TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Steuerungs- und Regelungstechnik

An Arbitrated Networked Control Systems Approach to Cyber-Physical Systems

Harald Voit

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. rer. nat. Thomas Hamacher
Prüfer der Dissertation:	
	1. Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
	2. Prof. Anuradha Annaswamy, Ph.D. Massachusetts Institute of Technology, USA

Die Dissertation wurde am 25. Juni 2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07. Oktober 2013 angenommen.

Acknowledgment

This thesis summarizes the results of my research conducted in the past four years at the Institute of Automatic Control Engineering, LSR, at Technische Universität München, Germany. I will always keep the time at LSR in fond memory. Thus, I would like to thank some people who supported and promoted me or with whom I had some nice chats.

First of all, I would like to thank Dr. Anuradha Annaswamy of Massachusetts Institute of Technology, Cambridge, MA, USA, who gave me the opportunity to work in a very exciting and innovative area. She always demanded one hundred percent commitment and motivation, but was prepared to listen to research-related questions at any time and continuously provided valuable hints and ideas. Furthermore, I would like to thank her for the possibility to spend some months at the Active Adaptive Control Lab at MIT and to gain a lot of priceless experiences.

I would also like to thank Prof. Martin Buss and Prof. Sandra Hirche for providing an inspiring working environment and a wonderful team at LSR.

Furthermore, I would like to thank Prof. Samarjit Chakraborty for introducing me into his field of research and supporting me with a lot of dedication, knowledge, and positive criticism.

This thesis was made possible by the financial support of the Institute of Advanced Study (IAS), funded by the German Excellence Initiative and by Deutsche Forschungsgemeinschaft (DFG) through the International Graduate School of Science and Engineering (IGSSE) of TU München. I would like to thank them for the numerous international conferences that I could attend and the long-term stay abroad at MIT.

Special thanks goes to Arman Kiani who shared all ups and downs of my work and with whom I conducted several 2pm-meetings. Another special thanks goes to my longtime office mate and friend Benjamin Passenberg for all the chats about research, soccer, and so on. Also, I would like to thank all other colleagues at LSR for the inspiring and pleasant atmosphere. In this context, I would like to highlight my office neighbor Mrs. Werner, who provided me with appreciated distractions during my working day.

I would also like to thank Mr. Heinrich Dorsch, IABG, who released me for four years from my work and thus helped me a lot in reaching my personal goal.

Finally, I would like to thank my parents and my brother Albert with all my heart for their steady support in all this years of studying and learning.

My biggest thanks goes to my wife Christiane, who also supported me at any time and was always understanding when I had to work on the weekend. Without her, I would not be where I am now. Last but not least, I would like thank my son Paul, even if it will take some years until he can read and understand this thesis.

Munich, June 2013 Harald Voit

Danksagung

Diese Arbeit fasst die Ergebnisse meiner Forschung der vergangenen vier Jahre am Lehrstuhl für Steuerungs- und Regelungstechnik (LSR) an der Technischen Universität München zusammen. Ich werde die Zeit am LSR immer in guter Erinnerung behalten. Daher möchte ich einigen Menschen besonders danken, die mich in dieser Zeit unterstützt, gefördert oder gefordert haben, oder mit denen ich einfach nur viele nette Gespräche geführt habe.

An erster Stelle möchte ich Dr. Anuradha Annaswamy vom Massachusetts Institute of Technology, Cambridge, MA, USA, danken, die mir die Gelegenheit gegeben hat, auf einem äußerst spannenden und innovativen Gebiet zu forschen. Sie verlangte stets hundertprozentigen Einsatz und Leistungsbereitschaft, hatte aber auch jederzeit ein offenes Ohr für die vielen Fragen, die während der Forschungsarbeit aufkamen sowie immer einen guten Ratschlag oder eine Idee parat. Ich danke ihr auch für die Möglichkeit, einige Monate in ihrem Labor am MIT zu verbringen und viele unschätzbare Erfahrungen zu sammeln.

Ebenso danke ich Prof. Martin Buss und Prof. Sandra Hirche, dass ich am LSR forschen und in diesem wunderbaren Team arbeiten durfte.

Des Weiteren möchte ich Prof. Samarjit Chakraborty danken, dass er mich in sein Arbeitsgebiet eingeführt und stets mit viel Engagement, Wissen und positiver Kritik begleitet und unterstützt hat.

Meine Arbeit wurde erst durch die finanzielle Unterstützung des Institute of Advanced Study (IAS), gefördert durch die Exzellenzinitiative sowie durch die Deutsche Forschungsgemeinschaft (DFG) über die International Graduate School of Science and Engineering (IGSSE) der TU München, ermöglicht. Für die zahlreichen internationalen Konferenzen, an denen ich teilnehmen und den mehrmonatigen Auslandsaufenthalt, den ich im Rahmen dieser Arbeit absolvieren konnte, möchte ich mich ganz herzlich bedanken.

Ein besonderer Dank geht auch an Arman Kiani, der mich durch viele Höhen und Tiefen meiner Arbeit begleitet und zahlreiche 2-Uhr-Telefonkonferenzen mit mir durchgemacht hat. Ein weiterer besonderer Dank geht an meinen langjährigen Zimmerkollegen und Freund Benjamin Passenberg für die vielen Gespräche über Forschung, Fußball und so weiter. Ebenso möchte ich allen anderen Kollegen am LSR für die inspirierende und freundliche Atmosphäre danken. In diesem Zusammenhang möchte ich meine Büronachbarin Frau Werner hervorheben, die immer für willkommene Abwechslung in meinem Arbeitsalltag gesorgt hat.

Darüber hinaus möchte ich ebenfalls Herrn Heinrich Dorsch, IABG, danken, dass er mich vier Jahre lang freigestellt und mir so beim Erreichen meines persönlichen Zieles sehr geholfen hat.

Zu guter Letzt möchte ich meinen Eltern und meinem Bruder Albert von ganzem Herzen danken, dass sie mich all die Jahre des Studierens und Lernens unterstützt haben und stets hinter mir standen.

Mein allergrößter Dank geht an meine Frau Christiane, die mich beständig unterstützt hat und immer Verständnis für lange Arbeitswochenenden hatte. Ohne sie wäre ich nicht da, wo ich jetzt bin. Nicht zuletzt möchte ich meinem Sohn Paul danken, auch wenn es noch einige Jahre dauern wird, bis er diese Arbeit lesen und verstehen kann.

München, Juni 2013 Harald Voit

Abstract

The term Cyber-Physical Systems (CPS) refers to a relatively new idea and describes the need for integrated research in control and communication. Although CPS are ubiquitous in our everyday life, their design and development is difficult, expensive, and without a sound theoretical foundation. Traditionally, the design of the controller and its stability and performance analysis is done independently from the design and analysis of the communication and computation architecture which in turn aggravates the verification of the resulting CPS. Since the generic term CPS describes many different systems with different features, the focus in this thesis is on a special class of CPS where the arbitration of messages on a shared communication medium is a dominant feature of the CPS. For this class of systems the term *Arbitrated Networked Control Systems (ANCS)* is coined. By emphasizing the arbitration of messages, i.e., the communication network, additional information about the network can be used for the design of the controller.

In this thesis several approaches for the integrated co-design of control and communication of ANCS are presented. In particular, ANCS with hierarchical and hybrid communication networks are studied. It is shown that an ANCS provides additional information about the network and how this information can be used to improve the control performance of an NCS with a hierarchical schedule. To this end a novel performance metric is developed which can be computed analytically and a proof is provided that the metric is monotonically increasing with decreasing communication delay. This is necessary since the delay is computed with the Real-Time Calculus (RTC) framework that can determine the worst-case end-to-end delay of messages in arbitrary communication networks. Finally, the novel performance metric augmented with other standard performance metrics and the RTC framework are used to compute the optimal controller and scheduler parameters through the solution of nonlinear optimization problem. Since any system contains uncertainties, which might be due to changes in the environment or unforeseen disturbances, adaptive controllers are introduced. It is shown, how adaptive controllers can utilize the additional information provided by the ANCS to improve the control performance compared to non-adaptive controllers.

Hybrid protocols belong to the most advanced scheduling policies used in embedded systems. They combine the advantages of time-triggered and event-triggered schemes in one protocol. In this thesis, an ANCS is presented that uses a hybrid, co-designed control and communication protocol for the control of multiple applications. The co-design consists of three parts: (i) a hybrid communication protocol with a time-triggered and an event-triggered segment, (ii) a switching scheme that decides whether a control-related message is sent over the time-triggered or the event-triggered segment, and (iii) an adaptive controller in order to cope with any uncertainties that may be present. The resulting adaptive switching controller is then shown to be stable with a proof of global boundedness and asymptotic tracking of a desired signal in the presence of a class of disturbances. It is shown through extensive simulation studies that the proposed co-design achieves better performance than other standard controller implementations.

Zusammenfassung

Der Begriff Cyber-Physical Systems (CPS) bezieht sich auf eine relativ neue Idee und beschreibt die Notwendigkeit einer integrierten Forschung in der Regelungs- und Informationstechnik. Obwohl CPS in unserem täglichen Leben allgegenwärtig sind, ist ihre Konstruktion und Entwicklung schwierig, teuer und ohne eine solide theoretische Grundlage. Traditionell wird die Auslegung des Reglers und die dazugehörige Stabilitäts- und Performanzanalyse unabhängig von der Konstruktion und Analyse der Kommunikations- und Prozessorarchitektur ausgeführt, dies wiederum erschwert die Verifizierung der resultierenden CPS. Da der Begriff CPS viele verschiedene Systeme mit unterschiedlichsten Funktionen umfasst, liegt der Fokus in dieser Arbeit auf einer speziellen Klasse von CPS, in denen die Arbitrierung der Nachrichten auf einem gemeinsam genutzten Kommunikationsmedium ein dominantes Merkmal der CPS darstellt. Für diese Klasse von Systemen wird der Begriff *Arbitrated Networked Control Systems (ANCS)* geprägt. Durch die Betonung der Arbitrierung von Nachrichten, d.h. des Kommunikationsnetzes, können zusätzliche Informationen über das Netzwerk selbst für das Design des Reglers verwendet werden.

In dieser Arbeit werden verschiedene Ansätze zum integrierten Co-Design von Reglern und Kommunikationsarchitekturen für ANCS vorgestellt. Insbesondere werden ANCS mit hierarchischen und hybriden Kommunikationsnetzen untersucht. Es wird gezeigt, dass ein ANCS zusätzliche Informationen über das Netzwerk zur Verfügung stellt und wie sich diese nutzen lassen, um das Regelverhalten eines NCS mit einem hierarchischen Scheduler zu verbessern. Zu diesem Zweck wird eine neue Performance-Metrik entwickelt, die analytisch berechnet werden kann, und es wird der Beweis erbracht, dass diese Metrik monoton mit abnehmender Zeitverzögerung wächst. Dies ist notwendig, da die Zeitverzögerung mittels Real-Time Calculus (RTC) berechnet wird, welches die Worst-Case Ende-zu-Ende-Verzögerung von Nachrichten in beliebigen Kommunikationsnetzen bestimmt. Schließlich wird die neue Performance-Metrik – erweitert um weitere Standard-Performance-Metriken – und RTC verwendet, um die optimalen Regler- und Scheduler-Parameter durch die Lösung eines nichtlinearen Optimierungsproblems zu berechnen. Da jedes System Unsicherheiten enthält, die durch Veränderungen in der Umwelt oder unvorhergesehene Störungen hervorgerufen werden können, werden adaptive Regler eingeführt. Es wird gezeigt, wie adaptive Regler die zusätzlichen Informationen im ANCS verwenden können, um die Regelungsperformanz im Vergleich zu nicht-adaptiven Steuerungen zu verbessern.

Hybride Protokolle gehören zu den fortschrittlichsten Schedulingmethoden in eingebetteten Systemen. Sie kombinieren die Vorteile der zeitgesteuerten und ereignisgesteuerten Systeme in einem Protokoll. In dieser Arbeit wird ein ANCS vorgestellt, das ein hybrides, gemeinsam entworfenes Regelungs- und Kommunikationsprotokoll für die Regelung von mehreren Anwendungen verwendet. Das Co-Design besteht aus drei Teilen: (i) ein hybrides Kommunikationsprotokoll mit einem zeitgesteuerten und einem ereignisgesteuerten Segment, (ii) ein Schaltalgorithmus, der entscheidet ob ein Steuersignal über das zeitgesteuerte oder das ereignisgesteuerte Segment gesendet wird

und (iii) einer adaptiven Regelung, um alle eventuellen Unsicherheiten bewältigen zu können. Die Stabilität des resultierenden adaptiven, schaltenden Reglers und dessen Fähigkeit einem Referenzsignal in Gegenwart einer Klasse von Störungen zu folgen, wird mittels eines ausgefeilten Beweises gezeigt. Durch aussagekräftige Simulationen wird gezeigt, dass das vorgestellte Co-Design eine höhere Performanz als andere Standardregler erreicht.

Contents

List of Figures

List of Tables

1 Introduction

Summary: *In this chapter, the motivation to study Cyber-Physical Systems is described. Next, the basic ideas of feedback control systems, adaptive control and networked control systems are described as these three provide the baseline for the later chapters and discussions. Also, an overview of related work is given. This chapter is concluded by an outline of the thesis and a summary of its main contributions.*

1.1 Motivation

Cyber-Physical Systems (CPS) are ubiquitous in our everyday life. They can be found in areas as diverse as aerospace, automotive, chemical processes, energy, manufacturing, transportation, civil infrastructure, entertainment, and consumer appliances. This variety of areas shows that CPS are a key technology for the future. The term CPS is a general term for computer systems that are embedded into the physical world and are linked together through communication networks. One reason for their pervasiveness is that the capabilities of CPS are increasing steadily while the costs are declining at an even higher rate. A difference between traditional computers and CPS is the way of interaction with human operators. A traditional computer accepts inputs through a mouse, keyboard, or joystick and provides outputs on a screen. A CPS, however, interacts directly and most of the time independently of human operators with its physical environment by sensing the environment, computing a required action, and applying the action. All this has to be done with limited resources like limited communication bandwidth, limited power, limited computational power, and so on. However, CPS can achieve this task since they are designed for one specific problem, in contrast to traditional computers that are designed for general purpose. Since CPS are embedded into physical devices, their lifetime is long compared to other computer systems. This makes it necessary to verify at the design stage of CPS that they operate reliable and safely over their entire lifetime.

CPS can be described by the following properties. For all of these properties individual solutions in different research areas are known.

CPS are tightly coupled to the physical world: As mentioned above, CPS interact directly with their environment by means of sensors, controllers, and actuators. All this is part of the research in the area of automatic control.

CPS are resource-constrained: The way CPS are used puts some constraints on their design. Many CPS will be battery operated to allow simple deployment. Limited

power also means limited computational power. Besides from this constraints, the allowable thermal dissipation might be a limiting factor in the design of CPS. Problems like constraint computation are discussed in the area of embedded systems.

CPS are long-living: Since CPS are embedded into physical systems, it is natural for them to have a long lifetime. A consequence of the long lifetime is that the environment of the CPS might change dramatically over time. For example, the physical system might fail or external components like senors are exchanged. In order to ensure that the CPS continues working properly, methods like adaptive controllers are studied.

CPS are made of many interacting heterogeneous components: A key feature of CPS is the exchange of information over communication networks that might have thousands of nodes. The nodes attached to the network are in general very heterogeneous. For example, a sensor sends its measurements in a periodic manner, but the controller only sends control values when necessary. The design of such distributed heterogeneous control systems is part of the research in networked control systems.

The above properties demonstrate that CPS are made up of several software and hardware technologies. For each of the above mentioned properties solutions exist, but the challenge is to combine these solutions into one integrated approach with a sound theoretical foundation. This is also often referred to as *co-design of control and communication.*

In this thesis, several co-design approaches will be presented. Based on the three main areas of research automatic and adaptive control, networked control systems, and embedded control systems as the key technologies for CPS, co-design approaches for a special class of CPS will be discussed. Since the generic term CPS describes many different systems with different features, the focus in this thesis is on a special class of CPS where the arbitration of messages on a shared communication medium is a dominant feature of the CPS. For this class of systems the term *Arbitrated Networked Control Systems (ANCS)* is coined. By emphasizing the arbitration of messages, i.e., the communication network, additional information about the network can be used for the design of the controller. In particular, ANCS with communication networks with hierarchical and hybrid architectures are studied. It is shown how the additional information about the network, provided by the ANCS, can be used to design more efficient controllers. For hierarchical schedules this is done by defining a novel performance metric and combining this metric with the Real-Time Calculus framework into one optimization problem. The solution of this optimization problem is the optimal choice of controller and scheduler parameters. In order to cope with uncertainties, an adaptive controller is also integrated into the co-design. For hybrid protocols, the structure of the protocol is utilized to improve the control performance while reducing

the cost of the communication bus. It is shown by simulations that the proposed co-design is superior compared to other common controller implementations.

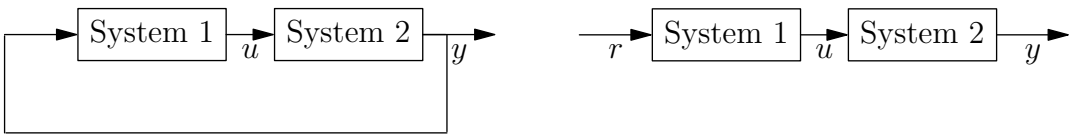
Next, a brief introduction to each of the three main areas will be given, followed by an overview of related work, an overview of the main contributions, and a detailed outline of the thesis. The rest of the thesis is organized as follows: In Chapter 2, the theoretical basics for the thesis are discussed. In Chapter 3, the ANCS approach to Cyber-Physical Systems will be presented. In Chapter 4, a co-design for hierarchical schedules is presented and in Chapter 5 for hybrid schedules. A simulation study for hybrid schedules is presented in Chapter 6 and the thesis is concluded with Chapter 7.

1.2 Control, Networked Control Systems, and Embedded Control Systems

1.2.1 Feedback Control and Adaptive Control

Feedback Control

The terms *feedback* and *control* are closely related to each other [1]. The term feedback describes a situation in which the output of one dynamical system is the input of another dynamical system and vice versa. Through this interconnection the dynamics of both systems are strongly coupled. The term control refers to the output a dynamical system generates in order to lead a second dynamical system to a desired behavior. In general one can distinguish between *open-loop* and *closed-loop* as shown in Figure 1.1. In a closed-loop system the systems are connected by a feedback loop, whereas in an open-loop system this feedback is not given.



(a) Closed-loop **(b)** Open-loop

Figure 1.1: Closed-loop and open-loop system

Using feedback in a control system has several advantages. Because a feedback controller has the output of the process as input, it can take immediate action if the output deviates from the desired value. This might be due to external disturbances or changes of the process parameters. Furthermore, because of the close connection of the two systems, a feedback controller may stabilize an unstable process. Of course, feedback controllers have disadvantages. The tight interconnection may also lead to instabilities causing oscillations or runaway behavior. Another drawback is that the sensors reading the process' output can introduce noise into the signal. Overall, if a

feedback controller is carefully designed, its use can improve the system performance in terms of speed, accuracy, cost, and efficiency.

Feedback control cannot only be found in engineering systems, but also in biological and economical systems. For example the regulation of the glucose concentration in the human bloodstream can be modeled as a feedback control system. Another example at a larger scale are population dynamics, e.g. the number of fishes that fishermen are allowed to catch without harming the fish population while maximizing the revenue can be derived by modeling the number of fishes and the harvesting rate in a feedback control system manner. In economics, the wholesale energy market can be modeled as a feedback control system [2].

Numerous examples for feedback control systems can be found in the engineering domain. One of the first feedback controllers was the centrifugal governor for steam engines by James Watt in 1788. Today, feedback controllers are extensively used in the automotive and aerospace domain but also in robotics, computer networks, the chemical industry and many others.

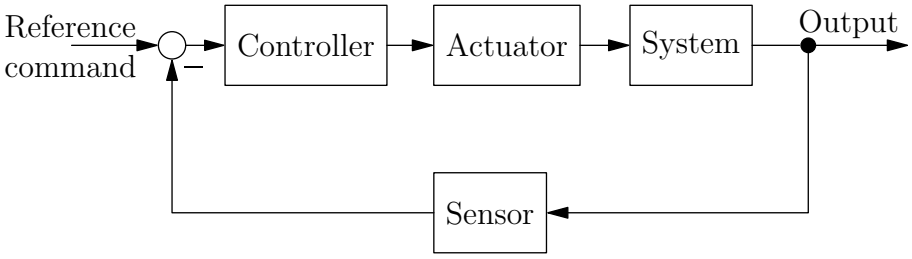


Figure 1.2: Schematic Feedback Control System

Typically, a feedback control system consists of sensors to monitor the system performance, actuators to introduce corrections to the system behavior, a reference command that indicates the desired behavior, and controllers that compute the timing and magnitude of these corrections (see Fig. 1.2 for a schematic). The field of control theory and engineering is dedicated to the analysis and synthesis of various control methods that determine the control structure for a given system.

A typical design procedure begins with the determination of the underlying model of the system to be controlled. Many physical systems can be described with ordinary differential equations (ODE) in continuous time. These ODEs may be based on conservation equations and constitutive relations. In some cases, difference equations are used to describe the system, and may be derived using a *System-Identification* based methodology that consists of evaluating input-output responses and determining a discrete-time system that best fits these responses. Similar to the system-model, an actuator model is determined as well. Once these models are derived, a controller is designed using state-space [3], frequency-domain [4], or geometric methods [5]. The success of the control performance is not only determined by the accuracy of the system models and the specific control method used, but also on the accuracy of modeling the implementation architecture.

The implementation architecture itself introduces errors of different kinds into the system [6]. These implementation errors can be modeled together with external disturbances in the form of a delay and a disturbance (see Fig. 1.3) that represent the effect of sampling, scheduling, discretization and quantization errors.

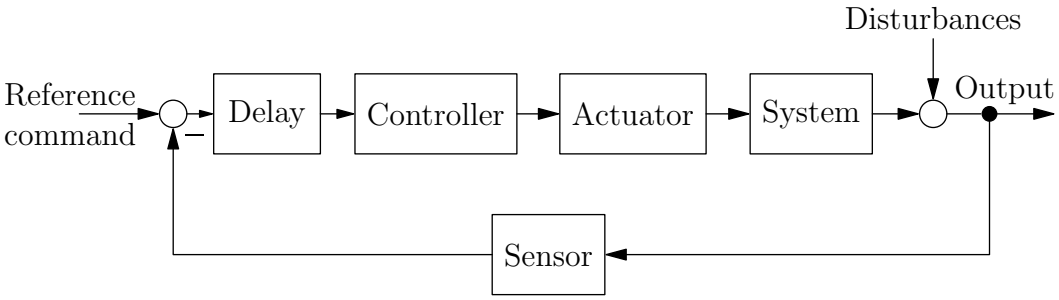


Figure 1.3: Schematic feedback control system with time delay and disturbance

The two main properties that need to be addressed while discussing Quality of Control are stability and performance. Stability is concerned with the well-behavedness of the system signals in the presence of feedback control. In particular, it is desired that the system signals do not exceed a certain bound. A stricter measure of stability is concerned with the regulation of the underlying error signals to zero.

Closed-loop control performance objectives typically include tracking of a command signal in addition to rejecting or minimizing the effects of disturbances, measurement noise, and modeling errors. The modern approach to characterizing these objectives is to analyze relevant matrix norms. Optimizing various performance objectives over the set of stabilizing controllers is the main thrust of recent optimal control methods, such as L_1, H_2, H_∞ control.

Adaptive Control

In contrast to classical feedback control where it is assumed that all parameters – system parameters, condition of the environment, disturbances, etc. – are completely known a priori, adaptive control refers to the control of partially known systems [7]. The characteristics of a real system are rarely known accurately enough and may change over time due to a various number of effects. For example, aging of a system can change its parameters such that a classically designed controller is no longer able to stabilize the system. Other effects such as changes in the environment, changes in the operating conditions, or unforeseen disturbances can also lead to a loss of stability with a classically designed controller. A conventional controller may not be able to guarantee satisfactory performance in the entire range over which the characteristics of the system may vary.

A definition of an adaptive system is difficult. Many different definitions have been discussed in the early years of research on adaptive control. A pragmatic definition of an adaptive controller is given in Definition 1.2.1.

Definition 1.2.1 (Adaptive controller [8])**.** An adaptive controller is a controller with adjustable parameters and a mechanism for adjusting the parameters.

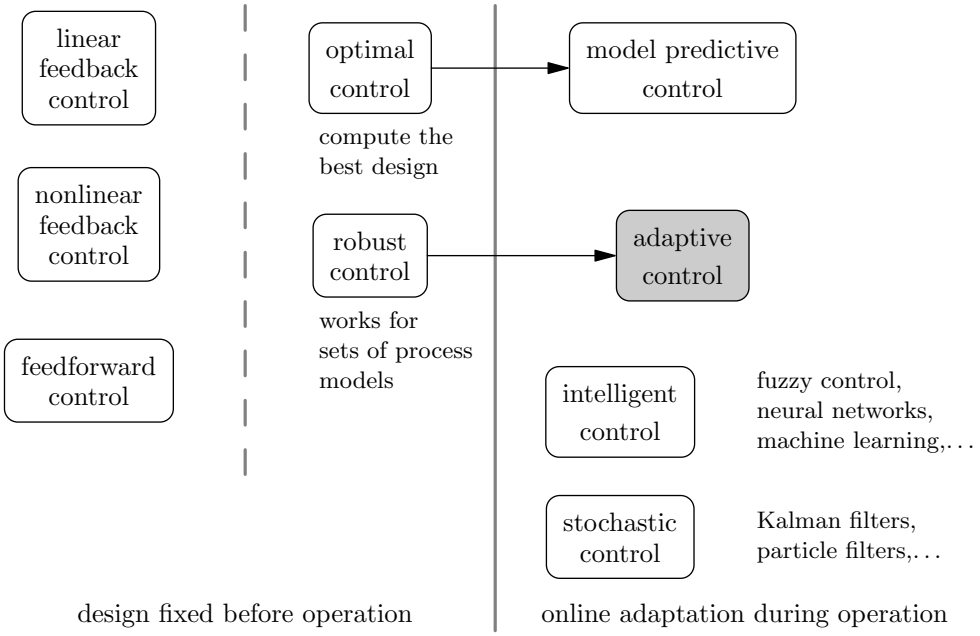


Figure 1.4: A (not exhaustive) landscape of control

Over the years, different types of adaptive control have been developed. The most well-known types are listed below.

Gain Scheduling: Controller parameters are adjusted based on measurable variables that correlate well with changes in the process dynamics.

Model Reference Adaptive Systems: The controller consists of two loops. The inner loop is an ordinary feedback loop. The outer loop adjusts the controller parameters such that the error between model and process output is small. The adjustment rules are designed so as to ensure stability of the overall nonlinear feedback system.

Self-Tuning Regulators: The controller consists of two loops. The inner loop is an ordinary feedback loop. The parameters of the controller are adjusted by the outer loop, which is composed of a recursive parameter estimator and a design calculation.

Extremum Seeking Control: This is an optimal control approach that deals with situations when the plant model and/or the cost to optimize are not available to the designer but it is assumed that measurements of plant input and output signals are available.

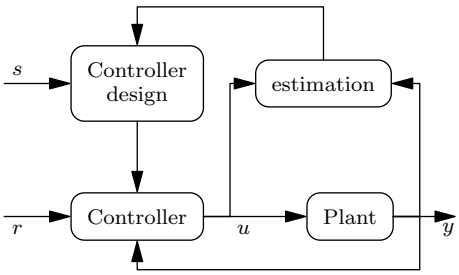


Figure 1.5: Indirect Control or Explicit Identification: Estimate parameters on-line and update control parameters based on estimates.

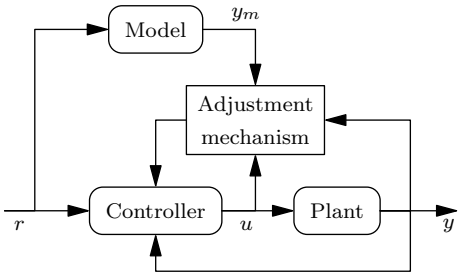


Figure 1.6: Direct Control or Implicit Identification: Control parameters are not identified explicitly, but directly adjusted to improve a performance index.

If a system is only partially known or the disturbances acting on the the system are not known, identification procedures can be used to gather information on the system and its environment. However, if the goal is to control the system, identification alone is not sufficient. Neither is it advantageous to control the system without knowing its parameters. An adaptive controller has thus to solve two problems simultaneously: First, it has to determine the characteristics of the system, and second, it has to determine the necessary control actions that lead to the desired goal. There are two different approaches how to solve this problem. The first approach is to estimate the parameters on-line and update the control parameters based on the estimate, illustrated in Figure 1.5. This approach is referred to as *indirect control* or *explicit identification*. The second approach does not identify the system parameters explicitly, but the control parameters are adjusted directly in order to improve a performance index, see Figure 1.6. This approach is referred to as *direct control* or *implicit identification*.

Since their first use in the late 1950s in autopilots for high-performance aircrafts ([9]) and the proofs of stability in the 1970s and robustness in the 1980s ([10–12]), adaptive controllers have been widely applied to industrial and academic problems. These include steering control of ships ([13, 14]), autonomous underwater vehicles ([15]), process control for distillation columns ([16]), load-frequency and generator exciter control of power systems ([17]), robot manipulators ([18]), blood pressure and muscular control in the field of bioengineering ([19, 20]), and unmanned aerial vehicles ([21, 22]).

1.2.2 Networked Control Systems

Networked Control Systems (NCS) are traditionally concerned with the modeling, design, and analysis of spatially distributed dynamical systems that are connected via some form of shared communication network (see Figure 1.7). The first commercial applications of NCS can be found in the 1970's [23] in automobile industry. The driving forces for this were to reduce the cost for cabling, to increase the flexibility, to handle the increasing amount of control loops, and to modularize the systems. NCS

can also be identified in a biological system. Neurons are the messages, muscles the actuators, sensory organs like skin or eyes are the sensors, neural pathways are the communication network, and the cerebral cortex acts as controller.

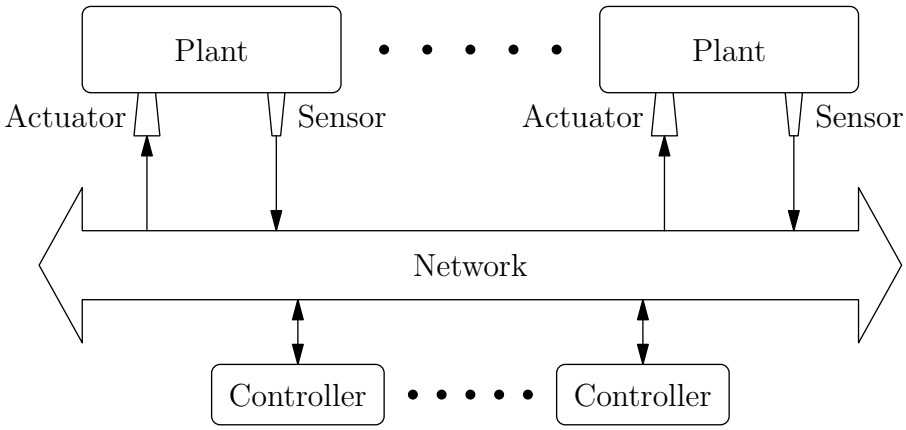


Figure 1.7: A general NCS

NCS have become a widely used technique in many fields. For example, NCS are currently successfully used in automobiles [24], remote surgery [25], environmental monitoring and surveillance [26], building automation [27], refineries and power systems [28], haptics collaboration over the Internet [29], and many more. Many of these examples illustrate the advantages of a NCS compared to classical (digital) hardwired connections. NCS cause no or little cost when components are added or removed, increase the reconfigurability of the systems, simplify the diagnosis and maintenance, and increase the overall flexibility of control systems.

Recent research topics in the area of NCS are dealing with "controlling the formation of ad hoc networks of spatially distributed systems, system-dependent data rate requirements in digital feedback channels, real-time fusion and registration of data from distributed heterogeneous sensors, and the theory of cooperative control of networks of autonomous agents." [30].

The theoretical foundations of classical control with continuous-time and continuous-state models were laid between 1930 and 1955 by the work of Nyquist, Bode, Nichols, and Evans [31, 32]. The next step in the evolution of control systems was the introduction of digital control with discrete-time and quantized states in the 1950's. Distributing the components spatially was easier with digital control systems, but still required hardwiring the components. Through technological advances it is nowadays possible to put low-cost processing units at remote locations and transmit the information over shared buses or wireless in a secure and reliable way.

Challenges of control over networks

Since NCS combine two different areas, namely control theory and communication theory, the challenges faced in NCS are different from classical control systems. Among these challenges are the following:

Limited-network resources: The use of communication networks restricts the amount of information that can be exchanged between two nodes. Especially, if many nodes are transmitting data over the same network, questions like scheduling of messages and network bandwidth arise [33–35].

Sampling and Delay: Physical systems produce continuous-time signals which have to be sampled to transmit them over a communication network [6]. The amount of time it takes to send a message over the network, is subject to many parameters. The overall delay is influenced by the arbitration policy of the network, i.e., when is which node allowed to access the network, and the network transmission time [36, 37].

Packet Dropouts: Since real communication networks are non-ideal channels, the loss of packets is possible. This might be due to errors in the physical layer or due to congestion. Another source of packet dropouts can be very large time-delays, e.g., when outdated packets are discarded by the receiver [30, 38, 39].

Synchronization: To give two nodes in the network the same or almost the same clock value is called clock synchronization and is a research area itself [23, 40, 41]. The timing of sensing and actuation actions has to be coordinated in a NCS in order to guarantee Quality of Service (QoS) requirements. Synchronization of local clocks is also important for keeping time delays as small as possible.

1.2.3 Embedded Control Systems

Embedded control systems are ubiquitous and can be found in several applications including aircraft, automobiles, process control, buildings, and even kitchen appliances. An embedded control system is one in which the computer system is designed to perform dedicated functions with real-time computational constraints [42]. Typical features of such embedded control systems are shared networks used by different components of the systems to communicate with each other, a large number of sensors as well as actuators, and their distributed presence in the overall system.

One of the first embedded systems was the Apollo Guidance Computer created in 1961 for the Apollo program by NASA [43]. Since then, the capabilities regarding processing power and functionality of embedded systems have been increased dramatically. This started with the first microprocessor, the Intel 4004, designed for a pocket calculator and is not over yet. The main breakthrough was the ability to reuse the hardware of a general-purpose computer by changing the software if the purpose of the system changes. Another reason why embedded systems are almost everywhere is that many sophisticated control algorithms can only be implemented with microprocessors. This is especially true in the automobile domain. In modern cars up to 100 processors are used.

The characteristics of embedded systems are different from the ones for standard PCs.

Complexity: The algorithms implemented on microprocessors are highly sophisticated. This can be for example advanced control algorithms or complicated filtering functions.

Real-time: Timing constraints are important for many embedded systems. Missing information or delayed execution of a program on the embedded systems might lead to a fatal fault or degrades the performance which makes the user unhappy.

Multirate: Embedded systems usually control many different things at the same time. Some might run at slow rates, others at high rates, but all must meet their deadlines. The audio and video portion in a multimedia stream are a prime example of tasks with very different rates.

Cost: Manufacturing costs are important for any commercial product.

Energy: Power consumption directly affects the cost of embedded systems, but also heat dissipation is an important factor.

In the design of embedded systems, several challenges have to be taken into account that are different from programming a standard PC [44]:

Hardware: The amount of hardware has to be a trade-off between the ability to meet all deadlines and the cost for the embedded system.

Deadlines: In order to meet all deadlines, the CPU speed might be increased which makes the system more expensive. However, this is no guarantee to meet all deadlines since the speed of other components such as memory might not be increasable.

Power consumption: If the embedded system runs on battery it is essential to minimize power consumption. One way to do this is slowing down the processor, but this may lead to missed deadlines.

Upgrades: The ability to change the software in order to add new features is important for embedded systems since they might run for several years.

Reliability: Since embedded systems are often used in safety-critical systems, it is important that the number of bugs is minimized before the system is shipped out.

Embedded systems with microprocessors make it possible to use sophisticated algorithms everywhere in a cheap way. However, the programming and design of an embedded system is much more difficult than programming a standard PC.

1.3 Related Work

The problem of bridging the gap between the semantics of control models and that of the implementation platform has of late attracted a lot of attention, for example in [45–51, 51, 52]. Ngheim *et al.* in [45] and Yazarel *et al.* in [46] have quantified this exact gap/error for linear controllers implemented on time-triggered platforms. In [47, 48], an empirical co-design is proposed by which a controller is first derived that is robust to specified sampling period and delays, and then a scheduling procedure is proposed that is shown in simulation results using the Jitterbug toolbox [49] to lead to a desired QoC and meet the scheduling constraints. However, no analytical guarantees are provided for this co-design and no closed-form expressions of the message delay from the scheduler parameters is derived. Also, no hybrid protocols are considered. If any changes are made, this entire process of control and scheduling designs is repeated. A similar sequential co-design is proposed in [53], where a control system is designed using a delay-impulsive modeling approach first, and then a physical architecture with an EDF scheduling policy is chosen. A schedulability analysis is carried out that guarantees that all control-related messages can be transmitted in finite time. If schedulability is not feasible, the design procedure starts over. Notable shortcomings are that no analytical guarantee is provided and the design methods are not incremental (i.e., the entire design cycle repeats in the case of future changes). Reference [53] addresses analysis and implementation of a distributed control system on a network of communicating control unit via control system analysis in terms of sampling times and delays, mapping of control loops to computation/communication hardware components, and scheduling analysis. Other efforts in the same direction include [50], [51], [52] where the goal was again to estimate suitable choices of scheduler periods or priorities with the aim of improving control performance. In cases, such as in [51], analytical solutions have also been found.

A common feature in networked control systems is the need for shared resources. Constrained by space, speed, and cost, often information has to be transmitted using a shared communication network. In order to manage the flow of information, protocols that are time-triggered [45] and event-triggered [50, 54] have been suggested over the years. In order to maintain flexibility in scheduling while minimizing delays for critical functions and conserving resources, a hierarchical schedule such as FlexRay has been proposed in [55] and has been used in several applications [56]. While several papers have considered control using TT protocols (see for example, [45, 57]) and ET protocols (see for example, [50, 54, 58, 59]), control using hybrid protocols has not been studied in the literature.

The digital implementation of the communication network introduces a delay. Therefore, for accurate performance, the control system has to explicitly incorporate the delay in its design. A good deal of the research on networked control systems (NCS) is focused on this effort (for example, [39, 60, 61] and references therein). A specific aspect of NCS is the combined design of its two major components of control and communication. This co-design is being explored more recently in [48, 51, 53, 53, 57, 62–

66, 66–78]. In [65], a Control Server was used to reduce the input-output latency. In addition, different strategies such as a feedback loop for sampling times were used to minimize jitter. In [74], feedback-feedforward loop was used to adjust the sampling period of the control tasks in order to optimize a performance criterion. In [67], the problem of optimal control and scheduling was solved by transforming the system into a mixed integer quadratic programming problem.

