate from x_i due to model uncertainties on $A_{K,i}$, disturbances, and the effect of the non-perfect decoupling. Furthermore, the agent i broadcasts the new measurement to its neighbors, which also update their estimations according to the new value received from agent i.

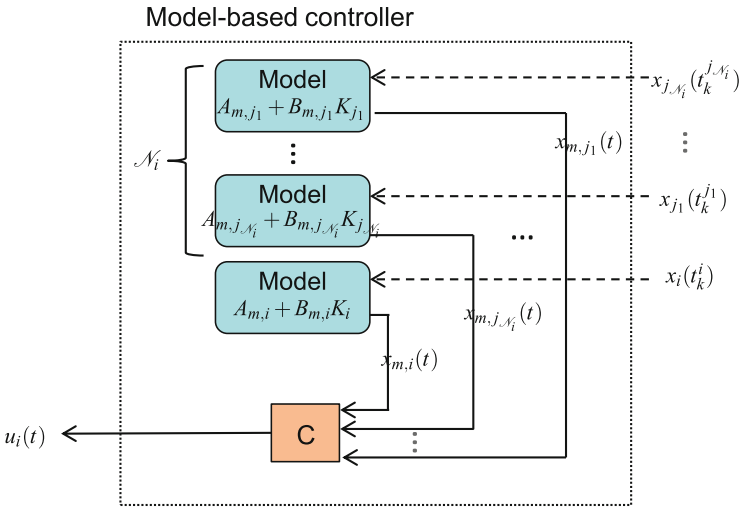

Fig. 7.9 Model-based control scheme for the node i

7.6.2.1 Main Result

If we consider the trigger function defined in (7.26) and for the new error defined in (7.62), the state will be also bounded by (7.28). However, the lower bound for the inter-event time will have a different expression.

Definition 7.2 Let us define

$$\begin{aligned} \delta A &:= A_m - A \\ \delta B &:= B_m - B \\ \delta A_K &:= A_{mK} - A_K = \delta A + \delta B K, \end{aligned} \tag{7.64}$$

i.e., the model uncertainty of the overall system without interconnections.

Assumption 7.5 We assume that the values of δ_0 and δ_1 and the initial conditions $x(0)$ satisfy the following constraint:

$$\frac{\sqrt{N_a}(\delta_0 + \delta_1)}{\|x(0)\| + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta} + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \beta}} < \kappa(V)\frac{\|A_K + \Delta\| - \|\delta A_K\| - \|\Delta\|}{\|A_{mK}\|}, \tag{7.65}$$

where $\alpha_\Delta = |\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|$.

Remark 7.4 Equation (7.65) is feasible only if the right-hand side is strictly positive, since $\delta_0 + \delta_1 > 0$. This gives a maximum value of the model uncertainty for a given bound on the norm of the coupling terms matrix or vice versa.

Theorem 7.4 *If Assumption 7.5 holds, the lower bound of the broadcasting period for the system (7.22), under the control law (7.61), and with triggering functions (7.26),* $0 < \beta < |\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|$*, is greater than (7.29).*

Proof The proof can be found in Appendix A.

7.6.2.2 Simulation Example

Next, the performance of the model-based approach is demonstrated and compared to the results of Sect. 7.4.3. Let us consider trigger functions $F_{e,i}(t, \varepsilon_i(t)) = 0.02 + 0.5e^{-0.8t}$. Figure 7.10 compares the output of Agent 1 of a chain of four inverted pendulums. Observe that, for this case, the model-based approach reduces the number of events in more than a third (from 23 (in red) to 9 (in blue)). Note that the control law is not a constant piecewise function.

Table 7.5 compares the results of the first row of Table 7.2 with the model-based design. Note that when the controller uses a model, the average and the minimum values of the inter-event times are enlarged, as predicted by Theorem 7.4.

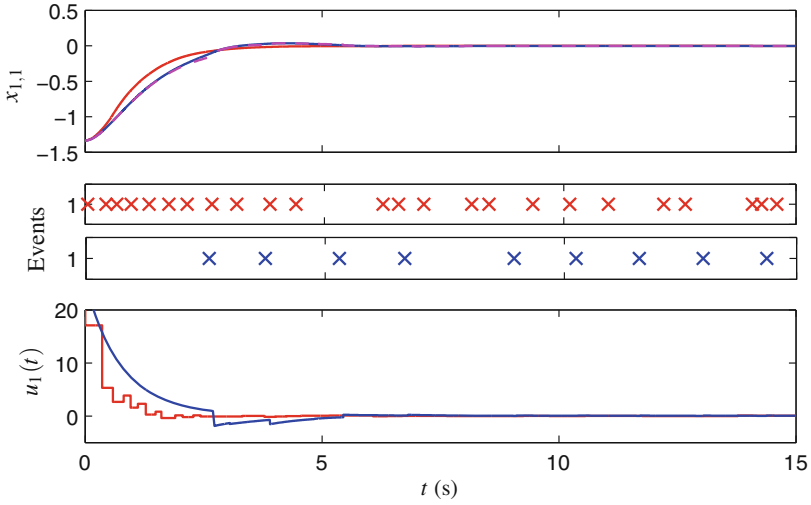

Fig. 7.10 Simulation result with trigger functions (7.26) for the design of Sect. 7.4 (*red*) and the distributed model-based control (*blue*). The *dashed line* (*magenta*) represents the piecewise function $x_{m1,1}$

Table 7.5 Inter-event times for different N_a

	N_a (s)	10	50	100	150	200
Trigger condition (7.26), Sect. 7.4	$\{T_k^i\}_{min}$	0.053	0.031	0.015	0.019	0.009
	$\{T_k^i\}_{\text{mean}}$	0.565	0.565	0.567	0.572	0.568
Trigger condition (7.26), MB control	$\{T_k^i\}_{min}$	0.6816	0.3025	0.219	0.0963	0.132
	$\{T_k^i\}_{\text{mean}}$	1.430	1.500	1.477	1.668	1.581

7.7 Conclusions

A distributed event-based control strategy for interconnected subsystems has been presented. The events are generated by the agents based on local information only, broadcasting their state over the network. The proposed trigger functions preserve the desired convergence properties and guarantee the existence of a strictly positive lower bound for the broadcast period, excluding the Zeno behavior.

Because most of the hardware platforms only provide periodical implementations of the measurement and actuation tasks, the analysis has been extended to discrete-time systems.

Additionally, the way in which the actuation rate can be reduced in an interconnected system if triggering functions are also used in the update of the control law has been illustrated. The existing trade-off between communication and actuation has been shown analytically and through simulations.

Finally, a model-based approach has been proposed showing that the minimum inter-event times can be enlarged if the model uncertainty satisfies certain conditions.

Chapter 8
Distributed Event-Based Observers for LTI Systems

Pablo Millán, Carlos Vivas and Carlo Fischione

8.1 Introduction

In the past few years, there has been a renewed interest in the problem of event-based distributed estimation. Most developments have been driven by applications in the field of mobile multi-agent systems [202], where several vehicles (agents) are intended to move in a coordinated way. In this field, the problem generally takes the form of a consensus problem and two different schemes of triggering events are usually found: centralized event-based consensus control and, decentralized event-based consensus control. The term *centralized* in this context refers to the case when there is only a single-event generator determining the global triggering instants for each agent. The triggering events are thus assumed to be synchronized [37, 41, 112, 152]. On the other hand, decentralized approaches require every agent to equip event generators in order to transmit their local information to neighborhoods in asynchronous instants [71, 232, 272, 278].

P. Millán (✉)
Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería,
Universidad Loyola Andalucía, Seville, Spain
e-mail: pmillan@uloyola.es

C. Vivas
Dpto. de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingenieros,
Universidad de Sevilla, Seville, Spain
e-mail: vivas@us.es

C. Fischione
Access Linnaeus Center, Department of Electrical Engineering,
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: carlofi@kth.se

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_8

Besides mobile multi-agent systems, probably the most common approach to event-based distributed estimation has been the distributed Kalman filter (DKF) based on consensus strategies. The methodology implies correcting the local estimations performed at each node based on the information received from their neighbors. See, for instance, [4, 157, 200, 203].

There exists also a number of works that propose different approaches. For example, distributed moving horizon schemes are employed in [72], where the solution requires each sensor to solve a quadratic optimization problem at every sampling time. A finite-horizon paradigm is proposed in [55] and [233] to design distributed observers who take into account quantization errors and successive packet dropouts. Another significant work from the same authors lying in this filed is that in [235], in which a stochastic sampling between nodes is considered. A very interesting research direction considers more general models of the plant, including nonlinearities, inner delays, or different Markov-chain-driven dynamical modes. See, for instance, [234], [148] or [149].

In this chapter, we focus our interest in designing distributed estimators for LTI systems under the assumption of perfect communication channels, that is, communication delays are neglected, and reliable protocols are employed such that no packet dropouts are considered. Every sensor in the network (agent) is assumed to have all or part of the following capabilities: measurement of a subset of the plant states, compute estimations, and communicate to neighboring nodes. Based on local Luenberger-like observers in combination with consensus strategies, the proposed method allows for a network of sensors to estimate the whole plant state under mild assumptions of local observability. The observer design problem is solved via linear matrix inequalities. It is shown that globally ultimately uniformly boundedness (GUUB) of the estimation errors into an arbitrary small ultimate bound region can be achieved when using the proposed event-based implementation.

As in Chap. 7, this chapter develops methodologies for distributed estimation for the case of asynchronous communications. As already mentioned, event-based methods are more efficient from the point of view of bandwidth use, as communications are invoked only when significant information requires to be transmitted. This approach becomes specially beneficial in the context of distributed estimation over networks, as the limitations imposed by the network render the frequency at which the system communicates.

This chapter is organized as follows. Section 8.2 describes and motivates the problem under consideration. Section 8.3 describes the general methodology and assumptions made for the problem at hand. First, the case for periodic communications is developed in Sect. 8.3.1, and later the asynchronous design is described in Sect. 8.3.2. Finally, an illustrative example is given in Sect. 8.4 and conclusions end the chapter.

8.2 Problem Statement

The system to observe is an autonomous linear time-invariant plant given by the following equations:

$$x(k+1) = Ax(k), \tag{8.1}$$

$$y_i(k) = C_i x(k), \quad \forall i \in \mathscr{V}, \tag{8.2}$$

where $x(k) \in \mathbb{R}^n$ is the state of the plant, $y_i(k) \in \mathbb{R}^{m_i}$ are the system outputs, and N_a is the number of the agents of the network.

We assume that $\forall i \in \mathscr{V}$, agent i directly accesses the output y_i, and that it can communicate with its neighborhood $\mathscr{N}_i$, see Fig. 8.1. Clearly, if the pair (A, C_i) is observable for some i, then node i is capable to reconstruct the full state x of the plant directly from y_i, without the necessity of communicate with any other node (*local observability* [200]). Here, we consider instead the case of *collective observability*, that is, the systems are observable only if we put together all the nodes, i.e., the pair (A, C) is observable, where C is a matrix stacking the output matrices C_i of all the agents [200].

Let us define $\bar{C}_i$ as a matrix stacking the matrix C_i and matrices C_j for all $j \in \mathscr{N}_i$. It is assumed that each pair $(A, \bar{C}_i)$ is observable. This is a necessary condition that impose some restrictions on the network topology and the information that is sent via each connection.

The local observer structure we use is given by the following equations:

$$\begin{aligned} \hat{x}_i(k+1) &= A\hat{x}_i(k) + M_i\left(\hat{y}_i(k) - y_i(k)\right) + \sum_{j \in \mathscr{N}_i} N_{ij}\left(\hat{x}_j(k) - \hat{x}_i(k)\right), \\ \hat{y}_i(k) &= C_i\hat{x}_i(k), \ \forall i = 1, 2, ..., N_a, \end{aligned} \tag{8.3}$$

where $\hat{x}_i(k) \in \mathbb{R}^n$ is the state estimate of the agent i. With this scheme, every agent tries to reconstruct the full state of the plant. Notice that each estimator is given by

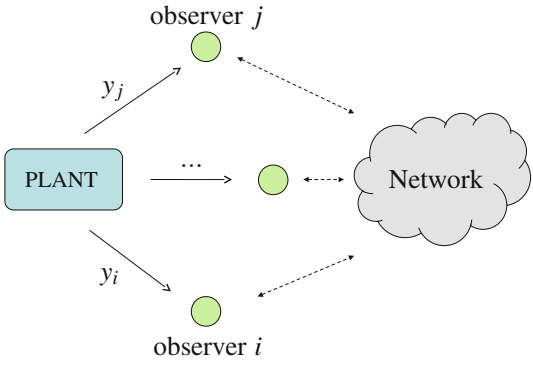

Fig. 8.1 System architecture. Each observer can measure some plant outputs and receives information from its neighbors

two pieces: a local Luenberger observer and a consensus part. The Luenberger-like observer corrects the estimated state of the plant based on the measured output $y_i(k)$ through the matrices M_i, while an additional correction is performed by consensus through the matrices N_{ij}, which take into account the information received from the neighborhood.

The problem we address in this chapter is split into two parts: first, we aim at designing the matrices set $\mathscr{M} = \{M_i, \quad i \in \mathscr{V}\}$ and $\mathscr{N} = \{N_{ij}, \quad (i,j) \in \mathscr{E}\}$ to stabilize the observations error of every node by assuming periodic communication. Then, given the set of matrices $\mathscr{M}$ and $\mathscr{N}$ designed above, the objective is to design an event-based mechanism to reduce the amount of communication among the nodes, while ensuring GUUB of the observation error.

8.3 Observer Design

In this section, we show how to solve the problem of distributed event-based estimation described so far.

Before proceeding further, let us define the observation error of the observer i as $e_i(k) = \hat{x}_i(k) - x(k)$, i.e., the difference between the estimation of agent i and the state of the plant, and let us define the vector $e(k)$ as the stack of the observation errors, i.e., $e^T(k) = [e_1^T(k) \dots e_{N_a}^T(k)]$.

We remark that for periodic communication among the nodes, asymptotic stability of the closed loop will be proved, although the network is affected by waste of energy and high traffic load because of unnecessary communication. For the event-based communication, both the energy expenditure and the network traffic load are drastically reduced, but asymptotically stability is no longer achieved. However, for this second case, GUUB into an arbitrary small region is still ensured.

8.3.1 Periodic Case

In the case of periodic communication, taking into account Eqs. (8.1)–(8.3), the dynamics of $e(k)$ is given by

$$e(k+1) = (\Phi(\mathscr{M}) + \Lambda(\mathscr{N}))\, e(k), \tag{8.4}$$

where matrices $\Phi(\mathscr{M})$ and $\Lambda(\mathscr{N})$ depend on the sets $\mathscr{M}$ and $\mathscr{N}$, and they have the following structure:

$$\Phi = \operatorname{diag}\{A + M_1 C_1, \dots, A + M_{N_a} C_{N_a}\}, \tag{8.5}$$

$$\Lambda(\mathscr{N}) = \sum_{(ij)\in\mathscr{E}} \Theta_{ij}(N_{ij}) \tag{8.6}$$

and

$$\Theta_{ij}(N_{ij}) = \begin{array}{c} \begin{matrix} column & i & & j & \end{matrix} \\ \begin{bmatrix} 0 \cdots & 0 & \cdots & 0 & \cdots 0 \\ \vdots \ \vdots & \vdots & \vdots & \vdots & \vdots \ \vdots \\ 0 \cdots & -N_{ij} & \cdots & N_{ij} & \cdots 0 \\ \vdots \ \vdots & \vdots & \vdots & \vdots & \vdots \ \vdots \\ 0 \cdots & 0 & \cdots & 0 & \cdots 0 \end{bmatrix} \end{array} row\ i \tag{8.7}$$

The design method resorts to a Lyapunov-based approach to prove that the procedure guarantees the asymptotic convergence of all the agents' estimates to the actual state of the plant. Concretely, we use the following Lyapunov function:

$$V(e) = e^T P e, \tag{8.8}$$

where $P = \text{diag}\{P_1, P_2, \ldots, P_{N_a}\}$, $P_i > 0$, $P_i \in \mathbb{R}^n$, $i \in \mathscr{V}$. The following result provides a design method in terms of a linear matrix inequality (LMI).

Lemma 8.1 *If there exists any positive scalar* $0 < \mu < 1$ *such that the LMI* (8.9) *has a feasible solution for positive definite matrix* P*, and matrices* W_i, X_{ij}, $i \in \mathscr{V}$ $(i, j) \in \mathscr{E}$,

$$\begin{bmatrix} -\mu P & * \\ \Phi(\mathscr{W}) + \Lambda(\mathscr{X}) & -P \end{bmatrix} < 0, \tag{8.9}$$

where

$$\Phi(\mathscr{W}) = P\Phi(\mathscr{M}), \mathscr{W} = \{W_i \triangleq P_i M_i, i \in \mathscr{V}\},$$
$$\Lambda(\mathscr{X}) = P\Lambda(\mathscr{N}), \mathscr{X} = \{X_{ij} \triangleq P_i N_{ij}, (i, j) \in \mathscr{E}\},$$

the estimations of all the observers asymptotically converge to the plant state by designing the observation matrices as $M_i = P_i^{-1} W_i$, $i \in \mathscr{V}$, *and* $N_{ij} = P_i^{-1} X_{ij}$, $i \in \mathscr{V}$, $(i, j) \in \mathscr{E}$.

Proof Choose the Lyapunov function (8.8). The forward difference can be computed as

$$\Delta V(k) = e^T(k+1) P e(k+1) - e^T(k) P e(k).$$

By taking into account Eq. (8.4) and Lemma 8.1, the forward difference can be expressed in the following way[1]:

[1] We remove the functional dependences to alleviate the notation.

$$\begin{aligned}\Delta V(k) &= e^T(\Phi+\Lambda)^T P(\Phi+\Lambda)e - e^T Pe \\ &\leq e^T(\Phi P+\Lambda P)^T P^{-1}(P\Phi+P\Lambda)e - e^T\mu Pe.\end{aligned}$$

Next, we impose that the Lyapunov function decreases for $e(k) \neq 0$. Using Schur complement, one can obtain easily that if the next matrix inequality holds, then $\Delta V(k) < 0$, which implies that the observation error of all the agents tends asymptotically to zero:

$$\begin{bmatrix} -\mu P & * \\ P\Phi(\mathscr{M}) + P\Lambda(\mathscr{N}) & -P \end{bmatrix} < 0.$$

This matrix inequality is not linear because of the products of variable matrices in the non-diagonal terms. However, from the diagonal structure of matrices P and $\Phi(\mathscr{M})$ and the special structure of matrix $\Lambda(\mathscr{N})$, given by (8.6) and (8.7), it is possible to define the change of variable in Lemma 8.1, thus obtaining the LMI (8.9), which concludes the proof.

Remark 8.1 By comparing the proposed method with the works [200], our method is not restricted to use a scalar gain on the consensus part, but it introduces matrix gains which can be different for each link.

8.3.2 Event-Based Implementation

In the previous section, we proposed a methodology to design the set of the observers' gains that solve the distributed estimation problem under the assumption of periodic communication. In this section, we exploit Lemma 8.1, developing an event-based strategy to reduce both the traffic load in the network and the energy consumption of the nodes due to unnecessary transmissions. In the event-based strategy we present here, each observer decides when it is necessary to broadcast its estimates to its neighbors, based on the difference between its current estimate and the last transmitted estimate. The price to be paid in this case is that asymptotic stability is no longer guaranteed, but GUUB of the estimation errors can be proved. The trade-off between performance and communication reduction can be adjusted with an adequate tuning of a free parameter that determines the size of the ultimate bound region and the amount of communication required.

The dynamics of each estimator, in the case of event-based implementation, is given by

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + M_i\left(\hat{y}_i(k) - y_i(k)\right) + \sum_{j\in\mathscr{N}_i} N_{ij}(\hat{x}_j(l_j) - \hat{x}_i(k)), \quad (8.10)$$

$$\hat{y}_i(k) = C_i\hat{x}_i(k), \ \forall i = 1, 2, ..., N_a, \quad (8.11)$$

where $l_j \leq k$ is the last time instant when the observer j communicated its estimated state to its neighborhood. Equation (8.10) takes in consideration aperiodic communication through the variable l_j, which can be distinct for each observer $i \in \mathscr{V}$. Equation (8.10) can be rewritten as follows:

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + M_i\left(\hat{y}_i(k) - y_i(k)\right) + \sum_{j\in\mathscr{N}_i} N_{ij}\left(\hat{x}_j(k) - \hat{x}_i(k)\right) + \sum_{j\in\mathscr{N}_i} \varepsilon_j(k), \tag{8.12}$$

where

$$\varepsilon_j(k) = N_{ij}\left(\hat{x}_j(l_j) - \hat{x}_j(k)\right). \tag{8.13}$$

Based on (8.12), the evolution of the observers with periodic communication is equivalent to the evolution with event-based communication, difference being in the terms $\varepsilon_j(k)$, which are given by Eq. (8.13). Every time the node j broadcasts its state to his neighborhood $\mathscr{N}_j$, we have $\varepsilon_j(k) = 0$. Therefore, $\varepsilon_j(k)$ can be interpreted as an external perturbation due to the discontinuous flow of information between neighbors, which is reset to zero at every transmission time. It is worth to point out that the disturbance that each observer j induces on its neighbors is unknown to other neighbors, but can be tracked by j, which has access to its own local estimations.

Since we are considering the designed observers' matrices to satisfy the conditions of Lemma 8.1, here the matrices describing the dynamic of the error are not variables and are fixed, that is, $\Phi(\mathscr{W}) = \Phi$, $\Lambda(\mathscr{X}) = \Lambda$. By proceeding analogously to the periodic case, the augmented observation error vector can be written as

$$e(k+1) = \Xi e(k) + \Gamma\varepsilon(k), \tag{8.14}$$

where

$$\Xi = \Phi + \Lambda, \quad \varepsilon(k)^T = [\varepsilon_1^T \ \ldots \ \varepsilon_p^T],$$

$$\Gamma(\mathscr{N}) = \sum_{(i,j)\in\mathscr{E}} \Psi_{ij}$$

and

$$\Psi_{i,j} = \begin{matrix} \\ \\ \\ \\ \end{matrix} \begin{matrix} \overset{column \qquad\qquad j}{} \\ \\ \\ \\ \end{matrix} \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & N_{ij} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \begin{matrix} \\ \\ row \quad i \\ \\ \end{matrix}$$

An event-based communication policy to ensure GUUB of the error dynamics (8.14) is given by the following important result.

Theorem 8.1 *Consider the observers 8.10–8.11 with gains as defined in Lemma 8.1. Suppose each node j broadcasts its estimate $\hat{x}_j$ to his neighbors $\mathcal{N}_j$ at each k such that $\|\varepsilon_j(k)\|_\infty \geq \delta$, with ε_j defined in (8.13). Then, the estimation error e is GUUB with bound*

$$\|e\|_2 \leq \delta \sqrt{\frac{n\lambda_{max}(P)}{\lambda_{min}(P)}}(\alpha\|\Xi\|_\infty + \|\Gamma\|_\infty),$$

$$\alpha = \frac{\|\Gamma^T P \Xi\|_\infty + \sqrt{\|\Gamma^T P \Xi\|_\infty^2 + (1-\mu)\lambda_{min}(P)\|\Gamma^T P \Gamma\|_\infty}}{(1-\mu)\lambda_{min}(P)}. \tag{8.15}$$

Proof From Theorem 8.1 and Eq. (8.14), the evolution of the Lyapunov function is given by[2]

$$\begin{aligned}\Delta V = {} & [e^T \Xi^T + \varepsilon^T \Gamma^T]P[\Xi e + \Gamma\varepsilon] - e^T P e \leq -e^T(1-\mu)Pe + 2\varepsilon^T \Gamma^T P \Xi e \\ & + \varepsilon^T \Gamma^T P \Gamma \varepsilon \leq -(1-\mu)\lambda_{min}(P)\|e\|_\infty^2 + 2\|\Gamma^T P \Xi\|_\infty \|\varepsilon\|_\infty \|e\|_\infty \\ & + \|\Gamma^T P \Gamma\|_\infty \|\varepsilon\|_\infty^2. \end{aligned} \tag{8.16}$$

The right-hand side of Eq. (8.16) is an algebraic second-order equation in $\|e\|_\infty$, such that it is easy to see that the Lyapunov function $V(k)$ decreases whenever $\|e(k)\|_\infty > \alpha\|\varepsilon(k)\|_\infty$, where α is given by Eq. (8.15). Given that each of the observers has access only to local information, and monitories the disturbance caused in its neighbors, if one takes the infinity norm of the disturbance vector ε, then it is possible to bound it bellow a desired level $\|\varepsilon\|_\infty < \delta$ using only local information in each node. In this way, it yields that $\Delta V(k) < 0$ in the region $\|e(k)\|_\infty > \alpha\delta$.

Now consider k^* as the time instant when the estimation error enters in the region $\|e(k)\|_\infty < \alpha\delta$. Then, taking into account the error dynamics given by (8.14), one can easily obtain that

$$\max\|e(k^*+1)\|_\infty = (\alpha\|\Xi\|_\infty + \|\Gamma\|_\infty)\delta$$

and the error can leave the region $\|e(k)\|_\infty \leq \alpha\delta$. After this, the Lyapunov function needs to decrease again, so the space enclosed by maximum of the Lyapunov function in k^*+1 is an ultimate bound for the estimation error. Using the inequalities $\lambda_{min}(P)\|e\|_2^2 \leq e^T P e \leq \lambda_{max}(P)\|e\|_2^2$ and $\|e\|_2 < \sqrt{n}\|e\|_\infty, \forall e \in \mathbb{R}^n$, one can easily compute the ultimate bound given in this theorem.

[2]We remove time indices to alleviate the notation.

Remark 8.2 As has been shown, the parameter δ is related with the size of the ultimate bound region of the estimation error e. By enlarging the value of δ, it is possible to reduce the amount of transmission of the nodes, while by reducing it, a better estimation performance is achieved, since the observation error is bounded in a smaller region.

8.4 Illustrative Example

In this section, we illustrate the proposed methodology by an example. Let $x := [x_1 \quad x_2 \quad x_3]^T \in \mathbb{R}^3$, and consider the system

$$x(k+1) = \begin{bmatrix} 0.95 & 0 & 0 \\ 0 & 0.809 & 1 \\ 0 & -0.3455 & 0.809 \end{bmatrix} x(k).$$

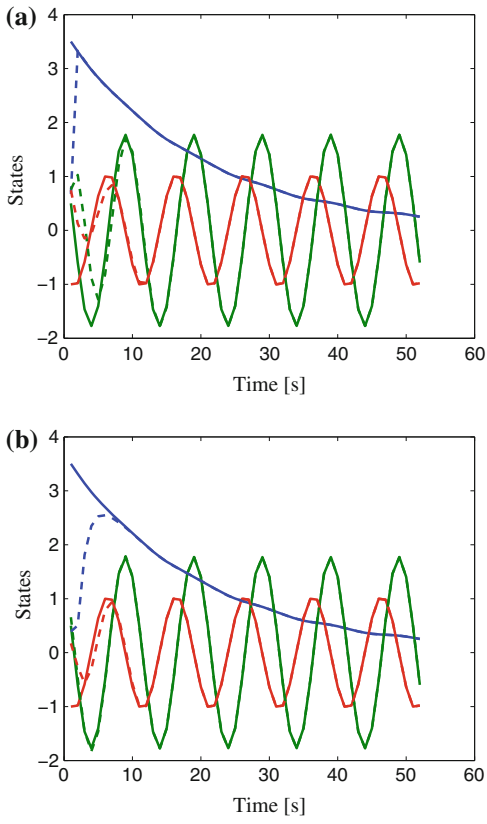

Fig. 8.2 Observers' estimates in the periodic case. **a** Estimation of the first observer. **b** Estimation of the second observer

We assume that two devices are estimating the state of the plant measuring distinct outputs. Specifically, the first node has access to the first state, that is, $y_1 = [1\ 0\ 0]x$, while the second node measures the output $y_2 = [0\ 1\ 1]x$.

Figure 8.2 illustrates the simulation results in case of periodic communications. The dotted line represents the estimations of each node, while the actual states of the plant are plotted in continuous line. It can be appreciated how the estimations of the two observers asymptotically converge to the actual states of the plant.

Figure 8.3 depicts instead the evolutions of the plant state and the estimations of the observers for the event-based communication policy. The triggering threshold is set to $\delta = 0.9$. It is interesting to see how in this case the estimate error of both the observers is bounded. In the periodic case, we experienced 102 transmissions between the observers, while in the event-based case, we counted 51 transmissions, resulting in a save of communication of 50 %.

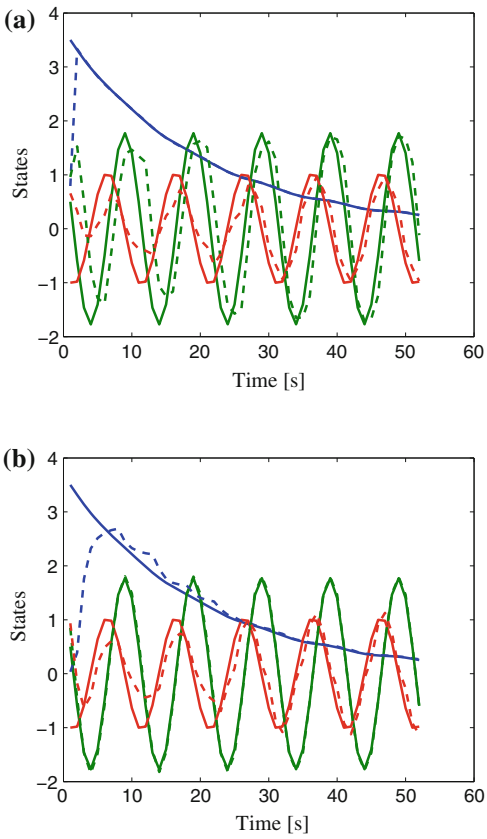

Fig. 8.3 Observers' estimates in the event-based case. **a** Estimation of the first observer. **b** Estimation of the second observer

8.5 Conclusions

This chapter has presented a novel distributed event-based estimator. The event-based method is compared to a periodic implementation, showing that it effectively achieves a reduction of the communication among the nodes. Such reduction of communication has the likely effect of reducing both the traffic load in the network and the energy expenditure of the nodes, still ensuring acceptable performance of the closed-loop system.

The next chapter includes both distributed observers and distributed controllers based on the same structure, while Chap. 11 extends the ideas in this chapter to the case of unreliable networks, considering packet dropouts and time delays.

Chapter 9
Suboptimal Distributed Control and Estimation: Application to a Four Coupled Tanks System

Francisco R. Rubio, Karl H. Johansson and Dimos V. Dimarogonas

9.1 Introduction

In this chapter, we distinguish between decentralized control strategies where agents only have access to local measurements, from distributed control strategies where agents have access to local measurements and the measurements from neighboring agents. A decentralized control strategy can be sufficiently effective when the couplings between agents are weak, [164, 255, 256]. If the agents' coupling is not weak, a full distributed feedback control approach must be employed, where each agent uses its state, the local measurements, and that of neighboring agents to build a control action. This mechanism of information exchange in the network, can assure asymptotic stability with stronger agent coupling than decentralized control strategy, [255, 259].

A natural extension of the single-plant NCS framework is considering the control of systems consisting of interconnected subsystems. These can be understood as the combination of networked control system, multi-agent systems, and large-scale systems. Much effort is still to be done to properly understand multiple interconnected systems over realistic channels working together in a distributed fashion, [14]. Such a class of systems include formation control, [187] and multi-agent cooperative control problems, [127]. Most of these works have in common that the primary control objective is governing the behavior of the agents themselves, focusing on the research of distributed cooperative strategies.

F.R. Rubio (✉)
Dpto. de Ingeniería de Sistemas y Automática, Escuela Superior de Ingenieros, Universidad de Sevilla, Seville, Spain
e-mail: rubio@us.es

K.H. Johansson · D.V. Dimarogonas
School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: kallej@kth.se

D.V. Dimarogonas
e-mail: dimos@kth.se

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_9

The approach in this work is different in the sense that a WSN is employed to collectively control a large-scale plant, not the agents themselves. The proposed problem considers a discrete linear time-invariant (LTI) process being controlled by a network of agents that may both, collect information about the evolution of the plant, and apply control actions to achieve a given goal. The problem makes full sense for geographically distributed processes where the agents have access only to partial information and actuate, possibly, only on specific control channels. In other words, no agent has the information, neither the control capabilities, to estimate and drive the overall process on its own. In this context, the networked structure of the agents plays an essential role as neighboring agents are allowed to share information and cooperate to achieve the system-wide goal.

The chapter provides an innovative estimation and control scheme so that the joint action of all agents allows to control the system and to monitor its state from different locations. To this goal, the agents exhibit all or part of the following features: sensing, actuating, computation, and communication. Each node implements an observer&controller structure based on local Luenberger-like observers in combination with consensus strategies, the first part being responsible for updating node's estimation based on local sensed information, while the former takes into account the data transmitted from neighboring nodes.

Two different communication policies are presented. First, a periodic time-driven pattern, where the nodes are assumed to communicate at every sampling time; then, an event-driven scheme that triggers agent's communications only when significant information requires to be transmitted. Event-driven approaches, see [61, 99, 154, 239], are specially beneficial in the context of WSNs, as a reduction in the transmission frequency implies bandwidth savings but also an improvement in average transmission delays and packet collisions, for back-off retransmissions are reduced. Moreover, in WSNs the battery life span is of great importance, and the energy consumption is mainly related to the number of transmissions of the device.

As it will be shown, in the proposed approach the Separation Principle does not hold, this forcing to a joint design of control and estimation. As a first extension of our previous works in [175, 177, 205] to tackle both, distributed control and estimation, this chapter neglects the effects of transmission delays and dropouts in the network. Although both are admittedly relevant phenomena in real applications, these have been dropped for the present work in favor of obtaining a tractable mathematical design method.

The stability of the solution is guaranteed through discrete-time Lyapunov functions, from which the design problem is cast as an optimization problem subject to matrix inequalities. Asymptotic stability and global ultimately uniformly boundedness (GUUB) of solutions are proved, respectively, for the time-driven and the event-driven approaches. Remarkably, to the best of our knowledge, this is the first approach that considers distributed estimation and control in a cost-guaranteed scheme using an event-based sampling policy.

As a conjunction of the above-mentioned features, the chapter presents a method to design the distributed estimation and control system that allows the user to trade-off control performance, control effort, and communication savings, guaranteeing closed-loop system stability.

The chapter is organized as follows. Section 9.2 describes the distributed problem under consideration. In Sect. 9.3, the suboptimal control and observation problem is formally stated. Next, Sects. 9.4 and 9.5 describe the proposed solution for both, the time-driven and event-driven scenarios. Section 9.6 presents a four-tank level control experimental setup on which experiments were conducted to show the effectiveness of the approach. Finally, Sect. 9.7 outlines the main conclusions and future work.

Notation The superscripts T and -1 stand for matrix transposition and matrix inverse, respectively. $\mathbb{R}^n$ denotes the n-dimensional Euclidean space and the notation $P > 0$ means that P is real symmetric and positive definite. $diag\{\ldots\}$ stands for a block-diagonal matrix and $tr\{\cdot\}$ means the trace of a matrix. $\|\cdot\|_2$ refers to the Euclidean norm for vectors and induced two-norm for matrices. In block matrices, the symbol $*$ refers to the corresponding block inferred from matrix symmetry.

9.2 System Description: Initial Considerations

Consider the distributed control and estimation scheme depicted in Fig. 9.1. The process is monitored and controlled through a network of agents. The problem consists of designing a fully distributed scheme such that the collective behavior of all agents drives the plant to stability with a cost-guaranteed control performance. This problem is of interest when the agents only have access to partial information of the plant states and can actuate only on a subset of the plant's control channels. That is, no single agent is capable to control or estimate the plant state on its own.

In the following, the different elements composing the distributed system are described in detail.

9.2.1 Plant

Consider a discrete LTI system described in the state-space representation. As Fig. 9.1 illustrates, the plant is being controlled and/or sensed by a set of p agents, each one possibly managing a different control signal. The dynamics of the system can be described as

$$x(k+1) = Ax(k) + \sum_{i=1}^{p} B_i u_i(k), \tag{9.1}$$

where $x \in \mathbb{R}^n$ is the state of the plant and $u_i \in \mathbb{R}^{d_i}$ $(i = 1, \ldots, p)$ is the control signal that agent i applies to the system. Matrices $A \in \mathbb{R}^{n \times n}$ and $B_i \in \mathbb{R}^{n \times d_i}$ are known. For those agents with no direct access to plant inputs, matrices B_i are set to zero.

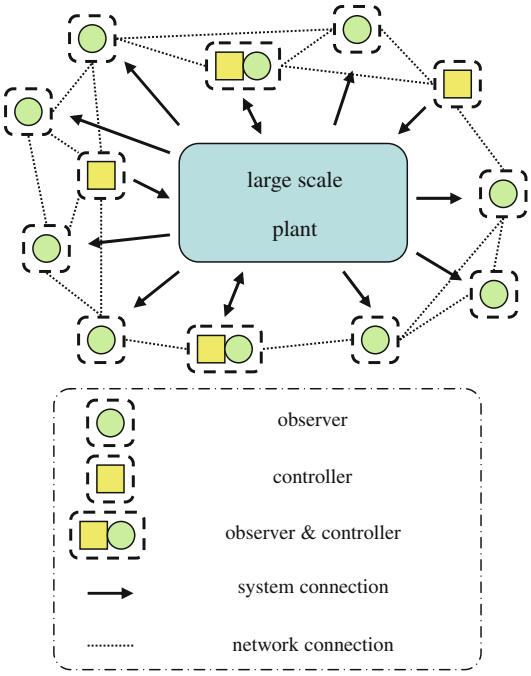

Fig. 9.1 Distributed scheme for the control of a large-scale plant

Defining an augmented control matrix as

$$B \triangleq \left[B_1 \; B_2 \; \ldots \; B_p \right]$$

and an augmented control vector

$$\mathscr{U}(k) \triangleq \left[u_1^T(k) \; u_2^T(k) \; \ldots \; u_p^T(k) \right]^T, \tag{9.2}$$

Equation (9.1) can be compactly rewritten as

$$x(k+1) = Ax(k) + B\mathscr{U}(k), \tag{9.3}$$

where $\mathscr{U}(k) \in \mathbb{R}^d$, with $d = \sum_{i=1}^{p} d_i$.

9.2.2 Network

The network in Fig. 9.1 is topologically defined by its graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with l links between p agents. The graph $\mathscr{G}$ is, in general, directed, with nodes $\mathscr{V} = \{1, 2, \ldots, p\}$ and links $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$. The set of agents connected to node i is named *the neighborhood of* i and is denoted by $\mathscr{N}_i \equiv \{j : (i, j) \in \mathscr{E}\}$. Link (i, j) implies that agent i receives information from agent j.

9.2.3 Agents

Each agent of the network exhibits all or part of the following capabilities: sensing plant outputs, estimating the state of the plant, applying control actions, and communicating with neighboring agents.

The approach adopted in this work is an observer-based scheme in which every agent is assumed to build its own estimate of the plant's states based on the information locally collected by the agent (plant output) and that shared with neighboring agents.

In general, agent i measures a specific plant output, y_i:

$$y_i(k) = C_i x(k) \in \mathbb{R}^{r_i}, \tag{9.4}$$

where the matrices $C_i \in \mathbb{R}^{r_i \times n}$ are known. If an agent j has no sensing capabilities, then its corresponding matrix C_j is set to zero.

Let C denote an augmented output matrix defined as

$$C \triangleq \left[C_1^T \; C_2^T \; \ldots \; C_p^T \right]^T .$$

It is assumed that the pair (A, B) in (9.3) is stabilizable and (A, C) is detectable.

On the other hand, the control counterpart of each agent generates an estimation-based control input to the plant, $u_i(k)$, in the form

$$u_i(k) = K_i \hat{x}_i(k) \in \mathbb{R}^{d_i}, \tag{9.5}$$

where $\hat{x}_i \in \mathbb{R}^n$ denotes the estimation of agent i, and $K_i \in \mathbb{R}^{d_i \times n}$ $(i = 1 \in \mathscr{V})$ are local controllers to be designed. Let K denote the augmented control matrix, defined by

$$K = \left[K_1^T \; K_2^T \; \ldots \; K_p^T \right]^T .$$

Every agent $i \in \mathscr{V}$ implements an estimator of the plant's state based on the following structure

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + BK\hat{x}_i(k) + M_i(y_i(k) - C_i\hat{x}_i(k)) + \sum_{j \in \mathscr{N}_i} N_{ij}(\hat{x}_j(k) - \hat{x}_i(k)). \tag{9.6}$$

Looking at Eq. (9.6), each agent has two different sources of information to correct its estimates. The first one consists of the output measured from the plant, $y_i(k)$, which is used similarly to a classical Luenberger observer, $M_i(y_i(k) - \hat{y}_i(k))$, being $M_i,\ i \in \mathscr{V}$, the observers to be designed. The second source of information comes from the estimates received from neighboring nodes, which are also used to correct estimations through the terms $N_{ij}(\hat{x}_j(k) - \hat{x}_i(k)),\ \forall j \in \mathscr{N}_i$, where $N_{ij},\ (i, j) \in \mathscr{E}$, are consensus gains to be synthesized.

It is worth recalling that the individual agents have no information about the exact control signal being applied to the plant, as each agent implements different control actions based on its particular state estimation (9.5), that is,

$$BK\hat{x}_i(k) \neq B\mathscr{U}(k) = \sum_{j=1}^{p} B_j K_j \hat{x}_j(k), \ \forall i.$$

Ideally, Eq. (9.6) should be implemented using the augmented control vector $\mathscr{U}(k)$ that the network, as a whole, applies to the plant. However, this information is not available at the agents. To circumvent this difficulty and make Eq. (9.6) realizable, the proposed solution consists, roughly speaking, of letting each agent to run its observer with the augmented control vector obtained from its particular estimate. In general, estimated and actual control inputs are different, but if the observers are properly designed and the nodes estimations converge to the plant states, these differences progressively vanish.

9.3 Problem Formulation

In the previous section, the notation used throughout the chapter has been introduced as well as the dynamics of the elements involved in the proposed distributed scheme. Next, the problem to be solved is formally stated:

Suboptimal distributed control and estimation problem Consider a discrete LTI plant with dynamics given by (9.1). The plant is monitored and controlled by a set of p agents connected through a network whose topology can be represented by the directed graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. The dynamics of the agents are given by (9.6), each of them receiving a measurement from the plant defined by (9.4), and applying a control signal defined by (9.5). Then, given the cost function

$$J = \sum_{j=k_0}^{\infty} x^T(j) Q_x x(j) + \sum_{j=k_0}^{\infty} \sum_{i \in \mathscr{V}} \left[e_i^T(j) Q_i e_i(j) + u_i^T(j) R_i u_i(j) \right], \qquad (9.7)$$

with Q_x, Q_i, $R_i > 0$ for all $i \in \mathscr{V}$, the suboptimal distributed control and estimation problem consists of finding observers M_i, $i \in \mathscr{V}$, consensus gains N_{ij}, $(i, j) \in \mathscr{E}$, and controllers K_i, $i \in \mathscr{V}$, such that:

- The state of the system $x(k)$ is asymptotically stable.
- The estimation errors $e_i(k) \triangleq x(k) - \hat{x}_i(k)$, $i \in \mathscr{V}$, are asymptotically stable.
- A cost-guaranteed solution is obtained by minimizing the upper bound of the cost function (9.7).

9.4 Periodic Sampling Case

Let us consider first the case of periodic communications between agents. That is, each agent receives/communicates information from/to other elements in the network at every discrete time step $k \in \mathbb{N}$.

9.4.1 Dynamics of the State and Estimation Error

First, the following sets must be defined:

$$\mathscr{M} = \{M_i, i \in \mathscr{V}\}, \tag{9.8}$$
$$\mathscr{N} = \{N_{ij}, (i, j) \in \mathscr{E}\}, \tag{9.9}$$
$$\mathscr{K} = \{K_i, i \in \mathscr{V}\}. \tag{9.10}$$

Proposition 9.1 *The dynamics of the plant state $x(k)$ is given by*

$$x(k+1) = (A + BK)\, x(k) + \Upsilon(\mathscr{K}) e(k), \tag{9.11}$$

where

$$\Upsilon(\mathscr{K}) = \left[-B_1 K_1 \; -B_2 K_2 \; \ldots \; -B_p K_p \right].$$

Proof The proof is immediate from Eq. (9.3) and the definition of the estimation errors. The following proposition studies the evolution of the error vector, defined as $e(k) \triangleq [e_1^T(k), \ldots, e_p^T(k)]^T \in \mathbb{R}^{np}$.

Proposition 9.2 *The dynamics of the error vector $e(k)$ is given by*

$$e(k+1) = (\Phi(\mathscr{M}) + \Psi(\mathscr{K}) + \Lambda(\mathscr{N}))\, e(k), \tag{9.12}$$

where

$$\Phi(\mathscr{M}) = diag\{(A - M_1 C_1), \ldots, (A - M_p C_p)\},$$
$$\Psi(\mathscr{K}) = diag\{BK, \ldots, BK\} + \begin{bmatrix} -B_1 K_1 & \ldots & -B_p K_p \\ \vdots & \ddots & \vdots \\ -B_1 K_1 & \ldots & -B_p K_p \end{bmatrix},$$
$$\Lambda(\mathscr{N}) = \sum_{(i,j) \in \mathscr{E}} \Theta(N_{ij}),$$

and

$$\Theta(N_{ij}) = \begin{array}{c} \begin{matrix} \textit{column} & i & & j & \\ \end{matrix} \\ \begin{bmatrix} 0 \cdots & 0 & \cdots & 0 & \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 \cdots & -N_{ij} & \cdots & N_{ij} & \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 \cdots & 0 & \cdots & 0 & \cdots 0 \end{bmatrix} \end{array} \begin{matrix} \\ \\ \textit{row } i \\ \\ \\ \end{matrix}$$

Proof The proof is found in Appendix A.

Consider now the augmented state vector defined as $\xi(k) \triangleq \left[x^T(k)\ e^T(k)\right]^T$, whose evolution is given next.

Proposition 9.3 *The evolution of the augmented state vector $\xi(k)$ is given by*

$$\xi(k+1) = \Omega(\mathscr{M}, \mathscr{N}, \mathscr{K})\xi(k), \tag{9.13}$$

where

$$\Omega(\mathscr{M}, \mathscr{N}, \mathscr{K}) = \left[\begin{array}{c|c} A + BK & \Upsilon(\mathscr{K}) \\ \hline 0 & \Phi(\mathscr{M}) + \Psi(\mathscr{K}) + \Lambda(\mathscr{N}) \end{array}\right].$$

The structure of (9.13) reveals that the separation principle does not hold in this case, for matrix $\Psi(\mathscr{K})$ depends on the controllers to be designed. This can be easily justified if we recall that the agents ignore the actual control signal being applied to the plant, and resort to estimations based on the knowledge of the distributed controllers. However, despite this drawback, it will be shown that it is possible to propose an unified design in which all the elements, namely controllers and observers, can be designed to guarantee the overall system stability.

Proposition 9.4 *The cost function (9.7) can be written as*

$$J = \sum_{j=k_0}^{\infty} \xi^T(j)(Q + \bar{K}^T R \bar{K})\xi(j), \tag{9.14}$$

where

$$\begin{aligned} Q &= diag\{Q_x, Q_1, Q_2, \ldots, Q_p\}, \\ R &= diag\{R_1, R_2, \ldots, R_p\}, \\ \bar{K} &= \begin{bmatrix} K_1 & -K_1 & 0 & \ldots & 0 \\ K_2 & 0 & -K_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_p & 0 & 0 & \ldots & -K_p \end{bmatrix}. \end{aligned}$$

This result can be easily proved by substituting matrices $Q, R, \bar{K}$ and the augmented vector $\xi(k)$ in (9.14).

