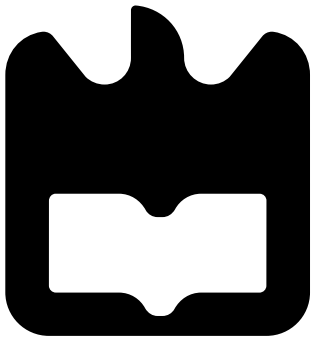
Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática,
2014

Milton Armando
Cunguara

Sobre a aplicação de Técnicas de Controlo em Redes Industriais com Falhas

On the Application of Optimal Control Techniques in Lossy Fieldbuses



Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2014

Milton Armando Cunguara

Sobre a aplicação de Técnicas de Controlo em Redes Industriais com Falhas

On the Application of Optimal Control Techniques in Lossy Fieldbuses

"New problems cannot be solved using the old thinking that created them"

— Albert Einstein

Milton Armando Cunguara

Sobre a aplicação de Técnicas de Controlo Óptimo em Redes Industriais

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Electrotécnica, realizada sob orientação científica dos Professores Doutor Tomás António Mendes Oliveira e Silva, Professor associado do Departamento Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e Paulo Bacelar Reis Pedreiras, Professor Auxilar do mesmo Departamento.

o júri / the jury

presidente / president	**António Manuel Rosa Pereira Caetano** Professor Catedrático da Universidade de Aveiro (por delegação do Reitor da Universidade de Aveiro)
vogais / examiners committee	**Ana Lusa Lopes Antunes** Professora Adjunta do Instituto Superior Técnico de Setúbal
	Luís Miguel Pinho de Almeida Professor associado da Universidade do Porto
	Alexandre Manuel Moutela Nunes da Mota Professor Associado da Universidade de Aveiro
	Tomás António Mendes Oliveira e Silva (orientador) Professor Associado da Universidade de Aveiro

agradecimentos / acknowledgements

. . . à todos os que me apoiaram durante este percurso, em especial, aos meus familiares.

Aos meus orientadores. . .

Aos membros do júri por terem encontrado tempo para rever e comentar esta humilde tese.

Ao governo da republica portuguesa, que através da Fundação para a Ciência e a Tecnologia, por meio da Bolsa de Doutoramento de contracto número SFRH/BD/62178/2009/J53651387T4E, ajudou financeiramente a realização desta tese. Igualmente, agradeço ao governo da república portuguesa, que através do Projecto Serv-CPS -PTDC/EEA-AUT/122362/2010, prestou ajuda financeira que conduziu a aquisão de bens materiais que foram de vital importância para a execução desta tese.

Á todos os que não tendo sido acima mencionados, tenham contribuído positivamente para a realizacção desta tese.

Palavras-chave Controlo, Controlo Óptimo, Controlo em Rede, Redes Industriais, Redes com Falhas

Resumo

Em resultado de várias tendências que têm afetado o mundo digital, o desempenho das redes de comunicação em tempo-real está continuamente a ser melhorado. No entanto, tais tendendências não só introduzem melhorias, como também introduzem uma série de não idealidades, tais como a latência, o *jitter* da latência de comunicação e uma maior probabilidade de perda de pacotes.

Esta tese tem o seu cerne em falhas de comunicação que surgem em tais redes, sob o ponto de vista do controlo automático. Concretamente, são estudados os efeitos das perdas de pacotes em redes de controlo, bem como arquitecturas e técnicas óptimas de compensação das mesmas.

Primeiramente, é proposta uma nova abordagem para colmatar os problemas que surgem em virtude de tais perdas. Essa nova abordagem é baseada no envio simultâneo de vários valores numa única mensagem. Tais mensagens podem ser de sensor para controlador, caso em que as mesmas são constituídas por um conjunto de valores passados, ou de controlador para actuador, caso em que tais mensagens contêm estimativas de futuros valores de controlo. Uma série de testes revela as vantagens de tal abordagem.

A abordagem acima explanada é seguidamente expandida de modo a acomodar o controlo óptimo. Contudo, ao contrário da abordagem acima apresentada, que passa pelo não envio deliberado de certas mensagens com vista a alcançar um uso mais eficiente dos recursos de rede; no presente caso, as técnicas são usadas para reduzir os efeitos da perda de pacotes.

Em seguida são estudadas abordagens de controlo óptimo que em situações de perda de pacotes empregam formas generalizadas da aplicação de valores de saída. Este estudo culmina com o desenvolvimento de um novo controlador óptimo, bem como a função, entre as funções generalizadas do funcionamento do actuador, que conduz o sistema a um desempenho óptimo.

É também apresentada uma linha de investigação diferente, relacionada com a oscilação da saída que ocorre em consequência da utilização de técnicas e algoritmos clássicos de co-desenho de controlo e redes industriais. O algoritmo proposto tem como finalidade permitir que tais algoritmos clássicos possam ser executados sem causar oscilações de saída, oscilações que por sua vez aumentam o valor da função de custo. Tais aumentos da função do custo, podem, em certas circunstâncias, pôr em causa os benefícios da aplicação das técnicas de co-desenho clássico.

Resumo (Continuação)

Numa outra linha de investigação, foram estudadas formas, mais eficientes que as contemporâneas, de geração de sequências de execuções de tarefas que garantam que pelo menos um dado número de tarefas activadas serão executadas por cada conjunto contíguo composto por um número predefinido de activações. Esta técnica poderá, no futuro, ser aplicada na geração dos padrões de envio de mensagens que é empregue na abordagem de utilização eficiente dos recursos de rede acima referida. A técnica proposta de geração de tarefas é melhor que as anteriores no sentido em que a mesma é capaz de escalonar sistemas que não são escalonáveis pelas técnicas clássicas.

A tese tambm apresenta um mecanismo que permite fazer o encaminhamento multi-caminho em redes de sensores sem fios com falhas sem causar a contagm em duplicado. Assim sendo a mesma técnica melhora o desempenho das redes de sensores sem fios, tornando as mesmas mais maleável as necessidades do controlo aumático em redes sem fios.

Como foi referido acima, a tese foca-se em técnicas de melhoria de desempenho de sistemas de controlo distribuído em que os vários elementos de controlo encontram-se interligados por meio de uma rede industrial que pode estar sujeita a perda de pacotes. As primeiras três abordagens cingem-se a este tema, sendo que primeiras duas olham para o problema sob um ponto de vista arquitetural, enquanto que a terceira olha sob um ponto de vista mais teórico. A quarta abordagem garante que outras propostas que podem ser encontradas na literatura e que visam atingir resultados semelhantes aos que se pretendem atingir nesta tese, possam fazê-lo sem causar outros problemas que invalidem as soluções em questão. Seguidamente, é apresenta-se uma abordagem ao problema proposto nesta tese que foca-se na geração eficiente de padrões para subsequente utilização nas abordagens acima referidas. E por fim, apresentar-se-a uma técnica de optimização do funcionamento de redes sem fios que promete melhorar o controlo em tais redes.

Palavras-chave

Control, Optimal Control, Networked Control, Real-time Networks, Fieldbuses, Lossy Networks

Abstract

The performance of real-time networks is under continuous improvement as a result of several trends in the digital world. However, these tendencies not only cause improvements, but also exacerbates a series of unideal aspects of real-time networks such as communication latency, jitter of the latency and packet drop rate.

This Thesis focuses on the communication errors that appear on such real-time networks, from the point-of-view of automatic control. Specifically, it investigates the effects of packet drops in automatic control over fieldbuses, as well as the architectures and optimal techniques for their compensation.

Firstly, a new approach to address the problems that rise in virtue of such packet drops, is proposed. This novel approach is based on the simultaneous transmission of several values in a single message. Such messages can be from sensor to controller, in which case they are comprised of several past sensor readings, or from controller to actuator in which case they are comprised of estimates of several future control values. A series of tests reveal the advantages of this approach.

The above-explained approach is then expanded as to accommodate the techniques of contemporary optimal control. However, unlike the aforementioned approach, that deliberately does not send certain messages in order to make a more efficient use of network resources; in the second case, the techniques are used to reduce the effects of packet losses.

After these two approaches that are based on data aggregation, it is also studied the optimal control in packet dropping fieldbuses, using generalized actuator output functions. This study ends with the development of a new optimal controller, as well as the function, among the generalized functions that dictate the actuator's behaviour in the absence of a new control message, that leads to the optimal performance.

The Thesis also presents a different line of research, related with the output oscillations that take place as a consequence of the use of classic co-design techniques of networked control. The proposed algorithm has the goal of allowing the execution of such classical co-design algorithms without causing an output oscillation that increases the value of the cost function. Such increases may, under certain circumstances, negate the advantages of the application of the classical co-design techniques.

Abstract (Continuation)

A yet another line of research, investigated algorithms, more efficient than contemporary ones, to generate task execution sequences that guarantee that at least a given number of activated jobs will be executed out of every set composed by a predetermined number of contiguous activations. This algorithm may, in the future, be applied to the generation of message transmission patterns in the above-mentioned techniques for the efficient use of network resources. The proposed task generation algorithm is better than its predecessors in the sense that it is capable of scheduling systems that cannot be scheduled by its predecessor algorithms.

The Thesis also presents a mechanism that allows to perform multi-path routing in wireless sensor networks, while ensuring that no value will be counted in duplicate. Thereby, this technique improves the performance of wireless sensor networks, rendering them more suitable for control applications.

As mentioned before, this Thesis is centered around techniques for the improvement of performance of distributed control systems in which several elements are connected through a fieldbus that may be subject to packet drops. The first three approaches are directly related to this topic, with the first two approaching the problem from an architectural standpoint, whereas the third one does so from more theoretical grounds. The fourth approach ensures that the approaches to this and similar problems that can be found in the literature that try to achieve goals similar to objectives of this Thesis, can do so without causing other problems that may invalidate the solutions in question. Then, the thesis presents an approach to the problem dealt with in it, which is centered in the efficient generation of the transmission patterns that are used in the aforementioned approaches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Control theory is a stand-alone discipline for more than a century, having separated itself from pure Mathematics. Unsurprisingly, nowadays many of the initial challenges of Control theory have been overcome and the discipline has reached a state of maturity. Notwithstanding, there are new problems, arising from a variety of technological issues, that are not solvable by the standard control approaches. This Thesis focus on one of such problems, namely, the problem of control on lossy networks.

Meanwhile, information technologies in general and networking in particular, experienced astronomical improvements in price, performance and reliability over the last 5 decades. This improvement rate continues unabated for the foreseeable future, thus promising new and more capable devices. Beyond the fact that these improvements bring about more capable devices, they also open the possibility for new types of devices. One such novelty is the possibility of using (general purpose) networks in control, which is at the heart of this Thesis.

In fact, these new developments in networking propelled the development of a new type of control system, namely distributed control systems. These new systems take advantage of the above mentioned novel network architectures that are expected to provide a seamless, cheaper and easier to control system. Advantages that were readily incorporated by the more profit-driven users of control systems, namely the industrial sector.

However, most control networks are not simple mathematical objects with ideal characteristics, such as no jitter/latency and no packet drops, nor can be approximated to networks that have such characteristics. Therefore, it is paramount to find alternative ways to conjugate both sides of the problem in order to achieve a solution that is closer to the intended goals of the global system.

This problem, though of easy statement, has a complex solution, which motivated its partition into several smaller problems regarding aspects of network and/or control, each of which, in turn, has a whole research community devoted to it.

1.1 Motivation

The disciplines of control and networking (more specifically fieldbuses) have historically been disjoint, not having any sort of direct cooperation between them. Both control theory and practice assume the use of an ideal (control) network, devoid of any limitations imposed by the physical world. Evidently, such type of network only exists in the mathematical (models and) abstractions that are made in order to reduce the problem of controller design

into a tractable one.

Similarly, network designers assume that all imperfections that arise from their work can be accounted for, or at least tolerated by, the network users. This assumption fails for a number of reasons, the most important ones being 1) even though it is possible to know in advance the type of deviations from a given set of ideal points it is not simple to determine its amount, i.e., it is more qualitative than quantitative, and 2) the deviations are usually stochastic which means that they can be dealt with in a stochastic sense, but their effects cannot be completely canceled.

With these limitations in mind, it is possible to envision a possible design framework in which the control informs the network on which types of deviations cause less errors. This is paramount because the different network designs cause different types of control errors, hence, even though it is not possible to make an ideal network, it is possible to make a real network in which the deviations from the ideal have a smaller impact on control. Furthermore, the network design can inform the control of which types of deviation are more likely to happen as well as give a description of such deviations, thereby allowing the design of controllers that attenuate the type of maladies that are more likely to happen.

This approach had already been tried before for the case of network latency/jitter and for the problem of sampling period assignment, being under the *umbrella* of co-design. Though the **current Thesis** is based on a similar concept there are a number of aspects in which it is considerably **different**:

The object of study — in this Thesis the **networks** aspects that are under study are their **failures**, how they affect control and what can be done to minimize their effects.

The methodology — this Thesis aims at methods that achieve the aforementioned goals, while considering questions regarding **optimality** of choices and proposals and not limiting itself to the proposal of new protocols (by processes similar to trial and error) but also concerning itself with the **formal proofs** of or justifications for the respective choices. In a sentence, this Thesis aims at having elements of both science and engineering, as opposed to only the latter.

1.2 Assumptions

Due to the physical systems nature of the systems that are the object of this Thesis, it is necessary to make a number of abstractions that allow for simpler models. Such models are, normally, of easier mathematical treatment resulting in (mathematically) proven optima. Such results are then incorporated into the original physical systems.

Nonetheless, it must be stressed that, even though assumptions are made, the current Thesis presents several contributions that allow for the relaxation of the assumptions usually made about the respective systems.

The general assumptions made on this Thesis are:

Linear dynamics — all systems are either linear or can be transformed into a system with linear dynamics, though not necessarily with linear input and/or output.

Time invariance — the behavior of the systems are independent of the instant in which the experiment takes place.

Statistically treatable network — the network is a channel with well defined, though not necessarily known, upper bounds for packet drop rate, jitter and latency and they are not necessarily equal at all of its streams (e.g. due to an implementation of Quality of Service).

Channel independence — each of the messages, though sent through the same channel, has a set of characteristics that are independent of each other.

Job schedulability — all jobs that are set up for execution meet their respective deadline.

Beyond these general assumptions, it is also assumed throughout the Thesis, a very specific type of network. This network is comprised of three types of devices: sensors, controllers and actuators. Sensors send their data to controllers, and not vice-versa. Similarly, controllers transmit control values to actuators. Furthermore, as pointed out above, the network need not be perfect, though, only the packet drop will be studied in detail. It may be considered that the network provides a mechanism for acknowledging the reception of a given message. But such acknowledgements are done completely at lower levels of the network, as to not violate the assumptions regarding which groups of nodes transmit and which receive. Figure 1.1 depicts such an architecture with the respective perturbations that emerge from sampling and actuation, and with a representation of the discretization process.

1.3 The Thesis

There are several possible ways in which a network may differ from an ideal communication channel. This work addresses only a number of them. In its essence, this work discusses the techniques for and the conditions under which it is possible to improve the control over lossy networks. Specifically, this work supports the following thesis:

> It is possible to significantly increase, comparing with methods present in the literature, the overall quality of control in networks with failures by using estimators and/or more complex control laws. It is also possible to use this type of control techniques to reduce resource usage, i.e., the number of messages (hence the bandwidth), of control communications in networks without network errors such as with the employment of $(m, k)-$firm schedulers and/or switched controllers with an appropriate framework that allows them to switch without causing an oscillation. Moreover, it is possible to improve the delivery rate of networks with high message drop rate, hence of the control network built on top of them, using

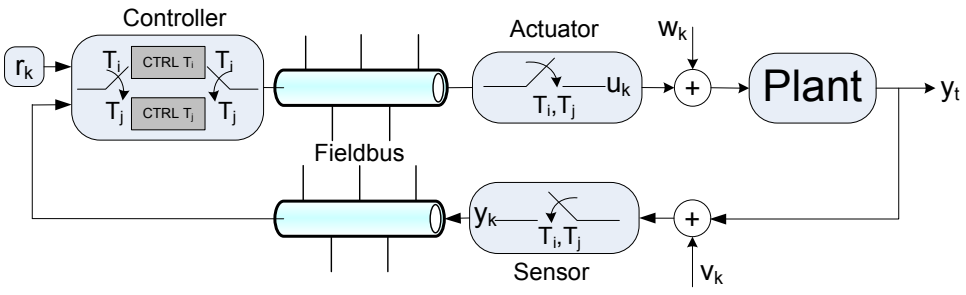


Figure 1.1: Control architecture.

a combination of multi-path routing and algorithms that allow to reconstruct the data with a low number of message exchanges.

The results of the Thesis are established using a variety of methods. Some of the parts of the Thesis are proven on purely mathematical grounds and then tested to assert their validity in practical settings. Other parts are postulated and then showed to be true (proven right) with a lower degree of mathematical abstraction. Furthermore, the Thesis is established by proving a number of intersecting lines regarding control over communication networks with packet losses.

One such line is the **use of estimators**. This Thesis will provide the basic conditions on which the use of estimators can result into a lower bandwidth consumption. Furthermore, it shows that it is possible to change the **actuation rate** without changing the **sampling rate** and vice-versa. This is crucial because provided that the sampling is done at a rate that allows for the correct estimation, then the quality of control can still be increased by increasing the control rate. This has some far reaching implications, such as the fact that the number of messages among the various devices in a distributed control system can be reduced while maintaining the quality of control for as long as it is found a mechanism to generate at the actuator the new control values (which are also provided in this work).

The Thesis proves empirically that the use of estimators in the controller of a distributed control system can improve substantially the quality of control. The Thesis also establishes mathematically the **optimal control law in networks with failures** in a particular type of output strategy. This new control is extended in a way that it produces the classical control laws as well as the control law of another output strategy, which is present in the literature. This extension produces a new parameter which when minimized yields a controller that produced output values that are similar to the values produces by the use of estimators, thereby both proving mathematical optimality of the use of estimators and proving the existence of a **duality** between the use of **estimators in the controller** and the **introduced parameter in the output**.

The Thesis will also discuss aspects related with the **certainty equivalence principle** of control in **networks with failures**, thereby setting boundaries to the amount of knowledge that can be extracted from the current control framework while giving directions into a possible way of creating an optimal controller for more restricted situations.

The possibility of using controllers in error laden scenarios brings back the possibility of using **(m,k)-firm schedulers** (schedulers in which at least m messages out of any consecutive group of k messages are successfully scheduled), or at least to model the successful delivery of control messages as an (m,k)-firm process. In this aspect, the contribution of this Thesis is two fold: first a new (m,k)-firm scheduler was devised that 1) provides **harder guarantees** and 2) achieves a **higher cpu/network utilization**, and second the introduction of the optimal scheduler gives the first steps in the direction of proposing an **optimal control law** for this type of scenarios.

The resource optimization normally requires the use of multiple controllers, each one tuned for a given scenario, which are switched as the condition of the overall control network changes. However, this normally introduces an oscillation in the output. This Thesis provides a mechanism, which is both mathematically and simulation wise proven, to guarantee **oscillations free controller change**.

At last, a mechanism that introduces and makes an **efficient exploitation of path redundancy on multi hop communication networks** is also presented. Such mechanism

has a potential to simplify the use of controllers in communication networks composed of several hops.

1.4 Structure of the Dissertation

The remaining of the Dissertation is organized as follows: the **first part** of this Dissertation deals with basic and **introductory concepts**. These concepts will be fundamental at establishing the nature of the contribution that is presented in the following chapters. The first part of the document has two chapters (not including the present chapter), one covering control related topics and another covering topics in network and real-time scheduling. The second part of this document deals with the Dissertation *per se*. All chapters in the second part of this document include an initial state-of-the-art of the topic in question as well as some form of validation at their respective ends.

The **first part** of this document is organized as follows:

Chapter 1 is this chapter, which provides an overview of this document and presents the Thesis.

Chapter 2 starts with an introduction to control and ends with a survey of topics in modern control.

Chapter 3 provides an overview of topics related to modern real-time systems and networks.

The **second part** of this document is organized as follows:

Chapter 4 deals with the use of estimators for the reduction of control bandwidth.

Chapter 5 deals with the control over lossy networks.

Chapter 6 deals with control systems with multiple controllers in which the change of controllers cause an output oscillation. A solution for this problem is presented and validated.

Chapter 7 deals with $(\mathrm{m}, \mathrm{k}) -$ firm systems. More specifically, the creation of a deterministic (m,k)-firm scheduler that can successfully fulfill all of its (requirements) properties, which is motivated by the fact that none of the schedulers found in the literature had all the properties sought for.

Chapter 8 presents a novel way to increase the probability of a value of a given node in a wireless sensor network reach the (or a, according to the case) node that communicates with the external network.

1.5 Contributions of this Thesis

The specific contributions of this Thesis are:

Use of buffers and estimators for bandwidth reduction — this mechanism is similar to the send-on-delta mechanism, in the sense that the various transmissions are triggered by crossings of some thresholds, but it differs from it because 1) the transmission thresholds are set on the state variable as opposed to the traditional input and/or output, 2) the

outputs are not set to constants but they vary in a way very similar to the way the outputs would vary if all messages were received and 3) the controller constructs an image of the system with a number of message exchange as low as possible.

Use of buffers and estimators to mitigate the effects of packet loss — this mechanism allows the actuator to have a reasonable estimation of the output value in the periods in which it does not receive any message from the controller. A similar effect takes place in the sensor to controller communication.

Analytical solution for the control over lossy networks with the hold strategy — this solution allows for an optimal application of this type of output strategy

Analytical extension for linear output types — a general optimal control over lossy networks in which the output is equal to the previous output times a multiplying matrix is given. Furthermore, the optimal value of the multiplying matrix itself is also derived. In doing so, it is proven that such generalized control is similar to the control using buffers and estimators, without the need for the transmission of the extra messages. Other differences are presented which help in the quantification of the effects of the transmission of control values.

Analytical solution for the problem of oscillation free controller change — this solved the dilemma that appears due to controller change causing oscillations that negated the gains of the controller change. The method is based on a change of basis matrix that is used to compute a novel state value at the switching time.

A novel deterministic (m,k)-firm scheduler — past(m,k)-firm schedulers were either stochastic, which *guarantee* the (m,k)-firm constraint in a stochastic manner, or were deterministic.

Multipath aggregation of duplicate sensitive summaries on WSN — this contribution addresses the problem that arise when aggregating values from different paths. It is solved by 1) finding an algorithm to break down the various aggregates and it says to each node which aggregate it should be sent and 2) a reconstruction algorithm that allows for nodes to make an aggregation that dismisses duplicates and is the best possible in the absence of a given message or set of messages.

1.5.1 List of Publications

- **Milton Armando Cunguara**, Tomás Oliveira e Silva, Paulo Bacelar Reis Pedreiras: "*On oscillation free controller changes. Systems & Control Letters*", 62(3): 262–268 (2013). **Journal**.

- **Milton Armando Cunguara**, Tomás António Mendes Oliveira e Silva, Paulo Bacelar Reis Pedreiras: "*On the control of lossy networks with hold strategy*". SIES 2013: 290–298, Porto, Portugal.

- **Milton Armando Cunguara**, Tomás Oliveira e Silva, Paulo Bacelar Reis Pedreiras: "*Optimal control in the presence of state uncertainty*". ETFA 2012: 1-4, Krakow, Poland

- **Milton Armando Cunguara**, Tomás António Mendes Oliveira e Silva, Paulo Bacelar Reis Pedreiras: "*A loosely coupled architecture for networked control systems*", IEEE International Conference on Industrial Informatics - INDIN , Lisbon, 2011.
- **Milton Armando Cunguara**, Tomás Oliveira Silva, Paulo Pedreiras: "*On Multi-Path Aggregation of Duplicate-Sensitive Functions*". 12 Conferncia sobre Redes de Computadores (CRC' 2012), November 15–16, 2012, Aveiro, Portugal.
- **Milton Armando Cunguara**, Tomás Silva and Paulo Pedreiras: "*Multipath Data Aggregation on WSN*", Workshop on Signal Processing Advances in Sensor Networks, in conjunction with the CPSWEEK'13, Philadelphia, PA, USA. April 8 2013;
- **Milton Armando Cunguara**, Tomás Oliveira e Silva, Paulo Pedreiras: "*A Path to Oscillation Free Controller Changes*", 9th IEEE International Workshop on Factory Communication Systems (WFCS' 2012), May 21–24, 2012, Lemgo/Detmold, Germany.

Chapter 2

Background

> If I have seen further it is by standing on the shoulders of giants.
>
> — Isaac Newton

This chapter presents a number of basic concepts that are paramount to fully understand the contributions that are made in the subsequent chapters. In a sense it *sets the stage* in which the Thesis will be *played*. The organization of this chapter reflects the organization of the Thesis which was put forward at the end of the previous chapter. Before entering the more advanced aspects of each subject, it is given a brief didactic introduction.

2.1 Principles of System Representation

For as long as the human species is capable of reasoning there is a necessity to think about the various aspects of life. All of them require that the thinking individual engages in a series of manipulations done over an internal representation of the object of thought. (In this context, internal refers to the fact that the thinking individual manipulates its representations and not the object that it is being represented).

In automatic control there is also a necessity to represent the various systems in a manner that is amenable to manipulations both by control designers as well as by the devices that will ultimately implement the control in question. Such representation is usually called *System Representation* and it is upon it that all of the control theory is developed.

In essence, a system representation is a mapping between a set of characteristic of the object being mapped and a series of characteristics of the objects that reside inside the thinking entity. This mapping is not always a bijection, i.e., a one to one correspondence, since only a finite set of characteristics (usually those of interest) may be mapped, but it is an endomorphism, which can be understood as being composed of two steps. The first is a bijective representation in which all aspects of the world are mapped. The second step which is applied to the intermediate mapping, removes all the aspects that are irrelevant to the entity that requires the mapping. Hence, it is an endomorphism because the representation does not have all the characteristics of the original system — endomorphisms are relations that stem from a given set to one of its proper subsets, or relations in which the codomain is contained in the domain. Moreover, each of the possible internal states of the thinking entity that is used to represent the system is called an **image of the system**.

The representations used in control systems are internal because changes in the representations do not (automatically) change the conditions of the systems. In fact, to modify their

surroundings, the controllers use devices called **actuators**. Similarly, due to the internal nature of such representation, the fact that a control system has a given representation of its surroundings does not cause its surroundings to be in that particular state. Hence there is a need to probe the environment to discover the values of such variables. The devices used to perform this task are called **sensors**. This notion is fundamental in control systems in general, but is more so in the context of this Thesis since the Thesis is about systems in which the actuator may or may not produce an output as well as the sensor may or may not produce a reading.

It is important to stress the difference between sensing and observing, in the context of automatic control. Sensing is related to the acquisition of a value associated to a particular variable, whereas observing is related to the acquisition of an image of a variable, in many cases by inference. Obviously, all aspects of a given system that can be sensed are observable, but the converse is not necessarily true. A similar relation exists between actuation and control.

An entity that controls a system, henceforth a **controller**, that cannot cause an influence, i.e., actuate, on the system that it is supposed to control or that cannot sense a system that it is supposed to observe is not of much use. It would simply perform a series of computation on its representations without any form of interaction with the physical world. The general effect of the network errors is a reduction of the coupling between both the internal representation (through sensing) and the desired physical state (through control) with the actual state of the physical world. This effect is one of the main object of study of this Thesis.

Even though a proper representation of the system to be controlled is essential to a successful control, as discussed above, it should be stressed that from the physical point-of-view, the controller is simply another element that receives a set of sensor readings and outputs a set of actuator values. Hence, for as long as the controller outputs the values that drive the system into the desired state, the controller's internal representations become irrelevant. Evidently, this last observation significantly relaxes the requirements of a representation.

2.1.1 Linearity

The physical world can be thought of as being made of a series of smaller subsystems. Each one of these subsystems obey a set of mathematical rules, of varying complexity, which are normally stated as equations. Due to such complexity, it is common to use simplified versions of these equations. A simplified version, or an approximation, produces results that are qualitatively similar but quantitatively inferior to the ones produced by the non-simplified equation. However, they have the benefits of simplicity, namely: ease of implementation from a computationally standpoint, ease of deduction, lower processing time, among others.

Evidently, this drive to simplicity makes the simplest systems the most used ones. One of the simplest types of systems, if not the simplest, are the linear ones. A system is said to be linear if it verifies the following two conditions:

Homogeneity A system is said to be homogeneous if it responds to the stimulus $u(t)$ with $y(t)$ then it also responds to the stimulus $\alpha u(t)$ with $\alpha y(t)$, $\forall \alpha \in \mathbb{C}$.

Superposition A system is said to obey the principle of superposition if it responds to the stimuli $u_1(t)$ and $u_2(t)$ with the stimuli $y_1(t)$ and $y_2(t)$ respectively, then it responds to the stimulus $u_1(t) + u_2(t)$ with the stimulus $y_1(t) + y_2(t)$.

In general a system is linear if it is possible to write its output in the form

$$y(t) = H\left[u(t)\right] \tag{2.1a}$$
$$= Ku(t) \tag{2.1b}$$

in which $u(t)$ is the input signal of the system, H is an operator that acts in signal $u(t)$ to produce another signal, $y(t)$ is the output signal and K is a factor that does not depend on $u(t)$.

If, in addition to these two properties, the system's response does not depend on the instant in which the stimuli are applied then it is said to be Linear Time Invariant (LTI).

Continuous-Time Representations

Due to the continuous nature at the scales normally used in control theory (i.e., ignoring quantum mechanical effects) the physical world is more readily described as a continuous-time system.

Continuous-time representations are characterized by the fact that the variable that represents *time* can take any real value. In practice, this range is limited by the time of interest (duration of the experiment, simulation, etc.) which sets lower and upper bounds. Nonetheless, the *time* variable is not constrained in the values that it can have in between.

Physical systems not only vary continuously in the time dimension, they also vary continuously in the amplitude of their respective variables. The rate of variation of a given variable, physical or otherwise, with the passage of time, in the Liebniz notation, is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}f(t) = \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{2.2}$$

which is the first derivative of the function $f(t)$. Higher order derivatives are denoted $\frac{\mathrm{d}^k}{\mathrm{d}t^k}f(t)$ with k the order of the derivative. In this Thesis only integer derivatives will be considered with the 0^{th} order derivative being the function itself and the negative order corresponding to multiple integrals, though, in general, such integrals seldom appear because the respective equations that give rise to them are rewritten by taking derivatives of an order that make all the integrals vanish. Continuous-time linear systems of this subclass, i.e., finite order with no delays and integral order derivatives only, have the general form

$$\sum_{j=0}^{N_B} \beta_j \frac{\mathrm{d}^j}{\mathrm{d}t^j} u(t) = \sum_{i=0}^{N_A} \alpha_i \frac{\mathrm{d}^i}{\mathrm{d}t^i} y(t) \tag{2.3}$$

in which N_A and N_B are finite, non-negative integer numbers characteristic of each system. Note that this equation can describe both Single-Input Single-Output (SISO) systems, if the respective coefficients are scalars or it can represent a Multiple-Input Multiple-Output (MIMO), if the respective coefficients are matrices.

Consider the definition of a new *operator*

$$p \overset{def}{=} \frac{\mathrm{d}}{\mathrm{d}t} \tag{2.4}$$

which allows Equation (2.3) to be written as

$$\sum_{j=0}^{N_B} \beta_j p^j u(t) = \sum_{i=0}^{N_A} \alpha_i p^i y(t) \tag{2.5}$$

in which $u(t)$ and $y(t)$ can be *removed* from inside the summation sign. This fact, in turn, induce the definition of two new polynomials, namely

$$B(p) = \sum_{j=0}^{N_B} \beta_j p^j \tag{2.6a}$$

$$A(p) = \sum_{i=0}^{N_A} \alpha_i p^i \tag{2.6b}$$

which further simplifies Equation (2.3) into

$$A(p)y(t) = B(p)u(t). \tag{2.7}$$

In most situations, it is assumed that $A(p)$ and $B(p)$ are co-prime polynomials, i.e., they have no common factors, since models with co-prime polynomials can be simplified into models without common factors that produce exactly the same input-output response, for example, by deconvolution of both polynomials by the common term. Furthermore, it is assumed (if not, then normalization is carried out) that the polynomial $A(p)$ is monic, i.e., the term associated with the highest order of p is 1 ($\alpha_{N_A} = 1$). However, this assumption is challenged in situations in which there is another exogenous variable (i.e., a variable that affects the system but that is not one of the *sensed* inputs) in the system. In fact, the presence of an exogenous variable leads the description into

$$A(p)y(t) = B(p)u(t) + C(p)e(t) \tag{2.8}$$

in which $e(t)$ is the exogenous variable and $C(t)$ is the *differential* polynomial associated to it. Note that the introduction of an exogenous variable changes the mutually prime assumption stated above into: it is assumed that $A(p)$, $B(p)$ and $C(p)$ are mutually prime polynomials, for the same reasons as in above.

The orders of these polynomials ($A(p)$, $B(p)$ and $C(t)$), namely N_A, N_B and N_C are not necessarily finite. However, in general, when they are not finite, it is still possible to approximate the polynomial, using Taylor function expansions into a number of other functions. The most common ones include the exponential (which corresponds to a (time) delay of the signal in question) and fractional derivatives. Nevertheless, in this Thesis it is assumed that N_A, N_B and N_C are all finite.

Moreover, it is also assumed that $N_A \geq N_B, N_C$, ensuring causality. In systems in which this condition is not verified, a response can temporal precede its cause. A system is said to be causal if, and only if, its responses always temporally succeed the inputs that cause them.

A given value p_p is said to be a system pole if $B(p_p) = 0$, i.e., if it nullifies the polynomial associated with the input. Similarly, a given value p_z is said to be a system zero is $A(p_z) = 0$, i.e., if it nullifies the polynomial associated with the output. More advanced aspects of representation of linear systems, such as delays, can be consulted, for example, in [CF03].

Discrete-Time Representations

Despite the continuous-time behavior of the physical world in general, there are a number of processes that are discrete in nature, for example the chess moves. These systems are also linear but are characterized by the fact that the *time* variable can only take a predefined set of values.

Discrete-time systems have a representation that is similar to the continuous-time representation, namely

$$A(q)y(k) = B(q)u(k) + C(q)e(k) \tag{2.9}$$

in which k is the discrete-time variable, $u(k)$ is the input signal, $e(k)$ is the exogenous signal, $y(k)$ is the output signal, q is the forward operator, i.e., $qx(k) = x(k+1)$, and $A(q)$, $B(q)$ and $C(q)$ are polynomials of degrees N_A, N_B and N_C in the variable q defined as

$$A(q) = \sum_{i=0}^{N_A} \alpha_i q^{-i} \tag{2.10a}$$

$$B(q) = \sum_{i=0}^{N_B} \beta_i q^{-i} \tag{2.10b}$$

$$C(q) = \sum_{i=0}^{N_C} \varsigma_i q^{-i}. \tag{2.10c}$$

Note that none of the polynomials has a term with a positive exponent of q, which is due to causality requirements. Note also that α_i, β_i and ς_i are not necessarily equal to their continuous-time counterparts, such those that emerge from Equation (2.8). Advanced topics, regarding the representation of discrete-time systems, can be consulted, for example, in [HRS07].

Test Signals

The linearity (homogeneity and superposition) property of the systems implies that if a given signal $u(t)$ produces an output $y(t)$ then $H(p)u(t)$ (with $H(p)$ a linear filter) produces an output equal to $H(p)y(t)$. This fact implies that the response of any other signal can be easily computed once the response for a well defined signal is known.

The signal best suited for serving as the *basic* input (test) signal is called an (unitary) *impulse distribution* and it can be defined, for example, by the following limits

$$\delta(t) = \lim_{A\to\infty} A \cdot rect\left(\frac{t}{A}\right) \tag{2.11a}$$

$$\delta(t) = \lim_{\sigma\to 0} \mathcal{N}(0,\sigma) \tag{2.11b}$$

in which $rect(t)$ is the rectangle function defined as

$$rect(t) = \begin{cases} 0 & \text{if } |2t| > 1 \\ 1 & \text{if } |2t| < 1 \\ \frac{1}{2} & \text{if } |2t| = 1 \end{cases}. \tag{2.12}$$

and $\mathcal{N}(\bar{x},\sigma)$ is the Normal distribution. The former definition has the advantage of being readily intelligible whereas the latter definition has the advantage of being infinitely differentiable. Nevertheless, the impulse can be defined as the limit of any other function in which the limit in question has unity area and is zero everywhere except at the origin. The impulse is related to the integral operator by the relation

$$\int f(t)\delta^{(n)}(\tau - t)\,\mathrm{d}t = f^{(n)}(\tau) \tag{2.13}$$

where $(\cdot)^{(n)}$ denotes the n^{th} repetitive derivative. Equation (2.13) is a very important property of the impulse because it implies that ($h(t) \stackrel{def}{=} H[\delta(t)]$)

$$H[u(t)] = H\left[\int u(\tau)\delta(t-\tau)\,\mathrm{d}\tau\right] \tag{2.14a}$$

$$= \int u(\tau)H[\delta(t-\tau)]\,\mathrm{d}\tau \tag{2.14b}$$

$$= \int u(\tau)h(t-\tau)\,\mathrm{d}\tau \tag{2.14c}$$

$$= u(t) * h(t) \tag{2.14d}$$

Equation (2.14a) follows directly from Equation (2.13) and Equation (2.14b) follows from the linearity of the integral, though there are some degenerate linear transformations that do respect this equality. However, these transformations either seldom appear in control practice or can be replace by a sequence of operators that do respect it, i.e., a work around, thus, such cases are outside of the scope of this Thesis. Equation (2.14c) follows from the definition of $h(t)$. Equation (2.14d) defines an operator called convolution. Equation (2.14) states that the response of a system to a given signal is equal to the convolution of the signal in question with the impulse response of the system.

Equation (2.14) can be rewritten as

$$H[u(t)] = H(p)u(t), \tag{2.15}$$

where $H(p) \stackrel{def}{=} \frac{A(p)}{B(p)}$ has the property that $A(p)\delta(t) = B(p)h(t)$. However, due to the superposition principle, the output of the system is also dependent of the initial values and of other input signals.

There are other test signals, some of them related with the impulse by the simple formula $p^n u(t) = \delta$. The simplest of such signals are the unitary (Heaviside) step, the unitary ramp and unitary parabola for $n = 1, 2, 3$ respectively, i.e.

$$u(t) = \begin{cases} \frac{t^n}{n!} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{2.16}$$

There are other less common test signals such as the (complex) exponentials which have a high importance for describing systems according to their frequency response. Such signals are not used enough times within this work to deserve an in depth description. Furthermore, such signals are test signals in the sense that they are used to (physically) *test* the response of systems and not to describe the properties of the systems based on the response to these signals.

Similarly to the continuous-time, it is possible to define a discrete-time impulse, i.e.

$$\delta(k) = \begin{cases} 0 & \text{if } k \neq 0 \\ 1 & \text{if } k = 0 \end{cases}. \tag{2.17}$$

The discrete-time impulse has properties of the continuous-time impulse, starting with

$$\sum_{\kappa=-\infty}^{\infty} f(\kappa)\delta(k-\kappa) = f(k), \tag{2.18}$$

which, using arguments similar to the continuous-time case, can be proven that

$$H\left[u(k)\right] = \sum_{\kappa=-\infty}^{\infty} u(\kappa)h(k-\kappa) \tag{2.19a}$$

$$= u(k) * h(k). \tag{2.19b}$$

It is also possible to define a discrete-time *unitary* family of signals associated with the impulse and it is done so according to the formula $(1-q^{-1})^n u(k) = \delta(k)$. The first three elements of this family are the discrete-time unitary (Heaviside) step, the unitary ramp and the unitary parabola, for $n = 1, 2, 3$ respectively. Similarly it is possible to define the (complex) exponentials and other test signals discrete-time domain.

A comprehensive analysis of test signals, both in continuous-time as in discrete-time, is provided in [CF03].

2.1.2 Discretization

As stated in previous subsection, at the scales relevant for contemporary control practice, most natural phenomena are continuous-time in nature. Nonetheless, in the previous decades there have been an explosion in microprocessor's price-performance ratios, which led to their wide spread adoption. Since microprocessors are discrete-time computational devices, it created a need to obtain a discrete representation of continuous-time processes.

The (uniform) discretization of a signal consists in defining a second signal

$$s_d(k) = s_c(kT_s + \varphi), \tag{2.20}$$

with s_c the continuous-time signal, s_d the discrete-time signal, k the discrete-time variable which represents the sample sequence number, T_s the sampling period and φ a phase in the sampling time which normally is equal to zero. Under certain conditions, i.e., if it appears due to a delay between an action and a response, φ is called dead time. Note that beside its application to denote the time difference between the instant in which a value is in the output to the instant in which it is discretized — i.e., sampling time, the term dead-time is also used to denote the time between the application of a given control value and the appearance of its response in the output.

This process is similar to the sampling process described by the Nyquist/Shannon sampling theorem which requires the sampling frequency (defined as $F_s T_s = 1$) to be at least twice the highest frequency contained in the signal Fourier Transform. However, the type of signals that usually appear on control systems include signals for which there is no frequency for which they become zero. In fact, all forms of reconstruction are based on the system's input/output response plus the knowledge of the value of the signal in a number of (discrete) points. This fact renders the Nyquist/Shannon sampling theorem of little use.

Output Signal "Analogization"

In digital systems there is a need to apply the discrete-time input signal into the physical world. Assuming that linearity is preserved (using the arguments presented above), any such *output* of the input signal can be produced by applying a static value, equal to the discrete-time input value at the beginning, into an analog filter for the duration of a period, which implies that

$$A(p)y(t) = B(p)O(p)u_s(k), \quad kT_s + \varphi < t \leq (k+1)T_s + \varphi \tag{2.21}$$

with some boundary conditions on $y(kT_s + \varphi)$.

$O(p)$, in essence, defines interpolating values for u_s. Choices of $O(p)$ are limited by the amount of electronic instrumentation that can be used, which in turn is limited by another set of factors, such as, price, size of the marginal performance gain of using more electronics, desired level of integration, i.e., total component size, etc. However, due to their low-cost and simplicity, a family of output filters have gained prominence, namely the n^{th} order hold, which are defined as

$$(pT_s)^{n+1} O(p) = \left(1 - e^{-pT_s}\right)^{n+1}, \tag{2.22}$$

which implies that the signal at the input of the continuous-time system is

$$u(t) = u_s(k-1) + \left(\frac{t - (kT_s + \varphi)}{T_s}\right)^{n+1} (u_s(k) - u_s(k-1)), \quad kT_s + \varphi < t \leq (k+1)T_s + \varphi. \tag{2.23}$$

The last equation is called the backward or causal version, because it uses information regarding a sample from the past ($u_s(k-1)$). There is a similar approach that employs samples from the future $u_s(k+1)$ and is called forward n^{th} order hold and is in essence the past equation forwarded by one sample.