Control theoretic principles based on linear systems, feedback control, and optimization are used to determine parameters such as sampling period and resource allocation so as to maintain both an efficient control performance and CPS utilization. In [70] and [51], the schedulability analysis of real-time tasks with respect to the stability of control functions is discussed for relatively simple platforms. The problem discussed in [51] is a continuation of the work in [70]. The control delay, which has a large impact on the control performance, is considered in the computation and assignment of optimal task periods. However, the platforms are simple platforms and the focus of this thesis is the use of more advanced platforms. In [72], modeling the real-time scheduling process as a dynamic system, an adaptive self-tuning regulator is proposed to adjust the bandwidth of each single task in order to achieve an efficient CPS utilization. But the focus, however, is on a single processor. In [48], for subsets of the set of applications a schedule is derived with an iterative procedure. Theses schedules are stored on the processor if they are necessary and fit into the memory. In [78] the problem of controlling multiple single-input single-output (SISO) systems using a single processor using task scheduling is considered. A good survey paper on co-design can be found in [66].

Connections between control performance and architecture design have been explored in a number of papers (see, for example, [51, 57, 58, 70, 79–83]). The idea in most of these papers is that better control performance is achieved by redesigning the architecture in a suitable manner. In [58, 79], a closed-loop state-based strategy is used to determine an aperiodic sampling sequence and therefore a corresponding scheduling strategy that ensures the desired control performance. Other event-based strategies are used in [51, 80, 81] in order to determine the sampling sequence. In [82, 83], automaton-based approaches are utilized for the design of online schedulers which in turn guarantee robust control performance with LQG controllers.

In [48, 51, 70–74], the co-design of control in CPS has been addressed, with particular focus on the design of scheduling policies for effective Quality of Control (QoC). Control theoretic principles based on linear systems, feedback control, and optimization are used to determine parameters such as sampling period and resource allocation so as to maintain both an efficient control performance and CPS utilization.

Adaptive systems with switching and tuning using multiple models have been considered extensively in the past decade (see for example, [84–88]). However, as will be seen in the subsequent sections, the properties of the switching protocol in the DES introduces a switch in the underlying delay in the plant to be controlled. Such classes of plant models have not been addressed thus far in the literature, making both stan-

dard adaptive solutions as well as switching adaptive control solutions inadequate. Switched control systems and related areas of hybrid systems and supervisory control have been studied extensively in the literature (see e.g., [84–86, 88–93]). However, in most of these cases, the design or dependence of these switches on the implementation platform has not been made.

Adaptive control in NCS is a topic that is being explored relatively recently. While a gain-scheduled approach has been used more commonly [94, 95], an adaptive approach which combines elements of parameter estimation followed by an update in the control algorithm has been investigated minimally (see [96, 97] for example). In [96], a least-squares error is used to adjust the estimate of the unknown time-delay, while in [97], the controller gains are recalculated every so often, assuming that the delay has changed and is known, so that a desired performance metric is optimized. No theoretical discussion of the underlying stability is provided in either of these papers.

A final point to note is the event-based actions that are addressed in this thesis. Papers [54, 58, 59] are similar to what is proposed in this thesis in that both approaches use an "event" to trigger control decisions. This event is based on a system error falling below, or exceeding a certain threshold, in this work and in [54, 58, 59], respectively. However, the actual control decision that is taken following the event is distinctly different in the two cases. In [54, 58, 59], this event triggers the sending or non-sending of a control message. In our case, as will be evident in the forthcoming sections, this event triggers a switching between two control strategies as well as switching between two protocols, with the overall switching closed-loop system remaining stable.

1.4 Major Contributions

In the following, the main contributions of this thesis are summarized.

Arbitrated Networked Control Systems: Traditional NCS research is concerned with challenges like message loss, jitter, or varying delays which arise in the context of distributed control over a given network. However, in embedded platforms the network itself is not fixed a priori, but offers the possibility to jointly design and optimize the network and the controller. To emphasize the connection of controller design and network design – especially the arbitration policy of the shared communication medium – this is referred to as *Arbitrated Networked Control Systems (ANCS)*. The term is novel and different examples of ANCS and solutions based on ANCS are presented in this thesis.

ANCS with Hierarchical Schedules: The additional information provided by an ANCS is used to improve the performance of an NCS with hierarchical schedules. The following contributions are made:

- Representing the network as a delay, excludes a lot of information. If the network is modeled in more detail, tighter bounds on the delay can

be derived. An ANCS provides this information and thus the Real-Time Calculus (RTC) framework can be used to compute the worst-case end-to-end delays in a NCS.

- A novel performance metric is presented. Since RTC provides only upper bounds on the delay, but the average delay is smaller, it is shown analytically that with this novel performance metric the performance monotonically increases with decreasing delay.
- The RTC framework and the performance metric are then combined into a co-design of control and communication by solving a nonlinear optimization problem with multiple performance metrics. In contrast to other co-design approaches, this approach provides the optimal choice of controller and scheduler parameters by the one-time solution of an optimization problem.
- By applying the co-design to a platform with a hierarchical schedule, it is shown how the performance of complex arbitration schemes can be optimized.
- Since uncertainties are present in any system, the concept of adaptive control is added to the ANCS, resulting in improved performance of the overall system.

ANCS with Hybrid Schedules: A co-design of control and communication with hybrid protocols is presented. The following contributions are made:

- The specific solution proposed is a co-design where the switches of the controller are aligned synergistically with the switches in the embedded platform. While part of the switches are introduced due to disturbances and therefore are due to the environment, part of the switches are by design.
- The plant being controlled is assumed to be unknown, and an adaptive control solution is employed to accommodate the uncertainty. This significantly complicates the problem and necessitates a fairly involved proof of stability. The main challenge in demonstrating the stability of the underlying switching adaptive controller is the fact that the delay in the underlying plant-model switches between two values which, in turn, is due to the hybrid use of the TT and ET protocols.
- Since the hybrid protocol is highly attractive because of the advantages it offers and is becoming more and more prevalent among embedded systems, the proposed solution will have a significant impact.
- The proposed co-design is supported by a full-fledged simulation. To this end the functionality of the TrueTime toolbox had to be extended. With this simulation it is shown that the proposed co-design improves the performance of an ANCS.

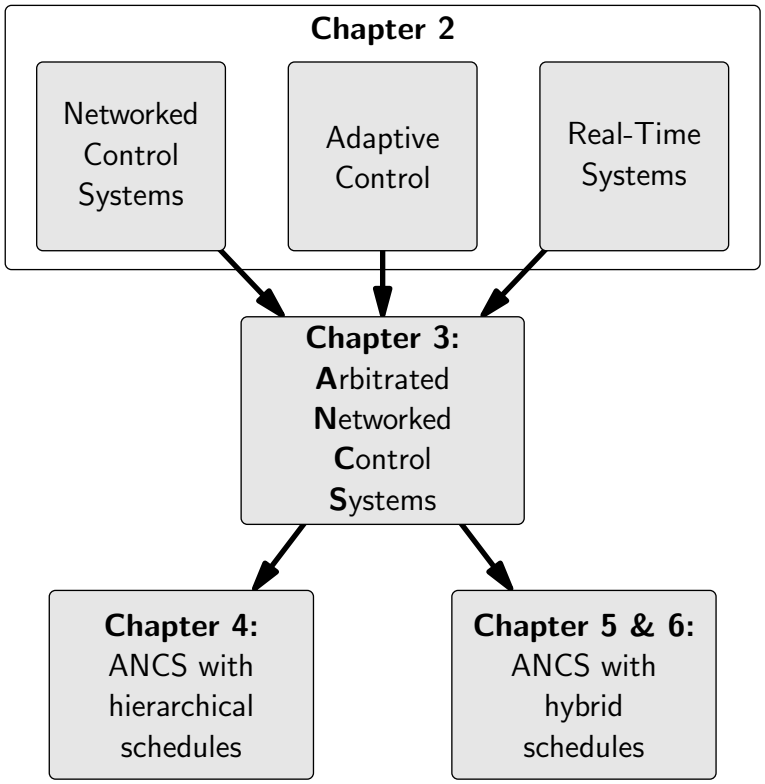


Figure 1.8: Outline of the dissertation

1.5 Thesis Outline

The general outline of this thesis is shown in Figure 1.8. In Chapter 2, the basic concepts that are needed later are introduced and described. First, an introduction to feedback control systems with emphasis on sampled-data systems is given. Next, the concept of adaptive control is introduced and an extension of the persistent excitation concept is presented. Finally, an overview of real-time systems is presented. Within this section, the Real-Time Calculus (RTC) Framework [98, 99] is introduced which allows to compute worst-case end-to-end delays of arbitrary platforms.

In Chapter 3, the three concepts described in Chapter 2 are used to introduce the Arbitrated Networked Control Systems (ANCS) approach to Cyber-Physical Systems. The term Cyber-Physical Systems refers to a relatively new idea and describes the need for integrated research in control and communication. The Arbitrated Networked Control System Approach refers to a special class of Cyber-Physical Systems where the emphasis is on the arbitration in the communication network. Some solutions for problems arising within Cyber-Physical Systems will be shown. It is shown that ANCS provide additional information about the network and that this information can be used to design better controllers. The following three chapters provide novel solutions for ANCS with hierarchical and hybrid communication networks. This chapter follows the publication [100].

In Chapter 4, it is shown how the information provided by an ANCS can be used to

improve the control performance of an NCS with a hierarchical schedule. In particular, in the first part of this chapter a novel performance metric is developed that allows to compute the optimal controller parameters for a given worst-case delay together with other standard performance metrics. The first part follows the publications [98, 99]. In the second part, it is shown how the performance of a NCS can be improved using an adaptive controller by utilizing information from the ANCS. This part is described in [101, 102].

In Chapter 5, an ANCS is presented that uses a hybrid, co-designed control and communication protocol for the control of multiple applications. The co-design consists of an adaptive switching controller and a hybrid communication architecture that switches between a time-triggered and event-triggered protocol governed by a switching algorithm. An adaptive controller is proposed in order to cope with any uncertainties that may be present. The resulting adaptive switching controller is then shown to be stable and to achieve regulation and tracking in the presence of a class of disturbances. The results for the regulation case are described in [103] and for the tracking case in [104]. In Chapter 6, numerical case studies with a full-fledged simulation are presented. It is shown through simulations that the co-design proposed in Chapter 5 achieves a better performance than a fixed controller design. Finally, it is shown that the proposed co-design uses fewer expensive TT slots than a fixed controller design. This part is published in [105].

Finally, a conclusion of this thesis and an outlook for future work is provided in Chapter 7.

2 Background

Summary: *In this chapter, basic concepts that are needed later are introduced and described. First, an introduction to feedback control systems with emphasis on sampled-data systems is given. Next, the concept of adaptive control is introduced and an extension of the persistent excitation concept is presented. Finally, an overview of real-time systems is presented. Within this section, the Real-Time Calculus Framework is introduced which allows to compute worst-case end-to-end delays of arbitrary platforms.*

2.1 Feedback Control Systems

Physical, mechanical or biological processes are commonly described by ordinary differential equations (ODEs). Often, a state of a system is defined by the rate of change of this state. This leads naturally to differential equations. A definition of an ODE is given by Definition 2.1.1.

Definition 2.1.1 (Ordinary Differential Equation (ODE))**.** An *Ordinary Differential Equation (ODE)* of order n is an equation of the form

$$f\left(t, y, \dot{y}, \ldots, y^{(n)}\right) = 0, \tag{2.1}$$

where y is a function of t, $\dot{y} = \frac{dy}{dt}$, and $y^{(n)} = \frac{d^n y}{dt^n}$.

A control system cannot be described with an ODE since no influence on the system is possible. Hence, control systems are commonly described by ODEs with input:

Definition 2.1.2 (State Space Model)**.** A *state space model* is given by the differential equation

$$\frac{dx}{dt} = f(x, u) \qquad y = h(x, u) \tag{2.2}$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ and $h : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^q$ are smooth mappings. The vector $x \in \mathbb{R}^n$ is called the *state vector*, $u \in \mathbb{R}^p$ is the *input vector*, and $y \in \mathbb{R}^q$ is the *output vector* and contains the measured signals.

If f and h are linear functions in x and u, the *linear state space model* can be represented by

$$\dot{x} = Ax + Bu \qquad y = Cx + Du \tag{2.3}$$

with A, B, C, and D matrices of appropriate dimensions.

Many analysis and design techniques in automatic control theory use the *frequency domain representation* of a linear state space model, the *transfer function*. For the system given in Equation (2.3) the transfer function is given by

$$G(s) = C(sI - A)^{-1}B + D. \tag{2.4}$$

For Single-Input Single-Output (SISO) Systems with the first-order linear differential equation

$$a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \ldots + a_0 y(t) = b_m u^{(m)}(t) + \ldots + b_0 u(t) \tag{2.5}$$

with the initial conditions $y^{(i)}(t_0) = 0, i = 0, \ldots, n$ and $u^{(i)}(t_0) = 0, i = 0, \ldots, m$, the transfer function is given by

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_0}. \tag{2.6}$$

2.2 Sampling of Continuous-time Systems

Consider a control system which controls a plant described by a linear continuous-time model

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t - \tau) \\ y(t) &= C_c x(t) \end{aligned} \tag{2.7}$$

with $A_c \in \mathbb{R}^{n \times n}$, $B_c \in \mathbb{R}^{n \times 1}$, $C_c \in \mathbb{R}^{1 \times n}$, and a time-delay $\tau > 0$. Since the controller is implemented in a digital manner, a *sampled* version of (2.7) is necessary.

Sampling is the process of replacing a continuous-time signal by its values at a discrete set of points ([6]). Let $\{t_k : k \in \mathbb{Z}\} \subset \mathbb{R}$ be the sampling instants, then the sequence $\{f(t_k) : k \in \mathbb{Z}\}$ is the sampled version of the continuous-time signal f. If there exists $h \in \mathbb{R}$ such that $t_k = k \cdot h$, then the sampling is called *periodic sampling* and h the *sampling period*. Periodic sampling is the most common way of sampling, although there are other types of sampling like multirate sampling where multiple sampling periods are used for different control loops [106]. The question when a continuous-time signal can be reconstructed exactly from its periodically sampled version is answered by the following theorem.

Theorem 2.2.1 (Shannon's sampling theorem [107])**.** *A continuous-time signal with a Fourier transform that is zero outside the interval $(-\omega_0, \omega_0)$ is given uniquely by its values in equidistant points if the sampling frequency is higher than $2\omega_0$. The continuous-time signal can be computed from the sampled signal by the interpolation formula*

$$f(t) = \sum_{k=-\infty}^{\infty} f(kh) \frac{\sin(\omega_s(t - kh)/2)}{\omega_s(t - kh)/2} \tag{2.8}$$

where ω_s is the sampling angular frequency in rad/s.

A proof of Theorem 2.2.1 can be found in [6]. The reconstruction of a continuous-time signal using Equation (2.8) is not a causal reconstruction, is difficult to compute, and is only applicable to periodic sampling. Therefore other reconstructions are needed that can be easily applied to control systems. One way to obtain a simple causal reconstruction that can be applied to any type of sampling is given by

$$f(t) = f(t_k); \quad t_k \leqslant t < t_{k+1}. \tag{2.9}$$

The reconstruction given in Equation (2.9) is called *zero-order hold*, see Figure 2.1. The reconstructed signal is piecewise constant, continuous from the right, equal to the sampled signal at the sampling instants, and held constant until the next sampling instant. Although the zero-order hold is not an exact reconstruction for almost all signals, it is the most commonly used reconstruction because of its simplicity.

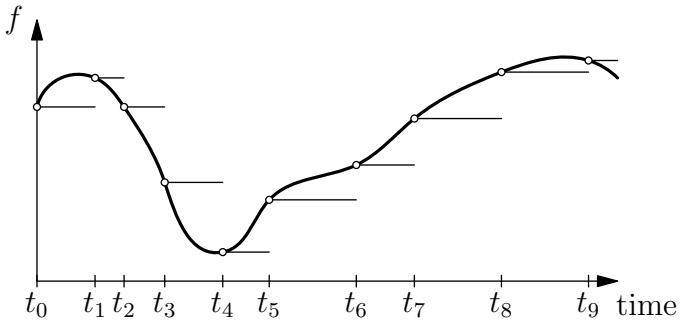


Figure 2.1: Non-periodic Sampling of a continuous-time signal f and zero-order hold reconstruction

In order to obtain the discrete-time equivalent of the continuous-time system (2.7), the system is sampled periodically and it is assumed that the continuous-time control input is reconstructed via zero-order hold. It is further assumed that the total time delay in the system τ is less than one sampling period h. The state of the plant at the sampling time $(k+1)h$ is then given by

$$\begin{aligned} x((k+1)h) &= e^{A_c h}x(kh) + \int_{kh}^{kh+h} e^{A_c(kh+h-s')}B_c u(s'-\tau)ds' \\ &= e^{A_c h}x(kh) + u(kh-h)\int_{kh}^{kh+\tau} e^{A_c(kh+h-s')}B_c ds' \\ &\quad + u(kh)\int_{kh+\tau}^{kh+h} e^{A_c(kh+h-s')}B_c ds' \end{aligned} \tag{2.10}$$

where the last step follows from the fact that the control input is held constant over one sampling period but changes due to the time delay (see Figure 4.14). With

$$\begin{aligned} \Psi(\tau) &= e^{A_c h} \\ \Gamma_0(\tau) &= \int_0^{h-\tau} e^{A_c s}ds B_c \\ \Gamma_1(\tau) &= e^{A_c(h-\tau)}\int_0^{\tau} e^{A_c s}ds B_c \end{aligned} \tag{2.11}$$

the sampled version of (2.7) can be written as

$$x(kh+h) = \Psi x(kh) + \Gamma_1 u(kh-h) + \Gamma_0 u(kh). \tag{2.12}$$

The state-space form of the sampled version of the system in Equation (2.7) is thus given by:

$$\begin{aligned} \begin{bmatrix} x(kh+h) \\ u(kh) \end{bmatrix} &= A_s(\tau) \begin{bmatrix} x(kh) \\ u(kh-h) \end{bmatrix} + B_s(\tau) u(kh) \\ y(kh) &= C_s(\tau) \begin{bmatrix} x(kh) \\ u(kh-h) \end{bmatrix} \end{aligned} \tag{2.13}$$

where

$$A_s = \begin{bmatrix} \Psi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \qquad B_s = \begin{bmatrix} \Gamma_0 \\ 1 \end{bmatrix} \qquad C_s = \begin{bmatrix} C & 0 \end{bmatrix}. \tag{2.14}$$

The dependence of A_s, B_s, and C_s in (2.14) on τ is due to the dependence of Ψ, Γ_0, and Γ_1 as in (2.11).

Due to the time delay an additional state variable has to be added which stores the previous control input signal. In case of a time-delay τ larger than the sampling period h, $\lceil \frac{\tau}{h} \rceil$ additional states are needed.

Since only single-input single-output systems are being considered, Equation (2.13) can be transformed into an input-output model of the form

$$y(k) = H(q^{-1})u(k) \tag{2.15}$$

with q^{-1} being the backward shift operator and

$$H(q^{-1}) = \left[C_s (I - q^{-1} A_s)^{-1} q^{-1} B_s \right] \tag{2.16}$$

and A_s, B_s, and C_s defined as in (2.14). Since A_s, B_s, and C_s depend on τ, it is clear that also $H(q^{-1})$ depends on τ.

An equivalent input-output representation of the model in Equation (2.15) is given by

$$A(q^{-1})y(k) = q^{-d} B(q^{-1})u(k); \qquad k \geqslant 0 \tag{2.17}$$

where the polynomials A and B are given by

$$A(q^{-1}) = 1 + \sum_{l=1}^{m_1} a_l q^{-l} \qquad B(q^{-1}) = q^d B'(q^{-1}) = b_0 + \sum_{l=1}^{m_2} b_l q^{-l} \tag{2.18}$$

2.2.1 Control of Deterministic Models

In this section a controller structure that ensures tracking of a reference signal y_{ref} is described.

For any delay d, Equation (2.17) can be expressed in a *predictor form* as follows [108]:

$$y(k+d) = \alpha(q^{-1})y(k) + \beta(q^{-1})u(k) \tag{2.19}$$

with

$$\begin{aligned} \alpha(q^{-1}) &= \alpha_0 + \alpha_1 q^{-1} + \ldots + \alpha_{m_1-1} q^{-(m_1-1)} \\ \beta(q^{-1}) &= F(q^{-1})B(q^{-1}) \\ &= \beta_0 + \beta_1 q^{-1} + \ldots + \beta_{m_2+d-1} q^{-(m_2+d-1)} \end{aligned} \tag{2.20}$$

where $F(q^{-1}) = f_0 + f_1 q^{-1} + \ldots + f_{d-1} q^{-(d-1)}$ and $\alpha(q^{-1}) = \alpha_0 + \alpha_1 q^{-1} + \ldots + \alpha_{m_1-1} q^{-(m_1-1)}$ are the unique polynomials that satisfy the equation

$$1 = F(q^{-1})A(q^{-1}) + q^{-d}\alpha(q^{-1}). \tag{2.21}$$

Equation (2.19) can be expressed as

$$y(k+d) = \theta^{*T}\Phi(k) \tag{2.22}$$

$$= \vartheta^{*T}\phi(k) + \beta_0 u(k) \tag{2.23}$$

where $\phi(k)$, ϑ^*, $\Phi(k)$, and θ^* are defined as

$$\phi(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k-m_1+1) \\ u(k-1) \\ \vdots \\ u(k-m_2-d+1) \end{bmatrix} \quad \vartheta^* = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{m_1-1} \\ \beta_1 \\ \vdots \\ \beta_{m_2+d-1} \end{bmatrix} \tag{2.24}$$

$$\Phi(k) = \begin{bmatrix} \phi(k) \\ u(k) \end{bmatrix} \text{ and } \theta^* = \begin{bmatrix} \vartheta^* \\ \beta_0 \end{bmatrix} \tag{2.25}$$

with $\phi(k) \in \mathbb{R}^{m_1+m_2+d-1}$, $\vartheta^* \in \mathbb{R}^{m_1+m_2+d-1}$, $\Phi(k) \in \mathbb{R}^{m_1+m_2+d}$, $\theta^* \in \mathbb{R}^{m_1+m_2+d}$, and $\alpha_j, j = 0, \ldots, m_1-1$ and $\beta_j, j = 0, \ldots, m_2+d-1$ the coefficients of the polynomials in (2.20) with respect to the delay d and finite initial conditions

$$\begin{aligned} y(k-i) &= y_0(i) \quad i = 0, \ldots, m_1-1, \\ u(k-i) &= u_0(i) \quad i = 1, \ldots, m_2+d-1. \end{aligned} \tag{2.26}$$

From Equations (2.22)-(2.25), it can be observed that a feedback controller of the form

$$u(k) = \frac{1}{\beta_0}\left(y_{\text{ref}}(k+d) - \vartheta^{*T}\phi(k)\right) \tag{2.27}$$

realizes the objective of stability and follows the desired bounded trajectory $y_{\text{ref}}(k)$ in the absence of disturbances for any d. Designing a stabilizing controller $u(k)$ essentially

boils down to a problem of implementing (2.27) with the controller gain ϑ^*. It should be noted that the dimension of $\phi(k)$, ϑ^* as well as the entries of ϑ^* depend on the delay d.

Since $H(q^{-1})$ depends on tau, it is clear that the parameter vector θ^* directly depends on the delay τ, i.e., there exists some function f with

$$\theta^* = f(\tau). \tag{2.28}$$

In order to illustrate the dependency of θ^* on τ, the delayed single-input single-output system

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t-\tau) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned} \tag{2.29}$$

with the sampling rate $h = 20\,\text{ms}$ is considered and θ^* is computed for different values of τ. In Figure 2.2, the variation of θ^* with the time-delay τ for $\tau \in [1\,\text{ms}; 19\,\text{ms}]$ is shown. It should be noted that α_0 and α_1 are not shown since they do not change with respect to τ. As can be seen from the figure, the parameter variation can be rather significant and can therefore impact the control performance.

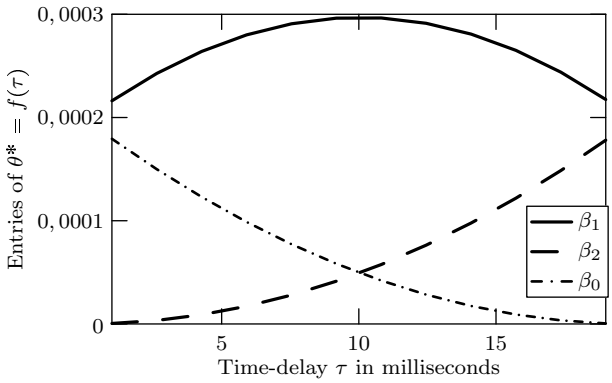


Figure 2.2: Entries of θ^* when τ varies from 1 ms to 19 ms.

2.2.2 Disturbance Model

In order to improve the model of the plant, a disturbance term is added. The model in Equation (2.17) thus becomes

$$A(q^{-1})y(k) = q^{-d}B(q^{-1})u(k) + G(q^{-1})D(k) \qquad k \geqslant 0 \tag{2.30}$$

where it is assumed that $G(q^{-1}) = 1$, i.e., the disturbances are impulses. The predictor form of (2.30) is then given by

$$y(k+d) = \alpha(q^{-1})y(k) + \beta(q^{-1})u(k) + \gamma(q^{-1})D(k) \tag{2.31}$$

with $\alpha(q^{-1})$ and $\beta(q^{-1})$ as in (2.20) and

$$\gamma(q^{-1}) = F(q^{-1})G(q^{-1}) = F(q^{-1}) \tag{2.32}$$

where $F(q^{-1})$ is one of the unique polynomials that satisfy the Diophantine Equation (2.21). Equation (2.31) can be expressed as

$$\begin{aligned} y(k+d) &= \theta^{*T}\Phi(k) + \mu^T\Delta(k) && (2.33)\\ &= \vartheta^{*T}\phi(k) + \beta_0 u(k) + \mu^T\Delta(k) && (2.34) \end{aligned}$$

where $\phi(k)$, ϑ^*, $\Phi(k)$, θ^* are given in Equations (2.24) and (2.25) and μ and $\Delta(k)$ are defined as

$$\mu = \begin{bmatrix} f_0 \\ \vdots \\ f_{d-1} \end{bmatrix} \quad \Delta(k) = \begin{bmatrix} D(k) \\ \vdots \\ D(k-d+1) \end{bmatrix}. \tag{2.35}$$

2.3 Adaptive Control

In this section, a basic adaptive controller for discrete-time systems is introduced and an extension of the persistent excitation property is presented.

2.3.1 A Basic Adaptive Controller

Given a system like the one in Equation (2.7) with unknown system parameters A_c, B_c, and C_c, the goal is to design an adaptive controller to stabilize the system and follow a reference trajectory with a digital controller. Hence, the controller has to be of a discrete nature and the controller given in Equation (2.27) might be used. Since the system parameters A_c, B_c, and C_c are unknown, it is clear from the previous sections that also ϑ_d^* and β_0^d given in Equations (2.24) and (2.25) are unknown and thus the controller in Equation (2.27) cannot be used.

In order to utilize the structure of (2.27), ϑ^* and β_0 are replaced by their parameter estimates. Then the following adaptive control input is derived

$$u(k) = \frac{1}{\hat{\theta}_{d,\nu}(k)}\left(y_{\text{ref}}(k+d) - \hat{\vartheta}_d(k)^T\phi_d(k)\right) \tag{2.36}$$

where $\hat{\theta}_{d,\nu}(k)$ denotes the (m_1+m_2+d)-th element of the parameter estimation $\hat{\theta}_d(k)$ and is the estimate of β_0^d. $\hat{\theta}_d(k)$ is adjusted according to the adaptive update law [108]:

$$\hat{\theta}_d(k) = \hat{\theta}_d(k-1) + \frac{a(k)\Phi_d(k-d)\nu_d(k)}{1+\Phi_d(k-d)^T\Phi_d(k-d)} \tag{2.37}$$

$$a(k) = \begin{cases} 1 & \text{if } \nu\text{-th element of right-hand side of (2.37) evaluated} \\ & \text{using } a(k)=1 \text{ is } \neq 0 \\ \gamma_d & \text{otherwise, where } 0<\gamma_d<2, \gamma_d \neq 1 \end{cases} \tag{2.38}$$

$$\nu_d(k) = y(k) - \hat{\theta}_d(k-1)^T\Phi_d(k-d), \tag{2.39}$$

with $\hat{\theta}_d(k) = \left[\hat{\vartheta}_d(k)^T \;\; \hat{\theta}_{d,\nu}(k)\right]^T$. Equation (2.38) is necessary to avoid division by zero in the control law (2.36). For the design of the adaptive controller, the error dynamics are necessary.

The stability of the overall adaptive system with the plant in (2.23) and the adaptive controller in (2.36)-(2.39) has been analyzed extensively in the adaptive control literature (for example, [108, 109]). Before the underlying result in Theorem 2.3.1 is stated, the error dynamics and an assumption are stated.

Defining the tracking error

$$e(k) := y(k) - y_{\text{ref}}(k) \tag{2.40}$$

it can be shown that

$$e(k+d) = \tilde{\vartheta}_d(k)^T \phi_d(k) + \tilde{\beta}_0^d(k) y_{\text{ref}}(k+d) \tag{2.41}$$

where $\tilde{\vartheta}_d(k) = \vartheta_d^* - \dfrac{1}{\hat{\theta}_{d,\nu}(k)}\hat{\vartheta}_d(k)$ and $\tilde{\beta}_0^d(k) = 1 - \dfrac{\beta_0^d}{\hat{\theta}_{d,\nu}(k)}$.

Assumption 2.3.1. 1) An upper bound for the orders of the polynomials in (2.18) is known and 2) all zeros of $B_i(q^{-1})$ lie strictly inside the closed unit disk.

Theorem 2.3.1. *Let $D(k) \equiv 0$. Subject to Assumption 2.3.1 and given a fixed delay d, the adaptive controller (2.36) with the update law (2.37) guarantees that*

1. *if $|e(0)| \leqslant M$ where $M > 0$ is a finite constant, then $|e(k)| \leqslant \gamma M$ for all $k > 0$ with $\gamma > 0$ a finite constant,*
2. $\lim_{k\to\infty} e(k) = 0$,
3. $\lim_{k\to\infty} \nu_d(k) = 0$, *and*
4. *the sequences $\{\hat{\theta}_d(k)\}$, $\{y(k)\}$ and $\{u(k)\}$ are bounded for all k.*

The reader is referred to Theorem 6.3.1 in [108] or Theorem 5.1 in [109] for the proof of Theorem 2.3.1.

2.3.2 Persistent Excitation and Sufficient Richness

Theorem 2.3.1 shows that the tracking error converges to zero, i.e., the output of the system gets arbitrarily close to the reference signal. However, no statement is made if the parameter estimate converges to the real parameters. The concept of persistent excitation guarantees that also the parameter error converges to zero.

The following definitions related to persistent excitation are needed to introduce a switching controller in Chapter 5. The terms *persistently exciting* and *sufficiently rich* are defined as follows:

Definition 2.3.1 ([110]). A sequence $x(t) \in \mathbb{R}^n$ is said to be *persistently exciting (PE) (in N steps)*, if there exists $N \in \mathbb{Z}^+, \alpha > 0$ such that

$$\sum_{t=t_0+1}^{t_0+N} x(t)x(t)^T \geqslant \alpha I \tag{2.42}$$

uniformly in t_0.

Definition 2.3.2 ([110]). A sequence $x(t) \in \mathbb{R}^n$ is said to be *sufficiently rich (SR) of order m (in N steps)*, if there exists $N \in \mathbb{Z}^+, \alpha > 0$ such that

$$\sum_{t=t_0+1}^{t_0+N} \xi_m(t)\xi_m(t)^T \geqslant \alpha I \tag{2.43}$$

with $\xi_m(t) = \begin{bmatrix} x(t+1) & x(t+2) & \cdots & x(t+m) \end{bmatrix}^T$ uniformly for all t_0.

The following Lemma is useful to prove Theorem 2.3.5.

Lemma 2.3.2 ([110]). *Suppose that $y_1(t)$ and $y_2(t)$ are two bounded sequences taking values in $\mathbb{R}^n$ satisfying $\sum_{t=1}^{\infty} \|y_1(t) - y_2(t)\| < \infty$. Then $y_1(t)$ is SR of order p if and only if $y_2(t)$ is SR of order p.*

The reader is referred to [110] for the proof of Theorem 2.3.5.

Lemma 2.3.3. *Consider the discrete time system*

$$X(t+1) = AX(t) + BU(t) \tag{2.44}$$

with $X(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times l}$, and $U(t) \in \mathbb{R}^l$. Assume that (2.44) is completely reachable and that the input $U(t)$ is SR of order $p, p \in [1, n]$. Then,

$$\operatorname{rank}\left(\sum_{t=t_0+1}^{t_0+n+N} X(t)X(t)^T\right) = p \tag{2.45}$$

for all $t_0 \geqslant 0$.

Proof. First, Equation (2.44) is rewritten as

$$X(t+j) = A^j X(t) + \sum_{i=1}^{j} A^{j-i} BU(t+i-1) \tag{2.46}$$

and define the characteristic polynomial of A to be

$$p(z) = z^n + a_1 + z^{n-1} + \ldots + a_n \tag{2.47}$$

and let

$$V(t) = X(t+n) + a_1 X(t+n-1) + \ldots + a_n X(t). \tag{2.48}$$

Then, from the Cayley-Hamilton theorem it follows that

$$V(t) = G\begin{bmatrix} U^T(t) & \dots & U^T(t+n-1) \end{bmatrix} \tag{2.49}$$

where $G = \begin{bmatrix} A^{n-1}B + a_1 A^{n-2}B + \ldots + a_{n-1}B, \ldots, B \end{bmatrix}$. Let

$$Y(t_0) = \begin{bmatrix} V(t_0+1) & V(t_0+2) & \dots & V(t_0+N) \end{bmatrix}. \tag{2.50}$$

Then,

$$Y(t_0)Y^T(t_0) = \sum_{t=t_0+1}^{t_0+N} V(t)V^T(t) \geqslant \alpha\gamma(I_p \oplus 0) \tag{2.51}$$

where "$\oplus$" denotes the direct sum, I_p is the $p \times p$ identity matrix, and $0 < \gamma = \sigma_{\min}(GG^T)$ is the minimal singular value of GG^T. $Y(t_0)$ can also be stated in terms of $X(t)$ in the following way

$$\begin{aligned} Y(t_0) &= \left[X(t_0+1), X(t_0+2), \ldots, X(t_0+n+N)\right] P \\ &= WP \end{aligned} \tag{2.52}$$

where $P \in \mathbb{R}^{(N+n)\times N}$ is given by

$$P = \begin{bmatrix} a_n & 0 & \cdots & 0 \\ \vdots & a_n & \ddots & \vdots \\ 1 & \vdots & \ddots & 0 \\ 0 & 1 & & a_n \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.53}$$

Then,

$$Y(t_0)Y^T(t_0) = WPP^TW^T \leqslant \sigma_{\max}(PP^T)WW^T \tag{2.54}$$

and hence using (2.51),

$$\sum_{t=t_0+1}^{t_0+n+N} X(t)X(t)^T = WW^T \geqslant \frac{\alpha\gamma}{\sigma_{\max}(PP^T)}(I_p \oplus 0) \tag{2.55}$$

That is,

$$\operatorname{rank}\left(\sum_{t=t_0+1}^{t_0+n+N} X(t)X(t)^T\right) \geqslant p. \tag{2.56}$$

From Theorem 1 in [110], it follows that an input signal which is SR of order p implies a persistent excitation of at most p directions in a n-dimensional space. Thus,

$$\operatorname{rank}\left(\sum_{t=t_0+1}^{t_0+n+N} X(t)X(t)^T\right) = p. \tag{2.57}$$

□

Remark 2.3.1. Much of the existing results pertaining to such as [110] on persistent excitation pertain to the case when the external input $U(t)$ is SR of order n. Lemma 2.3.3 above as well as Corollary 2.3.4 stated below address the case when $U(t)$ is SR of order $p, p \in [1, n]$, which has not been examined in the literature to the knowledge of the author. In Section 5.4 the goal will be tracking of an arbitrary signal and not parameter identification, it is thus not necessary for the SR-order to be n, but arbitrary and fixed at some $p \in [1, n]$.

Corollary 2.3.4. *Consider the discrete time system*

$$X(t+1) = AX(t) + BU(t) \tag{2.58}$$

with $X(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times l}$, *and* $U(t) \in \mathbb{R}^l$. *Assume that (2.58) is completely reachable and that the input* $U(t)$ *is SR of a fixed order* $p, p \in [1, n]$. *Then, there exists a subspace* $\Omega_p \subset \mathbb{R}^n$ *such that*

$$X(t) \in \Omega_p \ \forall t \geqslant t_0 \quad \textit{with } \dim \Omega_p = p. \tag{2.59}$$

That is, the columns of $X(t)$ *span the subspace* Ω_p.

Proof. This follows directly from Lemma 2.3.3 and the fact that for any complex matrix Γ the following is true

$$\operatorname{rank}(\Gamma) = \dim \operatorname{Im} \Gamma \tag{2.60}$$

where Im denotes the image of the linear transformation Γ. □

The following assumption is made which refers to an invariant property of persistent excitation.

Assumption 2.3.2. $y_{\text{ref}}(k)$ is sufficiently rich of constant order $p, p \in [1, n]$ for all k.

Theorem 2.3.5 connects the property of $y_{\text{ref}}(k)$ in Assumption 2.3.2 with boundedness and asymptotic tracking of the baseline adaptive system.

Theorem 2.3.5. *Let* $D(k) \equiv 0$. *Suppose the adaptive controller (2.36)-(2.39) is used to control the plant in (2.23) and let Assumptions 2.3.1 and 2.3.2 hold. Then*

(i) $\lim_{k\to\infty} e(k) = \lim_{k\to\infty} y(k) - y_{ref}(k) = 0$, *and*

(ii) $\Phi_d(k) \in \Omega_p \subset \mathbb{R}^n$ *as* $k \to \infty$

(iii) $\tilde{\theta}_d(k) = \hat{\theta}_d(k) - \theta_d^*$ *converges to* $\bar{\Omega}_{n-p}$ *where* $\bar{\Omega}_{n-p}$ *is defined as*

$$\bar{\Omega}_{n-p} := \left\{ x \mid \Phi_d(k)^T x = 0 \text{ for } k \to \infty \right\} \tag{2.61}$$

where $\Phi_d(k)$ *is given in (2.25).*

Proof. Item (i) follows directly from Theorem 2.3.1 as it is independent of any persistent excitation of the reference signal $y_{\text{ref}}(k)$. Item (ii) follows by noting that the adaptive system in (2.23) and (2.36)-(2.39) becomes asymptotically linear, and this linear system in turn has a state that satisfies (2.59) due to Assumption 2.3.2. Item (iii) follows from (i) and the fact that $e(k) = \Phi_d(k-d)^T \tilde{\theta}_d(k)$. □

2.4 Real-Time Systems

A *Real-Time System (RTS)* can be described as a system that has to provide an output by a certain point in time, which is called the *deadline* [42]. Examples for real-time systems can be found in many areas including control of sampled-data systems, high-level control, signal processing, databases, and multimedia applications. Common to most real-time systems is that they are not visible to the user as long as they work properly. For real-time systems it is thus essential to provide guarantees that all deadlines can be met. Usually, the timing constraints are divided into two types: hard and soft.

Hard RTS: Hard RTS are systems with hard deadlines. A deadline is *hard* if missing the deadline leads to disastrous or very serious consequences. It is indispensable to validate with deterministic methods that all deadlines can be met even under worst-case assumptions.

Soft RTS: Soft RTS are systems with soft deadlines. A deadline is *soft* if it can be missed sometimes without leading to a fatal fault. However, missing soft deadlines will degrade the performance of the system. These systems can be validated with statistical methods.

In order for the tasks in a RTS to meet their deadlines, some kind of scheduling has to be applied. Three different approaches are commonly used: clock-driven, priority-driven, and weighted round-robin. In the following, these three scheduling approaches will be described briefly.