9.4.2 Controller and Observer Design

The design method resorts to a Lyapunov-based approach to prove asymptotic stability of the plant state and the estimation errors. It is a centralized design in which both, controllers and observers, are designed together.

The following theorem proposes a centralized design method through an optimization problem subject to a nonlinear matrix inequality.

Theorem 9.1 *Given matrices Q and R of the cost function (9.14), the **suboptimal distributed control and estimation problem** can be solved by finding the sets $\mathscr{M}, \mathscr{N}, \mathscr{K}$ in (9.8)–(9.10), and a positive definite matrix $P = diag(P_x, P_e)$, with $P_x \in \mathbb{R}^n$ and $P_e \in \mathbb{R}^{np}$, that resolve the following optimization problem:*

$$\min_{P,\mathscr{M},\mathscr{N},\mathscr{K}} \alpha, \tag{9.15}$$

subject to

$$\begin{bmatrix} -P & \Omega^T & I & \bar{K}^T \\ * & -P^{-1} & 0 & 0 \\ * & * & -\alpha Q^{-1} & 0 \\ * & * & * & -\alpha R^{-1} \end{bmatrix} < 0. \tag{9.16}$$

Proof Consider the following quadratic Lyapunov function:

$$V(\xi) = \xi^T(k) P \xi(k), \tag{9.17}$$

where P is a positive definite matrix. The forward difference is defined as

$$\Delta V(k) = \xi^T(k+1) P \xi(k+1) - \xi^T(k) P \xi(k).$$

Using Proposition 9.3, the forward difference can be expressed in the following way:

$$\begin{aligned} \Delta V(k) &= \xi^T(k) \Omega^T P \Omega \xi(k) - \xi^T(k) P \xi(k), \\ &= \xi^T(k) \left(\Omega^T P \Omega - P \right) \xi(k). \end{aligned}$$

From Lyapunov stability, the state of the system and the estimation errors are asymptotically stable if, and only if, matrix $\Omega^T P \Omega - P$ is negative definite [50].

Using Schur complement, the following inequalities are equivalent:

$$\Omega^T P \Omega - P < 0 \leftrightarrow \begin{bmatrix} -P & \Omega^T \\ * & -P^{-1} \end{bmatrix} < 0.$$

The previous inequality is guaranteed by (9.16). This way, the stability properties of the estimation errors and the plant state are proved.

Let us move to the optimality of the method. Note that condition (9.16) implies that

$$\Omega^T P \Omega - P + \frac{1}{\alpha} Q + \frac{1}{\alpha} \bar{K}^T R \bar{K} < 0.$$

Therefore

$$\xi^T(k) \left(\Omega^T P \Omega - P \right) \xi(k) < -\xi^T(k) \frac{1}{\alpha} (Q + \bar{K}^T R \bar{K}) \xi(k).$$

Taking into account the previous considerations, it yields

$$\begin{aligned} \Delta V(k) &= \xi^T(k) \left(\Omega^T P \Omega - P \right) \xi(k) \\ &< -\frac{1}{\alpha} \xi^T(k) (Q + \bar{K}^T R \bar{K}) \xi(k). \end{aligned} \tag{9.18}$$

Calculating the summation of both sides of (9.18) from k_0 to k:

$$\sum_{j=k_0}^{k} \Delta V(j) < -\frac{1}{\alpha} \sum_{j=k_0}^{k} \xi^T(j) (Q + \bar{K}^T R \bar{K}) \xi(j).$$

Observe that $\sum_{j=k_0}^{k} \Delta V(j) = \sum_{j=k_0}^{k} (V(j+1) - V(j)) = V(k+1) - V(k_0)$. When $k \to \infty$, it holds that,

$$\lim_{k \to \infty} V(k) - V(k_0) < -\frac{1}{\alpha} \sum_{j=k_0}^{\infty} \xi^T(j) (Q + \bar{K}^T R \bar{K}) \xi(j).$$

Then, taking into account Proposition 9.4 and the positive definiteness of the Lyapunov function $V(k)$, it holds that

$$J < \alpha V(k_0).$$

The value of $V(k_0)$ depends on the initial condition. Therefore, by minimizing α an upper bound of the cost function J is minimized regardless of the initial conditions.

Remark 9.1 The solution of the optimization problem ensures the positiveness of α, as otherwise, it is not possible to satisfy the nonlinear matrix inequality (9.16). Additionally, the cost function (9.7) has a positive finite lower bound by definition.

Note that inequality (9.16) is nonlinear in the decision variables because of the presence of the term P^{-1}. Next, two solutions are presented to deal with this nonlinearity.

(1) Constraint on P^{-1} Introduce the following additional constraint: $-P^{-1} < -\frac{1}{\mu}I$, being μ a positive design scalar. Note that previous condition is equivalent to $P < \mu I$. Then, the nonlinear constraint in Theorem 9.1 can be replaced by:

$$\begin{cases} \Upsilon < 0, \\ P < \mu I \end{cases} \tag{9.19}$$

where Υ is the matrix required to be definite negative in (9.16), but substituting the term P^{-1} by $\frac{1}{\mu}I$.

Compared with the method introduced in [275] in which $P \triangleq \mu I$, the one proposed here cover a much wider range of possible solutions in the space of positive definite matrices.

(2) Cone complementary algorithm We can also adapt an extended procedure (see [64]), which lets us address the nonlinearity P^{-1} by introducing some new matrix variables and constraints.

First, define a new variable T such that $P^{-1} \geq T$, and replace the inequality (9.16) with

$$\begin{bmatrix} -P & \Omega^T & I & \bar{K}^T \\ * & -T & 0 & 0 \\ * & * & -\alpha Q^{-1} & 0 \\ * & * & * & -\alpha R^{-1} \end{bmatrix} < 0, \quad P^{-1} \geq T. \tag{9.20}$$

Since $P^{-1} \geq T$ is equivalent to $P \leq T^{-1}$, condition (9.20) is equal to

$$\begin{bmatrix} -P & \Omega^T & I & \bar{K}^T \\ * & -T & 0 & 0 \\ * & * & -\alpha Q^{-1} & 0 \\ * & * & * & -\alpha R^{-1} \end{bmatrix} < 0, \begin{bmatrix} T^{-1} & I \\ I & P^{-1} \end{bmatrix} \geq 0,$$

by applying Schur complement to the second inequality. Then, introducing new variables $\hat{T}$, $\hat{P}$, the original condition (9.16) can be represented by

$$\begin{bmatrix} -P & \Omega^T & I & \bar{K}^T \\ * & -T & 0 & 0 \\ * & * & -\alpha Q^{-1} & 0 \\ * & * & * & -\alpha R^{-1} \end{bmatrix} < 0, \begin{bmatrix} \hat{T} & I \\ I & \hat{P} \end{bmatrix} \geq 0, \ \hat{T} = T^{-1}, \ \hat{P} = P^{-1}.$$

Using a cone complementarity problem, one can obtain feasible solutions for the optimization problem in Theorem 9.1 by solving the following problem:

$$\text{Minimize } \mathbf{Tr}\left(\hat{P}P + \hat{T}T\right)$$

subject to

$$\begin{cases} \begin{bmatrix} -P & \Omega^T & I & \bar{K}^T \\ * & -T & 0 & 0 \\ * & * & -\alpha Q^{-1} & 0 \\ * & * & * & -\alpha R^{-1} \end{bmatrix} < 0, \begin{bmatrix} \hat{T} & I \\ I & \hat{P} \end{bmatrix} \geq 0 \\ \begin{bmatrix} P & I \\ I & \hat{P} \end{bmatrix} \geq 0, \begin{bmatrix} T & I \\ I & \hat{T} \end{bmatrix} \geq 0, \ \alpha < \alpha^*, \end{cases} \tag{9.21}$$

where α^* is minimized using a bisection algorithm.

In order to find a solution for this problem, the iterative algorithm introduced in [64] can be applied. See [170] for further details.

Regarding the complexity, the number of decision variables depends not only in the number of nodes and links, but also in the dimension of the control inputs and system outputs:

$$O_P\left\{\frac{n(n+1)}{2} + \frac{np(np+1)}{2}\right\}, \ O_{\mathscr{M}}\left\{n\sum^{p} r_i\right\}, \ O_{\mathscr{N}}\left\{2ln^2\right\}, O_{\mathscr{K}}\left\{n\sum^{p} d_i\right\}, \ O_\alpha\{1\}.$$

The second method also requires the computation of three additional variables, $\hat{P}, T, \hat{T}$, with the same dimension that P. Furthermore, three additional LMIs must be solved, increasing the computational effort. It is possible to force matrix P to be block diagonal, i.e., $P = \{P_x, P_1, \ldots, P_p\}$, reducing drastically the number of variables to $O_P\left\{p\frac{n(n+1)}{2}\right\}$. Although the outperformed experiments suggest that this modification does not introduce hard constraints, in general this might incur in feasibility problems.

Comparing both solutions in terms of conservatism, the former obtains worse results due to the additional constraint.

Remark 9.2 The design method that stems from Theorem 9.1 can be performed off-line prior to the implementation, and requires some centralized information: the network topology, the information that every node collects from the plant, what control channels it has access to, etc. Nonetheless, once the observers and controllers are designed, their implementation is fully distributed, and each agent requires only available local information to operate.

9.5 Event-Based Sampling Case

As briefly discussed in the introduction, event-based control is a means to reduce network usage by invoking a communication among the nodes only if significant information deserves transmission, [61, 99]. Furthermore, event-based schemes are usually more efficient in terms of energy consumption, as most of the energy expended in distributed tasks is associated with communications, specially in the case of wireless networks.

For these reasons, the event-based sampling is an interesting approach with relevant practical implications. The idea is simple: Instead of taking/sending a measurement each time instant, sampling is only triggered by an event. The different definitions of *event* yields a variety of published results. One of the most used, yet most intuitively appealing, is the one that triggers an event whenever some variable of interest has exceeded a tolerance bound. This concept has been adapted to the problem at hand as it is discussed in the following section.

9.5.1 Triggering Rule

In the proposed control architecture, most of the energy consumption is due to the inter-agent communications, that periodically exchanges the plant state estimates. In this section, network usage and energy expenditure is reduced by triggering the transmissions only at specific time instants when an event occurs. Let $l_j(k)$ denote the last time instant before k when node j sent its estimated state to its neighbors. Next, a norm-based rule to trigger the communication events is defined.

Triggering rule Given a threshold $\delta_\omega \geq 0$, at instant k agent j broadcasts its estimates to every neighbor i if

$$\|\hat{x}_j(l_j(k)) - \hat{x}_j(k)\|_\infty \geq \delta_\omega, \quad \text{for } k > l_j(k). \tag{9.22}$$

This rule is completely compatible with the discrete-time setting used in throughout the chapter. Each node does not transmit its state at every k, but only at those k's verifying the previous condition. Observe that the agent only requires local information (past and present) to check the triggering condition.

9.5.2 Remodeling the System Dynamics

From a modeling point of view, the main difference between the time-driven and the event-driven paradigms described here is the non-uniform pattern of transmission of information. This modifies the behavior of the agents, whose dynamics can be now described as follows:

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + BK\hat{x}_i(k) + M_i(y_i(k) - C_i\hat{x}_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}(\hat{x}_j(l_j(k)) - \hat{x}_i(k)). \tag{9.23}$$

Equation (9.23) takes into consideration aperiodic communication through the variable $l_j(k)$, which can be different for each agent $j \in \mathcal{V}$. This equation can be rewritten as

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + BK\hat{x}_i(k) + M_i(y_i(k) - C_i\hat{x}_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}(\hat{x}_j(k) - \hat{x}_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}\omega_j(k), \tag{9.24}$$

where

$$\omega_j(k) = \hat{x}_j(l_j(k)) - \hat{x}_j(k). \tag{9.25}$$

Based on (9.24)–(9.25), the evolution of the agents with event-based communication is equivalent to the evolution with periodic communication, difference being in the term $\omega_j(k)$, given by Eq. (9.25). The term $\omega_j(k)$ can be interpreted as an external perturbation due to the discontinuous flow of information between neighbors that is reset to zero at every transmission time. This way, whenever any agent j broadcasts its state to its neighbors $\mathcal{N}_j$, it holds that $\omega_j(k) = 0$. It is worth pointing out that the disturbance that each agent j induces on its neighbors is unknown to them, but can be tracked by agent j, which has access to its own local estimations.

The following result is the counterpart of Proposition 9.2 for event-based communication. Due to its similarities, its proof is omitted.

Proposition 9.5 *Let $\omega(k) \triangleq \left[\omega_1^T(k) \ \ldots \ \omega_p^T(k)\right]^T$. Then, for the event-based sampling case the evolution of the state $x(k)$ is given by Proposition 9.1, and the dynamics of the estimation error $e(k)$ is given by*

$$e(k+1) = (\Phi(\mathscr{M}) + \Psi(\mathscr{K}) + \Lambda(\mathscr{N}))\, e(k) - \Gamma(\mathscr{N})\omega(k), \tag{9.26}$$

where the functions $\Phi(\mathscr{M})$, $\Psi(\mathscr{K})$, $\Lambda(\mathscr{N})$ are defined as in Proposition 9.2, and

$$\Gamma(\mathscr{N}) = \sum_{(i,j)\in\mathscr{E}} \Psi(N_{ij}),$$

where

$$\Psi(N_{ij}) = \begin{array}{c} \begin{matrix} column\ i & & j & \end{matrix} \\ \begin{bmatrix} 0\cdots 0\cdots & 0 & \cdots 0 \\ \vdots \quad\ \vdots & \vdots & \ \vdots \\ 0\cdots 0\cdots & N_{ij} & \cdots 0 \\ \vdots \quad\ \vdots & \vdots & \ \vdots \\ 0\cdots 0\cdots & 0 & \cdots 0 \end{bmatrix} \end{array} row\ i.$$

9.5.3 Stability and Trade-off Between Communication Reduction and Final Boundedness

In terms of stability, the event-based sampling approach makes more difficult to analyze the system stability. In fact, it will be shown that the presented approach will only allow to prove the system to be GUUB, that is, the system is attracted and restricted to lay within an arbitrarily small region around the equilibrium point.

Theorem 9.1 allows the design of suboptimal distributed controllers and observers for uniformly sampled systems. A question that naturally arises is: is it possible to use designs obtained from Theorem 9.1 in an event-sampling context? And in this case, how much does the performance deteriorate with respect to uniform sampling?

It will be shown that, under certain conditions, the answer to the first question is affirmative and performance degradation can be traded off with respect to transmission reductions. More precisely, in the event-based sampling policy considered here, the agents are allowed to communicate only in case that the difference between their current estimates and the last transmitted estimates exceed a given threshold. The price to be paid in this case is that asymptotic stability of the estimation errors—and, hence, of the state of the system—is no longer guaranteed. However, GUUB stability of the state $x(k)$ and error $e(k)$ can be proved.

Note that this section does not develop a new design method, but applies the results of Theorem 9.1 with a different sampling policy. Henceforth, the notation $\Upsilon, \Phi, \Psi, \Lambda, \Gamma$ instead of $\Upsilon(\mathscr{K}), \Phi(\mathscr{M}), \Psi(\mathscr{K}), \Lambda(\mathscr{N}), \Gamma(\mathscr{N})$ will be used to remark that sets $\mathscr{M}, \mathscr{K}, \mathscr{N}$ are assumed to be designed.

Theorem 9.2 *Consider the evolution of the state (9.11) and the error (9.26) with observers, controllers, and Lyapunov function $P = diag(P_x, P_e)$ designed through Theorem 9.1. Assume a communication policy where each node j broadcasts its estimate $\hat{x}_j$ to its neighbors $\mathscr{N}_j$ at each k when $\|\omega_j(k)\|_\infty \geq \delta_\omega$, with ω_j defined in (9.25). Then, the estimation error $e(k)$ and the state of the system $x(k)$ are GUUB with bounds*

$$\|e(k)\|_2 \leq \delta_e,$$
$$\|x(k)\|_2 \leq \delta_x,$$

where the bounds are given by

$$\delta_e = \delta_\omega \sqrt{\frac{np\lambda_{max}(P_e)}{\lambda_{min}(P_e)}}(\|\Phi + \Psi + \Lambda\|_\infty \alpha_e + \|\Gamma\|_\infty), \tag{9.27}$$

$$\delta_x = \delta_e \sqrt{\frac{\lambda_{max}(P_x)}{\lambda_{min}(P_x)}}(\|A + BK\|_2 \alpha_x + \|\Upsilon\|_2), \tag{9.28}$$

being α_x, α_e positive scalars defined by

$$\alpha_e = \frac{k_e + \sqrt{k_e + \lambda_{min}(X_e)\|\Gamma^T P_e \Gamma\|_\infty}}{\lambda_{min}(X_e)}, \tag{9.29}$$

$$\alpha_x = \frac{k_x + \sqrt{k_x + \lambda_{min}(X_x)\|\Upsilon^T P_x \Upsilon\|_2}}{\lambda_{min}(X_x)}, \tag{9.30}$$

with $k_x = \|\Upsilon^T P_x (A + BK)\|_2$, $k_e = \|\Gamma^T P_e (\Phi + \Psi + \Lambda)\|_\infty$, and X_e, X_x being the unique positive definite matrices such that:

$$(\Phi + \Psi + \Lambda)^T P_e (\Phi + \Psi + \Lambda) - P_e = -X_e \tag{9.31}$$

$$(A + BK)^T P_x (A + BK) - P_x = -X_x. \tag{9.32}$$

Proof Consider the following Lyapunov function for the observation error:

$$V(e) = e^T(k) P_e e(k),$$

with P_e obtained from Theorem 9.1. The forward increment of the Lyapunov function is given by

$$\begin{aligned} \Delta V(e) &= V(k+1) - V(k) \\ &= e^T(k+1) P_e e(k+1) - e^T(k) P_e e(k). \end{aligned}$$

Using the dynamics of the observation error given in Proposition 9.5, it turns out[1]:

$$\begin{aligned} \Delta V(e) &= [(\Phi + \Psi + \Lambda)e + \Gamma\omega]^T P_e [(\Phi + \Psi + \Lambda)e + \Gamma\omega] - e^T P_e e \\ &= e^T (\Phi + \Psi + \Lambda)^T P_e (\Phi + \Psi + \Lambda)e - e^T P_e e + \omega^T \Gamma^T P_e \Gamma\omega \\ &\quad + 2\omega^T \Gamma^T P_e (\Phi + \Psi + \Lambda)e. \end{aligned}$$

From Theorem 9.1, the error $e(k)$ is asymptotically stable with periodic sampling, so there exists the positive definite matrix X_e defined in (9.31). Then, it holds

$$\Delta V(e) = -e^T X_e e + \omega^T \Gamma^T P_e \Gamma\omega + 2\omega^T \Gamma^T P_e (\Phi + \Psi + \Lambda)e.$$

Using the infinity norm, previous equation can be bounded as

$$\begin{aligned} \Delta V(e) \leq &-\lambda_{min}(X_e)\|e\|_\infty^2 + \|\Gamma^T P_e \Gamma\|_\infty \|\omega\|_\infty^2 \\ &+ 2\|\Gamma^T P_e (\Phi + \Psi + \Lambda)\|_\infty \|\omega\|_\infty \|e\|_\infty. \end{aligned}$$

[1] Time indexes have been removed to alleviate the notation.

The right-hand side of the equation above is an algebraic second order equation in $\|e\|_\infty$. The roots can be obtained by imposing $a\|e\|_\infty^2 + b\|e\|_\infty + c = 0$, where

$$\begin{aligned} a &= -\lambda_{min}(X_e), \\ b &= 2\|\Gamma^T P_e(\Phi + \Psi + \Lambda)\|_\infty \|\omega\|_\infty, \\ c &= \|\Gamma^T P_e \Gamma\|_\infty \|\omega\|_\infty^2. \end{aligned}$$

Note that $a < 0$ and $b, c > 0$. The unique positive root is exactly

$$\|e\|_\infty = -\frac{b + \sqrt{b^2 - 4ac}}{2a} = \alpha_e \|\omega\|_\infty,$$

where α_e is given in Eq. (9.29). Because of the sign of a, it is easy to see that the Lyapunov function $V(k)$ decreases whenever $\|e(k)\|_\infty > \alpha_e \|\omega(k)\|_\infty$. Using the bound on $\|\omega(k)\|_\infty$, it yields that $\Delta V(k) < 0$ in the region $\|e(k)\|_\infty > \alpha_e \delta_\omega$.

Now consider k^* as the time instant when the estimation errors enters in the region $\|e(k)\|_\infty \leq \alpha_e \delta_\omega$. Then, taking into account the error dynamics given by (9.26), one can easily obtain that

$$\max \|e(k^* + 1))\|_\infty = (\|\Phi + \Psi + \Lambda\|_\infty \alpha_e + \|\Gamma\|_\infty)\delta_\omega,$$

so the error can leave the region $\|e(k)\|_\infty \leq \alpha_e \delta_\omega$. After that, the Lyapunov function decreases again. Using the inequality $\|e\|_2 < \sqrt{np}\|e\|_\infty, \forall e \in \mathbb{R}^{np}$ the maximum of the 2-norm in $k^* + 1$ can be obtained as

$$\|e(k^* + 1))\|_2 = \sqrt{np}(\|\Phi + \Psi + \Lambda\|_\infty \alpha_e + \|\Gamma\|_\infty)\delta_\omega.$$

Note that the decreasing of the Lyapunov function does not ensure the decreasing of neither the two-norm nor the infinity norm, but the P_e-induced norm. The maximum of the P_e-induced norm for all instant $k > k^*$ is given by

$$\|e(k)\|_{P_e} = \sqrt{e^T P_e e} \leq \sqrt{\lambda_{max}(P_e)}\|e(k^* + 1)\|_2,$$

where inequality $\lambda_{min}(P_e)\|e\|_2^2 \leq e^T P_e e \leq \lambda_{max}(P_e)\|e\|_2^2$ has been employed. Taking into account the equation above, the final two-norm can be bounded as

$$\|e(k)\|_2 \leq \sqrt{\frac{\lambda_{max}(P_e)}{\lambda_{min}(P_e)}}\|e(k^* + 1)\|_2, \forall k > k^*.$$

This way, the boundedness of the estimation error has been proved.

Consider now the following Lyapunov function for the state of the plant

$$V(x) = x^T(k) P_x x(k),$$

with P_x obtained from Theorem 9.1. The forward increment of the Lyapunov function is given by

$$\begin{aligned}\Delta V(x) &= V(k+1) - V(k) \\ &= x^T(k+1)P_x x(k+1) - x^T(k)P_x x(k).\end{aligned}$$

Given the system dynamics in Proposition 9.1, it turns out that

$$\begin{aligned}\Delta V(x) &= [(A+BK)x + \Upsilon e]^T P_x[(A+BK)x + \Upsilon e] - x^T P_x x \\ &= x^T(A+BK)^T P_x(A+BK)x - x^T P_x x + e^T \Upsilon^T P_x \Upsilon e \\ &\quad + 2e^T \Upsilon^T P_x(A+BK)x.\end{aligned}$$

From Theorem 9.1, the state $x(k)$ is asymptotically stable under periodic sampling, so there exists the positive definite matrix X_x defined in (9.32). Then, it holds

$$\Delta V(x) = -x^T X_x x + e^T \Upsilon^T P_x \Upsilon e + 2e^T \Upsilon^T P_x(A+BK)x.$$

Taking norms, the forward difference can be bounded as follows:

$$\Delta V(x) \leq -\lambda_{min}(X_x)\|x\|_2^2 + \|\Upsilon^T P_x \Upsilon\|_2\|e\|_2^2 + 2\|\Upsilon^T P_x(A+BK)\|_2\|e\|_2\|x\|_2.$$

The right-hand side of the equation above is again an algebraic second order equation in $\|x\|_2$. Operating as before, it is easy to see that the Lyapunov function $V(x)$ decreases whenever $\|x(k)\|_2 > \alpha_x\|e(k)\|_2$. Using the derivations above, it yields that $V(x)$ decreases if $\|x(k)\|_2 > \alpha_x\delta_e$, where δ_e is given in Eq. (9.27).

Now consider k^* as the time instant when the state enters in the region $\|x(k)\|_2 < \alpha_x\delta_e$. Then, taking into account the system dynamics given in Proposition 9.5, one can easily obtain that

$$\|x(k^*+1))\|_2 \leq (\|A+BK\|_2\alpha_x + \|\Upsilon\|_2)\,\delta_e.$$

If the state leaves the region $\|x(k)\|_2 \leq \alpha_x\delta_e$, the Lyapunov function will decrease again. Hence, the P_x-induced norm of the state decreases. The maximum of the P_x-induced norm for all instant $k > k^*$ is given by

$$\|x(k)\|_{P_x} \leq \sqrt{\lambda_{max}(P_x)}\|x(k^*+1)\|_2.$$

Then, the final Euclidean norm can be bounded as

$$\|x(k)\|_2 \leq \sqrt{\frac{\lambda_{max}(P_x)}{\lambda_{min}(P_x)}}\|x(k^*+1)\|_2, \forall k > k^*.$$

This way, the boundedness of the state is proved. This ends the proof.

Remark 9.3 It is worth pointing out that the choice of the infinity norm of $\omega(k)$ as triggering condition in Theorem 9.2 is not arbitrary. Given that each observer has access only to local information, the infinity norm can be practically implemented using just local information: As each node sends its information whenever $\|\omega_i(k)\|_\infty \geq \delta_\omega$, it turns out that at intersampling times $\|\omega(k)\|_\infty < \delta_\omega$, thus it is possible to bound $\|\omega(k)\|_\infty$ from below resorting only to local information at each node.

The parameter δ_ω is related to the size of the ultimate bound region of the estimation error $e(k)$ and, indirectly, with the boundedness of the final region of $x(k)$. By enlarging the value of δ_ω, it is possible to reduce the amount of transmission between the nodes, while by reducing it, a better control and estimation performance is achieved, since the plant state and the observation error are finally bounded in a smaller region. This trade-off, typical in an event-based framework, will be shown up in the following section.

9.6 Application Example

The performance of the proposed distributed control scheme has been experimentally tested in a water level control system. Next, the experimental setup and its model are described providing all the considerations related to the distributed scheme.

9.6.1 Plant Description

The experiments were performed on the 33–041 Coupled Tanks System of Feedback Instruments, see [114] (see Fig. 9.2). This plant, a variant of the quadruple-tank process originally proposed in [124], is a model of a fragment of a chemical plant. It is composed of four tanks, each one equipped with a pressure sensor to measure the water level. The couplings between the tanks can be changed using seven manual valves to modify the dynamics of the system. Water is delivered to the tanks by two independently controlled, submerged pumps. Drain flow rates can be modified using suitable orifice caps. Notation related to the plant is given in Table 9.1.

The coupled tanks are controlled using Simulink and an Advanced PCI1711 Interface Card. For the experiments, the following configuration were chosen:

- Input water is delivered to the upper tanks. Pump 1 feeds tank 1 and pump 2 feeds tank 3.
- Tanks 1 and 3 are coupled by opening the corresponding valve.

Fig. 9.2 Plant of four coupled tanks

Table 9.1 Notation related to the plant

	Description
h_i	Water level of tank i
v_i	Voltage of pump i
h_i^0	Reference level of tank i
v_i^0	Reference voltage of pump i
Δh_i	Increment of h_i with respect to h_i^0
Δv_i	Increment of v_i with respect to v_i^0
s	System output
r	Reference to be tracked
Δh_r	Reference level with respect to h^0
Δv_r	Reference voltage with respect to v^0

Although the plant is a compact educational platform, it realistically represents all the relevant elements of a real network-controlled physically distributed plant; for example, large-scale chemical plants, where coupled processes (represented by the coupled tanks), can be located hundred of meters away from each other.

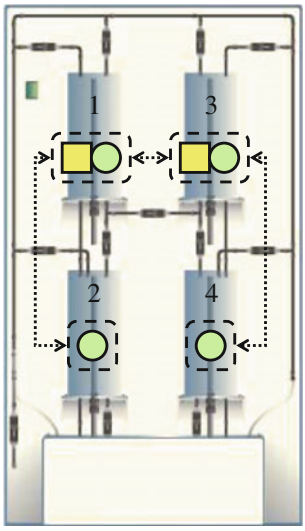

Fig. 9.3 Distributed control scheme with four agents. Agents *1* and *3* are observers&controllers; agents *2* and *4* are observers. The *dotted lines* represent the communication links

The distributed control scheme proposed in this work is applied considering a network with four agents, two of them being observers and the other two observers&controllers (see Fig. 9.3). Each agent has been tagged from 1 to 4 according to the number of the tank whose level measures. Agent 1 (respectively 3) measures the water level in tank 1 (3) and applies the control signal to pump 1 (2). Agents 2 and 4 measure the level in the tanks 2 and 4, respectively. The communication topology is: $2 \Leftrightarrow 1 \Leftrightarrow 3 \Leftrightarrow 4$.

The objective of the experiments is twofold. First, all four states of the plant must be estimated from every agent. Second, the water level of the two lower tanks is to be controlled. Notice that with this configuration, the agents applying the control signals (agents 1 and 3) do not have direct measurement of the variables being controlled (levels in tanks 2 and 4).

9.6.2 Plant Modeling

The coupled tanks system admits the following nonlinear model:

$$\begin{aligned}
\frac{dh_1(t)}{dt} &= -\frac{a_1}{S}\sqrt{2gh_1(t)} + \eta v_1(t) - \frac{a_{13}}{S}\sqrt{2g(h_1(t) - h_3(t))},\\
\frac{dh_2(t)}{dt} &= \frac{a_1}{S}\sqrt{2gh_1(t)} - \frac{a_2}{S}\sqrt{2gh_2(t)},\\
\frac{dh_3(t)}{dt} &= -\frac{a_3}{S}\sqrt{2gh_3(t)} + \eta v_2(t) + \frac{a_{13}}{S}\sqrt{2g(h_1(t) - h_3(t))},\\
\frac{dh_4(t)}{dt} &= \frac{a_3}{S}\sqrt{2gh_3(t)} - \frac{a_4}{S}\sqrt{2gh_4(t)},
\end{aligned}$$

where $h_i(t)$ $(i = 1, \ldots, 4)$ denote the water level in the corresponding tank and v_i $(i = 1, 2)$ are voltages applied to the pumps. a_i $(i = 1, \ldots, 4)$ are the outlet area of the tanks, a_{13} is the outlet area between tanks 1 and 3; η is a constant relating the control voltage with the water flow from the pump, S is the cross-sectional area of the tanks, and g is the gravitational constant.

This system is linearized around the equilibrium point given by h_i^0 and u_i^0, yielding

$$\Delta\dot{h}(t) = A\Delta h(t) + B\Delta v(t), \tag{9.33}$$

where $\Delta h(t) = \left[h_1(t) - h_1^0 \ \ldots \ h_4(t) - h_4^0\right]^T$ and $\Delta v(t) = \left[v_1(t) - v_1^0 \ v_2(t) - v_2^0\right]^T$. Matrices A and B in (9.34) have been obtained by using a Taylor expansion of the nonlinear equations of the model around the equilibrium point:

$$A = \begin{bmatrix} -\frac{a_1 g}{S\sqrt{2gh_1^0}} - \frac{a_{13} g}{S\sqrt{2g(h_1^0 - h_3^0)}} & 0 & \frac{a_{13} g}{S\sqrt{2g(h_1^0 - h_3^0)}} & 0 \\ \frac{a_1 g}{S\sqrt{2gh_1^0}} & -\frac{a_2 g}{S\sqrt{2gh_2^0}} & 0 & 0 \\ \frac{a_{13} g}{S\sqrt{2g(h_1^0 - h_3^0)}} & 0 & -\frac{a_3 g}{S\sqrt{2gh_3^0}} - \frac{a_{13} g}{S\sqrt{2g(h_1^0 - h_3^0)}} & 0 \\ 0 & 0 & \frac{a_3 g}{S\sqrt{2gh_3^0}} & -\frac{a_4 g}{S\sqrt{2gh_4^0}} \end{bmatrix}, \quad B = \begin{bmatrix} \eta & 0 \\ 0 & 0 \\ 0 & \eta \\ 0 & 0 \end{bmatrix} \tag{9.34}$$

Discretizing this continuous model with sampling time T, yields

$$\Delta h(k+1) = A_D \Delta h(k) + B_D \Delta v(k), \tag{9.35}$$

where $\Delta h(k) = \left[h_1(k) - h_1^0 \ \ldots \ h_4(k) - h_4^0\right]^T$ and $\Delta v(k) = \left[v_1(k) - v_1^0 \ v_2(k) - v_2^0\right]^T$. Matrices A_D and B_D are the discrete counterpart of A and B.

The control objective is not only to stabilize the plant around the linearization point, but also to track references. To do so, the system's output is set as $s \triangleq C_r \Delta h$, where C_r is a matrix that selects the water level of tanks 2 and 4. The references are given by the vector r. In the equilibrium points, it should be verified $s \simeq r$ and $\Delta h(k+1) \simeq \Delta h(k) \simeq \Delta h_r(k)$. To perform the tracking task, the incremental equilibrium points $(\Delta h_r, \Delta v_r)$ associated with reference r are found as follows.

$$\begin{aligned} \Delta h_r(k) &= A_D \Delta h_r + B_D \Delta v_r, \\ r &= C_z \Delta h_r. \end{aligned}$$

Rewriting in blocks the equation above yields

$$\begin{bmatrix} 0 \\ r \end{bmatrix} = \begin{bmatrix} A_D - I & B_D \\ C_z & 0 \end{bmatrix} \begin{bmatrix} \Delta h_r \\ \Delta v_r \end{bmatrix},$$

so that the incremental equilibrium point associated with r can be obtained as

$$\begin{bmatrix} \Delta h_r \\ \Delta v_r \end{bmatrix} = \begin{bmatrix} A_D - I & B_D \\ C_z & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ r \end{bmatrix}.$$

It is assumed that the references are reachable by the system, that is, the inverse above does exist. Finally, to track references, the following system must be stabilized.

$$x(k+1) = A_D x(k) + B_D u(k), \tag{9.36}$$

where $x(k) \triangleq \Delta h(k) - \Delta h_r$ and $u(k) \triangleq \Delta v(k) - \Delta v_r$. Notice that this system has the same structure that the one described in (9.3).

9.6.3 Experimental Results

The proposed distributed control method has been tested with the model parameters identified as shown in Table 9.2.

A comparison of both periodic and event-based sampling possibilities was performed. First, a periodic distributed control and estimation system was designed taking weighting matrices in (9.7) as

$$Q_x = diag(0.1, 100, 0.1, 100),$$
$$Q_1 = diag(1, 10, 1, 0.1),$$

Table 9.2 Parameters of the plant

	Value	Unit	Description
h_i	0–25	cm	Water level of tank i
v_i	0–5	V	Voltage level of pump i
S	0.01389	m^2	Cross-sectional area
a_i	50.265e−6	m^2	Outlet area of tank i
a_{13}	50.265e−6	m^2	Outlet area between tanks 1 and 3
η	2.2e-3	$\frac{m^3}{V \cdot s}$	Constant relating voltage and flow
h_1^0	9.8	cm	Nominal water level of tank 1
h_2^0	17.4	cm	Nominal water level of tank 2
h_3^0	7.5	cm	Nominal water level of tank 3
h_4^0	13.6	cm	Nominal water level of tank 4
v_1^0	3.3	V	Nominal voltage applied to pump 1
v_2^0	2.6	V	Nominal voltage applied to pump 2
T	1	s	Sampling period

$$
\begin{aligned}
Q_2 &= 10^{-2} \cdot diag(1, 1, 1, 1), \\
Q_3 &= diag(1, 0.1, 1, 10), \\
Q_4 &= 10^{-2} \cdot diag(1, 1, 1, 1), \\
R &= 10^{-6} \cdot diag(1, 1).
\end{aligned}
$$

The bijective search and the cone complementary algorithm took 160 s running over a computer with a 2.53 GHz processor and 4 GB of RAM. May the reader recall that once the controllers are found, the method is implemented distributedly.

Figure 9.4 depicts the performance of the distributed control scheme proposed in this chapter. A satisfactory tracking of references in tanks 2 and 4 can be observed. The effects of the chosen weighting matrices become apparent in the overshooting in tanks 1 and 3, which purpose is to achieve a fast reference tracking in the lower tanks. The average rise time of the system response is around 100 s, about one third of the natural time constant of the open-loop system.

Figure 9.5 compares the performance of the proposed distributed control scheme with respect to a centralized implementation of the same controllers. The information provided by the sensors (the four plant states) is gathered in a central unit, so no estimation is required. The control is thus applied as that of a centralized controller with full plant information, using the same controller gains obtained from the distributed design. As it can be observed, the control performances of the centralized and distributed implementations are very similar. This implies that the consensus-based, distributed monitoring of the plant state exhibits a good performance, so the distributed implementation can be applied without loosing control performance. Notice that the offset between references and output is due to the mismatches between the nonlinear plant and the linearized model. This offset can be corrected using an offset cancelation technique.

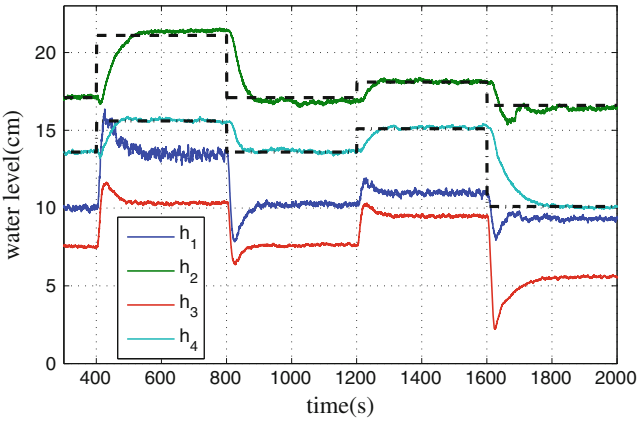

Fig. 9.4 Tracking performance with periodic sampling. The references are shown in *dashed lines*

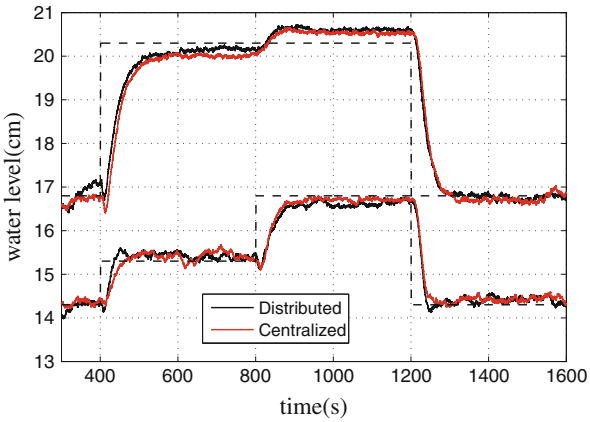

Fig. 9.5 Comparison between distributed and centralized control implementation. The water levels depicted correspond to those of the lower tanks

The water levels are shown in Fig. 9.6, together with the estimations of these level by Agent 1. It is worth pointing out that Agent 1 has no direct access to level measurements in tanks 2 and 4, but it estimates these magnitudes from the information sent by its neighbors.

Next, the effect of the weighting factors in (9.7) is shown. Two experiments were conducted, both with the same weighting for matrices Q_x and Q_i $(i = 1, \ldots, 4)$ as in the previous experiment, and different values for R: $R_1 = 10^{-6} \cdot diag(1, 1)$ to obtain a fast references tracking, and $R_2 = 10^2 \cdot diag(1, 1)$ to weight control effort in the cost functional. Figure 9.7 shows the evolution of the water levels and the control actions. As expected, a tighter tracking performance is observed for the experiment with R_1, at the cost of more aggressive control signals. This result shows how control performance can be traded off with respect to control actions by appropriately tuning the weighting gains.

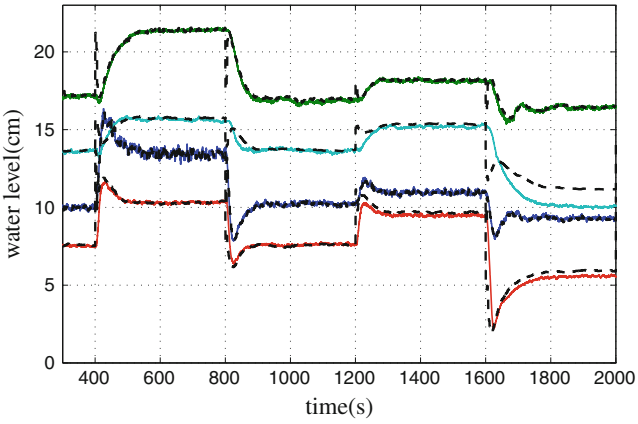

Fig. 9.6 Observation performance of Agent 1 with periodic sampling. The estimates are depicted in *dashed lines*

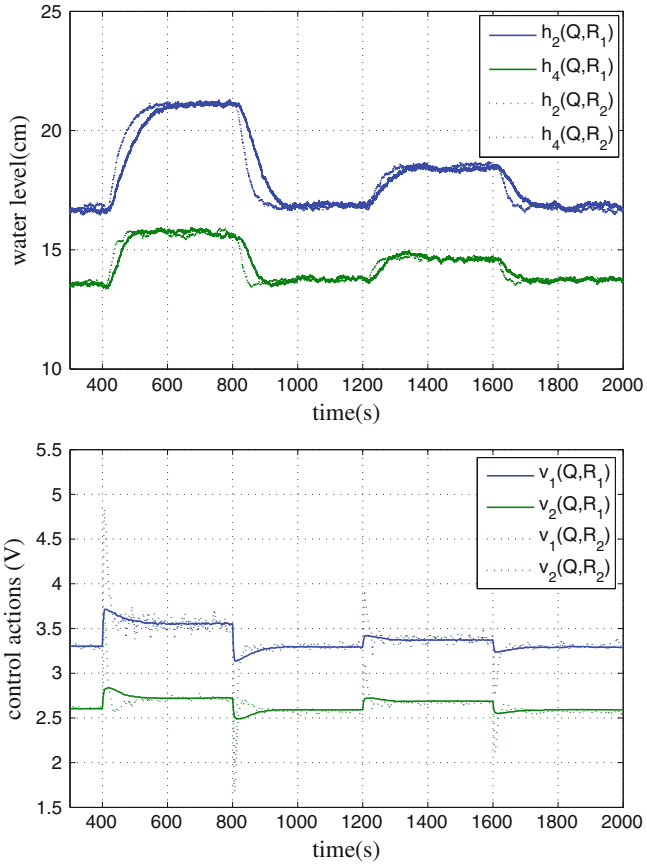

Fig. 9.7 Tracking performance and control signals with periodic sampling and different weighting matrices

The experiment shown in Fig. 9.8 was designed to show the decoupling capabilities of the proposed control strategy. Tank 2 was set to track references whereas the reference for tank 4 was kept constant. To modify the level of tank 2, tank 1 must be filled or emptied, and due to the coupling valve, tank 3 varies its level also affecting the level in tank 4. The distributed controllers achieved a remarkable decoupling of the closed-loop dynamics as can be observed from these experiments.

Lastly, the event-driven control scheme performance is examined. In this case, the same weighting matrices of the first experiment were chosen, and different thresholds to trigger the events were employed: $\delta_\omega = 0.1$, $\delta_\omega = 0.3$ and $\delta_\omega = 0.6$. The results of these experiments are shown in Fig. 9.9, where the tracking performance in tank 2 is shown. It is observed, as expected, that the larger the event threshold (larger δ_ω) is set, the poorer the tracking performance becomes. Figure 9.10 shows the observed states for node 1 with $\delta_\omega = 0.6$.

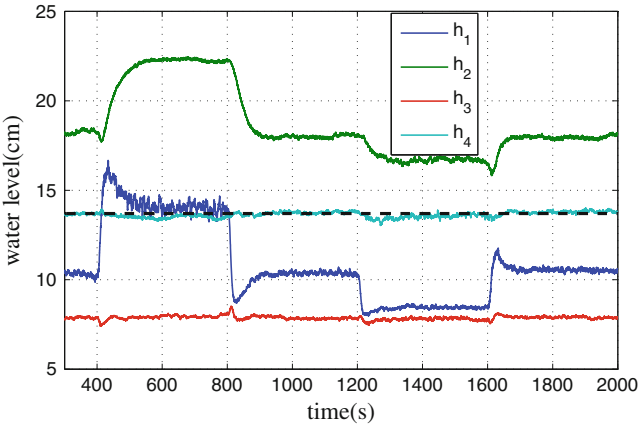

Fig. 9.8 Control decoupling: a change of reference is set for tank 2 while reference of tank 4 remains constant

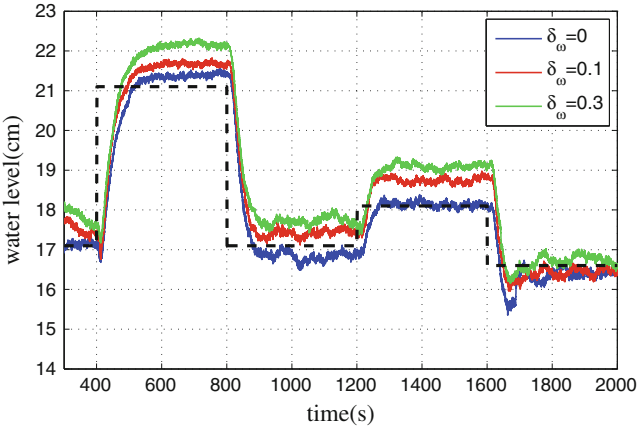

Fig. 9.9 Tracking of references in tank 2 for different values of δ_ω

The performance degradation due to the event-based communication scheme is now apparent when compared to the results with periodic communications. On the other hand, the event-based policy significantly reduces the number of required transmissions, as it is depicted in Fig. 9.11, which shows the ratio of transmitted packets with respect to the periodic case.

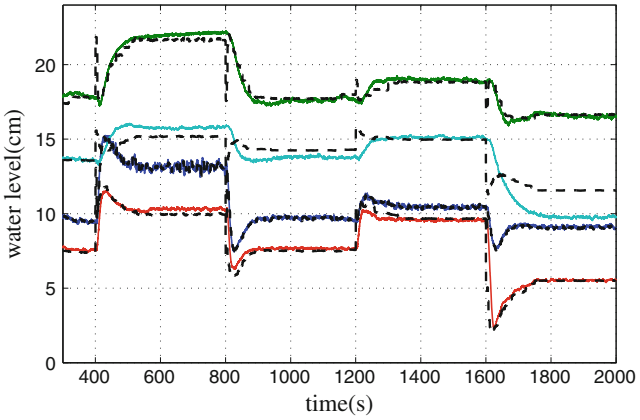

Fig. 9.10 Estimation performance of Agent 1 with $\delta_\omega = 0.6$. The estimates are shown in *dashed lines*

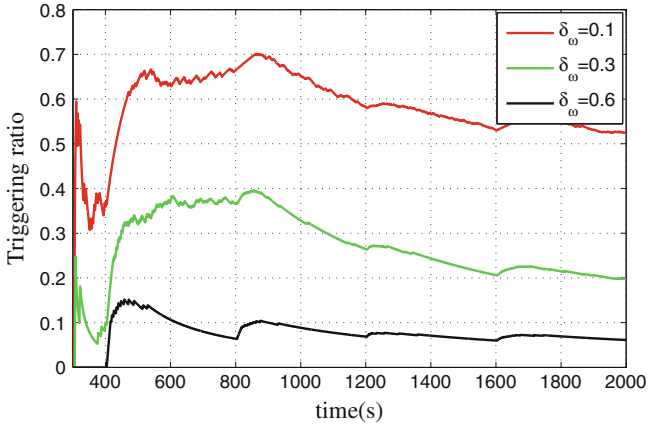

Fig. 9.11 Ratio of packets sent wrt. periodic communication

9.7 Summary

This chapter proposes a cost-guaranteed distributed estimation and control scheme for networked control systems, where the sensing and control capabilities are shared by a number of agents. The agents are assumed to collect partial information of the evolution of the plant states and have access, in general, to a subset of the plant control channels. The work proposes a fully distributed control and estimation scheme so that the collective behavior of all agents controls the system. The technique is of application to large-scale systems where centralized or classical decentralized schemes not advisable.