The most used order is zero (order hold). This choice appears to be too simplistic to be useful. However, its simplicity is the main reason why it reached such a wide adoption, i.e., it only requires that a value is maintained constant for a given amount of time. The other reason is that the *flaws* associated to its simplicity can be overcome by sampling the system at period shorter than the minimum necessary, also called oversampling, which is made possible by the ever increasing capacity of microprocessors.

The process of generating a continuous-time signal from a discrete-time signal is called Digital-to-Analog Conversion.

Derivative Approximations

The most simple approximations of the derivative operator are called forward and backward Euler operators and they discretize the continuous-time representation by approximating the derivative operator by a (linear) function of the delay operator. The approximations are

$$p \simeq \frac{1 - q^{-1}}{T_s} \tag{2.24a}$$

$$p \simeq \frac{q - 1}{T_s} \tag{2.24b}$$

with Equation (2.24a) corresponding to the backward (computed using a previous sample) Euler approximation and (2.24b) corresponding to the forward (computed using a future sample) Euler approximation.

With these transformations, and assuming that the p^n is approximated by applying the derivative approximation n times, then, the system defined in Equation (2.7) is discretized into

$$A\left(\frac{1 - q^{-1}}{T_s}\right) y(kT_s) = B\left(\frac{1 - q^{-1}}{T_s}\right) u(kT_s) \tag{2.25a}$$

$$A\left(\frac{q - 1}{T_s}\right) y(kT_s) = B\left(\frac{q - 1}{T_s}\right) u(kT_s) \tag{2.25b}$$

according to the backward and the forward Euler approximations respectively.

Another approach is to use the average of these two values to produce an Euler *mean* method, i.e.

$$p \simeq \frac{q - q^{-1}}{2T_s} \tag{2.26}$$

It can be proven that this approximation is equal to the Tustin, also called Trapezoidal or Bilinear approximation (see Equation (2.27)) in the sense that their transformations of continuous-time models into discrete-time models are always equal. The last statement can be proven by considering the approximation of the inverse of the derivative (the integral operator), i.e., the Euler mean approximation is the average of the approximation of the derivative of the two Euler approximations, whereas in the Tustin case the inverse of the derivative approximation is equal to the mean of the inverse of the Euler approximations. Despite both mean Euler and the Tustin approximations produce discrete-time approximations that have the same square error quality, they do not produce the same values of the $y(t)$, with the mean Euler assuming a constant value between two samples and the Tustin approximation assuming that $y(t)$ varies linearly between $y(kT_s)$ and $y((k+1)T_s)$.

$$p \simeq \frac{1 + q\frac{T_s}{2}}{1 - q\frac{T_s}{2}} \tag{2.27}$$

Another important aspect regarding these approximations concerns their respective stability. In fact, it can be proven that the discretization of systems using the forward Euler approximation does not preserve stability in the sense that there are stable continuous-time systems that are transformed by the forward Euler approximation into unstable discrete-time system. In the same line, both the backward Euler and the Tustin approximations preserve stability. However, in the inverse transformation, i.e., the transformation that turns the discrete-time approximation into the continuous-time system that originated it, the backward Euler is the only one that does not maintain stability. Therefore, of the three, the Tustin approximation is the only one that maintains stability in both cases, hence it is the most natural choice for an approximation, among the considered options, from a stability viewpoint.

A related issue is the quality of the approximation. It can be proven, for example, by performing a Taylor expansion of the systems response plus the fact that $q = exp(pT_s)$, that the two basic Euler approaches produce an error term of the orders greater than T_s whereas the Tustin approximation produces errors of orders greater than T_s^2, which is another aspect in which the latter approximation is superior.

Regarding the positions of the singularities, i.e., zeros and poles, in all three cases they are transformed according to the respective (transformation) approximation. However, the Tustin approximation introduces a new set of zeros in $q = -1$ with a multiplicity that makes the numerator and the denominator have the same order. One consequence of this fact is that in all such approximations, the poles and (some) zeros tend to the point $q = 1$. This causes some poles and zeros to have similar values, hence some poles and zeros approximately cancel each other, which in turn requires a microprocessor with higher bit precision.

A different approach to discretization is the *impulse invariant* method. The impulse invariant discrete-time system is defined as the discrete-time system to which a discrete-time impulse produces the same output as the discretized continuous-time impulse response. Under this approach, the continuous-time system can be written as (assuming that $B(q)$ has no pole

with multiplicity higher than one)

$$\frac{A(p)}{B(p)} = \sum_{i=0}^{N_B-1} \frac{\gamma_i}{p - p_i} \tag{2.28}$$

which coincides with its respective impulse response. The discretization of the signal given by the last equation is then

$$\frac{A_d(p)}{B_d(p)} = \sum_{i=0}^{N_B-1} \frac{qT_s\gamma_i}{q - e^{p_i T_s}} \tag{2.29}$$

which implies that the continuous-time poles are transformed according to $q_i = exp(p_i Ts)$ whereas the zeros are transformed in a rather more complex manner. However, as $T_s \to 0$, $q_i \to 1$ and it can be proven that as $T_s \to 0$, N_A zeros of the system tend to one.

The remaining $N_B - N_A$ (which is positive due to causality assumptions, furthermore is an integer if the system is rational) zeros tend to a different limit. In fact, as the zeros and poles tend to $q = 1$, an excess of poles (once more, due to causality assumptions) appear at $q = 1$. Then at this point the system behaves as if it was given by (with $n = N_B - N_A$)

$$\frac{A(p)}{B(p)} = \frac{1}{p^n}$$

which has an impulse response of:

$$\begin{aligned}
\frac{A_d(p)}{B_d(p)} &= \frac{1}{n!}\sum_{k=0}^{\infty}(T_s k)^n && \text{(2.30a)}\\
&= T_s^n \frac{(1-q^{-1})^n}{n!(1-q^{-1})^n}\sum_{k=0}^{\infty} k^n && \text{(2.30b)}\\
&= \frac{T_s^n}{n!(1-q^{-1})^n}\sum_{k=0}^{\infty}\sum_{j=0}^{n}(-1)^j\binom{n}{j}q^{-j}k^n && \text{(2.30c)}\\
&= \frac{T_s^n}{n!(1-q^{-1})^n}\sum_{j=0}^{n}\sum_{k=0}^{n}(-1)^j\binom{n}{j}q^{-j}k^n && \text{(2.30d)}\\
&= \frac{T_s^n}{n!(1-q^{-1})^n}\sum_{j=0}^{n}\sum_{k=0}^{j}(-1)^j\binom{n}{j}q^{-j}k^n && \text{(2.30e)}\\
&= \frac{T_s^n}{n!(1-q^{-1})^n}\sum_{j=0}^{n}\sum_{k=0}^{j}(-1)^j\binom{n}{j}q^{-j}(j-k)^n && \text{(2.30f)}
\end{aligned}$$

the first equality stems from the discretization of the impulse response of $\frac{1}{p^n}$ which is $\frac{t^{n-1}}{(n-1)!}$. The second equality is a simple multiplication (and division) by the known denominator inferred from the known multiple poles at $q = 1$. The third equality follows the application of the binomial theorem. The fourth equality stems from the fact that $\sum_{j=0}^{n}(-1)^j\binom{n}{j}(x-j)^i = 0$ for any $x \in \mathbb{C}, i \in \mathbb{N}$. This statement can be easily proven by induction. In fact, the last proposition implies that all partial sums should be zero, however, the impulse response of the system is zero for values of k (time variable) lower than zero, which breaks the symmetry and

allows the first terms of the partial sum to not be zero because they do not yet satisfy the above proposition. The fifth equality follows due to the fact delays of j samples are identically zero for the first $j-1$ terms of any causal system. The last equality follows by a reordering of the summation in k, i.e., from the end to the start.

The last equation completely defines the polynomial associated with the remaining zeros. There is the possibility to make *step invariant* or of higher order discretizations which basically consists of 1) defining the response of the system to the signal in question, 2) discretize the response signal, 3) finding the equivalent discrete model and finally 4) compensate for the input signal, i.e., finding a discrete system that when excited by the discrete version of the excitation signal generates the discretized output signal. However, regardless of the input signal, the over-sampling singularity clustering effect does not disappear.

An extensive discussion on the topics on the passage from the discrete-time domain to the continuous-time domain and vice-versa can be found in [Lev96].

2.1.3 Stability

There are many possible definitions of *Stability*. In fact, the word stable has a popular association with finiteness. However, there are problems with this definition, since any non identically zero linear (and most non-linear) system(s) can be made (in theory) to produce an infinity response by exciting them with a signal of infinite amplitude. This fact leads to a more rigid definition of stability, called Bounded-Input Bounded-Output (BIBO). Note that the term BIBO is applied to describe the system and not its current output, which can be made to change by changing the input.

Nevertheless, BIBO systems are not to be confused with well behaved systems, since being non chaotic, i.e., responded in a easy to visualize way, is one of the requisites for stability in popular notion of stability. In fact, a system does not have to always respond in the same manner to the same input signal, i.e., it may be stochastic, in order to be BIBO stable. The only criteria a system must meet in order to be BIBO stable is that it must produce a bounded response for all input signals.

A signal $u(t)$ (if continuous) or $u(k)$ (if discrete) is said to be bounded, according to a given norm if, and only if, there is a value M

$$|u(t)| \leq M, \forall t \in \mathbb{R} \tag{2.31a}$$

$$|u(k)| \leq M, \forall k \in \mathbb{Z} \tag{2.31b}$$

where $|\cdot|$ represents a suitable norm. The last equation means that a signal is bounded if its norm is never bigger than a certain value.

In the set of continuous-time rational linear functions, a function is BIBO if and only if all of its poles have a negative real part. For discrete-time rational functions, a function is BIBO if and only if all of its poles are inside the unity circle. This can be proven by showing that systems that are claimed to be unstable respond to certain bounded signals, such as the step functions, with an unbounded signal, and that the signals which are claimed to be bounded respond to bounded signals with bounded signals, provable by upper bounding the convolution that defines the output.

A notable case is that of the so called *marginally stable* systems which are systems in which there is/are single pole(s) at the boundaries (real part of the poles equal to zero or modulo of the poles equal to one, for continuous-time and discrete-time systems, respectively). Note

that there may be more than one such singularities, but each and every one of them must have multiplicity one. These systems are not BIBO because when excited by a signal that has a pole at that exact location, they produce a signal that grows linearly (in norm-2 squared) with time, hence being unbounded. However, they meet most criteria for stability when the particular type of signal is not introduced into the system. Systems that present this type of behavior are also called resonant. In fact, this term is used more broadly even for systems that have complex conjugate poles inside the region of convergence, but are relatively closed to the boundaries (also called under dumped).

Note, however, it is possible to excite a non BIBO system with a signal that makes it produce a bounded output. Theoretically, this can be achieved by producing a signal with an appropriate zero (with positive real part or outside of the unit circle according to the case), in essence doing what is called a zero-pole cancelling.

A less strict test of stability (of linear non-stochastic systems), and also more used, is the Root Mean Square (RMS) test of the impulse response, with some variants that include the so called impulse power test. If this value is zero, i.e., zero mean-power, then the system is stable. If it is finite but non-zero, then the system is marginally stable and if it is infinite, then it is unstable. The RMS of a signal is defined as, in the continuous and discrete-time respectively as

$$h_{RMS} = \lim_{T\to\infty} \sqrt{\frac{1}{T}\int_0^T |y(t)|^2 \,\mathrm{d}t} \tag{2.32a}$$

$$h_{RMS} = \lim_{k\to\infty} \sqrt{\frac{1}{k}\sum_0^k |y_\kappa|^2}. \tag{2.32b}$$

Note that the last test is not sensitive for continuous-time functions that contain $\delta(t)$.

Further introductory topics in stability can be found in [HRS07] and [CF03].

2.2 State Space Representations

There are many books that present a comprehensive view of state space representations. For a more in-depth than the approach presented in here see [CF03] or [HRS07]. For a full presentation, see [Lev96]. The latter reference also has a comprehensive presentation of Kalman filters.

The hitherto presented polynomial representation cannot take into account all relevant aspects of a given system. An example is an unstable system which is stabilized by putting on its input a compensator that cancels all destabilizing poles. Under such circumstances, the representations presented thus far would imply that the new system is stable. However, if there is a non-zero initial value, associated with an destabilizing pole, on the inputs of the systems/outputs of the compensator (stemming for example by the presence of a non-fully depleted capacitor), then the internal signal will grow unboundedly. This type of systems are called internally unstable.

Of course, it is possible to deal with this type of issues by using feedback or any other controller technique. However, from a system representation point-of-view, it shows that there are aspects that are not fully captured by the polynomial representation.

An alternative is the use of the so called state space representation. This representation has two *dynamic* equations. The first equation, also called the update equation, encapsulates

all of the aspects of the system dynamics, i.e., how the state evolves with time. In spite of the names of the respective equations, this is the only equation that has any form of dynamics. The second equation is called the output equation and it specifies how the (output) visible aspects of the system are produced from the system state. As for the state *per se* it is a variable that encapsulates all the history of the system up to that point. In other words, knowing the state of a system allows to compute the value of all variables of the system that can be determined by knowing the history of the system up to that point.

In the continuous-time linear case the dynamic equations are given by

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \check{\mathbf{A}}x(t) + \check{\mathbf{B}}u(t) \tag{2.33a}$$

$$y(t) = \check{\mathbf{C}}x(t) + \check{\mathbf{D}}u(t), \tag{2.33b}$$

where $x(t), u(t)$ and $y(t)$ are the state, input and output variables (signals) respectively, at instant t. These variables are in the general case, i.e., MIMO, column vectors with dimensions $(1 \times n), (1 \times q)$ and $(1 \times p)$ respectively. $\check{\mathbf{A}}, \check{\mathbf{B}}, \check{\mathbf{C}}$ and $\mathbf{D}$ are the transition, input, output and direct transmission matrices respectively with dimension $(n \times n), (n \times q), (p \times n)$ and $(p \times q)$ respectively.

Matrix $\check{\mathbf{A}}$ is responsible to transmit the signal from the current state to the next. Matrix $\check{\mathbf{B}}$ conveys the input into the future states. Matrix $\check{\mathbf{C}}$ describes how the state appears in the output, and matrix $\check{\mathbf{D}}$ conveys how the input can be/is propagated directly into the output. However, in causal systems with propagation delays — virtually all physical systems — matrix $\check{\mathbf{D}}$ is equal to a matrix of zeros of appropriate size, i.e., $\mathbf{0}_{p\times q}$. This notation (which will be used throughout this Thesis) means that it is a matrix with p lines and q columns in which all entries are 0.

The state space representation is invariant to the similarity transformation, i.e., given any non-singular, i.e., $\det(\mathbf{P}) \neq 0$, matrix $\mathbf{P}$, the transformation $\tilde{x}(t) = \mathbf{P}x(t)$ does not change the response of a system given its input. Nevertheless, the similarity transformation changes the values of the matrices according to (due to the simplicity of the transformation no proof is given)

$$\tilde{\mathbf{B}} = \mathbf{P}^{-1}\check{\mathbf{B}} \tag{2.34a}$$

$$\tilde{\mathbf{A}} = \mathbf{P}^{-1}\check{\mathbf{A}}\mathbf{P} \tag{2.34b}$$

$$\tilde{\mathbf{C}} = \check{\mathbf{C}}\mathbf{P}. \tag{2.34c}$$

This transformation is used in this Thesis to establish one of its results.

The response of the system state to an input signal can be easily computed,

$$px(t) = \check{\mathbf{A}}x(t) + \check{\mathbf{B}}u(t) \tag{2.35a}$$

$$px(t) - \check{\mathbf{A}}x(t) = \check{\mathbf{B}}u(t) \tag{2.35b}$$

$$x(t) = \left(p\mathbf{I}_n - \check{A}\right)^{-1}\check{\mathbf{B}}u(t). \tag{2.35c}$$

The output response is derived by recalling Equation (2.33b), leading to

$$y(t) = \check{\mathbf{C}}\left(p\mathbf{I}_n - \check{\mathbf{A}}\right)^{-1}\check{\mathbf{B}}u(t) + \check{\mathbf{D}}u(t) + \check{\mathbf{C}}e^{\check{\mathbf{A}}t}x(0). \tag{2.36}$$

in which $\mathbf{I}_n$ is the identity matrix of size n which is a square matrix (of size n) in which the diagonal entries are comprised by 1 and the remaining entries are all equal to zero. The $x(0)$

denotes the initial state and the term in question can be proven by repeating the analysis with an initial state and no input plus the application of the superposition principle. This equation proves the similarity invariance statement made before.

The last equation implies that the poles of the system are equal to the eigenvalues of matrix $\tilde{\mathbf{A}}$. Therefore, continuous-time, linear time-invariant systems are BIBO stable if all eigenvalues of the matrix $\tilde{\mathbf{A}}$ have negative real part; they are marginally stable if there are eigenvalues, with multiplicity one, with real part equal to zero and no eigenvalues with positive real part; they are unstable if there are eigenvalues with real part equal to zero and multiplicity higher than one or if there is at least one eigenvalue with positive real part.

It is also possible to write a discrete-time state space equation, see [HRS07] for more details,

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k) \tag{2.37a}$$
$$y(k) = \mathbf{C}x(k) + \mathbf{D}u(k), \tag{2.37b}$$

in which the various variables and matrices have meaning and sizes similar to their continuous-time counterparts. As with Equation (2.36), the output response in the discrete-time is

$$y(k) = \mathbf{C}\left(q\mathbf{I}_n - \mathbf{A}\right)^{-1}\mathbf{B}u(k) + \mathbf{D}u(k) + \check{\mathbf{C}}\check{\mathbf{A}}^k x(0). \tag{2.38}$$

Unlike the other representations, in the state space one, if the intersample behavior of the input is known, the effects of discretization can be fully taken into account, i.e., the discretized representation is not an approximation. Moreover, this Thesis focus on the discretization of continuous-time systems subject to inputs with a zero order hold. Under these circumstances and using Equation (2.36) (actually its state equivalent) with $\mathbf{D} = \mathbf{0}_{p\times q}$ and starting at time $t = kT_s$ (as opposed to $t = 0$), then (after solving the resulting differential equation)

$$x((k+1)T_s) = e^{\check{\mathbf{A}}T_s}x(kT_s) + \int_0^{T_s} e^{\check{\mathbf{A}}(T_s-t)}\mathrm{d}t\,\check{\mathbf{B}}u(kT_s) \tag{2.39}$$

in which the checked matrices $\check{(\cdot)}$ denote continuous-time matrices.

A comparison of Equation (2.37) with Equation (2.39) implies that

$$\mathbf{A} = e^{\check{\mathbf{A}}T_s} \tag{2.40a}$$
$$\mathbf{B} = \int_0^{T_s} e^{\check{\mathbf{A}}(T_s-t)}\mathrm{d}t\,\check{\mathbf{B}} \tag{2.40b}$$
$$\mathbf{C} = \check{\mathbf{C}}. \tag{2.40c}$$

The equality of the output matrices stems from the fact that the output equation does not possess any dynamics, hence it is not affected by the discretization process. Equations (2.40a) and (2.40b) can be rewritten as

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0}_{r\times n} & \mathbf{I}_r \end{bmatrix} = \exp\left(\begin{bmatrix} \check{\mathbf{A}} & \check{\mathbf{B}} \\ \mathbf{0}_{r\times n} & \mathbf{0}_{r\times r} \end{bmatrix} T_s\right). \tag{2.41}$$

2.2.1 Static Non-Linearity

The previous section assumed that the system was LTI. However, there are some non LTI systems in which the above theory can be extended into without significant changes. Such

systems have a so called static non-linearity, which means that the dynamic equation is still linear.

Among the static non-linear systems there are two non-disjoint subgroups: the Wiener and Hammerstein systems. The Wiener systems are characterized by a non-linearity in the input, but a linear dynamic. This is typical in industrial systems in which the input is normally comprised of motors with non-linear response. Such systems are normally linearized by dividing the system input-output response into a series of segments that are approximated by lines.

Hammerstein systems are characterized by a non-linear output sampling, i.e., the sampled value is a non-linear function of the state variable, but have a linear dynamic equation. Due to advances in microcontrollers, this type of systems are less common nowadays. In fact, modern digital-to-analog converters produce samples with non-linear components that are negligible for most applications.

Throughout this Thesis, it is assumed that the controlled system has a linear dynamics.

2.2.2 Linear Systems with Noise

Equation (2.33a) defined continuous-time (linear) system dynamics. However, it is rather common for physical systems to not evolve exactly as described by such equations. Some times due to perturbations, other times due to deficiencies of the model and most times due to a combination of both.

Nevertheless, it is possible to write the dynamic equations using a noise vector that accounts for the difference between the value produced by the model and the value of the physical system. This transforms Equation (2.33a) into

$$\mathrm{d}x(t) = \check{\mathbf{A}}x(t)\mathrm{d}t + \check{\mathbf{B}}u(t)\mathrm{d}t + \mathrm{d}\check{w}(t) \tag{2.42}$$

in which $\check{w}(t)$ is the variable that encapsulates all the aspects of the physical system that are not taken into account by the model. The reason why the last equation looks rather different from past dynamic equations (with respect to the derivative) is that the stochastic variable $\check{w}(t)$ is (in general) not differentiable. The solution of such equation, which is also a stochastic variable, is a stochastic integral which is one of the objects of study of the discipline of stochastic calculus and is out of the scope of this Thesis.

A causal (non causal) noise sequence is said to be white if its previous (all other) values cannot be used to predict the value of the current sample, i.e., in a white noise sequence, the samples are statistically independent. A color is attributed to a noise sequence according to the color of a filter that could potentially produce that noise sequence by filtering a white noise sequence. In turn, the color of a filter is the color that an optical filter, with a shape similar to the shape of the filter in question, would produce when illuminated with white light.

The auto-covariance of a given stochastic variable m is defined as

$$cov(m) = \mathbb{E}\left[(m - \mathbb{E}[m])(m - \mathbb{E}[m])'\right], \tag{2.43}$$

in which $(\cdot')$ denotes (Hermitian) transposition.

The variable $\check{w}(t)$ is itself modelled as being a zero mean, white gaussian noise with an auto-covariance of $\check{\mathbf{Q}}$. The *whiteness* assumption does not necessarily lead to a loss of

generality since under certain assumptions the $\check{w}(t)$ could be modelled as having a dynamic given by

$$\mathrm{d}\check{w}(t) = \check{\mathbf{A}}_w \check{w}(t)\mathrm{d}t + \mathrm{d}\check{\varepsilon}(t) \tag{2.44}$$

and under these circumstances Equation (2.42) can be rewritten as

$$\begin{bmatrix} \mathrm{d}x(t) \\ \mathrm{d}\check{w}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{I}_n \\ \mathbf{0}_{n\times n} & \mathbf{A}_w \end{bmatrix} x(t)\mathrm{d}t + \begin{bmatrix} \mathbf{B} \\ \mathbf{0}_{n\times q} \end{bmatrix} u(t)\mathrm{d}t + \begin{bmatrix} \mathbf{0}_{n\times n} \\ \mathbf{I}_n \end{bmatrix} \check{\varepsilon}\mathrm{d}t, \tag{2.45}$$

which is driven by the white noise $\check{\varepsilon}$.

Regarding the gaussian noise assumption, its non-validity is irrelevant, as will be seen below. Regarding the zero mean assumption, note that this is a standard noise assumption. Moreover, when it is not verified, it is said that there is an offset in the input which is normally (implicitly) corrected by $u(t)$.

System Noise Discretization

As discussed in the previous section, the input noise is stochastic. The discretization of the input noise should aim at generating a (stochastic) discrete sequence of noise terms that produces an effects on the state variable, at the sampling instants, that are similar to the effects of the continuous-time noise (at the same instants). However, the input noise is not known which makes the generation of a discrete noise sequence rather difficult. Instead, it is assumed that there is a (stochastic) discrete sequence which is modelled as a gaussian noise with known parameters, derived from the effects of the continuous-time input noise effects on the state variable.

Due to the assumptions regarding $\check{w}$ (i.e., that it is zero mean) and since the noise only undergoes linear operations then the discrete-time noise variable is also zero mean. Regarding its variance, according to the discussion on the previous paragraph

$$\mathrm{d}\hat{w}(t) = \mathbf{A}\hat{w}(t)\mathrm{d}t + \mathrm{d}\check{w}(t) \tag{2.46a}$$

$$w(kT_s) = \mathbf{0}_{n\times 1} \tag{2.46b}$$

$$w(k) \overset{def}{=} \lim_{\epsilon\to 0^+} w(kT_s - \epsilon) \tag{2.46c}$$

The first two lines define another noise sequence that evolves as the system noise but is reset at the beginning of every period. The discrete-time noise is equal to the value that the auxiliary noise sequence had right before the rest. This definition of the discrete-time noise implies that

$$E\left[w(k)w(k)'\right] = \lim_{\epsilon\to 0} E\left[w((k+1)T_s - |\epsilon|)w((k+1)T_s - |\epsilon|)'\right] \tag{2.47a}$$

$$= \lim_{\epsilon\to 0} E\left[\int_{kT_s}^{(k+1)T_s-|\epsilon|} e^{\mathbf{A}((k+1)T_s-t)}\check{w}(t)\left(e^{\mathbf{A}((k+1)T_s-t)}\check{w}(t)\right)'\mathrm{d}t\right] \tag{2.47b}$$

$$= \int_{kT_s}^{(k+1)T_s} e^{\mathbf{A}((k+1)T_s-t)} E\left[\check{w}(t)\check{w}(t)'\right] e^{\mathbf{A}'((k+1)T_s-t)}\mathrm{d}t \tag{2.47c}$$

$$= \int_0^{T_s} e^{\mathbf{A}(T_s-t)}\check{\mathbf{Q}}_w e^{\mathbf{A}'(T_s-t)}\mathrm{d}t \tag{2.47d}$$

$$= \int_0^{T_s} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'t}\mathrm{d}t \tag{2.47e}$$

For stable systems, it is possible to rewrite

$$E\left[w(k)w(k)'\right] = \int_0^{T_s} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'t}\mathrm{d}t \tag{2.48a}$$

$$= \int_0^{\infty} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'t}\mathrm{d}t - \int_{T_s}^{\infty} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'t}\mathrm{d}t \tag{2.48b}$$

$$= \int_0^{\infty} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'t}\mathrm{d}t - e^{\mathbf{A}T_s}\int_0^{\infty} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'t}\mathrm{d}t e^{\mathbf{A}'T_s} \tag{2.48c}$$

$$= \mathbf{W} - e^{\mathbf{A}T_s}\mathbf{W}e^{\mathbf{A}'T_s} \tag{2.48d}$$

in which

$$\mathbf{W} \stackrel{def}{=} \int_0^{\infty} e^{\mathbf{A}t}\check{\mathbf{Q}}_w e^{\mathbf{A}'T_s}\mathrm{d}t. \tag{2.49}$$

This method only works for stable systems because otherwise the integral that defines $\mathbf{W}$ does not converge. For non-stable (including marginally stable) systems, the covariance matrix is computed by breaking down the integral in question and solving it for each entry. This method has the disadvantage of not using the various tools available to solve the more specific types of equations. $\mathbf{W}$ itself can be computed by integrating the last equation by parts leading to

$$e^{\mathbf{A}T_s}\mathbf{W} + \mathbf{W}e^{\mathbf{A}'T_s} = \check{\mathbf{Q}}_w. \tag{2.50}$$

Equation (2.50) is an important and recursively appearing equation in Control Theory and it is called Continuous-Time Lyapunov equation. This equation is essential in one of the contributions of this Thesis.

There is another type of *perturbation* associated with an uncertainty in the output, i.e., sampling error. This uncertainty is in general not dynamic. However, the noise is possibly a colored sequence (for example, an Analog-to-Digital Converter used at a relatively high frequency introduces a small correlation between its readings). If such correlations exist then it is always possible to give this noise term a treatment similar to the treatment given to colored input perturbation, thus rendering the output noise white.

The whiteness of the output noise implies that its covariance matrix is independent of the sampling period since by definition having more data regarding a white variable does not lead to a decrease of the variance of its estimates. In fact, a more in-depth explanation would require the theory of Power Spectral Analysis, which is outside the scope of this Thesis. The overall effect of this sampling period independence is that it renders this perturbation easy to discretize because by definition its mean and covariance matrices are equal to their respective continuous-time counterparts. Nevertheless, if the sampling device is operated at a frequency high enough as to introduce a correlation between the readings, the correlated part can be modelled with a state equation, leading to a discretization that has the two types of components discussed above.

With these two new noise terms the state space discrete-time linear system is written as

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k) + w(k) \tag{2.51a}$$

$$y(k) = \mathbf{C}x(k) + v(k), \tag{2.51b}$$

with $v(k)$ the perturbation that arises from sampling: $v(k)$ is not a property of the system *per se*, it is a property of the sampling device, though it may causally affect the system if there is a form of feedback in which the output is used to compute the input.

Without lossy of generality, it is assumed that

$$E\left[\begin{bmatrix} w_j \\ v_j \end{bmatrix}\begin{bmatrix} w_k \\ v_k \end{bmatrix}'\right] = \begin{bmatrix} \mathbf{Q} & \mathbf{0}_{n\times r} \\ \mathbf{0}_{r\times n} & \mathbf{R} \end{bmatrix}\delta_{jk}, \tag{2.52}$$

where δ_{jk} is the Kroenecker delta, which is one if $j = k$ and zero otherwise. This assumption (no correlation between input and output perturbations) does not cause a loss of generality because it is always possible to decompose the output noise into two components, one correlated and another uncorrelated with the input noise respectively. Subsequently, the part of the output noise that is correlated with the input noise is *sent into the input* by removing it from the input and replacing it with an input noise that generates exactly the same (present and future) outputs, even though it would generate a different state. In short, the correlated part is made to belong only to the state.

2.2.3 Continuous-time Kalman Filter

The noise components discussed in the previous subsection have an effect in the amount of information (as measured by a suitable norm of the state auto-covariance matrix) that is known about the system, either due to the input noise causing an (or increasing the) uncertainty in future states and/or due to the output noise diminishing the amount of information obtained by sampling.

In most cases, the predictions of a model are useful only if the predictions are related to the physical variable that the model predicts. For this reason, whenever there is the possibility that the model predictions and the physical variables differ there is a need to measure and possible correct such differences.

An usual such measure, which is used throughout this Thesis, is the square of the so called norm-2 and is denoted

$$||e(t)||_2^2 = ||\hat{x}(t) - x(t)||_2^2 \tag{2.53a}$$

$$= \mathbb{E}\left[(\hat{x}(t) - x(t))\,(\hat{x}(t) - x(t))'\right] \tag{2.53b}$$

however, neither $e(t)$ nor the measure $||e(t)||^2$ are known. If they were known then it would be possible to *correct* the models prediction, so that it would become $\mathbf{0}_{n\times 1}$. Hence if such values were deterministic then they would be zero. This means, that they only make sense in expectation, hence a covariance matrix $||e(t)||^2$. A similar argument implies that the expectation of $e(t)$ (of a non biased) prediction error is always zero.

For the linear system in question, it implies that the dynamic equations, when applied to the expected state (note that the dynamic stochastic equation is equal to the dynamic equation of the physical system because the system is linear) leads to

$$\frac{\mathrm{d}\hat{x}(t)}{\mathrm{d}t} = \check{\mathbf{A}}\hat{x}(t) + u(t) \tag{2.54}$$

in which a new variable $\hat{x}$ signals the state estimation. Subtracting Equation (2.51a) from Equation (2.54) and remembering the definition of $e(k)$ yields,

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} = \check{\mathbf{A}}e(t) - w(t). \tag{2.55}$$

Note the presence of the minus sign in $w(t)$. Since the value of $w(t)$ is not known, for example, it could be $+1$ with a plus sign or a -1 with a minus sign, this sign is of little relevance leading many (to the best of the author's knowledge, all) classical theories to ignore it and assume that $w(t)$ itself had the opposite sign. However, in this Thesis it will be presented a result that requires that $w(t)$ to appear with different signs in the error equation and in the physical state equation, thus making it relevant.

Right multiplying Equation (2.55) by $e(t) - w(t)$, which is the integral of $\mathrm{d}(e(t) - w(t))$, for motives that will soon become apparent, leads to

$$\mathrm{d}e(t)\,(e(t) - w(t))' = \left(\check{\mathbf{A}}e(t)\mathrm{d}t - \mathrm{d}w(t)\right)(e(t) - w(t))' \tag{2.56a}$$

$$\mathrm{d}e(t)e'(t) = \check{\mathbf{A}}e(t)e'(t)\mathrm{d}t + \mathrm{d}w(t)w'(t). \tag{2.56b}$$

The last transformation is warranted due to the assumption that $w(t)$ is white, more precisely, since e is a causal function of values of w and by the whiteness assumption, the current value of w does not depend on its previous values, then $e(t)$ and $w(t)$ are uncorrelated. Nevertheless, once more, this assumption is general, in the sense that the state can always be augmented in a way that it becomes true. Continuing,

$$\mathrm{d}e(t)e'(t) + e(t)\mathrm{d}e'(t) = \check{\mathbf{A}}e(t)e'(t)\mathrm{d}t + e(t)(\check{\mathbf{A}}e(t))'\mathrm{d}t + \mathrm{d}w(t)w'(t) + w(t)\mathrm{d}w'(t) \tag{2.57a}$$

$$\mathrm{d}\left(e(t)e'(t)\right) = \check{\mathbf{A}}e(t)e'(t)\mathrm{d}t + e(t)e'(t)\check{\mathbf{A}}'\mathrm{d}t + \mathrm{d}\left(w(t)w'(t)\right) \tag{2.57b}$$

$$\mathrm{d}\check{\mathbf{P}}(t) = \check{\mathbf{A}}\check{\mathbf{P}}(t)\mathrm{d}t + \check{\mathbf{P}}(t)\check{\mathbf{A}}'\mathrm{d}t + \check{\mathbf{Q}}_w\mathrm{d}t \tag{2.57c}$$

$$\frac{\mathrm{d}\check{\mathbf{P}}(t)}{\mathrm{d}t} = \check{\mathbf{A}}\check{\mathbf{P}}(t) + \check{\mathbf{P}}(t)\check{\mathbf{A}}' + \check{\mathbf{Q}}_w. \tag{2.57d}$$

The first equality follows by adding its (own) transpose to Equation (2.56a). The second equality follows by the product rule (in reverse) of derivatives and is the reason why $\mathrm{d}(e(t) - w(t)$ was chosen as a factor. The third equality follows by taking expectations and the definitions $P(t) \overset{def}{=} E\left[e(t)e'(t)\right]$ and (the already presented definition of) $\check{\mathbf{Q}}_w$. The last equality follows by noting that the right side of the previous equation is a multiple of $\mathrm{d}t$ and that the derivative of $\check{\mathbf{P}}(t)$ is a well defined quantity.

The last equation is sometimes called the update (or innovation) equation because it defines how the error is updated. Other times it is called the *a prior* equation because it states what happens before a new measurement is assimilated by the model.

However, in continuous-time both the update of the covariance matrix and the *reception* of a new value happen in the same time instant, which suggests that a better approximation of the physical state is

$$\frac{\mathrm{d}\hat{x}(t)}{\mathrm{d}t} = \check{\mathbf{A}}\hat{x}(t) + u(t) + \check{\mathbf{K}}(t)\left(y(t) - \check{\mathbf{C}}\hat{x}(t)\right) \tag{2.58}$$

in which $\check{\mathbf{K}}(t)$ is a (time-dependent) gain matrix. The gain matrix is assumed to be linear due to the linearity of the underlying equation. Equation (2.58) implies that the prediction error evolves according to

$$\frac{\partial e(t)}{\partial t} = \check{\mathbf{A}}e(t) - \mathrm{d}w(t) + \check{\mathbf{K}}(t)\left(v(t) - \check{\mathbf{C}}e(t)\right), \tag{2.59}$$

which when treated as Equation (2.55) (multiplied by $\left(e(t) - w(t) + \frac{1}{2}\check{\mathbf{K}}(t)v(t)\right)'$, then added to its own transpose, group similar terms and derive in order to t), gives

$$\frac{\partial \check{\mathbf{P}}(t)}{\partial t} = \check{\mathbf{A}}\check{\mathbf{P}}(t) + \check{\mathbf{P}}(t)\check{\mathbf{A}}' + \check{\mathbf{Q}}_w - \left(\check{\mathbf{K}}(t)\check{\mathbf{C}}\check{\mathbf{P}}(t) + \check{\mathbf{P}}(t)\left(\check{\mathbf{K}}(t)\check{\mathbf{C}}\right)'\right) + \check{\mathbf{K}}(t)\check{\mathbf{R}}\check{\mathbf{K}}'(t). \quad (2.60)$$

It is possible to choose a value of the free parameter $\check{\mathbf{K}}(t)$ in order to minimize $\check{\mathbf{P}}(t)$. This is achieved by differentiating the last equation in order to $\check{\mathbf{K}}(t)$ and equating it to zero, i.e., (with $\check{\mathbf{R}} \stackrel{def}{=} E\left[v(t)v'(t)\right]$)

$$\frac{1}{2}\frac{\partial^2 trace\left(\check{\mathbf{P}}(t)\right)}{\partial \check{\mathbf{K}}(t)\partial t} = -\check{\mathbf{C}}\check{\mathbf{P}}(t) + \check{\mathbf{R}}\check{\mathbf{K}}'(t) = \mathbf{0}_{p\times n} \quad (2.61)$$

which is solved for

$$\frac{\mathrm{d}\check{\mathbf{P}}(t)}{\mathrm{d}t} = \check{\mathbf{A}}\check{\mathbf{P}}(t) + \check{\mathbf{P}}(t)\check{\mathbf{A}}' + \check{\mathbf{Q}}_w - \check{\mathbf{P}}(t)\left(\check{\mathbf{K}}(t)\check{\mathbf{C}}\right)' \quad (2.62a)$$

$$K(t) = \check{\mathbf{P}}(t)\check{\mathbf{C}}'\check{\mathbf{R}}^{-1}. \quad (2.62b)$$

In the last equation, $\check{\mathbf{P}}(t)$ was taken with a total derivative. That is due to the fact that it no longer is a function of $\check{\mathbf{K}}(t)$, but it is a function of $\check{\mathbf{P}}(t)$ itself, of some covariance matrices that for all effects and purposes are constants. Hence the only free variable is time.

Equation (2.62), which dictates the evolution of the state estimate covariance matrix, and Equation (2.58), which dictates the evolution of the state estimates, are called the Kalman-Bucy filter and they are the optimal way to filter continuous-time perturbations.

It is somewhat common to see Equation (2.62a) written as

$$\frac{\mathrm{d}\check{\mathbf{P}}(t)}{\mathrm{d}t} = \check{\mathbf{A}}\check{\mathbf{P}}(t) + \check{\mathbf{P}}(t)\check{\mathbf{A}}' + \check{\mathbf{Q}}_w - \check{\mathbf{K}}(t)\check{\mathbf{R}}\check{\mathbf{K}}(t)' \quad (2.63)$$

due to the equality $\check{\mathbf{K}}(t)\check{\mathbf{R}}\check{\mathbf{K}}(t) = \check{\mathbf{P}}(t)\left(\check{\mathbf{K}}(t)\check{\mathbf{C}}\right)'$ which is easily verifiable taking into account the definition of $\check{\mathbf{K}}(t)$.

2.2.4 Discrete-time Kalman Filters

Despite their names, the only Kalman filter that were proposed by Robert Kalman is the discrete version which will be presented in this section. The continuous-time version was introduced by Richard Snowden Bucy, but due to the similarities with the discrete version it also carries the Kalman name.

As in the continuous-time case, Equation (2.51) defines the dynamic of the state. And the dynamics of the state estimation is given by

$$\hat{x}(k+1|k) = \mathbf{A}\hat{x}(k|k) + \mathbf{B}u(k), \quad (2.64)$$

in which $\hat{x}(i|j)$ denotes the estimate of the state at instant i given all the information that was received up to the instant j.

Equation (2.64) implies that

$$e(k+1|k) = \mathbf{A}e(k|k) - w(k) \quad (2.65)$$

which when multiplied by its own transpose and using the definitions of the covariance matrices ($\mathbf{P}(k) \stackrel{def}{=} E\left[e(k)e'(k)\right]$ and of $\mathbf{Q}_w$ presented before)

$$\mathbf{P}(k+1|k) = \mathbf{APA}' + \mathbf{Q}_w. \tag{2.66}$$

Equation (2.64) and Equation (2.66) are called the update (or innovation or *a priori*) equations and together they define how the state estimation and its associated covariance matrix evolve from the estimation of the previous to the current.

The notation $\hat{x}(m+n|m)$ and $\mathbf{P}(m+n|m)$ denote the state estimation at instant $m+n$ given all the information about the system available up to instant m, i.e., $[u_0\, u_1 \cdots u_{n+m}]$ and $[y_0\, y_1 \cdots y_m]$. Note that the information points regarding the input goes up to $m+n$ whereas for the output goes only to m. For $n > 0$, these types of estimates are acquired by applying n (consecutive) times the update equations over $\hat{x}(m|m)$ and $\mathbf{P}(m|m)$. For $n < 0$, this leads to a special type of Kalman filter called Fixed-lag Smoothers.