Clock-Driven Approach: In a *clock-driven* approach, the decision which task will be executed next is made at specific time instants. These time instants, and thus the schedule, can be computed off-line and stored in the system in order to minimize the computational overhead. Since the scheduling is done off-line, the schedule can be optimized such that all hard real-time tasks meet their deadlines. This approach is also called *time-triggered approach*.

Priority-Driven Approach: The *priority-driven approach*, also called *event-triggered approach*, computes the schedule on-line. It assigns priorities to each task and executes at each scheduling decision time the task with the highest priority. This approach minimizes the idle time of a system since it executes a task whenever at least one task is ready for execution. Compared to a clock-driven schedule, a priority-driven schedule does not require all information (e.g. release time and execution time) a priori and is thus suited for tasks with varying time and resource requirements. Priority-driven schedules are typically not used for hard real-time tasks since the timing behavior becomes non-deterministic when job parameters can vary [42].

Weighted Round-Robin: The *Weighted Round-Robin* approach is an extension of the round-robin approach. In the round-robin approach each task is executed for the same amount of time. If it cannot be completed within this time slot, it is preempted, moved to end of all tasks that are ready for execution, and the next task is executed. Instead of giving each task the same amount of processor time, the weighted round-robin algorithm assigns different fractions of processor time to different tasks.

The above paragraphs describe the situation of scheduling tasks on processors. If the tasks are messages that are transmitted over some network, the algorithms are different from the ones described in this section. Some protocols used for communication over shared networks are described in the next section.

2.4.1 Network Protocols

Naturally, communication networks can be categorized into two groups: *wired* and *wireless* networks. Wired networks possess many advantages like high reliability, high bandwidth, and high security. On the other hand, the need for wires decreases their flexibility and mobility. Wireless networks, however, show a remarkable flexibility and mobility. The limited bandwidth and reduced reliability are the disadvantages of wireless networks. Common for both, wired and wireless networks, is the need of a network protocol depending on the field of application.

Networks can be grouped into two areas. *Data networks* are designed for communication networks where large data packets have to be sent with high throughput. On the other hand, *control networks* are designed to transmit a large amount of small packets at a high frequency between a large amount of nodes and meet the real-time constraints. The main difference between data networks and control networks is the capability of control networks to meet real-time constraints [111].

Network protocols are typically organized in different layers. Each layer has its own specifications and the interface to the layer above and below is carefully designed. With this, developers can focus on the design of one layer and still the overall network keeps working because of the fixed interface definitions between the individual layers. The most popular layer model is the Open Systems Interconnection (OSI) model from the International Organization for Standardization (ISO) [112]. It describes the network in seven layers (see Figure 2.3). The Media Access Control (MAC) protocol, according to the OSI-Reference model part of the Data Link layer, plays an important role in determining the characteristics of delays and packet dropouts. Since only one node can send a message over the shared network at any given time and as several nodes are connected to the network, access to the network has to be controlled; this is done by the MAC protocol. It is therefore important to also consider the MAC protocol in the analysis of a NCS. Common examples of the MAC protocols are the *Time Division Multiple Access* (TDMA) protocol , the *Carrier Sense Multiple Access*

(CSMA) protocol, and the *Token passing* protocol. In the following, these three MAC protocols are described.

<table>
<tr><td></td><td>Data</td><td>Layer</td></tr>
<tr><td rowspan="5">Host Layers</td><td rowspan="3">Data</td><td>7. Application</td></tr>
<tr><td>6. Presentation</td></tr>
<tr><td>5. Session</td></tr>
<tr><td>Segment</td><td>4. Transport</td></tr>
<tr><td>Packet</td><td>3. Network</td></tr>
<tr><td rowspan="2">Media Layers</td><td>Frame</td><td>2. Data Link</td></tr>
<tr><td>Bit</td><td>1. Physical</td></tr>
</table>

Figure 2.3: The OSI Layer model of network protocols

TDMA: The TDMA protocol assigns each node one or more time slots, in which only this node is allowed to send messages. This so called schedule has to be fixed before operation and cannot be changed during operation. Furthermore, all nodes need to know the schedule. This MAC protocol can only be used if all nodes use the same time basis. Examples where the TDMA protocol is used are FlexRay (see also Section 2.4.2), TTP [113], and the Time-triggered CAN (TT-CAN) [114].

Token-passing: All nodes in a token-passing network are arranged into a logical ring and can transmit messages over the network only if they hold the token. Only one node holds the token at any time. If a node holds the token, it can send messages for a maximum amount of time, or if it runs out of messages it can pass the token to its logical successor. The logical ring structure does not have to be identical to a physical ring structure. The token bus protocol is a deterministic protocol with excellent throughput and is especially efficient at high network load. It has also the advantage that nodes can be dynamically added and removed. Furthermore, the protocol guarantees a maximum time between network access for each node. Typical examples of token-passing networks are PROFIBUS and ControlNet [115].

CSMA: The CSMA protocol allows each node to send messages over the network whenever the network is available. As this can lead to collisions, two techniques are used to deal with collisions. These are CSMA/CD (CSMA with collision detection) and CSMA/CA (CSMA with collision avoidance).

Before a node starts transmitting a message under the CSMA/CD protocol, it listens to the network until it is idle. Once this happens, the node starts transmitting immediately. If two or more nodes decide to send messages simultaneously, the collision is detected by all nodes and all nodes stop transmitting and wait for a random length of time before trying to retransmit. The most well-known example where the CSMA/CD protocol is used, is Ethernet. At low network loads, the Ethernet protocol shows almost no communication delay. However, due to the non-deterministic nature of the CSMA/CD protocol, the collisions may lead to large time-delays at high network loads. Nevertheless, Ethernet is very popular because it is cost-effective and has high bandwidth.

In contrast to the CSMA/CD protocol, where the network reacts after collisions occurred, the CSMA/CA protocol prevents collisions a priori by a priority based arbitration scheme. To this end, each message is assigned a priority. Similar to the CSMA/CD protocol, nodes listen to the network until it is idle. Once the network is idle, the nodes start to transmit their priority bitwise. Higher bits overwrite lower bits and the other nodes know that their message has a lower priority and stop sending. The node with the highest priority message is granted access to the network. The CAN protocol, which is widely used in the automobile industry, uses the CSMA/CA protocol. Compared to Ethernet, CAN is a deterministic protocol and higher priority messages have a guaranteed transmission delay. Although the CAN network supports only a relatively low data rate (maximum of 500 Kbps), it can be found in almost all modern cars.

For wireless networks, the IEEE 802.11 standard, commonly referred to as *Wireless Ethernet*, is one of the most common protocols. Similar to wired Ethernet, wireless Ethernet utilizes a "listen before talk" mechanism to control access to the shared medium. A wireless network protocol, however, has to deal with challenges that are inherently to the wireless nature of the shared medium. These challenges are interferences and the so-called "hidden client" problem, i.e., a client which is in range of one node, but out of range of another node. Also, while a node is sending a message, it cannot reliable monitor the idle/busy state of the medium [116]. Nevertheless, wireless Ethernet provides a robust and secure communication. Similar to the wired Ethernet, wireless Ethernet uses the CSMA/CA protocol. However, collisions cannot be fully avoided. Thus, after a packet has been sent, the sender waits for an acknowledgement packet. In case the sender does not receive an acknowledgement packet, it assumes that there was a collision and retransmits the packet. Before a node starts sending a packet, it waits for a randomly chosen amount of time if the network is busy. If another node starts transmitting during that time, the node has to wait for another idle time. Collisions may still occur due to random nature of the waiting time. This results in non-deterministic transmission delays.

Hierarchical Schedules

Since most schedulers have advantages in one field, it is difficult to find the scheduler that is best suited for a general purpose system. A solution to this problem is to arrange different schedulers in a hierarchy such that for each application the appropriate scheduler is available. The top-level arbitration is done by a *root scheduler*, the individual applications or messages are then scheduled by *leaf schedulers*. In Figure 4.5, an example for a hierarchical TDMA/FP scheduler is shown. Each control application is allocated a TDMA slot, and message streams from the same controller are scheduled using fixed priority within the allocated TDMA slot.

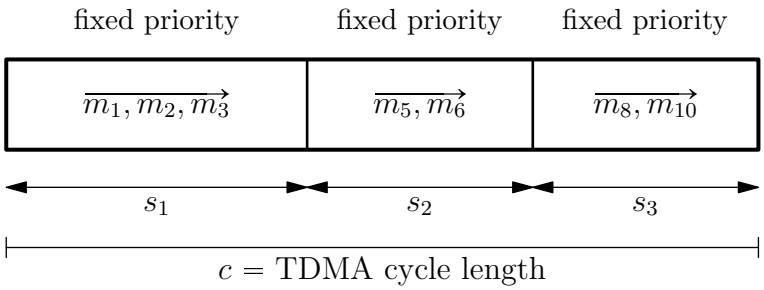


Figure 2.4: Hierarchical TDMA/FP Scheduler

The advantages of hierarchical schedule arrangements have been well studied in the literature [117, 118] and may be summarized as follows:

- Each application in the schedule hierarchy has its own scheduler and can be executed according to different scheduling algorithms. Consequently, each application may exploit the advantages of the scheduling policy which best meets the application's requirements.
- The integration of several applications on a single resource becomes easier as each subsystem may be treated independently and is unaffected of changes to parameters such as periods or priorities during the integration phase.
- Hierarchical schedules provide temporal isolation among several applications that are executed on a single resource. That is, each subsystem may run in an isolated environment and errors caused by one of the subsystems cannot propagate and interfere with other subsystems.

2.4.2 The FlexRay Protocol

In order to illustrate the concepts in this dissertation, the FlexRay protocol is described as it is considered to become the de-facto standard for future automotive in-vehicle networks.

FlexRay is a deterministic, fault-tolerant and high-speed automotive bus system. The FlexRay Consortium started to develop the FlexRay protocol in 2000 after several

leading automobile manufacturers and suppliers identified the need of a new communication protocol for automotive applications. The goals were twofold. Economic goals for the development of FlexRay were to reduce the overall costs by making the protocol specifications license free and to establish the FlexRay protocol as the standard protocol for future cars. Technical goals were a high data rate of 10 Mbit/s, redundant communication channels, a globally synchronized time basis, guaranteed transmission of messages, flexibility in adding and removing nodes, time- and event-triggered communication, and fault-tolerance. FlexRay covers the Physical Layer (OSI-Layer 1) [119] and the Data Link Layer (OSI-Layer 2) [120] (see Figure 2.3).

The most important properties of FlexRay are the following:

Data rate of 10 Mbit/s: The theoretically achievable maximum data rate of 10 Mbit/s for each of the two communication channels depends on the configuration of the bus.

Synchronized Time basis: A highly accurate time basis is the base for the arbitration on the bus. The time basis is autonomously established by the FlexRay protocol.

High degree of determinism Since the FlexRay protocol uses periodic cycles as basis for communication, upper bounds on the arrival time of messages can be guaranteed.

Scalable redundancy: It is possible to transmit some messages multiple times in order to increase redundancy. Though it is not necessary to transmit all messages multiple times.

Flexibility and scalability: The FlexRay protocol is designed to be highly flexible and scalable. Adding or removing nodes does not require any changes in the existing nodes.

Basic Functionalities of the FlexRay protocol

The basic functionalities of the FlexRay protocol are described in this section. Further details on the FlexRay specifications can be found in [121] or [120].

The FlexRay protocol supports bus (see Figure 2.5) and star topologies (see Figure 2.6). Also combinations are allowed. FlexRay can handle up to two channels, which do not necessarily have to have the same physical topology (see Figure 2.7). This makes FlexRay a very flexible protocol.

The FlexRay protocol is organized in a sequence of communication cycles of fixed length. Further, every such cycle is subdivided into a *static* segment (ST), a *dynamic* segment (DYN), the *symbol window*, and the *network idle time* (NIT). Only the static segment and the NIT are compulsory. The symbol window and the NIT are not further discussed here as they are not relevant for the transmission of control related messages. The next cycle starts immediately after the end of the previous cycle. The length of

each cycle, T, can be set in the design process. Each cycle has a cycle number which is incremented from 0 to 63. After that, the cycle counter is reset to 0.

Since FlexRay offers as well time-triggered communication in the static segment as well as event-triggered communication in the dynamic segment, FlexRay belongs to the group of *hybrid protocols*. Another example for a hybrid protocol is TTCAN [114], which is an extension of CAN capable of sending hard real-time messages in a time-triggered way. In general one can say that time-triggered communication offers highly predictable temporal behavior, and event-triggered communication provides efficient bandwidth usage.

Access to the static segment is organized with a TDMA scheme. The static segment is partitioned into time windows of fixed and equal length which are referred as *slots*. Each processing unit is assigned one or more slots indexed by a slot number $S \in \mathcal{S}_{\mathrm{ST}}$ that indicates available time windows for bus access in the static segment.

The dynamic segment is partitioned into *minislots* of much smaller duration than the static slots. Similar to the static segment, the minislots are indexed by a slot number to indicate allowable message transmissions. However, dynamic slots are of varying size depending on the size of the message which is transmitted in a certain slot $S \in \mathcal{S}_{\mathrm{DYN}}$, where $\mathcal{S}_{\mathrm{DYN}}$ is the set of available slot numbers in the dynamic segment. If no message is ready for transmission in a particular slot only one small minislot is consumed and the slot number is incremented to the next minislot. However, if a message is transmitted in a slot $S \in \mathcal{S}_{\mathrm{DYN}}$ then the slot number increments with the next minislot after which the message transmission has been completed. Hence, bus resources are only utilized if messages are actually transmitted on the bus; otherwise only one minislot is consumed. If many nodes send messages in the dynamic segment, a message with a low priority might experience a very large delay or might even be not sent in the current cycle.

The FlexRay protocol does not require that every message is sent in all cycles. For each message a *base cycle* and a *cycle repetition rate* have to be specified. The base cycle length takes values from 0 to 63 and indicates the offset of the message within the 64 cycles. The cycle repetition rate specifies the number of cycles that must elapse between two consecutive allowable transmissions. This offers the possibility that one slot is used to transmit one message in every odd cycle, whereas the same slot is used for another message in every even cycle.

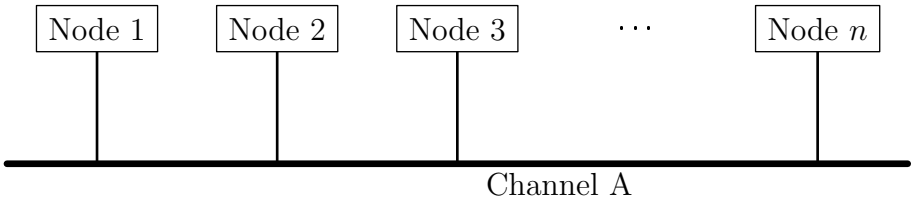


Figure 2.5: Bus topology with one channel

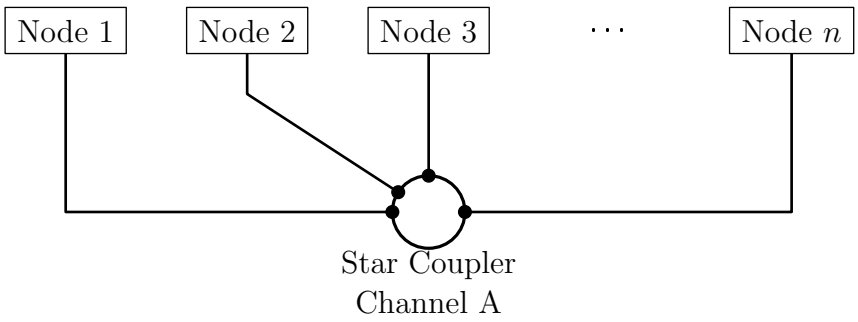


Figure 2.6: Star topology with one channel

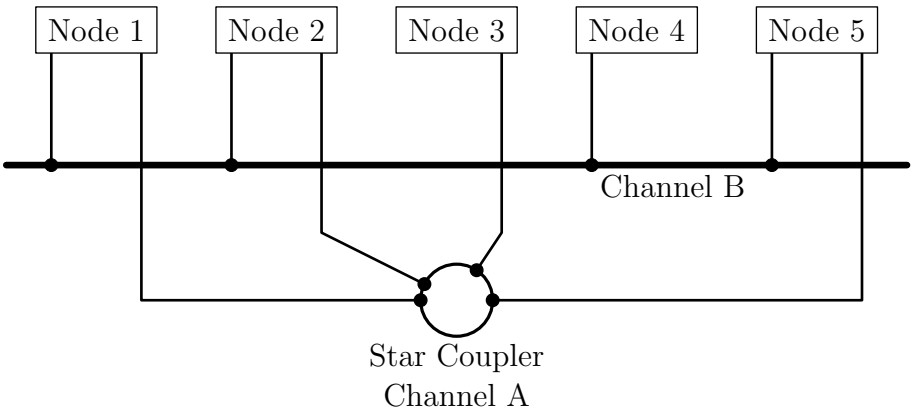


Figure 2.7: Mixed Bus-Star topology with two channels

2.4.3 Real-time Calculus

The *Real-time calculus* (RTC) framework [122, 123] is an analytical method that uses tools from three different areas to provide a unified framework for analyzing performance properties of distributed embedded real-time systems. It belongs to the class of so-called deterministic queuing theories that always gives hard upper and lower bounds on the performance indicator, in contrary to probabilistic queuing theories that provide information on the average performance of a system. The RTC framework links the Max-Plus Linear System Theory [124–126], Network Calculus [125, 127–129], and real-time scheduling. In the following, the basic concepts of the RTC framework are discussed.

Max-Plus and Min-Plus Algebra

Max-plus and min-plus algebra are special algebras that are associative, commutative, and distributive. Max-plus and min-plus algebras are used in various applications such as optimization, algebraic geometry, control theory, machine scheduling, discrete-event systems, telecommunication systems, and traffic control [126, 130–132]. Max-plus algebra has the nice property that many non-linear descriptions of these applications become linear in max-plus algebra. In the following max-plus and min-plus algebra will be defined and some of their properties will be described briefly.

Before max-plus and min-plus algebras will be defined, a *dioid* needs to be defined.

Definition 2.4.1. A *dioid* $(D, \boxplus, \boxtimes)$ is a set D endowed with two closed operations called *addition*, denoted by $\boxplus$, and *multiplication*, denoted by $\boxtimes$. The operations of a dioid must satisfy the following axioms $\forall a, b, c \in D$:

Commutativity of addition: $a \boxplus b = b \boxplus a$

Associativity of addition: $(a \boxplus b) \boxplus c = a \boxplus (b \boxplus c)$

Associativity of multiplication: $(a \boxtimes b) \boxtimes c = a \boxtimes (b \boxtimes c)$

Distributivity of multiplication over finite sums: $(a \boxplus b) \boxtimes c = (a \boxtimes c) \boxplus (b \boxtimes c)$ and $c \boxtimes (a \boxplus b) = (c \boxtimes a) \boxplus (c \boxtimes b)$

Existence of a zero element for addition: $\exists e \in D : e \boxplus a = a \boxplus e = a$

Absorption by the zero element $e \boxtimes a = a \boxtimes e = e$

Existence of unity for multiplication: $\exists \varepsilon \in D : \varepsilon \boxtimes a = a \boxtimes \varepsilon = a$

Idempotency of addition: $a \boxplus a = a$

If the operation $\boxtimes$ is commutative, i.e. $\forall a, b \in D : a \boxtimes b = b \boxtimes a$, then $(D, \boxplus, \boxtimes)$ is called a *commutative dioid*.

Note that the above axioms are almost the same as for a *ring*. The only difference is that the existence of a symmetric element $(-a)$ with $a \boxplus (-a) = e$ is replaced by the idempotency of addition. Another name for dioid is thus *idempotent semi-ring*.

In classical algebra the system of interest is usually the algebraic structure $(\mathbb{R}, +, \times)$. A min-plus algebra can now be defined by replacing $+$ with inf (or min if it exists) and $\times$ with $+$, and including $+\infty$ in the base set.

Definition 2.4.2. A *min-plus algebra* is an algebraic structure over the set $\mathbb{R}_{\min} = \mathbb{R} \cup \{+\infty\}$ with the addition operations $\boxplus$ defined as

$$x \boxplus y = \min(x, y) \tag{2.62}$$

and the multiplication operation $\boxtimes$ defined as

$$x \boxtimes y = x + y. \tag{2.63}$$

The identity element for the addition is $+\infty$, and the identity element for the multiplication is 0.

It is easy to show that $(\mathbb{R}_{\min}, \boxplus, \boxtimes)$ is a dioid. Similarly, a max-plus algebra can be defined.

Definition 2.4.3. A max-plus algebra is an algebraic structure over the set $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$ with the addition operation $\boxslash$ defined as

$$x \boxslash y = \max(x, y) \tag{2.64}$$

and the multiplication operation $\boxtimes$ defined as

$$x \boxtimes y = x + y. \tag{2.65}$$

The identity element for the addition is $-\infty$, and the identity element for the multiplication is 0.

Again, one can easily show that $(\mathbb{R}_{\max}, \boxslash, \boxtimes)$ is a dioid.

Diods cannot only be based on the set of real numbers $\mathbb{R}$, but also on function sets. In Network Calculus and RTC the following set of functions is of special interest:

Definition 2.4.4 (Wide-sense increasing function and $\mathcal{F}$)**.** A function $f : \bar{\mathbb{R}} \to \mathbb{R}^+$, with $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ and $\mathbb{R}^+ = [0, +\infty]$, is *wide-sense increasing* if and only if $f(s) \leqslant f(t)$ for all $s \leqslant t$.

The set $\mathcal{F}$ is defined as

$$\mathcal{F} := \{f : \bar{\mathbb{R}} \to \mathbb{R}^+ \mid f \text{ is wide-sense increasing and } f(t) = 0 \text{ for } t < 0\} \tag{2.66}$$

Examples for elements of $\mathcal{F}$ are the peak rate function, the burst delay function, affine functions, step functions, or staircase functions [129]. It is not difficult to see that $(\mathcal{F}, \min, +)$ and $(\mathcal{F}, \max, +)$ are diods. Important operations in RTC are the convolution and deconvolution of elements of $\mathcal{F}$. For min-plus calculus, convolution and deconvolution are defined as follows:

Definition 2.4.5 (min-plus convolution and deconvolution)**.** Let $f, g \in \mathcal{F}$. The *min-plus convolution* denoted by $\otimes$ is the function

$$\big(f \otimes g\big)(t) = \inf\big\{f(t-s) + g(s) \;\mid\; 0 \leqslant s \leqslant t\big\}. \tag{2.67}$$

The *min-plus deconvolution* denoted by $\oslash$ is the function

$$\big(f \oslash g\big)(t) = \sup\big\{f(t+u) - g(u) \mid u \geqslant 0\big\}. \tag{2.68}$$

The above definitions of convolution are the natural extensions of the well-known operation of convolution in the conventional algebra $(\mathbb{R}, +\times)$ with "addition" replaced by "inf" and "multiplication" replaced by "+". A similar definition can be made for max-plus calculus.

Definition 2.4.6 (max-plus convolution and deconvolution)**.** Let $f, g \in \mathcal{F}$. The *max-plus convolution* denoted by $\overline{\otimes}$ is the function

$$\big(f \overline{\otimes} g\big)(t) = \sup\big\{f(s) + g(t-s) \;\mid\; 0 \leqslant s \leqslant t\big\}. \tag{2.69}$$

The *max-plus deconvolution* denoted by $\overline{\oslash}$ is the function

$$\big(f \overline{\oslash} g\big)(t) = \inf\big\{f(t+u) - g(u) \;\mid\; u \geqslant 0\big\}. \tag{2.70}$$

Timing Analysis

The basic idea of RTC is to model event streams, such as message streams or task activations, as well as processing resources, such as bus or processor capacities, in a count-based abstraction. In the following, the abstract system representation (depicted in Fig. 2.8) is used to illustrate the applicability of the timing analysis framework. The system consists of two processing units PU1 and PU2 and a shared communication medium. The processing units execute the control tasks T_1, T_2 and T_3 and transmit the processed output over the shared bus.

Given several distributed applications that are mapped onto different processing and communication resources, e.g., in the architecture depicted in Fig. 4.1, the goal is to compute the maximum end-to-end delay experienced by each of the *control applications*.

The input of a task can be represented by an arrival pattern $A(t)$ of a data stream, i.e., sensor readings. The cumulative function $A(t)$ denotes the total number of events that arrive during the time interval $(0, t]$. However, the number of events that may arrive in *any* time interval of length Δ that trigger a task execution or a message transmission can be upper- and lower-bounded by the pair of *arrival curves* $\alpha = (\alpha^u, \alpha^l)$. Formally, the following mathematical inequality are obtained:

$$\forall \Delta \geqslant 0,\ \forall\, t \geqslant 0 : \alpha^l(\Delta) \leqslant A(\Delta + t) - A(t) \leqslant \alpha^u(\Delta). \tag{2.71}$$

In other words, $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$ denote the maximum and minimum number of events that can arrive within *any* time interval of length Δ. Thus, using this event

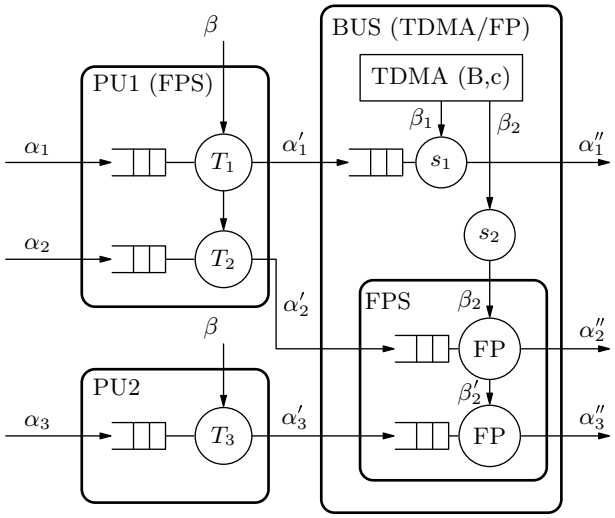


Figure 2.8: System With Hierarchical Bus Scheduling

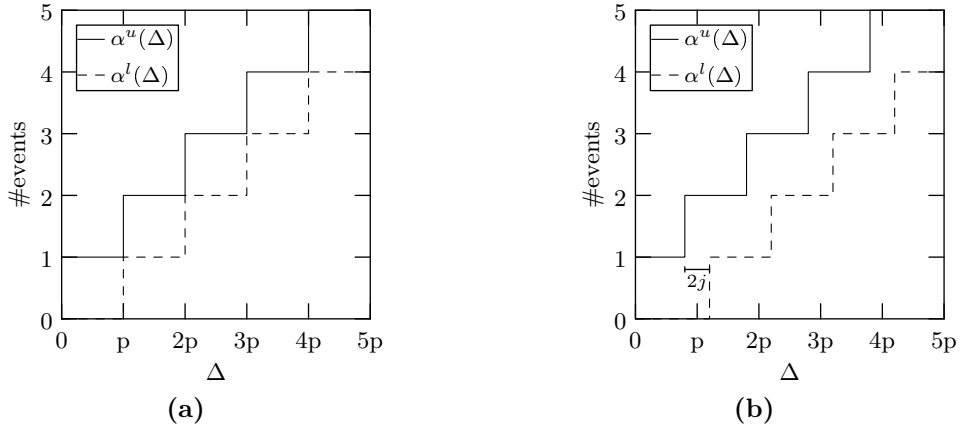


Figure 2.9: (a) Periodic Arrival Curves (b) Periodic with jitter j

model, the timing properties of standard event models - like *periodic, periodic with jitter* and *sporadic* - as well as arbitrary arrival patterns can be represented by an appropriate choice of $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$. Fig. 2.9 (a) illustrates an example for the upper and lower arrival curves representing a strictly periodic event stream with period p. Fig. 2.9 (b) depicts a pair of arrival curves with period p and jitter j, i.e., in a periodic sensor stream with jitter the sensor samples arrive at an average time interval of p time units, but can deviate from the ideal periodic arrival.

Similarly, the available resource capacities on a processor or a bus to process a task or transmit a message can be modeled. The cumulative function $C(t)$ captures the number of events that can be processed in $(0, t]$. The service available to a task or message is upper- and lower-bounded by the pair of *service curves* $\beta = (\beta^u, \beta^l)$. Thus, the following mathematical inequality hold true:

$$\forall \Delta \geqslant 0,\ \forall\, t \geqslant 0 : \beta^l(\Delta) \leqslant C(\Delta + t) - C(t) \leqslant \beta^u(\Delta) \tag{2.72}$$

Hence, $\beta^u(\Delta)$ and $\beta^l(\Delta)$ denote the maximum and minimum number of events that can be processed within *any* time interval of length Δ. The service curves can also be expressed in terms of the maximum and minimum number of resource units (e.g., processor cycles, execution times) that are required to process an event within any Δ. For this purpose, $\beta^u(\Delta)$ and $\beta^l(\Delta)$ are scaled with the execution requirement demanded by a task or message due to each activation. When running several tasks on a shared resource, e.g., T_1 and T_2 on $PU1$, the bounds on the service available to the tasks using this resource depend on the scheduling policy being used. On $PU1$ the tasks are executed according to FPS. Hence, the full service β is available to the highest priority task T_1 to process the input stream α_1. Consequently, the remaining service β' after processing α_1 serves as an input for the execution of T_2 processing α_2. Further, the processed data streams $\alpha' = (\alpha^{u'}, \alpha^{l'})$ and the remaining service $\beta' = (\beta^{u'}, \beta^{l'})$ can be computed by the following equations:

$$\alpha^{u'} = \min\Big\{ \big(\alpha^u \otimes \beta^u\big) \oslash \beta^l,\ \beta^u \Big\} \tag{2.73}$$

$$\alpha^{l'} = \min\Big\{ \big(\alpha^l \oslash \beta^u\big) \otimes \beta^l,\ \beta^l \Big\} \tag{2.74}$$

$$\beta^{u'} = \big(\beta^u - \alpha^l\big)\ \overline{\oslash}\ 0 \tag{2.75}$$

$$\beta^{l'} = \big(\beta^l - \alpha^u\big)\ \overline{\otimes}\ 0 \tag{2.76}$$

with the convolution and deconvolution operators as defined in (2.67)-(2.70).

Given $\alpha^u(\Delta)$ and $\beta^l(\Delta)$ the maximum backlog b at the input buffer, and the maximum delay d that is experienced by the event stream α can be computed as follows:

$$b = \sup\big\{\alpha^u(\Delta) - \beta^l(\Delta)\ \mid\ \Delta \geqslant 0\big\} \tag{2.77}$$

$$d = \sup\big\{\inf\big\{\tau \geqslant 0 \mid \alpha^u(s) \leqslant \beta^l(s + \tau)\big\} \mid s \geqslant 0\big\} \tag{2.78}$$

Note that (2.77) and (2.78) denote the maximum vertical and horizontal deviations between $\alpha^u(\Delta)$ and $\beta^l(\Delta)$.

The processed data streams α_1', α_2' and α_3' trigger messages to be transmitted on the shared bus according to a *hierarchical* TDMA/FP scheduling policy. The top-level scheduler is modeled as a TDMA resource with a total bandwidth of B and a cycle length of c. Within every TDMA cycle the time slot s_i is assigned to one *control application*. In Fig. 2.8, slot s_1 is assigned to the data stream α_1' whereas slot s_2 is assigned to α_2' and α_3'. The service bounds for a TDMA resource can be modeled as follows:

$$\beta_i^l = B \max\left\{ \left\lfloor \frac{\Delta}{c} \right\rfloor s_i, \Delta - \left\lceil \frac{\Delta}{c} \right\rceil (c - s_i) \right\} \tag{2.79}$$

$$\beta_i^u = B \min\left\{ \left\lceil \frac{\Delta}{c} \right\rceil s_i, \Delta - \left\lfloor \frac{\Delta}{c} \right\rfloor (c - s_i) \right\} \tag{2.80}$$

During the time interval $\Delta = (c - s_i)$ no service is guaranteed to any data stream that is assigned to any slot s_i. According to the *hierarchical* scheduling policy the service available to the data streams assigned to any time slot s_i is determined according to the FPS. The full service β_1 is available for transmitting the data stream α_1'. The input streams α_2' and α_3' share slot s_2 providing the service β_2 where α_2' is assigned a higher priority than α_3'. Thus, the full service β_2 is available to α_2' while the remaining service β_2' is available for transmitting α_3' on the bus. Now, the output event streams α_1'', α_2'' and α_3'' can serve again as an input to further processing units, e.g., executing actuator tasks.

Considering the system in Fig. 2.8, the end-to-end delay can be computed as the sum of the individual delays at the different processing components (due to task processing and bus communication) experienced by an input data stream, e.g., the data stream α_1 experiences a delay d_1 due to task processing of T_1 and another delay d_2 due to the message transmission on the bus. Thus, using (2.78), the total delay experienced by α_1 is computed as $\tau = d_1 + d_2$. Hence, using the compositional performance model presented in this section, it is possible to compute the maximum end-to-end delay τ experienced by any message along the path from the sensor to the actuator, of any distributed *control application*. For example, in Fig. 4.1 the worst case end-to-end delay experienced by *control application 3* is computed by $\tau = d_{T_8} + d_{m_8} + d_{T_9} + d_{T_{10}} + d_{m_{10}} + d_{T_{11}}$.

2.5 Summary

In this chapter, the basic concepts and ideas for this dissertation were introduced. First, a short introduction to feedback control systems and some representations for control systems were given. This was done with continuous-time models. In the next section, the transition to discrete-time systems was made. To this end, the concept of sampling was introduced and a first controller for discrete-time systems was presented. In order to be more realistic, a disturbance model was introduced. Since in practice, the model of a control system always contains parametric uncertainties or the plant

itself changes over time, the concept of adaptive control and a basic adaptive controller were introduced. In the next section, a novel approach to persistent excitation is presented. The chapter is concluded by an introduction to real-time systems. Within this, several network models are discussed, with special emphasis on FlexRay since it is often used in later sections and its practical significance is steadily increasing. Finally, a framework to analyze the timing properties of arbitrary real-time systems, the Real-Time Calculus, is presented.

3 Arbitrated Networked Control Systems

Summary: *In this chapter, Cyber-Physical Systems are introduced. The term Cyber-Physical Systems refers to a relatively new idea and describes the need for integrated research in control and communication. Furthermore, the Arbitrated Networked Control System Approach to Cyber-Physical Systems is presented to describe a special class of Cyber-Physical Systems where the emphasis is on the arbitration in the communication network. Some solutions for problems arising within Cyber-Physical Systems will be shown.*

3.1 Cyber-Physical Systems

The term *Cyber-Physical Systems (CPS)* (see Figure 3.1) refers to a relatively new idea [133–135]. On the one hand, the systems and control domain has focused on designing and developing methods like state-space models, system-identification, filtering, and robust control in order to analyze physically motivated, yet abstract models of real processes. On the other hand, the computer science domain has developed powerful techniques for real-time computing, programming languages and compilers, embedded systems, and cyber security. These methods use very mighty modeling formalisms and verification tools. Although both areas have a long history and play an essential role in almost all technical devices, there is no common theoretical foundation. The research in CPS aims to change this.

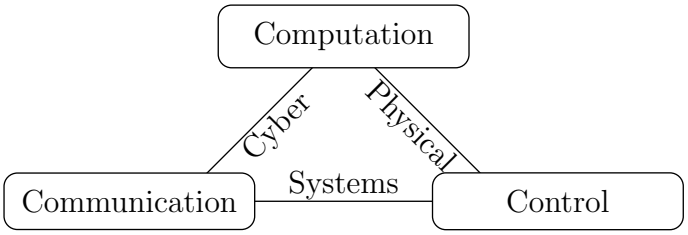


Figure 3.1: Cyber-Physical Systems (CPS)

Typically, the design process of an embedded control system is decoupled. First, the design, analysis, simulation, and verification of the control system is done by control engineers with limited knowledge of the hardware the control system will be implemented on. Once all requirements regarding control performance, robustness,

and disturbance rejection are met, the control system is implemented on a piece of hardware with periodic deadlines. This is done by real-time engineers who have only a limited knowledge of the control techniques and requirements. The effort for combining and verifying several – in modern cars hundreds of control loops are implemented – individually designed control loops in one embedded platform is very time consuming and extremely expensive.

Not only small scale systems like cars or airplanes can be viewed as CPS, but also a large scale system like the power grid. The problem here is that the controller and the plant are physically distributed. Hence there is a need for communication, which can only be carried out over the Internet. The challenge is to understand which part of a control system can be located far away from the generator as this information can be reliable transmitted over the Internet, and which part has to be close to the generator since it is essential for its proper operation.

The above shows that CPS are present on all scales and their importance will grow in the future. The following shows were research in CPS is necessary (see also [133, 136]).

Abstraction and Modeling: Models and abstractions of the architectures and control systems are needed that allow a holistic description of control, communication and computation. Once these models are available, the design and deployment of CPS will be simplified and accelerated, leading to decreasing costs.

Distributed Computation and Networked Control: The operation of CPS in a complex environment over many spatial and temporal scales poses challenges like time- and event-triggered communication, reconfiguration, adaptability, variable delays, and bridging the gap between continuous and discrete-time systems. Again, frameworks and models are needed to describe this heterogeneous setup of cooperating components.

Verification and Validation: Hardware, software and related tools and algorithms need to be developed that are highly dependable, certifiable, and reconfigurable such that verification and validation of the overall system is possible at design stage.

In the next sections, some solutions to problems arising in CPS will be shown.

3.2 The Arbitrated Networked Control Systems Approach to CPS

The domain of NCS has traditionally been concerned with modeling and designing distributed controllers in the presence of control message loss, varying delay and jitter. Here, the characteristics of the network are assumed to be given and the focus has largely been on the controller. While this is meaningful in the context of control over wireless networks, where the network is assumed to be predesigned, in many

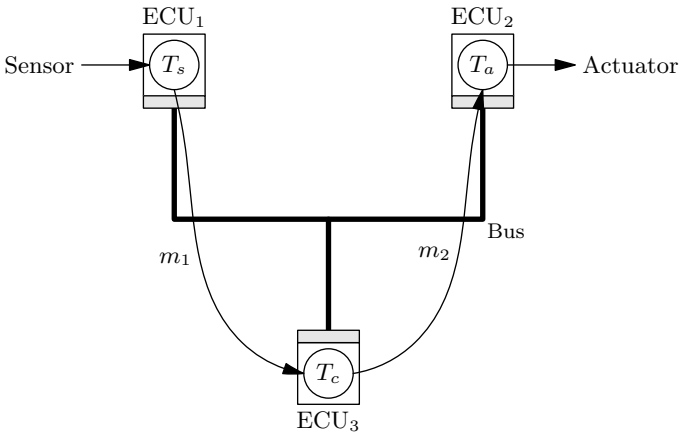
other domains there is considerable flexibility in designing the network itself. For example, in an automotive architecture, distributed controllers are implemented on multiple electronic control units (ECUs) that communicate over CAN or FlexRay buses. The network design problem in this case consists of mapping control tasks to ECUs, and determining both the scheduling parameters for the ECUs as well as those for the communication buses. Given a choice of these parameters, the controller design problem is similar to that studied in NCS, i.e., that of designing/analyzing the controller. However, the joint design and optimization of the network and the controller gives rise to new research questions, that have neither been studied within the NCS, nor within the embedded systems domain. From the previous section it is clear that these systems belong to the class of CPS. Since the systems described above represent only a subset of all CPS, they are referred to as *Arbitrated Networked Control Systems (ANCS)* to emphasize the fact that the arbitration policy in the network also needs to be determined. In the following sections, the properties of an ANCS are discussed.