Both, a periodic sampling and event-based communication policy have been discussed. Using a four-tank level control system, experiments were conducted to show that in practice, little performance degradation is, in general, observed for the event-based compared to the periodic-based scheme, while the number of packets transmitted is drastically reduced. The adopted cost-guaranteed approach has shown also very convenient in practical applications as allows the trade-off between control effort, performance degradation, and average packet transmission rates.

As future work, this technique will possibly be extended to consider network-induced communication problems, as time-delays or packet dropouts. Furthermore, it will be studied the implementation of asynchronous, self-trigger communication policies, and the integration of fault-detection and fault-tolerant systems. Another promising line of research includes a technique to carry out the observers/controllers design in a distributed way.

Chapter 10
Distributed Event-Based Control for Non-reliable Networks

María Guinaldo, Daniel Lehmann and José Sánchez

10.1 Introduction

Even though event-based control has been shown to reduce the communication to face the problem of reduced bandwidth, network delays and packet losses cannot be avoided [23]. However, up to now, only a reduced number of works have considered the effect of these issues on event-based control and just a few works have addressed a decentralized implementation to cope with them. Early works [34, 212] study simple stochastic systems and investigate the event-based control performance in dependence upon the medium access mechanism applied. In [78, 142], delays are compensated by model-based event-triggered approaches and the measurement of the delay. However, these schemes are difficult to implement in a distributed scenario since measuring transmission delays for any transmission between two nodes requires clock synchronization in the entire network.

In distributed control, one paper that takes into account delays and packet losses is [259]. As stated in the cited paper, one problem that might present trigger functions of the form $\|\varepsilon_i(t)\| \leq \sigma_i \|x_i(t)\|$ is that for unreliable networks a lower bound for the broadcasting period cannot be guaranteed when the system approaches the origin, being this the main drawback of the cited work. Additionally, an analytical expression for the minimum inter-event time is provided in [90] under the assumption of perfect decoupling of the subsystems and for deadband distributed controllers.

M. Guinaldo (✉) · J. Sánchez
Dpto. de Informática y Automática, Escuela Técnica Superior de Informática,
UNED, Madrid, Spain
e-mail: mguinaldo@dia.uned.es

J. Sánchez
e-mail: jsanchez@dia.uned.es

D. Lehmann
School of Electrical Engineering, Royal Institute of Technology (KTH),
Stockholm, Sweden
e-mail: dlehmann@kth.se

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_10

This chapter presents possible solutions to deal with the problem of unreliable networks in distributed control systems where the communication is event-triggered. Section 10.2 discusses the problems behind delays and packet losses in distributed networks from the analytical point of view. Two network protocols are described in Sect. 10.3 to control the flow of information between nodes. The first protocol is based on the work of [259] and requires the synchronous update of all the nodes in a given neighborhood when the transmission of data is subject to delay and packet losses. This limitation is solved by the second protocol.

Moreover, under certain requirements, upper bounds on the allowable delay and the maximum number of consecutive packet losses can be derived for different triggering mechanisms in Sects. 10.4 and 10.5. We additionally prove that the system can asymptotically converge to the equilibria while the Zeno behavior is excluded with exponential trigger functions, which also provide larger upper bounds on the delay than deadband control. An example to illustrate the analytical results is given in Sect. 10.6 for the same setup used in Chap. 7.

10.2 Problem Statement: Ideal Versus Non-ideal Networks

Consider the interconnected linear system described in Chap. 7 (7.12)

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + \sum_{j \in \mathcal{N}_i} H_{ij} x_j(t), \quad \forall i = 1, \ldots, N_a \tag{10.1}$$

and the control law (7.13)

$$u_i(t) = K_i x_{b,i}(t) + \sum_{j \in \mathcal{N}_i} L_{ij} x_{b,j}(t), \quad \forall i = 1, \ldots, N_a. \tag{10.2}$$

In an ideal network scenario, the detection of an event, the broadcast of the corresponding state $x_{b,i}$, and its reception in all neighboring nodes are assumed to be simultaneous.

However, in a non-reliable network, a broadcast state may be received in the neighbors with delay, or even more, not be received at all. This may yield *state inconsistency*. In this context, this concept was introduced for the first time by [259], and it can be defined as follows.

Definition 10.1 A distributed event-based control design preserves *state consistency* if any broadcast state is updated synchronously in each neighboring agent.

Example 10.1 An example of state inconsistency is presented in Fig. 10.1. Assume that the piecewise signal $x_{b,1}$ is updated at event times denoted by t_k^1, $k \in \mathbb{N}$, and sent through the network to update the copy of the signal $x_{b,12}$ accordingly. We denote by τ_k^{12}, $k \in \mathbb{N}$, the communication delay experienced in the broadcast.

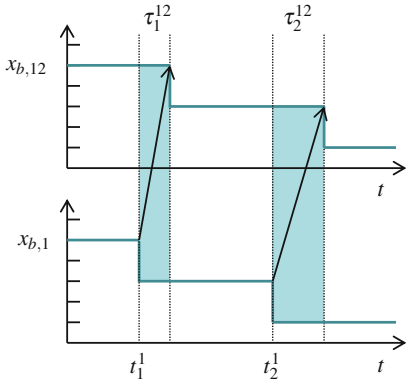

Fig. 10.1 Example of state inconsistency of the signal $x_{b,1}$ and its copy $x_{b,12}$ in other node of the network

If the transmission is not subject to delay, both signals $x_{b,1}$ and $x_{b,12}$ are identical. However, this is not the situation in the example of Fig. 10.1. In the time intervals $[t_1^1, t_1^1 + \tau_1^{12})$ and $[t_2^1, t_2^1 + \tau_2^{12})$ both signals are not equal. Hence, there is a state inconsistency since $x_{b,1}(t) \neq x_{b,12}(t), \forall t \in [t_1^1, t_1^1 + \tau_1^{12}) \cup [t_2^1, t_2^1 + \tau_2^{12})$.

The previous example illustrates the need of communication protocols to avoid state inconsistencies or to deal with them. In this chapter, two different protocols are proposed. The first one preserves the state consistency by the transmission of additional signals to synchronize the nodes in the neighborhood. This constraint is relaxed by the second protocol which allows the neighboring agents to use different versions of the broadcast states.

10.3 Transmission Protocol

Before describing the protocols, let us first introduce some notation.

Definition 10.2 We denote by τ_k^{ij} the delay in the transmission of the state $x_i(t_k^i)$ of agent i to its neighbor j, $j \in \mathcal{N}_i$, at time t_k^i, and by τ_k^i

$$\tau_k^i = \max\{\tau_k^{ij},\, j \in \mathcal{N}_i\}.$$

Definition 10.3 We denote by $n_{p,k}^{ij}$ the number of successive packet losses in the transmission of the state $x_i(t_k^i)$ of agent i to its neighbor j, $j \in \mathcal{N}_i$, at time t_k^i, and by $n_{p,k}^i$ the maximum of $n_{p,k}^{ij}$ for all $j \in \mathcal{N}_i$.

We now introduce the basic assumption that imposes constraints on delays and the number of consecutive packet dropouts. In practice, several consecutive packet losses introduce additional delays.

Assumption 10.1 The maximum delay and the number of successive packet dropouts which occur in the transmission of information from the subsystem i to its neighbors $j \in \mathcal{N}_i$, denoted by τ^i_{max} and n^i_p, respectively, are such that no event is generated before all the neighbors have successfully received the broadcast state $x_{b,i}$.

The second important consideration is that a packet is treated as lost if the sender i does not get an acknowledgment signal (ACK) from the receiver node j before a *waiting time* denoted by T^i_W. How to determine T^i_W is analyzed later on, but it seems logical to set this value larger than the maximum delay.

If agent i has not received an acknowledgment of the reception of all the neighbors after the waiting time T^i_W, we propose two alternatives that denoted by *Wait for All* (WfA) and *Update when Receive* (UwR).

10.3.1 WfA Protocol

The WfA works as follows. Consider the situation described above (no ACK signal is received after T^i_W). Then, the state at $t^i_k + T^i_W$ is broadcast again to all the neighbors. If after waiting T^i_W an ACK is not received from all $j \in \mathcal{N}_i$, the retransmission takes place again, and so on. This process can occur at most $n^i_p + 1$ times. Once all the neighbors have successfully received the data, agent i sends a *permission signal* (PERM) so that the previously transmitted data can be used to update the control law (10.2). Both signals ACK and PERM are assumed to be delivered with a delay approximated by zero over a reliable channel.

A very similar protocol is presented in [259]. As stated there, the reason to use a PERM signal and to retransmit the state to all the neighbors instead of only retransmitting to those from which an ACK signal has not been received is to preserve the *state consistency* (see Definition 10.1). Since the broadcast data is not valid until a PERM signal is received from agent i, all the neighboring agents update the value at the same time and therefore, the value of the error ε_i is the same in all nodes. This allows to define stack vectors for the state $x(t)$ and the error signal $\varepsilon(t)$ so that the stability of the overall system can be studied as in the ideal network case.

10.3.2 UwR Protocol

The previous protocol simplifies the analysis but it has some drawbacks. First, all nodes in the neighborhood have to wait for the slower connection (longer delay) to process the received data. Second, the WfA protocol may involve unnecessary transmission, since if an agent did not receive the measurement, the broadcast would take place again with an updated measurement for all the neighbors. Finally, the ACK signal is vastly used in network protocols to guarantee reliability of packet transfers,

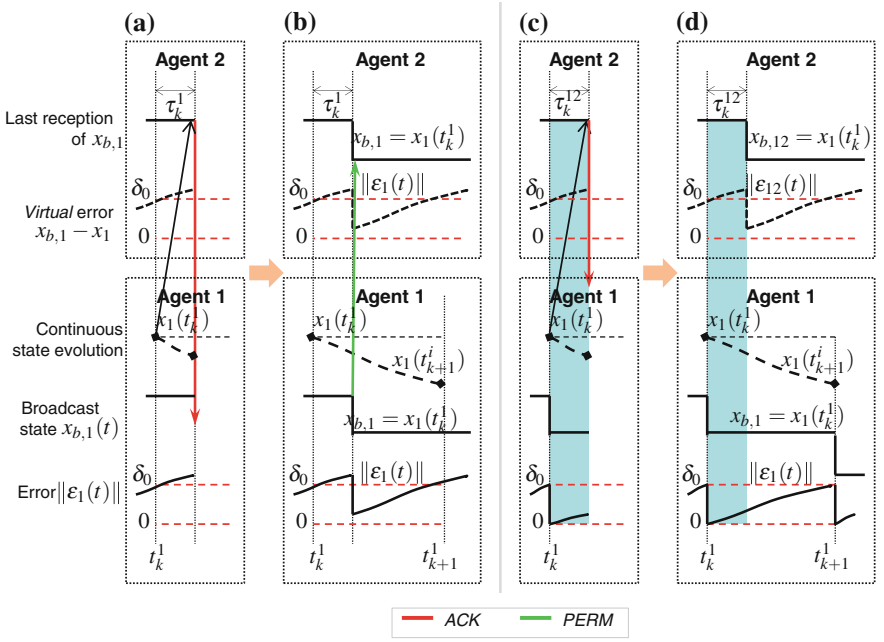

Fig. 10.2 Update mechanism of WfA (**a**, **b**) and UwR (**c**, **d**) protocols

but the PERM demands a more involved communication protocol. In order to overcome these drawbacks, in the new protocol agent i waits T_W^i to get acknowledgments from the neighbors. To those agents $j \in \mathcal{N}_i$ from which it did not receive the ACK signal, it sends the state $x_i(t_k^i + T_W^i)$ at time $t_k^i + T_W^i$. Agent i may transmit before the next event at most $n_p^i + 1$ times. According to Assumption 10.1, the number of consecutive packet losses and the network delay are upper bounded for each node i. Then, all neighbors have successfully received the state of agent i before the next event time t_{k+1}^i, that is, there exist instances of time t_k^{ij} such that $t_k^i \leq t_k^{ij} < t_{k+1}^i$.

Example 10.2 A simple example is given in Fig. 10.2 to clarify the difference between both protocols. A system with two agents is depicted and a constant threshold δ_0 for the trigger function (deadband control) is considered for simplicity. Assume that Agent 1 detects an event at time t_k^1 and broadcasts its state $x_1(t_k^1)$ to its neighbor Agent 2. The transmission is delayed by τ_k^1 and Agent 2 sends then the ACK signal. In the scenario of WfA protocol, once the ACK signal is received by Agent 1 (see Fig. 10.2a), the PERM signal is sent (both signals are assumed to be sent and received instantaneously), and both agents update the broadcast state in the control law at the same time $t_k^1 + \tau_k^1$ (see Fig. 10.2b). Thus, $x_{b,1}$ takes the same value at any time in both agents and, hence, $\varepsilon_1(t)$ is the same in the dynamics of Agent 1 and 2.

For the UwR protocol, the update in Agent 1 is applied immediately at time t_k^1 (see Fig. 10.2c), whereas the receiver updates the state information at time $t_k^1 + \tau_k^{12}$

(τ_k^1 and τ_k^{12} are the same), as illustrated in Fig. 10.2d. Thus, in the interval $[t_k^1, t_k^1+\tau_k^{12})$ the broadcast state $x_{b,1}$ has different values in the two nodes and consequently the error ε_1 considered in Agent 1 differs from the error affecting the dynamics of Agent 2.

This chapter refers to numbered agents as "Agent 1", "Agent 2", etc. Note that Agent 2 does not monitor ε_1 since it only knows the state of Agent 1 at event times. It is drawn in the figure to clarify the difference between the two protocols.

The performance of both protocols is analyzed next. We first assume that perfect decoupling ($\Delta_{ij} = 0$) can be achieved, since the analysis is simplified and moreover, upper bounds on the delay and packet losses can be derived for each agent, giving less conservative results. The results for the general case when the matching condition does not hold are derived afterwards. For simplicity, Assumption 7.1 applies (diagonalization of $A_{K,i}$).

Two cases are analyzed independently. First, we assume that perfect decoupling ($\Delta_{ij} = 0$) can be achieved, since the analysis is simplified and upper bounds on the delay and the number of consecutive of packet losses can be derived for each agent, giving less conservative results. These results are generalized afterwards. Regarding the design of the trigger functions, we first consider constant bounds for the error (deadband control) that provide analytical expressions for the delay and maximum number of consecutive packet losses. Later on, in this chapter, the focus lies on exponential bounds. In this case, a positive upper bound on the delay and consecutive number of packet losses can also be derived. The obtained expression can be solved for some given parameters to provide a numerical solution.

10.4 Performance Analysis for Perfect Decoupling

10.4.1 Properties of Deadband Control Using WfA Protocol

Let us consider trigger functions (7.24)

$$F_{e,i}(\varepsilon_i(t)) = \|\varepsilon_i(t)\| - \delta,\ \delta > 0. \tag{10.3}$$

Let us first assume that the communication can only experience delays but no packet dropouts.

10.4.1.1 Communication with Delays

Proposition 10.1 *Let us consider trigger functions of the form (10.3) and the WfA protocol. If Assumption 10.1 holds, the error of any subsystem i is upper bounded by $\|\varepsilon_i(t)\| < 2\delta$.*

Proof Assume that the last event occurred at time t_k^i and that the maximum transmission delay to its neighbors is τ_k^i. It follows that

$$\| \int_{t_k^i}^{t_k^i+\tau_k^i} \dot{\varepsilon}_i(s)ds \| = \|\varepsilon_i(t_k^i + \tau_k^i) - \varepsilon_i(t_k^i)\| < \delta, \tag{10.4}$$

has to be satisfied (see (10.3)) because no event is generated in the time interval $[t_k^i, t_{k+1}^i)$ from Assumption 10.1. Since an event has occurred at time t_k^i, $\|\varepsilon_i(t_k^i)\| = \delta$ holds and, from (10.4) it holds $\|\varepsilon_i(t_k^i + \tau_k^i)\| < 2\delta$, which is valid for any time t.

The previous result allows stating the next theorem. An analytical upper bound on the delay is derived, which is also the lower bound on the inter-event time, while the convergence of $x_i(t)$ to a region around the equilibrium is guaranteed.

Theorem 10.1 *If the network delay is upper bounded by*

$$\tau_{max}^i = \frac{\delta}{\|A_{K,i}\|\kappa(V_i)\|x_i(0)\| + \mu_i \left(1 + \frac{\|A_{K,i}\|\kappa(V_i)}{|\alpha_{max}(A_{K,i})|}\right) 2\delta}, \tag{10.5}$$

where $\mu_i = \|B_i K_i\| + \sum_{j \in \mathcal{N}_i} \|B_i L_{ij}\|$, *and* $\alpha_{max}(A_{K,i})$ *and* $\kappa(V_i)$ *are defined according to (7.2) and (7.1), respectively, then any broadcast state* $x_{b,i}$ *of any subsystem* $i \in 1, \ldots, N_a$ *is successfully received by the neighbors* $j \in \mathcal{N}_i$ *before a new event occurs, and the inter-event time is lower bounded* $t_{k+1}^i - t_k^i \geq \tau_{max}^i$.

Moreover, for all initial conditions $x_i(0)$ *and* $t > 0$ *it holds*

$$\|x_i(t)\| \leq \kappa(V_i) \left(\frac{\mu_i 2\delta}{|\alpha_{max}(A_{K,i})|} + e^{-|\alpha_{max}(A_{K,i})|t} \left(\|x_i(0)\| - \frac{\mu_i 2\delta}{|\alpha_{max}(A_{K,i})|} \right) \right). \tag{10.6}$$

Proof The proof can be found in Appendix A.

10.4.1.2 Communication with Delays and Packet Losses

The previous analysis was focused on the effect of delays exclusively. However, in practice, delays and packet losses may occur simultaneously. The next corollary extends the results of Theorem 10.1 when both phenomena are considered. The proof can be found in Appendix A, but the main idea behind it is that n_p consecutive packet losses can be seen as a delay if a packet is discarded when no acknowledgment is received after some waiting time. Thus, setting a value for n_p and knowing an upper bound for the total delay (packet losses plus transmission delay), an upper bound for the transmission delay can be computed.

Corollary 10.1 *Assume that the maximum number of consecutive packet losses is upper bounded by n_p^i, and the transmission delay τ_k^i is upper bounded by*

$$\bar{\tau}^i = \frac{\tau_{max}^i}{n_p^i + 1}, \tag{10.7}$$

where τ_{max}^i is given by (10.5). Assume also that the waiting time T_W^i of the WfA protocol is set to $\bar{\tau}^i$. Then, there is a successful broadcast before the occurrence of a new event and the state of each agent i is bounded by (10.6).

Proof The proof can be found in Appendix A.

Remark 10.1 Note that the maximum number of consecutive packet dropouts n_p^i and the maximum tolerable delay $\bar{\tau}^i$ are correlated. A large value of n_p^i implies small values of $\bar{\tau}^i$ and vice versa. This way, there is a trade-off between both parameters.

10.4.2 Properties of Deadband Control Using UwR Protocol

In this section, we study the UwR protocol, where the main issue is to keep track of the different versions of the broadcast states. First, some definitions are introduced to adapt the previous analysis to this new scenario.

Definition 10.4 We denote $\{t_k^{ij}\}$ the set of successful broadcasting times from agent i to agents $j \in \mathcal{N}_i$, and the error

$$\varepsilon_{ij}(t) = x_{b,ij}(t_k^{ij}) - x_i(t), \quad t \in [t_k^{ij}, t_{k+1}^{ij}), \tag{10.8}$$

where $x_{b,ij}(t_k^{ij})$ is the last successful broadcast state from agent i to agent j, $j \in \mathcal{N}_i$.

With this definition, the dynamics of agent i is given by

$$\dot{x}_i(t) = A_{K,i}x_i(t) + B_i K_i \varepsilon_i(t) + \sum_{j \in \mathcal{N}_i} B_i L_{ij} \varepsilon_{ji}(t) \tag{10.9}$$

with $\varepsilon_i(t)$ the agent i's version of the error. We assume that agent i automatically updates its broadcast state in its control law and does not need to wait to receive an acknowledgment of successful receptions from its neighbors. With these prerequisites the following theorem is obtained.

Theorem 10.2 *If the network delay is upper bounded by*

$$\bar{\tau}^i = \frac{\tau_{max}^i}{n_p^i + 1}, \tag{10.10}$$

where n_p^i is the maximum number of consecutive packet losses and

$$\tau_{max}^i = \frac{\delta}{\|A_{K,i}\|\kappa(V_i)\|x_i(0)\| + \bar{\mu}_i\left(1+\frac{\|A_{K,i}\|\kappa(V_i)}{|\alpha_{max}(A_{K,i})|}\right)2\delta}, \tag{10.11}$$

with $\bar{\mu}_i = \frac{1}{2}\|B_i K_i\| + \sum_{j\in N_i}\|B_i L_{ij}\|$, and the waiting time T_W^i of the UwR protocol is set to $\bar{\tau}^i$, then any broadcast state $x_{b,i}$ is successfully received by all the neighbors of the subsystem i before a new event occurs. Moreover, the local inter-event times $t_{k+1}^i - t_k^i$ are lower bounded by (10.11), and for any initial condition $x_i(0)$ and for any $t > 0$, it holds

$$\|x_i(t)\| \leq \kappa(V_i)\left(\frac{\bar{\mu}_i 2\delta}{|\alpha_{max}(A_{K,i})|} + e^{-|\alpha_{max}(A_{K,i})|t}\left(\|x_i(0)\| - \frac{\bar{\mu}_i 2\delta}{|\alpha_{max}(A_{K,i})|}\right)\right). \tag{10.12}$$

Proof The proof can be found in Appendix A.

Note that the delay bound for WfA and UwR protocols are different (see (10.5), (10.11)). Since $\bar{\mu}_i < \mu_i$, under the same initial conditions UwR allows larger delays. This difference lies on the fact that the upper bound of the local error ε_i in the agent i is different in the two protocols, since the node i waits for neighboring nodes in the WfA protocol, but it does not in the UwR protocol.

10.4.3 Pure Exponential Trigger Functions

Let us consider trigger functions (7.26) with $\delta_0 = 0$ and $\delta_1 > 0$ for simplicity:

$$F_{e,i}(t, \varepsilon_i(t)) = \|\varepsilon_i(t)\| - \delta_1 e^{-\beta t}, \quad \beta > 0. \tag{10.13}$$

The case $\delta_0, \delta_1 > 0$ is equivalent to having a constant threshold from the analytical point of view.

The motivation of trigger functions of the form (10.13) has been already discussed. Besides, in Chap. 7, it has been proved that the inter-event time is lower bounded if $\beta < |\alpha_{max}(A_K)|$ for perfect decoupling because $\|\Delta\| = 0$.

Hence, under Assumption 10.1, it seems reasonable that it is possible to derive an upper bound on the delay allowing less conservative results. We next briefly present the obtained results for WfA and UwR protocols. The proofs have been moved to Appendix A.

10.4.3.1 Performance of WfA Protocol

A result equivalent to Proposition 10.1 is derived.

Proposition 10.2 *Let us consider trigger functions of the form (10.13) and WfA protocol. If Assumption 10.1 holds, the error of any subsystem i is upper bounded by $\|\varepsilon_i(t)\| < \delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}$, where $\tau_{max} > 0$ is the maximum transmission delay in the system.*

Proof The proof can be found in Appendix A.

Note that the value of τ_{max} is unknown. Its existence is assumed, and the following theorem will prove it, giving the expression to compute it numerically.

Theorem 10.3 *Let $\beta < |\alpha_{max}(A_{K,i})|, \forall i = 1, \ldots, N_a$. If the network delay for any broadcast in the system (10.1) is upper bounded by*

$$\tau_{max} = \min\{\tau^i_{max},\ i = 1, \ldots, N_a\} \tag{10.14}$$

being τ^i_{max} the solution of

$$\left(\frac{k_{1,i}}{\delta_1} + \frac{k_{2,i}}{\delta_1}(1+e^{\beta\tau^i_{max}})\right)\tau^i_{max} = e^{-\beta\tau^i_{max}}, \tag{10.15}$$

and

$$k_{1,i} = \|A_{K,i}\|\kappa(V_i)\|x_i(0)\| \tag{10.16}$$

$$k_{2,i} = \left(\|A_{K,i}\|\kappa(V_i)\frac{1}{|\alpha_{max}(A_{K,i})| - \beta} + 1\right)\mu_i\delta_1, \tag{10.17}$$

then any broadcast state $x_{b,i}$ is successfully received by the neighbors $j \in \mathcal{N}_i$ before a new event occurs. Hence, the inter-event times are lower bounded $t^i_{k+1} - t^i_k \geq \tau_{max}$. Moreover, for all initial conditions $x_i(0)$ and $t > 0$ it holds

$$\|x_i(t)\| \leq \kappa(V_i)\left(\frac{\mu_i\delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta} + e^{-|\alpha_{max}(A_{K,i})|t}\left(\|x_i(0)\| - \frac{\mu_i\delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta}\right)\right). \tag{10.18}$$

Proof The proof can be found in Appendix A.

10.4.3.2 Performance of UwR Protocol

Under this protocol, the system dynamics is given by (10.9). Note that equivalently to the results for constant threshold, it holds that $\|\varepsilon_i(t)\| \leq \delta_1 e^{-\beta t}$ and $\|\varepsilon_{ij}(t)\| < \delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}$, where $\tau_{max} > 0$ is the upper bound on the delay derived in the next theorem.

Theorem 10.4 *Let $\beta < |\alpha_{max}(A_{K,i})|, \forall i = 1, \ldots, N_a$. If the network delay for any broadcast in the system (10.1) is upper bounded by*

$$\tau_{max} = \min\{\tau^i_{max},\ i = 1, \ldots, N_a\} \tag{10.19}$$

being τ^i_{max} *the solution of*

$$\left(\frac{k_{1,i}}{\delta_1} + \frac{k_{2,i}}{\delta_1} + \frac{k_{3,i}}{\delta_1}\left(1 + e^{\beta\tau^i_{max}}\right)\right)\tau^i_{max} = e^{-\beta\tau^i_{max}}, \tag{10.20}$$

and

$$k_{1,i} = \|A_{K,i}\|\kappa(V_i)\|x_i(0)\| \tag{10.21}$$

$$k_{2,i} = \|B_i K_i\|\left(1 + \frac{\kappa(V_i)\|A_{K,i}\|}{|\alpha_{max}(A_{K,i})| - \beta}\right)\delta_1 \tag{10.22}$$

$$k_{3,i} = \sum_{j\in\mathcal{N}_i}\|B_i L_{ij}\|\left(1 + \frac{\kappa(V_i)\|A_{K,i}\|}{|\alpha_{max}(A_{K,i})| - \beta}\right)\delta_1, \tag{10.23}$$

then any broadcast state $x_{b,i}$ *is successfully received by the neighbors* $j \in \mathcal{N}_i$ *before a new event occurs. Hence, the inter-event times are lower bounded* $t^i_{k+1} - t^i_k \geq \tau_{max}$. *Moreover, for all initial conditions* $x_i(0)$ *and* $t > 0$ *it holds*

$$\|x_i(t)\| \leq \kappa(V_i)\left(\frac{\bar{\mu}_i(\tau_{max})\delta_1 e^{-\beta t}}{|\alpha_{max}(A_{K,i})| - \beta} + e^{-|\alpha_{max}(A_{K,i})|t}\left(\|x_i(0)\| - \frac{\bar{\mu}_i(\tau_{max})\delta_1 e^{-\beta t}}{|\lambda_{max}(A_{K,i})| - \beta}\right)\right), \tag{10.24}$$

where $\bar{\mu}_i(\tau_{max}) = \|B_i K_i\| + \sum_{j\in\mathcal{N}_i}\|B_i L_{ij}\|(1 + e^{\beta\tau_{max}})$.

Proof The proof can be found in Appendix A.

Note that trigger functions (10.13) ensure the asymptotic convergence to the origin while guaranteeing a lower bound for the minimum inter-event time if the delay is below τ_{max}. This cannot be achieved if the triggering conditions are of the form $\|\varepsilon_i(t)\| \leq \sigma_i\|x_i(t)\|$, as pointed out in [259].

Example 10.3 Let us consider the chain of inverted pendulums of Fig. 7.3 and the control design described in Sect. 7.4.3.2. This example illustrates the influence of the parameters of the trigger functions on the upper bound on the delay.

Table 10.1 shows the most conservative computed delay among the subsystems in the network for different values of δ in trigger functions (10.3) and for WfA and

Table 10.1 Delays bounds (10.5) and (10.11) for different values of δ and for WfA and UwR protocols

δ (ms)	0.01	0.02	0.05	0.1
$(\tau^i_{max})_{\text{WfA}}$	0.347	0.613	1.140	1.624
$(\tau^i_{max})_{\text{UwR}}$	0.363	0.666	1.329	2.054

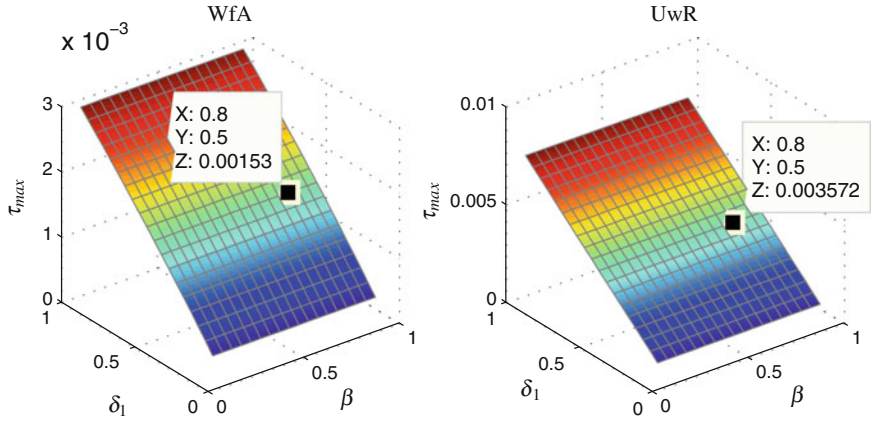

Fig. 10.3 Influence of δ_1 and β on the delay bound (10.14) (*left*) and (10.19) (*right*). The case $\delta_1 = 0.5$, $\beta = 0.8$ are 1.53 and 3.57 ms, respectively

UwR protocols. Note that the difference between the value of τ^i_{max} given by the two protocols increases with δ and that the UwR protocol always allows larger (less conservative) values on the delay.

Trigger functions (10.13) depend on two parameters δ_1 and β. Figure 10.3 depicts the bounds on the delay for a set of values of $\delta_1 \in [0.1, 1]$ and $\beta \in [0.1, 0.95]$ so that $\beta < |\alpha_{max}(A_K)| = 1$ is satisfied. The figure on the left shows the results for the WfA protocol (solution of (10.15)), and the one on the right for UwR (solution of (10.20)). Observe that τ_{max} is always greater when the transmissions are ruled by the UwR protocol.

If the solutions given for constant and exponential thresholds are compared, it can be noticed that the results are better in the second case. Furthermore, if we take the values of the parameters used in Sect. 7.4.3.2, deadband control ($\delta = 0.02$) gives values of τ_{max} around 0.6 ms, whereas for the exponential threshold ($\delta_1 = 0.5$ and $\beta = 0.8$), $\tau^\star$ is three (WfA) and five (UwR) times greater. It can be concluded that time-dependent trigger functions are a better choice because they provide asymptotic convergence and they also allow longer delays in the network.

10.5 Performance Analysis for Non-perfect Decoupling

If perfect decoupling cannot be assumed, the formulation changes. In order to illustrate it, let us consider an ideal network first. As it has been shown in Chap. 7, the dynamics of each agent can be rewritten in terms of the error as

$$\dot{x}_i(t) = A_{K,i}x_i(t) + B_i K_i \varepsilon_i(t) + \sum_{j \in \mathcal{N}_i} \left(\Delta_{ij} x_j(t) + B_i L_{ij} \varepsilon_j(t)\right).$$

Note that if $\Delta_{ij} \neq \mathbf{0}$, the dynamics of $\dot{x}_i(t)$ explicitly depends on $x_j(t)$, $\forall j \in \mathcal{N}_i$. Thus, $\|x_i(t)\|$ cannot be upper bounded if $\|x_j(t)\|$ is not. But at the same time, the dynamics of $x_j(t)$ depends on the neighborhood, and then there is a vicious circle.

Hence, one possible solution to this problem is to treat it as in Chap. 7, and rewrite the equations in terms of the overall system state and error as

$$\dot{x}(t) = (A_K + \Delta)x(t) + BK\varepsilon(t), \tag{10.25}$$

where all the matrices and vectors are defined in (7.16)–(7.21).

Let us assume that the communication is subjected to delays and packet losses. If the state consistency is preserved, for instance if WfA protocol is considered, (10.25) holds because the update of broadcast states is synchronized. Under certain assumptions on the error bound (e.g., Proposition 10.1), an equivalent analysis to the perfect decoupling case can be inferred for (10.25). However, if the state consistency cannot be guaranteed (UwR protocol), a different approach is required to handle the problem.

For the sake of simplicity, we next show the formulation which solves this situation for constant thresholds, but an equivalent procedure can be followed for other type of trigger functions.

10.5.1 Solving the State Inconsistency

Let us recall the definition of the error (10.8). If perfect decoupling does not hold and the transmissions over the network are governed by the UwR protocol, the dynamics of each subsystem is given by

$$\dot{x}_i(t) = A_{K,i}x_i(t) + \sum_{j\in\mathcal{N}_i} \Delta_{ij}x_j(t) + B_iK_i\varepsilon_i(t) + \sum_{j\in\mathcal{N}_i} B_iL_{ij}\varepsilon_{ji}(t). \tag{10.26}$$

Let us define the following set of matrices

$$\mathrm{M}_i = B_i\begin{pmatrix}L_{i1} & L_{i2} & \dots & L_{ii-1} & K_i & L_{ii+1} & \dots & L_{iN_a}\end{pmatrix}, \forall i = 1, \dots, N_a, \tag{10.27}$$

with $L_{ij} = 0$ if $j \notin \mathcal{N}_i$, and the matrix

$$\mathrm{M} = \begin{pmatrix}\mathrm{M}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathrm{M}_2 & \dots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathrm{M}_{N_a}\end{pmatrix}, \tag{10.28}$$

where $\mathbf{0}$ is a $n \times nN_a$ matrix whose elements are all zero.

Denote by

$$\overrightarrow{\varepsilon}_i^T = \left(\varepsilon_{1i}^T \ \varepsilon_{2i}^T \ \dots \ \varepsilon_{i-1,i}^T \ \varepsilon_i^T \ \varepsilon_{i+1,i}^T \ \dots \ \varepsilon_{N_a i}^T\right), \forall i = 1, \dots, N_a, \tag{10.29}$$

with $\varepsilon_{ji} = \mathbf{0}$ if $j \notin \mathcal{N}_i$, and

$$\overrightarrow{\varepsilon}^T = \left(\overrightarrow{\varepsilon}_1^T \ \dots \ \overrightarrow{\varepsilon}_{N_a}^T\right). \tag{10.30}$$

With these definitions, the dynamics of the overall system is

$$\dot{x}(t) = (A_K + \Delta)x(t) + \mathrm{M}\overrightarrow{\varepsilon}(t). \tag{10.31}$$

Proposition 10.3 *If Assumption 10.1 holds and trigger functions (10.3) and the UwR protocol are considered, the error (10.30) is bounded by*

$$\|\overrightarrow{\varepsilon}(t)\| \leq \delta \sqrt{N_a + 4\sum_{i=1}^{N_a} |\mathcal{N}_i|} = \bar{\delta}, \tag{10.32}$$

where $|\mathcal{N}_i|$ is the cardinality of the set $\mathcal{N}_i$.

The proof of this result is straightforward, since it holds that

$$\|\overrightarrow{\varepsilon}(t)\| \leq \sqrt{\sum_{i=1}^{N_a} \|\varepsilon_i(t)\|^2 + \sum_{i=1}^{N_a} \sum_{j \in \mathcal{N}_i} \|\varepsilon_{ij}(t)\|^2}.$$

Considering the upper bounds on ε_i and ε_{ij} that UwR protocol provides, it yields $\|\overrightarrow{\varepsilon}(t)\| < \sqrt{\sum_{i=1}^{N_a} \delta^2 + \sum_{i=1}^{N_a} \sum_{j \in \mathcal{N}_i} (2\delta)^2} = \sqrt{\delta^2 (N_a + 4\sum_{i=1}^{N_a} |\mathcal{N}_i|)}$, which is equivalent to (10.32).

This shows that due to the state inconsistency, the bound on the error increases. For instance, if WfA protocol is used, the error is bounded by $\|\varepsilon(t)\| < 2\sqrt{N_a}\delta$, which is a lower upper bound than (10.32). Otherwise, if the opposite is assumed, it follows that $\frac{3}{4}N_a > \sum_{i=1}^{N_a} |\mathcal{N}_i|$ must hold by enforcing $\bar{\delta} = \delta\sqrt{N_a + 4\sum_{i=1}^{N_a} |\mathcal{N}_i|} < 2\sqrt{N_a}\delta$. However, this cannot be satisfied for a connected topology.

Larger upper bounds on the error involve more conservative upper bounds on the maximum delay. Hence, it can be expected that the analytic results for the state inconsistency and non-perfect decoupling are more tight. The outcome is enounced in the next theorem.

Theorem 10.5 *If the network delay is upper bounded by*

$$\tau_{max} = \frac{\delta}{\|A_K + \Delta\|\kappa(V)\|x(0)\| + \mu_{max}\left(1 + \frac{\|A_K + \Delta\|\kappa(V)}{|\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|}\right)\bar{\delta}}, \tag{10.33}$$

Table 10.2 Delays for different values of δ and N_a

N_a	δ			
	0.01	0.02	0.05	0.1
10	0.089	0.110	0.191	0.284
20	0.063	0.077	0.129	0.196
50	0.040	0.048	0.080	0.122
100	0.028	0.034	0.057	0.086
200	0.020	0.024	0.040	0.061

where $\mu_{max} = max\{\|M_i\|, i = 1, \ldots, N_a\}$*, then any broadcast state* $x_{b,i}$ *is successfully received by the neighbors* $j \in \mathcal{N}_i$ *before a new event occurs. Hence, the inter-event times are lower bounded* $t_{k+1}^i - t_k^i \geq \tau_{max}$*. Moreover, for all initial conditions* $x(0)$ *and* $t > 0$ *it holds*

$$\|x(t)\| \leq \kappa(V) \left(\frac{\mu_{max}\bar{\delta}}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|} + e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t} \left(\|x(0)\| - \frac{\mu_{max}\bar{\delta}}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|} \right) \right). \quad (10.34)$$

Proof The proof can be found in Appendix A.

Example 10.4 In this example we illustrate the conservatism of Theorem 10.5 when estimating τ_{max}, even though the UwR protocol provides better results for perfect decoupling.

Let us consider the system specifications of Sect. 7.6.2.2. The upper bound on the delay τ_{max} computed according to (10.33) for different values of δ and N_a is given in Table 10.2. The values are expressed in milliseconds.

Given that $\bar{\delta}$ depends on N_a, the tolerable delay is reduced when the number of agents increases. This fact did not have influence in the case of perfect decoupling. Moreover, the increase of the dimension of the matrices with N_a also influences the bound negatively.

Remark 10.2 The conservatism of (10.33) comes from the fact that the individual dynamics of the subsystems cannot be decoupled and the system has to be treated as a whole. However, this does not mean that the system, in practice, cannot tolerate longer delays, simply just the analytical approach taken only guarantees stability for $\tau \leq \tau_{max}$.

10.6 Simulation Results

10.6.1 Performance

To illustrate the theoretical results, let us consider the system in Fig. 7.3 with $N_a = 4$ and $x(0) = (-0.9425 \quad 0 \quad 1.0472 \quad 0 \quad 0.6283 \quad 0 \quad -1.4137 \quad 0)^T$. The system behavior is investigated in three situations:

1. Ideal communication channel.
2. Non-ideal network using WfA protocol.
3. Non-ideal network using UwR protocol.

Let us consider static trigger functions. The upper bounds on the delay have been already computed for WfA and UwR protocols and for different values of the parameter δ and summarized in Table 10.1.

Let $\delta = 0.05$ and a delay generated randomly between zero and the corresponding upper bound specified in Table 10.1 (1.150 ms for WfA and 1.329 for UwR). The state of subsystem 2, the events time, and the control input $u(t)$ are depicted in Fig. 10.4 for the three situations stated above. The behavior of the subsystem is similar in the three cases as the effect of delays in the performance is mitigated by means of the two proposed protocols.

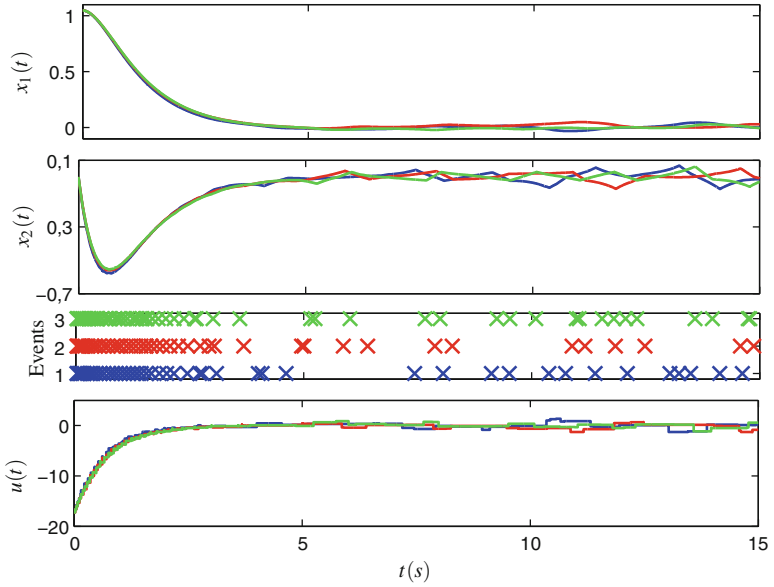

Fig. 10.4 Behavior of the subsystem 2 with WfA (*red*), UwR (*green*) protocols, and an ideal network (*blue*)

Note that even though the delay does not significantly affect the performance, it has an impact on the sequence of events. This is an interesting property of event-based control, because the delay in one transmission affects the occurrence of future events.

10.6.2 Exponential Trigger Functions

If time-dependent trigger functions of the form (10.13) with parameters $\delta_1 = 0.5$ and $\beta = 0.8$ are taken, the upper bound on the delay is 1.43 ms (WfA) and 3.57 ms (UwR), according to Fig. 10.3. Thus, the UwR protocol will be used in this example, as it provides a less restrictive result.

The performance of the system under the time-dependent trigger functions is compared with the behavior using the static-trigger functions for $\tau^i_{max} = 3.57$ ms. The results are shown in Fig. 10.5. The state of Agent 2 (x_{2_1}, x_{2_2}) is depicted in Fig. 10.5a, and Fig. 10.5b shows the broadcast states $(x_{b,2_1}, x_{b,2_2})$. The broadcast state for the constant threshold looks like a continuous function due to the high frequency of events detection, whereas piecewise constant functions are clearly appreciated in the time-dependent trigger function case.

Note that the number of updates in the broadcast state (number of events) decreases with trigger functions (10.13) and the performance around the equilibria is better with respect to (10.3). Moreover, the minimum and mean inter-event times have been computed according for these simulation results, resulting in 3.9 and 353 ms, respectively, for (10.3), and 1.2 ms, which agrees with the results of Table 10.1, and 215 ms for (10.13). Hence, the time-dependent trigger functions are an interesting alternative in non-ideal networks.

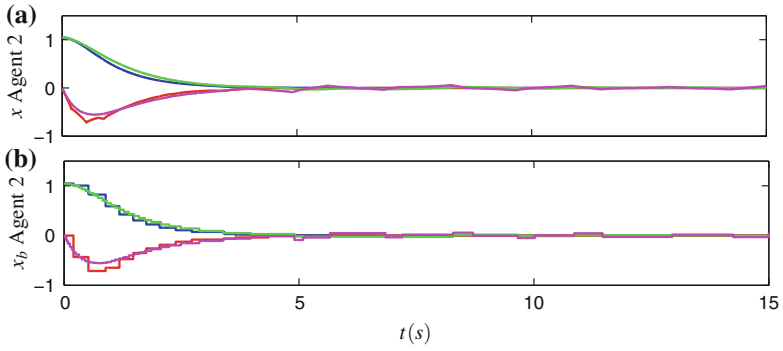

Fig. 10.5 Behavior of the Agent 2 with trigger functions (10.3) ($\delta = 0.05$) (*green*, *magenta*) and (10.13) ($\delta_1 = 0.5$, $\beta = 0.8$) (*blue*, *red*), with 3.57 ms as upper bound on the delay. **a** (x_{2_1}, x_{2_2}), **b** $(x_{b,2_1}, x_{b,2_2})$

10.7 Conclusions

This chapter has presented an extension of the distributed control design of Chap. 7 to non-reliable networks. Two transmission protocols have been proposed as means of dealing with the effects of non-reliable networks. Upper bounds on the delay and maximum number of consecutive packet dropouts have been derived for different situations. One of the main contributions of this chapter is the proof of the existence of a lower bound on the inter-event times and the asymptotic convergence to the origin if time-dependent trigger functions are used.

Chapter 11
Distributed Estimation in Networked Systems

Francisco R. Rubio, Luis Orihuela and Carlos Vivas

11.1 Introduction

Following the series of results on distributed estimation for NCSs, this chapter tackles the problem for the case where communications are taking place over an unreliable asynchronous network.

Most of the problem setups and assumptions are common to those described in Chaps. 9 and 10. A network of interconnected agents is assumed, each one having partial access to measurements from a linear plant, and broadcasting their estimations to their neighbors. The objective is again to reach a reliable estimation of the plant state from every agent location. The observers' structure implemented in each agent is based on local Luenberger-like observers in combination with consensus strategies. The chapter now focusses on the case of a unreliable transmission networks featuring communication delays and eventual packet dropouts. As for Chaps. 9 and 10, both periodic and event-based schemes are analyzed.

There exists a vast literature related to the problem of distributed estimation in sensor networks (see Chaps. 1 and 8 for a brief discussion on the topic). Despite the great deal of effort developed in distributed estimation, there is much room for research in the topic. Specifically, network-induced problems have historically received little attention. When a communication network is used to communicate, designs must be aware of network-induced constraints, significantly, delays and packet dropouts.

F.R. Rubio (✉) · C. Vivas
Dpto. de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingenieros, Universidad de Sevilla, Seville, Spain
e-mail: rubio@us.es

C. Vivas
e-mail: vivas@us.es

L. Orihuela
Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería, Universidad Loyola Andalucía, Seville, Spain
e-mail: dorihuela@uloyola.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_11

These effects degrade the performance of a given estimation scheme, causing eventually the resulting observed system not to converge to the actual plant values, see [104].

Furthermore, the estimation strategy can be designed in time-driven fashion, where the agents are required to broadcast their states at periodic frequency, or event-based, where the information asynchronously transmitted according to rules designed for every agent. As already discussed in Chaps. 9 and 10, the latter methodology is generally more efficient from the point of view of bandwidth use, as communications are invoked only when significant information requires to be transmitted, see [61, 154, 239]. This approach becomes specially beneficial in the context of distributed estimation over networks, as the limitations imposed by the network render the frequency at which the system communicates. A reduction in the transmission frequency implies bandwidth saving but also an improvement in average transmission delays and packet collisions, and for back-off retransmission, algorithms are reduced. The reduced average transmission rate is more appreciated in the case of wireless sensor networks, where extending the battery life span is of paramount importance. These two facts motivate the use of aperiodic communication policies, which allow to avoid the transmission of irrelevant data, reducing network traffic and energy expenditure.

