Ph.D. Dissertation

Contributions to Networked and Event-Triggered Control of Linear Systems

(Contribuciones al Control en Red Basado en Eventos para Sistemas Lineales)

María Guinaldo Losada

Physicist and Computer Scientist
from Universidad de Salamanca

Departamento de Informática y Automática

E.T.S. de Ingeniería Informática

Universidad Nacional de Educación a Distancia

Madrid, 2013

Department	Informática y Automática
Faculty	E.T.S. de Ingeniería Informática
Dissertation Title	Contributions to Networked and Event-Triggered Control of Linear Systems
Author	María Guinaldo Losada
Academic Degree	Physicist and Computer Scientist Universidad de Salamanca
Advisors	Dr. Sebastián Dormido Bencomo Dr. José Sánchez Moreno

Abstract

The development of new network technologies in the last decades has made possible its application to a noticeable heterogeneity of control systems. This application has given birth to a new family of control architectures, known as Networked Control Systems (NCS). In NCS, sensors, actuators, and controllers are connected through a shared band-limited digital communication network. In this regard, event-based control has been examined to reduce the communication between the components of a networked control loop. Moreover, NCS also suffer from other communication imperfections such as delays and packet losses. The goal of this thesis is to design and implement novel strategies to help solve some of the problems that arise in NCS. The framework is restricted to linear systems.

First, a single loop architecture is investigated and the proposed solution consists of designing new elements in the control loop which act as interfaces between the conventional components (controllers, sensor, and actuators) and the network. The use of event-triggered control helps to decrease the frequency of communication, and an anticipative controller, based on a model, computes finite-length signal predictions to better cope with network delays and packet losses. The proposed method has been implemented and two applications have been developed to facilitate the inclusion of conventional controllers in NCS.

Secondly, several distributed event-based control strategies are provided to reduce communication and actuation in networked interconnected systems. The proposed design guarantees the system stability, certain level of performance, and the existence of a positive lower bound for the inter-event time. The possible model uncertainties that characterize large-scale interconnected systems, and the effect of network imperfections such as delays or packet losses are analyzed. Two communication protocols are investigated to deal with this latter problem.

Finally, tools to implement distributed event-based control in networked mobile robots are provided. An interactive simulator has been developed to offer flexibility when defining the conditions of the experiment. The distributed event-based control algorithms are also applied in a testbed of mobile robots.

To my family

Acknowledgments

My most sincere gratitude to all the people that made this thesis possible.

First of all, to my advisors José Sánchez Moreno and Sebastián Dormido Bencomo, for giving me the opportunity to do this work, for their support, encouragement, trust and inspiration during all these years, which have helped me to grow personally and professionally. Thank you so much.

To professors Denis Gillet, Karl H. Johansson, and Carlos Canudas de Wit, for inviting me to visit their home institutions, for broadening my research interest, and for the shared work. I would also like to thank Dr. Christophe Salzmann for helping me with the first experimental results of my Ph.D., Dr. Dimos V. Dimarogonas, who supervised my work at KTH, for all the meetings and useful suggestions, and Dr. Daniel Lehmann for his advice on event-based control.

To all the professors at the Departamento de Informática y Automática. I would like to extend special thanks to Raquel Dormido and Natividad Duro for facilitating so much my start as a docent, and to Sebastián Dormido-Canto for all his advice and help. I would also like to thank Pilar Riego for all the help given with administrative issues.

To my colleagues at the department Miguel Ángel, Víctor, David, and Luis, and also to the past members. Special thanks to Dictino for his useful comments regarding stability, to Ernesto for helping me with the experimental results, and to Jesús who is always willing to help. I would also like to express my gratitude

to Héctor Vargas for his friendship and his support at the beginning of my Ph.D. I had the pleasure of working with Gonzalo Farias as well. Thank you for the help with Ejs.

To Eladio Sanz and Belén Pérez-Lancho, who introduced me into the automatic control. Thank you Eladio for affording me the opportunity of meeting Sebastián, and for all your help and support.

During my Ph.D. I have met a lot of people from academia. I would like to thank the people from the Automatic Control Laboratory (KTH) and to the NeCS team (INRIA), where I had the opportunity to work. My special thanks go to António, Pablo, Giovanni, Ruggero and Luis for making it easier to be away from home, and for their friendship.

To my family, especially to my mother Filo and my father Víctor for giving me the best they can and for the values they taught me such as responsibility and perseverance. To my brother Víctor for being always there. To María Panero for her positiveness and enthusiasm.

To my friends who are the best support one can have. My special gratitude to Raquel and Ana for being my family in Madrid during these years. Thank you to David, Noelia, Chisi, Pilar and Bea for their encouragement, to *los Molineros* for all the unforgettable moments, and to the friends I made in Colegio César Carlos.

Finally, there is one person remaining who has been essential for me in this time, Luis. Thank you so much for all your support, patience and love.

María Guinaldo Losada

Madrid, May 2013

Contents

List of Abbreviations

AAC	Anytime Attention Control
ACK	Ackwnowledgment
AODV	Ad-hoc On Demand distance Vector
API	Application Programming Interface
CAL	Controller Adaptive Layer
CBE	Control Basado en Eventos
CCD	Charge-Coupled Device
DAQ	Data Acquisition
DUN	Dial-up Networking
Ejs	Easy Java Simulation
EPFL	École Polytechnique Fédérale de Lausanne
GUI	Graphical User Interface
DEBC	Distributed Event-Based Control
DSR	Dynamic Source Routing
GUUB	Globally Uniformly Ultimately Bounded
IDTS	Input Difference Transmission Scheme
I/O	Input/Output
IAE	Integral Absolute Error
IFAC	International Federation of Automatic Control
LEACH	Low Energy Adaptive Clustering Hierarchy
LMI	Linear Matrix Inequality
LAN	Local-Area Network
LTI	Linear Time Invariant
MB	Model Based
MIMO	Multiple Input Multiple Output
MAC	Minimum Attention Control
MaSS	Multi-agent Systems Simulator

MPC	Model Predictive Control
NCS	Networked Control Systems
PAL	Process Adaptive Layer
PAN	Personal Area Network
PID	Proportional Integral Derivative
PERM	Permission
PPP	Point-to-Point Protocol
RF	Radio Frequency
RPL	Routing Protocol for Low power and lossy networks
RTT	Round Trip Time
TCP	Transmission Control Protocol
TS	Time Stamp
SCR	Sistemas de Control en Red
SISO	Single Input Single Output
UDP	User Datagram Protocol
UNED	Universidad de Educación a Distancia
UwR	Update when Receive
VI	Virtual Instruments
VRML	Virtual eality Markup Language
WfA	Wait for All
WNCS	Wireless Networked Control System
WSAN	Wireless Sensor/Actuator Network
WSN	Wireless Sensors Network
ZOH	Zero Order Hold

List of Symbols

Indices

$(\cdot)^*$	Conjugate transpose of a matrix
$(\cdot)^{-1}$	Inverse of a matrix
$(\cdot)^T$	Transpose of a vector or matrix
$\overrightarrow{(\cdot)}$	Collection of elements
$\hat{(\cdot)}$	Estimated signal or matrix
$(\cdot)_0$	Initial value
$(\cdot)_b$	Broadcasted signal
$(\cdot)_C$	Refering to the controller (Signal or matrix)
$(\cdot)_d$	Discrete-time (matrix)
$(\cdot)_u$	Refering to the control input u
$(\cdot)_x$	Refering to the state x
$(\cdot)_\Delta$	Refering to the coupling Δ
$(\cdot)_i/(\cdot)^i$	Refering to a subsystem i
$(\cdot)_{i\to j}/(\cdot)^{i\to j}$	Refering to transmission from a subsystem i to j
$(\cdot)_{max}$	Maximum value
$(\cdot)_{min}$	Minimum value

Scalars

n	Dimension of the state vector
m	Dimension of the input vector
r	Dimension of the output vector

k	Counter
ℓ	Counter
t	Continuous time
τ	Delay
T_s	Sampling time
T_W	Waiting time
c	Event constant threshold
t_k	Event time
k_i	Event time (discrete time)
$\tau^\star$	Delay bound
$P^\star$	Maximum number of consecutive packet losses
$\lambda(A)$	Eigenvalues of A
$\kappa(A)$	Condition number of A ($\kappa(A) = \|A\|\|A^{-1}\|$)

Vectors

x	State vector
u	Input vector
w	Disturbance vector
y	Output vector
v	Noise vector
e	State error
e_y	Output error
y_{sp}	Set-point
ϵ	Set-point tracking error
ξ	Augmented state vector
$\mathbf{r}$	Desired inter-vehicle relative position vector
$\mathbf{1}$	Column vector whose components are ones

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 ($A_K = A + BK$)
$\mathbf{U}_k$	Control sequence
$\hat{\mathbf{X}}_k$	State prediction sequence
$\hat{\mathbf{Y}}_k$	Output prediction sequence
A_{CL}	Closed loop matrix for output measurement
H_{ij}	Interaction term between i and j
L_{ij}	Decoupling gain
Δ_{ij}	Coupling term between i and j
$\mathbf{0}$	Null matrix of appropriate size

Functions

$f(\cdot)$	Trigger function
$\mathcal{O}$	Order
$\mathcal{S}$	Size
L_f	Fréchet derivative of a matrix function f

Sets

$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of Real Numbers
$\mathbb{C}$	Set of complex numbers
N_i	Set of neighbors of agent i
$\mathcal{V}$	Set of vertices of a graph $\mathcal{G}$
$\mathcal{E}$	Set of edges of a graph $\mathcal{G}$

List of Tables

List of Figures

1 Introduction

Summary

Control loops that are closed over a communication network have become more and more common as the hardware devices for networks and network nodes have become cheaper. The advantages of using digital communication networks are manifold, and not only from the point of view of the applications. However, networks also pose some challenges such as bandwidth limitations, delays or packet losses.

This chapter presents an overview of networked control systems, including aspects related to the network and the control. In particular, the influence of the type of network, the system architecture, and some of the most relevant approaches to deal with the communication imperfections that characterize networks are discussed. The focus also lies on event-based control, which has been shown to be effective in control over networks. In addition, the contributions and the outline of this thesis are given, where insights about the main goals of the dissertation and a brief description of each chapter are provided.

Finally, the results obtained in the development of the thesis (conference and journal papers) and the research projects that supported this work are mentioned.

1.1 Control over networks

1.1.1 General issues

The development of network technology in the last decades has made possible its application to control systems. Now, networked control is a strong research area in control systems. Indeed, at least two of the technical areas of the International Federation of Automatic Control (IFAC) are devoted to this field, and there also exists an increasing number of specialized conferences and workshops.

In Networked Control Systems (NCS), sensors, actuators, and controllers are connected through a shared band-limited digital communication network. The use of a multipurpose shared network to connect spatially distributed elements results in manifold advantages that are the reason of the success of NCS:

- Network structured systems offer flexible architectures, making easier the reconfiguration of the system parts and allowing a simpler addition of new elements to it.
- They generally reduce installation and maintenance costs, due to the reduction in the wiring required in a point-to-point paradigm.
- As a consequence of the previous statement, diagnosis and fault detection are easier tasks.

NCS have also opened up a complete new range of real-world applications, such as mobile sensor networks [HE04, OFL04], distributed power systems and smart grids [MAW05, BZ11] (see Figure 1.1), intelligent transportation systems [MS06], formation control of autonomous vehicles [SS05, GKKP06], surveillance [BCM$^+$10, CM02] (see Figure 1.2), remote surgery [MWC$^+$04] (see Figure 1.3), and many more.

However the use of a shared network introduces new challenges and makes the analysis and design of NCS complex. Conventional control theories with many ideal assumptions, such as synchronized control, non-delayed sensing and actuation and unlimited bandwidth, must be re-evaluated before they are applied to

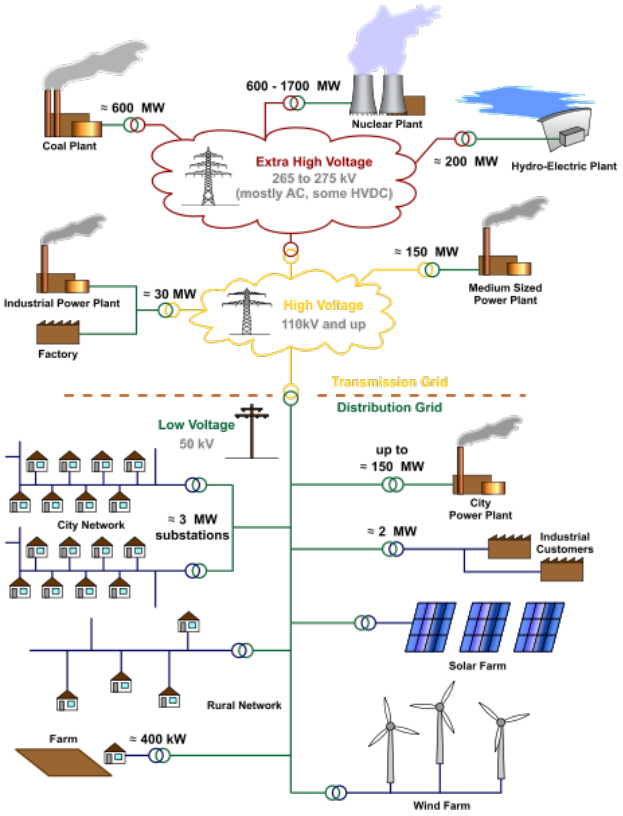

Figure 1.1: General layout of electrical grids. By MBizon [CC-BY-3.0 [com13]], via Wikimedia Commons.

NCS. Improving communication networks and protocols to increase the reliability is a partial solution. Thus, new control algorithms must be developed in order to deal with communication imperfections and constraints [Hee10], which can be summarized as follows:

- *Limited bandwidth:* Any communication network can only carry a finite amount of information per unit of time, and this can have a severe effect on the control system. In most digital networks, data are transmitted in atomic units called packets. Packets can be divided into the payload (user data) and the control information (headers) required for the transmission. The maximum size of the payload depends on the protocol and goes from 1500 bytes in Ethernet to 8 bytes in some Radio Frequency protocols [mOw10].

- *Variable communication delays:* The transmission of one packet from one node in the network to another node is not instantaneous and can take a variable amount of time which depends on highly variable network condi-

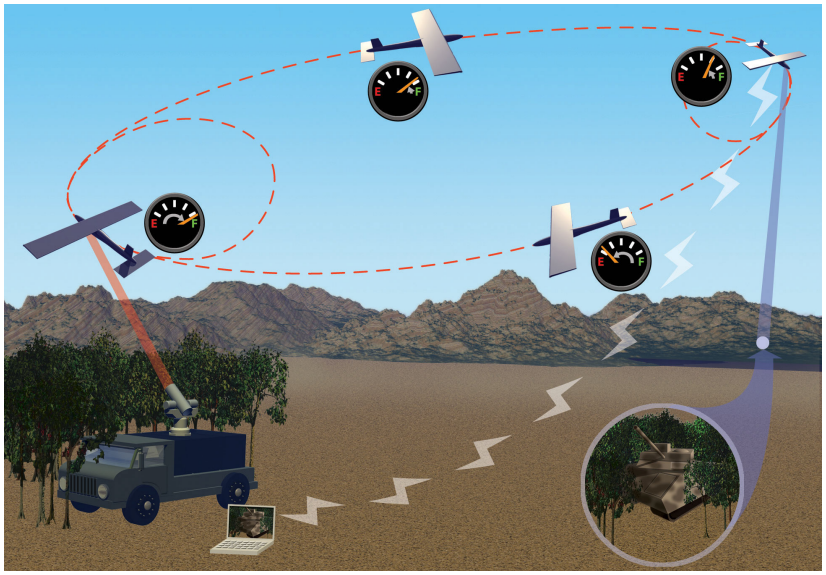

Figure 1.2: Unmanned aerial vehicles providing surveillance and communications, borrowed from [Las10].

tions such as congestion, channel quality or the protocol. This can affect the control performance in several ways. First, the transmitted information is delayed. Time-delayed systems are by themselves a topic of research. They have been vastly investigated out of the NCS context [NRC07], showing that the stability analysis is more involved than delay-free systems. Secondly, delays can induce variable sampling intervals. A significant number of results have attempted to characterize a maximum upper bound on the sampling interval for which stability of the system can be guaranteed (see [HNY07] and references there in).

- *Packet dropouts:* A packet can be lost due to transmission errors in physical network links, channel congestion or corrupted packets rejected in-transit. Also, in control applications packets can be discarded and treated as lost if they arrive "out of date", for instance, if an updated packet had already been received.

- *Quantization:* A quantizer is a function that maps a real-valued function into a piecewise constant function taking on a finite set of values. In NCS, the finite word length of the packets induces errors in the transmitted signals over the network.

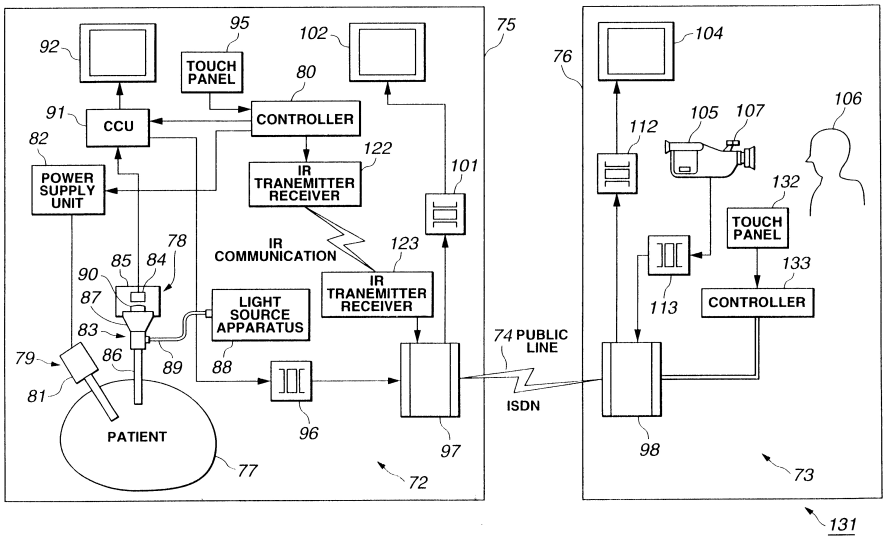

Figure 1.3: Remote surgery support system, borrowed from [Uch03].

Another key issue that distinguishes NCS from conventional control systems is the architecture. The general structure of NCS is depicted in Figure 1.4. It consists of several (sub-)systems, which may be physically interconnected. The respective sensor (S), actuator (A), and controller (C) nodes may be widespread within the entire system and connected arbitrarily through the network. Section 1.2 covers this aspect in more detail.

1.1.2 Work in the area of control over network

Most of the work in the area of NCS is inclined to model them into conventional control systems with some of the communication constraints described above. This usually involves a complete, and often complex, re-design of conventional controllers, which, moreover, can result to be conservative [ZLR09].

In [Hee10], an overview of the main lines of research in stabilizing controller synthesis of NCS is given. All the methods assume hard bounds on the communication imperfections such as delays, transmission intervals and maximum number of consecutive packet dropouts. Another usual restriction in these designs is that the synthesis conditions are LMI-based, which restricts the problem to linear plants. These approaches can be summarized as follows:

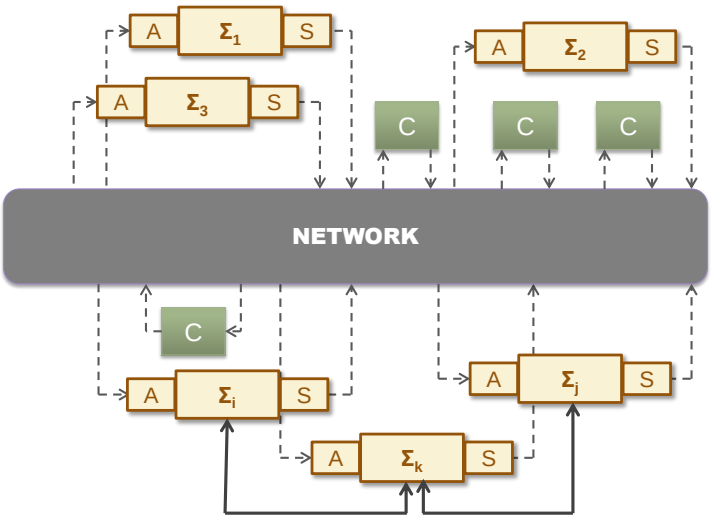

Figure 1.4: Generic NCS architecture.

- The discrete-time approach, in which NCS are modeled in discrete-time. The delay and the sampling interval represent the uncertainty of the system based on which LMI-based stability conditions are derived. See for instance [DHvdWH11].

- The sampled-data approach uses (impulsive) delay-differential equations. Stability conditions are LMIs resulting of extensions of the classical Lyapunov-Krasovskii functional for linear plants and controllers. See [NHT08].

- In the emulation approach [HTVdWN10], the stability of NCS is assessed by combining continuous-time Lyapunov functions of the network-free closed-loop system and the network protocol.

There are also some contributions in the field of robust control. H_∞ and H_2/H_∞ controllers have been proposed to deal with the presence of unreliable channels in the control loops, such as [YHL05, SS05, MOB+12].

The following two sections will analyze the particularities of two types of networks: Internet-like networks and wireless networks, and how they can differently affect the control system.

Internet-like networks

Internet-based control systems allow remote monitoring and adjustment of plants over the Internet, which makes the control systems benefit from the ways of retrieving data and reacting to plant fluctuations from anywhere around the world at any time.

Internet-like communication networks are based on packets that can carry a larger amount of data than is required for a control system without consuming additional network resources. Thus, rather than sending individual values, finite-length signal predictions can be transmitted. This is the motivation of the so called *packet-based control*.

A common approach is to use model-based control to emulate future states of the plant and, therefore predictions for the control signal. The idea of combining packet-based control and Model Predictive Control (MPC) was first introduced in [Bem98] in the context of teleoperation. Since then, other authors have exploited the principle of MPC in packet-based NCS [KJA06, QSG07, MJVR08, VF09, ICMS11].

Other alternatives have been studied to reduce the computational effort required by MPC. One of them is the so called *anticipative control* which estimates the future state of the system based on a model that considers delays [NH06], but with no optimization. The resultant control sequence may not be optimal, but its calculation consumes an insignificant slot of time compared as to MPC.

Anticipative controllers have been proposed in [ESDCM07, GSD11] for different network architectures.

Wireless networks

Recently, some work devoted to enabling control applications over wireless networks has begun to appear. The motivation of the interest in wireless networks comes from the cheap deployment and the increase of flexibility in getting rid of cabling. Early works focus the controller design on wireless sensor/actuator networks (WSANs) under ideal network assumption. However, the communica-

tion imperfections cannot be neglected from the implementation point of view. Moreover, these imperfections are much severe than in wired networks.

The communications community is also directing efforts into enabling reliable wireless networks for control applications, so that low-latency and hard-real-time constraints could be met [Maz10]. Also, WSANs arise new challenges with respect to cabled NCS. The most important one is the energy efficiency in devices powered by batteries, which impose computation and communication constraints on the WSAN design. Several factors determine the energy consumption. One of them is the data rate transmission, which also has influences on the network delay and packet dropouts. In general, reducing the number of transmitted packets per second yields a larger battery life for the wireless device. The packet size also has an impact on energy consumption, which increases with the packet size. However, the energy consumption per transmitted byte decreases with packet size due to the cost of protocols overheads [JER+10]. Most of the implementations use small and fixed size packets, since the required data for control applications are few.

In the proposed solutions in the literature, communication protocols are designed mainly to achieve high reliability and high energy efficiency for various applications of Wireless Sensor Networks (WSNs) and not specifically for control applications [AKK04, BDWL10]. More recently, a joint design of control and communication parameters has been proposed in [AAJ+11] to optimize energy consumption while guaranteeing a desired control performance.

However, the majority in the literature considers periodic transmission, sampling, and actuation. Recently, aperiodic sampling techniques have been proposed to more efficiently address the issues of wireless networks. A review of these techniques is presented in Section 1.3.

1.2 System architecture

The general architecture of NCS has been shown in Figure 1.4, in which the nodes are widespread across the network. Particular schemes that are considered in this thesis are described more in detail next.

1.2.1 Single loop schemes

In a single loop scheme, there are a controller node (C) and a plant with the corresponding actuator (A) and sensor (S). How these elements are located gives the different NCS configuration depicted in Figure 1.5. Note that the information is transmitted at discrete instances of time through the network. It is assumed that the element that transmits data to the network has the ability of converting data from continuous to discrete time, and the element that receives information from the network is able to transform a discrete signal into a continuous signal. We further consider full state $x(t)$ and output $y(t)$ measurement scenarios. The dashed lines represent the flow of information at discrete instances of time, generically denoted as t_k, such discrete instances of time being either equidistant in time or not.

Figure 1.5a shows the case of a collocated controller with the sensor, and the control inputs are transmitted through the network. Examples of works in which this is the preferred scheme are [ESDCM07, QSG08]. In the architecture of Figure 1.5b the controller is collocated with the actuator and only the plant measurements are sent through the network. Hence, the control signal is directly applied to the actuator. This is the considered scheme in, for example, [GCHM06, LL10, LL11b].

A more general scheme is depicted in Figure 1.5c, in which the network is at both sides of the controller. When a measurement is received by the controller, a new control input is computed and transmitted back to the plant. Assume that the information transmitted through the network is delayed. While in the collocated controllers of figures 1.5a and 1.5b the delay can be measured because the loop is closed through the network only between two elements of the loop, in a remote controller scheme the delays from the sensor to the controller and from the controller to the actuator cannot be known independently and only the sum of both delays can be computed.

In this thesis, the studied architecture for the single-loop case corresponds to the remote controller (Figure 1.5c), which is the focus of the study of the chapters

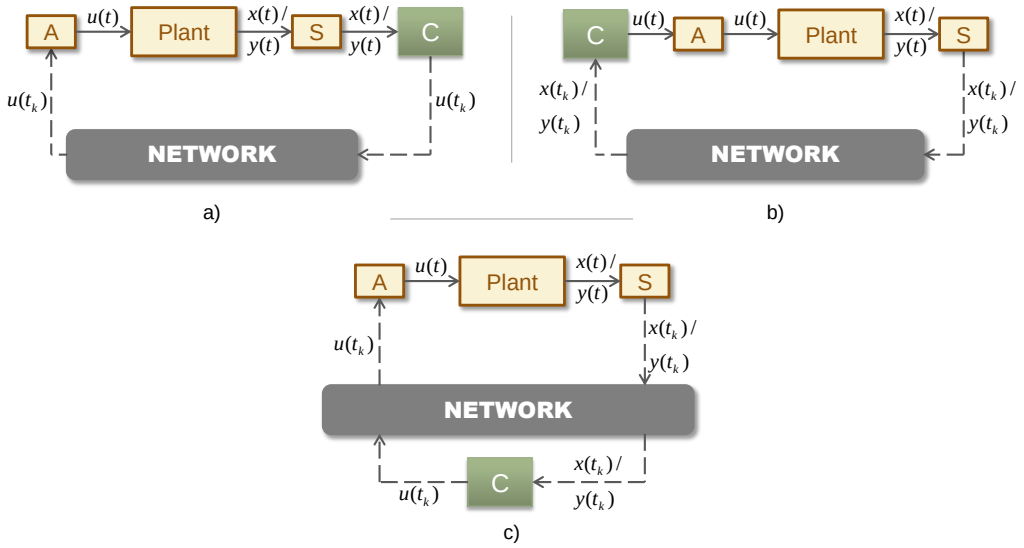

Figure 1.5: Single loop schemes in NCS.

2 and 3.

1.2.2 Multi-loop schemes

For multi-loop schemes, the possibilities of how to locate the different elements of the control loops are manifold. We focus on two architectures, which are the most common in the literature and from the implementation point of view.

The first architecture has a single controller which receives measurements from different sensors and send control updates to one or more actuators (*centralized control*). In general, these elements may be attached to the same system or to different plants.

Secondly, a *distributed control* approach is considered. Specifically, it is assumed that each node in the network has its own controller and needs to communicate with other nodes in the network for control purposes.

Centralized control

The centralized control approach for a multi-loop architecture is shown in Figure 1.6. The sensor nodes transmit the measurements to the controller through the network. Whereas in Figure 1.6 we consider that each plant has a sensor and an actuator, it might be the case of having a single plant with several sensors

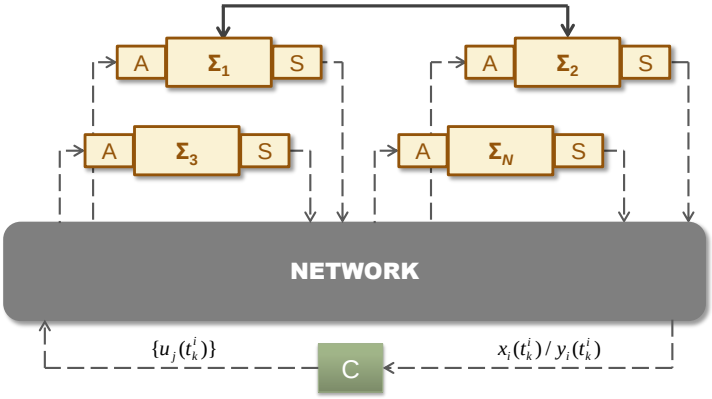

Figure 1.6: Centralized control of multiple plants.

responsible for monitoring the state of the plant and several actuators. The interconnections between the subsystems are represented by the bold solid line.

Depending on the complexity of the problem, the controller may have to switch between different sub-controllers if, for instance, there is a sub-controller for each plant, or, a single multi-variable controller can be designed in the case of a single plant with several sensors and actuators.

Regardless the nature of the problem, the controller has to process the different measurements received from the sensor nodes, denoted as $x_i(t_k^i)$ (full-state measurement) or $y_i(t_k^i)$ (output measurement), and compute the corresponding control inputs $u_j(t_k^i)$, where t_k^i denotes the sampling time instance. We consider that a measurement $x_i(t_k^i)/y_i(t_k^i)$ can involve the computation of more than one control input $u_j(t_k^i)$ since coupling between the sub-systems is accepted.

A centralized networked controller usually requires the use of buffers that induce additional delays to the network and computational delays. This *processing* delay, which is the elapsed time between the reception of the measurement $x_i(t_k^i)/y_i(t_k^i)$ and the start of the computation of $u_j(t_k^i)$, is not negligible when the number of control loops increases.

Though some strategies can be taken to mitigate this problem, for instance, the design of a network scheduler which decides when a node transmits [AAJ+11], a centralized controller is not convenient for large-scale systems.

For this reason, there has been ongoing interest in the design of decentralized control systems. We note, however, that this is not an easy task. Even basic

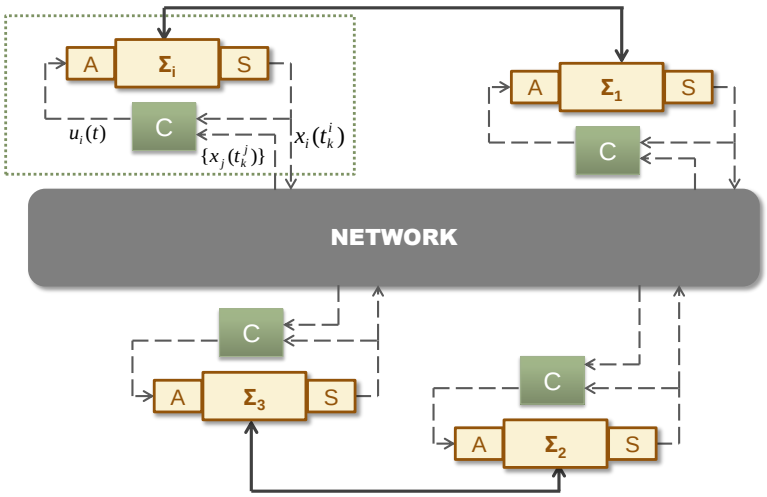

Figure 1.7: Distributed control of multiple plants.

notions such as stability become non trivial in a decentralized framework [WD73, GSS05]. In some implementations where the number of sensors and actuators is not large, a networked centralized controller can outperform the decentralized one if the reliability of the communication channel is guaranteed [SGQ08].

Distributed control

Many control systems are built in decentralized structure using a large number of simple single-input single-output (SISO) controllers enabling a stable operation of most unit operations. However, this approach is not the optimal control solution because in these structures the subsystems do not communicate between them even if they significantly interact. By contrast, the distributed control approach relies on the assumption that the information about the controllers from other loops is exploited in designing the controllers. Since controllers may require to communicate, the communication network turns to be part of the design problem.

Two types of interconnections are distinguished. In the first one, a subsystem can be physically interconnected to others, 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. This includes large-scale MIMO plants that can be split up into a set of physically coupled systems. The second type of interconnection is when the need of communication between the controllers

comes from the fact that the system tries to achieve a common objective. 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 control system is depicted in Figure 1.7. Each node i can be physically encapsulated (dotted line) and includes the subsystem and the local controller, which receives at discrete instances of time t_k^i the local state $x_i(t_k^i)$ and also the set of states $x_j(t_j^k)$ of other subsystems (also called agents) measured at different instances of time t_j^k. The agents that transmit information to i are known as its neighborhood and correspond to the ones that are interconnected with agent i.

1.3 Event-based control

1.3.1 Preliminaries

Though the physical world is analogical, most of the control applications are implemented in digital platforms that require the information in a control loop to be exchanged in a discrete-time manner between sensors, actuators, and controllers. Traditionally, instant times at which this exchange is carried out are equidistant between each other, i.e., are given by a sampling period. The frequency of sampling 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 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 had an increasing interest. In event-based control systems information is exchanged in the control loop when a certain condition in 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 of event-based control, event-triggered con-

trol, 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 the implementation.

Despite of its recent popularization, event-based sampling is not actually a new concept, and its origins date back to the late 50s when [Ell59] argued that the most appropriate sampling method is to transmit data when there is a significant change in the signal. Later on the 60s and 70s, an heuristic method called *adaptive sampling* [DFP62] 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, [Arz99] implemented the event-based control into a PID controller showing that the number of control updates was reduced without degrading the performance of the system. In [HGvZ+99] 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 [rB03], showing that under certain conditions the event-based control has a better performance than the periodic control. But the real impulse to the event-based control came out few years later when many researchers realized the benefits of applying this theory to networked control systems. Next section presents a literature review of event-based control applied to NCS.

1.3.2 Event-based control and NCS

Last decade has been prolific in the field of event-based control, and the lack of analytical results has been overcome. Also, experimental results have demonstrated a more efficient usage of the network bandwidth than periodic transmission.

In most implementations, an event is triggered when the error of the plant exceeds a tolerable bound. How this error and this bound are defined distinguish the different approaches in the literature.

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, the

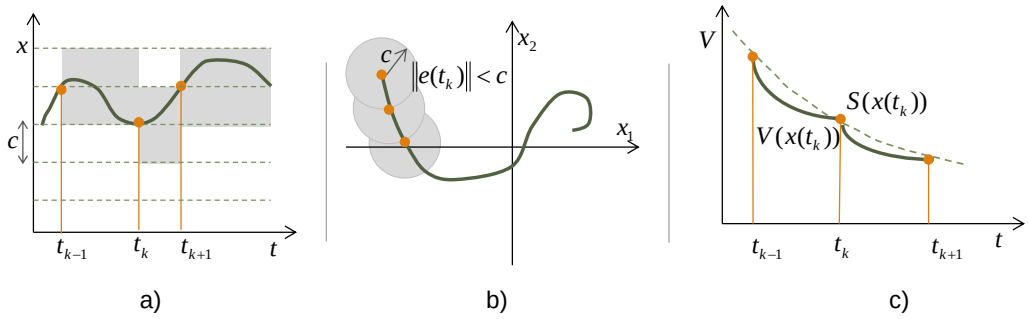

Figure 1.8: Examples of triggering rules.

trigger rule is

$$\|e(t)\| = \|x(t) - x(t_k)\| \leq c,$$

and the usual terminology to define it is deadband control. t_k refers to the instant of the last event and t is the current instant of time. The value of c determines the performance of the system and the ultimate set in which the state of the plant is confined around the equilibrium. Figures 1.8a and 1.8b depict two examples of deadband control for an unidimensional state and two dimensional state, respectively. Related works to deadband control are [HSvdB08, San06].

Deadband control does not generally yield asymptotic stability. And so, some researchers have investigated triggering rules to fulfill this. One example is presented in [Tab07] and the error is bounded by the state at the current time

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

This approach yields the system asymptotic stability but the inter-event times become shorter when the system reaches the equilibrium. However, in [Tab07] it is shown that a minimal inter-event time is guaranteed to exist only under suitable assumptions. This is an important issue in event-based control: the Zeno behavior, i.e. the occurrence of two consecutive events at the same time, has to be excluded. The parameter σ is designed according to some properties of the Lyapunov function.

Other authors have exploited the idea of using Lyapunov methods to define the triggering rule [MAT09]. An event is triggered when the value of the Lyapunov function of the closed loop system for the last broadcasted state reaches a

certain threshold of performance (see Figure 1.8c):

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

Recently, other time-dependent triggering rules have been proposed to reach the desired point asymptotically. In [GDJ+11, SDJ13], the trigger functions for single integrators and linear interconnected systems, respectively, bound the error as

$$\|e(t)\| \leq c_1 e^{-\alpha t},$$

which has the aforementioned property, guaranteeing a lower bound for the inter-execution times.

Sensor networks are 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 [AT10b, MT08, Ara11] that the most of the energy consumed in a sensor node comes from the task of monitoring the measured variable(s) rather than the transmission. The event-triggering rules 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 waken 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 [VMF03]. Self-triggered control can be regarded as a software-based emulation of event-triggered control. It has been studied for linear systems [WL09, MAT10], and applied to sensor and actuator networks in [MT08, TFJB10, AAJ+11, CMV+10]. 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 [WL10]. A so-called

hybrid sensor communication is proposed in [Ara11] as a trade-off between the event-based and self-triggered mechanism in Wireless Networked Control Systems (WNCS) for linear systems. Still, the disturbance rejection cannot be completely guaranteed and the control is centralized, which makes difficult the extension to large number of nodes.

Another approach is the Minimum Attention Control (MAC). It maximizes the time interval between executions of the control task, while guaranteeing a certain level of closed-loop performance [AT10a, DTH12]. 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. Although in [DTH12] an approach based on extended control Lyapunov functions allows to solve the problem online alleviating the computational burden as experienced in [AT10a], 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 [GQ10, GFB11, Gup09]. The Anytime Attention Control (AAC), proposed in [AT10a], 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.

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 [DH12].

The existing output-based event-triggered controllers can be categorized in observer-based or not. [LL11a, LL11c] belong to the first category. The measured state is replaced in the trigger function by the estimated state provided by the observer [LL11a] or the filter [LL11c]. The second direction is to use a different structure in the controller. A dynamical output-based controller is proposed in [DH10]. 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 [KB06], where a LTI continuous-time controller is emulated.

All the approaches described above consider Zero-order hold at the actuator, i.e., the control input computed at event times is hold constant till the next event occurrence. Although this consideration of "doing nothing" between events simplifies the analysis, it is 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 [LL10]. The idea of taking advantage of a model in NCS and working in open loop is not new and was introduced in [MA02, MA03a], but the updates from the system are periodic not event-triggered. However, emulation approaches such as [LL10] 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.

Some of the cited works and others consider a multi-loop architecture and a decentralized controller such as [MT11, MC11, Mol11, GA12, DH12] or a distributed control [WL08, WL11, GDJ$^+$11, SDJ13].

Finally, few existing works explicitly consider the effect of other communication constraints such as delays, packet dropouts or quantization in event-based control. One of the most relevant results is presented in [LL12], which is an extension of the previous paper [LL10]. Also, in [GA11a] an implementation to compensate delays is presented, and [WL11, GLS$^+$12] discuss distributed implementations of event-triggering in imperfect networks.

1.4 Model-based control in NCS

As said before, most of the approaches in the literature consider a Zero Order Hold (ZOH) which holds the last received value so that the output is a piecewise constant signal.

Consider the scheme of a collocated controller with the actuator as in Figure 1.5a. The last measurement is held constant and the controller computes the

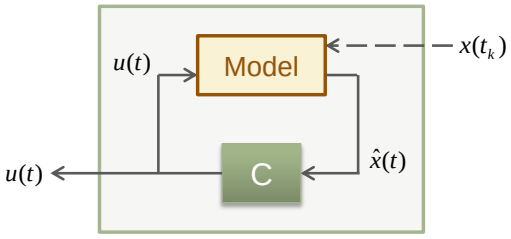

Figure 1.9: Model-based controller.

control signal $u(t)$ which is piecewise constant. Now assume that a model of the plant is available and, instead of holding the last received value, an estimate of the state of the plant can be used between two consecutive receptions. This is the idea of the model-triggered control. The controller is replaced by the element depicted in Figure 1.9. The controller C computes the control output based on the state predicted by the model $\hat{x}(t)$, which is initialized when a new measurement $x(t_k)$ is received.

As mentioned in the previous section, the concept of model-based NCS (MB-NCS) was first introduced by [MA02, MA03a]. Since then, Antsaklis and co-workers have published different extensions such as discrete-time models [EA09] or time-varying transmission times [MA04].

Other control approaches, that use a model in the design where usually a discrete-time model is iterated to generate future control inputs, have been commented in Section 1.1.2.

An event-based framework in which a model is used in both event detector and the controller to emulate the continuous state feedback controller is presented in [LL10, Leh11], as already mentioned.

Finally, a MB-NCS framework with event-triggered control is also presented in [GA11a] in which time-varying delays and model uncertainties are considered. Also, decentralized [GA12] and distributed [GLS+12] MB implementations have been proposed.

In general, model-based control allows increasing the sampling period (periodic control) or enlarging the inter-event times (event-based control). Ideally, if there are not model uncertainties and the system is not affected by any disturbance, the model perfectly estimates the state of the plant by knowing the initial

state. Hence, the purpose of combining model-based and event-triggered control is to reduce the number of events with respect to a ZOH approach.

1.5 Objectives of the Thesis

The main objective of this thesis is to contribute to solve some of the problems that arise in Networked Control Systems, with a special focus on event-based control. The contributions are twofold: the design of such strategies and their implementation. This main objective can be split into:

- *The design of a new architecture for networked control systems*, by including new elements in the control loop which act as interfaces between the conventional components (controllers, sensor, and actuators) and the network. The purpose of the new architecture is to take advantage of the flexibility offered by the use of the network while maintaining the stability of the system and dealing with the communication imperfections. More specifically, the objective is to decrease the frequency of communication via event-based control and to cope with network delays and packet losses sending finite-length signal predictions.

- *The implementation of prototypes* of the proposed solutions so that the conventional controllers can be reused without spending much time in their conversion to a networked control system.

- *The design of distributed event-based control approaches* for networked systems. The objective is to design transmission and actuation policies that decrease the amount of communication while guaranteeing a certain level of performance and the exclusion of Zeno behavior. The proposed design will deal with the possible model uncertainties that characterize large-scale interconnected systems. Also, communication protocols will be investigated to deal with the problem of network delays and data dropouts, while the existence of a positive lower bound for the inter-event time has to be preserved, since this is one of the major problems of the existing approaches

in the literature.

- *To provide and implement tools to apply distributed event-based control.* The motivation behind this objective is twofold: the application to the education environment in which we are immersed as well as a means of testing the control algorithms under a wide range of scenarios before the implementation in real platforms. Also, the validation over a testbed will be given.

1.6 Outlines and Contributions

This thesis has been structured as follows:

- **Chapter 2. Anticipative Control Design in Internet-like Networks.** Chapter 2 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. The design of two middleware layers between the process and the network, and between the controller and the network is proposed as a means of hiding the elements which do not belong to a conventional control loop. Event-based transmission rules are proposed to save network bandwidth. The stability of the system is proved to be Globally Ultimately Uniformly Boundeded (GUUB) when some constraints are imposed on the network delay. Different extensions such as disturbance estimators, output measurement and LTI anticipative controllers are discussed, preserving the Globally Ultimately Uniformly Boundedness property of the system. Finally, a centralized remote controller for a multi-loop architecture is presented. This work was published in part in the IET Control Theory and Applications (see [GSD11]), presented in the 49th IEEE Conference on Decision and Control (see [GSD10]), and included in the Proceedings of the *XXXIII Jornadas de Automática* (see [GSDD12]).

- **Chapter 3. Implementation and Experimental Evaluation of the Anticipative Control.** The experimental framework in which the anticipative controller presented in Chapter 2 has been evaluated is reported in Chapter 3. The description of the plants, the implementations of the middleware layers in LabVIEW and the experimental results enhancing the goodness of the proposed design are also given. The related publications to this chapter are the same as for Chapter 2.

- **Chapter 4. Distributed Event-Based Control for Interconnected Linear Systems.** Chapter 4 presents a distributed event-based control strategy for a networked dynamical system consisting of N linear time-invariant interconnected subsystems. The proposed triggering rules, which depend on local information only, can guarantee the asymptotic convergence to the equilibrium and the existence of a lower bound for the broadcasting period. The problem is initially solved for perfect decoupled systems, and then the results are extended for non-perfect decoupling, since that constraint is difficult to meet in practice. The coupling terms are treated as a perturbation of the nominal system, and the existing classical analysis on the sensitivity of the matrix exponential and matrix powers is applied to infer constraints on the coupling terms so the asymptotic stability property is preserved. This work was presented in the 50th IEEE Conference on Decision and Control [GDJ+11] and an extended version has been accepted for publication [GDJ+13].

- **Chapter 5. Extensions and Improvements of the Distributed Event-Based Control.** Chapter 5 focuses on two aspects. The first aspect is to study of the effect of realistic communication in the distributed event-based control design presented in Chapter 4. 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. 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 for different situations. Secondly, two improvements are proposed. The first one 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. Based on this model, it estimates its state continuously and computes the control law accordingly. A certain model uncertainty is assumed and the performance of the Chapter 4 approach and model based designs are compared based on this model uncertainty. Parts of this work were presented in the CDC of 2011 and 2012 (see [GDJ+11, GLS+12]), and the model based approach is also included in the accepted paper mentioned in Chapter 4.

- **Chapter 6. Simulation Tools and Application Example of the DEBC: Networked Mobile Robots.** The formation control of networked mobile robots can be taken 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. In particular, the formation control from a consensus problem point of view under a wide range of network conditions and multiple experiments can be studied with this platform. The interactivity of the tool with the final user has been in the focus of the developers, as well as offering flexibility to define the experiment conditions. Moreover, multiple parameters can be changed on-line while running a simulation by simple click-and-drag actions in the graphical interface. The DEBC algorithms have been also implemented in a testbed of mobile robots, and the results are presented. This work has

been published in the IEEE Network Magazine (simulation tool) [GFF+12] and the application to a real platform has been submitted to Sensors.

- **Chapter 7. Conclusions and Future Work**. The conclusions and future research steps are given.

1.7 Publications and projects

Journal Papers

1. M. Guinaldo, J. Sánchez, S. Dormido. A co-design strategy of NCS for treacherous network conditions. *IET Control Theory & Applications*, 5(16): 1906-1915, 2011.

2. M. Guinaldo, G. Farias, E. Fabregas, J. Sánchez, S. Dormido-Canto, S. Dormido. An Interactive Simulator for Networked Mobile Robots. *IEEE Network Magazine*, 26(3): 14-20, 2012.

3. M. Guinaldo, D.V. Dimarogonas, K.H. Johansson, J. Sánchez, S. Dormido. Distributed Event-Based Control Strategies for Interconnected Linear Systems. *IET Control Theory & Applications*, 2013, *Accepted on* 1^{st} *February 2013*, doi: 10.1049/iet-cta.2012.0525.

4. M. Guinaldo, E. Fabregas, G. Farias, S. Dormido-Canto, D. Chaos, J. Sánchez, S. Dormido. Mobile robots experimental environment with event-based wireless communications. *Submitted to Sensors (current state: major revision).*

5. E. Fabregas, G. Farias, S. Dormido-Canto, M. Guinaldo, J. Sánchez, S. Dormido. Virtual and real laboratory for teaching mobile robotic. *Submitted to IEEE Transactions on Industrial Electronics.*

Conference Papers

1. M. Guinaldo, J. Sánchez, S. Dormido. Diseño de un Sistema de Control Anticipativo Basado en Paquetes para Control en Red. *9ª Conferencia Iberoamericana en Sistemas, Cibernética e Informática* (CISCI 2010), July 2010, Orlando.

2. M. Guinaldo, J. Sánchez, S. Dormido. A Packet-based Network Control System Architecture for Teleoperation and Remote Laboratories. *49th IEEE Conference on Decision and Control* (CDC), December 2010, Atlanta.

3. M. Guinaldo, D.V. Dimarogonas, K.H. Johansson, J. Sánchez, S. Dormido. Distributed Event-Based Control for Interconnected Linear Systems. *50th IEEE Conference on Decision and Control and European Control Conference* (CDC-ECC), December 2011, Orlando.

4. M. Guinaldo, J. Sánchez, S. Dormido, M.A. Delgado. Control en red basado en eventos de múltiples plantas remotas. *XXXIII Jornadas de Automática*, September 2012, Vigo.

5. M. Guinaldo, D. Lehmann, J. Sánchez, S. Dormido, K.H. Johansson. Distributed Event-Triggered Control with Network Delays and Packet-losses. *51th IEEE Conference on Decision and Control* (CDC), December 2012, Maui.

6. M. Guinaldo, J. Sánchez, S. Dormido. Contribuciones al control en red basado en eventos para sistemas lineales. *XI Simposio CEA de Ingeniería de Control*, April 2013, Valencia.

7. M. Guinaldo, D. Lehmann, J. Sánchez, S. Dormido, K.H. Johansson. Reducing communication and actuation in distributed control systems. *Submitted to the 51th IEEE Conference on Decision and Control*, 2013.

Other Publications

1. M. Guinaldo, B. Pérez-Lancho, E. Sanz. Laboratorio virtual para el aprendizaje del control térmico en edificios. *V Jornadas de Enseñanza a Través de Internet/Web de la Ingeniería de Sistemas y Automática*, September 2007, Zaragoza.

2. M. Guinaldo, E. Sanz, S. Dormido. Laboratorio Virtual Basado en Web para Aprendizaje de Física usando Ejs. *XXIX Jornadas de Automática*, September 2008, Tarragona.

3. M. Guinaldo, J. Sánchez, H. Vargas, S. Dormido. Laboratorio basado en Web del sistema bola y viga para el entrenamiento de estrategias de control avanzado. *XXX Jornadas de Automática*, September 2009, Valladolid.

4. M. Guinaldo, H.Vargas, J. Sánchez, S. Dormido. Web-Based Control Laboratory: The Ball and Beam System. *8th IFAC Symposium on Advances in Control Education* (ACE09), October 2009, Kumamoto.

5. M. Guinaldo, J. Sánchez, H. Vargas, S. Dormido. An Advanced Web-based Control Laboratory for the Ball and Beam System. *9th Portuguese Conference on Automatic Control* (CONTROLO'2010), September 2010, Coimbra.

6. M. Guinaldo, L. de la Torre, R. Heradio, S. Dormido. A Virtual and Remote Control Laboratory in Moodle: The Ball and Beam System. *Submitted to the 10th IFAC Symposium Advances in Control Education*, 2013.

Research Projects

The results obtained in the framework of this dissertation have been supported by different research projects:

- *Event-based modeling, simulation, and control* (2007-2012). Spanish Ministry of Science and Technology, CICYT (Ref. DPI2007-61068). Participants: UNED (Spain), University of Murcia (Spain). Directed by Prof. Sebastián Dormido Bencomo.

- *MACROBIO: Modeling, simulation, control and optimization of photobioreactors* (2012-2014). Spanish Ministry of Economy and Competitiveness, CICYT (Ref. DPI2011-27818-C02-2). Participants: UNED (Spain). Directed by Prof. José Sánchez Moreno.

- *Event-based control of distributed and collaborative systems* (2012-2014). Spanish Ministry of Economy and Competitiveness, CICYT (Ref. DPI2012-31303). Participants: UNED (Spain). Directed by Prof. Sebastián Dormido Bencomo.

2

Anticipative Control Design in Internet-like Networks

Summary

This chapter presents the analysis and design of anticipative controllers for packet-based NCS. The remote controller uses a model of the plant and a basis controller to compute a sequence of future control actions to compensate the effect of delays and packet dropouts. This sequence is stored into the actuator buffer and is applied synchronously at each sampling time.

Two middleware layers between the process and the network, and between the controller and the network are designed to hide the elements which do not belong to a conventional control loop.

First, an scheme in which the sensor sends the measurements periodically is presented initially, and an event-based approach is proposed afterwards for a more efficient usage of the network bandwidth.

The system results to be GUUB when some constraints are imposed to the network delay. Different extensions such as disturbance estimators, output measurement and LTI anticipative controllers are discussed, preserving the GUUB property of the system. Finally, a centralized remote controller for a multi-loop architecture is presented.

2.1 Introduction

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

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

To achieve this, a common approach is to use model-based control to emulate future states of the plant and, therefore predictions for the control signal. The idea of combining packet-based control and Model Predictive Control (MPC) was first introduce in [Bem98] in the context of teleoperation. Since then, other authors have exploited the principle of MPC in packet-based NCS [KJA06, QSG07, MJVR08, VF09, ICMS11].

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

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

Whereas these approaches results in a more efficient usage of the network bandwidth and possible enlargement of transmission intervals, few publications have combined receding horizon control and event triggering. In [ESDCM07], a transmission protocol named as Input Difference Transmission Scheme (IDTS),

that calculates a new control sequence at every time step but only transmits to the actuator when the difference between the new sequence and that in the buffer has exceeded a threshold. In [GDJ+11] the sensor sends measurements to the controller if the difference between the predicted state by a model, which is sent with the predictions of the control signal, and the measured state crosses a given level. More recently, a model-based periodic event-triggered control is exploited to reduce the number of transmissions [HD13], where two frameworks are proposed, perturbed linear and piecewise linear systems, to achieved global exponential stability and ℓ_2 gain performance.

The outline of this chapter is as follows. The original contributions are given in Section 2.2. Section 2.3 states the assumptions that are taken in this chapter. The guidelines of the design of middleware layers are given in sections 2.4 and 2.5, which are adapted to an event-triggered scheme in Section 2.6. The stability of the system is studied in Section 2.7, and sections 2.8-2.10 present different extensions such as disturbance estimation, output measurement, and the centralized control of N loops. Finally, conclusions end the chapter.

2.2 Contributions of this chapter

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

One of the novel contributions is that the proposed design is aware of a more efficient usage of the bandwidth but also of facing delays and packet losses without assuming clock synchronization of the elements in the control loop, in contrast to other works in the literature such as [NH06, QSG07, QSG08, ZLR09, HD13].

Moreover, the model-based controller avoids the additional delays caused by the computational time required by MPC, and so the proposed approach seems especially adequate in processes with fast dynamics. Also, the theoretical analysis ensures the stability of the system if the network delay is upper bounded.

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

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

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

2.3 Assumptions

In the sequel of this chapter the following assumptions hold:

- *System architecture:* There is a single control loop with a remote controller, i.e., Figure 1.5c. We assume that the sensor and the actuator have a very limited computation capacity and the controller is the element which makes the computation effort. The actuator processes the incoming packets and store the data into a buffer. The sensor measures the state at each sampling time and is able to compare it to a reference value and, in case, to trigger an event. The remote controller also has a buffer to store the incoming

measurements.

- *System dynamics:* We consider linear plants and a sampling period denoted by T_s, that meets Nyquist criteria. Thus, the system dynamics is given by

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

$$y(k) = Cx(k) + v(k), \tag{2.2}$$

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

$$A_d = e^{AT_s} \tag{2.3}$$

$$B_d = \int_0^{T_s} e^{As} B ds. \tag{2.4}$$

The pair (A, B) is controllable.

- *Measures of time*: All the instances or intervals of time, such as delays, the occurrence of events, etc. are given as integer numbers $k \in \mathbb{N}$, so that the measurements in units of time are t_k which are multiple of the sampling period, i.e., $t_k = kT_s$.

- *Controller:* The basis controller is assumed to be state feedback if the full state is measurable, and LTI if only the output is measured. However, the framework can be extended to other controllers that stabilize the plant in a network-free system configuration.

- *Clocks synchronization:* The elements at the plant side (sensor, actuator and event detector[1]) are ruled by the same clock and hence, clock synchronization is assumed for them. By contrast, the remote controller clock is not synchronized with any of the other elements in the control loop and works

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

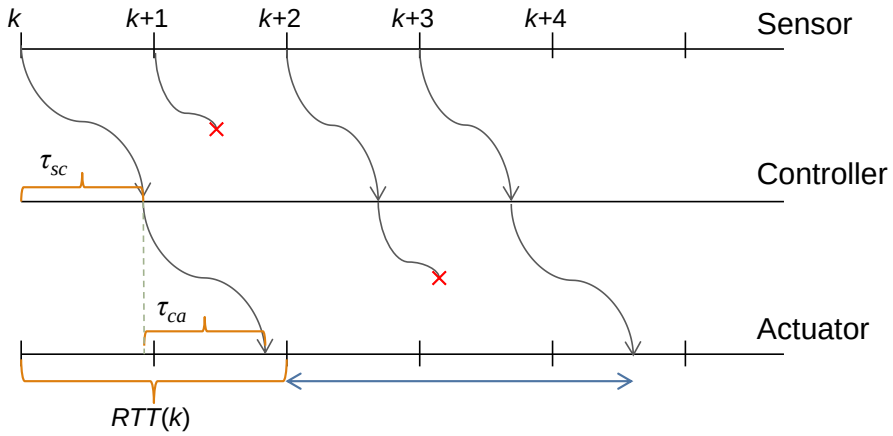

Figure 2.1: Examples of delays and data dropouts.

asynchronously. When a new measurement is received by the controller, it computes a new sequence of future control values.

- *Network:* We consider Internet-like networks. Hence, network protocols such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) can be used, and so the size of the packets payload is around 500 bytes [Eva98]. It is assumed that there are not powered-batteries devices so that energy consumption is not affected by the packet length. Moreover, packets can experience long delays or be lost during transmissions across the network.

Example 2.1: A simple chronogram is shown in Figure 2.1 to illustrate the phenomenon of delays and packet losses. 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.,

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

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

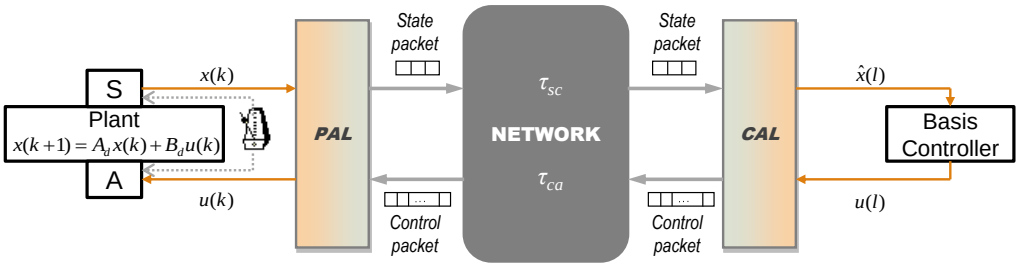

Figure 2.2: Proposed architecture for packet-based NCS.

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

We next briefly describe these two layers:

- The Controller Adaptation Layer (CAL) receives and processes the state packets sent from the plant side. Its main tasks are to estimate the future states of the plant and to interact with the basis controller to compute the sequences of future control actions. Hence, the main element of this layer is the model of the plant.

- The Process Adaptation Layer (PAL) receives the control packets and decides which control input is applied at each sampling time. Also, in an event-based approach, it decides when to transmit the measurements to the controller through the network.

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

2.4 The Controller Adaptation Layer (CAL)

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

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

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

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

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

2.4.1 Packets processing

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

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

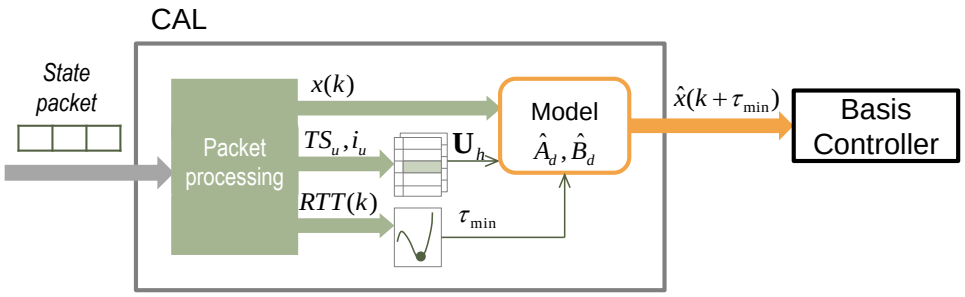

Figure 2.3: Packets processing by the CAL layer.

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

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

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

2.4.2 Control sequence computation

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

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

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

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

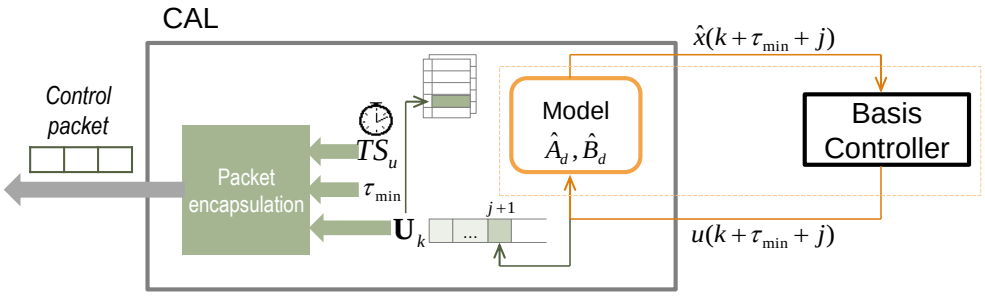

Figure 2.4: Control sequence computation by the CAL layer.

next sampling time is

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

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

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

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

2.5 The Process Adaptation Layer (PAL)

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

prior states of the plant. Thus, there is a queue for the incoming packets and one buffer which contains the current control sequence that is being handled.