3.3 Characteristics

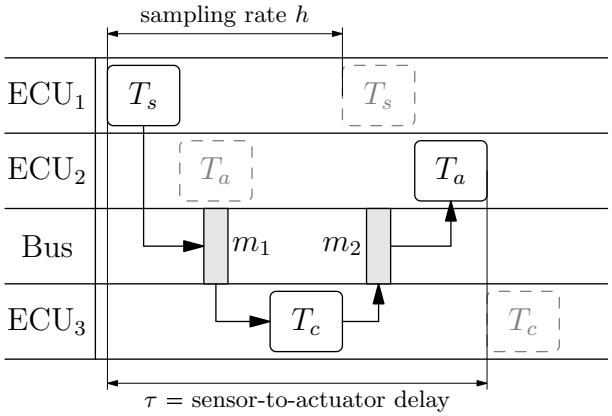
In order to illustrate an ANCS, a typical controller implementation in an automotive architecture is considered in Figure 3.2, where an electronic control unit (ECU) collects sensor data (denoted as a task T_s). A communication bus (e.g., FlexRay or CAN) then transmits the data as message m_1 to a second ECU (marked as ECU_3), where the resident control algorithm is implemented (denoted as task T_c). The output of the controller is then sent as a message m_2 to the actuator in ECU_2, which activates the actuator task T_a. Here, the sampling period is h. The total end-to-end delay between the time the signal is measured and the time the actuator inputs the computed signal from the controller into the plant is indicated by τ (see Figure 3.2(b)). The question is whether the implementation structure can be utilized to obtain prior information about τ and suitably exploit it in a control design. In what follows, two different examples are considered where such information is indeed available.

3.3.1 Prior Information on Delays

As illustrated above, the process of implementing multiple control applications on a DES typically involves breaking the control action into individual tasks, mapping these tasks to processors, and scheduling all relevant tasks and messages on the processors and the communication bus. The processing and communication inevitably introduce delays, and in the presence of uncertainties in the availability of resources introduce uncertainties in these delays. A typical networked control systems approach is to model this problem as one with a delay and represent the uncertainties either using a stochastic or a deterministic structure with upper bounds. The current problem clearly falls in the same category, and as such, one could pose the problem of



(a) Task allocation



(b) Time line

Figure 3.2: A typical distributed embedded system (DES)

designing the controller in a DES as a time-delay control problem with the network effects represented as an uncertain delay. It should be noted however, that there is indeed significant information available about the DES in terms of its overall structure and from the details of its parameters and protocols, which is what is proposed to be utilized in an ANCS-based design approach. Even more importantly, and as already mentioned, there is flexibility in designing this structure via a selection of a specific topology with desired scheduling protocols and suitable parameters. A specific example of this transparency and flexibility is included below.

The behavior of a simple networked control system is considered, as shown in Figure 3.3. This case corresponds to a scenario where two plants communicate with their controllers via a single communication bus. This bus therefore has to arbitrate or schedule messages sent by both plants to their controllers and vice versa. Here, when Plant 1 is sampled and the sensor measurement is sent through the network, Controller 1 can begin its computation (Slot 1). However, the schedule does not grant Controller 1 access to the network in Slot 2 when its message is ready for transmission. Therefore, Controller 1 has to wait until Slot 3. This results in a known amount of delay – which

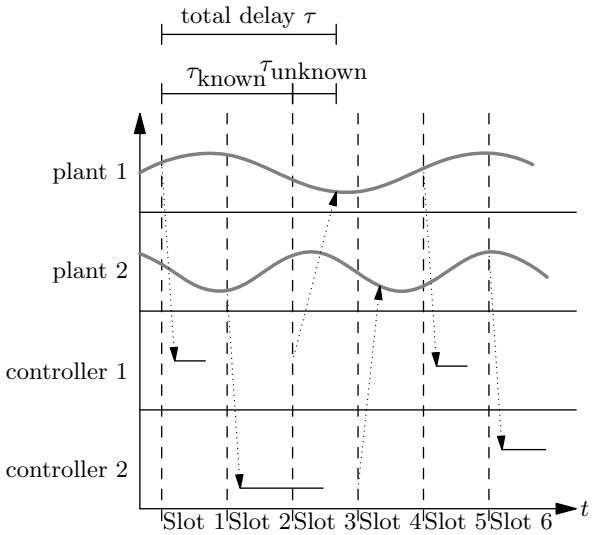


Figure 3.3: Effect of Scheduling on the Control of Multiple Plants

can be inferred from the scheduling or arbitration policy – that is twice the base slot size, i.e. $\tau_{\text{known}} = 2$. In Slot 3, Controller 1 is allowed to transmit its message. This message might arrive after an unknown transmission time. Hence, an unknown time delay τ_{unknown} (which might be zero or approximately zero in many systems) related to the communication is added to the total delay experienced by the system. Therefore the total delay that the system experiences is composed of two parts, (i) a known part τ_{known} due to the schedule, (ii) an unknown part τ_{unknown} probably due to factors like communication delay, with $\tau = \tau_{\text{known}} + \tau_{\text{unknown}}$. In other words, by making use of the information available about the DES, a large part of the delay that the messages experience is actually known. One can therefore design a control system to explicitly include this information in its design, and as a result reduce the amount of pessimism that is incurred.

3.3.2 Varying Delays in an ANCS

In this section, an example where the communication bus in question implements a FlexRay protocol (which is common in automotive architectures) is considered. In FlexRay, each communication cycle of the bus is divided into a time-triggered (or static) and an event-triggered (or dynamic) segment. On the time-triggered segment, the tasks are given access to the bus (or allowed to send messages) only at their predefined slots. On the other hand, the tasks are assigned priorities in order to arbitrate for the access to the event-triggered segment. By virtue of their segment, the temporal effects of these segments on the control messages vary. For example, for all control messages mapped onto the static time-triggered segment of the bus, the time-triggered slots and the task schedules on the ECUs may be perfectly synchronized, resulting in zero communication delays (waiting time). However, it is widely believed that as application complexity and hence communication requirements continue to

grow, the bandwidth of the time-triggered segment will not suffice and a purely time-triggered implementation might be overly expensive on one hand. Event-triggered implementations, on the other hand, are priority-driven and as such are subjected temporal non-determinism depending on the priorities assigned to messages. Control messages mapped to the event-triggered segment of the bus can be subjected to a non-zero, varying, and possibly unknown delay that depends on the characteristics of the higher priority messages mapped on the same segment.

3.4 Co-design of Control and Architecture

The previous sections show that the design of embedded controllers and the communication and computational networks that support them poses a number of challenges in their analysis and synthesis including network protocols, compatibility of operating systems, and methods for optimizing the combined performance of the control and real-time computing systems.

Such co-design brings up new challenges and opportunities that have not been studied in either the NCS or the embedded systems literature. In terms of new opportunities, since now also the network is designed, its characteristics such as delay, jitter or message loss (e.g., when they are overwritten in some internal buffers) can be chosen as desired, and hence can be better exploited when designing controllers. This is in contrast to NCS, where such characteristics are not assumed to be readily available, thereby leading to more conservatism in controller design and analysis. Further, in this context, the control design needs to accommodate architecture-level constraints, e.g., the availability or cost of computation and communication resources, paving the way for new control paradigms.

3.4.1 Co-design of Hierarchical ANCS

Hierarchical schedules as described in Section 2.4.1 have several advantages. While scheduling in general introduces a delay, the use of a hierarchical scheduling policy introduces a more specific structure into the problem. In particular, the hierarchical scheduling services different parts of the subsystem at specific time-instances thereby providing more information about the inherent delay. That is, while latency is indeed present between inputs and outputs due to the presence of a network, the network together with a hierarchical schedule allows a part of this delay to be known to the control designer. Furthermore, each subsystem can be integrated independently of the other subsystems without affecting the timing properties of messages in other branches of the hierarchy. The isolation of individual applications is also an advantage with respect to error handling. An error in one application does not propagate to other applications.

The above shows that the use of hierarchical schedules is highly attractive from an embedded systems point of view. From a control systems point of view, hierar-

chical schedules also have advantages. The hierarchical structure allows to keep the assumption of periodicity for transmission of some messages. For example, putting all sensor messages in a branch with a TDMA protocol guarantees that all sensor messages are transmitted and received on time. Furthermore, this also minimizes the communication delay. The control design can thus use this knowledge to design the controller accordingly. As outlined in the previous section, the knowledge about the schedule leads to additional information about the communication delay. For hierarchical schedules this is also true and the information can be used to improve the control performance.

In Chapter 4, a co-design of hierarchical schedules and controllers is presented. The hierarchy of the schedule allows to easily compute the worst-case end-to-end delays of all messages transmitted over the shared bus. Moreover, the information about the delay is used in combination with adaptive controllers by choosing the initial conditions appropriately.

3.4.2 Co-design of Hybrid ANCS

Another way of combining the two main paradigms in communication, time-triggered and event-triggered schedules, are hybrid protocols. Time-triggered protocols have the advantages of high quality of control (QoC) due to the possibility of reduced or zero delays, but leads to poor utilization of the communication bandwidth, high cost, overall inflexibility, and infeasibility as the number of control applications increase. On the other hand, event-triggered protocols often result in poor control performance due to the unpredictable temporal behavior of control messages and the related large delays which occurs due to the lack of availability of the bus. Hybrid protocols offer both scheduling methods. Examples for hybrid protocols are given in Section 2.4.2.

Similar to hierarchical schedules, hybrid schedules are highly attractive for the design of embedded systems. For the design of control systems, hybrid schedules also offer new possibilities. Designing a controller that suitably switches between the two schedules exploits their combined advantages of high QoC, efficient resource utilization, and low cost. Combining advanced controllers such as adaptive controllers and a hybrid protocol can be used to obtain an embedded control system that is working over a wide range of uncertainties while exploiting the resources optimally.

In Chapter 5, a co-designed hybrid switching protocol is presented. It is designed such that, if necessary, control related messages are transmitted over the time-triggered segment of the hybrid protocol. Otherwise the cheaper event-triggered segment is used. The switches between the time-triggered and the event-triggered schedule are governed by a switching algorithm. Some of the switches are introduced by external events, others are by design. The overall adaptive switching controller results therefore in improved performance.

4 ANCS with Hierarchical Protocols

Summary: *In this chapter, it is shown how the information provided by an ANCS can be used to improve the control performance of an NCS. In particular, in the first part of this chapter a novel performance metric is developed that allows to compute the optimal controller parameters for a given worst-case delay together with other standard performance metrics. In the second part, it is shown how the performance of a NCS can be improved using an adaptive controller by utilizing information from the ANCS.*

4.1 Introduction

As embedded systems become more complex and distributed – consisting of multiple processing units and communication buses – the gap between high-level control models and their actual implementations seem to widen. Control engineers are typically concerned with analyzing and simulating controllers based on well-defined semantics of both the plant and the controller being designed. Once a design is complete, has been analyzed and simulated, it is the task of the embedded systems engineer to come up with software implementations (e.g., in C) of the different control blocks (e.g., from MATLAB specifications) and implement the software on a hardware architecture or platform. Here, *platform* describes the hardware on which the controller is implemented along with the operating system layer between the hardware and the software.

Often, the platform architecture consists of multiple processors connected by one or more communication buses. Further, multiple control applications share such a platform and need to be scheduled appropriately. A natural question that arises in such a setting is: *How does one quantify or account for the semantic gap between the control models and their implementations?* This is important because many of the assumptions on periodicity or zero-delay/jitter, that are common for control models, no longer hold in real-life implementations.

While scheduling in general introduces a delay, the use of a hierarchical scheduling policy introduces a more specific structure into the problem. In particular, the hierarchical scheduling services different parts of the subsystem at specific time-instances thereby providing more information about the inherent delay. That is, while latency is indeed present between inputs and outputs due to the presence of a network, the network together with a hierarchical schedule allows a part of this delay to be known to the control designer.

In the first part of this chapter, Section 4.2, the hierarchical schedule is utilized to compute the worst-case end-to-end delay of the network. This delay is then used to optimally choose the controller parameters. To this end a novel performance metric is developed that captures stability and tracking performance of a control system with delay. Since only the worst-case delay can be computed, it is shown that this performance metric is monotonically increasing with decreasing delay. Finally, it is shown how to optimally choose the parameters of hierarchical schedules on the communication bus in order to improve multiple control performance metrics. In the second part of this chapter, Section 4.3, the fact that some part of the delay is known because of the hierarchical schedule is used to design an adaptive controller that performs better than a fixed controller. The chapter is concluded with a discussion in Section 4.4.

4.2 Optimizing Hierarchical Schedules for Improved Control Performance

In this section multiple feedback controllers being implemented on a platform consisting of multiple processing units (PUs) that communicate over a shared bus are considered. Each controller is split into multiple tasks that are then mapped onto different PUs. Hence, each PU consists of tasks from different control applications, and tasks from different PUs communicate over a single shared bus. The goal is to develop a systematic method for scheduling the different message streams on the bus, in order to jointly maximize multiple control performance metrics (e.g., those related to stability cost, and transient and steady-state performance). In the *hierarchical schedules* that are used here, each control application is allocated a Time Division Multiple Access (TDMA) slot (in order to provide temporal isolation between them), and message streams from the same controller are scheduled using fixed priority within the allocated TDMA slot. The problem is that of computing the values of the different parameters of the controller and of the hierarchical scheduler in order to maximize control performance. The parameters for the hierarchical scheduler include the TDMA cycle length and the different slot sizes (with the priorities assumed to be constant).

It may be noted that the setup consists of *multiple controllers* and *multiple performance metrics*. Hence, certain choices of parameter values that improve the performance of one controller or one metric might reduce that of another, which leads to the *design space* being non-trivial. Further, although control performance improves monotonically with decreasing the delay experienced by the different message streams (at least for the controllers we study here), the *rate* of improvement is not constant. Hence, identifying the sweet spot that leads to optimal control performance is in general a non-trivial optimization problem (whose complexity would depend on both the controllers, the performance metrics and the underlying architecture).

For the platform architecture and scheduling policies studied here, an analytical expression for the worst case end-to-end delay as a function of the various scheduler

parameters is derived. Then a general control performance metric that is a combination of stability and tracking performance is considered, and it is shown that this metric is a monotonic function of the delay. Finally, it is shown that in a general control application, the optimal control performance obtained is not only a function of the delay but also the control parameters, which clearly illustrates that a co-design of the control and the platform is needed. This co-design is formulated as a combined nonlinear optimization problem.

The implementation platform used in this section is described in Section 4.2.1. Next, the individual parts that are used for the co-design are presented in Section 4.2.2. Within this section first the design space exploration of the platform is described. Next, the formal plant description is stated and details on the used performance metrics are provided. It should be noted that in contrast to the rest of this dissertation, the plant is given by a continuous-time transfer function. The proposed monotonicity of the novel performance metric is proved in Section 4.2.3 and the co-design procedure is presented in Section 4.2.4. Then simulation results are shown to illustrate the hypothesis – the need for controller- and performance metric-specific architectures/schedulers (Section 4.2.5) and provide a complete case study of a realistic setup (Section 4.2.5).

4.2.1 Implementation Platform Design

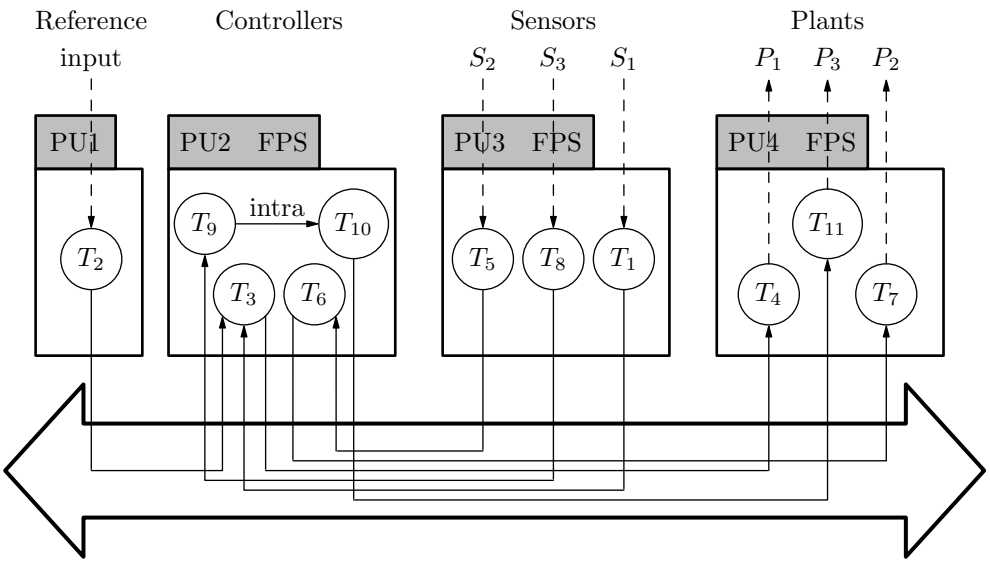


Figure 4.1: System Architecture With Three Distributed Control Applications

A platform architecture with multiple distributed control applications (see Figure 4.1) is considered. The control applications are partitioned into a number of tasks that are mapped onto different PUs. The PUs communicate via a shared communication bus and run different tasks from one or more control applications. Scheduling on the processors and arbitration on the shared bus introduces delays in various control

signals. These delays are dependent on the arbitration/scheduling policy on the buses and on the PUs.

The platform architecture, the mapping of various control tasks on it, and the hierarchical scheduling policy on the shared bus are described in the following.

Hardware/Software architecture: Three control applications are considered: *controller 1, 2 and 3* are shown in Figure 4.2, 4.3 and 4.4. Each controller is connected to an actuator and a sensor. The aim of each control application is to compute the control input to the plant such that the closed-loop system meets desired performance. Towards this, the controllers read sensors values to determine the state of the closed-loop system and compute commands based on the deviation from the desired performance. Control applications are partitioned into a number of tasks, which are mapped onto different PUs connected via the communication bus. In the architecture considered here (see Figure 4.1), PU1 hosts the tasks responsible for reading the reference command from the user, PU2 hosts all tasks that compute control commands, PU3 hosts the tasks responsible for reading sensors, and the tasks responsible for providing plant commands are mapped onto PU4.

Control applications: *Control application 1* (Figure 4.2) is partitioned into four tasks: T_1, T_2, T_3 and T_4. Task T_1 reads sensor S_1 and sends sensor signal m_1 via the communication bus to the task T_3. Similarly, task T_2 reads the reference command from the user and sends the reference command m_2 via the communication bus to task T_3. The control input is computed in T_3 using the messages m_1 and m_2. The processed output of task T_3 is sent via the communication bus to task T_4. The plant P_1 receives the control input from the task T_4. Figure 4.2 (a) and (b) show the *task graph* and the *closed-loop block diagram* for *control application 1*.

Control application 2 (Figure 4.3) is partitioned into three tasks: T_5, T_6 and T_7. Task T_5 reads the sensor S_2 and sends the sensor signal m_5 via the communication bus to the task T_6. Controller task T_6 computes the control command based on the sensor feedback signal and the constant predefined reference command. The task T_6 sends the control command to task T_7 via the communication bus. The plant P_2 receives input signal from the task T_7. Figure 4.3 (a) and (b) show the *task graph* and the *closed-loop block diagram* for *control application 2*.

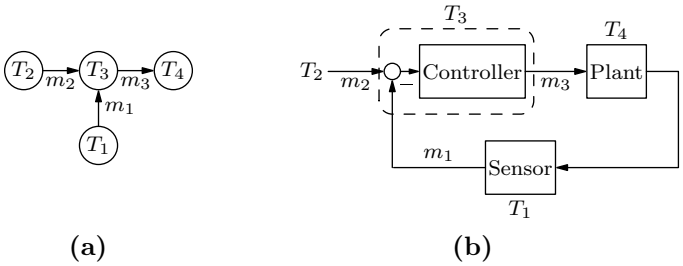


Figure 4.2: *Control Application 1*: (a) The Task Graph (b) Closed-Loop Block Diagram

Control application 3 (Figure 4.4) is partitioned into four tasks: T_8, T_9, T_{10} and T_{11}.

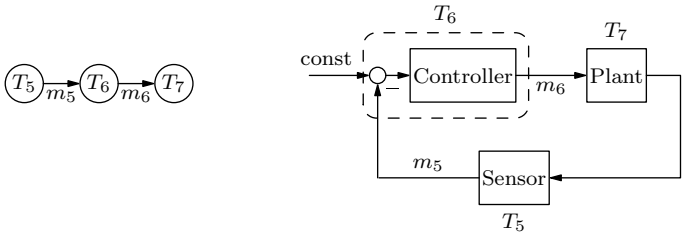


Figure 4.3: *Control Application 2*: (a) The Task Graph (b) Closed-Loop Block Diagram

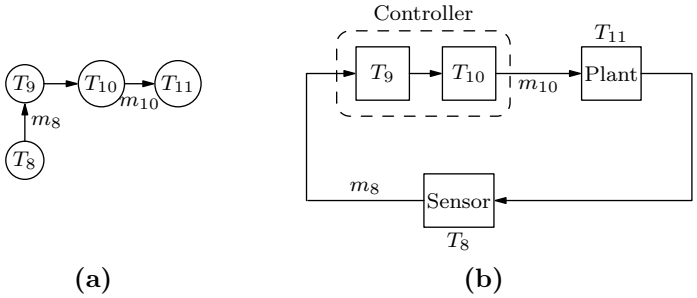


Figure 4.4: *Control Application 3*: (a) The Task Graph (b) Closed-Loop Block Diagram

Task T_8 reads the sensor S_3 and sends the sensor signal m_8 via the communication bus to the task T_9. Task T_9 computes one part of the control signal and sends the resulting command to the task T_{10}. Task T_{10} computes the control command m_{10} and sends it via the communication bus to the task T_{11} which sends the control command to the plant P_3. Figure 4.4 (a) and (b) show the *task graph* and the *closed-loop block diagram* for *control application 3*.

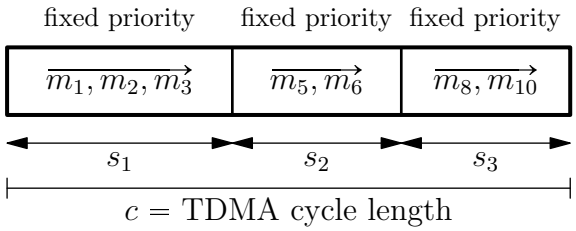


Figure 4.5: The Communication Bus: Hierarchical TDMA/FP Scheduler

Communication scheduling: The *shared communication bus* follows a hierarchical TDMA/FP scheduling policy (Figure 4.5). At the top-level of the scheduler runs a TDMA scheduler, i.e., the communication bandwidth is divided into equal cycles of length c. The TMDA cycles are further divided into three slots, i.e., s_1, s_2 and s_3 which are assigned to the *control applications 1, 2* and *3*. Therefore, *control application i (i=1, 2, 3)* can only send input streams or messages in slot s_i. For example, *control application 1* can only access the communication bus in s_1. Further, all the

streams or the messages coming into/from the *control application* which are transmitted via the communication bus follow a fixed priority scheduler (FPS). For example, *control application 1* exchanges three messages among its various tasks: m_1, m_2 and m_3. These messages can only be transmitted during the slot s_1. If all three messages are ready to be transmitted via s_1 then first m_1 gets access to the bus. Subsequently, m_2 and m_3 are transmitted respectively.

4.2.2 Prerequisites

Design space exploration

In Section 2.4.3, the analysis of timing properties in distributed embedded systems, e.g., the maximum end-to-end delay experienced by a data stream, was discussed. Given an architecture such as in Figure 4.1, the goal is to explore every possible system configuration within a specified range of interest. Towards this, the system configurations are generated while varying the TDMA cycle length c and the slot size s_i assigned to any control application C_i according to Algorithm 1 (lines 1 and 2). Towards exploring all the possible system configurations for a particular control application C_i, the slot size s_i that is assigned to this controller is varied while keeping the slot sizes for the other controllers C_j fixed (and equal to the maximum size of the messages being transmitted in those slots). For every such system configuration, the maximum end-to-end delay τ encountered by every control application (line 3) is computed according to the timing analysis technique described in Section 2.4.3. If the delays encountered by all three controllers are *finite*, the corresponding system configuration is included into the *set* of *feasible* system configurations Ω (lines 4 and 5). Therefore, at the end of the design space exploration for a particular controller C_i, a set of *feasible* system configurations with corresponding TDMA bus parameters is evaluated.

Algorithm 1 Design space exploration of system configurations.

Require: C_i, C_j, c_{min}, c_{max}, s_{min}, s_{max}, $step_c$, $step_s$

1: **for** $c \in [c_{min} + n \times step_c \leqslant c_{max}], n \in [0, 1, 2...]$ **do**
2: **for** $s_i \in [s_{min} + k \times step_s \leqslant s_{max}], k \in [0, 1, 2...]$ **do**
3: $\tau = computeDelay(),\ \forall C_i, C_j$
4: **if** $\tau \in [0, \infty),\ \forall C_i, C_j$ **then**
5: add configuration to Ω *//set of feasible configurations*
6: **else**
7: discard configuration
8: **end if**
9: **end for**
10: **end for**

Control Application Description

A general control application in continuous-time as described in Figure 1.3 is considered, with a forward-loop transfer function $G_{n,m}$ consisting of the controller, the plant, the actuator, and the sensor, and the delay due to the implementation platform, given by

$$G_{n,m}(s) = \frac{K\prod_{i=1}^{n}(s+p_i)e^{-\tau s}}{\prod_{i=1}^{m}(s+q_i)} \tag{4.1}$$

with $K > 0$, $p_i, q_i \in \mathbb{C}$ and $m \geqslant n \in \mathbb{N}$. It should be noted that complex zeros and poles have to occur as conjugate pairs. Then the closed-loop transfer function with negative unity feedback corresponding to $G_{n,m}$ is given by

$$H_{n,m}(s) = \frac{G_{n,m}(s)}{1+G_{n,m}(s)} = \frac{K\prod_{i=1}^{n}(s+p_i)e^{-\tau s}}{K\prod_{i=1}^{n}(s+p_i)e^{-\tau s}+\prod_{i=1}^{m}(s+q_i)}. \tag{4.2}$$

Next, the Fourier-transform of $H_{n,m}$ is considered, which is given by

$$h_{n,m}(j\omega) = \frac{K\prod_{i=1}^{n}(j\omega+p_i)(\cos(\tau\omega)-j\sin(\tau\omega))}{K\prod_{i=1}^{n}(j\omega+p_i)(\cos(\tau\omega)-j\sin(\tau\omega))+\prod_{i=1}^{m}(j\omega+q_i)}. \tag{4.3}$$

The product $\prod_{i=1}^{n}(j\omega+p_i)$ can be rewritten as $j\omega b_n+c_n$ where b_n and c_n are recursively defined by

$$\begin{aligned} b_1 &= 1 \\ c_1 &= p_1 \\ b_i &= p_i b_{i-1} + c_{i-1} \\ c_i &= p_i c_{i-1} - \omega^2 b_{i-1}. \end{aligned} \tag{4.4}$$

With (4.4), Equation (4.3) can be written as

$$h_{n,m}(j\omega) = \frac{K(j\omega b_n+c_n)(\cos(\tau\omega)-j\sin(\tau\omega))}{K(j\omega b_n+c_n)(\cos(\tau\omega)-j\sin(\tau\omega))+(j\omega d_m+e_m)} \tag{4.5}$$

where d_m and e_m are defined analogously to b_n and c_n according to Equation (4.4). The representation of the closed-loop system as in (4.4)-(4.5) will be useful in deriving the monotonicity property of the tracking performance.

Performance Metrics

Typical measures of stability of a feedback system are gain and delay margins. The latter is the focus here, which is used to define a stability cost function $\mathcal{P}_0$ as

$$\mathcal{P}_0 \stackrel{\text{def}}{=} \tau_0 \tag{4.6}$$

where τ_0 is the delay margin of the plant P with a controller C. The delay margin τ_0 is the amount of time delay which can be tolerated by the system before it gets unstable.

As second performance metric, $\mathcal{P}_\omega$ is defined as

$$\mathcal{P}_\omega(\omega^*) \stackrel{\text{def}}{=} \frac{1}{|h(j\omega^*)|} \tag{4.7}$$

where $h(j\omega)$ is the Fourier transform of H and ω^* is a fixed frequency in the interval $[0;\infty[$ chosen by the designer. Picking ω^* as the peak frequency, $\mathcal{P}_\omega$ reduces to the well-known H_∞-norm. It will be shown analytically in Section 4.2.3 that for fixed parameters $\mathcal{P}_\omega$ decreases with increasing delay, i.e., a larger delay impairs the performance of the overall system.

A desired controller can be defined as one that maximizes the cost function J defined as

$$J = \lambda_\infty \mathcal{P}_\omega + \lambda_0 \mathcal{P}_0 \tag{4.8}$$

where λ_∞ and λ_0 are suitably chosen weights. It is clear from (4.8) that equal values of λ_∞ and λ_0 imply an equal weighting in stability and performance, while a larger value of λ_0 relative to λ_∞ implies a greater emphasis on stability. It should be noted that the above choices for performance metrics are typical examples, and reflect measures of stability and performance costs associated with command tracking. In general any norm could be used to describe certain performance specifications.

In the following sections, an analytical expression for the performance metric $\mathcal{P}_0$ is given.

Computation of the Delay Margin

The delay margin τ_0 is determined by the phase of the forward-loop system $G_{n,m}(s)$ at the crossover frequency, ω_c, where the forward-loop gain is unity. The crossover frequency ω_c, which is independent of the delay τ, is given by

$$\omega_c = \sqrt{\frac{e_m^2 - c_n^2 K^2}{b_n^2 K^2 - d_m^2}} \tag{4.9}$$

with b_n, c_n, d_m, e_m defined in (4.4). The phase $\arg(G_{n,m}(j\omega))$ is given by

$$\arg(G_{n,m}(j\omega)) = \arctan\left(\frac{b_n\omega}{c_n}\right) - \arctan\left(\frac{d_m\omega}{e_m}\right). \tag{4.10}$$

A closed-loop system is stable as long as the phase margin of the open-loop system, i.e., the distance of the phase at the crossover frequency from $-\pi$, is positive. If the phase margin is positive, the delay margin can be computed by

$$\tau_0 = \frac{\arg(G_{n,m}(j\omega_c)) + \pi}{\omega_c}. \tag{4.11}$$

The performance metric $\mathcal{P}_0$ for system $G_{n,m}$ is then given by

$$\mathcal{P}_0 = \tau_0. \tag{4.12}$$

4.2.3 Performance Monotonicity

Since the RTC framework described in Section 2.4.3 is a deterministic framework, it provides an upper bound on the delay that each message experiences. However, the average delay experienced by a message is lower than the delay computed by RTC. It is thus necessary to show that the performance is not decreasing when the delay is decreasing. To this end it will be shown in this section for systems of the form (4.1) that $\mathcal{P}_\omega$ is monotonically decreasing with increasing delay τ for $\tau \in [0, \bar{\tau}]$ with some $\bar{\tau}$.

With (4.5), $|h_{n,m}(j\omega)|^2$ can be written as

$$|h_{n,m}(j\omega)|^2 = \frac{K^2(c_n\cos(\tau\omega) + \omega b_n \sin(\tau\omega))^2}{(\ldots)^2 + (\ldots)^2} + \frac{K^2(\omega b_n \cos(\tau\omega) - c_n \sin(\tau\omega))^2}{(\ldots)^2 + (\ldots)^2}. \tag{4.13}$$

The denominator is omitted for ease of exposition. In order to show that $\mathcal{P}_\omega$ decreases w.r.t. τ, it is shown that $|h_{n,m}(j\omega)|^2$ is monotonically increasing w.r.t. τ. To this end, it is shown that there exists a $\bar{\tau}$ such that the derivative of $|h_{n,m}(j\omega)|^2$ is greater than zero for all $\tau \in [0; \bar{\tau}]$. The derivative of (4.13) w.r.t. τ is given by

$$\frac{\partial|h_{n,m}(j\omega)|^2}{\partial\tau} = \frac{2K^3\omega(c_n^2 + b_n^2\omega)}{(\ldots)^2}\times \\ ((c_n d_m - b_n e_m)\omega\cos(\tau\omega) + (c_n e_m + b_n d_m\omega^2)\sin(\tau\omega)). \tag{4.14}$$

It is clear that (4.14) has infinitely many zeros in the interval $[0, \infty[$. The following Lemma will be used to find the smallest zero crossing.

Lemma 4.2.1. *Let $f(t) = a\sin(ct) + b\cos(ct)$ with constants $a, b, c > 0$. Then*

$$t = \frac{2}{c}\arctan\left(\frac{a + \sqrt{a^2 + b^2}}{b}\right) \tag{4.15}$$

is the smallest positive solution of $f(t) = 0$.

Using Lemma 4.2.1, the smallest $\bar{\tau}$ such that

$$\left.\frac{\partial|h_{n,m}(j\omega)|^2}{\partial\tau}\right|_{\tau=\bar{\tau}} = 0$$

can be found. $\bar{\tau}$ is then given by

$$\bar{\tau} = \frac{2}{\omega}\arctan\left(\frac{c_n e_m + b_n d_m\omega^2 + \theta}{(c_n d_m - b_n e_m)\omega}\right) \tag{4.16}$$

with

$$\theta = \sqrt{(c_n e_m + b_n d_m\omega^2)^2 + (c_n d_m - b_n e_m)^2\omega^2}.$$

In order to ensure that $|h_{n,m}(j\omega)|^2$ monotonically increases w.r.t. τ, it is checked if $\frac{\partial|h_{n,m}(j\omega)|^2}{\partial\tau}\Big|_{\tau=0} > 0$. For $\tau = 0$, (4.14) becomes

$$\frac{\partial|h_{n,m}(j\omega)|^2}{\partial\tau}\Big|_{\tau=0} = \frac{2(c_n d_m - b_n e_m)K^3\omega^2(c_n^2 + b_n^2\omega^2)}{(\ldots)^2}. \tag{4.17}$$

From (4.17) it is clear that, if

$$c_n d_m - b_n e_m > 0, \tag{4.18}$$

then

$$\frac{\partial|h_{n,m}(j\omega)|^2}{\partial\tau}\Big|_{\tau=0} > 0.$$

With this, it is clear that for any ω there exists a $\bar{\tau}$ such that $|h_{n,m}(j\omega)|^2$ increases for $\tau \in [0, \bar{\tau}]$ as long as $c_n d_m - b_n e_m > 0$. This in turn implies that the control parameters in a given application need to be suitably chosen in order to ensure monotonicity. Choosing ω to be the peak frequency ω_p results in $\mathcal{P}_\omega$ being the H_∞-norm. Any ω close to ω_p, results in an approximation of the H_∞-norm.

4.2.4 Co-design

Using the performance metrics described above, an optimization procedure that results in a minimal J is outlined. This optimization procedure is both over the control parameters and the configuration parameters, and hence results in a co-design of control and platform. The main idea is illustrated in Figure 4.6. Based on the delay values $\tau_1, \tau_2, \ldots$ obtained from the scheduling analysis, the optimal controller parameters are chosen for each delay. Together with the scheduler parameters, the optimal configuration consisting of scheduler parameters and controller parameters is chosen. In Figure 4.7, the relation between the delays obtained by the RTC procedure described in Section 2.4.3 and the performance metrics described in Section 4.2.2 is illustrated.

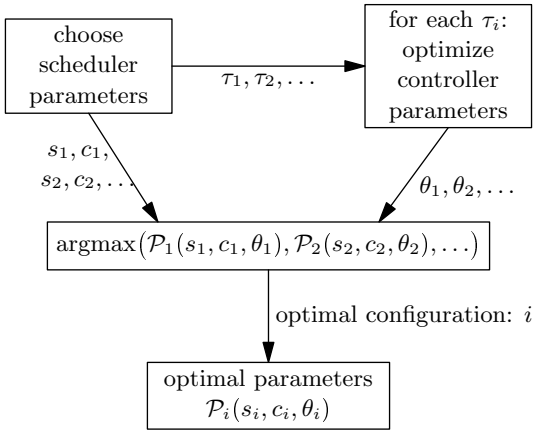


Figure 4.6: The proposed co-design

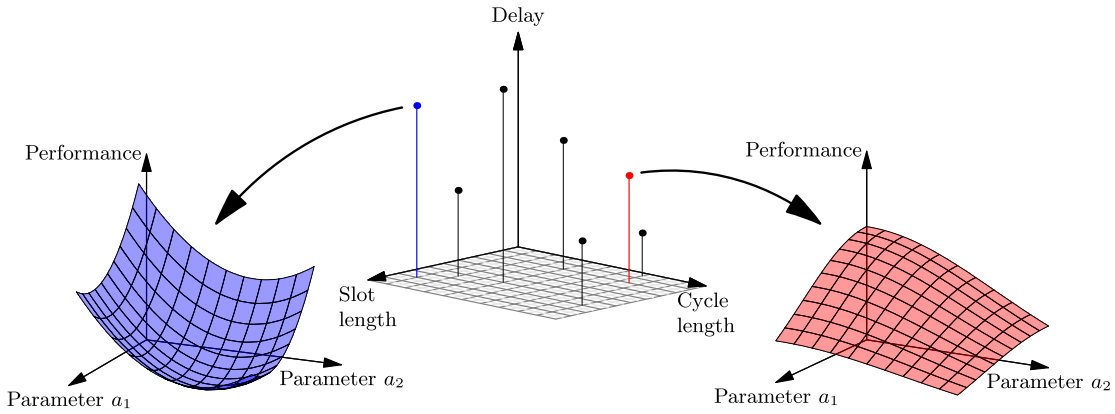


Figure 4.7: Illustration of the optimization surfaces for different delay values. Each delay value is the result of a specific scheduler configuration (Slot length and Cycle length). The performance surface over the controller design space (here Parameter a_1 and Parameter a_2) looks different for each delay value and thus a different optimal configuration is obtained for different delay values.

The cost J in (4.8) is represented using the notation $\mathcal{P}_{k,i}^{j}$ for the performance metrics, where index $k = 0, 1$ corresponds to the two different performance metrics $\mathcal{P}_0$ and $\mathcal{P}_\omega$, $i = 1, 2, 3, \ldots, |\Omega|$ represents the $|\Omega|$ feasible system configurations, and $j = 1, 2, 3, \ldots, m$ represents m copies of the controller in the embedded system architecture. The goal is to find a feasible configuration and a vector of controller parameters, such that the overall cost J which corresponds to the overal QoC is maximized. In order to find this optimal value, the total cost $\bar{\mathcal{P}}_i^j$ for a given controller j and a configuration i is considered, which is defined as

$$\bar{\mathcal{P}}_i^j(\theta_j) = \sum_{k=0}^{1} \lambda_k \mathcal{P}_{k,i}^{j}(\theta_j) \tag{4.19}$$

where θ_j represents the parameters of the control application j.