Once $\hat{x}(k+1|k)$ and $\mathbf{P}(k+1|k)$ become known there is the problem of integrating the output measurement, i.e., $y(k)$, into $\hat{x}$ and $\mathbf{P}$. This can be achieved by once again assuming that linearity is not lost, i.e.,

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + \mathbf{K}_{k+1}\left(y(k+1) - \mathbf{C}\hat{x}(k+1|k)\right) \tag{2.67}$$

with a new free variable K_{k+1}. This choice has the added benefit of ensuring that the error is not a function of the state variable, which is an important property in control systems. With this definition of $\hat{x}(k+1|k+1)$, the error associated with state estimation becomes

$$e(k+1|k+1) = e(k+1|k) + \mathbf{K}_{k+1}\left(v(k+1) - \mathbf{C}e(k+1|k)\right), \tag{2.68}$$

which when right-multiplied by its own transpose and taking expectation yields

$$\mathbf{P}(k{+}1|k{+}1) = \left(\mathbf{I}_n - \mathbf{K}(k+1)\mathbf{C}\right)\mathbf{P}(k{+}1|k)\left(\mathbf{I}_n - \mathbf{K}(k+1)\mathbf{C}\right)' + \mathbf{K}(k{+}1)\mathbf{R}\mathbf{K}'(k{+}1). \tag{2.69}$$

This can be minimized with respect to $\mathbf{K}(k+1)$ by taking the derivative of $\mathbf{P}(k+1|k+1)$ in order to $\mathbf{K}(k+1)$ and equating it to zero, i.e.

$$\frac{\mathrm{d}\,\mathrm{Trace}\left(\mathbf{P}(k+1|k+1)\right)}{\mathrm{d}\mathbf{K}'(k+1)} = \left(\mathbf{I}_n - \mathbf{K}(k+1)\mathbf{C}\right)\mathbf{P}(k+1|k)\mathbf{C}' + \mathbf{K}(k+1)\mathbf{R} = \mathbf{0}_{n\times p}, \tag{2.70}$$

which is solved for

$$\mathbf{K}(k+1) = \mathbf{P}(k+1|k)\mathbf{C}'\left(\mathbf{C}\mathbf{P}(k+1|k)\mathbf{C}' + \mathbf{R}\right)^{-1} \tag{2.71}$$

which when substituted back into Equation (2.69) yields

$$\mathbf{P}(k+1|k+1) = \left(\mathbf{I}_n - \mathbf{K}(k+1)\mathbf{C}\right)\mathbf{P}(k+1|k). \tag{2.72}$$

Equations (2.67), (2.71) and (2.72) comprise the so-called correction step of the Kalman filter. They define how the prediction or update made from the previous state and the dynamic of the system is corrected by the noise outputs.

2.3 Control

The act of, and the need for, controlling the world is as old as the human kind itself. It has been the main drive behind tool building throughout millennia. However, the types of control that arise from simple interaction of the human brain (or any other animal's) with the world, through its senses and limbs, are far from the topics that normally appear on the contemporary control theory, though the possibility of using this type of control is not precluded. For example, the recent emergence of robots made questions such as the optimality of teacup handling a control topic.

One of the most basic control entities, called a servomechanism, is an **automatic** device of the actuator family, that can maintain a certain input value (obviously, with sensing, *computing* and actuating capacity). Evidently, it is necessary to define the word **automatic**, but before this is done there are a number of other terms that need to be defined.

A transducer is defined as being any device that converts signals from one domain into another. Examples of such domains include the electrical, temperature, pressure, humidity, liquid levels, etc. A transducer that translates a physical quantity into a quantity that can be treated, either by humans or by machines, is called a sensor. Sensors can be further classified into indicators and registers. Indicators are sensors in which their readings are used (mostly in displays for human consumption, hence their name) and discarded, whereas registers are sensors in which readings are kept in some form. There is another group of transducers, called actuators, which are used to transduce quantities produced either by humans or by machines into the physical world. Due to the electronic nature of most machines, normally sensors transduce into the electrical domain whereas actuators transduce from the electrical domain.

A control system is said to be **manual** if there is a human being sensing the system, building a mental representation, using a number of rules to derive new control values, and then applying such values to the external world. A control system is said to be **automatic** if the control loop is closed without human intervention. Nevertheless, there are cases in between. For example a human typing sensed values (through vision) into a keyboard that is connected to a computer that computes the control values and produces the outputs. Another example is that of a computer system with cameras (vision) and controllers but with limited actuation capabilities (for example, the operation of some industry machinery). In this case, the algorithm is run by the computer but the actuation is done by human beings. Usually a system is called manual or automatic according to who/what is doing the control, regardless of who/what is doing the sensing and actuation.

The oldest automatic control processes on record are those of mechanical watches dating back for several centuries B.C. Other typical ancient processes that do not required human intervention were the *sacred* motions within temples in which apparent lack of a reason for the motion in question was upheld as a proof for the actions of a deity, for example, the doors of Heron. Other early examples of automatic control systems include windmills.

However, the first control system to be designed and later analysed in a manner that resembles modern forms of control was the Ball (or centrifugal) Governor invented by James Watt, and was fundamental in the development of the steam engine which fueled the industrial revolution. A governor is a device intended to keep a given speed constant. In a more modern vernacular it was a non-zero regulator or a speed servomechanism. Arguably the first publication regarding control systems theory was done by James Clerk Maxwell in 1868 entitled on "*On Governors*" [MAX68]. The publication in question introduced some fundamental concepts that are still in use in modern control.

However, Control Theory remained an almost dormant field until the 1920/1930 when the electronics and telecommunications revolution created both a need and means to realize the control systems. One of the most important revolutions of this time came from Oliver Heaviside, which used the Fourier transform:

$$F(j\omega) = \mathfrak{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\mathrm{d}t \tag{2.73a}$$

$$f(t) = \mathfrak{F}^{-1}\{F(j\omega)\} = \frac{1}{2\pi j}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}\mathrm{d}j\omega \tag{2.73b}$$

and the (bilinear) Laplace transform

$$F(s) = \mathfrak{L}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-st}\mathrm{d}t \tag{2.74a}$$

$$f(t) = \mathfrak{L}^{-1}\{F(s)\} = \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st}\mathrm{d}s \tag{2.74b}$$

in which the first equation of each group defines the transform and the second the inverse. The inverse Laplace transform must be taken for σ greater than the real part of all the singularities of $F(s)$.

It can be shown that the continuous-time linear rational systems, as defined in Subsection 2.1.1, can be transformed into a polynomial equation in the variable s. In fact, the Fourier/Laplace transform the p operator into $j\omega/s$ respectively. This framework for the analysis of Control Systems became known has the frequency domain approach.

In 1932 Harry Nyquist at Bell Labs introduced the first test of BIBO stability [Nyq32]. It was a graphical method that determined the stability of feedback Control Systems with its working principle enrooted in the theory of complex analysis. The next landmark in Control Theory occurred with the publication of Harold Hazen [Haz34] in 1934. Though the article did not introduce many new and important results, it introduced important concepts such as dumping factors and time constants which are extensively used nowadays.

Harold Black made an important contribution regarding systems with linear feedback [Bla34] that nowadays is commonly taught in introductory control classes. He proved that it is possible to exchange gain for bandwidth, i.e., linear systems have a constant gain-bandwidth product and the value of each variable is a function of the amount of feedback.

The following landmark contribution came from Hendrik Bode that realised that there was a relation between amplifier's frequency response gain and phase. This fact limited the range of possible realizable systems, facilitating their analysis. In fact, Bode perfected the Nyquist plots into their current form and introduced a new type of asymptotic plots that carry his name. Bode also introduced the concept of gain margin, i.e., the reciprocal of the amplifiers' gain at the (first) frequency that generate a phase of 180°, and phase margin, i.e., the 180° minus the phase at the (first) frequency in which the gain is unitary. Most of Bode's work, though extensive, were presented in one single article [Bod40].

In 1948 Walter R. Evans developed one of the most intuitive graphical methods named Root Locus [Eva50], in which the open loop singularities are used to derive the properties of the closed loop system. More concretely, it describes the path of the closed loops poles as a given parameter of the system is changed.

2.3.1 Modern Control

Ironically, Modern Control emerged out of concepts that were developed at the turn of the last century, with the Aleksandr Lyapunov and (Jules) Henri Poincaré works on stability of linear systems, that gave birth to the modern understanding of stability.

Lyapunov Stability and Equilibrium

Consider (non)linear systems given by the state dynamic $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t))$ and $x(0) = x_0$ (initial state). Consider also that there is an equilibrium point denoted x_e in which the state can be at rest ($f(x_e) = 0_{n\times 1}$).

The equilibrium is said to be Lyapunov stable if there is a region surrounding the equilibrium point such that whenever the state enters the region, it never leaves it. Such equilibrium is said to be asymptotically stable if once a state enters this region it converges asymptotically to the equilibrium point. And is said to be exponentially stable if the asymptotes of convergence are negative exponential functions.

An useful analogy for the Lyapunov stability is that $f(x)$ is landscape with peaks and valleys. The equilibrium points are the valleys (peaks) such that $f(x) = 0$, i.e. points that are locally horizontal. An equilibrium point is stable if it is surrounded by points that are above it, is *indifferent* (in a given direction) if the height of the landscape does not change if the state is moved in that direction, and an equilibrium point is unstable if it is surrounded by points that are below it. It is asymptotically stable if when *dropped* close enough to the equilibrium point, it will end up falling into the equilibrium and exponentially stable if the fall is exponential.

Lyapunov then established that a system is stable if there is a positive definite function (called Lyapunov function) which has a negative definite time derivative that satisfies certain conditions. In certain systems finding a candidate function with this property may be non-trivial. For physical systems, the energy of the system is always a Lyapunov function, and it is interpreted as: the system is stable (in a given region) if its energy is strictly monotonic decreasing (for non-zero value).

Remarks similar to the ones made above can be made regarding discrete-time system.

For the particular case of linear systems, the Lyapunov stability degenerates into the stability test regarding the eigenvalues (poles) of the system presented in Subsection 2.1.3. For more information into discrete-time stability criteria see [HRS07], as for the continuous-time case, consult [CF03].

Controllability and Observability

The Lyapunov analysis does not answer questions regarding the (possibility of) stabilization of unstable systems. It was limited to establishing whether a system is stable when unforced. The principles that will be presented shortly are attributed to Robert Kalman and address this problem for discrete-time linear function. A continuous-time version is analogous and will not be presented.

A system is said to be fully controllable if (after a given predefined amount of time) it can be taken from one state into any other and is called partially controllable if only a subset of its states are controllable. It can be proven that a system is fully controllable if the matrix

$$\mathcal{C} = \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \cdots & \mathbf{A}^{n-1}\mathbf{B}\end{bmatrix} \tag{2.75}$$

has rank equal to n, recall that n is the number of state variables. The rank of a matrix is defined as the number of linearly independent rows/columns.

Similarly, a system is said to be observable (discrete-time) if the matrix

$$\mathcal{O} = \left[\mathbf{C}\,\mathbf{CA}\,\mathbf{CA}^2 \cdots \mathbf{CA}^{n-1}\right]' \tag{2.76}$$

is full rank. By observable it is meant that it is possible to infer its current state by knowing a number of past outputs.

A system may not be fully controllable (observable). The states that contribute for the non-controllability (non-observability) are called non-controllable (non-observable) and the remaining are called controllable (observable). This defines four types of states: controllable observable, non-controllable observable, controllable non-observable and non-controllable non-observable. It is possible to prove that non-controllable observable states in one representation can be transformed, through a similarity transformation, into a controllable non-observable state in another representation. The study of non-controllable non-observable states may seem like a futile endeavor, however, such states of a system may become controllable observable if the system in question is connected to other systems. In fact, this was one of the main historical motivation to describe this types of states.

Regarding stability, a system is said to be stabilizable if all of its unstable states are controllable. Similarly, a system is said to be detectable if all of its unstable states are observable.

Besides these input to state and state to output relation, it is also defined the input to output controllability (full input observability is almost never possible). State controllability does not imply output controllability and output controllability does not imply state controllability. A system is output controllable if and only if the matrix

$$\mathcal{M} = \left[\mathbf{CB}\ \mathbf{CAB}\ \mathbf{CA^2B} \cdots \mathbf{CA}^{n-1}\mathbf{B}\right] \tag{2.77}$$

has rank n.

These concept are explored in more detail in [CF03] and [Lev96].

Pole placement

Pole placement is a control (observer) technique that uses a feedback matrix to position the closed loop poles at *any* location of the root locus. Though, this technique has a number of limitations, namely, 1) the number of poles does not change, 2) the energy necessary for position the poles at regions may be restrictive, and 3) the position of the zeros is unchanged.

In pole placement and in many of its successor techniques, the position of the controller and observer poles can be set independently. This allows to perform the two processes separately. The control (observer) equations are given by (respectively)

$$u(k) = -K_{ctrl}x(k) \tag{2.78a}$$

$$x(k+1) = Ax(k) + Bu(k) + K_{obs}\left(y - Cx(k)\right). \tag{2.78b}$$

A possible solution for SISO systems is called the Ackerman formula and is equal to

$$K_{ctrl} = \left[\,\mathbf{0}_{1\times(n-1)}1\,\right]\mathcal{C}^{-1}p(A) \tag{2.79a}$$

$$p(\lambda) = \prod_{i=0}^{n-1}\left(\lambda - \lambda_i\right), \tag{2.79b}$$

in which λ_i are the desired poles. A similar mechanism exists for the observer gain, which can be found by commuting $\mathbf{B} \leftrightarrow \mathbf{C}'$ and $\mathbf{A} \leftrightarrow \mathbf{A}'$. However, this method has two fundamental problems: 1) the Ackerman formula is numerical highly instable and 2) it is only valid for SISO systems. Other methods have been proposed that do not have these drawbacks. A good introduction for extensions of the pole placement method can be found in [Var00], in which a review of several such methods is made.

The observer is not equal to the Kalman filter, due to two reasons. The first reason is that the observer gain is set according to error filtration requirements which is an ideal framework when pole placement is employed, whereas in the Kalman filter the gain is set according to an optimality requirements computed from the covariance matrix of the two terms. However, it is possible to choose an optimal observer gain, which leads to the second difference, i.e., the structure of the error correction in which for the Kalman filter is done after the update whereas in the observer gain is done (before) simultaneously. A similar comparison can be made between pole placement control and the Linear Quadratic Regulator that will presented in the next subsection.

An in-depth presentation of the subject of pole placement is presented in [Lev96].

Linear Quadratic Regulators/Gaussian

Regulators are a subtype of controllers that are used to set the output variable at a constant value. The opposite type of controllers are called *tracking controllers* because they are optimized to track a physical or imaginary function.

Linear Quadratic Control (LQC) is used to denominate the control of linear systems with the goal of minimizing a quadratic (cost) function, and Linear Quadratic Gaussian (LQG) is the result of the combination of an LQC with a Kalman filter. For the continuous-time case the cost function is of the form (which, in some sense, is the simplest)

$$J(t_0, t_1) = \int_{t_0}^{t_1} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}' \begin{bmatrix} \check{\mathbf{Q}} & \check{\mathbf{N}} \\ \check{\mathbf{N}}' & \check{\mathbf{R}} \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \mathrm{d}t. \tag{2.80}$$

A number of properties must be verified by J in order to ensure that it is a cost function. The first one regards linearity, i.e., both J and αJ must be minimized by the same input, and it is easy to verify that the definition above has this property, i.e., α appears by multiplying the cost sub-matrices by the value in question, which obviously has no effects on the values of the input or state variables.

A second property of cost functions is their definiteness meaning that the second derivative of the cost function never changes its value, or its definiteness in case it is a matrix. The definiteness can be changed by multiplying it by -1, thus changing into a maximization (minimization) problem into a minimization (maximization). This is an important property because it prevents the existence of several inconsistent *control actions*, such as the possibility of reducing the (absolute value of) cost function by taking an action that has cost with a sign contrary of the current cost.

This itself implies that $\check{\mathbf{Q}}$ and $\check{\mathbf{R}}$ are themselves (semi-)definite (consider exciting the cost function with either $x(t) = x$ and $u(t) = \mathbf{0}_{1\times r}$ or $x(t) = \mathbf{0}_{1\times q}$ and $u(t) = u$, in a given instant). Furthermore, the matrix $\check{\mathbf{Q}} - \check{\mathbf{N}}\check{\mathbf{R}}^{-1}\check{\mathbf{N}}'$ is itself (semi-)definite. This notion is formalized in the Schur complement theorem [Zha05].

Throughout the rest of this Thesis, the following notation will be used $\lambda(\mathbf{M})$ denotes the set of all eigenvalues of matrix $\mathbf{M}$ and $\mathbf{M} > s$, $\mathbf{M} \geq s$, $\mathbf{M} < s$ and $\mathbf{M} \leq s$ denotes that $\max\{\lambda(\mathbf{M})\} > s$, $\max\{\lambda(\mathbf{M})\} \geq s$, $\max\{\lambda(\mathbf{M})\} < s$ and $\max\{\lambda(\mathbf{M})\} \leq s$ respectively.

The goal of the controller is to find a signal $u(t)$ that minimizes $J(t_0, t_1)$ subject to the dynamic equation of the system. Though it may be the case that the goal of the minimization is to find the path of smaller cost from the point $x(t_0)$ at $t = t_0$ to the point $x(t_1)$ at $t = t_1$. In these cases, an extra cost is added to the final state to *punish* final states that are not equal to the destiny, but in most cases it is possible to simply shift the state space in a way that puts the final state at the origin, thus reducing this problem into a simpler tracking control problem.

The minimization of the cost function itself is a problem of the calculus of variations but, on this Thesis, a simplified version of such topic will be treated, leaving out any subtopic that is outside its respective scope. The most appropriate framework to deal with this problem is the Lagrangian multipliers, due to the constraint (in fact, it is not a constraint since if it was not present it would be impossible to affect the state) imposed by the dynamic equations. Hence, after the introduction of the Lagrangian multiplier, Equation (2.80) becomes

$$J(t_0, t_1) = \int_{t_0}^{t_1} \left\{ \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}' \begin{bmatrix} \check{\mathbf{Q}} & \check{\mathbf{N}} \\ \check{\mathbf{N}}' & \check{\mathbf{R}} \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + \left(\check{\mathbf{A}} x(t) + \check{\mathbf{B}} u(t) - \frac{\mathrm{d}}{\mathrm{d}t} x(t) \right)' \lambda(t) \right\} \mathrm{d}t. \quad (2.81)$$

Equation (2.81) has the disadvantage of having a time derivative of $x(t)$ which has an implicity *correlation* with $x(t)$. To deal with this problem, Equation (2.81) is integrated by parts yielding

$$\begin{aligned} J(t_0, t_1) = &- x'(t_1)\lambda(t_1) + x'(t_0)\lambda(t_0) + \int_{t_0}^{t_1} \left\{ \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}' \begin{bmatrix} \check{\mathbf{Q}} & \check{\mathbf{N}} \\ \check{\mathbf{N}}' & \check{\mathbf{R}} \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \right. \\ &\left. + \left(\check{\mathbf{A}} x(t) + \check{\mathbf{B}} u(t) \right)' \lambda + x'(t) \frac{\mathrm{d}}{\mathrm{d}t} \lambda(t) \right\} \mathrm{d}t. \end{aligned} \quad (2.82)$$

Taking the derivative of these two equations (the $u(t)$ derivative can be taken from either equation, however, the $x(t)$ derivative is taken from Equation (2.82) and the $\lambda(t)$ is taken from (2.82)). Such derivatives yield three integrals that are equated to zero for every value of the upper limit of integration, thereby implying that the integrand themselves must be equal to zero, i.e.

$$\frac{\partial J}{\partial x'} = \check{\mathbf{Q}} x(t) + \check{\mathbf{A}}' \lambda(t) + \frac{d}{\mathrm{d}t} \lambda(t) = \mathbf{0}_{n \times 1} \quad (2.83a)$$

$$\frac{\partial J}{\partial u'} = \check{\mathbf{R}} u(t) + \check{\mathbf{B}}' \lambda(t) = \mathbf{0}_{q \times 1} \quad (2.83b)$$

$$\frac{\partial J}{\partial \lambda'} = \check{\mathbf{A}} x(t) + \check{\mathbf{B}} u(t) - \frac{d}{\mathrm{d}t} x(t) = \mathbf{0}_{n \times 1}. \quad (2.83c)$$

Equation (2.83b) when solved in order to $u(t)$ yields

$$u(t) = -\check{\mathbf{R}}^{-1} \lambda(t), \quad (2.84)$$

which when substituted into the other two sub-equations leads to

$$\begin{bmatrix} \dot{\lambda}(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} -\check{\mathbf{A}}' & -\check{\mathbf{Q}} \\ -\check{\mathbf{B}}\check{\mathbf{R}}^{-1}\check{\mathbf{B}}' & \check{\mathbf{A}} \end{bmatrix} \begin{bmatrix} \lambda(t) \\ x(t) \end{bmatrix}. \tag{2.85}$$

$\lambda(t)$ is called the co-state. Equation (2.85) has both the (feedback) state equation (at the bottom) and the co-state equation (at the top). Due to its form, Equation (2.85) is certainly not stable, but this is not an issue because the unstable part of the dynamic equation can be attributed to $\lambda(t)$ which goes backwards in time, thus being non-causal but being stable and the stable part can be attributed to $x(t)$ that is both causal and stable.

Equation (2.85) is a linear differential equation with a known solution, namely

$$\begin{bmatrix} \lambda(t) \\ x(t) \end{bmatrix} = e^{\tilde{\mathbf{A}}(t-\tau)} \begin{bmatrix} \lambda(\tau) \\ x(\tau) \end{bmatrix}, \tag{2.86}$$

where $\tilde{\mathbf{A}}$ is the transition matrix of Equation (2.85). If τ is chosen in such that $\lambda(\tau) = \mathbf{0}_{n\times 1}$, then it is possible to write $\lambda(t) = \tilde{\mathbf{P}}(t)x(\tau)$ and $x(t) = \hat{\mathbf{P}}(t)x(\tau)$, for some $\tilde{\mathbf{P}}(t)$ and $\hat{\mathbf{P}}(t)$ that come from the partitioning of matrix $e^{\tilde{\mathbf{A}}(t-\tau)}$, which implies that

$$\lambda(t) = \check{\mathbf{P}}(t)x(t) \tag{2.87}$$

with $\check{\mathbf{P}} = \tilde{\mathbf{P}}\hat{\mathbf{P}}^{-1}$ and $det(\hat{\mathbf{P}}) = 0$. The last equation introduces a relationship between $\lambda(t)$ and $x(t)$ that allows to write optimal control equations without the co-state, although a new matrix $\check{\mathbf{P}}$ is introduced. When this relation is introduced into Equation (2.85) leads to

$$\begin{bmatrix} \dot{\check{\mathbf{P}}}(t)x(t) + \check{\mathbf{P}}(t)\dot{x}(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} -\check{\mathbf{A}}' & -\check{\mathbf{Q}} \\ -\check{\mathbf{B}}\check{\mathbf{R}}^{-1}\check{\mathbf{B}}' & \check{\mathbf{A}} \end{bmatrix} \begin{bmatrix} \check{\mathbf{P}}(t)x(t) \\ x(t) \end{bmatrix}. \tag{2.88}$$

which when left multiplied by $\left[I_n, -\check{\mathbf{P}}\right]$ yields

$$\dot{\check{\mathbf{P}}}(t)x(t) = \left(-\check{\mathbf{A}}'\check{\mathbf{P}}(t) - \check{\mathbf{P}}(t)\check{\mathbf{A}} + \check{\mathbf{P}}(t)\check{\mathbf{B}}\check{\mathbf{R}}^{-1}\check{\mathbf{B}}'\check{\mathbf{P}} - \check{\mathbf{Q}}\right)x(t). \tag{2.89}$$

Since Equation (2.89) is valid for any value of $x(t_0)$ (hence for any value of $x(t)$), then

$$-\dot{\check{\mathbf{P}}}(t) = \check{\mathbf{A}}'\check{\mathbf{P}}(t) + \check{\mathbf{P}}(t)\check{\mathbf{A}} - \check{\mathbf{P}}(t)\check{\mathbf{B}}\check{\mathbf{R}}^{-1}\check{\mathbf{B}}'\check{\mathbf{P}} + \check{\mathbf{Q}}. \tag{2.90}$$

The last equation determines the dynamics of $\check{\mathbf{P}}$. Nevertheless, the control value is still undefined. This can be solved by combining Equation (2.83b) which defined $u(t)$ in terms of $\lambda(t)$ and Equation (2.87) which defined $\lambda(t)$ in terms of $x(t)$, then

$$u(t) = -\check{\mathbf{R}}^{-1}\check{\mathbf{B}}'\check{\mathbf{P}}(t)x(t). \tag{2.91}$$

Equation (2.90) is an important part of continuous-time control. In fact, it is the control law itself. The control law is the function that defines the (optimal) control values given the (expected) system state. Equation (2.91) implies that the control law for continuous-time linear quadratic regulators is also a linear function, i.e., $u(t) = -\check{\mathbf{L}}(t)x(t)$ with $\check{\mathbf{L}}(t)$ implicitly defined.

Nevertheless, Equation (2.90) which is used in Equation (2.91) to define the control law is a differential equation which defines a family of possible solutions. Hence, there is a need for finding the particular function within this family that actually minimizes the cost function. One possibility (the most used one) is to define a boundary condition. Such condition can be imposed by equating Equation (2.83a) (i.e., the derivative of the cost function minus the lagrangian term) computed at $t = t_1$, which leads to a condition in λ hence of $\check{\mathbf{P}}$.

A substitution of the optimal values, i.e., Equation (2.91) for the optimal control value and Equation (2.87) for the optimal co-state into Equation (2.82) leads into (by subtracting a sum of the sub-equations (2.83) multiplied by the respective variable taken the derivative of and taking into consideration the boundary condition of λ)

$$J^*(t_0, t_1) = x'(t_0)\check{\mathbf{P}}(t_0)x(t_0) \tag{2.92}$$

in which the J^* is used to denote the optimal value of the variable in question.

Readers interested in more information on regulators can consult, for example, either [Lev96] or [CF03].

Pontryagin Minimum Principle and Bellman-Jacobi-Hamilton Theorem

The solution presented above, though limited to linear systems, has a number of canonical properties that appear in all optimal control problems, hence, they will also appear in the control problems that are the object of study of this Thesis. Therefore, explicitly stating these properties of the solutions of optimal control reduces the number of steps for solving future occurrences of similar problems.

The first such property is called Pontryagin minimum principle, first formulated by Lev Pontryagin in 1956, states that 1)J^* is an extremal of J and 2) it is always possible to include a new variable $J + \lambda_0$ (with λ_0 constant) such that J^* is zero at the final time and $\lambda_0 < 0$. This was done implicitly above, when the initial (final) conditions were set. In fact, the reason this principle seems so obvious is that it was first formulated for systems with constraints and on such more complex setting the principle is not so self-evident. This principle is a necessary condition but not a sufficient.

The second general property is a necessary and sufficient condition called Bellman-Jacobi-Hamilton theorem. The theorem states that the minimum cost attainable from the optimal control function at a given instant is equal to the optimal control value from the instant in question until some point in the future plus the optimal cost from the future point in question until the end, or for the linear quadratic case

$$J^*(t_0, t_1) = \overbrace{\min_{u(t)} \int_{t_0}^{t'} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}' \begin{bmatrix} \check{\mathbf{Q}} & \check{\mathbf{N}} \\ \check{\mathbf{N}}' & \check{\mathbf{R}} \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \mathrm{d}t}^{J^*(t_0, t')} + J^*(t', t_1) \tag{2.93}$$

A commonly used notation, that is justified by the Bellman-Jacobi-Hamilton theorem, also adopted in this Thesis is

$$V(x(t), t) \stackrel{def}{=} \min_{u(\tau)} J(t, t_1) \tag{2.94}$$

whereas the $u(\tau)$ denotes all the control values from t to t_1. More information on the discrete-time formulation of these principles can be found, for example, at [HRS07].

Discrete-Time Linear Quadratic Regulators

The goal of this subsection is to find the control sequence that minimizes a discrete-time function, which is equivalent to the cost function used for the continuous-time case. There is the possibility of using Lagrange multipliers as it was done for the continuous-time case. However, that approach was used in the continuous-time case for pedagogical and presentational reasons. For the discrete-time it will be proven using the Bellman-Jacobi-Hamilton theorem (which implicitly use Lagrange multipliers) and input noise ($w(k)$) will also be consider i.e.

$$V(x(k),k) = x'(k)\mathbf{P}x(k) + c(k) \tag{2.95}$$

which implies that (by applying the Bellman-Jacobi-Hamilton theorem to the $(k+1)^{th}$ step)

$$V(x(k),k) = \min_{u(k)} \left[\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}' \begin{bmatrix} \mathbf{Q} & \mathbf{N} \\ \mathbf{N}' & \mathbf{R} \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x'(k+1)\mathbf{P}(k+1)x(k+1) + c(k+1) \right] \tag{2.96}$$

taking the derivative of $J(k)$ in order to $u(k)$ and equating it to zero

$$\frac{\partial J(k)}{2\partial u(k)} = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}' \begin{bmatrix} \mathbf{Q} & \mathbf{N} \\ \mathbf{N}' & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n\times q} \\ I_q \end{bmatrix} + x'(k+1)\mathbf{P}(k+1)\frac{\partial x(k+1)}{\partial u(k)} = \mathbf{0}_{1\times q}. \tag{2.97}$$

The derivative is taken in order to $J(k)$ because $V(x(k),k)$ is not a function of $u(k)$. More concretely, $J(k)$ is a function of $u(k)$ in the sense that different values of $u(k)$ can be chosen which in turn will lead to different values of $J(k)$. However, $V(x(k),k)$ is not a function of $u(k)$, because it is equal to $J(k)$ when a particular optimal value of $u(k)$ (designated $u^*(k)$) is used.

The dynamic equation(s) implies that $\frac{\partial x(k+1)}{\partial u(k)} = \mathbf{B}$. Substituting the dynamics of the system into $x(k+1)$, substituting the partial derivative and performing the matrix multiplications, yields

$$x'(k)\mathbf{N} + u^{*\prime}(k)\mathbf{R} + (\mathbf{A}x(k) + \mathbf{B}u^*(k) + w(k))'\mathbf{P}(k+1)\mathbf{B} = \mathbf{0}_{1\times q}, \tag{2.98}$$

which by transposing and putting all terms in $u^*(k)$ in the left side leads to

$$\left(\mathbf{B}'\mathbf{P}(k+1)\mathbf{B} + \mathbf{R}\right)u^*(k) = -\mathbf{B}'\mathbf{P}\left(\mathbf{A}x(k) + w(k)\right) - \mathbf{N}'x(k). \tag{2.99}$$

The optimal control value stems from the left multiplication of the appropriate matrix and taking expectations over $w(k)$, i.e.

$$u^*(k) = -\left(\mathbf{B}'\mathbf{P}(k+1)\mathbf{B} + \mathbf{R}\right)^{-1}\left(\mathbf{B}'\mathbf{P}\mathbf{A} + \mathbf{N}'\right)x(k). \tag{2.100}$$

The last equation implicitly defines an optimal control gain as in $u^*(k) = -\mathbf{L}^*(k)x(k)$.

As for the values of $\mathbf{P}(k)$ and $c(k)$, a subtraction of an appropriate form of Equation (2.97)

from Equation (2.96) yields,

$$J(k) - \frac{\partial J(k)}{2\partial u(k)} u^*(k) = \left(\mathbf{Q}x(k) + \mathbf{N}u^*(k)\right)' x(k) + x(k+1)'P(k+1)\left(x(k+1) - \frac{\partial x(k+1)}{\partial u(k)} u^*(k)\right) + c(k) \tag{2.101a}$$

$$= x'(k)\left(\mathbf{Q} - \mathbf{NL}(k)\right) x(k) + \left(\left(\mathbf{A} - \mathbf{BL}(k)\right) x(k) + w(k)\right)' \mathbf{P}(k+1)\left(\mathbf{A}x(k) + w(k)\right) + c(k+1) \tag{2.101b}$$

$$= x'(k)\left(\mathbf{Q} - \mathbf{NL}(k)\right) x(k) + \left(\mathbf{A} - \mathbf{BL}(k)\right)' \mathbf{P}(k+1)\mathbf{A}x(k) + w'(k)\mathbf{P}(k+1)w(k) + c(k+1) \tag{2.101c}$$

$$= x'(k)\mathbf{P}(k)x(k) + c(k) \tag{2.101d}$$

with

$$\mathbf{P}(k) = \mathbf{Q} - \mathbf{NL}(k) + \left(\mathbf{A} - \mathbf{BL}(k)\right)' \mathbf{P}(k+1)\mathbf{A} \tag{2.102a}$$

$$c(k) = c(k+1) + \text{Trace}\left(\mathbf{Q}_w \mathbf{P}(k+1)\right). \tag{2.102b}$$

Equation (2.101a) follows by simple subtraction. Equation (2.101b) follows by substituting $u^*(k)$ and $x(k+1)$. Equation (2.101c) follows by the grouping of terms with $x(k)$ and $w(k)$ and finally Equation (2.101d) follows from the definition of the respective terms and the format of (2.101c). The format in question implies Equation (2.102) in particular Equation (2.102a) follows directly. However, this recursive equation does not define the value of $\mathbf{P}(N)$, i.e., it last value. Nevertheless, by analyzing the initial cost function at the last step it follows that $\mathbf{P}(N) = \mathbf{Q}(N)$ (in fact the cost function has terms in $u(N)$, nonetheless, $u(N) = \mathbf{0}_{q\times 1}$ because $u(N)$ can no longer influence the state variable, hence if it were not zero it would incur a cost but would not imply a reduction of the cost through $x(N+1)$).

Regarding Equation (2.102b), it is derived below, see Equation (2.103). In fact, this type of equation occurs repetitively throughout the Thesis, hence it is of uttermost importance to establish it:

$$E\left[w'(k)\mathbf{P}(k+1)w(k)\right] = E\left[\text{Trace}\left(w'(k)\mathbf{P}(k+1)w(k)\right)\right] \tag{2.103a}$$

$$= E\left[\text{Trace}\left(w(k)w'(k)\mathbf{P}(k+1)\right)\right] \tag{2.103b}$$

$$= \text{Trace}\left(E\left[w(k)w'(k)\right]\mathbf{P}(k+1)\right) \tag{2.103c}$$

$$= \text{Trace}\left(\mathbf{Q}_w \mathbf{P}(k+1)\right). \tag{2.103d}$$

In which Equation (2.103a) follows because a 1×1 matrix (a number) is equal to its own trace. Equation (2.103b) follows because the argument of the trace function does not vary upon cyclical permutation. Equation (2.103c) follows because the trace function is linear and the expectation operator commutes with linear operators and finally, Equation (2.103d) follows from the definition of $\mathbf{Q}_w$.

A few notes are in order: 1) as in the continuous-time case, $\mathbf{P}$ and $\mathbf{L}$ do not depend on the value of x, thus vastly simplifying their computation, 2) these matrix converge after a few iteration. The number of such iterations (k_{ss}, with ss for steady state) is a function of the dynamics of the system and as $T_s \to 0$, $k_{ss}T_s = t_{ss}$ which is itself the amount of time

necessary for the continuous-time equation to converge. Upon convergence the matrices in question become $\mathbf{P}_{ss}$ and $\mathbf{L}_{ss}$ and 3) these equations are solved backwards in time, a fact that can become an inconvenient as it will be seen in the main body of the Thesis.

An in-depth presentation of this topic is available at [Lev96] and at [HRS07].

2.3.2 Receding Horizon Control

Optimal control, or at least, optimal linear control with quadratic cost or LQG, depends in future cost matrices that are normally not known at the time that the control gain matrices ($\mathbf{L}(k)$) are computed.

Nevertheless, as pointed out above, both $\mathbf{L}(k)$ and $\mathbf{P}(k+1)$ converge relatively fast, which implies that to compute $\mathbf{L}(k)$ and $\mathbf{P}(k+1)$ it is necessary to know the cost matrices from k up to $k+k_{ss}$. This fact helps to alleviate a shortcoming of LQG and it is at the heart of Model Predictive Control (MPC), in which the general idea is to compute the optimal control values up to a given moving horizon that guarantees that the present control value is equal to the optimal LQG control.

In effect, this creates a cycle in which at each time instant a single future cost matrix, above the current horizon, is acquired. Then the optimal control values up to the horizon are computed. The current control value is applied and the remaining are discarded. This cycle is the reason why these techniques are called *receding*.

At least this was the initial motivation. More recent applications of receding horizon control 1) use an horizon shorter than the optimal horizon, i.e., shorter than the horizon related to the convergence time and 2) no longer use the same horizon for the state and control value costs. The first difference is due to the fact that unlike in LQG, in MPC these computations are done online which makes the online receding horizon control more computationally demanding. The second aspect emerged empirically by studying the behaviour of such systems in real settings in which it was noted that (see references below) it was possible to remove certain groups of cost without a severe degradation of the quality of control.

Generalized Predictive Control

In the past years, predictive control has been one of the most researched areas of control, consequently, it is also one of the areas that advanced the most. Predictive control broke a long standing tradition in which theoretical control had almost not influenced practical applications of control.

A control technique is said to be a Generalized Predictive Control (GPC) if for computing the control actions (inputs) the technique uses predictions of future outputs. In GPC control actions are computed by minimizing a cost function[JW08]:

$$J(N_1, N_2, N_u) = \sum_{i=N_1}^{N_2} [y(k+i) - r(k+i)]^2 + \sum_{i=1}^{N_u} \lambda(i)[\Delta u(t+i-1)]^2; \tag{2.104}$$

where $y(k)$, $r(k)$ and $u(k)$ denote the output, reference and control signals respectively. N_1 is the minimum costing horizon, N_2 is the maximum costing horizon, N_u is the control horizon and $\lambda(i)$ is a control weighting sequence.

N_1 is chosen as the first output value that can be controlled from the present. For example, if a system has a dead-time of $1.2T_s$ it would be superfluous to try to minimize the difference

$y(k+1) - r(k+1)$ since this quantity although in the future, cannot be influenced by present or future control actions. However, when there is no *a priori* knowledge of the dead-time, it is usual to set N_1 with the value of 1. Similarly, it is pointless to attribute to N_u a value higher than $N_2 - N_1$, since only control actions up to this point would have an effect in the output within the horizon. Nonetheless, in recent contributions, such as [JW08], the value of N_u has been considerably lower (typically 1 or 2), due to the fact that both practical realization and simulation studies have shown little or no improvement when larger values of N_u were used.

From a theoretical standpoint, the value of N_2 ought to be set as equal to the number of unstable or under-dumped poles, though sometimes the number of maximum phase zeros, i.e., zeros with positive real part, are also considered. Since when the number is lower than that, the controller cannot foresee a situation in which it will minimize the current cost, but drive the system state to a point that substantially increase the future values of the cost function. Nevertheless, experimental realization have shown that the system's rise time is a good value to N_2, and that larger values do not produce significant improvements in performance.

After computing the control sequence that minimizes the cost function, only the first input corresponding to $u(t) = u(t-1) + \Delta u(t)$ is applied. The remaining values of Δu are discarded. In the next sampling instant the process is repeated, but now all sequence are shifted one sample to the future. So both control horizon and output horizon are receding, thus GPC is a form of receding control. When both N_2 and N_u approach infinity the GPC become an optimal control strategy and in a sense it matches the definition of LQG.

The cost function can be represented in matrix form, yielding

$$J = (\tilde{y} - \tilde{r})'\mathbf{P}(\tilde{y} - \tilde{r}) + \tilde{u}'\mathbf{Q}\tilde{u} = \mathbf{G}\tilde{u} + \tilde{f} + e \tag{2.105a}$$

in which $\tilde{\mathbf{y}}$ is a vector of future outputs taken from N_1 to N_2, $\tilde{\mathbf{r}}$ is a vector of desired future outputs in the same range, $\mathbf{f}$ is a vector whose entries are the influences of current states in future output values and $\mathbf{e}$ is a vector of perturbation that will affect future outputs (which is not known when $\tilde{\mathbf{u}}$ is computed) and $\tilde{\mathbf{u}}$ is the set of future inputs. G is a $N_2 - N_1$ by N_u matrix of impulse response of the transfer function from $\tilde{\mathbf{u}}$ to $\tilde{\mathbf{y}}$, that is:

$$\mathbf{G} = \begin{bmatrix} g_{N_1} & 0 & \dots & 0 \\ g_{N_1+1} & g_{N_1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{NU} & g_{NU-1} & \vdots & g_{N_1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{N_2-1} & g_{N_2-2} & \vdots & g_{N_2-N_u} \end{bmatrix} \tag{2.106}$$

where g_i is the i^{th} value of impulse response from $\tilde{\mathbf{u}}$ to $\tilde{\mathbf{y}}$. And $\tilde{\mathbf{u}}$ is a vector of future control sequences Δu as defined in the original cost function. Note that in SISO systems g_i is a number, but in general g_i is a matrix and $\mathbf{G}$ is a block matrix.

Minimization of the cost function yields the control law

$$\tilde{\mathbf{u}} = \left(\mathbf{G}^\mathbf{T}\mathbf{P}\mathbf{G} + \mathbf{Q}\right)^{-1}\mathbf{G}^\mathbf{T}\mathbf{P}(\tilde{\mathbf{r}} - \tilde{\mathbf{f}}) \tag{2.107}$$

Up to this point we have presented GPC as a cost function minimization problem, though GPC is normally presented as a cost function that must be minimized under certain constrains.

Typical constrains are: the amplitude of the input (control) values, the rate of change of the control values (amplitude of Δu), amplitude of the output. With constrains in effect the cost function minimization is no longer a simple linear algebra problem. It becomes either a linear programming (LP), quadratic programming (QP) or non-linear programming (NLP) problem; [Kr99] presents the various approaches developed hitherto to cope with such constrains.

Model Predictive Control

MPC is a specific form of GPC in which future output (or states) are computed using an explicit model of the system, hence its name. Much of what was said about GPC can be said about MPC. Linear MPC is by far the most used type of MPC. In fact, the equation presented in the last section implicitly assumed that there is a linear relation between the input and the output. This reliance in linear models are in part due to the fact that good non-linear models are still hard to obtain, hence, state space linear model are the dominant form of representation. The other motive for the reliance in linear models is that there are optimal solutions for linear problems whereas there is no general optimal solution for non-linear problem and under certain circumstances it is better to have a suboptimal representation, i.e., linear, and an optimal controller for the representation in question than to have an optimal representation and a suboptimal controller.

The feasibility of MPC under constrains was a thriving research topic in the decade of 1990s. The research into input constrains related to saturation, was the main drive of this field. Despite the accomplished advances, when the optimal input variable to the MPC is outside the allowed values of the input variable or such optimal input drives the output to outside of the allowed output values, then, nothing can be done but to let the system have a suboptimal response. However, an approach to the problem has been presented [ZM95] in which slack variables, borrowed from the field of linear programming, were introduced to force the state of the system to be in a region that can reach any other state while still meeting the input constrains. This makes the system have a behavior that is always suboptimal, however, it frees the system from reaching a state in which there is no possible input (satisfying all constrains) that can drive the system to the next desired state. This approach is widely used in industry, as can be seen in the survey [QB96].

In case the output constrains are impossible to fulfill, i.e., the goals are not consistent, it is treated as a design issue, as opposed to physically impossible. That is, if the description of the required output is infeasible, then, it is better to find a feasible desired output and preferably to do so in offline.

The stability of MPC (and GPC) is still an open topic of research. In general, when a system is unstable it cannot (in a guaranteed manner) be stabilized by an MPC with input constrains. Nonetheless some classes of MPC's were proposed in which stability is guaranteed. One such MPC introduced in [PY93] adds a new constrain to the MPC in order to guarantee that the norm of the state will contract and according to the Lyapunov theory, stability follows easily. Approaches that do not introduce other constrains to MPC are rather more difficult to analyze.