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

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

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

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

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

Example 2.2: Assume that the sampling period for a given plant is $T_s = 5ms$ and that, at a given instance of time, the value of τ_{min} is of two sampling periods, i.e., 10 ms. Thus, for a measurement $x(k)$ received by the controller, the value of $\hat{x}(k+2)$ is estimated and the

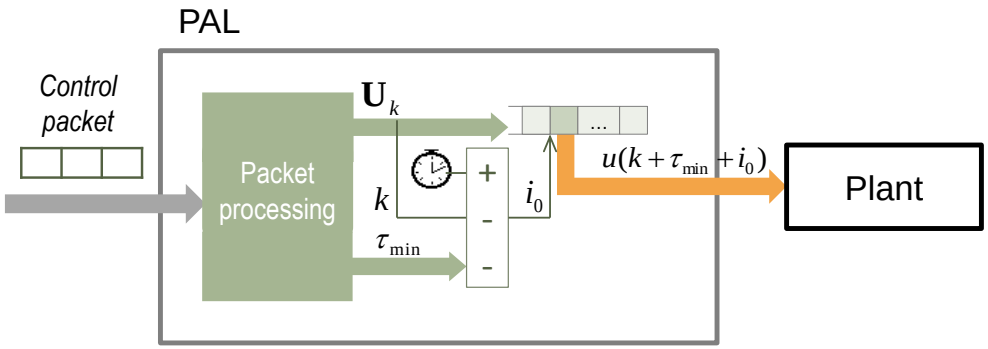

Figure 2.5: Packets processing by the PAL layer.

control sequence $\mathbf{U}_k$ computed as described before. Assume that when $\mathbf{U}_k$ is received at the PAL, the local time is $k+5$, that is, between the measurement of $x(k)$ and the reception of the control sequence, five sampling periods have passed (25 ms). Hence, the first three elements of $\mathbf{U}_k$ are discarded and $\mathbf{U}_k(4)$ is applied since $i_0 = (k+5) - k - \tau_{min} = k+5-k-2 = 3$.

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

2.6 Event-based anticipative control

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

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

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

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

to be

$$f(e(k)) = \|e(k)\| - c, \tag{2.8}$$

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

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

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

- *Event detector:* The event detector is the physical element which monitors the trigger condition. This element is collocated with the sensor and when an event is detected, the measurement is sent to the controller.

2.6.1 CAL design for event-based control

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

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

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

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

Figure 2.6 illustrates the new design of the CAL. The predicted states se-

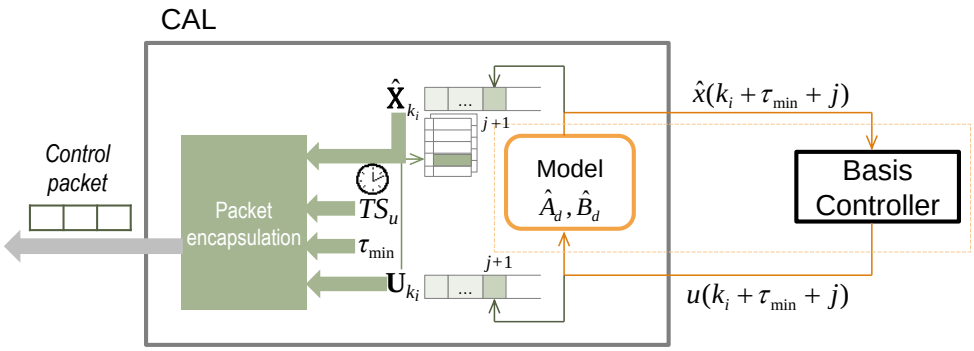

Figure 2.6: Control and state sequences computation by the CAL layer in the event-based design.

quence $\hat{\mathbf{X}}_{k_i}$ is highlighted respect to the rest of the elements which are encapsulated to result in a control packet.

Remark 2.1. Note that the length of the control sequence $\mathbf{U}_{k_i}$ has to be cut down to include $\hat{\mathbf{X}}_{k_i}$ in the control packet. Specifically, if we denote by Q_P the length of $\mathbf{U}_k$ when measurements are sent to the controller periodically, and by Q_{EB} the length of $\mathbf{U}_{k_i}$ in the event-based design, it holds that

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

where n is the state dimension and m the number of control inputs.

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

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

where $\mathcal{S}$ is the operator *size of*. We assume that all the elements of the control packets are float. Therefore, the number of future control sent in an control packet is 125.
However, an event-based design cuts down this value to $Q_{EB} = \frac{125}{n+1}$. For example if $n = 4$, then $Q_{EB} = 25$.

2.6.2 PAL design for event-based control

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

However, the detection of events has to be included in the PAL design. The code of this new module of the PAL is given by Algorithm 2.1. The control and state predictions sequences are received as inputs as well as the computed index i_0. The error and the trigger function are initialized to default values (lines 2-3). The state of the plant is measured at each sampling time, and the error and trigger functions are computed (lines 7-9). The event condition is checked at each sampling time (line 10). In case an en event is triggered, the module delivers $x(k_i)$ and the index value as outputs.

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

Input: $\mathbf{U}_{k_i}, \hat{\mathbf{X}}_{k_i}, i_0$ with $k_i < k$
Output: $x(k_{i+1}), i_0 + j$
1: $j := 0$
2: $e(k) := \mathbf{0}$
3: $f(e(k)) := -1$
4: **while** $i_0 + j < \bar{Q}$ **and** $f(e(k)) < 0$ **do**
5: $\quad j := j + 1$
6: $\quad$ Apply $\mathbf{U}_{k_i}(i_0 + j)$
7: $\quad$ Measure $x(k)$
8: $\quad \hat{x}(k) := \hat{\mathbf{X}}_{k_i}(i_0 + j)$
9: $\quad e(k) := \hat{x}(k) - x(k)$
10: $\quad$ Compute $f(e(k))$
11: **end while**

Algorithm 2.1: PAL event-detection algorithm.

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

2.7 Stability analysis

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

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

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

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

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

$$x(k+1) = (A_d + B_d K)x(k) + B_d K e(k). \tag{2.10}$$

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

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

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

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

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

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

$$\|e(k)\| \leq 2c. \tag{2.11}$$

Proof. From Assumption 2.1 it follows that

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

since no event is detected in this interval.

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

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

which concludes the proof. □

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

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

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

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

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

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

$$A_{dK}^T P A_{dK} - P = -Q, \tag{2.13}$$

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

$$V(x) = x^T P x. \tag{2.14}$$

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

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

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

where

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

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

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

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

which is upper bounded by

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

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

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

where σ is given in (2.16).

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

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

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

2.7.1 Analysis of the maximum RTT and the model uncertainties

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

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

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

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

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

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

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

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

The maximum RTT can be derived imposing that

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

which yields to a complicated expression which depends not only on the system and model dynamics but also on the state at the last event $x(k_i)$. It is not possible to derive an analytical solution for it, but the feasibility of the solution requires

a bound for $x(k_i)$ $\forall k_i$. Its existence is guaranteed from the results in Theorem 2.1.

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

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

Thus, according to Proposition 2.1, it holds that

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

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

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

which is independent of the value of c.

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

$$\|A_d^{\tau_{max}} - I\| < \nu.$$

Example 2.4: Assume that the scalar system

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

is sampled with a sampling period T_s. An anticipative controller based on events is designed for this system, in which the event detector detects an event whenever the error crosses a threshold c. Assume that there is no model uncertainty in the anticipative controller. Let us compute the maximum allowable RTT for the system (2.22).
It holds that $A_d = e^{aT_s}$. Thus, according to (2.21), it holds that

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

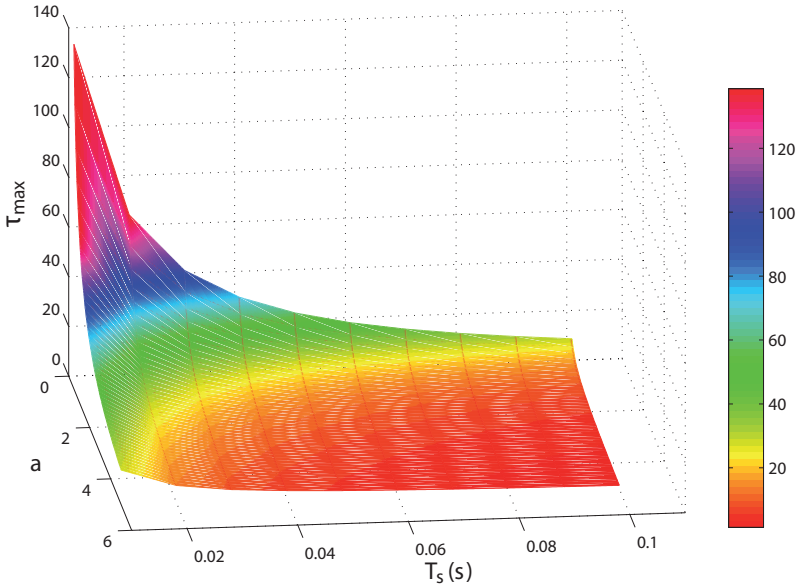

Figure 2.7: Surface defined by (2.23).

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

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

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

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

2.7.2 Analysis of the error bounds

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

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

Also, if the delay τ can be measured because the architecture has a different configuration (e.g. Figure 1.5a), the state of the plant at the time instance $k+\tau$ can be estimated, and the error is reset to zero if the model is perfect.

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

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

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

While the prediction that the model gives is

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

Then, it follows that the error is

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

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

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

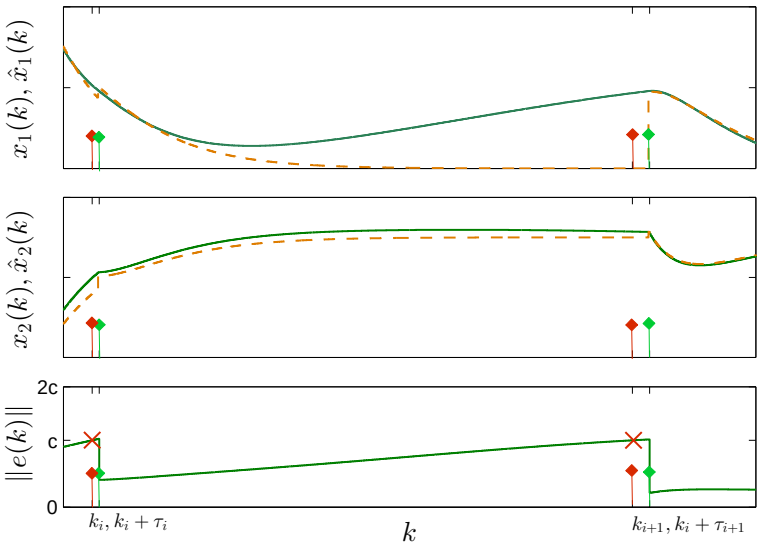

Figure 2.8: Comparative of the state (solid line) and the model (dotted line), and the error bound. k denotes the sampling time, k_i, k_{i+1} the events occurrence, and τ_i, τ_{i+1} the delays.

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

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

2.8 Disturbance estimator

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

In [LL10], disturbances are estimated at event times assuming that they are

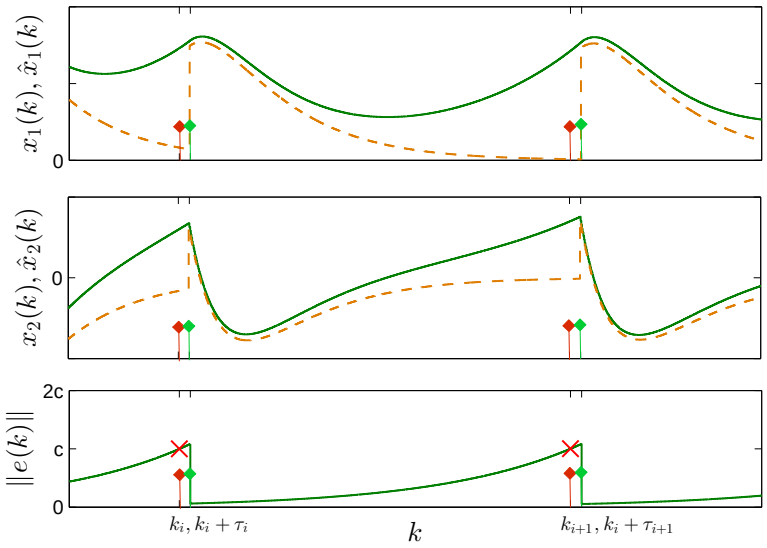

Figure 2.9: Comparative of the state (solid line) and the model (dotted line), and the error bound. k denotes the sampling time, k_i, k_{i+1} the events occurrence, and τ_i, τ_{i+1} the delays.

constant between events in the proposed emulation approach, which mimics the continuous state feedback control. One constraint of the design is that the matrix A must be invertible, which excludes integrators from the dynamics of the system.

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

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

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

  where $\hat{w}(k)$ is the estimated disturbance at time k.
- The state $x(k)$ is measurable.
- When a state packet is received with a measurement taken at time k, the disturbance is estimated before computing the next control sequence $\mathbf{U}_k$, and held constant in the next steps.

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

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

$$e(k+1) = \hat{x}(k+1) - x(k+1) = \hat{A}_d\hat{x}(k) - A_d x(k) + (\hat{B}_d - B_d)u(k) + (\hat{w}(k) - w(k)), \tag{2.26}$$

where $u(k)$ is given by (2.6). The disturbance $w(k)$ could be calculated if the rest of the terms of (2.26) were known. However, the model mismatch is unknown. Therefore, if the approximation $\bar{A}_d \approx \mathbf{0}, \bar{B}_d \approx \mathbf{0}$ is taken, the value of $w(k)$ can be computed at the next sampling time $k+1$ (after measuring $x(k+1)$) as

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

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

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

when disturbances are included in the model.

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

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

Example 2.6: Consider that the system is a double integrator:

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

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

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

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

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

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

Note that $e(k)$ in (2.28), which denotes the error between the estimated state and the measured state at the time of the reception of the control sequence is in general non-zero. This information as well as the error at the time of the detection of the event must be known. This implies that both values have to be transmitted from the PAL to the CAL. Thus, the *state packets* must include the following information:

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

According to this, the code executed by the CAL is illustrated in Algorithm 2.2.

Note that once $\hat{w}(k_i)$ is estimated, it is used in the estimation of $\hat{x}(k_i+\tau_{min})$ and the computation of $\mathbf{U}_{k_i}, \hat{\mathbf{X}}_{k_i}$.

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

Algorithm 2.2: Code executed in the CAL for disturbance estimation.

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

Remark 2.7. Note that $\hat{w}(k)$ includes the effect of the disturbance $w(k)$ and also of the model uncertainty $\bar{A}_d, \bar{B}_d$. Both effects cannot be separated with the proposed approach, but are compensated, though.

2.8.1 Stability analysis

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

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

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

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

$$x(k+1) = A_{dK}x(k) + B_dKe(k) + w(k). \tag{2.29}$$

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

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

where

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

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

$$\delta_b = \|(B_dK)^T PA_{dK}\|2c + \|PA_{dK}\|w_{max} \tag{2.33}$$

$$\delta_c = \|(B_dK)^T PB_dK\|4c^2 + 4\|PB_dK\|w_{max}c + \lambda_{max}(P)w_{max}^2. \tag{2.34}$$

Proof. The proof can be found in the Appendix B on page 245. □

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

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

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

Figure 2.10 shows the state of the system (solid line), the prediction given by the model (dashed line), the norm of the error, the control input and the real and estimated disturbances. Note that the major number of the events occur for small values of time (when the state of the system is far from the equilibrium), and when the value of the disturbance changes. The difference between the real and the estimated states is not well appreciated due to the scale and the small value of c.

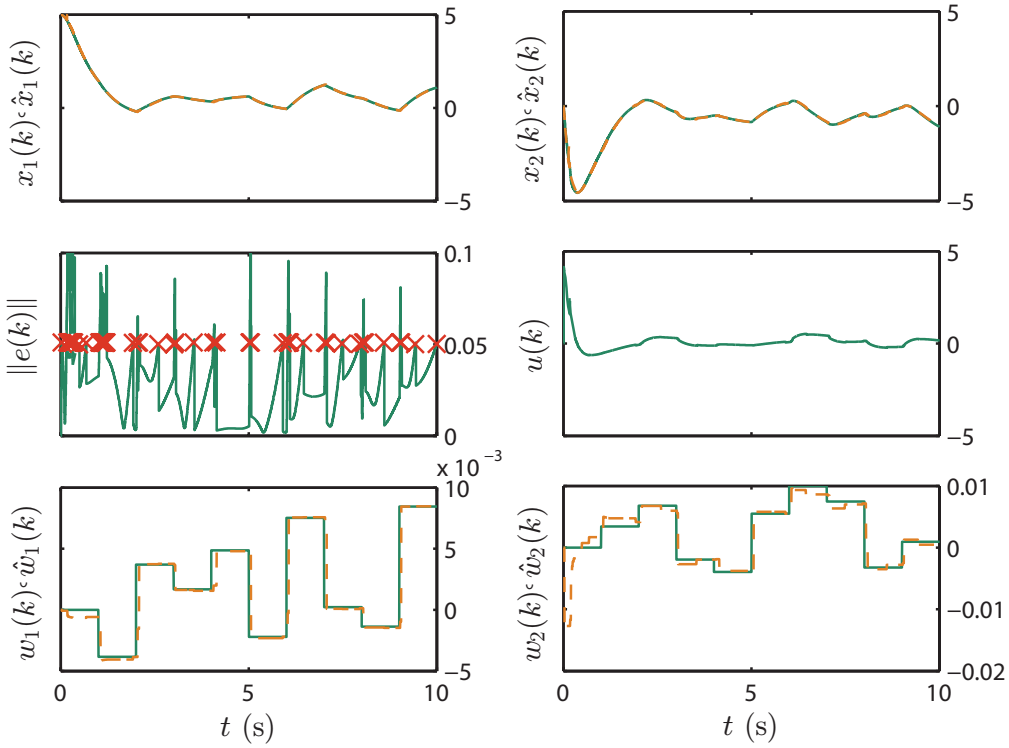

Figure 2.10: Disturbances estimation. The estimated values are represented by the dotted line, and the actual values by the solid line.

2.9 Output-based event-triggered control

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

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

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

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

where $\tilde{x}(k)$ is the observed state. The analysis shows that the property of GUUB is preserved. In [LL11c], a Kalman filter approach is presented.

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

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

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

$$f(e_y(k)) = \|e_y(k)\| - c_y. \tag{2.36}$$

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

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

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

for the basis controller. $x_C(k)$ is the state of the controller, and A_C, B_C, C_C and D_C are matrices of the appropriate dimensions. We assume that the controller is designed to render the system asymptotically stable when $\hat{y}(k)$ is replaced by $y(k)$. We further assume that the pair (A_d, C) is observable and that a model is

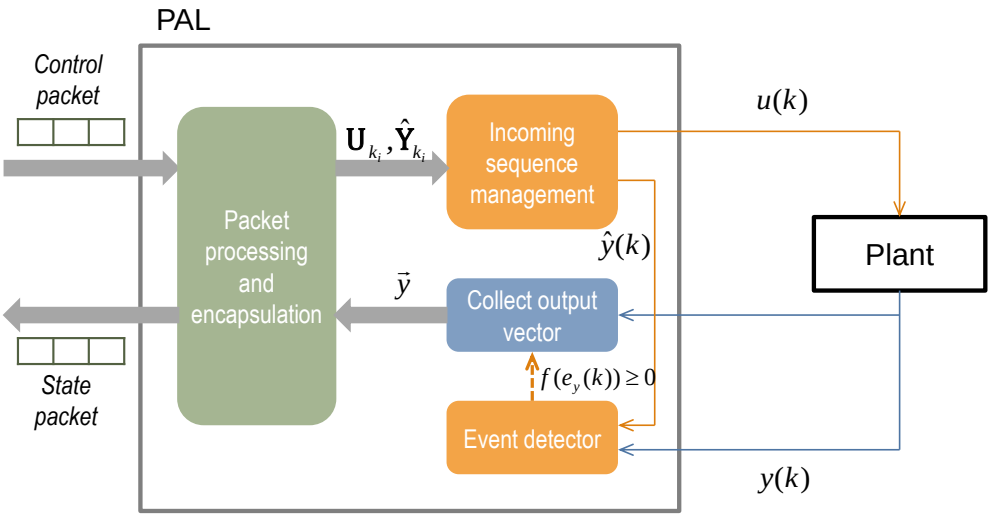

Figure 2.11: PAL design for output measurement.

available and it is given by $(\hat{A}_d, \hat{B}_d)$, and $\hat{C} = C$. Finally, disturbances affecting the system (2.2) are not considered for simplicity. However, the measurement noise $v(k)$ might not zero but bounded by v_{max}.

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

2.9.1 PAL design for output measurement

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

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

- *Incoming sequence management:* This module is in charge of selecting the control input at each sampling time, as described in Section 2.5. Since event-triggering is supported, it also manages sequence of predictions given by the model. This sequence has been denoted as $\hat{\mathbf{X}}_{k_i}$ for state measurement. For output-based control, the controller sends $\hat{\mathbf{Y}}_{k_i}$ referring to a sequence of predicted outputs. The details of how $\hat{\mathbf{Y}}_{k_i}$ is computed can be found in the next section.

- *Event detector:* It monitors the system output at each sampling time. If the

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

Algorithm 2.3: PAL event-detection algorithm for output measurement.

error (2.35) exceeds a certain threshold, i.e., the trigger function becomes positive, an event is generated. This is illustrated in Algorithm 2.3.

- The completely novel module in the PAL for output measurement is the one in charge of *collecting an output vector* denoted as $\overrightarrow{y}$. The measured outputs $y(k)$ at each sampling time k between the reception of a control packet and the detection of a new event are stored in $\overrightarrow{y}$ (see Algorithm 2.3). This information is used by the PAL to estimate the state of the plant via an state observer. Note that $\overrightarrow{y}$ can actually be a matrix if the system has multiple outputs. Since the inter-event time is limited by the fact that an event is triggered when the index of the control sequence reaches the value $\bar{Q}$, there is no need in imposing an additional constraint to the length of $\overrightarrow{y}$.

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

2.9.2 CAL design for output measurement

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

- The controller structure: The new basis controller is given by (2.37)-(2.38). Hence, it receives from the model the predicted output of the plant $\hat{y}(k)$, it

computes its next internal state according to (2.37) and the control input as (2.38).

- The model needs to compute an estimation of the state of the plant $\hat{x}(k)$ according to (2.5), but only $\hat{y}(k) = C\hat{x}(k)$ is required by the controller.

- In 2.6.1, a predicted states sequence $\hat{\mathbf{X}}_{k_i}$ is generated and sent to the process. This information is not useful anymore since the state is not measurable, therefore, a predicted outputs sequence $\hat{\mathbf{Y}}_{k_i}$ is used instead. Since we have assumed that $\hat{C} = C$, it holds that $\hat{\mathbf{Y}}_{k_i} = C\hat{\mathbf{X}}_{k_i}$. Note that one advantage of this approach is that the length of $\mathbf{U}_{k_i}$ for an output-based scheme is, in general, larger than for full state measurement in event-based communications (see Remark 2.1).

- Since $x(k_i)$ is no longer available, it is estimated by a state observer using the information in $\overrightarrow{y}$, generating future states of the plant after that. We next describe this more in detail.

A Luenberger observer of the state

A Luenberger observer of the form

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

is used to obtain the state $x(k)$, being $(\hat{A}_d - LC)$ Hurwitz. We use the notation $\tilde{x}(k)$ rather than $\hat{x}(k)$ to differentiate it from the model predictions given by (2.5). Anytime a new state packet is received at the controller side, the code of Algorithm 2.4 is executed. The length of $\overrightarrow{y}$ 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 by checking them in a look-up table (see Section 2.4). Then, the Luenberger observer estimates the

Input: $\overrightarrow{y}, TS_u, i_u$
Output: $\tilde{x}(k_i)$
1: $n := \dim(\overrightarrow{y})$
2: $[u(k_i - n - 1) \dots u(k_i)] :=$ getFromLookupTable(TS_u, i_u, n)
3: **for** $j = 1 \rightarrow n$ **do**
4: $\quad y(k_i - n - 1 + j) := \overrightarrow{y}(j)$
5: $\quad \tilde{x}(k_i - n + j) := (\hat{A}_d - LC)\tilde{x}(k_i - n - 1 + j) + \hat{B}_d u(k_i - n - 1 + j) + Ly(k_i - n - 1 + j)$
6: $\quad \tilde{x}(k_i - n - 1 + j) := \tilde{x}(k_i - n + j)$
7: **end for**

Algorithm 2.4: Luenberger observer state estimation.

plant state $x(k_i)$ corresponding to the last output measurement $y(k_i)$, which is the last element of the output vector $\overrightarrow{y}$.

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

2.9.3 Stability analysis

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

$$x(k+1) = A_d x(k) + B_d u(k) \tag{2.39}$$

$$y(k) = Cx(k) + v(k) \tag{2.40}$$

$$x_C(k+1) = A_C x_C(k) + B_C \hat{y}(k) \tag{2.41}$$

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

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

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

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

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

$$F = \begin{pmatrix} B_d D_C \\ B_C \end{pmatrix}. \tag{2.44}$$

Thus, the closed-loop system-controller dynamics is

$$\xi(k+1) = A_{CL}\xi(k) + Fe_y(k) + Fv(k). \tag{2.45}$$

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

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

We define the Lyapunov function

$$V(\xi) = \xi^T(k) P \xi(k). \tag{2.46}$$

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

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

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

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

THEOREM 2.3. *If Assumption 2.1 holds, the augmented state $\xi(k)$ of the system-controller* (2.45)*, when the event detector is defined by* (2.36) *and the measure-*

ment noise is bounded $\|v(k)\| \leq v_{max}$*, is GUUB with bound*

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

where

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

Proof. The proof can be found in the Appendix B on page 246. □

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

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

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

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

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

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

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

2.9.4 PI anticipative control

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

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

State representation of a PI controller

A conventional continuous-time PI controller can be written as

$$u(t) = K_p\Big((by_{sp} - y(t)) + \frac{1}{T_i}\int_0^t (y_{sp} - y(\tau))d\tau\Big).$$

The state of the controller x_C can be defined as

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

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

$$u(t) = x_C(t) + K_p(by_{sp} - y(t)). \tag{2.51}$$

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

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

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

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

The output is

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

and the control input

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

Control and predicted states sequences computation

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

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

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

where

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

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

$$\hat{\mathbf{Y}}_{k_i}(j) = \begin{pmatrix} C & 0 \end{pmatrix} \Big[\hat{A}_{CL}^j \hat{\xi}(k_i + \tau_{min}) + \sum_{l=0}^{j-1} \hat{A}_{CL}^l \hat{F}_b y_{sp} \Big], \tag{2.55}$$

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

$$\mathbf{U}_{k_i}(j+1) =(-K_PC \quad 1)\Big[\hat{A}_{CL}^{j}\hat{\xi}(k+\tau_{min})+\sum_{l=0}^{j-1}\hat{A}_{CL}^{l}\hat{F}_b y_{sp}\Big]+K_Pby_{sp}. \quad (2.56)$$

2.10 Centralized anticipative control for N subsystems

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

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

- The system dynamics is given by

$$x_i(k+1) = A_d^i x_i(k)+B_d^i u_i(k)+w_i(k), \quad x_i(0)=x_{0,i} \quad (2.57)$$

$$y_i(k) = C^i x_i(k)+v_i(k), \quad i=1,\ldots,N, \quad (2.58)$$

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

- There is a model for each plant given by

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

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

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

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

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

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

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

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

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

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

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

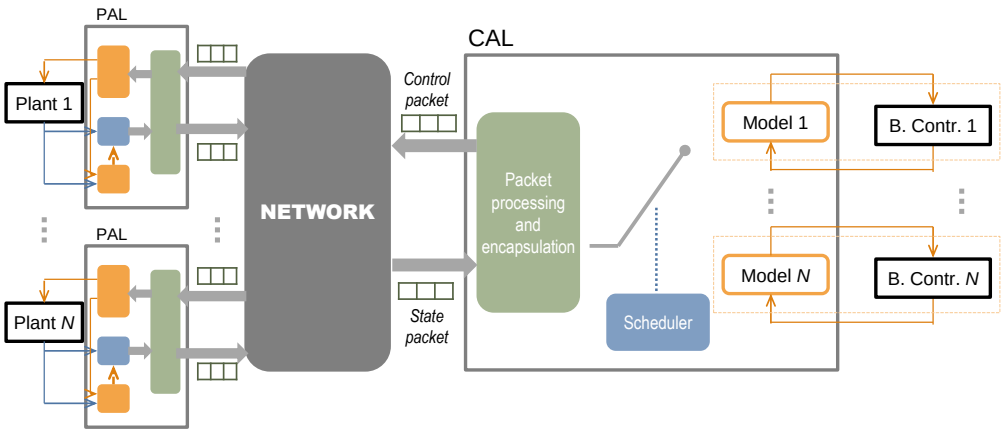

Figure 2.12: CAL design of a centralized anticipative controller for N plants.

the number of subsystems.

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

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

Example 2.8: Let us consider a set of N scalar systems

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

Each of them is sampled with a sampling period h_i. In the Example 2.4 an upper bound on the delay has been derived for the maximum RTT τ_{max} when a single system is considered so that Assumption 2.1 holds. This value is given by $\tau_{max} < \frac{\log 2}{ah}$.
When the number of controlled plants increase, the waiting delays on the packet queues in the controller also grows. If T_c^j is the computational time required to process a packet, compute the arguments of a new control packet, and encapsulate the new control packet for the process j, the worst case of the waiting time τ_W^i for another process i can be computed

as

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

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

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

2.10.1 The scheduler

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

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

Mathematically, the priority can be computed as

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

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

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

Table 2.1: Parameters of the subsystems.

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

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

The priority π^i assigned by the scheduler to each subsystem is shown in Figure 2.13c. This π^i is computed taking into account the values of i_u^i (Figure 2.13b) and τ_{min}^i (Figure 2.13a), both of which change during the simulation period. The subsystem 1 (blue line) and subsystem 2 (green line) has an initial priority of 1. The subsystem 3 (red line) has $\pi_0^3 = 2$ and, finally, $\pi_0^4 = 3$ (magenta line). Notice that, for instance, $\pi^3 > \pi^4$ in $k \in [64, 71]$ even though $\pi_0^3 < \pi_0^4$. Also, in $k \in [11, 13]$, $\pi^1 > \{\pi^2, \pi^3, \pi^4\}$ although it has the lowest initial priority. The reason is that $\tau_{min}^1 = \max\{\tau_{min}^i, i = 1, \ldots, 4\}$ for this period of time.

2.11 Conclusions

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

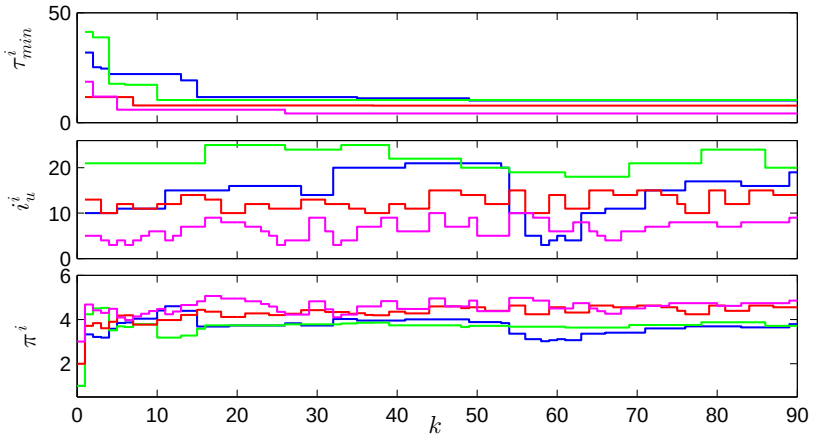

Figure 2.13: Priority assigned by the scheduler to each of the subsystems.

architecture.

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

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

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

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

3

Implementation and Experimental Evaluation of the Anticipative Control

Summary

This Chapter describes the implementation of the anticipative controller of Chapter 2 and the experimental results obtained with this implementation. First, the experimental framework is described. The PAL is hosted in a server, which acquires the measurements and is connected to the network. The controller and the CAL can be at any computer connected to the Internet.

Two are the processes in which the design has been evaluated: a servo motor and a flexible link. The description of these prototypes is given, and the design of the basis controller is presented.

The experimental evaluation consists of testing the design over a range of possible situations and also finding the best set of parameters. The influence of the trigger function and the effect of disturbances and network imperfections are analyzed. Finally, the results are discussed thoroughly.

3.1 Contributions of this chapter

The main contribution of this chapter is the implementation of the two middleware layers proposed in Chapter 2 from a high-level programming point of view. The proposed architecture facilitates the reuse of conventional controllers

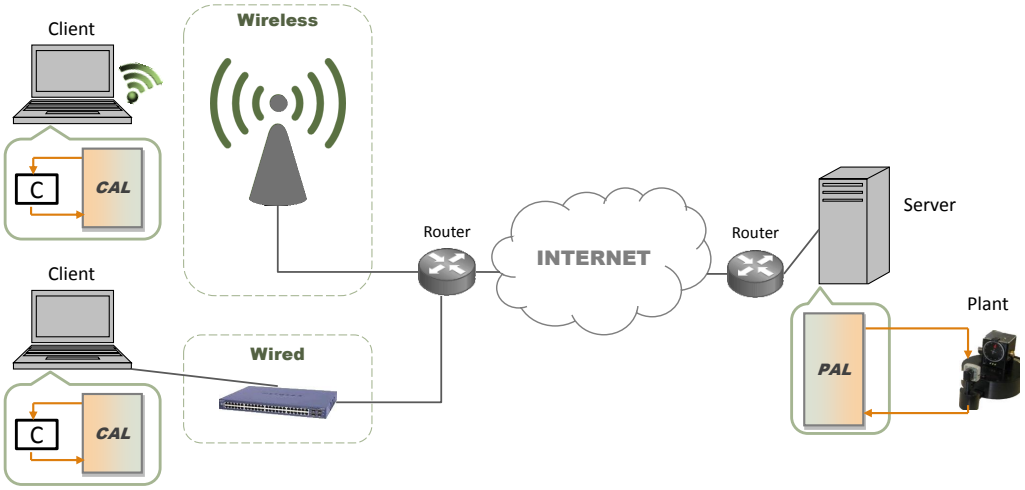

Figure 3.1: Scheme of the experimental framework.

in networked control with little effort if a model of the process is available.

Moreover, several experiments have been designed to test the controller under treacherous network conditions.

3.2 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 [Var10], which is based on a *client-server* architecture [KCC04].

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) [VSD09].

Hence, the implementation changes to adapt this environment to work using the remote anticipative controller described in Chapter 2 are given in next pages. An scheme of the experimental framework is shown in Figure 3.1. The PAL is hosted at the server side, which is connected to the Internet. The remote controller and the CAL are 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.

The description of the plants used in the experiments is given next.

3.2.1 Prototypes overview

The QUANSER SRV-02 setup

The SRV-02 device (see Figure 3.2) is a DC servo-motor located at the Automatic Control Laboratory of UNED in Madrid. It is specifically designed to experiment with angular position, as it has a decoder which determines the angle of the gear. For this reason, this setup can be used for different experiments. The speed is not measured, so it must be estimated from position measurements. The model for the plant can be found in Appendix A, which yields

$$G(s) = \frac{-46.7}{s^2 + 33.3s}. \tag{3.1}$$

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.

The plant is sampled every 10 ms to satisfy the Nyquist-Shannon theorem. Thus, the discrete-time system is

$$\hat{A}_d = \begin{pmatrix} 1 & 0.008505 \\ 0 & 0.7168 \end{pmatrix}, \quad \hat{B}_d = \begin{pmatrix} -0.002096 \\ -0.3972 \end{pmatrix}, \quad C = (1 \quad 0).$$

A PI controller has been design to control the angle. The parameters of this

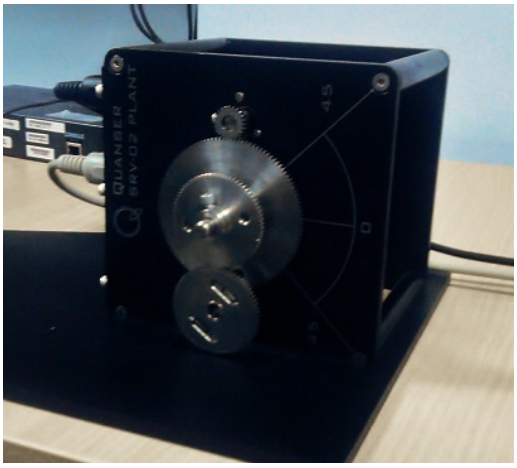

Figure 3.2: QUANSER SRV-02 gear.

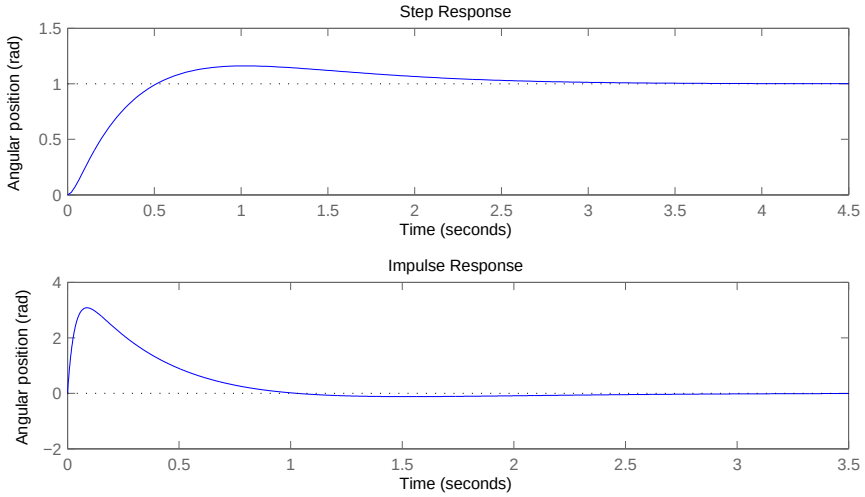

Figure 3.3: Step and impulse response of the SRV-02 gear model (3.1) with the PI controller (3.2).

controller are

$$K_P = -2.5, \quad T_i = 1. \tag{3.2}$$

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.

The step and impulse responses of the SRV-02 model (3.1) for this controller design are given in Figure 3.3.

The flexible link: QUANSER SRV-02 series

Figure 3.4 depicts the Flexible Link module coupled to the SRV02 plant described above. The module is attached to the SRV02 load gear by two thumbscrews.

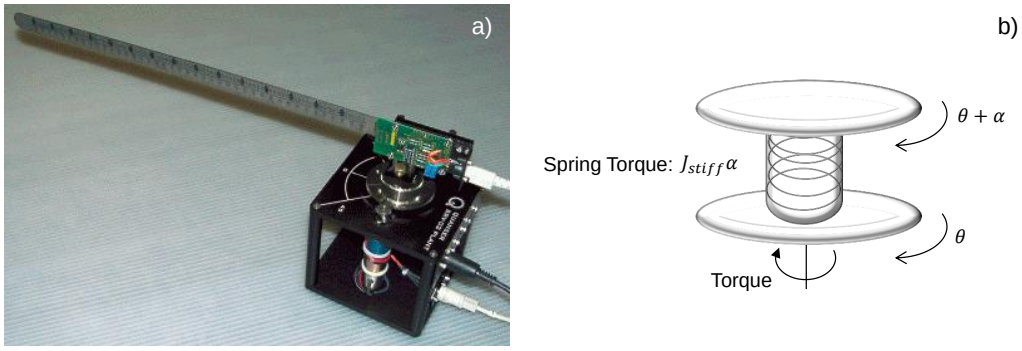

Figure 3.4: The flexible link: a) View of the module and b) model.

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. Figure 3.4b depicts the simplified model. The derivation of the model can be found in Appendix A. It results in

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 592 & 32 & 0 \\ 0 & -947.3 & -32 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -56.2 \\ 56.2 \end{pmatrix} u, \tag{3.3}$$

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.

A feedback gain K is design to control the system:

$$K = (17.3205 \quad -24.7388 \quad 1.7164 \quad 0.5007), \tag{3.4}$$

that sets the poles at $\{-48.13, -35.34, -8.43 + 11.50i, -8.43 - 11.50i\}$. If there is an external reference in the angular position, the input to the controller is the control error. The response of the model (3.3) with the feedback gain (3.4) to step and impulse inputs is illustrated in Figure 3.5. Note that the control error and not the state are depicted.

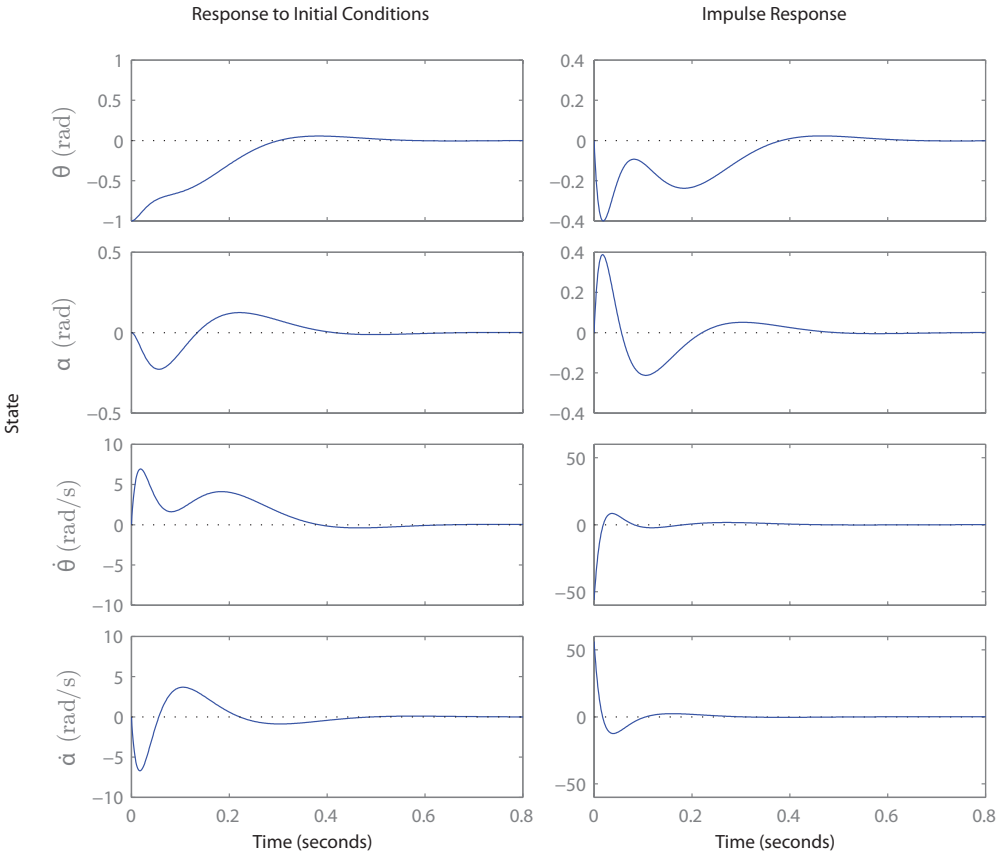

Figure 3.5: Step (left) and impulse (right) response of the flexible link model (3.3) with the feedback gain (3.4).

3.3 Implementing the CAL and the PAL in LAbVIEW

LabVIEW is a graphical programming language developed by National Instruments in 1968 [Lab13]. In its origins, LabVIEW was developed for data acquisition and instrumentation control. However, last versions of LabVIEW allow users to use it for many other purposes: process control, industrial automation, modeling and simulation, digital signal processing, remote operation, real-time programming, etc.

Programs developed with G, the graphical language of LabVIEW, are named VI (Virtual Instruments) due to its instrumentation-related origins. A VI is made up connecting multiple blocks, existing libraries of blocks for many purposes: vi-

sion, I/O hardware, mathematical calculations, simulation, Internet protocols, process control, database access, etc. One of the main reasons to use LabVIEW is its simplicity since users with low knowledge of programming can develop programs, difficult to write using traditional programming languages [Tra00, Blu07].

In a traditional experimentation environment, i.e., when the execution of the *all* control tasks (readings from sensors, executions of the control code, and writing to actuators) takes place in a local application, the required regularity is achieved by using threads and DAQ board interrupts.

However, the proposed design for remote experimentation demands to implement additional threads to manage the communication. LabVIEW seems adequate here, since it provides simple multi-thread programming using Timed Loop blocks. These blocks allow programmers to include multiple threads in a single virtual instrument (VI), and running these threads at different sampling periods and with different priorities. Moreover, these threads can also work asynchronously, i.e., the next execution of the loop will not take place until a new event occurs, for example, the reception of a new packet in the communication thread.

In the remote experimentation framework described in [Var10], the control thread receives a critical time priority, whereas the communication task has a lower priority, referring both of which to the server side. However, when the control is remote, the communication thread has a higher priority since the control loop is closed through the network.

The main features of the middleware layers are given next.

3.3.1 Implementation of the PAL

This layer has three threads:

- *Receive commands:* Once an UDP connection is open, the loop is not executed until a request arrives. When a new control packet is received, the data is desencapsulated and enqued in a buffer created for this purpose. This thread also sends back a small packet with the time stamp to the

address from which it received the control packet, but to a different port. This allows measuring the RTT.

- *Hardware access:* It is the most important thread, since it reads from sensors, writes to actuators, selects the control input to apply, collects the output vector in output based control, and decides when a new measurement has to be transmitted to the controller. It is executed at the frequency required to sample the plant.

  The most important block is the one that manages the control and state/output sequences. At each sampling time, it checks the queue when the data from the new incoming packets is stored. If a most recent data is available, it discards the sequences that were being processed and decides the first element in each of the sequences that will be used.

- *Send measurements:* It is executed with the same frequency that the hardware access. This thread checks the measurements queue. If new measurements have been enqueued, i.e., if a new event has been triggered, the data is encapsulated in a new *state packet* and transmitted to the controller through the network.

It can be argued that sending back a packet to measure the RTT increases the network traffic. However, this action is similar to the *acknowledgment* that many protocols use but it is not required by UDP. The reason for using UDP, and for instance not TCP, is that UDP reduces latency over reliability, and this is preferable as the experimental results will show. The block diagram of this layer as well as the GUI at the server can be found in Appendix C.

3.3.2 Implementation of the CAL

In order to execute the controller, the application at the server side must be running, otherwise an error is reported. This layer also has three threads:

- *Receive measurements:* This thread opens an UDP connection that remains listening until a new state packet is received. It extracts the data and enqueues it in the measurement queue.

- *Control and send commands:* This is the most critical thread, since it process the measurements and computes the state/output and control sequences. The measurement queue is checked. If it is not empty, it reads the data, estimates the disturbance and the state that the plant will have after τ_{min} sampling periods, and then the computation of the anticipative controller starts. The anticipative controller block, the most important, has three inputs: the aforementioned estimated state and the disturbance, the basis controller, and the model of the plant. And it has two output: the state/output sequence and the set of future control values. Once this computation is finished, it encapsulates the data, and sends a new control packet to the server.

  A screenshot of the block diagram for this thread is shown in Figure 3.6. Two cases are distinguished. When the controller type is "0", this is interpreted as state feedback controller so that the state sequence is computed, whereas the output sequence is generated if the controller type is "1", i.e., a PI controller.

- *Measure RTT:* As indicated in the implementation of the PAL, packets are sent back with the time stamp to measure the quality of the channel (RTT). This loop, with lower priority than the other two, is in charge of extracting this information.

The complete block diagram of the CAL as well as the GUI at the client side can be found in the Appendix C.

3.4 Experimental results

In order to evaluate the performance of the controller design proposed in Chapter 2, the influence of the parameter c of the trigger function is studied, and the step response of the anticipative controller is compared to other frameworks. Moreover, the effectiveness of the disturbance estimation is assessed.

The flexible link described in Section 3.2.1 is used for this evaluation since

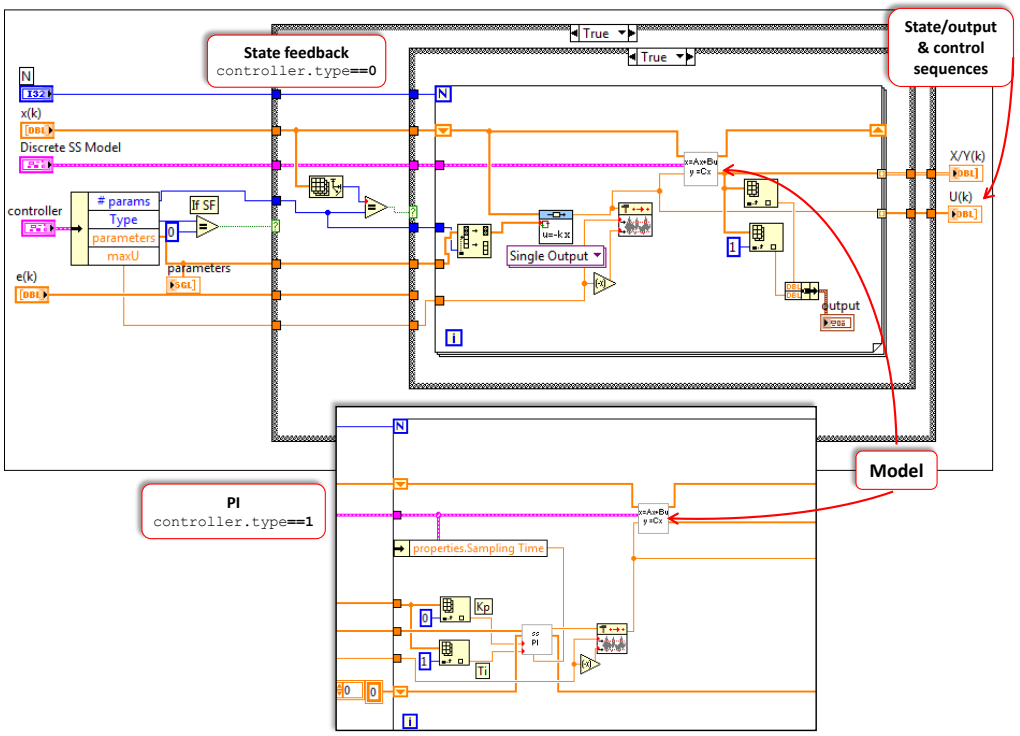

Figure 3.6: Screenshot of the anticipative controller block in LabVIEW.

it is a higher order process than the SRV-02 setup, although the PI anticipative controller is tested over this device. Finally, some properties of the network are analyzed and additional experiments are performed on the SRV-02 gear.

3.4.1 Performance of event-based control

An anticipative controller based on the model (3.3) and the feedback gain (3.4) is designed for the flexible link presented in Figure 3.4. The system is sampled with $h = 10$ ms, and the driver provided by Quanser is used to emulate full state measurement because the sensors only provides measurements of θ and α.

Influence of the trigger function

First, we afford the study of the influence of the parameter c of the trigger function (2.8). The response to an angular position step command is shown in Figure 3.7. Three values are considered: $c = 0.05$ (blue), $c = 0.1$ (red), and $c = 0.2$ (green). The angle of the gear x_1, the arm deflection x_2, the control input, and the events execution are depicted.

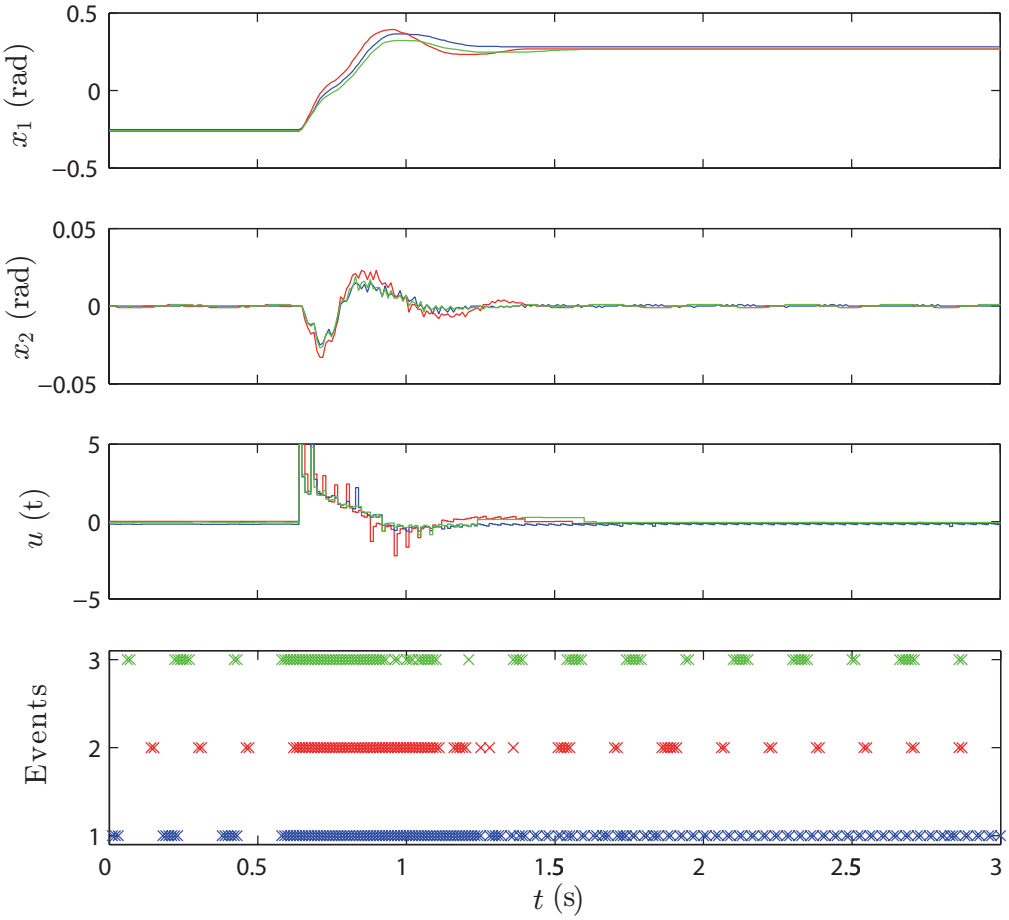

Figure 3.7: Step response for the event based controller with $c = 0.05$ (blue), $c = 0.1$ (red), and $c = 0.2$ (green).

Note that the performance in the three situations is similar, and the main differences are in the number of triggered events. If $c = 0.05$, 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 ($c = 0.1$). 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 $\bar{Q}$ (see Algorithm 2.1).

Performance comparative

Figure 3.8 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).

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 $c = 0.1$.

3. *Remote state feedback controller with anticipative strategy:* The anticipative controller proposed with the state feedback basis controller with sequence length $Q = 20$ (length of control and prediction sequences), and the trigger function with $c = 0.1$.

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 3.1. 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 transmission is obtained.

The Integral Absolute Error (IAE) is computed for the three frameworks and for the first component of the output vector as

$$\text{IAE} = \int_{t_0}^{t_f} |\epsilon(t)| dt,$$

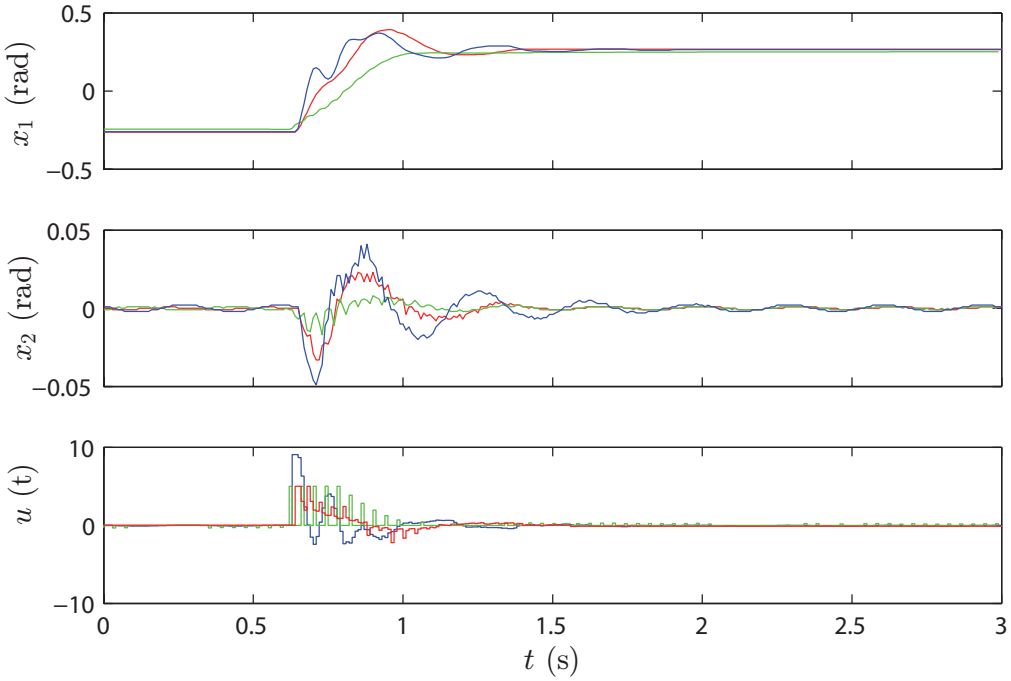

Figure 3.8: Performance comparative of the local controller (blue), the remote controller with $Q = 1$ (green), and the remote controller with $Q = 20$ (red).

where $\epsilon(t) = y_{SP} - y(t)$. The IAE is increased with event-triggered due to the existence of a stationary error that varies with c. Nevertheless, the IAE is reduced with the anticipative strategy (framework 3) respect to the second framework.

3.4.2 Response to disturbances

The disturbance estimator is evaluated if a step disturbance is induced in the input while a step command is given to the angular position. Figure 3.9 shows the system response in two situations: When the disturbance estimator module is included in the CAL, and when it is not. Observe that the system rejects better the disturbance in the first case. Moreover, the number of events accounts for

Table 3.1: Performance parameters of the three frameworks depicted in Figure 3.8.

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

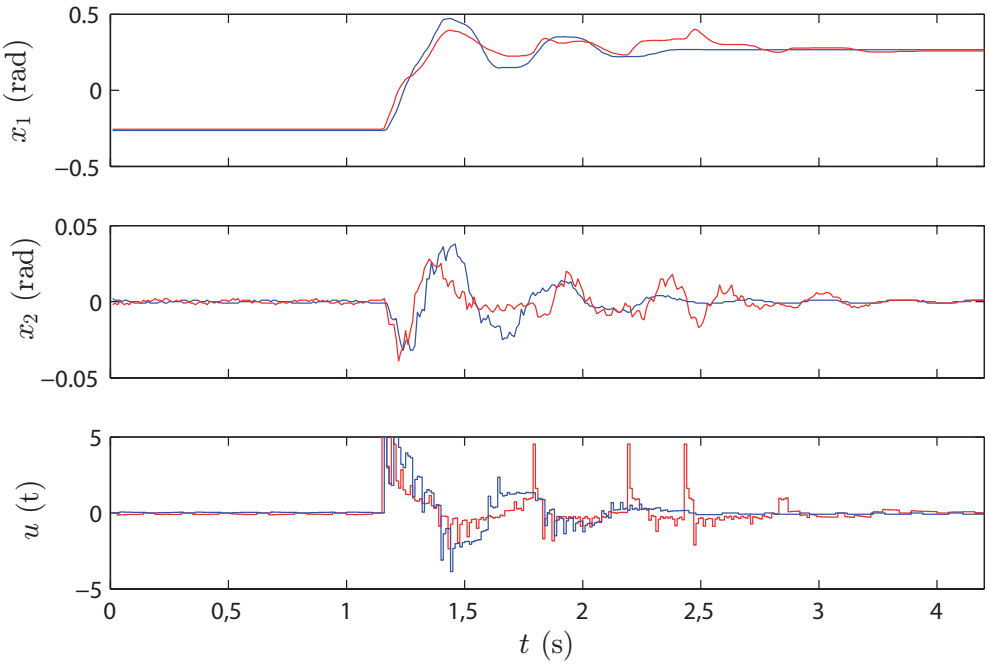

Figure 3.9: Disturbance rejection with (blue) and without (red) disturbance estimation.

50.13% and 59.84% of the sampling times, respectively. Thus, the disturbance estimator not only provides better performance but also a lower transmission rate.

Another example is given in Figure 3.10. 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. Note that the controller exhibits much better behavior when the disturbance is estimated. Such estimation is shown in Figure 3.11. Each plot corresponds to a component of the estimated disturbance vector $\hat{w}(k)$. 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.

The number of events is reduced even more and accounts for almost the half of the events without disturbances estimation. In order to avoid that the noise has influence on the disturbance computation, a threshold is defined so that $\hat{w}(k)$ is set to zero if the estimation is below this threshold. If this strategy is not taken, additional events may be generated.

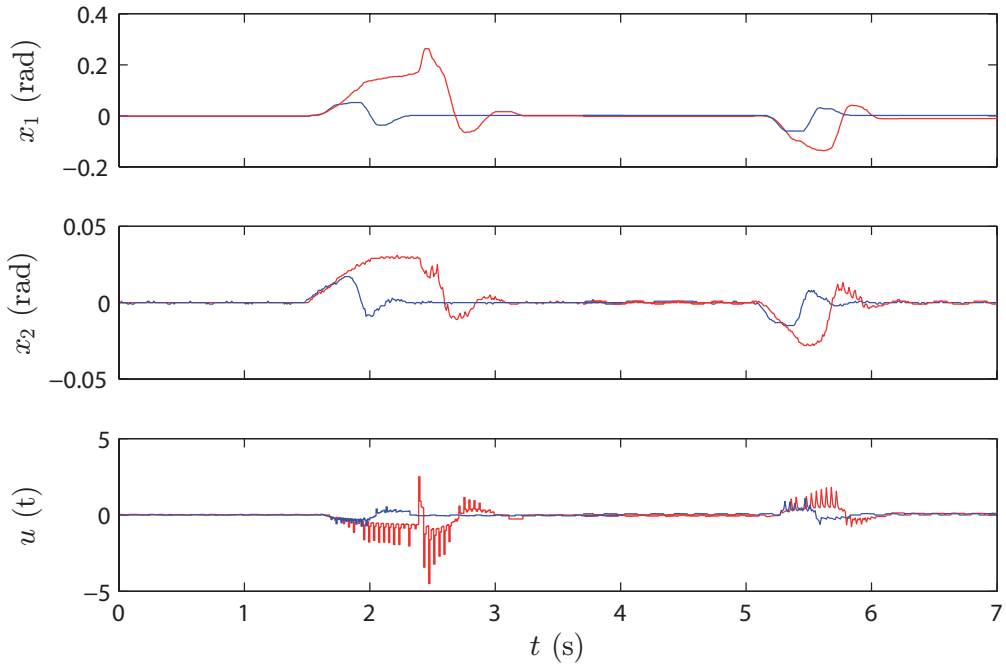

Figure 3.10: Disturbance rejection with (blue) and without (red) disturbance estimation.