The total cost over all control applications, for a given configuration i, denoted as $\tilde{\mathcal{P}}_i(\theta_j)$ is given by

$$\tilde{\mathcal{P}}(\theta_j) = \sum_{j=1}^{m} \bar{\mathcal{P}}_i^j(\theta). \tag{4.20}$$

In the previous sections four conditions were derived that have to be fulfilled in order to ensure a monotonic behavior of the cost function $\tilde{\mathcal{P}}_i(\theta)$: (i) $\bar{\tau}$ given in (4.16) has to be larger than the time-delay of the current application, i.e., $\bar{\tau} > \tau_i$ for $i = 1, \ldots, |\Omega|$. (ii) The inequality in Equation (4.18) has to be satisfied. (iii) In order to ensure stability, the delay margin τ_0 also needs to be larger than the time-delay of the current application, i.e., $\tau_0 > \tau_i$ for $i = 1, \ldots, |\Omega|$. (iv) In order for the delay margin to exist, the radicand of the square root in Equation (4.9) has to be larger than zero. The overall optimal cost can therefore be summarized as the solution of the constrained

optimization problem

$$\begin{aligned} \mathcal{P}^*(\theta^*, i^*) = \max_{\substack{i=1,\dots,|\Omega| \\ \theta\in\Gamma}} & (\tilde{\mathcal{P}}_i(\theta)) \\ \text{s.t.} \quad & \bar{\tau} > \tau_i \\ & \tau_0 > \tau_i \\ & c_n d_m - b_n e_m > 0 \\ & \frac{e_m^2 - c_n^2 K^2}{b_n^2 K^2 - d_m^2} > 0 \end{aligned} \tag{4.21}$$

where $\mathcal{P}^*$ is the maximum performance obtained for an optimal configuration i^* and optimal control application parameters $\theta^* \in \Gamma = \{\text{all feasible parameters}\}$, $\bar{\tau}$ is the maximum delay for which monotonicity is guaranteed, and τ_0 is the delay margin.

While (4.21) clearly lays out the optimization procedure that has to be carried out to determine the optimal co-design, no analytical guarantees can be provided that a solution to (4.21) exists, as the underlying problem is nonlinear.

4.2.5 Numerical Analysis

In this section the proposed co-design procedure is illustrated with some numerical examples. First, two different control applications are considered and it is shown that the optimal performance decreases with increasing delay where the set of delay values is chosen arbitrarily without reference to any particular configuration. It is also shown that the optimal control parameters depend on the delay, which underscores the need for a co-design of the platform and the controller. Next, a complete co-design study is performed, where the delays are derived for all possible platform configurations using Algorithm 1 outlined in Section 4.2.2, and a combined optimization procedure, outlined in Section 4.2.4, is carried out when three control applications are present.

Design with Two Control Applications

Two cases of the general control application given in (4.1) are considered, with forward-loop transfer functions

$$G_{1,2}(s) = \frac{K}{s+a}\frac{s+10}{s+5}e^{-\tau s} \tag{4.22}$$

$$G_{2,3}(s) = \frac{K}{s+a}\frac{(s+1)(s+10)}{(s+3)(s+4)}e^{-\tau s} \tag{4.23}$$

where $C(s) = \frac{K}{s+a}$ is assumed to be the controller.

For both control applications, $G_{1,2}$ and $G_{2,3}$, the best achievable performance P^* with respect to a delay τ was computed. The frequency ω^* in (4.7) was chosen close to the peak frequency as

$$\omega^* = \frac{1}{G_{n,m}(0)}.$$

The delay τ is varied linearly from 0 ms to 150 ms with a step size of $\Delta\tau = 5$ ms. For both systems the parameters were chosen from the intervals $a \in [0.1, \infty[$ and $K \in [0.1, \infty[$. The weights λ_∞ and λ_0 in (4.19) are chosen as $\lambda_\infty = 1$ and $\lambda_0 = 1$. The optimal values for the parameters K, a, and τ were then determined using (4.21).

In Figure 4.8, the best possible performance with respect to the delay for systems $G_{1,2}$ and $G_{2,3}$ is shown. It can be seen from Figure 4.8, that the best achievable performance degrades with increasing delay. That is,

$$J(a(\tau_1), K(\tau_1), \tau_1) \leqslant J(a(\tau_2), K(\tau_2), \tau_2) \tag{4.24}$$

for all $0.15\,\text{s} \geqslant \tau_1 > \tau_2 > 0\,\text{s}$. However, the optimal parameters K^* and a^* depend on the delay τ, as shown in Figure 4.9 for system $G_{1,2}$. This clearly shows the advantage of a co-design of control and communication.

Figure 4.10 shows that the optimal performance depends on the weights λ_∞ and λ_0. It can be seen that for a high value of λ_0, large delay values can have the same or even better performance as small systems with small delay values. This is because the optimization procedure tries to make the delay margin as small as possible, i.e., as close to the delay τ.

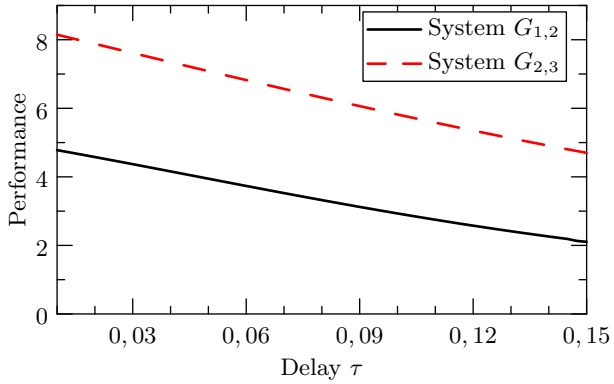


Figure 4.8: Optimal performance of systems $G_{1,2}$ and $G_{2,3}$ with respect to the delay $\tau \in [0, 0.15]$.

Co-design with Three Control Applications

In this section, the proposed co-design approach is illustrated with a complete case study of a realistic embedded control setup. Throughout this section, it is assumed that the control application has a transfer function as in Equation (4.22). First, a set of feasible platform configurations is derived and then the optimal control values are computed for every configuration, resulting in a co-designed control-platform-architecture.

In order to carry out the optimal co-design outlined in Section 4.2.4, the delays τ_j, $j = 1, 2, 3$ are determined using the *design space exploration* method outlined in Section 4.2.2. Given the system architecture shown in Figure 4.1 together with the task

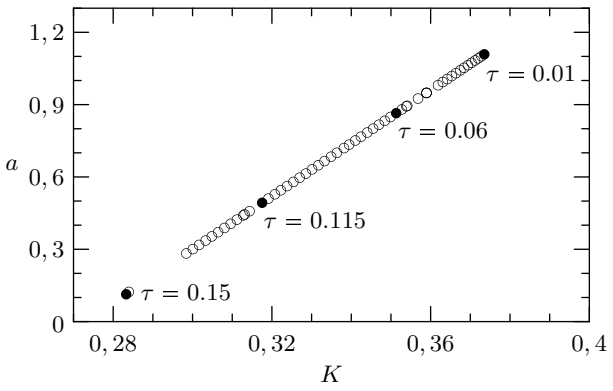


Figure 4.9: Each dot represents a delay value with solid dots indicating the actual delay value. The Figure illustrates how K and a vary for each of these values. For example, for a delay value $\tau = 60\,\text{ms}$, the optimal parameters are $K = 0.268$ and $a = 0.864$. This figure shows that the best performance for two different delay values of τ is achieved for two different controller parameters a and K.

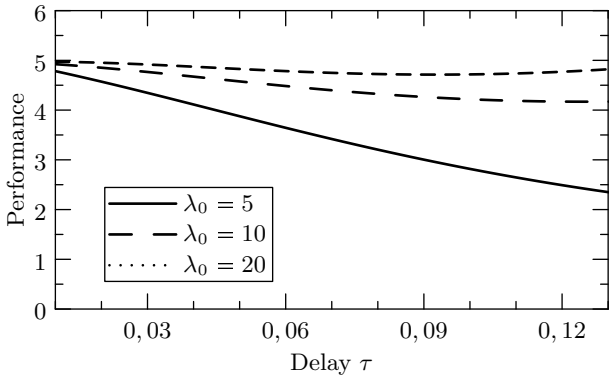


Figure 4.10: Optimal performance of system $G_{1,2}$ with respect to the delay $\tau \in [0, 0.13]$ for different values of λ_0.

mapping for the three control applications shown in Figure 4.2–4.4, sensor S_1 is chosen to send the sensor reading periodically at an interval of 0.02 s and sensors S_2 and S_3 to send the sensor streams periodically with periods 0.03 s and 0.04 s respectively with jitters 0.006 s and 0.002 s. The architecture shown in Figure 4.1 has 11 tasks mapped onto various PUs and are executed according to FP schedulers. The execution demand of the tasks are (in s): $T_1 = 0.002$; $T_2 = 0.002$; $T_3 = 0.003$; $T_4 = 0.003$; $T_5 = 0.001$; $T_6 = 0.002$; $T_7 = 0.004$; $T_8 = 0.001$; $T_9 = 0.002$; $T_{10} = 0.003$ and $T_{11} = 0.006$. The 11 tasks generate 7 messages which are transmitted via the shared communication bus according to a *hierarchical* TDMA/FP scheduling policy. The transmission times of the messages are (in s): $m_1 = 0.004$; $m_2 = 0.002$; $m_3 = 0.002$; $m_5 = 0.003$; $m_6 = 0.002$; $m_8 = 0.002$; $m_{10} = 0.002$.

For the design space exploration, the TDMA cycle length c is varied from $c_{min} = 0.010$ s to $c_{max} = 0.040$ s in steps of $step_c = 0.005$ s according to Algorithm 1. While varying the TDMA cycle length, the slot size s_1 of controller C_1, that we are interested to optimize, is also varied, while keeping the other two slot sizes s_2 and s_3 fixed, i.e., s_1 is varied from $s_{min} = 0.001$ s to $s_{max} = (c_{max} - s_2 - s_3)$ in steps of $step_s = 0.0005$ s. Further, the corresponding maximum end-to-end delay τ encountered by each controller is computed and the feasible system configurations where the delay is *finite* are retained. In total 273 system configurations were explored and $|\Omega| = 15$ *feasible* configurations out of them (see Table 4.1) were retained. For each of the i configurations 1 through 15, the corresponding delays are indexed as τ_{j_i}, $i = 1, \ldots, 15$. The optimal co-design now consists of choosing $\mathcal{P}^*(\theta^*, i^*)$ defined in (4.21). The set of all feasible parameters $\Gamma^j_{k,i}$ is defined as

$$\Gamma^j_{k,i} := \{(a, K) \,|\, \tau_0(a, K) < \tau_{j_i}\}. \tag{4.25}$$

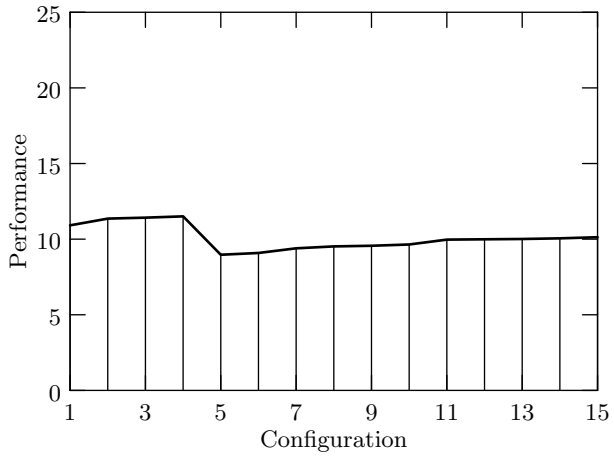


Figure 4.11: Optimal performance of system $G_{1,2}$ over all feasible configurations.

Figure 4.11 shows the best achievable performance for each of the 15 feasible configurations for system $G_{1,2}$. The weights in (4.8) were chosen as $\lambda_\infty = 1.0$ and $\lambda_0 = 0.5$. The same data is illustrated in a different manner in Figure 4.12 which shows the

config.	c	s_1	Delay [ms]		
	[ms]	[ms]	Cont. App. 1	Cont. App. 2	Cont. App. 3
1	10	3.5	64	56	74
2	10	4.0	43	56	74
3	10	4.5	40	56	74
4	10	5.0	36	56	74
5	15	5.0	80	99	113
6	15	5.5	69	102	116
7	15	6.0	57	99	116
8	15	6.5	51	99	116
9	15	7.0	49	99	116
10	15	7.5	47	99	113
11	15	8.0	35	96	113
12	15	8.5	34	96	113
13	15	9.0	33	96	113
14	15	9.5	31	96	113
15	15	10.0	30	96	110

Table 4.1: Cycle length, slot length, and delay values of the 15 feasible configurations.

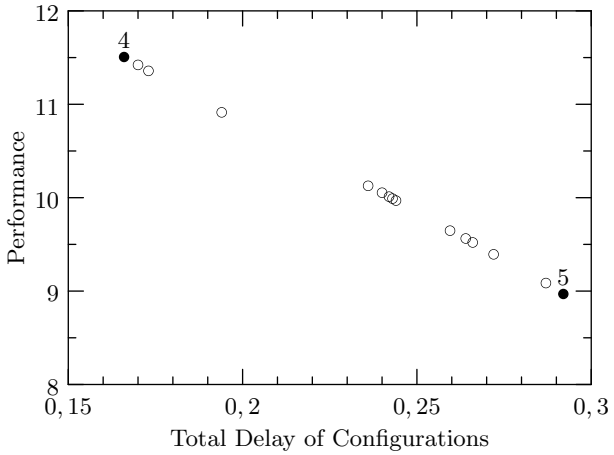


Figure 4.12: The variation in performance with each delay for the three applications for the same data illustrated in Figure 4.11. Each dot refers to one configuration; solid dots refer to the best and worst configuration. It is clear from this plot that Configuration 4 is the most optimal one.

total delay of each configuration versus the best performance. It can be seen that configuration 4 achieves the best performance with the optimal parameters for each of the three controllers given in Table 4.2.

Control Application	a	K
1	9.2441	7.6055
2	6.4446	0.24399
3	0.35735	0.46758

Table 4.2: Optimal parameters for configuration 4

4.3 Adaptive Control and Hierarchical Schedules

In this section, an adaptive controller is used with an ANCS. An adaptive controller is proposed to accommodate the effect of uncertainties and it is shown that this adaptive controller can stabilize the system, provide asymptotic tracking of a desired signal, and make use of any prior information that may be available regarding the unknown parameters including the delay. It is shown using simulation studies that such a design leads to improved performance compared to non-adaptive NCS.

In Section 4.3.1, the underlying problem of control using an embedded system with a hierarchical schedule is stated and a description of the system is given. The adaptive controller which accommodates the uncertainties, stabilizes the system, and makes use of the structure of the hierarchical scheduler given in Section 4.3.2. Finally, numerical simulations in Section 4.3.3 which show the advantage of the proposed adaptive controller over non-adaptive ones proposed in the literature is presented.

4.3.1 Problem Formulation

In this section the NCS model that is used in the rest of the chapter is introduced and the problem formulation is given. The communication bus in the embedded control system is digital as a result of which it is necessary to sample the plant to get digital signals and to translate the control inputs into piecewise-continuous signals. While the first is done by periodically sending sensor measurements of the plant output (or states) to the controller, the latter is done by a zero-order hold (ZOH) device or any higher order hold device, see also Section 2.2. The overall structure of the NCS model is shown in Figure 4.13. Each device or processing unit that has access to the network is called a node.

The second aspect of an embedded control system is the presence of a shared communication medium. As described in previous sections and chapters, the presence of

a schedule introduces a communication delay. As denoted in Figure 4.13, the network introduces two time delays, τ_{sc} and τ_{ca}. These two time delays are communication induced. The third time delay, τ_c represents the computation time in the controller node. It is assumed that the sensor is clock-driven, while the controller and the actuator are event-driven. It is also possible to consider a clock-driven controller as in [62] but this case is not considered here. The total time-delay of the NCS is given by $\tau = \tau_{sc} + \tau_{ca} + \tau_c$. Usually, the computational delay τ_c is small compared to τ_{sc} and τ_{ca} and is therefore neglected in the following. The network here is assumed to have a hierarchical TDMA/FPS schedule as described in Section 4.2.1.

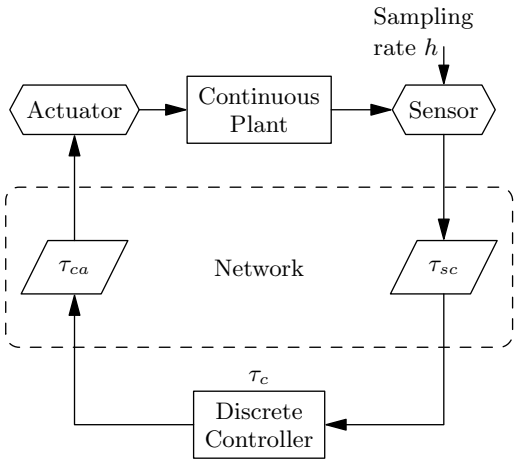


Figure 4.13: The underlying NCS Model

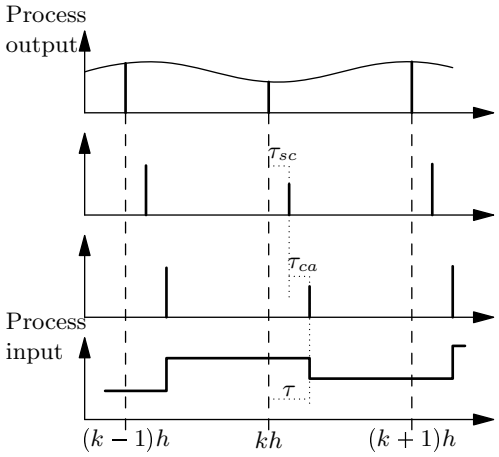


Figure 4.14: Timing of an event-driven control for a single plant

An embedded control system is considered which controls a plant described by a linear continuous-time model (see Section 2.2)

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t-\tau) \\ y(t) &= C_c x(t) \end{aligned} \tag{4.26}$$

with $A_c \in \mathbb{R}^{n\times n}$, $B_c \in \mathbb{R}^{n\times 1}$, and $C_c \in \mathbb{R}^{1\times n}$ and a time-delay $\tau > 0$ which represents the combined effect of lags due to communication and computation in the embedded system. Since the controller is implemented in a digital manner, a sampled version of (4.26) is derived according to Section 2.2. The total time delay in the system τ is assumed to be less than one sampling period h. The sampled version of (4.26) is written in predictor form according to Section 2.2.1 as

$$y(k+d) = \theta^{*T}\Phi(k) \tag{4.27}$$

$$= \vartheta^{*T}\phi(k) + \beta_0 u(k) \tag{4.28}$$

where $\phi(k)$, ϑ^*, $\Phi(k)$, and θ^* are defined as in Equations (2.24) and (2.25).

The goal is to control the continuous plant in (4.26) respectively the discrete plant in (4.28) on an embedded platform using a discrete-time adaptive controller while the operating conditions of the communication network or of the plant change or are uncertain.

4.3.2 Choice of Initial Conditions

As described in Section 2.3.1, a discrete-time adaptive controller is chosen as in Equations (2.37)-(2.39) with the control law

$$u(k) = \frac{1}{\hat{\theta}_\nu(k)} \left(y_{\text{ref}}(k+d) - \hat{\vartheta}_d(k)^T \phi_d(k) \right) \tag{4.29}$$

where $\hat{\theta}_\nu$ and $\hat{\vartheta}(k)^T$ are the estimates of β_0 and ϑ^*.

As described in the section on ANCS (Section 3), the partial knowledge of the time delay experienced by the system can be used to design more efficient controllers. Here, this knowledge is used to choose the initial values of the parameter estimates $\hat{\theta}$. That is, given that $\tau = \tau_{\text{known}} + \tau_{\text{unknown}}$ as discussed in Section 3, noting that $\theta^* = f(\tau)$ is unknown, the initial condition for the parameter estimate $\hat{\theta}(0)$ is chosen as

$$\hat{\theta}(0) = f(\tau_{\text{known}}) \tag{4.30}$$

where f is given in Equation (2.28).

It should be pointed out that the adaptive control design proposed here explicitly makes use of the prior knowledge of τ, by incorporating the knowledge of τ_{known} in the initial parameter estimates of the controller, and adapting to the uncertainty due to τ_{unknown}.

4.3.3 Simulation Results

In this section, numerical simulations are provided. It is shown that the prior knowledge of τ used in the choice of the initial parameter estimates of the adaptive controller improves the performance of the overall system. To this end, the delayed single-input single-output continuous-time system

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t-\tau) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned} \tag{4.31}$$

with the time-delay $\tau = 10\,\text{ms}$ is considered. The goal is to stabilize (4.31) and have its output follow that of a reference model given by

$$y_{\text{ref}}(k) \equiv 7 \quad \forall k = 0, 1, 2, \ldots \tag{4.32}$$

It is assumed that the sampling rate $h = 20\,\text{ms}$. First, a sampled version of (4.31) is derived. Following the procedure described in Section 2.2, the discrete-time version of (4.31) is given by

$$\begin{aligned} (1 - 1.9798q^{-1} + 0.9802q^{-2})y(k) = \\ (4.98 \times 10^{-5}q^{-1} + 2.97 \times 10^{-4}q^{-2} + 4.92 \times 10^{-5}q^{-3})u(k). \end{aligned} \tag{4.33}$$

From (4.33) a control law with fixed parameters can be derived according to (2.27). The parameters of the controller are given by

$$\beta_0^1 = 4.98 \times 10^{-5} \qquad \text{and} \qquad \vartheta_1^* = \begin{bmatrix} 1.9798 \\ -0.9802 \\ 2.97 \times 10^{-4} \\ 4.92 \times 10^{-5} \end{bmatrix}. \tag{4.34}$$

It should be noted that this controller is designed for a system with a fixed-delay of $\tau = 10\,\text{ms}$. As mentioned above, the plant is subject to changes in the operating conditions and uncertainties in the model and therefore an adaptive controller is designed according to (4.29). Both controllers, the one with the fixed parameters as well as the adaptive one, are compared through numerical simulations. According to Section 4.3.2, the initial conditions of the adaptive controller are chosen as

$$\hat{\theta}_\nu(0) = \beta_0 \quad \hat{\vartheta}(0) = \vartheta^*. \tag{4.35}$$

Additionally, an adaptive controller without using the known information about the delay τ is designed, i.e., a controller according to (2.36) is designed but the initial conditions are chosen as

$$\hat{\theta}_\nu(0) = 0 \quad \hat{\vartheta}(0) = 0. \tag{4.36}$$

It should be noted that both the controller with the fixed parameters and the adaptive controller have a similar, output-feedback based structure which does not require filtering as it is necessary for the standard continuous-time output-feedback MRAC controller [7].

Three simulations were carried out using the setup shown in Figure 4.15. The controller is implemented on a digital processor and can either be the fixed controller given in Equation (2.27) with the parameters given in Equation (4.34) or the adaptive controller described in (4.29). In the first simulation, the time delay $\tau(k)$ is held constant at 10 ms, i.e., $\tau(k) = 10\,\text{ms}\ \forall k$. In the second simulation, the delay $\tau(k)$ is generated using a uniformly distributed random variable between $[10\,\text{ms}, 20\,\text{ms}]$. Note that the maximum delay is equal to the sampling rate $h = 20\,\text{ms}$ and thus all messages arrive within one sample. In the third simulation, the delay $\tau(k)$ is generated using a uniformly distributed random variable between $[10\,\text{ms}, 80\,\text{ms}]$. The maximum delay in this simulation is larger than the sampling rate which means that the order of messages is not preserved.

Figures 4.16–4.18 show the closed-loop responses with the adaptive and fixed controller. It can be seen from Figure 4.16 that the adaptive controller with initial condition (4.35) and the fixed controller are able to track the reference signal. The adaptive controller with initial condition (4.36) also tracks the reference signal but takes longer to achieve the task. If the delay is no longer a constant, but varying randomly between 10 ms and 20 ms, it can be seen from Figure 4.17 that the fixed controller is not able to follow the reference signal, whereas both adaptive controllers easily accommodate the

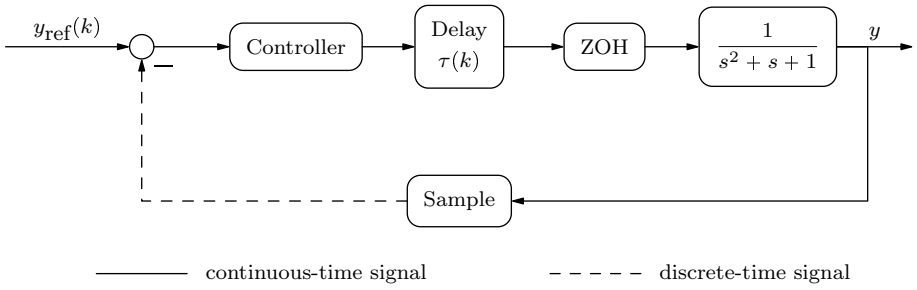


Figure 4.15: Setup used for the simulations

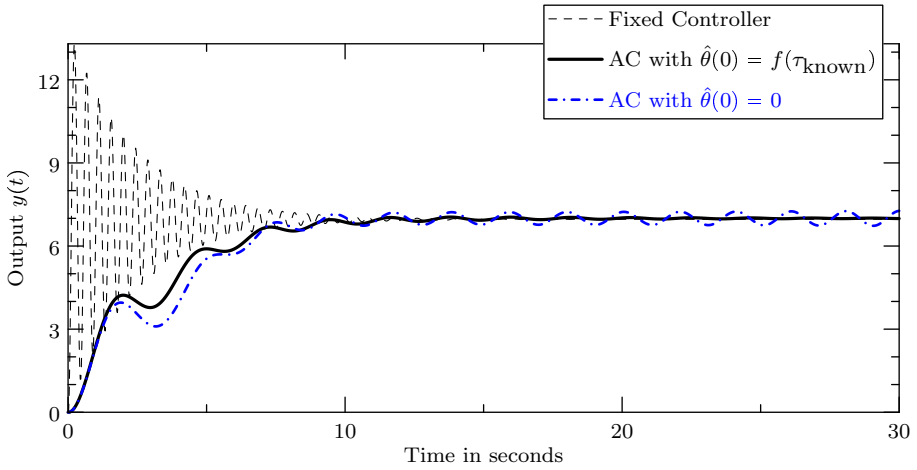


Figure 4.16: Output of the plant with Adaptive Controller with initial conditions using $\tau_{\text{known}} = 10\,\text{ms}$ and with zero initial conditions and the Fixed Controller with fixed time-delay $\tau = 10\,\text{ms}$

tracking task. If the range of the time-delay is further increased (see Figure 4.18), the system with the fixed controller becomes unstable, whereas both adaptive controllers easily stabilize the system and track the reference signal.

4.4 Summary

In this chapter, ANCS with hierarchical schedules were considered. In the first part, a co-design scheme for optimizing the parameters of a hierarchical bus scheduler and the controller parameters in order to improve various control performance metrics was proposed. A novel performance metric was introduced and it was shown that with this metric the performance of the control application decreases with increasing delay. With the information on the delay, the selection of the platform parameters and the controller parameters could be combined into one nonlinear constrained optimization problem. It was further shown that the performance depends on the platform parameters as well as the controller parameters, which clearly illustrates that a co-design is needed.

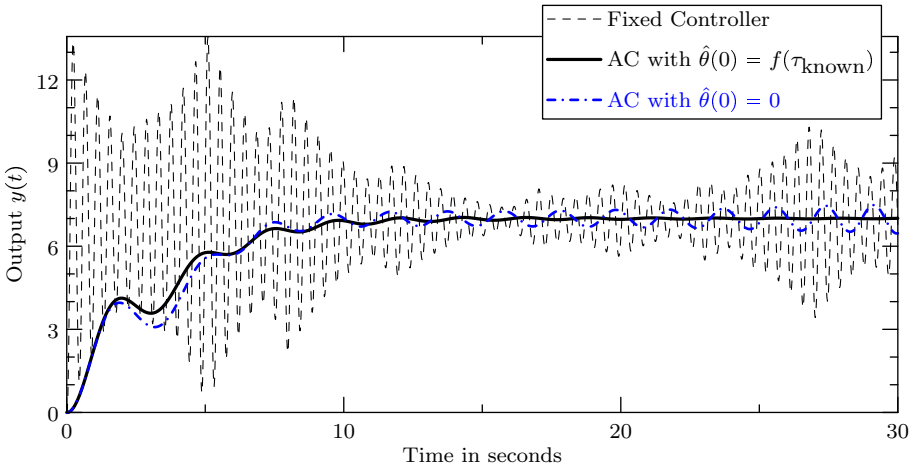


Figure 4.17: Output of the plant with Adaptive Controller with initial conditions using $\tau_{\text{known}} = 10\,\text{ms}$ and with zero initial conditions and the Fixed Controller with random time-delay $\tau \in [10\,\text{ms}; 20\,\text{ms}]$

In the second part, an adaptive controller was used to control the system. The ANCS view of the system provided further information on the platform, in this case the partial knowledge of the communication delay, which was used to improve the performance of the controller compared to a fixed controller. The effect of uncertainties that may be present either in the plant to be controlled or in the network were addressed by introducing an adaptive controller. This adaptive controller is shown to accommodate the uncertainties, stabilize the system, and make use of the structure of the hierarchical scheduler. Numerical simulations were carried out and shown to support the theoretical derivations.

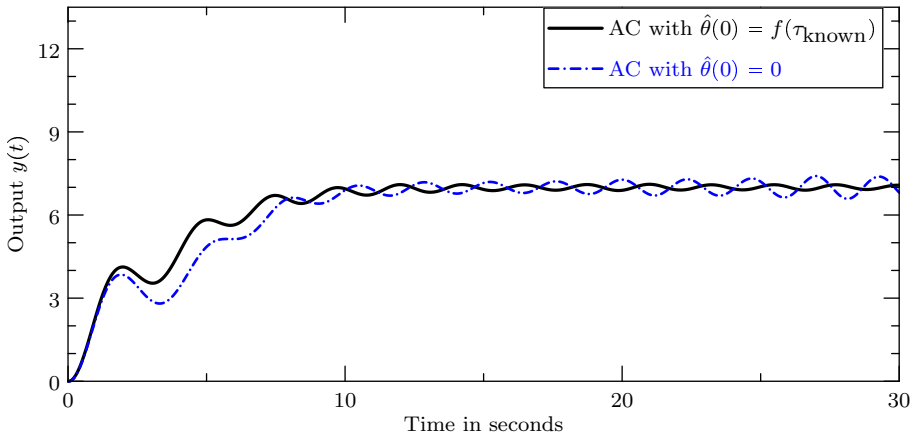


Figure 4.18: Output of the plant with Adaptive Controller with initial conditions using $\tau_{\text{known}} = 10\,\text{ms}$ and with zero initial conditions with random time-delay $\tau \in [10\,\text{ms}; 80\,\text{ms}]$. The output of the plant with the fixed controller is not shown here since the system is unstable.

5 ANCS with Hybrid Communication Protocols

Summary: *In this chapter an ANCS is presented that uses a hybrid, co-designed control and communication protocol for the control of multiple applications. The co-design consists of an adaptive switching controller and a hybrid communication architecture that switches between a time-triggered and event-triggered protocol. An adaptive controller is proposed in order to cope with any uncertainties that may be present. The resulting adaptive switching controller is then shown to be stable and to achieve regulation and tracking in the presence of a class of disturbances.*

5.1 Introduction

A commonality of Cyber-Physical Systems (see also Section 3.1) is that complex tasks have to be carried out using communication and computation with shared resources [47, 69, 101, 114, 137, 138]. In order to manage the flow of information in the communication network, various protocols based on a time-triggered [45] or event-triggered algorithm [54, 58, 75] have been suggested over the years. Associated with each of these communication protocols are different set of advantages and disadvantages. The assignment of time-triggered (TT) slots to all control-related signals has the advantage of high quality of control (QoC) due to the possibility of reduced or zero delays, but leads to poor utilization of the communication bandwidth, high cost, overall inflexibility, and infeasibility as the number of control applications increase. On the other hand, event-triggered (ET) schedules often result in poor control performance due to the unpredictable temporal behavior of control messages and the related large delays which occurs due to the lack of availability of the bus. These imply that a hybrid protocol that suitably switches between these two schedules offers the possibility of exploiting their combined advantages of high QoC, efficient resource utilization, and low cost [139], and are increasingly being used in [140, 141].

The focus in this chapter is on a hybrid switching protocol, and the design of a switching controller that is customized to the properties of the hybrid protocol. The second unique property of the proposed controller is adaptivity. It is assumed that multiple plants need to be controlled and that these plants have parametric uncertainties, and an adaptive switching controller is proposed and its implementation using a hybrid protocol is evaluated. The main result of this chapter is the demonstration that this switching adaptive controller is stable and has bounded solutions in the case of

regulation to zero and in the case of tracking a reference signal. For ease of exposition the simpler case of regulation is addressed first, followed by the tracking case. In both cases, a multiple set of Lyapunov functions is used in order to address stability.

The following are the main contributions of this chapter:

- The specific solution proposed is a co-design where the switches of the controller are aligned synergistically with the switches in the embedded platform. While part of the switches are introduced due to disturbances and therefore are due to the environment, part of the switches are by design.
- The plant being controlled is assumed to be unknown, and an adaptive control solution is employed to accommodate the uncertainty. This significantly complicates the problem and necessitates a fairly involved proof of stability. The main challenge in demonstrating the stability of the underlying switching adaptive controller is the fact that the delay in the underlying plant-model switches between two values which, in turn, is due to the hybrid use of the TT and ET protocols.
- Since the hybrid protocol is highly attractive because of the advantages it offers and is becoming more and more prevalent among embedded systems, the proposed solution will have a significant impact.

This chapter is organized as follows: The proposed co-design is described in Section 5.2, starting with the problem formulation (Section 5.2.1) and followed by a description of the operating modes (Section 5.2.2), the switching scheme (Section 5.2.3), and the design of the adaptive controller in Section 5.2.4. The main results are proved in Sections 5.3 and 5.4. Finally, concluding remarks are presented in Section 5.5.

5.2 The Proposed Co-design

In this section the control-communication co-design is presented. First, the problem formulation is given, followed by a formal definition of the operating modes. Next, the main switching scheme is presented, and the section is concluded by a formal description of the controller design.

5.2.1 Problem Formulation

The problem that is addressed in this chapter is the simultaneous control of n plants, C_i, $i = 1, \ldots, n$, in the presence of impulse disturbances that occur sporadically, using a hybrid communication protocol that switches between time-triggered and event-triggered communication protocols.

The plant to be controlled is assumed to have a discrete time model with a description as given in Equation (2.30), that is

$$A(q^{-1})y(k) = q^{-d}B(q^{-1})u(k) + G(q^{-1})D(k) \qquad k \geqslant 0 \tag{5.1}$$

where $u(k)$ and $y(k)$ are the input and output of the i-th control application, respectively, at the time-instant t_k and $d \geqslant 1$ is a time-delay. For ease of exposition, it is assumed that all n plants have identical dynamics. The disturbance $D(k)$ is assumed to be a set of impulses whose inter-arrival time is lower-bounded by a finite constant. The parameters of the i-th plant are given by a_l, $l = 1, \ldots, m_1$, b_l, $l = 0, \ldots, m_2$ and are assumed to be unknown. It is further assumed that the sampling time of the controller is a constant h, so that $t_{k+1} = t_k + h$. The goal is to choose the control input $u(k)$ such that $y(k)$ tracks a desired signal $y_{\text{ref}}(k)$, which can be equal to zero, with all signals remaining bounded.

As outlined in Section 2.2.2, the model in (5.1) can be expressed as

$$y(k+d) = \theta^{*T}\Phi(k) + \mu^T\Delta(k) \tag{5.2}$$
$$= \vartheta^{*T}\phi(k) + \beta_0 u(k) + \mu^T\Delta(k) \tag{5.3}$$

with $\phi(k)$, ϑ^*, $\Phi(k)$, θ^*, μ, and $\Delta(k)$ as given in Equations (2.24), (2.25) and (2.35).

5.2.2 The Operating Mode

Three different sets of modes are considered in this section: (i) The system modes, (ii) the communication modes, and (iii) the operating modes. These modes are being defined in this section.

System modes: In order to ensure quality performance, i.e., that the peak error does not become too large and to recover from external disturbances as fast as possible, a threshold e_{th} is set on the output. If $|y(k) - y_{\text{ref}}(k)| > e_{\text{th}}$, the system is said to be in a transient mode and therefore an aggressive control action is warranted. If the output magnitude does not exceed this threshold, i.e., if $|y(k) - y_{\text{ref}}(k)| \leqslant e_{\text{th}}$, the system is said to be in a steady-state mode and requires a less aggressive control action. In other words, the controller needs to adapt to two different situations and needs to suitably switch and tune. Denoting the two system modes as $\mathcal{M}_{\text{tr}}$ for the transient mode and $\mathcal{M}_{\text{ss}}$ for the steady-state mode, $\mathcal{M}_{\text{System}}(k)$ is defined as

$$\mathcal{M}_{\text{System}}(k) := \begin{cases} \mathcal{M}_{\text{tr}} & \text{if } |y(k) - y_{\text{ref}}(k)| > e_{\text{th}} \\ \mathcal{M}_{\text{ss}} & \text{if } |y(k) - y_{\text{ref}}(k)| \leqslant e_{\text{th}}, \end{cases} \tag{5.4}$$

where $\mathcal{M}_{\text{System}}(k)$ denotes the mode of the system at time t_k. It should be noted that the switch $\mathcal{M}_{\text{ss}} \to \mathcal{M}_{\text{tr}}$ happens due to disturbances or other external influences. However, the switch $\mathcal{M}_{\text{tr}} \to \mathcal{M}_{\text{ss}}$ can be achieved by choice of a suitable controller.

Communication modes: As a hybrid communication protocol is used, it can be decided if a message is sent in the static or in the dynamic segment. Therefore, if an event-triggered schedule is used the bus is denoted to be in the mode $\mathcal{M}_{\text{ET}}$ and if a time-triggered schedule is used in mode $\mathcal{M}_{\text{TT}}$, i.e.,

$$\mathcal{M}_{\text{Bus}}(k) := \begin{cases} \mathcal{M}_{\text{ET}} & \text{if } S \in \mathcal{S}_{\text{DYN}} \\ \mathcal{M}_{\text{TT}} & \text{if } S \in \mathcal{S}_{\text{ST}} \end{cases} \tag{5.5}$$

where $\mathcal{M}_{\text{Bus}}(k)$ denotes the mode of the bus at time t_k.
Operating mode: The resulting mode of the overall system (consisting of the plant and the bus) is therefore given by the combination of the system ($\mathcal{M}_{\text{tr}}/\mathcal{M}_{\text{ss}}$) and the communication mode ($\mathcal{M}_{\text{ET}}/\mathcal{M}_{\text{TT}}$). Hence, the operating mode $\mathcal{M}_{\text{OpMode}}(k)$ can take four different modes given by

$$\mathcal{M}_{\text{OpMode}}(k) := \begin{cases} \mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{TT}} \\ \mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{ET}} \\ \mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{TT}} \\ \mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{ET}} \end{cases} \tag{5.6}$$

5.2.3 Switching Scheme

The focus of this problem is the simultaneous control of n applications $\mathcal{C}_i, i = 1, \ldots, n$ for stabilization and tracking. That is, the goal is to choose $u(k)$, the input of the ith control application, such that $y(k)$, its output, converges to $y_{\text{ref}}(k)$, for $i = 1, 2, \ldots, n$. It is assumed that there are m TT communication slots that are to be *shared* by the n control applications, and the remaining bandwidth to be allocated by ET communication schedules. $m = 0$ implies that the entire time is devoted to ET, and $m = n$ implies that each application uses its own TT slot all the time (which as explained before is the more expensive implementation option). Each application C_i is assigned to one TT slot $\mathcal{P}_j$, i.e., the set $\mathcal{P}$ of all applications is partitioned into m disjoint subsets $\mathcal{P}_j, j = 1, \ldots, m$ with $\mathcal{P} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_m$. If $m < n$, at least one $\mathcal{P}_j$ contains more than one control application. In the context of the problem under consideration, all control applications are partitioned into a sensor task T_s, a controller task T_c, and an actuator task T_a. In Figure 5.1, this is depicted for one application. Using *time-triggered* communication schedules, denoted as $\mathcal{M}_{\text{TT}}$, applications are allowed to send messages only at their assigned slots. Since multiple applications are assigned to the same TT slot, only one application can use its assigned slot at any point in time. In $\mathcal{M}_{\text{TT}}$, the tasks are triggered synchronously with the bus, i.e., it is assumed that the communication delay is negligible and hence the delay d in (2.19) is equal to 1. On the other hand, in an *event-triggered* schedule, denoted as $\mathcal{M}_{\text{ET}}$, the tasks are assigned priorities in order to arbitrate for access to the bus. Multiple control applications share the same bus and hence multiple control messages have to be sent using a common bus. This means that messages might experience a communication delay τ when the higher priority tasks access the event-triggered segment. The event-triggered communication schedule is chosen such that the sensor-to-actuator delay τ is within $(d_2 - 1)$ sample intervals, i.e., $0 < \tau \leqslant (d_2 - 1)h$ for the control-related messages and hence the delay d is at most equal to d_2 with $d_2 \geqslant 2$. In summary, the delay $d = 1$ if $\mathcal{M}_{\text{Bus}}(k) = \mathcal{M}_{\text{TT}}$ and $d = d_2$ if $\mathcal{M}_{\text{Bus}}(k) = \mathcal{M}_{\text{ET}}$ where $\mathcal{M}_{\text{Bus}}(k)$ denotes the protocol used at time-instant t_k.