Since MPC and GPC are receding horizon control techniques, when a control sequence is computed only the first samples of the control signal is used. This leads to differences between the control sequence that is computed in a given moment and the control sequence that actually gets implemented. The only cases in which they do not differ is when the horizons are infinite and/or the system is noiseless. Considering that exists a control sequence that

leads to a stable system, then it follows that MPC would be stable if it was given a long enough horizon. In fact, [PN97] proposed a metric for the difference between the cost of future control actions and outputs that are computed at a given moment, and the cost that they actually have (i.e., based in the values that they actually get in the future). They have shown that this difference decreases with the size of the horizons and they proposed that this metric was used for choosing the horizon.

Some extension for MPCs were recently proposed, such as, [SJLM11] in which MPCs are solved using wavelets transforms at varying rates, [SJC+12] that approaches the problem of iterative learning control by stating it as a MPC with a cost function similar to a Proportional-Integral-Derivative (PID) cost function.

In [HLC12] it was studied the behavior of distributed systems in which 1) various control elements were physically separated from each other and 2) the system to be controlled was non-linear. This raised some issues due to the lack of information regarding the control value that the other controllers were producing and the non separability of the various control actions that stem from the non-linearity. A similar extension is explored in [ZLQ13].

In [CF13] the authors used an MPC that searched through various value of the systems parameters. The authors called each combination of the parameters a scenario. The proposed controller choose a control value based in a weighting of the control value given a scenario and the probability of the scenario in question to be realized.

Other instances of MPC that can be found in the literature are concerned with its respective implementation. For example, [KM11], in which MPC is used to control the flow of a fluid in a tank, [NTZ+10] in which the current that goes through an electromagnet in order to control its respective axial flux, [ZLQ13] that report on the application of MPC in a distributed network, though no results on optimality (in the context of MPC as opposed to the general sense) were discussed. In [LL12] an MPC was employed in order to ensure that a mechatronic servo tracked a given signal using a servo that presented some input constraints.

2.3.3 Send-on-Delta, Adaptive Sampling and Event-triggered Control

Most control systems do not start in an equilibrium point, in the Lyapunov sense. In those non-equilibrium points it is necessary to generate the respective control values that ensure that the system has the required behaviour. However, under certain circumstances, the system may be in an equilibrium point and require a slowly varying control value.

In fact, the scenario presented in the previous paragraph is an extreme. In general, it is possible to set the control rate as a function of a distance of the system state to a desired equilibrium point. Under this circumstance, a new measure and/or control value is triggered whenever $\gamma_{\text{last}} - \gamma_{\text{actual}} > \delta$, for a given variable of interest γ (usually the output of the system). δ is a design parameter that defines the allowed error in the variable in question, γ_{last} is the value that the variable in question had at the last exchange instant and γ_{actual} is the current value of such variable. The described schema is called Send-on-Delta (SOD).

The image of SOD presented in the last paragraph is incomplete, because if the parameter δ is too small, the amount of time between to sampling/control instants may become smaller than the smaller amount of time that the control system can manage. This problem is solved by stating that a new value is produced only if a certain predefined amount of time, T_{min} has elapsed, regardless of whether the trigger condition was met.

It is also defined a T_{max} which is the maximum amount of time that the various control elements can go without communicating. This variable is defined in order to distinguish situ-

ations in which there was no communications because the trigger condition was not met, from situations in which there was no communications because a control element is not functioning properly. For this reason, the messages sent when this condition is met are called *I am alive* (or *heart beat*) messages.

To the best of our knowledge the first SOD was proposed in [TB66]; the 60's and 70's saw many advances in adaptive sampling. [VV06] extended this work by proposing a Proportional Integral Derivative (PID) controller that can also cope with some static problems, for example, *sticking* in which the output value has a slow derivative, hence low updating rate, because , for instance, it is near an overshoot point and not necessarily due to reaching a static point. More generally, even observable systems may be temporarily in a set of states that differ only by a state component that is not mensurable, i.e., $\mathbf{C}\left(\mathrm{x}(\mathrm{k}+1)-\mathrm{x}(\mathrm{k})\right)=\mathbf{0}_{\mathrm{q}\times 1}$.

In [PVK10] it is presented the results of a comparison of several PID controllers with SOD sampling. The comparison involves a periodically sampled PID, a PID with SOD, a PID with SOD applied to the integral of the absolute value of the error, a PID with SOD applied to a linear prediction of the output, such as the approaches presented in [Suh07] and [SKM11]. The authors debate the most known problems related to naïve PID implementations, such as the *sticking* problem. The *sticking* causes the sampler to not send any message because the output is not changing and *sticking* causes the controller to not change its value because no new sensor message was received. The other problem that is investigated is the *limit cycles* problem in which due to the SOD condition, the control is awaken in a point lower than the reference point, produces a control value that sends the output *up*, and then is awaken again when the output is above the reference point, so on and so forth, causing an oscillation. At the end of the paper, the authors built a table with strong and weak points of each of the approaches. Nevertheless, the authors of [PVK10] only consider some rather simple SOD PID controller and only one system, of which the dynamic equations were not specified. Moreover, it is not clear that the advantages that each of the SOD types displayed for this system will also be present in other systems.

As for the PID problems that stem from the non-periodic nature of SOD sampling, [WÅÅ02] presents a PID definition in which the parameters are explicitly written in order to the period. Then, [VK07] used this definition and applied to the SOD case in which at each control execution the period was set as the duration since the last execution. This ameliorates the problem but does not completely solves it, since the integral and differential approximations rely, at a fundamental level, on the assumption that the difference between any two samples will be small. When this assumption is not verified a phenomenon called derivative/Integral *wind up* takes place, in which the respective estimates drift away from the respective estimated variable. Under these circumstances, there are no set of PID parameters that solves the problem. Another, not mutually exclusive, possible solution is to reduce the maximum inter-messages time.

Still in the PID family, [BVDS11] presented a study of Proportional Integrator (PI) controllers with SOD. On their particular case, the SOD behavior was implemented at the level of the process error and its respective integral. There are actually used two different parameters and two different messages for each of them. A more in-depth study of the two parameter case is studied in [BDSV12]. The paper studies the properties of equilibrium points, i.e., the input-output error level that are eigenstates of the system, or simply, a system is at equilibrium if at a sampling instant k it has a given input and a given error level, then at sampling instant $\mathrm{k}+1$ it will have the same input and will not trigger the SOD condition. Basically, the paper aims at minimizing the effects of limit cycles. Nevertheless, a system with zero

disturbances was considered — too simplistic, it was not discussed the condition for which the equilibrium point would be around no error, the only requirement was that it did not leave a certain error band and the authors did not discuss the effects of the *I am alive messages* in the notion of equilibrium points, since one of the drives to search for this point, hence the main contribution of the paper is rather trivial, i.e., once at equilibrium there would be no need for sending more data.

In [PDRPS12] it is discussed the use of SOD with an estimation control co-design. First a Kalman-like filter is proposed, but the authors note that it has a forbidding complexity. Then an observer gain matrix scheduler is proposed, i.e two gain matrix, one for the case in which the condition is met, hence there is new data and another for the case in which the condition is not met. A suboptimal algorithm to design the scheduled gains was also presented. Then another suboptimal algorithm for choosing δ was put forward.

A first study of the possibility of using SOD controllers to reduce network loads was presented in [OMT02]. More recently it has been proposed that SOD controllers may be used to save energy in network elements. Other application of SOD controllers include the dynamical adaptation of (pseudo-)uniform sampling to the network load, by for example increasing the value of δ when the network is overloaded. By doing so, it is expected that the control deteriorates less than it would have if the messages where completely lost, such an approach was first proposed in [APM05].

Another relatively highly deployed approach, that is used to cope with missing data, is Dead-Reckoning and it consists on performing approximations of the current state variable based on the previous previous state variable. Actually, this is not how it is usually framed and it is (was before the emergence of the Global Positioning System, which provides a more accurate odometry mechanism) highly deployed in the aviation industry, where the position of an aircraft was estimated from its previous position, estimated velocity and acceleration, whenever a more accurate reading from a ground station was not available. This had some problems, since it does not take in account the air drag which is a form of friction, it assumed that the system was of order 2, assumption that differs from the common assumptions of the field and that were left unjustified. In modern control vernacular: the authors performed a *"state estimation without any knowledge of the system dynamics"*. [YPZ$^+$10] presents a stereotypical approach to dead reckoning. Other examples of dead reckoning include [GRSK11] in which the dead reckoning is used to determine the position of a pedestrian that wears a strapped down integrated circuity, [WBK11] reports on the application of global positioning system readings plus dead reckoning for a maritime exploration robot, [EAFR12] presented a case in which a version of dead reckoning was used to correct the bias introduced by a gyroscope and [CCC$^+$10] achieved a similar goals using a system with more input methods, i.e., a gyroscope, an accelerometer and a global positioning system, and also reports smaller error.

Note that in most such reports the term dead reckoning is used with an abuse of language, since there was a sample available at all sampling instants. However, the systems used the *dead reckoning* either to determine the value of the state variable, usually position, in the time between two receptions or to join measurements from different sources. The former case it is normally done by a simple interpolation, whereas the later is called a fusion. Note also that both of these steps are done optimally in the Kalman filter, though if the data is used more frequently then the sampling rate, then there will instants that the update part of the Kalman filter will be executed but the correction step will not.

Networked Gaming is another field in which dead-reckoning is commonly used. Its neces-

sity stems from the fact that the communication latencies between each gaming system and the server is considerably higher than the human reaction time. Hence, the end-user gaming system provides a dead-reckoned scenario to the user in question until it receives data from the server detailing the action of the other users. From a system stand-point, this problem is more severe than the previous one, since there is not even an approximation of the system dynamics. This scenario gives rise to other problems regarding the fusion of local and servers data. An example of such approach is presented in [HBG10] which proposes that dead reckoning in simulation environments can be improved if neural networks are used to produce the state variables in the dead periods. In [HHLH10] it is proposed a method to deal with dead reckoning in wireless gaming settings and the respective extra challenge that emerges, i.e., low energy. In [MWW12] it is presented a stereotypical multiplayer dead reckoning system with an SOD component, i.e., the message exchange at the server is triggered by the crossing of a server's internal error measure.

An approach similar to dead reckoning is presented in [YTS02] in which there is a distributed control system consisting of several controllers, each one with its respective sensor and actuator and each controlling a subpart of a larger system, hence there is a need for the controllers to coordinate. On this setting, it was proposed in [YTS02] that each controller simulated the execution of the remaining controllers whenever new data from such controllers was not available. It was noticed that the noise term that is the subject of the Kalman corrections are only available to its respective local controller, hence, cannot be simulated remotely without the reception of a new message. Though the general idea is relevant, its concretization was very poor, being riddled with basic mathematical flaws.

In [XLL12] and in references therein, it is discussed the use of moving horizons in estimation, in which the current best estimate is the value that minimizes a quadratic cost function of a fixed number, or window, of past sensor readings. Even though this is a suboptimal approach by design, it presents a rather good behavior under situations in which the optimal estimator is not known, i.e., it is a robust approach. Example of such situations include non-gaussian perturbation terms, state variable constraints, etc. In [XLL12] it is explored the possibility of using this framework to reduce the effects of packet losses, in a rather straightforward way. Nevertheless, the authors noted that the usual definition of moving horizon does not penalize the missing samples. A heuristic was presented to counter this fact.

The SOD approach, in its essence, reduces the number of transmissions by using the so called *event triggered* paradigm in which sampling and/or messaging is triggered by the occurrence of a given event, as opposed to the *time triggered* paradigm in which sampling and/or messaging is triggered by the arrival of certain instants.

After the proposal of the SOD framework, some more complex triggering rules were proposed. More recent examples include [SKM11] which proposes that the receiver should have future predictions based on a Taylor expansion based extrapolation. The authors explore this concept for first and second orders, i.e., linear and quadratic approximations. The sampler executes a similar predictor and the triggering condition takes into account the error between the predicted variable and its readings. This approach had previously been presented in [Suh07].

In [XXY+10] the authors propose the use of a predictor for reducing the amount of energy spent by wireless body sensor networks without providing any details as for the type of the predictors.

In [NS08] it is proposed a new triggering rule, i.e., $\int_{t_{last}}^{t} (y(t) - y(t_{last}))\, dt < \delta$. This approach is dubbed Send-on-Area. The authors show a simulation in which for certain values

of the parameter δ their approach outperforms the standard SOD approach. However, the paper has some unconventional and unjustified assumptions regarding the noise terms. One such assumption is that the integral of the error on the periods that no message is received, is uniformly distributed, which is not a far fetched assumption. However, they assume that this error plus the process error form a Gaussian error, and variance of the joint process is used in a modified Kalman Filters.

In [SNR07] the authors of the previous paper proposed the use of a basic SOD. They also proposed the use of a filter similar to the filter of the previous paper, with the same two shortcoming, namely, the addition of the covariance of the term $y(t) - y(t_{last})$ into the covariance error of the output noise, and maintaining the last sensor reading in the Kalman filter.

In [NS07] the authors acknowledge that the assumptions made in [NS08] and in [SNR07] are not sensible. Furthermore, they restated the problem of Kalman filtering with SOD sampling in terms that bare a resemblance to Bayesian filtering conditional on the change in the last sample have been smaller than δ in the sampling periods that no new message is received. Nevertheless, there are still some unjustified simplifying assumptions, and more importantly, the new modification was not followed by an increase in the complexity of the estimator, i.e., despite their acknowledgement and the restatement of the problem, the presented solution has barely changed. In [NS09] the authors consider the *I am alive* messages in their analysis.

It should be remarked that there are two types of event-triggered paradigms in the control literature. The first type is a continuous-time event-triggered paradigm in which the new event triggers a process at the moment that it occurs. The second type is a discrete-time event-triggered paradigm in which the event triggers a process that is carried out in the next period, i.e., if a given event does not take place within a given period, then the associated process is not activated. Traditional SOD, which encompasses the SOD schemes presented in this section are of the continuous-time type, whereas [NS07, NS08, SNR07, NS09] are examples of the discrete-type.

In [VK10] it was presented a comparison of three approaches for PID control using SOD: standard SOD, SOD with an observer and a heuristic based SOD PID controller. The authors pointed out that during long periods without transmissions, which is typical of SOD, the variables associated with the integral and derivative part of the PID loose their meaning. In the heuristic case, it was attempted to develop a heuristic that subverted this problem. However, the presented simulation results proved otherwise, in fact, the authors concluded that the three approaches are in general equivalent. However, this conclusion was reached using a suboptimal implementation of the observer and with a heuristic that was presented without any form of justification. Nevertheless, the approach presented in [VK10] is relevant to the current discussion due to the use of observers which are executed with a constant period at the controller, ensuring that it always has an estimate of the state/output at the control instant.

In [XH05] it presented the results of a study the behavior of the estimates, on a continuous-time framework, of systems in which the transmission is regulated by a poison process. They assume that 1) the sensor can replicate the functioning of the receiver Kalman filter and 2) the sensor *knows* whether a given packet was lost. Two situations are studied, 1) the sensor imposes a rate control that varies the value of the Poison rate and 2) the network always drops the packets with a constant Poison rate. The paper presents an extensive discussion of the Poison rates that ensure stability, however, 1) it does not present any filter/controller that allows to achieve such stability bounds and 2) it does not present any justification for

why a control user would choose to employ a poison process to improve/manage the quality of control, as opposed to simpler approaches, for example, by varying a period, hence, it cannot be used in SOD type of controllers. Note that for the network induced errors the reported results are useful for researchers that deal with that type of errors.

Similar contributions [XH04a] and [XH04b], by the same authors, discuss the stability of the networked control framework presented in [VK10]. Once more, some rather unconventional assumptions were made. In fact, the assumption regarding the evolution of each sub-state that is available to each controller defies the definition of single system. Other assumptions are relatively limiting, such as the assumption that all states can be perfectly measured by at least one controller. The last assumption is limiting because one of the most important goals of networked control in networks with packet losses is to find an observable/filter that can cope with missing observations.

In [LSF+03] the authors proved that in every estimation problem, i.e., extensions of the Kalman filter, there is a critical packet drop rate, such that if the network packet drop rate is smaller than such value then exists a filter that guarantees a bounded error auto-covariance matrix and if the the network packet drop rate is not smaller than the critical rate then there will be no filter that guarantees a bounded error auto-covariance matrix. The paper also presented upper and lower bounds for the critical rate. However, the upper bound is equal to the traditional upper bound and the lower bound is a loose bound, i.e., in general the presented lower bound is significatively lower than the actually lower bound.

The authors of [SCS07] claim that progress on complex MIMO models appears to be halted, hence, proposed a SISO *optimal filter*. However, the paper uses inconsistent system dynamics assumptions, i.e., it assumes both that whenever the actuator does not receives a message it outputs the previous outputted value and that whenever a sensor message is not received the output has the same value, which are clearly contradictory. A similar contradiction is found in the indices of the boolean variables used to describe the message receptions. Furthermore, the authors assume that certainty equivalence and separation principle still hold, which is not the case since their method introduce a control dependence in the error covariance matrix.

In [Sch08] it is presented a scenario in which there is a variable communication delay. The sensing element always sends its data at the beginning of a sample period. However, due to packet delays, the messages do not always arrive within their respective sampling periods. It is proposed in [Sch08] to place an infinity sized buffer at the receiver. The sensor would time stamp the samples in order to allow the receiver to reorder them, in case they arrive out of order. The receiver stores the estimate and the respective error covariance matrix of the sample corresponding to the maximum delay related to the size of the buffer. These estimates are computed using the common modified Kalman filter. The authors also consider the possibility of the sensor performing the estimation and send the estimates instead of the samples, whereas the receiver would perform a projection of the state whenever no new message is received and simple copy the value at a message into its state, whenever a new message is received.

A similar approach is presented in [BB08] in which whenever the triggering condition is met the sensor sends the respective message for the controller and the controller responds by sending a message with estimated future system outputs, with an horizon that is a design parameter. Transmissions are triggered when the estimates received from the controller differ from the samples by a given amount. Whenever the controller does not receive a message, it assumes that the trigger condition was not verified.

In [GSHM05] it is considered a scenario in which it is used more than one sensor, not necessarily sensing the same variable. The authors only consider the cases in which the controller and the actuator are collocated or that there is a mechanism in place that informs the controller of whether the actuator correctly received the message, because these are the only possibilities that preserve the applicability of the separation principle. The sensors, running a Kalman filter that assumes that the plant had no input, sends an *encoded* version of their data. The controller assimilate the sensor data (whenever there is new data from a given sensor) by adding the received sensor data to an estimate that takes into account the plant inputs. The authors propose that the controller can be designed using modern control design techniques since the separation principle holds. However, the separation principle does not hold because the sensors cannot successfully apply the proposed algorithm because they do not know the past control sequence. In fact, even though the usual formulation of the Kalman filter does not depend explicitly on the inputs, it depends on the errors of the estimates and the estimates depend on the control sequence. It follows that the controller is also suboptimal. Nevertheless, the two groups of simulations made by the authors showed an improvement over classical approaches, that was due to the fact that as the plant measurements are sent in separate packets, there is an increase of general information available at the controller. And in the second example, it was considered a case in which all control values were zero, hence, the separation principle held artificially.

In [CB08] it is proposed a mechanism to compensate for the network induced delays by incorporating into the control message a series of future control action. The paper deals with non-linear systems and, as many such papers, it does not provide any type of controller. However, it attempts to prove the feasibility of this approach, more concretely, it discusses the Lyapunov stability of such system.

In [KBAF11] it is considered the case in which there were probabilistic acknowledgments and not acknowledgement messages from the actuator to the controller, reporting the correct message reception. Under such assumptions, the UDP-like scenario is equivalent to having an acknowledgement probability of zero and an acknowledgement probability of one corresponds to the TCP-like case. The authors note that the separation principle only holds if the acknowledgement probability is equal to one, which was already known from the TCP-like protocols. Hence, the analysis of this family of networked control systems can only be carried out, using conventional/contemporary mathematical tools, in a subset of cases that had already been computed using conventional tools. In the same article, it is also considered the case in which the sensor/actuator are distributed, with each one having its own link error variable/probability. Once more, the presented results have a structure similar to *classical* approaches.

In [GSGC12] a more in-depth study of such networked systems is done. Optimal estimators and controllers are presented. The presented estimator is equivalent to the estimator in [CB08]. The presented controller has a rather significant mistake on the proof of its optimality, hence, putting its optimality under question. The mistake is related to the fact that the authors consider the expectation of the minimum to be equal to the minimum of the expectation on a function that is clearly non-linear on the variable being minimized.

The authors [HSJ08] presented one of the earliest papers addressing the problem of actuator compensation of missing control messages. In [HSJ08] it is proposed a simple filter that 1) detects the presence of loads, i.e., other slow varying signals that also drive the input of the plant, and it can set the actuator value to compensate for such loads whenever a new control value is not available. Furthermore, the authors propose to tune the actuator's internal filter

according to the load dynamics. This initial paper had many of the *intuitions* of the more complex papers that came after it.

In [MQF12] it is presented a framework intended to provide different Quality of Service (QoS) to control processes that are experiencing different conditions. This is achieved by controlling the packet loss rate associated with the controller in question, though no strategy to achieve this goal is presented. The problem is defined as being a (typical) MPC with additional constraints related to the packet loss probability.

In [MSSG12] and in references therein, it is discussed the situation in which a smart sensor, i.e., a sensor with significant computational capabilities, computes its local estimates of the state variable and sends it to a central node. The goal of the paper is to find a smart sensor transmission policy that minimizes the communication rate while guaranteeing that the error covariance matrix is within a pre-specified bounds. Note the similarities of this problem with the SOD problem. The paper in question further assumed that the communication link was lossy and derived bounds for the minimum rate that the smart sensor should attempt to transmit in order to bound the error at the central node.

2.3.4 Co-design

The co-design problem is relatively old. It stemmed from the necessity to harmonize application level requirements, which in the context of this Thesis are the requirements of the control system, and the functions provided by the control network.

Concretely, the control side has a certain set of requirements related to the latency — average delay, jitter — variance of the delay, packet drop rate, control period, etc., and the network has a number of trade off variables such as the communication speed versus packet drop rate versus energy, jitter versus delay, etc. In the co-design approach, the control and the network are co-designed in order to achieve a better quality of control, hence: the network side chooses its respective trade-off variables in such a way that allows for a better quality of control and the controller side compensates for the non ideal aspects of the network while simultaneously choosing control strategies that allows for the relaxation of certain network requirements.

As pointed out in [YWL+11], there are three main aproaches to the codesign problem: the Central Processing Unit (CPU) scheduling problem, the bandwidth assignment problem and the control theoretic problem. Each one of these problems is dealt with by researchers from their respective areas. In this literature review section, only aspects related to control theory will be presented. A similar review related to CPU and Network scheduling will be presented in Section 3.4.

One of the first papers to discuss the issue of control and network co-design was [HA88]. At the time it was known as *Integrated Communication and Control Systems* (ICCS). At these early times, the main focus was on jitter compensation at the controller side, though with a relatively small degree of success. More successful was the analysis of stochastic stability under jitter, in which some initial results were instrumental to the current understanding of stochastic stability of control systems in networks with communication jitter. An example of such posterior advances is [CvdWHN06], in which it is proven that even stable systems, for each possible delay, can become unstable under a variable delay. On this same paper a tighter stability condition is derived.

Another influential contribution was given in [ZBP01], where it was provided a general overview of the various, already diversified, aspect of control and network co-design, hence, in

a sense, making it a subfield of its own. One of the most important reasons for such success was its publication in a venue that had a relatively high audience with the skills required to jump start the subfield. As referred above the paper did not introduced many new concepts but it presented a discussion, regarding the stability of networked control systems with jitter, very similar to the discussion on [HA88]. Moreover, it presented a condition for the stability of networked systems with packet drops.

[LW12] presented a condition that provides a rather tight bound, based on the Lyapunov-Krasovskii criteria. However, it is applied to $\mathcal{H}_\infty$ minimization problem, which is not the type of problem tackled on this Thesis. [GJ10] deals with an expanded version of the previous problem, i.e., it is also considered that a message may be lost and under these circumstances the actuator holds its value until the correct reception of a new message. Though some results were reported, the tightness was not established.

[MSZ11] considered a controller/scheduling system in which each process had a constant activation rate. However, the scheduler could decide to not schedule a given controller in a certain period. Whenever this happened the respective actuator used the hold strategy. A necessary condition, for the rate at which each controller is executed, was presented. In fact, the condition is equal to the condition for the lossy networks that had already been presented in the literature, as discussed in the last subsection. Under these assumptions, the goal is to choose the miss ratio of the jobs associated with each controller process in a manner that guarantees the stability of all processes. The authors state, without a formal proof, that this problem is NP-hard, i.e., that in general it cannot be solved in polynomial time, hence, justifying the use of a suboptimal heuristics. Nevertheless, a few important points remained open, the first being related with the fact that no schedulability test is proposed. In fact, this problem is minor since the authors propose to test a sufficient condition for stability at the beginning of the execution. The second aspect is the fact that the system does not use all resources available to it, it simple gives to each controller a rate that guarantees stability and it does not increase it if there is slack in the utilization, even in the start up.

In [CQG11] the authors take a network approach to a system with an input perturbation. The perturbations are used to define the signal to noise ratio, which in turn defines the channel capacity as defined in the Shanon channel capacity. Then the authors determine the smallest total capacity, in which total capacity denotes the sum of the capacity of each input channel, that ensures the stability of the system. In [CZQ12] the authors noted two issues with their previous proposal: 1) given that the separation principle holds, then any stabilizable system continues being so under their assumptions. In fact, their assumption are in all similar to the traditional assumptions, but framed in a different manner. 2) Each input has one perturbation variable associated with it. This assumption is far from realistic since in practice there are correlations between such terms and often the perturbation matrix does not have the rank of the input matrix. The authors then propose to use of their framework in a scenario in which the inputs passed through an amplifier with a multiplying variable, i.e., random, gain. This is a rather common situation that can be induced by, for example, temperature variations, thus, providing a usefulness to their approach. Further developments were proposed in [QGC13]. Nevertheless, the authors still do not address the fact that the application of classical approaches to this problem yield results better than the results of their framework.

In [DLG10] it was proposed a possible network scheduling/control co-design approach. It was first noted that systems with varying delays can become unstable if the control variable is changed at each control period. Hence, it was proposed that each controlled system acted in

open loop whenever it did not receive any new control loop, i.e., use the zero strategy. Then it is proven that, on average, the controller lowers the state variable during period of closed loop operation more than it increments the state variable during the control periods in which the system is in open loop, then stability is guaranteed. The authors proceed by proposing a periodic scheduler in which at a given period only a subset of the controllers is activated, in order to ensure schedulability, and that each controller is activated with a given period — an integer number of the controller rate, to ensure stability.

This line of inquiry had already been presented in [HV01] and in references therein, in which a subset of controllers were given the ability to control their respective processes for a limited amount of time, in order to drive the processes into a low error zone. This was repeated with a periodicity that resembles a static schedule.

In [LZFA05] it is investigated the possible improvements of the $\mathcal{H}_\infty$ norms that can be achieved by properly designing a network. The authors focused on the design of the quantizer and of the possibility of system destabilization due to quantization error.

Chapter 3

Real-Time Systems

> The roots of scholarship are bitter, but its fruits are sweet.
>
> — Aristotle

This chapter provides an overview of real-time systems topics related to control theory. Hence, it will focus on contributions that have an impact on the system codesign. A brief introduction into real-time systems will be provided in order to make the chapter self contained.

3.1 Real-time Systems

Computational systems, in general, are composed of one or more processes or applications. To these processes, it is associated a series of resources, such as computational resources, memory, printers, etc..., required to carrying out the requests of the user and/or of other processes.

Processes are themselves divided into tasks which are assigned some well defined part of the process. For instance, a task can be assigned to communicate with the printer, or to find the solution of a particular equation, or to read inputs from a keyboard. Tasks communicate with each other in other to achieve the goals of the process.

Tasks are also divided into jobs. There are two main strategies to perform this division: in the first one, and historically the first emerge, each job was defined as each instance of the execution of a given task. Under this approach, the notion of job was a dynamical concept since it depended on the amount of processor time assigned to the task in that particular instance. The second strategy to define a job, which also appeared later, is to divide the task in a series of pre-specified points, followed by an assignment of a group of characteristics (that will be presented in this chapter) to the respective interval. These intervals are the so called jobs.

Modern computational systems are capable of executing several processes using a mechanism called time sharing. More recently, with the appearance of multi-core platforms, jobs can also be executed with a certain degree of parallelism.

A task is defined as being real-time if the moments at which it results become available are as important as the logical correctness of the results in question [SR89, Kop97]. A task is said to be Hard Real-Time if failing to complete the task in its respective deadline may lead to catastrophic outcomes. A process is said to be Soft Real-Time if not meeting its dead-line

causes a graceful reduction of its respective usefulness [Kop97]. A real-time system may be composed of several real-time tasks. A hard real-time system comprises at least one hard real-time task. Otherwise the system is said to be soft real-time.

When characterized according to periodicity tasks can be periodic or non-periodic. In periodic tasks, a job is activated every T units of time, with T called the period. Non-periodic tasks are those in which there is not a constant interval of time between consecutive activations. Non-periodic tasks are in turn subdivided into sporadic tasks in which there is a minimum amount of time between two consecutive activations, such value is denoted minimum inter-arrival time (*mit*), or simply denoted T. Aperiodic tasks are those that do not have any of the patterns described above.

3.1.1 Task Models

There are many possible task models, with varying degrees of detail and complexity, for example, in this Thesis it will be necessary to use two more parameters to describe systems that are fail tolerant.

In one of the simplest task models, periodic and sporadic tasks are defined by: a Worst Case Execution Time (WCET), denoted by C, which is the maximum amount of CPU time that a process requires in order to finish its computation; a relative Deadline, denoted D, which is the latest moment counting from the activation of the process for which the results of the computation remain useful; a Period or minimum time between activations, denoted T.

These characteristics are bundled together to describe a single task as

$$\tau = (C, T, D). \tag{3.1}$$

For periodic tasks, it is also common to define a phase (or offset) which is the instant counting from a global starting point in which the process is activated for the first time.

The set of all tasks in a system is usually referred to as the task set. These and other introductory topics can be found in [Bur91].

3.1.2 Classification of Real-Time Systems

This subsection presents the classification of systems according to the characteristic that have the highest impact in the design of real-time systems.

Static Vs Dynamic Priority Assignment

When classified according to their priority assignment, there are two type of systems:

- *Static (fixed).* Task's priorities do not change during execution, i.e., each job from a given tasks always supersede or is superseded by a job from another task.
- *Dynamic.* Task's priorities change during execution, i.e., the priority relationship between different jobs from a given set of tasks changes throughout time.

Among dynamic task, the jobs can be classified as

- *job-static.* If the job's priority does not change with time.

- *job-dynamic*. If the job's priority changes with time.

Static (task) priority assignment schedulers are normally less efficient at scheduling task sets than their dynamic counterparts. However, static algorithms have many other advantages that in certain applications can outweigh their inefficiency. The greatest advantage of static scheduling is its predictability in the presence of overload. Static scheduling techniques are predictable in the presence of overload in the sense that it can be known *a priori* the order by which task's deadlines will not be met (normally those of lower priority). By doing so, it assures that some tasks will meet their deadlines even under transient overloads. This assurance is crucial in Hard Real-Time systems.

The greatest advantage of dynamic priority assignment algorithms is their efficiency. For instance, for independent task sets, the schedulability of dynamic priority systems can be as high as 100% of the system utilization, whereas for static systems the utilization can be limited to 69%, see Subsection 3.1.3.

Static Vs Dynamic Systems

Systems may be classified as static or dynamic. Static systems are those in which the set of tasks does not change throughout time. Conversely, dynamic systems are those in which the task set may change in runtime. A task set can change by the insertion, removal or change of the defining characteristics of one or more tasks.

The online inclusion of a new task into the task set, may imply, if schedulability guarantees need to be provided, the use of an online schedulability test that increases the complexity of the system.

Resource Schedulability

A general problem in the allocation of shared resources is to define which process gets which resource at which instant. To guarantee a smooth operation it is necessary that each resource is used by a limited number of processes at a time, taking in consideration that all processes that are competing for the possession of the resources need them to complete their tasks.

The scheduling discipline deals with finding a resource allocation order, i.e., resource schedule, that ensures that resources will be shared orderly, so that all the processes will be able to access all the resources they need and complete their execution within well defined deadlines.

From the CPU standpoint, which is one of the shared resources, a system is schedulable if, and only if, there is at least one execution order of its tasks, such that all of them meet their respective deadlines. This implies that a task is schedulable if its deadline is higher than the sum of its preemption, execution and blocking time, see [Obe94].

In some systems, a schedulability analysis is carried out whenever the task set is about to be changed. The change is rejected if it results in a situation in which schedulability is not guaranteed. Other systems do not include such mechanisms, thus any change to the task set is allowed without providing any schedulability guarantees. These systems are called best-effort.

Online Vs Offline Scheduling

Scheduling can be done offline if activation times are known *a priori* or online as tasks become ready.

Offline scheduling allows the use of complex scheduling algorithms and optimization, thus, it can be more efficient. Furthermore, offline schedules have lower run-time computational demands, since the schedules are already built. However, offline scheduling of dynamic systems is not an option, since whenever the task set changes the schedule has to be rebuilt. Adding to this downside, offline scheduling may also have high memory requirements, in order to store the scheduling table. This is worse for large task sets in which the periods are relative primes.

Resource Sharing Control

In many scheduling algorithms it is assumed that tasks run independently of each other. However, in most applications, the tasks are not independent. In fact, in a typical computational system there are many shared resources that must be accessed in a mutually exclusive manner, thus leading to interdependency. In resource sharing, there are situations in which a low priority process acquires a resource then gets preempted by a medium priority process. During the time in which the referred low priority process is preempted, all processes with priorities higher than the medium priority process, that require the blocked resource to continue their execution are blocked. This is called an (indirect) priority inversion between the medium priority and the higher priority processes. In general, a system is said to be in a priority inversion if a task priority supersedes an higher priority one. Without proper mitigating measures such blocking can reach long periods, thus causing deadline misses.

To mitigate this effect, protocols such as Priority Ceiling Protocol (PCP) and Priority Inheritance Protocol (PIP)[SRL90], can be used. In PIP, whenever a process gains access to a shared resource, it is attributed a priority equal to the priority of the highest priority process that is also competing for the possession of the resource in question. In PCP a process is allowed to enter a critical section only if the critical section will be executed at a priority level higher than any preempted critical section. At this point the priority of the critical section is ceiled to the new process entering it.

With these two mechanisms in place, a system is protected from unbounded priority inversion. For PCP it is also protected from deadlock.

Statical priority assignment analysis has been extended to incorporate blocking processes. To this end, first consider the concept of Maximum Blocking Time (B_i), which in analogy with the C_i, B_i is the maximum amount of time that jobs from lower priority tasks will be executing while a job from the task in question is waiting due to blocking.

3.1.3 Scheduling Algorithms

This subsection presents the most common scheduling algorithms. Starting from static priority assignment protocols and ending with dynamic priority assignment schedulers.

Rate Monotonic Scheduling

Rate Monotonic Scheduling (RMS) is one of the oldest scheduling mechanisms, due, in part, to its simplicity which allows it to be implemented in rather slow systems. It is a static

priority scheduling algorithm.

In RMS, processes are assigned priorities as a monotonic increasing function of their rates, i.e., $\forall \tau_i, \tau_j \in \tau, T_i < T_j \Rightarrow P_i > P_j$.

Classical RMS analysis has a series of assumption, introduced in [LL73] and further discussed in [SKG91]:

- Instantaneous switching between tasks;
- All execution time is dedicated to tasks;
- Tasks do not interact with each other;
- Tasks become ready to execute precisely at the beginning of their periods and relinquish the CPU only when execution is completed;
- Tasks deadlines are always in the start of the next period (deadlines are equal to periods);
- Tasks with shorter periods are assigned higher priorities, regardless of task criticality;
- A lower priority task is never executed when a higher priority task is ready.

The essentials of RMS analysis were developed in 1973 by Liu and Layland [LL73] where they showed that a set of processes is guaranteed to be schedulable if:

$$U(N) = \sum_{\tau_i \in \tau} \frac{C_i}{T_i} \leq N(2^{1/N} - 1) \tag{3.2}$$

where U is the CPU utilization factor, i.e., the fraction of time that the CPU is being used, and N is the number of tasks. The utilization factor converges to $ln(2)$ ($\approx 69.31\%$) as $N \to \infty$.

Another schedulability test was presented in [BBB03] and is called the hyperbolic bound test:

$$\prod_i \left(1 + \frac{C_i}{T_i}\right) \leq 2 \tag{3.3}$$

Both of these are sufficient tests, i.e., U being lower than the test bound is a sufficient condition for the system to be schedulable. However, they are not necessary conditions, i.e., if U is higher than the test bound, but with $U \leq 1$, then the test is inconclusive. It was shown in [BBB03] that the hyperbolic bound is tighter than the Liu and Layland bound in the sense that for every task set that the Liu and Layland bound is verified the hyperbolic test is also verified, however, there are cases in which the Liu and Layland is inconclusive but the hyperbolic test is conclusive.

RMS is an optimal scheduling mechanism within its assumption set, in the sense that if exists a scheduler that satisfies RMS assumptions and is capable of schedule a given system, then RMS is also capable of scheduling the system in question.

The fact that U only provides a sufficient condition, prompted the appearance of less conservative tests, such as the response time test, first introduced in [ABRW91] for the Deadline Monotonic Scheduling, and simplified to the RMS case in [Obe94], and is based on the fact that was proven in the paper in question: "For a set of independent, periodic tasks, if each task meets its first deadline, with worst-case task phasing, the deadline will always be

met". Or more bluntly, a task set is schedulable if the response time of each task is lower than its respective deadline.

The response time is defined as being the largest amount of time that goes from the activation of a job to its completion. The response time can be computed using the following formula, in which tasks are sorted by their respective priorities, i.e., $\forall_{i<j}, P_i \geq P_j$:

$$a_i^k = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_i^k}{T_j} \right\rceil C_j, \qquad \text{where } a_0 = \sum_{j=1}^{i} C_j$$

The response time is the value of a_i for which $a_i^{k+1} = a_i^k$, which is the stopping condition of this recursive definition of response time, though its computation can also be halted if $a_i^k > D_i$.

The last test is both necessary and sufficient, hence better than the test introduced in [LL73]. However, it carries a computational burden that makes the initial test an attractive option in systems with low computing power.

Deadline Monotonic Scheduling

Though RMS was almost established as the *de facto* static scheduling mechanism, it had a rather restrictive set of assumptions. Deadline Monotonic Scheduling (DMS) was proposed to lessen one of RMS limitations, namely the *deadline equal to period* assumption made in RMS. In DMS the deadline of a process is allowed to be smaller than its period, and the priorities are attributed as a monotonically increasing function of the inverse of the process deadline. It is worth stressing that DMS in which all deadlines are equal to their respective periods is equivalent to RMS.

In [LW82] it was shown that DMS is an optimal priority assignment strategy, meaning that if there is any static priority assignment capable of scheduling a given set of tasks that satisfy DMS assumptions, then DMS also is.

DMS analysis adds a set of assumptions and conventions, relatively to RMS: $\forall \tau_i \in \tau$: $C_i \leq D_i \leq T_i$ and the processes are sorted with $i = 1$ being the process with highest priority (lowest deadline) and $i = n$ being the process with lowest priority (highest deadline), i.e., $\forall_{i<j}, Pr(i) \geq Pr(j)$.

The first proposed schedulability test for DMS is a sufficient but not necessary test. That was derived from the same idea that Liu and Layland used to derive the RMS utilization bound: the worst case for schedulability is the case in which all processes are launched simultaneously. This moment is denoted critical moment; the test determines if it is possible to schedule a given set of task when it starts from a critical moment. Moreover, If a set of processes is schedulable when it starts from a critical moment then it will be schedulable in all other situations. This and other tests are explored in depth in [ABRW91]. Numerically, the test is given by:

$$\forall i : 1 \leq i \leq n : \frac{C_i}{D_i} + \frac{I_i}{D_i} \leq 1 \qquad \text{where } I_i = \sum_{j=1}^{i-1} \left\lceil \frac{D_i}{T_j} \right\rceil C_j \tag{3.4}$$

Note that I_i is the amount of interference that process i suffers from higher priorities processes that have been launched before its deadline. This test is conservative (since it is sufficient but not necessary) because it considers that all higher priority processes will be launched before

the lower priority ones, and more importantly will terminate before the low priority ones have started, which may not always be the case. [ABRW91] presented a, more complex, necessary and sufficient test based on this observations.

Scheduling Tasks With Arbitrary Deadlines

In Subsection 3.1.3, the principles of schedulability analysis of task sets scheduled under DMS was presented. However, in many real-world applications deadlines are not restricted to cases in which all of them are not greater than their respective periods.

The DMS schedulability test can be summarized into:

$$a_i^{k+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{a_i^k}{T_j} \right\rceil C_j \qquad \text{with } a_i^0 = C_i \tag{3.5}$$

where hp(i) is the set of all tasks that have a priority higher than the priority of task i, a_i is the response time of process i, and a_i^k is the k^{th} iteration of its computation. The test stops either if $a_i^k = a_i^{k+1}$ or if $a_i^k > D_i$, and the system is schedulable $\iff$ $a_i \leq D_i$.

This test is not valid for the case in which exists at least one task with a deadline greater than its period, essentially because the computing time of preceding jobs of task i are not accounted for. To introduce the previous effect, [Leh90] introduced a new concept: *busy period* - a busy period level i is the maximum amount of time in which jobs of tasks of processes with priority equal to or greater than the priority of task i may be executed.

Making considerations similar to those made in section 3.1.3, or those that where argued for in [Leh90], and later formally shown in [JP86] and [TBW94] it was shown that:

$$w_i^{k+1}(q) = (q+1)C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^k(q)}{T_j} \right\rceil C_j. \tag{3.6}$$

In this equation $w_i(q)$ is the busy period level i relative to the time interval $[t_0, t_0 + qT_i]$, and $w_i^k(q)$ is the k^{th} iteration of its computation. The computation ends when it converges. The maximum release time of process i is given by:

$$r_i = \max_{q \in \mathbb{N}} (w_i(q) - qT_i) \tag{3.7}$$

and the process i is schedulable $\iff$ $r_i \leq D_i$.

Since $w_i(q)$ is never lower than C_i it is usual to initialize $w_1^0(q) = C_i$. Similarly, since $w_i(q+1) \geq w_i(q)$ then $w_i^0(q+1) = w_i(q)$ is a good initialization value.