The approach in this chapter for a distributed cooperative estimation framework is discussed based on local Luenberger-like observers in combination with consensus strategies. Remarkably, network-induced delays and packet dropouts are considered. The efficient use of the network resources receives important attention in both time-driven periodic an event-based communication between the agents. The former approach reduces the amount of information communicated over the network resorting to two different ideas: on the one hand, only neighbors are allowed to communicate, reducing transmissions with respect to all-to-all communication schemes. On the other hand, the design of the observers contemplates the possibility of sharing only a part of the estimated state between neighbors, instead of communicating the whole estimated vector state. This economy in the use of network resources is, by its own nature, further improved with the event-driven communication approach.

11.2 Problem Description and Motivation

Consider a sensor network intended to estimate the state of a linear plant in a distributed way, where the sensors measure some variables (outputs), compute a local estimation of the overall state of the system, and broadcast to a set of neighbors of some information related with their own estimations. As Fig. 11.1 illustrates, the set of agents are connected by means of a communication network, which may introduce delays and packet dropouts. When the local information received for each of the different agents is not sufficient to estimate the complete state of the plant, then the proposed type of distributed observation makes sense.

The concepts of *local observability* and *collective observability* refer, respectively, to a situation in which the measurement performed by any sensor is sufficient to

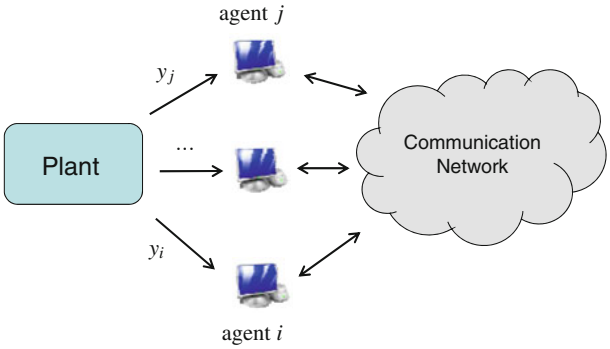

Fig. 11.1 Distributed observation scheme with a set of agents sharing information with neighboring devices

guarantee observability of the process state; and to a situation in which all the sensors, if put together, guarantee this property. See [200] for a complete explanation. In this chapter, it is assumed that all the sensors must estimate the overall state of the system even when local observability does not hold.

To motivate the problem, consider a possible application where the state of a plant is monitored from different geographically distributed locations, provided that only some local information of the plant can be directly measured from each location. This scenario might consist of a number of observers having access to some, generally different plant outputs. The plant is not necessarily fully observable from any of the observers. The different observers are able to communicate themselves by sharing information with a set of neighbors in order to estimate the complete state of the plant. The communication among the different observers is assumed to be implemented through a communication network, in which time delays and possible packet dropouts have to be taken into consideration. A different situation in which the framework considered in this work might be of application could be that in which the observers are connected using a shared medium, managing traffic information in a urban environment.

Taking into account the aforementioned ideas, the present chapter focuses on network-related issues, specifically communication efficiency and robustness against network-induced problems. The main contributions are briefly summarized subsequently. The chapter provides an observer design method to operate with time-driven communication between the agents, being the objective to reach a common reliable estimate of the system state, despite of the presence of delays and dropouts. After this, an extension is proposed to include an event-based communication strategy between agents, aiming at reducing the traffic over the network and the energy consumption.

In the latter case, the estimation error will eventually enter into an arbitrary small region around the equilibrium point. Similar to Chap. 5, the size of this region depends on a free parameter that sets the threshold triggering the communication events, which allows to trade-off between communication savings and estimation performance.

The chapter is organized as follows. Section 11.2 describes and motivates the problem under consideration. First, in Sect. 11.3, the particular case of periodic communications is presented, followed by a discussion of the necessary changes to be introduced to accommodate the observers structure to the asynchronous case in Sect. 11.4. Practical stability of the proposed approach is proved in Sect. 11.4.2. The chapter closes with some illustrative examples in Sects. 11.5 and 11.6.

11.2.1 Network Topology

The communication network is represented using a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = 1, 2, \ldots, N_a$ being the set of agents (observers) of the graph (network), and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, being the set of links. Assuming the cardinality of $\mathcal{E}$ equal l, and defining $\mathcal{L} = \{1, 2, \ldots, l\}$, it is obvious that a bijective function $g : \mathcal{E} \to \mathcal{L}$ can be built so that a given link can be either referenced by the pair of agents that connects $(i, j) \in \mathcal{E}$ or the link index $r \in \mathcal{L}$, so that $r = g(i, j)$. The set of agents connected to agent i is termed the neighborhood of i, and denoted as $\mathcal{N}_i \triangleq \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. Directed communications are considered, so that link (i, j) implies that agent i receives information from agent j.

11.2.2 System Description

In this work, the system to be observed is assumed to be an autonomous linear time-invariant plant given by the following equations:

$$x(k+1) = Ax(k), \tag{11.1}$$

$$y_i(k) = C_i x(k), \quad \forall i \in \mathcal{V}, \tag{11.2}$$

where $x(k) \in \mathbb{R}^n$ is the state of the plant and $y_i(k) \in \mathbb{R}^{d_i}$ are the system outputs. In general, each of the different N_a observers has access to a distinct output of the plant. Collective observability is assumed, i.e., the pair (A, C) is observable, where C is a matrix stacking the output matrices C_i of all the agents.

Furthermore, the observers can communicate with each other by means of a communication network that can be represented by the graph $\mathcal{G}$. More precisely, each neighbor j of the observer i communicates some estimated outputs $\hat{y}_{ij} = C_{ij}\hat{x}_j$. It is assumed that agent i knows the matrix C_{ij} corresponding to the output $\hat{y}_{ij}$. Exchanging estimates instead of the measurements from the plant provides some freedom and flexibility to choose the information sent through the network. In this way, taking into account the plant dynamics and the output measured by a particular agent, it is possible to collect only the information from its neighbors that allows estimation, leading to a policy in which only relevant information for each agent is transmitted.

Let us define $\bar{C}_i$ as a matrix stacking the matrix C_i and matrices C_{ij} for all $j \in \mathcal{N}_i$. It is assumed that each pair $(A, \bar{C}_i)$ is observable. This is a necessary condition that imposes some restrictions on the network topology and the information that is sent via each connection.

11.3 Periodic Time-Driven Communication Between Agents

This section is devoted to the observer design method under periodic communication between agents. Next, a description of the agent dynamics is given in detail.

11.3.1 Agent Dynamics

The communication between agents may be affected by delays and packet dropouts. Therefore, it is convenient that the observer to be proposed takes both effects under consideration. Figure 11.2 illustrates a possible time scheduling in which both effects appear. Let $\tau_{ij}(k) \in \mathbb{N}$ represent the time difference between the current time instant k and the instant in which the last packet received by agent i was sent from its neighbor j. This constant includes the effect of delays and packet dropouts. Note that packet dropouts have the effect of enlarging $\tau_{ij}(k)$.

Assuming that the number of consecutive data dropouts is bounded by n_p and that the effective network-induced delays are also bounded by h_{min} and h_{max}, it is obvious that $\tau_{ij}(k)$ belongs to the interval $[\tau_{min}, \tau_{max}]$, where

$$\tau_{min} = h_{min}, \tag{11.3}$$

$$\tau_{max} = n_p + h_{max}. \tag{11.4}$$

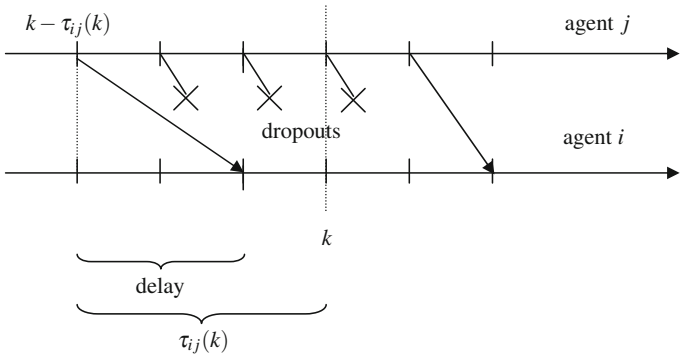

Fig. 11.2 Time scheduling for a typical communication between two agents

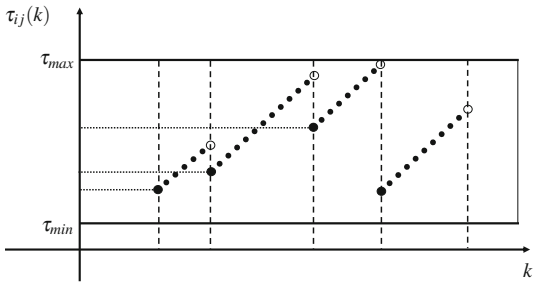

Fig. 11.3 Qualitative evolution of $\tau_{ij}(k)$

Figure 11.3 illustrates a characteristic shape for $\tau_{ij}(k)$. For the sake of simplicity in the technical developments to come, we assume that the minimum delay bound is exactly zero, $\tau_{min} = 0$, and the maximum delay will be denoted by τ_{max}. In other works as [176] or [204], this assumption is relaxed, leading to a more cumbersome, but still solvable, design problems.

Also note that each τ_{ij} is directly associated to a link $(i, j) \in \mathscr{E}$. Then, it is possible to establish a relation between each connection and the corresponding τ_{ij}. Numbering the links from 1 to l, we can use the following equivalent notation:

$$\tau_r(k) = \tau_{ij}(k), \ r = 1, \ldots, l, \tag{11.5}$$

where $r = g(i, j)$, that is, we can either refer τ to a pair of agents (τ_{ij}) or to a link (τ_r).

Once the considerations about delays and packet dropouts have been done, we propose a structure for the observers given by the following equation:

$$\begin{aligned}\hat{x}_i(k+1) = A\hat{x}_i(k) + M_i(C_i\hat{x}_i(k) - y_i(k)) \\ + \sum_{j \in \mathscr{N}_i} N_{ij}(C_{ij}\hat{x}_j(k - \tau_{ij}(k)) - C_{ij}\hat{x}_i(k - \tau_{ij}(k))),\end{aligned} \tag{11.6}$$

for $i \in \mathscr{V}$. The structure of the observers comprises two main parts, namely,

- A local Luenberger-like observer, weighted with matrices M_i, which corrects the estimated state of the plant based on the measured output $y_i(k)$ accessible for each observer i.
- A consensus-based observer, weighted with matrices N_{ij}, which takes into account the information received from neighboring observers.

The name consensus comes from the fact that all the agents will eventually achieve the same value of the estimated state. Note that agent i must known the exact value of the actual artificial delay $\tau_{ij}(k)$, as it needs to compare the received information $C_{ij}\hat{x}_j(k - \tau_{ij}(k))$ with past values of its own estimated state $C_{ij}\hat{x}_i(k - \tau_{ij}(k))$. To do this, the agents must be synchronized. Assuming that some kind of synchronization algorithm is running, the delay $\tau_{ij}(k)$ can be known by adding a timestamp to every

data packet. Furthermore, each agent must buffer all its past estimates until instant $k - \tau_{max}$.

Let us consider now the observation error of a generic observer i defined as $e_i(k) = \hat{x}_i(k) - x(k)$, i.e., the difference between the estimation of agent i and the state of the plant. Taking into account Eqs. (11.1) and (11.6), the dynamics of the observation error can be written as

$$e_i(k+1) = (A + M_i C_i)e_i(k) + \sum_{j \in \mathcal{N}_i} N_{ij} C_{ij}(e_j(k - \tau_{ij}(k)) - e_i(k - \tau_{ij}(k))). \quad (11.7)$$

Considering that the number of observers is given by N_a, the dynamic equations of the observation errors can be written in a compact form defining a stacked error vector as $e^T(k) = [e_1^T(k) \quad e_2^T(k) \quad \dots \quad e_{N_a}^T(k)]$:

$$e(k+1) = \Phi(\mathcal{M})e(k) + \Lambda(\mathcal{N})d(k) \quad (11.8)$$

where $d(k)$ is a delayed version of the stacked error vector taking into account the delays of the different links (see Eq. (11.5)) $d^T(k) = [e^T(k - \tau_1(k)) \quad \dots \quad e(k - \tau_l(k))^T]$, or equivalently the delays of the communications between neighbors i and j. The matrices $\Phi(\mathcal{M})$ and $\Lambda(\mathcal{N})$ depend on the sets of observers to be designed: $\mathcal{M} = \{M_i, i \in \mathcal{V}\}$ and $\mathcal{N} = \{N_{ij}, i \in \mathcal{V}, j \in \mathcal{N}_i\}$. It is not difficult to see that the structure of such matrices is given by

$$\Phi(\mathcal{M}) = \begin{bmatrix} A + M_1 C_1 & 0 & \cdots & 0 \\ 0 & A + M_2 C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A + M_{N_a} C_{N_a} \end{bmatrix} \quad (11.9)$$

$$\Lambda(\mathcal{N}) = [\Lambda_1 \ \Lambda_2 \ \cdots \ \Lambda_l] \quad (11.10)$$

where $\Lambda_r, \ r = g(i,j) \in \{1, \dots, l\}$, are block matrices in correspondence with each of the links r communicating the observer i with j, in which the only blocks different from zero are $-N_{ij}C_{ij}$ and $N_{ij}C_{ij}$ in the (i,i) and (i,j) positions, respectively:

$$\Lambda_r = \begin{array}{c} \begin{matrix} column & i & & j & \end{matrix} \\ \begin{bmatrix} 0 \cdots & 0 & \cdots & 0 & \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 \cdots & -N_{ij}C_{ij} & \cdots & N_{ij}C_{ij} & \cdots 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 \cdots & 0 & \cdots & 0 & \cdots 0 \end{bmatrix} \end{array} \begin{matrix} \\ \\ row\ i. \\ \\ \\ \end{matrix}$$

11.3.2 Observer Design

In the following, the main result of this section is introduced. For periodic communication between agents, the next theorem states a sufficient condition for the asymptotic convergence of the estimates of each observer to the plant state.

Theorem 11.1 *The estimations of all the observers asymptotically converge to the plant state if, given a maximum delay bound τ_{max} according to Eq. (11.4), the nonlinear matrix inequality*

$$\begin{bmatrix} \Xi & \Theta & 0 & \Phi^T(\mathscr{M})P & (\Phi^T(\mathscr{M})-I)P\tau_{max} \\ * & \Upsilon & \Theta^T & \Lambda^T(\mathscr{N})P & \Lambda^T(\mathscr{N})P\tau_{max} \\ * & * & \Omega & 0 & 0 \\ * & * & * & -P & 0 \\ * & * & * & * & -\frac{1}{l}PZ_2^{-1}P \end{bmatrix} < 0 \tag{11.11}$$

has a feasible solution for positive definite matrices $Z_1, Z_2, P_i,\ i \in \mathscr{V}$, and observers matrices $M_i, N_{ij},\ i \in \mathscr{V},\ j \in \mathscr{N}_i$, where

$$\begin{aligned} P &= diag\{P_1, P_2, \ldots, P_{N_a}\}, \\ \Xi &= -P + Z_1 - lZ_2, \\ \Theta &= \overbrace{[Z_2 \quad Z_2 \quad \cdots \quad Z_2]}^{l \text{ times}}, \\ \Upsilon &= diag\overbrace{\{-2Z_2, \ldots, -2Z_2\}}^{l \text{ times}}, \\ \Omega &= -Z_1 - lZ_2. \end{aligned}$$

The proof of this theorem uses the Lyapunov–Krasovskii theory to ensure the stability of the observation errors. The details can be found in Appendix A.

As it is clearly seen from (11.11), the matrix inequality to be solved in order to design the observers is nonlinear because of the presence of the terms $\Phi^T(\mathscr{M})P$, $\Lambda^T(\mathscr{N})P$, and $PZ_2^{-1}P$.

The first two nonlinearities, related to $\Phi^T(\mathscr{M})P$, $\Lambda^T(\mathscr{N})P$, can be trivially settled by defining $M_iP_i = W_i$ and $N_{ij}P_i = X_{ij}$. In this way, these terms are now a function of the new matrices in the change of variables, i.e., $\Phi^T(\mathscr{M})P \to \Phi(\mathscr{W})$ and $\Lambda^T(\mathscr{N})P \to \Lambda(\mathscr{X})$, where the sets are defined as $\mathscr{W} = \{W_i, i \in \mathscr{V}\}$ and $\mathscr{X} = \{X_{ij}, i \in \mathscr{V}, j \in \mathscr{N}_i\}$.

The nonlinearity $PZ_2^{-1}P$ cannot be sorted with this or any other direct technique that let us transform the nonlinear condition into a linear one. To solve this problem, two solutions can be implemented. The first one requires the introduction of an additional constraint

$$-PZ_2^{-1}P < -\frac{1}{\mu}P$$

which let us address the problem by means of a set of linear matrix inequalities. The second solution uses the cone complementary algorithm to transform the nonlinear inequality into an iterative optimization problem with linear constraints. Comparing both solutions, the former is more conservative (as it introduces an additional condition), but it is computationally more efficient. Both methods are explained in detail in Appendix B.

Remark 11.1 The computational burden required to solve the conditions in Theorem 11.1 (through any of the proposed methods) directly depends on the dimension of the system, the number of agents, and the number of links between them. Although the implementation of the estimation scheme is completely distributed, the design procedure is centralized, as all the weighting matrices are designed together. With respect to centralized schemes, the number of links is now the bottleneck of all the proposed solutions in the literature. It would be of undeniable interest to distribute the mathematical calculus, in such a way that each agent does not need information of the rest of the agents, but only of its neighbors, to design its observer. This is matter of future research.

11.4 Event-Based Communication Between Agents

This section analyzes an asynchronous event-based communication policy between agents to reduce the energy consumption and make an efficient use of the network resources.

Transmissions between neighbors are now assumed to be triggered at specific time instants, when a triggering condition is satisfied. Let $\ell_{ij}(k)$ denote the time instant when agent j sent the more recent packet available for the agent i at current time instant k. Then, $\{\ell_{ij}(k)\} \subset \mathbb{N}$, as agent j only sends packets when an event is triggered.

Triggering condition Given a threshold δ, at instant k, agent j broadcasts its estimates to every agent i such that $j \in \mathcal{N}_i$ if

$$\|\hat{x}_j(\ell_{ij}(k)) - \hat{x}_j(k)\|_\infty \geq \delta, \quad \text{for } k > \ell_{ij}(k). \tag{11.12}$$

In this section, packet dropouts are not considered, so only delays affect the communication between agents.

11.4.1 Remodeling of the Observer Dynamics

This section introduces the modifications needed to remodel the system dynamical equations according to the approach introduced above. Consider a generic agent i at time instant k. As it was explained in Chap. 5 for the centralized scheme, there are

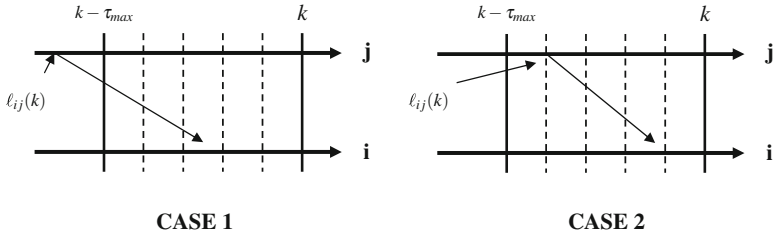

Fig. 11.4 Different cases regarding the transmission of information from agent j to i

also two possible situations with respect to the information received from each of its neighbors $j \in \mathcal{N}_i$ (see Fig. 11.4):

- **Case 1** The last packet received in agent i was sent before $k-\tau_{max}$. It is obvious that $\ell_{ij}(k) < k-\tau_{max}$. In this case, agent i does not have in memory[1] the estimate $\hat{x}_i(\ell_{ij}(k))$. Then, it compares its older buffered state, that is, $\hat{x}_i(k-\tau_{max})$, with the available state of its neighbor $\hat{x}_j(\ell_{ij}(k))$ to correct its estimation:

$$\hat{x}_i(k+1) \underset{\text{CASE 1}}{=} A\hat{x}_i(k) + M_i(\hat{y}_i(k) - y_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}C_{ij}\left(\hat{x}_j(\ell_{ij}(k)) - \hat{x}_i(k-\tau_{max})\right).$$

- **Case 2** The last packet received in agent i was sent after $k-\tau_{max}$, so $\ell_{ij}(k) \geq k-\tau_{max}$. Given that the estimate $\hat{x}_i(\ell_{ij}(k))$ is still in the buffer of the agent i, it compares it with the received information $\hat{x}_j(\ell_{ij}(k))$:

$$\hat{x}_i(k+1) \underset{\text{CASE 2}}{=} A\hat{x}_i(k) + M_i(\hat{y}_i(k) - y_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}C_{ij}\left(\hat{x}_j(\ell_{ij}(k)) - \hat{x}_i(\ell_{ij}(k))\right).$$

Taking into account the above considerations, the dynamics of a generic agent i can be rewritten compactly as

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + M_i(\hat{y}_i(k) - y_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}C_{ij}\left(\hat{x}_j(\ell_{ij}(k)) - \hat{x}_i(k-\mu_{ij}(k))\right), \tag{11.13}$$

where

$$\mu_{ij}(k) = \begin{cases} \tau_{max}, & \ell_{ij}(k) < k-\tau_{max}, \\ k-\ell_{ij}(k), & \ell_{ij}(k) \geq k-\tau_{max}. \end{cases}$$

[1] Recall that each agent stores only a finite amount of past estimates, as was explained in Sect. 11.3.

Considering the addition of the null term $\hat{x}_j(k-\mu_{ij}(k)) - \hat{x}_j(k-\mu_{ij}(k))$ and defining

$$\varepsilon_{ij}(k) = \hat{x}_j(\ell_{ij}(k)) - \hat{x}_j(k-\mu_{ij}(k)).$$

Equation (11.13) can be rewritten as

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + M_i(\hat{y}_i(k) - y_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}C_{ij}\left(\hat{x}_j(k-\mu_{ij}(k)) - \hat{x}_i(k-\mu_{ij}(k))\right) + \sum_{j\in\mathcal{N}_i} N_{ij}C_{ij}\varepsilon_{ij}(k).$$

In this way, the observer have a dynamics equivalent to that of the periodic communication case, difference being in the terms that depend on $\varepsilon_{ij}(k)$, which can be interpreted as an external perturbation due to the discontinuous flow of information between neighbors that is reset to zero at every transmission time. In this case, $\mu_{ij}(k)$ plays the role of $\tau_{ij}(k)$ in Eq. (11.7). It is straightforward to check that $0 \leq \mu_{ij}(k) \leq \tau_{max}$ for Cases 1 and 2.

Moreover, it is easy to see that

$$\|\varepsilon_{ij}(k)\|_\infty < \delta,$$

in both cases. In the Case 2, $\varepsilon_{ij}(k) = 0$. In Case 1, it holds $\varepsilon_{ij}(k) = \hat{x}_j(\ell_{ij}(k)) - \hat{x}_j(k-\tau_{max})$, with $\ell_{ij}(k) < k - \tau_{max}$. Since no packet has been transmitted between $\ell_{ij}(k)$ and $k-\tau_{max}$ (because, otherwise, this packet had been available in agent i at current instant k), it implies that $\|\hat{x}_j(\ell_{ij}(k)) - \hat{x}_j(k-\tau_{max})\|_\infty < \delta$ (see the triggering condition defined above). Therefore, $\|\varepsilon_{ij}(k)\|_\infty < \delta$ holds for both cases.

The dynamics of the augmented observation vector $e(k)$ is similar to that of the time-driven case, but including an additional term related with these disturbances:

$$e(k+1) = \Phi(\mathcal{M})e(k) + \Lambda(\mathcal{N})d(k) + \Gamma(\mathcal{N})\varepsilon(k), \tag{11.14}$$

where $\varepsilon^T(k) = [\varepsilon_1^T(k), \ldots, \varepsilon_r^T(k), \ldots, \varepsilon_l^T(k)]$, with $r = g(i,j)$. As before, it is not difficult to see that matrix $\Gamma(\mathcal{N})$ has the following structure:

$$\Gamma(\mathcal{N}) = \left[\, \Gamma_1(\mathcal{N}) \cdots \Gamma_r(\mathcal{N}) \cdots \Gamma_l(\mathcal{N}) \,\right],$$

where $\Gamma_r(\mathcal{N})$, $r = g(i,j) \in \{1, \ldots, l\}$, are vectors of matrices, in which the only block different from zero is $N_{ij}C_{ij}$ in the i row:

$$\Gamma_r(\mathcal{N}) = \begin{bmatrix} 0 \\ \vdots \\ N_{ij}C_{ij} \\ \vdots \\ 0 \end{bmatrix} row\ i.$$

As has been mentioned, in this section, we consider that the observers are designed according to Theorem 11.1, so in the following, notations Φ, Λ, and Γ will be used instead of $\Phi(\mathcal{M})$, $\Lambda(\mathcal{N})$, and $\Gamma(\mathcal{N})$, respectively.

11.4.2 Practical Stability for Delayed Asynchronous Systems

Next, the main result of this section is introduced. Given the distributed observer synthesized by Theorem 11.1, the following result proves that, by implementing the event-based sampling policy described above, the observation error $e(k)$ can be ultimately bounded into an arbitrary small region that depends on the triggering threshold δ. The theorem is based on the Lyapunov–Krasovskii theory, through the same functional given in Theorem 11.1. The proof, which can be found in the appendix, makes use of the following two facts:

- The functional can be written as a quadratic function:

$$V(k) = \zeta^T(k)\Psi\zeta(k), \tag{11.15}$$

  where

$$\zeta(k) = \begin{bmatrix} e(k) \\ e(k-1) \\ e(k-2) \\ \vdots \\ e(k-\tau_{max}) \end{bmatrix},$$

  and Ψ is a positive definite matrix that can be easily found, due to the structure of the functional. The interested reader can find an example in Chap. 5. It has been omitted here, since it has no influence in the following theorem.
- The evolution of the functional $\Delta V(k)$ can be bounded with another quadratic function:

$$\Delta V(k) \leq \xi^T(k)L_1\xi(k)$$

  where $\xi(k)$ is an augmented state vector and L_1 a negative definite matrix, both defined in the proof of Theorem 11.1.

Theorem 11.2 *Consider a set of distributed observers designed by Theorem 11.1. Then, using an event-based communication with triggering condition (11.12), the estimation error $e(k)$ whose dynamics is given in Eq. (11.14) is ultimately bounded by*

$$\|e(k)\|_\infty \leq \delta \sqrt{\frac{\lambda^{\Psi}_{\max}}{\lambda^{P}_{\min}}} \left[(\|\Phi\|_\infty + \|\Lambda\|_\infty) D_1 + \|\Gamma\|_\infty \right],$$

where matrices P, Φ, Λ, and Γ are given in Theorem 11.1 and

$$\begin{aligned} D_1 &= \frac{\|L_2\|_\infty + \sqrt{\|L_2\|^2_\infty + \lambda^{Q}_{\min} \|L_3\|_\infty}}{\lambda^{Q}_{\min}}, \\ L_2 &= \Gamma^T P \left[\Phi \ \Lambda \ 0 \right] + \Gamma^T Z_2 \left[(\Phi - I) \ \Lambda \ 0 \right], \\ L_3 &= \Gamma^T P \Gamma + \Gamma^T Z_2 \Gamma, \end{aligned}$$

being Q any positive definite matrix such that $-Q > L_1$.

Observe that the final bound of $e(k)$ depends on the threshold δ that triggers the sampling. By choosing $\delta = 0$, the events are triggered at each sampling time, obtaining a time-driven scheme. Larger values of δ imply larger periods without events. Therefore, parameter δ can be used to trade-off the number of retransmission and the performance of the estimation.

11.5 Simulation Results

Consider a plant whose dynamics is given by

$$x(k+1) = \begin{bmatrix} 0.99 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.005 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9945 & -0.08757 & 0 & 0 \\ 0 & 0 & 0.1248 & 0.9945 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 & 0.09 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x(k).$$

Agent 1 measures the output $y_1 = x_1$, while observers 2, 3, and 4 receive $y_2 = x_2$, $y_3 = x_4$, and $y_4 = x_6$, respectively. The agents are connected according to an incomplete communication graph, depicted in Fig. 11.5. The outputs measured from every agent and the received estimates from their neighborhood are summarized in Table 11.1.

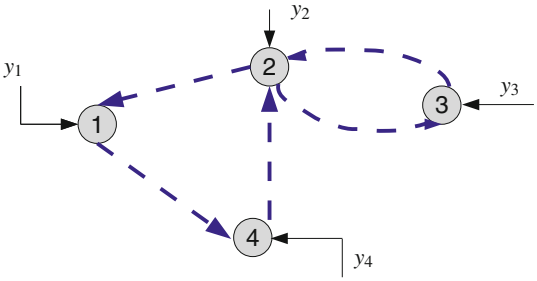

Fig. 11.5 Graph representing the network connectivity

Table 11.1 Outputs and information shared with neighbors

	Measurements	Received outputs
Agent 1	$y_1 = C_1 x = [1\ 0\ 0\ 0\ 0\ 0]x$	$C_{12} = [C_2; C_3; C_4]$
Agent 2	$y_2 = C_2 x = [0\ 1\ 0\ 0\ 0\ 0]x$	$C_{23} = C_3,\ C_{24} = [C_1; C_4]$
Agent 3	$y_3 = C_3 x = [0\ 0\ 0\ 1\ 0\ 0]x$	$C_{32} = [C_1; C_2; C_4]$
Agent 4	$y_4 = C_4 x = [0\ 0\ 0\ 0\ 0\ 1]x$	$C_{41} = [C_1; C_4]$

Note that local observability is not achieved from any of the observers. The design of the observation matrices and the simulations have been performed for a maximum delay of $\tau_{max} = 3$ in all links.

Figures 11.6 and 11.7 represent the evolution of the plant states (continuous lines) and the estimated states (dashed lines) for agents 1 and 4, respectively. The initial states for all the observers are set to zero. The plant's initial state is $x_0 = \left[-4\ 1.2\ 0.5\ -1\ 3.5\ 5\right]^T$. Observe that these states which can be locally measured in an agent converge faster to the actual plant states, as they are not affected by communication effects (delays and asynchronicity).

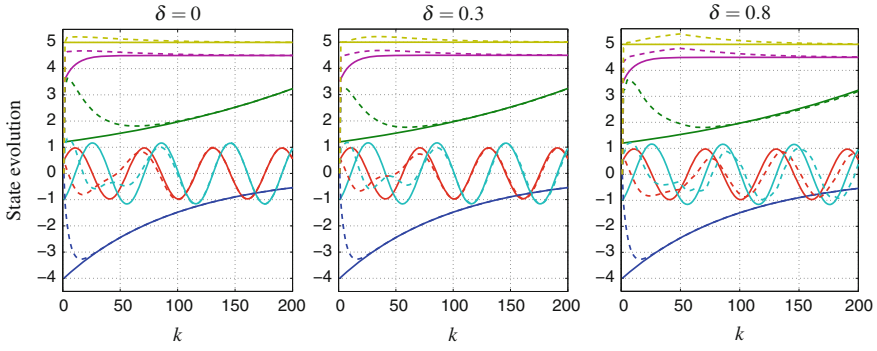

Fig. 11.6 Evolution of the estimates for Agent 1

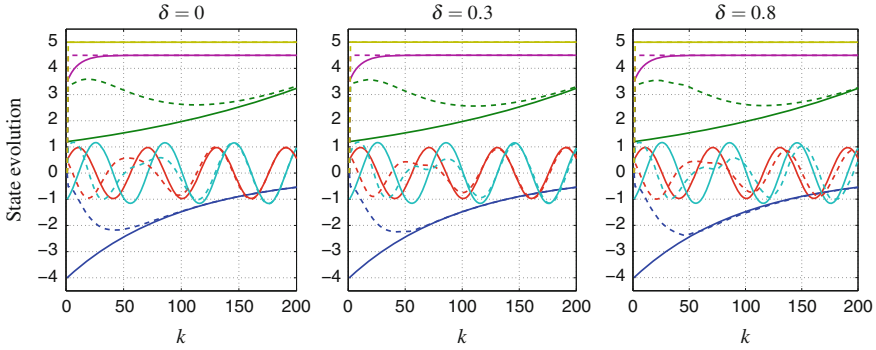

Fig. 11.7 Evolution of the estimates for Agent 4

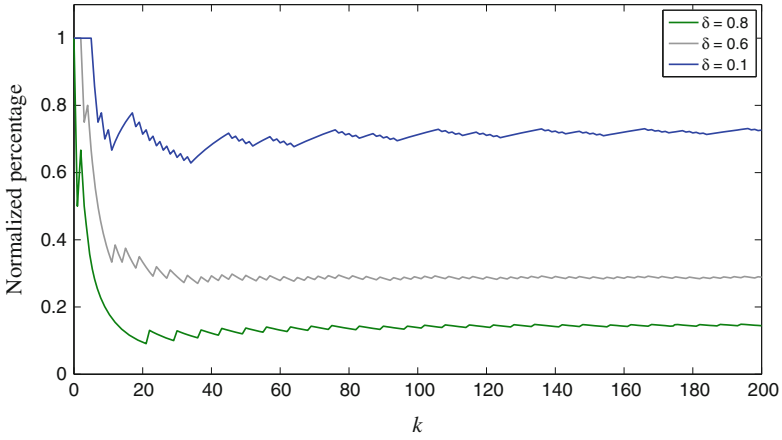

Fig. 11.8 Normalized percentage of transmitted packets for different thresholds

For both observers, the simulations on the left correspond to periodic communication policy ($\delta = 0$). As expected, when the threshold to communicate is enlarged, the observers broadcast less information to their neighbors and the estimation performance progressively diminishes. Nonetheless, it is possible to find an adequate trade-off between estimation performance and communication savings, achieving remarkable reduction on the network traffic load while maintaining an estimation performance close to the periodic communication case.

Finally, Fig. 11.8 shows the percentage of packets transmitted with respect to periodic communication policy for Agent 1 and different communication thresholds.

11.6 Conclusions

In this chapter, the problem of distributed estimation considering network-induced delays and dropouts is solved. Two schemes are analyzed, namely, periodic time-driven and event-based approaches, the latter being specially beneficial in terms of economy of use of network resources. For both scenarios, the observers employ a local Luenberger-like structure and consensus matrices to weight the information received from neighbors. The information shared between neighbors does not need to be necessarily the complete estimated state, but can be selected to reduce communication requirements. Stability proofs are provided and performance of the design methods is showed by a simulation example.

Future research may include the consideration of different delay bounds for each link as well as the robustification of the method to deal with parametric uncertainties in the system model and exogenous perturbations or noise. Other important line of research consists in reducing the computational requirements of the proposed algorithms, by distributing the design algorithms.

Chapter 12
Networked Mobile Robots: An Application Example of the Distributed Event-Based Control

Gonzalo Farias, María Guinaldo and Sebastián Dormido

12.1 Introduction

In large-scale systems, the interconnection of subsystems can be physical or introduced through the control law, such as the case of cooperative control problems in multi-agent systems. The focus of this chapter is on the second type of interconnections, which is also an interesting field to apply the decentralized event-triggered strategies studied in previous chapters for physically coupled systems.

One example of a group objective for multi-agent systems is the reaching of a state agreement or consensus, i.e., all agents are supposed to converge to a common point or state. Such consensus problems have a variety of applications in flocking, attitude synchronization in satellite swarms, distributed sensor networks, congestion control in communication networks, or formation control [201]. We are particularly interested in the last field of application since achieving a stable formation is analogous to reaching consensus. The formation control problem based on consensus algorithms can be applied to mobile robots [219], which can be modeled as non-holonomic vehicles.

A centralized approach to formation control makes difficult the scalability of the problem and it is more sensitive to failure or joining of agents, obstacles in the operating environment, or other external influences, than a neighbor-based coordination strategy. Recent developments in the fields of communication technology, wireless

G. Farias
Escuela de Ingeniería Eléctrica, Pontificia Universidad
Católica de Valparaiso, Valparaiso, Chile
e-mail: gonzalo.farias@ucv.cl

M. Guinaldo (✉) · S. Dormido
Dpto. de Informática y Automática, Escuela Técnica Superior de Informática,
UNED, Madrid, Spain
e-mail: mguinaldo@dia.uned.es

S. Dormido
e-mail: sdormido@dia.uned.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_12

technology, and embedded devices have made possible the implementation of these decentralized techniques in autonomous mobile robots, since agents are able to exchange information through a shared communication network, mainly wireless.

Although the problem of multi-agent systems with event-based communications has been recently addressed [52, 232], the study of the effect of communication networks over the control performance, and in particular over the formation control, requires still many simulations because of the mutual and complex influence between control and communication algorithms. Normally, the simulation of networked control systems is done for a specific scenario. Researchers generally write their programming codes for their particular problems to obtain the simulation results or they use commercial software such as MATLAB®/Simulink® to develop simulation tools. The main drawback of these solutions is that, in general, they are not so flexible and interactive, and it may be necessary to connect them to additional software to simulate the network counterpart.

Apart from the lack of simulation tools for multi-agent systems, the evaluation of the cited communication strategies has been not carried out so far in an experimental platform.

This chapter describes in Sect. 12.2 the problem of formation control for mobile robots from a consensus perspective and the issues to account for in a practical implementation. Section 12.3 depicts a developed simulation tool that fills the gap of integrated tools to simulate the formation control of autonomous agents. The aspects regarding the control and the communication of a group of networked robots are all merged in a single tool, which is, moreover, license free. The high degree of interactivity and flexibility provides a large set of possible experiments in which the coupling between control and communication can be analyzed.

The triggering mechanisms described in Chap. 7 as well as periodic communications are implemented. This implementation has been also carried out over a test bed of mobile robots. The experimental platform as well as the results are presented in Sect. 12.4. Efficiency in the communications and energy consumed are evaluated, showing the benefits of using event-driven communications.

12.2 Formation Control for Networked Mobile Robots

This section backgrounds the problem under consideration. First, an overview of multi-agent systems and the consensus problem is given. After this, the formation control is studied as an extension of the consensus problem. The model for nonholonomic vehicles that has been used in the implementation is provided subsequently. Finally, the possible transmission policies and the communication protocols for wireless robotics are discussed.

12.2.1 Multi-agent Systems and the Consensus Problem

To set the complete model of this setup, we need to define two features: the agents' dynamics and the communication. The simplest model to represent the communication topology of a multi-agent system is a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the nodes $\mathcal{V}$ correspond to agents and the edges $\mathcal{E}$ between nodes represent communication links between agents. We say that $\mathcal{G}$ is connected if there is a path for any pair of nodes in the network.

According to [202], a simple consensus algorithm to reach an agreement regarding the state of N_a single-integrator agents of the form $\dot{x}_i(t) = u_i(t)$ can be expressed as an nth order linear system on a graph

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t)). \tag{12.1}$$

The dynamics of the group of agents can be written as

$$\dot{x}(t) = -Lx(t) \tag{12.2}$$

where L is the Laplacian matrix of the network (or the communication graph) and its elements are defined as follows:

$$l_{ij} = \begin{cases} -1 & \text{if } j \in \mathcal{N}_i \\ |\mathcal{N}_i| & \text{if } j = i, \end{cases} \tag{12.3}$$

where $|\mathcal{N}_i|$ refers to the number of neighbors of the agent i.

Let us also define the *adjacency* matrix A of $\mathcal{G}$ with entries

$$a_{ij} = \begin{cases} 1 & \text{if } (j, i) \in \mathcal{E} \\ 0 & \text{otherwise}, \end{cases}$$

and the *degree* matrix D as the diagonal matrix with diagonal elements d_i equal to $|\mathcal{N}_i|$, so that $L = D - A$.

Example 12.1 Assume a five-node multi-agent network with the communication graph depicted in Fig. 12.1a. In this example, the node 1 can communicate with nodes 2 and 3 but not with nodes 4 or 5. It holds that $\mathcal{V} = \{1, 2, 3, 4, 5\}$ and

$$\mathcal{E} = \{(1, 2), (1, 3), (2, 1), (2, 3), (2, 5), (3, 1), (3, 2), (3, 4), (4, 3), (4, 5), (5, 2), (5, 4)\},$$

and it follows that $D = diag(2, 3, 3, 2, 2)$ and

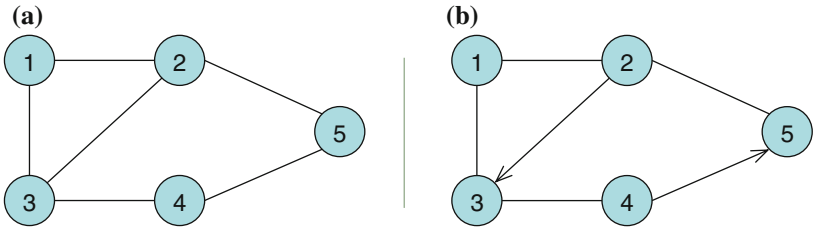

Fig. 12.1 Examples of **a** undirected and **b** directed graphs

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

Based on analytical tools from algebraic graph theory, it can be shown that if the graph is connected, then there is an unique equilibrium state for (12.2) of the form $x_{eq} = \alpha \mathbf{1}$, where $\alpha = \frac{1}{N_a} \sum_{i=1}^{N_a} x_i(0)$ and $\mathbf{1} = (1 \ldots 1)^T$ [201].

The above results hold for undirected graphs, i.e., for bidirectional communications: for any pair $(i, j) \in \mathscr{E}$, there is another edge $(j, i) \in \mathscr{E}$. In a directed communication graph, there is at least one pair of nodes whose communication is unidirectional. An example is given in Fig. 12.1b. In this case, the node 2 transmits to node 3, but the communication is not allowed in the opposite direction. Also, nodes 4 and 5 are unidirectionally connected.

For directed graphs, $\mathscr{N}_i$ is defined as the set of agents from which agent i receives information. Note that the Laplacian and adjacency matrices are not symmetric in this case. Still a consensus can be reached if there is a directed path connecting any two arbitrary nodes (i, j) of the graph [201].

There are also in the literature extensions regarding the agents dynamics [194, 218, 219, 230]. For double-integrator dynamics $\dot{x}_{i,1}(t) = x_{i,2}(t)$, $\dot{x}_{i,2}(t) = u_i(t)$, the consensus algorithm as introduced in [218] is given by

$$u_i(t) = \sum_{j \in \mathscr{N}_i} (x_{j,1}(t) - x_{i,1}(t)) + \gamma (x_{j,2}(t) - x_{i,2}(t)), \tag{12.4}$$

where $\gamma > 0$.

For connected and undirected graphs, [218] shows that the consensus of double integrators is achieved, but the agents' states do not converge to a constant value but to a state of constant velocity $v_i = \frac{1}{N_a} \sum_{j \in \mathscr{N}_i} x_{i,2}(0)$, and

$$\lim_{t \to \infty} x_{i,1}(t) = \frac{1}{N_a} \sum_{i=1}^{N_a} x_{i,1}(0) + \frac{t}{N_a} \sum_{i=1}^{N_a} x_{i,2}(0).$$

In [220], results for nth order consensus are given, and [194, 230] study the consensus when the dynamics of each agent is an nth order linear control system. For instance, for N identical agents of the form $\dot{x}_i(t) = Ax_i(t) + Bu_i(t)$, a feedback gain K can be found so that the consensus is reached with the following control law:

$$u_i(t) = K \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t)).$$

12.2.2 Formation Control

In the past years, the formation control problem of multi-vehicle systems has attracted the attention of the control community due to its commercial and military applications. There are several approaches in the literature to distributed formation control [201]. Here the focus is on consensus-based controllers in which formations are represented by vectors of relative positions of neighboring agents.

Let us denote by $\mathbf{r}_{ij}$ the desired inter-vehicle relative position vector (see example in Fig. 12.2). For single-integrator agents, the following control law

$$u_i(t) = \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t) - \mathbf{r}_{ij}), \tag{12.5}$$

yields the group to achieve the objective of the formation [73].

According to (12.5), the overall system dynamics is given by

$$\dot{x}(t) = -Lx(t) - \mathbf{r}, \tag{12.6}$$

where $\mathbf{r}_i = \sum_{j \in \mathcal{N}_i} \mathbf{r}_{ij}$, $i = 1, \ldots, N_a$.

There are some extensions of the protocol above regarding agents dynamics. For instance, in [137], the vehicle dynamics are modeled as linear systems, and a feedback gain is derived under certain conditions.

An example of three different formations in the plane is given in Fig. 12.2. In this case, $\mathbf{r}_{ij}$ are the (x, y) coordinates of the desired distance between nodes.

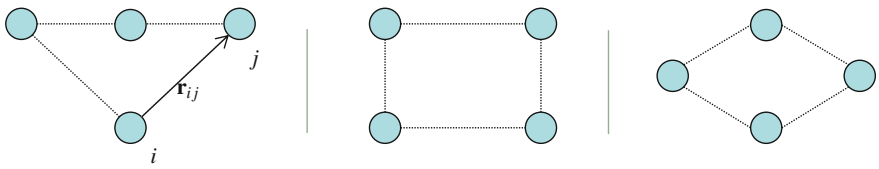

Fig. 12.2 Examples of formations of four agents in the plane

12.2.2.1 Formations with Leaders

A special situation occurs when one of the agents does not receive information from any of the others. Essentially, this means that the others are forced to arrange their positions in response to the motion of this agent, which is called the *leader* of the formation. This problem is known as *leader–follower* consensus [147]. Note that if there are multiple leaders where two of them are not coordinately moving, then the formation cannot be asymptotically reached. Hence, let us assume from now ahead that there is only one leader, if any, in the formation.

The existence of a leader makes the communication graph $\mathscr{G}$ directed by definition, and the row corresponding to the leader in L, A, and D is zero, and therefore, these matrices are not invertible. For this reason, some authors [73, 110, 194] define L, A, and D for the group of vehicles excluding the leader, and define a new diagonal matrix B, whose diagonal entries are $b_i = 1$ if agent i receives information from the leader, and $b_i = 0$ otherwise.

The leader will move according to its dynamics and initial conditions, or by a given control law, and the rest of the agents will follow it to maintain the formation. For example, if the leader moves with constant velocity v_0, $\dot{x}_0(t) = v_0$, the following protocol can be defined for the single-integrator followers:

$$u_i(t) = \sum_{j \in \mathscr{N}_i} (x_j(t) - x_i(t) - \mathbf{r}_{ij}) + b_i(x_0(t) - x_i(t) - \mathbf{r}_{i0}). \tag{12.7}$$

12.2.3 Model of Non-holonomic Mobile Robots

Single or double integrators do not describe properly the dynamics of most of commercial mobile robots, since these cannot move in any direction instantaneously. In robotics, holonomicity refers to the relationship between the controllable and total degrees of freedom of a given robot. If the controllable degrees of freedom are fewer than the total degrees of freedom, the vehicle is non-holonomic.

The non-holonomic model of a mobile robot is depicted in Fig. 12.3. The distance between the back and the front wheels is denoted by d. It is assumed a single front wheel in this case.

The equations of motion are given by [218]

$$\begin{aligned} \dot{x}_i &= v_i \cos\theta_i \\ \dot{y}_i &= v_i \sin\theta_i \\ \dot{\theta}_i &= \omega_i \\ m_i \dot{v}_i &= F_i \\ J_i \dot{\omega}_i &= T_i \end{aligned} \tag{12.8}$$

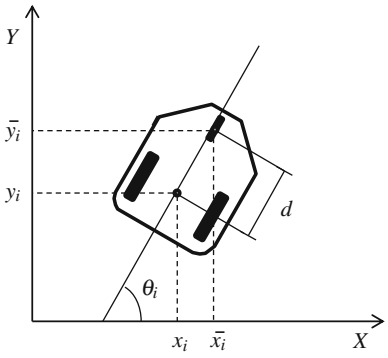

Fig. 12.3 Non-holonomic mobile robot

where (x_i, y_i) is the position in the plane of agent i, θ_i is the orientation, v_i is the linear velocity, ω_i is the angular velocity, F_i is the force, T_i is the torque, m_i is the mass, and J_i is the mass moment of inertia.

To avoid the non-holonomic constraint introduced by (12.8), let us define

$$\begin{aligned} \bar{x}_i &= x_i + d\cos\theta_i \\ \bar{y}_i &= y_i + d\sin\theta_i, \end{aligned} \tag{12.9}$$

according to Fig. 12.3.