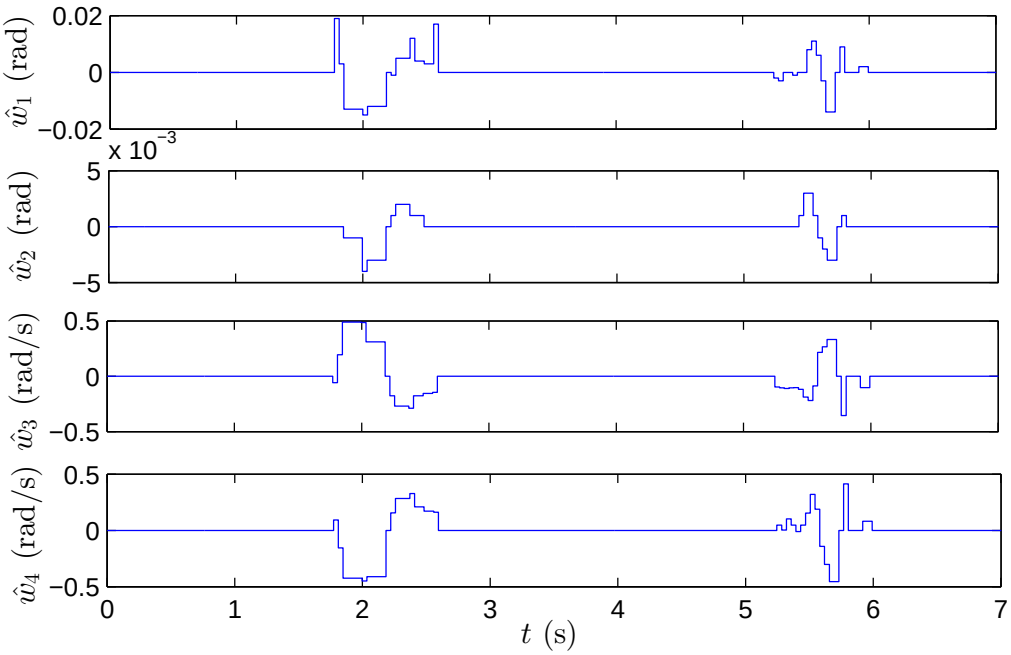

Figure 3.11: Disturbance estimation. Top: $\hat{w}_1$ (blue), $\hat{w}_2$ (cyan). Bottom: $\hat{w}_3$ (blue), $\hat{w}_4$ (cyan).

3.4.3 PI anticipative controller

In order to test the design for LTI anticipative controllers, the PI controller (3.2) is taken as the basis controller. This has been the proposed solution when

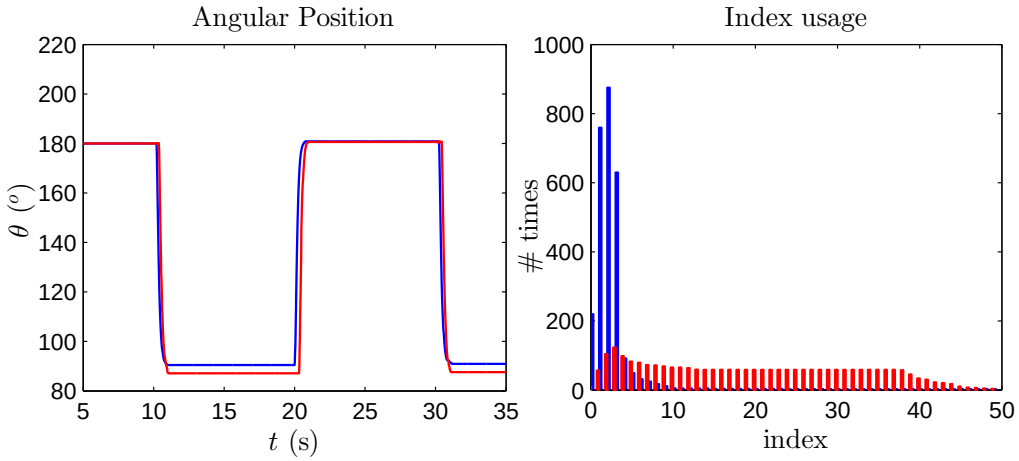

Figure 3.12: Comparison of time-based (blue) and event-based (red) PI anticipative controllers.

only output measurements are available. For the SRV-02 gear, the output is the angular position θ. As remarked in Section 2.9, one of the benefits of measuring the output is that the horizon of the predictions can be enlarged.

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 $N = 50$, but with periodic transmission from the sensor to the controller.

2. Such transmission is event-triggered (2.36) with $c_y = 0.1$ rad, and the rest of the parameters are the same.

The results for a particular experience are shown in Figure 3.12. 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 c_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 c_y.

On the right hand side of Figure 3.12 the parameter denoted *Index usage* is depicted. This parameter represents the number of times that the element *index*, $index = 1, \ldots, N$ of any control sequence $\{\mathbf{U}_k, k \in \mathbb{N}\}$ has been applied by the actuator.

We remind that the first i_0 elements are discarded according to the current

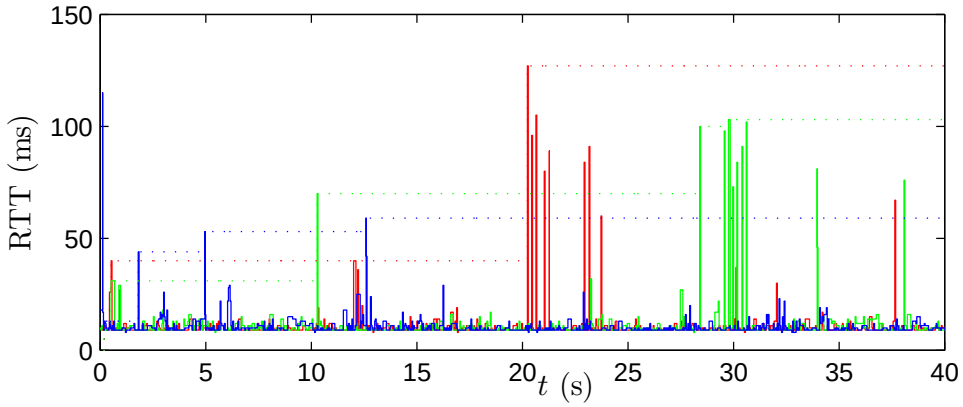

Figure 3.13: Measured RTT in three experiments: 10 AM (red), 3 PM (green), 8 PM (blue).

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 c_y or the index reaches the bound $\bar{Q}$ in event-triggering.

3.4.4 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 Figure 3.13. 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.

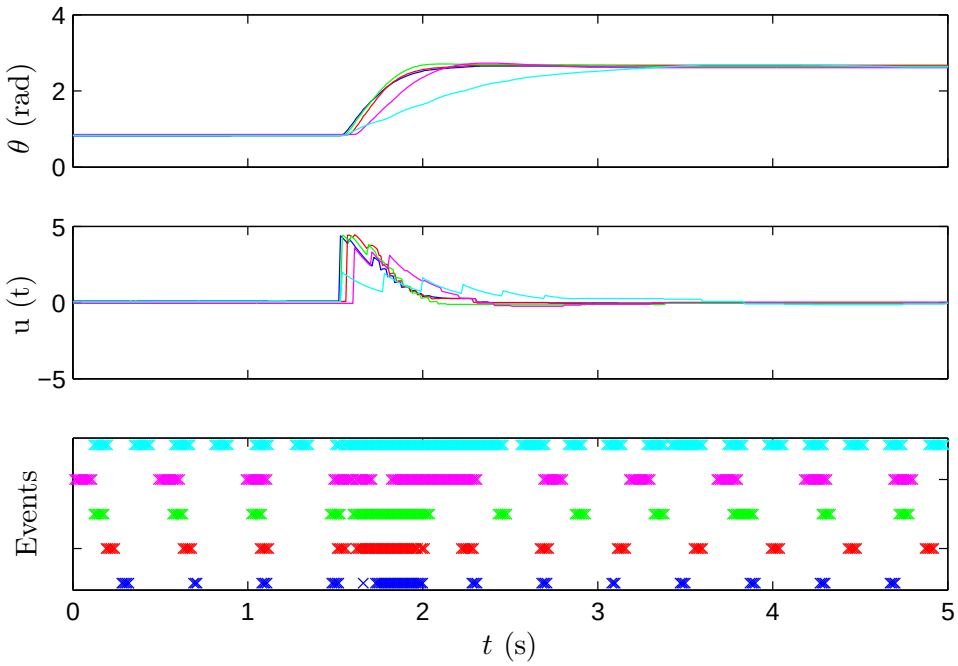

Figure 3.14: Output, control signal and events generation when the anticipative controller runs in the server (blue), the average delay is 20 ms (red), 50 ms (green), 100 ms (magenta), and 230 ms (cyan).

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.

Study of the delay impact

If the theoretical upper bound is computed according to (2.21) for the SRV-02 gear model (3.1), it follows that $\|A_d^\tau - I\| < 1$ 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 ms, 50 ms, 100 ms, and 230 ms has been designed by introducing artificial delays. Figure 3.14 shows the obtained results for the SRV-02 gear for the PI controller (3.2). If the delay increases, the performance of the system degrades, slowing down the response to

Table 3.2: 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

a step change in the reference. However, the controller achieves acceptable results even for RTT values of 230 ms, that is, more than 20 times the plant sampling period. Moreover, the behavior is really closed to the model (see Figure 3.3) for delays of 20 and 50 ms.

Note that the number of events increases with the delay, since the sensor sends an new measurement after the detection of an event if it has note received an updated control packet.

The rise time, settling time, and overshoot have been computed for the five cases described above. The results are summarized in Table 3.2. The settling time increases with the RTT, whereas the rise time is preserved in adequate values except in the last case.

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.

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

Table 3.3: 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

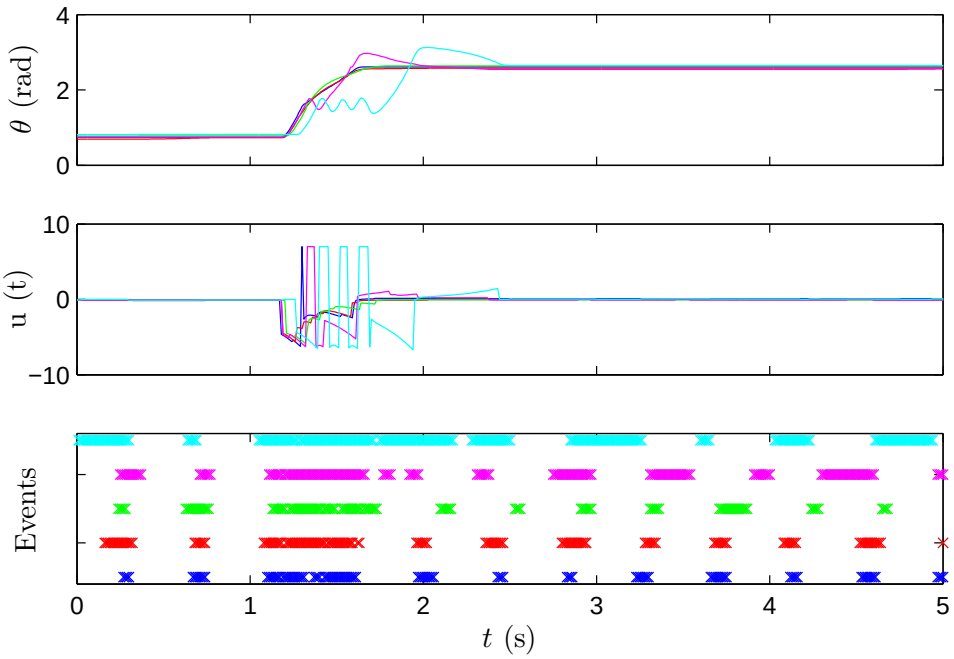

Figure 3.15: Output, control signal and events generation for $p = 0$ (blue), $p = 0.2$ (red), $p = 0.4$ (green), $p = 0.6$ (magenta), and $p = 0.8$ (cyan).

packet is $p = 1 - q$. As an example, Figure 3.15 shows 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 output of the system, the control input and the events triggered for values of $p = 0$, 0.2, 0.4, 0.6, and 0.8, are depicted. The system exhibits good behavior if $p \leq 0.6$ as the rise and settling times are almost the same in this interval (see Table 3.3), 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 (see Table 3.2).

If the generation of events is analyzed, it can be noticed that a high rate of packet losses involve a lower transmission rate that large delays. Thus, we can say that packet losses are preferable to large latency of the network.

3.5 Conclusions

The proposed architecture in Chapter 2 has been implemented and evaluated in a framework in which the remote controller communicates with the process

through the Internet.

The middleware layers CAL and PAL have been implemented in LabVIEW, which provides a simple multi-thread programming framework required for applications in which tasks such as data-acquisition and communication are critical.

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.

Moreover, the designed disturbance estimator has been also evaluated, showing that disturbances can be rejected effectively.

4

Distributed event-based control for interconnected linear systems

Summary

This chapter presents a distributed event-based control (DEBC) strategy for a networked dynamical system consisting of N linear time-invariant interconnected subsystems. Each subsystem broadcasts its state over the network according to certain triggering rules which depend on local information only. The system can converge asymptotically to the equilibrium point under the proposed control design, and the existence of a lower bound for the broadcasting period is guaranteed. The problem is solved assuming that the control law is able to decouple the subsystems and a continuous time system, and the results are extended to non-perfect decoupling and discrete-time systems afterwards.

4.1 Introduction

Power or traffic networks can be seen as the interconnection of subsystems through a network, characterized by a large number of variables and uncertainties. The centralized control of such large-scale systems in a networked environment would require a very accurate knowledge of the interaction between these subsystems and the consumption of a lot of computation and network resources. Hence, there is a natural interest in applying event-triggering to decentralized NCS.

There are some recent contributions on distributed event-triggered control [DFJ12, DPSW11, GDJ+11, MT11, SDJ13, WL11]. The basic idea in all these contributions is that each subsystem (also called agent or node) 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.

A distributed event-triggered control has been proposed in [DFJ12, SDJ13] restricted to multi-agent systems and average consensus problems. In [MT11] self-triggered policies are proposed to avoid the constant checking of the trigger condition. However, the control system is less robust against disturbances under these policies since these cannot be detected in the inter-event times.

In [WL11] a decentralized control[1] for large-scale systems is proposed under the assumption of week coupling. The design of the event triggering threshold is based on Lyapunov methods and it ensures input-to-state stability of nonlinear systems. However, a positive lower bound for the broadcasting period, i.e., the minimum difference between successive broadcasting times, may not be achievable when the system approaches the origin. This might cause severe problems since it would require the detection of events and transmission of data infinitely fast due to possible Zeno behaviors. In a previous work of the same authors [WL08], the design is restricted to linear systems with perfect decoupling.

A distributed event-triggered control has also been examined in [DPSW11], in which the gains measuring the degree of interconnection satisfy a generalized small-gain condition. This design does not prevent from Zeno behavior, and a constant threshold-like condition is proposed to overcome this issue.

This chapter presents a DEBC for interconnected linear systems, which can be represented as different nodes which communicates through a shared network, as in Figure 4.1. Solid lines represent the coupling between nodes. Neighboring relationships are defined in the sense of dynamical interactions between subsystems. Measurements are transmitted through the network to achieve the equilibria of

[1]Decentralized control neglects the interaction between the subsystems and designs a local controller for each subsystem, whereas the local regulators exchange information among them in distributed control

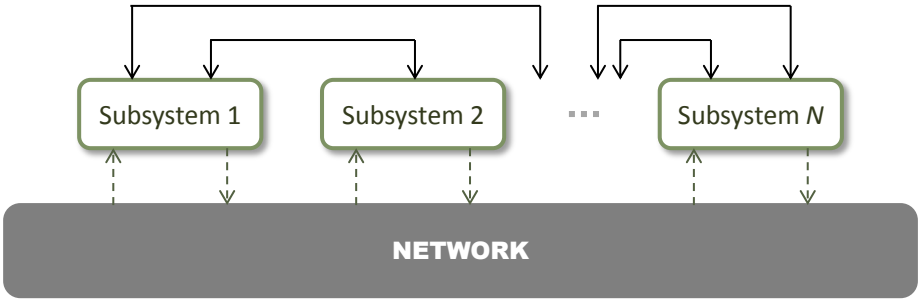

Figure 4.1: Networked interconnected system.

the system. The outline of this chapter is as follows. Section 4.2 summarizes the contributions of this chapter. Some background on matrix analysis and the problem statement are given in Section 4.3. The proposed event-triggered control is presented in Section 4.4. Results for perfect decoupled systems are given in Section 4.5, and Section 4.6 widens these results to a more general framework. An extension to discrete-time systems is given in Section 4.7. Finally, conclusions end the chapter.

4.2 Contributions of this chapter

One of the issues mentioned in the introduction is the difficulty to achieve asymptotic convergence to the equilibria while excluding the Zeno behavior. We show that this can be accomplished by the proposed trigger mechanism with time-dependent trigger functions. If the parameters of these trigger functions are adequately selected, the system presents asymptotic stability while guaranteeing a lower bound for the minimum inter-event time, which can be analytically derived. With regards to [WL11, WL08, GA12], the triggering mechanism does not continuously depend on the state of the system but on the error between the current and the latest broadcasted state, which results in that the number of generated events decreases when the system is close to the equilibrium point.

The problem is initially solved for perfect decoupled systems, and then the results are extended for non-perfect decoupling, since that constraint is difficult to achieve in practice. Moreover, the interconnection terms are not required to by symmetric in contrast to [WL08], [GA12].

The coupling terms are treated as a perturbation of the nominal system, and the existing classical analysis on the sensitivity of the matrix exponential [VL77] and matrix powers [AMH09] is applied to infer constraints on the coupling terms so the asymptotic stability property is preserved.

4.3 Background and problem statement

Some classical results from matrix analysis, which are used for obtaining the analytical results of this chapter, are presented first. The problem statement is given afterwards.

4.3.1 Matrix and perturbations analysis

Let $A \in \mathbb{C}^{n\times n}$ be a complex matrix, and let us denote

$$A^* = (\bar{a}_{ji}), \tag{4.1}$$

$$\lambda(A) = \{\lambda : \det(A - \lambda I) = 0\}, \tag{4.2}$$

$$\kappa(A) = \|A\|\|A^{-1}\| \quad (0 \notin \lambda(A)), \tag{4.3}$$

$$\lambda_{max}(A) = \max\{\Re e(\lambda) : \lambda \in \lambda(A)\}, \tag{4.4}$$

where $\|\cdot\|$ denotes the induced 2-norm.

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 $\|e^{At}\|$.

Bounding the matrix exponential

In [VL77] 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

$$\|e^{At}\| = \|Ye^{Bt}Y^{-1}\| \leq \kappa(Y)e^{\mu_{max}(B)t}, \tag{4.5}$$

where $\kappa(Y)$ is defined according to (4.3).

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 (4.5), it holds that

$$\|e^{At}\| \leq \kappa(V)e^{\mu_{max}(D)t} = \kappa(V)e^{\lambda_{max}(D)t} = \kappa(V)e^{\lambda_{max}(A)t}, \tag{4.6}$$

where $\lambda_{max}(A)$ is defined according to (4.4).

- *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 [VL77]

$$\|e^{At}\| \leq m \cdot \kappa(X)e^{\lambda_{max}(A)t} \max_{0\leq r\leq m-1} \frac{t^r}{r!}. \tag{4.7}$$

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

$$Q^* A Q = D + N, \tag{4.8}$$

where D is the diagonal matrix $D = \text{diag}(\lambda_i)$ and N is strictly upper triangular. The following upper bound can be obtained [VL77]

$$\|e^{At}\| \leq e^{\lambda_{max}(A)t} \sum_{k=0}^{n-1} \frac{\|Nt\|^2}{k!}. \tag{4.9}$$

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 theorem studies the perturbation of the eigenvalues of a matrix A when A is diagonalizable:

THEOREM [BF60]. *If A is diagonalizable ($V^{-1}AV = D$), the eigenvalues $\tilde{\lambda}_i$ of $A + E$ satisfy*

$$\min_{\lambda_j \in \lambda(A)} |\tilde{\lambda}_i - \lambda_j| \leq \kappa(V)\|E\|. \tag{4.10}$$

The previous result has been extended to defective, i.e., non diagonalizable, matrices in [Chu86]:

THEOREM [CHU86]. *Let consider the Schur decomposition* (4.8). *Then for $\tilde{\lambda}_i \in \lambda(A + E)$*

$$\min_{\lambda_j \in \lambda(A)} |\tilde{\lambda}_i - \lambda_j| \leq max\{\theta_1, \theta_1^{1/n}\}, \tag{4.11}$$

where $\theta_1 = \|E\| \sum_{k=0}^{n-1} \|N\|^k$.

Finally, a result from semigroup theory (see [Kat66]) states that if $\|e^{At}\| \leq ce^{\beta t}$ for some constants c and β, then

$$\|e^{(A+E)t}\| \leq ce^{(\beta + c\|E\|)t}. \tag{4.12}$$

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 [Hig08]. 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

$$f(A+E) - F(A) - L_f(A,E) = \mathcal{O}(\|E\|^2),$$

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 [AMH09]. *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

$$(A+E)^p = A^p + \sum_{j=0}^{p-1} A^{p-1-j} E A^j + \mathcal{O}(\|E\|^2).$$

Then, it is a logical consequence the following

$$\|(A+E)^p\| \leq \|A^p\| + \|\sum_{j=0}^{p-1} A^{p-1-j} E A^j\| + \mathcal{O}(\|E\|^2). \tag{4.13}$$

4.3.2 Problem statement

Consider a system of N linear time-invariant subsystems. The dynamics of each subsystem is given by

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + \sum_{j\in N_i} H_{ij} x_j(t), \quad \forall i = 1, ..., N \tag{4.14}$$

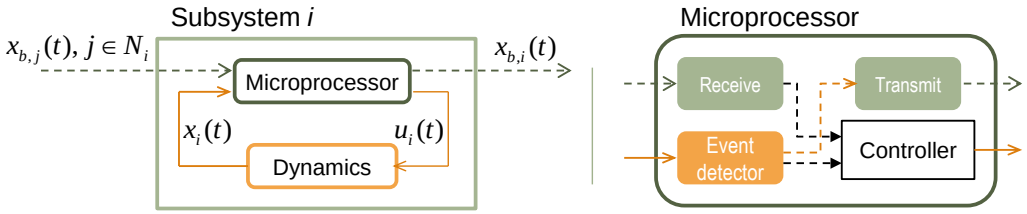

Figure 4.2: Scheme of a node, consisting of a digital microprocessor and dynamics (left), and block diagram of the tasks carried out by the microprocessor.

where N_i is the set of "neighbors" of the subsystem i, i.e., 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, computing the control signal and the communication tasks (see Figure 4.2).

Each agent i sends its state through the network at discrete time instances. Specifically, the agent i can only communicate with the set of agents on its neighborhood 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 broadcasted state is denoted by $x_{b,i}$. The broadcasted 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

$$u_i(t) = K_i x_{b,i}(t) + \sum_{j \in N_i} L_{ij} x_{b,j}(t), \quad \forall i = 1, ..., N \tag{4.15}$$

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 $e_i(t)$ between the state and the latest broadcasted state as

$$e_i(t) = x_{b,i}(t) - x_i(t) = x_i(t_k^i) - x_i(t), \quad t \in [t_k^i, t_{k+1}^i). \tag{4.16}$$

Rewriting (4.14) in terms of $e_i(t)$ and the control law (4.15), we obtain

$$\dot{x}_i(t) = A_{K,i}x_i(t) + B_iK_ie_i(t) + \sum_{j\in N_i}(\Delta_{ij}x_j(t) + B_iL_{ij}e_j(t)), \tag{4.17}$$

where $A_{K,i} = A_i + B_iK_i$, and $\Delta_{ij} = B_iL_{ij} + H_{ij}$ 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

$$A_K = \text{diag}(A_{K,1}, A_{K,2}, ..., A_{K,N}) \tag{4.18}$$

$$B = \text{diag}(B_1, B_2, ..., B_N) \tag{4.19}$$

$$K = \begin{pmatrix} K_1 & L_{12} & \cdots & L_{1N} \\ L_{21} & K_2 & \cdots & L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & K_N \end{pmatrix} \tag{4.20}$$

$$\Delta = \begin{pmatrix} 0 & \Delta_{12} & \cdots & \Delta_{1N} \\ \Delta_{21} & 0 & \cdots & \Delta_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{N1} & \Delta_{N2} & \cdots & 0 \end{pmatrix} \tag{4.21}$$

and the stack vectors

$$x = (x_1^T, x_2^T, ..., x_N^T)^T \tag{4.22}$$

$$e = (e_1^T, e_2^T, ..., e_N^T)^T \tag{4.23}$$

as the state and error vectors of the overall system. Note that $H_{ij}, L_{ij}, \Delta_{ij} := 0$ if $j \notin N_i$. Let also be $n = \sum_{i=1}^{N} n_i$ the state and error dimension.

The dynamics of the overall system is given by

$$\dot{x}(t) = (A_K + \Delta)x(t) + BKe(t). \tag{4.24}$$

As the broadcasted states $x_{b,i}$ remain constant between consecutive events, the error dynamics in each interval is given by

$$\dot{e}(t) = -(A_K + \Delta)x(t) - BKe(t). \tag{4.25}$$

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, (4.24) and (4.25) do not generally hold. The extension to non-reliable communications is given in Chapter 5.

4.4 Event-based control strategy

The occurrence of an event, i.e., a broadcast over the network and a control law update, is defined by the trigger functions f_i which depend on local information of agent i only and take values in $\mathbb{R}$. The sequence of broadcasting times t_k^i is determined recursively by the event trigger function as $t_{k+1}^i = \inf\{t : t > t_k^i, f_i(t) > 0\}$. Particularly, we consider trigger functions of the form

$$f_i(t, e_i(t)) = \|e_i(t)\| - (c_0 + c_1 e^{-\alpha t}), \alpha > 0 \tag{4.26}$$

where $c_0 \geq 0, c_1 \geq 0$ but both parameters cannot be zero simultaneously, and the error is defined in (4.16).

The motivations of these trigger functions (4.26) are the following. On one hand, static trigger functions ($c_1 = 0$) have been vastly studied in the literature, see e. g. [HSvdB08, LL10]. In that case, the error is bounded by $\|e_i(t)\| \leq c_0$ $\forall t$ and c_0 determines the ultimate set in which the state of the plant is confined around the equilibrium. Large values of c_0 allow reducing the number of events but degrades the performance. On the contrary, small values of c_0 give better performance but the average inter-event time, defined as $T_k^i = t_{k+1}^i - t_k^i$ decreases considerably.

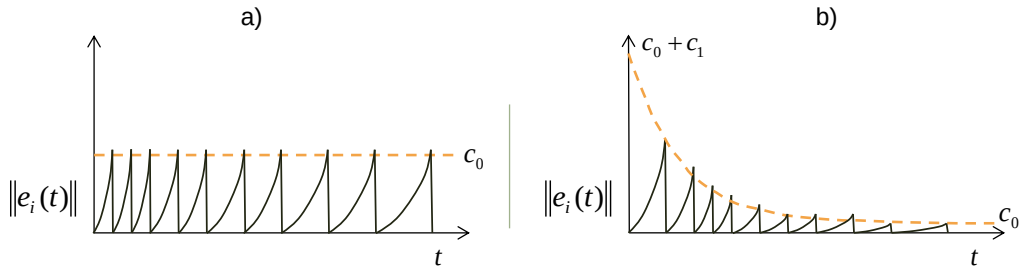

Figure 4.3: a) Static trigger functions, b) Proposed trigger functions.

On the other hand, event-triggering rules derived using Lyapunov analysis are usually of the form $\|e_i(t)\| \leq \sigma_i \|x_i(t)\|$. 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 ([WL11, GA12]). In contrast, we will prove that trigger functions (4.26) can give good performance while decreasing the number of events and guaranteeing a minimum inter-event time even if $c_0 = 0$, if the parameters are adequately selected.

Throughout this chapter we will refer to the Zeno behavior. This phenomenon must be avoided and the design has to guarantee the existence of a lower bound for the inter-event time, since it might cause severe problems because it would require the detection of events and transmission of data infinitely fast.

Example 4.1: A static trigger function is depicted on the left hand side of Figure 4.3. The error is bounded by the constant threshold c_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.
On the right side, trigger functions of the form (4.26) are represented. Note that the threshold decreases with time and the error is bounded by $c_0 + c_1$ at $t = 0$ and by c_0 when $t \to \infty$. If $c_0 = 0$, this bound goes to zero when time increases and asymptotic stability can be achieved.

The stability properties of the proposed event-based control are derived next. First, the results for perfect decoupling are presented, i.e., when $\Delta = \mathbf{0}$ holds. The extension to the *perturbed* case is given afterwards.

Though perfect decoupling is difficult to achieve in practice, this case is analyzed because the analytical results are simpler, and then the effect of the coupling terms can be examined from them.

4.5 Results for perfect decoupling

If the *matching condition* holds, i.e., $\Delta_{ij} + B_i L_{ij} = 0$, (4.17) is transformed into

$$\dot{x}_i(t) = A_{K,i} x_i(t) + B_i K_i e_i(t) + \sum_{j \in N_i} B_i L_{ij} e_j(t), \tag{4.27}$$

and the dynamics of the overall system is given by

$$\dot{x}(t) = A_K x(t) + BKe(t). \tag{4.28}$$

Assumption 4.1. We assume that $A_{K,i}$, $i = 1, \ldots, N$ is diagonalizable so that there exists a matrix $D_i = \mathrm{diag}(\lambda_k(A_{K,i}))$ and an invertible matrix of eigenvectors V_i such that $A_{K,i} = V_i D_i V_i^{-1}$.

This assumption facilitates the calculations since (4.5) can be applied, but the extension to general Jordan blocks is achievable as discussed later in the section.

The following theorem states that the system (4.28) with trigger functions defined as in (4.26) converges to a specified region around the equilibrium point which, without loss of generality, is assumed to be $(0, \ldots, 0)^T$. Moreover, if $c_0 = 0$ the convergence is asymptotical to the origin. The functions (4.26) bound the errors $\|e_i(t)\| \leq c_0 + c_1 e^{-\alpha t}$, since an event is triggered as soon as the norm of $e_i(t)$ crosses the threshold $c_0 + c_1 e^{-\alpha t}$.

Theorem 4.1. *Consider the closed-loop system* (4.28) *and trigger functions of the form* (4.26)*, with* $0 < \alpha < |\lambda_{max}(A_K)|$*. Then, for all initial conditions* $x(0) \in \mathbb{R}^n$*, and* $t > 0$*, it holds*

$$\begin{aligned} \|x(t)\| \leq & \kappa(V)\Big(\frac{\|BK\|\sqrt{N}c_0}{|\lambda_{max}(A_K)|} + e^{-|\lambda_{max}(A_K)|t}(\|x(0)\| - \\ & \|BK\|\sqrt{N}(\frac{c_0}{|\lambda_{max}(A_K)|} + \frac{c_1}{|\lambda_{max}(A_K)| - \alpha})) + \frac{e^{-\alpha t}\|BK\|\sqrt{N}c_1}{|\lambda_{max}(A_K)| - \alpha}\Big), \end{aligned} \tag{4.29}$$

where $\lambda_{max}(A_K)$ *and* $\kappa(V)$ *are defined according to* (4.4) *and* (4.3)*, respectively, being* V *the matrix of the eigenvectors of* A_K*.*

Furthermore, the closed-loop system does not exhibit Zeno behavior.

Proof. The analytical solution of (4.28) is

$$x(t) = e^{A_K t}x(0) + \int_0^t e^{A_K(t-s)}BKe(s)ds. \tag{4.30}$$

From Assumption 4.1 the matrix A_K is diagonalizable by construction, because each diagonal block A_{K_i} is. Then it follows that $e^{A_K} = Ve^DV^{-1}$, with $V = \text{diag}(V_i)$ is a block diagonal matrix too. According to (4.5), $\|e^{A_K t}\|$ can be bounded by $\kappa(V)e^{\lambda_{max}(A_K)t}$. Because A_K is Hurwitz, $\lambda_{max}(A_K) < 0$. Thus,

$$\|e^{A_K t}\| \leq \kappa(V)e^{-|\lambda_{max}(A_K)|t}.$$

Consequently, the state can be bounded by

$$\|x(t)\| \leq \kappa(V)(e^{-|\lambda_{max}(A_K)|t}\|x(0)\| + \int_0^t e^{-|\lambda_{max}(A_K)|(t-s)}\|BK\|\|e(s)\|ds).$$

The trigger condition $f_i(t, e_i(t)) > 0$ enforces $\|e_i(t)\| \leq c_0 + c_1e^{-\alpha t}$ so that

$$\|e(s)\| \leq \sqrt{N}(c_0 + c_1e^{-\alpha s}), \quad \forall s \in [0, t].$$

It follows that

$$\begin{aligned}\|x(t)\| \leq &\kappa(V)(e^{-|\lambda_{max}(A_K)|t}\|x(0)\| \\ &+ \int_0^t \sqrt{N}e^{-|\lambda_{max}(A_K)|(t-s)}\|BK\|(c_0 + c_1e^{-\alpha s})).\end{aligned}$$

If the integral is solved:

$$\begin{aligned}\|x(t)\| \leq &\kappa(V)(e^{-|\lambda_{max}(A_K)|t}\|x(0)\| + \frac{\|BK\|\sqrt{N}c_0}{|\lambda_{max}(A_K)|}(1 - e^{-|\lambda_{max}(A_K)|t}) \\ &+ \frac{\|BK\|\sqrt{N}c_1}{|\lambda_{max}(A_K)| - \alpha}(e^{-\alpha t} - e^{-|\lambda_{max}(A_K)|t})),\end{aligned}$$

which by reordering terms yield (4.29), proving the first part of the theorem.

Note that (4.29) can be upper bounded by

$$\|x(t)\| \leq \kappa(V)(\|x(0)\|e^{-|\lambda_{max}(A_K)|t} + \tfrac{\|BK\|\sqrt{N}c_0}{|\lambda_{max}(A_K)|} + \tfrac{\|BK\|\sqrt{N}c_1}{|\lambda_{max}(A_K)|-\alpha}e^{-\alpha t}), \quad (4.31)$$

by omitting the negative terms.

We next show that the broadcasting period is lower bounded. Let us first assume that $c_0, c_1 \neq 0$. If t^* refers to the last event time occurrence, $\|e_i(t^*)\| = 0$, and $f_i(t^*) = -c_0 - c_1 e^{-\alpha t^*} < 0$. Therefore agent i cannot trigger at the same time instant. From (4.16) it falls out that between two consecutive events we have $\dot{e}_i(t) = -\dot{x}_i(t)$, thus

$$\|e_i(t)\| \leq \int_{t^*}^{t} \|\dot{x}_i(s)\| ds.$$

Furthermore, from (4.28) it can be derived

$$\|\dot{x}(t)\| \leq \|A_K\|\|x(t)\| + \|BK\|\|e(t)\| \leq \|A_K\|\|x(t)\| + \|BK\|\sqrt{N}(c_0 + c_1 e^{-\alpha t^*}).$$

As from definition (4.22) we have $\|\dot{x}_i(t)\| \leq \|\dot{x}(t)\|$, and if the last event occurred at time $t^* > 0$

$$\|e_i(t)\| \leq \int_{t^*}^{t} \|\dot{x}(s)\| ds \leq \int_{t^*}^{t} (\|A_K\|\|x(s)\| + \|BK\|e(s)) ds,$$

and $\|x(t)\| \leq \|x(t^*)\|$ holds in (4.31). Thus, defining the following constants

$$k_1 = \kappa(V)\|A_K\|\|x(0)\| \quad (4.32)$$

$$k_2 = \|BK\|\sqrt{N}c_1(\frac{\kappa(V)\|A_K\|}{|\lambda_{max}(A_K)| - \alpha} + 1) \quad (4.33)$$

$$k_3 = \|BK\|\sqrt{N}c_0(\frac{\kappa(V)\|A_K\|}{|\lambda_{max}(A_K)|} + 1), \quad (4.34)$$

the error can be bounded as

$$\|e_i(t)\| \leq \int_{t^*}^{t} \|\dot{x}(s)\| ds \leq \int_{t^*}^{t} (k_1 e^{-|\lambda_{max}(A_K)|s} + k_2 e^{-\alpha s} + k_3) ds.$$

Because $e^{-|\lambda_{max}(A_K)|s} \leq e^{-|\lambda_{max}(A_K)|t^*}$ and $e^{-\alpha s} \leq e^{-\alpha t^*}$, $\forall s \geq t^*$, it holds that

$$\begin{aligned}\|e_i(t)\| &\leq \int_{t^*}^{t} (k_1 e^{-|\lambda_{max}(A_K)|t^*} + k_2 e^{-\alpha t^*} + k_3) ds \\ &= (k_1 e^{-|\lambda_{max}(A_K)|t^*} + k_2 e^{-\alpha t^*} + k_3)(t - t^*) \leq (k_1 + k_2 + k_3)(t - t^*).\end{aligned} \tag{4.35}$$

The next event will not be triggered before $\|e_i(t)\| = c_0 + c_1 e^{-\alpha t} \geq c_0$. Thus a lower bound on the inter-events time is given by

$$T_{min} = \frac{c_0}{k_1 + k_2 + k_3}, \tag{4.36}$$

which is a positive quantity. Hence, the inter-event time is lower bounded, and the Zeno behavior is excluded. □

Remark 4.1. Note that the integrability of $e(t)$ in (4.30) is justified by the definition of the event-triggered functions $f_i(e_i(t))$, which by continuity guarantee that $e_i(t)$ cannot be updated to zero immediately after it had done so. Thus there is an arbitrarily small, yet positive lower bound on the interexecution times. Thus the right hand side of the ODE that described the closed loop system is piecewise continuous. Note that the specific lower bound on the interexecution times is established in the final part of the proof.

Remark 4.2. Theorem 4.1 establishes a bound for the overall system state $x(t)$. In the perfect decoupling case, it can be proved, following the same procedure than in the proof of Theorem 4.1, that the state of each agent i is bounded by

$$\begin{aligned}\|x_i(t)\| \leq &\kappa(V_i)\Big(\frac{\mu_i c_0}{|\lambda_{max}(A_{K,i})|} + e^{-|\lambda_{max}(A_{K,i})|t}\big(\|x_i(0)\| - \\ &\mu_i\big(\frac{c_0}{|\lambda_{max}(A_{K,i})|} + \frac{c_1}{|\lambda_{max}(A_{K,i})| - \alpha}\big)\big) + \frac{e^{-\alpha t}\mu_i c_1}{|\lambda_{max}(A_{K,i})| - \alpha}\Big),\end{aligned}$$

where $\mu_i = \|B_i K_i\| + \sum_{j \in N_i} \|B_i L_{ij}\|$. This expression can be derived due to the fact that L_{ij} perfectly decouples the system and the bound on the errors is $\|e_j(t)\| \leq c_0 + c_1 e^{-\alpha t}$.

However, the bound (4.29) allows to derive the extension to non-perfect de-

coupling easily.

Remark 4.3. If Assumption 4.1 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 (4.7) holds, or the Schur decomposition bound (4.9). In both cases the bound is governed by the exponential of $\lambda_{max}(A_K)$, which is negative. Thus, the stability of the system is guaranteed though the speed of convergence to the equilibria decreases.

We next analyze two particular cases: static trigger functions, i.e., $c_1 = 0$, and pure exponential trigger functions, i.e., $c_0 = 0$.

4.5.1 Static trigger functions

If $c_1 = 0$ in (4.26), the error is bounded by $\|e_i(t)\| \leq c_0$. The analytical expressions of Theorem 4.1 can be adapted to this case, and the state is bounded by

$$\|x(t)\| \leq \kappa(V)\Big(\frac{\|BK\|\sqrt{N}c_0}{|\lambda_{max}(A_K)|} + e^{-|\lambda_{max}(A_K)|t}\big(\|x(0)\| - \|BK\|\sqrt{N}\frac{c_0}{|\lambda_{max}(A_K)|}\big)\Big).$$

Also, the lower bound for the inter-event time (4.36) becomes $T_{min} = \frac{c_0}{k_1+k_3}$, because $k_2 = 0$ if $c_1 = 0$.

4.5.2 Pure exponential trigger functions

In the case $c_0 = 0$ the error is bounded by $\|e_i(t)\| \leq c_1 e^{-\alpha t}$, and so the error goes to zero when times goes to infinity. The state bound (4.31) can be particularized for pure exponential trigger functions as follows

$$\|x(t)\| \leq \kappa(V)\big(\|x(0)\|e^{-|\lambda_{max}(A_K)|t} + \tfrac{\|BK\|\sqrt{N}c_1}{|\lambda_{max}(A_K)|-\alpha}e^{-\alpha t}\big). \tag{4.37}$$

In order to prove that the Zeno behavior is excluded, we consider the bound on $\|e_i(t)\|$ defined in (4.35) before the last inequality, i.e.

$$\|e_i(t)\| \leq (k_1 e^{-|\lambda_{max}(A_K)|t^*} + k_2 e^{-\alpha t^*})T,$$

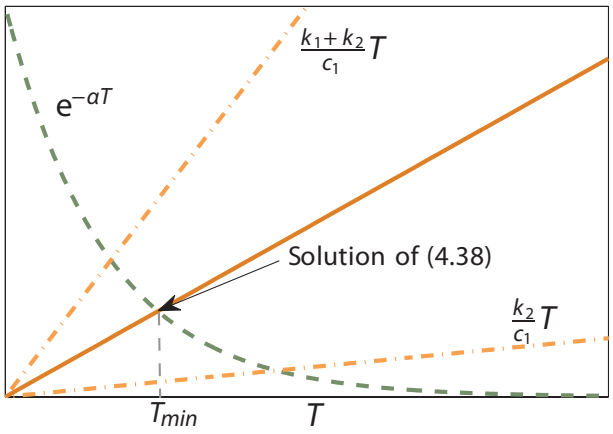

Figure 4.4: Graphical solution of (4.38).

where $T = t - t^*$ and k_1, k_2 are defined in (4.32)-(4.33). Note that $k_3 = 0$ since $c_0 = 0$.

The next event is not triggered before $\|e_i(t)\| = c_1 e^{-\alpha t}$. Thus, a lower bound on the inter-event intervals is given by

$$\left(\frac{k_1}{c_1} e^{(\alpha - |\lambda_{max}(A_K)|)t^*} + \frac{k_2}{c_1}\right) T = e^{-\alpha T}. \tag{4.38}$$

The right hand side of (4.38) is always positive. Moreover, for $\alpha < |\lambda_{max}(A_K)|$ the left hand side is strictly positive as well, and the term in brackets is upper bounded by $\frac{k_2+k_1}{c_1}$ and lower bounded by k_2/c_1, and this yields to a positive value of T for all $t^* \geq 0$. The existence of the solution T_{min} can also be depicted graphically (see Figure 4.4). The solution is given by the intersection of the exponential curve $e^{-\alpha T}$ and the straight line between the two bounds whose slope depends on t^*. Thus, there is no Zeno behavior.

4.5.3 Simulation results

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 Figure 4.5. This setup will

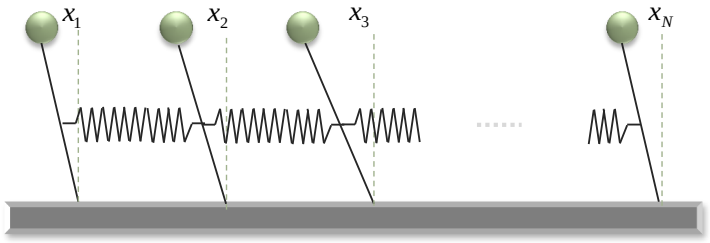

Figure 4.5: Scheme of the network of the inverted pendulums.

be used throughout this and the next chapter.

The problem of coupled oscillators has numerous applications in fields as medicine, physics or communications [Ste07, DGA08], 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 broadcasted to its neighbors in the chain at discrete times given by the communication strategy.

Each subsystem can be described as follows:

$$\dot{x}_i(t) = \begin{pmatrix} 0 & 1 \\ \frac{g}{l} - \frac{a_i k}{ml^2} & 0 \end{pmatrix} x_i(t) + \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix} u_i + \sum_{j \in N_i} \begin{pmatrix} 0 & 0 \\ \frac{h_{ij} k}{ml^2} & 0 \end{pmatrix} x_j(t)$$

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 $h_{ij} = 1, \forall j \in N_i$ 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:

$$u_i(t) = \begin{pmatrix} -3ml^2 & a_i k - \frac{ml^2}{4}(8 + \frac{4g}{l}) \end{pmatrix} x_{b,i}(t) + \sum_{j \in N_i} \begin{pmatrix} -k & 0 \end{pmatrix} x_{b,j}(t)$$

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

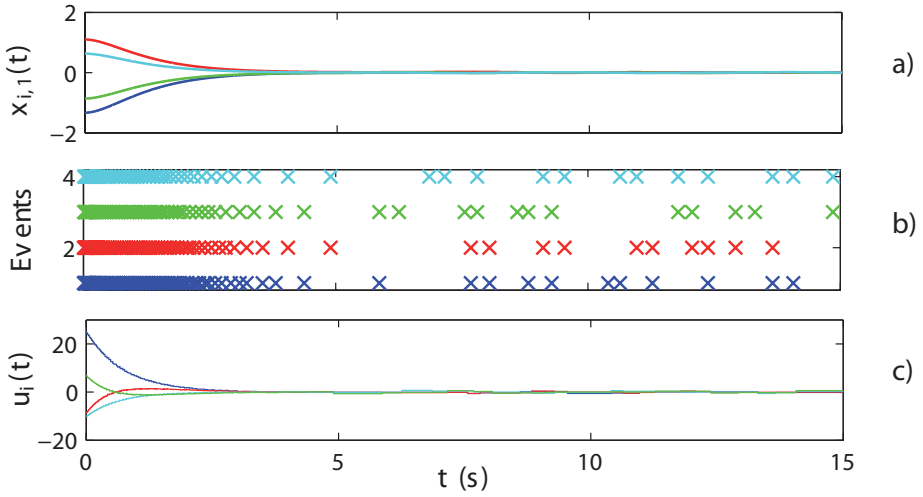

Figure 4.6: Simulation results with trigger function (4.26) with $c_0 = 0.02$, $c_1 = 0$.

set to $g = 10, m = 1, l = 2,$ and $k = 5$.

We next provide several simulation results in order to enhance the theoretical results presented previously in the section and to illustrate the advantages of trigger functions (4.26) respect to constant threshold triggering. Furthermore, we also compare some of these results with the ones obtained in [WL08].

Static trigger functions

The output of the system and the sequence of events for $N = 4$ with trigger functions (4.26) with $c_0 = 0.02$, $c_1 = 0$ is shown in Figure 4.6 for initial conditions $x(0) = (-1.3352 \quad 0 \quad 1.0996 \quad 0 \quad -0.8639 \quad 0 \quad 0.6283 \quad 0)^T$. The output, the time instances at which events are generated in each subsystem, and the applied control signal are depicted. Note that the inter-event times are very small when the system is far from the equilibria.

Time-dependent trigger functions

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 (4.26) with parameters $c_0 = 0.02, c_1 = 0.5,$ and $\alpha = 0.8$ is shown in Figure 4.7.

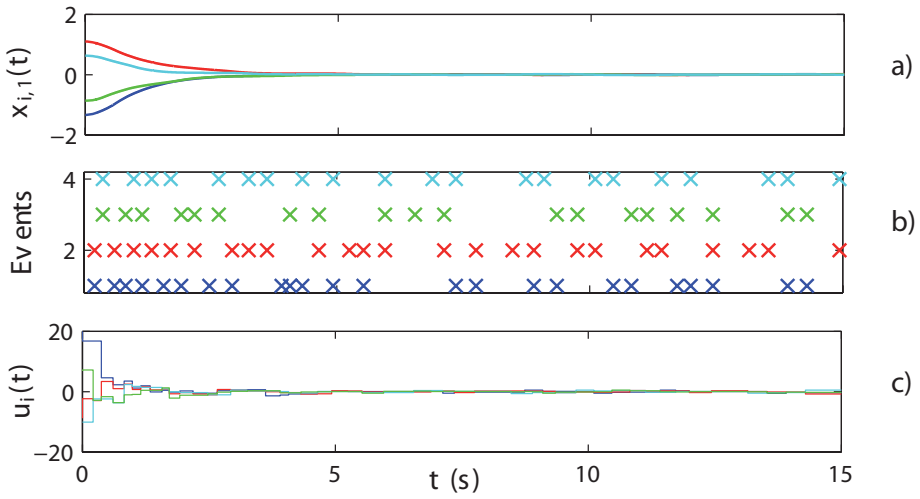

Figure 4.7: Simulation results with trigger functions (4.26) with $c_0 = 0.02$, $c_1 = 0.5$, $\alpha = 0.8$.

The convergence of the system to a small region ($c_0 = 0.02$) around equilibrium point is guaranteed due to the time-dependency in the trigger functions. The event generation is shown in Figure 4.7.b. 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 seconds. Table 4.1 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 [FPW97], 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 (Figure 4.7c). They are updated if an event is triggered by the agent or its neighbors.

Table 4.2 extends this study for a larger number of agents. Several simulations

Table 4.1: 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 4.2: Inter-event times for different N.

	N	10	50	100	150	200
Trigger condition (4.26)	$\{T_k^i\}_{\min}$ (s)	0.053	0.031	0.015	0.019	0.009
	$\{T_k^i\}_{\text{mean}}$ (s)	0.565	0.565	0.567	0.572	0.568
Trigger condition of [WL08]	$\{T_k^i\}_{\text{mean}}$ (s)	0.1149	0.1175	0.1152	0.1180	0.1177

were performed for different initial conditions for each value of N. 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.

Moreover, if we compare these results to [WL08], we see that the proposed scheme can provide around five times larger broadcast periods. For example, for a number of pendulums of $N = 100$, trigger functions of the form (4.26) give a mean broadcasting period of 0.567, while the trigger functions in [WL08] give a mean value of 0.1152.

Though the scheme in [WL08] ensures asymptotic stability, we guarantee the convergence to an arbitrary small region around the origin with $c_0 \neq 0$. Alternatively, one can choose $c_0 = 0$ to get rid of this drawback.

4.6 The non-perfect decoupling case

In this section the effect of the coupling terms Δ_{ij} is analyzed. In (4.24) Δ can be seen as a perturbation to A_K. Thus, the stability results of Section 4.5 are modified according to this perturbation.

The next lemma provides a bound for $\|\Delta\|$ that ensures that $A_K + \Delta$ is Hurwitz. We assume that Assumption 4.1 holds. The discussion to defective matrix is given afterwards.

LEMMA 4.1. *If $\kappa(V)\|\Delta\| < |\lambda_{max}(A_K)|$ holds, the eigenvalues $\tilde{\lambda}_i$ of $A_K + \Delta$ have negative real part.*

Proof. According to the Bauer-Fike theorem (see (4.10) on page 102), 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 $\lambda_{max}(A_K)$ (4.4), then it yields $|\lambda_{max}(A_K)| \leq |\alpha_j|, \forall j$. Moreover, if $\kappa(V)\|\Delta\| < |\lambda_{max}(A_K)|$, $\kappa(V)\|\Delta\|$ is also upper bounded by $|\alpha_j|, \forall j$. Thus, $\tilde{\alpha}_i$ is negative, because if it was positive

$$|\tilde{\alpha}_i - \alpha_j| = \tilde{\alpha}_i + |\alpha_j| > |\alpha_j| \geq |\lambda_{max}(A_K)| > \kappa(V)\|\Delta\|,$$

that would contradict the theorem of Bauer-Fike. Hence, $\tilde{\alpha}_i$ is negative, and this concludes the proof. □

Remark 4.4. 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 (4.11), enforcing $\max\{\theta_1, \theta_1^{1/n}\} < |\lambda_{max}(A_K)|$.

Hence, before stating the main results of this section, the following assumption is required.

Assumption 4.2. The coupling terms Δ_{ij} are such that $\kappa(V)\|\Delta\| < |\lambda_{max}(A_K)|$ holds.

The next theorem generalizes the results of Theorem 4.1 when $\|\Delta\|$ is constrained by Assumption 4.2. The proof can be found in the Appendix B on page 247.

Theorem 4.2. *Consider the closed-loop system* (4.24) *and trigger functions of the form* (4.26)*, with* $0 < \alpha < |\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|$*. Then, if assumptions 4.1 and 4.2 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:

$$\begin{aligned}\|x(t)\| \leq & \kappa(V)\Big(\tfrac{\|BK\|\sqrt{N}c_0}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|} + e^{-(|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\big(\|x(0)\|- \\ & \|BK\|\sqrt{N}\big(\tfrac{c_0}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|} + \tfrac{c_1}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|-\alpha}\big)\big) \\ & + e^{-\alpha t}\tfrac{\|BK\|\sqrt{N}c_1}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|-\alpha}\Big). \end{aligned} \tag{4.39}$$

Furthermore, the inter-event times are lower bounded by

$$T_{\Delta,min} = \frac{c_0}{k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3}}, \tag{4.40}$$

where

$$k_{\Delta,1} = \kappa(V)\|A_K + \Delta\|\|x(0)\| \tag{4.41}$$

$$k_{\Delta,2} = \|BK\|\sqrt{N}c_1\big(\frac{\kappa(V)\|A_K + \Delta\|}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\| - \alpha} + 1\big) \tag{4.42}$$

$$k_{\Delta,3} = \|BK\|\sqrt{N}c_0\big(\frac{\kappa(V)\|A_K + \Delta\|}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|} + 1\big). \tag{4.43}$$

Remark 4.5. Less conservative results can be derived from Theorem 4.2 if the coupling terms are small enough that the following approximations can be taken

$$\begin{aligned} e^{\kappa(V)\|\Delta\|t} &\approx 1 + \kappa(V)\|\Delta\|t, \\ \frac{1}{1 - \frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}} &\approx 1 + \tfrac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}, \\ \frac{1}{1 - \frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|-\alpha}} &\approx 1 + \tfrac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|-\alpha}. \end{aligned}$$

In this situation, it falls out that the state can be upper bounded by

$$\begin{aligned}\|x(t)\| \leq & \kappa(V)\Big(\tfrac{\|BK\|\sqrt{N}c_0\beta_0}{|\lambda_{max}(A_K)|} + e^{-|\lambda_{max}(A_K)|t}(1 + \kappa(V)\|\Delta\|t)(\|x(0)\|- \\ & \|BK\|\sqrt{N}\big(\tfrac{c_0\beta_0}{|\lambda_{max}(A_K)|} + \tfrac{c_1\beta_1}{|\lambda_{max}(A_K)|-\alpha}\big)\big) + e^{-\alpha t}\tfrac{\|BK\|\sqrt{N}c_1\beta_1}{|\lambda_{max}(A_K)|-\alpha}\Big), \end{aligned} \tag{4.44}$$

where $\beta_0 = 1 + \frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|}, \beta_1 = 1 + \frac{\kappa(V)\|\Delta\|}{|\lambda_{max}(A_K)|-\alpha}$.

A similar approximation will be useful when dealing with discrete-time systems in Section 4.7, and this is why it is brought up here.

It can also be proven (see the Appendix B, page 249) that if these approximation are taken, the lower bound for the inter-event times can be computed as follows:

$$T'_{\Delta,min} = \frac{-b_\Delta + \sqrt{b_\Delta^2 + 4a_\Delta c_0}}{2a_\Delta}, \tag{4.45}$$

where

$$a_\Delta = 0.5\kappa(V)\|\Delta\|k_{\Delta,1}$$
$$b_\Delta = k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3},$$

if $k_{\Delta,2}$ and $k_{\Delta,3}$ are approximated as

$$k_{\Delta,2} \approx \|BK\|\sqrt{N}c_1\left(\tfrac{\kappa(V)\|A_K+\Delta\|\beta_1}{|\lambda_{max}(A_K)|-\alpha} + 1\right),$$

$$k_{\Delta,3} \approx \|BK\|\sqrt{N}c_0\left(\tfrac{\kappa(V)\|A_K+\Delta\|\beta_0}{|\lambda_{max}(A_K)|} + 1\right).$$

4.6.1 Simulation results

The effect of the coupling terms over the performance of the system is illustrated next. Let us consider the same setup that for the perfect decoupling (see Figure 4.5). Assume that the length of the chain of inverted pendulums is $N = 50$ and that the matching condition does not hold, that is the terms $\Delta_{ij} \neq 0$ for $j \in N_i$. Specifically, coupling terms Δ_{ij} are randomly induced such that $\|\Delta_{ij}\| \leq 0.1\|H_{ij}\|, j \in N_i$, and these terms act as a disturbance to the system.

If $\|\Delta\|$ and $\lambda_{max}(A_K + \Delta)$ are computed for this simulation, we get 0.3242 and -0.7433, respectively. Thus, a more conservative value of α than in the perfect decoupled case is required to ensure the stability of the system. Thus, the selected parameters of (4.26) are $c_0 = 0.02$, $c_1 = 0.3$ and $\alpha = 0.5$ to guarantee equivalent performance specifications.

Figure 4.8 shows the output, the events generated and the control signal for the nodes 2, 3, 4, 10 and 50, respectively. A disturbance is induced at time $t = 7s$ at the pendulum 3 (blue line). Observe how the disturbance also affects the neighbors of the third node, 2 and 4 (red lines). This effect can be noticed in

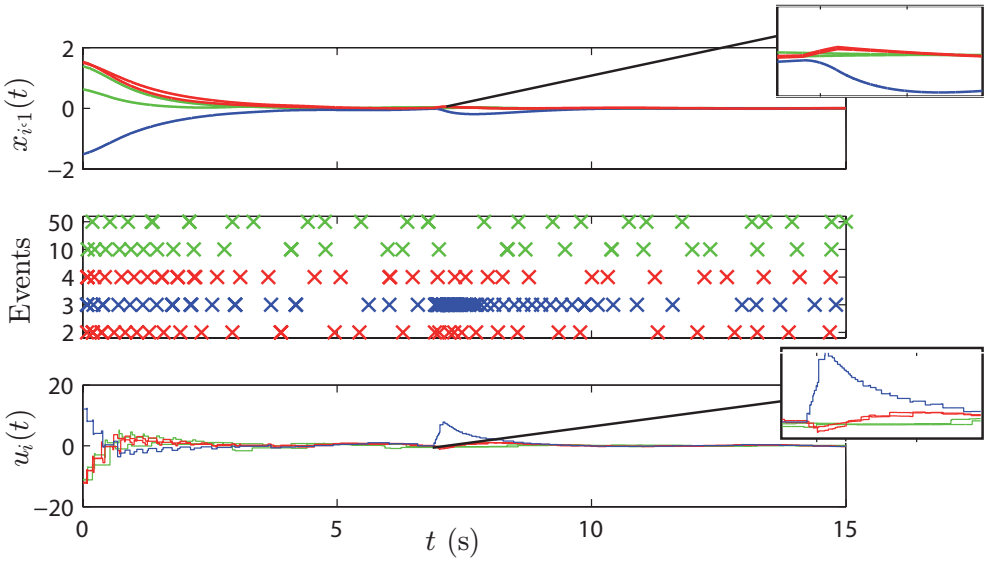

Figure 4.8: System response when N=50 for $\|\Delta_{ij}\| \leq 0.1\|H_{ij}\|$. Trigger function parameters: $c_0 = 0.02, c_1 = 0.3, \alpha = 0.5$.

the output of the subsystem, the events generation and the control law. However, there is no effect over nodes which are far away from the third one (green lines).

We can conclude that the event-based communication respects somehow the idea of neighborhood in a large scale system. Specifically, in a interconnected linear system, even if the system is not perfectly decoupled, the generation of events at a node takes place when something occurs (for instance, a disturbance), and an area around this node starts communicating in order to reject the disturbance, but the rest of the system is not affected.

4.7 Extension to discrete-time systems

4.7.1 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. And let us denote by

$$A_{d,i} = e^{A_i T_s}, \quad B_{d,i} = \int_0^{T_s} B_i e^{A_i s} ds, \quad H_{d,ij} = \int_0^{T_s} H_{ij} e^{A_i s} ds \tag{4.46}$$

Thus, the discrete-time dynamical equation describing each subsystem is

$$x_i(\ell+1) = A_{d,i} x_i(\ell) + B_{d,i} u_i(\ell) + \sum_{j \in N_i} H_{d,ij} x_j(\ell). \tag{4.47}$$

The control law is given by

$$u_i(\ell) = K_{d,i} x_{b,i}(\ell) + \sum_{j \in N_i} L_{d,ij} x_{b,j}(\ell), \tag{4.48}$$

where $x_{b,i}(\ell)$ is the last broadcasted state, $K_{d,i}$ is the feedback gain and $L_{d,ij}$ are the decoupling gains for the discrete-time subsystem i. The error is defined again as the difference between the last broadcasted state and the measured state. Thus,

$$e_i(\ell) = x_{b,i}(\ell) - x_i(\ell), \tag{4.49}$$

and (4.47) can be rewritten in terms of the error $e_i(\ell)$ as

$$x_i(\ell+1) = A_{dK,i} x_i(\ell) + B_{d,i} K_{d,i} e_i(\ell) + \sum_{j \in N_i} \Delta_{d,ij} x_j(\ell) + B_{d,i} L_{d,ij} e_j(\ell), \tag{4.50}$$

where $A_{dK,i} = A_{d,i} + B_{d,i} K_{d,i}$ and $\Delta_{d,ij} = B_{d,i} L_{d,ij} + H_{d,ij}$. $K_{d,i}$ is designed so that all the eigenvalues of $A_{dK,i}$ lie inside the unit circle.