The properties of the varying delay of the TT and ET protocol are directly exploited in the control design in the following way. Whenever the error between the plant

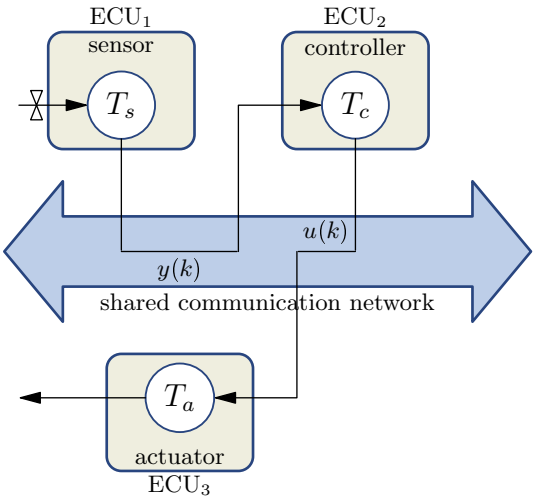


Figure 5.1: Schematic of a cyber-physical control system

output and its desired value is above some threshold e_{th} and if a TT slot is available, the control messages are sent over the TT protocol, as this guarantees an aggressive control action with minimal communication delay. Otherwise, the control messages are sent using the ET protocol. That is,

$$\mathcal{M}_{\text{Bus}}(k) = \begin{cases} \mathcal{M}_{\text{TT}} & \text{if } |e(k)| > e_{\text{th}} \text{ and a TT slot is available.} \\ \mathcal{M}_{\text{ET}} & \text{otherwise} \end{cases} \tag{5.7}$$

That is, the mode of the bus switches depending on the state of the control application, as in (5.7).

A *switching algorithm* is proposed which governs the mode transitions of all the n control applications. For the case of regulation to zero, Algorithm 2 gives a pseudo-code representation of the switching scheme and is described in the following.

In line 1, all necessary variables are initialized with the following values: dwelltime = T_{dw}, stepCount = 0, dwell = **true**, and modeChangeRequest = **false**. Also, each control application C_i is started one by one with its initial values in the operating mode $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{TT}}$ as one cannot assume that the system is in a well-behaved mode when it is started. After this initialization phase, all applications are in the operating mode $\mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{ET}}$ (i.e., the applications are stable and the communication is event-triggered) and the system is ready to run (line 2). In lines 3-27, the actual switching algorithm is presented. If no disturbance occurs the control application stays in the operating mode $\mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{ET}}$ (line 26). Otherwise, it is checked if the system was in the mode $\mathcal{M}_{\text{TT}}$ in the previous step (line 4). Each control application C_i in a subgroup $\mathcal{P}_j$ is assigned a priority. Note that although the access to the time-triggered slot is "arbitrated" by assigning priorities, the main difference is the time delay which is 0 in the time-triggered slot and upper-bounded by h in the event-triggered slot. The switching algorithm allows the application with the highest priority (among all

Algorithm 2 The switching algorithm

```
1: initialize()
2: while system is running do
3:   if system in steady-state or mode change was requested  or system is in TT
     mode (dwell == true) then
4:     if system is in TT mode then /* dwell == true */
5:       if stepCount < dwelltime then
6:         increase step counter
7:         dwell ← true
8:         continue with TT
9:       else
10:        stepCounter ← 0
11:        dwell ← false
12:        continue with ET
13:      end if
14:    else
15:      request mode change
16:      if TT is available then
17:        stepCount ← 1
18:        dwell ← true
19:        modeChangeRequest ← false
20:        continue with TT
21:      else
22:        continue with ET
23:      end if
24:    end if
25:  else
26:    continue with ET
27:  end if
28: end while
```

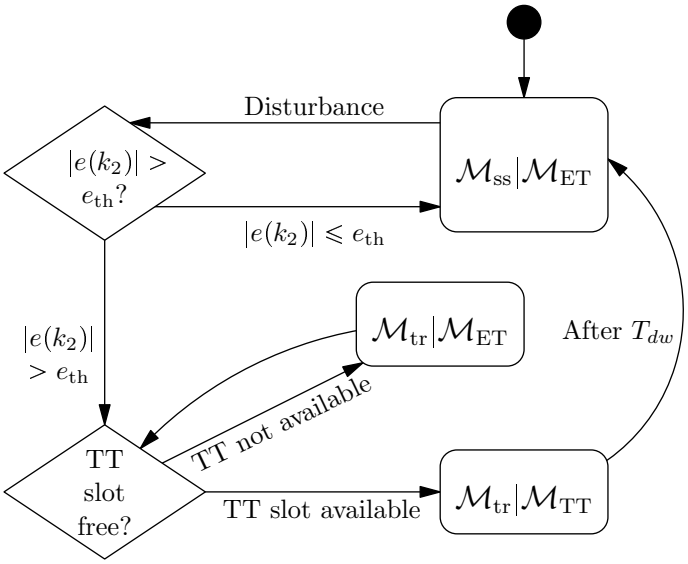


Figure 5.2: The switching algorithm for stabilization

applications in $\mathcal{P}_j$ requesting a mode change) to change its communication mode from $\mathcal{M}_{\text{ET}}$ to $\mathcal{M}_{\text{TT}}$ (lines 16–20). Subsequently, once application $C_i \in \mathcal{P}_j$ is in $\mathcal{M}_{\text{TT}}$, it must reside at $\mathcal{M}_{\text{TT}}$ for at least T_{dw} samples, where T_{dw} is a given dwell time (lines 5–8). During this period, all applications in $\mathcal{P}_j \backslash \{C_i\}$ that requested a mode change are required to wait in the mode $\mathcal{M}_{\text{ET}}$ (line 22). Usually, these parameters T_{dw} are chosen such that they decrease in magnitude with the priority of the application.

In Figure 5.2, the overall switching design with all possible transitions in the stabilization case is illustrated with a flowchart diagram. After initialization, at time k_0, the system is assumed to be in mode $\mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{ET}}$. If the initial conditions are such that at some $k_0' > k_0$, the tracking error $|e(k_0')|$ has a magnitude greater than the threshold e_{th} ($\mathcal{M}_{\text{tr}}$), and if a TT slot is available, the system switches to $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{TT}}$. At some $k_1 > k_0'$, due to the action of a controller, $|e(k_1)|$ falls below the threshold ($\mathcal{M}_{\text{ss}}$), thereby switching to the communication mode $\mathcal{M}_{\text{ET}}$. At a time $k_2 > k_1$, a disturbance is assumed to occur. The system switches to $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{TT}}$ immediately if $|e(k_2)|$ exceeds the threshold and if a TT slot is available, but stays in $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{ET}}$ otherwise. Let $k_2' \geqslant k_2$ be the time at which the system is switched to $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{TT}}$. The same process defined above continues for all $k \geqslant k_2'$. If k_2' does not exist, it implies that the system stays in $\mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{ET}}$ forever.

In Figure 5.3, the overall switching design with all possible transitions in the tracking case is illustrated. The overall switching scheme is similar to the one for stabilization except the initialization. At time k_0, the system is turned on with being in mode $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{TT}}$. After the dwell-time T_{dw} at time k_1, the system switches to mode $\mathcal{M}_{\text{tr}}|\mathcal{M}_{\text{ET}}$ and stays there regardless how large $|e(k_2)|$ becomes. After the dwell-time T_{dw} at time k_2, the system stays in mode $\mathcal{M}_{\text{ss}}|\mathcal{M}_{\text{ET}}$ and is now in normal operating mode. From here on the switching scheme is the same as for the stabilization case.

In what follows, different controllers for when the communication mode is $\mathcal{M}_{\text{TT}}$ and

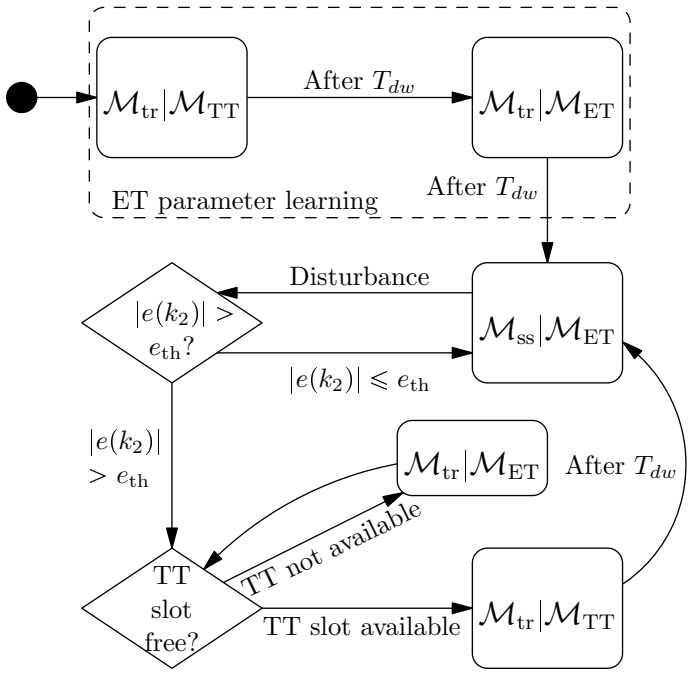


Figure 5.3: The switching algorithm for tracking

$\mathcal{M}_{\text{ET}}$ are designed, and in such a manner that the above switching process is a stable one, and that the algorithm continues to return to $\mathcal{M}_{\text{ET}}$ as soon as possible. This ensures a satisfactory co-design of the controller and the platform, with all closed-loop signals remaining bounded, with a minimal utilization of the TT slots since multiple controllers share a common TT slot.

5.2.4 Controller Design

The switching controller is described in this section. The control input $u(k)$ is chosen as follows: If $\mathcal{M}_{\text{Bus}}(k) = \mathcal{M}_{\text{TT}}$,

$$
\begin{aligned}
u(k) &= \frac{1}{\hat{\theta}_{1,\nu}(k)} \left(y_{\text{ref}}(k+1) - \hat{\vartheta}_1(k)^T \phi_1(k) \right) \\
\nu_1(k) &= y(k) - \hat{\theta}_1(k-1)^T \Phi_1(k-1) \\
\hat{\theta}_1(k) &= \hat{\theta}_1(k-1) + \frac{a(k)\Phi_1(k-1)\nu_1(k)}{1 + \Phi_1(k-1)^T \Phi_1(k-1)} \\
a(k) &= \begin{cases} 1 & \text{if } \nu\text{-th element of right-hand side of update law evaluated using } a(k) = 1 \text{ is } \neq 0 \\ \gamma_1 & \text{otherwise, with } 0 < \gamma_1 < 2, \gamma_1 \neq 1 \end{cases}
\end{aligned}
\tag{5.8}
$$

where $\phi_1(k)$ is given in Equation (2.24), $\Phi_1(k)$ is given in Equation (2.25), $\hat{\theta}_1(k) = \left[\hat{\vartheta}_1(k)^T \;\; \hat{\theta}_{1,\nu}(k)\right]^T$ is the estimation of the controller gains θ_1^* (Equation (2.25)), and $\gamma_1 \in (0, 2)$.

If $\mathcal{M}_{\text{Bus}}(k) = \mathcal{M}_{\text{ET}}$, the control input is given by

$$\begin{aligned}
u(k) &= \frac{1}{\hat{\theta}_{2,\nu}(k)} \left(y_{\text{ref}}(k+d_2) - \hat{\vartheta}_2(k)^T \phi_2(k) \right) \\
\nu_2(k) &= y(k) - \hat{\theta}_2(k-1)^T \Phi_2(k-d_2) \\
\hat{\theta}_2(k) &= \hat{\theta}_2(k-1) + \frac{a(k)\Phi_2(k-d_2)\nu_2(k)}{1+\Phi_2(k-d_2)^T\Phi_2(k-d_2)} \\
a(k) &= \begin{cases} 1 & \text{if } \nu\text{-th element of right-hand side of update law evaluated using } a(k)=1 \text{ is } \neq 0 \\ \gamma_2 & \text{otherwise, with } 0<\gamma_2<2, \gamma_2 \neq 1 \end{cases}
\end{aligned} \tag{5.9}$$

where $\phi_2(k)$ is given in Equation (2.24), $\Phi_2(k)$ is given in Equation (2.25), $\hat{\theta}_2(k) = \left[\hat{\vartheta}_2(k)^T \;\; \hat{\theta}_{2,\nu}(k)\right]^T$ is the estimation of the controller gains θ_2^* (Equation (2.25)), and $\gamma_2 \in (0,2)$.

In the subsequent sections it will be shown that this co-design results in a stable system with bounded states and achieves regulation to zero (Section 5.3) and tracking of an arbitrary reference signal (Section 5.4).

5.3 Stabilization

In this section the first main result of this chapter is presented. The simpler case of regulation to zero, $y_{\text{ref}}(k) \equiv 0 \; \forall k$ is considered first. Global boundedness is stated and proved in Theorem 5.3.1.

The following definitions are useful for the rest of this chapter. The instants of time when the switch from TT to ET occurs are denoted as k_p, $p = 1,3,5,\ldots$, and the instants of time when the switch from ET to TT occurs as k_p, $p = 2,4,6,\ldots$. That is, the TT protocol is applied for $k \in [k'_{2p}; k_{2p+1}], p \in \mathbb{N}_0$ and the ET protocol is applied for $k \in [k'_{2p+1}; k_{2p}], p \in \mathbb{N}_0$ with $k'_p := k_p + 1$ and switches occurring between $[k_p; k'_p], p \in \mathbb{N}$ (see Figure 5.4).

Assumption 5.3.1. The disturbance $D(k)$ in (2.19) is an impulse train, with the distance between any two consecutive impulses greater than a constant T_{dw}.

Theorem 5.3.1. *Let Assumptions 2.3.1 and 5.3.1 be satisfied and $y_{ref}(k) \equiv 0$ for all k. Consider the closed-loop adaptive system given by the plant in (2.19), together with the switching adaptive controller in (5.8) and (5.9) with the hybrid protocol in (5.7) and the following parameter estimate selections at the switching instants*

$$\hat{\theta}_1(k_p) = 0, \qquad p = 0,2,4,\ldots \tag{5.10}$$

$$\hat{\vartheta}_2(k_p - 1) = 0, \qquad p = 1,3,5,\ldots \tag{5.11}$$

$$\hat{\theta}_{2,\nu}(k_p - 1) = 1, \qquad p = 1,3,5,\ldots \tag{5.12}$$

$$\hat{\theta}_2(k_p) = 0, \qquad p = 1,3,5,\ldots. \tag{5.13}$$

Then there exists a positive constant T_{dw}^* such that for all $T_{dw} \geqslant T_{dw}^*$, the closed loop system has globally bounded solutions. That is, the sequences $\{y(k)\}$, $\{u(k)\}$, and the parameter estimation error sequences $\{\tilde{\theta}_1(k)\}$ and $\{\tilde{\theta}_2(k)\}$ are bounded for all k. If there are no disturbances, the ET protocol will stay en vogue and the output $y(k)$ converges to zero.

The outline of the proof of Theorem 5.3.1 is as follows:
First, it is shown in Stage 1 that a switching sequence exists, both from TT to ET and from ET to TT as k increases. Next, it is shown in Stage 2 that the tracking error $e(k)$ is bounded for all k. As in any adaptive system, additional steps have to be used in order to show that the states of the adaptive system are bounded since they depend on the tracking error $e(k)$ and the parameter error $\tilde{\theta}_d(k), d = 1, 2$. In stage 3, the boundedness of the latter is established. This is used, in the final stage 4, to show the system state $\Phi(k)$ and therefore the control input $u(k)$ is bounded.

Noting that the error $e(k)$ defined in (2.40) is related to the parameter estimates and the latter vary with the underlying protocol, the errors $e_1(k)$ and $e_2(k)$ are defined as follows:

$$e(k) := \begin{cases} e_1(k) & \text{if } \mathcal{M}_{\text{TT}} \text{ is active} \\ e_2(k) & \text{if } \mathcal{M}_{\text{ET}} \text{ is active.} \end{cases} \tag{5.14}$$

Proof of Theorem 5.3.1: In what follows, the effect of the impulse disturbances is represented as a jump in the plant output at the instances where the disturbances occur. When the algorithm is in mode $\mathcal{M}_{\text{TT}}$, the underlying error equation, using (2.41), can be derived as

$$\begin{aligned} e_1(k+1) &= \left(\vartheta_1^* - \frac{1}{\hat{\theta}_{1,\nu}(k)} \hat{\vartheta}_1(k) \right)^T \phi_1(k) \\ &= \tilde{\vartheta}_1(k)^T \phi_1(k). \end{aligned} \tag{5.15}$$

since $y_{\text{ref}}(k) = 0$ with $\tilde{\vartheta}_1(k) = \vartheta_1^* - \hat{\vartheta}_1(k)$. When the system is in mode $\mathcal{M}_{\text{ET}}$, the error equation can be derived again using (2.41) as

$$\begin{aligned} e_2(k+2) &= \left(\vartheta_2^* - \frac{1}{\hat{\theta}_{2,\nu}(k)} \hat{\vartheta}_2(k) \right)^T \phi_2(k) \\ &= \tilde{\vartheta}_2(k)^T \phi_2(k). \end{aligned} \tag{5.16}$$

with $\tilde{\vartheta}_2(k) = \vartheta_2^* - \hat{\vartheta}_2(k)$.

Define θ_a^* as

$$\theta_a^* = \begin{cases} \begin{bmatrix} \theta_1^{*T} & 0 \end{bmatrix}^T & \text{if } \mathcal{M}_{\text{Bus}}(k) = \mathcal{M}_{\text{TT}} \\ \theta_2^* & \text{if } \mathcal{M}_{\text{Bus}}(k) = \mathcal{M}_{\text{ET}} \end{cases} \tag{5.17}$$

Choose a Lyapunov function $V(k)$ as

$$V(k) = \tilde{\theta}_a(k)^T \tilde{\theta}_a(k) \tag{5.18}$$

where $\tilde{\theta}_a(k) = \theta_a^* - \hat{\theta}_a(k)$. Let $\Delta V(k) = V(k) - V(k-1)$.

The proof consists of the following four stages:

Stage 1: Let there exist a sequence of finite switching times $\{k_l\}_{l\in\mathbb{N}}$ with the properties described above. Then the errors $e_1(k)$ and $e_2(k)$ are bounded for all k.
The proof of Stage 1 is established using the following three steps:

Step 1-1 There exists a $\Delta \in \mathbb{N}$ such that $\forall \varepsilon \in \,]0; e_{\text{th}}] : \ |e_1(k_1)| < \varepsilon \leqslant e_{\text{th}}$ where $k_1 = k_0 + \Delta + m_1$.

Step 1-2 During $\mathcal{M}_{\text{TT}}$ ($\mathcal{M}_{\text{ET}}$), the error $e_1(k)$ ($e_2(k)$) is bounded.

Step 1-3 There exists a constant $M_1 < \infty$ with $|e_1(k'_p)| \leqslant M_1$, for $p = 2, 4, 6, \ldots$.

Stage 2: The length of the interval $[k'_p; k_{p+1}]$ is greater than 2, i.e., $k_{p+1} - k'_p > 2, p = 1, 3, 5, \ldots$
Stage 2 is established using the following steps:

Step 2-1 $\forall e_{\text{th}} > 0 \ \exists \varepsilon_p \in \,]0; e_{\text{th}}]$ such that $(|e_1(k_p)| \leqslant \varepsilon_p \Rightarrow |e_2(k'_p)| \leqslant e_{\text{th}})$ for $p = 1, 3, 5, \ldots$

Step 2-2 $|e_2(k'_p + 1)| \leqslant e_{\text{th}}$ for $p = 1, 3, 5, \ldots$

Stage 3: $V(k)$ is bounded for all $k \in \mathbb{N}_0$.
The following steps will be used to establish Stage 3:

Step 3-1 $\Delta V(k'_p) \leqslant M_2 < \infty$ and $\Delta V(k'_{p+1}) \leqslant M_3 < \infty$, for all $p \in \mathbb{N}_0$.

Step 3-2 $\Delta V(k) \leqslant 0$ during $\mathcal{M}_{\text{TT}}$ and during $\mathcal{M}_{\text{ET}}$

Stage 4: The control input is bounded for all k and hence all signals are bounded.
The following two steps will be used to prove Stage 4:

Step 4-1 $|u(k)| \leqslant M_4 \quad \forall k$

Step 4-2 All signals are bounded.

In the following, a detailed proof is provided for each of the above steps.

Stage 1: In this stage it is shown that the sequence $\{k_p\}, p = 0, 1, \ldots$ exists. That the sequence $\{k_i\}$ exists for $i = 2, 4, \ldots$ is immediate since disturbances occur persistently and since TT slots will become available within a finite wait time. In what follows it is shown that this sequence exists for $i = 1, 3, \ldots$. That is, the algorithm periodically switches to ET. This stage consists of three steps.
Step 1-1: $\forall e_{th} > 0 \ \exists \varepsilon \in \,]0; e_{th}]$ *such that* $(|e_1(k_p)| \leqslant \varepsilon \Rightarrow |e_2(k'_p)| \leqslant e_{th})$ *for* $p = 1, 3, 5, \ldots$

Proof of Step 1-1: By Step 1-1, it is known that for all $\varepsilon \leqslant e_{\text{th}}$ there exists a k_1 such that $|e_1(k_1)| < \varepsilon \leqslant e_{\text{th}}$. Now, it needs to be ensured that the jump in the output error is no larger than e_{th} and hence the ET protocol stays en vogue. This is established by showing that (i) there exists a $\varepsilon < e_{\text{th}}$ such that when $|e_1(k_1)| \leqslant \varepsilon$ then $|e_2(k'_1)| \leqslant e_{\text{th}}$ and (ii) $|e_2(k'_p)| \leqslant e_{\text{th}}$ for $p = 3, 5, \ldots$.

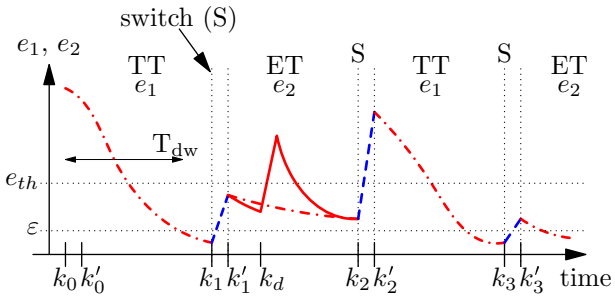


Figure 5.4: Schematic evolution of the error with a given sequence of switching times. Impulses in $D(k)$ are assumed to occur at $k_p, p = 0, 2, 4, \ldots$. Two scenarios are depicted when a disturbance arrives at k_d in the interval $[k_1', k_2]$: (i) TT slot is available (dashed line) at k_d, and (ii) TT slot is not available (solid line) at k_d but at k_2. The dwell time T_{dw} is determined by the settling time of the system with the TT controller.

Consider the switch from TT to ET at time k_p', $p = 1, 3, 5, \ldots$. Then,

$$\left|e_2(k_p')\right| = \left|\tilde{\vartheta}_2(k_p-1)^T\phi_2(k_p-1)\right| = \left|\left(\vartheta_2^* - \frac{1}{\hat{\theta}_{2,\nu}(k)}\hat{\vartheta}_2(k_p-1)\right)^T\phi_2(k_p-1)\right|. \quad (5.19)$$

Choosing $\hat{\vartheta}_2(k_p-1)$ according to (5.11) and $\hat{\theta}_{2,\nu}$ according to (5.12), it follows that

$$\left|e_2(k_p')\right| \leqslant \sum_{i=0}^{m_1-1}|\alpha_i^2||y(k_p-1-i)| + \sum_{i=1}^{m_2+1}|\beta_i^2||u(k_p-1-i)|. \quad (5.20)$$

Replacing $|u(k_p-1-i)|$ by (5.8), using (5.29), and noting that

$$\left|\hat{\vartheta}_1(k_p-1-i)^T\phi_1(k_p-1-i)\right| \leqslant |y(k_p-i)| + |\nu_1(k_p-i)| \leqslant 2\varepsilon, \quad (5.21)$$

according to Theorem 2.3.1(ii), $i = 1, \ldots, m_2+1$, it follows that

$$\left|e_2(k_p')\right| \leqslant \varepsilon\left(\sum_{i=0}^{m_1-1}|\alpha_i^2| + 2\sum_{i=1}^{m_2+1}|\beta_i^2|\right) \leqslant M_0\varepsilon \quad (5.22)$$

as the second term in the last equation in (5.22) is constant and thus smaller than some constant M_0.

For $\varepsilon = \dfrac{e_{\text{th}}}{1+M_0}$, it follows that

$$\left|e_2(k_p')\right| \leqslant \frac{M_0}{1+M_0}e_{\text{th}} \leqslant e_{\text{th}}. \quad (5.23)$$

Step 1-2: $|e_2(k_p'+1)| \leqslant e_{th}$ *for* $p = 1, 3, 5, \ldots$

Proof of Step 1-2: Using (5.13), the error $e_2(k_p'+1) = e_2(k_p+2)$ is bounded by

$$|e_2(k_p+2)| \leqslant \sum_{i=0}^{m_1-1}|\alpha_i^2||y(k_p-i)| + \sum_{i=1}^{m_2+1}|\beta_i^2||\hat{\vartheta}_1(k_p-i)^T\phi_1(k_p-i)| \quad (5.24)$$

Noting that $|y(k_p - i)|, i = 0, \ldots, m_1 - 1$ is smaller than ε and that

$$\left|\hat{\vartheta}_1(k_p - i)^T \phi_1(k_p - i)\right| \leqslant |y(k_p + 1 - i)| + |\nu_1(k_p + 1 - i)| \leqslant 2\varepsilon, \tag{5.25}$$

$i = 1, \ldots, m_2 + 1$, it follows that

$$|e_2(k'_p)| \leqslant \varepsilon \left(\sum_{i=0}^{m_1-1} |\alpha_i^2| + 2 \sum_{i=1}^{m_2+1} |\beta_i^2| \right). \tag{5.26}$$

With the same ε and M_0 as in Step 2.1, it follows that

$$|e_2(k_p + 2)| \leqslant \frac{M_0}{1 + M_0} e_{\text{th}} \leqslant e_{\text{th}} \tag{5.27}$$

at $k = k_p$, $|e_2(k)| \leqslant e_{\text{th}}$ for at least $k \in [k'_p, k'_p + 1]$.

Step 1-3: *$|e_2(k)| \leqslant \min(e_{th}, M(e_{th}, D_{\max}))$ for $k \in [k'_p; k_{p+1}]$, $p = 1, 3, 5, \ldots$ where $M(D_{\max})$ is a finite constant depending on $D_{\max}$.*

Proof of Step 1-3: Assume a disturbance occurs at time instant $k_d \in [k'_p + 1; k_{p+1}], p = 1, 3, 5, \ldots$. Two cases have to be considered here: The disturbance is such that (i) $|e_2(k)| \leqslant e_{\text{th}}, k \in [k_d, k_{p+1}]$, (ii) $|e_2(k)| \geqslant e_{\text{th}}, k \in [k_d, k_{p+1}]$.

If Case (i) occurs, then $|e_2(\bar{k})| \leqslant e_{\text{th}}$ is trivially true. If Case (ii) occurs it implies that no TT slot is available for $k \in [k_d, k_{p+1}]$. In this case, it will be shown that $|e_2(k)| \leqslant M_0 D_{\max}, k \in [\bar{k}, k_{p+1}]$ where M_0 is a finite constant. Since the ET protocol is en vogue for $[k_d, k_{p+1}]$, Theorem 2.3.1(i) implies that there exists a finite $M > 0$ such that $|e_2(\bar{k})| \leqslant M D_{\max}$, and $|e_2(k)| \leqslant M_0 D_{\max}$ with $M_0 = \gamma M$ where γ is a finite constant, which proves Case (ii).

Stage 2: In this stage it is shown that the tracking error $e_1(t)$ is bounded during $\mathcal{M}_{\text{TT}}$ and $e_2(t)$ is bounded during $\mathcal{M}_{\text{ET}}$. This stage consists of three steps.

Step 2-1: *There exists a $\Delta \in \mathbb{N}$ such that $\forall \varepsilon \in]0; e_{th}]$: $|e_1(k_1)| < \varepsilon \leqslant e_{th}$ where $k_1 = k_0 + \Delta + m_1$.*

Proof of Step 2-1: Let $k = k_0$ when an impulse disturbance occurs, and the system is in mode $\mathcal{M}_{\text{TT}}$. It follows that $|e_1(k_0)| = y(k_0) + D_{\max}$ where $D_{\max}$ is an upper bound on $D(k)$. Since $D(k) \equiv 0$ for $k \leqslant k_0 + T^*_{dw}$, Theorem 2.3.1 implies that there exists a $\Delta_{\text{TT}} \in \mathbb{N}$ such that for $k \geqslant k_0 + \Delta_{\text{TT}}$, $|e_1(k)| < \varepsilon \leqslant e_{\text{th}}$. Choose $k_1 = k_0 + T^*_{dw}$, with

$$T^*_{dw} = \Delta_{\text{TT}} + m_1. \tag{5.28}$$

Then, it follows that no disturbance can occur between k_0 and k_1, and

$$|e_1(k_1 - i)| < \varepsilon \leqslant e_{\text{th}} \forall 0 \leqslant i \leqslant m_1. \tag{5.29}$$

Note that the same arguments as above can be applied for the interval $[k_{2p}, k_{2p+1}], p = 1, 2, \ldots$. For $k \in [k_1 + 1; k_2]$ the ET protocol with controller (5.9) is applied.

Step 2-2: *During $\mathcal{M}_{TT}$ ($\mathcal{M}_{ET}$), the error $e_1(k)$ ($e_2(k)$) is bounded.*

Proof of Step 2-2: Theorem 2.3.1 ensures that during $\mathcal{M}_{\mathrm{TT}}$ ($\mathcal{M}_{\mathrm{ET}}$), the sequence $\{y(k)\}$ is bounded. Hence, the error $e_1(k)$ ($e_2(k)$) is also bounded during $\mathcal{M}_{\mathrm{TT}}$ ($\mathcal{M}_{\mathrm{ET}}$). $e_2(k)$ ($e_1(k)$), by definition, is bounded during $\mathcal{M}_{\mathrm{TT}}$ ($\mathcal{M}_{\mathrm{ET}}$).

Step 2-3: *There exists a constant* $M_1 < \infty$ *with* $|e_1(k'_p)| \leqslant M_1$, *for* $p = 2, 4, 6, \ldots$.

Proof of Step 2-3: Let $M_1 = e_{\mathrm{th}} + D_{\max}$. Then, since $|e_1(k_{2p})| \leqslant e_{\mathrm{th}}$ by definition, it follows that

$$|e_1(k'_{2p})| \leqslant M_1 \quad \forall p \in \mathbb{N}. \tag{5.30}$$

Stage 3: The boundedness of the parameter error is shown in this stage during $\mathcal{M}_{\mathrm{TT}}$, $\mathcal{M}_{\mathrm{ET}}$ and all switching instants. This stages consists of two steps.

Step 3-1: $\Delta V(k'_p) \leqslant M_2 < \infty$, $p = 1, 3, \ldots$ *and* $\Delta V(k'_p) \leqslant M_3 < \infty$, $p = 2, 4, \ldots$.

Proof of Step 3-1: At $k = k'_1$, $\Delta V(k'_1)$ is given by

$$\begin{aligned}
\Delta V(k'_1) &= V(k_1 + 1) - V(k_1) && (5.31)\\
&= \tilde{\theta}_a(k_1 + 1)^T \tilde{\theta}_a(k_1 + 1) - \tilde{\theta}_a(k_1)^T \tilde{\theta}_a(k_1) && (5.32)\\
&= \tilde{\theta}_2(k_1 + 1)^T \tilde{\theta}_2(k_1 + 1) - \tilde{\theta}_1(k_1)^T \tilde{\theta}_1(k_1) && (5.33)\\
&\leqslant \tilde{\theta}_2(k_1)^T \tilde{\theta}_2(k_1) \\
&\quad + \frac{\gamma_2 \left(\tilde{\theta}_2(k_1)^T \Phi_2(k_1 - d_2 + 1)\right)^2}{1 + \Phi_2(k_1 - d_2 + 1)^T \Phi_2(k_1 - d_2 + 1)} \\
&\quad \times \left(\frac{\gamma_2 \Phi_2(k_1 - d_2 + 1)^T \Phi_2(k_1 - d_2 + 1)}{(1 + \Phi_2(k_1 - d_2 + 1)^T \Phi_2(k_1 - d_2 + 1))} - 2 \right). && (5.34)
\end{aligned}$$

Noting that $\hat{\theta}_2(k_1) = 0$ according to (5.10) and that the last term in (5.34) is negative as $0 < \gamma_2 < 2$, it follows that

$$\Delta V(k'_1) \leqslant \theta_2^{*T} \theta_2^* \leqslant M_2 \leqslant \infty \tag{5.35}$$

where M_2 is a finite constant.

Let $p = 3, 5, 7 \ldots$. It can be shown that

$$\begin{aligned}
\Delta V(k'_p) &= V(k_p + 1) - V(k_p) && (5.36)\\
&\leqslant \tilde{\theta}_2(k_p)^T \tilde{\theta}_2(k_p) \\
&\quad + \frac{\gamma_2 \left(\tilde{\theta}_2(k_p)^T \Phi_2(k_p - d_2 + 1)\right)^2}{1 + \Phi_2(k_p - d_2 + 1)^T \Phi_2(k_p - d_2 + 1)} \\
&\quad \times \left(\frac{\gamma_2 \Phi_2(k_p - d_2 + 1)^T \Phi_2(k_p - d_2 + 1)}{(1 + \Phi_2(k_p - d_2 + 1)^T \Phi_2(k_p - d_2 + 1))} - 2 \right). && (5.37)
\end{aligned}$$

Since the last term in (5.37) is negative, it follows that

$$\Delta V(k'_p) \leqslant \tilde{\theta}_2(k_p)^T \tilde{\theta}_2(k_p) = \left(\theta_2^* - \hat{\theta}_2(k_p)\right)^T \left(\theta_2^* - \hat{\theta}_2(k_p)\right). \tag{5.38}$$

According to (5.13), $\hat{\theta}_2(k_p) = \hat{\theta}_2(k_{p-1})$, and thus

$$\Delta V(k_p') \leqslant \tilde{\theta}_2(k_{p-1})^T \tilde{\theta}_2(k_{p-1}) \leqslant M_2 \leqslant \infty \tag{5.39}$$

since $\tilde{\theta}_2(k_{p-1})$ is bounded by Theorem 2.3.1 and where M_2 is a finite constant. Hence, the jump of V at $k_p, p = 3, 5, 7, \ldots$, from TT to ET, is bounded.

Let $p = 2, 4, 6, \ldots$. At k_p', when the controller switches from ET to TT

$$\begin{aligned}
\Delta V(k_p') &= V(k_p + 1) - V(k_p) \\
&= \tilde{\theta}_a(k_p + 1)^T \tilde{\theta}_a(k_p + 1) - \tilde{\theta}_a(k_p)^T \tilde{\theta}_a(k_p) \\
&= \tilde{\theta}_1(k_p + 1)^T \tilde{\theta}_1(k_p + 1) - \tilde{\theta}_2(k_p)^T \tilde{\theta}_2(k_p) \\
&\leqslant \tilde{\theta}_1(k_p)^T \tilde{\theta}_1(k_p) \\
&\quad - 2\gamma_1 \frac{\tilde{\theta}_1(k_p)^T \Phi_1(k_p) \tilde{\theta}_1(k_p)^T \Phi_1(k_p)}{1 + \Phi_1(k_p)^T \Phi_1(k_p)} \\
&\quad + \frac{\gamma_1^2 \left(\tilde{\theta}_1^T(k_p) \Phi_2(k_p) \right)^2 \Phi_1(k_p)^T \Phi_1(k_p)}{\left(1 + \Phi_1(k_p)^T \Phi_1(k_p)\right)^2}.
\end{aligned} \tag{5.40}$$

Note that the ET protocol is not active at time k_p, and $\hat{\theta}_1(k_p) = 0$ according to (5.11). Thus,

$$\begin{aligned}
\Delta V(k_p') &= V(k_p + 1) - V(k_p) \\
&\leqslant \theta_1^{*T} \theta_1^* - 2\gamma_1 \frac{\theta_1^{*T} \Phi_1(k_p) \theta_1^{*T} \Phi_1(k_p)}{1 + \Phi_1(k_p)^T \Phi_1(k_p)} + \frac{\gamma_1^2 \left(\theta_1^{*T} \Phi_1(k_p) \right)^2 \Phi_1(k_p)^T \Phi_1(k_p)}{\left(1 + \Phi_1(k_p)^T \Phi_1(k_p)\right)^2}.
\end{aligned} \tag{5.41}$$

Any disturbance occurring at k_p' is bounded by $D_{\max}$ and hence, when the TT algorithm is employed $|\theta_1^{*T} \Phi_1(k_2)| \leqslant |y(k_2 + 1)| + D_{\max}$. Then,

$$\begin{aligned}
\Delta V(k_p') &\leqslant \theta_1^{*T} \theta_1^* + \frac{\gamma_1^2 \left(|y(k_p + 1)| + D_{\max} \right)^2 \Phi_1(k_p)^T \Phi_1(k_p)}{\left(1 + \Phi_1(k_p)^T \Phi_1(k_p)\right)^2} \\
&\leqslant \theta_1^{*T} \theta_1^* + \frac{\gamma_1^2 \left(|y(k_p + 1)| + D_{\max} \right)^2}{1 + \Phi_1(k_p)^T \Phi_1(k_p)} \\
&\leqslant \theta_1^{*T} \theta_1^* + \frac{\gamma_1^2 \left(e_{\text{th}} + D_{\max} \right)^2}{1 + \Phi_1(k_p)^T \Phi_1(k_p)} \leqslant M_3 \leqslant \infty
\end{aligned} \tag{5.42}$$

where the fact is used that $|y(k_2 + 1)| \leqslant e_{\text{th}}$ which follows since $y(k_2 + 1)$ is computed before the disturbance occurs and M_3 is a finite constant. The above implies that the jumps at k_p and $k_{p+1}, \forall p \in \mathbb{N}$, corresponding to TT to ET and ET to TT, respectively, are bounded.