This analysis can be easily extended to include the effects of jitter and to the *bursty* behavior of some processes as it was done in [TBW94].

First Come First Served

In the First Come First Served (FCFS) policy, whenever a process is activated it is attributed the smallest priority. As time passes, the process ages, and its priority increases. The highest priority process is scheduled next.

FCFS is one of the simplest possible dynamic priority assignment algorithm, hence it has found applicability among systems with low computational capabilities.

A task set is FCFS schedulable if, even if, all tasks arrive simultaneously in the queue, without considering re-arrivals, it is still schedulable; i.e.

$$\forall_{i:\tau_i \in \tau} \quad \sum_{j:\tau_j \in \tau} C_j \leq D_i \tag{3.8}$$

Earliest Deadline First

Earliest Deadline First (EDF) is a dynamic scheduling priority assignment algorithm. In EDF the next job to be scheduled is the job with earliest (nearest) deadline. It has a utilization bound of one, assuming that all deadlines are equal to their respective periods, i.e.,

$$U = \sum_{i:\tau_i \in \tau} \frac{C_i}{T_i} \leq 1. \tag{3.9}$$

In so being, EDF is an optimal scheduling policy, among dynamic priority assignment policies in which all processes deadlines are equal to their respective periods, since no scheduling policy can achieve utilizations higher than one.

A priority assignment algorithm is said to be predictable under overload if whenever some task set is not schedulable, then it is known *a priori* which subset of the task set will not be scheduled. This is the case of RMS and DMS, but is not the case in EDF.

EDF has two main disadvantages: it introduces a considerable overhead during its dynamic priority assignment and it is not predictable under overload. These two factors have delayed EDF industrial deployments despite its wide acceptance in the research community.

EDF is optimal even when deadlines are smaller than their respective periods, and in that situation the following is a sufficient but not necessary test:

$$U' = \sum_{i:\tau_i \in \tau} \frac{C_i}{D_i} \leq 1 \tag{3.10}$$

The last test is equivalent to increase the utilization, since $U < U'$, and due to its sufficiency, when it holds then the task set is schedulable. However, if it does not hold then more complex tests are required.

A formal analysis of EDF with deadlines not greater than their respective periods was made in [Spu96] and [LG96]. Here some results are presented. To perform a sufficient and necessary test it is necessary to introduce a new concept, namely the Demand Function.

The demand function of a process is defined as the amount of computation necessary to be done between instants t_1 and t_2 so that the process is schedulable. Or putting it more formally:

$$df_i(t_1, t_2) = \sum_{\substack{a_i \geq t_1 \\ d_i \leq t_2}} C_i \tag{3.11}$$

where a_i is the moment at which process i was activated, d_i is the deadline of process i, and C_i is the amount of computation performed by process i. So in a sense the demand function of process in an interval $[t_1, t_2]$ is the worst case amount of computation that process i has to perform in that period.

The demand function of a set of processes on a given interval is defined as the sum of the demand functions of each of its processes:

$$df(t_1, t_2) = \sum_{i:\tau_i \in \tau} df_i. \tag{3.12}$$

The definition of demand function implies that "A task is (EDF) schedulable if $\forall t_1, t_2 \ : \ t_2 > t_1 \quad df(t_1, t_2) \leq t_2 - t_1$". For example, computing in 5 seconds what the hardware can only compute in 6 seconds is impossible. This remark provides a necessary and sufficient schedulability test for EDF. However, it would be necessary to test an infinity number of intervals in order to decide about the task set schedulability. To overcome this hurdle, the notion of critical moment introduced earlier in this section must be incorporated. So consider only the situation in which at the instant $t = 0$ all processes are activated. So, the test becomes:

$$\forall t \quad df(0, t) \leq t \tag{3.13}$$

The last observation induces the definition of a new measure, namely the Demand Bound Function:

$$dbf(L) = \max_t(df(t, t + L)) = df(0, L). \tag{3.14}$$

Equation (3.14) defines a unidimensional function, which can be used is a unidimensional test, as opposed to a bi-dimensional test in t_1 and t_2), but it still necessary to test an infinity number time instants. If all processes of the set are periodic then it is possible to reduce it to a finite number of test based, for example, up to the least common multiple of all periods. The resulting test range is denoted L^*. Note that this test does not need to test all the instant of the interval in question, it suffices to test the instants in which the dbf changes.

Least Slack First

The Slack of a process is defined as the maximum amount of time that its execution can be delayed while still meeting its deadline.

Least Slack First (LSF) or Least Laxity First (LLF) is a scheduling algorithm that was proposed to deal with a deficiency of EDF: in EDF the next process to be executed is always the one with the closest deadline, even if there is another process that will meet its deadline only if scheduled next, due to its high execution time.

LLF tries to improve EDF by executing the process with the lowest slack. Therefore, situations as those described above do not result in a failure to meet the deadline.

However, LLF has a high temporal locality LLF is local in the sense that the slack using in the scheduling is computed in a given time instant and will not be equal in the following instants. LLF's high temporal locality causes two problems, one of them being *thrashing* that occur whenever two or more jobs have the minimum slack of the system. Whenever one of them is scheduled, after running for a unit of time, the other becomes the job with the least laxity.

[HGT99] proposed a solution for the *thrashing* problem: whenever two or more processes have the same slack the one with lowest deadline is executed and the others are excluded. These processes continue excluded until the execution of the current process is halted or is preempted by another process. A process may preempt an excluding process if it is not excluded and its laxity is the lowest in the system.

Another solution to the *thrashing* problem is given in [WD07]. This solution builds upon the previous one and is based on fuzzy rules (thresholds). [WD07] presents many solutions to the *thrashing* problem that are between the two presented in this section.

Servers

Aperiodic servers emerged to fill the necessity to include aperiodic tasks in the model set. Servers allow to schedule together periodic and aperiodic tasks, while guaranteeing that the later have a desired priority level and a bounded interference on the former.

Aperiodic servers are defined by their replenishment period and an execution budget. An aperiodic server is a logical task created to handle randomly arriving requests, with a priority, replenishment periods and execution budget attributed based on some scheduling policy. The execution budget of a server is diminished each time that it handles a request. The server itself acts as a periodic process to which resources are attributed, moreover, it is defined as any regular process τ_i.

The simplest aperiodic server is the background server. Basically, whenever there are no ready periodic tasks, the background server attributes the execution to the highest priority aperiodic task, if any. Under this server the aperiodic tasks have no influence over the schedulability of periodic tasks. Furthermore the aperiodic ones have the lowest priority of all. This raises serious constrains since the periodic tasks may never be all in simultaneous idleness, causing the aperiodic tasks to never be executed.

To solve this problem, the polling server was introduced. The polling server has a given period, priority and execution time, like any other periodic tasks. When it is scheduled, it polls the aperiodic tasks associated with itself, and if the ready queue of aperiodic tasks is not empty it executes the higher priority one. The complexity of this server is comparable to the complexity of the background server. Under this, server aperiodic tasks interfere with periodic ones in a controlled and bounded way and their priority is not necessarily the smallest. However, aperiodic tasks have a poor reactivity, since if they are activated immediately after the pooling server, they must wait a full period, even if the server has not executed, as illustrated in Figure 3.1.

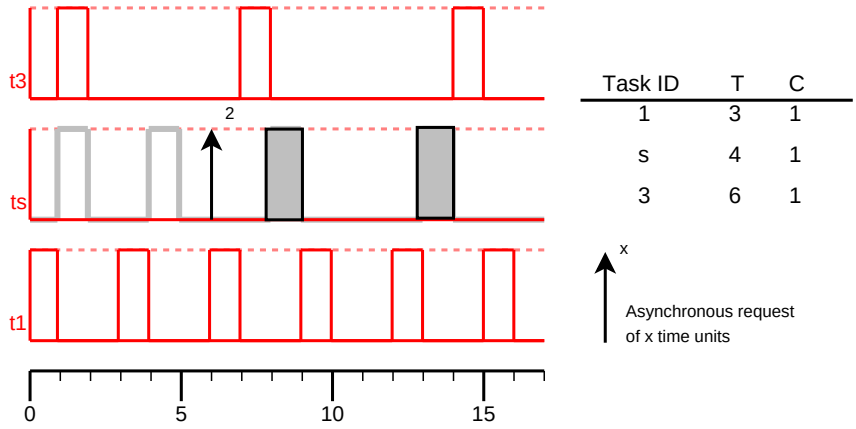


Figure 3.1: Example of an Aperiodic Server — Polling Server, aperiodic jobs were released sooner, but were executed only at the polling instants

To solve this issue it was introduced the deferrable server. The deferrable server differs

from the polling server because if the deferrable server becomes the highest priority task, it only executes if there is an aperiodic task associated to it waiting for execution, furthermore, in these instants its capacity is not spent. Otherwise, it defers its execution until such instant. In the deferrable server, provided that there is capacity available to do so, aperiodic tasks can be served at any time during the server period. This solves the main issues of the polling server. Nevertheless, it introduces another one, namely this server interferes with periodic processes in a way that is not equivalent to a periodic task, thus reducing the schedulability bound of the system and increasing the response time of periodic processes. For instance, one such interference is under RMS, in which the assumption that active tasks execute whenever they become the highest priority tasks.

One of the servers introduced to solve the problems in the deferrable server is the sporadic server. In the sporadic server the execution budget is not replenished in the beginning of each period. While in the polling Server and the deferrable server, the budget is replenished periodically, in the sporadic server the replenishment is normally scheduled whenever the server capacity starts to be consumed. The sporadic server is more complex than the background server.

There are also servers for dynamic priority systems, such as the Total Bandwidth Server (TBS) [SB94] and Constant Bandwidth Server (CBS) [AB98].

3.2 Real-Time Communication Networks

The last decades were marked by a steady development of real-time communication networks, fueled by the steady development of automation that took place in the industry. Informally, a Real-time communication network is a set of communication links placed to provide connectivity between field devices, e.g. sensors, actuators, controllers.

Real-time communication architecture differ from general purpose communication systems mainly because Real-time communication network have to meet specific real-time requirements, namely: efficient short messages handling, support periodic and aperiodic traffic, bounded response time, redundancy, low implementation and maintenance costs.

In the remainder of this section an overview of some basic Real-time communication concepts will be presented as well as an overview of some of the most relevant Real-time network protocols.

3.2.1 Real-Time Communication Architecture

Real-time communications are usually implemented based on some kind of layered multiple access networks communication architecture [MZ95]. Each layer has a set of protocols responsible for carrying out specific operations that are made available to higher layers. Figure 3.2 shows the architecture of the International Standard Organization (ISO) Reference Model for Open Systems Interconnection (OSI) [Con10]. However, real-time communication networks frequently employ a *collapsed* OSI-based architecture, in which the upper 5 layers are merged into a single *application layer*, as shown in Figure 3.2.

Following this philosophy the following layers are not usually implemented in real-time communication stacks: application interface, providing common services required to the particular applications; presentation layer, to provide a vendor independent data access, in order to allow for interoperability; session layer, to allow opening and closing dialogs between senders and receivers; a transport layer, to handle end-to-end communications; a network

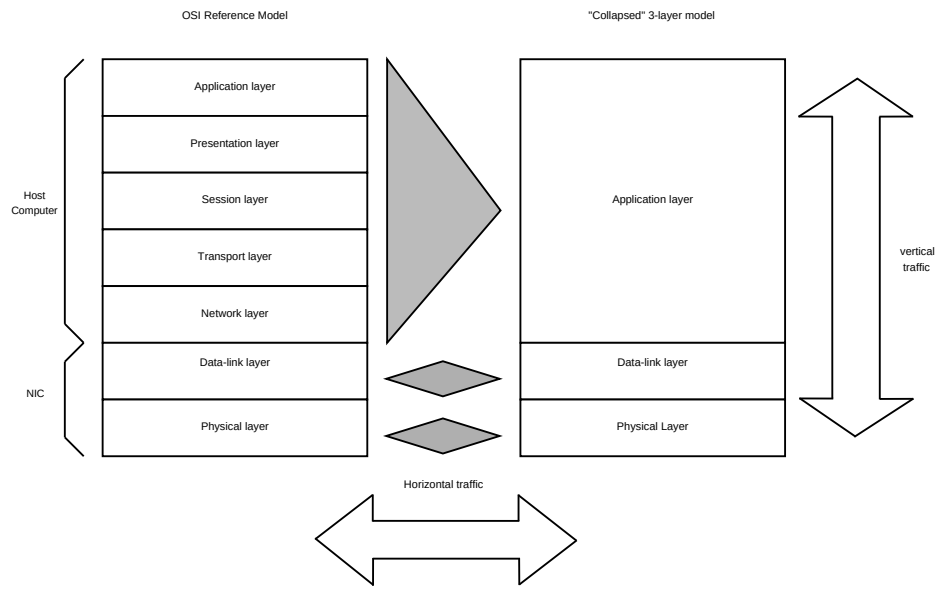


Figure 3.2: Left OSI model, Right collapsed OSI model with marking of vertical and horizontal traffic.

---

layer, to handle node addressing and message routing. Still under these philosophy, the following layers are usually implemented in real-time communication system: data-link, layer responsible for the access to the communication medium and logical data transfer; and finally a physical layer, to handle the physical transmission of messages over the communication medium (pin assignments, number of wires, electrical characterization, repeaters, etc).

The performance of the communication system as a whole strongly depends on the performance of each layer. New techniques have recently been proposed to reduce the time spent in the internal processing at the different protocol layers, for example by providing distinct queues and paths for real-time and non-real-time traffic (e.g. [SJH02]). However, the data-link layer is important, because it is the layer responsible for deciding when and for how long nodes can access the bus.

The real-time communication physical layers are chosen according to its intended characteristic: data rate, distance between nodes, voltages and currents, cabling systems (coaxial cable, twisted pair, infrared, current carrier, optical fiber, to name a few). Another physical layer design constraint is the location in which it will operate: (high) temperature, (high) pressure, (rapidly) varying electromagnetic field, (deployment site) ease of access. The last example is an area that wireless real-time communication promise to revolutionize. The real-time communication physical layer has an impact on its respective error statistics.

3.2.2 Real-Time Communication Cooperation Models

Distributed systems comprise several autonomous processing units that cooperate to achieve a common goal. Information exchange is carried by a suitable communication system and consists not only of the physical transmission of messages but also in the way that they are distributed. Depending on the particular application, nodes may need data that resides in one or more other nodes (many-to-one). Conversely, the same data can also be needed in several distinct nodes (one-to-many). More generally, communications can be one-to-one,

one-to-many, many-to-one and many-to-many.

Figure 3.2, beside presenting the aforementioned OSI and collapsed-OSI reference models, also presents the concept of vertical traffic and horizontal traffic among two nodes. Horizontal traffic is depicted by the horizontal line as spanning both nodes, and vertical traffic as spanning the several layers of each node's protocol stack.

The client-server is a data exchange model, in which nodes that are sources of some data behave as servers. Nodes that need the data (clients) issue requests to the respective server, which in turn replies with the data in question. This communication model is inherently one-to-one, and can lead to both spatial and temporal data inconsistency problems when used to support one-to-many or many-to-one communications. For instance, if the same data is required in several nodes, different nodes issue the respective requests to the server. If the data value changes during this period, the successive replies of the server will carry different values of the same entity, resulting in spatial inconsistency. On the other hand, when a node needs data from different servers, it must issue the requests sequentially, one after the other, which can result in temporal inconsistency. Another problem posed by this model is related with the internal servers scheduling and processing of requests.

Data exchanges can also be defined in terms of producer-consumer, as opposed to client-server. In the producer-consumer framework, as introduced in [TN89], a given physical variable, e.g. temperature, is associated to a system variable. Entities that generate or convert the physical value into the respective system value are called producers (of the referred variable). The producer-consumer model associates unique logical handles to each message type. Messages are generated and received based only on these logical handles, without any explicit reference to the message's source or destination nodes.

Due to its broadcast nature, the producer-consumer model inherently supports one-to-one and one-to-many communication, without causing spatial data consistency problems. However, this property may be lost if the underlying network does not support atomic broadcasts.

This model, however, does not solve the problem of temporal consistency, that can be introduced, for instance, by the contention resulting from a multiplicity of producers on the network. This problem has been solved by the producer-distributor-consumer (PDC) model [TN89], which adds coordination to the producer-consumer model. In the PDC model the producers behave as slaves with respect to an arbitrator node. Such node is fed with the message properties and temporal requirements of the messages that are exchanged in the bus and builds a suitable schedule, which, is then used to grant to a single producer at a time the right to transmit.

Real-time communication systems are comprised of two types of traffic [FV92]: 1) *Vertical traffic* which is comprised mainly of requests from upper-level entities to lower level entities, and subsequent confirmations (acknowledgments) from lower level entities to higher ones; this type of traffic follows a client-server philosophy. 2) *Horizontal traffic* which is comprised of data exchange between entities of the same layer, e.g. between the physical layer of a sensor and a controller, or between the application layer of two controllers. Note that in the latter case, for the correct exchange of horizontal information, vertical exchanges would also be necessary.

Traffic Model

Real-time communication systems normally include time specifications. These specifications are used to verify if a set of messages can be delivered in a manner that guarantees

the coherence of actions and/or data. The most used time specifications are: periodicity, jitter, application process response time, and some possible relations between the timing specifications of two different messages.

The periodicity, in this context, refers to the (minimum) amount of time between two transfers. The jitter is a measure of time variation in relation to a pre-specified time set. As examples of possible real-time communication jitters include, the transmission instant, the reception instant, transmission duration, etc. For instance, [But97] defines two types of jitter, relative jitter, which is the maximum deviation of the start time of two consecutive instances $\mathrm{RRJ_i} = \max_k \{(\mathrm{s_{i,k}} - \mathrm{r_{i,k}}) - (\mathrm{s_{i,k-1}} - \mathrm{r_{i,k-1}})\}$, where $\mathrm{s_{i,k}}$ is the starting time of the $\mathrm{k^{th}}$ instance of $\mathrm{i^{th}}$ task and $\mathrm{r_{i,k}}$ is the respective release time; and [But97] also define absolute jitter as the maximum deviation of the start time among all instances: $\mathrm{ARJ_i} = \max_k (\mathrm{s_{i,k}} - \mathrm{r_{i,k}}) - \min_k (\mathrm{s_{i,k}} - \mathrm{r_{i,k}})$. High (relative) values of jitter degrade the quality of the real-time communication.

The response time is the (maximum) delay between a request and a confirmation, or between a demand and a response, or any other two similar events. Response time can be measured either locally, for example, time from the moment that the request was sent to when the response was received or between distributed occurrences. For more in time constrains of real-time communication systems, refer to [Des85].

Real-time communication are used to transmit data between distributed control elements. Most of this traffic is periodic in nature, with a periodicity that is defined by the control algorithm. Thus, this type of traffic is time-triggered (or at least time-driven), in the sense that it occurs when a given instant is reached. Another type of traffic is called event-triggered and it occurs when a given event takes place. Event-triggered transfers are particularly useful for variables that do not change their values regularly. A deeper analysis of periodic and aperiodic (time/event-triggered) traffic (or messaging) is performed in [Kop91].

3.2.3 Real-time Communication Medium Access Control

Real-time communication systems Medium Access Control (MAC) protocols can be divided into two main classes: controlled and uncontrolled. A MAC is controlled if there is a signal (explicit or not) that informs the nodes about who is next to transmit, hence preventing the occurrence of collisions. In uncontrolled MACs there is no (global) arbitration scheme, therefore collisions may happen. Usually, in such cases, measures are taken to prevent and/or detect the occurrence of such collisions.

A controlled MAC is said to be centralized if there is a (central) node that handles the messaging schedule, otherwise it is said to be distributed. In distributed unontrolled MACs, all nodes willing to transmit in a given time slot/frame are involved in the arbitration. These notions and other related with fieldbuses are discussed in detail in [Tho98].

Controlled MACs provide real-time guarantees, are predictable, thus are appropriated to control application; however, most controlled MACs are inflexible. Uncontrolled MACs are flexible but tend to not provide real-time guarantees.

The controlled MACs presented so far are (mostly) static, always behaving in the same way. For introducing some dynamic behavior, thus allowing the MAC to be more adjustable to the situation in hand, [Foh93] proposes a mechanism that allows the MAC to have different operating modes. In many protocols that implement CSMA-CA there is already a dynamic behavior, in which the protocol suggests different values for transmission and pre-transmission (waiting to confirm that the medium is free) windows, for depending in the medium usage:

idle vs congestion.

Examples of uncontrolled MAC protocols include: all protocols based in Carrier Sense Multiple Access (CSMA) as BATIBUS (French Standard NFC 46620), EIBUS (European Installation Bus), EHS, LON, CAN [Rob91], Ethernet [Soc08]. The first four use a simple version of CSMA, CAN uses CSMA Non-destructive Bitwise Arbitration (CSMA-NBA), Ethernet uses CSMA-CD where CD stands for Collision Detection, which allows nodes to sense a collision and stop transmitting. Other versions of CSMA include CSMA-CA (Collision Avoidance), though this version is primarily used in wireless systems. Wireless fieldbuses are now starting to enter in the control arena, as in the case of wirelessHART [wa].

Examples of centralized controlled MAC protocols include: some tokens passing protocols, and master polling. The representatives are: Profibus-PA, WorldFIP [AC98], P-Net, Interbus-S [Int87], ASI. Possible TTP [KG94] and [KAGS05] are the most widely deployed decentralized controlled MAC.

3.2.4 Effects of Available Real-Time Communication Networks on Networked Control

As mentioned on Section 2.1, control networks require a physical implementation, that nowadays, due to the development of certain key technologies, is accomplished by the use of real-time communication networks.

The real-time communication networks have an enormous effect on the quality of control of the systems that operate under it. Due to the inherent characteristics, such as latency, bit rate, message jitter and associated synchronization metrics, minimum and maximum message size and packet drop rate and the presence/lack of of an associated packet acknowledgement.

Regarding the latency (and the sampling to actuation delay, which are normally lumped into the average dead-time of the system), it is low varying by nature, since the fast varying component is accounted for in the jitter. For this reason, it is possible to compensate for it at the control level. As for the bit rate, higher bit rates, can in principle, allow for control using smaller sampling times. Nonetheless, will be discussed in this Thesis a mechanism that decouples the fieldbus sampling rate from the control sampling rate that is effective if the system dynamics is well known.

The jitter can have a nefarious effect on the quality of control and it is relatively difficult to compensate for at the control level. For this reason, some real-time communication networks have an isochronous mode, i.e., low jitter, as will be presented in this section. Nevertheless, it is possible to ease the effects of jitter by trading latency for jitter, i.e., use a synchronization point at which it is guaranteed, with a given probability, that the message will have been received.

Current real-time communication networks, i.e., the Ethernet based ones, have minimum message sizes that are larger than the size of an average control message. The extra *space* can be used to send data that normally would not be sent.

The packet drop rate and the message reception acknowledgement are two fundamental aspects of lossy networks that, as will be discussed in Thesis, can be essential to optimally control this type of networks.

3.2.5 Available Real-Time Communication Networks

In this subsection, a summary description of some available fieldbuses is made. Among the many types of fieldbuses that exist, were chosen those that either are representative of their class of fieldbuses, thus are a landmark in understanding fieldbuses, or have gained a significant acceptance among the industrial networking community. Given these criteria there are presented at Appendix A the following protocols: CAN, TT-CAN, FTT-CAN, CANopen, TT-Ethernet, Ethercat, TTP, Ethernet POWERLINK, FlexRay, Profinet and WorldFip.

Table 3.1 summarizes the most relevant features of such fieldbuses.

3.3 (m,k)-firm Systems

Nowadays, the use of electronic devices is widespread into virtually all sectors of modern life. Moreover, virtually all such electronic devices can have more than one task, rendering the scheduler a paramount piece of the software puzzle.

Most such devices model their tasks in a restrictive manner, allowing only for the existence of the two types of task, i.e., hard and soft. However, as will be seen in Section 3.3.1, there are situations in which there is a need for tasks that are neither hard nor soft, but can tolerate the miss of a number job executions.

Remembering that a scheduler is said to be optimal, within a given class, if for every task set that can be scheduled by an algorithm that belongs to this class, can also be scheduled by the scheduler in question. In particular, an (m, k)–firm scheduler is said to be optimal if it can schedule all feasible (m, k)–firm systems. However, in (m, k)–firm systems there is also a drive to maximize the utilization of the system, either on a fairly manner or not. This fact introduces a new optimality constraint, namely that the optimal scheduler must also achieve the highest possible utilization.

The classification of (m, k)–firm schedulers that is employed in this Thesis was introduced in [JYQ06] and the prime reason for choosing this classification is its relevance for the correct understanding of the interplay between (m, k)–firm and control systems. The classification in question is done according to the system determinism and they are classified into deterministic or classical — which are explored in this Thesis — and probabilistic, which are more concerned with probabilistic guarantees. In deterministic (m, k)–firm systems, in every group

Protocol	Bitrate	Jitter	Latency	Cycle	Topologies	Real–Time	Philosophy
CAN	– –	– –	– –	No	Basic	Yes	Peer–to–Peer
TTCAN	– –	±	– –	Yes	Basic	Yes	Master–Slave
FTTCAN	– –	±	– –	Sync/Async	Basic	Yes	Master–Slave
CANopen	– –	–	– –	No	Basic	Yes	Exported Functionalities over Peer–to–Peer
WorldFIP	±	–	–	Yes	Complex	Yes	Producer/distributer /consumer
FlexRay	+	±	±	No	Complex	Yes	Segmented Peer–to–Peer
TTP	+	+	+	Yes	Basic	Yes	Master–Slave
TTE	++	+	+	Yes	Basic	Yes	Master–Slave
EtherCAT	++	–	±	No	Complex	Yes	Master–Slave w\ packet concatenation
Ethernet Powerlink	++	++	++	3 part cycle	Basic	Yes	Master–Slave
PROFINET	++	++	++	3 part cycle	Extra layer of supervision	Yes	Supervised Master-slave

Table 3.1: Most prominent Fieldbuses and their most relevant characteristics.

of k consecutive periodic activations of an (m, k)–firm task there are at least m executions, which is in contrast with probabilistic (m, k)–firm systems, where the ratio between activated and executed jobs, for a large number of activations, must be no smaller than m/k.

A task is said to be in a dynamic failure if on the last set of k periodic job activations, there were less than m job executions. Evidently, under deterministic (m, k)–firm scheduling, tasks are supposed to never be in a dynamic failure.

In the strict sense, static schedulers cannot execute the jobs that they mark as optional, since the execution of these jobs may either jeopardise the completion of mandatory jobs, hence, causing the system to enter a dynamic failure or imply some form of policing to guarantee the timely completion of the mandatory jobs, thereby rendering the overall system a dynamic scheduler. Furthermore, if a given static job that belongs to a repeating cycle is guaranteed to complete without causing any mandatory job to fail, then it is worthwhile to set it as mandatory as well.

3.3.1 Related Work

In [Ram99] it was put forth an implementation which managed to gather a significative momentum in the community. It was first proposed that the scheduler should divide its task's jobs into mandatory and optional ones. Each group of k consecutive jobs should have m mandatory jobs. This was in total contrast with the then established use of distance to failure. Jobs are set as mandatory/optional based on an upper mechanical word, defined as

$$a = \left\lfloor \left\lceil a\frac{m_i}{k_i} \right\rceil \frac{k_i}{m_i} \right\rfloor . \tag{3.15}$$

in which a is the number of the job instance, i.e., τ_i is activated at instant aT_i, $a = 0, 1, 2, \ldots$. Whenever the previous condition is verified the job is set as mandatory, otherwise it is set as optional. For example, for an $(3, 7)$–firm task, the first seven executions, of the right-side of Equation (3.15), in which the first one has $a = 0$, would be equal to [0 2 2 4 4 7 7], hence generating the $(3, 7)$–firm frame [1 0 1 0 1 0 0]. The frames generated by the rule defined in Equation (3.15) are called upper mechanical words and it generates an evenly distributed mandatory activation pattern. In [BC01] this model was expanded by first considering normal (optional) and panic (mandatory) jobs (thus a bimodal scheduler), and then trying to provide a schedule of panic jobs that guarantees that they will always meet their deadlines. However, there approach was not based in upper mechanical words or other mathematical extensions, but rather on the authors intuition of what the repeating frame should be. A contribution for this approach is given in [LC06], in which it is proposed that the mandatory jobs should be assigned in a manner that minimizes a certain metric, dubbed Granularity of Quality of Service-Reward (-reward).

In [QH00] it was discussed the fact that the scheduler presented in [Ram99] introduces a periodic critical instant, of which, one of the instances is at $t = 0$. Thus, the even distribution of mandatory jobs is not necessarily the best one. This led the authors to propose three different algorithms for finding a suitable schedule for the mandatory tasks. These algorithms differ primarily in their computational efficiency. The first resembles a test of all possible schedules, the second is a genetic algorithm that at each round tests the task set for schedulability, the third one introduces a metric that drives the search parameters of the genetic algorithm. Notwithstanding the exhaustive nature of the proposed algorithm, in general, it fails to reach the optimal schedule because, as will be shown in this Thesis, the

search space of the algorithm is smaller than the solution space. Furthermore, the *solutions* that the paper proposes are still haunted by the problem that it intended to solve, namely, in many situations, the proposed scheduler introduces the same type of critical instants that were introduced by the upper mechanical words of [Ram99], though it introduces them in different instants.

A different type of static pattern was presented in [KS95], the so called Deeply Red (or R-) pattern and is defined by setting all jobs that verify

$$j_i \bmod k_i < m_i \tag{3.16}$$

as mandatory and setting the remaining ones as optional, where j_i is a non-negative integer that is incremented by one whenever a job from the task in question is released. Note that if $j_i = 0$ at the release of the first job of the task, then the generated static sequence will consist of m_i mandatory jobs in the beginning of the cycle, obviously, followed by $k_i - m_i$ optional jobs. For example, for a $(3, 7)$–firm task that is initialized with $j = 0$, then the sequence would be [1 1 1 0 0 0 0]. If $j_i \neq 0$ at the release of the first job of the task, then there will be a phase, equal to j_i on the previously defined sequence. Thus, for a certain extent, the parameter j can be used to introduce a relation of phases between tasks, but it is absolutely impossible to escape from a critical instant if there is no special relation among the task's periods.

It was proved in [NQ06] that if a given (m, k)–firm task set is schedulable using a Deeply Red pattern then it is schedulable using any other static k jobs sequence with m job activations, which also follows from the above discussion regarding critical instants. However, this is not the same as optimality which implies that if there exists a scheduler that can schedule a given system, then the proposed scheduler can also schedule the system in question. In fact, the proof presented in [NQ06] proves that there is no scheduler, in the set of schedulers that use a static frame with the respective (m, k)–firm properties, that performs worse than the Deeply Red pattern.

In [JYQ06] it is noted this limitation regarding the provisioning of deterministic (m, k)–firm guarantees using static frames of length k with m job activations and it argues that the deterministic approach "*does not contribute to reducing the resource requirements in general*". However, this conclusion is based on 1) static frames introduce critical instants, 2) tasks with implicit deadlines can be scheduled starting from a critical instant if, and only if, the task set is also EDF-schedulable, therefore, 3) all task sets that provide deterministic guarantees are EDF-schedulable. However, at the core of such line of reasoning is the assumption that the results of [QH00, Ram99] and other contemporary (m, k)–firm proposals are optimal. But, as will be shown in this chapter, contemporary schedulers are suboptimal in such a manner that they introduce critical instants, that in turn validate the remarks made in [JYQ06]. However, the generalization that is made in the paper in question is not warranted.

Regarding the implementation end of the field, consider what (to the best of the author's knowledge) was one of the first papers to tackle the problem of (m, k)–firm scheduling and is reported in [HR95], in which it was presented a scheme in which messages were supposed to achieve probabilistic (m, k)–firm guarantees by attributing to each task a *distance to failure* and attribute higher priorities to tasks with lower distances, i.e., closer, to failure. The distance to failure is the number of future consecutive jobs/messages that have to not be executed in order to put the task/message in a dynamical failure. In case of a draw, i.e., several tasks with the same difference to failure, its choice was made based on EDF. This

approach was dubbed Distance Based Priority (). DBP had a number of downturns that stemmed from the fact that the algorithm had no awareness of tasks periods and deadlines, nor the relationships among them. A similar approach was put forth in [Kim10], in which an (m, k)–firm model for wireless networks was presented. The authors introduced some aspects relevant to wireless networks such as the distance to the sink (signal strength). However, in general, it inherited all the issues of its predecessor. Furthermore, it only minimizes the probability of a failure, thereby not providing any guarantee that (m, k)–firm constraints will always be met, i.e., it is a probabilistic (m, k)–firm scheduler. Such probabilistic approach was first proposed in [KKH04].

Other variants of DBP include Integrated DBP () [ZWxS04], which used not only the distance to failure but also the distance to recovery upon failure, that is not very effective at avoiding a system's entrance into dynamic failure, but significantly reduces the *domino effect* in which, as in EDF, the failure in one job/task is propagated into other jobs/tasks. Matrix-DBP ([CSWS04], also uses the distance to failure to schedule message streams, but it differs at a fundamental level from other variants since, by virtue of its intended purpose, it cannot be preemptive. It employed a rather large set of other variables such as periods, (execution) service time and relationship between streams. An expanded version of the matrix-DBP into the , which allows for non-periodical messages was presented in [CSWS04]. Also [LY08] improves upon DBP introducing a Total BDP () which introduces a *total distance* to failure that include other relevant parameters, such as distance to enter invalidation (dynamic failure) and to exit it, which in much resembles the goals of IDBP [ZWxS04].

In [KC11] it is presented a multiprocessor implementation of a protocol of the DBP family. An analysis of its probability of being in a dynamic failure is also presented. In [LK12] it is proposed a protocol called Local DBP () which has the goal of providing (m, k)–firm guarantees to multimedia streams over wireless sensor networks. It works by augmenting the distance of DBP with local node information regarding link congestion and failure rates.

In [Eve10] it is discussed the possibility of performing an online schedulability test that admits optional jobs if and only if it can be assured that they will not cause a mandatory job to miss a deadline. This is further enforced by putting the jobs into two different execution queues, a high priority queue for mandatory jobs and a low priority one for optional jobs, with jobs from the low priority queue executing if and only if the high priority queue is empty. The proposed schedulability test is not tight, in the sense that it is possible that an optional job can execute without causing any deadline miss of a mandatory task but it gets declined by the scheduler. This phenomenon is sometimes called a false negative. The approach presented in this Thesis also uses an online schedulability test. However, the proposed approach is less restrictive, in the sense that optional jobs are allowed to interfere with mandatory ones for as long as they do not cause a deadline miss.

In [CCP10] it is presented the approach that is most similar to the one presented herein. The authors use both the time to failure approach (DBP) and a deadline awareness that mitigate some of the issues related to the original DBP. Two algorithms are presented, the guaranteed on-line scheduling algorithm () being the most powerful, the other being a lightweight version with a lower computational demand. In GDPA it is decided to put a given job in the execution queue based on a local (temporal) state of the queue. Thus it can put a job that has a long distance to failure in the queue (as long as there are no jobs of tasks with lower distances at the moment) while in the following scheduling instant a job from a task close to a dynamic failure is released, which cannot be executed because the queue is already full. An example of this effect is explored in-depth in this Thesis.

Applications of (m, k)–firm scheduling in control are presented, for example, in [Ram97] in which the fact that many control applications are resilient to a small number of failures is used to reduce the resource usage. Controller modifications are also discussed. In [LSSL06] the graceful QoS degradation provided by (m, k)–firm scheduling is explored. More concretely, DBP is analyzed and sufficient, though not tight, conditions for deterministic schedulability and their applications for control purposes were proposed. [FNSLY08] tries to devise an optimal (m, k)–firm frame (repeating pattern) for control purposes. However, the approach presented therein did not made any consideration regarding the schedulability of the resulting (m, k)–firm sequence, which could potentially invalidate the underlying assumptions. In fact, this is a general issue regarding the utilization of contemporary (m, k)–firm schedulers for control, i.e., they do not guarantee the schedulability of the respective (m, k)–firm tasks, which in turn causes a series of obvious problems. Furthermore, the authors used a rather short integration time in their ISE computation, thus sequences that had mandatory tasks in the beginning of the meta-frame ended up having lower ISE, even among sequences that were rotations of one another.

Moreover, the actuation scheme that was used in the control related contributions share a common problem, namely, though the schedulers assume that optional jobs may or may not be executed, the controllers are designed assuming that optional jobs will never be executed. The main problem caused by this assumption disparity is that upon the reception of an optional message, the controller will assume the reception of a mandatory message in its respective time, thus generating the corresponding control value, which obviously will be erroneous. Additionally, upon the reception of the next mandatory message, the controller will assume that it is receiving the mandatory message that would follow that one, and so on. To the best of the author's knowledge, the only controller that does not have this problem is [CeSP11], which was proposed by the authors of this Thesis and presented in Chapter 4 and Chapter 5, and the main reason why this controller does not have this problem is that it uses sequence numbers that allow it to find out how many messages/jobs have passed since the last one was served.

In [WJ08] it is presented a study of the application of (m, k)–firm schedulers in multimedia settings, with a special focus on the DBP scheduler. Multimedia is one of the areas in which (m, k)–firm scheduling has the potential to provide considerable improvements, as can be attested in [WJ08] and references therein. For example, for the JPEG video case, there are two types of frames. The most important type (key frames), used to decode the other less important frames, must always be sent, which conflicts with many dynamic schedulers, because dynamic schedulers do not provide guarantees regarding which job will be executed. For example, DBP only cares about the distance to failure, completely disregarding the importance of the associated frame.

3.4 Feedback Scheduling

The co-design of real-time systems and networks with their respective applications, which usually is control, led the various applications that have time varying computational demands. For example, if the controllers are allowed to change their sampling periods, then it would be reasonable that the minimum inter-arrival time of their respective tasks are also allowed to change. Moreover, there is a need to choose the parameters of such tasks in a manner that optimizes, or that at least provides some guarantees, regarding the behavior of the application.

This fact stimulated the development of schedulers with the aforementioned characteristics and they are called feedback schedulers and are roughly defined as any scheduler that consider the current state of the system on its scheduling decisions.

In [LSST02] it is presented an extensive discussion on the application of control theoretical approaches to feedback scheduling. Their approach use the transfer function approach to devise feedback controllers with zero static error, i.e., the actual task utilization is equal to the user specified value, and with some requirements regarding the transitory behaviour. Nevertheless, the approach is applied to only one task at a time, with little regard to the fact that the sum of such utilizations is equal to a constant, i.e., one. This approach is applied for the dynamic scheduling of multiprocessor systems in [SCH11]. This framework is also in [YZL+12] for scheduling a systems that change their operating frequency, hence its effective utilization, in order to deal with processors' thermal fluctuations. The framework in question is used in the scheduling the inner loop, i.e., the actual tasks parameters.

In [APLW02] the authors explored the concept of reservation applied to feedback server scheduling. In this context, each server would reserve a certain amount of time, i.e., a period and maximum utilization that it will require on such amount of time. The scheduler uses a control theory based approach to schedule the various servers. [Lin07] presents an extensive survey of reservation based feedback server scheduling. However, it must also be noticed that this has not been an active area of research on the last years.

In [SLB08] it is considered the possibility of dynamically changing requirements, i.e., period and capacity, of servers of static priority processes in a feedback scheduling scenario. This work, though from the same authors, differs from the past because it allows for online scheduling. Requests that reduce the utilization of servers are handled seamlessly. However, requests to increase the requirements are handled by first performing a schedulability test to ensure that of the new task. If the new task set is schedulable, then the request is processed, otherwise the server in question is given the highest utilization that is schedulable. The paper proposes five different schedulability test algorithms with different degrees of tightness and complexity.

In [BLCA02] it is presented the elastic model, which is also a type of feedback scheduler. The elastic model is based on an analogy between with of strained springs and a series of task weights. Each task has an individual weight, such weights can be set according to several criteria. Moreover, the weights are normally used to control a task's relative level of adaptivity. However, the exact criteria is not relevant to the elastic model. The analogy that drives the elastic model is the fact that in a composite spring the sum of the displacements is equal to the total displacement and the displacement of each string is inversely proportional to its elastic coefficient. [BLCA02] is used to ensure that a task set never exceeds a given utilization whilst giving more resources to tasks with higher weights.

In [BGSS08] it is presented the co-design of multiple control loops into a single CPU from a non-linear programming standpoint. It assumes an infinite horizon quadratic cost function which is approximated by a constant term, equal to the continuous time cost matrix, plus a term that is quadratic on the period. A global cost function is defined as being a weighted sum of the individual cost functions. The global cost function is solved using the Karush-Kuhn-Tucker conditions, see [BV04], which are used to solve non-linear optimization problems. Even though the rationale of the paper is sound, it is riddled with mathematical inconsistencies that greatly undermine its usefulness. In [EHA00] it is proposed a similar approach, also using Karush-Kuhn-Tucker conditions, but using somewhat more conventional mathematical tools, and arrived at results that, according to the simulations performed, were better. The work

in [EHA00] was improved in [CEBÅ02] by allowing the control tasks to inform the schedule of mode changes, which allows the scheduler to better react to such situations. [CEBÅ02] also showed that these various approach are equivalent to the elastic model [BLCA02], up to the elastic coefficients. In [HC05] the authors applied the equations in question to a first and second order systems and it became evident that the scheduling is state dependent. However, the issues that appear upon dynamic change of task parameters, which are exacerbated by the fact that this change is state dependent, are not addressed.

In [RS00] it is considered the co-design of a type of (m, k)–firm control system in which all tasks have a utilization of one, when scheduled for execution. It is also assumed that the (m, k)–firm sequences are static, i.e., periodic. The authors find the optimum for this particular case by considering the global cost function, defined almost as in [BGSS08], over all possible (m, k)–firm sequences. In [LB02] it was proposed the use of pruning to reduce the search space of the algorithm.

In [MFFR02] it is discussed whether it is better to have a constant controller to actuator delay or to have a constant control period, as measured by the instant in which the controller is launched. The analysis yielded the obvious result that the former is best. The authors, then take the results used to establish their conclusion and propose a new metric for Quality of Control, which uses a combination of Integral of Absolute Errors with the errors defined in a variety of ways. This metric is used to define the Quality of Control of the overall system, given the period assigned to each task. It is also proposed a sketch of a mechanism to switch task's properties without causing a transient of deadline misses.