As proposed in [138], the dynamics of the mobile robot can be reformulated in terms of these coordinates as

$$\begin{pmatrix} \dot{\bar{x}}_i \\ \dot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix} \begin{pmatrix} v_i \\ \omega_i \end{pmatrix}. \tag{12.10}$$

If v_i and ω_i are considered as the control inputs, the dynamics of the mobile robot is modeled by a first-order model. Alternatively, second time derivatives can be computed to produce a second-order model:

$$\begin{pmatrix} \ddot{\bar{x}}_i \\ \ddot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} -v_i\omega_i\sin\theta_i - d\omega_i^2\cos\theta_i \\ v_i\omega_i\cos\theta_i - d\omega_i^2\sin\theta_i \end{pmatrix} + \begin{pmatrix} \frac{1}{m}\cos\theta_i & -\frac{d}{J}\sin\theta_i \\ \frac{1}{m}\sin\theta_i & \frac{d}{J}\cos\theta_i \end{pmatrix} \begin{pmatrix} F_i \\ T_i \end{pmatrix}, \tag{12.11}$$

where F_i and T_i are the control inputs.

Therefore, the formation control problem is formulated as follows: design control laws so that the formation is reached while applying a consensus-based coordination scheme. According to this, in the next pages, a control law is designed for the first-order model (12.10). After this, the example proposed in [218, 231] to control the second-order model (12.11) is presented.

12.2.3.1 Formation Control of First-Order Non-holonomic Mobile Robots

The control law $u_i = (v_i, \omega_i)$ in (12.10) to reach the desired formation is

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{j\in\mathcal{N}_i}(\bar{x}_j - \bar{x}_i - (r_{x,j} - r_{x,i})) \\ \sum_{j\in\mathcal{N}_i}(\bar{y}_j - \bar{y}_i - (r_{y,j} - r_{y,i})) \end{pmatrix}, \tag{12.12}$$

where $(r_{x,i}, r_{y,i})$ are the predefined relative position offsets with respect to the formation center. From (12.10) and (12.12), it follows that

$$\begin{pmatrix} \dot{\bar{x}}_i \\ \dot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} \sum_{j\in\mathcal{N}_i}(\bar{x}_j - \bar{x}_i - (r_{x,j} - r_{x,i})) \\ \sum_{j\in\mathcal{N}_i}(\bar{y}_j - \bar{y}_i - (r_{y,j} - r_{y,i})) \end{pmatrix}. \tag{12.13}$$

If stack vectors for the overall system are defined as $\bar{x}^T = (\bar{x}_1 \dots \bar{x}_{N_a})$, $\bar{y}^T = (\bar{y}_1 \dots \bar{y}_{N_a})$, $r_x^T = (r_{x,1} \dots r_{x,N})$, and $r_y^T = (r_{y,1} \dots r_{y,N_a})$, it yields

$$\begin{pmatrix} \dot{\bar{x}} \\ \dot{\bar{y}} \end{pmatrix} = -\begin{pmatrix} L & \mathbf{0} \\ \mathbf{0} & L \end{pmatrix} \begin{pmatrix} \bar{x} - r_x \\ \bar{y} - r_y \end{pmatrix}. \tag{12.14}$$

Note that the control law (12.12) decouples the system, giving an equivalent results to (12.6) for single integrator. The group of robots reaches, the formation and the center of this formation is the average of the robots initial positions.

12.2.3.2 Formation Control of Second-Order Non-holonomic Mobile Robots

According to [218], the following feedback linearization can be used in order to transform the dynamics (12.11) to two decoupled double integrators

$$\begin{pmatrix} F_i \\ T_i \end{pmatrix} = \begin{pmatrix} \frac{1}{m}\cos\theta_i & -\frac{d}{J}\sin\theta_i \\ \frac{1}{m}\sin\theta_i & \frac{d}{J}\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} v_i\omega_i\sin\theta_i + d\omega_i^2\cos\theta_i + \bar{F}_i \\ -v_i\omega_i\cos\theta_i + d\omega_i^2\sin\theta_i + \bar{T}_i \end{pmatrix}, \tag{12.15}$$

which yields to

$$\begin{pmatrix} \ddot{\bar{x}}_i \\ \ddot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} \bar{F}_i \\ \bar{T}_i \end{pmatrix}.$$

The control law (12.4) can be extended to the formation control problem, giving the following coordination rule for the group of mobile robots:

$$\begin{pmatrix} \bar{F} \\ \bar{T} \end{pmatrix} = -\begin{pmatrix} L & \gamma_x L & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & L & \gamma_y L \end{pmatrix} \begin{pmatrix} \bar{x} - r_x \\ \dot{\bar{x}} \\ \bar{y} - r_y \\ \dot{\bar{y}} \end{pmatrix}, \tag{12.16}$$

where $\gamma_x, \gamma_y > 0$, $\bar{F} = (\bar{F}_1 \dots \bar{F}_N)^T$, and $\bar{T} = (\bar{T}_1 \dots \bar{T}_N)^T$.

The center of the formation depends on the robots initial positions, and the group moves with a velocity equal to the average of the initial velocities.

12.2.4 Time-Schedule Control

The formation control laws (12.12) and (12.16) have been proposed for the first- and second-order models, respectively, of non-holonomic mobile robots. However, these control laws require the continuous measurement of the robot and the neighbors state, which is not achievable in practice, as we have already discussed.

The most common approach is to set a periodic scheduling of measurement samplings, control updates, and broadcasting over the network. Alternatively, event-triggering policies for multi-agent systems [52, 232] can be adapted to the formation control problem by redefining the previous control laws as a function of last broadcast states:

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{j\in\mathcal{N}_i} (\bar{x}_{b,j} - \bar{x}_{b,i} - (r_{x,j} - r_{x,i})) \\ \sum_{j\in\mathcal{N}_i} (\bar{y}_{b,j} - \bar{y}_{b,i} - (r_{y,j} - r_{y,i})) \end{pmatrix} \tag{12.17}$$

$$\begin{pmatrix} \bar{F}_i \\ \bar{T}_i \end{pmatrix} = \begin{pmatrix} \sum_{j\in\mathcal{N}_i} \left(\hat{\bar{x}}_j - \hat{\bar{x}}_i - (r_{x,j} - r_{x,i}) + \gamma_x(\dot{\bar{x}}_{b,j} - \dot{\bar{x}}_{b,i})\right) \\ \sum_{j\in\mathcal{N}_i} \left(\hat{\bar{y}}_j - \hat{\bar{y}}_i - (r_{y,j} - r_{y,i}) + \gamma_y(\dot{\bar{y}}_{b,j} - \dot{\bar{y}}_{b,i})\right) \end{pmatrix}, \tag{12.18}$$

where

$$\begin{aligned} \hat{\bar{x}}_i &= \bar{x}_{b,i} + (t - t_k^i)\dot{\bar{x}}_{b,i} \\ \hat{\bar{y}}_i &= \bar{y}_{b,i} + (t - t_k^i)\dot{\bar{y}}_{b,i}, \end{aligned}$$

and $\bar{x}_{b,i}$, $\bar{y}_{b,i}$, $\dot{\bar{x}}_{b,i}$, and $\dot{\bar{y}}_{b,i}$ are the last broadcast values of $\bar{x}_i$, $\bar{y}_i$, $\dot{\bar{x}}_i$, and $\dot{\bar{y}}_i$, respectively, and t_k^i refers to the last broadcasting time of the robot i. The position $(\bar{x}_i, \bar{y}_i)$ is approximated by $(\hat{\bar{x}}_i, \hat{\bar{y}}_i)$ in (12.18), as proposed in [231]. This can be assimilated to a model-based estimation.

The occurrence of an event is determined by trigger functions, where the error of the robot i is defined as

$$\varepsilon_i = \begin{pmatrix} \varepsilon_{x,i} \\ \varepsilon_{y,i} \end{pmatrix} = \begin{pmatrix} \bar{x}_{b,i} - \bar{x}_i \\ \bar{y}_{b,i} - \bar{y}_i \end{pmatrix}$$

for the first-order dynamics, and for the second-order systems as

$$\varepsilon_i = \begin{pmatrix} \varepsilon_{x,i} \\ \gamma_x \varepsilon_{\dot{x},i} \\ \varepsilon_{y,i} \\ \gamma_y \varepsilon_{\dot{y},i} \end{pmatrix} = \begin{pmatrix} \hat{\bar{x}}_i - \bar{x}_i \\ \gamma_x (\dot{\bar{x}}_{b,i} - \dot{\bar{x}}_i) \\ \hat{\bar{y}}_i - \bar{y}_i \\ \gamma_y (\dot{\bar{y}}_{b,i} - \dot{\bar{y}}_i) \end{pmatrix}.$$

12.2.5 Robot Wireless Communication Protocols

In early robot wireless communications , infrared technology was applied in a large scale because of its low cost. Since infrared waves cannot pass through obstacles, the communication rate using this technology is poor and the transmission reliability low. Currently, radio-frequency (RF) technology has become the preferred in the design of mobile robot communication systems. Robots can communicate with others by RF point-to-point links or broadcasting mechanisms. The proliferation of Internet-like networks has motivated the research to address wireless LAN (IEEE 802.11), Bluetooth standards, and ad hoc networking in mobile robot systems.

The main features of these three wireless communication technologies for mobile robot communications are illustrated in Table 12.1. Wi-Fi (the brand name for products following IEEE 802.11 standards) uses the same radio frequencies as Bluetooth, but with higher power, resulting in higher bit rates and better range from the base station. The nearest equivalents to Bluetooth are the DUN (dial-up networking) profile, which allows devices to act as modem interfaces, and the PAN (personal area network) profile, which allows for ad hoc networking.

A wireless communication link is characterized by long bandwidth delay, dynamic connectivity, and error-prone transmission. The robots are often equipped with low-cost low-power short-range wireless network interfaces, which only allow direct

Table 12.1 Wireless communication technologies for mobile robots

	Infrared	IEEE 802.11b/g/n	Bluetooth
Band (GHz)		2.4/2.5	2.4/2.5
(Up to) Data-rate (Mbps)	0.1–0.4	11/54/150	1–3
Range (m)	4	140–250	5–100
Power (W)	5E-3	0.4–4	1E-3-0.1
Network structure	PPP	Infrastructure and ad hoc	Ad hoc

communication with their near neighbors. Hence, it is virtually impossible for each node to know the entire network topology at a given time [260].

Moreover, many in the field on networking argued that Internet protocols were not convenient to achieve robustness and scalability for such distributed architecture [134], and there has been a proliferation of new protocols, plenty of good ideas from the academic and commercial domains but with few impact in the real world. Examples of routing protocols for mobile ad hoc networks are ad hoc on-demand distance vector (AODV) [209] and dynamic source routing (DSR) [123], while low-energy adaptive clustering hierarchy (LEACH) [103] is a cluster-based protocol that includes distributed cluster formation and a hierarchical clustering algorithm. Finally, routing protocol for low power and lossy networks (RPL) [262] is an IP-based protocol for this kind of networks.

Three important features usually serve to evaluate the performance of a protocol [160]:

- **Energy efficiency** Low energy consumption is a major objective for battery equipped devices.
- **End-to-end reliability** Reliability is measured as the packet delivery ratio from each transmitter to the destination. A maximization of the reliability may require a large number of packet overhead and retransmissions, thus increasing the energy consumption.
- **End-to-end delay** At network layer, delay is computed for successfully received packets at the destination. A minimization of the delay requires a high utilization of the transmission resources and a very low duty cycling between nodes, thus requiring high energy expenditure.

Hence, there is a trade-off between latency, packet losses, and energy consumption in the protocol design. Moreover, when the protocol is devoted to control applications, it must guarantee the stability of the control system. Despite the proposal of numerous routing protocols for energy efficient wireless networks, there is not yet a definite solution.

12.3 Interactive Simulation Tools

The many control and system configuration options needed to simulate a multi-agent system demand graphical user interfaces (GUIs) with high degree of interactivity and flexibility. The GUI designed in this work is intended to make rapid prototyping and simulation of wireless autonomous agents which execute distributed control algorithms and perform event-based communications.

The simulator allows users to define the characteristics of the network and test the control algorithm and the triggering mechanism that rules the control updates under many possible scenarios before implementing them into a real platform of networked robots. Nevertheless, the simulator has been designed keeping the interaction with

the user relatively simple and intuitive in order to also be used as a pedagogical tool for advanced engineering control courses.

For this purpose, Easy Java Simulations (EJS) was chosen for the development of the simulation platform. EJS is a free software tool that helps to create dynamic, interactive scientific simulations in Java language and which offers a high degree of flexibility as well as high-level graphical capabilities and an increased degree of interactivity [69]. EJS is based on an original simplification of the *model-view-control* paradigm, structuring the simulation into two main parts: the model and the view. The model describes the behavior of the system using variables, ordinary differential equations, and Java code. The view is intended to (1) provide a visual representation of the more relevant properties of the model and its dynamic behavior; and (2) facilitate the user's interaction on the model. Additional libraries in Java can also be imported.

In this section, a short background of other existing tools is given. After this, the simulator is described, including the GUI and the system modeling with EJS.

12.3.1 Existing Tools

Most of the simulators for the formation control of a team of networked vehicles/robots use different softwares to emulate the real control and the network counterparts, which are connected and synchronized afterwards.

Some companies dedicated to the design and manufacture of autonomous mobile robotic systems provide the software to simulate their products. For example, MobileSim [250] is an open-source Amigobot and Pioneer simulator [180], also provided by Mobile Robots Inc. It has a customizable interface for users to design and simulate different models of MobileRobots/ActivMedia robots [180]. However, most operations are run through commands and the supported protocols are specific for these robots.

Also in the academical world, some research groups have developed software to compare experimental test beds. Because the MATLAB/Simulink is a well-known environment in the control and communication community, it is frequently present in these developments. For example, in PiccSIM [192], the dynamics and the control algorithm are implemented in Simulink, and ns-2 [197] is used for the simulation of the network. In [162], we can find another example. In this case, the software is produced with the MATLAB Virtual Reality Markup Language (VRML) Toolbox. Outside the robotic community, NetMarSys is a specific simulator for networked marine vehicles [246] and it is also based on MATLAB.

All of these tools have a common characteristic: the lack of interactivity and flexibility. Although they usually have a GUI, most of the operations carried out by the user are through commands, or the change of the parameters requires to restart the simulation and run it again.

On the contrary, the interactivity provided by the proposed simulator allows user to immediately appreciate the effect of any change in the control or the network

counterparts over the system. Moreover, when more than one software tool needs to be installed, the communication between them becomes a tough problem and the final user has to spend some time installing and synchronizing them. The main advantage of integrating all in one tool is that it is easy to study all aspects of communication and control in networked robots, including the interaction between them.

12.3.2 Description of the GUI

The user interface of the simulator is shown in Fig. 12.4. It has five main panels, a menu bar, and a small task bar. The two upper panels of the interface provide a quick view of the multi-agent system and a time plot of the output and control signals. The top left panel (No. 3) shows an animation of the complete multi-agent system. Each agent is numbered and shows a trace of its trajectories. Network links are depicted as arrows between agents.

By default, the links provide bidirectional communication, although one-way communication is also allowed. So, in Fig. 12.4, it is simulated a multi-agent system with four robots linked by three bidirectional links: $0 \leftrightarrow 1$, $1 \leftrightarrow 2$, $2 \leftrightarrow 3$. Finally, this panel allows speeding up or down the simulation by dragging the slider *Simulation vel*.

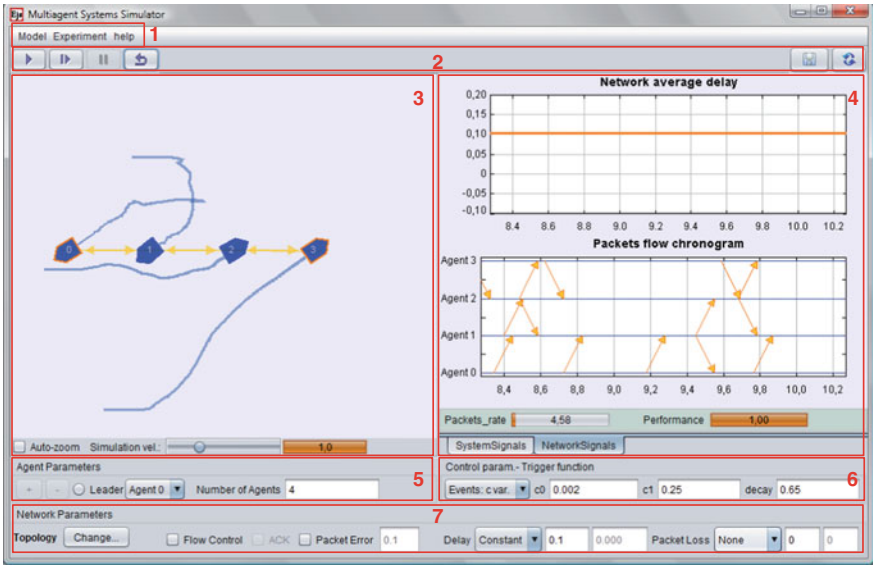

Fig. 12.4 View of the GUI

The lower left panel, named *Agent Parameters* (No. 5), allows users to set the number of agents in the system, as well as to add and remove a particular agent. Using this panel, it is also possible to set an agent as leader, which means that the agent can be moved (i.e., dragged by the user) freely in the coordinate system, and the rest of the agents moves to keep the desired formation. The links between the leader and its neighbors are changed to unidirectional links automatically.

The two time plots on the top right panel (No. 4), which are grouped in the *System Signals* tab, display the relative distance to the desired formation as well as the control actions of each agent. There are also plots grouped in the *Network Signals* tab (shown in Fig. 12.4), which provide mainly information about the dispatched and arrival time of the packets. The average network delay is also shown in this tab.

The lower panel, named *Network Parameters* (No. 7), is devoted to configure the behavior of the network. Users can choose the drop-down list *Delay* to set a constant or variable network delay. Packet loss probability can be set using the drop-down list *Packet Loss*. The topology of the network (bidirectional or unidirectional links) can be changed after pressing the button *Topology* and clicking on the agents to be connected. Advanced network functionalities, such as flow control or automatic acknowledgment packets, can be set as well. Additionally, the user can configure a bit error rate in the transmission of packets.

The lower right panel *Control Parameters* (No. 6) is used to specify the time-scheduling communication and control. This option specifies the conditions that trigger the sending of packets from one agent to its neighbors in order to update the control actions. The events can be triggered periodically or when the position of an agent has changed and it is greater than a threshold (send on delta policy).

The components of the interface described earlier provide the basic functionality required to operate the application. However, there are also advanced options available in the menu bar (No. 1) which allows some additional features such as the possibility to:

- Specify the dynamic model of the agents (first order, second order) and define coupling terms which dynamically couple neighboring agents.
- Select a predefined multi-agent system configuration to perform with the simulator.
- Load and save user-defined multi-agent system configurations.

The interface is completed with a top task bar (No. 2) that provides buttons to start, pause, and reset the simulation. Finally, there is a button to save the state variables of the agents and the system configuration in MATLAB language to perform further analysis of a simulation.

12.3.3 Modeling a Multi-agent System in EJS

The model of each node in the multi-agent system is basically the same described in Chap. 7 (Fig. 12.5) for interconnected systems, but specifically assuming that the information is sent through the network in packets of a given structure, which is detailed later.

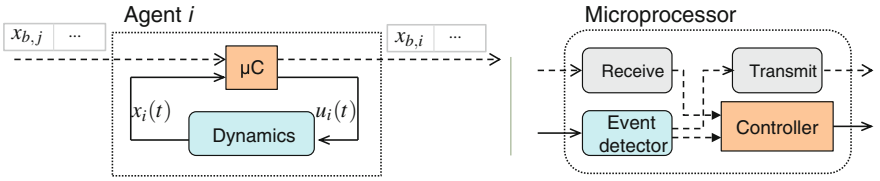

Fig. 12.5 Scheme of one node

Hence, the simulator has to implement the following:

- The system dynamics, including the agent dynamics and the topology of the system.
- The tasks performed by the microprocessor, i.e., deciding when to broadcast the state, computing the control law and transmitting and receiving the packets through the network.
- The network itself, that is, the process of transmitting information from one node to another, taking into account the properties defined by the user.

We next describe the implementation of the three aspects mentioned above.

12.3.3.1 The System Dynamics

EJS provides an interface to define the dynamics of a system through differential equations. Moreover, we can specify the dynamics of a set of entities. Several pages of differential equations are allowed but only one can be enabled at a given time instant. It is the programmer task to take care of possible inconsistencies when switching from one dynamics to another.

Example 12.2 Figure 12.6 shows the EJS pages where the dynamics of the multi-agent system is defined. The page above, which is enabled by default, corresponds to the first-order model (12.10). The page for the second-order model (12.11) is shown below. When the user changes the model of the agents through the GUI, a method is executed to enable the selected dynamics and to disable the old ones. It also captures the current time as the initial time of the new experiment and resets the control inputs.

Besides the dynamics of the agents, the communication graph is initialized to a default value and it is updated when a new experiment is selected or new links are added/removed by the user.

12.3.3.2 The Microprocessor

The first task that the microprocessor performs is the detection of events by monitoring the state of the agent (event-based policies) and the internal clock (periodic

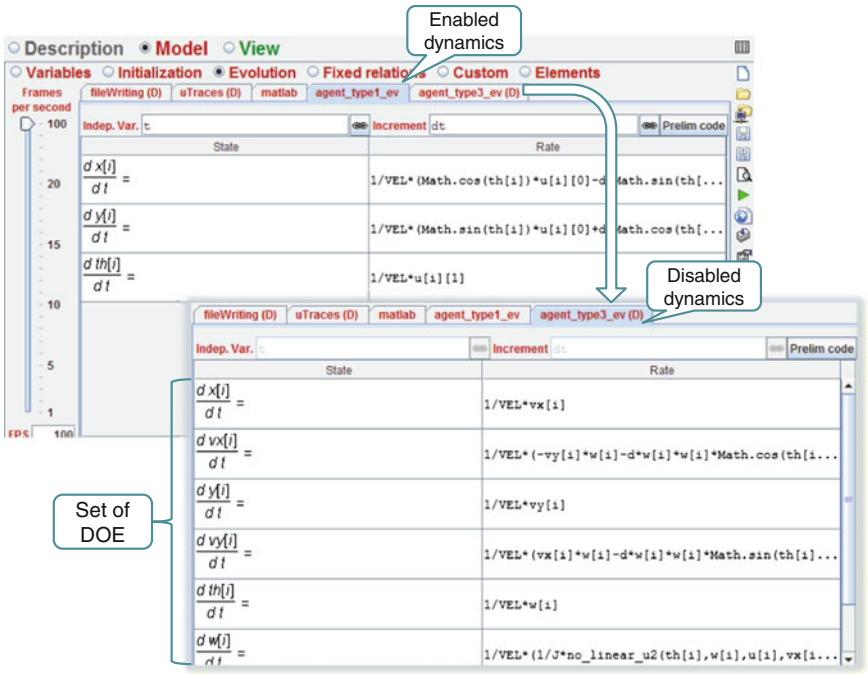

Fig. 12.6 Screenshot of *Evolution pages* in EJS

triggering). EJS provides specific pages to define the events detection and the routine to execute after the triggering. This routine is executed at the time of the detection of the event, i.e., the simulation is "stopped" until the procedure is completed.

Listings 12.1 and 12.2 show an example for first-order dynamics. An event is detected when the variable `tol` reaches the zero value. If the transmission policy is event-triggered policy, `tol=0` when `e[i]` ($\|\varepsilon_i\|$) reaches the threshold `d` (δ). Note that in the periodic case the event to detect is the time instance that equals the next sampling time (line 11 in Listing 12.1), where `incr` is an internal count and `tpo` is the time at which the periodic sampling started.

The broadcast state of an agent i to a neighbor j is denoted as (`x_b[i][j]`, `y_b[i][j]`). Hence, (`x_b[i][i]`,`y_b[i][i]`) refers to the last-broadcast state of the agent i, which immediately updates the value. The value of (`x_b[i][j]`, `y_b[i][j]`) may neither be the same in different neighbors nor equal to (`x_b[i][i]`, `y_b[i][i]`) at a given time for unreliable networks.

The second task executed by the microprocessor is the computation of the control law, i.e., (12.12) for the first-order model and (12.15) for the second-order model. The update of the control law is performed only at event times (line 3 in Listing 12.2) and holds constant between events.

Finally, the microprocessor is in charge of encapsulating the data into a packet, sending it over the network, reading the incoming packets, and extracting the data.

Listing 12.1 Code to detect events

```
/* Event-triggering */
if(control_type.equals("Events: d cte")||control_type.equals("
    Events: d var."))
{
  d=(d0+d1*Math.exp(-beta*(t-ti)));
  e[i]=computeError(x_b[i][i],x[i],y_b[i][i],y[i]);
  tol=d-e[i];
}
/* Periodic sampling. Parameter d is reconverted to Ts */
else {
    d=d0;
    tol=incr*d-(t-tpo);
}
return tol;
```

Listing 12.2 Routine of treatment of events

```
x_b[i][i]=x[i];
y_b[i][i]=y[i];
u[i]=control1(i,x_b[i],y_b[i],th,x_ref,y_ref);
broadcast(i);
t_b[i]=t;
colorTransmission[i]=new java.awt.Color(255,128,0);
N_events=N_events+1;
if(control_type.equals("Periodic"))
  incr=incr+1;
```

The structure of a packet is shown in Fig. 12.7. The *header* field identifies the packet and contains the sender and the receiver address devices. The *payload* contains the data to transmit (state and time stamp). The *frame control* and *check sequence* are not used in this implementation. For each packet, the simulator associates a value to the delay, determines if the packet is successfully transmitted or not, and corrupts the data in the packet with a given probability. This is represented on the right of Fig. 12.7.

Moreover, if the *Acknowledgment* signal is required, the receiver sends an ACK packet to confirm the reception. The ACK packets are assumed to be always delivered with a short delay (10 ms) due to its small size. If an ACK is not received before a given *waiting time*, the packet is treated as lost, but not retry occurs and the agent will send a new packet after the occurrence of a new event.

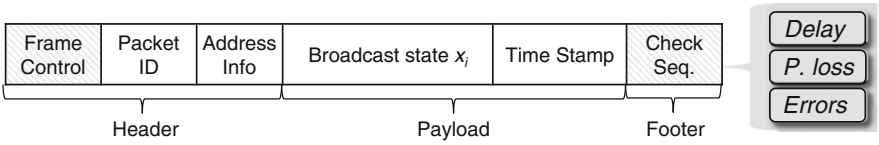

Fig. 12.7 Structure of a data packet

Listing 12.3 Code to compute the next reception time of a packet (lines 2–9) and to execute after the detection of an event (lines 11–16)

```
1 /* Code to detect the time of the next packet reception time */
2 double t_min0=100+t;
3 t_min=t_min0;
4 double tol=0.1;
5 t_min=getNextPacketTime();
6 if (t_min<t_min0) {
7   tol=-(t-t_min);
8 }
9 return tol;
10 /* Routine executed when an event is detected */
11 NpendPacket=getPacketsToAttend();
12 int dim_st=2;
13 for(int i=0;i<NpendPacket;i++) {
14   processPacket(i,dim_st);
15   prepareNextReception(i);
16 }
```

12.3.3.3 The Network

The simulator implements the network as a collection of *buffers*. The packets are stored in these data structures and the reaching to the destination is also handled by EJS events. A class named as *packet* and the corresponding methods have been implemented to encapsulate the communications functions.

Short examples of code are given in Listing 12.3. In lines 2–9, the code to detect the next network event is shown. When the simulation time `t` reaches the value `t_min`, the routine of treatment of events is executed (lines 11–16). The variable `dim_st` refers to the dimension of the state, which is 2 in this example (first-order model).

More than one network event may need to be handled at a given time, for instance, if the network delay is set to be constant or zero, a broadcast state should be received at the same time in all the neighbors. In this case, the requests are processed sequentially though from the simulation time point of view all receptions are simultaneous, preserving the distributed architecture of the system.

Listing 12.4 shows extracts of the functions `getNextPacketTime()` (line 5 in Listing 12.3) and `getPacketsToAttend()` (line 11 in Listing 12.3). Each receiver agent j has a buffer of a given capacity `buffer_cap`, in which the packets are virtually stored until they are processed or discarded.

12.3.4 Using the Simulator

We next describe some examples of how to use the simulator.

Listing 12.4 Extracts of code of getNextPacketTime() and getPacketsToAttend()

```
/* Begin of getNextPacketTime */
double getNextPacketTime () {
  double t_min=100+t;
  ...
  int i_buffer=0;
  while(i_buffer<buffer_cap) {
    p2=nextPacket(buffer[j]);
    if(p2!=null && ((packet)p2).getTS()!=-1.0) {
       if((((packet)p2).getDelay()+((packet)p2).getTS())<t_min)
           {
           t_min=((packet)p2).getDelay()+((packet)p2).getTS();
       }
    }
    i_buffer+=1;
  }
  ...
  return t_min;
}
/* Begin of getPacketsToAttend() */
int getPacketsToAttend() {
  int NpendPackets=0;
  ...
  p2=nextPacket(pendPackets[j]);
  if(p2!=null && ((packet)p2).getTS()!=-1.0) {
     if((((packet)p2).getDelay()+((packet)p2).getTS())==t_min) {
        i_min[NpendPackets]=i;
        j_min[NpendPackets]=j;
        NpendPackets+=1;
    }
  }
  ...
  return NpendPackets;
}
```

12.3.4.1 Experiment 1: Bidirectional Communication Links, Constant Delay, First-Order Model

Let us assume that the system goes from an initial configuration to the formation of Fig. 12.4 with the following bidirectional links: $0 \leftrightarrow 1$, $1 \leftrightarrow 2$, $2 \leftrightarrow 3$.

In this case, the communication is bidirectional and delayed 100 ms in every link. The trigger mechanism is defined as $F_{e,i}(t, \varepsilon_i(t)) = \|\varepsilon_i(t)\| - 0.002 - 0.25e^{-0.65t}$, which means that the threshold to trigger the events is not constant and decrease with time. Thus, large errors are allowed when the robots are far away from the desired formation, but when they get close, events are triggered to prevent from stationary errors.

The chronogram at the bottom of the right-hand side of Fig. 12.4 reflects the described characteristics of the links. For example, at time 8.5 s, Agent 2 broadcasts its state to its neighbors, which are Agent 3 and 1 from the definition of the topology. The orange arrows represent packets correctly delivered. Because the probability of losing a packet is zero, all packets are delivered. Moreover, all arrows have the same slope since the delay of the network is constant and equal to 100 ms.

12.3.4.2 Experiment 2: Directed Graph, Random Delay, Packet Losses, Second-Order Model

Let us change the topology of the network removing the link $1 \rightarrow 2$ and adding the links $1 \rightarrow 3$, $3 \rightarrow 0$. This can be done *online* by clicking on the button *Change...*, then on the two agents involved in the link, and finally selecting *Delete* or *Add*.

Let us give a value of 20 % to the probability of losing a packet, define the delay as random with maximum value of 300 ms, and set to true the checkbox of *Flow control*. Moreover, let us change the dynamics of the vehicles to a second-order nonholonomic model, and select a different experiment to change the desired formation. Figure 12.8 shows the packets flow chronogram in a time slot.

The red arrows correspond to lost packets. For example, at time 17.35 s, Agent 1 sent a packet to its neighbor 0, but for some reason the communication link was broken. Note that defining a packet loss probability gives a time-varying topology. Packet losses can be caused by interference with other networks, the presence of obstacles or packet collisions, and these losses have a direct effect over the control performance. Note that because we have changed the topology, Agent 1 does not transmit information to Agent 2 (as in Fig. 12.4), but to Agent 3; and Agent 3 does so to Agent 0.

The third type of arrow is green colored and corresponds to packets arrived correctly but discarded because they contained old information. In Fig. 12.8, Agent 2 discards one packet from Agent 3 at time 16.1 s because it has already received and measured from 3 which was taken later on time. Other packets are discarded at time 17.54 and 17.62 s.

12.3.4.3 Experiment 3: Designating a Leader

An interesting experiment is the behavior of the system when there is a leader in the group. A leader determines by itself the actions to take, which from the communica-

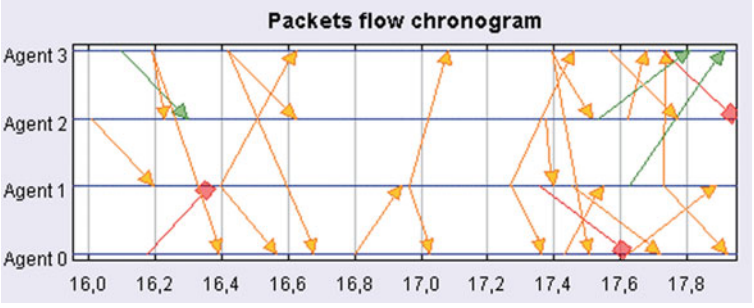

Fig. 12.8 Example of chronogram. Delivered packets are in *orange*, *red arrows* are lost packets, and *green arrows* correspond to discarded packets

tion point of view means that it sends its state to its neighbors but it does not receive information from them.

Thus, once the system has reached the formation aforementioned, we add a new agent, labeled as 4, and we define it as leader of the group (see Fig. 12.9a). Figure 12.9b, c, d depict the animation of how the other agents move to get the formation around the leader at different instants of time. If you look at the *red cross*, you see that the leader did not move. Note that at the beginning, the four agents were spatially distributed in a circle of a given radius, which seems as a square. Because we add a new agent, the desired formation changes to a pentagon to keep the agents equidistant between each other.

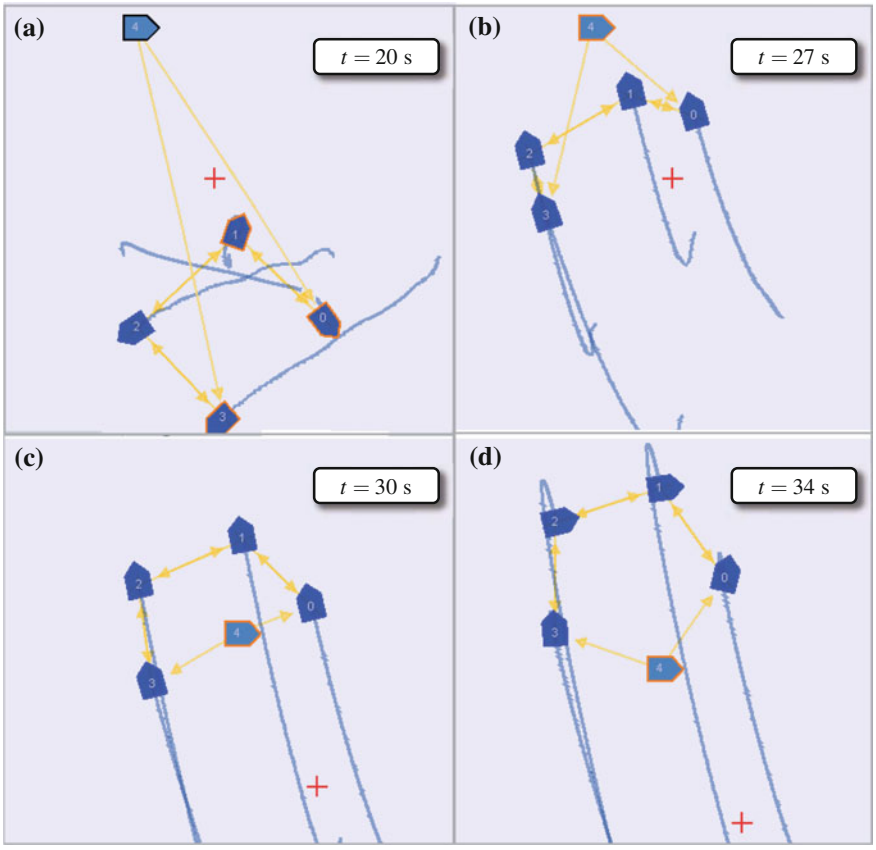

Fig. 12.9 Screenshots of experiment 3 at different instants of time

12.3.4.4 Experiment 4: Save Data to MATLAB File

Another interesting capability of the simulator is to store the data in a MATLAB file to further analysis. This is very useful when other same experiment is performed to the system under the variation of a parameter and compare the performance afterward. So, as an example, let us consider the following scenario:

- **Agents' model** Four first-order non-holonomic vehicles.
- **Desired formation** A square of 0.85 u. side.
- **Communication method** Event-triggered communication with constant threshold $\delta_0 = 0.025$.
- Acknowledgment of packets.
- **Network delay** Constant, 100 ms,

and let us analyze the results when (1) packets dropouts are modeled as a Bernoulli distribution of probability $p = 0.1$, and (2) packet dropouts are influenced by transmission rate assuming that when this rate increases so does the probability of packet collisions.

The results are presented in Figs. 12.10 and 12.11. The trajectories of the agents for both cases are depicted in Fig. 12.10. The initial and final positions are marked with crosses and circles, respectively. The blue lines correspond to Case 1 and the red lines to Case 2. Observe that in both cases the formation is reached, though the final positions are different because the consensus algorithm only preserves relative distances and the final absolute positions depend on initial conditions and disturbances.

In Fig. 12.11a, the distance to the formation $d_i = \sqrt{(\bar{x}_i - r_{x,i})^2 + (\bar{y}_i - r_{y,i})^2}$ is depicted. Observe that in Case 2 there is an oscillatory behavior and the formation is reached later due to the degradation of the network performance (Fig. 12.11c). The network performance is defined as the ratio of delivered to sent packets.

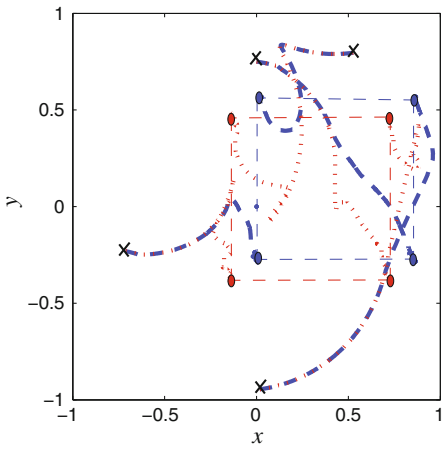

Fig. 12.10 MATLAB figure corresponding to the trajectories of the agents in the experiment 4

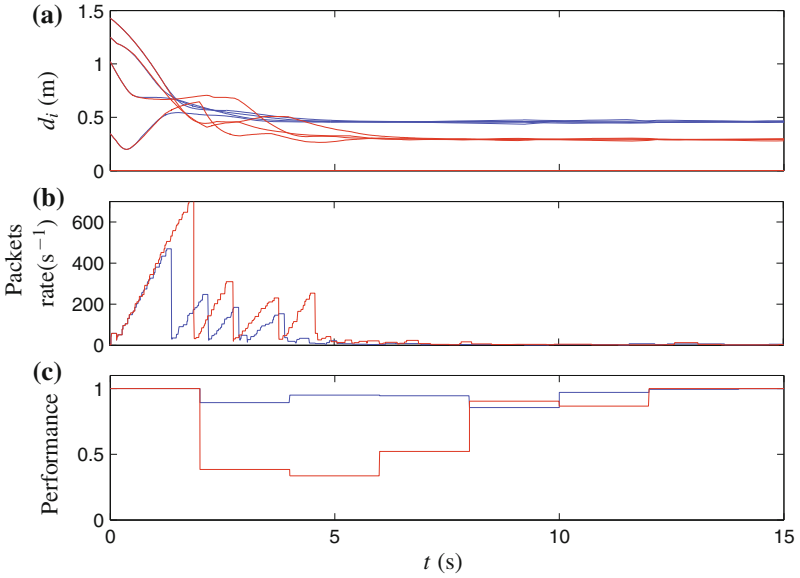

Fig. 12.11 Distance to the formation, packets rate, and performance corresponding to experiment 4 (MATLAB figure)

The packet transmission rate is shown in Fig. 12.11b. One can conclude that the degradation of the performance of the network has a direct effect into the performance of the system.

12.4 Application Example to a Real Test bed

The distributed event-based control has been implemented in a test bed of real mobile robots. The prototypes taken for experimentation are the *Moway* robots [186], which are built on low-cost components but still have high potential for experimentation in an educational environment.

The experimental framework and the experimental results are presented next.

12.4.1 Experimental Framework

The experimental framework used to test DEBC is part of a remote laboratory developed to teach robotics. A full description of this laboratory can be found in [70].

12.4.1.1 Moway Mobile Robots

Moway robots are autonomous small programmable robots mainly designed to develop practical applications in an educational environment. The components of these robots are a microcontroller, two independent servo motors, a battery, a light sensor, a temperature sensor, four infrared sensors, two infrared line sensors, four LEDs diodes, three-axis accelerometer, a speaker tone generator, and a microphone. All these peripherals are connected to the microcontroller responsible for governing the robot [186]. Another important component is the RF module that enables wireless communication with other RF devices.

The main handicap of these prototypes is the need of an external device to measure the position and orientation. In order to overcome this problem, additional components are required. The description of these elements is provided next.

12.4.1.2 Measurement and Communication Systems

Measurement and communication systems are depicted in Fig. 12.12. Several hardware and software components exchange information to perform both tasks:

- **CCD camera** It is installed on the ceiling of the laboratory, and it captures the video that will be processed by a software tool to determine the robots poses. It is connected to a computer via a *FireWire* port.
- **SwissTrack** This application is an open-source tool developed at EPFL to track objects using a camera or a recorded video as input source [47, 153]. Hence, the values of x_i, y_i, and θ_i are determined by this software, which processes the

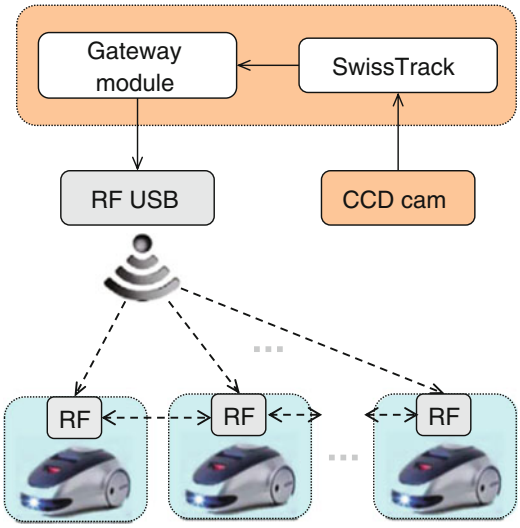

Fig. 12.12 Experimental framework block diagram. *Dotted lines* represent wireless communications, and the exchange of information between hardware and software components is symbolized by *solid lines*

incoming images from the CCD camera. This information is retrieved via TCP/IP in form of packets.

- **Gateway Module** This application is running at the same computer than SwissTrack, and it is in charge of processing the measurements and sending the data through the RF USB device to the robots.

This architecture emulates the position sensors of the robots. Each of them receives its state, sends it to its neighbors when required, and computes the control law (v_i and ω_i).

12.4.2 Experimental Results

12.4.2.1 Experiment 1: Consensus Protocol

Description of the Experiment

Let us consider four mobile robots, labeled as 2, 3, 4, and 5. The communication topology as well as the initial and desired formation is depicted in Fig. 12.13. The graph has directed links, but the consensus can be reached, and hence, the formation, because there is a directed path connecting any two arbitrary nodes of the graph [201]. The initial conditions are

$$\begin{aligned} x(0) &= (0\ 60\ -50\ 0)^T \\ y(0) &= (0\ 0\ 0\ 40)^T \\ \theta(0) &= (198\ 280\ 262\ 179)^T, \end{aligned}$$

and the components of the relative position offsets vector (r_x, r_y) are

$$\begin{aligned} r_x &= (-20\ -20\ 20\ 20)^T \\ r_y &= (-20\ 20\ 20\ -20)^T. \end{aligned} \tag{12.19}$$

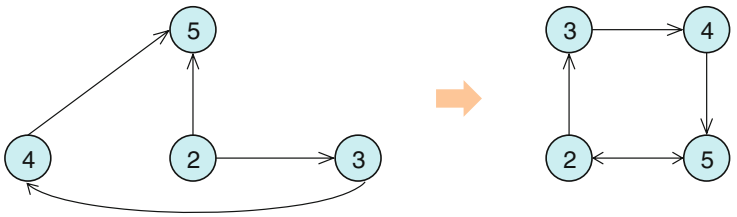

Fig. 12.13 Scheme of the communication topology, initial formation (*left*) and desired formation (*right*)

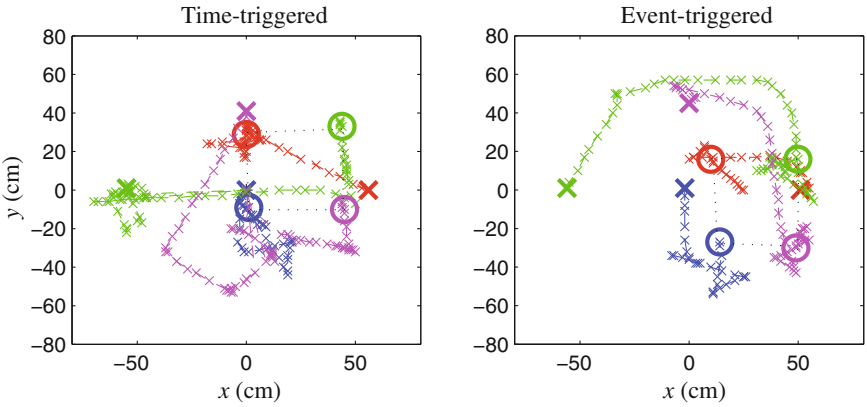

Fig. 12.14 Representation in the plane of the trajectories of each robot for time-triggered (*left*) and event-triggered (*right*) communications and consensus protocols. Agent 2 (*blue*), Agent 3 (*red*), Agent 4 (*green*), and Agent 5 (*magenta*). The initial an final positions are marked with *crosses* and *circles of the same color*, respectively

The control law is computed according to (12.17).

Time-Triggered Versus Event-Triggered Communications

The experiment described above is performed with time-triggered communications and with event-triggered communications. The period is set to $T_s = 250\,\text{ms}$ in the time-triggered case. The value of T_s should not be below 200 ms due to the constraints imposed by the measurement and communication systems, and by the robots microcontroller. The threshold of the trigger function is set to a constant value $\delta_0 = 4\,\text{cm}$. A larger value would cause an excessive formation error, and a smaller value would not provide much benefit respect to periodic communication. Moreover, the measurements taken by the camera have an estimated error around 2 cm.

The results for both approaches are illustrated in Fig. 12.14. The formation is reached in both cases but the center of the formation is different although the initial conditions are the same. This is a side effect of real communications, since the trajectories of the robots are affected by delays, communication losses, etc.