Step 3-2: $\Delta V(k) \leqslant 0$ *during* $\mathcal{M}_{TT}$ *and during* $\mathcal{M}_{ET}$

Proof of Step 3-2: For $k \in [k'_p; k_{p+1}], p = 0, 2, 4, \ldots$ and $d = d_1 = 1$ and for $k \in [k'_p; k_{p+1}], p = 1, 3, 5, \ldots$ and $d = d_2$, $\Delta V(k)$ is given by

$$\begin{aligned}
\Delta V(k) &= \tilde{\theta}_d(k)^T\tilde{\theta}_d(k) - \tilde{\theta}_d(k-1)^T\tilde{\theta}_d(k-1) \\
&= \left(\tilde{\theta}_d(k-1) - \frac{\gamma_d\Phi_d(k-d)\tilde{\theta}_d(k-1)^T\Phi_d(k-d)}{1+\Phi_d(k-d)^T\Phi_d(k-d)}\right)^T \\
&\quad \times \left(\tilde{\theta}_d(k-1) - \frac{\gamma_d\Phi_d(k-d)\tilde{\theta}_d(k-1)^T\Phi_d(k-d)}{1+\Phi_d(k-d)^T\Phi_d(k-d)}\right) \\
&\quad - \tilde{\theta}_d(k-1)^T\tilde{\theta}_d(k-1) \\
&= \frac{\gamma_d^2\left(\tilde{\theta}_d(k-1)^T\Phi_d(k-d)\right)^2\Phi_d(k-d)^T\Phi_d(k-d)}{\left(1+\Phi_d(k-d)^T\Phi_d(k-d)\right)^2} \\
&\quad - \frac{2\gamma_d\tilde{\theta}_d(k-1)^T\Phi_d(k-d)\tilde{\theta}_d(k-1)^T\Phi_d(k-d)}{1+\Phi_d(k-d)^T\Phi_d(k-d)} \\
&= \frac{\gamma_d\nu_d(k)^2}{1+\Phi_d(k-d)^T\Phi_d(k-d)}\left(\frac{\gamma_d\Phi_d(k-d)^T\Phi_d(k-d)}{1+\Phi_d(k-d)^T\Phi_d(k-d)} - 2\right) \leqslant 0
\end{aligned} \tag{5.43}$$

with $0 < \gamma_d < 2$. This shows that the parameter estimation error decreases during $\mathcal{M}_{\mathrm{TT}}$ and $\mathcal{M}_{\mathrm{ET}}$.

Stage 4: Finally, in this stage boundedness of the system state is shown and therefore of the control input. All previous stages are used to conclude the boundedness of all signals of the switched adaptive system. This stage consists of two steps.

Step 4-1: $|u(k)| \leqslant M_4 \quad \forall k$

Proof of Step 4-1: It will be shown that if in the interval $[k'_{p-1}; k_p]$ the boundedness condition $|u(k)| \leqslant \gamma_{p-1}Q_{p-1}$ is satisfied, then $|u(k)| \leqslant \gamma_p Q_p$ for $p = 1, 2, 3, \ldots$ where all γ_p, Q_p are finite constants for all $p \in \mathbb{N}$.

Let $p = 1$. As all initial conditions (2.26) are bounded, i.e., $|u(k'_0)| \leqslant Q_0$, it follows from Theorem 2.3.1 that there exists a constant γ_0 with $0 < \gamma_0 < \infty$ such that

$$|u(k)| \leqslant \gamma_0 Q_0 \tag{5.44}$$

for $k \in [k'_0; k_1]$. Consider the switch from TT to ET at k'_1. $|u(k'_1)|$ is then computed by

$$\begin{aligned}
|u(k'_1)| &= \left|\frac{1}{\hat{\theta}_{2,\nu}(k'_1)}\left(y_{\mathrm{ref}}(k'_1+d_2) - \hat{\vartheta}_2(k'_1)^T\phi_2(k'_1)\right)\right| \\
&= \left|\frac{1}{\hat{\theta}_{2,\nu}(k'_1)}\left(y_{\mathrm{ref}}(k'_1+d_2) - \hat{\vartheta}_2(k'_1)^T\begin{bmatrix} y(k_1+1) \\ \vdots \\ y(k_1-m_1+2) \\ u(k_1) \\ \vdots \\ u(k_1-m_2-d_2+2)\end{bmatrix}\right)\right| \leqslant Q_1
\end{aligned} \tag{5.45}$$

where Q_1 is a finite constant since $\hat{\vartheta}_2(k_1')$ and $\hat{\theta}_{2,\nu}(k_1')$ are bounded by Step 3-2, $u(k_1), \ldots, u(k_1 - m_2 - d_2 + 2)$ are bounded by $\gamma_0 M_0$ (Equation (5.44)), and $|y(k_1')|, \ldots, |y(k_1' - m_1 + 2)|$ are less than e_{th}. Hence, the interval $[k_1'; k_2]$ begins with bounded values and thus it follows from Theorem 2.3.1 that there exists a γ_1 such that

$$|u(k)| \leqslant \gamma_1 Q_1 \tag{5.46}$$

for $k \in [k_1'; k_2]$. Consider the switch from ET to TT at k_2'. $|u(k_2')|$ is then computed by

$$\begin{aligned} |u(k_2')| &= \left| \frac{1}{\hat{\theta}_{1,\nu}(k_2')} \left(y_{\text{ref}}(k_2' + 1) - \hat{\vartheta}_1(k_2')^T \phi_1(k_2') \right) \right| \\ &= \left| \frac{1}{\hat{\theta}_{1,\nu}(k_2')} \left(y_{\text{ref}}(k_2' + 1) - \hat{\vartheta}_1(k_2')^T \begin{bmatrix} y(k_2 + 1) \\ \vdots \\ y(k_2 - m_1 + 2) \\ u(k_2) \\ \vdots \\ u(k_2 - m_2 + 1) \end{bmatrix} \right) \right|. \end{aligned} \tag{5.47}$$

Since $\hat{\vartheta}_2(k_2')$ and $\hat{\theta}_{2,\nu}(k_2')$ are bounded by Step 3-2, $u(k_2), \ldots, u(k_2 - m_2 + 1)$ is bounded by $\gamma_1 M_1$ (Equation (5.46)), and $|y(k_2')|, \ldots, |y(k_2 - m_1 + 2)|$ are less than e_{th} it follows that $|u(k_2')| \leqslant Q_2$ where Q_2 is a bounded function of e_{th}, γ_0, and Q_0. Using the same arguments as above it can be concluded that there exist finite constants γ_3 and Q_3 such that

$$|u(k)| \leqslant \gamma_3 Q_3 \tag{5.48}$$

for $k \in [k_2'; k_3]$ and finite constants γ_4 and Q_4 such that

$$|u(k)| \leqslant \gamma_4 Q_4 \tag{5.49}$$

for $k \in [k_3'; k_4]$, where $\gamma_i, Q_i, i = 3, 4$ are bounded functions of e_{th} and $\gamma_j, Q_j, j = 1, 2$. The rest follows by induction. Assume that

$$|u(k)| \leqslant \gamma_p Q_p \tag{5.50}$$

for $k \in [k_{p-1}'; k_p]$ with finite constants γ_p and Q_p. Either the ET or the TT protocol is used during $[k_{p-1}'; k_p]$. In case of the former it can be concluded using the same arguments as for the interval $[k_2'; k_3]$ that there exist finite constants γ_{p+1} and Q_{p+1} such that

$$|u(k)| \leqslant \gamma_{p+1} Q_{p+1} \tag{5.51}$$

for $k \in [k_p'; k_{p+1}]$. In case of the latter it can be concluded using the same arguments as for the interval $[k_0'; k_1]$ that there exist finite constants γ_{p+1} and Q_{p+1} such that

$$|u(k)| \leqslant \gamma_{p+1} Q_{p+1} \tag{5.52}$$

for $k \in [k'_p; k_{p+1}]$. In all of the above, γ_{p+1} and Q_{p+1} are analytic functions of $\gamma_j, j \leqslant p$. Hence, there exist finite constants γ_p and $Q_p, p = 0, 1, 2, 3, \ldots$ such that

$$|u(k)| \leqslant \gamma_p Q_p \tag{5.53}$$

for all k. Since all γ_p and Q_p are finite there exists a constant $M_4 = \max_{p=0,1,2,3,\ldots}\{\gamma_p Q_p\}$ such that $u(k) \leqslant M_4 \; \forall k$.

Step 4-2: *All signals are bounded.*

Proof of Step 4-2: In Steps 1-2 to 2-2, it is shown that the error $e(k)$ is bounded for all k and hence the output $y(k)$ is also bounded for all k. In Step 4-1 it is shown that $u(k)$ is bounded for all k. Thus, $\Phi_1(k)$ and $\Phi_2(k)$ are bounded for all k. In Steps 3-1 and 3-2 it is shown that the parameter estimation error $\tilde{\theta}(k)$ is bounded for all k. Hence, also the parameter estimation $\hat{\theta}(k)$ is bounded for all k. Overall, all signals are bounded and the theorem is proved if a T^*_{dw} exists as in (5.28). ■

Remark 5.3.1. Theorem 5.3.1 implies that the plant in (2.19) can be guaranteed to have bounded solutions with the proposed adaptive switching controller in (5.8) and (5.9) and the hybrid protocol in (5.7), in the presence of disturbances. The latter is assumed to consist of impulse-trains, with their inter-arrival lower bounded. Note that if no disturbances occur, then the choice of the algorithm in (5.7) implies that these switches cease to exist, and the event-triggered protocol continues to be applied. And switching continues to occur with the onset of disturbances, with Theorem 5.3.1 guaranteeing that all signals remain bounded with the tracking errors e_1 and e_2 converging to e_{th} before the next disturbance occurs.

Remark 5.3.2. The nature of the proof is similar to that of all switching systems. Multiple Lyapunov functions $V(k)$ defined by (5.18) together with the switches as in (5.17) were chosen, and shown to have a finite jump during switches and to decrease in between switches (in Stage 3). Unlike linear switching controllers, in the adaptive controller, these Lyapunov functions only ensure the boundedness of parameter estimates, which are a part of the state of the overall system. The remaining states were shown to be bounded using the boundedness of the tracking errors e_1 and e_2 (in Stage 1) and the parameter estimates (in Stage 3), and the control input using the method of induction (in Stage 4). Since the switching instants themselves were functions of the states of the closed-loop system, it was necessary to show that indeed these switching sequences exist, which was demonstrated in Stage 2. It is the latter that distinguishes the adaptive controller proposed in this chapter as well as the methodology used for the proof from existing adaptive switching controllers and their proofs in the literature.

5.4 Tracking

In this section Theorem 5.3.1 is extended to the tracking case. Since in this case $y_{\text{ref}}(k) \neq 0$, as adaptation proceeds, certain amount of parameter adaptation takes

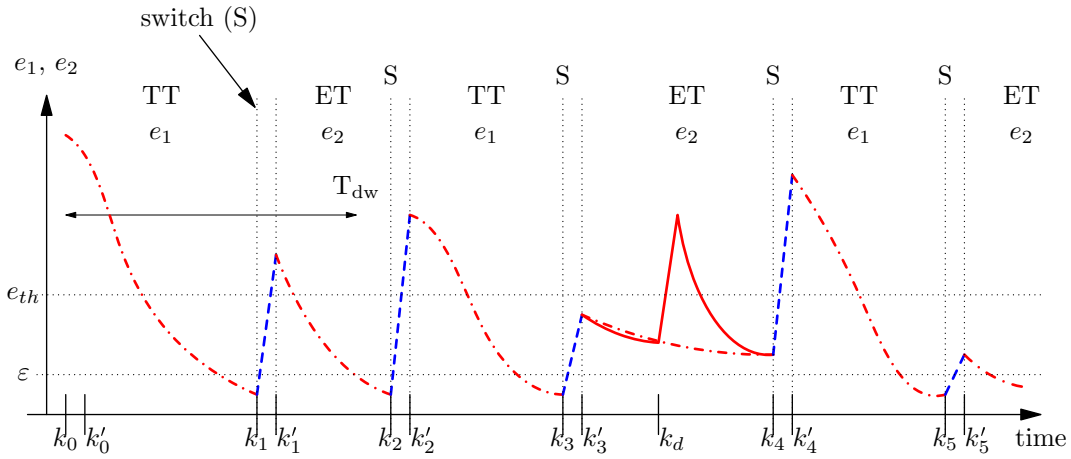


Figure 5.5: Schematic evolution of the error with a given sequence of switching times. Impulses in $D(k)$ are assumed to occur at $k_p, p = 0, 2, 4, \ldots$. When the system switches to ET at k_1', the tracking error might exceed e_{th}. This is not followed by a switch to $\mathcal{M}_{\text{TT}}$ as the ET controller needs some time to learn the parameter values. It is assumed that no disturbance occurs in the interval $[k_0, k_0 + 2T_{dw}]$. Two scenarios are depicted when a disturbance arrives at k_d in the interval $[k_1', k_2]$: (i) TT slot is available (dashed line) at k_d, and (ii) TT slot is not available (solid line) at k_d but at k_4. The dwell time T_{dw} is determined by the settling time of the system with the TT controller and the ET controller.

place depending on persistent excitation properties of the external signal, which in turn affects the magnitude of the tracking error. In order to exploit these properties, the way in which the parameter estimates at switching instants are chosen is changed in the control design. That is, the parameter estimates are set $\hat{\theta}_2(k_1) = 0$ in ET, and subsequent parameter estimates $k_p, p = 3, 5, 7, \ldots$, the beginning of each ET phase, are initialized to the values at $k_{p-1}, p = 3, 5, 7, \ldots$, the end of the previous ET phase.

The main difference between Theorem 5.3.1 and the following Theorem 5.4.1 is the choice of parameter estimates at the switching instants. In Theorem 5.3.1, all parameter estimates are set to zero at the switching instants (Equations (5.10)-(5.13)) and thus information is lost. In Theorem 5.4.1 however, the parameter estimates from the conclusion of previous ET phase are used as initial values of the parameter estimates in the next ET phase. This and the fact that $y_{\text{ref}}(k)$ is no longer equal to zero make it necessary, in contrast to Theorem 5.3.1, to use persistent excitation properties stated in Section 2.3.2. Specifically, Assumption 2.3.2 and Theorem 2.3.5(ii)-(iii) are used in the proof of Theorem 5.4.1.

Theorem 5.4.1. *Let the plant, disturbance D and $y_{ref}(k)$ satisfy Assumptions 2.3.1-2.3.2. Consider the switching adaptive controller in (5.8) and (5.9) with the hybrid protocol in (5.7) and the following parameter estimate selections at the switching in-*

stants

$$\hat{\theta}_1(k_p) = 0, \qquad p = 0, 2, 4, \ldots \tag{5.54}$$

$$\hat{\theta}_2(k_1) = 0 \tag{5.55}$$

$$\hat{\theta}_2(k_p + l) = \hat{\theta}_2(k_{p-1}), \qquad p = 3, 5, 7, \ldots, \tag{5.56}$$

$$l = 0, 1, \ldots, m_2 + d - 1. \tag{5.57}$$

Then there exists a positive constant T_{dw}^* such that for all $T_{dw} \geqslant T_{dw}^*$, the closed-loop system has globally bounded solutions if the switching instants $k_1 = k_0 + T_{dw}$ and $k_2 \geqslant k_0 + 2T_{dw}$.

Proof of Theorem 5.4.1: First, an equivalent reference signal y'_{ref} that combines the effect of both $y_{\text{ref}}(k)$ and the disturbance D is defined as

$$y'_{\text{ref}}(k) := y_{\text{ref}}(k) + D'(k) \tag{5.58}$$

where $D'(k)$ is given by

$$D'(k) := G^{-1}(q^{-1})D(k). \tag{5.59}$$

and $G(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})}$ is the transfer function of the plant (2.17). Also, a reference model signal ϕ_i^* is defined as

$$\phi_i^*(q^{-1}) = G_i(1 + \vartheta_i^{*T} G_i)^{-1} y'_{\text{ref}} \tag{5.60}$$

where the transfer functions $G_i, i = 1, 2$ is given by

$$G_i(q^{-1}) = \begin{bmatrix} W(q^{-1})q^{-1} \\ \vdots \\ W(q^{-1})q^{-m_1+1} \\ q^{-1} \\ \vdots \\ q^{-m_2-d_i+1} \end{bmatrix} \tag{5.61}$$

and the optimal feedback gain $\vartheta_i^*, i = 1, 2$ is given by (2.24). The overall ideal closed-loop system is given by the block diagram shown in Figure 5.6. Note that when there is no disturbance, the output ϕ_i^* corresponds to the desired regressor vector, and its first element of the vector corresponds to $y_{\text{ref}}(k)$.

The proof is divided into the same steps as the one of Theorem 5.3.1 except for Stage 2 which differs significantly due to $y_{\text{ref}}(k) \neq 0$. Stage 2 consists of the following steps:

Stage 2: The length of the interval $[k'_p; k_{p+1}]$ is greater than 2, i.e., $k_{p+1} - k'_p > 2, p = 1, 3, 5, \ldots$
Stage 2 is established using the following steps:

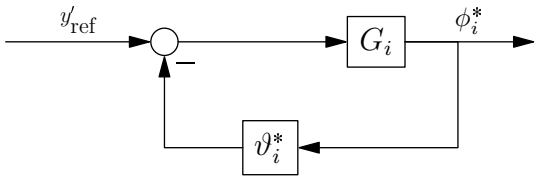


Figure 5.6: Block diagram of the reference model for $i = 1, 2$

Step 2-1 If $\lim_{k\to\infty} |y(k) - y_{\text{ref}}(k)| = 0$ then $\exists \bar{k}$ such that for $k > \bar{k} : \|\phi_i(k) - \phi_i^*(k)\| \leqslant \delta$

Step 2-2 If $|y(k) - y_{\text{ref}}(k)| \to 0$, then $\phi_i(k_j) \in \Omega_{p,i}, i = 1, 2, j = 1, 2, 3, \ldots$

Step 2-3 $|e_2(k'_p)| = |\tilde{\vartheta}_2(k_{p-1})^T \phi_2(k'_p - d_2) + \tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k'_p)| \leqslant e_{\text{th}}$ for $p = 3, 5, 7, \ldots$

Step 2-4 $|e_2(k'_p + l)| \leqslant e_{\text{th}}, p = 3, 5, 7, \ldots$ and $l = 1, 2, \ldots, m_2 + d_2 - 1$

A detailed proof of Steps 2-1 to 2-4 is provided below.

Step 2-1: $\lim_{k\to\infty} |y(k) - y_{\text{ref}}(k)| = 0$ *then* $\exists \bar{k}$ *such that for* $k > \bar{k} : \|\phi_i(k) - \phi_i^*(k)\| \leqslant \delta$

Proof of Step 2-1: In this step it is shown that if the tracking error $|y(k) - y_{\text{ref}}(k)|$ is small, the state signal error $\|\phi_i(k) - \phi_i^*(k)\|$ is also small.

The signal $\phi_i(k) - \phi_i^*(k)$ is the output produced by the following transfer function H with $|y(k) - y_{\text{ref}}(k)|$ as the input:

$$H = \begin{bmatrix} 1 \\ q^{-1} \\ \vdots \\ q^{-m_1+1} \\ W^{-1}(q^{-1})q^{-1} \\ W^{-1}(q^{-1})q^{-2} \\ \vdots \\ W^{-1}(q^{-1})q^{-m_2-d_i+1} \end{bmatrix} \tag{5.62}$$

where $W^{-1}(q^{-1})$ is the inverse of the plant transfer function $W(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})}$ with the input signal $y(k) - y_{\text{ref}}(k)$ and $\phi_i^*(k)$ given in (5.60). From Assumption 2.3.1, it follows that $W^{-1}(q^{-1})$ is a stable transfer function. Hence, as $|y(k) - y_{\text{ref}}(k)|$ tends to zero, $\|\phi_i(k) - \phi_i^*(k)\|$ also tends to zero. From this it immediately follows that for any $\delta > 0$ there exists a $\bar{k}$ such that for all $k > \bar{k} : \|\phi_i(k) - \phi_i^*(k)\| \leqslant \delta$.

Step 2-2: *If* $|y(k) - y_{ref}(k)| \to 0$, *then* $\phi_i(k_j) \in \Omega_{p,i}, i = 1, 2, j = 1, 2, 3, \ldots$

Proof of Step 2-2: First it is shown that $\phi_d^*(k) \in \Omega_{p,i}$ for $i = 1, 2$ and $j = 1, 2, 3, \ldots$. Note that the reference model given in (5.60) is a linear system and hence there exists a state space representation

$$\phi_i^*(k+1) = R\phi_i^*(k) + Sy'_{\text{ref}} \tag{5.63}$$

with (R, S) being completely reachable. Then it follows directly from Lemma 2.3.4 that $\phi_i^*(k_j) \in \Omega_{p,i}$ for $i = 1, 2$ and $j = 1, 2, 3, \ldots$. Together with Step 2-1 it follows that if $|y(k) - y_{\text{ref}}(k)| \to 0$ for $k \to \infty$, then $\phi_i(k_j) \in \Omega_{p,i}$. Since the $y_{\text{ref}}(k)$ is SR of constant order p, $\phi_i(k_j' - d_i) \in \Omega_{p,i}$ also.

Step 2-3: $|e_2(k_p')| \leqslant e_{th}$ *for* $p = 3, 5, 7, \ldots$

Proof of Step 2-3: Starting from Equation (2.41), it follows that

$$|e_2(k_p')| = |\tilde{\vartheta}_2(k_{p-1})^T \phi_2(k_p' - d_2) + \tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k_p')| \tag{5.64}$$

$$= |\tilde{\vartheta}_2(k_{p-1})^T (\phi_2(k_p' - d_2) - \phi_2^*(k_p' - d_2) + \phi_2^*(k_p' - d_2)) + \tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k_p')| \tag{5.65}$$

$$\leqslant |\tilde{\vartheta}_2(k_{p-1})^T (\phi_2(k_p' - d_2) - \phi_2^*(k_p' - d_2))| + |\tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k_p') + \tilde{\vartheta}_2(k_{p-1})^T \phi_2^*(k_p' - d_2))|. \tag{5.66}$$

From Step 2-1 it follows that the first term is bounded above by $\delta \leqslant e_{\text{th}} - \varepsilon$. From Theorem 2.3.1 it follows that $|\tilde{\vartheta}_2(k_{p-1})^T \phi_2(k_{p-1}) + \tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k_p')| \leqslant \varepsilon$ and from Step 1-2 it follows that $\phi_2^*(k_p' - d_2) \in \Omega_2$ just as $\phi_2(k_{p-1})$. Hence,

$$|\tilde{\vartheta}_2(k_{p-1})^T \phi_2^*(k_p' - d_2) + \tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k_p')| \leqslant \varepsilon.$$

Thus, $|e_2(k_p')| \leqslant e_{\text{th}}$ for $p = 3, 5, 7, \ldots$.

Step 2-4: $|e_2(k_p' + l)| \leqslant e_{th}, p = 3, 5, 7, \ldots$ *and* $l = 1, 2, \ldots, m_2 + d_2 - 1$

Proof of Step 2-4: This step shows that the error at the beginning of the ET mode is below the threshold for at least $m_2 + d_2$ steps.

From Step 2-3 it follows that $|e_2(k_p')| \leqslant e_{\text{th}}$. According to the parameter choice in (5.13), the controller uses a constant initial value for the first $m_2 + d_2 - 1$ steps. Thus, the error $|e_2(k_p' + l)| = |\tilde{\vartheta}_2(k_{p-1})^T \phi_2(k_p' + l - d_2) + \tilde{\theta}_{2,\nu}(k_{p-1}) y_{\text{ref}}(k_p' + l)| \leqslant e_{\text{th}}$ because Steps 2-1 to 2-3 can be applied. ■

Remark 5.4.1. Since $y_{\text{ref}}(k)$ is not equal to zero in the tracking case, the choice of the parameter estimates at the switching instants to $\mathcal{M}_{\text{ET}}$ has to be changed compared to the stabilization case. That is, the parameter estimates at the beginning of an ET phase are set to the values at the end of the previous ET phase. This, in turn, necessitates some amount of time for the parameter estimates in the first ET phase following the first TT phase to converge. Therefore, the inter-arrival time of disturbances T_{dw} has to include the convergence time of the error both in TT and ET to e_{th}. This convergence time is given by $\Delta_{\text{TT}} + \Delta_{\text{ET}} + 2m_1$ where Δ_{TT} and Δ_{ET} are the settling times of the TT and the ET controller, respectively.

Remark 5.4.2. The nature of the proof is similar to that of Theorem 5.3.1, except for Stage 2. Again, the switching instants themselves were functions of the states of the closed-loop system, and thus it was necessary to show that these switching sequences exist. To this end, sufficient richness properties of the reference signal (Assumption 2.3.2) were utilized to show that the underlying signals in the reference model, and therefore the closed-loop adaptive system, converge to the same subspace.

This in turn led to the conclusion that the tracking error at the switch from TT to ET stays below the threshold e_{th}. It is the latter that distinguishes the adaptive controller proposed in this chapter, as well as the methodology used for the proof, from existing adaptive switching controllers and their proofs in the literature.

Remark 5.4.3. The two main components of the proposed controller are adaptation and switching. In order to examine the role of these two components, this controller is compared with (i) a switching, fixed controller, (ii) a non-switching, adaptive controller, and (iii) a non-switching, fixed controller. In terms of the proposed controller versus either (ii) or (iii), as this setup contains a hybrid communication bus, the usage of a non-switching controller would lead to an increased resource consumption as more slots in the static segment would then have to be used and hence are not suitable in a setup where many processing units have to arbitrate for bandwidth on the bus. Hence, using a switching protocol is advantageous.

In terms of comparison between the proposed controller and the controller in (i), the following comments can be made. Since any physical application being controlled inevitably includes uncertainties, and more unknown components may be introduced during the implementation phase in the cyber-domain, an adaptive controller that can accommodate these uncertainties on-line, in real-time, is attractive. For this reason, an adaptive controller has been chosen. Into this controller, switching was introduced, with the switches to occur whenever transients are triggered in the environment, such as due to an external disturbance.

This qualitative discussion is going to be supported by an extensive simulation in Chapter 6.

5.5 Summary

In this chapter, the control of multiple control applications using a hybrid communication protocol for sending control-related messages was considered. These protocols switch between time-triggered and event-triggered methods, with the switches dependent on the closed-loop performance, leading to a co-design of the controller and the communication architecture. In particular, this co-design consisted of switching between a TT and ET protocol depending on the amplitude of the tracking error, and correspondingly between two different adaptive controllers that are predicated on the resident delay associated with each of these protocols. The resulting adaptive switching controller was shown to result in bounded solutions for the task of stabilization, in the presence of a disturbance of impulse-trains, with the inter-arrival time between any two impulses greater than a finite constant. Furthermore, an extension to the task of tracking was presented and by using persistent excitation properties of the reference signal it was shown that the solutions remain bounded.

6 Numerical Case Studies

Summary: *In this chapter, numerical case studies with a full-fledged simulation are presented. It is shown through simulations that the co-design proposed in Chapter 5 achieves a better QoC than a fixed controller design. Finally, it is shown that the proposed co-design uses fewer expensive TT slots than a fixed-controller design.*

6.1 Introduction

Simulation is a very powerful technique that is used extensively in many areas. A simulation is the imitation of a real-world process or an abstraction of it that might exist or not. It helps to analyze the real system, to design a system in a systematic way or by trial-and-error, and to explore the boundaries of a real system without building the real system. Doing simulations has many advantages.

- All aspects of a system, including its limits, can be tested without spending money on prototypes.
- The possibility to expand time in a simulation gives a greater insight into the underlying process.
- To compress time in a simulation gives the possibility to see what happens to a real system in several years.
- Users can be trained on the real system without the danger of breaking the system or risking anyone's life.

Simulations also have disadvantages.

- Building a good model is a very difficult task and a simulation can only be as good as its model.
- The interpretation of the results is difficult and can lead to wrong decisions.

More advantages and disadvantages of simulations can be found in [142].

In this chapter, the co-design approach presented in Chapter 5 is simulated with a full-fledged Simulink model using the TrueTime toolbox [65, 143] and several case studies are carried out with this simulation.

TrueTime is a Matlab/Simulink-based library to simulate advanced real-time control systems. TrueTime facilitates co-simulation of controller task execution in real-time

kernels, network transmissions, continuous plant dynamics, and offers a variety of wired and wireless network blocks such as Ethernet, CAN, FlexRay, and 802.11b WLAN. TrueTime is written in C++ MEX and is freeware. The source code of TrueTime is open and thus is was possible to extend the functionality of TrueTime such that the proposed co-design scheme could be simulated.

This chapter is organized as follows: In Section 6.2 the TrueTime-based implementation of the proposed co-design scheme is presented and the problem is formulated. In Section 6.2.1, the performance of the proposed adaptive switching controller is compared with other common controllers. In Section 6.2.2, it is shown that the proposed adaptive switching controller reduces the number of TT slots necessary. The chapter is concluded with a summary of the results in Section 6.3.

6.2 Simulations

In Chapter 5, it was proved that the adaptive switching controller proposed within this chapter has globally bounded solutions in the case of regulation to zero and tracking a reference signal. In this chapter, it will be shown that the proposed controller design works as desired, has a better performance compared to other controllers, and reduces the number of TT slots necessary.

The simulation in Simulink uses a FlexRay ([120, 121, 141]) network block since the FlexRay protocol is a hybrid protocol that has a static segment, that is used for time-triggered communication, as well as a dynamic segment, which is used for event-triggered communication. Figure 6.1 shows a screen shot of the top-level Simulink model. The send blocks (green colored blocks) have an additional input named *dynamicSeg* that controls whether the static or the dynamic segment will be used for sending the current data value, i.e., the control input coming from the controller or the sensor value coming from the plant. This additional input is an extension of the original TrueTime block. The control algorithms described in Section 5.2.4 for the TT and ET segment are implemented in the pink blocks labeled *adaptive controller d=1* and *adaptive controller d=2*. An interference node, sending random data over the network, was used to simulate other non-control applications communicating over the same shared bus.

Using the same overall setup of the Simulink model, a fixed switching controller with the control law given in Equation (2.27) was also implemented.

The following linear open-loop unstable SISO plant was used for the simulations

$$G(s) = \frac{s+10}{s^2 + \frac{3}{2}s - 1}. \tag{6.1}$$

In order to compare the performance of the different simulations that were carried out, the cost function given by

$$J = \int_0^T e(t)^2 dt + \sum_{k=0}^{T/h} u(k)^2 \tag{6.2}$$

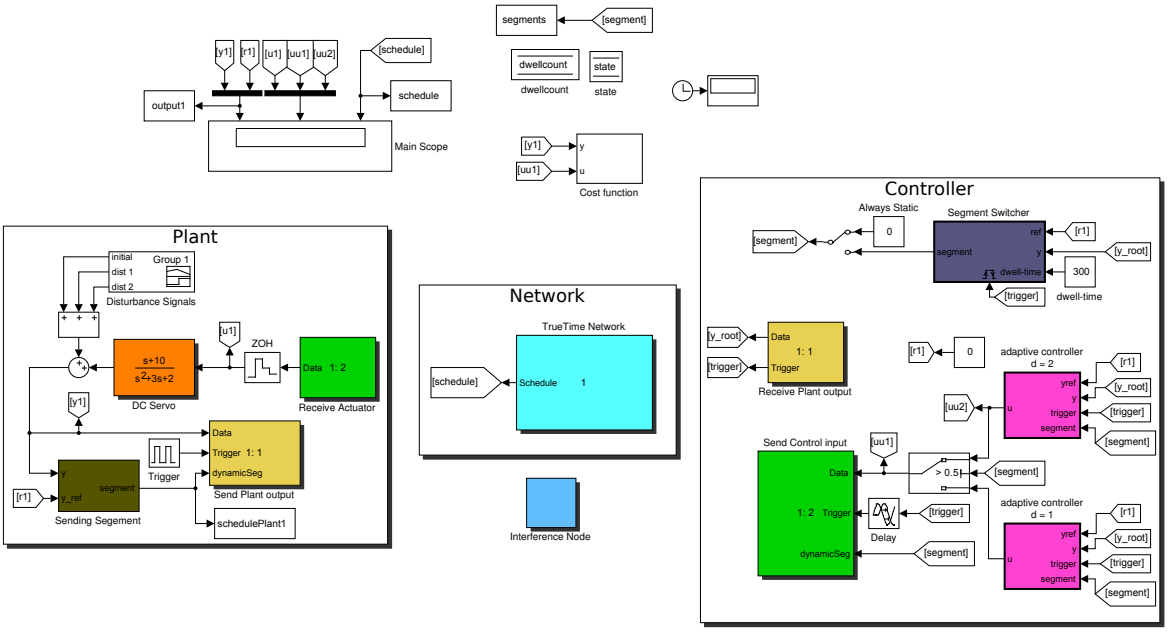


Figure 6.1: Simulink Block diagram

Parameter	Value
Data rate (bits/s)	80000
Minimum frame size (bits)	80
Slotsize (bits)	80
Static Schedule	[2 1]
Mini slot size (bits)	40
Dynamics Schedule	[3 3 3 2 1 3]

Table 6.1: Parameters of the FlexRay bus

where $e(t) = y(t) - y_{\text{ref}}(t)$, was used.

The sampling rate in the simulations was $h = 10\,\text{ms}$ and the FlexRay parameters given in Table 6.1 were used for the network. According to Section 2.2, the sampled version of $G(s)$ for $d = 1$ and $d = 2$, respectively, has the parameters

$$\begin{aligned} d = 1: \quad \vartheta_1^* &= \begin{bmatrix} 1.985 & -0.9851 & -0.009431 \end{bmatrix} & \beta_0^1 &= 0.01042 \\ d = 2: \quad \vartheta_2^* &= \begin{bmatrix} 2.955 & -1.9554 & 0.0113 & -0.0187 \end{bmatrix} & \beta_0^2 &= 0.01042. \end{aligned} \tag{6.3}$$

However, the parameters given in Equation (6.3) are not used for the controller in the simulation since it is assumed that the exact transfer function $G(s)$ is not known.

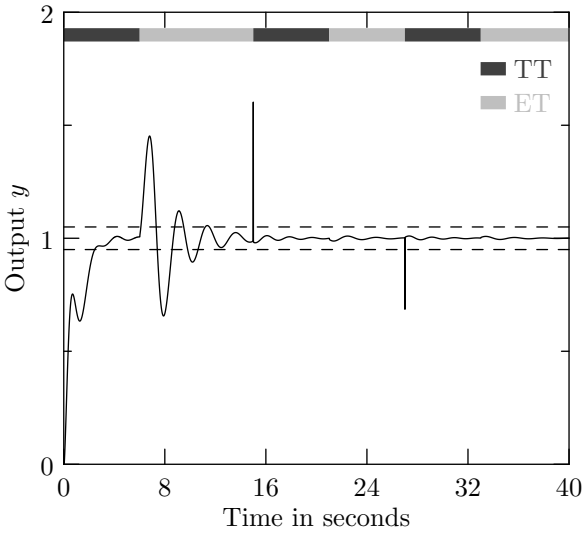


Figure 6.2: Evolution of the output y with the proposed switching adaptive controller for tracking

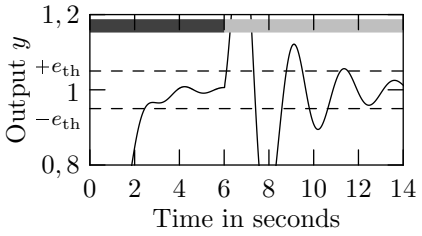


Figure 6.3: Detail of the learning phase

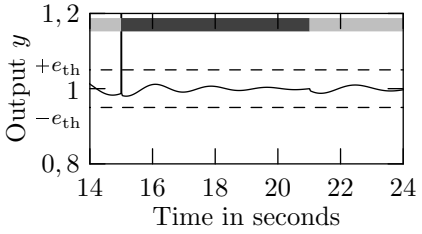


Figure 6.4: Detail of the switch from ET to TT at time 15 s and the switch from TT to ET at time 21 s

The parameters implemented are given by

$$\begin{aligned} d=1: \quad \vartheta_1^* &= \begin{bmatrix} 1.9 & -0.9 & -0.009 \end{bmatrix} & \beta_0^1 &= 0.02 \\ d=2: \quad \vartheta_2^* &= \begin{bmatrix} 2.9 & -1.9 & 0.01 & -0.02 \end{bmatrix} & \beta_0^2 &= 0.04. \end{aligned} \tag{6.4}$$

Before proceeding with the case studies, the switching adaptive controller and the switching algorithm are illustrated in detail. The threshold e_{th} for switching from TT to ET is given by $e_{\text{th}} = 0.05$, the reference signal is $y_{\text{ref}}(k) \equiv 1$ for all k and the dwell-time T_{dw} was chosen as 6 seconds.

Figure 6.2 shows the evolution of the output y with the proposed switching adaptive controller. The system is turned on at time $t_0 = 0\,\text{s}$ with zero initial conditions. According to Figure 5.2, the system is in mode $\mathcal{M}_{\text{TT}}$ and stays there for the dwell-time T_{dw}, i.e., until time $t_1 = 6\,\text{s}$. Since at time t_1, the error $|e(t_1)| < e_{\text{th}}$, at t_1, the system switches to mode $\mathcal{M}_{\text{ET}}$. Since the controller switches to a new algorithm and due to initialization of parameter estimates as in (5.55), the error becomes larger than e_{th} (see Figure 6.3). As time evolves, with the tracking action of the controller in $\mathcal{M}_{\text{ET}}$, the error becomes smaller than e_{th} at time $t_2 = 12\,\text{s}$. Since T_{dw} is chosen such that sufficient learning of the parameters has taken place, at t_2, the learning phase of the ET controller parameters is over. Since the error $|e(t_2)| < e_{\text{th}}$, the system continues to stay in $\mathcal{M}_{\text{ET}}$ until $t = 15\,\text{s}$, at which a disturbance occurs (see Figure 6.4), following which the system switches to $\mathcal{M}_{\text{TT}}$. Since the dwell time is chosen to be six seconds, the algorithm stays in $\mathcal{M}_{\text{TT}}$ until $t = 21\,\text{s}$, even though the tracking error has become well within e_{th} even at $t = 16\,\text{s}$. At $t = 21\,\text{s}$, the system switches to

$\mathcal{M}_{\text{ET}}$. Since the algorithm has adapted to $\mathcal{M}_{\text{ET}}$, despite the switch and the resetting of the parameters as in (5.56), the error continues to remain small and within the desired threshold. Therefore, the algorithm continues to stay at $\mathcal{M}_{\text{ET}}$. The switch from mode $\mathcal{M}_{\text{TT}}$ to $\mathcal{M}_{\text{ET}}$ at time t_4 does not cause a large error due to the learning phase at the beginning. The same procedure is repeated for the second disturbance which occurs at time $t_5 = 27\,\text{s}$. If no further disturbance occurs the system will stay in $\mathcal{M}_{\text{ET}}$ forever.

With the setup described above, two different case studies were carried out: The first one compares the performance of the proposed adaptive switching controller described in Section 5.2.4 to the performance of a non-switching adaptive controller, a fixed switching controller, and a non-switching fixed controller. The non-switching adaptive controller always sends its messages over the TT slot and uses the controller given in Equation (5.8). The fixed controllers use the control law given by Equation (2.27) with the parameters given by (6.4). The second case study is concerned with the question if the adaptive switching controller performs better than the fixed switching controller with less TT slots, i.e., when lower priority applications have to wait longer. In other words, the question that is addressed is if the adaptive switching controller requires less TT slots than a fixed controller in order to realize a given QoC. The results of these simulations are described below.