If we define:

$$A_{dK} = \text{diag}(A_{dK,1}, A_{dK,2}, ..., A_{dK,N}) \tag{4.51}$$

$$B = \text{diag}(B_{d,1}, B_{d,2}, ..., B_{d,N}) \tag{4.52}$$

$$K_d = \begin{pmatrix} K_{d,1} & L_{d,12} & \cdots & L_{d,1N} \\ L_{d,21} & K_{d,2} & \cdots & L_{d,2N} \\ \vdots & \vdots & \ddots & \vdots \\ L_{d,N1} & L_{d,N2} & \cdots & K_{d,N} \end{pmatrix} \tag{4.53}$$

$$\Delta_d = \begin{pmatrix} 0 & \Delta_{d,12} & \cdots & \Delta_{d,1N} \\ \Delta_{d,21} & 0 & \cdots & \Delta_{d,2N} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{d,N1} & \Delta_{d,N2} & \cdots & 0 \end{pmatrix} \tag{4.54}$$

and the overall system state and error, respectively, as

$$x = (x_1^T, x_2^T, ..., x_N^T)^T \tag{4.55}$$

$$e = (e_1^T, e_2^T, ..., e_N^T)^T, \tag{4.56}$$

it follows that

$$x(\ell + 1) = (A_{dK} + \Delta_d)x(\ell) + B_d K_d e(\ell) \tag{4.57}$$

4.7.2 Discrete-time trigger functions

Trigger functions of the form (4.26) are difficult to implement in digital platforms since they involve a decaying exponential. Therefore, for discrete-time systems we propose the following functions

$$f_i(e_i(\ell)) = \|e_i(\ell)\| - (c_0 + c_1\alpha_d^\ell), \quad 0 < \alpha_d < 1 \tag{4.58}$$

since they can be assimilated to (4.26) 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:

$$\ell_{k+1}^i = \inf\{\ell > \ell_k^i, f_i(e_i(\ell)) \geq 0\}.$$

Example 4.2: Let us consider a trigger function $f_i(e_i(t)) = \|e_i(t)\| - (0.01 + 0.5e^{-0.8t})$ in continuous-time t, which bounds the error $\|e_i(t)\| \leq (0.01 + 0.5e^{-0.8t})$. This bound is depicted in Figure 4.9 (green 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 (4.58) can be defined with the same values for c_0 and c_1 and

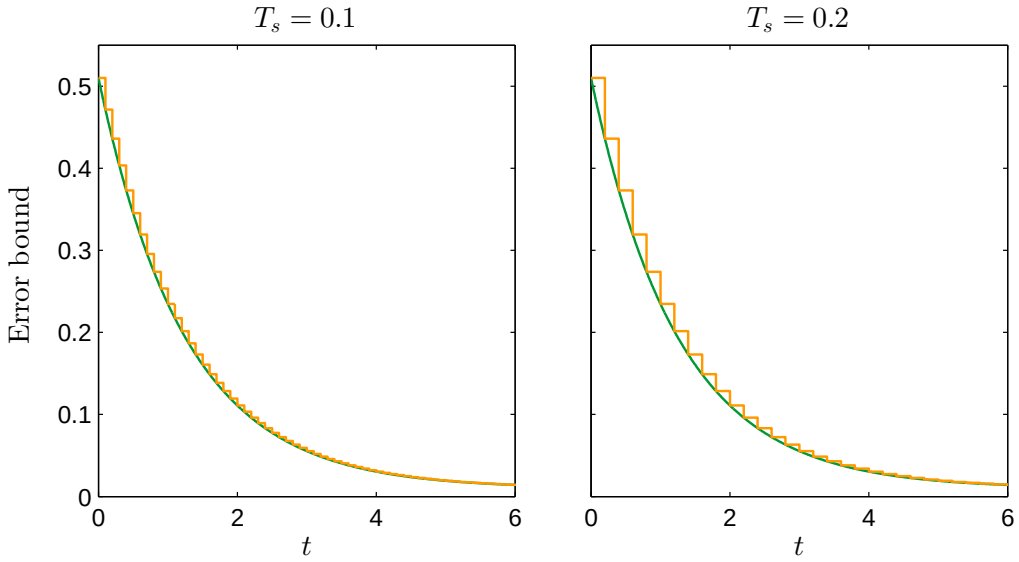

Figure 4.9: Comparative of time-continuous (green) and discrete-time (orange) trigger functions, $T_s = 0.1$ (left), $T_s = 0.2$ (right).

with $\alpha_d = e^{-\alpha T_s}$. This yields values $\alpha_d = 0.9231$ and $\alpha_d = 0.8521$, respectively.

The error bounds for both cases is shown in Figure 4.9. Note that this bound is a piecewise constant function and changes at the sampling time instances.

4.7.3 Stability analysis

Theorems 4.1 (perfect decoupling) and 4.2 (non perfect decoupling) sum up the stability results for the continuous time system. Equivalent results can be derived for the discrete-time system (4.57).

However, a remark should be pointed out first. Whereas in continuous time the state is monitored continuously and this ensures that the error $e_i(t)$ is strictly upper bounded by $c_0 + c_1 e^{-\alpha t}$, in discrete-time systems it might occur that for a given ℓ, $\|e_i(\ell)\| < c_0 + c_1\alpha_d^{\ell}$, but $\|e_i(\ell+1)\| > c_0 + c_1\alpha_d^{\ell+1}$, 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 4.3. Fast sampling is assumed [HI01] so that events occur in all probability at the sampling times ℓ. Hence, $\|e_i(\ell_k^i)\| \approx c_0 + c_1\alpha_d^{\ell_k^i}$ for some $\ell = \ell_k^i$.

The next theorem states that the system (4.57), when trigger functions (4.58) are used, converges to a region around the origin, which depends on c_0.

The proof of the theorem can be found in the Appendix B on page 250, being two the clues to follow the proof. First, all the eigenvalues of A_{dK} lie inside the unit circle, so that $|\lambda_M(A_{dK})|^\ell < 1, \forall \ell \geq 0$ and $|\lambda_M(A_{dK})|^\ell \xrightarrow{\ell\to\infty} 0$, being $\lambda_M(A_{dK})$ the maximum of the eigenvalues of A_{dK}. Secondly, the perturbation analysis for matrix powers, and in particular (4.13), can be applied.

Before enouncing the theorem, the following assumption is required:

Assumption 4.4. A_{dK} is diagonalizable so that $A_{dK} = V_d D_d V_d^{-1}$, and the coupling terms are such that $\kappa(V_d)\|\Delta_d\| < 1 - |\lambda_M(A_{dK})|$, where $\kappa(V_d) = \|V_d\|\|V_d^{-1}\|$ and $\lambda_M(A_{dK})$ is the eigenvalue of A_{dK} with the closer magnitude to 1. Furthermore, it is assumed that Δ_d is such that the second order terms can be approximated to zero $\mathcal{O}(\|\Delta_d\|^2) \approx 0$.

Note that when $\alpha_d \neq 0$, and additional constraint is imposed to the coupling terms. Specifically, the condition $|\lambda_M(A_d^K)| + \kappa(V_d)\|\Delta_d\| < \alpha < 1$ ensures the convergence to the equilibria.

THEOREM 4.3. *Consider the closed-loop system* (4.57) *and trigger functions of the form* (4.58)*, where* $|\lambda_M(A_d^K)| + \kappa(V_d)\|\Delta_d\| < \alpha < 1$*. If Assumptions 4.3 and 4.4 hold, then, for all initial conditions* $x(0) \in \mathbb{R}^n$ *and* $\ell > 0$*, it holds*

$$\begin{aligned}\|x(\ell)\| \leq \kappa(V_d)\Bigg(&\tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|}\beta_{d,0} + |\lambda_M(A_{dK})|^\ell\Big(\|x(0)\| - \tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|}\beta_{d,0} \\ &- \tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\beta_{d,1} + \tfrac{\kappa(V_d)\|\Delta_d\|}{|\lambda_M(A_{dK})|}\ell\big(\|x(0)\| - \tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|} - \tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\big)\Big) \\ &+ \alpha_d^\ell \tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\beta_{d,1}\Bigg),\end{aligned} \tag{4.59}$$

where

$$\beta_{d,0} = 1 + \frac{\kappa(V_d)\|\Delta_d\|}{1 - |\lambda_M(A_{dK})|} \tag{4.60}$$

$$\beta_{d,1} = 1 + \frac{\kappa(V_d)\|\Delta_d\|}{\alpha - |\lambda_M(A_{dK})|}. \tag{4.61}$$

Note that the results are equivalent to (4.44) for discrete-time systems.

Remark 4.6. If perfect decoupling can be achieved, then $\|\Delta_d\| = 0$, which yields

Table 4.3: Values of $\|\Delta_d\|^2$ for several sampling times and number of subsystems, and $\|A_{dK}\|$.

N	10	20	50	100	200	$\|A_{dK}\|$
$T_s = 1$ ms	0.739e^{-7}	0.986e^{-7}	0.929e^{-7}	0.989e^{-7}	1.019e^{-7}	0.0037
$T_s = 5$ ms	1.780e^{-6}	1.785e^{-6}	2.435e^{-6}	2.732e^{-6}	2.736e^{-6}	0.0185
$T_s = 10$ ms	0.729e^{-5}	0.868e^{-5}	0.924e^{-5}	0.923e^{-5}	1.044e^{-5}	0.0369
$T_s = 20$ ms	2.655e^{-5}	3.610e^{-5}	3.754e^{-5}	4.066e^{-5}	4.627e^{-5}	0.0735

$\beta_{d,0}, \beta_{d,1} = 1$. Thus, (4.59) is simplified:

$$\|x(\ell)\| \leq \kappa(V_d)\Big(\tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|} + |\lambda_M(A_{dK})|^\ell\Big(\|x(0)\| - \tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|} - \tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\Big) + \alpha_d^\ell \tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\Big).$$

Example 4.3: In this example we study how restrictive is the approximation taken in Assumption 4.4 concerning the norm of $\|\Delta_d\|$, i.e., $\mathcal{O}(\|\Delta_d\|^2) \approx 0$. Let us consider the scenario of Section 4.6.1, in which $\|\Delta\|$ was computed to be 0.3242.

The value of $\|\Delta_d\|^2$ has been computed for various sampling times and number of agents. Table 4.3 depicts the results. Observe that the change of $\|\Delta_d\|^2$ with the number of agents is not remarkable, and that it increases with the sampling period. The last column on the right shows the value of $\|A_{dK}\|$ for each sampling period, as it remains constant with the number of agents.

If $\|A_{dK}\|$ and $\|\Delta_d\|^2$ are compared, the smaller T_s, the better the rate. Since fast sampling is assumed (see Assumption 4.3), it can be concluded that the approximation is fair enough.

4.8 Conclusions

A novel distributed event-based control strategy for linear 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 time-dependent 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.

The perfect decoupling case has been considered first to simplify the analysis. Since perfect decoupling is difficult to achieve in many situations due to model uncertainties or the matrix B_i does not have full rank, the influence of the coupling terms has been analyzed and the constraints that guarantee that the stability of the system is preserved have been derived.

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.

5

Extensions and improvements of the distributed event-based control

Summary

This chapter is focused on two aspects. The first aspect is the study of how realistic communication affects the distributed event-based control presented in Chapter 4. We analyze the consequences of a non-reliable channel, and upper bounds on the delay and the number of consecutive packet losses are obtained for different situations. Two communications protocols are proposed and analytical results are derived for perfect and non-perfect decoupling.

The second aspect that this chapter accounts for is the proposal of some improvements to the design described in Chapter 4. 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 broadcasted 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 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 continuously and computes the control law accordingly. Model uncertainty is assumed and the performance of the Chapter 4's and model-based designs are compared.

5.1 Introduction

Event-based control and non-reliable networks

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 [BA11]. However, up to now, only a reduced number of papers has 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 papers [CH08, RJ09] study simple stochastic systems and investigate the event-based control performance in dependence upon the medium access mechanism applied.

In [GA11a, LL12], 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 [WL11]. As stated in this paper, one problem that might present trigger functions of the form $\|e_i(t)\| \leq \sigma_i \|x_i(t)\|$ 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. The proposed approach in Section 5.3 solves this problem for linear systems.

Reducing the actuation in event-based control

The importance of reducing the number of control actions in order to save energy has been showed up in recent publications such as [JHC07, RJJ08, DTH12]. In [JHC07] a first-order linear stochastic process is sampled periodically and a sporadic controller decides whether to apply a new control action based on the cost of control actions. In [RJJ08] and [DTH12] optimization problems are solved in order to not exceed certain limits on the switching rate, and to maximize the time elapsed between two consecutive executions of the control task, respectively.

Furthermore, reducing actuation is also important because some actuators are subject to wear. After some time in operation, this wear may result in phenomena that deteriorate the control performance, such as friction or hysteresis in mechanical actuators [rH06].

In a single control loop the reduction of communication usually implies the reduction of actuator updates [Tab07, ESDCM07]. However, this does not necessary hold in distributed systems.

Recent contributions in distributed event-triggered control follow basically two directions. The first approach assumes sophisticated measurement devices in order to get relative measurements of neighboring nodes. It focuses on the design of triggering rules to reduce the number of the actuator updates for a more efficient usage of the limited resources of embedded processors, in which the control task shares computational and communication resources with other tasks [DFJ12, PF12]. The second approach tries to reduce the communication between the subsystems, as already described in detail in this thesis (see for instance [GDJ$^+$11, MT11, PTNA11, SDJ13, WL11]).

On the one hand, the drawback of the first direction is obvious and lies in the requirement of the measurement devices to provide the relevant relative information. On the other hand, the second approach might lead to a very frequent adaption of the control input, specially if the number of neighbors is large. In fact, the control signal is updated whenever a new measurement is received from a neighboring agent. To the best of our knowledge, both aspects, i.e., reduction

of actuation and communication, have not been considered simultaneously in the context of distributed control systems. This is addressed in Section 5.4.

Distributed model-based control

The previous approaches hold a constant control input in the inter-event time. In contrast, model-based control [MA03b] takes advantage of the knowledge of the dynamics of the system to generate a control signal based on the prediction given by the model. However, just a few publications have exploited this idea with event-triggered sampling. An emulation approach is presented in [LL10] for a single loop in which a control input generator and an event generator emulate the continuous-time state feedback controller. In [SDJ11], the control signal is sampled by a first-order hold, according to the double-integrator dynamics. In [GDJ$^+$11], a distributed model-based design for perfect decoupled interconnected linear systems is presented. Finally, in [GA12], centralized and decentralized approaches of model-based event-triggered control are presented for symmetric interconnections.

In Section 5.5 the results of [GDJ$^+$11] are extended to non-perfect decoupling showing that the transmission rate can be reduced if the model is accurate enough.

5.2 Contributions of this chapter

The first contribution of this chapter is the design of a network protocol that does not require the synchronous update of all the nodes in a given neighborhood when the transmission of data is subject to delay and packet losses, in contrast to [WL11]. Under certain requirements, upper bounds on the allowable delay and the maximum number of consecutive packet losses can be derived.

Another contribution related to the unreliability of the network, is that the system can asymptotically converge to the equilibria while the Zeno behavior is excluded with the proposed design. Moreover, we show that time-dependent trigger functions can provide larger upper bounds on the delay than constant thresholds.

The design proposed in Section 5.4 addresses the problem of reducing communication and actuation simultaneously, which has not been yet studied in distributed control systems.

Another contribution is the analysis of the inter-event times for a distributed model-based approach with model uncertainty. For instance, in the decentralized model-based approach of [GA12] no analysis of the inter-event times is conducted. We also prove that, when the model uncertainty fulfills a certain condition, the model-based approach gives larger minimum inter-event times. Respect to the aforementioned work of [LL10], we assume that the dynamics of each subsystem are not perfectly known and we evaluate the effect of these model uncertainties. Moreover, in the design of [LL10] an invertibility condition is imposed to the matrix A which describes the system-free dynamics. This constraint is not required in Section 5.5.

5.3 Extension to non-reliable network

Consider the linear interconnected system described in Chapter 4 (4.14)

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + \sum_{j \in N_i} H_{ij} x_j(t), \quad \forall i = 1, ..., N, \tag{5.1}$$

and the control law (4.15)

$$u_i(t) = K_i x_{b,i}(t) + \sum_{j \in N_i} L_{ij} x_{b,j}(t), \quad \forall i = 1, ..., N. \tag{5.2}$$

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 broadcasted 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 [WL11].

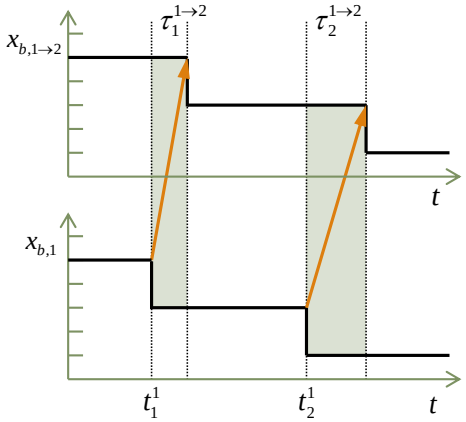

Figure 5.1: Example of state inconsistency of the signal $x_{b,1}$ and its copy $x_{b,1\to2}$ in other node of the network.

Definition 5.1. A distributed event-based control design preserves *state consistency* if any broadcasted state is updated synchronously in each neighboring agent.

Example 5.1: In Figure 5.1 an example of state inconsistency is presented. 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 $x_{b,1\to2}$ accordingly. We denote by $\tau_k^{1\to2}$, $k \in \mathbb{N}$, the communication delay experienced in the broadcast. If the transmission is not subject to delay, both signals $x_{b,1}$ and $x_{b,1\to2}$ are identical. However, this is not the situation in the example of Figure 5.1. In the time intervals $[t_1^1, t_1^1 + \tau_1^{1\to2})$ and $[t_2^1, t_2^1 + \tau_2^{1\to2})$ both signals are not equal. Hence, there is a state inconsistency since $x_{b,1}(t) \neq x_{b,1\to2}(t), \forall t \in [t_1^1, t_1^1 + \tau_1^{1\to2}) \cup [t_2^1, t_2^1 + \tau_2^{1\to2})$.

Therefore, a communication protocol should be defined to avoid state inconsistencies or to deal with them. In this thesis, two different protocols are proposed. The first one is designed to preserve 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 broadcasted states.

5.3.1 Transmission protocols

Before describing the proposed protocols, let us first introduce some notation.

Definition 5.2. We denote by $\tau_k^{i\to j}$ the delay in the transmission of the state $x_i(t_k^i)$ of agent i to its neighbor j, $j \in N_i$, at time t_k^i, and by $\bar{\tau}_k^i$

$$\bar{\tau}_k^i = \max\{\tau_k^{i\to j}, j \in N_i\}.$$

Definition 5.3. We denote by $P_{i\to j}^k$ the number of successive packet losses in the transmission of the state $x_i(t_k^i)$ of agent i to its neighbor j, $j \in N_i$, at time t_k^i, and by P_i^k the maximum of $P_{i\to j}^k$ for all $j \in N_i$.

We now introduce the basic assumption that imposes constraints on delays and the number of consecutive packet dropouts.

Assumption 5.1. We assume that the maximum delay and the number of successive packet dropouts which occur in the transmission of information from the subsystem i to its neighbors $j \in N_i$, denoted by $(\tau^\star)^i$ and $P_i^\star$, respectively, are such that no event is generated before all the neighbors have successfully received the broadcasted state $x_{b,i}$.

The second important consideration is that the sender i knows that the data has been successfully received by j by getting an acknowledgment signal (ACK). If an ACK is not received before a *waiting time* denoted by T_W^i, the packet is treated as lost. How to determine T_W^i 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_W^i, we propose two alternatives that denoted by *Wait for All* (WfA) and *Update when Receive* (UwR).

WfA protocol

The state at $t_k^i + T_W^i$ is broadcasted again to all the neighbors. If after waiting T_W^i an ACK is not received from all $j \in N_i$, the retransmission takes place again, and so on. This process can occur at most $P_i^\star + 1$ 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

(4.15). 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 [WL11]. 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 5.1). Since the broadcasted 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 e_i is the same in all nodes. This allows to define stack vectors for the state $x(t)$ and the error signal $e(t)$ so that the stability of the overall system can be studied as in the ideal network case.

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. Secondly, the WfA protocol may involve unnecessary transmission, since if an agent did not received 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, 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 $j \in N_i$ 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 $P_i^\star + 1$ times.

- Let us denote by $\mathcal{N}_i(t) \subseteq N_i$ the agents to which the subsystem i transmits information at time t. In contrast to WfA, agent i only transmits a new measurement to those agents from which it has not received the ACK signal. If the last event occurred at time t_k^i and $t \in [t_k^i, t_{k+1}^i)$, thus

$$\forall j \in N_i, \notin \mathcal{N}_i(t) \quad \exists t_k^{i\to j} : t_k^i \leq t_k^{i\to j} < t,$$

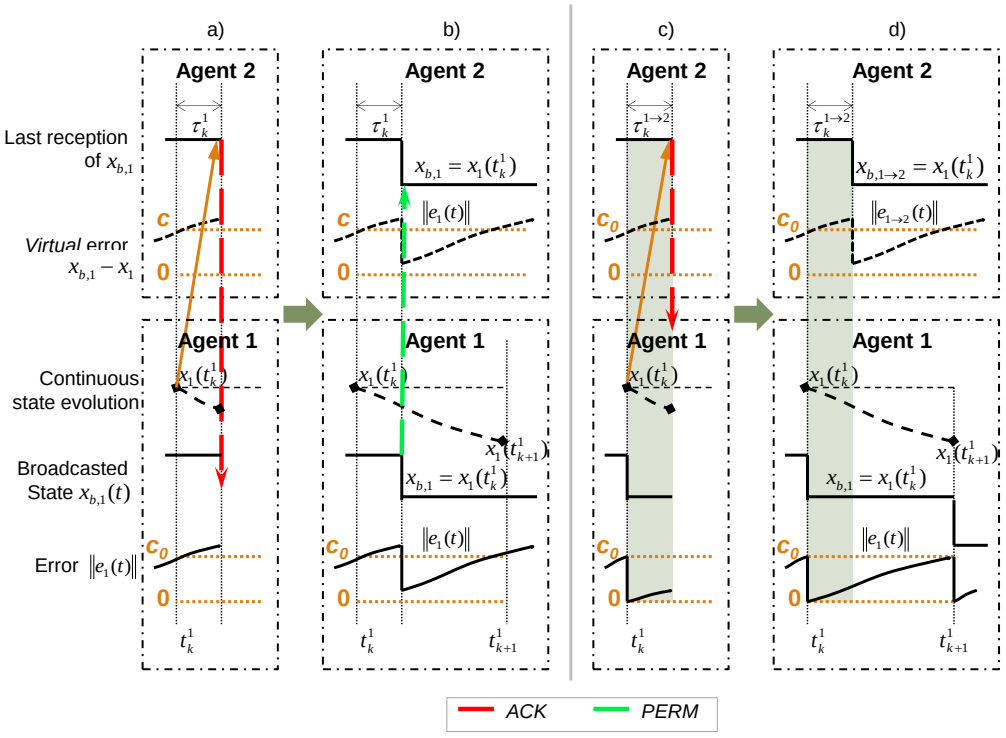

Figure 5.2: Update mechanism of WfA (a) and b)) and UwR (c) and d)) protocols.

where $t_k^{i\to j}$ is the time of the successful broadcast to agent j. Hence if at time t the node j is not in $\mathcal{N}_i(t)$ it means that it has correctly received a broadcasted state after the occurrence of the last event and it has confirmed this reception with an ACK signal.

- The number of consecutive packet losses and the network delay are upper-bounded for each agent i, according to Assumption 5.1. Thus, it must hold

$$\mathcal{N}_i((t_{k+1}^i)^-) = \emptyset,$$

where $(t_{k+1}^i)^-$ refers to the instant time before t_{k+1}^i. I.e., all neighbors have successfully received the state of agent i before the next event occurance.

Example 5.2: In order to clarify the difference between both protocols, a simple example is given in Figure 5.2. A system with two agents is depicted. Assume that Agent 1 detects an event at time t_k^1 and broadcast 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 Figure 5.2a), the PERM signal is sent (both signals are assumed to be sent and received instantaneously), and both agents update the broadcasted state in the control law at the same time $t_k^1 + \tau_k^1$ (see 5.2b). Thus, $x_{b,1}$ takes the same value at any time in both agents and, hence, $e_1(t)$ 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 Figure 5.2c), whereas the receiver updates the state information at time $t_k^1 + \tau_k^{1\to 2}$ (τ_k^1 and $\tau_k^{1\to 2}$ are the same), as illustrated in Figure 5.2d. Thus, in the interval $[t_k^1, t_k^1 + \tau_k^{1\to 2})$ the broadcasted state $x_{b,1}$ has different values in the two nodes and consequently the error e_1 considered in Agent 1 differs from the error affecting the dynamics of Agent 2.
Note that Agent 2 does not monitor e_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, as already discussed in Remark 4.2. The results for the general case when the matching condition does not hold are derived afterwards. For simplicity, Assumption 4.1 applies (diagonalization of $A_{K,i}$).

The focus of the analysis is on constant trigger functions, since they allow to derive analytical expressions for the delay. However, a positive upper bound on the delay can also be derived for time-dependent trigger functions and the proof can be found in the Appendix B. The obtained expression can be solved for some given parameters to provide a numerical solution.

5.3.2 Performance analysis for perfect decoupling

We firstly investigate the performance of the event-based control with constant threshold obtained by using WfA protocol. After that, we extend these results to the situation which uses UwR. Finally, we discuss an extension to time-dependent trigger functions.

Properties of event-triggered control using WfA protocol

Let us consider trigger functions (4.26) with $c_1 = 0$ and $c_0 > 0$, that is

$$f_i(e_i(t)) = \|e_i(t)\| - c_0,\ c_0 > 0. \tag{5.3}$$

Let us first assume that the communication can only experience delays but no packet dropouts.

Communication with delays

Proposition 5.1. *Let us consider trigger functions of the form* (5.3) *and the WfA protocol. If Assumption 5.1 holds, the error of any subsystem i is upper bounded by $\|e_i(t)\| < 2c_0$.*

Proof. Assume that the last event occurred at time t_k^i and that the maximum transmission delay to its neighbors is $\bar{\tau}_k^i$. It follows that

$$\| \int_{t_k^i}^{t_k^i + \bar{\tau}_k^i} \dot{e}_i(s) ds \| = \|e_i(t_k^i + \bar{\tau}_k^i) - e_i(t_k^i)\| < c_0, \tag{5.4}$$

has to be satisfied (see (5.3)) because no event is generated in the time interval $[t_k^i, t_{k+1}^i)$ from Assumption 5.1. Since an event has occurred at time t_k^i, $\|e_i(t_k^i)\| = c_0$ holds and, from (5.4) it holds $\|e_i(t_k^i + \bar{\tau}_k^i)\| < 2c_0$, 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 5.1. *If the network delay is upper bounded by*

$$(\tau^\star)^i = \frac{c_0}{\|A_{K,i}\|\kappa(V_i)\|x_i(0)\| + \mu_i\left(1 + \frac{\|A_{K,i}\|\kappa(V_i)}{|\lambda_{max}(A_{K,i})|}\right)2c_0}, \tag{5.5}$$

where $\mu_i = \|B_i K_i\| + \sum_{j \in N_i} \|B_i L_{ij}\|$, and $\lambda_{max}(A_{K,i})$ and $\kappa(V_i)$ are defined ac-

cording to (4.4) *and* (4.3)*, respectively, then any broadcasted state* $x_{b,i}$ *of any subsystem* $i \in 1,\dots,N$ *is successfully received by the neighbors* $j \in N_i$ *before a new event occurs, and the inter-event time is lower bounded* $t_{k+1}^i - t_k^i \geq (\tau^\star)^i$*. Moreover, for all initial conditions* $x_i(0)$ *and* $t > 0$ *it holds*

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\frac{\mu_i 2c_0}{|\lambda_{max}(A_{K,i})|} + e^{-|\lambda_{max}(A_{K,i})|t}(\|x_i(0)\| - \frac{\mu_i 2c_0}{|\lambda_{max}(A_{K,i})|})\Big). \tag{5.6}$$

Proof. In order to prove the theorem, let us assume that Assumption 5.1 holds.

The analysis will derive an upper bound for the delay which preserves this assumption. The error in the time interval $[t_k^i, t_k^i + \bar{\tau}_k^i)$ is given by

$$e_i(t_k^i + \bar{\tau}_k^i) - e_i(t_k^i) = x_i(t_k^i) - x_i(t_k^i + \bar{\tau}_k^i),$$

since the broadcasted state $x_{b,i}$ is not updated in any agent before the time instance $t_k^i + \bar{\tau}_k^i$ according to the WfA protocol, so that $x_{b,i}(t_k^i + \bar{\tau}_k^i) = x_{b,i}(t_k^i) = x_i(t_{k-1}^i)$ holds. This yields

$$e_i(t_k^i+\bar{\tau}_k^i)-e_i(t_k^i) = (I-e^{A_{K,i}\bar{\tau}_k^i})x_i(t_k^i)+\int_0^{\bar{\tau}_k^i} e^{A_{K,i}s}(B_iK_ie_i(s)+B_i\sum_{j\in N_i} L_{ij}e_j(s))ds,$$

based on which the upper bound for the delay $\bar{\tau}_k^i$ can be derived as

$$\begin{aligned}(\tau^\star)_k^i = &\arg\min_{\bar{\tau}_k^i \geq 0}\{\|(I - e^{A_{K,i}\bar{\tau}_k^i})x_i(t_k^i) \\ &+ \int_0^{\bar{\tau}_k^i} e^{A_{K,i}s}(B_iK_ie_i(s) + B_i\sum_{j\in N_i} L_{ij}e_j(s))ds\| = c_0\}.\end{aligned}$$

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 $x_i(t) = e^{A_{K,i}t}x_i(0) + \int_0^t e^{A_{K,i}(t-s)}(B_iK_ie_i(s) + B_i\sum_{j\in N_i} L_{ij}e_j(s))ds$. The error is bounded by $\|e_i(t)\| < 2c_0, \forall i$ by Proposition 5.1. Thus, a bound on $x_i(t)$ can be calculated following the methodology of Chapter 4 as (5.6).

Note that (5.6) is upper bounded by

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\frac{\|B_iK_i\|2c_0 + (\sum_{j\in N_i}\|B_iL_{ij}\|)2c_0}{|\lambda_{max}(A_{K,i})|} + \|x_i(0)\|\Big), \quad \forall t, \tag{5.7}$$

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

$$\dot{e}_i(t) = -A_{K,i}x_i(t) - B_iK_ie_i(t) - \sum_{j\in N_i} B_iL_{ij}e_j(t)$$

holds in the interval $t \in [t^i_{k-1}+\bar{\tau}^i_{k-1}, t^i_k+\bar{\tau}^i_k)$ for any two consecutive events t^i_{k-1}, t^i_k, and, in particular, it holds in the subinterval $[t^i_k, t^i_k + \bar{\tau}^i_k) \subset [t^i_{k-1} + \bar{\tau}^i_{k-1}, t^i_k + \bar{\tau}^i_k)$. Hence, $\dot{e}_i(t)$ can be bounded as

$$\begin{aligned}\|\dot{e}_i(t)\| =& \|A_{K,i}x_i(t) + B_iK_ie_i(t) + \sum_{j\in N_i} B_iL_{ij}e_j(t)\| \\ &\leq \|A_{K,i}\|\|x_i(t)\| + \|B_iK_i\|\|e_i(t)\| + \sum_{j\in N_i}\|B_iL_{ij}\|\|e_j(t)\|. \end{aligned}\tag{5.8}$$

The state $x_i(t)$ can be bounded according to (5.7), and for the error it holds that $\|e_i(t)\| < 2c$ (see Proposition 5.1). Thus, (5.8) can be integrated straightforward in the interval $[t^i_k, t^i_k + \bar{\tau}^i_k)$, and it yields

$$\begin{aligned}\|e_i(t^i_k + \bar{\tau}^i_k) - e_i(t^i_k)\| \leq \Big(&\|A_{K,i}\|\kappa(V_i)\big(\|x_i(0)\| + \\ &\frac{(\|B_iK_i\| + \sum_{j\in N_i}\|B_iL_{ij}\|)2c_0}{|\lambda_{max}(A_{K,i})|}\big) + \big(\|B_iK_i\| + \textstyle\sum_{j\in N_i}\|B_iL_{ij}\|\big)2c_0\Big)\bar{\tau}^i_k.\end{aligned}$$

Thus, the delay bound (5.5) for agent i ensures that Assumption 5.1 is not violated, and this concludes the proof. □

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.

Corollary 5.1. *Assume that the maximum number of consecutive packet losses is upper bounded by $P^\star_i$, and the transmission delay τ^i_k is upper bounded by a*

constant $\bar{\tau}^i$ *given by*

$$\bar{\tau}^i = \frac{(\tau^\star)^i}{P_i^\star + 1}, \tag{5.9}$$

where $(\tau^\star)^i$ *is given by* (5.5)*. 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* (5.6)*.*

Proof. Assuming that an event was triggered at time t_k^i. The accumulated error after $P_i^\star$ consecutive packet losses and a transmission delay $\bar{\tau}_k^i \leq \bar{\tau}^i$ is

$$\begin{aligned} &\underbrace{(e_i(t_k^i + T_W^i) - e_i(t_k^i)) + (e_i(t_k^i + 2T_W^i) - e_i(t_k^i + T_W^i)) + ...}_{P_i^\star \text{ times}} \\ &\quad + (e_i(t_k^i + P_i^\star T_W^i + \bar{\tau}_k^i) - e_i(t_k^i + P_i^\star T_W^i)) \\ &\quad = e_i(t_k^i + P_i^\star T_W^i + \bar{\tau}_k^i) - e_i(t_k^i). \end{aligned} \tag{5.10}$$

Since $P_i^\star T_W^i + \bar{\tau}_k^i \leq P_i^\star T_W^i + \bar{\tau}^i = (P_i^\star + 1)\bar{\tau}^i = (\tau^\star)^i$, and $(\tau^\star)^i$ is also the minimum inter-event time for the system, this implies that $\|e_i(t_k^i + P_i^\star T_W^i + \bar{\tau}_k^i) - e_i(t_k^i)\| < c_0$. Hence, $\|e_i(t)\| < 2c_0$ holds and so does the bound (5.6). □

Remark 5.1. Note that the maximum number of consecutive packet dropouts $P_i^\star$ and the maximum tolerable delay $\bar{\tau}^i$ are correlated. A large value of $P_i^\star$ implies small values of $\bar{\tau}^i$ and vice versa. This way, there is a trade-off between both parameters.

Properties of event-triggered 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 broadcasted states. First, some definitions are introduced to adapt the previous analysis to this new scenario.

Definition 5.4. We denote $\{t_k^{i\to j}\}$ the set of successful broadcasting times from agent i to agents $j \in N_i$, and the error

$$e_{i\to j}(t) = x_{b,i\to j}(t_k^{i\to j}) - x_i(t), \quad t \in [t_k^{i\to j}, t_{k+1}^{i\to j}), \tag{5.11}$$

where $x_{b,i\to j}(t_k^{i\to j})$ is the last successful broadcasted state from agent i to agent

j, $j \in N_i$.

With this definition, the dynamics of agent i is given by

$$\dot{x}_i(t) = A_{K,i}x_i(t) + B_iK_ie_i(t) + \sum_{j\in N_i} B_iL_{ij}e_{j\to i}(t) \tag{5.12}$$

with $e_i(t)$ the agent i's version of the error. We assume that agent i automatically updates its broadcasted 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 5.2. *If the network delay is upper bounded by*

$$\bar{\tau}^i = \frac{(\tau^\star)^i}{P_i^\star + 1}, \tag{5.13}$$

where $P_i^\star$ is the maximum number of consecutive packet losses and

$$(\tau^\star)^i = \frac{c_0}{\|A_{K,i}\|\kappa(V_i)\|x_i(0)\| + \bar{\mu}_i\left(1+\frac{\|A_{K,i}\|\kappa(V_i)}{|\lambda_{max}(A_{K,i})|}\right)2c_0}, \tag{5.14}$$

with $\bar{\mu}_i = \frac{1}{2}\|B_iK_i\| + \sum_{j\in N_i}\|B_iL_{ij}\|$, and the waiting time T_W^i of the UwR protocol is set to $\bar{\tau}^i$, then any broadcasted 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 (5.14)*, and for any initial condition $x_i(0)$ and for any $t > 0$, it holds*

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\frac{\bar{\mu}_i 2c_0}{|\lambda_{max}(A_{K,i})|} + e^{-|\lambda_{max}(A_{K,i})|t}\big(\|x_i(0)\| - \frac{\bar{\mu}_i 2c_0}{|\lambda_{max}(A_{K,i})|}\big)\Big). \tag{5.15}$$

Proof. According to the UwR protocol, $\|e_i(t)\| \leq c_0$ holds and $e_i(t) \neq e_{i\to j}(t)$, in general. However, as Assumption 5.1 applies, $\|e_{i\to j}(t_k^{i\to j}) - e_i(t_k^i)\| < c_0$ yields $\|e_{i\to j}(t)\| < 2c_0$.

Thus, a bound on the state can be derived from (5.12) in a similar way as in Theorem 5.1 and (5.15) holds. The proof of the first part of the theorem can be obtained by following the proof of Theorem 5.1, since in the interval $[t_k^i, t_k^{i\to j})$ the

state information $x_{b,i\to j}$ remains constant in the agent j, so that $\dot{e}_{i\to j}(t) = -\dot{x}_i(t)$ holds. If the error $e_{i\to j}(t)$ is integrated in the interval $[t_k^i, t_k^{i\to j})$ considering that the state is bounded by (5.15), and that the error is bounded as discussed above, then (5.14) is derived. Finally, (5.13) can be derived as in Corollary 5.1. □

Remark 5.2. Note that the delay bound for WfA and UwR protocols are different (see (5.5), (5.14)). Since $\bar{\mu}_i < \mu_i$, under the same initial conditions UwR allows larger delays.

Time-dependent trigger functions

Let us consider trigger functions (4.26) with $c_0 = 0$ and $c_1 > 0$ for simplicity:

$$f_i(e_i(t)) = \|e_i(t)\| - c_1 e^{-\alpha t}, \quad \alpha > 0. \tag{5.16}$$

The case $c_0, c_1 > 0$ is equivalent to having a constant threshold from the analytical point of view.

The motivation of trigger functions of the form (5.16) has been already discussed. Besides, in Chapter 4, it has been proved graphically that the inter-event time is lower bounded if $\alpha < |\lambda_{max}(A_K)|$ (see Section 4.5.2).

Hence, under Assumption 5.1, it seems reasonable that 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 B, since they are derived following similar steps to the previous results.

Performance of WfA protocol. A result equivalent to Proposition 5.1 is derived.

Proposition 5.2. *Let us consider trigger functions of the form* (5.16) *and WfA protocol. If Assumption 5.1 holds, the error of any subsystem i is upper bounded by $\|e_i(t)\| < c_1(1 + e^{\alpha\tau^\star})e^{-\alpha t}$, where $\tau^\star > 0$ is the maximum transmission delay in the system.*

Proof. The proof can be found in Appendix B on page 252. □

Note that the value of $\tau^\star$ is unknown. Its existence is assumed, and the following theorem will prove it, giving the expression to compute it numerically.

THEOREM 5.3. *Let $\alpha < |\lambda_{max}(A_{K,i})|, \forall i = 1, \dots, N$. If the network delay for any broadcast in the system* (5.1) *is upper bounded by*

$$\tau^\star = \min\{(\tau^\star)^i,\ i = 1, \dots, N\} \tag{5.17}$$

being $(\tau^\star)^i$ the solution of

$$\left(\tfrac{k_{1,i}}{c_1} + \tfrac{k_{2,i}}{c_1}(1 + e^{\alpha(\tau^\star)^i})\right)(\tau^\star)^i = e^{-\alpha(\tau^\star)^i}, \tag{5.18}$$

and

$$k_{1,i} = \|A_{K,i}\|\kappa(V_i)\|x_i(0)\| \tag{5.19}$$

$$k_{2,i} = (\|A_{K,i}\|\kappa(V_i)\frac{1}{|\lambda_{max}(A_{K,i})|-\alpha} + 1)\mu_i c_1, \tag{5.20}$$

then any broadcasted state $x_{b,i}$ is successfully received by the neighbors $j \in N_i$ before a new event occurs. Hence, the inter-event times are lower bounded $t^i_{k+1} - t^i_k \geq \tau^\star$. Moreover, for all initial conditions $x_i(0)$ and $t > 0$ it holds

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\frac{\mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha} + e^{-|\lambda_{max}(A_{K,i})|t}\big(\|x_i(0)\| - \frac{\mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha}\big)\Big). \tag{5.21}$$

Proof. The proof can be found in Appendix B on page 253. □

PERFORMANCE OF UWR PROTOCOL. Under this protocol, the system dynamics is given by (5.12). Note that equivalently to the results for constant threshold, it holds that $\|e_i(t)\| \leq c_1 e^{-\alpha t}$ and $\|e_{i\to j}(t)\| < c_1(1 + e^{\alpha\tau^\star})e^{-\alpha t}$, where $\tau^\star > 0$ is the upper bound on the delay derived in the next theorem.

THEOREM 5.4. *Let $\alpha < |\lambda_{max}(A_{K,i})|, \forall i = 1, \dots, N$. If the network delay for any broadcast in the system* (5.1) *is upper bounded by*

$$\tau^\star = \min\{(\tau^\star)^i,\ i = 1, \dots, N\} \tag{5.22}$$

being $(\tau^\star)^i$ *the solution of*

$$\Big(\frac{k_{1,i}}{c_1}+\frac{k_{2,i}}{c_1}+\frac{k_{3,i}}{c_1}(1+e^{\alpha(\tau^\star)^i})\Big)(\tau^\star)^i = e^{-\alpha(\tau^\star)^i}, \tag{5.23}$$

and

$$k_{1,i} = \|A_{K,i}\|\kappa(V_i)\|x_i(0)\| \tag{5.24}$$

$$k_{2,i} = \|B_iK_i\|(1+\frac{\kappa(V_i)\|A_{K,i}\|}{|\lambda_{max}(A_{K,i})|-\alpha})c_1 \tag{5.25}$$

$$k_{3,i} = \sum_{j\in N_i}\|B_iL_{ij}\|(1+\frac{\kappa(V_i)\|A_{K,i}\|}{|\lambda_{max}(A_{K,i})|-\alpha})c_1, \tag{5.26}$$

then any broadcasted state $x_{b,i}$ *is successfully received by the neighbors* $j\in N_i$ *before a new event occurs. Hence, the inter-event times are lower bounded* $t^i_{k+1}-t^i_k\geq\tau^\star$. *Moreover, for all initial conditions* $x_i(0)$ *and* $t>0$ *it holds*

$$\|x_i(t)\|\leq\kappa(V_i)\Big(\frac{\bar\mu_i(\tau^\star)c_1e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha}+e^{-|\lambda_{max}(A_{K,i})|t}\big(\|x_i(0)\|-\frac{\bar\mu_i(\tau^\star)c_1e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha}\big)\Big), \tag{5.27}$$

where $\bar\mu_i(\tau^\star)=\|B_iK_i\|+\sum_{j\in N_i}\|B_iL_{ij}\|(1+e^{\alpha\tau^\star})$.

Proof. The proof can be found in Appendix B on page 254. □

Note that trigger functions (5.16) ensures the asymptotic convergence to the origin while guaranteeing a lower bound for the minimum inter-event time if the delay is below $\tau^\star$. This cannot be achieved if the triggering conditions are of the form $\|e_i(t)\|\leq\sigma_i\|x_i(t)\|$, as pointed out in [WL11].

Example 5.3: Let us consider the chain of inverted pendulums of Figure 4.5 and the control design described in Section 4.5.3. This example illustrates the influence of the parameters of the trigger functions on the upper bound on the delay.

Table 5.1 shows the most conservative computed delay among the subsystems in the network for different values of c_0 in trigger functions 5.3 and for WfA and UwR protocols. Note that the difference between the value of $(\tau^\star)^i$ given by the two protocols increases with c_0 and that the UwR protocol always allows larger (less conservative) values on the delay.

Trigger functions 5.16 depend on two parameters c_1 and α. Figure 5.3 depicts the bounds on the delay for a set of values of $c_1\in[0.1,1]$ and $\alpha\in[0.1,0.95]$ so that $\alpha<|\lambda_{max}(A_K)|=1$

Table 5.1: Delays bounds (5.5) and (5.14) for different values of c_0 and for WfA and UwR protocols.

c_0	0.01	0.02	0.05	0.1
$(\tau^\star)^i_{\text{WfA}}$ (ms)	0.347	0.613	1.140	1.624
$(\tau^\star)^i_{\text{UwR}}$ (ms)	0.363	0.666	1.329	2.054

is satisfied. The figure on the left shows the results for the WfA protocol (solution of (5.18)), and the one on the right for UwR (solution of (5.23)). Observe that $\tau^\star$ is always greater when the transmissions are ruled by the UwR protocol.

If the solutions given for constant trigger functions and time-dependent trigger functions 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 Section 4.5.3, constant thresholds ($c_0 = 0.02$) gives values of $\tau^\star$ around 0.6 ms, whereas for the exponential threshold ($c_1 = 0.5$ and $\alpha = 0.8$), $\tau^\star$ is three (WfA) and five times (UwR) 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.

5.3.3 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 Chapter 4, the dynamics of each agent can be rewritten in terms of the error as

$$\dot{x}_i(t) = A_{K,i}x_i(t) + B_iK_ie_i(t) + \sum_{j\in N_i}\left(\Delta_{ij}x_j(t) + B_iL_{ij}e_j(t)\right).$$

Note that if $\Delta_{ij} \neq \mathbf{0}$, the dynamics of $\dot{x}_i(t)$ explicitly depends on $x_j(t)$, $\forall j \in N_i$. 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 Chapter 4, and rewrite the equations in terms of the overall system state and error as

$$\dot{x}(t) = (A_K + \Delta)x(t) + BKe(t), \tag{5.28}$$

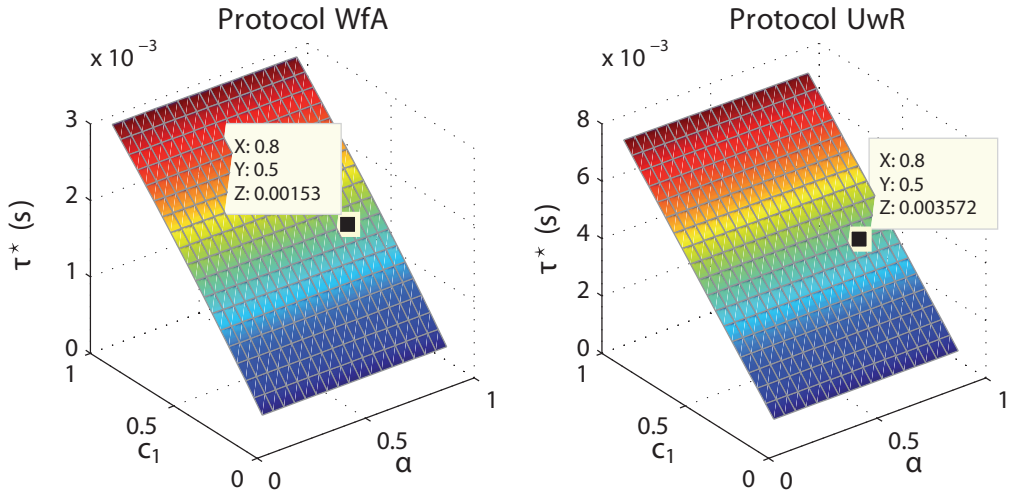

Figure 5.3: Influence of c_1 and α on the delay bound (5.17) (left) and (5.22) (right). The case $c_1 = 0.5$, $\alpha = 0.8$ are 1.53 ms and 3.57 ms, respectively.

where all the matrices and vectors are defined in (4.18)-(4.23).

Let us assume that the communication is subject to delays and packet losses. If the state consistency is preserved, for instance if WfA protocol is considered, (5.28) holds because the update of broadcasted states is synchronized. Under certain assumptions on the error bound (e.g., Proposition 5.1), an equivalent analysis to the perfect decoupling case can be inferred for (5.28). 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 trigger functions (5.3).

Solving the state inconsistency

Let us recall the definition of the error (5.11). 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

$$\dot{x}_i(t) = A_{K,i}x_i(t) + \sum_{j\in N_i} \Delta_{ij}x_j(t) + B_iK_ie_i(t) + \sum_{j\in N_i} B_iL_{ij}e_{j\to i}(t). \qquad (5.29)$$

Let us define the following set of matrices

$$\mathrm{M}_i = B_i(L_{i1}\ L_{i2}\ \dots\ L_{ii-1}\ K_i\ L_{ii+1}\ \dots\ L_{iN}), \forall i = 1, \dots, N, \tag{5.30}$$

with $L_{ij} = 0$ if $j \notin N_i$, and the matrix

$$\mathrm{M} = \begin{pmatrix} \mathrm{M}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathrm{M}_2 & \dots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathrm{M}_N \end{pmatrix}, \tag{5.31}$$

where $\mathbf{0}$ is a $n \times nN$ matrix whose elements are all zero.

Denote by

$$\overrightarrow{e}_i^T = (e_{1\to i}^T\ e_{2\to i}^T\ \dots\ e_{i-1\to i}^T\ e_i^T\ e_{i+1\to i}^T\ \dots\ e_{N\to i}^T), \forall i = 1, \dots, N, \tag{5.32}$$

with $e_{j\to i} = \mathbf{0}$ if $j \notin N_i$, and

$$\overrightarrow{e}^T = (\overrightarrow{e}_1^T\ \dots\ \overrightarrow{e}_N^T). \tag{5.33}$$

With these definitions, the dynamics of the overall system is

$$\dot{x}(t) = (A_K + \Delta)x(t) + \mathrm{M}\overrightarrow{e}(t). \tag{5.34}$$

Lemma 5.1. *If Assumption 5.1 holds and trigger functions* (5.3) *and the UwR protocol are considered, the error* (5.33) *is bounded by*

$$\|\overrightarrow{e}(t)\| \le c_0\sqrt{N + 4\sum_{i=1}^{N}|N_i|} = \bar{c}_0. \tag{5.35}$$

Proof. From (5.33) it follows that

$$\|\overrightarrow{e}(t)\| \le \sqrt{\sum_{i=1}^{N}\|e_i(t)\|^2 + \sum_{i=1}^{N}\sum_{j\in N_i}\|e_{i\to j}(t)\|^2}.$$

Under the UwR protocol, $\|e_i(t)\| \leq c_0$ and $\|e_{i\to j}(t)\| < 2c_0$ hold. It yields

$$\|\overrightarrow{e}(t)\| < \sqrt{\sum_{i=1}^{N} c_0^2 + \sum_{i=1}^{N} \sum_{j\in|N_i|} (2c_0)^2} = \sqrt{c_0^2(N + 4\sum_{i=1}^{N} |N_i|)},$$

which is equivalent to (5.35). □

The previous result 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 $\|e(t)\| < 2\sqrt{N}c_0$, which is a lower upper bound than (5.35). Otherwise, if the opposite is assumed, it follows that $\frac{3}{4}N > \sum_{i=1}^{N} |N_i|$ must hold by enforcing $\bar{c}_0 = c_0\sqrt{N + 4\sum_{i=1}^{N} |N_i|} < 2\sqrt{N}c_0$. 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 5.5. *If the network delay is upper bounded by*

$$\tau^\star = \frac{c_0}{\|A_K+\Delta\|\kappa(V)\|x(0)\|+\mu_{max}\left(1+\frac{\|A_K+\Delta\|\kappa(V)}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|}\right)\bar{c}_0}, \tag{5.36}$$

where $\mu_{max} = max\{\|M_i\|, i = 1, \ldots, N\}$*, then any broadcasted state* $x_{b,i}$ *is successfully received by the neighbors* $j \in N_i$ *before a new event occurs. Hence, the inter-event times are lower bounded* $t_{k+1}^i - t_k^i \geq \tau^\star$*. Moreover, for all initial conditions* $x(0)$ *and* $t > 0$ *it holds*

$$\begin{aligned}\|x(t)\| \leq & \kappa(V)\Big(\frac{\mu_{max}\bar{c}_0}{|\lambda_{max}(A_{K,i})|-\kappa(V)\|\Delta\|} \\ & + e^{-(|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\big(\|x(0)\| - \frac{\mu_{max}\bar{c}_0}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|}\big)\Big).\end{aligned} \tag{5.37}$$

Proof. The proof can be found in Appendix B on page 255. □

Example 5.4: In this example we illustrate the conservativism of Theorem 5.5 when estimating $\tau^\star$, even though the UwR protocol provides better results for perfect decoupling.

Table 5.2: Delays for different values of c_0 and N.

	c_0			
N	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

Let us consider the system specifications of Section 4.6.1. The upper bound on the delay $\tau^\star$ computed according to (5.36) for different values of c_0 and N is given in Table 5.2. The values are expressed in milliseconds.

Given that $\bar{c}_0$ depends on N, 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 also influences the bound negatively.

Remark 5.3. The conservativism of (5.36) 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 $\tau \leq \tau^\star$.

5.3.4 Simulation results

Performance

To illustrate the theoretical results, let us consider the system in Figure 4.5 with $N = 4$ 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.

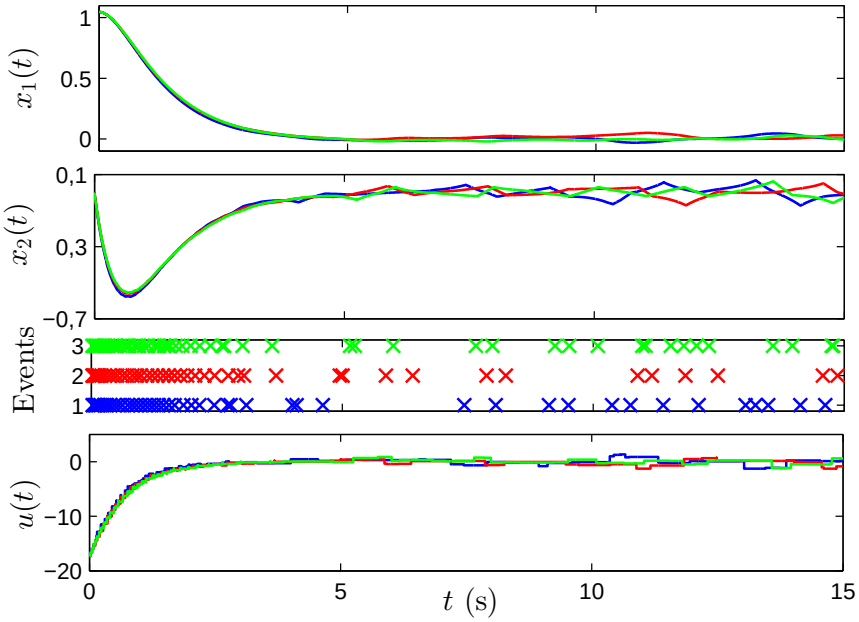

Figure 5.4: Behavior of the subsystem 2 with WfA (red), UwR (green) protocols, and a ideal network (blue).

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 c_0 and summarized in Table 5.1.

Let $c_0 = 0.05$ and a delay generated randomly between zero and the corresponding upper bound specified in Table 5.1 (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 Figure 5.4 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.

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.

WfA vs. UwR

In order to illustrate the difference between WfA and UwR in more detail, Figure 5.5 extracts a short time interval showing how the broadcasted state $x_{b,2}$ of Agent

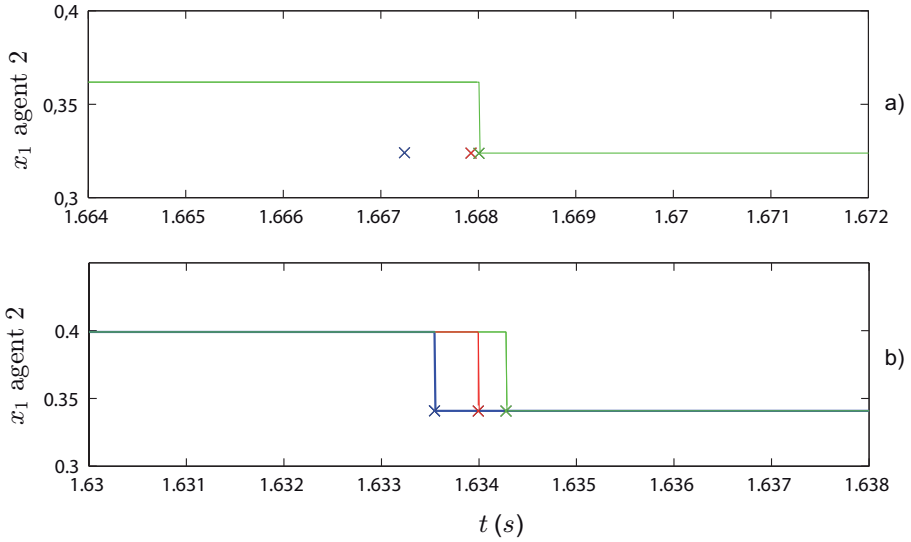

Figure 5.5: Difference between a) WfA and b) UwR protocols in updating the state information. Only the first component of the state is depicted: $x_{b,2}$ (blue), $x_{b,2\to1}$ (red), and $x_{b,2\to3}$ (green).

2 is used in the system. Since Agent 2 is an inner pendulum, it has two neighbors. For WfA protocol the three copies of $x_{b,2}$ (one in Agent 2, one in Agent 1, and the third in Agent 3) are identical. All the neighbors wait for the last reception ($x_{b,2\to3}$ in the depicted case) at time $t = 1.668$ s to update the value of x_b (Figure 5.5a), which is depicted by the green solid line. In contrast, using the UwR protocol (Figure 5.5b), whenever an event is triggered in Agent 2, its state is broadcasted and immediately updated in Agent 2. The neighbors also update as soon as they receive the broadcasted state. Note that the events times are not the same in the two protocols because the time of one update affects the generation of future events, as mentioned before.

Time-dependent trigger function

If time-dependent trigger functions of the form (5.16) with parameters $c_1 = 0.5$ and $\alpha = 0.8$ are taken, the upper bound on the delay is 1.43 ms (WfA) and 3.57 ms (UwR), according to Figure 5.3. 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 $(\tau^\star)^i = 3.57$ ms.

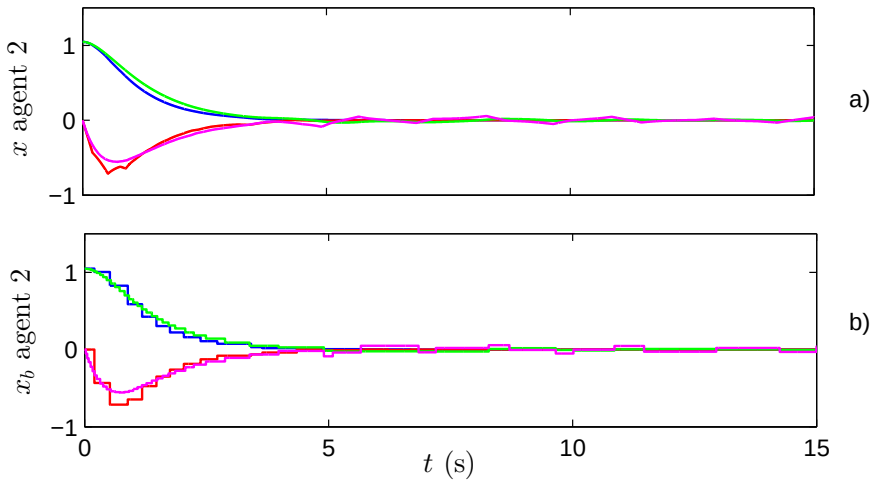

Figure 5.6: Behavior of the agent 2 with trigger functions (5.3) ($c_0 = 0.05$) (green, magenta) and (5.16) ($c_1 = 0.5$, $\alpha = 0.8$) (blue, red), with 3.57 ms as upper bound on the delay. a) (x_{2_1}, x_{2_2}), b) $(x_{b,2_1}, x_{b,2_2})$.

The results are shown in Figure 5.6. The state of agent 2 (x_{2_1}, x_{2_2}) is depicted in Figure 5.6a, and Figure 5.6b shows the broadcasted states $(x_{b,2_1}, x_{b,2_2})$. The broadcasted 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 broadcasted state (number of events) decreases with trigger functions (5.16) and the performance around the equilibria is better with respect to (5.3). Moreover, the minimum and mean inter-event times have been computed according for these simulation results, resulting in 3.9 ms and 353 ms, respectively, for (5.3), and 1.2 ms, which agrees with the results of Table 5.1, and 215 ms for (5.16). Hence, the time-dependent trigger functions are an interesting alternative in non-ideal networks.

5.4 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 [Tab07, ESDCM07], which does not necessary hold

in distributed systems.

The control law is computed in (4.15) based on the broadcasted 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.

5.4.1 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 latest broadcasted state $x_{b,i}$ as

$$e_{x,i}(t) = x_{b,i}(t) - x_i(t), \tag{5.38}$$

and we consider time-dependent trigger functions defined by

$$f_{x,i}(t, e_{x,i}(t)) = \|e_{x,i}(t)\| - c_{x,0} - c_{x,1}e^{-\alpha t}, \tag{5.39}$$

with $c_{x,0} > 0, c_{x,1} \geq 0$, and $\alpha > 0$. An event is detected when $f_{x,i}(t, e_{x,i}(t)) > 0$, and the error $e_{x,i}$ is reset to zero. Note that the error remains bounded by

$$\|e_{x,i}(t)\| \leq c_{x,0} + c_{x,1}e^{-\alpha t}. \tag{5.40}$$

This type of trigger functions has been shown to decrease the number of events while maintaining a good performance of the system. The case $c_{x,0} = 0$ is excluded. The reason is discussed later. However, the case $c_{x,1} = 0$ 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 $\{t_l^i\}_{l=0}^{\infty}, \forall i = 1, \ldots, N$.

The control law is defined for the inter-event time period as

$$u_{b,i}(t) = K_i x_{b,i}(t_l^i) + \sum_{j \in N_i} L_{ij} x_{b,j}(t_l^i), t \in [t_l^i, t_{l+1}^i). \tag{5.41}$$

In order to determine the occurrence of an event, we define

$$e_{u,i}(t) = u_{b,i}(t) - u_i(t), \tag{5.42}$$

where $u_i(t)$ is given by (4.15). The set of trigger functions is given by

$$f_{u,i}(e_{u,i}(t)) = \|e_{u,i}(t)\| - c_u, \quad c_u > 0. \tag{5.43}$$

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 (4.15) 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 $\bar{N}_i = i \cup N_i$ and $\{t_k^{\bar{N}_i}\}$ the set $\{t_k^i\} \cup \{t_k^j\}$, $j \in N_i$. Thus,

$$t_{l+1}^i = \inf\{t_k^{\bar{N}_i} : t_k^{\bar{N}_i} > t_l^i, f_{u,i}(t_k^{\bar{N}_i}) > 0\}.$$

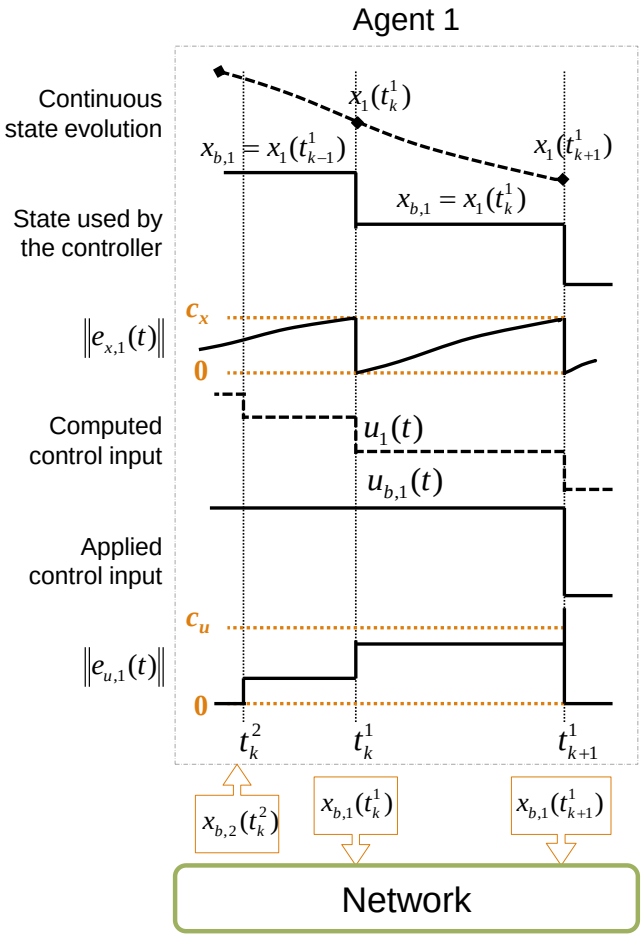

Figure 5.7: Illustrative example of transmission and control update events between a system compound of two agents.