A relatively different type of feedback scheduling is the so called Real-Time Physical Scheduling and is discussed, for example, in [DVF12] and in references therein. This type of schedulers differs from classical schedulers because what is scheduled are the physical variables themselves, i.e., it is primarily used in systems in which only one variable can be controlled at a time and the scheduling regards the variable that should be controlled. For example, [DVF12] considers the case of two freezers in which there is enough power to turn on only one freezer at a time. This type of scheduling inherently uses feedback (though it is possible to change it in order to work in open loop) because it makes scheduling decision based on the state of each system.

Feedback scheduling is not the only approach to the problem of integrating control tasks into a real-time system. The most prominent of such other approaches is Event-driven control. In event-driven control, the next control instant, i.e., $t + h$, is decided based on the equation

$$x'(t+h)\mathbf{M}_1 x(t+h) = \eta x'(t)\mathbf{M}_2 x(t), \tag{3.17}$$

where $x(t)$ and $x(t+h)$ are the current and future states, and $\mathbf{M}_1$ and $\mathbf{M}_2$ are two n-by-n (semi-)positive definite matrices that define a cost like value for having a given error at a certain instant, and η is a scalar. These three variables are used as design parameters of the event-driven control. Usually, η is chosen last and it is given a value that ensures that h has a mean equal to a certain predefined value. Having such a mean provides some guarantees regarding the schedulability of the system. Nevertheless, event-driven control has been haunted by a lack of interest on issues relating to schedulability. [LMV10] is a relatively recent article that surveys contribution on event-driven control and makes a comparison with feedback scheduling.

Co-design is not limited to the control systems and real-time/control networks co-design case. For example, [CPA+10] presents a framework for the integration of application level

Quality-of-Service requirements, i.e., execution time, period, deadline miss ratio, with resource level Quality-of-Service requirements, such as power requirements, inter-activation times, resource speed, etc.

Chapter 4

Network Decoupled Control Architecture

4.1 Introduction

In recent years networked and related electronic systems have experienced a rapid development that rendered the systems suitable for control purposes [JJ07, NP04]. In spite of such rapid development, which facilitated many operations, some issues remain far from being solved, such as the (optimal) assignment of network traffic characteristics, such as periods (which are application dependent), phases (which may be interdependent), and other schedule related issues. Hence, under certain circumstances, some messages may not be delivered to their destinations at the desired time instants, e.g. because a message has a low priority and was superseded, or simply due to a network error. When this happens, the overall effect is the deterioration of the control performance.

Nowadays, it is common practice for Distributed Control Systems (DCS) to be deployed using a fieldbus at the communication layer. Under certain circumstances such networks (fieldbuses) may introduce errors that prevent all messages from being properly delivered, causing a plethora of control issues. This chapter presents a control architecture that aims to increase the tolerance of the control system to network errors. Summarizing, the increased tolerance is achieved through a series of data buffering and future state predictions, the specific details of the technique will be presented in this chapter.

From the control standpoint, it would be desirable that all such issues could be masked in a manner that allowed a presentation of the system that was not entangled with the peculiarities of the underlying network. There already exists a vast body of work that tries to solve this problem, as can be seen in, for example, the survey [HNX07] and references therein. It is the author's opinion that the most promising approach appears to be the introduction of a network-aware layer in the controller design and vice-versa.

This chapter proposes one possible realization of such a layer, which is implemented by an architecture that allows the controller to have a more transparent view of the network, thereby allowing the controller to make a better use of the available network resources.

The architecture that is presented in this chapter will be used as a framework in subsequent chapters. This framework will allow to understand the ideas that are presented in this Thesis. Furthermore, its use is instrumental to establish the Thesis. Nevertheless, most of the final results of the Thesis can be ported into other architectures.

4.2 Architecture Rationale

It has been shown that networks can be made somewhat more resilient to external disturbances provided that a suitable error correction mechanism is in place. In fact, Shannon [Sha48] proved that the maximum error-free transmission bandwidth is a function of the communication medium bandwidth and signal-to-noise ratio.

The scheme presented here tries to achieve a bandwidth-noise tradeoff. For example, in systems with no perturbations, once the model and its respective state variable at a given time instant are known, a controller can, in principle, compute all future states and control actions in advance. Therefore, only one message — long but of finite duration due to the finiteness of the control duration — would need to be sent, hence leading to a smaller transmission rate.

From a network perspective, it is a well known fact that the number of nodes and the bandwidth requirements of each node have been increasing steadily in the past decades. Hence, if it continued to increase without any form of intervention on other parts of the network, then it would lead to a saturation of the channel capacity of the existing fieldbuses. A solution that was employed several times, is to increase the bandwidth of the network. Another one, and the one that is followed in this Thesis, is to trade bandwidth for computation, as is done, for example, in network coding. The scheme presented here has the potential to do so efficiently. The rationale is simple: as the average computational capabilities of network nodes increases, nodes are starting to have spare computational capacity while experiencing network congestion. Such scheme could either be used to increase the number of nodes, or to increase the bandwidth available to each connection.

As control perturbations increase, the transmission rate must also increase in order to guarantee a small output error. It should be stressed that what is being increased is the message transmission rate, not the control execution rate. The former is determined based on the conditions of the output (error) and the maximum bit rate reserved for control. The latter is determined by the dynamic of the system, using some empiric rule [Oga92, KJA97, CHH85]. This has two implications, namely, 1) the messages that are exchanged encode regularly spaced samples/control values. Nevertheless, the messages neither need to be exchanged with the same period nor need to be exchanged regularly. In fact, if for example, a given system has a high output error, its associated sampler/controller could send more messages with the same internal period, thereby reducing the number of times that the actuator applies a prediction, but preserving the internal sampling/control period. This is in line with the assumption that the transmission cost of sending one message is almost independent of the size of the message, at least for small messages in large packets — typical in Digital Control Systems, but such cost is very sensitive to the number of sent messages. 2) this scheme does not require that sampling and actuation are performed at the same rate, which can be an advantage in applications in which, for example, sampling is less expensive than control.

The size of the messages introduces a new design challenge, i.e., the number of sensor/controller values that are sent in each message. From the controller point of view, it is pointless to send a prediction of a future control value if between the present and the future point in question the controller will (successfully) send another message. From the sampler point of view, it is pointless to send a past output value if it is either known that the controller has already received it or if the reception of the present values are already enough to have a decent estimate. These observations can be turned into conditions regarding the number of messages that go in one packet.

A related and equally important aspect is to decide whether all packets messages will

contain the same number of messages or if the the number of messages is a function of the expected instant at which a new packet will be sent. However, following the above discussion regarding the assumptions of bandwidth cost, it is pointless to change the number of messages online, since it does not have any effect on the cost.

Since the sent packets may contain many sets of messages/values, receivers of such packets must be capable of discovering which value corresponds to which time instants. The receiver could potentially attain such knowledge by counting the number of periods that it has not received a new message. However, due to clock drifts, the difference in periods reported by the various units is not guaranteed to be equal. Hence, in this work, the sender includes a sequence number that allows the receiver to find out the number of messages that it did not receive.

4.3 Description of the Architecture

The control systems presented here have the usual distributed control structure, being comprised of sampler, controller and actuator nodes, see Figure 4.1. As the name states, sampler nodes sample a number of physical variables and send the values to the controller. The controller receives a number of sampled variables and computes a series of control outputs, that are then sent to the actuators. Actuators receive messages containing the desired values for physical variables, and apply them to the physical world. Figure 4.1 depicts a control system with the proposed architecture.

4.3.1 Sampler

The sampler is a simple system, akin to the one proposed in [ESTM08], though there is a big semantic difference associated with the values stored in the buffer. The sampler stores in a circular buffer the N last sampled values. The buffer is obviously updated whenever there is new data. Whenever this happens, the sensor sends the buffer contents plus a sequence number. The sequence number tells the receivers (controllers) if any messages were lost, and if so, how many. If less than N consecutive messages are lost, then no information is lost, though part of it is received late.

Messages containing all such values are sent in batches and an algorithm in the controller node produces the best estimate given the received messages.

Due to overhead, sending an extra value in a message requires less bits than sending the same value in a different message. Hence, sending bulk values into smaller number of messages implies a lower utilization of network resources. However, it must be pointed out that this

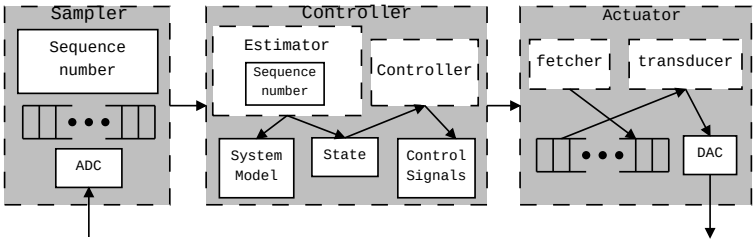


Figure 4.1: Architecture Diagram

strategy provide smaller gains in networks that have small maximum allowed payload, e.g. CAN [Rob91].

4.3.2 Controller

Many application domains are cost-sensitive, thus the processing power tends to be as low as possible. Hence, the controller is most likely the only node with relatively high processing power. For this reason, for as much as possible, most of the *heavy lifting* is done by controllers.

The controllers proposed in this paper are composed of two subunits or tasks, that are executed with a certain degree of independence.

1. Whenever a message is not received, the estimation is based on the previous estimate and the previous control signal, i.e., one step prediction. When a new message is received the states from the instant in which the previous message was received up to the present are recomputed, based on the actual sampled values, through a process explained in the next two paragraphs. Thus, assuming that the buffers are long enough to avert a data loss, all estimates built in the absence of samples are discarded, and in the long run all estimates are made based on actual samples. This has two advantages: i) at any time there is the best possible estimation (even when the best estimation is not based on a sample), and ii) the state estimation error covariance matrix is certainly bounded, and from time to time equal to the lowest possible, i.e., the state estimation error covariance matrix of the case in which all samples were received.

   The first part of the controller interacts with the samplers. The controller has a sequence number of its own, that represents the last sensor message that was successfully received. Whenever a new message is received, its sequence number is compared to the sequence number of the controller. If it is the next number in sequence then no message has been lost. In general, if the difference is n, $n > 1$, then the total number of lost messages is $n - 1$. After receiving a message, the controller sets its sequence number to the last received message.

   Knowing the number of lost messages, the controller extracts this same number of past input values from the current message (buffer), plus the current sample. Thus, the controller, with the information of its previously computed control signals, reconstructs the state trajectory of the system, using the estimation algorithm that would have been used in ordinary, i.e., without this mechanism, systems. It should be remarked that, in doing, so the estimates computed without the sampled values are discarded.

   To summarize, this task produces an estimate of the system state, and if necessary, a system identification is performed. This task is executed whenever there is a new message from a sensor.

2. The other task running in the controller is responsible for computing control values and sending them to the respective actuators. To this end, the controller generates a series of M control values, one for each of the next M actuation instants. The first one is computed directly from the state estimation produced by the first task. Subsequent control values are computed assuming the estimated state given the computed control signal, and the control law.

   After having this set of control signal, the controller sends them, plus a sequence number, in a single message to the respective actuators. Once again, sending larger messages

may have implications that were discussed in the sampler description.

This second task is both self-activated or activated by the first task. It is activated by the estimator when the estimator reaches the conclusion that the states estimated by the controller are considerable different from those reported by the sampler. It activates itself if it remains inactive for a constant number of periods after its last activation.

An SOD-like system was implemented, in which the sensor and the controller have different threshold values. Whenever the sampled value is outside a given range of the last sent sample value or a given amount of time has passed, a new message is sent. Similarly, when the control signals estimated in the last message differ significantly from the control signals computed given the actually state trajectory, or a given amount of time has passed, the controller sends a new control message. The results obtained are presented in Section 4.5.

4.3.3 Actuator

The actuator *translates* the latest available value into the physical world. The actuator is synchronous, changing its value at well defined time instants. However, sending the actuation instants alongside the control values, may or may not invalidate the assumptions regarding the cost of sending message with aggregated values. If it does violate the assumptions in question, then its better to use a periodic actuator, otherwise, the choice should be made according to control requirements.

Note that the actuator can apply the values periodically, or with any other pattern, even when it is not receiving the control messages in the same pattern because it already possesses the capacity to store future actuation values. Naturally, this can be used to reduce the impact of communication jitter.

Whenever a new message is received, the next output value is set to the first value of the array. If the time to apply the next value is reached before a new message is received, the second value of the array is applied instead. Obviously, this assumes that the messages that have not arrived until a given moment were lost, as opposed, for example, to messages arriving out of order. Whenever all values of the packet have been applied, the actuator keeps applying either zero or the last value. However, this situation will occur with a very low probability, since the array size is/can be chosen in order to avoid it.

4.4 Related Work

This work encompasses many aspects that can be found in the control/network literature, namely:

- **The ability to change the control rate at run time**. One such approach was presented in [APM05], which suggested the design of several controllers, each one for a different sampling rate, and the use of a selection mechanism responsible to switch the controller (and corresponding sampling rate) when certain error criteria were met (or not met) as discussed on Section 2.3.3. In [VMF+10] controller rate adaptation was also discussed, but with a focus on the optimization of the use of centralized resources, e.g. CPU. A different approach is presented in [AT09]. There it is proposed that the rate should be proportional to the quadratic error and to a constant given to each process, in

order to enforce priorities. A similar approach had been proposed before, see [MLV+08] and in references therein.

These approaches may cause slight oscillations, due to the fact that the controller task is a) given a shorter period when it has large errors, then b) using a smaller period lowers the error, eventually, triggering a switch to c) a slower rate controller, in which the error may increase leaving the system in state d) of large error and slow period, causing another commutation into a), so on and so forth. This problem does not occur in systems with properly designed static schedulers or in systems that use a schedule that minimizes the expected error over a large run-time. In order to avoid such oscillations, the periods of the tasks must be changed based on a low-pass filtered version of the error, not its instantaneous value.

- **The use of buffers for improving state estimation in lossy networks**. In [ESTM08], it is shown that a buffer in the sampler can be used to improve state estimation, thereby allowing the controller to know past output values that were dropped by the network. In the same paper, the probability of the estimation error covariance matrix being smaller than a given matrix is explored in detail.

  The simpler systems deal with missing actuation values by either holding their outputs or by outputting zero in the absence of new control messages. Systems in which the controller can somehow compute a new value whenever a sensor message is not present, represent a logical evolution.

  From the perspective of controller estimation, a scheme to compensate for networks delays was presented in [ZBP01]. There it is suggested that the controller should predict the state of the system at the control instant, and then apply a control law based in the predicted state. A similar approach was studied in [LMVF08a], but with the prediction of a future (network delayed) state and the application of the control law. Less sophisticated approaches include the use of basic interpolators to achieve the same goal.

Another contribution that relates to the approach presented in this chapter is the (SOD) [TB66], [OMT02], in which data is sent if either a significant change has occurred or a given amount of time has passed since the last message was sent (*I am alive* message). In [YTS02] it is argued that computation can be exchanged with bandwidth, in the sense that the number of messages exchanged in the network can be reduced if more complex control techniques are used. An example of a system with an estimator/controller capable of such trade-off is provided. A more comprehensive review of related contribution on the literature was presented on Chapter 2.3.

One particular contribution stands out, when comparing the existing literature of the subject with the present chapter, the work in [Sch08] in which there is considered the case of a variable communication delay between a sensor and some other node that is building an image of the system. The sensors always send their messages at the beginning a sampling period. But, the network delays can cause the respective packet to arrive at a different sampling period. The authors of [Sch08] propose to use the aforementioned buffer, which is assumed to be infinite, at the receiver. The receiver would store the estimates and the respective error covariance matrix of the sample corresponding to the maximum delay related to the size of the buffer. The sensor used time stamps to allows the receiver to reorder the samples. The estimates are computed using the common modified Kalman filter, which

handles missing data points by only performing the innovation step at the referred sample periods. The authors also consider the possibility of the sensor performing the estimation and send the estimates instead of the samples, whereas the receiver would perform a projection of the state whenever no new message is received and simple copy the value at a message into its state, whenever a new message is received. This mechanism is, in many ways, similar to the sensor controller communication part of the architecture discussed in this chapter. The main difference is on the placement of the buffer, since in the approach proposed here, the buffer is at the sensor side which allows to a single sample to be sent more than once, hence increasing (redundancy) the probability of being received, whereas in the approach put forth in [Sch08], each sample is sent once, another advantage is related with the efficient use of the network resources, as pointed out above.

4.5 Performance Assessment

To perform an experimental validation of the proposed architecture — in this case a verification of the reduction of used network resources — two systems were simulated, in the Matlab® numerical computation environment. TrueTime was used to implement the network (Ethernet) and kernels.

The simulated plant had the following continuous-time state-space representation:

$$S1 : \dot{x} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u, \quad y = [0 \quad 1]\, x + v \tag{4.1}$$

which has poles (in the s-plane) at -1 and 0. The system had a sampling (actuation) period of 10 ms. Pole-placement was used to put a closed-loop double pole at 0.98 (z-plane). The input of the system was perturbed with white noise, for six equidistant RMS points in the range $0 - 10$.

Two groups of experiments were performed for the output noise. In the first group the output noise was absent, and in second it was present and it simulated the type of noise generated by a 10 bit Analog to Digital Converter (ADC) with no other non-idealities (10 bit due to its high availability in contemporary micro-controllers).

Each experiment was performed for six transmission threshold values. The threshold was the value of the error variable that triggered the transmission of a new message. The threshold values of the sampler and the controller were set proportionally, i.e., $Th_c = k * Th_s$, where Th_c is the threshold in the controller, Th_s is the threshold in sensor and k is a constant factor. A value for the *gain* k that gave a good relation between used bandwidth and the Integral Squared-Error (ISE) was found by trial-and-error, being 0.1, and such value was used in all simulations. Due to the random nature of the experiments, each experiment was performed 160 times.

Figure 4.2 depicts the variation of the ISE as a function of the SOD threshold value for several input noise levels (the ISE graphs with output noise are similar to the ones presented here, thus, they were not included). The of the controller is to have the lowest ISE possible. From this figure it follows that, using this control strategy, the greatest contribution to the ISE comes from the SOD threshold, and that the input and output noise levels have a very small effect on the ISE. Therefore, the threshold values can be used reliable to set a desired quality of control.

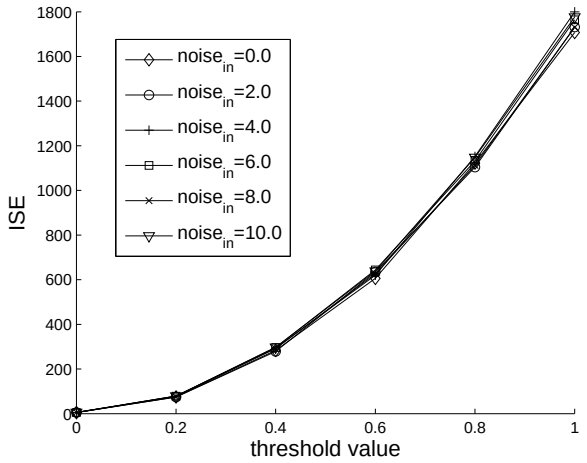


Figure 4.2: ISE versus threshold for several noise values

The next simulation results show the influence of the noise level and threshold on the amount of bandwidth that is saved by this architecture. Note that in systems without the presented architecture, the bandwidth savings are, by definition, zero.

Figures 4.3 show a series of bandwidth measurements. More precisely, they show the saved bandwidth in percentage, which is computed as:

$$sb = 100 \times \frac{b_{\text{standard}} - b_{\text{used}}}{b_{\text{standard}}}, \tag{4.2}$$

where b_{used} is the bandwidth that was measured in the simulation and b_{standard} is the bandwidth used by a standard controller, under similar systems and networks.

The first set of graphs (up-left) shows that the saved bandwidth tends to 87.5% when the threshold values grow unboundedly. This happens because the messages are scheduled to be transmitted with a periodicity of N (as described before), even when the thresholds are not reached. Thus, when the threshold is significantly larger than the error in the signals, 1 out of N messages is sent. Therefore, the saved bandwidth tends to $100 \times (N-1)/N$ percent. In this case, $N = 8$ (thus converging to 87.5%). Since Ethernet was used, a message with 8 blocks of 2 bytes is well within the maximum payload of 1500 bytes, even when compared with the minimum payload, which is of 46 bytes.

On a similar experiment (up-left) done with ADC reading error, the transmission rate was limited by the ADC noise, i.e, its effects were enough to trigger a transmission (sooner than the effects of the input noise alone). Hence, the saved bandwidth was less sensitive to the input noise.

The controller actuator transmission without ADC reading errors — top right — is similar to the sensor-actuator counterpart. However, in the presence of ADC reading errors — bottom left — there is no significant qualitative difference, mostly because those errors got filtered out due to the higher rate at which the sensor send its messages. The bottom right figure depicts the controller to actuator case, in the presence of ADC noise.

Not all messages sent from the sensor to controller triggered a message from the controller to the actuator. If the controller notes that the new messages do not change significantly its state estimate, and consequently its control outputs, no new messages will be sent. That is the primarily reason why the two rightmost graphs of Figure 4.3 are rather similar.

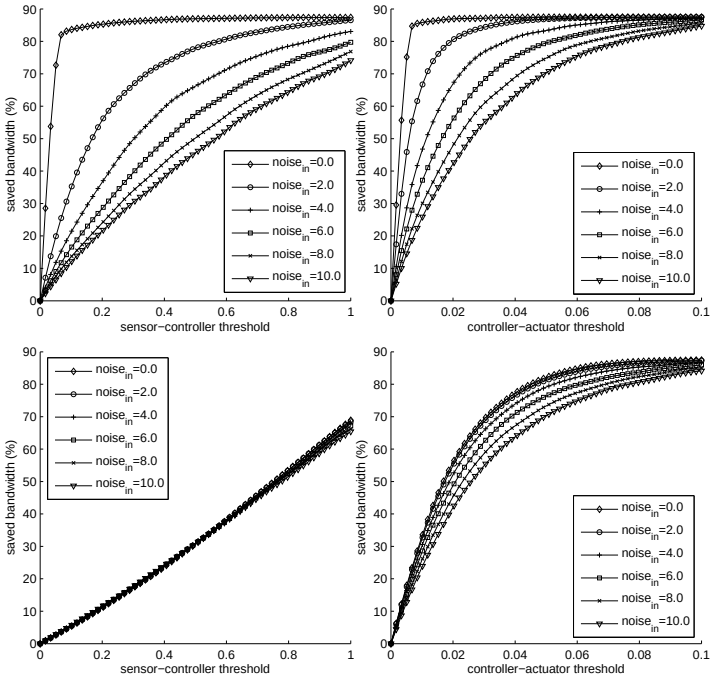


Figure 4.3: Threshold vs saved channel capacity for several noise values. Up in the absence of ADC noise. Down in the presence of ADC noise. Left sensor to controller. Right controller to actuator.

Note that the curves are converging to the value mentioned above at different rates. This is due to the fact that, under the same threshold value, systems with lower noise transmit less messages. In the extreme case of no noise, as depicted in the top curve of the two top graphs in Figure 4.3, the curve converged to a step function, i.e., it sends all the messages if the threshold is zero and it only sends 1 message out of N if the threshold is not zero.

4.6 Summary and Conclusion

Networked control systems have a number of non ideal characteristics that stem from the differences between the assumptions made in control and network designs. However, a new trend on control network is emerging, which can be summarized as the use of networks, e.g. Ethernet based fieldbuses, with a large minimum allowed packet size.

These developments shift the type of control architectures that achieve the best possible control bandwidth tradeoff. Hence, a new control architecture was proposed. The new architecture takes advantage of the fact that the cost of sending additional data may have a reduced or no cost, depending on network technology and the amount of additional data, to send extra control (sensor) data, thereby reducing the total number of message exchanges, which results in a overall bandwidth reduction.

The new architecture was tested on a simple system. Results corroborated the initial expectations and showed a clear reduction of used bandwidth, under Ethernet, without a significant ISE increase.

Chapter 5

Control over Lossy Networks

5.1 Introduction

In the previous chapter, it was shown that many networks used in control (in particular Ethernet), have a minimum data length that allows for the transmission of extra control (sensor) data at small or no cost. However, such extra sent data was primarily used for reducing the overall bandwidth requirements, which was achieved by reducing the number of transmitted messages.

Even though during the last four decades the average processing and communication capabilities of the nodes employed in distributed control systems have increased steadily, these systems are frequently deployed in harsh environments, being subject to strong electromagnetic interference, noisy power supplies, mechanical vibration, among other perturbations. Thus, despite all aforementioned advances, communications are still affected by non-negligible error-rates. The move towards wireless media, currently observed in many application areas, exacerbates this problem. Obviously, the performance of control applications supported by this kind of communication infrastructure can be severely deteriorated. Thus, to achieve an adequate quality of control, it is necessary to develop a control framework that takes into account the effects of packet losses in control and estimation.

The main drive behind this chapter is to investigate the possibility of using the type of control architecture developed in the previous chapter to lessen the effects of packet drops. Evidently, this is a problem of estimation and control in networks with packet drops.

Over the last few years, the problem of optimal estimation and control over lossy networks has been studied in detail. However, unlike the approach in this Thesis, most of such works assume that in the absence of a new actuation value, the actuator either applies zero or the previous applied value. Some other extensions have been proposed, but even though they are presented as different approaches to the same problem, they present similar performance.

In this chapter it is argued that these methods are not optimal. Moreover, a new approach in which the controller sends a number of predictions of future actuation values, which are applied when the actuator does not receive a new message, is presented. Under this new approach it is shown that the controller/estimator herein presented performs equal to or better than any previously introduced controller, the equality being verified only in the absence of communication errors. A modified Kalman filter as well as a specific optimal controller for this novel actuation model are also presented. For the sake of simplicity and without prejudice to the assumption of a fully distributed architecture, this chapter assumes that batch

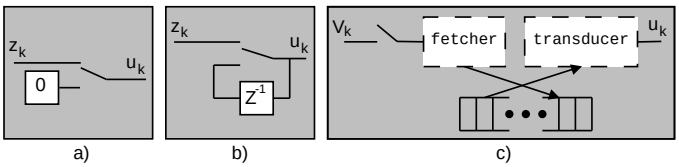


Figure 5.1: Types of output strategies. a) output zero, b) hold, c) proposed approach.

transmissions occurs only between the controller and the actuator. The sensor controller case can be studied either in a similar way or by, for example, applying the control/estimation duality.

Depending on the existence or not of explicit acknowledgment signals, communication protocols are usually classified into TCP-like and UDP-like categories, respectively [IYB06]. Since it is assumed a fully-distributed architecture, under TCP-like protocols the controller receives an acknowledgment from the actuator signaling the correct reception of the last actuation value. Therefore, in TCP like protocols, whenever a state-estimation is performed, the estimator has perfect knowledge of the inputs of the system. However, under UDP-like protocols, there are no such acknowledgments, thus the estimator cannot know which input was used in the previous actuation instant.

Such lack of information, under UDP-like protocols, regarding the correct reception of messages by the actuator renders the estimation error covariance matrix a function of the input, thus leading to a complex (i.e., non-quadratic) *optimization problem*, which has no known optimal solution. A more in depth discussion of these type of problems can be found, for example, in [SSF+07]. The use of TCP-like protocols suffers less from this ailments. However, it requires acknowledgments that, *a*) are also sent over a lossy network, and therefore are themselves subject to errors *b*) increase the transmission error probability *c*) increase the network load and complexity of the communication stacks.

Note, that the terms TCP-like and UDP-like do not refer to the protocols that are used in the Internet Protocol Stack (IP), i.e., OSI level 4. Instead, they are simply used to denote protocols whether it is possible to the sender to know if the message was correctly transmitted.

The literature presents two main strategies to cope with missing actuation data. In the first one, called output zero, if there is no new message the actuator applies zero, shown in Figure 5.1 a). In the second one, called *hold*, in the absence of a new message the actuator applies the previous value (Figure 5.1 b)). This Thesis argues that the best option is to use an estimation of the control value given the set of previous control and output values (Figure 5.1 c)). This scheme provides a simple and optimal realization. The control wise optimality is proven in this chapter, whereas the bandwidth wise optimality is assumed from the network assumptions.

This chapter also shows that for TCP-like protocols the separation principle holds, i.e., the error covariance matrix is independent of the controller gain, while for UDP-like protocols the separation principle does not hold in general, though, as will be shown, for the class of controllers presented in this chapter, the estimation auto-covariance matrix is independent of the state. Therefore, for the controller presented herein, both for TCP and for UDP-like protocols, the optimal estimator is independent of the optimal controller.

5.2 Related Work

The subject of estimation and control over lossy networks is relatively old. The first appearances of it in the literature try to find a correspondence between the Shannon information theory [Sha48] and control theory. More specifically, they tried to answer the question: *what is the smallest bit-rate necessary to stabilize a given system* or, equivalently, what is the maximum transmission error rate that a network can have that ensures the stability of the system? Furthermore, *which techniques (architecture) achieve this threshold?* is a question that, to the best of the author's knowledge, has not yet been addressed by the scientific community and it is hopped that our work aids in its response. A number of early attempts to answer the first question, i.e., the connection with Shannon's information theory, are presented in the survey [HOV02, HNX07].

In [As03] the authors attempt to extend the classical theory of optimal control [Ber07, KJA97]. However, they conclude that the separation principle, which allows a simplification of the results, does not hold in the general case. However, other simpler results are shown, such as, that a system is stable if the independent and identically distributed (i.i.d.) variables that describe the error rate (α) verify $\alpha < [\max|\lambda(\mathbf{A})|]^{-2}$, assuming that the system is unstable, where $\lambda(\mathbf{X})$ is the set of all eigenvalues of the matrix $\mathbf{X}$ and $\mathbf{A}$ is the discrete-time state-transition matrix. Similar claims are made in [HY07].

In [SSF+07] it is made a more in-depth study of such issues, formally presenting the UDP and TCP-like protocol cases. TCP-like protocols are more malleable to the existing optimal control theory, whereas UDP-like protocols, under the assumptions made therein (output zero) yields a complex optimization problem, since the estimation error covariance matrix at any given time step will depend on previous control values. In [SSF+07] it is also shown that, under their assumptions, TCP-like protocols have an error-rate similar to the one found in the previous papers, but UDP-like protocols have lower error-rate bounds that guarantees stability.

Another related development is given in [HY08], in which state information is sent in more than one packet. This fragmentation is done primarily in an attempt to reduce the impact of packet losses, since loosing a packet with a significant large number of state variables would be worst than loosing one packet with only one variable.

In [KKH+08] it is taken a radically different approach, i.e., passive networks — networks that can be implemented using only passive elements — are used as a mean to provide jitter immunity for the controlled system. However, no reasons to believe that passive networks are more immune to jitter are presented, and no mechanism to design passive controllers to achieve certain metrics, e.g. state-tracking or input tracking under noisy conditions, are provided.

In [GSC08] the authors introduce a new concept, i.e., the packet acknowledgment is also probabilistic. By varying the probability of an acknowledgment, it is possible to go from no acknowledgment (UDP-like protocols) through a grey zone up to the situation in which there is an acknowledgment with probability one (TCP-like protocols). Therefore, in a certain way, they extended the notion of UDP-like versus TCP-like protocols. Even though it is a very active research topic, this extension was not considered in this Thesis because: 1) the author is not aware of any network protocols that support such an extension, 2) it is normally dealt with as in the case of UDP-like protocols, i.e., the separation principle does not hold, hence, classical minimization techniques cannot be applied, therefore, the extension is not actually solved.

In [KF11] the output schemes are extended from the zero versus hold spectrum by allowing the value that is held at the actuator to decay with time, that is,

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}s_k + w_k \tag{5.1a}$$
$$s_k = \mathbf{L}_k r_k \tag{5.1b}$$
$$r_k = \theta_k u_k + (1-\theta_k)\mathbf{M}_k r_{k-1}, \tag{5.1c}$$

in which $\mathrm{s_k}$ is the value that is actually outputted, $\mathrm{r_k}$ is the actuator internal state representation which is allowed to be a matrix, $\mathrm{u_k}$ is such a representation computed at the controller using information from the sensors and it is transmitted in a bulk message, $\mathbf{M}_\mathrm{k}$ is a state transition matrix of the actuator state, and $\mathbf{L}_\mathrm{k}$ is the respective controller gain (actually, in [KF11], $\mathrm{s_k}$ it is assumed to be equal to the first column of $\mathrm{r_k}$ and constraints are placed on $\mathbf{M}_\mathrm{k}$). The authors of the paper in question also proposed a suboptimal controller given the respective architecture of their controller. However, it can be argued that the initial representation is flawed. The first flaw is related to the fact that this approach both sends a large number of messages and executes a rather complex actuator, thereby doing the opposite of trading bandwidth for computation. Second, it has an internal actuator state which is a matrix and it is well known from the theory of realizations that the total number of entries of such state can be chosen such that it is not larger that the size of the state vector, i.e., order of the system, *per se*.

In [GHBVdW12] it is proposed the use of two mechanisms to cope with missing messages. The first mechanism is employed in the controller, which assumes that control messages always arrive at its destination, whereas the process errors are filtered through an observer gain, executed only when there is a new sensor message. The second mechanism is similar and is employed at the actuator side, i.e., it assumes that the controller always receives the sensor messages and it uses an observer gain also used only when a new controller message is received. The authors also provide a stability analysis for systems in which their assumptions are valid. In essence, the paper is similar to [KF11] in which the matrix $\mathbf{M}_\mathrm{k}$ is chosen as the closed-loop state transition matrix. It is given no reason for the use of this particular choice nor it is proven that their choice has the behaviour in question, two points that are covered in this chapter. Furthermore, the authors failed to notice 1) the suboptimality of observers as opposed to the LQG, which implies that the optimal LQG gain applied to the observer in the actuator simply returns the value received from the controller which in turn renders the observer part of the actuator pointless. 2) the use of an actuator with observing capabilities renders the use of the controller pointless, since the sensor could send its data directly into the actuator where the data would receive a proper treatment. In fact, the use of an intermediary network element only increases the end-to-end error probability.

An aspect related to the last critique of the previous paper is explored in [RK07] in which it is shown that in physically long networks the closer the controller is to the actuator the better the quality of control and (the relevant part) the best results are achieved when the controller and the actuator are collocated. This implies that the quality of control may be improved by moving some of the functionalities of the controller into the actuator, which is a general trend on the field, as can be attested by the contribution reported in this section. However, some proposals fully replicate the controller into the actuator. In this chapter, the internal function of the actuator is simulated in the controller but the converse is not true and this choice was made because there are no guarantees that the actuator will be, computationally wise, capable of performing the required operations.

In [MQF12] two new concepts are introduced to the subfield of control over lossy networks. The first one consists in the use of differentiated services (DiffServ) on control networks, which is intended to provide different QoS to different networked control processes. The second new concept is the use of MPC in which the cost function is minimized over a limited number of steps into the future and only the current control value is outputted whereas the remaining are discarded. The two *parts* are connected by a scheduler that uses Linear Programming (LP) to schedule the messages. However, due to the connection of various areas that are still open problems, the overall solution is suboptimal.

In [ESM07] the authors propose to estimate whether a given controller message reached the actuator. The remaining of the paper further develops the idea. Nevertheless, the assumption related to the type of system, i.e., rank of the various matrices, considerably limits the domain of application of the work in question. Furthermore, the problem set for which their approach is a solution of, has a trivial structure. Thus, optimal solutions to it, are already known from other theories.

In [VH09] it is considered the behaviour of the auto-covariance matrix of the state estimation error. In particular, the fact that it does not converge as in the case of the regular Kalman filtering. The analysis is done using the Stieltjes transform which transforms the set of (expected) eigenvalues of a given stochastic matrix into a polynomial in the complex plane. The Stieltjes transform of a probability density function is defined as $\int_D \frac{f(t)}{\lambda - t} \mathrm{d}t$, D represents the support domain of the distribution.

5.3 Controller Design Rationale

As referred to before, the basic idea proposed of this chapter is the application of the architecture presented in the previous chapter in lossy networks. The reason why this may be sensible is the fact that even if the optimal value is missing, due to communication errors, the actuator can output an acceptable value if the controller had previously predicted and sent a set of future actuation values, which are then stored locally at the actuator. Thereby, the extra data that in the previous chapter were used to reduce the used bandwidth, is used in this chapter to increase the tolerance of the control system to network errors. Figure 5.2 depicts such a scenario.

Nevertheless, the architecture of the previous chapter requires a few adjustments to be applied on a lossy network, since in the approach presented in Chapter 4 the various control/network elements were capable of deciding which packets were not transmitted/received,

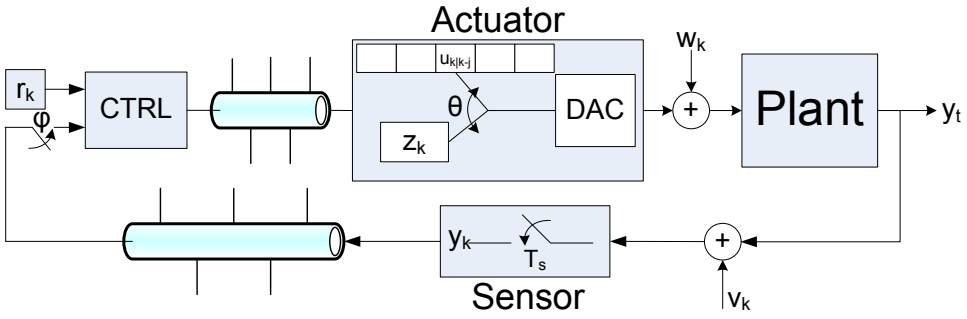


Figure 5.2: Proposed Control Architecture

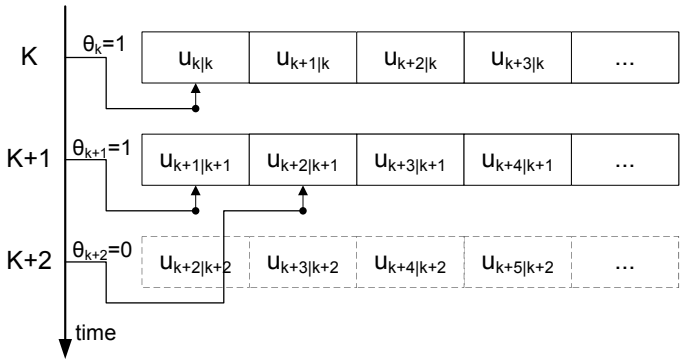


Figure 5.3: Example of Actuator Output Sequence.

whereas in the case discussed in this chapter, the due to the random nature of the network errors, the (sensor) controller cannot predict which messages will be successfully delivered. Therefore, in each execution, the controller computes several future control values, given the past information that it has, and then sends them to the actuator. Whenever a message is either lost or delivered late, the actuator can simply apply an estimated value from the last successfully received message, hence, the controller cannot know which value was effectively applied. Figure 5.3 provides an example of such process, where the message sent at time $K + 2$ is lost and thus it is used an estimate sent at time $K + 1$.

Note that systems with dominant, discrete-time, i.e., after the discretization, closed-loop poles close to 1, produce control sequences that change relatively slowly, i.e. any two consecutive control values differ by a small amount. Hence, in these circumstances and if the cost of control is relatively small, applying the hold strategy provides good performance. Conversely, for systems in which present actuation values are not correlated with past values and the cost of control is relatively high, it makes sense to output zero in the event of message losses. This is the case where the output zero strategy performs better than the hold strategy. However, the idea presented in this chapter is optimal in either extreme and in the cases in between.

Furthermore, though it is presented as a mechanism in which the controller sends a bulk of control values to the actuator, it is possible to implement a similar control system in which the controller sends its estimate of the system state and the actuator computes future estimates whenever it does not receive a new message. The approach adopted in this chapter has the advantage of requiring only one network element with a relatively high computational power, as discussed before. In Chapter 6 will be presented an actuator based compensation and a comparison with the approach in this chapter.

It is assumed that the network drops the control messages as an i.i.d process. However, this assumption can be relaxed, since error bursts that occur consecutively but within the span of the same message are already considered in the analysis. Furthermore, the expected number of message transmissions that guarantees that at every instant there is a control value with a given probability, is used to determine size of the buffers. This already deals with non i.i.d. packet drop processes because if, for example, the errors occur in burst of N messages, then the buffer need to have only $N + 1$ entries.

Obviously, the approach proposed in this chapter has an associated overhead. Regarding the network, as discussed on Chapter 4, current fieldbuses have a payload that easily accommodates the additional data. As for the computational capabilities, most of the ad-

ditional burden is put on the controller node, which normally is a node with relatively high computational capabilities.

5.4 Problem Statement and Solution

Consider a fully distributed system, as depicted in Figure 5.2, with linear dynamics. Notwithstanding its linear dynamics, there are some differences with the previous chapter that makes it is necessary to redefine some of its variables and to introduce others. The system under this architecture has the state-space representation:

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}z_k + w_k \tag{5.2a}$$

$$y_k = \mathbf{C}x_k + v_k, \tag{5.2b}$$

where x is the state variable, z is the control value that is applied by the actuator, w is the input noise, y is the output as sampled by the sampler, v is the output noise, and k is the sample index. Note that the variable u, that appeared in previous chapters, is the control value as computed by the controller, as opposed to z, the value applied by the actuator. $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are the state transition, input and output matrices, respectively. Let $\hat{x}_{k|k-j}$ be the optimal estimate of x_k given all the information known at instant $k-j$.

Let there also be defined

$$\mathbf{P}_{k|k-j} = \mathbb{E}\left[(\hat{x}_{k|k-j} - x_k)\,(\hat{x}_{k|k-j} - x_k)'\right] \tag{5.3a}$$

$$\mathbb{E}\left[[w_k\ v_k][w_i\ v_i]'\right] = \begin{bmatrix}\mathbf{Q} & \mathbf{0}\\ \mathbf{0} & \mathbf{R}\end{bmatrix}\delta_{ki}, \tag{5.3b}$$

where $\mathbf{P}_{k|k-j}$ is the auto-covariance matrix of the estimation of x_k given all the information received up to the instant $k-j$. $\mathbf{Q}$ and $\mathbf{R}$ are the input and output error covariance matrices, respectively. These last two matrices are not necessarily constant. However, since this fact does not have a direct impact in the results presented herein, it will be assumed that they are constant. δ_{ki} is the Kronecker delta which is 1 if $k = i$ and 0 otherwise. It is also assumed that w and v are uncorrelated white Gaussian noise sequences.

Considering a linear controller and that the certainty equivalence principle holds, as will be shown in a subsequent section, each possible output of the controller can be written as:

$$u_{k|k-i} = -\mathbf{L}_k\hat{x}_{k|k-i} \tag{5.4}$$

in which $\hat{x}_{k|k-i}$ is the state estimation as defined above, $\mathbf{L}_k$ is the controller gain matrix at the k^{th} step and $u_{k|k-i}$ is the value that the controller expected at instant $k-i$ to be the optimal control value at instant k.