Six shots of the experiment for the event-based communication case are shown in Fig. 12.15. Note that the formation is almost reached at $t = 10\,\text{s}$. This is also illustrated if the distance to the formation is computed as

$$d_i = \sqrt{(\bar{x}_i - r_{x,i})^2 + (\bar{y}_i - r_{y,i})^2}$$

for each robot. Figure 12.16 depicts d_i over time. For the event-based case, d_i is almost equal for the four agents at $t = 11.5\,\text{s}$. However, disturbances possibly affect the robot 4, which is the latest robot to stop. If periodic and event-driven communications are compared, the time instant at which the formation is reached is similar in both approaches. However, if the amount of communication required is computed for both

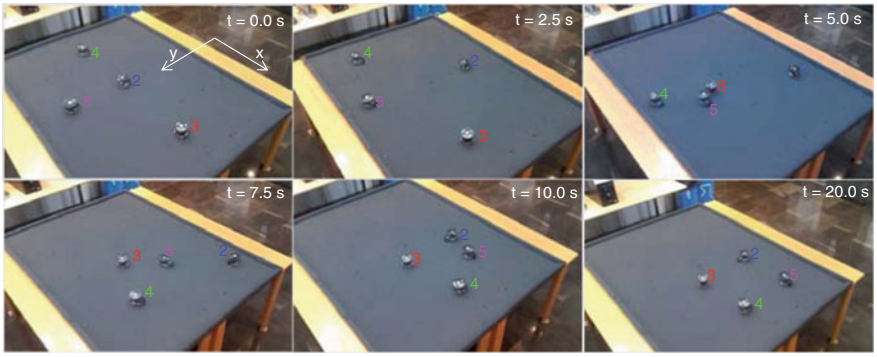

Fig. 12.15 Shots of the consensus protocol experiment with event-triggered communications at six instants of time

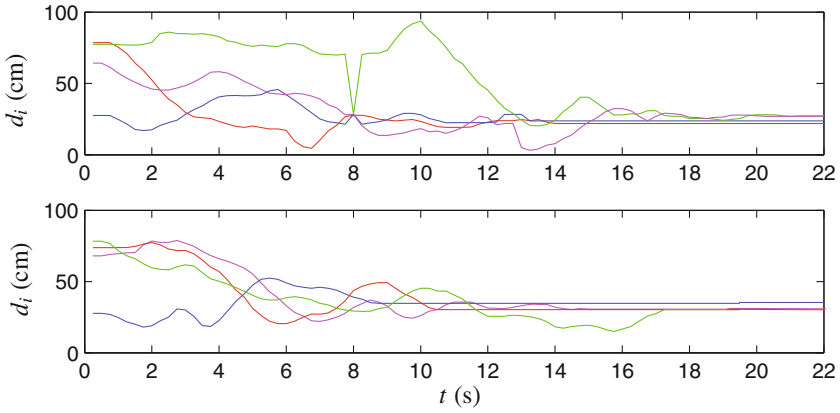

Fig. 12.16 Distance to the formation over time for time-triggered (*above*) and event-triggered (*below*) communications and consensus protocols. Agent 2 (*blue*), Agent 3 (*red*), Agent 4 (*green*), and Agent 5 (*magenta*)

cases, the goodness of the event is highlighted. The number of events is summarized in Table 12.2. The total number of communications is the number of events plus the result of multiplying the number of events by the number of agents to which each robot sends information. This accounts for 356, whereas the number of transmissions for the periodic approach is

$$\left(\frac{t_f - t_0}{T_s} + 1\right) \times (\text{No. robots} + \text{No. links}) = 89 \times (4 + 5) = 801 \text{ transmissions}, \tag{12.20}$$

where $t_0 = 0$ and $t_f = 22\,\text{s}$.

Table 12.2 Number of events generated by each robot

Robot	No. events	No. broadcasts
2	26	52
3	36	36
4	54	54
5	49	49
Total	165	191

Energy Consumption

The number of transmissions is related to the energy consumption. The RF module of the Moway robots has the following characteristics [185]:

- Transmission current I_t: 11.3 mA.
- Reception current I_r: 12.3 mA.
- Average voltage V_{RF}: 2.75 V.
- Duration of transmission/reception δt_{RF}(estimated): 10 ms.

Note that the number of receptions and transmissions has to be considered separately to compute the energy consumption. When an event is detected by the *Gateway module*, the state is transmitted to the robot. Thus, energy is consumed in the reception at the robot. Then, the robot sends this information to the neighbors. This process consumes energy in the transmission (at the sender) and in the reception (at the receiver). Thus, the energy consumption due to the transmission/reception of the RF modules for the event-triggered approach is

$$\begin{aligned} E_{ET} &= V_{RF}\delta t_{RF}\big(\text{No. events} \times I_r + \text{No. broadcasts} \times (I_r + I_t)\big) \\ &= 2.75[\text{V}] \cdot 0.01[\text{s}] \cdot \big(165 \cdot 0.123[\text{A}] + 191 \cdot (0.123[\text{A}] + 0.113[\text{A}])\big) = 1.80[\text{J}]. \end{aligned}$$

According to the aforesaid and to (12.20), it follows that the energy consumption for the time-driven approach is

$$E_{TT} = 2.75[\text{V}] \cdot 0.05[\text{s}] \cdot (809 \cdot 0.123[\text{A}] + 89 \cdot 5 \cdot 0.113[\text{A}]) = 4.09[\text{J}].$$

Thus, the energy consumption is reduced 56 % with event-triggered communications in this experiment.

However, the following question can be raised: Does this reduction in communication cause an increase in the energy consumption by other tasks such as actuation? The evolution of control law values is depicted in Fig. 12.17. The values of v_i and w_i obtained from (12.17) are scaled by constant gains, and then the library that controls the motors of the Moway robots converts the calculated signals into commands that are applied to each motor. Hence, it is difficult to evaluate the energy consumed, but still possible to compare the efficiency of time-driven and event-driven approaches if the following parameters are computed:

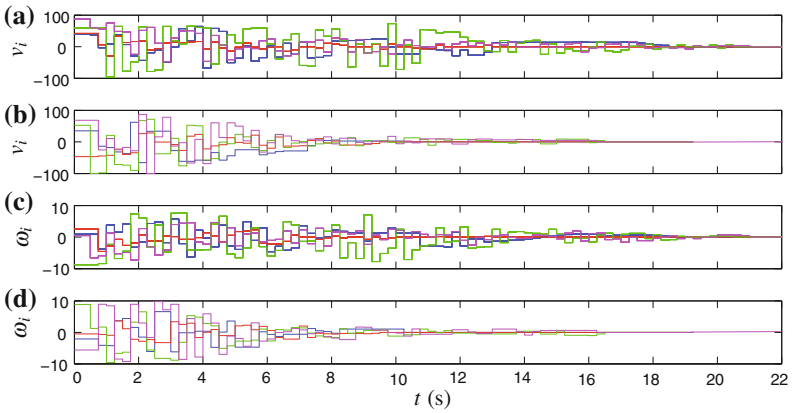

Fig. 12.17 Control signals: **a** v_i time-triggered, **b** v_i event-triggered, **c** ω_i time-triggered, **d** ω_i event-triggered approaches. Agent 2 (*blue*), Agent 3 (*red*), Agent 4 (*green*), and Agent 5 (*magenta*)

Table 12.3 W_{v_i} and W_{ω_i} with event-based and periodic communications for each robot

Robot	Periodic		Event-based	
	W_{v_i}	W_{ω_i}	W_{v_i}	W_{ω_i}
2	404.43	35.35	285.50	22.28
3	119.68	11.47	166.55	13.89
4	658.92	66.16	360.01	36.85
5	375.92	35.29	332.80	44.80

$$W_{v_i} = \int_{t_0}^{t_f} |v_i(t)|dt \tag{12.21}$$

$$W_{\omega_i} = \int_{t_0}^{t_f} |\omega_i(t)|dt. \tag{12.22}$$

The results are summed up in Table 12.3. As expected, event-driven communications does not yield an increase in V_{v_i} and W_{ω_i}. Moreover, an additional benefit is the reduction of the microprocessor load, since the actuation task is updated less often, and this also reduces the energy consumption.

12.4.2.2 Experiment 2: Leader–Follower Protocol

The experiment described in the previous section is repeated when a leader of the group is defined. Specifically, the communication graph is redefined as depicted in Fig. 12.18, and the robot 2 is the leader. It computes its control law to reach its desired position $[-20, -20]$ as

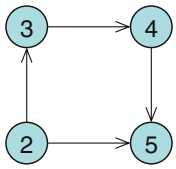

Fig. 12.18 Scheme of the communication topology, initial formation (*left*) and desired formation (*right*)

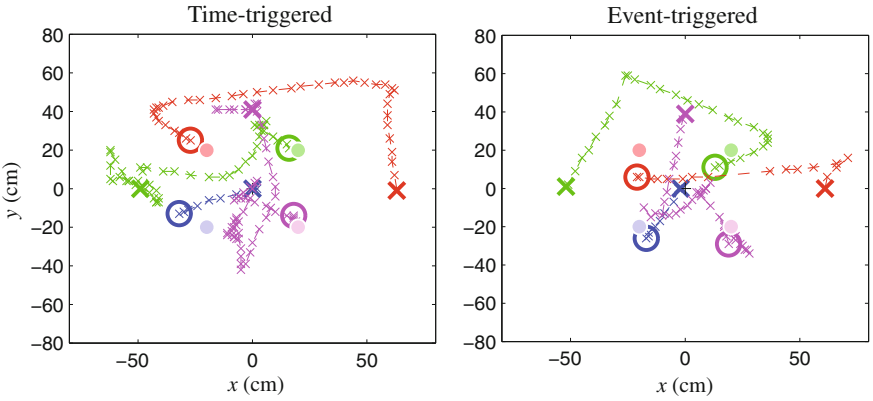

Fig. 12.19 Representation in the plane of the trajectories of each robot for time-triggered (*left*) and event-triggered (*right*) communications. Agent 2 (*blue*), Agent 3 (*red*), Agent 4 (*green*), and Agent 5 (*magenta*). The initial an final positions are marked with *crosses* and *circles of the same color*, respectively. The desired formation is represented by the *circles in light colors*

$$\begin{pmatrix} v_2 \\ \omega_2 \end{pmatrix} = \begin{pmatrix} \cos\theta_2 & -d\sin\theta_2 \\ \sin\theta_2 & d\cos\theta_2 \end{pmatrix}^{-1} \begin{pmatrix} -20 - \bar{x}_{b,2} \\ -20 - \bar{y}_{b,2} \end{pmatrix}.$$

The experiment is performed with time-triggered communications and with event-triggered communications, and the results comparative is illustrated in Fig. 12.19. The consensus is no longer reached due to the existence of a leader, and the final positions are equal to the position offsets (r_x, r_y) (12.19).

In the event-based approach, the number of events for each robot accounts for 9, 21, 37, and 48 (see Table 12.4), whereas in the periodic case the number of executions of the measurement task is 73 for each robot. This yields a total number of transmissions of 191 for the event-based case, and 584 for the periodic case (computed as in (12.20)). Similar conclusions can be extracted for the energy consumption as in the consensus protocol.

Note that the robots do not exactly achieve the formation. This might be because of measurement errors by the camera, actuators deadzone, etc. Another problem that is illustrated in the experiment is the loss of communication. For instance, the robot

Table 12.4 Number of events generated by each robot

Robot	No. events	No. broadcasts
2	9	18
3	21	21
4	37	37
5	48	0
Total	111	76

3 does not receive its position from the measurement system (see Fig. 12.19 right). Hence, events are not detected even if the robot is moving, the control law is not updated, and the current state is not send to the neighbors.

12.5 Conclusions

The formation control of mobile robots has been presented as an application example in which the use of distributed event-based control can be useful. An interactive simulator has been developed in which the system dynamics, the control task, and the network effects have been modeled. The tool offers high flexibility and allows to test the model under a wide range of scenarios. Several examples of how the simulator can be used have been provided.

The distributed event-based control has been also implemented over a real test bed of mobile robots. The experimental results illustrate the reduction in the number of transmissions with event-based communications, which implies energy saving. They also manifest some of the problems that need to be faced when dealing with a real system such as loss of communication, measurement errors, or actuators deadzones.

Chapter 13
Conclusions

María Guinaldo, Pablo Millán and Luis Orihuela

13.1 Summary of the Book

This book has presented different approaches to asynchronous control for networked systems. Following an introduction to the motivation and objectives of this book, the volume has been divided into two parts: Centralized and distributed solutions for control and estimation in networked control systems.

The first part of the book consists of five chapters. Chapters 2 and 3 do not take into account the network effects in the analysis. Chapter 2 has been devoted to study the properties of limit cycles, which are an important phenomenon that occurs in asynchronous control when a level-crossing sampling scheme is applied, which is one of the sampling strategies that are extensively used in the literature, and in particular, through this book. Moreover, Chap. 3 has proposed to maximize dynamically the inter-sampling times, resorting to a self-trigger approach and a stability analysis that can be posed as a quadratic optimization problem. The next sampling instant is decided by the controller to guarantee practical stability and the sensor might enter into a sleep or low-consumption mode so that energy consumption is reduced.

M. Guinaldo (✉)
Dpto. de Informática y Automática, Escuela Técnica Superior de Informática,
UNED, Madrid, Spain
e-mail: mguinaldo@dia.uned.es

P. Millán · L. Orihuela
Dpto. de Matemáticas e Ingeniería, Escuela Técnica Superior de Ingeniería,
Universidad Loyola Andalucía, Seville, Spain
e-mail: pmillan@uloyola.es

L. Orihuela
e-mail: dorihuela@uloyola.es

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9_13

Chapters 4–6 have presented different strategies that not only aims at reducing the number of samplings due to event-triggering, but also at dealing with delays and packet losses in remote controller architectures. Chapter 4 has combined anticipative controllers, which estimate future states of the system based on a model that considers delays, and the advantages of packet-based networks that allow the transmission of hundreds of bytes without increasing the bandwidth cost. In Chap. 5, an H_2/H_∞ design that joins optimal control and disturbance attenuation has been proposed. Finally, the combination of predictive controllers with actuator buffers has been suggested in Chap. 6 as an alternative to cope with packet losses and reduce network traffic.

The second part of the book is composed of six chapters. Two aspects have been covered in Part II: distributed control and distributed estimation in LTI large-scale systems. As in Part I, the first chapters have dealt with asynchronous solutions that consider ideal network conditions. In particular, Chap. 7 has focused on distributed event-triggering. Different control strategies have been proposed to reduce communication between agents while guaranteeing stability and avoiding Zeno behavior. Chapter 8 has addressed the design of distributed estimators. The nodes in the network run consensus algorithms to estimate the whole system state exchanging local measurements in an event-based fashion. Chapter 9 has tackled the joint problem of distributed estimation and control in a network, in which the sensing and control capabilities are shared by a number of agents.

Extensions of Chaps. 7 and 8 for imperfect communication channels have been presented in Chaps. 10 and 11, respectively. Networked control strategies have been designed to deal with delays and data dropouts. Finally, Chap. 12 has presented an application example in which the use of distributed event-based techniques can be useful: the formation control of mobile robots.

13.2 Comparison Between the Different Solutions

This section intends to be a guide to help readers to choose between the different solutions that have been presented in this book. For this purpose, we focus on several aspects that will help to compare the strategies.

Centralized versus distributed The advantages of one or the other approach have been deeply discussed in previous chapters. To decide which the preferred architecture is, we recommend to answer the following questions: How many elements has the control loop? Are they spatially distributed? Can the system be modeled as an interconnection of subsystems or of devices? How well known are these interconnections? Do the sensor nodes have access to total or partial information? When the system has few elements, these are not far away from each other, and a central agent can collect all the available information of the system; then, a centralized architecture will achieve high performance. Otherwise, a distributed architecture will be more efficient, flexible, cheap, and robust against node failures. Additionally, the

decision might be imposed by the nature of the system or plant. Not only can the sensors/actuators be situated at remote difficult-to-access locations, but also the system might be split itself. Consider, for instance, the formation control of mobile robots studied in Chap. 12. This is a typical situation in which centralized approaches lose interest.

Network conditions As it has been deeply explained, the transmission through a communication network is affected by several harming effects of different natures: delays, dropouts, quantization, etc. However, some of these undesired network effects might be neglected when the dynamics of the plant is slow enough, compared to the transmission rate and/or bandwidth available. Furthermore, some network protocols achieve the reliable delivery of packets, at a cost of higher delays. In any of these cases, the reader may choose between any of the solutions proposed in Part I and II in which ideal network conditions have been considered, that is, Chaps. 2, 3, and 7–9, respectively. When dealing with fast-dynamics plants, the effects caused by the delays or dropouts can be terribly dangerous. Here, the reader is offered different solutions (Chaps. 4 –6 and 10–12) that cope with these undesired effects, maintaining the stability of the closed loop and still achieving some interesting performance measurements.

Model uncertainties and disturbances Model-based controllers have been used in Chaps. 3, 4, and 6, and proposed as an improvement for distributed control in Chap. 7. In general, model-based control allows larger inter-event times compared with zero-order hold strategies. For instance, it has been shown that how the model-based version of the distributed controller of Chap. 7 provides around three times longer broadcasting periods. The anticipative and predictive controllers presented in Chaps. 4 and 6, respectively, also use a model to compensate delays and packet losses, and even more, to estimate external disturbances in Chap. 4. However, in practice, perfect model matching is difficult to achieve. A complementary approach that allows to compensate disturbances and model uncertainties, as well as delays, has been proposed in Chap. 5. Finally, we should keep in mind the John von Neumann's aspiration: *All stable processes we shall predict. All unstable processes we shall control.*

Computation capabilities Another aspect to take into account is the platform where the controller will be implemented. The computing power of today's computers is not an issue to worry about anymore. In this regard, the effort required by the optimization processes of Chaps. 3 and 6 is perfectly achievable by conventional PCs and PLCs for most of the plants (one exception might be processes with tight real-time constraints). However, the recent popularization of low-cost open-hardware platforms such as Arduino [11], which have limited computation and memory capacities as well as software installation constraints, makes more suitable the usage of other alternatives, such as the anticipative controller of Chap. 4, though the optimality is lost.

Energy consumption As in the previous case, the flexibility that low-cost wireless components offer has motivated a change of traditional point-to-point architectures. Batteries in many cases power the new, smart sensor, actuator, and controller nodes.

In this regard, the energy consumption awareness has been in the focus of several chapters. In general, asynchronous techniques cut down the energy consumption required by the transmission of information, since it has been demonstrated to reduce the number of communications when compared with periodic control. However, there are other issues that affect the energy efficiency. For instance, Chap. 3 has presented a self-triggering approach that avoids the continuous monitoring of the plant by the sensor nodes that other event-triggered strategies require. Additionally, the frequency of actuation has been taken into account in Chaps. 2 and 7 in different contexts.

Full-state and output measurement When full-state measurement cannot be assumed, two alternatives have been discussed. First, Chap. 4 has proposed LTI controllers to deal with output measurements. A more challenging aspect is the state estimation from partial output measurements in distributed architectures. This has been addressed in Chaps. 8 and 11 for reliable and non-reliable networks, respectively. Finally, Chap. 9 has provided an innovative estimation and control scheme so that the joint action of all agents allows to control the system and to monitor its state from different locations.

13.3 Concluding Remarks

There is no doubt that as smart sensors and actuator become more flexible, sophisticated, and cheaper, distributed networked control will gain increasing interest. Accompanied by the development of new communication protocols, wireless devices, routing protocols, and control algorithms, the distributed networked approach will extend both: the number of classical application that will follow this paradigm and the number of new ones that will become feasible, thanks to these techniques.

In this regard, as distributed networked control become ubiquitous and the number of devices concurrently interconnected grows, the need of asynchronous sampling methods will grow. The reduction of the bandwidth usage in wireless industrial processes and the expansion of batteries lifetime in application with portable or onboard sensors will be always key objectives to be fulfilled. To this purpose, asynchronous techniques are undoubtedly one of the most valuable strategies that control theory can provide.

This book contains an up-to-date review of the state-of-the-art concerning distributed and asynchronous control, as well as a reasonable number of control and estimation strategies that can found applications in different scenarios.

May the reader find it interesting and fruitful.

Appendix A
Proofs

A.1 Chapter 4

A.1.1 Proof of Theorem 4.2

The forward difference of the Lyapunov function (4.10) for (4.25) is

$$\begin{aligned}
&\Delta V(k) = x^T(k+1)Px(k+1) - x^T(k)Px(k)\\
&= (A_K x(k) + BK\varepsilon(k) + w(k))^T P\,(A_K x(k) + BK\varepsilon(k) + w(k)) - x^T(k)Px(k)\\
&= -x^T(k)Qx(k) + 2\varepsilon^T(k)(BK)^T PA_K x(k) + \varepsilon^T(k)(BK)^T PBK\varepsilon(k)\\
&+ 2w^T(k)PA_K x(k) + 2w^T(k)PBK\varepsilon(k) + w^T(k)Pw(k)\\
&\leq -\lambda_{min}(Q)\|x(k)\|^2 + 2\|(BK)^T PA_K\|\|\varepsilon(k)\|\|x(k)\| + \|(BK)^T PBK\|\|\varepsilon(k)\|^2\\
&+ 2\|PBK\|\|w(k)\|\|\varepsilon(k)\| + 2\|PA_K\|\|w(k)\|\|x(k)\| + \lambda_{max}(P)\|w(k)\|^2.
\end{aligned} \tag{A.1}$$

The error and the disturbance are bounded by $\|\varepsilon(k)\| \leq 2\delta$ and $\|w(k)\| \leq w_{max}$. Thus, the Lyapunov function decreases if (A.1) is negative. This holds whenever

$$\|x(k)\| \geq \frac{\sigma_b + \sqrt{\sigma_b^2 + 4\sigma_a\sigma_c}}{2\sigma_a} = \sigma_w,$$

where σ_a, σ_b, σ_c and σ_w are defined in (4.27)–(4.30).

The state decreases until it reaches this bound. Let us denote k^* the time instant at which the state enters this region. According to (4.25), the norm of the state at the next step is, in the worst case:

$$\|x(k^*+1)\| \leq \|A_K\|\sigma_w + \|BK\|2\delta + w_{max}.$$

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9

So if the state leaves the region, the Lyapunov function decreases again. Using the property of the Lyapunov function $\lambda_{min}(P)\|x\|^2 \leq x^T P x \leq \lambda_{max}(P)\|x\|^2$, the state $x(k)$ remains bounded by (4.26) $\forall k \geq k^*$, and this concludes the proof.

A.1.2 Proof of Theorem 4.3

The forward difference of the Lyapunov function (4.42) for (4.41) is

$$\begin{aligned}
\Delta V(k) &= \xi^T(k+1)P\xi(k+1) - \xi^T(k)P\xi(k) \\
&= \left(A_{CL}\xi(k) + F(\varepsilon_y(k) + v(k))\right)^T P \left(A_{CL}\xi(k) + F(\varepsilon_y(k) + v(k))\right) \\
&\quad - \xi^T(k)P\xi(k) \\
&= -\xi^T(k)Q\xi(k) + 2(\varepsilon_y^T(k) + v^T(k))F^T P A_{CL}\xi(k) \\
&\quad + (\varepsilon_y^T(k) + v^T(k))F^T P F(\varepsilon_y(k) + v(k)) \\
&\leq -\lambda_{min}(Q)\|\xi(k)\|^2 + 2\|F^T P A_{CL}\|\|\varepsilon_y(k) + v(k)\|\|\xi(k)\| \\
&\quad + \|F^T P F\|\|\varepsilon_y(k) + v(k)\|^2. \qquad \text{(A.2)}
\end{aligned}$$

The right-hand side of (A.2) is an algebraic second-order equation in $\|\xi(k)\|$ such that the Lyapunov function decreases whenever

$$\|\xi(k)\| \geq \sigma_\xi \|\varepsilon_y(k) + v(k)\|,$$

where σ_ξ is given in (4.44).

Because the error ε_y is bounded by $2\delta_y$ and the noise by v_{max}, $\Delta V < 0$ in the region $\|\xi(k)\| > \sigma_\xi(2\delta_y + v_{max})$. 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 (4.41), it follows that

$$\|\xi(k^* + 1)\| \leq (\sigma_\xi \|A_{CL}\| + \|F\|)(2\delta_y + v_{max}).$$

Then the state can leave the region so the Lyapunov function decreases again. If the inequalities $\lambda_{min}(P)\|\xi\|^2 \leq \xi^T P \xi \leq \lambda_{max}(P)\|\xi\|^2$ are used, it is straightforward to see that the state $\xi(k)$ remains bounded by (4.43) $\forall k \geq k^*$, and this concludes the proof.

A.2 Chapter 5

A.2.1 Proof of Proposition 5.2

Consider, first, the system without disturbances ($w(k) \equiv 0$). The forward difference can be calculated as

$$\begin{aligned}\Delta V(k) = {} & x^T(k+1)Px(k+1) - x^T(k)Px(k) \\ & + x^T(k)Z_1x(k) - x^T(k-\tau_{max})Z_1x(k-\tau_{max}) \\ & + x^T(k)Z_2x(k) - x^T(k-\tau_{min})Z_2x(k-\tau_{min}) \\ & + \tau_{max}^2 \Delta x^T(k)Z_3\Delta x(k) - \tau_{max}\sum_{j=k-\tau_{max}}^{k-1} \Delta x^T(j)Z_3\Delta x(j) \\ & + \Delta\tau^2 \Delta x^T(k)Z_4\Delta x(k) - \Delta\tau\sum_{j=k-\tau_{max}}^{k-\tau_{min}-1} \Delta x^T(j)Z_4\Delta x(j),\end{aligned}$$

where $\Delta\tau = \tau_{max} - \tau_{min}$. The summations can be divided into two parts as follows:

$$\begin{aligned}-\sum_{j=k-\tau_{max}}^{k-1} \Delta x^T(j)Z_3\Delta x(j) = {} & -\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x^T(j)Z_3\Delta x(j) \\ & -\sum_{j=k-\tau(k)}^{k-1} \Delta x^T(j)Z_3\Delta x(j), \\ -\sum_{j=k-\tau_{max}}^{k-\tau_{min}-1} \Delta x^T(j)Z_4\Delta x(j) = {} & -\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x^T(j)Z_4\Delta x(j) \\ & -\sum_{j=k-\tau(k)}^{k-\tau_{min}-1} \Delta x^T(j)Z_4\Delta x(j).\end{aligned}$$

The resulting terms can be bounded using the Jensen inequality:

$$\begin{aligned}-\tau_{max}\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x^T(j)Z_3\Delta x(j) &\le -\left[\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x(j)\right]^T Z_3 \left[\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x(j)\right], \\ -\tau_{max}\sum_{j=k-\tau(k)}^{k-1} \Delta x^T(j)Z_3\Delta x(j) &\le -\left[\sum_{j=k-\tau(k)}^{k-1} \Delta x(j)\right]^T Z_3 \left[\sum_{j=k-\tau(k)}^{k-1} \Delta x(j)\right],\end{aligned}$$

$$-\Delta\tau \sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x^T(j) Z_4 \Delta x(j) \leq -\left[\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x(j)\right]^T Z_4 \left[\sum_{j=k-\tau_{max}}^{k-\tau(k)-1} \Delta x(j)\right],$$

$$-\Delta\tau \sum_{j=k-\tau(k)}^{k-1} \Delta x^T(j) Z_4 \Delta x(j) \leq -\left[\sum_{j=k-\tau(k)}^{k-\tau_{min}-1} \Delta x(j)\right]^T Z_4 \left[\sum_{j=k-\tau(k)}^{k-\tau_{min}-1} \Delta x(j)\right].$$

Since the augmented state vector has been defined as $\xi^T(k) = [x^T(k)\ x^T(k - \tau(k))\ x^T(k - \tau_{min})\ x^T(k - \tau_{max})]$, the forward difference of the functional can be written as

$$\Delta V(k) = \xi^T(k) \left(M + \tilde{A}^T P \tilde{A} + (\tilde{A} - \tilde{I})^T (\tau_{max}^2 Z_3 + \Delta\tau^2 Z_4)(\tilde{A} - \tilde{I})\right) \xi(k), \quad \text{(A.3)}$$

satisfying Assumption 5.2. for $w \equiv 0$.

Consider now the case with disturbances. Including $w(k)$ into the augmented state vector, the forward difference (A.3) can be written now as

$$\Delta V(k) = \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix}^T \left(\begin{bmatrix} M & 0 \\ * & 0 \end{bmatrix} + \begin{bmatrix} \tilde{A}^T \\ B_w^T \end{bmatrix} P \begin{bmatrix} \tilde{A} & B_w \end{bmatrix} + \begin{bmatrix} (\tilde{A} - \tilde{I})^T \\ B_w^T \end{bmatrix} (\tau_{max}^2 Z_3 + \Delta\tau^2 Z_4) \begin{bmatrix} (\tilde{A} - \tilde{I}) & B_w \end{bmatrix} \right) \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix}. \quad \text{(A.4)}$$

Now some null terms are added to the forward difference:

$$\Delta V(k) = \Delta V(k) + z_\infty^T(k) z_\infty(k) - \gamma^2 w^T(k) w(k) + \gamma^2 w^T(k) w(k) - z_\infty^T(k) z_\infty(k).$$

The first two terms are included in the quadratic product (A.4) taking into account that $z_\infty^T(k) z_\infty(k) = \xi^T(k) \tilde{C}_\infty^T \tilde{C}_\infty \xi(k)$, where $\tilde{C}_\infty = [C_\infty\ \ D_\infty K\ \ 0\ \ 0]$. Thus,

$$\Delta V(k) = \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix}^T \left(\begin{bmatrix} M + \tilde{C}_\infty^T \tilde{C}_\infty & 0 \\ * & -\gamma^2 I \end{bmatrix} + \begin{bmatrix} \tilde{A}^T \\ B_w^T \end{bmatrix} P \begin{bmatrix} \tilde{A} & B_w \end{bmatrix} + \begin{bmatrix} (\tilde{A} - \tilde{I})^T \\ B_w^T \end{bmatrix} (\tau_{max}^2 Z_3 + \Delta\tau^2 Z_4) \begin{bmatrix} (\tilde{A} - \tilde{I}) & B_w \end{bmatrix} \right) \begin{bmatrix} \xi(k) \\ w(k) \end{bmatrix} + \gamma^2 w^T(k) w(k) - z_\infty^T(k) z_\infty(k). \quad \text{(A.5)}$$

Equation (A.5) has the same structure as (5.10), so the proposed functional satisfies all the conditions given in Assumption 5.2.

Finally, matrix Ψ can be obtained after some careful mathematical manipulations.

A.3 Chapter 7

A.3.1 Proof of Theorem 7.1

The analytical solution of (7.22) is

$$x(t) = e^{(A_K+\Delta)t}x(0) + \int_0^t e^{(A_K+\Delta)(t-s)}BK\varepsilon(s)ds. \tag{A.6}$$

From Assumption 7.1, the matrix A_K is diagonalizable and

$$\|e^{A_K t}\| \leq \kappa(V)e^{-|\alpha_{max}(A_K)|t}.$$

Thus, (7.10) can be used to bound $e^{(A_K+\Delta)t}$ as

$$\|e^{(A_K+\Delta)t}\| \leq \kappa(V)e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t}.$$

Note that the exponent is negative since $|\alpha_{max}(A_K)| - \kappa(V)\|\Delta\| > 0$ from Assumption 7.2. Let us denote $\alpha_\Delta = |\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|$.

Consequently, the state can be bounded by

$$\|x(t)\| \leq \kappa(V)\left(e^{-\alpha_\Delta t}\|x(0)\| + \int_0^t e^{-\alpha_\Delta(t-s)}\|BK\|\|\varepsilon(s)\|ds\right).$$

The overall system error is bounded by

$$\|\varepsilon(t)\| \leq \sqrt{N_a}\left(\delta_0 + \delta_1 e^{-\beta t}\right).$$

This yields

$$\begin{aligned}\|x(t)\| &\leq \kappa(V)\left(e^{-\alpha_\Delta t}\|x(0)\| + \int_0^t \sqrt{N_a}e^{-\alpha_\Delta(t-s)}\|BK\|(\delta_0 + \delta_1 e^{-\beta s})\right)\\ &= \kappa(V)\left(e^{-\alpha_\Delta t}\|x(0)\| + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta}\left(1 - e^{-\lambda_\Delta t}\right)\right.\\ &\quad \left. + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \beta}\left(e^{-\beta t} - e^{-\alpha_\Delta t}\right)\right),\end{aligned}$$

which by reordering terms and restoring $\alpha_\Delta = |\alpha_{max}(A_K)| - \kappa(V)\|\Delta\|$ yield (7.28), proving the first part of the theorem. Note that the previous expression can be upper

bounded by

$$\|x(t)\| \leq \kappa(V)\left(\|x(0)\|e^{-\alpha_\Delta t} + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta} + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \alpha}e^{-\beta t}\right), \tag{A.7}$$

by omitting the negative terms.

We next show that broadcasting period is lower bounded. If t^* refers to the last event time occurrence, $\|\varepsilon_i(t^*)\| = 0$, and $F_{e,i}(t^*) = -\delta_0 - \delta_1 e^{-\beta t^*} < 0$.

Because $\dot{\varepsilon}_i(t) = -\dot{x}_i(t)$ and $\|\varepsilon_i(t)\| \leq \int_{t^*}^{t} \|\dot{x}_i(t)\| \leq \int_{t^*}^{t} \|\dot{x}(t)\|$ hold, and from (7.22) we derive

$$\|\dot{x}(t)\| \leq \|A_K + \Delta\|\|x(t)\| + \|BK\|\sqrt{N_a}\left(\delta_0 + \delta_1 e^{-\beta t^*}\right).$$

If the last event occurred at time $t^* > 0$

$$\|\varepsilon_i(t)\| \leq \int_{t^*}^{t} \|\dot{x}(t)\| \leq \int_{t^*}^{t} (\|A_K + \Delta\|\|x(s)\| + \|BK\|\varepsilon(s))\, ds,$$

and $\|x(t)\| \leq \|x(t^*)\|$ holds in (A.7). Thus, defining the following constants:

$$\begin{aligned}
k_1 &= \kappa(V)\|A_K + \Delta\|\|x(0)\| \\
k_2 &= \|BK\|\sqrt{N_a}\delta_1\left(\frac{\kappa(V)\|A_K + \Delta\|}{\alpha_\Delta - \beta} + 1\right) \\
k_3 &= \|BK\|\sqrt{N_a}\delta_0\left(\frac{\kappa(V)\|A_K + \Delta\|}{\alpha_\Delta} + 1\right),
\end{aligned}$$

the error can be bounded as

$$\|\varepsilon_i(t)\| \leq \int_{t^*}^{t} (k_1 + k_2 + k_3) ds = (k_1 + k_2 + k_3)(t - t^*).$$

The next event will not be triggered before $\|\varepsilon_i(t)\| = \delta_0 + \delta_1 e^{-\beta t} \geq \delta_0$. Thus a lower bound on the inter-events time is given by

$$T_{min} = \frac{\delta_0}{k_1 + k_2 + k_3}, \tag{A.8}$$

which is a positive quantity.

A.3.2 Pure Exponential Functions

Let us consider the case when $\delta = 0$ and, for simplicity, $\|\Delta\| = 0$. The error is bounded by $\|\varepsilon_i(t)\| \leq \delta_1 e^{-\beta t}$, and so the error goes to zero when times goes to infinity. The state bound (A.7) can be particularized for pure exponential trigger functions as follows:

$$\|x(t)\| \leq \kappa(V)\left(\|x(0)\|e^{-|\alpha_{max}(A_K)|t} + \frac{\|BK\|\sqrt{N_a}\delta_1}{|\alpha_{max}(A_K)|-\beta}e^{-\beta t}\right). \tag{A.9}$$

In order to prove that the Zeno behavior is excluded, we consider that the error $\|\varepsilon_i(t)\|$ is upper bounded by

$$\|\varepsilon_i(t)\| \leq \left(k_1 e^{-|\alpha_{max}(A_K)|t^*} + k_2 e^{-\beta t^*}\right)T.$$

Note that $k_3 = 0$ since $\delta_0 = 0$.

The next event is not triggered before $\|\varepsilon_i(t)\| = \delta_1 e^{-\beta t}$. Thus, a lower bound on the inter-event intervals is given by

$$\left(\frac{k_1}{\delta_1}e^{(\beta-|\alpha_{max}(A_K)|)t^*} + \frac{k_2}{\delta_1}\right)T = e^{-\beta T}. \tag{A.10}$$

The right-hand side of (A.10) is always positive. Moreover, for $\beta < |\alpha_{max}(A_K)|$ the left-hand side is strictly positive as well, and the term in brackets is upper bounded by $\frac{k_2+k_1}{\delta_1}$ and lower bounded by k_2/δ_1, and this yields to a positive value of T for all $t^* \geq 0$.

A.3.3 Proof of Theorem 7.2

Let us denote $F = A_K + \Delta$. The analytical solution of (7.34) is

$$x(\ell) = F^{\ell}x(0) + \sum_{j=0}^{\ell-1} F^{\ell-1-j}BK\varepsilon(j). \tag{A.11}$$

This can be bounded as

$$\|x(\ell)\| \leq \|F^{\ell}\|\|x(0)\| + \sum_{j=0}^{\ell-1}\|F^{\ell-1-j}\|\|BK\|\|\varepsilon(j)\|. \tag{A.12}$$

Let us assume that $\Delta^j \approx \mathbf{0}, \forall j \geq 2$. According to (7.11), $\|F^\ell\|$ can be bounded as

$$\begin{aligned}\|F^\ell\| = \|(A_K + \Delta)^\ell\| &\leq \|A_K^\ell\| + \|\sum_{j=0}^{\ell-1} A_K^{\ell-1-j}\Delta A_K^j\| + \mathscr{O}(\|\Delta\|^2) \\ &\approx \|A_K^\ell\| + \|\sum_{j=0}^{\ell-1} A_K^{\ell-1-j}\Delta A_K^j\|,\end{aligned}$$

since $\|\Delta\|^2 \approx 0$.

Moreover, as in the continuous time case, we assume that A_K is diagonalizable, and hence, $A_K = VDV^{-1}$. It also holds that $\|D\| = |\lambda_{max}(D)| = |\lambda_{max}(A_K)|$, where $\lambda_{max}(A_{dK})$ is the eigenvalue with the closer magnitude to 1. Thus,

$$\|A_K^\ell\| = \|VD^\ell V^{-1}\| \leq \kappa(V)|\lambda_{max}(A_K)|^\ell,$$

where $\kappa(V) = \|V\|\|V^{-1}\|$.

Similarly, the following bound can be computed for the sum:

$$\begin{aligned}\|\sum_{j=0}^{\ell-1} A_K^{\ell-1-j}\Delta A_K^j\| &\leq \kappa^2(V)\sum_{j=0}^{\ell-1}|\lambda_{max}(A_K)|^{\ell-1-j}\|\Delta\||\lambda_{max}(A_K)|^j \\ &= \kappa^2(V)|\lambda_{max}(A_K)|^{\ell-1}\|\Delta\|\ell.\end{aligned}$$

Thus, $\|F^\ell\|$ is bounded by

$$\|F^\ell\| \leq \kappa(V)|\lambda_{max}(A_K)|^\ell\left(1 + \ell\frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}\right). \tag{A.13}$$

If we consider the bound (A.13) in (A.12), it holds that

$$\begin{aligned}\|x(\ell)\| \leq{}& \kappa(V)|\lambda_{max}(A_K)|^\ell\left(1 + \ell\frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}\right)\|x(0)\| \\ &+ \sum_{j=0}^{\ell-1}\left(\kappa(V)|\lambda_{max}(A_K)|^{\ell-1-j}\left(1 + (\ell-1-j)\frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}\right)\|BK\|\|\varepsilon(j)\|\right).\end{aligned} \tag{A.14}$$

Moreover, from (7.39), the error can be bounded as $\|\varepsilon(j)\| \leq \sqrt{N_a}(\delta_0 + \delta_1\beta^j)$.

The sum in (A.14) can be computed taking into account that

$$\sum_{j=0}^{\ell-1} r^{\ell-1-j} = \frac{1-r^{\ell}}{1-r}$$
$$\sum_{j=0}^{\ell-1} (\ell-1-j) r^{\ell-1-j} = r \cdot \left(\frac{1-r^{\ell}}{(1-r)^2} - \frac{\ell r^{\ell-1}}{1-r} \right),$$

where r can be $|\lambda_{max}(A_K)|$ or $\frac{\beta}{|\lambda_{max}(A_K)|}$.

Thus, it yields that

$$\begin{aligned}
\|x(\ell)\| \leq \kappa(V) \Bigg(& \tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|} \left(1 + \tfrac{\kappa(V)\|\Delta\|}{1-|\lambda_{max}(A_K)|}\right) + |\lambda_{max}(A_K)|^{\ell} \Bigg(\|x(0)\| \\
& - \tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|} \left(1 + \tfrac{\kappa(V)\|\Delta\|}{1-|\lambda_{max}(A_K)|}\right) - \tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|} \left(1 + \tfrac{\kappa(V)\|\Delta\|}{\beta-|\lambda_{max}(A_K)|}\right) \\
& + \tfrac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|} \ell \left(\|x(0)\| - \tfrac{\|BK\|\sqrt{N_a}\delta_0}{1-|\lambda_{max}(A_K)|} - \tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|} \right) \Bigg) \\
& + \beta^{\ell} \tfrac{\|BK\|\sqrt{N_a}\delta_1}{\beta-|\lambda_{max}(A_K)|} \left(1 + \tfrac{\kappa(V)\|\Delta\|}{\beta-|\lambda_{max}(A_K)|}\right) \Bigg).
\end{aligned} \tag{A.15}$$

Defining γ_0, γ_1 as in (7.41)–(7.42), it yields to (7.40), which concludes the proof.

A.3.4 Proof of Lemma 7.4

Assume that the last broadcasting event on the subsystem i occurred at $t = t_k^{\bar{\mathcal{N}}_i}$, meaning that its own events and the neighbors' are included. If this last event does not yield a control update it means that $\|\varepsilon_{u,i}(t_k^{\bar{\mathcal{N}}_i})\| < \delta_u$. Assume that at $t = t_{k+1}^{\bar{\mathcal{N}}_i}$ there is a new broadcast in $\bar{\mathcal{N}}_i$ which triggers a control event. There are two possibilities:

- The subsystem i triggers the event. Thus,

$$\begin{aligned}
\|\varepsilon_{u,i}\left(t_{k+1}^{\bar{\mathcal{N}}_i}\right)\| &= \|\varepsilon_{u,i}\left(t_k^{\bar{\mathcal{N}}_i}\right) + u_i\left(t_k^{\bar{\mathcal{N}}_i}\right) - u_i\left(t_{k+1}^{\bar{\mathcal{N}}_i}\right)\| \\
&= \|\varepsilon_{u,i}\left(t_k^{\bar{\mathcal{N}}_i}\right) + K_i\left(x_{b,i}\left(t_k^{\bar{\mathcal{N}}_i}\right) - x_{b,i}\left(t_{k+1}^{\bar{\mathcal{N}}_i}\right)\right)\| \\
&\leq \|\varepsilon_{u,i}\left(t_k^{\bar{\mathcal{N}}_i}\right)\| + \|K_i\|\|x_{b,i}\left(t_k^{\bar{\mathcal{N}}_i}\right) - x_{b,i}\left(t_{k+1}^{\bar{\mathcal{N}}_i}\right)\|,
\end{aligned}$$

that is upper bounded by

$$\|\varepsilon_{u,i}\left(t_{k+1}^{\bar{\mathcal{N}}_i}\right)\| \leq \delta_u + \|K_i\|\left(\delta_{x,0} + \delta_{x,1} e^{-\beta t_{k+1}^{\bar{\mathcal{N}}_i}}\right).$$

- The event has been triggered for any neighbor $j \in \mathcal{N}_i$, it yields

$$\begin{aligned}\|\varepsilon_{u,i}\left(t_{k+1}^{\mathcal{N}_i}\right)\| &= \|\varepsilon_{u,i}\left(t_k^{\mathcal{N}_i}\right) + L_{ij}\left(x_{b,j}\left(t_k^{\mathcal{N}_i}\right) - x_{b,j}\left(t_{k+1}^{\mathcal{N}_i}\right)\right)\| \\ &\leq \delta_u + \|L_{ij}\|\left(\delta_{x,0} + \delta_{x,1}e^{-\beta t_{k+1}^{\mathcal{N}_i}}\right).\end{aligned}$$

  Since this holds for all t, and if the worst case is considered, it yields (7.52). Moreover, from (7.49), it follows that

$$\|\varepsilon_u(t)\| \leq \sqrt{\sum_{i=1}^{N_a}\|\varepsilon_{u,i}\|^2(t)} \leq \sqrt{\sum_{i=1}^{N_a}\bar{\delta}_{u,i}^2(t)} \leq \sqrt{N_a(\max\{\bar{\delta}_{u,i}(t)\})^2},$$

  which is equivalent to (7.53).

A.3.5 Proof of Theorem 7.3

The state of the system at any time is given by

$$x(t) = e^{(A_K+\Delta)t}x(0) + \int_0^t e^{(A_K+\Delta)(t-s)}(BK\varepsilon_x(s) + B\varepsilon_u(s))ds.$$

The error ε_x is bounded by $\sqrt{N_a}(\delta_{x,0} + \delta_{x,1}e^{-\beta t})$ and the bound on ε_u is derived in Lemma 7.4. Moreover, as already proved, it holds that

$$\|e^{(A_K+\Delta)t}\| \leq \kappa(V)e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t}.$$

With these considerations, the bound on $x(t)$ can be calculated following the used methodology in the previous proofs to derive (7.55), showing that the system is globally ultimately bounded. Furthermore, (7.55) is upper bounded by

$$\|x(t)\| \leq \sigma_1 + \kappa(V)\|x(0)\|e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t} + \sigma_2 e^{-\beta t}, \tag{A.16}$$

if the negative terms are omitted.

The Zeno behavior exclusion in the broadcasting and, as a consequence, in the control update, can also be proved similar to the previous results. Note that in the inter-event times $\|\dot{\varepsilon}_i(t)\| \leq \|\dot{x}_i(t)\| \leq \|\dot{x}(t)\|$, and $\|\dot{x}(t)\|$ can be bounded according to (7.50). Thus,

$$\|\varepsilon_i(t)\| \leq \int_{t^*}^t (\|A_K + \Delta\|\|x(s)\| + \|BK\|\|\varepsilon_x(s)\| + \|B\|\|\varepsilon_u(s)\|)ds.$$

If $x(t)$ is bounded according to (A.16), and the corresponding bounds on ε_x and ε_u are considered, it leads to the following lower bound for the inter-event time

$$T_{x,min} = \frac{\delta_{x,0}}{\gamma_1 + \sqrt{N_a}(\gamma_2 + \gamma_3 + \gamma_4)},$$

where

$$\begin{aligned}
\gamma_1 &= \kappa(V)\|x(0)\|\|A_K + \Delta\| \\
\gamma_2 &= (\|BK\| + \|B\|\|\mu(K)\|_{max})\delta_{x,0}\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\right) \\
\gamma_3 &= (\|BK\| + \|B\|\|\mu(K)\|_{max})\delta_{x,1}\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|-\beta}\right) \\
\gamma_4 &= \|B\|\delta_u\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\right).
\end{aligned}$$

A.3.6 Proof of Theorem 7.4

Define the overall system state estimation as $x_m = (x_{m,1}^T, \ldots, x_{m,N_a}^T)^T$. Let us prove that the bound for the inter-events time is larger in the model-based approach.