Hence, it holds $\{t_l^i\} \subset \{t_k^{\bar{N}_i}\}$.

Example 5.5: An example of the proposed design is given in Figure 5.7. Assume that Agent 1 sends and receives information to/from its neighborhood through a network. At $t = t_k^2$ it receives a broadcasted 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 c_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 $e_{x,1}$ reaches the threshold c_x. $x_1(t_k^1)$ is broadcasted through the network and u_1 is computed again. Given that $\|e_{u,1}\| < c_u$, $u_{b,1}$ is not modified. Finally, a new event occurs at $t = t_{k+1}^1$ resulting in a broadcast and a control update since $\|e_{u,1}\| \geq c_u$. Note that $u_{b,1}(t) = u_1(t)$ $(e_{u,1} = 0)$ for $t < t_k^2$ and $t \geq t_{k+1}^1$ but this does not hold in the meantime.

5.4.2 Performance analysis

The dynamics of the subsystems (4.14) with control law (5.41) 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 $e_{x,i}(t)$ and $e_{u,i}(t)$ handled by the trigger functions (5.39) and (5.43). respectively, as

$$\dot{x}_i(t) = A_{K,i} x_i(t) + \sum_{j \in N_i} \Delta_{ij} x_j(t) + B_i K_i e_{x,i}(t) + B_i \sum_{j \in N_i} L_{ij} e_{x,j}(t) + B_i e_{u,i}(t).$$

Let us define the stack vectors

$$\begin{aligned} e_x^T &= (e_{x,1}^T \dots e_{x,N}^T) \\ e_u^T &= (e_{u,1}^T \dots e_{u,N}^T), \end{aligned} \tag{5.44}$$

and consider the usual definitions for $x(t)$ and the matrices A_K, B, K, and Δ given in (4.18).

Accordingly, the overall system dynamics is given by

$$\dot{x}(t) = (A_K + \Delta)x(t) + BKe_x(t) + Be_u(t). \tag{5.45}$$

As the broadcasted states $x_{b,i}$ remain constant between consecutive events, the dynamics of the state error in each interval are given by

$$\dot{e}_x(t) = -(A_K + \Delta)x(t) - BKe_x(t) - Be_u(t). \tag{5.46}$$

The state error of the overall system is bounded by

$$\|e_x(t)\| \leq \sqrt{N}(c_{x,0} + c_{x,1}e^{-\alpha t})$$

according to (5.40). However, $e_u(t)$ is not strictly bounded by c_u because $u_i(t)$ is not a continuous function but piecewise constant.

The following assumption let us establish a bound on the control error.

Assumption 5.2. The occurrence of simultaneous transmission events in any neighborhood $\bar{N}_i$ is not allowed, i.e., two neighboring nodes cannot transmit at the same instance of time.

The previous assumption seems reasonable from the network protocol perspective. Assumption 5.2 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 has been already studied in Section 5.3. Hence, similar results could be inferred assuming that the induced delay were at most the bound derived for the transmission delay in the cited section.

Moreover, in case that two broadcasted states were received by one agent, it could enqueue the data and do the computation of the control law sequentially.

LEMMA 5.2. *If Assumption 5.2 holds, the control error of the subsystem i is bounded by*

$$\|e_{u,i}(t)\| \leq \bar{c}_{u,i}(t), \tag{5.47}$$

with

$$\bar{c}_{u,i}(t) = c_u + (c_{x,0} + c_{x,1}e^{-\alpha t}) \cdot max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}.$$

Proof. Assume that the last broadcasting event on the subsystem i occurred at $t = t_k^{\bar{N}_i}$, meaning that its own events and the neighbors' are included. If this last event does not yield a control update it means that $\|e_{u,i}(t_k^{\bar{N}_i})\| < c_u$. Assume that at $t = t_{k+1}^{\bar{N}_i}$ there is a new broadcast in $\bar{N}_i$ which triggers a control event. There are two possibilities:

- The subsystem i triggers the event. Thus,

$$\begin{aligned}
\|e_{u,i}(t_{k+1}^{\bar{N}_i})\| &= \|e_{u,i}(t_k^{\bar{N}_i}) + u_i(t_k^{\bar{N}_i}) - u_i(t_{k+1}^{\bar{N}_i})\| \\
&= \|e_{u,i}(t_k^{\bar{N}_i}) + K_i(x_{b,i}(t_k^{\bar{N}_i}) - x_{b,i}(t_{k+1}^{\bar{N}_i}))\| \\
&\leq \|e_{u,i}(t_k^{\bar{N}_i})\| + \|K_i\|\|x_{b,i}(t_k^{\bar{N}_i}) - x_{b,i}(t_{k+1}^{\bar{N}_i})\|,
\end{aligned}$$

that is upper bounded by

$$\|e_{u,i}(t_{k+1}^{\bar{N}_i})\| \leq c_u + \|K_i\|(c_{x,0} + c_{x,1}e^{-\alpha t_{k+1}^{\bar{N}_i}}).$$

- The event has been triggered for any neighbor $j \in N_i$, it yields

$$\begin{aligned}\|e_{u,i}(t_{k+1}^{\bar{N}_i})\| &= \|e_{u,i}(t_k^{\bar{N}_i}) + L_{ij}(x_{b,j}(t_k^{\bar{N}_i}) - x_{b,j}(t_{k+1}^{\bar{N}_i}))\| \\ &\leq c_u + \|L_{ij}\|(c_{x,0} + c_{x,1}e^{-\alpha t_{k+1}^{\bar{N}_i}}).\end{aligned}$$

Since this holds for all t, and if the worst case is considered, it yields (5.47).

□

Lemma 5.3. *If Assumption 5.2 holds, the control error of the overall system is bounded by*

$$\|e_u(t)\| \leq \sqrt{N}(c_u + \|\mu(K)\|_{max}(c_{x,0} + c_{x,1}e^{-\alpha t})) = \bar{c}_u(t), \tag{5.48}$$

where

$$\mu(K) = \begin{pmatrix} \|K_1\| & \|L_{12}\| & \cdots & \|L_{1N}\| \\ \|L_{21}\| & \|K_2\| & \cdots & \|L_{2N}\| \\ \vdots & \vdots & \ddots & \vdots \\ \|L_{N1}\| & \|L_{N2}\| & \cdots & \|K_N\| \end{pmatrix}, \tag{5.49}$$

and $\|\cdot\|_{max}$ denotes the entry-wise max norm of a matrix.

Proof. From (5.44) and (5.47) it follows that

$$\|e_u(t)\| \leq \sqrt{\sum_{i=1}^{N} \|e_{u,i}\|^2(t)} \leq \sqrt{\sum_{i=1}^{N} \bar{c}_{u,i}^2(t)} \leq \sqrt{N(\max\{\bar{c}_{u,i}(t)\})^2},$$

which is equivalent to (5.48). □

Remark 5.4. Note that, although constant trigger functions are defined for the update of the control actions, the effective bound on the control input is time variant due to the trigger mechanism applied on the state error.

Main result

Assume that Assumptions 4.1 and 4.2 hold. Thus, the following result can be stated.

THEOREM 5.6. *Consider the interconnected linear system* (5.45)*. If trigger functions* (5.39) *are used to broadcast the state with* $0 < \alpha < |\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|$*, and trigger functions* (5.43) *for the control update, then, for all initial conditions* $x(0)$ *and* $t \geq 0$*, it follows that*

$$\|x(t)\| \leq \sigma_1 + (\kappa(V)\|x(0)\| - \sigma_1 - \sigma_2)e^{-(|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|)t} + \sigma_2 e^{-\alpha t}, \quad (5.50)$$

where

$$\sigma_1 = \kappa(V)\sqrt{N}\frac{(\|BK\| + \|B\|\|\mu(K)\|_{max})c_{x,0} + \|B\|c_u}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|} \quad (5.51)$$

$$\sigma_2 = \kappa(V)\sqrt{N}\frac{(\|BK\| + \|B\|\|\mu(K)\|_{max})c_{x,1}}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\| - \alpha}. \quad (5.52)$$

Furthermore, the system does not exhibit Zeno behavior, being the lower bound for the inter-execution times

$$T_{x,min} = \frac{c_{x,0}}{\gamma_1 + \sqrt{N}(\gamma_2 + \gamma_3 + \gamma_4)}, \quad (5.53)$$

where

$$\gamma_1 = \kappa(V)\|x(0)\|\|A_K + \Delta\|$$

$$\gamma_2 = (\|BK\| + \|B\|\|\mu(K)\|_{max})c_{x,0}\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|}\right)$$

$$\gamma_3 = (\|BK\| + \|B\|\|\mu(K)\|_{max})c_{x,1}\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\| - \alpha}\right)$$

$$\gamma_4 = \|B\|c_u\left(1 + \frac{\kappa(V)\|A_K+\Delta\|}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|}\right).$$

Proof. The proof can be found in the Appendix B on page 256. □

The lower bound found for the inter-event times (see (5.53)) is strictly positive since $c_{x,0} > 0$.

Discussion

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 $c_{x,1} = 0$ yielding to static trigger functions. It follows that $\|e_{x,i}(t)\| \leq c_{x,0}$ and $\|e_{u,i}(t)\| \leq c_u + c_{x,0} \cdot \max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}$ $\forall t \geq 0$, according to (5.40) and (5.47), 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 5.2:

$$\begin{aligned}\|e_{u,i}(t) - e_{u,i}(t^*)\| = \|e_{u,i}(t)\| &\leq \sum_{k=1}^{n_e} c_{x,0} \cdot \max\{\|K_i\|, \|L_{ij}\| : j \in N_i\} \\ &= n_e c_{x,0} \max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}\end{aligned}$$

and the next control update event will not be triggered before

$$\|e_{u,i}\| = c_u \leq c_u + c_{x,0}\max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}.$$

Thus,

$$n_e^i \geq \frac{c_u}{c_{x,0}\max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}}. \tag{5.54}$$

Equation (5.54) shows the trade-off between c_u and $c_{x,0}$ and gives insights on how one of these parameters should be chosen according to the other one.

Moreover, (5.54) can be translated into a relationship between the inter-execution times of the control law (5.41), denoted $T_{u,min}^i$, and the minimum broadcasting period (5.53). It holds that

$$T_{u,min}^i \geq n_e^i T_{x,min} \geq \frac{c_u}{(\gamma_1 + \sqrt{N}(\gamma_2 + \gamma_4))\max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}}.$$

Note that $\gamma_3 = 0$ because we are analyzing the case $c_{x,1} = 0$. Let $T_{u,min}$ be

$T_{u,min} = \min\{T^i_{u,min}\}$. It yields

$$T_{u,min} \geq \frac{c_u}{(\gamma_1 + \sqrt{N}(\gamma_2 + \gamma_4))\|\mu(K)\|_{\max}}.$$

Hence, $c_{x,0}$ and c_u can be chosen to meet some constraints on $T_{x,min}$ and $T_{u,min}$.

In the design of Section 5.4.1 the case $c_{x,0} = 0$ was excluded and the reason is given next. Assume that $c_{x,0} = 0$. Thus, following the steps of the previous case, $\|e_{u,i}(t)\| \leq n_e c_{x,1} e^{-\alpha t^*} \max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}$, 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 $\|e_{u,i}\|$ reaches the threshold c_u. In this case, it holds that

$$n_e \geq \frac{c_u}{c_{x,1} e^{-\alpha t^*} \max\{\|K_i\|, \|L_{ij}\| : j \in N_i\}}. \tag{5.55}$$

Note that the lower bound for n_e in (5.55) 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 c_u to the decreasing bound on the state $c_{x,1} e^{-\alpha t}$.

5.4.3 Simulation results

System topology

Let us consider the system presented in Section 4.5.3 but with a different topology. Specifically, the mesh of inverted pendulums is depicted in Figure 5.8. 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.

Each subsystem can be described as

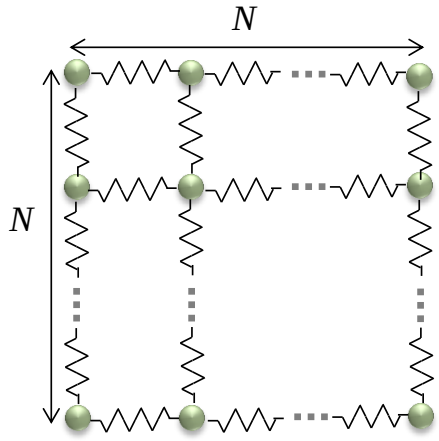

Figure 5.8: Scheme of the coupled pendulums mesh.

$$\dot{x}_i = \begin{pmatrix} A_i & \mathbf{0} \\ \mathbf{0} & A_i \end{pmatrix} x_i + \begin{pmatrix} B_i & \mathbf{0} \\ \mathbf{0} & B_i \end{pmatrix} u_i + \sum_{j \in N_i} \begin{pmatrix} H_{ij} & \mathbf{0} \\ \mathbf{0} & H_{ij} \end{pmatrix} x_j,$$

where

$$A_i = \begin{pmatrix} 0 & 1 \\ \frac{g}{l} - \frac{|N_i|k}{ml^2} & 0 \end{pmatrix}, \quad B_i = \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix}, \quad H_{ij} = \begin{pmatrix} 0 & 0 \\ \frac{k}{ml^2} & 0 \end{pmatrix},$$

and $x_i = (x_{i_1} \quad x_{i_2} \quad x_{i_3} \quad x_{i_4})^T$, $u_i = (u_{i_1} \quad u_{i_2})^T$.

The feedback gains K_i are designed to place the poles at $\{-2, -2, -1, -1\}$. The decoupling gains are designed to decouple the system with uncertainties bounded by $\kappa(V_i)\|\Delta_{ij}\| < 0.35|\lambda_{max}(A_{K,i})|$.

Performance

Figure 5.9 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 $c_{x,0} = 0.02, c_{x,1} = 0.5, \alpha = 0.6$, and $c_u = 0.1$ are considered.

Let us focus on one particular subsystem, for example the agent (2,2) (second row, second column). The state and the control signals are illustrated in Figure 5.10. 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

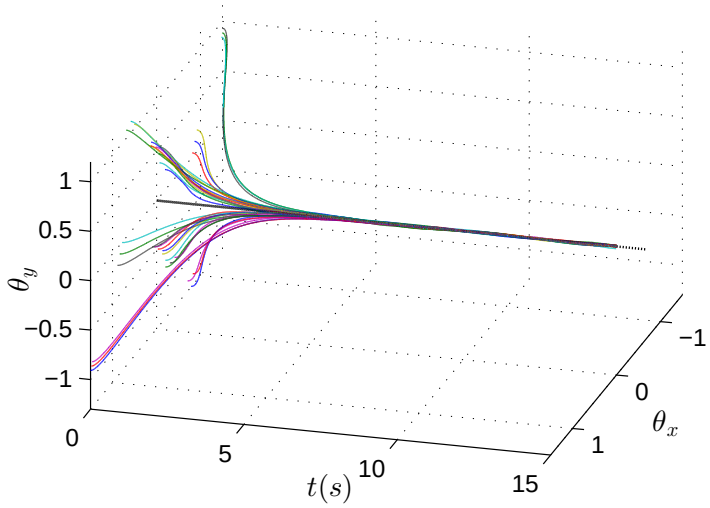

Figure 5.9: x_{i_1} for a 6×6 mesh of inverted pendulums.

the agent (2,2) is 90, so that 47% of the transmissions do not end into a control update because the threshold c_u is not reached.

If this experiment is repeated for the case in which trigger functions (5.43) are not considered (Chapter 4 approach), the number of broadcasting events in the neighborhood of (2,2) is 140, which is equal to the number of control updates.

Thus, as expected, the proposed design with trigger functions (5.43) 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 $\bar{T}_x = \frac{N^2 t_{sim}}{\text{No. events}}$ 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 in Table 5.3. Note that the variations of the average period with the number of agents are not significant.

The influence of the parameter c_u for given parameters $c_{x,0} = 0.02, c_{x,1} = 0.5$, and $\alpha = 0.6$ can be analyzed and the results are illustrated in Table 5.4. For a

Table 5.3: Average broadcasting period variations with N.

$N \times N$	16	36	64	81	100
$\bar{T}_x$	0.5422	0.5202	0.4813	0.4676	0.4765

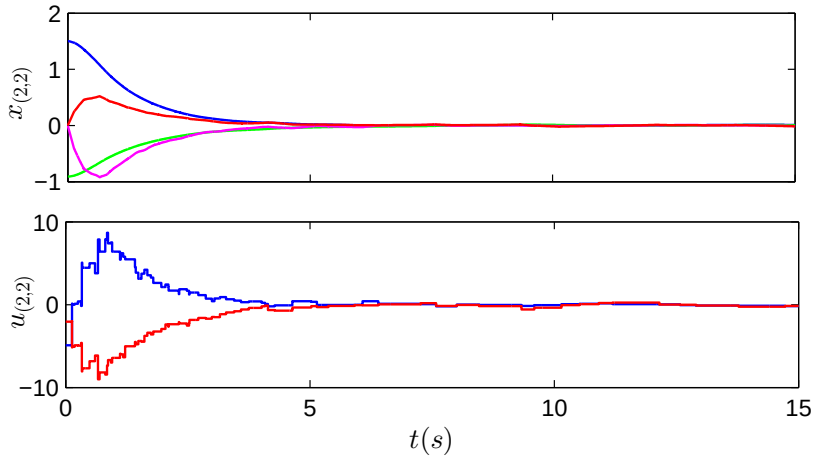

Figure 5.10: State (above) and control signals (below) for agent (2,2) with $c_x = 0.02 + 0.5e^{-0.6t}$, $c_u = 0.1$.

mesh of 6×6 subsystems the following values are computed for each value of c_u and simulation time $t = 15$ s:

- Average number of transmissions through the network defined as $\bar{n}_x = \frac{\sum_{i=1}^{N^2} |\{t_k^i\}|}{N^2} |\overline{N_i}|$, where $|\{t_k^i\}|$ is the cardinality of the set $\{t_k^i\}$ and $|\overline{N_i}|$ is the average for the number of neighboring agents.
- Average number of control updates defined as $\bar{n}_u = \frac{\sum_{i=1}^{N^2} |\{t_l^i\}|}{N^2}$.

Note that the best choice of c_u, $c_{x,0}$, and $c_{x,1}$ depends on the implementation costs of the communication and actuation processes, and the required values for the lower bounds on the transmission and control update inter-event times. For this particular example, it can be said that a value $c_u \in [0.05, 0.1]$ would be a good option because the decrease of the control events is notable while the increase in communication events is assumable. If $c_u = 0.02$ 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 data).

Table 5.4: Average transmission and control update events with c_u.

c_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

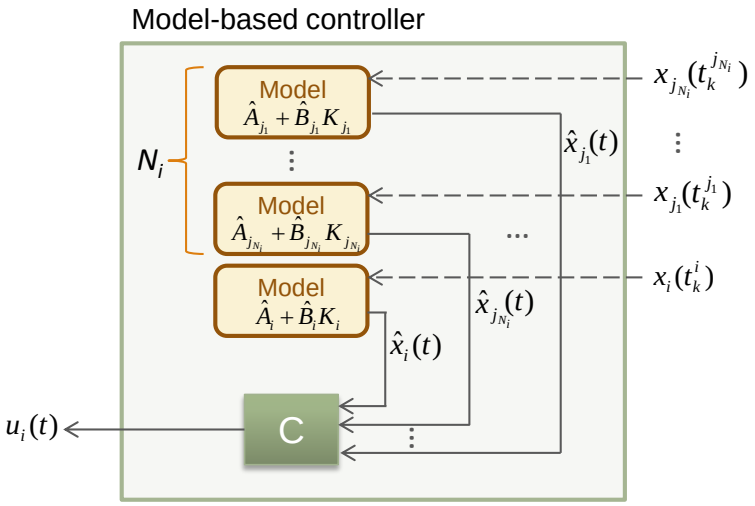

Figure 5.11: Model-based control scheme for the node i.

5.5 Model-based design

Model-based event-triggered control has been shown to reduce the amount of communication in a control loop [LL10]. Ideally, if the plant is stable, there are not 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 [Leh11].

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 Chapter 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

$$u_i(t) = K_i \hat{x}_i(t) + \sum_{j \in N_i} L_{ij} \hat{x}_j(t), \tag{5.56}$$

where $\hat{x}_i$ now represents the state estimation of x_i given by the model $(\hat{A}_i, \hat{B}_i)$ of each agent, and $\hat{A}_{K,i} = \hat{A}_i + \hat{B}_i K_i$. Let us define $\hat{A}_K = \text{diag}(\hat{A}_{K,1}, \ldots, \hat{A}_{K,n})$.

The error $e_i(t)$ is redefined as

$$e_i(t) = \hat{x}_i(t) - x_i(t), \tag{5.57}$$

and is reset at events' occurrence. In particular, $\hat{x}_i(t)$ is computed in the inter-event times as

$$\hat{x}_i(t) = e^{\hat{A}_{K,i}(t-t_k^i)} x_i(t_k^i), \quad \forall t \in [t_k^i, t_{k+1}^i). \tag{5.58}$$

Note that (5.58) 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 N_i. Based on this model, it estimates its state denoted as $\hat{x}_i(t)$ to compute $u_i(t)$ in (5.56). This idea is illustrated in Figure 5.11. Note that this is an extension of the model-based controller depicted in Figure 1.9. In the distributed approach the controller C has $N_i + 1$ inputs and one output. A block that represents the model of one subsystem is reset when a new broadcasted state is received.

When the state estimation $\hat{x}_i(t)$ 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, $\hat{x}_i$ 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.

Figure 5.12 shows an example of the previous idea. In previous sections, the control law of each agent i was computed based on the broadcasted measurements and it was a piecewise constant function. Now, the control is computed using a model-based approach in the inter-event times and is reset when a new event is triggered.

Remark 5.5. Note that $\hat{x}_i(t)$ is used instead of $x_i(t)$ in the control law (5.56) in order to preserve the property that the agent i and all its neighbors have the same version of $e_i(t)$. Alternatively, (4.24) can be redefined to deal with the aforementioned approach.

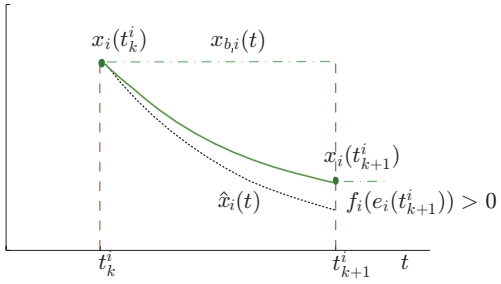

Figure 5.12: Comparison of model-based event-triggered control and Chapter 4 approach.

5.5.1 Main result

If we consider the trigger function defined in (4.26) and for the new error defined in (5.57), the state will be also bounded by (4.39). However, the lower bound for the inter-event time will have a different expression.

Definition 5.5. Let us define

$$\begin{aligned} \bar{A} &:= \hat{A} - A \\ \bar{B} &:= \hat{B} - B \\ \bar{A}_K &:= \hat{A}_K - A_K = \bar{A} + \bar{B}K, \end{aligned} \tag{5.59}$$

which represent the model uncertainty of the overall system without interconnections.

Assumption 5.3. We assume that the values of c_0 and c_1 and the initial conditions $x(0)$ satisfy the following constraint:

$$\frac{\sqrt{N}(c_0 + c_1)}{\|x(0)\| + \frac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta} + \frac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha}} < \kappa(V)\frac{\|A_K + \Delta\| - \|\bar{A}_K\| - \|\Delta\|}{\|\hat{A}_K\|}, \tag{5.60}$$

where $\lambda_\Delta = |\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|$.

Remark 5.6. Equation (5.60) is feasible only if the right hand side is strictly positive, since $c_0 + c_1 > 0$. 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 5.7. *If Assumption 5.3 holds, the lower bound of the broadcasting period for the system* (4.24)*, under the control law* (5.56)*, and with triggering functions* (4.26)*,* $0 < \alpha < |\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|$*, is greater than* (4.40)*.*

Proof. Define the overall system state estimation as $\hat{x} = (\hat{x}_1^T, \dots, \hat{x}_N^T)^T$. Let's 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 $\|e_i(t)\| \leq \int_{t^*}^{t} \|\dot{e}_i(s)\| ds$. In this interval, it also holds that

$$\|\dot{e}_i(t)\| = \|\dot{\hat{x}}_i(t) - \dot{x}_i(t)\| \leq \|\dot{\hat{x}}(t) - \dot{x}(t)\|.$$

Observe that

$$\begin{aligned}\dot{\hat{x}}(t) - \dot{x}(t) &= \hat{A}_K \hat{x}(t) - \big((A_K + \Delta)x(t) + BKe(t)\big) \\ &= (\bar{A}_K - \Delta)x(t) + (\hat{A}_K - BK)e(t).\end{aligned}$$

Then

$$\begin{aligned}\|\dot{e}_i(t)\| &\leq \|\bar{A}_K - \Delta\|\|x(t)\| + \|\hat{A}_K - BK\|\|e(t)\| \\ &\leq \|\bar{A}_K - \Delta\|\|x(t)\| + \|\hat{A}_K - BK\|\sqrt{N}(c_0 + c_1 e^{-\alpha t}) \qquad (5.61)\end{aligned}$$

Assume that $c_0, c_1 \neq 0$. It holds that $c_0 + c_1 e^{-\alpha t} \leq c_0 + c_1 e^{-\alpha t^*}$. As already stated, the bound on the state of Theorem 4.2 holds, and can be upper bounded as

$$\|x(t)\| \leq \kappa(V)\big(\|x(0)\|e^{-\lambda_\Delta t} + \tfrac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta} + \tfrac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha} e^{-\alpha t}\big).$$

Moreover, it holds that $\|\bar{A}_K - \Delta\| \leq \|\bar{A}_K\| + \|\Delta\|$. Thus, the error

$$\begin{aligned}\|e_i(t)\| \leq \int_{t^*}^{t} \|\dot{e}(s)\| ds \leq \Big(&(\|\bar{A}_K\| + \|\Delta\|)\kappa(V)\big(\|x(0)\|e^{-\lambda_\Delta t^*} + \tfrac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta} \\ &+ \tfrac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha} e^{-\alpha t^*}\big) + \|\hat{A}_K - BK\|\sqrt{N}(c_0 + c_1 e^{-\alpha t^*})\Big)(t - t^*), \qquad (5.62)\end{aligned}$$

It follows that $\|e_i(t)\| \leq (\hat{k}_{\Delta,1} + \hat{k}_{\Delta,2} + \hat{k}_{\Delta,3})(t - t^*)$, where

$$\begin{aligned}
\hat{k}_{\Delta,1} &= \kappa(V)\|x(0)\|(\|\bar{A}_K\| + \|\Delta\|) \\
\hat{k}_{\Delta,2} &= \big(\tfrac{\kappa(V)(\|\bar{A}_K\|+\|\Delta\|)\|BK\|}{\lambda_\Delta - \alpha} + \|\hat{A}_K - BK\|\big)\sqrt{N}c_1 \\
\hat{k}_{\Delta,3} &= \big(\tfrac{\kappa(V)(\|\bar{A}_K\|+\|\Delta\|)\|BK\|}{\lambda_\Delta} + \|\hat{A}_K - BK\|\big)\sqrt{N}c_0.
\end{aligned} \tag{5.63}$$

The next event will not occur before $\|e_i(t)\| = c_0 + c_1 e^{-\alpha t} \geq c_0$. This condition gives a lower bound for the broadcasting period

$$\hat{T}_{\Delta,min} = \frac{c_0}{\hat{k}_{\Delta,1} + \hat{k}_{\Delta,2} + \hat{k}_{\Delta,3}}, \tag{5.64}$$

that is larger than the lower bound in (4.40) if $\hat{k}_{\Delta,1} + \hat{k}_{\Delta,2} + \hat{k}_{\Delta,3} < k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3}$, which is equivalent to

$$\begin{aligned}
(\|\hat{A}_K - BK\| - \|BK\|)\sqrt{N}(c_0 + c_1) < (\|A_K + \Delta\| - \|\bar{A}_K\| - \|\Delta\|)\Big(\|x(0)\| \\
+ \tfrac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta} + \tfrac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha}\Big),
\end{aligned}$$

After some manipulations

$$\frac{\sqrt{N}(c_0 + c_1)}{\|x(0)\| + \frac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta} + \frac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha}} < \kappa(V)\frac{\|A_K + \Delta\| - \|\bar{A}_K\| - \|\Delta\|}{\|\hat{A}_K - BK\| - \|BK\|}. \tag{5.65}$$

The denominator on the right hand side can be bounded as:

$$\|\hat{A}_K - BK\| - \|BK\| \leq \|\hat{A}_K\| + \|BK\| - \|BK\| = \|\hat{A}_K\|.$$

Then if Assumption 5.3 holds, (5.65) is fulfilled. Thus, the lower bound for the broadcasting period is larger for the model-based approach. □

Remark 5.7. Note that in (5.65) it holds that $\|A_K + \Delta\| - \|\bar{A}_K\| - \|\Delta\| \leq \|A_K\| - \|\bar{A}_K\|$. Thus, if $\|\bar{A}_K\| \approx 0$, the right hand side of (5.65) can be approximated to $\kappa(V)$.

Example 5.6: Let us consider a system consisting of N identical subsystems whose closed

loop eigenvalues are $\{-1, -2\}$. Thus, $|\lambda_{max}(A_K)| = 1$. Assume that the coupling can be bounded as $\kappa(V)\|\Delta\| \leq 0.1|\lambda_{max}(A_K)|$ and that the parameters of the trigger functions are chosen so that

$$c_0 = 0.01|\lambda_{max}(A_K)|, \quad c_0 = 0.5|\lambda_{max}(A_K)|, \quad \alpha = 0.8|\lambda_{max}(A_K)|.$$

This yields $c_0/\lambda_\Delta = 0.011, c_1/(\lambda_\Delta - \alpha) = 5$.
Thus, the left hand side of (5.65) is $0.51|\lambda_{max}(A_K)|/_\kappa(V)(5.01\sqrt{N}\|BK\| + \|x(0)\|)$. If we approximate $\|\hat{A}_K\| \simeq \|A_K\|$, the constraint to the model uncertainty is

$$\|\bar{A}_K\| \leq \|\hat{A}_K\|\Big(1 - 0.51\frac{|\lambda_{max}(A_K)|}{\kappa(V)(5.01\sqrt{N}\|BK\| + \|x(0)\|)}\Big). \tag{5.66}$$

The first constraint for the feasibility of (5.66) is that the right hand side is positive, which imposes some conditions on the initial values. Then, for given values of $\kappa(V), \|BK\|, \|x(0)\|$ and $\hat{A}_K$, the upper bound on $\|\bar{A}_K\|$ can be computed.

5.5.2 Simulation results

Next, the performance of the model-based approach is demonstrated and compared to the results of Section 4.5.3. Let us consider trigger functions $f_i(e_i(t)) = 0.02 + 0.5e^{-0.8t}$. Figure 5.13 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.

The results of Table 4.2 compared the mean and minimum inter-event times of the approach of Chapter 4 and the approach proposed in [WL08] for a set of values in the number of subsystems. Table 5.5 extends those results with the model-based design.

Note that when the controller uses a model, the average and the minimum value of the inter-event times are enlarged, as predicted by Theorem 5.7.

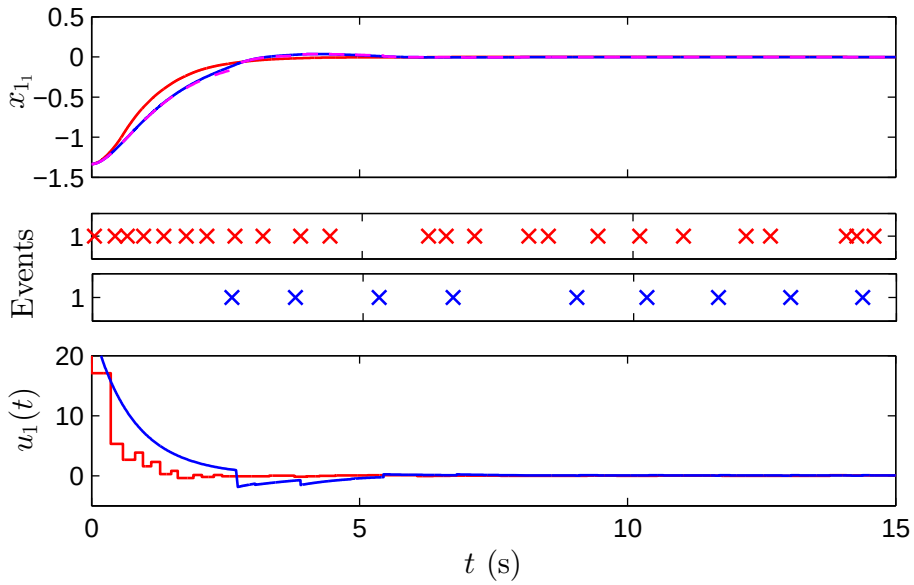

Figure 5.13: Simulation result with trigger functions (4.26) for the design of Chapter 4 (red) and the distributed model-based control (blue). The dashed line (magenta) represents the piecewise function $\hat{x}_{1,1}$.

Table 5.5: Inter-event times for different N.

		N	10	50	100	150	200
Trigger condition	$\{T_k^i\}_{\min}$ (s)		0.053	0.031	0.015	0.019	0.009
(4.26), Chapter 4	$\{T_k^i\}_{\text{mean}}$ (s)		0.565	0.565	0.567	0.572	0.568
Trigger condition	$\{T_k^i\}_{\min}$ (s)		0.6816	0.3025	0.219	0.0963	0.132
(4.26), MB control	$\{T_k^i\}_{\text{mean}}$ (s)		1.430	1.500	1.477	1.668	1.581
Trigger condition [WL08]	$\{T_k^i\}_{\text{mean}}$ (s)		0.1149	0.1175	0.1152	0.1180	0.1177

5.6 Conclusions

This chapter has presented an extension of the distributed control design of Chapter 4 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.

Additionally, it has been illustrated how the actuation rate can be reduced in interconnected system if triggering functions are used also in the update of the

control law. The existing trade-off between communication and actuation have 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.

6

Simulation Tools and Application Example of the DEBC: Networked Mobile Robots

Summary

The formation control of networked mobile robots is an example of multi-agent systems in which a group of robots achieve a common objective (the formation) by applying distributed control laws and event-based communications. This chapter gives a description of the problem, some of most common approaches to model these systems, and how the distributed event-triggered policies can be useful to reduce communication. An interactive simulator named *MaSS* (Multi-agent Systems Simulator) has been developed to emulate this kind of setups. The user interface of this tool is described, and the software implementation of the real counterparts is given. Some examples of usage are given to illustrate how the network and the model of the system can be configured interactively by the user. The DEBC algorithms have been also implemented in a testbed of real mobile robots, and the results are presented.

6.1 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 intercon-

nections, which is also an interesting field to apply the decentralized event-trigger 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 value. 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 [OSFM07]. 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 [RBA07], 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 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.

Though the problem of multi-agent systems with event-based communications have been recently addressed [DL12, SDJ13], 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 evalua-

tion of the cited communication strategies has been not carried out so far in an experimental platform.

The outline of the chapter is as follows. Section 6.3 presents an overview of the formation control for networked mobile robots. The description of the developed simulator MaSS is given in Section 6.4. The description of the testbed over which the DEBC algorithms have been implemented, and the obtained experimental results are presented.

6.2 Contributions of this chapter

The simulator described in this chapter 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 Chapter 4 as well as periodic communications are implemented. This implementation has been also carried out over a testbed of mobile robots, showing the benefits of using event-driven communications.

6.3 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 that, the formation control is studied as an extension of the consensus problem. The model for non-holonomic 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.

6.3.1 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 [OSM04], a simple consensus algorithm to reach an agreement regarding the state of N single integrator agents of the form $\dot{x}_i(t) = u_i(t)$ can be expressed as an n^{th} order linear system on a graph

$$\dot{x}_i(t) = \sum_{j \in N_i} (x_j(t) - x_i(t)). \tag{6.1}$$

The dynamics of the group of agents can be written as

$$\dot{x}(t) = -Lx(t) \tag{6.2}$$

where L is the Laplacian graph of the network (or the communication graph) and its elements are defined as follows

$$l_{ij} = \begin{cases} -1 & \text{if } j \in N_i \\ |N_i| & \text{if } j = i, \end{cases} \tag{6.3}$$

where $|N_i|$ 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 $|N_i|$, so that $L = D - A$.

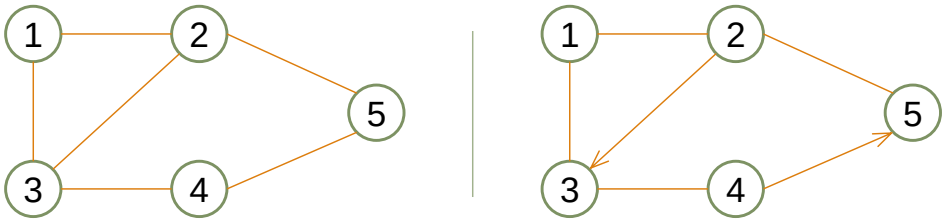

Figure 6.1: Examples of a) undirected and b) directed graphs.

Example 6.1: Assume a five node multi-agent network with the communication graph depicted in Figure 6.1a. 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

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

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 (6.2) of the form $x_{eq} = \alpha \mathbf{1}$, where $\alpha = \frac{1}{N} \sum_{i=1}^{N} x_i(0)$ and $\mathbf{1} = (1 \dots 1)^T$ [OSFM07].

The above results hold for undirected graphs, i.e., for bidirectional communications: for any pair $(i, j) \in \mathcal{E}$ there is another edge $(j, i) \in \mathcal{E}$. In a directed communication graph, there is at least one pair of nodes whose communication is unidirectional. An example is given in Figure 6.1b. 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, 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 [OSFM07].

There are also in the literature extensions regarding the agents dynamics [RBA07, RA07, SSB09, NC10]. 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 [RA07] is given by

$$u_i(t) = \sum_{j \in N_i} (x_{j,1}(t) - x_{i,1}(t)) + \gamma(x_{j,2}(t) - x_{i,2}(t)), \tag{6.4}$$

where $\gamma > 0$.

For connected and undirected graphs, [RA07] 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 $v_i = \frac{1}{N} \sum_{j \in N_i} x_{i,2}(0)$, and

$$\lim_{t \to \infty} x_{i,1}(t) = \frac{1}{N} \sum_{i=1}^{N} x_{i,1}(0) + \frac{t}{N} \sum_{i=1}^{N} x_{i,2}(0).$$

In [RMC06], results for n^{th} order consensus are given, and [SSB09, NC10] study the consensus when the dynamics of each agent is an n^{th} 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

$$u_i(t) = K \sum_{j \in N_i} (x_j(t) - x_i(t)).$$

6.3.2 Formation Control

Last years, the formation control problem of multi-vehicles 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 [OSFM07]. 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 Figure 6.2). For single integrator agents, the following control law

$$u_i(t) = \sum_{j \in N_i} (x_j(t) - x_i(t) - \mathbf{r}_{ij}), \tag{6.5}$$

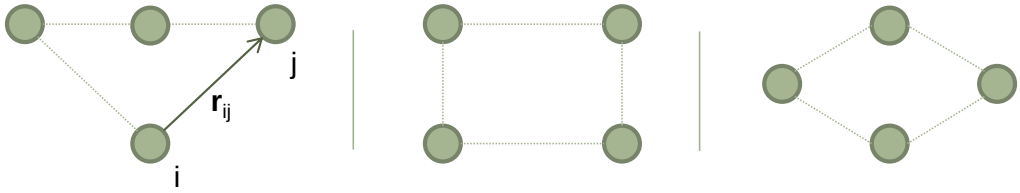

Figure 6.2: Examples of formations of four agents in the plane.

yields the group to achieve the objective of the formation [FM04].

According to (6.5), the overall system dynamics is given by

$$\dot{x}(t) = -Lx(t) - \mathbf{r}, \tag{6.6}$$

where $\mathbf{r}_i = \sum_{j \in N_i} \mathbf{r}_{ij}$, $i = 1, \ldots, N$.

There are some extensions of the protocol above regarding agents dynamics. For instance, in [LWCV05] the vehicles 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 Figure 6.2. In this case, $\mathbf{r}_{ij}$ are the (x, y) coordinates of the desired distance between nodes.

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 that agent, which is called the *leader* of the formation. This problem is known as *leader-follower* consensus [LDH09]. 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 $\mathcal{G}$ 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 [FM04, HH07, NC10] 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

$$u_i(t) = \sum_{j \in N_i} (x_j(t) - x_i(t) - \mathbf{r}_{ij}) + b_i(x_0(t) - x_i(t) - \mathbf{r}_{i0}). \tag{6.7}$$

6.3.3 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 less than the total degrees of freedom the vehicle is non-holonomic.

The non-holonomic model of a mobile robot is depicted in Figure 6.3. 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 [RA07]

$$\begin{aligned} \dot{x}_i &= v_i \cos\theta_i \\ \dot{y}_i &= v_i \sin\theta_i \\ \dot{\theta}_i &= \omega_i \\ m_i \dot{v}_i &= f_i \\ J_i \dot{\omega}_i &= \tau_i \end{aligned} \tag{6.8}$$

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, τ_i is the torque, m_i is the mass, and J_i is the mass moment of inertia.

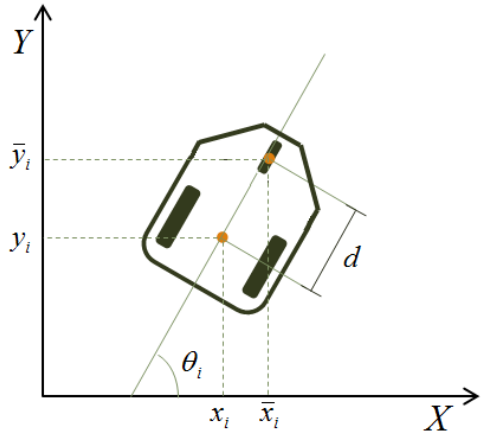

Figure 6.3: Non-holonomic mobile robot.

To avoid the non-holonomic constraint introduced by (6.8), let us define

$$\begin{aligned} \bar{x}_i &= x_i + d\cos\theta_i \\ \bar{y}_i &= y_i + d\sin\theta_i. \end{aligned} \tag{6.9}$$

according to Figure 6.3.

As proposed in [LBY03], the dynamics of the mobile robot can be reformulated in terms of these coordinates as

$$\begin{pmatrix} \dot{\bar{x}}_i \\ \dot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix} \begin{pmatrix} v_i \\ \omega_i \end{pmatrix}. \tag{6.10}$$

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

$$\begin{pmatrix} \ddot{\bar{x}}_i \\ \ddot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} -v_i\omega_i\sin\theta_i - d\omega_i^2\cos\theta_i \\ v_i\omega_i\cos\theta_i - d\omega_i^2\sin\theta_i \end{pmatrix} + \begin{pmatrix} \frac{1}{m}\cos\theta_i & -\frac{d}{J}\sin\theta_i \\ \frac{1}{m}\sin\theta_i & \frac{d}{J}\cos\theta_i \end{pmatrix} \begin{pmatrix} f_i \\ \tau_i \end{pmatrix}, \tag{6.11}$$

where f_i and τ_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 that, in the next pages a control law is de-

signed for the first order model (6.10). After that, the example proposed in [RA07, Sey10] to control the second-order model (6.11) is presented.

Formation control of first-order non-holonomic mobile robots

The control law $u_i = (v_i, \omega_i)$ in (6.10) to reach the desired formation is

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{j\in N_i}(\bar{x}_j - \bar{x}_i - (r_{x,j} - r_{x,i})) \\ \sum_{j\in N_i}(\bar{y}_j - \bar{y}_i - (r_{y,j} - r_{y,i})) \end{pmatrix}, \tag{6.12}$$

where $(r_{x,i}, r_{y,i})$ are the predefined relative position offsets with respect to the formation center. From (6.10) and (6.12), it follows that

$$\begin{pmatrix} \dot{\bar{x}}_i \\ \dot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} \sum_{j\in N_i}(\bar{x}_j - \bar{x}_i - (r_{x,j} - r_{x,i})) \\ \sum_{j\in N_i}(\bar{y}_j - \bar{y}_i - (r_{y,j} - r_{y,i})) \end{pmatrix}. \tag{6.13}$$

If stack vectors for the overall system are defined as $\bar{x}^T = (\bar{x}_1 \dots \bar{x}_N)$, $\bar{y}^T = (\bar{y}_1 \dots \bar{y}_N)$, $r_x^T = (r_{x,1} \dots r_{x,N})$, and $r_y^T = (r_{y,1} \dots r_{y,N})$, it yields

$$\begin{pmatrix} \dot{\bar{x}} \\ \dot{\bar{y}} \end{pmatrix} = -\begin{pmatrix} L & \mathbf{0} \\ \mathbf{0} & L \end{pmatrix} \begin{pmatrix} \bar{x} - r_x \\ \bar{y} - r_y \end{pmatrix}. \tag{6.14}$$

Note that the control law (6.12) decouples the system, giving an equivalent results to (6.6) for single integrators. The group of robots reaches the formation and the center of this formation is the average of the robots initial positions.

Formation control of second-order non-holonomic mobile robots

According to [RA07], the following feedback linearization can be used in order to transform the dynamics (6.11) to two decoupled double-integrators

$$\begin{pmatrix} f_i \\ \tau_i \end{pmatrix} = \begin{pmatrix} \frac{1}{m}\cos\theta_i & -\frac{d}{J}\sin\theta_i \\ \frac{1}{m}\sin\theta_i & \frac{d}{J}\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} v_i\omega_i\sin\theta_i + d\omega_i^2\cos\theta_i + \bar{f}_i \\ -v_i\omega_i\cos\theta_i + d\omega_i^2\sin\theta_i + \bar{\tau}_i \end{pmatrix}, \tag{6.15}$$

which yields to

$$\begin{pmatrix} \ddot{\bar{x}}_i \\ \ddot{\bar{y}}_i \end{pmatrix} = \begin{pmatrix} \bar{f}_i \\ \bar{\tau}_i \end{pmatrix}.$$

The control law (6.4) can be extended to the formation control problem, giving the following coordination rule for the group of mobile robots

$$\begin{pmatrix} \bar{f} \\ \bar{\tau} \end{pmatrix} = - \begin{pmatrix} L & \gamma_x L & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & L & \gamma_y L \end{pmatrix} \begin{pmatrix} \bar{x} - r_x \\ \dot{\bar{x}} \\ \bar{y} - r_y \\ \dot{\bar{y}} \end{pmatrix}, \tag{6.16}$$

where $\gamma_x, \gamma_y > 0$, $\bar{f} = (\bar{f}_1 \dots \bar{f}_N)^T$, and $\bar{\tau} = (\bar{\tau}_1 \dots \bar{\tau}_N)^T$.

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.

6.3.4 Time-Schedule Control

The formation control laws (6.12) and (6.16) have been proposed for 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 [SDJ13, DL12] can be adapted to the formation control problem by redefining the previous control laws as a function of last broadcasted states:

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -d\sin\theta_i \\ \sin\theta_i & d\cos\theta_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{j\in N_i} (\bar{x}_{b,j} - \bar{x}_{b,i} - (r_{x,j} - r_{x,i})) \\ \sum_{j\in N_i} (\bar{y}_{b,j} - \bar{y}_{b,i} - (r_{y,j} - r_{y,i})) \end{pmatrix} \tag{6.17}$$

$$\begin{pmatrix} \bar{f}_i \\ \bar{\tau}_i \end{pmatrix} = \begin{pmatrix} \sum\limits_{j\in N_i} (\hat{\bar{x}}_j - \hat{\bar{x}}_i - (r_{x,j} - r_{x,i}) + \gamma_x(\dot{\bar{x}}_{b,j} - \dot{\bar{x}}_{b,i})) \\ \sum\limits_{j\in N_i} (\hat{\bar{y}}_j - \hat{\bar{y}}_i - (r_{y,j} - r_{y,i}) + \gamma_y(\dot{\bar{y}}_{b,j} - \dot{\bar{y}}_{b,i})) \end{pmatrix}, \tag{6.18}$$

where

$$\hat{\bar{x}}_i = \bar{x}_{b,i} + (t - t_k^i)\dot{\bar{x}}_{b,i}$$
$$\hat{\bar{y}}_i = \bar{y}_{b,i} + (t - t_k^i)\dot{\bar{y}}_{b,i},$$

and $\bar{x}_{b,i}, \bar{y}_{b,i}, \dot{\bar{x}}_{b,i}$, and $\dot{\bar{y}}_{b,i}$ are the last broadcasted 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 (6.18), as proposed in [Sey10]. 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

$$e_i = \begin{pmatrix} e_{x,i} \\ e_{y,i} \end{pmatrix} = \begin{pmatrix} \bar{x}_{b,i} - \bar{x}_i \\ \bar{y}_{b,i} - \bar{y}_i \end{pmatrix}$$

for first-order dynamics, and for second-order systems as

$$e_i = \begin{pmatrix} e_{x,i} \\ \gamma_x e_{\dot{x},i} \\ e_{y,i} \\ \gamma_y e_{\dot{y},i} \end{pmatrix} = \begin{pmatrix} \hat{\bar{x}}_i - \bar{x}_i \\ \gamma_x(\dot{\bar{x}}_{b,i} - \dot{\bar{x}}_i) \\ \hat{\bar{y}}_i - \bar{y}_i \\ \gamma_y(\dot{\bar{y}}_{b,i} - \dot{\bar{y}}_i) \end{pmatrix}.$$

6.3.5 Robot Wireless Communication Protocols

In early robot wireless communications, infrared technology was applied in a large scale because of its low cost. But 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

Table 6.1: 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

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 6.1. 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 communication with their near neighbors. Hence, it is virtually impossible for each node to know the entire network topology at a given time [WLZ05].

Moreover, many in the field on networking argued that Internet protocols were not convenient to achieve robustness and scalability for such distributed architecture [KDHH$^+$11], 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) [PR97], Dynamic Source Routing (DSR) [JM96], while Low Energy Adaptive Clustering Hierarchy (LEACH) [HCB02] is a cluster-based protocol that includes distributed cluster formation and a hierar-

chical clustering algorithm. Finally, Routing Protocol for Low power and lossy networks (RPL) [WTB+12] is a IP-based protocol for this kind of networks.

Three important features usually serve to evaluate the performance of a protocol [Mar10]:

- *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.

6.4 MaSS: Multi-agent Systems Simulator

The many control and system configuration options needed to simulate a multi-agent system demand Graphical User Interfaces (GUI) 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 [Esq04]. 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 that, the simulator is described, including the GUI and the system modeling with Ejs.

6.4.1 Existing tools

Most of the simulators for the formation control of a team of networked vehicles/robots use different software 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 [VH08] is an open-source Amigobot and Pioneer simulator [Mob13], also provided by Mobile Robots Inc. It has a customizable interface for users to design and simulate different models of MobileRobots/ActivMedia robots [Mob13]. 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 [NPEJ07] the dynamics and the control algorithm are implemented in Simulink, and ns-2 [ntNS12] is used for the simulation of the network. In [Mas11], 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 [VAP08] 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.

6.4.2 Description of the GUI

The user interface of the MaSS simulator is shown in Figure 6.4. 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.

The agents connected by links are known as neighbors. By default, the links provide bidirectional communication, although one way communication is also allowed. So, in Figure 6.4, 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

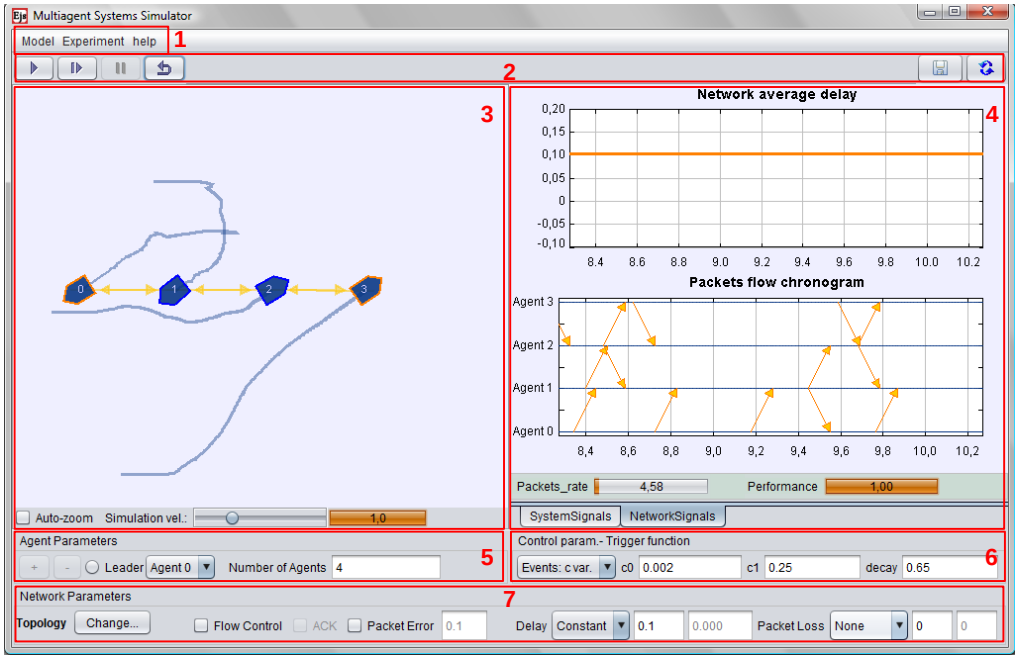

Figure 6.4: View of the GUI.

allows speeding up or down the simulation by dragging the slider *Simulation vel.*

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 Figure 6.4), 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 by 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 provides 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 in order to perform further analysis of a simulation.

6.4.3 Modeling a multi-agent system in Ejs

The model of each node in the multi-agent system is basically the same described in Chapter 4 (Figure 6.5) for interconnected systems, but specifically assuming that the information is sent through the network in packets of a given structure, which is detailed later.

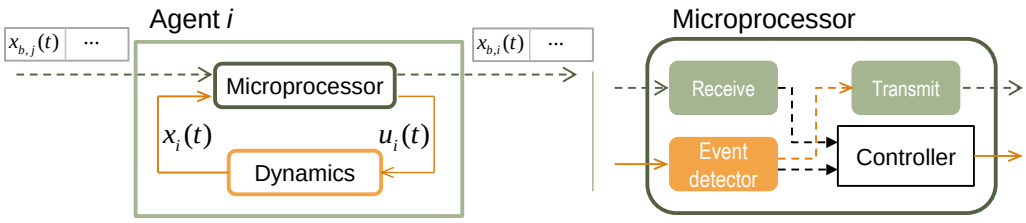

Figure 6.5: Scheme of one node.

Hence, the simulator has to implement:

- The system dynamics, including the agents 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.

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 6.2: Figure 6.6 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 (6.10). The page for the second order model (6.11) 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

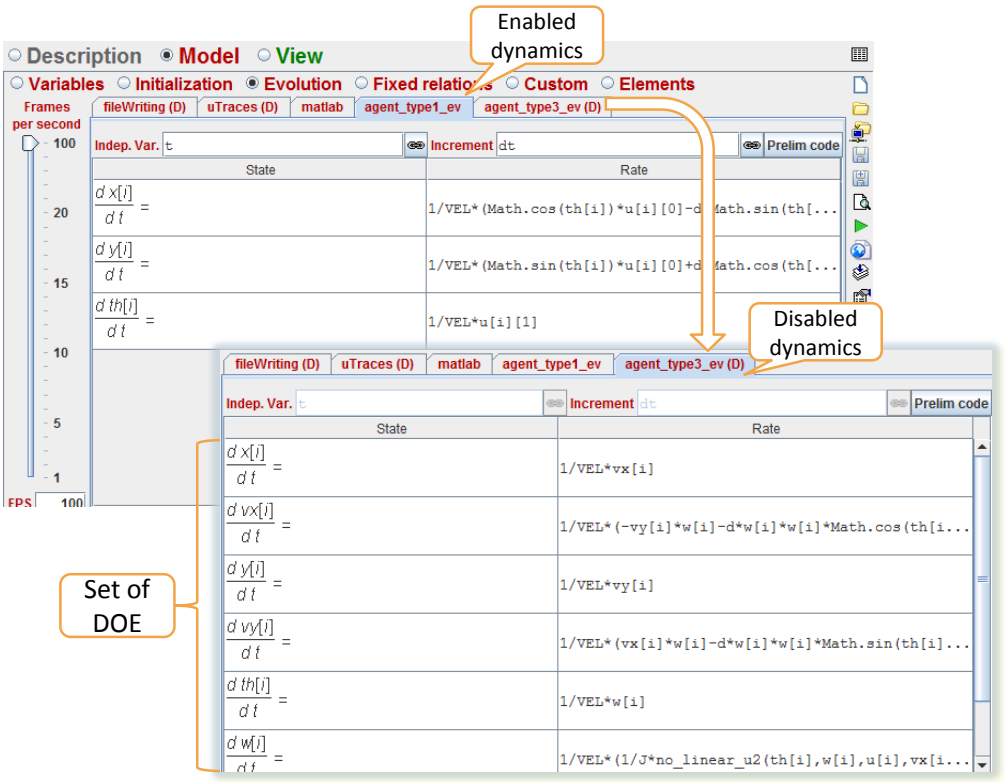

Figure 6.6: Screenshot of *Evolution pages* in Ejs.

a default value and it is updated when a new experiment is selected or new links are added/removed by the user.

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 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 6.1 and 6.2 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, `tol=0` when `e[i]` ($\|e_i\|$) reaches the threshold `c`. Note that in the periodic case the event to detect is the time instance that equals the next sampling time (line 11 in Listing 6.1), where `incr` is an internal count and `tpo`

Listing 6.1: Code to detect events.

```
1 /* Event-triggering */
2 if(control_type.equals("Events: c cte")||control_type.equals("Events: c var."))
3 {
4   c=(c0+c1*Math.exp(-alpha*(t-ti)));
5   e[i]=computeError(x_b[i][i],x[i],y_b[i][i],y[i]);
6   tol=c-e[i];
7 }
8 /* Periodic sampling. Parameter c is reconverted to Ts */
9 else {
10     c=c0;
11     tol=incr*c-(t-tpo);
12 }
13 return tol;
```

is the time at which the periodic sampling started.

The broadcasted 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 broadcasted 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., (6.12) for the first order model and (6.15) for the second order model. The update of the control law is performed only at event times (line 3 in Listing 6.2) 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.

The structure of a packet is shown in Figure 6.7. 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 MaSS 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 Figure 6.7.

Moreover, if the *Acknowledgment* signal is required, the receiver sends an ACK packet to confirm the reception. The ACK packets are assumed to be

Listing 6.2: 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;
```

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.

The network

The MaSS 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 6.3. 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 broadcasted 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

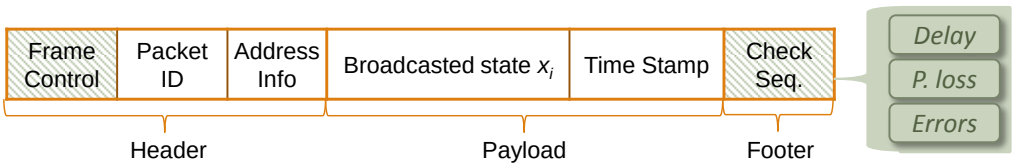

Figure 6.7: Structure of a data packet.

Listing 6.3: 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 }
```

receptions are simultaneous, preserving the distributed architecture of the system.

Listing 6.4 shows extracts of the functions `getNextPacketTime()` (line 5 in Listing 6.3) and `getPacketsToAttend()` (line 11 in Listing 6.3). 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.

6.4.4 Using MaSS

We next describe some examples of how to use MaSS.