Due to the fact that the architecture considered in this chapter is fully distributed, communication errors may happen both between the sensor and the controller and between the controller and the actuator. It is assumed that the underlying communication network provides suitable error correction/detection mechanisms so that messages are either correctly received or discarded. As illustrated in Figure 5.2, transmission errors are modeled by the boolean variables φ_k (sensor-controller) and θ_k (controller-actuator), which take the logical value 1 if the transmission is carried out correctly and 0 otherwise. Furthermore, regarding their stochastic behavior, it is assumed that these variables are i.i.d. Bernoulli variables, i.e., $Pr(\theta_k = 1) = Pr(\varphi_k = 1) = \bar{\theta}$ and $Pr(\theta_k = 0) = Pr(\varphi_k = 0) = 1 - \bar{\theta}$.

As stated before, in virtue of the employed architecture, in each period $(k-j)$ the controller computes u_{k-j} as well as a set of j future values $\{\hat{u}_{k-j+1|k-j}\,\hat{u}_{k-j+2|k-j}...\hat{u}_{k|k-j}\}$. These future values are computed based in the expectation of the system state x at their respective application instants. It is important to stress that these estimates are updated by the controller and sent to the actuator in each controller execution. Therefore, each one of the estimated control values $\hat{u}_{k-i|k-j}, i \in \{0..(j-1)\}$ is applied if and only if its predecessor $\hat{u}_{(k-i)-1|k-j}$ was also applied. This is so because the correct reception of a message at instant $(k-m)$ replaces all previous estimates in the interval $\{(k-m)...k\}$. This fact significantly simplifies the computation of $\hat{u}_{k|k-j}$.

Under the presented conditions above, the value that is actually used by the actuator at period k, i.e., z_k, may be different from the value generated by the controller at the same execution u_k, being equal to:

$$z_k = \sum_{j=0}^{k} \theta_j \left[\prod_{l=j+1}^{k} (1-\theta_l) \right] u_{k|j} \tag{5.5}$$

The boolean part of the argument of summation is 1 if the j^{th} message was received and all the future messages up to the instant k were not. Therefore, the whole summation tells 1) what was the last received message and 2) what was the estimate of u_k at that time.

Hence, the state in the next step, can be written as

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}z_k + w_k \tag{5.6a}$$

$$= \mathbf{A}x_k + \mathbf{B}\sum_{j=0}^{k} \theta_j \left[\prod_{l=j+1}^{k} (1-\theta_l) \right] u_{k|j} + w_k \tag{5.6b}$$

in which (5.6b) is a substitution of z_k, due to (5.5), into (5.6a); (5.6a) is one of the dynamic equations.

5.4.1 Noise Filtering

As discussed on chapter 2, in centralized control problem, the computation of the state estimates involves two complementary steps. In the first one, designated **innovation**, an estimation of the state in the next instant is made, followed by the the computation of its associated error covariance matrix. The estimates associated with the innovation step are computed solely based on the dynamic equations, with no information whatsoever of the output. The second step, designated **correction**, consists in a comparison of actual observations, i.e., the output as for example read by a sensor (with certain error covariance matrices), with the previously made estimations and the computation of new estimations with a lower error covariance matrix. Both of these steps are presented in detail in the following sections. Since UDP- and TCP-like protocols behave differently, a separate analysis for each of them is presented.

Innovation

The innovation step is based on Equation (5.6), consisting on the computation of the state variable at the next instant given the information available prior to the reception of

the sample of the instant in question. UDP-like protocols lack acknowledgment mechanisms, therefore the values of θ_j are unknown on the controller side. Given the type of control that is proposed, this behavior can be modeled with the help of three new variables, $t_k, \tilde{e}_k$ and ϵ_k, defined as follows:

- $t_k = \theta_k \hat{x}_{k|k} + (1-\theta_k)\left(\mathbf{A} - \mathbf{BL}_{k-1}\right) t_{k-1}$ is the state estimate that results from equation (5.5), representing the state estimation that was (or would have been if the system were centralized) used when computing the control value that the actuator actually applied to the environment;
- $\tilde{e}_k = \varphi_{k+1}\left((\mathbf{A} - \mathbf{BL}_{k-1})\hat{x}_{k-1|k-1} - \hat{x}_{k|k}\right)$ is the information acquired about the state variable by performing the k^{th} correction step, being equal to the difference between the uncorrected and corrected state estimation. If no sensor message is received and thereby with no correction, this variable is equal to zero;
- ε_k is a state variable associated with the input $\tilde{e}_k$, i.e., $\varepsilon_{k+1} = (\mathbf{A} - \mathbf{BL}_k)\varepsilon_k + \varphi_{k+1}\tilde{e}_{k+1}$, where φ_{k+1} is the boolean variable defined above, which models the communication errors between the sensor and the controller.

Since t and ε were defined recursively, it is also necessary to define boundary conditions. Given the physical meaning associated with the variables, it is sane to define $t_0 = \hat{x}_0$ and $\varepsilon_0 = 0$. The first equality follows from the definition by comparing the definitions of t_0 and $\hat{x}_0$. The second equality follows from the fact that at initialization time no corrections have been made yet. Note that $\varphi_{k+1}\tilde{e}_{k+1} = \tilde{e}_{k+1}$ since whenever $\varphi_{k+1} =$ '0' there is no update hence $\tilde{e}_{k+1} = 0$, hence the multiplication by φ_{k+1} is redundant, and whenever $\varphi_{k+1} =$ '1' it is the neutral element of multiplication, therefore, it is always redundant.

Under these assumptions $t_k = \hat{x}_{k|k} + (1-\theta_k)\,\varepsilon_k$. The last statement can easily be proven by induction, i.e., the initial value already satisfies the assumption since by definition $t_0 = \hat{x}_0$ and $\varepsilon_0 = 0$, and the recursion can be verified as follows

$$
\begin{aligned}
t_{k+1} &= \theta_{k+1}\hat{x}_{k+1|k+1} + (1-\theta_{k+1})\left(\mathbf{A} - \mathbf{BL}_k\right) t_k && \text{(5.7a)}\\
&= \theta_{k+1}\hat{x}_{k+1|k+1} + (1-\theta_{k+1})\left(\mathbf{A} - \mathbf{BL}_k\right)\left(\hat{x}_{k|k} + \varepsilon_k\right) && \text{(5.7b)}\\
&= \theta_{k+1}\hat{x}_{k+1|k+1} + (1-\theta_{k+1})\left[\left(\hat{x}_{k+1|k+1} + \varphi_{k+1}\tilde{e}_{k+1}\right) + \left(\mathbf{A} - \mathbf{BL}_k\right)\varepsilon_k\right] && \text{(5.7c)}\\
&= \hat{x}_{k+1|k+1} + (1-\theta_k)\left[\varphi_{k+1}\tilde{e}_{k+1} + \left(\mathbf{A} - \mathbf{BL}_k\right)\varepsilon_k\right] && \text{(5.7d)}\\
&= \hat{x}_{k+1|k+1} + (1-\theta_k)\,\varepsilon_{k+1}. && \text{(5.7e)}
\end{aligned}
$$

This new set of variables finds its application in the description of the dynamic equation since the state and the state estimation variables can be written as:

$$
\begin{aligned}
x_{k+1} &= \mathbf{A}x_k - \mathbf{BL}_k t_k + w_k && \text{(5.8a)}\\
\hat{x}_{k+1|k} &= \mathbf{A}\hat{x}_{k|k} - \mathbf{BL}_k \hat{t}_k && \text{(5.8b)}\\
\hat{t}_{k+1} &= \bar{\theta}\hat{x}_{k+1|k} + (1-\bar{\theta})\left(\mathbf{A} - \mathbf{BL}_k\right)\hat{t}_k && \text{(5.8c)}\\
\tilde{e}_k &= \left(\mathbf{A} - \mathbf{BL}_{k-1}\right)\hat{x}_{k-1|k-1} - \hat{x}_{k|k} && \text{(5.8d)}\\
\varepsilon_{k+1} &= \left(\mathbf{A} - \mathbf{BL}_k\right)\varepsilon_k + \varphi_{k+1}\tilde{e}_{k+1}, && \text{(5.8e)}
\end{aligned}
$$

in which Equation (5.8a) follows due to the assumption that t_k is the state variable used to produce z_k, i.e., the control value that was actually applied, hence, using the linearity

of the controller it follows that $t_k = -\mathbf{L}_k z_k$, which when substituted on Equation (5.2a) yields Equation (5.8a). Equation (5.8b) follows from Equation (5.8a) because it dictates the behaviour of the state estimates. Note the introduction of a new variable $\hat{t}_k$ which is an estimate of t_k which is the object of Equation (5.8c) in which the equation for the evolution of $\hat{t}_k$ is laid out. Equation (5.8d) and Equation (5.8e) do similar things to $\tilde{e}_k$ and ε_{k+1}. Note that the new ε_{k+1} does not come with an estimation sign, because the previously defined ε_{k+1} was already an estimate.

Equation (5.8) implies that $e_{k+1|k} = \hat{x}_{k+1|k} - x_k$ is equal to

$$e_{k+1|k} = \mathbf{A}e_{k|k} - w_k - \left(\bar{\theta} - \theta_k\right)\mathbf{B}\mathbf{L}_k\varepsilon_k. \tag{5.9}$$

The error auto-covariance matrix, i.e., $\mathbb{E}\left[e_{k+1|k}e'_{k+1|k}\right]$, is therefore

$$\mathbf{P}_{k+1|k} = \mathbf{A}\mathbf{P}_{k|k}\mathbf{A}' + \mathbf{Q} + \bar{\theta}\left(1-\bar{\theta}\right)\mathbf{B}\mathbf{L}_k\varepsilon_k\varepsilon'_k\left(\mathbf{B}\mathbf{L}_k\right)' \tag{5.10}$$

in which there are no terms in $\mathbb{E}\left[e_{k+1|k}\varepsilon'_k\right]$ despite the correlation of the two variables being non-zero, because all such terms come multiplied by $\mathbb{E}\left[\left(\bar{\theta} - \theta_k\right)\right]$ which is zero by (the) definition (of $\bar{\theta}$). The past equations fully describe the innovation step under UDP-like protocols.

Under TCP-like protocols, it is always known whether or not a message was delivered. Therefore, even though TCP-like protocols have a similar mechanism (involving ε), the applied value, z_k, is always known. For this reason, there is no need to estimate the value that was applied by the actuator. This implies that the innovation step is equal to the innovation step of the standard Kalman Filter, i.e.

$$\hat{x}_{k+1|k} = \mathbf{A}\hat{x}_{k|k} + \mathbf{B}z_k \tag{5.11a}$$
$$\mathbf{P}_{k+1|k} = \mathbf{A}\mathbf{P}_{k|k}\mathbf{A}' + \mathbf{Q}_k \tag{5.11b}$$

Correction

The correction step is independent of the type of protocol used to exchange messages between the controller and the actuator, because it depends only of the sensor-controller message transfer and at the time of the correction the controller knows whether or not it has received a new sensor message. Note that the notion of UDP vs TCP-like, in this particular scenario ought to be applied to the sensor, since it is the sender of the message, but the sensor is not the network element that is performing the computation. In fact, such element, is the controller, which is the one that supposedly sends a message acknowledging the reception of the sensor's message. In any case, the controller always know whether it received the sensor message at the time of the correction step.

The correction is made only when a sensor message is received, since otherwise there would be no sample to compare the estimates with. For this reason, K_{k+1} is multiplied by φ_{k+1}. The rest of the correction step is equal to the correction step of a standard Kalman filter, i.e.,

$$\mathbf{K}_{k+1} = \varphi_{k+1}\mathbf{P}_{k+1|k}\mathbf{C}'\left(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}' + \mathbf{R}_k\right)^{-1} \tag{5.12a}$$
$$\mathbf{P}_{k+1|k+1} = \left(\mathbf{I}_n - \mathbf{K}_{k+1}\mathbf{C}'\right)\mathbf{P}_{k+1|k} \tag{5.12b}$$
$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + \mathbf{K}_{k+1}\left(y_{k+1} - \mathbf{C}\hat{x}_{k+1|k}\right). \tag{5.12c}$$

Note that if $\varphi_{k+1} = 0$, then Equation (5.12) implies that no correction is made, i.e., $\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k}$ and $\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k}$.

5.4.2 Optimal Control over Lossy Networks — TCP-Like Protocols

Thanks to the existence of controller-actuator acknowledgments, the general principles behind the dynamic programming theory, as described in Section 2.3.1, are valid for TCP-like protocols. It is important to notice that in this case: 1) the controller knows the inner state of the actuator and its inputs, thus, it is able to determine the applied value, 2) once a controller message is delivered, its respective values will be outputted until the reception of a new controller message, and 3) the certainty equivalence principle is still valid since the error does not depend on the control variable and according, for example, to [TBS75] whenever this condition is verified, the certainty equivalence is valid. Taking these three observations in consideration, the computation of new control values under the proposed scheme is no different than the computation of such values under classical optimal control, which is presented below.

Let J_k be defined as:

$$J_k(x_k) = x'_N \mathbf{W}_N x_N + \sum_{j=k}^{N-1} (x'_j \mathbf{W}_j x_j + u'_j \mathbf{U}_j u_j) \tag{5.13}$$

with k and N the limits of the minimization window. Lets also define the value function $V_k(x) = x'_k \mathbf{P}_k x_k + c_k$ as the usual value function, in which $V_k(x_k) = \min_{u_k} J_k$, also known as the cost-to-go function, as defined in Chapter 2. Then using the Bellman-Hamilton-Jacobi theorem, $V_k(x_k)$ can be written as

$$V_k(x) = x'_k \mathbf{W}_k x_k + \min_{u_k} \Big(u'_k \mathbf{U}_k u_k + (\mathbf{A}x_k + \mathbf{B}u_k + w_k)' \mathbf{P}_{k+1} (\mathbf{A}x_k + \mathbf{B}u_k + w_k) \Big) + c_{k+1} \tag{5.14}$$

At each control iteration, the controller produces a set of values that are assumed to be applied according to the description provided above. This set of rules, pre-programmed into the various networked control elements ensures that a given control value is applied only if the previous control value was applied, as discussed above.

The reasoning presented in the last paragraph plus the fact that the separation principle holds for TCP-like protocols implies that the minimization can ignore the network side and simply minimize in order to u_k, i.e.

$$\frac{\partial V_k(x_k)}{2\partial u_k} = u'_k \mathbf{U}_k + (\mathbf{A}x_k + \mathbf{B}u_k + w_k)' \mathbf{P}_{k+1} \mathbf{B} = 0 \tag{5.15}$$

The last equation was equated to zero because it is known that the minimum is an extremum of $V_k(x_k)$. Furthermore, it is solved into (after transposition)

$$u_k^* = -\left(\mathbf{B}' \mathbf{P}_{k+1} \mathbf{B} + \mathbf{U}_k\right)^{-1} \mathbf{B}' \mathbf{P}_{k+1} \mathbf{A} x_k. \tag{5.16}$$

The previous equation implicitly defines $\mathbf{L}_k$ that appears in $u_k = -L_k x_k$. The equation defines the optimal controller in terms of the actual state variable, x_k, as opposed to the estimation of the state variable. This is allowed due to the certainty equivalence principle which is not broken under TCP-like protocols.

Subtracting an appropriate form of Equation (5.15) from Equation (5.14) yields

$$V_k(x_k) - \frac{\partial V_k(x_k)}{2\partial u_k} u_k = x'_k \mathbf{W}_k x_k + u'_k \mathbf{U}_k u_k + (\mathbf{A}x_k - \mathbf{B}\mathbf{L}_k x_k + w_k)' \mathbf{P}_{k+1} (\mathbf{A}x_k + w_k) + c_{k+1}. \tag{5.17}$$

$\mathbb{E}[x_k w_k'] = \mathbf{0}_{n\times n}$ because otherwise the system would either be non-causal in which case w_k would be able to affect at least one state that temporally preceded it, i.e., $\exists_{j\leq k} : x_j = x_j(w_k)$, or the system would violate the assumption that w_k is a white noise sequence in which case a previous perturbation could influence both the current state and the current input perturbation.

$\mathbb{E}[x_k w_k'] = 0$ eliminates the cross terms containing such terms from Equation (5.17). Then a comparison of the terms in the definition of $V_k(x_k)$ with Equation (5.17) implies that:

$$\mathbf{P}_k = \mathbf{W}_k + (\mathbf{A} - \mathbf{B}\mathbf{L}_k)' \mathbf{P}_{k+1}\mathbf{A} \tag{5.18a}$$

$$c_k = c_{k+1} + trace\,(\mathbf{Q}\mathbf{P}_{k+1}) \tag{5.18b}$$

with the boundary conditions $\mathbf{P}_N = \mathbf{W}_N$ and $c_N = 0$.

Note that the controller presented above is structural very close to the centralized one that can be found, for example, in [Kired].

5.4.3 Optimal Control Over Lossy Networks — UDP-Like Protocols

The computation of optimal control values under UDP-like protocols is normally more complex than the computation of optimal control values under TCP-like problems. As pointed out in the introduction section, under this type of protocol the separation and certainty equivalence principles are not guaranteed to hold. Furthermore, when these principles do not hold, the minimization problem, from which the optimal control value stems, is no longer quadratic.

However, in Section 5.4 was shown that $e_{k|k-1} = \hat{x}_{k|k-1} - x_k$ is independent of x_k even over UDP-like protocols. This fact implies that $\mathbf{P}_{k|k-1} = \mathbb{E}\left[e_{k|k-1}e_{k|k-1}'\right]$ must also be independent of x_k. In fact, it is possible to prove that some of the other extended inputs techniques discussed in Section 5.2 also have this property.

Having the above mentioned property is a sufficient condition for the certainty equivalence principle to hold, as shown in [TBS75]. This implies that the optimal controller can be derived as the *classical* optimal controller, as shown in the previous subsection.

Nevertheless, the actual control values will be different since they are computed from different state estimates, which, for example, have different error auto-covariance matrices.

5.5 Simulations and Results

In order to evaluate the validity of the methods proposed in this chapter, two sets of simulations were performed. The first set was centered in the validation of the method *per se*, and to show its benefits when compared to other methods, whereas the second set of simulations is a remake of a simulation performed in the paper [SSF+07] which was modified in order to include the methods presented herein. Both simulations were made using `Matlab`®.

The first set of simulations used were performed in a system that had the following discrete-

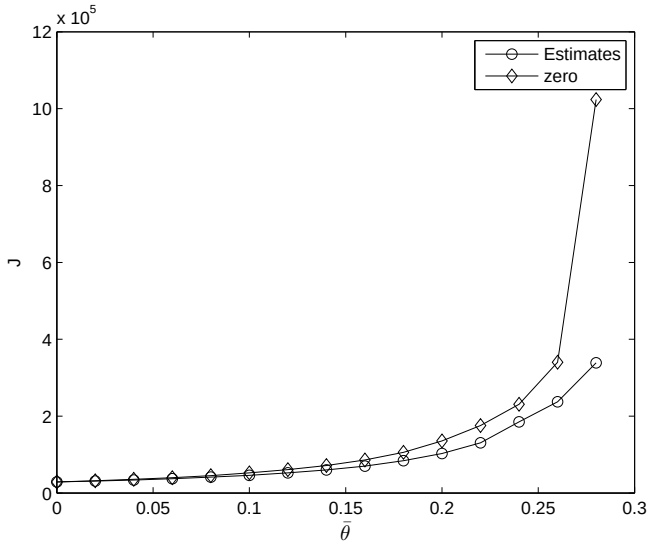


Figure 5.4: Output Zero versus Proposed Approach under TCP-like Protocols.

---

time parameters:

$$\mathbf{A} = \begin{bmatrix} 1.2 & 0.8 \\ 0.5 & 0.9 \end{bmatrix} \tag{5.19a}$$

$$\mathbf{B} = [1.1\ 0.9]' \tag{5.19b}$$

$$\mathbf{C} = [-0.9\ 1.1] \tag{5.19c}$$

$$\mathbb{E}\left[[w_k\ v_k][w_j\ v_j]'\right] = \begin{bmatrix} 1.2 & 0.7 & 0.00 \\ 0.7 & 1.1 & 0.00 \\ 0.0 & 0.0 & 0.60 \end{bmatrix} \delta_{kj} \tag{5.19d}$$

The simulated system had poles (z − plane) at 1.7 and 0.4. The simulations, in essence, varied the transmission error probability. For each transmission error probability there were made 230.000 simulations to filter out the stochastic nature of each run. At each run, there were made 300 steps, i.e., k varied from 0 at initialization to 300 at the end. The system had an initial value of $x_0 = [1.32,\ 0.97]'$, chosen once randomly, but used in all experiments, to increase the degree of repeatability of the experiments, though it ended up proving itself of low value given the rather long duration of each run.

The cost matrices were

$$\mathbf{W} = \begin{bmatrix} 0.81 & -0.99 \\ -0.99 & 1.21 \end{bmatrix} \tag{5.20a}$$

$$\mathbf{U} = 0.70 \tag{5.20b}$$

where $\mathbf{W}$ was arbitrarily set to $\mathbf{C}'\mathbf{C}$ and $\mathbf{U}$ was randomly selected.

The TCP-like case was simulated both with the approach proposed here and with the output zero strategy, i.e., $u_k = \theta z_k$. However, it was not simulated for the *hold* case. That is because the optimal controller for the output zero strategy was already presented in the literature (for example [SSF+07]) whereas the *hold* case, to the best of the authors knowledge, has not been extensively studied up to the moment in which this work was done. Chapter

6 will address this matter. The UDP-like protocol was simulated only with the solution provided in this paper, for reasons similar to the previous one, i.e., other output methods do not provide any clue on optimal control values.

Figure 5.4 compares the behaviour of the output zero strategy with the strategy proposed herein, for TCP-like protocols. The graph is shown with a (natural) logarithmic y-axis due to the rather wide range of total cost J. At low error probabilities the performance of both strategies is indistinguishable. However, as the error probability grows the performance difference grows dramatically, in favour of the estimation-based approach proposed in this chapter.

Figure 5.5 compares the behaviour of the strategy proposed herein for the TCP and UDP-like protocols. Once again, at low error probabilities both methods are indistinguishable. However, as the error probability increases the differences become evident. It is unfortunate, however, that the authors could not find another proposals in the literature claiming optimality under UDP-like protocols for a fair comparison.

5.5.1 Comparison with other solutions

The second set of experiments is based on a comparison previously made in [SSF+07] between the zero and hold strategies, extended with the prediction-based method presented herein.

It consists in a first order discrete-time system, in which $\mathbf{A} = 1.2$, $\mathbf{B} = \mathbf{C} = 1$ and $\mathbf{Q} = \mathbf{R} = 1$. In the original experiment the authors considered $\bar{\varphi} = 1$, i.e., no errors in sensor to controller transmissions, and $\bar{\theta} = 0.5$. This is true only if different network segments are used. However, most communication networks use a *bus* (with one or more segments) that connects all the nodes. In this scenarios it is more realistic to assume that any source of errors in one segment of the network is very likely to affects the other segment as well, with a similar probability. Or simply to assume that the whole network is one single segment. Therefore, it was used $\bar{\varphi} = \bar{\theta} = 0.5$.

As previously discussed in the rationale section, the hold strategy performs best when

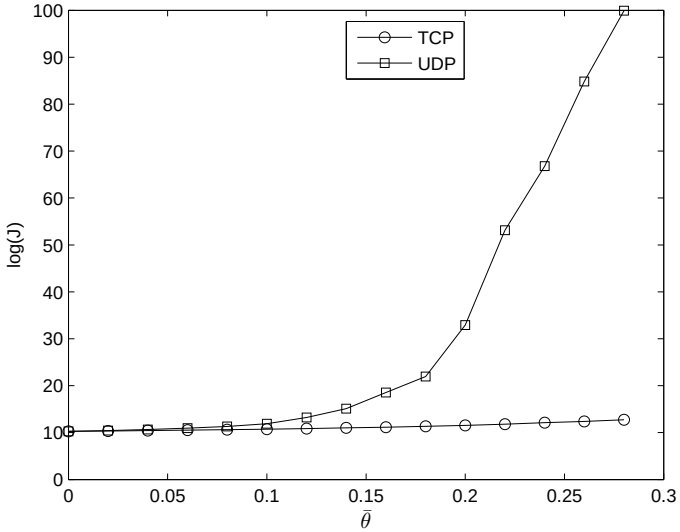


Figure 5.5: TCP-like versus UDP-like Protocols for Estimation Output Extensions.

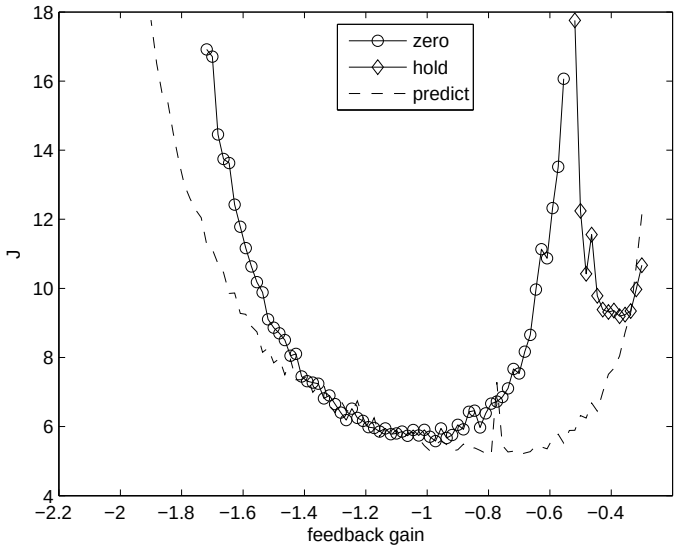


Figure 5.6: Output strategies comparison

the discrete-time closed-loop system has all of its poles close to 1, because in this situation the control value does not vary significantly between successive control periods. For a similar reason, the output zero strategy performs better when the closed loop system has all of its poles close to origin of the $z-$ plane, because in this situations the control value is almost always close to zero, thus applying zero approximates the optimal control value.

As shown in Figure 5.6, the performance of the zero and hold strategies depends strongly on the feedback gain, as expected, since the position of the poles depends on this parameter. It should also be remarked the high sensitivity to the feedback gain. Thus, minimizing the ISE is hard, mainly for practical systems where the exact parameter values are often difficult or even impossible to obtain.

The prediction case presents a very different behavior. Firstly it can be seen that it always offers a lower ISE than any of the other approaches, as claimed. In addition, the ISE is maintained near the minimum for a large range of feedback gain values, thus making the system much more amenable for use in practical scenarios. Therefore, it can be concluded that, in the presence of communication errors, the use of the prediction-based strategy, proposed in this paper, outperforms the zero and hold strategies in terms of ISE and is easier to use in practice due to its lower sensitivity to feedback gain.

5.6 Summary and Conclusions

The performance of distributed control systems is negatively affected by the presence of communication errors. Classical approaches to deal with missing actuation data consist in letting the controller output zero or the previous control value. This chapter shows that these techniques only perform well in particular circumstances and are hard to use in practice, due to their high sensitivity to feedback gain. To address these limitations, this chapter proposes a novel prediction-based technique, in which the controller sends the current actuation value as well as a set of future estimates, which are used only in case of communication errors. This approach ensures that the value that is applied by the actuator is as close as possible to the

value produced by the controller, even in the presence of network errors.

The basic mechanisms and theoretical foundation of the novel prediction-based technique are presented in the chapter. Simulation results show the feasibility of the proposed technique as well as that it outperforms classical approaches in terms of ISE, or other square-error measures, exhibiting at the same time a significantly lower sensitivity to feedback gain, thus being more amenable to use in practical scenarios.

Chapter 6

Control over Lossy Networks with Generalized Linear Output

The discipline of linear control in centralized systems has been extensively studied and optimal results can already be found in the literature. However, in recent years, new variants of control systems have appeared, among them are the control communication networks with various degrees of reliability. As discussed in previous chapters, two approaches have been defined to cope with missing control values in such non-ideal networks: output zero and hold the previous output. The optimal control law over output zero lossy networks has already been presented in the literature. In this chapter, we present the optimal control law for generalized linear output schemes. The control law derived from such generalized output scheme has more internal structure, being equal to the optimal controller of each one of its sub-cases, including the zero and the hold approaches. Furthermore, this chapter presents the optimal output strategy to which the optimal control under it produces the smallest cost. Two proofs are provided for the proposed control law, one based on the collection/gathering of terms and another based on a recursive differential expansion. The novel control law is tested via simulation and the obtained results are in perfect agreement with the presented theory. A connection is made between this approach and the control architecture that was presented in earlier chapters and it is proven that the optimal generalized hold controller is a complement to such architecture.

6.1 Introduction

Control theory is nowadays a well developed theory, with mathematically sound and experimentally proven foundations. Optimal control of linear centralized systems, i.e., systems where all of its components are located in the same physical device, was naturally the first to be discovered.

Improvements in networking technologies and computational platforms made Networked Control Systems (NCS) a reality. Despite bringing important benefits (deployment cost, deployment time, maintenance cost, to name just a few) the adoption of NCS also introduced several non-ideal aspects that were not taken into account in the initial formulation of optimal control. Such non-ideal aspects include jitter (variance in the arrival time), latency (average delay in the communication) and packet losses.

These effects have been the subject of intense research and appropriate solutions for some

of them are nowadays reported in the literature. For example, latency can be dealt with using predictive controllers that choose the best inputs based on the moment in which the control value is expected to be applied. For the case of lossy networks, there are two possible output strategies: zero and hold. In the zero approach, whenever a control value is not delivered, it is applied a zero. In an equivalent scenario, the hold approach outputs the value that was applied in the previous control cycle.

The optimal control law in output zero networks is now a standard part of control theory. The actual result maintains the overall structure of the optimal controller in centralized systems. However, the hold strategy leads to an optimal controller that has a structure that is significantly different from the optimal control law in centralized networks, though in the limit of no network error it converges to the optimal centralized control. In the author's opinion, this fact had an influence on the timing of the discoveries.

In this chapter it is presented a novel control law that is optimal in lossy control networks, in which the actuator applies a generalized linear function of the last applied value whenever there is a control message drop, as well as it is presented the optimal matrix of such linear function. Such optimal controller is then compared to the architecture proposed in this Thesis. Such comparison establishes the optimality of the architecture proposed earlier in this Thesis.

The remaining of this chapter is organized as follows: in Section 6.2 it is made a review of the contributions in the literature that are relevant to this work. In Section 6.3 it is presented the notation used throughout the chapter. In Section 6.4 it is presented a solution for the control over lossy networks with the output zero strategy, which will be compared to the solution of the control over lossy networks with the generalized hold strategy presented in this chapter. Section 6.5 is the main section of this chapter presenting two different, though equivalent, methods to solve the identified problem. The first one is an extension of classical approaches which keeps track of all the matrices that emerge due to the hold strategy. The second one uses the *Hessian* matrix to reduce the quadratic cost function into a more tractable form. Section 6.6 discusses aspects related to presented solutions as well as of the other types of control. In Section 6.7 it is deduced a condition for optimality of the linear function used at the actuator and several matrices that verify such condition, with varying degrees of complexity, are presented. In Section 6.8 it is made a comparison between the use of the architecture presented in previous chapters with the optimal linear transformation used in the actuator. In Section 6.9 it is made an evaluation, via simulation, of the control laws devised in this chapter. Finally, Section 6.10 summarizes the chapter and presents some concluding remarks.

6.2 Related Work

Optimal control is a relatively well-established discipline, arguably around 90 years old, which started with the work of Pontryagin and Richard Bellman, with Bellman's contribution being

$$V_k(x) = x_k' \mathbf{Q} x_k + \min_{u_k} \left(u_k' \mathbf{R} u_k + x_{k+1}' \mathbf{P}_{k+1} x_{k+1} \right) \tag{6.1}$$

where $V_k = x_k' \mathbf{P}_k x_k$ is the value function to be minimized, x is the state, u is the control signal, and $\mathbf{Q}$ and $\mathbf{R}$ are the weight/cost matrices. This equation assumes full state measurement and no state perturbations whatsoever. Hence, from a control practice standpoint, its usefulness is limited. This changed with the advent of Kalman filters, leading to the much heralded *Linear Quadratic Gaussian Regulators*. Estimation and control decoupling was explained by

the application of the certainty equivalence principle, which states that if $u_k = -\mathbf{L}_k x_k$, i.e., $\mathbf{L}_k$ is the optimal controller gain if the state is perfectly known, then if only a state estimation ($\hat{x}_{k|k}$) is available the optimal control value is $u_k = -\mathbf{L}_k \hat{x}_{k|k}$ [Wit71]. In [TBS75] it was shown that the certainty equivalence principle holds if, and only if, the estimation error covariance matrix is not a function of the state i.e.

$$\frac{\partial E\left[\left(\hat{x}_{k|k} - x_k\right)'\left(\hat{x}_{k|k} - x_k\right)\right]}{\partial x_k} = \mathbf{0}_{n\times n\times n} \tag{6.2}$$

Recent contributions in this line of work show that the certainty equivalence principle is still widely accepted (e.g. [BSJ07, HS05, ERA07, LZ10, MCH10, NFZE07, MH09, RSBJ11, Bat06]), though some works put its applicability in question for certain classes of systems, e.g., [FU98] and [FU95] show that in systems in which the parameters are unknown or because the system is non-linear, the use of the certainty equivalence leads to a suboptimal control, hence it is proposed an *heuristic* called *partial* certainty equivalence. Similar remarks were made in [Cha03] regarding self-tuned controllers.

Obviously, the subject of estimation and control over lossy networks is not as old as the subject of optimal control. The first appearances of optimal control in lossy networks in the literature tried to find a correspondence between the Shannon information theory [Sha48] and control theory. More specifically, they tried to answer the age old question: *what is the smallest bit-rate necessary to stabilize a given system* or, equivalently, what is the maximum transmission error rate that a network can have without compromising the stability of its systems? Moreover, *which techniques achieve this threshold?* This is a matter that, appears to be unexplored. A number of early attempts to answer the first question are presented in [HOV02][HNX07]. This chapter indirectly aides in such quest, by exploring a generalization of the control over lossy networks, which provides different types of possible controllers.

In [As03] the authors attempt to extend the classical theory of optimal control [Ber07][KJA97]. However, they conclude that the separation principle — which allows a simplification of the results — does not hold in the general case. However, other simpler results are shown, such as that a system is stable if the i.i.d. variables that describe the error rate (θ) verify $\theta < \max \lambda(\mathbf{A}^{-2})$, assuming that the system is unstable, where $\lambda(\mathbf{A})$ is the set of all eigenvalues of matrix $\mathbf{A}$ and $\mathbf{A}$ is the state-transition matrix. Similar claims are made in [HY07].

In [SSF+07] it is made a more in-depth study of such issues, presenting formally the UDP and TCP-like protocol cases. The TCP-like protocol is more amenable to the existing optimal control theory, whereas UDP-like protocol, under the assumptions made therein — $u_k = 0$, if $\delta_k = 0$[1]— leads to a complex optimization problem, since the estimation error covariance matrix at any given time step will depend on previous control values. It is shown in the same reference that, under their assumptions, TCP-like protocols have an error-rate similar to the one found in previous papers, but UDP-like protocols have lower error-rate bounds that guarantee stability.

Another somewhat related development is given in [HY08], in which state information is sent in more that one packet. This fragmentation is done primarily in an attempt to reduce impact of a packet loss, since loosing a packet with a significant large number of state variables would be worst than loosing one packet with only one variable.

In [KKH+08] it is taken a radically different approach. It uses passive networks as a mean to provide jitter immunity to the controlled system. However, no reasons to believe

[1] $\delta_k = 1$ if the message is delivered and $\delta_k = 0$ otherwise

that passive networks are more immune to jitter are presented, and no mechanism to design passive controllers to achieve certain metrics, e.g., state-tracking or input tracking under noisy conditions, are provided.

In [Sch09] it is presented an analysis similar to the analysis presented in this paper, in which an optimal controller was introduced. However, the authors placed a strong emphasis on the maximum drop rate that allowed for the stabilization of the system. The actual controller for the hold case was not fully presented/deduced, rather it was presented a reference to another article that provided the main intuition for the correctness of the proposed controller. Moreover, their main result only concerns the steady-state controller, not dealing the optimal controller over a finite number of steps. Note, however, that the results presented herein do not contradict the results in [Sch09]. In fact, the results presented herein extend them, both by presenting a demonstration of the controller for finite number of steps and extending it for other linear combination of the state and output variables.

In [KF11] the inputs schemes are extended from the zero versus hold spectrum by allowing the value that is held at the actuator to decay with time, that is,

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}s_k + w_k \tag{6.3a}$$
$$s_k = \mathbf{L}_k r_k \tag{6.3b}$$
$$r_k = \theta_k u_k + (1-\theta_k)\mathbf{M}_k r_{k-1} \tag{6.3c}$$

in which s_k is the value that is actually applied, r_k is the actuator internal state representation which is allowed to be a matrix, u_k is the control value computed at the controller using information from the sensors and it is transmitted in a bulk message, $\mathbf{M}_k$ is a state transition matrix for the actuator state and $\mathbf{L}_k$ is the respective controller gain. In fact, the authors of [KF11] assume that s_k is equal to the first column of r_k, hence, constraining $\mathbf{M}_k$. The authors of the paper in question proceeded by proposing a suboptimal controller given their control architecture. Nevertheless, no motivation for this particular control architecture was presented. In fact, a case can be made that their approach both sends a large number of messages and has a complex actuator. Second, the actuator internal state is matrix, i.e., rank 2 tensor, whereas it is well known, from realization theory, as can be seen in the survey [Sch00] and in references therein, that the total number of entries of such a state can be chosen such that it is not larger than the size of the state variable *per se*.

6.3 Problem Definition and Notation

In this chapter, are considered discrete-time LTI systems, which will be described using the state space representation

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k \tag{6.4a}$$
$$y_k = \mathbf{C}x_k, \tag{6.4b}$$

with x_k, u_k and y_k representing the system state, the system input (as applied by the actuator) and the system output vectors respectively, and $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are the state transition, the input and the output matrices, respectively. It is assumed that the variables x_k, z_k, u_k and y_k are column vectors of sizes $(n\times 1), (r\times 1), (r\times 1)$ and $(p\times 1)$ respectively. $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are matrices of appropriate dimensions. Two standard matrices are used frequently, namely the zero matrix

$\mathbf{0}_{n\times m}$, with n lines and m columns, and the identity matrix $\mathbf{I}_n$, which is a square matrix composed of 1 in the diagonal entries and 0 in all other entries.

The goal of the controller is to minimize, without loss of generality[2], a quadratic cost function, namely

$$J = x'_N \mathbf{Q}_N x_N + \sum_{k=0}^{N-1} x'_k \mathbf{Q}_k x_k + z'_k \mathbf{R}_k z_k, \tag{6.5}$$

with $\mathbf{Q}_k$ and $\mathbf{R}_k$ semi-positive definite matrices and N the duration of the minization.

This cost function is computed in a step by step manner — cost-to-go — which is essentially the cost of the current step plus the (minimum) cost-to-go of all future steps, i.e., $V_k(x_k) = x'_k \mathbf{Q}_k x_k + u'_k \mathbf{R} k u_k + V_{k+1}(x_{k+1})$.

Furthermore, it is assumed that the applied control value (z_k) depends on the delivery of a certain control massage as

$$z_k = \theta_k u_k + (1 - \theta_k)\mathbf{M} z_{k-1} \tag{6.6}$$

in which θ_k is a boolean variable that is '1' if the message is delivered and '0' otherwise and $\mathrm{u_k}$ is the control vector as produced by the controller. $\mathbf{M}_k$ is a square matrix, of suitable size. Whenever the actuator receives a new control value u_k, i.e., θ_k ='1', it outputs the received value, otherwise, i.e., θ_k ='0', it outputs a vector equal to the matrix $\mathbf{M}_k$ times the previous applied value. Note that if $\mathbf{M}_k = \mathbf{0}_{r\times r}$ then the output strategy will be equal to the output zero strategy and if $\mathbf{M}_k = \mathbf{I}_r$, then it will be equal to the hold strategy.

Beside the usual reasons why these equations are solved in the discrete-time, namely, regarding their implementation on digital microprocessors, there is another subtle reason for the discrete-time implementation of this type of control, namely, as the sampling period goes to zero, the control value become a Lebesgue function. A Lebesgue function, is a function that has a countable, though potentially infinite, number of discontinuities. This problem does not appear in physical systems because as the sampling time goes to zero the respective θ_k becomes correlated. However, in this chapter it is assumed that θ_k are i.i.d. variables. The issue of correlated θ_k is left for future work.

On a related note, though from a control perspective, network protocols can be organized in two groups, namely, TCP-like and UDP-like protocols, as discussed in Section 5.1 of Chapter 5, the main results of this chapter only consider TCP-like protocols. This choice was imposed by the fact that UDP-like protocols do not respect the separation principle, see [TBS75].

Note that, unlike the approaches followed in Chapters 4 and 5, the problem being tackled in this chapter does not allow for the use of batch transmissions, i.e., each control message only contains one value. In so being, the only change that is allowed is that of the respective control algorithms.

[2]It could have been minimized the more complete quadratic cost function

$$J = x'_N \mathbf{Q}_N x_N + \sum_{k=0}^{N} \begin{bmatrix} x_k \\ z_k \end{bmatrix}' \begin{bmatrix} \mathbf{Q}_k & \mathbf{N}_k \\ \mathbf{N}'_k & \mathbf{R}_k \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix},$$

but as will be shown, the result of both minimizations have essentially the same structure.

6.4 Optimal Control over Output Zero Lossy Networks

Even though the solution of this particular problem can be found in the literature (see [SSF+07]), it will be presented here, though with a different derivation, to allow for a comparison with the generalized optimal controller, presented in this chapter.

Theorem 1. *The value of the cost-to-go in a output zero lossy network with no estimation (and input) error is*

$$V_k(x_k) = x_k' P_k x_k \tag{6.7a}$$
$$P_k = Q_k + (A - \bar{\theta} B L_k)' P_{k+1} A \tag{6.7b}$$
$$L_k^* = (B' P_{k+1} B + R_k)^{-1} B' P_{k+1} A \tag{6.7c}$$

and $P_N = Q_N$.