If the last event occurred at t^*, the error in the inter-event time is $\|\varepsilon_i(t)\| \leq \int_{t^*}^{t} \|\dot{\varepsilon}_i(s)\|ds$. In this interval, it also holds that

$$\|\dot{\varepsilon}_i(t)\| = \|\dot{x}_{m,i}(t) - \dot{x}_i(t)\| \leq \|\dot{x}_m(t) - \dot{x}(t)\|.$$

Observe that

$$\begin{aligned}
\dot{x}_m(t) - \dot{x}(t) &= A_{mK}x_m(t) - ((A_K + \Delta)x(t) + BK\varepsilon(t)) \\
&= (\delta A_K - \Delta)x(t) + (A_{mK} - BK)\varepsilon(t).
\end{aligned}$$

Then

$$\begin{aligned}
\|\dot{\varepsilon}_i(t)\| &\leq \|\delta A_K - \Delta\|\|x(t)\| + \|A_{mK} - BK\|\|\varepsilon(t)\| \\
&\leq \|\delta A_K - \Delta\|\|x(t)\| + \|A_{mK} - BK\|\sqrt{N_a}(\delta_0 + \delta_1 e^{-\beta t}).
\end{aligned} \tag{A.17}$$

Assume that $\delta_0, \delta_1 \neq 0$. It holds that $\delta_0 + \delta_1 e^{-\beta t} \leq \delta_0 + \delta_1 e^{-\beta t^*}$. As already stated, the bound on the state of Theorem 7.1 holds, and can be upper bounded as

$$\|x(t)\| \leq \kappa(V)\left(\|x(0)\|e^{-\alpha_\Delta t} + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta} + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \beta}e^{-\beta t}\right).$$

Moreover, it holds that $\|\delta A_K - \Delta\| \leq \|\delta A_K\| + \|\Delta\|$. Thus, the error

$$\|\varepsilon_i(t)\| \leq \int_{t^*}^{t} \|\dot{\varepsilon}(s)\| ds \leq \Big((\|\delta A_K\| + \|\Delta\|)\kappa(V) \Big(\|x(0)\| e^{-\alpha_\Delta t^*} + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta} + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \beta} e^{-\beta t^*} \Big) + \|A_{mK} - BK\|\sqrt{N_a} \left(\delta_0 + \delta_1 e^{-\beta t^*} \right) \Big) (t - t^*). \tag{A.18}$$

It follows that $\|\varepsilon_i(t)\| \leq (k_{m,1} + k_{m,2} + k_{m,3})(t - t^*)$, where

$$\begin{aligned} k_{m,1} &= \kappa(V)\|x(0)\|(\|\delta A_K\| + \|\Delta\|) \\ k_{m,2} &= \left(\frac{\kappa(V)(\|\delta A_K\| + \|\Delta\|)\|BK\|}{\alpha_\Delta - \beta} + \|A_{mK} - BK\| \right) \sqrt{N_a}\delta_1 \\ k_{m,3} &= \left(\frac{\kappa(V)(\|A_{mK}\| + \|\Delta\|)\|BK\|}{\alpha_\Delta} + \|A_{mK} - BK\| \right) \sqrt{N_a}\delta_0. \end{aligned} \tag{A.19}$$

The next event will not occur before $\|\varepsilon_i(t)\| = \delta_0 + \delta_1 e^{-\beta t} \geq \delta_0$. This condition gives a lower bound for the broadcasting period

$$T_{m,min} = \frac{\delta_0}{k_{m,1} + k_{m,2} + k_{m,3}}, \tag{A.20}$$

that is larger than the lower bound in (7.29) if $k_{m,1} + k_{m,2} + k_{m,3} < k_1 + k_2 + k_3$, which is equivalent to

$$(\|A_{mK} - BK\| - \|BK\|)\sqrt{N_a}(\delta_0 + \delta_1) < (\|A_K + \Delta\| - \|\delta A_K\| - \|\Delta\|) \Big(\|x(0)\| + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta} + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \beta} \Big).$$

After some manipulations

$$\frac{\sqrt{N_a}(\delta_0 + \delta_1)}{\|x(0)\| + \frac{\|BK\|\sqrt{N_a}\delta_0}{\alpha_\Delta} + \frac{\|BK\|\sqrt{N_a}\delta_1}{\alpha_\Delta - \beta}} < \kappa(V) \frac{\|A_K + \Delta\| - \|\delta A_K\| - \|\Delta\|}{\|A_{mK} - BK\| - \|BK\|}. \tag{A.21}$$

The denominator on the right-hand side can be bounded as

$$\|A_{mK} - BK\| - \|BK\| \leq \|A_{mK}\| + \|BK\| - \|BK\| = \|A_{mK}\|.$$

Then if Assumption 7.5 holds, (A.21) is fulfilled. Thus, the lower bound for the broadcasting period is larger for the model-based approach.

A.4 Chapter 9

A.4.1 Proof of Proposition 9.2

The dynamics of a single node is given by (9.6):

$$\hat{x}_i(k+1) = (A+BK)\hat{x}_i(k) + M_i(y_i(k) - C_i\hat{x}_i(k)) + \sum_{j\in\mathcal{N}_i} N_{ij}(\hat{x}_j(k) - \hat{x}_i(k)).$$

And the observation error at instant $k+1$ can be obtained using Proposition 9.1:

$$\begin{aligned} e_i(k+1) &= x(k+1) - \hat{x}_i(k+1) \\ &= (A+BK)x(k) - \sum_{i=1}^{p} B_iK_ie_i(k) - (A+BK)\hat{x}_i(k) - M_i(y_i(k) \\ &\quad - C_i\hat{x}_i(k)) - \sum_{j\in\mathcal{N}_i} N_{ij}(\hat{x}_j(k) - \hat{x}_i(k)). \end{aligned} \tag{A.22}$$

We can write $e_i(k+1) = (tr1) - (tr2)$, where $(tr1)$ are the terms of (A.22) which do not depend on the neighbors and $(tr2)$ are the others. Consider first the terms $(tr1)$.

$$\begin{aligned} (tr1) &= (A+BK)e_i(k) - M_iC_ie_i(k) - \sum_{i=1}^{p} B_iK_ie_i(k) \\ &= (A - M_iC_i + BK)e_i(k) - \sum_{i=1}^{p} B_iK_ie_i(k). \end{aligned} \tag{A.23}$$

Consider now $(tr2)$:

$$\begin{aligned} (tr2) &= \sum_{j\in\mathcal{N}_i} N_{ij}(\hat{x}_j(k) - \hat{x}_i(k)) \\ &= \sum_{j\in\mathcal{N}_i} N_{ij}(e_i(k) - e_j(k)). \end{aligned} \tag{A.24}$$

Using Eqs. (A.23)–(A.24) the observation error at instant $k+1$ can be written as

$$e_i(k+1) = (A - M_iC_i)\, e_i(k) + BKe_i(k) - \sum_{i=1}^{p} B_iK_ie_i(k) - \sum_{j\in\mathcal{N}_i} N_{ij}(e_i(k) - e_j(k)).$$

Finally, since the error vector was defined as $e^T(k) = \left[e_1^T(k) \ldots e_p^T(k)\right]$, it is immediate that the dynamics of $e(k)$ is (9.12). The proof is completed.

A.5 Chapter 10

A.5.1 Proof of Theorem 10.1

In order to prove the theorem, let us assume that Assumption 10.1 holds.

The analysis will derive an upper bound for the delay which preserves this assumption. The error in the time interval $[t_k^i, t_k^i + \tau_k^i)$ is given by

$$\varepsilon_i(t_k^i + \tau_k^i) - \varepsilon_i(t_k^i) = x_i(t_k^i) - x_i(t_k^i + \tau_k^i),$$

since the broadcast state $x_{b,i}$ is not updated in any agent before the time instance $t_k^i + \tau_k^i$ according to the WfA protocol, so that $x_{b,i}(t_k^i + \tau_k^i) = x_{b,i}(t_k^i) = x_i(t_{k-1}^i)$ holds. This yields

$$\begin{aligned}\varepsilon_i(t_k^i + \tau_k^i) - \varepsilon_i(t_k^i) = &\left(I - e^{A_{K,i}\tau_k^i}\right) x_i(t_k^i) \\ &+ \int_0^{\tau_k^i} e^{A_{K,i}s} \left(B_i K_i \varepsilon_i(s) + B_i \sum_{j \in \mathcal{N}_i} L_{ij} \varepsilon_j(s) \right) ds,\end{aligned}$$

based on which the upper bound for the delay τ_k^i can be derived as

$$\begin{aligned}\tau_{max,k}^i = \arg\min_{\tau_k^i \geq 0} \Bigg\{ \| &\left(I - e^{A_{K,i}\tau_k^i}\right) x_i(t_k^i) \\ &+ \int_0^{\tau_k^i} e^{A_{K,i}s} \left(B_i K_i \varepsilon_i(s) + B_i \sum_{j \in \mathcal{N}_i} L_{ij} \varepsilon_j(s) \right) ds \| = \delta \Bigg\}.\end{aligned}$$

Note that this bound depends on $x_i(t_k^i)$. In order to guarantee the existence of the bound for the delay, we need to find an upper bound of the state for any t_k^i. The state at any time is given by $x_i(t) = e^{A_{K,i}t} x_i(0) + \int_0^t e^{A_{K,i}(t-s)} \left(B_i K_i \varepsilon_i(s) + B_i \sum_{j \in \mathcal{N}_i} L_{ij} \varepsilon_j(s) \right) ds$. The error is bounded by $\|\varepsilon_i(t)\| < 2\delta, \forall i$ by Proposition 10.1. Thus, a bound on $x_i(t)$ can be calculated following the methodology of Chap. 8 as (10.5).

Note that (10.5) is upper bounded by

$$\|x_i(t)\| \leq \kappa(V_i) \left(\frac{\|B_i K_i\| 2\delta + (\sum_{j \in \mathcal{N}_i} \|B_i L_{ij}\|) 2\delta}{|\alpha_{max}(A_{K,i})|} + \|x_i(0)\| \right), \quad \forall t, \tag{A.25}$$

if the negative terms are omitted, and using that $e^{-|\lambda_{max}(A_{K,i})|t} \leq 1, \forall t \geq 0$.

In order to derive an upper bound for the delay for any t, we recall that

$$\dot{\varepsilon}_i(t) = -A_{K,i}x_i(t) - B_iK_i\varepsilon_i(t) - \sum_{j\in\mathcal{N}_i} B_iL_{ij}\varepsilon_j(t)$$

holds in the interval $t \in [t_{k-1}^i + \tau_{k-1}^i, t_k^i + \tau_k^i)$ for any two consecutive events t_{k-1}^i, t_k^i, and, in particular, it holds in the subinterval $[t_k^i, t_k^i + \tau_k^i) \subset [t_{k-1}^i + \tau_{k-1}^i, t_k^i + \tau_k^i)$. Hence, $\dot{\varepsilon}_i(t)$ can be bounded as

$$\begin{aligned}\|\dot{\varepsilon}_i(t)\| &= \|A_{K,i}x_i(t) + B_iK_i\varepsilon_i(t) + \sum_{j\in\mathcal{N}_i} B_iL_{ij}\varepsilon_j(t)\| \\ &\leq \|A_{K,i}\|\|x_i(t)\| + \|B_iK_i\|\|\varepsilon_i(t)\| + \sum_{j\in\mathcal{N}_i}\|B_iL_{ij}\|\|\varepsilon_j(t)\|. \end{aligned} \tag{A.26}$$

The state $x_i(t)$ can be bounded according to (A.25), and for the error it holds that $\|\varepsilon_i(t)\| < 2\delta$ (see Proposition 10.1). Thus, (A.26) can be integrated straightforward in the interval $[t_k^i, t_k^i + \tau_k^i)$, and it yields

$$\begin{aligned}&\|\varepsilon_i(t_k^i + \tau_k^i) - \varepsilon_i(t_k^i)\| \\ &\quad\leq \left(\|A_{K,i}\|\kappa(V_i)\left(\|x_i(0)\| + \frac{(\|B_iK_i\| + \sum_{j\in\mathcal{N}_i}\|B_iL_{ij}\|)2\delta}{|\alpha_{max}(A_{K,i})|}\right)\right. \\ &\qquad\left. + (\|B_iK_i\| + \textstyle\sum_{j\in\mathcal{N}_i}\|B_iL_{ij}\|)2\delta\right)\tau_k^i.\end{aligned}$$

Thus, the delay bound (10.5) for agent i ensures that Assumption 10.1 is not violated, and this concludes the proof.

A.5.2 Proof of Corollary 10.1

Assuming that an event was triggered at time t_k^i. The accumulated error after n_p^i consecutive packet losses and a transmission delay $\tau_k^i \leq \bar{\tau}^i$ is

$$\begin{aligned}&\underbrace{\left(\varepsilon_i\left(t_k^i + T_W^i\right) - \varepsilon_i\left(t_k^i\right)\right) + \left(\varepsilon_i\left(t_k^i + 2T_W^i\right) - \varepsilon_i\left(t_k^i + T_W^i\right)\right) + \cdots}_{n_{p^i}\text{ times}} \\ &\quad + \left(\varepsilon_i\left(t_k^i + n_p^iT_W^i + \tau_k^i\right) - \varepsilon_i\left(t_k^i + n_p^iT_W^i\right)\right) \\ &\quad = \varepsilon_i\left(t_k^i + n_p^iT_W^i + \tau_k^i\right) - \varepsilon_i\left(t_k^i\right).\end{aligned} \tag{A.27}$$

Since $n_p^i T_W^i + \tau_k^i \leq n_p^i T_W^i + \bar{\tau}^i = (n_p^i + 1)\bar{\tau}^i = \tau_{max}^i$, and τ_{max}^i is also the minimum inter-event time for the system, this implies that $\|\varepsilon_i(t_k^i + n_p^i T_W^i + \tau_k^i) - \varepsilon_i(t_k^i)\| < \delta$. Hence, $\|\varepsilon_i(t)\| < 2\delta$ holds and so does the bound (10.6).

A.5.3 Proof of Theorem 10.2

According to the UwR protocol, $\|\varepsilon_i(t)\| \leq \delta$ holds and $\varepsilon_i(t) \neq \varepsilon_{ij}(t)$, in general. However, as Assumption 10.1, $\|\varepsilon_{ij}(t_k^{ij}) - \varepsilon_i(t_k^i)\| < \delta$ yields $\|\varepsilon_{ij}(t)\| < 2\delta$.

Thus, a bound on the state can be derived from (10.9) in a similar way as in Theorem 10.1 and (10.12) holds. The proof of the first part of the theorem can be obtained by following the proof of Theorem 10.1, since in the interval $[t_k^i, t_k^{ij})$ the state information $x_{b,ij}$ remains constant in the agent j, so that $\dot{\varepsilon}_{ij}(t) = -\dot{x}_i(t)$ holds. If the error $\varepsilon_{ij}(t)$ is integrated in the interval $[t_k^i, t_k^{ij})$ considering that the state is bounded by (10.12), and that the error is bounded as discussed above, then (10.11) is derived. Finally, (10.10) can be derived as in Corollary 10.1.

A.5.3.1 Proof of Proposition 10.2

Assume that the last event occurred at time t_k^i and that the maximum transmission delay to its neighbors is τ_k^i. From Assumption 10.1, it follows that

$$\| \int_{t_k^i}^{t_k^i + \tau_k^i} \dot{\varepsilon}_i(s) ds \| = \| \varepsilon_i \left(t_k^i + \tau_k^i \right) - \varepsilon_i \left(t_k^i \right) \| < \delta_1 e^{-\beta (t_k^i + \tau_k^i)}, \tag{A.28}$$

has to be satisfied (see (10.13)) because no event is generated in the time interval $[t_k^i, t_{k+1}^i)$. Since an event has occurred at time t_k^i, $\| \varepsilon_i \left(t_k^i \right) \| = \delta_1 e^{-\beta t_k^i}$ holds and, thus

$$\| \varepsilon_i \left(t_k^i + \bar{\tau}_k^i \right) \| < \delta_1 e^{-\beta t_k^i} + \delta_1 e^{-\beta (t_k^i + \tau_k^i)} = \delta_1 \left(1 + e^{\beta \tau_k^i} \right) e^{-\beta (t_k^i + \tau_k^i)},$$

must hold. Because this result is valid for any time t and $e^{\beta \tau_k^i} < e^{\beta \tau_{max}}$, $\forall \tau_k^i < \tau_{max}$, it follows that:

$$\|\varepsilon_i(t)\| < \delta_1 (1 + e^{\beta \tau_{max}}) e^{-\beta t}.$$

A.5.4 Proof of Theorem 10.3

The state at any time is given as

$$x_i(t) = e^{A_{K,i} t} x_i(0) + \int_0^t e^{A_{K,i}(t-s)} \left(B_i K_i \varepsilon_i(s) + B_i \sum_{j \in \mathcal{N}_i} L_{ij} \varepsilon_j(s) \right) ds.$$

According to Proposition 10.2, the error is bounded by $\|\varepsilon_i(t)\| < \delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}$. Thus, a bound on $x_i(t)$ can be calculated following the methodology of Chap. 8 as

$$\|x_i(t)\| \leq \kappa(V_i)\left(\frac{\mu_i\delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta} + e^{-|\alpha_{max}(A_{K,i})|t}\left(\|x_i(0)\| - \frac{\mu_i\delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta}\right)\right),$$

which proves the second part of the theorem.

Note that (10.18) can be upper bounded as

$$\|x_i(t)\| \leq \kappa(V_i)\left(\frac{\mu_i\delta_1(1+e^{\beta\tau_{max}})e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta} + e^{-|\alpha_{max}(A_{K,i})|t}\|x_i(0)\|\right). \tag{A.29}$$

Moreover, in the interval $t \in [t^i_{k-1}+\tau^i_{k-1}, t^i_k+\tau^i_k)$ it holds that

$$\dot{\varepsilon}_i(t) = -A_{K,i}x_i(t) - B_iK_i\varepsilon_i(t) - \sum_{j\in\mathcal{N}_i} B_iL_{ij}\varepsilon_j(t),$$

and this is particularly true in the subinterval $[t^i_k, t^i_k+\tau^i_k)$. Thus

$$\begin{aligned}\|\dot{\varepsilon}_i(t)\| &= \|A_{K,i}x_i(t) + B_iK_i\varepsilon_i(t) + \sum_{j\in\mathcal{N}_i} B_iL_{ij}\varepsilon_j(t)\| \\ &\leq \|A_{K,i}\|\|x_i(t)\| + \|B_iK_i\|\|\varepsilon_i(t)\| + \sum_{j\in\mathcal{N}_i}\|B_iL_{ij}\|\|\varepsilon_j(t)\|.\end{aligned}$$

Therefore, integrating the error in the interval $[t^i_k, t^i_k+\bar{\tau}^i_k)$ and noting that $\|x_i(t)\| \leq \|x_i(t^i_k)\|$ in (A.29) in this interval

$$\begin{aligned}\|\varepsilon_i(t^i_k+\tau^i_k) - \varepsilon_i(t^i_k)\| \leq \Bigg(\|A_{K,i}\|\kappa(V_i)\Bigg(&\frac{\mu_i\delta_1(1+e^{\beta\tau_{max}})e^{-\beta t^i_k}}{|\alpha_{max}(A_{K,i})|-\beta} + e^{-|\alpha_{max}(A_{K,i})|t^i_k}\|x_i(0)\|\Bigg) \\ &+ \mu_i\delta_1\left(1+e^{\beta\tau_{max}}\right)e^{-\beta t^i_k}\Bigg)\tau^i_k.\end{aligned}$$

Denote $k_{1,i} = \|A_{K,i}\|\kappa(V_i)\|x_i(0)\|$ and $k_{2,i} = (\|A_{K,i}\|\kappa(V_i)\frac{1}{|\alpha_{max}(A_{K,i})|-\beta}+1)\mu_i\delta_1$. From (A.28) in Preposition 10.2, it follows that the upper bound on the delay satisfies

$$\left(k_{1,i}e^{-|\alpha_{max}(A_{K,i})|t^i_k} + k_{2,i}\left(1+e^{\beta\tau_{max}}\right)e^{-\beta t^i_k}\right)\tau^i_k = \delta_1 e^{-\beta\left(t^i_k+\tau^i_k\right)}.$$

It yields that

$$\left(\frac{k_{1,i}}{\delta_1}e^{-(|\alpha_{max}(A_{K,i})|-\beta)t^i_k} + \frac{k_{2,i}}{\delta_1}\left(1+e^{\beta\tau_{max}}\right)\right)\tau^i_k = e^{-\beta\tau^i_k}.$$

The right-hand side is always positive and takes values in the interval $[0, 1)$. The left-hand side is also positive and its image is $[0, +\infty)$. Hence, there is a positive solution for the upper bound on the delay. Moreover, the left-hand side is upper bounded by $(\frac{k_{2,i}}{\delta_1} + \frac{k_{2,i}}{\delta_1}(1+e^{\beta\tau_{max}}))\bar{\tau}_k^i$ for $\beta < |\alpha_{max}(A_{K,i})|$. Hence, the most conservative bound on the delay τ_{max} is given as

$$\tau_{max} = \min\{\tau_{max}^i,\ i = 1, \ldots, N_a\},$$

where τ_{max}^i are the solutions of

$$\left(\frac{k_{1,i}}{\delta_1} + \frac{k_{2,i}}{\delta_1}\left(1 + e^{\beta\tau_{max}^i}\right)\right)\tau_{max}^i = e^{-\beta\tau_{max}^i}.$$

A.5.5 Proof of Theorem 10.4

The state at any time is given as

$$x_i(t) = e^{A_{K,i}t}x_i(0) + \int_0^t e^{A_{K,i}(t-s)}\left(B_iK_i\varepsilon_i(s) + B_i\sum_{j\in N_i}L_{ij}\varepsilon_{ji}(s)\right)ds.$$

Under the UwR protocol, it holds that $\|\varepsilon_i(t)\| \leq \delta_1 e^{-\beta t}$, and $\|\varepsilon_{ji}(t)\| < \delta_1(1 + e^{\beta\tau_{max}})e^{-\beta t}$. Hence, following the same steps as in the proof of Theorem 10.3, it yields

$$\|x_i(t)\| \leq \kappa(V_i)\left(\frac{\bar{\mu}_i(\tau_{max})\delta_1 e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta} + e^{-|\alpha_{max}(A_{K,i})|t}\left(\|x_i(0)\| - \frac{\bar{\mu}_i(\tau_{max})\delta_1 e^{-\beta t}}{|\alpha_{max}(A_{K,i})|-\beta}\right)\right),$$

where $\bar{\mu}_i(\tau_{max}) = \|B_iK_i\| + \sum_{j\in N_i}\|B_iL_{ij}\|(1 + e^{\beta\tau_{max}})$.
In the interval $[t_k^i, t_k^{ij})$, $\dot{\varepsilon}_{ij}(t) = -\dot{x}_i(t)$ holds. Thus, it can be derived easily that

$$\|\varepsilon_{ij}(t_k^{ij}) - \varepsilon_{ij}(t_k^i)\| \leq \left(k_{1,i}e^{-|\alpha_{max}(A_{K,i})|t_k^i} + \left(k_{2,i} + k_{3,i}(1 + e^{\beta\tau_{max}})\right)e^{-\beta t_k^i}\right)\tau_k^{ij},$$

$k_{1,i}$, $k_{2,i}$ and $k_{3,i}$ defined in (10.21)–(10.23).

According to Proposition 10.2, $\|\varepsilon_{ij}(t_k^{ij}) - \varepsilon_{ij}(t_k^i)\| < \delta_1 e^{-\beta t_k^{ij}}$. And the upper bound on the delay is the minimum value of τ_{max}^i which solves

$$\left(\frac{k_{1,i}}{\delta_1} + \frac{k_{2,i}}{\delta_1} + \frac{k_{3,i}}{\delta_1}\left(1 + e^{\beta\tau_{max}^i}\right)\right)\tau_{max}^i = e^{-\beta\tau_{max}^i}.$$

A.5.6 Proof of Theorem 10.5

From 10.31, the state at any time is given as

$$x(t) = e^{(A_K+\Delta)t}x(0) + \int_0^t e^{(A_K+\Delta)(t-s)}\mathrm{M}\overrightarrow{\varepsilon}(s)ds.$$

According to Proposition 10.3, the error $\overrightarrow{\varepsilon}(s)$ is bounded by $\bar{\delta}$. Moreover, since A_K is diagonalizable, $e^{(A_K+\Delta)t}$ can be bounded using (7.10). Thus, it follows that

$$\begin{aligned}\|x(t)\| \leq \kappa(V)\Big(&\|x(0)\|e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t} \\ &+ \frac{\|\mathrm{M}\|\bar{\delta}}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\left(1 - e^{-(|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\right)\Big).\end{aligned}$$

Reordering terms and noting that $\|\mathrm{M}\|$ is bounded by μ_{max} because is a block diagonal matrix, it falls out (10.34).

The upper bound on the delay can be derived easily noting that if the last event occurred at $t = t_k^i$, it holds that

$$\|\varepsilon_{ij}(t_k^{ij}) - \varepsilon_{ij}(t_k^i)\| \leq \int_{t_k^i}^{t_k^{ij}} \|\dot{\varepsilon}_{ij}(s)\|ds \leq \int_{t_k^i}^{t_k^{ij}} \|\dot{x}_i(s)\|ds \leq \int_{t_k^i}^{t_k^{ij}} \|\dot{x}(s)\|ds,$$

since $x_{b,ij}$ remain constant in the interval and $\|\dot{x}_i(s)\| \leq \|\dot{x}(s)\|$.

Because $\|\dot{x}(s)\| \leq \|A_K + \Delta\|\|x(s)\| + \|\mathrm{M}\|\|\overrightarrow{\varepsilon}(s)\|$, it yields

$$\begin{aligned}\|\varepsilon_{ij}(t_k^{ij}) - \varepsilon_{ij}(t_k^i)\| \leq \Big(&\|A_K + \Delta\|\kappa(V)\Big(\|x(0)\| + \\ &\frac{\|\mathrm{M}\|\bar{\delta}}{|\alpha_{max}(A_K)|-\kappa(V)\|\Delta\|}\Big) + \|\mathrm{M}\|\bar{\delta}\Big)(t_k^{ij} - t_k^i).\end{aligned}$$

According to Assumption 10.1, no event occurs before the broadcast state is successfully received and, therefore the increase of the error in the interval $[t_k^i, t_k^{ij})$ is bounded by δ, giving the upper bound on the delay (10.33).

A.6 Chapter 11

A.6.1 Proof of Theorem 11.1

Choose the following Lyapunov–Krasovskii functional:

$$V(k) = e^T(k)Pe(k) + \sum_{i=k-\tau_{max}}^{k-1} e^T(i)Z_1e(i) + l \times \tau_{max} \sum_{j=-\tau_{max}+1}^{0} \sum_{i=k+j-1}^{k-1} \Delta e^T(i)Z_2\Delta e(i), \tag{A.30}$$

where $\Delta e(k) = e(k+1) - e(k)$. Note that the third term is included l times, one for each communication link. The forward difference can be calculated as

$$\begin{aligned}
\Delta V(k) &= e^T(k+1)Pe(k+1) - e^T(k)Pe(k) + \\
&\quad e^T(k)Z_1e(k) - e^T(k-\tau_{max})Z_1e(k-\tau_{max}) + \\
&\quad l \times \tau_{max}^2 \Delta e^T(k)Z_2\Delta e(k) - l \times \tau_{max} \sum_{j=k-\tau_{max}}^{k-1} \Delta e^T(j)Z_2\Delta e(j) \\
&= \begin{bmatrix} e^T(k) & d^T(k) \end{bmatrix} \begin{bmatrix} \Phi^T(\mathscr{M}) \\ \Lambda^T(\mathscr{N}) \end{bmatrix} P \begin{bmatrix} \Phi(\mathscr{M}) & \Lambda(\mathscr{N}) \end{bmatrix} \begin{bmatrix} e(k) \\ d(k) \end{bmatrix} + \\
&\quad \begin{bmatrix} e^T(k) & e^T(k-\tau_{max}) \end{bmatrix} \begin{bmatrix} Z_1 - P & 0 \\ 0 & -Z_1 \end{bmatrix} \begin{bmatrix} e(k) \\ e(k-\tau_{max}) \end{bmatrix} + \\
&\quad l \times \tau_{max}^2 \Delta e^T(k)Z_2\Delta e(k) - l \times \tau_{max} \sum_{j=k-\tau_{max}}^{k-1} \Delta e^T(j)Z_2\Delta e(j).
\end{aligned}$$

Defining the augmented state vector

$$\xi(k) = \begin{bmatrix} e(k) \\ e(k-\tau_1(k)) \\ e(k-\tau_2(k)) \\ \vdots \\ e(k-\tau_l(k)) \\ e(k-\tau_{max}) \end{bmatrix} = \begin{bmatrix} e(k) \\ d(k) \\ e(k-\tau_{max}) \end{bmatrix},$$

the forward difference of the Lyapunov–Krasovskii functional can be written using the following quadratic form:

$$\begin{aligned}\Delta V(k) = \xi^T(k) \Bigg(&\begin{bmatrix} Z_1 - P & 0 & 0 \\ * & 0 & 0 \\ * & * & -Z_1 \end{bmatrix} + \begin{bmatrix} \Phi^T(\mathscr{M}) \\ \Lambda^T(\mathscr{N}) \\ 0 \end{bmatrix} P \begin{bmatrix} \Phi(\mathscr{M}) & \Lambda(\mathscr{N}) & 0 \end{bmatrix} \\ &+ l \times \tau_{max}^2 \begin{bmatrix} \Phi^T(\mathscr{M}) - I \\ \Lambda^T(\mathscr{M}) \\ 0 \end{bmatrix} Z_2 \begin{bmatrix} (\Phi(\mathscr{M}) - I) & \Lambda(\mathscr{M}) & 0 \end{bmatrix} \Bigg) \xi(k) \\ &- l \times \tau_{max} \sum_{j=k-\tau_{max}}^{k-1} \Delta e^T(j) Z_2 \Delta e(j).\end{aligned}$$

In order to take into account the delay of each different communication link $(\tau_r(k), \forall r = 1, \ldots, l)$, we split the last term in the above equation (which appears l times) into 2 terms, considering the delay in each specific link:

$$\begin{aligned}-\tau_{max} \sum_{j=k-\tau_{max}}^{k-1} \Delta e^T(j) Z_2 \Delta e(j) = \\ - \tau_{max} \sum_{j=k-\tau_{max}}^{k-\tau_r(k)-1} \Delta e^T(j) Z_2 \Delta e(j) - \tau_{max} \sum_{j=k-\tau_r(k)}^{k-1} \Delta e^T(j) Z_2 \Delta e(j).\end{aligned}$$

The resulting terms can be bounded using the Jensen inequality:

$$-\tau_{max} \sum_{j=k-\tau_{max}}^{k-\tau_r(k)-1} \Delta e^T(j) Z_2 \Delta e(j) \leq -\left[\sum_{j=k-\tau_{max}}^{k-\tau_r(k)-1} \Delta e(j)\right]^T Z_2 \left[\sum_{j=k-\tau_{max}}^{k-\tau_r(k)-1} \Delta e(j)\right],$$

$$-\tau_{max} \sum_{j=k-\tau_r(k)}^{k-1} \Delta e^T(j) Z_2 \Delta e(j) \leq -\left[\sum_{j=k-\tau_r(k)}^{k-1} \Delta e(j)\right]^T Z_2 \left[\sum_{j=k-\tau_r(k)}^{k-1} \Delta e(j)\right].$$

The terms in brackets can be cancelled in pairs, except the first and the last one in the summatory, yielding:

$$\begin{aligned}-\tau_{max} \sum_{j=k-\tau_{max}}^{k-\tau_r(k)-1} \Delta e^T(j) Z_2 \Delta e(j) \leq \\ - \left[e(k-\tau_r(k)) - e(k-\tau_{max})\right]^T Z_2 \left[e(k-\tau_r(k)) - e(k-\tau_{max})\right],\end{aligned}$$

$$-\tau_{max}\sum_{j=k-\tau_r(k)}^{k-1}\Delta e^T(j)Z_2\Delta e(j)\leq$$
$$-\left[e(k)-e(k-\tau_r(k))\right]^T Z_2\left[e(k)-e(k-\tau_r(k))\right].$$

The above terms are also written in the same quadratic manner as

$$-\tau_{max}\sum_{j=k-\tau_{max}}^{k-\tau_r(k)-1}\Delta e^T(j)Z_2\Delta e(j)\leq$$
$$\left[e^T(k-\tau_r(k))\ e^T(k-\tau_{max})\right]\begin{bmatrix}-Z_2 & Z_2\\ * & -Z_2\end{bmatrix}\begin{bmatrix}e(k-\tau_r(k))\\ e(k-\tau_{max})\end{bmatrix},$$
$$-\tau_{max}\sum_{j=k-\tau_r(k)}^{k-1}\Delta e^T(j)Z_2\Delta e(j)\leq$$
$$\left[e^T(k)\ e^T(k-\tau_r(k))\right]\begin{bmatrix}-Z_2 & Z_2\\ * & -Z_2\end{bmatrix}\begin{bmatrix}e(k)\\ e(k-\tau_r(k))\end{bmatrix}.$$

Including all the terms, the forward difference is

$$\Delta V(k)\leq\xi^T(k)L_1\xi(k)$$

where

$$L_1=\xi^T(k)\left(\begin{bmatrix}\Xi & \Theta & 0\\ * & \Upsilon & \Theta^T\\ * & * & \Omega\end{bmatrix}+\begin{bmatrix}\Phi^T(\mathscr{M})\\ \Lambda^T(\mathscr{N})\\ 0\end{bmatrix}P\left[\Phi(\mathscr{M})\ \Lambda(\mathscr{N})\ 0\right]\right.$$
$$\left.+\ l\times\tau_{max}^2\begin{bmatrix}\Phi^T(\mathscr{M})-I\\ \Lambda^T(\mathscr{M})\\ 0\end{bmatrix}Z_2\left[(\Phi(\mathscr{M})-I)\ \Lambda(\mathscr{M})\ 0\right]\right)\xi(k).\quad(A.31)$$

In order to ensure the error convergence to zero, it will be demonstrated that $\Delta V(k)<0$ for all $\xi(k)\neq 0$ through the negative definiteness of L_1. Applying Schur complements, one can obtain that the previous matrix is negative definite if and only if the following holds:

$$\begin{bmatrix}\Xi & \Theta & 0 & \Phi^T(\mathscr{M}) & (\Phi^T(\mathscr{M})-I)\tau_{max}\\ * & \Upsilon & \Theta^T & \Lambda^T(\mathscr{N}) & \Lambda^T(\mathscr{N})\tau_{max}\\ * & * & \Omega & 0 & 0\\ * & * & * & -P^{-1} & 0\\ * & * & * & * & -\frac{1}{l}Z_2^{-1}\end{bmatrix}<0.$$

Finally, pre- and post-multiplying the previous matrix by *diag* $\{I, I, I, P, P\}$ and its transpose, this condition is equivalent to the one stated in Theorem 11.1. Therefore, the negative definiteness of this matrix is ensured.

A.6.2 Proof of Theorem 11.2

Consider the Lyapunov–Krasovskii functional (A.30). Including the disturbances due to the asynchronous flow of information, the forward difference takes the form

$$\Delta V(k) \leq \xi^T(k) L_1 \xi(k) + 2\varepsilon^T(k) L_2 \xi(k) + \varepsilon^T(k) L_3 \varepsilon(k).$$

From Theorem 11.1 we can ensure that matrix L_1 is negative definite, so there exists a positive matrix Q such that $L_1 < -Q$. Taking norms, the forward difference can be bounded as follows:

$$\Delta V(k) \leq -\lambda_{min}^{Q} \|\xi(k)\|_\infty^2 + 2\|L_2\|_\infty \|\varepsilon(k)\|_\infty \|\xi(k)\|_\infty + \|L_3\|_\infty \|\varepsilon(k)\|_\infty^2.$$

The triggering condition (11.12) ensures that $\|\varepsilon(k)| \leq \delta$, in such a way that

$$\Delta V(k) \leq -\lambda_{min}^{Q} \|\xi(k)\|_\infty^2 + 2\|L_2\|_\infty \|\xi(k)\|_\infty \delta + \|L_3\|_\infty \delta^2.$$

We are interested in the values of $\xi(k)$ that achieve that $\Delta V(k) \leq 0$. In order to find a feasible region, the following second-order equation in $\|\xi(k)\|$ can be solved:

$$-\lambda_{min}^{Q} \|\xi(k)\|_\infty^2 + 2\|L_2\|_\infty \|\xi(k)\|_\infty \delta + \|L_3\|_\infty \delta^2 = 0.$$

By solving the previous equation, it can be ensured that $\Delta V(k) \leq 0$ for $\|\xi(k)\|_\infty > D_1\delta$, with $D_1 = \frac{\|L_2\|_\infty + \sqrt{\|L_2\|_\infty^2 + \lambda_{min}^{Q} \|L_3\|_\infty}}{\lambda_{min}^{Q}}$.

For a generic vector x and a positive scalar D, let B_D^x denote the region of the space defined by $\{x : \|x\|_\infty \leq D\}$. Please note that the above result implies that $V(k)$ decreases for every $\xi(k) \notin B_{D_1\delta}^{\xi}$. Hence, it is obvious that there exists a time instant k^* in which $\xi(k^*)$ enters into the region $B_{D_1\delta}^{\xi}$. The augmented state vector $\xi(k)$ includes the observation error $e(k)$, so it turns out that $e(k^*) \in B_{D_1\delta}^{e}$.

As $\xi(k^*) \in B_{D_1}^{\xi}$ for any realization of $\mu_r(k) \in [0, \tau_{max}]$, $r \in \mathscr{L}$, it also holds that $\zeta(k^*) \in B_{D_1}^{\zeta}$.

From instant k^* on, the functional is not necessarily decreasing and the augmented state may jump outside the region, that is, it may occur that $\xi(k^*+1) \notin B_{D_1\delta}^{\xi}$. Using

the dynamics of the observation error given in Eq. (11.14), it is possible to bound the error at instant $k^* + 1$ by

$$\begin{aligned}\|e(k^*+1)\|_\infty &< \|\Phi\|_\infty \|e(k^*)\|_\infty + \|\Lambda\|_\infty \|d(k^*)\|_\infty + \|\Gamma\|_\infty \|\varepsilon(k^*)\|_\infty \\ &< (\|\Phi\|_\infty + \|\Lambda\|_\infty) D_1 + \|\Gamma\|_\infty \delta.\end{aligned}$$

Then $e(k^*+1) \in B^e_{D_2\delta}$, where $D_2 = (\|\Phi\|_\infty + \|\Lambda\|_\infty)D_1 + \|\Gamma\|_\infty$. This way, $\xi(k^*+1)$, and hence $\zeta(k^*+1)$, may leave the regions $B^\xi_{D_1}$ and $B^\zeta_{D_1}$, respectively. In that case, the Lyapunov–Krasovskii functional must be decreasing again, implying that

$$\begin{aligned}\forall k > k^*+1, \quad V(k) &< \max\{V(k^*+1)\} = \max\{\zeta^T(k^*+1)\Psi\zeta(k^*+1)\} \\ &< \lambda^\Psi_{max} \max\{\|\zeta(k^*+1)\|^2_\infty\} \\ &< \lambda^\Psi_{max}(D_2\delta)^2.\end{aligned}$$

Finally, to get the final bound on $e(k)$ for $k > k^*+1$, note that all the terms of the Lyapunov functional involve positive definite matrices, so

$$e(k)^T P e(k) < V(k) < \lambda^\Psi_{max}(D_2\delta)^2, \forall k > k^*+1.$$

And using well-known properties, it yields

$$\begin{aligned}\lambda^P_{min}\|e(k)\|^2_\infty < e(k)^T P e(k) &< \lambda^\Psi_{max}(D_2\delta)^2 \\ \Rightarrow \|e(k)\|_\infty &< \sqrt{\frac{\lambda^\Psi_{max}}{\lambda^P_{min}}} D_2\delta.\end{aligned}$$

Appendix B
Dealing with Nonlinear Terms in Matrix Inequalities

Sometimes, when the control problems are posed as matrix inequalities, it is inevitable that some nonlinear terms appears, so the existing methods for LMIs cannot directly be applied. This appendix proposes two different solutions for some nonlinearities that are very common both in this book and in other approaches based in Lyapunov–Krasovskii theorem. By means of appropriate transformations and additional constraints, the nonlinear matrix inequality can be replaced by a problem with linear constraints.

Consider a nonlinear matrix inequality

$$\begin{bmatrix} f_{11}(X_1,\ldots,X_m) & \cdots & f_{1k}(X_1,\ldots,X_m) & \cdots & f_{1p}(X_1,\ldots,X_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f_{1k}^T(X_1,\ldots,X_m) & \cdots & g_{kk}(X_1,\ldots,X_m) & \cdots & f_{ip}(X_1,\ldots,X_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f_{1p}^T(X_1,\ldots,X_m) & \cdots & f_{kp}^T(X_1,\ldots,X_m) & \cdots & f_{pp}(X_1,\ldots,X_m) \end{bmatrix} < 0, \qquad \text{(B.1)}$$

where f are affine functions on the decision variables $X_1,\ldots,X_m$ and g are nonlinear functions with the following particular structure:

$$g(X_1,\ldots,X_m) = -X_i X_j^{-1} X_i,\ i \neq j.$$

Note that the nonlinear function appears in the diagonal of the inequality. In the following sections, two solutions are given to deal with the nonlinearity $X_i X_j^{-1} X_i$, $i \neq j$. The first introduces an additional constraint which lets us address the problem by means of a set of linear matrix inequalities. The second solution employs the *cone complementary algorithm* to transform the nonlinear inequality into an iterative optimization problem with linear constraints. Comparing both solutions, the former could be more conservative, but it is computationally more efficient, as the number of constraints and variables is lower.

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9

B.1 Direct Constraint

Consider the introduction of the following additional constraint:

$$-X_i X_j^{-1} X_i < -\frac{1}{\mu} X_i,$$

being μ a positive design scalar. Note that previous condition is equivalent to $X_j < \mu X_i$. Then, the nonlinear constraint in Eq. (B.1) can be replaced by

$$\begin{cases} \Upsilon(X_1, \ldots, X_m) < 0, \\ X_j < \mu X_i \end{cases} \tag{B.2}$$

where Υ is the matrix required to be negative definite in (B.1), but substituting the terms $g(X_1, \ldots, X_m) = -X_i X_j^{-1} X_i$ by $-\frac{1}{\mu} X_i$.

It is worth comparing the proposed method with that introduced in [275] and used in other papers to handle the same nonlinearity. While in [275] it is directly imposed X_j to be X_i times a given scalar, this method just restricts $X_j < \mu X_i$, which covers a much wider range of possible solutions in the space of positive definite matrices. Therefore, it leads to less conservative solutions.

B.2 Cone Complementary Algorithm

Another possibility consists in using the well-known *cone complementary algorithm*. The idea is the following: first, the nonlinear inequality can be addressed by solving an optimization problem with linear constraints. Then, a solution for this problem can be found with an extended algorithm whose convergence is theoretically ensured.

Following the idea of [184], define a variable T such that

$$X_i X_j^{-1} X_i \geq T > 0, \tag{B.3}$$

which is equivalent to

$$\begin{bmatrix} -T^{-1} & X_i^{-1} \\ X_i^{-1} & -X_j^{-1} \end{bmatrix} \leq 0. \tag{B.4}$$

Now, introducing some new variables,

$$\hat{X}_i = X_i^{-1}, \quad \hat{T} = T^{-1}, \quad \hat{X}_j = X_j^{-1}, \tag{B.5}$$

Equation (B.4) can be rewritten as

$$\begin{bmatrix} -\hat{T} & \hat{X}_i \\ \hat{X}_i & -\hat{X}_j \end{bmatrix} \leq 0. \tag{B.6}$$

Now, instead of using the original nonlinear inequality (B.1), consider the following nonlinear minimization problem involving LMI conditions:

$$\text{Minimize } \mathbf{Tr}\left(\hat{X}_i X_i + \hat{X}_j X_j + \hat{T}T\right) \tag{B.7}$$

subject to

$$\begin{cases} \Upsilon(X_1, \ldots, X_m) < 0, \\ \begin{bmatrix} -\hat{T} & \hat{X}_i \\ * & -\hat{X}_j \end{bmatrix} \leq 0, \begin{bmatrix} X_i & I \\ * & \hat{X}_i \end{bmatrix} \geq 0, \begin{bmatrix} X_j & I \\ * & \hat{X}_j \end{bmatrix} \geq 0, \begin{bmatrix} T & I \\ * & \hat{T} \end{bmatrix} \geq 0, \end{cases} \tag{B.8}$$

where Υ is as before the matrix required to be definite negative in (B.1), but substituting $X_i X_j^{-1} X_i$ by T. From Eqs. (B.3), it is immediate that, if $\Upsilon < 0$, then (B.1) holds. The minimization problem is introduced to force (B.5). When the LMIs in the second row of the restrictions (B.8) saturate, the optimum is reached and (B.1) holds.

In order to solve the aforementioned minimization problem (B.7) the following algorithm introduced in [64] can be implemented:

---

1. Set $k = 0$. Find a feasible solution under the conditions in (B.8):

   $$(X_1^0, X_2^0, \ldots, X_m^0, T^0, \hat{X}_i^0, \hat{X}_j^0, \hat{T}^0)$$

   If there is no solution, exit.
2. Solve the following optimization problem with LMI constraints with decision variables $(X_1, X_2, \ldots, X_m, T, \hat{X}_i, \hat{X}_j, \hat{T})$

   $$\min \quad \mathbf{Tr}\left(\hat{X}_i^k X_i + X_i^k \hat{X}_i + \hat{X}_j^k X_j + X_j^k \hat{X}_j + \hat{T}^k T + T^k \hat{T}\right)$$

   subject to LMIs in (B.8)

   Set $X_i^{k+1} = X_i, \hat{X}_i^{k+1} = \hat{X}_i, X_j^{k+1} = X_j, \hat{X}_j^{k+1} = \hat{X}_j, \hat{T}^{k+1} = \hat{T}, T^{k+1} = T$.
3. If the condition (B.1) is satisfied, exit. Otherwise, set $k = k + 1$ and return to Step 2.

The first and second steps of the algorithm are simple LMI problems, and they can be solved efficiently using an appropriate computational software. As it is stated in Theorem 2.1 in [64], the algorithm converges and then $\hat{X}_i X_i = I, \hat{X}_j X_j = I, \hat{T}T = I$.