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 Figure 6.4 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 $f_i(e_i(t)) = \|e_i(t)\| - 0.002 - 0.25e^{-0.65t}$, which means that the threshold to trigger the events is not constant and decrease as time increases. 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 Figure 6.4 reflects

Listing 6.4: 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;
}
```

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.

Experiment 2: Directed graph, random delay, packet losses, second order model

Let us change the topology of the network removing the link $1 \to 2$ and adding the links $1 \to 3$, $3 \to 0$. 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

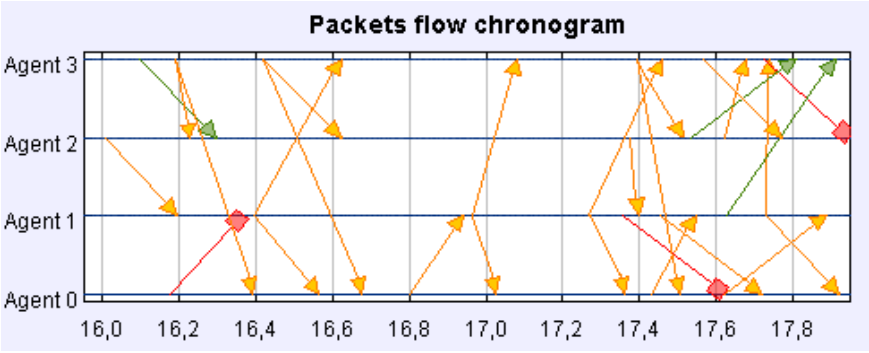

Figure 6.8: Example of chronogram. Delivered packets are in orange, red arrows are lost packets, and green arrows correspond to discarded packets.

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 non-holonomic model, and select a different experiment to change the desired formation. Figure 6.8 shows the packets flow chronogram in a time slot.

The red arrows correspond to lost packets. For example, at time 17.35 s, the 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, the agent 1 does not transmit information to agent 2 (as in Figure 6.4), 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 Figure 6.8, agent 2 discards one packet from agent 3 at time 16.1 s because it has already received a measured from 3 which was taken later on time. Other packets are discarded at time 17.54 s and 17.62 s.

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 communication point of view means that it sends its state to its neighbors but it

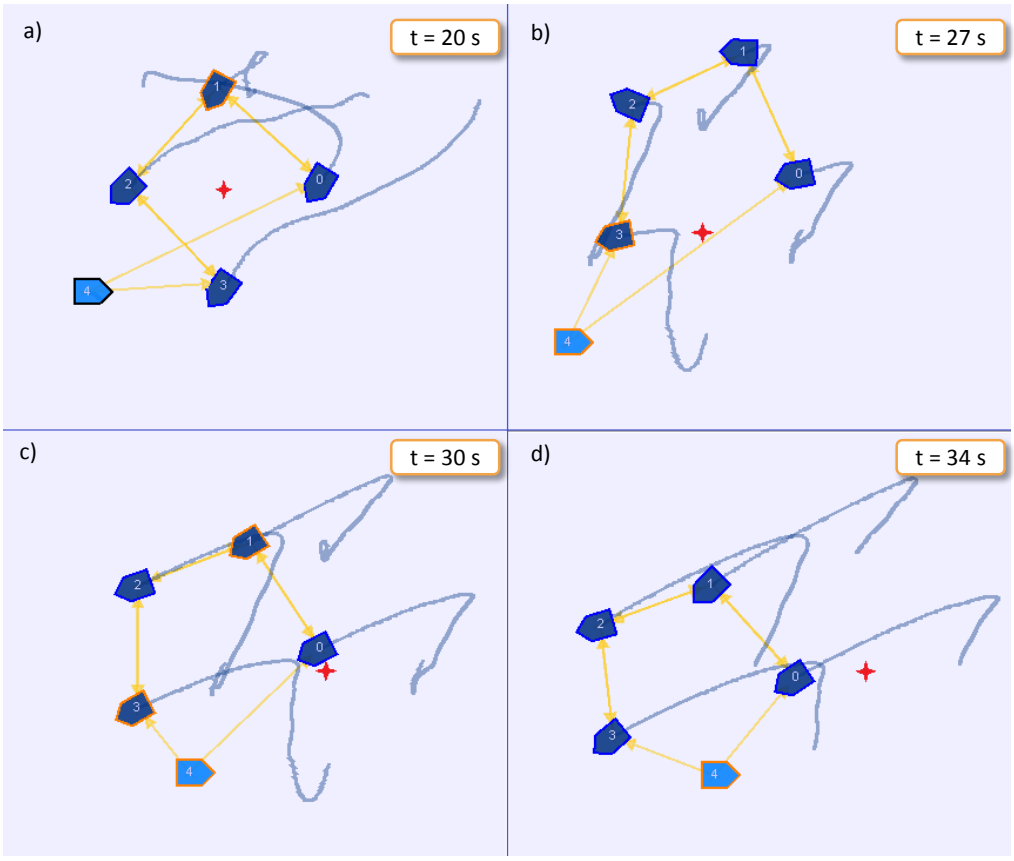

Figure 6.9: Example of experiment with a leader at different instants of time.

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 Figure 6.9a). Figures 6.9b, 6.9c, and 6.9d 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.

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

afterwards. So, as an example let us consider the following scenario:

- Agents' model: 4 first-order non-holonomic vehicles.
- Desired formation: A square of 0.85 u. side.
- Communication method: Event-triggered with constant threshold $c_0 = 0.025$.
- 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 Figure 6.10 and 6.11. The trajectories of the agents for both cases are depicted in Figure 6.10. 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 Figure 6.11a 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 (Figure 6.11c). The network performance is defined as the ratio of delivered to sent packets.

The packet transmission rate is shown in 6.11b. One can conclude that the degradation of the performance of the network has a direct effect into the performance of the system.

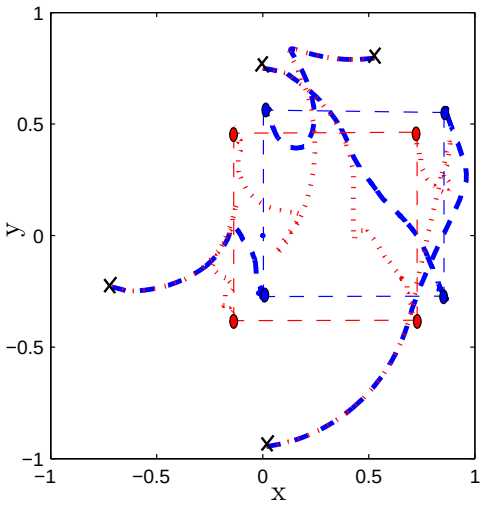

Figure 6.10: Matlab figure corresponding to the trajectories of the agents in the experiment 4.

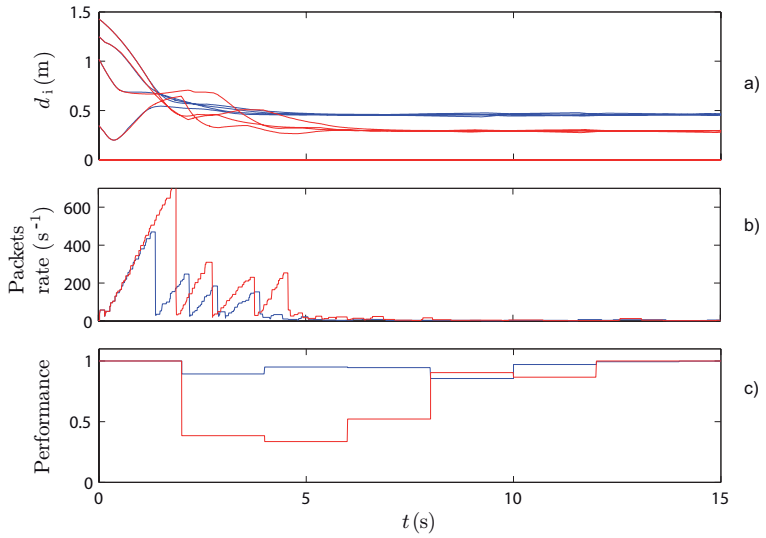

Figure 6.11: Distance to the formation, packets rate, and performance corresponding to experiment 4 (Matlab figure).

6.5 Application example to a real testbed

The DEBC has been implemented in a testbed of real mobile robots. The prototypes taken for experimentation are the *Moway* robots [mOw13], 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.

6.5.1 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 [Fab13].

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 [mOw13]. Another important component is the RF (Radio Frequency) 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.

Measurement and communication systems

Measurement and communication systems are depicted in Figure 6.12. 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 position and orientation. 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 [CSdM$^+$06, LRCC08]. Hence, the values of x_i, y_i and θ_i are determined by this soft-

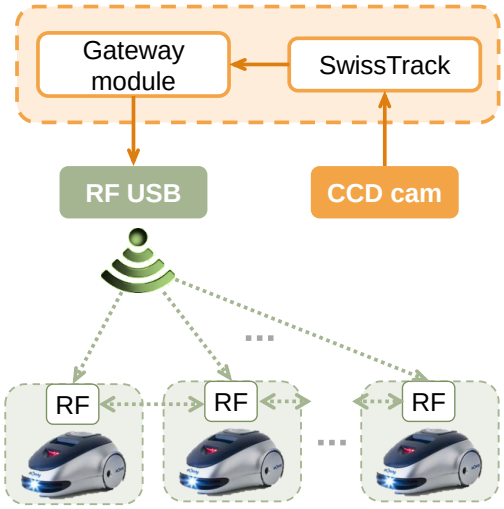

Figure 6.12: Experimental framework block diagram. Dotted lines represent wireless communications, and the exchange of information between hardware and software components is symbolized by solid lines.

ware, which processes the 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).

6.5.2 Experimental results

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 Figure 6.13. 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 [OSFM07]. The initial

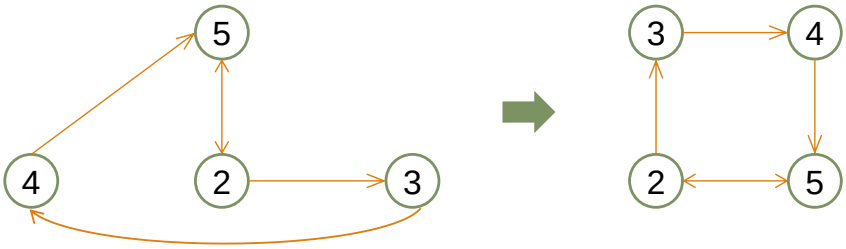

Figure 6.13: Scheme of the communication topology, initial formation (left) and desired formation (right).

conditions are:

$$x(0) = (0 \quad 60 \quad -50 \quad 0)^T$$
$$y(0) = (0 \quad 0 \quad 0 \quad 40)^T$$
$$\theta(0) = (198 \quad 280 \quad 262 \quad 179)^T,$$

and the desired inter-vehicle relative-position vector $\mathbf{r} = (r_x, r_y)$ is

$$r_x = (-20 \quad -20 \quad 20 \quad 20)^T$$
$$r_y = (-20 \quad 20 \quad 20 \quad -20)^T. \tag{6.19}$$

The control law is computed according to (6.17).

TIME-TRIGGERED VS. EVENT-TRIGGERED COMMUNICATIONS. The experiment described above is performed with time-triggered communications and with event-triggered communications. The period is set to $T_s = 250$ 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 $c_0 = 4$ 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 Figure 6.14. 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

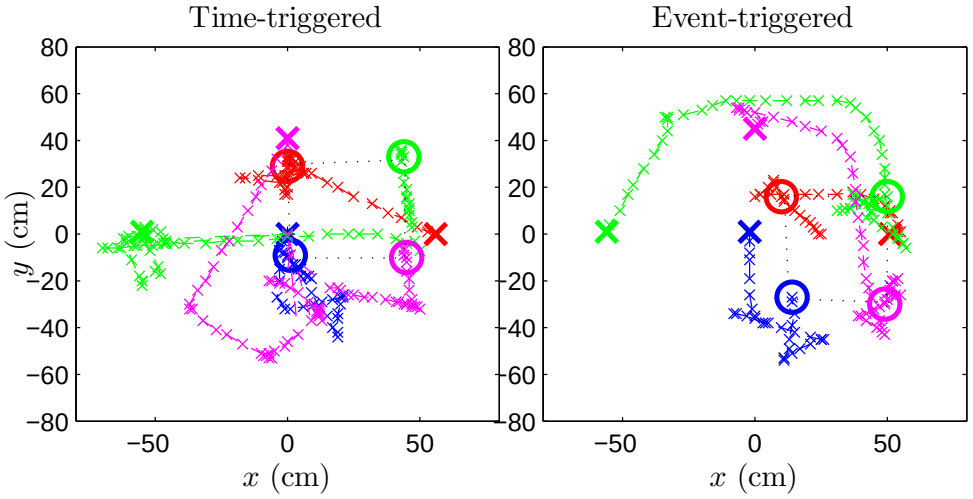

Figure 6.14: Representation in the plane of the trajectories of each robot for time-triggered (left) and event-triggered (right) communications and consensus protocols. Agent 2 (blue), agent 3 (red), agent 4 (green), and agent 5 (magenta). The initial an final positions are marked with crosses and circles of the same color, respectively.

the 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 Figure 6.15. Note that the formation is almost reached at $t = 10$ 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 6.16 depicts d_i over time. For the event-based case, d_i is almost equal for the four agents at $t = 11.5$ 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 cases, the goodness of the event is highlighted. The number of events is summarized in Table 6.2. 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

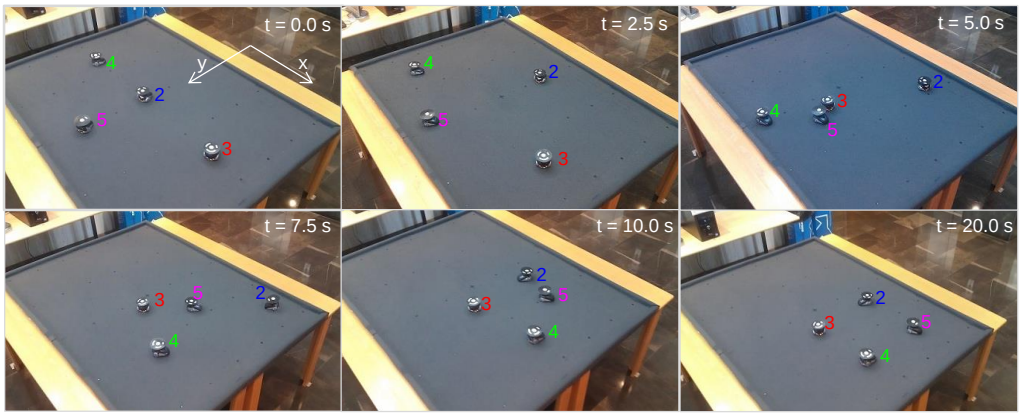

Figure 6.15: Shots of the consensus protocol experiment with event-triggered communications at six instants of time.

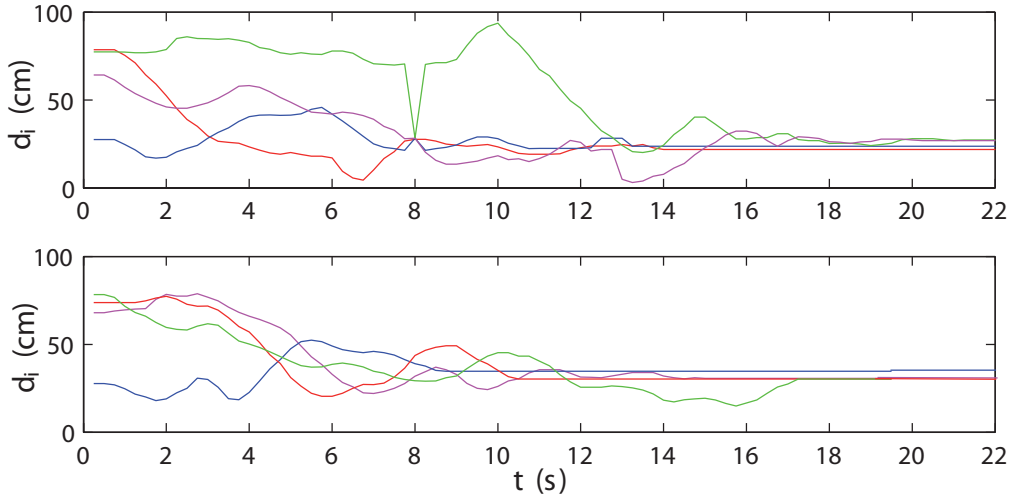

Figure 6.16: Distance to the formation over time for time-triggered (above) and event-triggered (below) communications and consensus protocols. Agent 2 (blue), agent 3 (red), agent 4 (green), and agent 5 (magenta).

for the periodic approach is

$$(\frac{t_f - t_0}{T_s}+1)\times(\text{No. robots}+\text{No. links}) = 89\times(4+5) = 801 \text{ transmissions}, \quad (6.20)$$

where $t_0 = 0$ s and $t_f = 22$ s.

Energy consumption. The number of transmissions is related to the energy consumption. The RF module of the Moway robots has the following characteristics [mOw10]:

- Transmission current I_t: 11.3 mA

Table 6.2: 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

- 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 have 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:

$$\begin{aligned} E_{ET} &= V_{RF}\delta t_{RF}(\text{No. events} \times I_r + \text{No. broadcasts} \times (I_r + I_t)) \\ &= 2.75[\text{V}] \cdot 0.01[\text{s}] \cdot (165 \cdot 0.123[\text{A}] + 191 \cdot (0.123[\text{A}] + 0.113[\text{A}])) = 1.80[\text{J}]. \end{aligned}$$

According to the aforesaid and to (6.20), 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

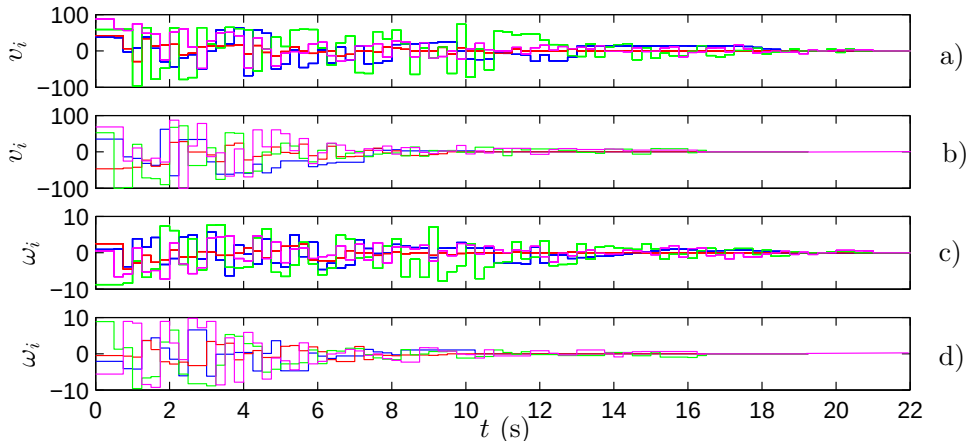

Figure 6.17: Control signals: a) v_i time-triggered, b) v_i event-triggered, c) ω_i time-triggered, d) ω_i event-triggered approaches. Agent 2 (blue), agent 3 (red), agent 4 (green), and agent 5 (magenta).

actuation? The evolution of control law values is depicted in Figure 6.17. The values of v_i and w_i obtained from (6.17) 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:

$$W_{v_i} = \int_{t_0}^{t_f} |v_i(t)|dt \tag{6.21}$$

$$W_{\omega_i} = \int_{t_0}^{t_f} |\omega_i(t)|dt. \tag{6.22}$$

The results are summed up in Table 6.3. 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.

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 Figure 6.18, and the robot 2 is the leader. It computes its control law to reach

Table 6.3: W_{v_i} and W_{ω_i} with event-based and periodic communications for each robot.

	Periodic		Event-based	
Robot	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

Table 6.4: 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

its desired position $[-20, -20]$ as

$$\begin{pmatrix} v_2 \\ \omega_2 \end{pmatrix} = \begin{pmatrix} \cos\theta_2 & -d\sin\theta_2 \\ \sin\theta_2 & d\cos\theta_2 \end{pmatrix}^{-1} \begin{pmatrix} -20 - \bar{x}_{b,2} \\ -20 - \bar{y}_{b,2} \end{pmatrix}.$$

The experiment is performed with time-triggered communications and with event-triggered communications, and the results comparative is illustrated in Figure 6.19. The consensus is not longer reached due to the existence of a leader, and the final positions are equal to the desired inter-vehicle relative-position vector $\mathbf{r}$ (6.19). In the event-based approach, the number of events for each robot accounts for 9, 21, 37, and 48 (see Table 6.4), whereas in the periodic case the number of executions of the measurement task is 73 for each robot. This yields

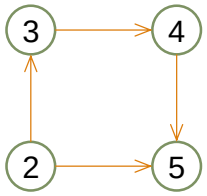

Figure 6.18: Scheme of the communication topology, initial formation (left) and desired formation (right).

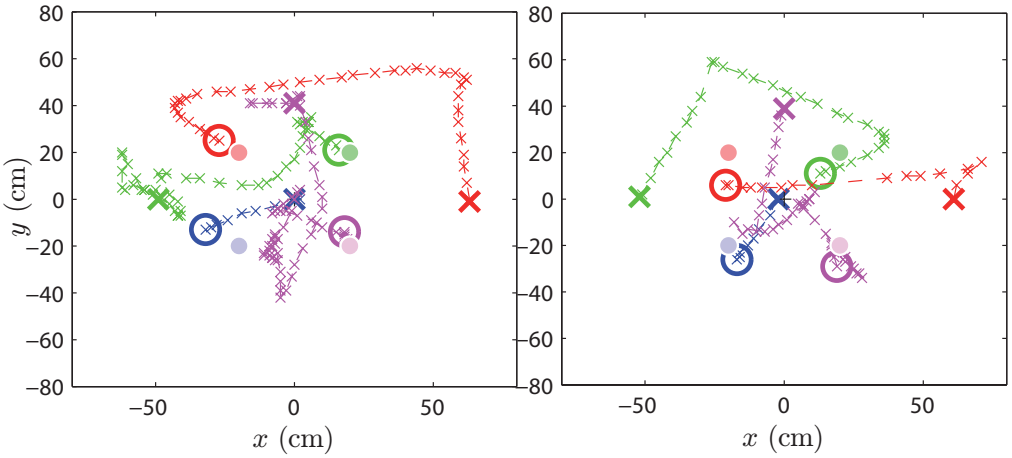

Figure 6.19: Representation in the plane of the trajectories of each robot for time-triggered (left) and event-triggered (right) communications. Agent 2 (blue), agent 3 (red), agent 4 (green), and agent 5 (magenta). The initial an final positions are marked with crosses and circles of the same color, respectively. The desired formation is represented by the circles in light colors.

a total number of transmissions of 191 for the event-based case, and 584 for the periodic case (computed as in (6.20)). Similar conclusions can be extracted for the energy consumption as in the consensus protocol.

Note that the robots does 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 2 does not receive its position from the measurement system (see Figure 6.19 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.

6.6 Conclusions

The formation control of mobile robots has been presented as an application example in which the use of DEBC 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 DEBC has been also implemented over a real testbed 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 deadzone.

7

Conclusions and Future Work

7.1 Conclusions

This thesis has addressed several problems of NCS, with special attention on the reduction of the amount of communication between the different components presented in the networked control loop, but also on tackling other induced problems such as delays or packet losses. Event-triggered policies have shown to be effective to cope with this problems in the studied scenarios, and several applications have been implemented to back the theoretical results.

More specifically, Chapter 2 has addressed the problem of remote controllers. The proposed solution lies on the design of two middleware layers which interface the conventional components with the network, being a model of the plant the most important element of the layer at the controller side. The iteration of the model and the basis controller allows us to generate finite-length signal predictions to cope with delays and packet losses effectively. Moreover, the inclusion of event-triggering policies let us reduce the frequency of the communication and gives a robust response to disturbances.

The lack of synchronization between the controller and the rest of the elements of the control loop does not allow to measure delays. This limitation has been overcome measuring the RTT from the controller side. Moreover, a disturbance estimator has been designed to get better predictions of the future states of the

process. If the full state cannot be measured, a mixed solution between the use of observers and LTI controllers has been proposed, and particularized to PI controllers. The stability of the system has been studied via Lyapunov-based analysis for the three situations aforementioned.

The implementation of the two middleware layers in LabVIEW has been described in Chapter 3. These applications let us reuse conventional controllers in networked control with little effort. Several experiments have been designed to test the controller under treacherous network conditions. The experimental results have illustrated the goodness of the proposed solution.

The second part of the thesis has focused on distributed control in NCS. In particular, event-triggered control has been proposed to reduce the amount of communication and actuation in interconnected linear systems. Chapter 4 tackles this problem for both perfect and non-perfect decoupled systems, showing that asymptotic convergence to the equilibrium for the overall system can be achieved while guaranteeing a lower bound for the inter-event times if time-dependent trigger functions are used.

Several extensions to the aforesaid design have been proposed in Chapter 5. Specifically, the problem of delays and packet losses in distributed event-triggered control has been addressed. Two network protocols have been designed, and one of them deals with the problem of state inconsistency providing larger upper bounds on the delay and packet losses. Moreover, the properties of asymptotic stability and minimum inter-event time are preserved with these protocols.

Two improvements have been also described in Chapter 5 to provide a more efficient usage of the the resources of embedded systems. If the frequency of actuation in the system cannot exceed a certain value, a second set of trigger functions can be designed to control the update of the control law in distributed control systems. Moreover, if the critical issue is the frequency of transmission, a distributed model-based design can be used to enlarge the inter-event times.

Finally, Chapter 6 has reported the development of a simulator to test distributed event-triggering in networked mobile robots. This tool provides a high degree of interactivity, and therefore it is suitable for an education environment,

and it makes possible to test the control algorithms under a wide range of scenarios as well. Finally, the validation over a testbed of mobile robots has been given.

7.2 Future work

This work can be extended in several directions. Some suggestions are listed below.

- Centralized and decentralized model-based event-triggered control approaches have been proposed in chapters 2 and 5 to enlarge the inter-event time in single loop and multi-loop schemes, respectively. If a perfect model cannot be assumed, the model uncertainty negatively affects the performance of the system and the frequency of generation of events. Possible solutions may be found on the study of adaptive control [rW95] or online parameters estimation techniques [Lju99]. Our guess is that there is not a trivial solution, because event-triggered systems can be seen as time-varying parameters systems, and there is not a priori knowledge of when the next event will take place. Moreover, the problem is more complicated in distributed paradigms because the effect of the interconnection terms and the model uncertainties cannot be distinguished a priori. There is a recent contribution [GA11b] that deals with the adaptive stabilization of centralized model-based control problem for discrete-time linear systems by variations of the Kalman Filter. However, the requirement of zero-mean white noise does not fit into the error profile introduced by event-based control.

- Even though the distributed event-based control approach addressed in this thesis facilitates the scalability of the system since the feedback and the decoupling gains are designed locally, the parameter α of the trigger functions is constrained by global information of the system such as $\lambda_{max}(A_K)$, $\kappa(V)$ or $\|\Delta\|$. The use of distributed algorithms [YFG+08, YFL08] in an event-based fashion to estimate global information of the system is another

possible direction of future research.

- There are many fields in which event-triggered control can be useful such as electrical grids, satellite swarms, traffic networks, irrigation channels, and photobioreactors. In this kind of systems, an energy aware design for distributed control is desirable, and event-triggering pursues this goal. In this regard, there is already an ongoing research project for photobioreactors [SÍ3], and recent collaborations for electrical grids and traffic networks.

- The middleware layers have been implemented in LabVIEW. However, it would be desirable that the application at the client side was built with a license-free software. Thus, Ejs seems adequate for this purpose, and this direction would also follow the scheme of remote laboratories proposed in [Var10], that would be transformed into *remote controlled laboratories*. Moreover, the modularity of the middleware layers, and in particular of the CAL, makes it suitable for its implementation as an Ejs *element*, which is quite similar to a Java library [FGElT$^+$12] but provides an application programming interface (API) for its customization. With respect to the MaSS simulator, the library and the code concerning the network could also be implemented as an Ejs element to improve its portability.

Bibliography

[AAJ+11] J. Araujo, A. Anta, M. Mazo Jr., J. Faria, A. Hernandez, P. Tabuada, and K. H. Johansson. Self-triggered control over wireless sensor and actuator networks. In *International Conference on Distributed Computing in Sensor Systems and Workshops*, pages 1–9, Barcelona, 2011.

[AKK04] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Transactions on Wireless Communications*, 11(6):6–24, 2004.

[AMH09] A. Al-Mohi and N. J. Higham. Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimate. *SIAM Journal on Matrix Analysis and Applications*, 30:1639–1657, 2009.

[Ara11] Jose 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.

[Arz99] K.E. Arzén. A simple event-based pid controller. In *IFAC World Congress*, pages 423–428, Beijing, 1999.

[AT10a] A. Anta and P. Tabuada. On the minimum attention and anytime attention problems for nonlinear systems. In *49th IEEE Conference on Decision and Control*, pages 3234–3239, Atlanta, 2010.

[AT10b] A. Anta and P. Tabuada. To sample or not to sample: self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042, 2010.

[BA11] R. Blind and F. Allgöwer. On the optimal sending rate for networked control systems with a shared communication medium. In *50th IEEE Conference on Decision and Control*, pages 4704–4709, Orlando, 2011.

[BCM+10] M. Baseggio, A. Cenedese, P. Merlo, M. Pozzi, and L. Schenato. Distributed perimeter patrolling and tracking for camera networks. In *49th IEEE Conference on Decision and Control*, pages 2093–2098, Atlanta, 2010.

[BDWL10] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung. Mac essentials for wireless sensor networks. *IEEE Communications Surveys and Tutorials*, 12(2):222–248, 2010.

[Bem98] A. Bemporad. Predictive control of teleoperated constrained systems with unbounded communication delays. In *37th Conference of Decision and Control*, pages 2133–2138, Tampa, 1998.

[BF60] F. L. Bauer and C. T. Fike. Norms and exclusion theorems. *Numer. Math.*, 2:137–141, 1960.

[Blu07] P. A. Blume. *The LabVIEWStyle Book*. Prentice Hall, 2007.

[BZ11] S. Bolognani and S. Zampieri. A gossip-like distributed optimization algorithm for reactive power flow control. In *IFAC World Congress*, pages 5700–5705, Milano, 2011.

[CB08] A. Chaillet and A. Bicchi. Delay compensation in packet switch-

ing networked controlled systems. In *47th IEEE Conference on Decision and Control*, pages 3620–362, Cancun, 2008.

[CH08] A. Cervin and T. Henningsson. Scheduling of event-triggered controllers on a shared network. In *47th IEEE Conference on Decision and Control*, pages 3601–3606, Cancun, 2008.

[Chu86] K.W.E. Chu. Generalization of the bauer-fike theorem. *Numerische Mathematik*, 49:685–691, 1986.

[CM02] C.F. Chiasserini and E. Magli. Energy consumption and image quality in wireless video-surveillance networks. In *13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 5, pages 2357–2361. IEEE, 2002.

[CMV+10] A. Camacho, P. Martí, M. Velasco, C. Lozoya, R. Villa, J.M. Fuertes, and E. Griful. Self-triggered networked control systems: an experimental case study. In *IEEE 2010 International Conference on Industrial Technology*, pages 123–128, Valparaiso, 2010.

[com13] Creative commons. Website, 2013. `http://creativecommons.org/licenses/by/3.0/`.

[CSdM+06] N. Correll, G. Sempo, Y. Lopez de Meneses, J. Halloy, and J.L. Deneubourg. Swistrack: A tracking tool for multi-unit robotic and biological systems. In *International Conference on Intelligent Robots and Systems*, Beijing, 2006.

[DFJ12] D.V. Dimarogonas, E. Frazzoli, and K.H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2012.

[DFP62] R.C. Dorf, M.C. Farren, and C.A. Phillips. Adaptive sampling for sampled-data control systems. *IEEE Transaction on Automatic Control*, 7(1):34–47, 1962.

[DGA08] A. Diaz-Guilera and A. Arenas. *Bio-Inspired Computing and Communication*, chapter Phase Patterns of Coupled Oscillators with Application to Wireless Communication. Springer-Verlag, Berlin, 2008.

[DH10] M. C. F. Donkers and 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*, pages 3246–3251, Atlanta, 2010.

[DH12] M.C.F. Donkers and W.P.M.H. Heemels. Output-based event-triggered control with guaranteed L_∞-gain and improved and decentralised event-triggering. *Transactions on Automatic Control*, 57(6):1362–1376, 2012.

[DHvdWH11] M.C.F. Donkers, W.P.M.H. Heemels, N. van de Wouw, and L. Hetel. Stability analysis of networked control systems using a switched linear systems approach. *IEEE Transactions on Automatic Control*, 56(9):2101 –2115, 2011.

[DL12] O. Demir and J. Lunze. Cooperative control of multi-agent systems with event-based communication. In *American Control Conference*, pages 4504–4509, Montreal, 2012.

[DPSW11] C. De Persis, R. Sailer, and F. Wirth. On a small-gain approach to distributed event-triggered control. In *18th IFAC World Congress*, pages 2401–2406, Milano, 2011.

[DTH12] M.C.F. Donkers, P. Tabuada, and W.P.M.H. Heemels. Minimum attention control for linear systems: A linear programming approach. *Discrete Event Dynamic Systems Theory and Applications*, 2012. DOI 10.1007/s10626-012-0155-x.

[EA09] T. Estrada and P. Antsaklis. Performance of model-based networked control systems with discrete-time plants. In *17th Mediter-*

ranean conference on control & automation, pages 628–633, Thessaloniki, 2009.

[Ell59] P. Ellis. Extension of phase plane analysis to quantized systems. *IRE Transactions on Automatic Control*, 4(2):43–54, 1959.

[ESDCM07] M. Epstein, L. Shi, S. Di Cairano, and R. M. Murray. Control over a network: Using actuation buffers to reduce transmission frequency. In *European Control Conference*, Kos, 2007.

[Esq04] F. Esquembre. Easy java simulations: A software tool to create scientific simulations in java. *Computer Physics Communications*, 156(2):199–204, 2004.

[Eva98] W.R Evans. *UNIX Networked Programming.* Prentice-Hall, 2nd edition, 1998.

[Fab13] E. Fabregas. *Plataformas de experimentación virtual y remota: Aplicaciones de control y robótica.* PhD thesis, UNED, 2013.

[FGElT+12] G. Farias, F. Gomez-Estern, L. De la Torre, D. Muñoz de la Peña, C. Sánchez, and S. Dormido. Enhancing virtual and remote labs to perform automatic evaluation. In *9th IFAC Symposium Advances in Control Education*, pages 276–281, Nizhny Novgorod, 2012.

[FM04] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[FPW97] G.F. Franklin, J.D. Powell, and M. Workman. *Digital control of dynamic systems.* Addison-Wesley. E.U.A, 1997.

[GA11a] E. Garcia and P. J. Antsaklis. Model-based event-triggered control with time-varying network delays. In *50th IEEE Conference on Decision and Control*, pages 1650–1655, Orlando, 2011.

[GA11b] Eloy Garcia and Panos J. Antsaklis. Adaptive stabilization of model-based networked control systems. In *American Control Conference*, pages 1094–1099, San Francisco, 2011.

[GA12] E. Garcia and P.J. Antsaklis. Decentralized model-based event-triggered control of networked systems. In *American Control Conference*, pages 6485–6490, Montreal, 2012.

[GBK09] S. Graham, G. Balinga, and P.R. Kumar. Abstractions, architecture, mechanisms and a middleware for networked control. *IEEE Transactions on Automatic Control*, 54(7):1490–1503, 2009.

[GCB12] Luca Greco, Antoine Chaillet, and Antonio Bicchi. Exploiting packet size in uncertain nonlinear networked control systems. *Automatica*, 48(11):2801–2811, 2012.

[GCHM06] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray. On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage. *Automatica*, 42(2):251–260, 2006.

[GDJ+11] M. Guinaldo, D. V. Dimaragonas, K. H. Johansson, J. Sánchez, and S. Dormido. Distributed event-based control for interconnected linear systems. In *50th Control and Decision Conference*, pages 2553–2558, Orlando, 2011.

[GDJ+13] M. Guinaldo, D.V. Dimarogonas, K.H. Johansson, J. Sánchez, and S. Dormido. Distributed event-based control strategies for interconnected linear systems. *IET Control Theory & Applications*, 2013.

[GFB11] L. Greco, D. Fontanelli, and A. Bicchi. Design and stability analysis for anytime control via stochastic scheduling. *IEEE Transactions on Automatic Control*, 56(3):571–585, 2011.

[GFF+12] M. Guinaldo, G. Farias, E. Fabregas, J. Sánchez, S. Dormido-

Canto, and S. Dormido. An interactive simulator for networked mobile robots. *IEEE Network Magazine*, 26(3):14–20, 2012.

[GKKP06] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *Robotics & Automation Magazine, IEEE*, 13(3):16–25, 2006.

[GLS+12] M. Guinaldo, D Lehmann, J. Sánchez, S. Dormido, and K. H. Johansson. Distributed event-triggered control with network delays and packet-losses. In *51th IEEE Conference on Decision and Control*, pages 1–6, Maui, 2012.

[GQ10] V. Gupta and D. E. Quevedo. On anytime control of nonlinear processes though calculation of control sequences. In *IEEE Conference on Decision and Control*, pages 7564–7569, Atlanta, 2010.

[GSD10] M. Guinaldo, J. Sánchez, and S. Dormido. A packet-based network control system architecture for teleoperation and remote laboratories. In *49th IEEE Conference on Decision and Control*, Atlanta, 2010.

[GSD11] M. Guinaldo, J. Sánchez, and S. Dormido. A co-design strategy of NCS for treacherous network conditions. *IET Control Theory & Applications*, 5(16):1906–1915, 2011.

[GSDD12] M. Guinaldo, J. Sánchez, S. Dormido, and M.A. Delgado. Control en red basado en eventos de múltiples plantas remotas. In *XXXIII Jornadas de Automática*, Vigo, 2012.

[GSS05] G.C. Goodwin, M.E. Salgado, and E.I. Silva. Time-domain performance limitations arising from decentralized architectures and their relationship to the RGA. *International Journal of Control*, 78(13):1045–1062, 2005.

[GT04] D Georgiev and DM Tilbury. Packet-based control. In *American Control Conference*, volume 1, pages 329–336, 2004.

[Gup09] V. Gupta. On an anytime algorithm for control. In *47th IEEE Conference on Decision and Control*, pages 6218–6223, Cancun, 2009.

[HCB02] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application speciffic protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.

[HD13] W.P.M.H. Heemels and M.C.F. Donkers. Model-based periodic event-triggered control for linear systems. *Automatica*, 49(3):698–711, 2013.

[HE04] Lingxuan Hu and David Evans. Localization for mobile sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 45–57. ACM, 2004.

[Hee10] N. Heemels, W.P.M.H. & van de Wouw. *Networked Control Systems*, volume 406, chapter Stability and stabilization of networked control systems, pages 203–253. Springer-Verlag, 2010.

[HGvZ+99] W. P. M. H. Heemels, R. J. A. Gorter, A. van Zijl, P. P. J. van den Bosch, S. Weiland, W. H. A. Hendrix, and M. R. Vonder. Asynchronous measurement and control: a case study on motor synchronization. *Control Engineering Practice*, 7(12):1467–1482, 1999.

[HH07] J. Hu and Y. Hong. Leader-following coordination of multi-agent systems with coupling time delays. *Physica A*, 374:853–863, 2007.

[HI01] A. Horch and A. J. Isaksson. Assessment of the sampling rate in control systems. *Control Engineering Practice*, 9:533–544, 2001.

[Hig08] N. J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, 2008.

[HJT12] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *51st IEEE Conference on Decision and Control*, pages 3270–3285, Maui, 2012.

[HNY07] J. P. Hespanha, P. Naghshtabrizi, and Xu Y. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

[HSvdB08] W. P. M. H. Heemels, J. Sandee, , and P. van den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590, 2008.

[HTVdWN10] W. P M H Heemels, A.R. Teel, N. Van de Wouw, and D. Nešić. Networked control systems with communication constraints: Tradeoffs between transmission intervals, delays and performance. *IEEE Transactions on Automatic Control*, 55(8):1781 –1796, aug. 2010.

[ICMS11] G. Irwin, J. Chen, A. McKernan, and W. Scanlon. Co-design of predictive controllers for wireless network control. *IET Control Theory and Applications*, 4(2):186–196, 2011.

[JER+10] J. Jiménez, R. Estepa, F.R. Rubio, N.F. Gómez-Estern, and A. Estepa. Wireless networked control system: 802.11 performance analysis. In *9th Portuguese Conference on Automatic Control*, Coimbra, 2010.

[JHC07] Erik Johannesson, Toivo Henningsson, and Anton Cervin. Sporadic control of first-order linear stochastic systems. In *Hybrid Systems: Computation and Control*, pages 301–314. Springer, 2007.

[JM96] D. B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[Kat66] T. Kato. *Perturbation Theory for Linear Operators.* Springer-Verlag, 1966.

[KB06] E. Kofman and J.H. Braslavsky. Level crossing sampling in feedback stabilization under data-rate constraints. In *45th IEEE Conference on Decision and Control*, San Diego, 2006.

[KCC04] C. Ko, B. Chen, and J. Chen. *Creating web-based Laboratories.* Springer, 2004.

[KDHH+11] J. Ko, S. Dawson-Haggerty, J. Hui, D. Culler, P. Levis, and A. Terzis. Connecting low power and lossy networks to the internet. *IEEE Communications Magazine: Recent Advances in IETF Standards*, 49(4):96–101, 2011.

[KJA06] W.-J. Kim, K. Ji, and A. Ambike. Real-time operating environment for networked control systems. *IEEE Transactions on Automation Science and Engineering*, 3(3):287–296, 2006.

[Lab13] LabVIEW. Website, 2013. `http://www.ni.com/labview`.

[Las10] LaserMotive. Website, 2010. `http://lasermotive.com/products/uav-power-links/`.

[LBY03] J.R.T. Lawton, R.W. Beard, and B.J. Young. A decentralized approach to formation maneuvers. *Robotics and Automation, IEEE Transactions on*, 19(6):933 – 941, dec. 2003.

[LDH09] Zhongkui Li, Zhisheng Duan, and Lin Huang. Leader-follower consensus of multi-agent systems. In *American Control Conference*, pages 3256–3261, 2009.

[Leh11] D. Lehmann. *Event-based state-feedback control.* PhD thesis, University of Bochum, 2011.

[Lju99] L. Ljung. *System Identification- Theory for the User.* Prentice-Hall, 2nd edition, 1999.

[LL10] J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46(1):211–215, 2010.

[LL11a] D. Lehmann and J. Lunze. Event-based output-feedback control. In *19th Mediterranean Conference on Control and Automaticon*, pages 982–987, Corfu, 2011.

[LL11b] D. Lehmann and J. Lunze. Extension and experimental evaluation of an event-based state-feedback approach. *Control Engineering Practice*, 19(2):101–112, 2011.

[LL11c] L. Li and 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.

[LL12] D. Lehmann and J. Lunze. Event-based control with communication delays and packet losses. *International Journal of Control*, 85(5):566–577, 2012.

[LRCC08] T. Lochmatter, P. Roduit, C. Cianci, and N. Correll. Swistrack - a flexible open source tracking software for multi-agent systems. In *International Conference on Intelligent Robots and Systems*, Nice, 2008.

[LWCV05] G. Lafferriere, A. Williams, J. Caughman, and J.J.P. Veerman. Decentralized control of vehicle formations. *Systems & Control Letters*, 54(9):899–910, 2005.

[MA02] L. A. Montestruque and P. Antsaklis. State and output feedback in model-based networked control systems. In *IEEE Conference on Decision and Control*, Las Vegas, 2002.

[MA03a] L. A. Montestruque and P. Antsaklis. On the model-based control of networked systems. *Automatica*, 39:1837–1843, 2003.

[MA03b] L.A. Montestruque and P.J. Antsaklis. On the model-based control of networked systems. *Automatica*, 39:1837–1843, 2003.

[MA04] L. A. Montestruque and P. Antsaklis. Stability of model-based networked control systems with time-varying transmission times. *IEEE Transactions on Automatic Control*, 49(9):1562–1572, 2004.

[Mar10] P. Di Marco. *Modeling and design of multi-hop energy efficient wireless networks for control applications.* Licentiate thesis, Royal Institute of Technology (KTH), 2010.

[Mas11] I. Mas. *Cluster Space Framework for Multi-Robot Formation Control.* Phd dissertation, Santa Clara University, School of Engineering, 2011.

[MAT09] M. Mazo, A. Anta, and P. Tabuada. On self-triggered control for linear systems: guarantees and complexity. In *European Control Conference*, pages 3767–3772, Budapest, 2009.

[MAT10] M. Mazo, A. Anta, and P. Tabuada. An iss self-triggered implementation for linear controllers. *Automatica*, 46(8):1310–1314, 2010.

[MAW05] S Massoud Amin and Bruce F Wollenberg. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.

[Maz10] Manuel Mazo. *Contributions to the Control of Networked Cyber-Physical Systems.* PhD thesis, University of California, 2010.

[MC11] M. Mazo and M. Cao. Decentralized event-triggered control with asynchronous updates. In *50th IEEE Conference on Decision and Control*, pages 2547–2552, Orlando, 2011.

[MJVR08] Pablo Millán, Isabel Jurado, Carlos Vivas, and Francisco R Rubio. Networked predictive control of systems with data dropouts. In

47th IEEE Conference on Decision and Control, pages 2704–2709, 2008.

[MOB+12] P. Millán, L. Orihuela, G. Bejarano, C. Vivas, T. Áamo, and F.R. Rubio. Design and application of suboptimal mixed H_2/H_∞ controllers for networked control systems. *IEEE Transactions on Control Systems Technology*, 20(4):1057–1065, 2012.

[Mob13] Adept MobileRobots. Website, 2013. `http://www.mobilerobots.com/`.

[Mol11] A. Molin. On the optimal design of decentralized event-triggered controllers for large-scale systems with contention-based communication. In *50th IEEE Conference on Decision and Control*, pages 4710–4716, Orlando, 2011.

[mOw10] mOway. moway user manual v2.1.0. Website, June 2010. `http://www.adrirobot.it/moway/pdf/mOway%20User%20Manua%202.1.0.pdf`.

[mOw13] mOway. Website, 2013. `http://moway-robot.com/en/`.

[MS06] J.A. Misener and S.E. Shladover. Path investigations in vehicle-roadside cooperation and safety: A foundation for safety and vehicle-infrastructure integration research. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 9–16. IEEE, 2006.

[MT08] M. Mazo and P. Tabuada. On event-triggered and self-triggered control over sensor/actuator networks. In *47th IEEE Conference on Decision and Control*, pages 435–440, Cancun, 2008.

[MT11] M. Mazo and P. Tabuada. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Transactions on Automatic Control*, 56(10):2456–2461, 2011.

[MWC+04] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian. Remote surgery case: robot-assisted teleneurosurgery. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 819–823, Barcelona, 2004.

[NC10] W. Ni and D. Cheng. Leader-following consensus of multi-agent systems under fixed and switching topologies. *Systems & Control Letters*, 59(3):209–217, 2010.

[NH06] P. Naghshtabrizi and J.P. Hespanha. Anticipative and non-anticipative controller design for network control systems. *Networked Embedded Sensing and Control*, 331:203–218, 2006.

[NHT08] P. Naghshtabrizi, J. P. Hespanha, and A. R. Teel. Exponential stability of impulsive systems with application to uncertain sampled-data systems. *Systems & Control Letters*, pages 378–385, 2008.

[NPEJ07] S. Nethi, M. Pohjola, L. Eriksson, and R. Jäntt. Platform for emulating networked control systems in laboratory environments. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Helsinki, June 2007.

[NRC07] J. Normey-Rico and E. Camacho. *Control of dead-time processes*. Springer Verlag, 2007.

[ntNS12] ns-2 the Network Simulator. Website, 2012. `http://nsnam.isi.edu/nsnam/index.php`.

[OFL04] P. Ogren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, 2004.

[OSFM07] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[OSM04] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520 – 1533, 2004.

[PF12] C. De Persis and P. Frasca. Self-triggered coordination with ternary controllers. In *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Santa Barbara, 2012.

[PR97] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *2nd IEEE workshop on mobile computing systems and applications*, pages 90–100, 1997.

[PTNA11] R. Postoyan, T. Tabuada, D. Nešič, and A. Anta. Event-triggered and self-triggered stabilization of distributed networked control systems. In *50th IEEE Conference on Decision and Control*, pages 2565–2570, Orlando, 2011.

[QSG07] D. Quevedo, E.I. Silva, and G. Goddwin. Packetized predictive control over erasure channels. In *Americal Control Coference*, pages 1003–1008, Portland, 2007.

[QSG08] D. E. Quevedo, E. I. Silva, and G. C Goodwin. Control over unreliable networks affected by packet erasures and variable transmission delays. *IEEE Journal on Selected Areas in Communications*, 26(4):672–685, 2008.

[RA07] W. Ren and E. Atkins. Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control*, 17(10):1002–033, 2007.

[rB03] K. Åström and B. Bernhardsson. Systems with lebesgue sampling. In Anders Rantzer and Christopher Byrnes, editors, *Directions in Mathematical Systems Theory and Optimization*, volume 286 of *Lecture Notes in Control and Information Sciences*, pages 1–13. Springer Berlin / Heidelberg, 2003.

[RBA07] W. Ren, R. Beard, and E. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.

[rH06] K.J. Åström and T. Hägglund. *Advanced PID Control*. ISA, 2006.

[RJ09] M. Rabi and K. H. Johansson. Scheduling packets for event-triggered control. In *European Control Conference*, Budapest, 2009.

[RJJ08] M. Rabi, K.H. Johansson, and M. Johansson. Optimal stopping for event-triggered sensing and actuation. In *47th IEEE Conference on Decision and Control*, pages 3607–3612, Cancun, 2008.

[RMC06] W. Ren, K. Moore, and Y. Chen. High-order consensus algorithms in cooperative vehicle systems. In *International conference on networking, sensing and control*, pages 457–462, 2006.

[rW95] K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, 2nd edition, 1995.

[SÍ13] J. Sánchez. Macrobrio: Modelado, simulación, control y optimización de fotobiorreactores. In *IX Simposio CEA en Ingeniería de Control*, Valencia, 2013.

[San06] J. Sandee. *Event-driven control in theory and practice*. PhD thesis, Technische Universiteit Eindhoven, 2006.

[SDJ11] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson. Control of multi-agent systems via event-based communication. In *IFAC World Congress*, pages 10086–10091, Milano, 2011.

[SDJ13] G.S. Seyboth, D.V. Dimarogonas, and K.H. Johansson. Event-based broadcasting for multi-agent average consensus. *Automatica*, 49(1):245–252, 2013.

[Sey10] G. Seyboth. Event-based control for multi-agent systems. Diploma thesis, Automatic Control Lab, Royal Institute of Technology (KTH), Sweden, 2010.

[SGQ08] E.I. Silva, G.C. Goodwin, and D.E. Quevedo. On networked control architectures for mimo plants. In *17th IFAC World Congress*, pages 8044–8049, Seoul, 2008.

[SS05] Pete Seiler and Raja Sengupta. An H_∞ approach to networked control. *IEEE Transactions on Automatic Control*, 50(3):356–364, 2005.

[SSB09] J. H. Seo, H. Shim, and J. Back. Consensus of high-order linear systems using dynamic output feedback compensator: Low gain approach. *Automatica*, 45(11):2659 – 2664, 2009.

[Ste07] A. Stefanovska. Coupled oscillators: Complex but not complicated cardiovascular and brain interactions. *IEEE Eng Med Biol Mag.*, 26(6):25–29, 2007.

[Tab07] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[TFJB10] U. Tiberi, C. Fischione, K. H. Johansson, and M. D. Di Benedetto. Adaptive selftriggered control over ieee 802.15.4 networks. In *49th IEEE Conference on Decision and Control*, pages 2099–2104, Atlanta, 2010.

[Tra00] J. Travis. *Internet Applications in LabVIEW*. Prentice Hall, 2000.

[Uch03] Akinobu Uchikubo. Remote surgery support system, August 2003.

[VAP08] F. Vanni, A. P. Aguiar, and A. M. Pascoal. Netmarsys- networked marine systems simulator. Technical Report WP6-0108, Instituto Superior Tecnico (Lisbon), May 2008.

[Var10] H. Vargas. *An Integral Web-based Environment for Control Engineering Education.* PhD thesis, UNED, 2010.

[VF09] P. Varutti and R. Findeisen. Compensating network delays and information loss by predictive control methods. In *European Control Conference*, Budapest, 2009.

[VH08] Richard Vaughan and Andrew Howard. The player project, 2008. `http://robots.mobilerobots.com/wiki/MobileSim`.

[VL77] C.F. Van Loan. The sensitivity of the matrix exponential. *SIAM Journal on Numerical Analysis*, 14(6):971–981, 1977.

[VMF03] Manel Velasco, Pau Martí, and Josep M. Fuertes. The self triggered task model for real-time control systems. In *24th IEEE Real-Time Systems Symposium (RTSS03)*, Cancun, 2003.

[VSD09] H. Vargas, J. Sánchez, and S. Dormido. The spanish university network of web-based laboratories for control engineering education: The automatl@bs project. In *10th European Control Conference*, pages 4623–4628, Budapest, 2009.

[WD73] S.H. Wang and E.J. Davidson. On the stabilization of decentralized control systems. *IEEE Transaction on Automatic Control*, 18(5):473–478, 1973.

[WL08] X. Wang and M. D. Lemmon. Event-triggered broadcasting across distributed networked control systems. In *American Control Conference*, pages 3139–3144, Seattle, 2008.

[WL09] X. Wang and M. D. Lemmon. Self-triggered feedback control systems with finite-gain L_2 stability. *IEEE Transactions on Automatic Control,*, 53(3):452–467, 2009.

[WL10] X. Wang and M. D. Lemmon. Self-triggering under state-independent disturbances. *IEEE Transactions on Automatic Control*, 55(6):1494–1500, 2010.

[WL11] X. Wang and M.D. Lemmon. Event-triggering in distributed networked control systems. *IEEE Transactions on Automatic Control*, 56(3):586–601, 2011.

[WLZ05] Z. Wang, L. Liu, and M. Zhou. Protocols and applications of ad-hoc robot wireless communication networks: An overview. *International Journal of Intelligent Control and Systems*, 10(4):296–303, 2005.

[WTB+12] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur. RPL: IPv6 routing protocol for low power and lossy networks, March 2012.

[YFG+08] F.P. Yang, R.A. Freeman, G.J. Gordon, K.M. Lynch, S. Srinivasa, and R. Sukthankar. Decentralized estimation and control of graph connectivity in mobile sensor networks. In *American Control Conference*, pages 2678–2683, Seattle, 2008.

[YFL08] P. Yang, R.A. Freeman, and K.M. Lynch. Multi-agent coordination by decentralized estimation and control. *IEEE Transaction on Automatic Control*, 53(11):2480–2496, 2008.

[YHL05] D. Yue, Q. L. Han, and J. Lam. Network-based robust H_∞ control of systems with uncertainty. *Automatica*, 41(6):999–1007, 2005.

[ZLR09] Y. Zhao, G. Liu, and D. Rees. Design of a packed-based control framework for networked control systems. *IEEE Transactions on Control Systems Technology*, 17(5):859–865, 2009.

APPENDICES

A

Prototypes models

This appendix describes the model of the prototypes used in Chapter 3. These mathematical models can also be found in the Student Handout by Quanser.

A.1 The QUANSER SRV-02 setup

The electrical part of the motor is examined first. The electrical scheme of the circuit is shown in Figure A.1. Using Kirchhoff's voltage law, it follows

$$V_m(t) - R_m I_m(t) - L_m \frac{dI_m(t)}{dt} - E_{emf}(t) = 0.$$

Since $L_m \ll R_m$, the motor inductance can be disregarded, and it yields

$$I_m = \frac{V_m - E_{emf}}{R_m}.$$

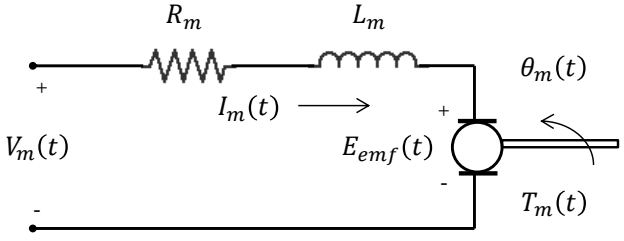

Figure A.1: Electrical circuit of the SRV-02 gear.

The back electromotive force (emf) created by the motor is proportional to the motor sharp velocity ω_m such that

$$I_m = \frac{V_m - K_m\dot{\theta}_m}{R_m}.$$

The dynamics of the motor shaft according to Newton's second law is

$$J_m\ddot{\theta}_m = T_m - \frac{T_l}{\eta_g K_g}, \tag{A.1}$$

where $\frac{T_l}{\eta_g K_g}$ is the load torque seen through the gears, and η_g is the efficiency of the gearbox.

If the Newton's second law is applied to the load of the motor:

$$J_l\ddot{\theta}_l = T_l - B_{eq}\dot{\theta}_l, \tag{A.2}$$

where B_{eq} is the viscous damping coefficient at the output.

From (A.1) and (A.2), it yields

$$J_i\ddot{\theta}_l = \eta_g K_g T_m - \eta_g K_g J_m\ddot{\theta}_m - B_{eq}\dot{\theta}_l. \tag{A.3}$$

Since $\theta_m = K_g\theta_l$ and $T_m = \eta_m K_t I_m$, where η_m is the efficiency of the motor, (A.2) can be rewritten as

$$J_l\ddot{\theta}_l + \eta_g K_g^2 J_m\ddot{\theta}_l + B_{eq}\dot{\theta}_l = \eta_g\eta_m K_g K_t I_m.$$

Finally, if the electrical and dynamical model are combined, it yields the following transfer function

$$\frac{\theta_l(s)}{V_m(s)} = \frac{\eta_g\eta_m K_t K_g}{J_{eq}R_m s^2 + (B_{eq}R_m + \eta_g\eta_m K_m K_t K_g^2)s},$$

where $J_{eq} = J_l + \eta_g J_m K_g^2$.

The parameters are listed in Table A.1 at the end of this appendix.

A.2 The flexible link: QUANSER SRV-02 series

The equations of motion involving a rotary flexible link, involves modeling the rotational base and the flexible link as rigid bodies. As a simplification to the partial differential equation describing the motion of a flexible link, a lumped single degree of freedom approximation is used. We first start the derivation of the dynamic model by computing various rotational moment of inertia terms. The rotational inertia for a flexible link is given by

$$J_{link} = \frac{mL^2}{3},$$

where m is the mass of the flexible link, and L is the length.

For a single degree of freedom system, the natural frequency is related with torsional stiffness and rotational inertia in the following manner

$$\omega_c = \sqrt{\frac{K_{stiff}}{J_{link}}},$$

where ω_c is found experimentally and K_{stiff} is an equivalent torsion spring constant as depicted in Figure 3.4.

In addition, any frictional damping effects between the rotary base and the flexible link are assumed negligible. Next, we derive the generalized dynamic equation for the tip and base dynamics using Lagrange energy equations in terms of a set of generalized variables α and θ, where α is the angle of tip deflection and θ is the base rotation.

The potential energy of the system is provided by the torsional spring, and it is given by

$$V = \frac{1}{2}K_{stiff}\alpha^2.$$

The total kinetic energy of the mechanical system is computed as the sum of the base and the flexible link:

$$T = \frac{1}{2}J_{eq}\dot{\theta}^2 + \frac{1}{2}J_{link}(\dot{\theta} + \dot{\alpha})^2.$$

Thus, the Lagrangian is

$$L = T - V = \frac{1}{2} J_{eq} \dot{\theta}^2 + \frac{1}{2} J_{link} (\dot{\theta} + \dot{\alpha})^2 - \frac{1}{2} K_{stiff} \alpha^2. \tag{A.4}$$