6.2.1 Case Study 1

In this case study, the performance of the proposed adaptive switching controller is compared with three other common controller configurations: a non-switching adaptive controller, a fixed switching controller, and a non-switching fixed-controller. Both non-switching controllers use the control law for the TT case with $d = 1$. The fixed controllers use the parameters given in (6.4). The scenario for this case study was the following: The total time of simulation is 25 s and two disturbances occur at 6 s with a magnitude of 0.8 and at 14 s with a magnitude of -1.5. Two sets of simulations are carried out in this case study: (i) the stabilization problem is considered as described in Section 5.3 and (ii) the tracking problem as described in Section 5.4.

For the stabilization case, Figure 6.5 shows that the proposed adaptive switching controller for stabilization works. After each disturbance the controller switches to the TT mode and stabilizes the system. After the dwell-time of 3 s, the system switches back to $\mathcal{M}_{\text{ET}}$ and stays therein if no further disturbance occurs.

In Figure 6.6, the output of the plant with a fixed controller is shown. The dwell-time is also 3 s. The goal of stabilization is achieved. It is obvious that the fixed controller results in a worse performance than the adaptive controller. Using the cost function $J = \int_0^T y(t)^2 dt + \sum_{k=0}^{T/h} u^2$ (since $y_{\text{ref}} \equiv 0$), the controllers have the costs shown in Table 6.2. The costs for stabilization $\int_0^T y(t)^2 dt$ and control effort $\sum_{k=0}^{T/h} u^2$ are listed separately, since the magnitudes of the costs are very different for the fixed and the adaptive controllers. Overall, the switching adaptive controller has a much lower

Controller	$\int_0^T y^2 dt$	$\sum_{k=0}^{T/h} u^2$
switching adaptive	0.144	0.57
non-switching adaptive	0.149	1.27
switching fixed	0.180	469.5
non-switching fixed	0.144	399.4

Table 6.2: Cost of the controllers for the stabilization case

cost than the switching fixed controller and as both non-switching controllers. It can be seen from Table 6.2, that the cost for stabilization is the same for the switching adaptive controller and the non-switching fixed controller. However, the control effort for the latter is much higher.

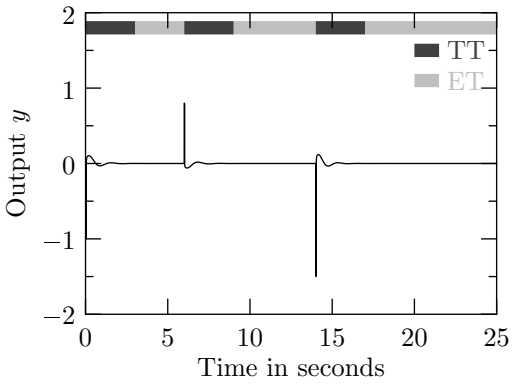


Figure 6.5: Evolution of the output y with the proposed switching adaptive controller for stabilization

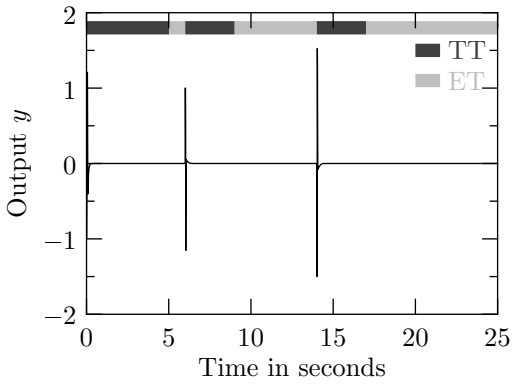


Figure 6.6: Evolution of the output y with a switching fixed controller

In the tracking case, Figure 6.7 shows the evolution of the output for the adaptive switching controller. The reference input was chosen constant $y_{\text{ref}}(k) \equiv 1 \, \forall k$. The dwell-time is 6 s. The learning phase described in Section 5.4 is implemented in the switching algorithm of the Simulink model. The dwell-time of the learning phase is 6 s, i.e., a disturbance should not occur between 6 s and 12 s. The first disturbance occurs at 15 s with a magnitude of 0.615. The second disturbance occurs at 27 s with a magnitude of −0.525. It can be seen from Figure 6.7 that the goal of tracking the reference signal is achieved. Figure 6.8 shows the evolution of the output with a fixed switching controller with parameters given in Equation (6.4). Using the cost function $J = \int_0^T e(t)^2 dt + \sum_{k=0}^{T/h} u^2$, the costs shown in Table 6.3 were obtained.

The switching algorithm for tracking stays in the ET mode for at least $t_l = 6$ s to let the adaptation of the controller parameters proceed no matter whether the error

Controller	$\int_0^T e^2 dt$	$\sum_{k=0}^{T/h} u^2$
switching adaptive	0.609	0.77
non-switching adaptive	0.394	0.62
switching fixed	0.303	2669.0
non-switching fixed	0.112	942.5

Table 6.3: Cost of the controllers for the tracking case

Controller	$\int_{t_l}^T e^2 dt$	$\sum_{k=t_l/h}^{T/h} u^2$
switching adaptive	0.007	0.26
non-switching adaptive	0.008	0.27
switching fixed	0.252	2472.0
non-switching fixed	0.083	824.3

Table 6.4: Cost of the controllers for the tracking case after the learning phase t_l

becomes larger than the threshold or not. As can be seen from Table 6.3 the cost $\int_0^T e^2 dt$ of the adaptive switching controller is larger than the cost of the other three controllers. The cost $\sum_{k=0}^{T/h} u^2$ of the fixed controllers, however, is several magnitudes larger than for the adaptive controllers. Considering in the cost function only for times after the learning phase, i.e., $t > t_l$, shows that the adaptive switching controller has the best performance compared to the other three controllers, see Table 6.4. The non-switching fixed controller shows also a good tracking performance, i.e., a small cost $\int_0^T e^2 dt$, but also shows a worse bandwidth utilization than the switching controllers.

The switching fixed controller stays in $\mathcal{M}_{\text{TT}}$ after the learning phase because the switch from TT to ET produces always a jump in the tracking error that is larger than the threshold e_{th}. The adaptive switching controller switches to $\mathcal{M}_{\text{ET}}$ (and stays there if no disturbance occurs) because the adaptive controller for the ET protocol has already good parameter estimates and thus the tracking error stays below e_{th}.

6.2.2 Case Study 2

In order to show that the proposed switching adaptive controller reduces the number of TT slots necessary, the following simulations were carried out.

The setup consists of four applications $\mathcal{C}_1, \ldots, \mathcal{C}_4$ and up to four TT slots $\mathcal{S}_i, i = 1, \ldots, 4$, each of them with the transfer function given in Equation (6.1). The threshold e_{th} for switching from TT to ET is given by $e_{\text{th}} = 0.05$, the reference signal is $y_{\text{ref}}(k) \equiv 1$

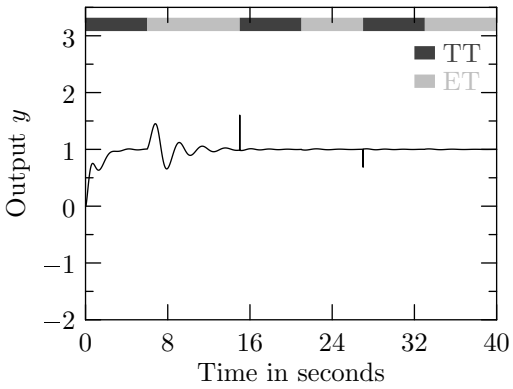


Figure 6.7: Evolution of the output y with the proposed switching adaptive controller for tracking

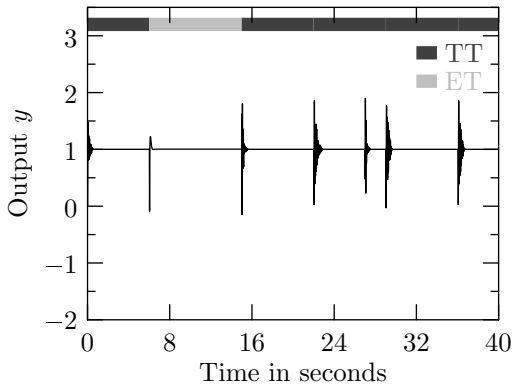


Figure 6.8: Evolution of the output y with a switching fixed controller for tracking

for all k and the dwell-time T_{dw} was chosen as 6 seconds. A disturbance occurs at 15 s with a magnitude of 0.57.

In the first simulation it is assumed that each of the four applications has its own TT slot $\mathcal{S}_1, \ldots, \mathcal{S}_4$ and thus can switch to $\mathcal{M}_{\text{TT}}$ at once if necessary. In the second simulation it is assumed that two applications share one TT slot, i.e., there are a total of three TT slots. That is, $(\mathcal{C}_1, \mathcal{C}_2) \rightarrow \mathcal{S}_1$, $\mathcal{C}_3 \rightarrow \mathcal{S}_2$ and, $\mathcal{C}_4 \rightarrow \mathcal{S}_3$. This means, if both $\mathcal{C}_1$ and $\mathcal{C}_2$ want to switch to the TT slot at the same time, the one with the lower priority has to wait. In the third simulation, the number of TT slots is further reduced by one and now three applications are being mapped onto one TT slot, i.e., $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3) \rightarrow \mathcal{S}_1$ and $\mathcal{C}_2 \rightarrow \mathcal{S}_2$. In the last simulation all applications are being mapped to one TT slot, i.e., $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4) \rightarrow \mathcal{S}_1$. In all four simulations it is assumed that $\mathcal{C}_1$ has the lowest priority of all four applications, and thus has to wait for the longest time period.

Table 6.5 shows the results of these four simulations. In the last two columns, the cost after the learning phase, i.e., $t > t_l$, is shown. It can be seen that the cost for the fixed switching controller is always higher than with the adaptive switching controller. Furthermore, the cost stays almost constant for the adaptive switching controller when the number of applications per TT slot is increased. Whereas for the fixed switching controller the cost increases.

6.3 Summary

In this chapter, the switching algorithm presented in Chapter 5 was validated and its superiority compared to other common controller implementations was shown. To this end a full-fledged implementation of the switching algorithm was implemented using Simulink with an extension of the TrueTime toolbox. Additional functionality was added to TrueTime that allowed to decide at runtime whether a message should

Apps per TT slot	Controller	$\int_0^T e^2 dt$	$\sum_{k=0}^{T/h} u^2$	$\int_{t_l}^T e^2 dt$	$\sum_{k=t_l/h}^{T/h} u^2$
1	Adaptive Switching	0.6143	0.7846	0.01308	0.2719
	Fixed Switching	0.931	13010.0	0.3188	5233.0
2	Adaptive Switching	0.6147	0.7819	0.01346	0.2692
	Fixed Switching	1.154	16830.0	0.5414	9048.0
3	Adaptive Switching	0.6147	0.7819	0.01346	0.2692
	Fixed Switching	1.158	16900.0	0.5454	9123.0
4	Adaptive Switching	0.6148	0.7819	0.01355	0.2692
	Fixed Switching	1.163	16940.0	0.551	9162.0

Table 6.5: Costs for different numbers of applications per TT slot

be sent in the static or dynamic segment. Using this simulation environment, two case studies were carried out to illustrate the advantages of the adaptive switching controller presented in Chapter 5 over a fixed controller design, both switching and non-switching. By means of a cost function, it was shown that the proposed co-designed switching adaptive controller shows a better performance under worst-case assumptions, both for the case of regulation to zero and the case of tracking a reference signal. Furthermore, it was shown with the switching adaptive controller that the number of time-triggered slots can be reduced compared to a fixed controller design. Overall, this shows that a co-designed switching adaptive controller is advantageous.

7 Conclusion and Future Work

Summary: *This chapter summarizes the contributions of this thesis and gives an outlook on future directions.*

7.1 Conclusion

Cyber-Physical Systems (CPS) are ubiquitous in our everyday life and thus are a key technology for the future. The term CPS is a general term for computer systems that are embedded into the physical world and are linked together through communication networks. CPS interact directly and most of the time independently of human operators with their physical environment by sensing the environment, computing a required action, and applying the action under the constraint of limited resources like limited communication bandwidth, limited power, limited computational power, and so on. One reason for their pervasiveness is that the capabilities of CPS are increasing steadily while the costs for procurement and operation are declining at an even higher rate. One can identify three main technical fields for the design and development of CPS: automatic control, networked control systems, and embedded systems. Traditionally, the design of the controller and its stability and performance analysis is done independently from the design and analysis of the communication and computation architecture which in turn aggravates the verification of the resulting CPS. In each of these fields, solutions for specific problems exist, but the challenge is to combine these solutions into one integrated approach – called co-design – with a sound theoretical foundation that is necessary in order to exploit the full capabilities of CPS.

Arbitrated Networked Control Systems: Traditional NCS research is concerned with challenges like message loss, jitter, or varying delays which arise in the context of distributed control over a given network. However, in embedded platforms the network itself is not fixed a priori, but offers the possibility to jointly design and optimize the network and the controller. To emphasize the connection of controller design and network design – especially the arbitration policy of the shared communication medium – this is referred to as *Arbitrated Networked Control Systems (ANCS)*. The term is novel and different examples of ANCS and solutions based on ANCS are presented in this thesis. By emphasizing the arbitration of messages, i.e., the communication network, additional information about the network can be used for the design of the controller.

ANCS with Hierarchical Schedules: Two co-design approaches are presented for ANCS with hierarchical schedules. First, the hierarchical schedules are opti-

mized for improved control performance. Second, adaptive controllers are used to improve the control performance. The former is achieved by the following steps: A detailed model of the implementation platform and the controller structure are developed. Next, the Real-Time Calculus (RTC) framework is used to find all feasible system configuration parameters with a finite worst-case end-to-end communication delay. This is done by an extensive design space exploration. Next, a novel performance metric is presented. Since RTC provides only upper bounds on the delay, but the average delay is smaller, it is shown analytically that this novel performance metric monotonically increases with decreasing delay. That is, the performance computed with the novel performance metric based upon the worst-case delay is a lower bound and it will be higher in reality. The RTC framework and the performance metric are then combined into a co-design of control and communication by solving a nonlinear optimization problem with multiple performance metrics. In contrast to other co-design approaches, this approach provides the optimal choice of controller and scheduler parameters by the one-time solution of an optimization problem. By applying the co-design to a realistic platform with a hierarchical schedule, it is shown that a co-design is necessary and advantageous since the performance depends on the platform parameters as well as the controller parameters.

Since uncertainties are present in any system, an adaptive controller is added to the ANCS. The additional information provided by the ANCS about the network, especially the known part of the delay, is used to design an adaptive controller that accommodates the uncertainties due to changes in the network and stabilizes the system. By means of numerical simulations it is shown that the designed adaptive controller achieves a better performance than a fixed controller design.

ANCS with Hybrid Schedules: A co-design of control and communication with hybrid protocols is presented. The co-design consists of three parts: (i) a hybrid communication protocol with a time-triggered and an event-triggered segment, (ii) a switching scheme that decides whether a control-related message is sent over the time-triggered or the event-triggered segment, and (iii) an adaptive controller in order to cope with any uncertainties that may be present. The specific solution proposed is a co-design where the switches of the controller are aligned synergistically with the switches in the embedded platform. While part of the switches are introduced due to disturbances and therefore are due to the environment, part of the switches are by design. The plant being controlled is assumed to be unknown, and an adaptive control solution is employed to accommodate the uncertainty. This significantly complicates the problem and necessitates a fairly involved proof of stability and it is shown that regulation to zero and tracking of a reference signal in the presence of a class of disturbances is achieved. The main challenge in demonstrating the stability of the underlying switching adaptive controller is the fact that the delay in the underlying plant-

model switches between two values which, in turn, is due to the hybrid use of the TT and ET protocols. Since the hybrid protocol is highly attractive because of the advantages it offers and is becoming more and more prevalent among embedded systems, the proposed solution will have a significant impact. The superiority of the proposed co-design over other common controller implementations is demonstrated by a full-fledged simulation. To this end the functionality of the TrueTime toolbox is extended to allow sending blocks to decide in runtime over which segment the messages should be sent.

7.2 Future Work

Future challenges and possible extensions to the results and insights gained in this thesis are discussed in the following:

- The performance metric introduced in Chapter 4 allows to prove analytically that it is monotonically increasing with decreasing delay. The same might be possible for other standard performance metrics.

- Instead of the worst-case delay computed with RTC, other non-deterministic queuing techniques could be used to compute the average delay a message experiences. This average delay could be then used to design the control law. However, this poses the problem to prove stability, since some samples might experience a higher delay and probably the order of the messages is no longer reserved.

- The design space exploration (Section 4.2.2) is an efficient, yet simple way to to find all feasible platform configurations. In future, this task might be formulated as an optimization problem and combined with the optimization of the controller parameters.

- Hierarchical and hybrid communication schedules are highly attractive. In this thesis, the focus was on communication schedules. It might be interesting to extend the co-design to the schedules on the processors where other features of the schedules play an important role. For example, in future, preemption might be considered. The interruption of a currently running job is called preemption. For example, higher priority jobs, i.e., more urgent jobs, can preempt lower priority jobs. Once the urgent job is completed, the lower priority job is resumed. The co-design for this kind of schedules is significantly more complex, from the timing analysis point of view as well as from the control design point of view.

- In this thesis, the mapping of computational tasks to processors was fixed, but it might be added to the co-design as well and would add an additional dimension to the problem.

- The co-design procedure presented in Section 4.2 assumes a simple controller structure. More advanced controllers, also nonlinear controllers like adaptive controllers, could be considered. This would complicate the overall procedure significantly.
- From the perspective of the implementation platform, in Chapter 5 only two communication modes have been considered. As a part of future work, multiple modes of the computation platform (i.e., the processors) could be taken into account. In particular, it is possible to investigate how to design a switched control strategy when the processors are not uniformly available because of other applications that have been mapped onto them.
- The proposed co-design in Chapter 5 is supported by a full-fledged simulation. For further validation of the theoretical results, an experimental setup and testing is necessary.

Bibliography

[1] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[2] A. Kiani, "Wholesale energy market in a smart grid: Dynamic modeling, stability, and robustness," Ph.D. dissertation, Technische Universität München, 2012.

[3] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[4] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Addison Wesley, 1995.

[5] A. Isidori, *Nonlinear Control Systems*. Springer, 1995.

[6] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Prentice Hall, 1990.

[7] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Dover Publications, 2005.

[8] K. J. Åström and B. Wittenmark, *Adaptive Control*. Dover Publications, 2008.

[9] D. R. Jenkins, *Hypersonics Before the Shuttle - A Concise History of the X-15 Research Airplane*. National Aeronautics and Space Administration (NASA), 2000.

[10] K. S. Narendra, Y.-H. Lin, and L. S. Valavani, "Stable adaptive controller design, Part II: Proof of stability," *IEEE Transactions on Automatic Control*, vol. 25, pp. 440 – 448, 1980.

[11] K. S. Narendra and L. S. Valavani, "Stable adaptive controller design – direct control," *IEEE Transactions on Automatic Control*, vol. 23, pp. 570 – 583, 1978.

[12] K. S. Narendra and L. S. Valavani, "Stable adaptive controller design, Part I: Direct control," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 1977.

[13] J. V. Amerongen, "Adaptive steering of ships - A model reference approach," *Automatica*, vol. 20, pp. 3 – 14, 1984.

[14] J. Hu, R. Bu, J. Yin, and Y. Wang, "Adaptive control for ship steering based on clonal selection model identification," in *Proc. of the Chinese Control and Decision Conference*, 2009.

[15] S. Sosnowski, J.-M. P. Franosch, L. Zhang, Y. Nie, S. Hirche, and J. L. van Hemmen, "Simulation of the Underwater Vehicle "Snookie": Navigating like a Fish," in *Proc. of the 1st International Conference on Applied Bionics and Biomechanics (ICABB)*, 2010.

[16] D. E. Seborg, T. F. Edgar, and S. L. Shah, "Adaptive control strategies for process control: A survey," *AIChE Journal*, vol. 32, pp. 881–913, 1986.

[17] A. Ghosh, G. Ledwich, O. Malik, and G. Hope, "Power System Stabilizer Based on Adaptive Control Techniques," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-103, pp. 1983 –1989, 1984.

[18] J.-J. E. Slotine and W. Li, "On the Adaptive Control of Robot Manipulators," *The International Journal of Robotics Research*, vol. 6, pp. 49–59, 1987.

[19] S. Enbiya, M. Hossain, and F. Mahieddine, "Multi-drug Infusion Control Using Model Reference Adaptive Algorithm," in *5th International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB)*. Springer, 2011.

[20] S. Enbiya, F. Mahieddine, and A. Hossain, "Model Reference Adaptive Scheme for Multi-drug Infusion for Blood Pressure Control," *Journal of Integrative Bioinformatics*, vol. 8, pp. 1–14, 2011.

[21] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Composite Adaptive Control of a Quadrotor UAV in the Presence of Actuator Uncertainty," in *AIAA Guidance, Navigation, and Control Conference*, 2010.

[22] Z. T. Dydek, A. M. Annaswamy, J.-J. E. Slotine, and E. Lavretsky, "Time delay resistant adaptive control of mini-UAVs," in *IFAC Workshop on Time Delay Systems*, 2010.

[23] J. Nilsson, "Real-time control systems with delays," Ph.D. dissertation, Department of Automatic Control, Lund Institute of Technology, Sweden, 1998.

[24] "CAN," May 2013. [Online]. Available: www.can-cia.org

[25] C. Meng, T. Wang, W. Chou, S. Luan, Y. Bang, and Z. Tian, "Remote surgery case: robot-assisted teleneurosurgery," in *IEEE International Conference on Robotics and Automation*, 2004.

[26] R. Daoud and H. Amer, "Fault-tolerant access points in a mobile IPv6 WiFi light urban traffic communication model," in *Canadian Conference on Electrical and Computer Engineering*, 2008.

[27] H. Newman, "Integrating building automation and control products using the BACnet protocol," *ASHRAE Journal*, vol. 38, pp. 36–42, 1996.

[28] T. Yang, J. Zhang, and H. Yu, "A new decentralised controller design method with application to power-system stabiliser design," *Control Engineering Practice*, vol. 7, pp. 537 – 545, 1999.

[29] E. Steinbach, S. Hirche, M. Ernst, F. Brandi, R. Chaudhari, J. Kammerl, and I. Vittorias, "Haptic communications," *Proceedings of the IEEE*, vol. 100, pp. 937–956, 2012.

[30] J. Baillieul and P. Antsaklis, "Control and communication challenges in networked real-time systems," *Proceedings of the IEEE*, vol. 95, pp. 9–28, 2007.

[31] S. Bennett, *A History of Control Engineering 1930-1955*. Peter Peregrinus, 1993.

[32] C. Bissell, "A history of automatic control," in *Springer Handbook of Automation*. Springer, 2009.

[33] G. Nair, F. Fagnani, S. Zampieri, and R. Evans, "Feedback Control Under Data Rate Constraints: An Overview," *Proceedings of the IEEE*, vol. 95, pp. 108–137, 2007.

[34] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1056 – 1068, 2004.

[35] S. Tatikonda and S. Mitter, "The capacity of channels with feedback," *IEEE Transactions on Information Theory*, vol. 55, pp. 323 –349, 2009.

[36] C.-C. Chen, S. Hirche, and M. Buss, "Sampled-data networked control systems with random time delay," in *Proc. of IFAC World Congress*, 2008.

[37] T. Matiakis, S. Hirche, and M. Buss, "Networked control systems with time-varying delay - stability through input-output transformation," *at - Automatisierungstechnik*, vol. 56, pp. 29–37, 2008.

[38] W. Zhang, M. Branicky, and S. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, pp. 84 –99, 2001.

[39] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, pp. 138–162, 2007.

[40] P. Varutti and R. Findeisen, "On the synchronization problem for the stabilization of networked control systems over nondeterministic networks," in *Proc. of American Control Conference (ACC)*, 2009.

[41] S. Johannessen, "Time synchronization in a local area network," *IEEE Control Systems Magazine*, vol. 24, pp. 61 – 69, 2004.

[42] J. W. S. Liu, *Real-Time Systems*. Prentice Hall, 2000.

[43] D. A. Mindell, *Digital Apollo - human and machine in spaceflight.* MIT Press, 2008.

[44] W. H. Wolf, *Computers as components: principles of embedded computing system design.* Elsevier Ltd, 2001.

[45] T. Nghiem, G. J. Pappas, R. Alur, and A. Girard, "Time-triggered implementations of dynamic controllers," in *Proc. of ACM & IEEE International Conference on Embedded Software*, 2006.

[46] H. Yazarel, A. Girard, G. J. Pappas, and R. Alur, "Quantifying the gap between embedded control models and time-triggered implementations," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 2005.

[47] S. Samii, A. Cervin, P. Eles, and Z. Peng, "Integrated scheduling and synthesis of control applications on distributed embedded systems," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2009.

[48] S. Samii, P. Eles, Z. Peng, and A. Cervin, "Quality-driven synthesis of embedded multi-mode control systems," in *Proc. of Design Automation Conference (DAC)*, 2009.

[49] B. Lincoln and A. Cervin, "Jitterbug: A Tool for Analysis of Real-Time Control Performance," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2002.

[50] A. Cervin and T. Henningsson, "Scheduling of event-triggered controllers on a shared network," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2008.

[51] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 2008.

[52] L. Palopoli, C. Pinello, A. Sangiovanni Vincentelli, L. Elghaoui, and A. Bicchi, "Synthesis of robust control systems under resource constraints," in *Proc. of Hybrid Systems: Computation and Control (HSCC).* Springer, 2002.

[53] P. Naghshtabrizi and J. P. Hespanha, "Analysis of distributed control systems with shared communication and computation resource," in *Proc. of American Control Conference (ACC)*, 2009.

[54] X. Wang and M. Lemmon, "Event-triggering in distributed networked control systems," *IEEE Transactions on Automatic Control*, vol. 56, pp. 586–601, 2011.

[55] AUTOSTAR, "Specification of flexray interface, ver. 3.0.3," June 2013. [Online]. Available: http://www.autosar.org

[56] "BMW brake system relies on flexray," June 2013. [Online]. Available: http://www.eetimes.com/electronics-news/4197804/BMW-Brake-system-relies-on-FlexRay

[57] L. Palopoli, C. Pinello, A. Bicchi, and A. Sangiovanni-Vincentelli, "Maximizing the stability radius of a set of systems under real-time scheduling constraints," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1790–1795, 2005.

[58] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1680–1685, 2007.

[59] M. Velasco, P. Marti, and E. Bini, "Control-driven tasks: Modeling and analysis," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 2008.

[60] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control system," *Control Engineering Practice*, vol. 11, pp. 1099–1111, 2003.

[61] T.-C. Yang, "Networked control system: a brief survey," *IEE Proceedings - Control Theory and Applications*, vol. 153, pp. 403–412, 2006.

[62] Y. Halevi and A. Ray, "Integrated Communication and Control Systems: Part I - Analysis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 367–373, 1988.

[63] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An introduction to control and scheduling co-design," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2000.

[64] G. Walsh and H. Ye, "Scheduling of networked control systems," *IEEE Control Systems Magazine*, vol. 21, pp. 57 –65, 2001.

[65] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime," *IEEE Control Systems Magazine*, vol. 23, pp. 16–30, 2003.

[66] F. Xia and Y. Sun, "Control-scheduling codesign: A perspective on integrating control and computing," *Dynamics of Continuous, Discrete and Impulsive Systems*, vol. 13, pp. 1352–1358, 2006.

[67] M. Gaid, A. Cela, and Y. Hamam, "Optimal integrated control and scheduling of networked control systems with communication constraints: application to a car suspension system," *IEEE Transactions on Control Systems Technology*, vol. 14, pp. 776–787, 2006.

[68] G. Che and Y. Jin, "Online co-design of feedback control and real-time scheduling for embedded systems with commnuication delays," in *Proc. of International Conference on Computer Science Education*, 2009.

[69] **H. Voit** and A. M. Annaswamy, "Local adaptive controllers for networked cooperative systems," in *Proc. of American Control Conference (ACC)*, 2010.

[70] D. Seto, J. Lehoczky, L. Sha, and K. Shin, "On task schedulability in real-time control systems," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 1996.

[71] J. Stankovic, C. Lu, S. Son, and G. Tao, "The case for feedback control real-time scheduling," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 1999.

[72] L. Abeni, L. Palopoli, and G. Buttazzo, "On adaptive control techniques in real-time resource allocation," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2000.

[73] J. Eker, P. Hagander, and K.-E. Årzén, "A feedback scheduler for real-time controller tasks," *Control Engineering Practice*, vol. 8, pp. 1369–1378, 2000.

[74] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, "Feedback-feedforward scheduling of control tasks," *Real-Time Systems*, vol. 23, pp. 25–53, 2002.

[75] A. Cervin and P. Alriksson, "Optimal on-line scheduling of multiple control tasks: A case study," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2006.

[76] M. Pajic, S. Sundaram, J. Le Ny, G. Pappas, and R. Mangharam, "The Wireless Control Network: Synthesis and robustness," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2010.

[77] A. Cervin and J. Eker, "Control-scheduling codesign of real-time systems: The control server approach," *Journal of Embedded Computing*, vol. 1, pp. 209–224, 2005.

[78] F. Zhang, Z. Shi, and W. Wolf, "A Dynamic Battery Model for Co-design in Cyber-Physical Systems," in *Proc. of IEEE International Conference on Distributed Computing Systems Workshops*, 2009.

[79] M. Mazo and P. Tabuada, "On event-triggered and self-triggered control over sensor/actuator networks," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2008.

[80] X. Wang and M. Lemmon, "State Based Self-Triggered Feedback Control Systems with L2 Stability," in *Proc. of IFAC World Congress*, 2008.

[81] T. Henningsson, E. Johannesson, and A. Cervin, "Sporadic event-based control of first-order linear stochastic systems," *Automatica*, vol. 44, pp. 2890 – 2895, 2008.

[82] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," in *Proc. of Hybrid Systems: Computation and Control (HSCC)*. Springer, 2007.

[83] R. Alur and G. Weiss, "Regular specifications of resource requirements for embedded control software," in *Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2008.

[84] A. S. Morse, "Supervisory control of families of linear set-point controllers Part I. Exact matching," *IEEE Transactions on Automatic Control*, vol. 41, pp. 1413–1431, 1996.

[85] K. S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE Transactions on Automatic Control*, vol. 42, pp. 171–187, 1997.

[86] K. S. Narendra and C. Xiang, "Adaptive control of discrete-time systems using multiple models," *IEEE Transactions on Automatic Control*, vol. 45, pp. 1669–1686, 2000.

[87] M. di Bernardo, U. Montanaro, and S. Santini, "Novel hybrid MRAC-LQ control schemes: synthesis, analysis and applications," *International Journal of Control*, vol. 81, pp. 940–961, 2008.

[88] V. Hassani, J. Hespanha, M. Athans, and A. Pascoal, "Stability analysis of robust multiple model adaptive control," in *Proc. of IFAC World Congress*, 2011.

[89] M. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 475–482, 1998.

[90] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine*, vol. 19, pp. 59–70, 1999.

[91] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 1999.

[92] D. Liberzon, *Switching in systems and control*. Birkhäuser, 2003.

[93] A. Rajhans, S.-W. Cheng, B. R. Schmerl, D. Garlan, B. H. Krogh, C. Agbi, and A. Bhave, "An architectural approach to the design and analysis of cyber-physical systems," *Electronic Communications of the EASST*, vol. 21, pp. 2–11, 2009.

[94] E. P. Godoy, A. J. V. Porto, and R. Y. Inamasu, "Sampling time adaptive control methodology for CAN-based Networked Control Systems," in *Proc. of IEEE/IAS International Conference on Industry Applications (INDUSCON)*, 2010.

[95] S. Piltz, M. Bjorkbom, L. Eriksson, and H. Koivo, "Step adaptive controller for networked mimo control systems," in *International Conference on Networking, Sensing and Control (ICNSC)*, 2010.

[96] L. Chunmao and X. Jian, "Adaptive delay estimation and control of networked control systems," in *Proc. of International Symposium on Communications and Information Technologies (ISCIT)*, 2006.

[97] P. Marti, J. Yepez, M. Velasco, R. Villa, and J. Fuertes, "Managing quality-of-control in network-based control systems by controller and message scheduling co-design," *IEEE Transactions on Industrial Electronics*, vol. 51, pp. 1159 – 1167, 2004.

[98] **H. Voit**, R. Schneider, D. Goswami, A. M. Annaswamy, and S. Chakraborty, "Optimizing Hierarchical Schedules for Improved Control Performance," in *Proc. of IEEE Symposium on Industrial Embedded Systems (SIES)*, 2010.

[99] **H. Voit**, R. Schneider, D. Goswami, A. M. Annaswamy, and S. Chakraborty, "Scheduler Design for Distributed Embedded Control Systems," *IEEE Transactions on Industrial Electronics*, 2013, in preparation.

[100] A. M. Annaswamy, S. Chakraborty, D. Soudbakhsh, D. Goswami, and **H. Voit**, "The arbitrated networked control systems approach to designing cyber-physical systems," in *Proc. of IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys)*, 2012.

[101] **H. Voit** and A. M. Annaswamy, "Adaptive Control of a Networked Control System with Hierarchical Scheduling," in *Proc. of American Control Conference (ACC)*, 2011.

[102] **H. Voit**, D. Goswami, and A. M. Annaswamy, "Arbitrated Networked Control Systems with Hierarchical Scheduling," *Asian Journal of Control*, 2013, submitted.

[103] **H. Voit**, A. M. Annaswamy, R. Schneider, D. Goswami, and S. Chakraborty, "Adaptive Switching Controllers for Systems with Hybrid Communication Protocols," in *Proc. of American Control Conference (ACC)*, 2012.

[104] **H. Voit**, A. M. Annaswamy, R. Schneider, D. Goswami, and S. Chakraborty, "Adaptive Switching Controllers for Tracking with Hybrid Communication Protocols," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2012.

[105] **H. Voit**, R. Schneider, D. Goswami, A. M. Annaswamy, and S. Chakraborty, "Adaptive Switching Controllers for Multi-Modal Embedded Systems," *IEEE Transactions on Automatic Control*, 2013, in preparation.

[106] E. I. Jury, "Sampling schemes in sampled-data systems," *IRE Transactions on Automatic Control*, vol. 6, pp. 86–88, 1961.

[107] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, pp. 10–21, 1949.

[108] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*. Prentice Hall, 1984.

[109] G. C. Goodwin, P. Ramadge, and P. Caines, "Discrete-time multivariable adaptive control," *IEEE Transactions on Automatic Control*, vol. 25, pp. 449–456, 1980.

[110] E. Bai and S. Sastry, "Persistency of excitation, sufficient richness and parameter convergence in discrete time adaptive control," *Systems & Control Letters*, vol. 6, pp. 153–163, 1985.

[111] F.-L. Lian, J. R. Moyne, and D. M. Tilbury, "Network protocols for networked control systems," in *Handbook of Networked and Embedded Control Systems*. Birkhäuser, 2005.

[112] H. Zimmermann, "OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, pp. 425 – 432, 1980.

[113] H. Kopetz and G. Grunsteidl, "TTP - a time-triggered protocol for fault-tolerant real-time systems," in *Proc. of the International Symposium on Fault-Tolerant Computing*, 1993.

[114] "ISO/CD11898-4, Road Vehicles Controller Area Network (CAN) Part 4: Time-Triggered Communication," International Standards Organization, Geneva, 2004.

[115] F.-L. Lian, J. Moyne, and D. Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," *IEEE Control Systems Magazine*, vol. 21, pp. 66 –83, 2001.

[116] C. Heegard, J. Coffey, S. Gummadi, P. Murphy, R. Provencio, E. Rossin, S. Schrum, and M. Shoemake, "High performance wireless Ethernet," *IEEE Communications Magazine*, vol. 39, pp. 64 –73, 2001.

[117] J. Regehr and J. A. Stankovic, "Hierarchical schedulers, performance guarantee, and resource management," *SIGOPS Oper. Syst. Rev.*, vol. 34, pp. 31–, 2000.

[118] F. Zhang and A. Burns, "Analysis of hierarchical edf pre-emptive scheduling," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 2007.

[119] FlexRay Consortium, *FlexRay Communications System – Electrical Physical Layer Specification, Version 3.0.1*, October 2010.

[120] FlexRay Consortium, *FlexRay Communications System – Protocol Specification, Version 3.0.1*, October 2010.

[121] M. Rausch, *FlexRay : Grundlagen, Funktionsweise, Anwendung.* Hanser, 2007.

[122] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2003.

[123] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proc. of IEEE International Symposium on Circuits and Systems*, 2000.

[124] R. Cuninghame-Green, "Minimax algebra and applications," *Fuzzy Sets and Systems*, vol. 41, pp. 251 – 267, 1991.

[125] M. Daniel-Cavalcante, M. Magalhaes, and R. Santos-Mendes, "The Max-Plus Algebra and the Network Calculus," in *International Workshop on Discrete Event Systems*, 2006.

[126] B. De Schutter and T. van den Boom, "Max-plus algebra and max-plus linear discrete event systems: An introduction," in *International Workshop on Discrete Event Systems*, 2008.

[127] R. Cruz, "A calculus for network delay. I. Network elements in isolation ," *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, 1991.

[128] J. Le Boudec, "Application of network calculus to guaranteed service networks," *IEEE Transactions on Information Theory*, vol. 44, pp. 1087–1096, 1998.

[129] J.-Y. Le Boudec and P. Thiran, *Network Calculus - A Theory of Deterministic Queuing Systems for the Internet.* Springer, 2001.

[130] S. Gaubert and M. Plus, "Methods and applications of (max,+) linear algebra," in *STACS 97.* Springer, 1997.

[131] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems.* John Wiley & Sons, 1992.

[132] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[133] R. Baheti and H. Gill, "Cyber-physical systems," in *The Impact of Control Technology.* IEEE Control System Society, 2011.

[134] W. Wolf, "Cyber-physical systems," *Computer*, vol. 42, pp. 88–89, 2009.

[135] Committee on Networked Systems of Embedded Computers, National Research Council, *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. National Academic Press, 2001.

[136] D. Goswami, R. Schneider, A. Masrur, M. Lukasiewycz, S. Chakraborty, **H. Voit**, and A. M. Annaswamy, "Challenges in automotive cyber-physical systems design," in *Proc. of the International Conference on Embedded Computer Systems: Architecture, Modeling and Simulation (SAMOS)*, 2012.

[137] W. Jiang, E. Fridman, A. Kruszewski, and J.-P. Richard, "Switching controller for stabilization of linear systems with switched time-varying delays," in *Proc. of IEEE Conference on Decision and Control (CDC)*, 2009.

[138] "The FlexRay Communication System Specifications, Ver. 2.1," www.flexray.com.

[139] A. Masrur, D. Goswami, R. Schneider, **H. Voit**, A. M. Annaswamy, and S. Chakraborty, "Schedulability analysis of distributed cyber-physical applications on mixed time-/event-triggered architectures with retransmissions," in *Proc. of IEEE Symposium on Industrial Embedded Systems (SIES)*, 2011.

[140] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," in *Proc. of Embedded World*, 2004.

[141] S. C. Talbot and S. Ren, "Comparision of FieldBus Systems CAN, TTCAN, FlexRay and LIN in Passenger Vehicles," in *Proc. of IEEE International Conference on Distributed Computing Systems Workshops (ICDCS)*, 2009.

[142] J. Banks, Ed., *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. Wiley-Interscience, 1998.

[143] A. Cervin and K.-E. Årzén, "TrueTime: Simulation tool for performance analysis of real-time embedded systems," in *Model-Based Design for Embedded Systems*. CRC Press, 2009.