References

1. K.J. Åström, T. Hägglund, Automatic tuning of simple regulators with specifications on phase and amplitude margins. Automatica **20**, 645–651 (1984)
2. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
3. A. Al-Mohi, N.J. Higham, Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimate. SIAM J. Matrix Anal. Appl. **30**, 1639–1657 (2009)
4. P. Alriksson, A. Rantzer, Distributed Kalman filtering using weighted averaging, in *17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan (2006)
5. S.M. Amin, B.F. Wollenberg, Toward a smart grid: power delivery for the 21st century. IEEE Power Energy Mag. **3**(5), 34–41 (2005)
6. A. Anta, P. Tabuada, Self-triggered stabilization of homogeneous control systems, in *American Control Conference*, Seattle, WA, United States, June 2008, pp. 4129–4134
7. A. Anta, P. Tabuada, On the minimum attention and anytime attention problems for nonlinear systems, in *49th IEEE Conference on Decision and Control*, Atlanta (2010), pp. 3234–3239
8. A. Anta, P. Tabuada, To sample or not to sample: self-triggered control for nonlinear systems. IEEE Trans. Autom. Control **55**(9), 2030–2042 (2010)
9. J. Araujo, A. Anta, M. Mazo Jr., J. Faria, A. Hernandez, P. Tabuada, K.H. Johansson, Self-triggered control over wireless sensor and actuator networks, in *International Conference on Distributed Computing in Sensor Systems and Workshops*, Barcelona (2011), pp. 1–9
10. J. Araujo, Design and implementation of resource-aware wireless networked control systems. Technical report TRITA-EE 2011:065, Royal Institute of Technology (KTH), September 2011. Licentiate Thesis
11. Arduino, Website (2015). http://www.arduino.cc/
12. K.E. Arzén, A simple event-based PID controller, in *IFAC World Congress*, Beijing (1999), pp. 423–428
13. K.J. Åström, B. Wittenmark, *Computer Controlled Systems: Theory and Design*, 3rd edn. (Prentice Hall, Upper Saddle River, 1997)
14. J. Baillieul, P.J. Antsaklis, Control and communication challenges in networked real-time systems. Proc. IEEE **95**(1), 9–28 (2007)
15. L. Bakule, Decentralized control: an overview. Annu. Rev. Control **32**(1), 87–98 (2008)
16. M. Baseggio, A. Cenedese, P. Merlo, M. Pozzi, L. Schenato, Distributed perimeter patrolling and tracking for camera networks, in *49th IEEE Conference on Decision and Control*, Atlanta (2010), pp. 2093–2098

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9

17. F.L. Bauer, C.T. Fike, Norms and exclusion theorems. Numer. Math. **2**, 137–141 (1960)
18. A. Bemporad, Predictive control of teleoperated constrained systems with unbounded communication delays, in *37th Conference of Decision and Control*, Tampa (1998), pp. 2133–2138
19. A. Bemporad, M. Morari, V. Dua, E.N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems. Automatica **38**(1), 3–20 (2002)
20. O. Bernard, Hurdles and challenges for modelling and control of microalgae for CO2 mitigation and biofuel production, in *11th International IFAC Symposium on Computer Applications in Biotechnology*, Leuven (2010)
21. M. Beschi, S. Dormido, J. Sánchez, A. Visioli, Characterization of symmetric send-on-delta PI controllers. J. Process Control (2012)
22. M. Beschi, S. Dormido, J. Sánchez, A. Visioli, Tuning rules for event-based SSOD-PI controllers, in *20th Mediterranean Conference on Control and Automation*, Barcelona (2012)
23. R. Blind, F. Allgöwer, On the optimal sending rate for networked control systems with a shared communication medium, in *50th IEEE Conference on Decision and Control*, Orlando (2011), pp. 4704–4709
24. S. Bolognani, S. Zampieri, A gossip-like distributed optimization algorithm for reactive power flow control, in *IFAC World Congress*, Milano (2011), pp. 5700–5705
25. M.S. Branicky, S.M. Phillips, W. Zhang, Stability of networked control systems: explicit analysis of delay, in *American Control Conference* (2000), pp. 2352–2357
26. J.H. Braslavsky, R.H. Middleton, J.S. Freudenberg, Feedback stabilization over signal-to-noise ratio constrained channels. Proc. Am. Control Conf. **179**(36), 4903–4908 (2004)
27. S.E. Butner, M. Ghodoussi, Transforming a surgical robot for human telesurgery. IEEE Trans. Robot. Autom. **19**(5), 818–824 (2003)
28. A. Camacho, P. Martí, M. Velasco, C. Lozoya, R. Villa, J.M. Fuertes, E. Griful, Self-triggered networked control systems: an experimental case study, in *IEEE 2010 International Conference on Industrial Technology*, Valparaiso (2010), pp. 123–128
29. E.F. Camacho, F.R. Rubio, M. Berenguel, L. Valenzuela, A survey on control schemes for distributed solar collector fields. Part I: modeling and basic control approaches. Sol. Energy **81**(10), 1240–1251 (2007)
30. E.F. Camacho, F.R. Rubio, M. Berenguel, L. Valenzuela, A survey on control schemes for distributed solar collector fields. Part II: advanced control approaches. Sol. Energy **81**(10), 1252–1272 (2007)
31. F. Camacho, M. Berenguel, F.R. Rubio, *Advanced Control of Solar Plants* (Springer, Berlin, 1997)
32. A. Casavola, E. Mosca, M. Papini, Predictive teleoperation of constrained dynamic systems via internetlike channels. IEEE Trans. Control Syst. Technol. **14**(4), 681–694 (2006)
33. A. Cervin, K.J. Åström, On Limit Cycles in Event-Based Control System, in *46th Conference on Decision and Control*, December 2007, pp. 3190–3195
34. A. Cervin, T. Henningsson, Scheduling of event-triggered controllers on a shared network, in *47th IEEE Conference on Decision and Control*, Cancun (2008), pp. 3601–3606
35. J. Chacon, J. Sanchez, A. Visioli, L. Yebraa, S. Dormido, Characterization of limit cycles for self-regulating and integral processes with pi control and send-on-delta sampling. J. Proc. Control **23**(6), 826–838 (2003)
36. A. Chaillet, A. Bicchi, Delay compensation in packet switching networked controlled systems, in *47th IEEE Conference on Decision and Control*, Cancun (2008), pp. 3620–362
37. W. Che, J. Wang, G. Yang, Observer-based h-infinity control in multiple channel networked control systems with random packet dropouts. J. Control Theory Appl. **8**(3):359–367 (2010). Control methods; Data missing; Disturbance attenuation levels; H-infinity; H-infinity control; H-infinity controller; Limited communication; Mean square; Multiple channels; Networked control system (NCS); Networked control systems; Numerical example; Packet dropouts; Stochastic variable
38. J.D. Chen, LMI approach to robust delay-dependent mixed H_2/H_∞ controller of uncertain neutral systems with discrete and distributed time-varying delays. J. Optim. Theory Appl. **131**(3), 383–403 (2006)

39. W.H. Chen, Z.H. Guan, X. Lu, Delay-dependent guaranteed cost control for uncertain discrete-time systems with delay. IEE Proc.—Control Theory Appl. **150**(4), 412–416 (2003)
40. W.H. Chen, Z.H. Guan, X.M. Lu, Delay-dependent output feedback guaranteed cost control for uncertain time-delay systems. Automatica **40**, 1263–1268 (2004)
41. X. Chen, F. Hao, Event-triggered average consensus control for discrete-time multi-agent systems. IET Control Theory Appl. **6**(16), 2493–2498 (2012). Average consensus; Consensus control; Event-triggered; LMI toolboxes; Multi agent system (MAS); Simulation example; Sufficient conditions; Theoretical result
42. Y.Q. Chen, Z. Wang. IEEE/RSJ international conference on formation control: a review and a new consideration, in *Intelligent Robots and Systems* (2005)
43. C.F. Chiasserini, E. Magli, Energy consumption and image quality in wireless video-surveillance networks, in *13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 5 (IEEE, 2002), pp. 2357–2361
44. K.W.E. Chu, Generalization of the bauer-fike theorem. Numerische Mathematik **49**, 685–691 (1986)
45. M. Cloosterman, N. Van de Wouw, W.P. Heemels, H. Nijmeijer, Stability of networked control systems with uncertain time-varying delays. IEEE Trans. Autom. Control **54**(7), 1575–1580 (2009)
46. R. Cogill, Event-based control using quadratic approximate value functions, in *48th IEEE Conference on Decision and Control*, Shangai, China, December 2009, pp. 5883–5888
47. N. Correll, G. Sempo, Y. Lopez de Meneses, J. Halloy, J.L. Deneubourg, Swistrack: a tracking tool for multi-unit robotic and biological systems, in *International Conference on Intelligent Robots and Systems*, Beijing (2006)
48. D. Limon, T. Alamo, E.F. Camacho, Stability analysis of systems with bounded additive uncertainties based on invariant sets: stability and feasibility of MPC, in *Proceedings of American Control Conference*, Anchorage, Alaska, USA (2002)
49. D. Muñoz, C. Panagiotis, Estimation-based networked predictive control of nonlinear systems. Dyn. Contin. Discret. Impuls. Syst. **14**, 52–58 (2007)
50. M.C. de Oliveira, J. Bernussou, J.C. Geromel, A new discrete-time robust stability condition. Syst. Control Lett. **37**(4), 261–265 (1999)
51. C. De Persis, R. Sailer, F. Wirth, On a small-gain approach to distributed event-triggered control, in *18th IFAC World Congress*, Milano (2011), pp. 2401–2406
52. O. Demir, J. Lunze, Cooperative control of multi-agent systems with event-based communication, in *American Control Conference*, Montreal (2012), pp. 4504–4509
53. A. Diaz-Guilera, A. Arenas, Phase patterns of coupled oscillators with application to wireless communication, *Bio-Inspired Computing and Communication* (Springer, Berlin, 2008)
54. D.V. Dimarogonas, E. Frazzoli, K.H. Johansson, Distributed event-triggered control for multi-agent systems. IEEE Trans. Autom. Control **57**(5), 1291–1297 (2012)
55. H. Dong, Z. Wang, H. Gao, Distributed filtering for a class of time-varying systems over sensor networks with quantization errors and successive packet dropouts. IEEE Trans. Signal Process. **60**(6), 3164–3173 (2012)
56. M.C.F. Donkers, W.P.M.H. Heemels, Output-based event-triggered control with guaranteed L_∞-gain and improved event-triggering, in *IEEE Conference on Decision and Control*, Atlanta (2010), pp. 3246–3251
57. M.C.F. Donkers, W.P.M.H. Heemels, Output-based event-triggered control with guaranteed L_∞-gain and improved and decentralised event-triggering. Trans. Autom. Control **57**(6), 1362–1376 (2012)
58. M.C.F. Donkers, P. Tabuada, W.P.M.H. Heemels, On the minimum attention control problem for linear systems: a linear programming approach, in *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)* (2011), pp. 4717–4722
59. P.M. Doran, *Bioprocess Engineering Principles* (Elsevier Science & Technology Books, 1997)
60. R.C. Dorf, M.C. Farren, C.A. Phillips, Adaptive sampling for sampled-data control systems. IEEE Trans. Autom. Control **7**(1), 34–47 (1962)

61. S. Dormido, J. Sánchez, E. Kofman, Muestreo, control y comunicación basado en eventos. Revista Iberoamericana de Automática e Informática Industrial **5**(1), 5–26 (2008)
62. D.Q. Mayne, J.B. Rawlings, C.V. Rao, Constrained model predictive control: stability and optimality. Automatica **36**, 789–814 (2000)
63. N.A. Duffie, An approach to the design of distributed machinery control systems. IEEE Trans. Ind. Appl. **18**(4), 435–442 (1982)
64. L. El Ghaoui, F. Oustry, M. AitRami, A cone complementary linearization algorithm for static output-feedback and related problems. IEEE Trans. Autom. Control **42**(8), 1171–1176 (1997)
65. N. Elia, S.K. Mitter, Stabilization of linear systems with limited information. IEEE Trans. Autom. Control **46**(9), 1384–1400 (2001)
66. P. Ellis, Extension of phase plane analysis to quantized systems. IRE Trans. Autom. Control **4**(2), 43–54 (1959)
67. P.H. Enslow Jr, What is a distributed data processing system? Computer **11**(1), 13–21 (1978)
68. M. Epstein, L. Shi, S. di Cairano, R.M. Murray, Control over a network: using actuation buffers to reduce transmission frequency, in *European Control Conference*, Kos, 2007
69. F. Esquembre, Easy java simulations: a software tool to create scientific simulations in java. Comput. Phys. Commun. **156**(2), 199–204 (2004)
70. E. Fabregas, Plataformas de experimentación virtual y remota: aplicaciones de control y robótica. Ph.D thesis, UNED (2013)
71. Y. Fan, G. Feng, Y. Wang, C. Song, Distributed event-triggered control of multi-agent systems with combinational measurements. Automatica **49**(2), 671–675 (2013). 3-D space; Continuous measurements; Control performance; Convergence analysis; Event-triggered; Hybrid control systems; Multi agent system (MAS); Numerical example;
72. M. Farina, G. Ferrari-Trecate, R. Scattolini, Distributed moving horizon estimation for sensor networks, in *1st IFAC Workshop on Estimation and Control of Networked Systems* (Venice, Italy, 2009), pp. 126–131
73. J.A. Fax, R.M. Murray, Information flow and cooperative control of vehicle formations. IEEE Trans. Autom. Control **49**(9), 1465–1476 (2004)
74. G.F. Franklin, J.D. Powell, M. Workman, *Digital Control of Dynamic Systems* (Addison-Wesley, E.U.A., 1997)
75. E. Fridman, A. Seuret, J.P. Richard, Robust sampled-data stabilization of linear systems: an input delay approach. Automatica **40**(9), 1441–1446 (2004)
76. E. Fridman, U. Shaked, Stability and guaranteed cost control of uncertain discrete delay systems. Int. J. Control **78**(4), 235–246 (2005)
77. H. Gao, T. Chen, New results on stability of discrete-time systems with time-varying state delay. IEEE Trans. Autom. Control **52**(2), 328–334 (2007)
78. E. Garcia, P.J. Antsaklis, Model-based event-triggered control with time-varying network delays, in *50th IEEE Conference on Decision and Control*, Orlando (2011), pp. 1650–1655
79. E. Garcia, P.J. Antsaklis, Decentralized model-based event-triggered control of networked systems, in *American Control Conference*, Montreal (2012), pp. 6485–6490
80. D. Georgiev, D.M. Tilbury, Packet-based control. Am. Control Conf. **1**, 329–336 (2004)
81. J.M. Gonçalves, *Constructive global analysis of hybrid systems*. Ph.D. thesis, Massachussetts Institute of Technology (2000)
82. J.M. Gonçalves, Regions of stability for limit cycle oscillations in piecewise linear systems. IEEE Trans. Autom. Control **50**(11), 1877–1882 (2005)
83. G.P. Liu, J.X. Mu, D. Rees, S.C. Chai, Design and stability analysis of networked control systems with random communication time delay using the modified MPC. Int. J. Control **79**, 288–297 (2006)
84. L. Greco, D. Fontanelli, A. Bicchi, Design and stability analysis for anytime control via stochastic scheduling. IEEE Trans. Autom. Control **56**(3), 571–585 (2011)
85. L. Greco, A. Chaillet, A. Bicchi, Exploiting packet size in uncertain nonlinear networked control systems. Automatica **48**(11), 2801–2811 (2012)
86. B. Grocholsky, J. Keller, V. Kumar, G. Pappas, Cooperative air and ground surveillance. Robot. Autom. Mag. IEEE **13**(3), 16–25 (2006)

87. J.A. Gubner, Distributed estimation and quantization. IEEE Trans. Inf. Theory **39**(4), 1465–1467 (1993)
88. M. Guinaldo, D.V. Dimaragonas, K.H. Johansson, J. Sánchez, S. Dormido, Distributed event-based control for interconnected linear systems, in *50th Control and Decision Conference*, Orlando (2011), pp. 2553–2558
89. M. Guinaldo, D.V. Dimarogonas, K.H. Johansson, J. Sánchez, S. Dormido, Distributed event-based control strategies for interconnected linear systems. IET Control Theory Appl. (2013)
90. M. Guinaldo, D Lehmann, J. Sánchez, S. Dormido, K.H. Johansson, Distributed event-triggered control with network delays and packet-losses, in *51th IEEE Conference on Decision and Control*, Maui (2012), pp. 1–6
91. M. Guinaldo, D. Lehmann, J. Sanchez, S. Dormido, K.H. Johansson, Reducing communication and actuation in distributed control systems, in *52nd Annual Conference on Decision and Control (CDC)*, December 2013, pp. 5288–5293
92. M. Guinaldo, J. Sánchez, S. Dormido, A co-design strategy of NCS for treacherous network conditions. IET Control Theory Appl. **5**(16), 1906–1915 (2011)
93. V. Gupta, On an anytime algorithm for control, in *47th IEEE Conference on Decision and Control*, Cancun (2009), pp. 6218–6223
94. V. Gupta, D.E. Quevedo, On anytime control of nonlinear processes through calculation of control sequences, in *IEEE Conference on Decision and Control*, Atlanta (2010), pp. 7564–7569
95. Y. Halevi, A. Ray, Performance analysis of integrated communication and control system networks. Trans. ASME. J. Dyn. Syst. Meas. Control **112**(3), 365–371 (1990)
96. Y. He, M. Wu, G.P. Liu, J.H. She, Output feedback stabilization for a discrete-time system with a time-varying delay. IEEE Trans. Autom. Control **53**(10), 2372–2377 (2008)
97. M. Donkers, W. Heemels, A. Teel, Periodic event-triggered control for linear systems. IEEE Trans. Autom. Control **58**(4), 847–861 (2013)
98. W.P.M.H. Heemels, R.J.A. Gorter, A. van Zijl, P.P.J. van den Bosch, S. Weiland, W.H.A. Hendrix, M.R. Vonder, Asynchronous measurement and control: a case study on motor synchronization. Control Eng. Pract. **7**(12), 1467–1482 (1999)
99. W.P.M.H. Heemels, J.H. Sandee, P.P.J. van den Bosch, Analysis of event-driven controllers for linear systems. Int. J. Control **81**(4), 571–590 (2008)
100. W.P.M.H. Heemels, M.C.F. Donkers, Model-based periodic event-triggered control for linear systems. Automatica **49**(3), 698–711 (2013)
101. W.P.M.H. Heemels, K.H. Johansson, P. Tabuada, An introduction to event-triggered and self-triggered control, in *51st IEEE Conference on Decision and Control*, Maui (2012), pp. 3270–3285
102. W.P.M.H. Heemels, N. van de Wouw, Stability and stabilization of networked control systems. Lect. Notes Control Inf. Sci. **406**, 203–253 (2010)
103. W. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application specific protocol architecture for wireless microsensor networks. IEEE Trans. Wirel. Commun. **1**(4), 660–670 (2002)
104. J. Hespanha, P. Naghshtabrizi, Y. Xu, A survey of recent results in networked control systems. Proc. IEEE **95**(1), 138–162 (2007)
105. J. Hespanha, A. Ortega, L. Vasudevan, Towards the control of linear systems with minimum bit-rate, in *Proceedings of the International Symposium on the Mathematical Theory of Networks and System* (2002)
106. J.P. Hespanha, M. McLaughlin, G.S. Sukhatme, M. Akbarian, R. Garg, W. Zhu, Haptic collaboration over the internet, in *5th PHANTOM Users Group Workshop*, vol. 40 (2000)
107. L. Hetel, J. Daafouz, C. Iung, Analysis and control of LTI and switched systems in digital loops via an event-based modelling. Int. J. Control **81**(7), 1125–1138 (2008)
108. N.J. Higham, *Functions of Matrices: Theory and Computation* (Society for Industrial and Applied Mathematics, Philadelphia, 2008)
109. A. Horch, A.J. Isaksson, Assessment of the sampling rate in control systems. Control Eng. Pract. **9**, 533–544 (2001)

110. J. Hu, Y. Hong, Leader-following coordination of multi-agent systems with coupling time delays. Phys. A **374**, 853–863 (2007)
111. L. Hu, D, Evans, Localization for mobile sensor networks, in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking* (ACM, 2004), pp. 45–57
112. S. Hu, X. Yin, Y. Zhang, E.G. Tian, Event-triggered guaranteed cost control for uncertain discrete-time networked control systems with time-varying transmission delays. IET Control Theory Appl. **6**(18), 2793–2804 (2012). Event-triggered control systems; Event-triggering schemes; Guaranteed cost control; Guaranteed cost controller; Linear matrix inequality techniques; Networked Control Systems (NCSs); Optimisation techniques; Quadratic cost functions
113. HydroBioNets. Seventh framework programme. Technical report, European Commission, 2011-2014
114. FeedBack Instruments. Data Sheet: 33-041 Coupled Tank System for Matlab (2012)
115. G. Irwin, J. Chen, A. McKernan, W. Scanlon, Co-design of predictive controllers for wireless network control. IET Control Theory Appl. **4**(2), 186–196 (2011)
116. A. Jaimes, S. Kota, J. Gomez, An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles UAV(s), in *IEEE International Conference on System of Systems Engineering* (IEEE, 2008), pp. 1–6
117. E.D. Jensen, *Decentralized Control, Distributed Systems: An Advanced Course* (Springer, Berlin, 1981)
118. E.D. Jensen, W.E. Boebert, Partitioning and assignment of distributed processing software, in IEEE COMPCON (1976), pp. 490–506
119. X. Jiang, Q.L. Han, Delay-dependent robust stability for uncertain linear systems with interval time-varying delay. Automatica **42**(6), 1059–1065 (2006)
120. X. Jiang, Q.L. Han, S. Liu, A. Xue, A new H_∞ stabilization criterion for networked control systems. IEEE Trans. Autom. Control **53**(4), 1025–1032 (2008)
121. Y. Jianyong, Y. Shimin, W. Haiqing, Survey on the performance analysis of networked control systems, in *IEEE International Conference on Systems, Man and Cybernetics* (The Haghe, Netherlands, 2004), pp. 5068–5073
122. X.J. Jing, D.L. Tan, Y.C. Wang, An LMI approach to stability of systems with severe time-delay. IEEE Trans. Autom. Control **49**, 1192–1195 (2004)
123. D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, *Mobile Computing* (Kluwer Academic Publishers, Boston, 1996)
124. K.H. Johansson, The quadruple-tank process: a multivariable laboratory process with an adjustable zero. IEEE Trans. Control Syst. Technol. **8**(3), 456–465 (2000)
125. C.Y. Kao, B. Lincoln, Simple stability criteria for systems with time-varying delays. Automatica **40**, 1492–1434 (2004)
126. T. Kato, *Perturbation Theory for Linear Operators* (Springer, Berlin, 1966)
127. M. Khatir, E.J. Davison, *Cooperative Control of Large Systems* (Springer, Berlin, 2006)
128. A.Y. Kibangou, Distributed estimation over unknown fading channels, in *IFAC Proceedings Volumes (IFAC-PapersOnline)* (2010), pp. 317–322
129. D.S. Kim, Y.S. Lee, W.H. Kwon, H.S. Park, Maximum allowable delay bounds of networked control systems. Control Eng. Pract. **11**(11), 1301–1313 (2003)
130. J.H. Kim, Robust mixed H_2/H_∞ control of time-varying delay systems. Int. J. Syst. Sci. **32**(11), 1345–1351 (2001)
131. J.H. Kim, Delay-dependent robust H_∞ control for discrete-time uncertain singular systems with interval time-varying delays in state and control input. J. Frankl. Inst. **347**(9), 1704–1722 (2010)
132. W.-J. Kim, K. Ji, A. Ambike, Real-time operating environment for networked control systems. IEEE Trans. Autom. Sci. Eng. **3**(3), 287–296 (2006)
133. C. Ko, B. Chen, J. Chen, *Creating Web-based Laboratories* (Springer, New York, 2004)
134. J. Ko, S. Dawson-Haggerty, J. Hui, D. Culler, P. Levis, A. Terzis, Connecting low power and lossy networks to the internet. IEEE Commun. Mag.: Recent Adv. IETF Stand. **49**(4), 96–101 (2011)

135. E. Kofman, J.H. Braslavsky, Level crossing sampling in feedback stabilization under data-rate constraints. Technical report, ACR Centre for Complex Dynamic Systems and Control, The University of Newcastle, Callaghan, Australia (2006)
136. H. Kopetz, Real time in distributed real time systems, in *Distributed Computer Control Systems* (1984), pp. 11–15
137. G. Lafferriere, A. Williams, J. Caughman, J.J.P. Veerman, Decentralized control of vehicle formations. Syst. Control Lett. **54**(9), 899–910 (2005)
138. J.R.T. Lawton, R.W. Beard, B.J. Young, A decentralized approach to formation maneuvers. IEEE Trans. Robot. Autom. **19**(6), 933–941 (2003)
139. D. Lehmann, Event-based state-feedback control. Ph.D. thesis, University of Bochum (2011)
140. D. Lehmann, K.H. Johansson, Event-triggered PI control subject to actuator saturation, in *IFAC Conference on Advances in PID Control* (2012)
141. D. Lehmann, J. Lunze, Event-based output-feedback control, in *19th Mediterranean Conference on Control and Automation*, Corfu (2011), pp. 982–987
142. D. Lehmann, J. Lunze, Event-based control with communication delays and packet losses. Int. J. Control **85**(5), 566–577 (2012)
143. L. Li, M.D. Lemmon, Weakly coupled event triggered output feedback control in wireless networked control systems, in *Allerton Conference on Communication, Control and Computing* (University of Illinois—Urbana-Champaign, 2011)
144. S. Li, Z. Wang, Y. Sun, Fundamental problems of networked control system from the view of control and scheduling. IEEE Conf. Control Theory Appl. **2002**, 2503–2508 (2002)
145. X. Li, C.E. Souza, LMI approach to delay-dependent robust stability and stabilization of uncertain linear delay system, in *34th Conference on Decision and Control*, New Orleans, December 1995, pp. 3614–3619
146. X.-G. Li, A. Cela, S.I. Niculescu, A. Reama, Some problems in the stability of networked-control systems with periodic scheduling. Int. J. Control **83**(5), 996–1008 (2010)
147. Z. Li, Z. Duan, L. Huang, Leader-follower consensus of multi-agent systems, in *American Control Conference* (2009), pp. 3256–3261
148. J. Liang, B. Shen, H. Dong, J. Lam, Robust distributed state estimation for sensor networks with multiple stochastic communication delays. Int. J. Syst. Sci. **42**(9), 1459–1471 (2011)
149. J. Liang, Z. Wang, X. Liu, Distributed state estimation for uncertain Markov-type sensor networks with mode-dependent distributed delays. Int. J. Robust Nonlinear Control **22**(3), 331–346 (2012)
150. D. Liberzon, J.P. Hespanha, Stabilization of nonlinear systems with limited information feedback. IEEE Trans. Autom. Control **50**(6), 910–915 (2005)
151. L.-L. Liu, Q.-L. Zhang, Stability analysis of non-linear real-time network control systems. Dongbei Daxue Xuebao/J. Northeast. Univ. **29**(3), 305–307 (2008)
152. Z. Liu, Z. Chen, Z. Yuan, Event-triggered average-consensus of multi-agent systems with weighted and direct topology. J. Syst. Sci. Complex. **25**(5), 845–855 (2012). Average-consensus; consensus; digraph; Event-triggered; Multi agent system (MAS)
153. T. Lochmatter, P. Roduit, C. Cianci, N. Correll, Swistrack—a flexible open source tracking software for multi-agent systems, in *International Conference on Intelligent Robots and Systems*, Nice (2008)
154. J. Lunze, D. Lehmann, A state-feedback approach to event-based control. Automatica **46**, 211–215 (2010)
155. T. Luzyanina, K. Engelborghs, D. Roose, Computing stability of differential equations with bounded distributed delays. Numer. Algorithms **34**, 41–66 (2003)
156. S. Ma, C. Zhang, Z. Cheng, Delay-dependent robust h_∞ control for uncertain discrete-time singular systems with time-delays. J. Comput. Appl Math. **217**, 194–211 (2010)
157. J.M. Maestre, P. Giselsson, A. Rantzer, Distributed receding horizon Kalman filter, in *49th IEEE Conference on Decision and Control*, Atlanta, December 2010, pp. 5068–5073
158. M.R. Mahfouz, G. To, M.J. Kuhn, Smart instruments: wireless technology invades the operating room, in *IEEE Topical Conference on Biomedical Wireless Technologies, Networks, and Sensing Systems* (2012), pp. 33–36

159. M.S. Mahmoud, A. Ismail, Role of delays in networked control systems. Proc. IEEE Conf. Control Syst. **153**(4), 40–43 (2003)
160. P. di Marco, Modeling and design of multi-hop energy efficient wireless networks for control applications. Licentiate thesis, Royal Institute of Technology (KTH) (2010)
161. N.C. Martins, M.A. Dahleh, Feedback control in the presence of noisy channels: "Bode-like" fundamental limitations of performance. IEEE Trans. Autom. Control **53**(7), 1604–1615 (2008)
162. I. Mas, Cluster space framework for multi-robot formation control. Ph.D. dissertation, Santa Clara University, School of Engineering (2011)
163. M. Mazo, A. Anta, P. Tabuada, On self-triggered control for linear systems: guarantees and complexity, in *European Control Conference* (2009)
164. M. Mazo, A. Anta, P. Tabuada, An ISS self-triggered implementation of linear controllers. Automatica **46**(8), 1310–1314 (2010)
165. M. Mazo, P. Tabuada, On event-triggered and self-triggered control over sensor/actuator networks, in *47th IEEE Conference on Decision and Control*, Cancun (2008), pp. 435–440
166. M. Mazo, P. Tabuada, Decentralized event-triggered control over wireless sensor/actuator networks. IEEE Trans. Autom. Control **56**(10), 2456–2461 (2011)
167. C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, Z. Tian, Remote surgery case: robot-assisted teleneurosurgery. IEEE Int. Conf. Robot. Autom. **1**, 819–823 (2004)
168. W. Michiels, V. van Assche, S.I. Niculescu, Stabilization of time-delay systems with a controlled time-varying delay and applications. IEEE Trans. Autom. Control **50**, 493–504 (2005)
169. P. Millán, I. Jurado, C. Vivas, F.R. Rubio, Networked predictive control of systems with large data dropouts, in *47th IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008, pp. 2704–2709
170. P. Millán, L. Orihuela, G. Bejarano, C. Vivas, T. Alamo, F.R. Rubio, Design and application of suboptimal mixed H_2/H_∞ controllers for networked control systems. IEEE Trans. Control Syst. Technol. **20**(4), 1057–1065 (2012)
171. P. Millán, L. Orihuela, I. Jurado, C. Vivas, F.R. Rubio, Distributed estimation in networked systems under periodic and event-based communication policies. Int. J. Syst. Sci. (2013). In press
172. P. Millán, L. Orihuela, D. Muñoz de la Peña, C. Vivas, F.R. Rubio, Self-triggered sampling selection based on quadratic programming, in *18th IFAC World Congress*, Milano, Italy, August–September 2011, pp. 8896–8901
173. P. Millán, L. Orihuela, C. Vivas, F.R. Rubio, Improved delay-dependent stability for uncertain networked control systems with induced time-varying delays, in *1st IFAC Workshop on Estimation and Control of Networked Systems* (Italy, Venice, September, 2009), pp. 346–351
174. P. Millán, L. Orihuela, C. Vivas, and F. R. Rubio. An optimal control L_2-gain disturbance rejection design for networked control systems, in *American Control Conference*, Baltimore, Maryland, USA, June–July 2010, pp. 1344–1349
175. P. Millán, L. Orihuela, C. Vivas, F.R. Rubio, Distributed consensus-based estimation considering network induced delays and dropouts. Automatica **48**(10), 2726–2729 (2012)
176. P. Millán, L. Orihuela, C. Vivas, F.R. Rubio, Control óptimo-L_2 basado en red mediante funcionales de Lyapunov-Krasovskii. Revista Iberoamericana de Automática e Informática Industrial **9**(1), 14–23 (2012)
177. P. Millan, L. Orihuela, I. Jurado, F.R. Rubio, Formation control of autonomous underwater vehicles subject to communication delays. IEEE Trans. Control Syst. Technol. **22**(2), 770–777 (2014). Autonomous underwater vehicles (AUVs); Communication delays; Feed-forward controllers; Formation control; Formation control problems; Inter vehicle communications; Of autonomous underwater vehicles; Underwater environments
178. J.A. Misener, S.E. Shladover, Path investigations in vehicle-roadside cooperation and safety: a foundation for safety and vehicle-infrastructure integration research, in *IEEE Intelligent Transportation Systems Conference, ITSC'06* (IEEE, 2006), pp. 9–16
179. Y. Mo, R. Ambrosino, B. Sinopoli, Sensor selection strategies for state estimation in energy constrained wireless sensor networks. Automatica **47**(7), 1330–1338 (2011)

180. Adept MobileRobots. Website (2013). http://www.mobilerobots.com/
181. L.A. Montestruque, P. Antsaklis, On the model-based control of networked systems. Automatica **39**(10), 1837–1843 (2003)
182. L.A. Montestruque, P. Antsaklis, Stability of model-based networked control systems with time varying transmission times. IEEE Trans. Autom. Control **49**(9), 1562–1572 (2004)
183. L.A. Montestruque, P. Antsaklis, State and output feedback in model-based networked control systems, in *IEEE Conference on Decision and Control*, Las Vegas (2002)
184. Y.S. Moon, P. Park, W.H. Kwon, Y.S. Lee, Delay-dependent robust stabilization of uncertain state-delayed systems. Int. J. Control **74**(14), 1447–1455 (2001)
185. mOway. moway user manual v2.1.0. Website, June 2010. http://www.adrirobot.it/moway/pdf/mOway20User20Manua202.1.0.pdf
186. mOway. Website (2013). http://moway-robot.com/en/
187. R.M. Murray, Recent research in cooperative control of multivehicle systems. J. Dyn. Syst. Meas. Control **129**(5), 571–583 (2007)
188. R.M. Murray, K.J. Astrom, S.P. Boyd, R.W. Brockett, G. Stein, Future directions in control in an information-rich world. IEEE Control Syst. Mag. **23**(2), 20–33 (2003)
189. P. Naghshtabrizi, J. Hespanha, Designing an observer-based controller for a network control system, in *44th IEEE Conference on Decision and Control and the European Control Conference*, Seville, Spain, December 2005, pp. 848–853
190. P. Naghshtabrizi, J.P. Hespanha, Anticipative and non-anticipative controller design for network control systems. Netw. Embed. Sens. Control **331**, 203–218 (2006)
191. G. Nair, R. Evans, I.M.Y. Mareels, Topological feedback entropy and nonlinear stabilization. IEEE Trans. Autom. Control **49**(9), 1585–1597 (2004)
192. S. Nethi, M. Pohjola, L. Eriksson, R. Jäntt, *Platform for emulating networked control systems in laboratory environments, in International Symposium on a World of Wireless* (Mobile and Multimedia Networks, Helsinki, 2007)
193. D. Nešić, A.R. Teel, Input-to-state stability of networked control systems. Automatica **40**(12), 2121–2128 (2004)
194. W. Ni, D. Cheng, Leader-following consensus of multi-agent systems under fixed and switching topologies. Syst. Control Lett. **35**(3–4), 209–217 (2010)
195. J. Nocedal, S.J. Wright, *Numerical Optimization*, 2nd edn. (Springer, Berlin, 2006)
196. N.H. Norsker, M.J. Barbosa, M.H. Vermuë, R.H. Wijffels, Microalgal production–a close look at the economics. Biotechnol. Adv. **29**, 24–27 (2011)
197. ns-2 the Network Simulator. Website (2012). http://nsnam.isi.edu/nsnam/index.php
198. P. Ogren, E. Fiorelli, N.E. Leonard, Cooperative control of mobile sensor networks: adaptive gradient climbing in a distributed environment. IEEE Trans. Autom. Control **49**(8), 1292–1302 (2004)
199. L.R. Leon Ojeda, A.Y. Kibangou, C. Canudas De Wit, Online dynamic travel time prediction using speed and flow measurements, in *European Control Conference* (2013)
200. R. Olfati-Saber, Distributed Kalman filtering for sensor networks, in *46th IEEE Conference on Decision and Control*, New Orleans, December 2007, pp. 5492–5498
201. R. Olfati-Saber, J.A. Fax, R.M. Murray, Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
202. R. Olfati-Saber, R.M. Murray, Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Autom. Control **49**(9), 1520–1533 (2004)
203. R. Olfati-Saber, J. Shamma, Consensus filters for sensor networks and distributed sensor fusion, in *44th IEEE Conference on Decision and Control and the European Control Conference*, Seville, Spain, December 2005, pp. 6698–6703
204. L. Orihuela, P. Millán, C. Vivas, F.R. Rubio, Robust stability of nonlinear time-delay systems with interval time-varying delay. Int. J. Robust Nonlinear Control **21**(7), 709–724 (2011)
205. L. Orihuela, P. Millán, C. Vivas, F.R. Rubio, Reduced-order H_2/H_∞ distributed observer for sensor networks. Int. J. Control (2014). doi:10.1080/00207179.2013.798746
206. L. Orihuela, P. Millan, C. Vivas, F.R. Rubio, Reduced-order H2/H distributed observer for sensor networks. Int. J. Control **86**(10), 1870–1879 (2013)

207. L. Orihuela, P. Millan, C. Vivas, F.R. Rubio, Distributed control and estimation scheme with applications to process control. IEEE Trans. Control Syst. Technol. (2014)
208. P. Millan, I. Jurado, C. Vivas, F.R. Rubio, Networked predictive control of systems with large data dropouts, in *Proceedings of the 47th IEEE Conference on Decision and Control (CDC)* (2008)
209. C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in *2nd IEEE Workshop on Mobile Computing Systems and Applications* (1997), pp. 90–100
210. D. Quevedo, E.I. Silva, G. Goddwin, Packetized predictive control over erasure channels, in *American Control Conference*, Portland (2007), pp. 1003–1008
211. D.E. Quevedo, E.I. Silva, G.C. Goodwin, Control over unreliable networks affected by packet erasures and variable transmission delays. IEEE J. Sel. Areas Commun. **26**(4), 672–685 (2008)
212. M. Rabi, K.H. Johansson, Scheduling packets for event-triggered control, in *European Control Conference*, Budapest (2009)
213. D.M. Raimondo, P. Hokayem, J. Lygeros, M. Morari, An iterative decentralized MPC algorithm for large-scale nonlinear systems. Estim. Control Netw. Syst. **1**(1), 162–167 (2009)
214. K. Åström, B. Bernhardsson, Systems with Lebesgue sampling, in *Directions in Mathematical Systems Theory and Optimization*, Lecture Notes in Control and Information Sciences, ed. by A. Rantzer, C. Byrnes (Springer, Berlin, 2003), pp. 1–13
215. K.J. Åström, T. Hägglund, *Advanced PID Control* (ISA, 2006)
216. K.J. Åström, B. Wittenmark, *Adaptive Control*, 2nd edn. (Addison-Wesley, Reading, 1995)
217. H. Rehbinder, M. Sanfridson, Scheduling of a limited communication channel for optimal control. Automatica **40**(3), 491–500 (2004)
218. W. Ren, E. Atkins, Distributed multi-vehicle coordinated control via local information exchange. Int. J. Robust Nonlinear Control **17**(10), 1002–033 (2007)
219. W. Ren, R. Beard, E. Atkins, Information consensus in multivehicle cooperative control. IEEE Control Syst. Mag. **27**(2), 71–82 (2007)
220. W. Ren, K. Moore, Y. Chen, High-order consensus algorithms in cooperative vehicle systems, in *International Conference on Networking, Sensing and Control* (2006), pp. 457–462
221. J. Rice, A simple and effective method for predicting travel times on freeways. IEEE Trans. Intell. Trans. Syst. **5**(3), 200–207 (2004)
222. J.P. Richard, Time-delay systems: an overview of some recent advances and open problems. Automatica **39**(10), 1667–1694 (2003)
223. A. Sahai, Anytime information theory. Ph.D. dissertation, MIT, Cambridge, MA (2001)
224. M. Sahebsara, T. Chen, S.L. Shah, Optimal h infinity filtering in networked control systems with multiple packet dropouts. Syst. Control Lett. **57**(9), 696–702 (2008)
225. T. Samad, P. McLaughlin, J. Lu, System architecture for process automation: review and trends. J. Process Control **17**(3), 191–201 (2007)
226. J. Sandee, Event-driven control in theory and practice. Ph.D. thesis, Technische Universiteit Eindhoven (2006)
227. E. Santacana, G. Rackliffe, L. Tang, X. Feng, Getting smart. With a clearer vision of the intelligent grid, control emerges from chaos. IEEE Power Energy Mag. **8**(2), 41–48 (2010)
228. F.A. Schraub, H. Dehne, Electric generation system design: management, startup and operation of IEA distributed collector solar system in Almería. Spain. Sol. Energy **31**(4), 351–354 (1983)
229. P. Seiler, R. Sengupta, An H_∞ approach to networked control. IEEE Trans. Autom. Control **50**(3), 356–364 (2005)
230. J.H. Seo, H. Shim, J. Back, Consensus of high-order linear systems using dynamic output feedback compensator: low gain approach. Automatica **45**(11), 2659–2664 (2009)
231. G. Seyboth, Event-based control for multi-agent systems. Diploma thesis, Automatic Control Lab, Royal Institute of Technology (KTH), Sweden (2010)
232. G.S. Seyboth, D.V. Dimarogonas, K.H. Johansson, Event-based broadcasting for multi-agent average consensus. Automatica **49**(1), 245–252 (2013)
233. B. Shen, Z. Wang, Y.S. Hung, Distributed H_∞-consensus filtering in sensor networks with multiple missing measurements: the finite-horizon case. Automatica **46**(10), 1682–1688 (2010)

234. B. Shen, Z. Wang, Y.S. Hung, G. Chesi, Distributed h_∞ filtering for polinomial nonlinear stochastic systems in sensor networks. IEEE Trans. Ind. Electron. **58**(5), 1971–1979 (2011)
235. B. Shen, Z. Wang, L. Xiaohui, A stochastic sampled-data approach to distributed h_∞ filtering in sensor networks. IEEE Trans. Circuits Syst. I: Regul. Papers **58**(9), 2237–2246 (2011)
236. D.D. Siljak, *Large-Scale Dynamic Systems: Stability and Structure* (North-Holland, New York, 1978)
237. A. Stefanovska, Coupled oscillators: complex but not complicated cardiovascular and brain interactions. IEEE Eng. Med. Biol. Mag. **26**(6), 25–29 (2007)
238. A. Stoorvogel, *TheH_∞Control Problem: A State Space Approach* (Prentice Hall, Englewood Cliff, 1992)
239. P. Tabuada, Event-triggered real-time scheduling of stabilizing. IEEE Trans. Autom. Control **52**(9), 1680–1685 (2007)
240. U. Tiberi, C. Fischione, K.H. Johansson, M.D. Di Benedetto, Adaptive selftriggered control over IEEE 802.15.4 networks, in *49th IEEE Conference on Decision and Control*, Atlanta (2010), pp. 2099–2104
241. U. Tiberi, K.H. Johansson, A simple self-triggered sampler for perturbed nonlinear systems. Nonlinear Anal.: Hybrid Syst. **10**, 126–140 (2013)
242. Y. Tipsuwan, M.Y. Chow, Control methodologies in networked control systems. Control Eng. Pract. **11**(10), 1099–1111 (2003)
243. United Nations Framework Convention on Climate Change UNFCCC. Website (2014). http://unfccc.int/
244. P. Vaidyanathan, S.F. Midkiff, Performance evaluation of communication protocols for distributed processing. Comput. Commun. **13**(5), 275–282 (1990)
245. C.F. Van Loan, The sensitivity of the matrix exponential. SIAM J. Numer. Anal. **14**(6), 971–981 (1977)
246. F. Vanni, A.P. Aguiar, A.M. Pascoal, Netmarsys-networked marine systems simulator. Technical report WP6-0108, Instituto Superior Tecnico (Lisbon), May 2008
247. H. Vargas, An integral web-based environment for control engineering education. Ph.D. thesis, UNED (2010)
248. H. Vargas, J. Sánchez, S. Dormido, The Spanish university network of web-based laboratories for control engineering education: the Automatl@bs project, in *10th European Control Conference*, Budapest (2009), pp. 4623–4628
249. P. Varutti, R. Findeisen, Compensating network delays and information loss by predictive control methods, in *European Control Conference*, Budapest, 2009
250. R, Vaughan, A. Howard, The player project (2008). http://robots.mobilerobots.com/wiki/MobileSim
251. M. Velasco, P. Martí, J.M. Fuertes, The self triggered task model for real-time control systems, in *24th IEEE Real-Time Systems Symposium (RTSS03)*, Cancun (2003)
252. G.C. Walsh, O. Beldiman, L.G. Bushnell, Asymptotic behavior of nonlinear networked control systems. IEEE Trans. Autom. Control **46**(7), 1093–1097 (2001)
253. G.C. Walsh, H. Ye, L.G. Bushnell, Stability analysis of networked control systems. IEEE Trans. Control Syst. Technol. **10**(3), 438–446 (2002)
254. J. Wang, C. Liu, K. Li, H stochastic control of a class of networked control systems with time delays and packet dropouts. Math. Probl. Eng. (2014). Closed-loop; Disturbance attenuation levels; H-infinity; H-infinity controller; Markovian jump linear systems; Networked Control Systems (NCSs); Packet dropouts; Stochastic control
255. X. Wang, M.D. Lemmon, Event-triggered broadcasting across distributed networked control-systems, in *American Control Conference*, Seattle, June 2008, pp. 3139–3144
256. X. Wang, M.D. Lemmon, Event-triggering in distributed networked systems with data dropouts and delays. Lect. Notes Comput. Sci. **5469**, 366–380 (2009)
257. X. Wang, M.D. Lemmon, Finite-gain l_2 stability in distributed event-triggered networked control systems with data dropouts, in *European Control Conference* (2009)
258. X. Wang, M.D. Lemmon, Self-triggering under state-independent disturbances. IEEE Trans. Autom. Control **55**(6), 1494–1500 (2010)

259. X. Wang, M.D. Lemmon, Event-triggering in distributed networked control systems. IEEE Trans. Autom. Control **56**(3), 586–601 (2011)
260. Z. Wang, L. Liu, M. Zhou, Protocols and applications of ad-hoc robot wireless communication networks: an overview. Int. J. Intell. Control Syst. **10**(4), 296–303 (2005)
261. H. Wenshan, G.P. Liu, D. Rees, Networked predictive control system with data compression, in *IEEE International Conference on Networking, Sensing and Control, ICNSC'07* (2007), pp. 52–57
262. T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, RPL: IPv6 routing protocol for low power and lossy networks. IETF Internet Draft: http://tools.ietf.org/html/rfc6550, March 2012
263. C. Canudas De Wit. Best-effort highway traffic congestion control via variable speed limits, in *50th IEEE Conference on Decision and Control and European Control Conference*, Orlando (2011)
264. B. Wittenmark, J. Nilsson, M. Torngren, Timing problems in real-time control systems, in *American Control Conference* (1995), pp. 2000–2004
265. L. Wu, J. Lam, X. Yao, J. Xiong, Robust guaranteed cost control of discrete-time networked control systems. Optim. Control Appl. Methods **32**(1), 95–112 (2011)
266. T. Wu, Q. Cheng, Distributed estimation over fading channels using one-bit quantization. IEEE Trans. Wirel. Commun. **8**(12), 5779–5784 (2009)
267. J.Z. Xu, J. Zhao, J. Qian, Y. Zhu, Nonlinear MPC using an identified LPV model. Ind. Eng. Chem. Res. **48**(6), 3043–3051 (2009)
268. S. Xu, J. Lam, Improve delay-dependent stability criteria for time-delay systems. IEEE Trans. Autom. Control **50**(3), 384–387 (2005)
269. C. Yang, S.-A. Zhu, W.-Z. Kong, L. Li-Ming, Application of generalized predictive control in networked control system. J. Zhejiang Univ.: Sci. **7**(2), 225–233 (2006)
270. T.C. Yang, Networked control system: a brief survey. IEEE Conf. Control Theory Appl. **153**(4), 403–412 (2006)
271. T.C. Yang, Computational delay in digital and adaptive controllers, in *IEE Control 90 Conference*, Brighton, UK (1990)
272. X. Yin, D. Yue, S. Hu, Distributed event-triggered control of discrete-time heterogeneous multi-agent systems. J. Frankl. Inst. **350**(3), 651–669 (2013). Communication schemes; Consensus problems; Event-triggered controls; Heterogeneous multi-agent systems; Information communication; Lyapunov functional method; Multiagent networks; Sufficient conditions
273. H.K. Yong, H.K. Wook, H.S. Park, Stability and a scheduling method for network-based control systems. IECON Proc. (Industrial Electronics Conference) **2**, 934–939 (1996)
274. M. Yu, L. Wang, T. Chu, G. Xie, An LMI approach to networked control systems with data packet dropout and transmission delays, in *43rd Conference on Decision and Control* (2004), pp. 3545–3550
275. D. Yue, Q.L. Han, J. Lam, Network-based robust H_∞ control of systems with uncertainty. Automatica **41**(6), 999–1007 (2005)
276. D. Yue, J. Lam, Suboptimal robust mixed H_2/H_∞ controller design for uncertain descriptor systems with distributed delays. Comput. Math. Appl. **47**(6–7), 1041–1055 (2004)
277. D. Yue, E. Tian, Y. Zhang, C. Peng, Delay-distribution-dependent robust stability of uncertain systems with time-varying delay. Int. J. Robust Nonlinear Control **19**(4), 377–393 (2009)
278. D. Yue, E. Tian, Q.-L. Han, A delay system method for designing event-triggered controllers of networked control systems. IEEE Trans. Autom. Control **58**(2), 475–481 (2013). Co-designing; Controller designs; Delay systems; Event-triggered; Event-triggering; Feedback gain; Lyapunov functionals; Network transmission; Network-induced delays
279. G. Zames, Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverses. IEEE Trans. Autom. Control **26**(2), 301–320 (1981)
280. H.S. Zhang, L.H. Xie, W. Wang, X. Lu, An innovation approach to H_∞ fixed-lag smoothing for continuous time-varying systems. IEEE Trans. Autom. Control **49**, 2240–2244 (2004)
281. L. Zhang, T. Cui, X. Zhang, Distributed estimation for sensor networks with channel estimation errors. Tsinghua Sci. Technol. **16**(3), 300–307 (2011)

282. W. Zhang, M.S. Branicky, S.M. Phillips, Stability of networked control systems. IEEE Control Syst. Mag. **21**(1), 84–99 (2001)
283. Y. Zhang, X. Zheng, G. Lu, Stochastic stability of networked control systems with network-induced delay and data dropout. J. Control Theory Appl. **6**(4), 405–409 (2008)
284. Y. Zhao, G. Liu, D. Rees, Design of a packed-based control framework for networked control systems. IEEE Trans. Control Syst. Technol. **17**(5), 859–865 (2009)

Index

M. Guinaldo Losada et al. (eds.), *Asynchronous Control for Networked Systems*,
DOI 10.1007/978-3-319-21299-9