The existence of two degrees of freedom (θ and α) provides two Lagrange equations for the system:

$$\frac{\partial}{\partial t}\Big(\frac{\partial L}{\partial \dot{\theta}}\Big) - \frac{\partial L}{\partial \theta} = T_{output} - B_{eq}\dot{\theta} \tag{A.5}$$

$$\frac{\partial}{\partial t}\Big(\frac{\partial L}{\partial \dot{\alpha}}\Big) - \frac{\partial L}{\partial \alpha} = 0. \tag{A.6}$$

Applying (A.4) into (A.5) and (A.6), it yields

$$J_{eq}\ddot{\theta} + J_{link}(\ddot{\theta} + \ddot{\alpha}) = T_{output} - B_{eq}\dot{\theta} \tag{A.7}$$

$$J_{link}(\ddot{\theta} + \ddot{\alpha}) + K_{stiff}\alpha = 0. \tag{A.8}$$

From the model detailed in the previous section, the output torque on the load from the motor is

$$T_{output} = \frac{\eta_m \eta_g K_t K_g (V_m - K_g K_m \dot{\theta})}{R_m}. \tag{A.9}$$

Finally, combining (A.7), (A.8) and (A.9), it results in the following state space model:

$$\begin{pmatrix} \dot{\theta} \\ \dot{\alpha} \\ \ddot{\theta} \\ \ddot{\alpha} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_{stiff}}{J_{eq}} & -\frac{\eta_m \eta_g K_t K_g^2 K_m + B_{eq} R_m}{J_{eq} R_m} & 0 \\ 0 & \frac{-J_{stiff}(J_{eq}+J_{link})}{J_{eq} J_{link}} & \frac{\eta_m \eta_g K_t K_g^2 K_m + B_{eq} R_m}{J_{eq} R_m} & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{pmatrix}.$$

The parameters of the flexible link are listed in Table A.2, and the parameters concerning the SRV-02 gear are listed in A.1.

Table A.1: SRV-02 model parameters.

Symbol	Description	Value (SI units)
V_m	Circuit imput voltage	
I_m	Circuit current	
R_m	Resistance	2.6
L_m	Inductance	
E_{emf}	Motor back-emf voltage	
θ_m	Motor shaft position	
ω_m	Motor shaft angular velocity	
θ_l	Load shaft position	
ω_l	Load shaft angular velocity	
T_m	Torque generated by the motor	
T_l	Torque applied at the load	
K_m	Back-emf constant	0.00767
K_t	Motor torque constant	0.00767
J_m	Motor moment of inertia	3.87×10^{-7}
J_{eq}	Equivalent moment of inertia at the load	2.0×10^{-3}
B_{eq}	Equivalent viscous damping coefficient	4.0×10^{-3}
K_g	SRV02 system gear ratio (motor→load)	70
η_g	Gearbox efficiency	0.9
η_m	Motor efficiency	0.69

Table A.2: Flexible link model parameters.

Symbol	Description	Value (SI units)
L	Length of flexible link	0.42
m	Mass of flexible link	0.065
θ	Servo load gear angle	
α	Arm deflection	
D	Link end point deflection	
ω_c	Link's damped natural frequency	18.85
J_{link}	Modeled link moment of inertia	0.0038

B
Proofs

B.1 Proof of Theorem 2.2

The forward difference of the Lyapunov function (2.14) for (2.29) is

$$\begin{aligned}
&\Delta V(k) = x^T(k+1)Px(k+1) - x^T(k)Px(k) \\
&= \big(A_{dK}x(k) + B_dKe(k) + w(k)\big)^T P\big(A_{dK}x(k) + BKe(k) + w(k)\big) - x^T(k)Px(k) \\
&= -x^T(k)Qx(k) + 2e^T(k)(B_dK)^T PA_{dK}x(k) + e^T(k)(B_dK)^T PB_dKe(k) \\
&\quad + 2w^T(k)PA_{dK}x(k) + 2w^T(k)PB_dKe(k) + w^T(k)Pw(k) \\
&\leq -\lambda_{min}(Q)\|x(k)\|^2 + 2\|(BK)^T PA_{dK}\|\|e(k)\|\|x(k)\| + \|(BK)^T PBK\|\|e(k)\|^2 \\
&\quad + 2\|PB_dK\|\|w(k)\|\|e(k)\| + 2\|PA_{dK}\|\|w(k)\|\|x(k)\| + \lambda_{max}(P)\|w(k)\|^2.
\end{aligned} \tag{B.1}$$

The error and the disturbance are bounded by $\|e(k)\| \leq 2c$ and $\|w(k)\| \leq w_{max}$. Thus, the Lyapunov function decreases if (B.1) is negative. This holds whenever

$$\|x(k)\| \geq \frac{\delta_b + \sqrt{\delta_b^2 + 4\delta_a\delta_c}}{2\delta_a} = \delta_x^w,$$

where $\delta_a, \delta_b, \delta_c$ and δ_x^w are defined in (2.31)-(2.34).

The state decreases until it reaches this bound. Let us denote k^* the time instant at which the state enters this region. According to (2.29), the norm of

the state at the next step is, in the worst case:

$$\|x(k^*+1)\| \leq \|A_{dK}\|\delta_x^w + \|B_d K\|2c + w_{max}.$$

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

B.2 Proof of Theorem 2.3

The forward difference of the Lyapunov function (2.46) for (2.45) is

$$\begin{aligned}
\Delta V(k) &= \xi^T(k+1)P\xi(k+1) - \xi^T(k)P\xi(k) \\
&= \big(A_{CL}\xi(k) + F(e_y(k)+v(k))\big)^T P\big(A_{CL}\xi(k) + F(e_y(k)+v(k))\big) \\
&\quad - \xi^T(k)P\xi(k) \\
&= -\xi^T(k)Q\xi(k) + 2(e_y^T(k)+v^T(k))F^T P A_{CL}\xi(k) \\
&\quad + (e_y^T(k)+v^T(k))F^T F(e_y(k)+v(k)) \\
&\leq -\lambda_{min}(Q)\|\xi(k)\|^2 + 2\|F^T P A_{CL}\|\|e_y(k)+v(k)\|\|\xi(k)\| \\
&\quad + \|F^T P F\|\|e_y(k)+v(k)\|^2. \qquad \text{(B.2)}
\end{aligned}$$

The right hand side of (B.2) is an algebraic second order equation in $\|\xi(k)\|$ such that the Lyapunov function decreases whenever

$$\|\xi(k)\| \geq \sigma_\xi \|e_y(k)+v(k)\|,$$

where σ_ξ is given in (2.48).

Because the error e_y is bounded by $2c_y$ and the noise by v_{max}, $\Delta V < 0$ in the region $\|\xi(k)\| > \sigma_\xi(2c_y + v_{max})$. 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 (2.45), it follows that

$$\|\xi(k^*+1)\| \leq (\sigma_\xi \|A_{CL}\| + \|F\|)(2c_y + v_{max}).$$

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

B.3 Proof of Theorem 4.2

The analytical solution of (4.24) is

$$x(t) = e^{(A_K+\Delta)t}x(0) + \int_0^t e^{(A_K+\Delta)(t-s)}BKe(s)ds. \tag{B.3}$$

From Assumption 4.1, the matrix A_K is diagonalizable and

$$\|e^{A_K t}\| \leq \kappa(V)e^{-|\lambda_{max}(A_K)|t}.$$

Thus, (4.12) can be used to bound $e^{(A_K+\Delta)t}$ as

$$\|e^{(A_K+\Delta)t}\| \leq \kappa(V)e^{-(|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|)t}.$$

Note that the exponent is negative since $|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\| > 0$ from Assumption 4.2. Let us denote $\lambda_\Delta = |\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|$.

Consequently, the state can be bounded by

$$\|x(t)\| \leq \kappa(V)(e^{-\lambda_\Delta t}\|x(0)\| + \int_0^t e^{-\lambda_\Delta(t-s)}\|BK\|\|e(s)\|ds).$$

The overall system error is bounded by

$$\|e(t)\| \leq \sqrt{N}(c_0 + c_1 e^{-\alpha t}).$$

This yields

$$\begin{aligned}\|x(t)\| &\leq \kappa(V)(e^{-\lambda_\Delta t}\|x(0)\| + \int_0^t \sqrt{N}e^{-\lambda_\Delta(t-s)}\|BK\|(c_0 + c_1 e^{-\alpha s}))\\ &= \kappa(V)\Big(e^{-\lambda_\Delta t}\|x(0)\| + \frac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta}(1-e^{-\lambda_\Delta t})\\ &+ \frac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha}(e^{-\alpha t} - e^{-\lambda_\Delta t})\Big),\end{aligned}$$

which by reordering terms and restoring $\lambda_\Delta = |\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|$ yield (4.39), proving the first part of the theorem. Note that the previous expression can be upper bounded by

$$\|x(t)\| \leq \kappa(V)(\|x(0)\|e^{-\lambda_\Delta t} + \tfrac{\|BK\|\sqrt{N}c_0}{\lambda_\Delta} + \tfrac{\|BK\|\sqrt{N}c_1}{\lambda_\Delta - \alpha}e^{-\alpha t}), \tag{B.4}$$

by omitting the negative terms.

We next show that broadcasting period is lower bounded. If t^* refers to the last event time occurrence, $\|e_i(t^*)\| = 0$, and $f_i(t^*) = -c_0 - c_1 e^{-\alpha t^*} < 0$.

Because $\dot{e}_i(t) = -\dot{x}_i(t)$ and $\|e_i(t)\| \leq \int_{t^*}^t \|\dot{x}_i(t)\| \leq \int_{t^*}^t \|\dot{x}(t)\|$ hold, and from (4.24) we derive

$$\|\dot{x}(t)\| \leq \|A_K + \Delta\|\|x(t)\| + \|BK\|\sqrt{N}(c_0 + c_1 e^{-\alpha t^*}).$$

If the last event occurred at time $t^* > 0$

$$\|e_i(t)\| \leq \int_{t^*}^t \|\dot{x}(t)\| \leq \int_{t^*}^t (\|A_K + \Delta\|\|x(s)\| + \|BK\|e(s))ds,$$

and $\|x(t)\| \leq \|x(t^*)\|$ holds in (4.31). Thus, defining the following constants

$$\begin{aligned}k_{\Delta,1} &= \kappa(V)\|A_K + \Delta\|\|x(0)\|\\ k_{\Delta,2} &= \|BK\|\sqrt{N}c_1(\frac{\kappa(V)\|A_K + \Delta\|}{\lambda_\Delta - \alpha} + 1)\\ k_{\Delta,3} &= \|BK\|\sqrt{N}c_0(\frac{\kappa(V)\|A_K + \Delta\|}{\lambda_\Delta} + 1),\end{aligned}$$

the error can be bounded as

$$\|e_i(t)\| \leq \int_{t^*}^{t} (k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3})ds = (k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3})(t - t^*).$$

The next event will not be triggered before $\|e_i(t)\| = c_0 + c_1 e^{-\alpha t} \geq c_0$. Thus a lower bound on the inter-events time is given by

$$T_{\Delta,min} = \frac{c_0}{k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3}}, \tag{B.5}$$

which is a positive quantity.

B.4 Derivation of (4.45)

The bound (4.44) can be upper bounded by

$$\begin{aligned}\|x(t)\| \leq & \kappa(V)\Big(\tfrac{\|BK\|\sqrt{N}c_0\beta_0}{|\lambda_{max}(A_K)|} + (1 + \kappa(V)\|\Delta\|t)\|x(0)\|e^{-|\lambda_{max}(A_K)|t} \\ & + \tfrac{\|BK\|\sqrt{N}c_1\beta_1 e^{-\alpha t}}{|\lambda_{max}(A_K)|-\alpha}\Big).\end{aligned}$$

As in the analysis of the theorem 4.1 and 4.2 the error between two consecutive events can be upper bounded as

$$\|e_i(t)\| \leq \int_{t^*}^{t} (\|A_K + \Delta\|)x(s) + \|BK\|e(s))ds.$$

With the definition of (4.41)-(4.43), and if $k_{\Delta,2}$ and $k_{\Delta,2}$ are approximated to $k_{\Delta,2} \approx \|BK\|\sqrt{N}c_1(\frac{\kappa(V)\|A_K+\Delta\|\beta_1}{|\lambda_{max}(A_K)|-\alpha} + 1)$, $k_{\Delta,3} \approx \|BK\|\sqrt{N}c_0(\frac{\kappa(V)\|A_K+\Delta\|\beta_0}{|\lambda_{max}(A_K)|} + 1)$, it follows that

$$\begin{aligned}&\|e_i(t)\| \leq \int_{t^*}^{t} (k_{\Delta,1}(1 + \kappa(V)\|\Delta\|s)e^{-|\lambda max(A_K)|t^*} + k_{\Delta,2}e^{-\alpha t^*} + k_{\Delta,3})ds \\ &= \tfrac{\kappa(V)\|\Delta\|k_{\Delta,1}e^{-|\lambda max(A_K)|t^*}}{2}(t^2 - t^{*2}) + (k_{\Delta,1}e^{-|\lambda max(A_K)|t^*} + k_{\Delta,2}e^{-\alpha t^*} + k_{\Delta,3})T \\ &\leq \tfrac{\kappa(V)\|\Delta\|k_{\Delta,1}}{2}T^2 + (k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3})T.\end{aligned}$$

where $T = t - t^*$ is the inter-event time. Note that it holds that $e^{-|\lambda_{max}(A_K)|t^*}(1 + \kappa(V)\|\Delta\|t^*) < 1, \forall t^* > 0$.

The next event will not be triggered before $\|e_i(t)\| = c_0 + c_1 e^{-\alpha t} \geq c_0$. Thus a lower bound on the inter-events time is the solution of

$$a_\Delta T^2 + b_\Delta T = c_0 \tag{B.6}$$

where

$$\begin{aligned} a_\Delta &= \frac{k_{\Delta,1}\kappa(V)\|\Delta\|}{2} \\ b_\Delta &= k_{\Delta,1} + k_{\Delta,2} + k_{\Delta,3}. \end{aligned} \tag{B.7}$$

Since a_Δ and b_Δ are positive constants, only one of the solutions of (B.6) is strictly positive and hence feasible, and is given by (4.45).

B.5 Proof of Theorem 4.3

Let us denote $F_d = A_{dK} + \Delta_d$. The analytical solution of (4.47) is

$$x(\ell) = F_d^\ell x(0) + \sum_{j=0}^{\ell-1} F_d^{\ell-1-j} B_d K_d e(j). \tag{B.8}$$

This can be bounded as

$$\|x(\ell)\| \leq \|F_d^\ell\|\|x(0)\| + \sum_{j=0}^{\ell-1} \|F_d^{\ell-1-j}\|\|B_d K_d\|\|e(j)\|. \tag{B.9}$$

Let us assume that $\Delta_d^j \approx \mathbf{0}, \forall j \geq 2$. According to (4.13), $\|F_d^\ell\|$ can be bounded as:

$$\begin{aligned} \|F_d^\ell\| = \|(A_{dK} + \Delta_d)^\ell\| &\leq \|A_{dK}^\ell\| + \|\sum_{j=0}^{\ell-1} A_{dK}^{\ell-1-j}\Delta_d A_{dK}^j\| + \mathcal{O}(\|\Delta_d\|^2) \\ &\approx \|A_{dK}^\ell\| + \|\sum_{j=0}^{\ell-1} A_{dK}^{\ell-1-j}\Delta_d A_{dK}^j\|, \end{aligned}$$

since $\|\Delta_d\|^2 \approx 0$.

Moreover, as in the continuous time case, we assume that A_{dK} is diagonalizable, and hence, $A_{dK} = V_d D_d V_d^{-1}$. It also holds that $\|D_d\| = |\lambda_M(D_d)| = |\lambda_M(A_{dK})|$, where $\lambda_M(A_{dK})$ is the eigenvalue with the closer magnitude to 1. Thus,

$$\|A_{dK}^{\ell}\| \leq \|V_d D_d^{\ell} V_d^{-1}\| \leq \kappa(V_d)|\lambda_M(A_{dK})|^{\ell},$$

where $\kappa(V_d) = \|V_d\|\|V_d^{-1}\|$.

Similarly, the following bound can be computed the sum:

$$\begin{aligned}\|\sum_{j=0}^{\ell-1} A_{dK}^{\ell-1-j}\Delta_d A_{dK}^{j}\| &\leq \kappa^2(V_d)\sum_{j=0}^{\ell-1}|\lambda_M(A_{dK})|^{\ell-1-j}\|\Delta_d\||\lambda_M(A_{dK})|^{j} \\ &= \kappa^2(V_d)|\lambda_M(A_{dK})|^{\ell-1}\|\Delta_d\|\ell.\end{aligned}$$

Thus, $\|F_d^{\ell}\|$ is bounded by

$$\|F_d^{\ell}\| \leq \kappa(V_d)|\lambda_M(A_{dK})|^{\ell}\Big(1 + \ell\frac{\kappa(V_d)\|\Delta_d\|}{|\lambda_M(A_{dK})|}\Big). \tag{B.10}$$

If we consider the bound (B.10) in (B.9), it holds that

$$\begin{aligned}\|x(\ell)\| \leq \kappa(V_d)|\lambda_M(A_{dK})|^{\ell}\Big(1 + \ell\frac{\kappa(V_d)\|\Delta_d\|}{|\lambda_M(A_{dK})|}\Big)\|x(0)\| \qquad (B.11)\\ + \sum_{j=0}^{\ell-1}\Big(\kappa(V_d)|\lambda_M(A_{dK})|^{\ell-1-j}(1 + (\ell-1-j)\frac{\kappa(V_d)\|\Delta_d\|}{|\lambda_M(A_{dK})|})\|B_d K_d\|\|e(j)\|\Big).\end{aligned}$$

Moreover, from (4.58), the error can be bounded as $\|e(j)\| \leq \sqrt{N}(c_0 + c_1\alpha_d^j)$.

The sum in (B.11) can be computed taking into account that

$$\begin{aligned}\sum_{j=0}^{\ell-1} r^{\ell-1-j} &= \frac{1-r^{\ell}}{1-r} \\ \sum_{j=0}^{\ell-1}(\ell-1-j)r^{\ell-1-j} &= r\cdot\Big(\frac{1-r^{\ell}}{(1-r)^2} - \frac{\ell r^{\ell-1}}{1-r}\Big),\end{aligned}$$

where r can be $|\lambda_M(A_{dK})|$ or $\frac{\alpha_d}{|\lambda_M(A_{dK})|}$.

Thus, it yields that

$$\begin{aligned}\|x(\ell)\| \leq \kappa(V_d)\Big(&\tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|}\big(1+\tfrac{\kappa(V_d)\|\Delta_d\|}{1-|\lambda_M(A_{dK})|}\big)+|\lambda_M(A_{dK})|^{\ell}\Big(\|x(0)\|\\ &-\tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|}\big(1+\tfrac{\kappa(V_d)\|\Delta_d\|}{1-|\lambda_M(A_{dK})|}\big)-\tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\big(1+\tfrac{\kappa(V_d)\|\Delta_d\|}{\alpha-|\lambda_M(A_{dK})|}\big)\\ &+\tfrac{\kappa(V_d)\|\Delta_d\|}{|\lambda_M(A_{dK})|}\ell\big(\|x(0)\|-\tfrac{\|B_dK_d\|\sqrt{N}c_0}{1-|\lambda_M(A_{dK})|}-\tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\big)\Big)\\ &+\alpha_d^{\ell}\tfrac{\|B_dK_d\|\sqrt{N}c_1}{\alpha_d-|\lambda_M(A_{dK})|}\big(1+\tfrac{\kappa(V_d)\|\Delta_d\|}{\alpha-|\lambda_M(A_{dK})|}\big)\Big).\end{aligned} \tag{B.12}$$

Defining $\beta_{d,0}, \beta_{d,1}$ as in (4.60)-(4.61), it yields to (4.59), which concludes the proof.

B.6 Proof of Proposition 5.2

Assume that the last event occurred at time t_k^i and that the maximum transmission delay to its neighbors is $\bar{\tau}_k^i$. From Assumption 5.1, it follows that

$$\|\int_{t_k^i}^{t_k^i+\bar{\tau}_k^i}\dot{e}_i(s)ds\| = \|e_i(t_k^i+\bar{\tau}_k^i)-e_i(t_k^i)\| < c_1e^{-\alpha(t_k^i+\bar{\tau}_k^i)}, \tag{B.13}$$

has to be satisfied (see (5.16)) 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, $\|e_i(t_k^i)\| = c_1e^{-\alpha t_k^i}$ holds and, thus

$$\|e_i(t_k^i+\bar{\tau}_k^i)\| < c_1e^{-\alpha t_k^i} + c_1e^{-\alpha(t_k^i+\bar{\tau}_k^i)} = c_1(1+e^{\alpha\bar{\tau}_k^i})e^{-\alpha(t_k^i+\bar{\tau}_k^i)},$$

must hold. Because this result is valid for any time t and $e^{\alpha\bar{\tau}_k^i} < e^{\alpha\tau^\star}$, $\forall\bar{\tau}_k^i < \tau^\star$, it follows

$$\|e_i(t)\| < c_1(1+e^{\alpha\tau^*})e^{-\alpha t}.$$

B.7 Proof of Theorem 5.3

The state at any time is given by

$$x_i(t) = e^{A_{K,i}t}x_i(0) + \int_0^t e^{A_{K,i}(t-s)}(B_iK_ie_i(s) + B_i\sum_{j\in N_i} L_{ij}e_j(s))ds.$$

According to 5.2, the error is bounded by $\|e_i(t)\| < c_1(1+e^{\alpha\tau^*})e^{-\alpha t}$. Thus, a bound on $x_i(t)$ can be calculated following the methodology of Chapter 4 as

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\frac{\mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha} + e^{-|\lambda_{max}(A_{K,i})|t}\big(\|x_i(0)\| - \frac{\mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha}\big)\Big),$$

which proves the second part of the theorem.

Note that (5.21) can be upper bounded as

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\frac{\mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha} + e^{-|\lambda_{max}(A_{K,i})|t}\|x_i(0)\|\Big). \tag{B.14}$$

Moreover, in the interval $t \in [t^i_{k-1} + \bar{\tau}^i_{k-1}, t^i_k + \bar{\tau}^i_k)$ it holds that

$$\dot{e}_i(t) = -A_{K,i}x_i(t) - B_iK_ie_i(t) - \sum_{j\in N_i} B_iL_{ij}e_j(t),$$

and this is particularly true in the subinterval $[t^i_k, t^i_k + \bar{\tau}^i_k)$. Thus

$$\begin{aligned}\|\dot{e}_i(t)\| &= \|A_{K,i}x_i(t) + B_iK_ie_i(t) + \sum_{j\in N_i} B_iL_{ij}e_j(t)\| \\ &\leq \|A_{K,i}\|\|x_i(t)\| + \|B_iK_i\|\|e_i(t)\| + \sum_{j\in N_i}\|B_iL_{ij}\|\|e_j(t)\|.\end{aligned}$$

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 (B.14) in this interval

$$\begin{aligned}\|e_i(t^i_k + \bar{\tau}^i_k) - e_i(t^i_k)\| \leq \Big(&\|A_{K,i}\|\kappa(V_i)\big(\frac{\mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t^i_k}}{|\lambda_{max}(A_{K,i})|-\alpha} + e^{-|\lambda_{max}(A_{K,i})|t^i_k}\|x_i(0)\|\big) \\ &+ \mu_i c_1(1+e^{\alpha\tau^\star})e^{-\alpha t^i_k}\Big)\bar{\tau}^i_k.\end{aligned}$$

Denote $k_{1,i} = \|A_{K,i}\|\kappa(V_i)\|x_i(0)\|$ and $k_{2,i} = (\|A_{K,i}\|\kappa(V_i)\frac{1}{|\lambda_{max}(A_{K,i})|-\alpha} + 1)\mu_i c_1$. From (B.13) in Preposition 5.2, it follows that the upper bound on the delay satisfies

$$\Big(k_{1,i}e^{-|\lambda_{max}(A_{K,i})|t_k^i} + k_{2,i}(1+e^{\alpha\tau^\star})e^{-\alpha t_k^i}\Big)\bar{\tau}_k^i = c_1 e^{-\alpha(t_k^i+\bar{\tau}_k^i)}.$$

It yields

$$\Big(\tfrac{k_{1,i}}{c_1}e^{-(|\lambda_{max}(A_{K,i})|-\alpha)t_k^i} + \tfrac{k_{2,i}}{c_1}(1+e^{\alpha\tau^\star})\Big)\bar{\tau}_k^i = e^{-\alpha\bar{\tau}_k^i}.$$

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 $(\frac{k_{2,i}}{c_1} + \frac{k_{2,i}}{c_1}(1+e^{\alpha\tau^\star}))\bar{\tau}_k^i$ for $\alpha < |\lambda_{max}(A_{K,i})|$. Hence, the most conservative bound on the delay $\tau^\star$ is given by

$$\tau^\star = \min\{(\tau^\star)^i,\ i = 1, \ldots, N\},$$

where $(\tau^\star)^i$ are the solutions of

$$(\tfrac{k_{1,i}}{c_1} + \tfrac{k_{2,i}}{c_1}(1+e^{\alpha(\tau^\star)^i}))(\tau^\star)^i = e^{-\alpha(\tau^\star)^i}.$$

B.8 Proof of Theorem 5.4

The state at any time is given by

$$x_i(t) = e^{A_{K,i}t}x_i(0) + \int_0^t e^{A_{K,i}(t-s)}(B_iK_ie_i(s) + B_i\sum_{j\in N_i}L_{ij}e_{j\to i}(s))ds.$$

Under the UwR protocol, it holds that $\|e_i(t)\| \leq c_1 e^{-\alpha t}$, and $\|e_{j\to i}(t)\| < c_1(1 + e^{\alpha\tau^\star})e^{-\alpha t}$. hence, following the same steps than in the proof of Theorem 5.3, it yields

$$\|x_i(t)\| \leq \kappa(V_i)\Big(\tfrac{\bar{\mu}_i(\tau^\star)c_1e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha} + e^{-|\lambda_{max}(A_{K,i})|t}(\|x_i(0)\| - \tfrac{\bar{\mu}_i(\tau^\star)c_1e^{-\alpha t}}{|\lambda_{max}(A_{K,i})|-\alpha})\Big),$$

where $\bar{\mu}_i(\tau^\star) = \|B_iK_i\| + \sum_{j\in N_i}\|B_iL_{ij}\|(1+e^{\alpha\tau^\star})$.
In the interval $[t_k^i, t_k^{i\to j})$, $\dot{e}_{i\to j}(t) = -\dot{x}_i(t)$ holds. Thus, it can be derived easily that

$$\|e_{i\to j}(t_k^{i\to j}) - e_{i\to j}(t_k^i)\| \leq \Big(k_{1,i}e^{-|\lambda_{max}(A_{K,i})|t_k^i} + \big(k_{2,i} + k_{3,i}(1+e^{\alpha\tau^\star})\big)e^{-\alpha t_k^i}\Big)\tau_k^{i\to j},$$

$k_{1,i}, k_{2,i}$ and $k_{3,i}$ defined in (5.24)-(5.26).
According to Proposition 5.2, $\|e_{i\to j}(t_k^{i\to j}) - e_{i\to j}(t_k^i)\| < c_1e^{-\alpha t_k^{i\to j}}$. And the upper bound on the delay is the minimum value of $(\tau^\star)^i$ which solves

$$\Big(\tfrac{k_{1,i}}{c_1} + \tfrac{k_{2,i}}{c_1} + \tfrac{k_{3,i}}{c_1}\big(1+e^{\alpha(\tau^\star)^i}\big)\Big)(\tau^\star)^i = e^{-\alpha(\tau^\star)^i}.$$

B.9 Proof of Theorem 5.5

From 5.34, the state at any time is given by

$$x(t) = e^{(A_K+\Delta)t}x(0) + \int_0^t e^{(A_K+\Delta)(t-s)}\mathrm{M}\overrightarrow{e}(s)ds.$$

According to Lemma 5.1, the error $\overrightarrow{e}(s)$ is bounded by $\bar{c}_0$. Moreover, since A_K is diagonalizable, $e^{(A_K+\Delta)t}$ can be bounded using (4.12). Thus, it follows

$$\begin{aligned}\|x(t)\| \leq& \kappa(V)\Big(\|x(0)\|e^{-(|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|)t} \\ &+ \tfrac{\|\mathrm{M}\|\bar{c}_0}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|}\big(1-e^{-(|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|)t}\big)\Big).\end{aligned}$$

Reordering terms and noting that $\|\mathrm{M}\|$ is bounded by μ_{max} because is a block diagonal matrix, it falls out (5.37).

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

$$\|e_{i\to j}(t_k^{i\to j}) - e_{i\to j}(t_k^i)\| \leq \int_{t_k^i}^{t_k^{i\to j}} \|\dot{e}_{i\to j}(s)\|ds \leq \int_{t_k^i}^{t_k^{i\to j}} \|\dot{x}_i(s)\|ds \leq \int_{t_k^i}^{t_k^{i\to j}} \|\dot{x}(s)\|ds,$$

since $x_{b,i\to j}$ remain constant in the interval and $\|\dot{x}_i(s)\| \leq \|\dot{x}(s)\|$.

Because $\|\dot{x}(s)\| \leq \|A_K + \Delta\|\|x(s)\| + \|\mathrm{M}\|\|\overrightarrow{e}(s)\|$, following equivalent steps as in Theorem 4.2, it yields

$$\|e_{i\to j}(t_k^{i\to j}) - e_{i\to j}(t_k^i)\| \leq \Big(\|A_K + \Delta\|\kappa(V)(\|x(0)\| + \frac{\|\mathrm{M}\|\bar{c}_0}{|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|}) + \|\mathrm{M}\|\bar{c}_0\Big)(t_k^{i\to j} - t_k^i).$$

According to Assumption 5.1, no event occurs before the broadcasted state is successfully received and, therefore the increase of the error in the interval $[t_k^i, t_k^{i\to j})$ is bounded by c_0, giving the upper bound on the delay (5.36).

B.10 Proof of Theorem 5.6

The state of the system at any time is given by

$$x(t) = e^{(A_K+\Delta)t}x(0) + \int_0^t e^{(A_K+\Delta)(t-s)}(BKe_x(s) + Be_u(s))ds.$$

The error e_x is bounded by $\sqrt{N}(c_{x,0} + c_{x,1}e^{-\alpha t})$ and the bound on e_u is derived in Lemma 5.3. Moreover, as already proved, it holds that

$$\|e^{(A_K+\Delta)t}\| \leq \kappa(V)e^{-(|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|)t}$$

With these considerations, the bound on $x(t)$ can be calculated following the used methodology in the previous proofs to derive (5.50), showing that the system is globally ultimately bounded. Furthermore, (5.50) is upper bounded by

$$\|x(t)\| \leq \sigma_1 + \kappa(V)\|x(0)\|e^{-(|\lambda_{max}(A_K)| - \kappa(V)\|\Delta\|)t} + \sigma_2 e^{-\alpha t}, \tag{B.15}$$

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 $\|\dot{e}_i(t)\| \leq \|\dot{x}_i(t)\| \leq \|\dot{x}(t)\|$, and $\|\dot{x}(t)\|$ can be bounded

according to (5.45). Thus,

$$\|e_i(t)\| \leq \int_{t^*}^{t} (\|A_K + \Delta\|\|x(s)\| + \|BK\|\|e_x(s)\| + \|B\|\|e_u(s)\|)ds.$$

If $x(t)$ is bounded according to (B.15), and the the corresponding bounds on e_x and e_u are considered, it leads to the following lower bound for the inter-event time

$$T_{x,min} = \frac{c_{x,0}}{\gamma_1 + \sqrt{N}(\gamma_2 + \gamma_3 + \gamma_4)},$$

where

$$\begin{aligned}
\gamma_1 &= \kappa(V)\|x(0)\|\|A_K + \Delta\| \\
\gamma_2 &= (\|BK\| + \|B\|\|\mu(K)\|_{\max})c_{x,0}\big(1 + \tfrac{\kappa(V)\|A_K+\Delta\|}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|}\big) \\
\gamma_3 &= (\|BK\| + \|B\|\|\mu(K)\|_{\max})c_{x,1}\big(1 + \tfrac{\kappa(V)\|A_K+\Delta\|}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|-\alpha}\big) \\
\gamma_4 &= \|B\|c_u\big(1 + \tfrac{\kappa(V)\|A_K+\Delta\|}{|\lambda_{max}(A_K)|-\kappa(V)\|\Delta\|}\big).
\end{aligned}$$

C
Software

This appendix provides additional information about the applications developed in this thesis and described in this manuscript in chapters 3 and 6.

First, screenshots of the LabVIEW applications are enclosed. The GUI as well as the block diagrams of the CAL and the PAL, in which the main blocks are highlighted, are depicted.

Secondly, complementary information of the MaSS tool is given in the developed user manual, which is also available at `http://lab.dia.uned.es/mass`. A more detailed description of the features of the simulator, and additional examples of usage are provided.

C.1 Implementation of the CAL and the PAL in LabVIEW

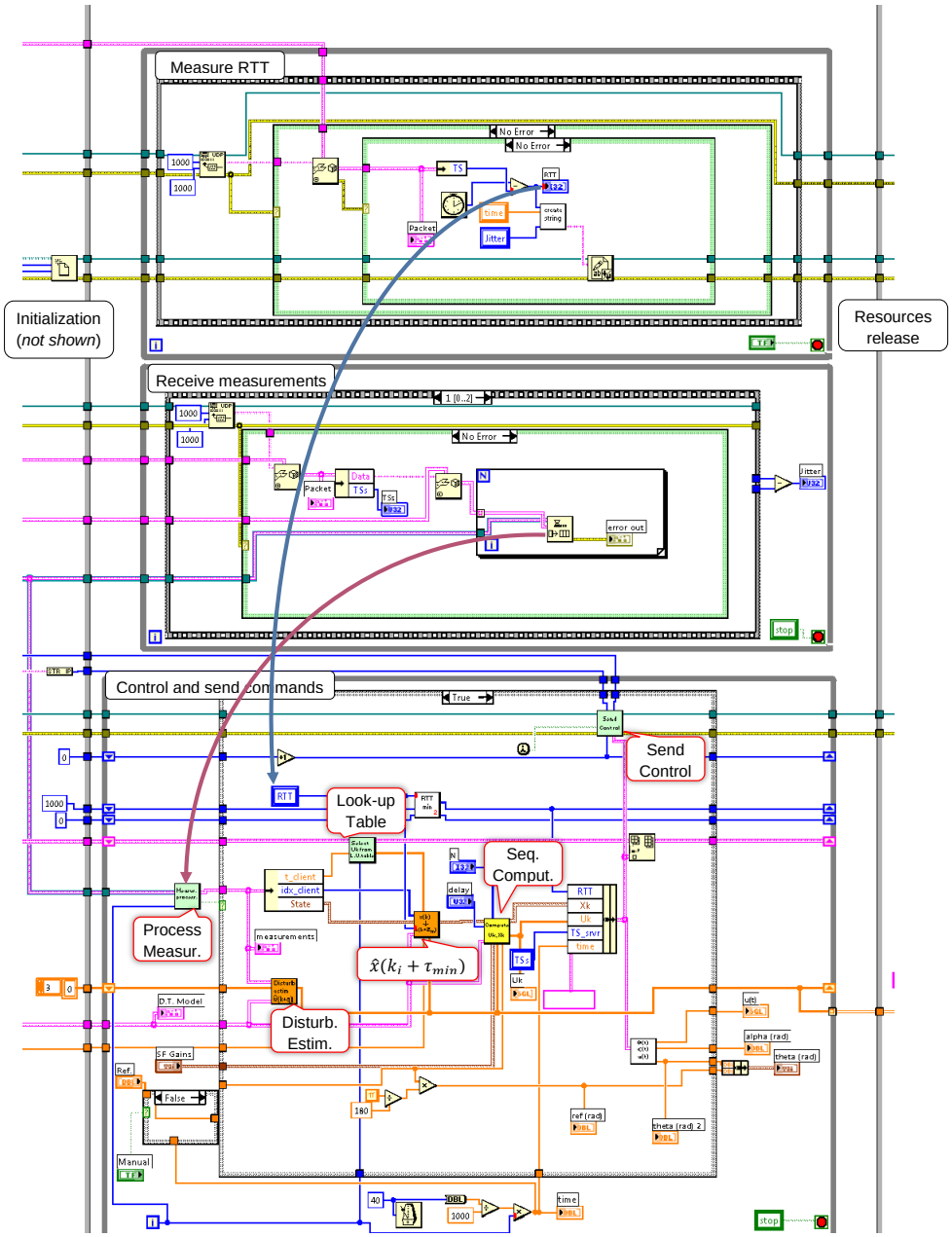

Figure C.1: Screenshot of the implementation of the CAL in LabVIEW.

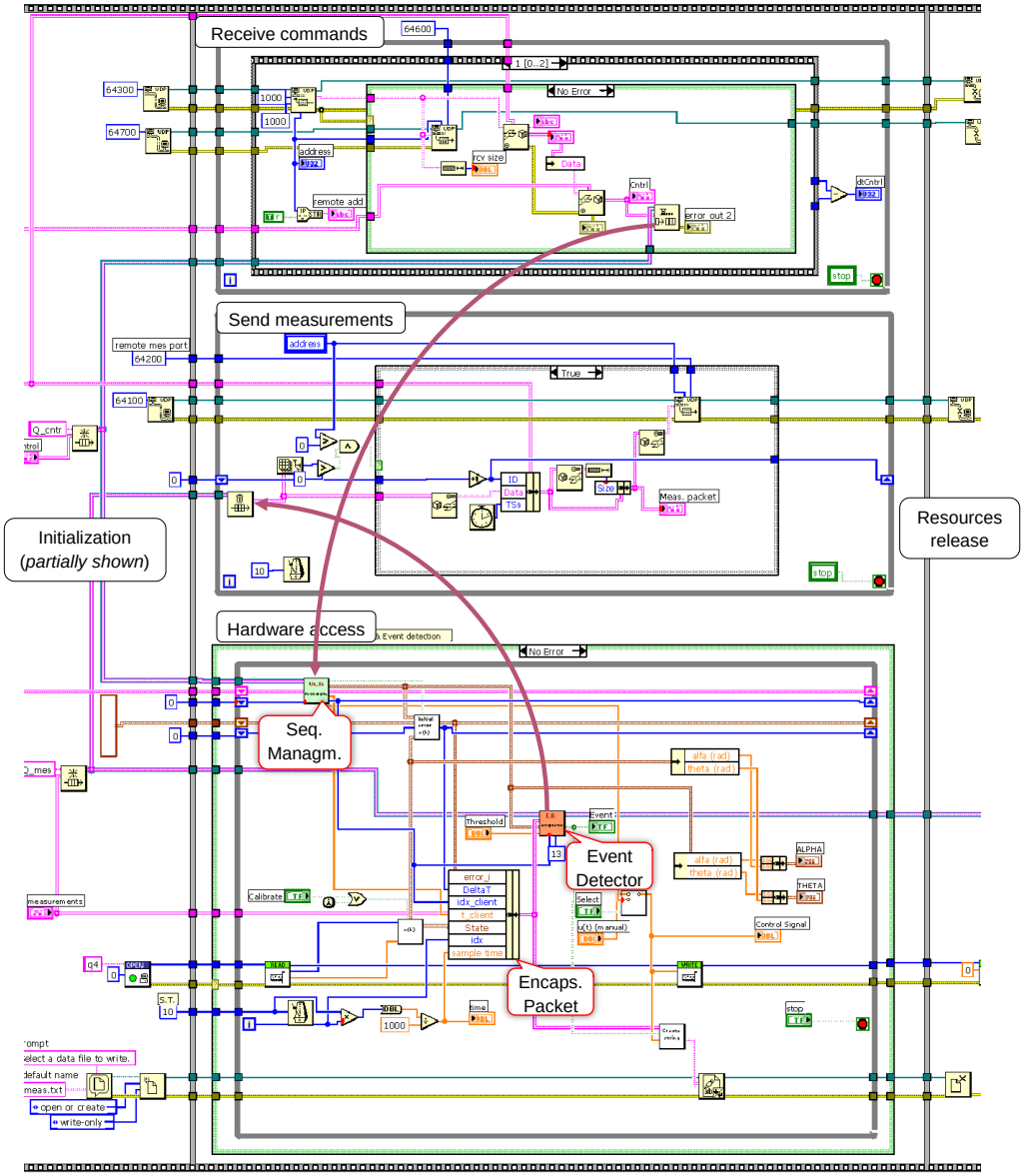

Figure C.2: Screenshot of the implementation of the PAL in LabVIEW.

Figure C.3: Screenshot of the GUI at the Client in LabVIEW.

Figure C.4: Screenshot of the GUI at the Server in LabVIEW.

C.2 User Manual of MaSS

The MaSS simulator is available online at `http://lab.dia.uned.es/mass`, where users can download a version of the simulator, view a video in which the usability of the tool is shown, and have access to a user manual. This user manual is reproduced next, and the cover is depicted in Figure C.5.

C.2.1 Background

What is a multi-agent system?

In a multi-agent system (MAS), a number of entities (agents) work together to cooperatively solve problems.

The application fields of MAS are quite heterogeneous. A group of autonomous vehicles which coordinately move to fulfill an objective or individual particles that interact with each other and model a thermodynamical system are examples of MAS. In this case, we are interested in the first example of systems.

The behavior of the system depends on the agents dynamics and the interconnections between the agents.

What is the Consensus Problem in a multi-agent system?

The agents of a MAS reach a common state (the average, the maximum, the minimum, etc.) from a reduced knowledge of the overall system. For instance, if the final state is the average of all initial states we call it average consensus.

The Consensus Problem has many applications and one of them is the autonomous vehicle formation, since reaching the formation is analogous to reaching Consensus.

What is the system topology?

The system topology describes how the agents interact with each other. In this case, interaction means communication.

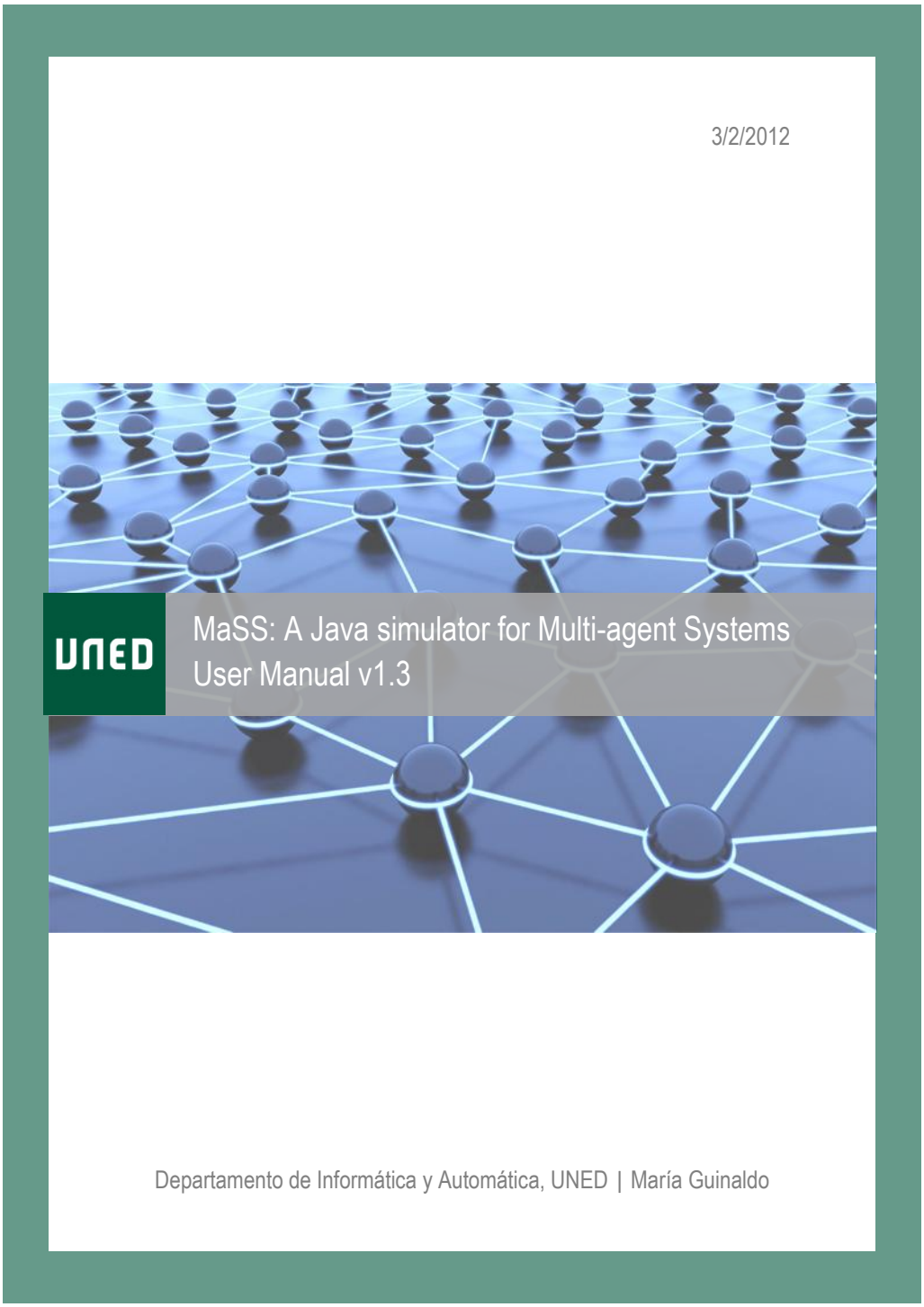

Figure C.5: Cover of the user manual of MaSS available at `http://lab.dia.uned.es/mass`.

Each agent can communicate with a reduced number of agents that we denote as neighbors. Each agent only has access to its own state and its neighbors'.

The topology is described by a graph whose vertices are the agents and the edges are the communication links.

What is the communication law of the system?

The agents communicate with each other through a network to give information about its state. The instants of time at which the communication occurs are discrete-time events, and so the communication law determines when these events take place.

Traditionally, the information is sent periodically, for example every 100 ms. One alternative is to use a communication law based on events in which the condition to send information does not depend on the instants of time but on the state of the agent. It seems logical that if the state of one agent has not varied there is no need of sending the same value periodically.

Some network concepts

A protocol is a set of rules that sets out how the information flows between two systems. A protocol gives specifications about the message formats, how to detect and correct errors, etc.

The network delay gives how much time it takes for a bit to go from one node to another in the network. The delay is influenced by several factors as queues, propagation delays, transmission rate, etc.

A packet (a message) is lost when the data sent through the network do not reach the final destination. The number of packet losses is influenced by the channel congestion, the degradation of the signal, etc.

C.2.2 The Graphical User Interface

The graphical user interface of the simulator has six panels and menu, which are labeled by number in Figure C.6 from 2 to 7 and 1, respectively. The most important panels are 3 and 4. In the first one, an interactive view of the simulation

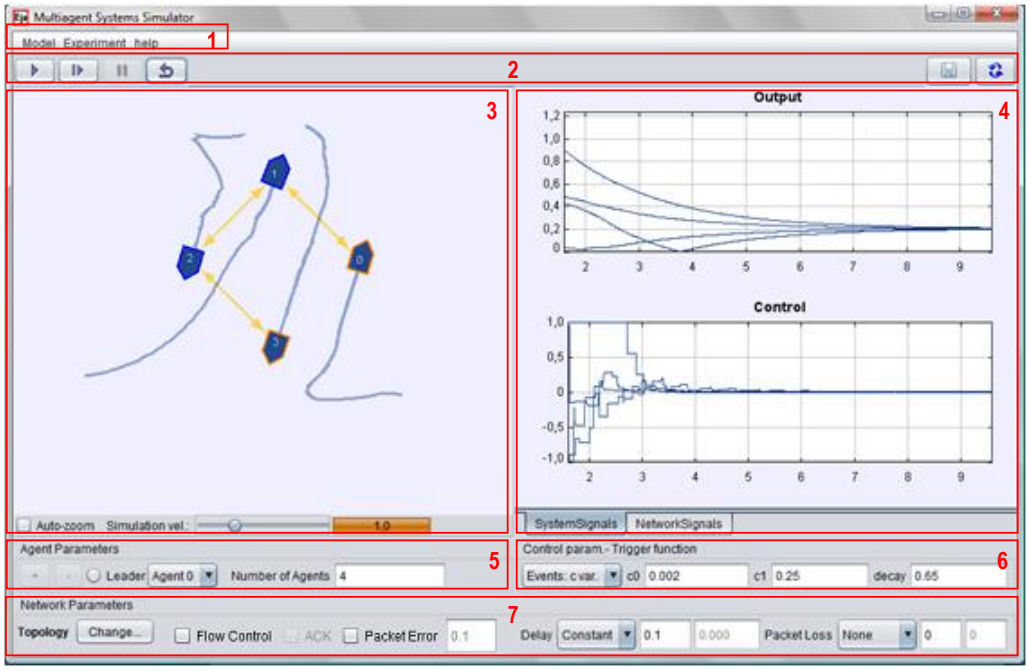

Figure C.6: Graphical User Interface.

of the agents is shown, and in the second one, a set of graphs provides information about the formation control and network status. The panel 5 allows users to change the number of agents and to define a group leader. The trigger mechanism is defined through the panel 6. All the configurable network parameters can be modified in the panel 7. Moreover, in the menu at the top of the interface, users can select the dynamical model of the agents and also the experiment to perform. Finally, the panel 2 contains the basic buttons to play, pause, reset and step the simulation, and an additional button to save the data into a Matlab file in order to perform further analysis.

We next describe in more detail each of the elements of the interface aforementioned.

Main menu: Performing Experiments and Changing the Model Dynamics

- *Model:* The vehicles have been modeled as non-honolomic. If the control signals are the linear and the angular velocity we have a first order model. On the other hand, the second order model uses the force and the torque as the control signals.

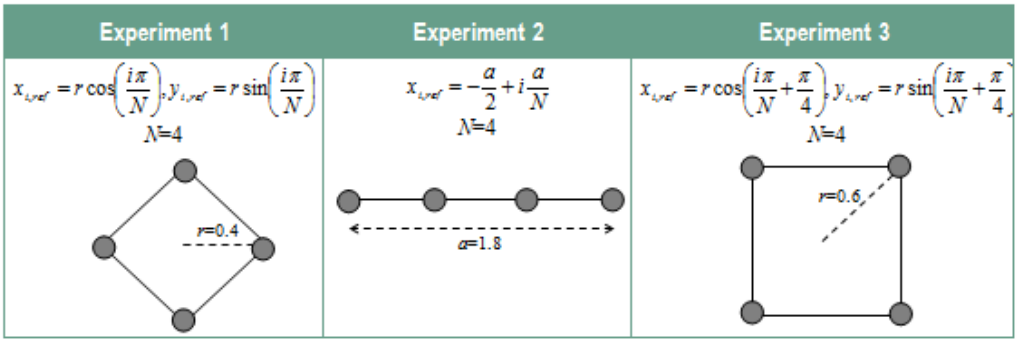

Figure C.7: Prefixed experiments.

- *Experiment:* There are three prefixed experiments, whose configurations are shown in Figure C.7. Moreover, the user can save the system configuration of a certain instant of time in a file, and restore this configuration at any time loading the corresponding file.

- *Help:* It gives access to this document.

Panel 2: Controlling the simulation execution

On the left side of the panel, users can find the buttons to start, step, stop, and restart the simulation.

On the right side, a button allows users to save some variables into a Matlab file, which can be executed afterwards to generate the corresponding graphs. If users push once the button, the icon changes and the data is stored in a buffer. The variables that are saved are the time, the coordinates (x, y) of each agent, the distance to the formation of each agent, the packets transmission rate and the performance of the network. If users push again the button, a dialogue box allows us to specify the name file to save the data. This file must have .m extension. If the code is load and run in Matlab, three graphs similar to the ones shown panel 4 should appear.

Panel 3: Interactive view of the multi-agent system

The view of the interactive simulation is shown in the panel 3. The agents are labeled from 0 to $N - 1$ and and traces show the path followed by the agents.

The arrows represent the communication links between the agents: an arrow from agent i to agent j means that agent i sends information to agent j.

The agents can be dragged to a new position, and the overall system moves to recover the formation, if the system is under control. The agents are dark blue colored and, if there is a leader of the group it is in light blue. The outline of an agent changes to orange when transmitting a packet and goes back to its normal state when the data is received.

At the bottom, we have two additional options:

- *Auto-zoom:* To zoom-in or zoom-out the simulation view.
- *Simulation vel.:* It speeds up or slows down the simulation. This is useful to adjust the movement of the agents to a real system.

Panel 4: Plots of system and network events

The panel 4 has two different tabs. The first one shows the distance to the formation at the top and the angular velocity (first order model) or the torque (second order model) at the bottom.

The second tab has two graphs as well. The first one shows the average delay for all the existing links in the network. The second one shows the last sent packet for a slot of time (see Figure C.8). Each arrow connects two lines that represent two agents. For example, if there is an arrow that connects 3 to 2 means that the vehicle 3 sent a packet to the vehicle 2. The x coordinate represents the time. The origin of an arrow represents the instant of time at which a packet was sent, and the end is the time at which it was received.

If we look at Figure C.8, we see that there are three types of arrows. The orange ones indicate that the packet was successfully received. The red arrows mean that the packet was dropped and did not reach the destination. Finally, a green arrow means that the packet was discarded because a packet with more recent information arrived before. This option can be activated in the *Flow Control* checkbox. Acknowledgments of packets are also drawn if the *ACK* checkbox is marked.

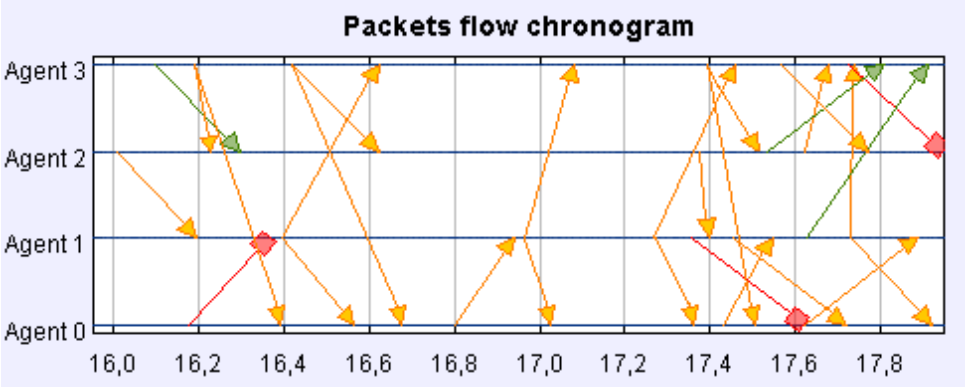

Figure C.8: Example of chronogram. Delivered packets are in orange, red arrows are lost packets, and green arrows correspond to discarded packets.

Below this chronogram, two indicators show the packets transmission rate computed on-line and a performance index which is updated every 2 seconds. This performance measures the packet delivery ratio of the network.

Panel 5: Modifying the number and type of agents

This panel allows users to add or withdraw agents and define a leader in the group. The number of agents is changed through the corresponding indicator. When a new agent is added, its position is generated randomly.

In order to define a leader, the user has to select the number of the agent from the drop-down list and then mark the checkbox *leader*. The leader differs from the rest of the agents because it sends its state to its neighbors but it does not receive information from any other agent, and so the rest of the agents move "around" the leader to reach the formation.

Panel 6: Defining Communication Strategy and Triggering Mechanism

The parameters referring to the control and communication triggering mechanism can be found in the panel 6. Three different choices can be made through the drop-down list.

- *Periodic:* Each agent sends periodically its state to its neighbors. For a first order model, the state are the coordinates and for a second order model the

state also includes the velocities in the x and y directions. The configurable parameter is the sampling period, T_s.

- *Events: c cte.:* In this case the packets are not sent periodically but when the error reaches a certain constant threshold. The error is defined as the difference between the current state and the last broadcasted state. The threshold is denoted by c and can be adjusted by the user.

- *Events: c var.:* In this case, the threshold to trigger the event is not constant and changes with time as $c = c_0 + c_1 e^{-\alpha t}$. Thus, there is a constant term and an exponential term decreasing with time. The user can configure the parameters c_0, c_1, and α.

Panel 7: Configuring Network Topology and Parameters

Some aspects of the Network can be configured in this panel.

- *Topology:* If users press the button *change...*, a window shows up and gives us instructions to add or withdraw links. A link is added or deleted by clicking first on the agent i and then on the agent j.

- *Flow Control:* If this checkbox is marked, packets containing old information are discarded.

- *ACK:* If this checkbox is marked, every agent does not sent new information after getting confirmation of reception of the previous packet.

- *Packet error:* This option allows to add bit error transmission into packets.

- *Delay:* Using this option, users can configurate a constant or random network delay.

- *Packet loss:* The probability of losing packets can be set up by using this option. The user can set a constant probability or, if the option *Model 1* is selected, the probability is calculated based on the transmission. If this rate increases, the probability of collision increases and so do packet losses.

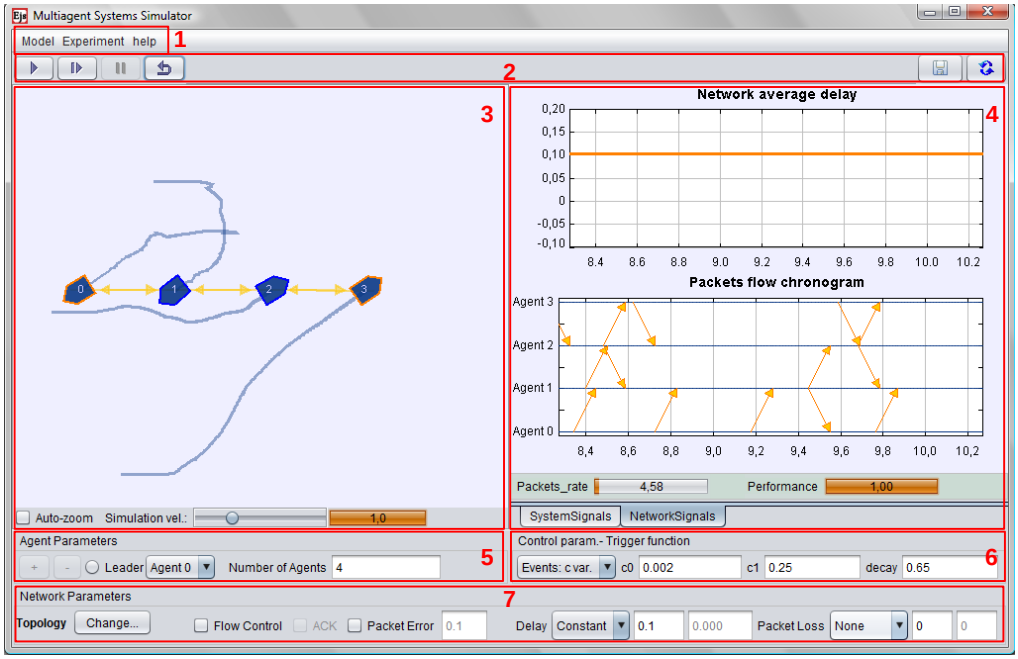

Figure C.9: Example of usage, snapshot 1.

C.2.3 Examples of Usage

Example 1: Interacting with the GUI

Let us now see an example of usage of the simulator. This example is the same than the available video at `http:/lab.dia.uned.es/MaSS`. Let us assume that the initial configuration is generated randomly by the code and that we want to save all the experiments in a Matlab file. Thus, we press the corresponding button to store some variables in a buffer (see Section C.2.2) and we let the system evolve to reach the formation of the *Experiment 2*. As a result, we should get something similar to Figure C.9.

Let us make few changes in the configuration. We first set the agents model to second order systems, hen we modify the network topology by deleting the link $1 \rightarrow 2$ and adding links $1 \rightarrow 3$ and $3 \rightarrow 0$. Furthermore, we we modify the position of the agents to the given in the Experiment 1 by selecting it from the Menu. If we restart the simulation, the system should evolve to something similar to Figure C.10. Note that the changes in the topology have been correctly performed. Moreover, since we have set the model to a second order system, the

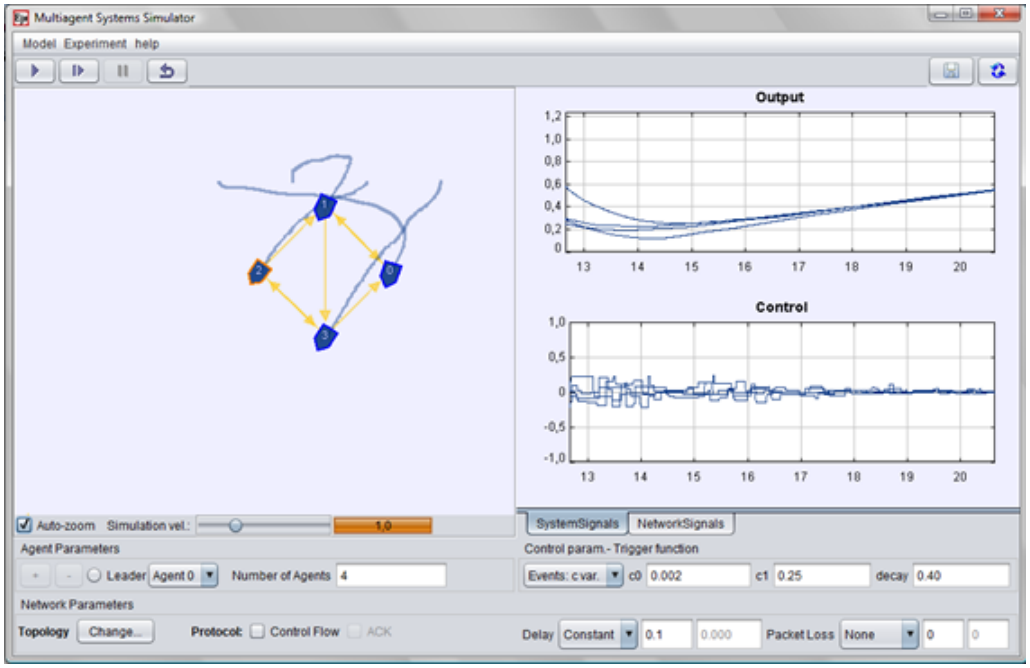

Figure C.10: Example of usage, snapshot 2.

system reaches the formation but it moves with a constant velocity (see panel 4, Figure C.10). Observe that we have activated the *Auto-zoom* checkbox so that we can keep following the trajectory of the system moving with constant velocity.

If two new agents are added (type 6 in the Number of Agents textbox) and the agent 3 is marked as a leader, the formation will follow it. Note that the topology is restored when the number of agents is changed.

Let us now change to a periodic communication strategy with $Ts = 0.2$ s, set the delay to random with a maximum value of 0.3 s, the probability of packet losses to 0.1 and mark the *Flow Control* checkbox. Note that every agent sends to its neighbors packets every 0.2 s. These packets can have different lengths (the delay is random). Since the maximum delay is larger than the sampling period, some packets should be discarded because the *Flow Control* is active. Moreover, some packets are dropped since the probability is set to 0.1.

Example 2: How to save the data into a Matlab file

As explained in Section C.2.2, the panel 2 has a button to save the workspace into a Matlab file. An example of how to do this is given below.

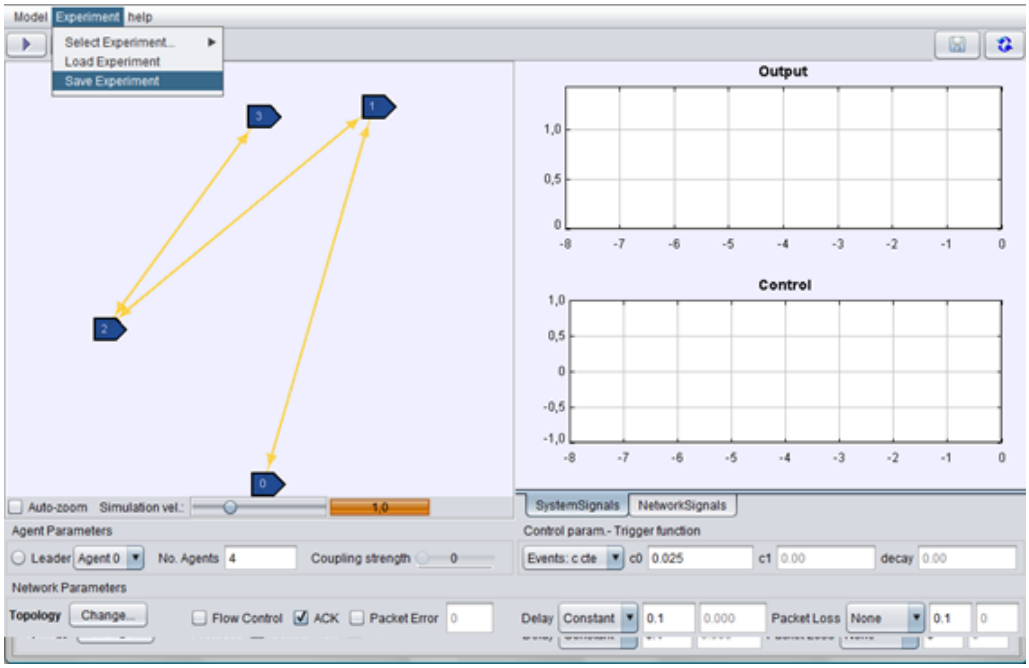

Figure C.11: Initial view of Example 2.

Assume that the initial configuration is the one given in Figure C.11 and that we save this configuration to load it again in order to to compare the performance of the system under two different conditions:

1. Packets drops are modeled as a Bernoulli distribution of probability $p = 0.1$.

2. Packet drops are influenced by the transmission rate assuming that when this rate increases so does the probability of packet collisions.

The number of agents is 4, the communication mechanism is based on events with constant threshold $c_0 = 0.025$, the ACK option is marked and there is a constant delay of 100 ms. To save the experiment, click on *Experiment*, *Save Experiment*, and name the file. Press the button to start recording the data and then play the simulation. Once the system reach the formation, we press again the mentioned button to stop recording the data and we type the name of the Matlab file to store them.

Let us restore the initial configuration by clicking on *Experiment* and then on *Load Experiment* to select the file where we saved it before. We next change the model of packet losses by selecting *Model 1* in the list and we repeat the same steps to save the variables into a Matlab file.

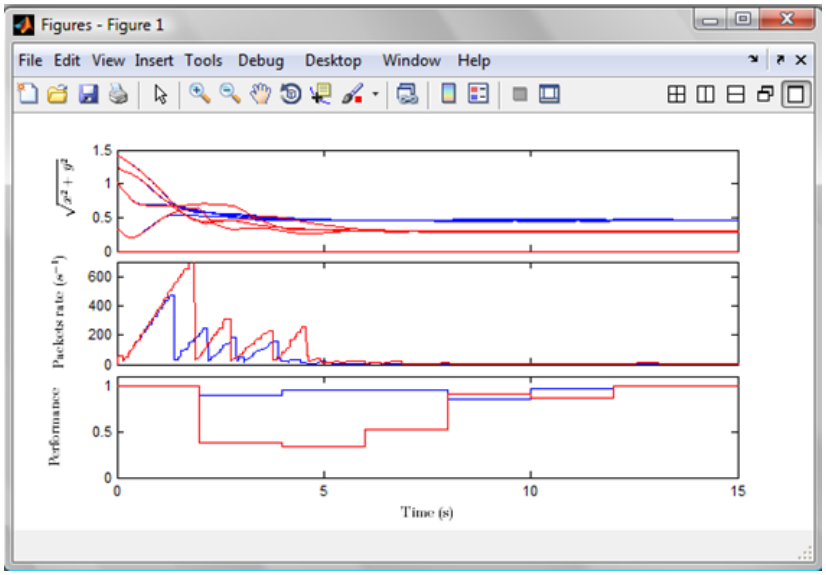

Figure C.12: Matlab figure for the experiments of Example 2.

Afterwards, if both Matlab files are open and executed, something similar to Figure C.12 will be obtained.

The first graph depicts the output, the second one the packets transmission rate per second and the third one the performance of the network, which is evaluated every 2 seconds as the packet delivery ratio. One can conclude that the degradation of the performance of the network has a direct effect into the performance of the system, since the formation is reached later in the second case (red lines) and the trajectories are less optimal.

D

Resumen en Castellano

Resumen

El cierre de lazos de control a través de redes de comunicaciones se ha ido extendiendo y ha incrementado su popularidad a medida que los dispositivos de red han abaratado sus costes. Muchas son las ventajas de usar redes de comunicación digital para el control, y no sólo desde el punto de vista de sus aplicaciones. Sin embargo, su uso también implica una serie de retos como las limitaciones que impone en el ancho de banda, retardos en la transmisión de información, e incluso pérdidas de datos.

A continuación se presenta un resumen en castellano de la tesis que incluye una reflexión sobre la influencia del tipo de red, la arquitectura del sistema, y las líneas de investigación más relevantes para hacer frente a las imperfecciones de la comunicación que caracterizan a estos medios. También se hace una revisión del control basado en eventos, ya que se ha demostrado su eficiencia en control a través de redes.

Asimismo, se incluyen las contribuciones, un pequeño resumen de cada capítulo, así como un listado de las publicaciones y de los proyectos de investigación en los que se ha participado.

Por último, el resumen finaliza con las conclusiones que se extraen de la tesis.

D.1 Control a través de redes

D.1.1 Cuestiones generales

El desarrollo técnico en las últimas décadas del mundo de las redes de comunicaciones ha posibilitado su aplicación a los sistemas de control. En la actualidad, el control sobre redes es de hecho una rama de investigación de gran importancia, como demuestra la existencia de al menos dos áreas técnicas específicas de IFAC (*International Federation of Automatic Control*), y un creciente número de conferencias y *workshops* especializados en este campo.

En los Sistemas de Control en Red (SCR) los diferentes elementos de un lazo de control (sensores, actuadores y controladores) se conectan a través de un medio de comunicación digital con ancho de banda limitado. El uso de redes compartidas de propósito múltiple para conectar elementos que están espacialmente distribuidos tiene varias ventajas que, por otro lado, son el principal motivo de su éxito:

- Ofrecen arquitecturas flexibles, haciendo más fácil reconfigurar partes del sistema o añadir nuevos elementos.
- En general, se reducen los costes de instalación y mantenimiento, debido a la reducción del cableado que se requiere en una arquitectura punto a punto.
- Como consecuencia de ello, resulta más fácil diagnosticar y detectar fallos.

Los SCR han abierto también un nuevo espectro de posibles aplicaciones en el mundo real, como redes de sensores móviles [HE04, OFL04], sistemas de energía distribuidos o inteligentes o *smart grids* [MAW05, BZ11] (véase la Figura D.1), sistemas de transporte inteligentes [MS06], formación de vehículos autónomos [SS05, GKKP06], vigilancia [BCM$^+$10, CM02] (véase la Figura D.2), operaciones quirúrgicas teleasistidas [MWC$^+$04] (véase la Figura D.3), entre otras.

Sin embargo, el uso de redes introduce nuevos retos y hace que el análisis y diseño de los SCR resulte más complejo. Las teorías de control convencional

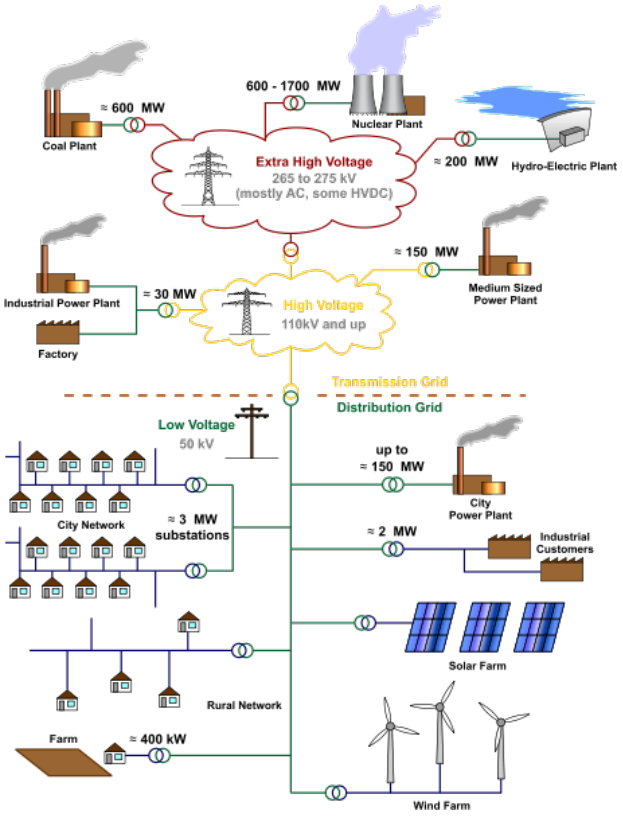

Figura D.1: Diseño genérico de redes de energía. Imagen por MBizon [CC-BY-3.0 [com13]], a través de Wikimedia Commons.

plantean una serie de hipótesis como la sincronización en el control, la ausencia de retardos en la medida y en la actuación y un ancho de banda ilimitado. Sin embargo, estas hipótesis han de replantearse antes de poder ser aplicadas a los SCR dadas sus características. Tratar de mejorar las redes de comunicaciones y los protocolos de red son sólo una solución parcial al problema. Por tanto, es necesario diseñar nuevos algoritmos de control con el fin de hacer frente a las imperfecciones de la comunicación y sus restricciones [Hee10], las cuales pueden resumirse en:

- *Ancho de banda limitado:* Cualquier red puede transmitir una cantidad de información finita por unidad de tiempo, y esto puede tener graves consecuencias en los sistemas de control. En la mayoría de las redes digitales, la información es transmitida en forma de unidades atómicas o paquetes. Dichos paquetes pueden dividirse en dos partes, el área de datos o *payload* y la cabecera, que contiene la información de control requerida para la transmisión. El máximo tamaño del área de datos es dependiente del

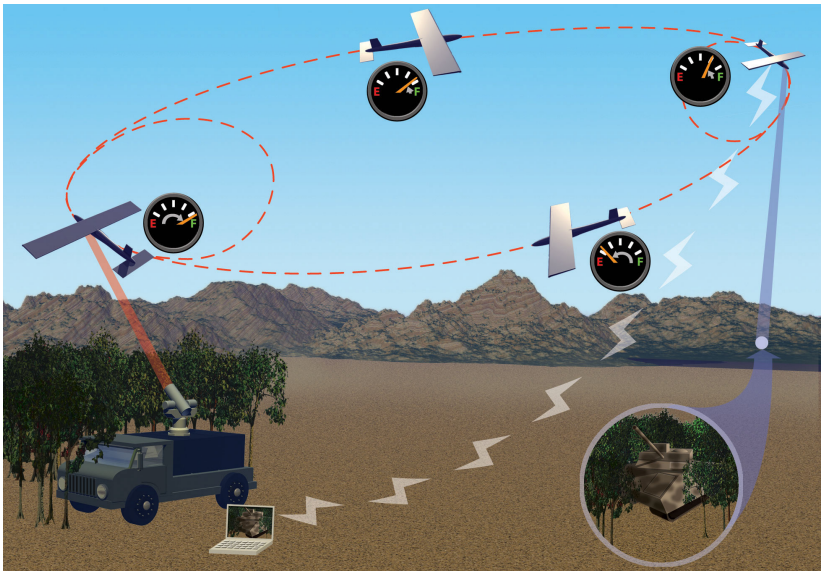

Figura D.2: Vehículos aéreos no tripulados para vigilancia y comunicaciones. Imagen tomada prestada de [Las10].

protocolo, y puede variar de los 1500 bytes en Ethernet a los 8 bytes de algunos protocolos de comunicación por radio frecuencia [mOw10].

- *Retardos variables en la comunicación:* La trasferencia de un paquete de un nodo a otro de la red no es instantáneo sino que puede suponer una cantidad variable de tiempo, que depende de condiciones variables de la red como la congestión, la calidad del canal o el protocolo. Esto puede afectar al rendimiento del lazo de control de varias maneras. Por un lado, la información transmitida es recibida con retraso. Los sistemas con retardos son en sí mismos una línea de investigación. Este área ha sido ampliamente investigada en el contexto de los SCR [NRC07], y se puede concluir que la estabilidad de estos sistemas es más compleja que la de los sistemas sin retardo. Por otro lado, los retardos pueden suponer que el sistema sea muestreado a intervalos de tiempo variables. Un número significativo de resultados (véase [HNY07] y referencias de ese artículo) han tratado de caracterizar la cota superior del tiempo inter-muestreo para la cual la estabilidad del sistema está garantizada.

- *Pérdidas de paquetes:* Un paquete puede perderse por errores en la transmisión a nivel físico entre enlaces de la red, lor la congestión del canal o

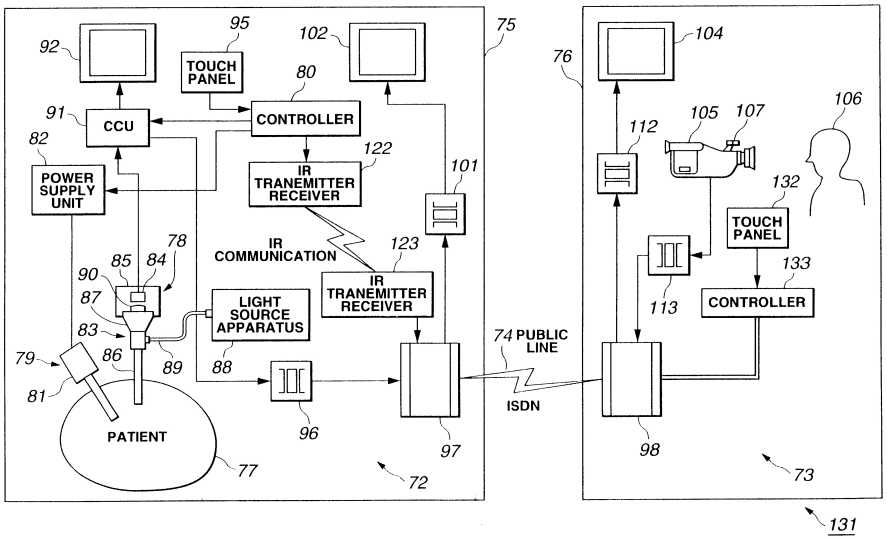

Figura D.3: Sistema de cirugía teleasistida. Imagen tomada prestada de [Uch03].

por la corrupción de los paquetes en tránsito. En aplicaciones de control un paquete puede ser descartado si contienen información desactualizada.

- *Cuantización:* Un cuantizador es una función que mapea una función real en una función a tramos con un número finito de valores constantes. En SCR, el tamaño finito de los paquetes puede introducir errores en las señales transmitidas.

Otro aspecto importante que diferencia a los SCR de un sistema de control convencional es su arquitectura. La Figura D.4 muestra la arquitectura genérica de un SCR. Puede consistir en varios subsistemas, los cuales pueden estar interconectados físicamente, al igual que sus respectivos sensores (S), actuadores (A) y controladores (C). Todos estos nodos pueden estar dispersos dentro del sistema y pueden conectarse de manera arbitraria a la red. En la Sección D.2 se detalla este aspecto.

D.1.2 Trabajos en el área de control sobre redes

La mayor parte del trabajo desarrollado en el área de los SCR se ha inclinado por modelarlos como sistemas de control en red convencionales con ciertas restricciones en la comunicación, ya descritas anteriormente. Normalmente esto supone

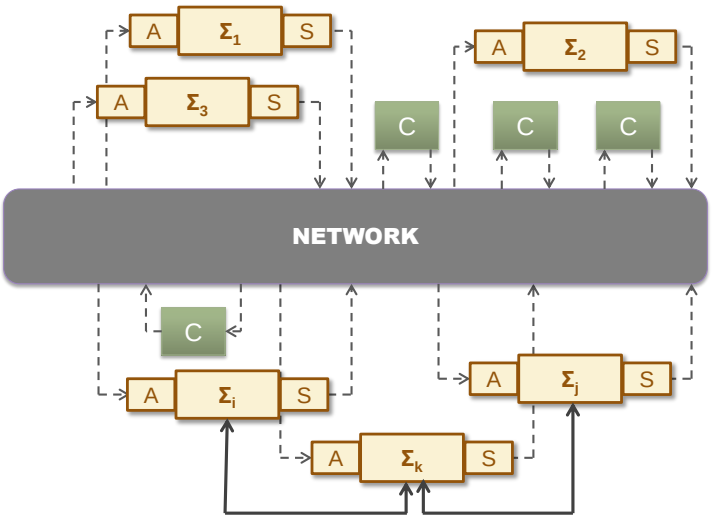

Figura D.4: Arquitectura genérica de un SCR.

el re-diseño de controladores convencionales que, además, pueden desembocar en diseños conservadores [ZLR09].

En [Hee10] se puede encontrar un resumen de las principales líneas de investigación centradas en la síntesis de controladores en red. Todos los métodos consideran límites bastante restrictivos en los retardos, intervalos de transmisión y el número máximo de pérdidas de paquetes. Otra de las restricciones más usuales es la síntesis de condiciones basadas en LMI, lo que limita el problema a plantas lineales:

- Una de las estrategias es modelar el sistema en tiempo discreto. El retardo y los intervalos inter-muestreo representan la incertidumbre del sistema, en la que se basa para derivar las condiciones de estabilidad a través de un LMI [DHvdWH11].

- La estrategia de *datos muestreados* hace uso de ecuaciones diferenciales (impulsivas) con retardo. Se pueden obtener condiciones de estabilidad basadas en LMI como resultado de la extensión de funcionales de Lyapunov-Krasovskii a plantas y controladores lineales. Véase, por ejemplo, [NHT08].

- En la estrategia de *emulación* [HTVdWN10], la estabilidad del SCR se obtiene combinando funciones de Lyapunov continuas del sistema de control en ausencia de red y el protocolo de red.

Mención especial merecen algunas contribuciones en el campo del control robusto. Se han propuesto controladores H_∞ y H_2/H_∞ para lidiar con los problemas ocasionados por la red en el lazo de control, como por ejemplo [YHL05, SS05, MOB$^+$12].

Las siguientes secciones analizan las particularidades de dos tipos de redes: las redes tipo Internet y las redes inalámbricas, y cómo puede afectar este hecho al sistema de control.

Redes tipo Internet

Los sistemas de control basados en Internet permiten la monitorización remota y el ajuste de plantas a través de Internet, lo que hace factible la recuperación datos y la reacción a posibles fluctuaciones de la planta desde cualquier parte y en cualquier momento.

Las redes del tipo Internet se basan en paquetes que pueden transportar más información que la requerida en un sistema de control convencional sin que esto suponga un mayor consumo de los recursos de la red. Por tanto, en lugar de enviar valores individuales, se pueden transmitir predicciones de tamaño limitado. Esta es la motivación del llamado *control basado en paquetes*.

Una estrategia relativamente común es recurrir a controladores basados en modelo para emular futuros estados de la planta, y así generar predicciones. La idea de combinar el control basado en paquetes con Control Predictivo (MPC, de *Model Predictive Control*) fue introducida por primera vez en el contexto de la teleoperación en [Bem98]. Desde entonces, otros autores han explotado los principios de MPC en SCR basados en paquetes [KJA06, QSG07, MJVR08, VF09, ICMS11].

Otras alternativas han sido estudiadas para intentar reducir el esfuerzo computacional que el MPC requiere. Unas de ellas es el llamado *control anticipativo* que efectúa una estimación de los futuros estados del sistema en base a un modelo que considera retardos [NH06], pero sin realizar un proceso de optimización. La secuencia de control resultante puede que no sea óptima, pero el tiempo empleado en su cálculo es insignificante comparado con el del MPC.

En [ESDCM07, GSD11] se proponen controles anticipativos para diferentes arquitecturas.

Redes inalámbricas

En los últimos tiempos, han aparecido algunos trabajos que han permitido aplicar control sobre redes inalámbricas. El interés en las redes inalámbricas proviene del bajo coste que supone su instalación y la flexibilidad que aporta prescindir del cableado. Los primeros trabajos se han centrado en el diseño del controlador en redes de sensores y actuadores bajo la hipótesis de redes ideales. Sin embargo, las imperfecciones de la comunicación no pueden obviarse, sobre todo desde el punto de vista de la implementación. Además, dichas imperfecciones son mucho más severas que en redes cableadas.

En el mundo de las telecomunicaciones, los esfuerzos se están dirigiendo a desarrollar redes inalámbricas más fiables para aplicaciones de control, de manera que se puedan garantizar valores bajos de latencia y tener en consideración las restricciones que demandan los sistemas de tiempo real [Maz10]. Asimismo, las redes de sensores y actuadores inalámbricas presentan nuevos retos con respecto a los SCR cableados. El más importante es el relativo a la eficiencia energética en dispositivos alimentados por baterías, que imponen restricciones tanto en la computación como en la comunicación de dichas redes. Hay varios factores que influyen en el consumo energético. Uno de ellos es la tasa de trasnmisión de datos, que también puede influir en el retardo y las pérdidas de paquetes. En general, se puede decir que, reduciendo el número de paquetes transmitidos por unidad de tiempo, se permite alargar la vida de las baterías de los dispositivos inalámbricos. Otro de los factores que influyen en el consumo es el tamaño de los paquetes, aunque la energía consumida por byte sea menor debido al coste que supone la cabecera de los paquetes [JER+10]. La mayoría de las implementaciones hacen uso de paquetes pequeños y de tamaño fijo, ya que la información requerida en aplicaciones de control es de pequeño tamaño.

Entre las soluciones propuestas en la literatura, se encuentra el diseño de nuevos protocolos que permiten alcanzar una alta fiabilidad y eficiencia energética

en distintas aplicaciones de redes de sensores inalámbricas, no diseñadas específicamente para el control [AKK04, BDWL10]. Más recientemente, se ha propuesto el diseño conjunto de parámetros del control y la comunicación [AAJ+11] con el objetivo de optimizar el consumo de energía a la vez que se garantiza un rendimiento en el control adecuado.

La mayoría de los trabajos existentes en la literatura considera que la transmisión se produce de manera periódica, aunque recientemente se han propuesto técnicas de muestreo aperiódico que abordan mejor los problemas de los sistemas de control inalámbricos. Una revisión de dichas técnicas se puede encontrar en la Sección D.3.

D.2 Arquitectura

En la Figura D.4 se ilustraba la arquitectura más genérica de un SCR, en la cual los nodos se encuentran repartidos por toda la red. A continuación, se particulariza esta arquitectura para dos casos concretos.

D.2.1 Sistemas con un único lazo de control

Si se dispone de un único lazo de control, tenemos tres nodos: el controlador (C), y los correspondientes sensor (S) y actuador (A). La localización de estos tres nodos da lugar a tres arquitecturas diferentes para el SCR, tal y como se muestra en la Figura D.5. Nótese que la información debe ser transmitida en tiempos discretos sobre la red. Se asume que el elemento que transmite el dato sobre la red es capaz de muestrear las señales continuas y transmitir la información a tiempo discreto. De igual modo, el elemento que recibe la información de la red debe tener la capacidad de transformar señales de tiempo discreto a señales continuas. Las líneas discontinuas representan el flujo de información a instantes de tiempo discretos que, en general, se denotan como t_k, pudiendo ser estos equidistantes o no.

Véase que se tiene en cuenta el caso en que se mide el estado completo $x(t)$ o sólo la salida del proceso $y(t)$.

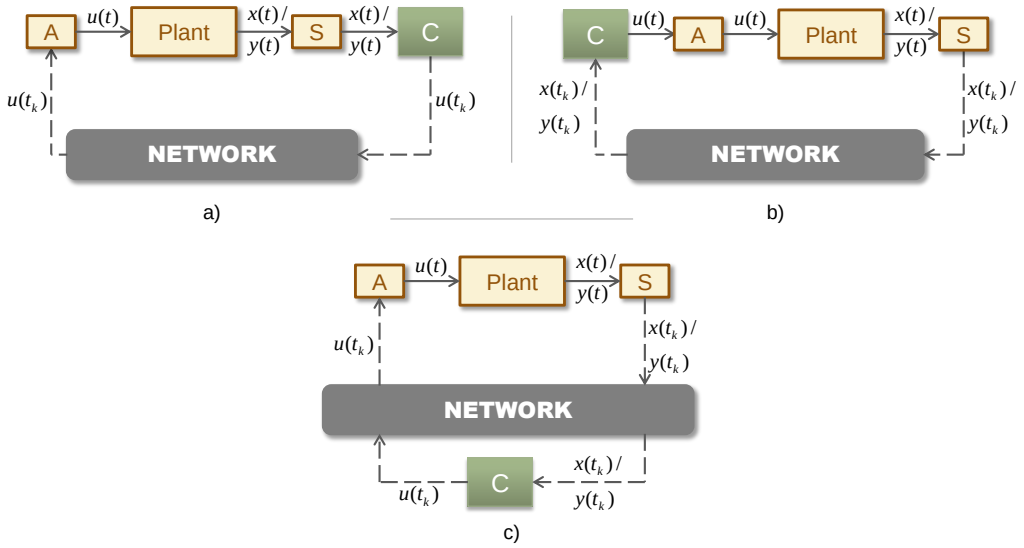

Figura D.5: Posibles esquemas de un SCR en un lazo de control.

La Figura D.5a muestra el caso en el que el controlador se encuentra junto al sensor, y las señales de control son transmitidas a través de la red. Ejemplos de trabajos en los cuales esta es la arquitectura preferida son [ESDCM07, QSG08]. En la Figura D.5b el controlador se sitúa junto al actuador y lo que se transmite a través de la red son las medidas de la planta. Este es el esquema considerado en, por ejemplo, [GCHM06, LL10, LL11b].

Un esquema más genérico se muestra en la Figura D.5c, en el que la red se encuentra a ambos lados del controlador. Cuando el controlador recibe una nueva medida, se calcula una nueva señal de control que es transmitida a la planta. Supongamos que la información es transmitida con un cierto retardo causado por la red. Mientras que en el caso de controladores *colocados* (D.5a y D.5b) se puede medir la latencia porque el cierre del lazo de control a través de la red ocurre sólo entre dos elementos, en el caso del controlador remoto los retardos del sensor al controlador y del controlador al actuador no pueden medirse de manera independiente sin asumir la sincronización de todos los elementos, y sólo se puede conocer su suma.

En esta tesis, la arquitectura que se estudia para el caso de que haya un único lazo de control se corresponde con la Figure D.5c, y se corresponde con los capítulos 2 y 3.

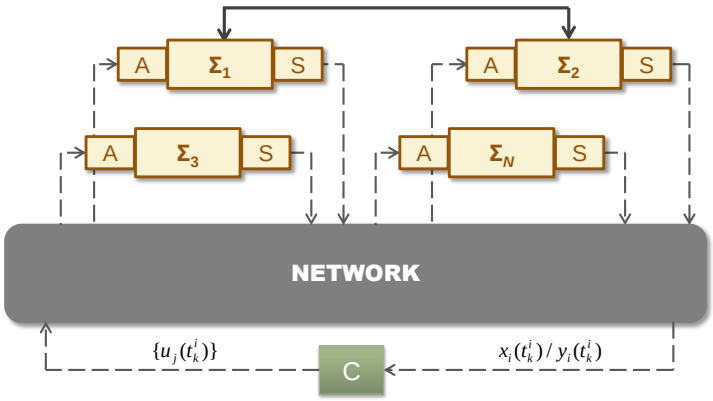

Figura D.6: Control centralizado de múltiples plantas.

D.2.2 Sistemas con múltiples lazos de control

Cuando se tienen varios lazos de control, las alternativas para ubicar los diferentes nodos son múltiples. Nos centramos en dos arquitecturas, que por otro lado son las más extendidas desde el punto de vista de la implementación.

La primera arquitectura tiene un único controlador que puede recibir información de varios sensores y enviar acciones de control a varios actuadores (*control centralizado*). En general, estos elementos pueden pertenecer al mismo sistema o a diferentes plantas.

Posteriormente, se considera una estrategia de *control distribuido*. En concreto, se asume que cada nodo en la red dispone de su propio controlador y que la necesidad de comunicación con otros nodos es por distintos objetivos del control.

Control centralizado

El control centralizado propuesto para una arquitectura multi-lazo se muestra en la Figura D.6. El sensor transmite a través de la red al controlador las medidas adquiridas. A pesar de que en la Figura D.6 se considera que cada planta tiene un único sensor y un actuador, también se contempla el caso de tener una única planta con varios sensores y actuadores.

Dependiendo de la complejidad del problema, el control tendrá que cambiar entre diferentes sub-controladores si, de hecho, hay varias plantas cada una con su controlador asociado. También se puede diseñar un único controlador multi-

variable para el segundo tipo de problema mencionado.

Independientemente de la naturaleza del problema, el controlador tendrá que procesar diferentes medidas recibidas de los sensores, denotadas como $x_i(t_k^i)$ (medida del estado) o $y_i(t_k^i)$ (medida de la salida), y calcular la(s) correspondiente(s) señal(es) de control $u_j(t_k^i)$, donde t_k^i hace referencia al tiempo de muestreo. Se considera que una medida $x_i(t_k^i)/y_i(t_k^i)$ puede suponer que varias señales de control $u_j(t_k^i)$ sean calculadas, ya que puede existir acoplamiento.

El uso de controles centralizados en red suele requerir el uso de *buffers* que inducen retardos adicionales a los ya causados por la red. Este retardo de *procesamiento*, que es el tiempo desde que se recibió una medida $x_i(t_k^i)/y_i(t_k^i)$ hasta que se comienza a calcular la correspondiente señal de control $u_j(t_k^i)$, no resulta despreciable cuando el número de lazos de control aumenta.

Aunque se pueden tomar algunas estrategias para tratar de mitigar este problema como, por ejemplo, el diseño de un planificador de red que decida qué nodo debe transmitir en cada momento [AAJ+11], el control centralizado no resulta ni conveniente ni eficiente en sistemas a gran-escala.

Por este motivo, el interés en el diseño de controles descentralizados ha ido creciendo en los últimos años. Sin embargo, conviene indicar que esta tampoco es una tarea fácil. Incluso las nociones más básicas como estabilidad se convierten en no triviales en un marco descentralizado [WD73, GSS05]. En algunas implementaciones cuando el número de sensores y actuadores no es muy grande, se demuestra que el control centralizado a través de red resulta más eficiente que el descentralizado si se puede garantizar la fiabilidad de la red [SGQ08].

Control distribuido

Existen muchos sistemas de control que han sido contruidos siguiendo una estructura descentralizada de un gran número de controladores SISO (*single-input-single-output*), permitiendo así la operación estable de la mayoría de las unidades de operación. No obstante, esta estrategia no es la solución de control más óptima ya que en estas arquitecturas los subsistemas no pueden comunicarse aunque la interacción entre ellos sea significativa. Por el contrario, el diseño de controladores

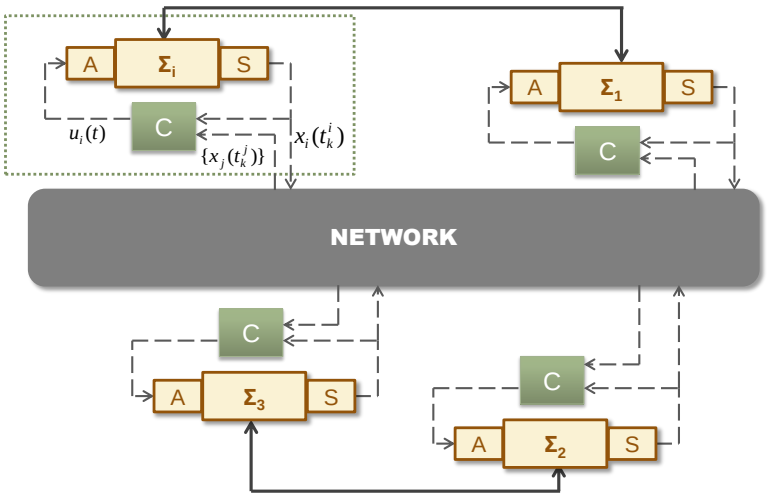

Figura D.7: Sistema de control distribuido.

distribuidos se basa en asumir que los controladores pueden compartir información. Dado que los controladores pueden necesitar comunicarse entre ellos, la red de comunicación forma parte del diseño del problema.

En un sistema de control distribuido se distinguen dos tipos de interacciones. En la primera, un subsistema puede estar físicamente interconectado con otros, es decir, el estado de un subsistema i rige la dinámica de otro sistema j. Este hecho podría ser tenido en cuenta a la hora de diseñar el controlador del subsistema j con el objetivo de compensar este efecto si el estado de i estuviese disponible en j. Esto incluye a sistemas MIMO de gran escala que pueden ser divididos en pequeñas porciones que interaccionan unas con otras. El segundo tipo de interacción proviene de la necesidad de comunicación entre los controladores para alcanzar un objetivo común. Esto lleva a lo que se conoce como control cooperativo. La terminología más usual para referirse a este tipo de sistemas, en los cuales se agrupa información de partes individuales para controlar el comportamiento global del sistema, es la de *sistemas multi-agentes*.

La Figura D.7 ilustra un esquema de control distribuido. Se puede encapsular cada nodo i (linea punteada) para incluir el subsistema y el controlador local, que recibe información en instantes de tiempo discreto t_k^i el estado local del subsistema $x_i(t_k^i)$ pero también un conjunto de estados $x_j(t_j^k)$ de otros subsistemas (también llamados agentes) medidos en instantes de tiempo diferentes t_j^k. Los agentes que transmiten información i se conocen como los vecinos de i.

D.3 Control basado en eventos

D.3.1 Preliminares

A pesar de que el mundo real es analógico, la mayoría de las aplicaciones de control se implementan en plataformas digitales que necesitan que la información en el lazo de control se intercambie de manera discreta entre los sensores, los actuadores y los controladores. Tradicionalmente, los instantes en los que este intercambio se produce son equidistantes, es decir, dados por el periodo de muestreo. La frecuencia de muestreo tiene que garantizar la estabilidad del sistema en todos los escenarios posibles, lo que a veces provoca una elección de la frecuencia realmente conservativa. Además, todas las tareas se ejecutan de manera periódica independientemente de las necesidades de la planta.

En los últimos años, ha habido un interés creciente en la idea de tener en cuenta el estado de la planta a la hora de decidir cuándo se ejecuta el control o se muestrea al sistema. En los sistemas de control basados en eventos, dichas tareas tienen lugar cuando se viola una cierta condición sobre el estado de la planta. Por tanto, hay una adaptación a las necesidades del proceso.

No hay una terminología uniforme a la hora de referirse a este concepto. Se pueden encontrar en la literatura distintas acepciones como control basado en eventos, control disparado por eventos, control *send-on-delta*, control por cruce de nivel, control autodisparado o *self-triggered*, control con mínima atención, control con atención en cualquier momento, entre otras.

A pesar de su reciente popularización, el muestreo basado en eventos no es en realidad una idea nueva. Sus orígenes se remontan a finales de los años 50 cuando [Ell59] planteaba que el método de muestreo más apropiado consiste en transmitir solamente datos cuando existe un cambio significativo en la señal que justifique la adquisición de una nueva muestra. Más tarde, en las décadas de los 60 y los 70, se popularizó un método heurístico llamado *muestreo adaptativo*. El objetivo que persigue es reducir el número de muestras, sin que se produzca una degradación en la respuesta del sistema, evaluando en cada intervalo el periodo

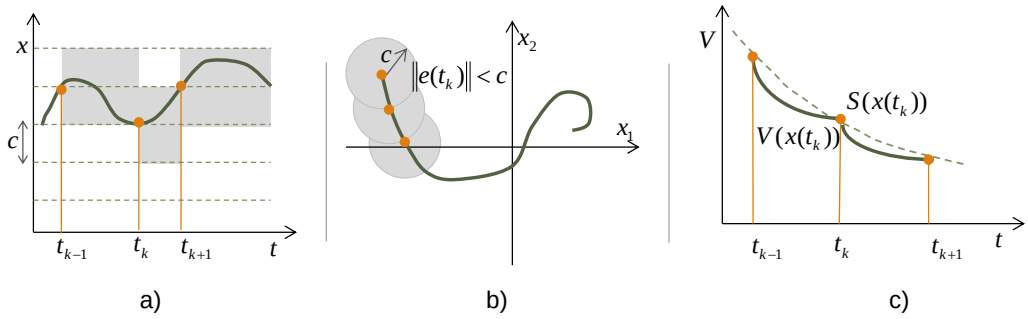

Figura D.8: Diferentes reglas de disparo por eventos.

de muestreo.

Más recientemente, en [Arz99] se implementa el control basado en eventos (CBE) en controladores PID mostrando que el número de acciones de control puede reducirse sin afectar al rendimiento del sistema. En [HGvZ+99] se utiliza el control por cruce de nivel para controlar el giro de un motor con sensores de baja resolución.

Los primeros resultados analíticos se obtuvieron para sistemas estocásticos de primer orden en [rB03], mostrando que bajo ciertas condiciones el CBE tiene un mejor rendimiento que el control periódico. Pero el gran impulso que ha recibido el control basado en eventos en estos últimos años ha venido motivado por su aplicación a sistemas de control en red.

A continuación se presenta una revisión de los trabajos más relevantes de la aplicación del CBE a SCR.

D.3.2 Control basado en eventos y SCR

La última década ha sido realmente prolífica en el campo de CBE, y la falta de resultados analíticos ha sido superada. También los resultados experimentales han mostrado que el uso del ancho de banda resulta más eficiente que en el caso del control periódico.

Una de las implementaciones más extendidas es considerar que se produce un evento cuando una función de error cruza un cierto límite. El cómo se define el error y este límite es lo que distingue las diferentes estrategias que existen en la literatura.

Si el error se define como la diferencia entre el estado en el instante de la última muestra y el estado actual de la planta y el límite es constante

$$\|e(t)\| = \|x(t) - x(t_k)\| \leq c,$$

la terminología más frecuente es la de control *deadband* o de zona muerta. t_k hace referencia al instante de tiempo del último evento y t es el tiempo presente. El valor de la constante c determina el rendimiento del sistema y la región en torno al equilibrio en la que el estado permanece confinado. Las figuras D.8a y D.8b representan dos ejemplos de control *deadband* para el caso unidimensional y bidimensional, respectivamente. Trabajos relevantes en esta categoría son [HSvdB08, San06].

Sin embargo, este tipo de control, en general, no asegura la estabilidad asintótica del sistema. Por este motivo, se han investigado otro tipo de reglas de disparo que permitan alcanzar esta propiedad. Un ejemplo se presenta en [Tab07], en el que el error es acotado por el estado del sistema:

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

Esta estrategia permite que el sistema sea asintóticamente estable pero presenta el inconveniente de que el tiempo entre eventos disminuye a medida que el sistema se acerca al equilibrio. No obstante, en [Tab07] se demuestra que un valor mínimo para dicho intervalo está garantizado bajo ciertas condiciones. Este aspecto es importante en el control basado por eventos ya que el efecto Zeno, es decir, la ocurrencia de dos o más eventos de manera consecutiva en el mismo instante de tiempo, debe ser evitado. Para ello, se diseña el parámetro σ de acuerdo con ciertas propiedades de la función de Lyapunov.

Recientemente, se han propuesto otras reglas de disparo que dependen del tiempo con el objetivo de alcanzar el punto deseado de manera asintótica. En [SDJ13, GDJ+11], se definen reglas de disparo del tipo

$$\|e(t)\| \leq c_1 e^{-\alpha t},$$

para sistemas multi-agentes y sistemas lineales interconectados, respectivamente, que son capaces de preservar la propiedad mencionada, garantizando además que el efecto Zeno no se produce.

Otros autores definen la regla de disparo teniendo en cuenta la función de Lyapunov [MAT09]. En concreto, un evento es detectado cuando el valor de la función de Lyapunov del sistema en lazo cerrado alcanza una cierto nivel de referencia $S(x,t)$ (véase la Figura D.8c):

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

Las redes de sensores son un caso especial de SCR en los que el consumo de energía tiene un papel fundamental. Por tanto, el muestreo basado en eventos resulta conveniente para reducir el número de emisiones. Sin embargo, tal y como se analiza en [AT10b, MT08, Ara11], la mayor parte de la energía consumida por el sensor se debe a la monitorización de la señal y no tanto a la transmisión. Las reglas de disparo analizadas hasta ahora requieren de la medida continua del estado. Por este motivo, una nueva estrategia conocida como *self-triggering*, o auto-disparo, ha emergido en los últimos años.

Las políticas basadas en self-triggering calculan el tiempo t_{k+1} en el cual tendrá lugar la siguiente ejecución de las tareas relacionadas con el control en función de la medida del último estado x_k. De esta manera, el sensor no monitoriza el proceso hasta que es despertado en el instante temporal t_{k+1}, adquiere la medida, la transmite, y recalcula el tiempo de la próxima ejecución. El concepto de self-triggering fue sugerido por primera vez en [VMF03]. El control por auto-disparo puede interpretarse como una emulación basada en software del control basado en eventos. Ha sido estudiado para sistemas lineales [WL09, MAT10], y aplicado a redes de sensores y actuadores [MT08, TFJB10, AAJ$^+$11, CMV$^+$10]. Un problema inherente a este esquema es el efecto de, por ejemplo, la existencia de incertidumbre en el modelo o la aparición de perturbaciones externas desconocidas. Para hacer frente a ello, se suelen derivar resultados bastante conservadores del tiempo inter-muestreo de manera que la estabilidad del sistema no se vea com-

prometida [WL10]. Una solución híbrida entre self-triggering y control basado en eventos se propone en [Ara11]. No obstante, el rechazo de perturbaciones no puede ser completamente garantizado y el control es centralizado, lo que la hace difícilmente extensible a sistemas con un número grande de nodos.

El *control por mínima atención* (MAC, de *Minimum Attention Control*) maximiza el siguiente tiempo de ejecución, mientras que garantiza un cierto nivel en el rendimiento del sistema [AT10a, DTH12]. La política es similar a la de self-triggering en el sentido de que el objetivo es que las tareas de control sean ejecutadas lo mínimo posible pero sin utilizar técnicas de emulación. A pesar de que en [DTH12] se propone un diseño que permite aliviar el problema de carga computacional que presentaba [AT10a], MAC es mucho menos robusto frente a retardos y perturbaciones que el control basado en eventos. El mismo tipo de problemas presenta el *control con atención en cualquier momento* (AAC, de *Anytime Attention Control*) [GQ10, GFB11, Gup09]. El diseño que se propone en [AT10a] asume que después de cada ejecución de la tarea de control, la señal de control no puede ser recalculada durante un cierto intervalo de tiempo determinado por otro elemento del sistema (el planificador), calculando así la señal de control que maximiza el rendimiento del sistema en lazo cerrado.

Todos los trabajos anteriores se basan en que el estado del sistema es accesible, aunque en la práctica esto no es posible en muchos casos. Si las mismas reglas tratan de utilizarse para controladores por realimentación de la salida, el efecto Zeno puede aparecer, tal y como se comenta en [DH12].

Los controladores basados en eventos con realimentación de la salida se dividen en dos categorías. A la primera pertenecen [LL11a, LL11c]. La medida del estado es reemplazada en la función de disparo por la estimación del estado proporcionada por un observador [LL11a] o un filtro [LL11c]. La segunda línea de investigación considera una estructura diferente en el controlador. Un controlador dinámico basado en la salida se propone en [DH10]. Haciendo uso de mecanismos de disparo mixtos, se puede garantizar que el estado permanece confinado en torno al equilibrio y que el efecto Zeno es excluido. Un diseño basado en muestreo por cruce de nivel se presenta en [KB06], donde se propone emular un controlador

LTI.

Todas las estrategias descritas hasta ahora consideran que el actuador mantiene constante la señal de control entre eventos (ZOH, Zero Order Hold). A pesar de que bajo esta consideración de "no hacer nada" se simplifica el análisis, se ha demostrado que si existe un modelo preciso de la planta, un generador de la señal de control puede tratar de emular el comportamiento de un sistema convencional de control en lazo cerrado para obtener un mejor rendimiento que con ZOH [LL10].

La idea de sacar ventaja del conocimiento de un modelo de la planta en SCR se introduce por primera vez en [MA02, MA03a], aunque las ejecuciones son periódicas y no basadas en eventos. No obstante, este tipo de estrategias como la propuesta en [LL10] requieren sincronización de todos los elementos en el lazo de control, lo que la hace difícilmente extensible a sistemas de control remoto y a esquemas distribuidos.

Alguno de los trabajos citados y otros tantos consideran arquitecturas con varios lazos de control y control descentralizado como [MT11, MC11, Mol11, GA12, DH12] o control distribuido [WL08, WL11, GDJ$^+$11, SDJ13].

Finalmente, pocas publicaciones han tenido en cuenta explícitamente el efecto de otras restricciones en la comunicación como retardos y/o pérdidas de datos. Un trabajo relevante es [LL12], que es una extensión del trabajo previo [LL10]. En [GA11a] se presenta una implementación que compensa parcialmente los retardos, y en [WL11, GLS$^+$12] se analizan implementaciones distribuidas para redes reales.

D.4 Control basado en modelo en SCR

La mayoría de las estrategias de control presentes en la literatura consideran el uso de ZOH de manera que la señal de control es constante a tramos.

Considérese un esquema de un controlador ubicado junto al sensor como en la Figura D.5a. La última medida $x(t_k)/y(t_k)$ se mantiene constante y el controlador calcula la señal de control $u(t)$ constante a tramos. Supóngase ahora que se dispone de un modelo del proceso y que, en lugar de mantener la señal de control

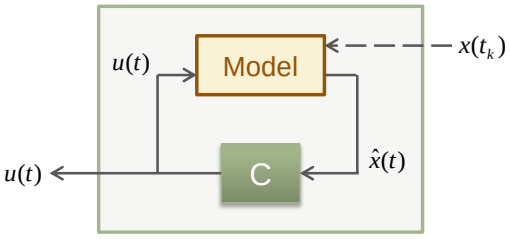

Figura D.9: Controlador basado en modelo.

constante, se estima el estado de la planta entre recepción y recepción. Para ello, el controlador es reemplazado por el elemento mostrado en la Figura D.9, de forma que se calcula la salida de control basándose en la predicción del estado $\hat{x}(t)$ dada por el modelo, y se inicializa cada vez que una nueva medida $x(t_k)$ es recibida.

Tal y como se comentó en la sección anterior, este concepto se introdujo en SCR por [MA02, MA03a]. Desde entonces, Antsaklis y colaboradores han publicado diferentes extensiones, como por ejemplo, sistemas de tiempo discreto [EA09] o con tiempos de transmisión variables [MA04].

Otras estrategias de control que recurren a un modelo del proceso ya han sido comentadas en la Sección D.1.2, en las que, normalmente, se itera un modelo de tiempo discreto para generar acciones de control futuras.

En [LL10, Leh11] se presenta una infraestructura en la que se usa un modelo tanto en el detector de eventos como en el controlador,con el objetivo de emular un controlador continuo por realimentación de estados, como ya se ha mencionado.

Finalmente, otros controladores basados en modelo han sido propuestos en [GA11a], donde se consideran retardos variables e incertidumbres en el modelo. También, se han presentado implementaciones descentralizadas [GA12] y distribuidas [GLS+12].

En general, se puede decir que los controladores basados en modelo permiten aumentar el periodo de muestreo (control periódico) o incrementar el tiempo entre eventos (control basado en eventos). En un supuesto ideal, si no hay incertidumbres en el modelo y el sistema no se ve afectado por perturbaciones, el modelo es capaz de estimar perfectamente el estado del sistema simplemente conociendo su estado inicial. Por tanto, el fin de combinar este tipo de control

con diseños basados en eventos es reducir el número de eventos con respecto al uso de ZOH.

D.5 Objetivos de la tesis

El objetivo principal de esta tesis es el de contribuir a resolver algunos de los problemas que surgen en los sistemas de control en red, con especial atención en el control basado en eventos. Las contribuciones siguen dos líneas: el diseño de estas estrategias de control y su posterior implementación. Este objetivo principal puede descomponerse en los siguientes puntos:

- *El diseño de una nueva arquitectura para sistemas de control en red*, incluyendo nuevos elementos en el lazo de control que sirvan de interfaces entre los componentes convencionales (controladores, sensores y actuadores) y la red. El objetivo de esta nueva arquitectura es aprovecharse de la flexibilidad que ofrecen las actuales redes de comunicación a la vez que se garantice la estabilidad del sistema y se haga frente a las imperfecciones de la comunicación. En concreto, se pretende disminuir la frecuencia de comunicación mediante técnicas basadas en eventos y hacer frente a los retardos de la red y a las pérdidas de datos a través del envío de predicciones de señales de tamaño finito.

- *La implementación de prototipos* de la solución propuesta de manera que los controladores convencionales puedan ser reutilizados sin la necesidad de gastar demasiado tiempo en su conversión a sistemas de control en red.

- *El diseño de estrategias distribuidas de control basado en eventos* para sistemas de control en red. El objetivo es diseñar políticas de transmisión y de actuación que disminuyan el número de comunicaciones pero que garanticen un cierto nivel de rendimiento, y la exclusión del comportamiento Zeno. El diseño propuesto debe hacer frente a posibles incertidumbres en el modelo de las interconexiones entre subsistemas que caracterizan a los sistemas de larga escala. También, se investigarán protocolos de comunicación para

mitigar los problemas inherentes a la red como los retardos y las pérdidas de paquetes, mientras que se preserva la existencia de una cota inferior para el tiempo entre eventos, ya que este es uno de los principales problemas de las estrategias que existen en la literatura.

- *Diseñar e implementar herramientas para la aplicación del control distribuido basado en eventos.* La motivación que persigue este objetivo es doble: La aplicación en un entorno educativo en el cual estamos inmersos, así como un medio para probar los algoritmos de control en un número elevado de situaciones antes de su implementación en plataformas reales. También se presentarán algunos datos experimentales de la implementación sobre un sistema real.

D.6 Guión y Contribuciones de la Tesis

La tesis se estructura de la siguiente manera:

- **Capítulo 2. Diseño de Control Anticipativo en Redes tipo Internet.** El Capítulo 2 presenta el análisis y el diseño de controladores remotos para SCR basados en paquetes, siguiendo los paradigmas del control anticipativo. El controlador remoto utiliza un modelo del proceso y un controlador base para generar futuras acciones de control con el objetivo de compensar el efecto de los retardos de la red y las pérdidas de paquetes. Se propone el diseño de dos capas de *middleware* entre el proceso y la red y el controlador y la red como medio para ocultar los elementos que no pertenecen a un sistema de control convencional. Se demuestra que la estabilidad del sistema es GUUB (*Globally Ultimately Uniformly Boundeded*) cuando se imponen ciertas condiciones al retardo de la red. Se proponen también diferentes extensiones como un estimador de perturbaciones, y se analiza el diseño de controladores LTI para sistemas en los que sólo se puede medir la salida, de tal modo que la propiedad de GUUB se preserve. Finalmente, se propone un controlador anticipativo centralizado para un sistema

con varios lazos de control. Este trabajo ha sido publicado en la revista *IET Control Theory and Applications* (véase [GSD11]) y presentado en la *49 IEEE Conference on Decision and Control* (véase [GSD10]).

- **Capítulo 3. Implementación y Evaluación Experimental del Control Anticipativo.** El marco experimental en el que el control anticipativo descrito en el Capítulo 2 ha sido evaluado se presenta en el Capítulo 3. También se encuentra en este capítulo la descripción de las plantas, la implementación de las capas de *middleware* en LabVIEW y los resultados experimentales que resaltan la bondad del diseño propuesto. Las publicaciones relacionadas son las mismas que para el Capítulo 2.

- **Capítulo 4. Control Distribuido Basado en Eventos para Sistemas Lineales Interconectados.** El Capítulo 4 presenta una estrategia de control distribuido basado en eventos para un sistema dinámico en red consistente en N sistemas interconectados LTI. Las reglas de disparo propuestas, que únicamente dependen de información local, garantizan la convergencia asintótica al equilibrio así como la existencia de una cota inferior para el tiempo entre eventos. El problema es resuelto inicialmente para sistemas perfectamente desacoplados, y estos resultados son generalizados para el caso en el que el desacoplo no sea perfecto. Los términos de acoplamiento son tratados como una perturbación del sistema nominal, y las herramientas clásicas para el análisis de la sensibilidad de la exponencial y potencias de matrices se aplican para obtener restricciones en los términos de acoplamiento de manera que la propiedad de estabilidad asintótica sea preservada. Este trabajo ha sido presentado en la *50 IEEE Conference on Decision and Control* [GDJ$^+$11] y también ha sido aceptado para su publicación [GDJ$^+$13].

- **Capítulo 5. Extensiones y Mejoras del Control Distribuido Basado en Eventos.** El Capítulo 5 se centra en dos aspectos. El primero de ellos es el estudio de los efectos que las comunicaciones reales pueden tener sobre el esquema de control distribuido presentado en el Capítulo 4. A pesar de

que el control basado en eventos se muestra efectivo a la hora de reducir la frecuencia de las comunicaciones en el lazo de control, no puede evitar la existencia de retardos y pérdidas. Por tanto, se analizan las consecuencias que tienen las redes no ideales en el cierre del lazo, y se calculan cotas superiores para el retardo y el número consecutivo de pérdidas de paquetes para varios escenarios posibles que garantizan la estabilidad el sistema y un cierto nivel de rendimiento. Posteriormente, se proponen dos mejoras. La primera está basada en el hecho de que si el número de nodos vecinos de un subsistema es extenso, la frecuencia de actuación puede ser elevada en esquemas de control distribuido, incluso si cada agente no transmite de manera muy frecuente. Para ello, se define una nueva función de error para la entrada de control y se propone un segundo conjunto de funciones de disparo para solventar el citado problema, actualizando la ley de control sólo cuando se viola una cierta condición. La segunda mejora se basa en la existencia de actuadores inteligentes capaces de aplicar señales continuas en lugar de constantes a tramos. Un controlador basado en modelo es propuesto de manera que cada agente tiene cierto conocimiento de la dinámica de sus vecinos en la red. En base a ese modelo, estima su estado y calcula la señal de control. Se asume una posible incertidumbre en el modelo, y el rendimiento del diseño propuesto en el Capítulo 4 y el basado en modelo es comparado en función de la incertidumbre existente. Partes de este trabajo han sido presentadas en el CDC de 2011 y de 2012 (véase [GDJ+11, GLS+12]), y el controlador basado en modelo está también incluido en el artículo de revista citado en el Capítulo 4.

- **Capítulo 6. Herramientas de Simulación y Ejemplo de Aplicación del CDBE.** El control de formaciones de robots móviles a través de redes es un ejemplo de sistemas multi-agentes en los que un grupo de robots alcanzan un objetivo común, en este caso la formación, gracias a leyes de control distribuidas basadas en eventos. Se ha desarrollado un simulador interactivo para reproducir este tipo de sistemas. En concreto, el control de formaciones desde el punto de vista del consenso para un gran número de posibles

condiciones de la red y múltiples experimentos puede ser estudiado a través de esta plataforma. La interactividad de la herramienta ha sido uno de los principales objetivos del desarrollo. Además, se pueden cambiar múltiples parámetros mientras la simulación está ejecutándose, simplemente pulsando y arrastrando elementos. Los algoritmos de control distribuido basado en eventos también han sido implementados en una plataforma real y se muestran los resultados experimentales. Este trabajo ha sido publicado en la revista *IEEE Network Magazine* [GFF$^+$12] (herramienta de simulación) y la aplicación experimental se ha enviado a *Sensors*.

- **Capítulo 7. Conclusiones y trabajos futuros**. Se enumeran las conclusiones y trabajos futuros.

D.7 Publicaciones y Proyectos

Artículos de Revista

1. M. Guinaldo, J. Sánchez, S. Dormido. A co-design strategy of NCS for treacherous network conditions. *IET Control Theory & Applications*, 5(16): 1906-1915, 2011.

2. M. Guinaldo, G. Farias, E. Fabregas, J. Sánchez, S. Dormido-Canto, S. Dormido. An Interactive Simulator for Networked Mobile Robots. *IEEE Network Magazine*, 26(3): 14-20, 2012.

3. M. Guinaldo, D.V. Dimarogonas, K.H. Johansson, J. Sánchez, S. Dormido. Distributed Event-Based Control Strategies for Interconnected Linear Systems. *IET Control Theory & Applications*, 2013, *Aceptado el 1 de febrero de 2013*, doi: 10.1049/iet-cta.2012.0525.

4. M. Guinaldo, E. Fabregas, G. Farias, S. Dormido-Canto, D. Chaos, J. Sánchez, S. Dormido. Mobile robots experimental environment with event-based wireless communications. *Enviada a Sensors (estado: major revision)*.

5. E. Fabregas, G. Farias, S. Dormido-Canto, M. Guinaldo, J. Sánchez, S. Dormido. Virtual and real laboratory for teaching mobile robotic. *Enviada a IEEE Transactions on Industrial Electronics.*

Artículos en Congresos y Conferencias

1. M. Guinaldo, J. Sánchez, S. Dormido. Diseño de un Sistema de Control Anticipativo Basado en Paquetes para Control en Red. *9ª Conferencia Iberoamericana en Sistemas, Cibernética e Informática* (CISCI 2010), julio 2010, Orlando.

2. M. Guinaldo, J. Sánchez, S. Dormido. A Packet-based Network Control System Architecture for Teleoperation and Remote Laboratories. *49th IEEE Conference on Decision and Control* (CDC), diciembre 2010, Atlanta.

3. M. Guinaldo, D.V. Dimarogonas, K.H. Johansson, J. Sánchez, S. Dormido. Distributed Event-Based Control for Interconnected Linear Systems. *50th IEEE Conference on Decision and Control and European Control Conference* (CDC-ECC), diciembre 2011, Orlando.

4. M. Guinaldo, J. Sánchez, S. Dormido, M.A. Delgado. Control en red basado en eventos de múltiples plantas remotas. *XXXIII Jornadas de Automática*, septiembre 2012, Vigo.

5. M. Guinaldo, D. Lehmann, J. Sánchez, S. Dormido, K.H. Johansson. Distributed Event-Triggered Control with Network Delays and Packet-losses *51th IEEE Conference on Decision and Control* (CDC), diciembre 2012, Maui.

6. M. Guinaldo, J. Sánchez, S. Dormido. Contribuciones al control en red basado en eventos para sistemas lineales. *XI Simposio CEA de Ingeniería de Control*, abril 2013, Valencia.

7. M. Guinaldo, D. Lehmann, J. Sánchez, S. Dormido, K.H. Johansson. Reducing communication and actuation in distributed control systems. *En-*

viado a 51th IEEE Conference on Decision and Control, 2013.

Otras publicaciones

1. M. Guinaldo, B. Pérez-Lancho, E. Sanz. Laboratorio virtual para el aprendizaje del control térmico en edificios. *V Jornadas de Enseñanza a Través de Internet/Web de la Ingeniería de Sistemas y Automática*, septiembre 2007, Zaragoza.

2. M. Guinaldo, E. Sanz, S. Dormido. Laboratorio Virtual Basado en Web para Aprendizaje de Física usando Ejs. *XXIX Jornadas de Automática*, septiembre 2008, Tarragona.

3. M. Guinaldo, J. Sánchez, H. Vargas, S. Dormido. Laboratorio basado en Web del sistema bola y viga para el entrenamiento de estrategias de control avanzado. *XXX Jornadas de Automática*, septiembre 2009, Valladolid.

4. M. Guinaldo, H.Vargas, J. Sánchez, S. Dormido. Web-Based Control Laboratory: The Ball and Beam System. *8th IFAC Symposium on Advances in Control Education* (ACE09), octubre 2009, Kumamoto.

5. M. Guinaldo, J. Sánchez, H. Vargas, S. Dormido. An Advanced Web-based Control Laboratory for the Ball and Beam System. *9th Portuguese Conference on Automatic Control* (CONTROLO'2010), septiembre 2010, Coimbra.

6. M. Guinaldo, L. de la Torre, R. Heradio, S. Dormido. A Virtual and Remote Control Laboratory in Moodle: The Ball and Beam System. *Enviado a 10th IFAC Symposium Advances in Control Education*, 2013.

Proyectos de Investigación

Los resultados obtenidos en el marco de esta Tesis han sido financiados por diferentes proyectos de investigación:

- *Modelado, simulación y control basado en eventos* (2007-2012). Ministerio de España de Ciencia y Tecnología, CICYT (Ref. DPI2007-61068). Parti-

cipantes: UNED (España), Universidad de Murcia (España). Dirigido por el Prof. Sebastián Dormido Bencomo.

- *MACROBIO: Modelling, simulation, control and optimization of photobioreactors* (2012-2014). Ministerio de España de Economía y Competitivdad, CICYT (Ref. DPI2011-27818-C02-2). Participantes: UNED (Spain). Dirigido por el Prof. José Sánchez Moreno.

- *Control basado en eventos de sistemas distribuidos y colaborativos* (2012-2014). Ministerio de España de Economía y Competitivdad, CICYT (Ref. DPI2012-31303). Participantes: UNED (España). Dirigido por el Prof. Sebastián Dormido Bencomo.

D.8 Conclusiones y líneas futuras de investigación

D.8.1 Conclusiones

En esta tesis se han abordado varios problemas de los SCR, con especial atención en la reducción de la comunicación entre los distintos elementos del lazo de control, pero también retardos y pérdidas de paquetes. Se ha demostrado que las políticas de transmisión basadas en eventos son efectivas para hacer frente a estos problemas en los escenarios estudiados, y se han implementado varias aplicaciones que respaldan los resultados teóricos.

En concreto, en el Capítulo 2 se ha afrontado el problema del diseño de controladores remotos. La solución propuesta radica en el diseño de dos capas de middleware que actúan de interfaces entre los componentes convencionales de un lazo de control y la red, siendo el modelo de la planta el elemento más importante en el lado del controlador. La iteración de dicho modelo con un controlador base permite generar predicciones de señales de tamaño finito para hacer frente a retardos y pérdidas de paquetes de manera efectiva. Además, las políticas de envío por eventos permiten reducir la frecuencia de comunicación y obtener una respuesta robusta a posibles perturbaciones.

La falta de sincronización entre el controlador y el resto de elementos impide que los retardos en la red puedan medirse de manera absoluta. Se ha superado esta limitación midiendo el RTT desde el controlador. Además, se ha diseñado un estimador de perturbaciones que proporciona una mejor predicción de los estados futuros del proceso. Si no se puede medir el vector de estado completo, se ha propuesto una solución mixta que utiliza observadores y controladores LTI, particularizando el diseño para controladores PI. Se ha estudiado la estabilidad del sistema en las tres situaciones mencionadas a través funciones de Lyapunov.

La implementación de las dos capas de middleware en LabVIEW se ha descrito en el Capítulo 3. Estas aplicaciones permiten reutilizar con poco esfuerzo controladores convencionales en control a través de redes. Se han diseñado varios experimentos que permiten estudiar el sistema bajo condiciones de la red relativamente extremas para el sistema. La bondad de la solución propuesta se ilustra en los resultados experimentales.

La segunda parte de la tesis se ha centrado en control distribuido a través de redes. En concreto, se ha propuesto el uso de control disparado por eventos para reducir tanto la frecuencia de comunicación como la de actuación en sistemas lineales interconectados. El capítulo 4 aborda este problema tanto para sistemas perfectamente desacoplados como para el caso en el que esta condición no es asumible, mostrando que, si se usan funciones de disparo que dependen del tiempo, se puede alcanzar la convergencia asintótica del sistema al equilibrio, mientras que se garantiza una cota inferior para el tiempo entre eventos.

Varias extensiones a este esquema se proponen en el Capítulo 5. Especificamente, se ha abordado el problema de existencia de retardos y pérdida de datos en control distribuido basado en eventos. Se han diseñado dos protocolos de red, uno de los cuales hace frente al problema de consistencia del estado permitiendo cotas máximas para el retardo y el número de pérdidas de datos consecutivas mayores que el otro protocolo. Ambos protocolos permiten también preservar las propiedades de estabilidad asintótica, así como la existencia de una cota inferior para el tiempo entre eventos.

En el Capítulo 5 también se han descrito dos posibles mejoras que permiten

un uso más eficiente de los recursos de sistemas embebidos. Si la frecuencia de actuación no debe exceder un cierto valor, se puede diseñar un segundo conjunto de funciones de disparo para controlar la actualización de la ley de control en sistemas de control distribuidos. Por otro lado, si el recurso más crítico es la frecuencia de transmisión, un diseño de control distribuido basado en modelo permite alargar el tiempo entre eventos.

Finalmente, en el Capítulo 6 se ha presentado el desarrollo de un simulador que permite probar el control distribuido disparado por eventos en un grupo de robots móbiles en red. La herramienta presenta un alto grado de interactividad y, por tanto, resulta adecuada para un entorno educativo. También permite probar el modelo del sistema bajo un amplio espectro de posibles situaciones. Por último, se han presentado resultados experimentales sobre una plataforma real de robots móbiles.

D.8.2 Trabajos futuros

Este trabajo puede extenderse en varias direcciones. A continuación se enumeran algunas ideas.

- En los capítulos 2 y 5, se han propuesto controladores basados en modelo tanto centralizados como descentralizados para disminuir la frecuencia en la ocurrencia de eventos. Si el modelo tiene incertidumbre, ésta afecta negativamente tanto al comportamiento del sistema como al número de eventos generados. Pensamos que posibles soluciones a este problema pasan por estudiar el control adaptativo [rW95] o técnicas de estimación de parámetros en línea [Lju99]. Nuestra intuición es que la solución no es trivial, ya que los sistemas basados en eventos pueden interpretarse como sistemas con parámetros variables en el tiempo, y tampoco hay un conocimiento previo de cuándo sucederá el próximo evento. Además, el problema es aún más complicado para el caso distribuido, ya que, a priori, el efecto de los términos de acoplamiento no puede distinguirse del producido por la incertidumbre del modelo. Hay un trabajo reciente [GA11b] que afronta el problema de la

estabilización adaptativa de controladores centralizados basados en modelo para sistemas de tiempo discreto usando distintas variaciones del filtro de Kalman. Sin embargo, el perfil del error que introduce el control basado en eventos no encaja con el requisito de ruido blanco de media nula que require Kalman.

- A pesar de que el esquema de control distribuido en eventos presentado en esta tesis facilita la escalabilidad del sistema, ya que las ganancias de realimentación y de desacoplamiento son diseñadas de manera local, los parámetros de las funciones de disparo están restringidos por información global del sistema como $\lambda_{max}(A_K)$, $\kappa(V)$ o $\|\Delta\|$. La utilización de algoritmos distribuidos [YFG+08, YFL08] para estimar información del sistema a instancias de tiempo dadas por eventos es otra de las futuras líneas de investigación.

- Hay muchas campos en los que el control disparado por eventos puede ser útil como en redes eléctricas, conjunto de satélites, redes de tráfico, canales de irrigación o fotobiorreactores. En este tipo de sistemas, resulta conveniente un diseño energético eficiente para control distribuido, que es uno de los objetivos perseguidos por el control basado en eventos. En este sentido, ya está en marcha un proyecto de investigación para fotobiorreactores [SÍ13], así como colaboraciones recientes en redes de energía y de tráfico.

- Las capas de middleware se han implementado en LabVIEW. Sin embargo, sería más adecuado que la aplicación del lado del cliente hubiese sido desarrollada con software libre de licencia. Por tanto, Ejs parece adecuado para este objetivo. Además, esta línea encajaría en el esquema de laboratorios remotos propuesto en [Var10], que serían transformados en *laboratorios controlados remotamente*. Más aún, la modularidad de las capas de middleware, y en particular de CAL, hace que su implementación encaje perfectamente como un *elemento* de Ejs. Un elemento de Ejs es muy similar a una librería Java [FGElT+12] pero proporciona una *interfaz de programación de la aplicación* (API, de *Application Programming Interface*) para

su personalización. Con respecto al simulador MaSS, los aspectos relativos a la red pueden también ser migradosa un elemento de Ejs con el objetivo de aumentar su portabilidad.