Proof. This theorem is proven by induction. Obviously it is valid for $k = N$, in which case the definition of the cost function and of V_k coincide. Considering previous steps leads to

$$V_k(x_k) = x_k' Q_k x_k + \theta_k u_k' R u_k + (A x_k + \theta_k B u_k)' P_{k+1} (A x_k + \theta_k B u_k) \tag{6.8}$$

calculating the derivative of $V_k(x_k)$ in order to u_k and equating it to zero, yields

$$\frac{\partial V_k(x_k)}{2 \partial u_k} = \theta_k u_k' R + \theta_k (A x_k + \theta_k B u_k)' P_{k+1} B = \mathbf{0}_{1 \times r} \tag{6.9}$$

which implies that

$$V_k(x_k) - \frac{\partial V_k(x_k)}{2 \partial u_k}\Big|_{u_k = u_k^*} u_k^* = x_k' Q_k x_k + (A x_k + \theta_k B u_k^*)' P_{k+1} A x_k. \tag{6.10}$$

The statement regarding the value of L_k^*, i.e., Equation (6.7c), is proven by solving Equation (6.9) in order to u_k followed by a comparison with the definition of L_k^*. However, it should be noticed that the value of u_k is relevant only if $\theta_k = 1$, since otherwise it will not be applied. The statement regarding $\mathbf{P}_k$ (Equation (6.7b)) is proven by comparing Equation (6.10) (after taking expectation over θ_k) with the definition of $V_k(x_k)$. □

6.5 Optimal Control over Generalized Hold Lossy Networks

In this section, two different proofs of the theorem regarding optimal control over output hold lossy networks are presented. They both produce the same optimal control law. Nevertheless, both of them will be presented in an effort to allow readers comfortable with only one of the approaches to be able to follow the arguments.

Due to the dependence of the current control value with current and past values of θ, in hold strategy and its generalizations, there is a need to include the current actuation value in the cost function to encapsulate its future contributions to the cost stemming from its application through the hold mechanism. For this motive, the current value of the cost function is denoted $V_k(x_k, z_k)$. Furthermore, the cost function at future instants is also dependent of both x_j and z_j, with j being the future instant in question. But from the perspective of the current instant, it is a function of z_k, $\theta_i, \forall k \leq i < j$ and of x_k. For this reason, for instance, the one-step expectation of the cost function, i.e., the expected cost-to-go starting from the next step, is $V_{k+1}(x_{k+1}; \theta_{k+1}, z_k)$ with the last two terms being used to determine z_{k+1}.

6.5.1 Recursive Matrix Derivation

Theorem 2. *The cost-to-go function of control in a lossy network is given by* $V_k(x_k, z_{k-1}) = \theta_k x_k' \mathbf{P}_k x_k + (1-\theta)(x_k' \mathbf{Q}_k x_k' + z_{k-1}' \mathbf{R}_k z_{k-1} + V_{k+1}(x_{k+1}, z_{k-1}))$ *with*

$$\mathbf{P}_k = \mathbf{Q}_k + \left(\mathbf{A}'\mathbf{I}_{k+1} - (\mathbf{B}\mathbf{L}_k^*)'\,\mathbf{H}_{k+1}\right)\mathbf{A} \tag{6.11a}$$

$$\mathbf{I}_k = \theta_k \mathbf{P}_k + (1-\theta_k)\left(\mathbf{Q}_k + \mathbf{A}'\mathbf{I}_{k+1}\mathbf{A}\right) \tag{6.11b}$$

$$\mathbf{H}_k = (1-\theta_k)\,\mathbf{M}'\left(\mathbf{B}'\mathbf{I}_{k+1}\mathbf{A} + \mathbf{H}_{k+1}\mathbf{A}\right) \tag{6.11c}$$

$$\mathbf{F}_k = (1-\theta_k)\,\mathbf{M}'\left(\mathbf{B}'\mathbf{I}_{k+1}\mathbf{B} + \mathbf{H}_{k+1}\mathbf{B} + \mathbf{B}'\mathbf{H}_{k+1} + \mathbf{F}_{k+1}\right)\mathbf{M} \tag{6.11d}$$

$$\hat{\mathbf{R}}_k = \mathbf{R}_k + (1-\theta_k)\,\mathbf{M}'\hat{\mathbf{R}}_{k+1}\mathbf{M} \tag{6.11e}$$

$$\mathbf{L}_k^* = \left(\mathbf{B}'\mathbf{F}_{k+1}\mathbf{B} + \hat{\mathbf{R}}_k\right)^{-1}\mathbf{B}'\mathbf{H}_{k+1}\mathbf{A} \tag{6.11f}$$

and $\mathbf{P}_N = \mathbf{Q}_N, \mathbf{I}_N = \mathbf{Q}_N, \mathbf{H}_N = \mathbf{0}_{r\times n}, \mathbf{F}_N = \hat{\mathbf{R}}_N = \mathbf{0}_{r\times r}$.

Proof. In this output strategy, it must be taken into account the fact that the cost-to-go function includes an extra term that does not depend on u_k, which is applied whenever $\theta_k = 0$. Hence, the cost-to-go function can be written as

$$V_k(x_k;\theta_k, z_{k-1}) = x_k'\mathbf{Q}_k x_k + (1-\theta_k)\Big(z_{k-1}'\mathbf{M}'\mathbf{R}_k\mathbf{M}z_{k-1} + V_{k+1}(x_{k+1};\theta_{k+1}, \mathbf{M}z_{k-1})\Big) + \theta_k\Big(u_k'\mathbf{R}_k u_k + \theta_{k+1}x_{k+1}'\mathbf{P}_{k+1}x_{k+1}(1-\theta_{k+1})V_{k+1}(x_{k+1};\theta_{k+1}, u_k)\Big). \tag{6.12}$$

Obviously, the term with $1-\theta_k$ cannot be minimized by u_k. However, Equation (6.12) applies to $V_i, k < i \leq N$ and in such cases, it is useful to consider the terms that were not minimized upon the computation of the respective u_i, i.e., expansions of the term $V_{k+1}(x_{k+1}; 1-\theta_{k+1}, u_k)$. Recall also, that these equations are solved backwards in time, starting at $k = N$ and finishing at $k = 0$, which implies that these expansions will in essence consider the fact that depending on the value of θ at present/future instants, a value applied at a given instant may be reapplied, up to a linear function, in later instants.

The proof of the theorem *per se*, is done by induction: $V_N(x_N) = x_N'\mathbf{Q}_N x_N$ according to the theorem and is the cost-to-go in the last step. Given that the previous step, which due to the fact that the cost is minimized backwards in time is actually the next step, held. Under the conditions of the theorem plus the structure of V_k as shown in Equation (6.12), then

$$V_k(x_k;\theta_k,z_{k-1}) = x_k'\mathbf{Q}_k x_k + \theta_k\Big(u_k'\mathbf{R}_k u_k + \theta_{k+1}(\mathbf{A}x_k + \mathbf{B}u_k)'\mathbf{P}_{k+1}(\mathbf{A}x_k + \mathbf{B}u_k) + (1-\theta_{k+1})\big(V_{k+1}(\mathbf{A}x_k + \mathbf{B}u_k;\theta_{k+1}, u_k)\big)\Big) + (1-\theta_k)\Big(z_{k-1}'\mathbf{M}'\mathbf{R}_k\mathbf{M}z_{k-1} + V_{k+1}(\mathbf{A}x_k + \mathbf{B}\mathbf{M}z_{k-1};\theta_{k+1}, \mathbf{M}z_{k-1})\Big) \tag{6.13}$$

which with the recursive expansion of all the terms enclosed within $V_i, \forall_i k < i \leq N$, yields

$$V_k(x_k;\theta_k, z_{k-1}) = x_k'\mathbf{Q}_k x_k + (1-\theta_k)\Big(z_{k-1}'\mathbf{M}'\mathbf{R}_k\mathbf{M}z_{k-1} + V_{k+1}(x_{k+1};\theta_{k+1}, \mathbf{M}z_{k-1})\Big) + \theta_k\sum_{i=k}^{N-1}\left[\prod_{j=k+1}^{i}(1-\theta_j)\right]\Big(u_k'\Big(M^{i-k}\Big)'R_i M^{i-k}u_k + \theta_{i+1}x_{i+1}'P_{i+1}x_{i+1} + (1-\theta_{i+1})x_{i+1}'Q_{i+1}x_{i+1}\Big). \tag{6.14}$$

Once more, the terms in θ_j (as opposed to terms in $1-\theta_j$) were not expanded because they are encapsulated into $\mathbf{P}_{i+1}$. By further expanding x_{i+1} (for the case in which $\theta_k = 1$ and $\theta_{[k+1,i]} = 0$) into

$$x_i = \mathbf{A}^{i-k}x_k + \sum_{j=k}^{i-1} \mathbf{A}^{i-j-1}\mathbf{B}\mathbf{M}^{j-k}u_k \tag{6.15}$$

leads Equation (6.14) into Equation (6.16)

$$\begin{aligned} V_k(x_k;\theta_k,z_{k-1}) = x_k'\mathbf{Q}_k x_k + (1-\theta_k)(z_{k-1}'\mathbf{M}'\mathbf{R}_k\mathbf{M}z_{k-1} + V_{k+1}(x_{k+1};\theta_{k+1},\mathbf{M}z_{k-1}))+ \\ \theta_k \sum_{i=k}^{N-1}\left[\prod_{j=k+1}^{i}(1-\theta_j)\right]\bullet \\ \left(u_k'(\mathbf{M}^{i-k})'\mathbf{R}_i\mathbf{M}^{i-k}u_k + \left(\mathbf{A}^{i-k}x_k + \sum_{j=k}^{i-1}\mathbf{A}^{i-j-1}\mathbf{B}\mathbf{M}^{j-k}u_k\right)'(\theta_{i+1}\mathbf{P}_{i+1}+(1-\theta_{i+1})\mathbf{Q}_{i+1})\left(\mathbf{A}^{i-k}x_k + \sum_{j=k}^{i-1}\mathbf{A}^{i-j-1}\mathbf{B}\mathbf{M}^{j-k}u_k\right)\right). \end{aligned} \tag{6.16}$$

Equation (6.16) itself can be rewritten into

$$\begin{aligned} V_k(x_k;\theta_k,z_{k-1}) = x_k'\mathbf{Q}_k x_k + (1-\theta_k)\Big(z_k'\mathbf{R}_k z_k + V_{k+1}(x_{k+1};\theta_{k+1},z_{k-1})\Big) \\ \theta_k\Big(u_k'\hat{\mathbf{R}}_k u_k + (\mathbf{A}x_k)'\mathbf{I}_{k+1}\mathbf{A}x_k + (\mathbf{B}u_k)'\mathbf{H}_{k+1}\mathbf{A}x_k + \\ (\mathbf{A}x_k)'\mathbf{H}_{k+1}'\mathbf{B}u_k + (\mathbf{B}u_k)'\mathbf{F}_{k+1}\mathbf{B}u_k\Big), \end{aligned} \tag{6.17}$$

with

$$\hat{\mathbf{R}}_k = \sum_{i=k}^{N-1}\left[\prod_{j=k+1}^{i}(1-\theta_j)\right](\mathbf{M}^{i-k})'\mathbf{R}_i M^{i-k} \tag{6.18a}$$

$$\mathbf{I}_k = \sum_{i=k}^{N-1}\left[\prod_{j=k+1}^{i}(1-\theta_j)\right](\mathbf{A}^{i-k})'\Big(\theta_{i+1}\mathbf{P}_{i+1}+(1-\theta_{i+1})\mathbf{Q}_{i+1}\Big)\mathbf{A}^{i-k} \tag{6.18b}$$

$$\mathbf{H}_k = \sum_{i=k}^{N-1}\left[\prod_{j=k+1}^{i}(1-\theta_j)\right]\left(\sum_{j=k}^{i-1}\mathbf{A}^{i-j-1}\mathbf{B}\mathbf{M}^{j-k}\right)'\Big(\theta_{i+1}\mathbf{P}_{i+1}+(1-\theta_{i+1})\mathbf{Q}_{i+1}\Big)\mathbf{A}^{i-k} \tag{6.18c}$$

$$\begin{aligned} \mathbf{F}_k = \sum_{i=k}^{N-1}\left[\prod_{j=k+1}^{i}(1-\theta_j)\right]\left(\sum_{j=k}^{i-1}\mathbf{A}^{i-j-1}\mathbf{B}\mathbf{M}^{j-k}\right)' \\ \Big(\theta_{i+1}\mathbf{P}_{i+1}+(1-\theta_{i+1})\mathbf{Q}_{i+1}\Big)\left(\sum_{j=k}^{i-1}\mathbf{A}^{i-j-1}BM^{j-k}\right). \end{aligned} \tag{6.18d}$$

All these variables have been defined in a more economic manner beforehand. Taking the derivative of V_k (Equation (6.17)) in order to $\mathrm{u_k}$ and equating it to zero yields

$$\frac{\partial J_k(x_k;\theta_k,z_{k-1})}{2\partial u_k} = \theta_k\Big(u_k'\hat{\mathbf{R}}_{\mathbf{k}} + (\mathbf{A}x_k)'\mathbf{H}_k'\mathbf{B} + (\mathbf{B}u_k)'\mathbf{F}_k\mathbf{B}\Big) = \mathbf{0}_{1\times r}. \tag{6.19}$$

Subtracting an appropriate form of equation (6.19) from (6.17) yields

$$V_k(x_k;\theta_k,z_{k-1})-\frac{\partial J_k(x_k;\theta_k,z_{k-1})}{2\partial u_k}\Big|_{u_k=u_k^*}u_k^*=(1-\theta_k)\Big(z_k'\mathbf{R}_k z_k+V_{k+1}(x_{k+1};\theta_{k+1},z_{k-1})\Big) \\ +x_k'\mathbf{Q}_k x_k+\theta_k\Big((\mathbf{A}x_k)'\mathbf{I}_{k+1}+(\mathbf{B}u_k)'\mathbf{H}_{k+1}\Big)\mathbf{A}x_k, \tag{6.20}$$

from which, substituting u_k^* for $-\mathbf{L}_k^* x_k$ and taking expectations over θ_k makes the term of V_k that appears multiplied by θ_k independent of u_k. Furthermore, such term is only multiplied by x_k and x_k' and it coincides with the definition of $\mathbf{P}_k$. By expanding the terms multiplied by $1-\theta_k$ and remembering that, under such circumstances $z_k=\mathbf{M}z_{k-1}$, concludes the proof with respect to V_k.

Regarding the optimal controller gain, starting from equation (6.19), transposing and grouping terms with u_k^* and x_k, yields

$$\theta_k\left(\mathbf{B}'\mathbf{F}_k\mathbf{B}+\hat{\mathbf{R}}_k\right)u_k^*+\theta_k\mathbf{B}'\mathbf{H}_k\mathbf{A}x_k=\mathbf{0}_{r\times 1} \tag{6.21}$$

in which it is assumed that $\theta_k=1$, since otherwise the cost function does not depend on u_k. Sending the second term to the right-side and considering the definition of L_k^* concludes the proof. □

6.5.2 Differential Matrix Derivation

Theorem 3. *The cost-to-go function of control in a lossy network is given by*

$$V_k(x_k)=\frac{1}{2}\begin{bmatrix}x_k\\ z_{k-1}\end{bmatrix}'\begin{bmatrix}\mathbf{P}_{xx,k}^- & \mathbf{P}_{xu,k}^-\\ \mathbf{P}_{ux,k}^- & \mathbf{P}_{uu,k}^-\end{bmatrix}\begin{bmatrix}x_k\\ z_{k-1}\end{bmatrix} \tag{6.22a}$$

$$\begin{bmatrix}\mathbf{P}_{xx,k}^+ & \mathbf{P}_{xu,k}^+\\ \mathbf{P}_{ux,k}^+ & \mathbf{P}_{uu,k}^+\end{bmatrix}=\hat{\mathbf{A}}'\begin{bmatrix}\mathbf{P}_{xx,k+1}^- & \mathbf{P}_{xu,k+1}^-\\ \mathbf{P}_{ux,k+1}^- & \mathbf{P}_{uu,k+1}^-\end{bmatrix}\hat{\mathbf{A}}+\begin{bmatrix}\mathbf{Q}_k & \mathbf{0}_{n\times r}\\ \mathbf{0}_{r\times n} & \mathbf{R}_k\end{bmatrix} \tag{6.22b}$$

$$\hat{\mathbf{A}}=\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0}_{r\times n} & \mathbf{I}_n\end{bmatrix} \tag{6.22c}$$

$$\begin{bmatrix}\mathbf{P}_{xx,k}^- & \mathbf{P}_{xu,k}^-\\ \mathbf{P}_{ux,k}^- & \mathbf{P}_{uu,k}^-\end{bmatrix}=\begin{bmatrix}\mathbf{P}_{xx,k}^+-\theta_k\mathbf{P}_{xu,k}^+\mathbf{L}_k^* & (1-\theta_k)\mathbf{P}_{xu,k}^+\mathbf{M}\\ (1-\theta_k)\mathbf{M}'\mathbf{P}_{ux,k}^+ & (1-\theta_k)\mathbf{M}'\mathbf{P}_{uu,k}^+\mathbf{M}\end{bmatrix} \tag{6.22d}$$

$$\mathbf{L}_k^*=\left(\mathbf{P}_{uu,k}^+\right)^{-1}\mathbf{P}_{ux,k}^+, \tag{6.22e}$$

and

$$\begin{bmatrix}\mathbf{P}_{xx,N}^- & \mathbf{P}_{xu,N}^-\\ \mathbf{P}_{ux,N}^- & \mathbf{P}_{uu,N}^-\end{bmatrix}=\begin{bmatrix}\mathbf{Q}_N & \mathbf{0}_{n\times r}\\ \mathbf{0}_{r\times n} & \mathbf{0}_{r\times r}\end{bmatrix}. \tag{6.23}$$

Proof. Since the cost function is quadratic it is always possible to rewrite it in the form

$$V_k(x_k,z_{k-1})=\frac{1}{2}\begin{bmatrix}x_k\\ z_{k-1}\end{bmatrix}'\frac{\mathrm{d}^2J_k}{\mathrm{d}\begin{bmatrix}x_k\\ z_{k-1}\end{bmatrix}\mathrm{d}\begin{bmatrix}x_k\\ z_{k-1}\end{bmatrix}'}\begin{bmatrix}x_k\\ z_{k-1}\end{bmatrix}. \tag{6.24}$$

The derivative is carried out in order to J_k instead of $V_k(x_k)$ because

$$V_k(x_k, z_{k-1}) \stackrel{def}{=} \min_{u_k} J_k(x_k, u_k, z_{k-1}).$$

Hence, V_k is not a function of u_k. In fact, the definition of V_k assumes that u_k is equal to a given value u_k^*. Continuing, deriving Equation (6.12) (twice) in order to $\begin{bmatrix} x_k' & z_{k-1}' \end{bmatrix}'$ using the chain rule leads to

$$\frac{\partial J_k(x_k)}{\partial x_k} = 2x_k' Q_k + \frac{\partial J_{k+1}(x_{k+1})}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial x_k} \tag{6.25a}$$

$$\frac{\partial J_k(x_k)}{\partial z_{k-1}} = 2\theta_k z_k' R_k + \frac{\partial J_{k+1}(x_{k+1})}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial z_k} + \frac{\partial J_{k+1}(x_{k+1})}{\partial z_k} \frac{\partial z_k}{\partial z_k}, \tag{6.25b}$$

taking the derivative of these two equations in order to x_k and z_k lead to (the derivative in order to $\partial x_k \partial z_k'$ was omitted due to the symmetry of the equations)

$$\frac{\partial^2 J_k(x_k)}{\partial x_k \partial x_k'} = 2Q_k + \left(\frac{\partial x_{k+1}}{\partial x_k}\right)' \frac{\partial^2 J_{k+1}(x_{k+1})}{\partial x_{k+1} \partial x_{k+1}'} \frac{\partial x_{k+1}}{\partial x_k} \tag{6.26a}$$

$$\frac{\partial^2 J_k(x_k)}{\partial z_k \partial x_k'} = \left(\frac{\partial x_{k+1}}{\partial z_k}\right)' \frac{\partial^2 J_{k+1}(x_{k+1})}{\partial x_{k+1} \partial x_{k+1}'} \frac{\partial x_{k+1}}{\partial x_k} + \left(\frac{\partial z_k}{\partial z_k}\right)' \frac{\partial J_{k+1}(x_{k+1})}{\partial z_k \partial x_{k+1}'} \frac{\partial x_{k+1}}{\partial x_k} \tag{6.26b}$$

$$\begin{aligned} \frac{\partial^2 J_k(x_k)}{\partial z_k \partial z_k'} = &\left(\frac{\partial x_{k+1}}{\partial z_k}\right)' \frac{\partial^2 J_{k+1}(x_{k+1})}{\partial x_{k+1} \partial x_{k+1}'} \frac{\partial x_{k+1}}{\partial z_k} + \left(\frac{\partial z_k}{\partial z_k}\right)' \frac{\partial^2 J_{k+1}(x_{k+1})}{\partial z_k \partial x_{k+1}'} \frac{\partial x_{k+1}}{\partial z_k} + \\ &\left(\frac{\partial x_{k+1}}{\partial z_k}\right)' \frac{\partial^2 J_{k+1}(x_{k+1})}{\partial x_{k+1} \partial z_k'} \frac{\partial z_k}{\partial z_k} + \left(\frac{\partial z_k}{\partial z_k}\right)' \frac{\partial^2 J_{k+1}(x_{k+1})}{\partial z_k \partial z_k'} \frac{\partial z_k}{\partial z_k} + 2R_k. \end{aligned} \tag{6.26c}$$

The last set of equations is recursive in nature. This fact can be used to write down a more economical version of them, starting with the substitution

$$\hat{A} \stackrel{def}{=} \frac{\partial \begin{bmatrix} x_{k+1} \\ z_k \end{bmatrix}}{\partial \begin{bmatrix} x_k \\ z_k \end{bmatrix}} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0}_{r \times n} & \mathbf{I}_r \end{bmatrix}, \tag{6.27}$$

the vector being differentiated (in the numerator) as a term in z_k as opposed to z_{k+1} because (as it will be discussed below) the act of producing a control value transforms the expression in z_{k+1} into an expression in z_k. Define also

$$\hat{\mathbf{P}}_k^- \stackrel{def}{=} \begin{bmatrix} \mathbf{P}_{xx,k}^- & \mathbf{P}_{xu,k}^- \\ \mathbf{P}_{ux,k}^- & \mathbf{P}_{uu,k}^- \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \frac{\partial^2 J_k}{\partial x_k' \partial x_k} & \frac{\partial^2 J_k}{\partial x_k' \partial z_k} \\ \frac{\partial^2 J_{k+1}}{\partial z_k' \partial x_k} & \frac{\partial^2 J_k}{\partial z_k' \partial z_k} \end{bmatrix}. \tag{6.28}$$

Due to the structure of $\hat{\mathbf{P}}_k^-$, the relation $\mathbf{P}_{xu,k}^{\pm} = (\mathbf{P}_{ux,k}^{\pm})'$ must hold. Putting it all together, the recursion becomes

$$\hat{\mathbf{P}}_k^+ = \begin{bmatrix} \mathbf{Q}_k & \mathbf{0}_{n \times r} \\ \mathbf{0}_{r \times n} & \mathbf{R}_k \end{bmatrix} + \hat{\mathbf{A}}' \hat{\mathbf{P}}_{k+1}^- \hat{\mathbf{A}}. \tag{6.29}$$

Once more, it must be stressed that, due to the quadratic nature of the cost function (before minimization $J_k(x_k, z_k)$ is a function of $\hat{\mathbf{P}}_k^+$)

$$J_k(x_k) = \frac{1}{2} \begin{bmatrix} x_k \\ z_k \end{bmatrix}' \hat{\mathbf{P}}_k^+ \begin{bmatrix} x_k \\ z_k \end{bmatrix}, \tag{6.30}$$

which taking the derivative in order to u_k and remembering that $\hat{\mathbf{P}}_k^+$ is not a function of u_k leads to

$$\frac{\partial J_k(x_k)}{\partial u_k} = \begin{bmatrix} x_k \\ z_k \end{bmatrix}' \hat{\mathbf{P}}_k^+ \begin{bmatrix} \mathbf{0}_{n\times r} \\ \theta_k \mathbf{I}_r \end{bmatrix} = \mathbf{0}_{1\times r} \tag{6.31}$$

which is solved into

$$u_k = -\left(\mathbf{P}_{uu,k}^+\right)^{-1} \mathbf{P}_{ux,k}^+ x_k, \tag{6.32}$$

which implicitly defines the controller gain $\mathbf{L}_k$. Remembering that $z_k = \theta_k u_k + (1-\theta_k)\mathbf{M} z_{k-1}$ and as just shown $u_k^* = \mathbf{L}_k^* x_k$, this can be substituted into (6.30) from which performing the matrix multiplication taking into account that $\theta_k(1-\theta_k) = 0$ and Equation (6.32) and grouping terms in x_k yields

$$V_k(x_k) = \frac{1}{2} \begin{bmatrix} x_k \\ z_{k-1} \end{bmatrix}' \hat{\mathbf{P}}_k^- \begin{bmatrix} x_k \\ z_{k-1} \end{bmatrix} \tag{6.33}$$

due to the idempotency of the product in Boolean algebra, i.e., $(1-\theta_k)(1-\theta_k) = (1-\theta_k)$,

$$\hat{\mathbf{P}}_k^- = \begin{bmatrix} \mathbf{P}_{xx,k}^+ - \theta_k \mathbf{P}_{xu,k}^+ \left(\mathbf{P}_{uu,k}^+\right)^{-1} \mathbf{P}_{ux,k}^+ & (1-\theta_k)\mathbf{P}_{xu,k}^+ \mathbf{M} \\ (1-\theta_k)\mathbf{M}'\mathbf{P}_{ux,k}^+ & (1-\theta_k)\mathbf{M}'\mathbf{P}_{uu,k}^+ \mathbf{M} \end{bmatrix}. \tag{6.34}$$

□

As referred above, the computation of u_k (due to the fact that it is a function of x_k) transformed Equation (6.30) with terms in z_k into Equation (6.33) with terms in z_{k-1}. The terms in z_k disappear after substituting them for their respective definition, which introduces terms in u_k and z_{k-1}. The terms in u_k are substituted for the control law leaving only terms in x_k and z_{k-1}.

6.6 Discussion

Two important notes are in order: first, previous sections presented two proofs of the optimal controller over hold lossy networks. The control laws that came out of these two sections appear to be different. However, the various sub-matrices of the *Hessian* matrix that appear in the second proof are related according to

$$\hat{\mathbf{P}}_k^- = 2 \begin{bmatrix} \mathbf{I}_k & \mathbf{H}_k \\ \mathbf{H}_k' & \mathbf{F}_k + \hat{\mathbf{R}}_k \end{bmatrix}. \tag{6.35}$$

The factor of two in the last equation stem from the slightly different cost functions used in each proof. The last equality can be established by a simply comparison of the values of the respective variables.

Second, in general this control law tracks a matrix of size $(n+r)\times(n+r)$, which is larger than the $\mathbf{P}_k$ matrix that is tracked in classical optimal control and in control over lossy networks with the output zero strategy. Once more, this seems to be a discrepancy. However, this apparent discrepancy stems from the (borrowing from the estimation lexicon) correction part, i.e., the mechanism that updates $\hat{\mathbf{P}}_k$ once the value of u_k is known. Since in the (generalized) hold case, before the correction the cost function is a function of the z_k. However, after the correction, once the control value is known, it is possible to write the cost

function as a function of z_{k-1}, by expanding the definition of z_k. Hence, the remaining cost function is a function of z_{k-1} and obviously, x_k. This is in contrast with centralized control, as well as in the zero strategy, once u_k is found and joined to the terms in x_k, z_k becomes equal to zero. Hence, the three sub-matrices associated with it become unnecessary.

This fact implies that only $\mathbf{P}_{xx,k}$ needs to be tracked. This has profound implications, one of them being the lowering of the effective rank of the other approaches.

It was pointed out in Section 6.3 that the introduction of cost terms relating to the correlation between the input and the output would not change the structure of the optimal controller. That statement can now be substantiated by noting that by repeating the proofs presented herein under such circumstances Equation (6.29) becomes

$$\hat{\mathbf{P}}_k^+ = \begin{bmatrix} \mathbf{Q}_k & \mathbf{N}_k \\ \mathbf{N}_k' & \mathbf{R}_k \end{bmatrix} + \hat{\mathbf{A}}'\hat{\mathbf{P}}_{k+1}^-\hat{\mathbf{A}} \tag{6.36}$$

whereas Equation (6.32) and Equation (6.34) are not transformed.

As pointed out before, this control strategy generalizes the types of actuator outputs. Hence, it is no surprise that for $\mathbf{M} = \mathbf{0}_{r\times r}$ it is equal to the controller presented in [SSF+07] and reproduced in Section 6.4.

6.6.1 Domain of Application

Even though the novel optimal controller was derived under a number of strict conditions, i.e., no noise/error of any kind, the optimal control law has a wider domain of application: substituting the controller derived on this chapter on certain systems that do not comply with the error/perturbation assumptions still yields the optimal control law. These systems can be found, for instance, by substituting the derived control law and checking for their optimality.

This control law can be trivially extended into the case of a (perfectly, e.g. full state measurement) known state but with state disturbances, see Equation (6.37), since the effects of w_k cannot be minimized by u_k.

$$\begin{aligned} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k + w_k \\ y_k &= \mathbf{C}x_k. \end{aligned} \tag{6.37}$$

More generally, if the separation principle holds, then by definition the derived controller will be optimal. This fact alone implies that this controller can be used in TCP-like network protocols, i.e., network protocols in which the delivery status is always known with a low latency by the sender.

6.7 Optimal Actuator State Transition Matrix

In Section 6.5, it was derived the optimal control over a lossy network, in which, whenever there was no new control message, the actuator applied a linear function which included the multiplication of the previous applied value by a constant matrix $\mathbf{M}$. However, the optimality was only with respect to u_k, i.e., on the controller side.

This section deals with the optimal value of the matrix $\mathbf{M}$ and, of course, its effects on the optimal value that the actuator should apply. In the previous sections, it was shown that z_k, hence both $\mathbf{M}_k z_{k-1}$ and u_k, only has a bearing on the cost function starting from Equations (6.29)-(6.30). Finding an optimal $\mathbf{M}_k$ implies the minimization of such equations in order

to both u_k and $\mathbf{M}_k$. However, the minimum over u_k is already known from past sections. Furthermore, due to the boolean nature of the θ variables, the derivative in order to u_k does not have terms in $\mathbf{M}_k$ and vice-versa.

Using the standard method for determining extremum points, a derivative of Equation (6.29) is taken in order to $\mathbf{M}_k$ and equated to zero. It is known that such extremum is a minimum because Equation (6.29) is convex in z_k, with a semi-positive definite Hessian. Continuing,

$$\frac{\partial J_k\left(x_k, z_{k-1}\right)}{2\partial \mathbf{M}_k} = z_{k-1}\begin{bmatrix} x_k \\ z_k \end{bmatrix}' \hat{\mathbf{P}}_k^+ \begin{bmatrix} \mathbf{0}_{n\times r} \\ (1-\theta_k) I_r \end{bmatrix} = \mathbf{0}_{r\times r}, \tag{6.38}$$

which can be extended into

$$(1-\theta_k) z_{k-1}\left(z_k' P_{uu,k}^+ + x_k' P_{xu,k}^+\right) = \mathbf{0}_{r\times r}. \tag{6.39}$$

Applying the distributive property of multiplication to the term $(1-\theta_k)$, especially to z_k, which, once more, due to the boolean nature of the variables in question, does not have any term in u_k. It yields, after dropping $(1-\theta_k)$ for being a common term

$$z_{k-1}\left(\left(\mathbf{M}_k^* z_{k-1}\right)' \mathbf{P}_{uu,k}^+ + x_k' \mathbf{P}_{xu,k}^+\right) = \mathbf{0}_{r\times r}. \tag{6.40}$$

It is possible to drop the term in $(1-\theta_k)$, which is equivalent to assume that $\theta_k =$'0', because whenever $\theta_k =$'1' the value of $\mathbf{M}_k$ does not matter, since in this case it will be applied the value sent by the controller. Continuing, by right multiplying the last equation by $\left(\mathbf{P}_{uu,k}^+\right)^{-1}$ and transposing the result, it becomes

$$\left(\mathbf{M}_k^* z_{k-1} + \left(\mathbf{P}_{uu,k}^+\right)^{-1} \mathbf{P}_{ux,k}^+ x_k\right) z_{k-1}' = \mathbf{0}_{r\times r} \tag{6.41}$$

which, according to Equation (6.32), i.e., the equation of the optimal control value u_k^*, implies that

$$\left(\mathbf{M}_k^* z_{k-1} - u_k^*\right) z_{k-1}' = \mathbf{0}_{r\times r}. \tag{6.42}$$

Equation (6.42) shows that $\mathbf{M}_k^*$ is a projection matrix that provides the optimal estimate of u_k^* given z_{k-1}, which is an unsurprising result, since intuitively, it was already expected that $\mathbf{M}_k^* z_{-1}$ was some approximation of u_k^*. $\mathbf{M}_k^*$ can be readily computed from the equations that describe the evolution of the system.

Just as $\mathbf{L}_k^*$ in the controller, $\mathbf{M}_k^*$ can be preprogrammed into the actuator in order to reduce the online computational load. Furthermore, the equations that describe both $\mathbf{L}_k^*$ and $\mathbf{M}_k^*$ converge to constant matrices.

Regarding the value of $V_k\left(x_k, z_{k-1}\right)$ given $\mathbf{M}_k^*$, Equation (6.34) which defines the Hessian of $V_k\left(x_k, z_{k-1}\right)$ still holds. However, two of the terms of Equation (6.34) with $\mathbf{M}_k$ cancel each other, more concretely:

$$\begin{aligned}\left(\mathbf{M}_k z_{k-1}\right)' \mathbf{P}_{ux,k}^+ x_k + x_k' \mathbf{P}_{xu,k}^+ \mathbf{M}_k z_{k-1} + \left(\mathbf{M}_k z_{k-1}\right)' \mathbf{P}_{xu,k}^+ \mathbf{M}_k z_{k-1} = \\ \left(\mathbf{M}_k z_{k-1}\right)' \mathbf{P}_{ux,k}^+ x_k + x_k' \mathbf{P}_{xu,k}^+ \mathbf{M}_k z_{k-1} + \left(\mathbf{M}_k z_{k-1}\right)' \mathbf{P}_{xu,k}^+ \mathbf{M}_k z_{k-1} \\ + \left(\mathbf{M}_k z_{k-1}\right)' \mathbf{P}_{xu,k}^+ \mathbf{M}_k z_{k-1} - \left(\mathbf{M}_k z_{k-1}\right)' \mathbf{P}_{xu,k}^+ \mathbf{M}_k z_{k-1}.\end{aligned} \tag{6.43}$$

The last equation has the trace of Equation (6.41) (and its transpose) times $\mathbf{M}_k^*$, which is zero by definition of $\mathbf{M}_k^*$. This implies that the first four terms, on the right side, of the previous equation are equal to $\mathbf{0}_{r\times r}$. Hence, the Hessian of the $V_k(x_k, z_{k-1})$ given that $\mathbf{M}_k = \mathbf{M}_k^*$ is

$$\hat{\mathbf{P}}_k^- = \begin{bmatrix} \mathbf{P}_{xx,k}^+ - \theta_k \mathbf{P}_{xu,k}^+ \left(\mathbf{P}_{uu,k}^+\right)^{-1} \mathbf{P}_{ux,k}^+ & \mathbf{0}_{n\times r} \\ \mathbf{0}_{r\times n} & -(1-\theta_k)\mathbf{M}_k^{*\prime}\mathbf{P}_{uu,k}^+\mathbf{M}_k^* \end{bmatrix}. \tag{6.44}$$

6.7.1 Computation of $\mathbf{M}_k^* z_{k-1}$

Equation (6.42) provides a condition for the optimality of $\mathbf{M}_k$. However, it requires two vectors u_k and z_{k-1} which can be a problem because 1) there is more than one value of $\mathbf{M}_k$ that satisfy the condition and 2) $\mathbf{M}_k$ is used when u_k is not known.

Hence, it is necessary to find a general formula for $\mathbf{M}_k^*$ that can be numerically computed and plugged into the equations derived herein.

A possible choice $\mathbf{M}_k^*$ that verifies Equation (6.42) is

$$\mathbf{M}_k^* = \mathbf{L}_k^* \left(\mathbf{A}\mathbf{L}_{k-1}^{*\dagger} - \mathbf{B}\right) \tag{6.45}$$

The choice in Equation (6.45) is rather parochial. It is based on 1) assuming that $\theta_{k-1} =$'1', which, *per se*, does not cause a loss of generality, 2) since, $u_{k-1} = -\mathbf{L}_k^* \hat{x}_{k|k}$, then it is assumed that $\hat{x}_{k|k} = -\mathbf{L}_{k-1}^{*\dagger} \mathrm{u}_{k-1}$, in which the $(\cdot)^\dagger$ stands for pseudo-inverse. Obviously, this assumption produces only an estimate of the actually state and is the only assumption which is not general, 3) it is applied the dynamic equation, i.e., multiplying $\hat{\mathrm{x}}_{k-1|k-1}$ by $\mathbf{A} - \mathbf{B}\mathbf{L}_k^*$ to produce $\hat{x}_{k|k-1}$ and finally 4) multiply the state by the control law $\mathbf{L}_k^*$.

Putting together all the steps outlined in the last paragraph leads to

$$\mathbf{M}_k^* = \mathbf{L}_k^* \left(\mathbf{A} - \mathbf{B}\mathbf{L}_{k-1}^*\right) \mathbf{L}_{k-1}^{*\dagger}. \tag{6.46}$$

Since $\mathbf{M}_k$ is always right multiplied by control type vectors, for instance it is right multiplied by z_{k-1} as in $\mathbf{M}_k z_{k-1}$. Such vectors are built by left multiplication by the matrix $\mathbf{L}_{k-1}$. For instance, here we assume u_{k-1} because it is assumed that $\theta_{k-1} =$'1', but if $\theta_{k-1} =$'0', then this would still be true since then z_{k-1} would have been used and $z_{k-1} = \mathbf{M}_{k-1} z_{k-2}$ and $\mathbf{M}_{k-1}$ is left multiplied by $\mathbf{L}_{k-1}^*$ as can be seen in (6.45) and (6.46). Hence,

$$\begin{aligned}
\mathbf{M}_k^* u_{k-1} =& \mathbf{L}_k^* \left(\mathbf{A} - \mathbf{B}\mathbf{L}_{k-1}^*\right) \mathbf{L}_{k-1}^{*\dagger} \mathbf{L}_{k-1}^* \hat{x}_{k-1|k-1} && \text{(6.47a)} \\
=& \mathbf{L}_k^* \left(\mathbf{A}\mathbf{L}_{k-1}^{*\dagger}\mathbf{L}_{k-1}^* - \mathbf{B}\mathbf{L}_{k-1}^*\mathbf{L}_{k-1}^{*\dagger}\mathbf{L}_{k-1}^*\right) \hat{x}_{k-1|k-1} && \text{(6.47b)} \\
=& \mathbf{L}_k^* \left(\mathbf{A}\mathbf{L}_{k-1}^{*\dagger}\mathbf{L}_{k-1}^* - \mathbf{B}\mathbf{L}_{k-1}^*\right) \hat{x}_{k-1|k-1} && \text{(6.47c)} \\
=& \mathbf{L}_k^* \left(\mathbf{A}\mathbf{L}_{k-1}^{*\dagger} - \mathbf{B}\right) \mathbf{L}_{k-1}^* \hat{x}_{k-1|k-1} && \text{(6.47d)} \\
=& \mathbf{L}_k^* \left(\mathbf{A}\mathbf{L}_{k-1}^{*\dagger} - \mathbf{B}\right) u_{k-1} && \text{(6.47e)}
\end{aligned}$$

in which the first equality follows from the control law, the second equality follows by applying the distributive property to the two terms on the left of $\hat{x}_{k-1|k-1}$. The third equality holds because for any matrix $\mathbf{S}$, $\mathbf{S}\mathbf{S}^\dagger\mathbf{S} = \mathbf{S}$. The fourth equality follows by putting the controller gain in evidence. The last equation, which coincides with Equation (6.45), follows by applying the control law.

Equation (6.45) has a problem, from a computational standpoint, since, it requires knowledge of $\mathbf{L}^*_{k-1}$. But $\mathbf{L}^*_{k-1}$ is computed based on $\mathbf{P}^+_{k-1}$ (Equation (6.32)) which is computed based on $\mathbf{P}^-_k$ (Equation (6.29)) which is computed based on $\mathbf{M}^*_k$ (Equations (6.34) and (6.44)). This means that these equations are actually not closed. But they can still be solved recursively, in a cycle in which at a given time the values of $\mathbf{M}^*_k$, with $k = 0 \dots N$, are assumed to be known and the values of $\mathbf{L}^*_k$, $k = 0 \dots N$, are computed and then use the freshly computed values of $\mathbf{L}^*_k$, $k = 0 \dots N$, to compute new values of $\mathbf{M}^*_k$, $k = 0 \dots N$, so on and so forth, until both sequences, $\mathbf{M}$ and $\mathbf{L}$, converge.

A different and more elaborate choice of $\mathbf{M}^*_k$ can be achieved by noting that Equation (6.45) uses a relatively poor choice of estimate of $\hat{x}_{k|k}$. A better estimate of $\hat{\mathrm{x}}_{k|k}$ can be obtained by the use of an observer or a Kalman filter. This approach would have the advantage that the information about the state acquired in a given instant could be used on a latter moment, thus improving the general quality of the estimate, as measured, for example, by its associated error covariance matrix. In a scenario without any perturbations, the actuator could acquire full knowledge of $\hat{x}_{k|k}$, thus becoming capable of computing the control values equal to the ones computed by controller.

On more realistic scenarios, such as those in which a Kalman filter would be applied, its innovation step should be

$$\hat{\hat{x}}_{k+1|k} = \mathbf{A}\hat{\hat{x}}_{k|k} + \mathbf{B}z_k \tag{6.48a}$$

$$\hat{\mathbf{P}}_{k+1|k} = \mathbf{A}\hat{\mathbf{P}}_{k|k}\mathbf{A}' + \hat{\mathbf{Q}}_k \tag{6.48b}$$

and the correction step should be

$$\hat{\mathbf{K}}_{k+1} = \theta_{k+1}\hat{\mathbf{P}}_{k+1|k}{\mathbf{L}^*_{k+1}}' \left(\mathbf{L}^*_{k+1}\hat{\mathbf{P}}_{k+1|k}{\mathbf{L}^*_{k+1}}'\right)^{-1} \tag{6.49a}$$

$$\hat{\mathbf{P}}_{k+1|k} = \left(\mathbf{I}_n - \hat{\mathbf{K}}_{k+1}\mathbf{L}^*_{k+1}\right)\hat{\mathbf{P}}_{k|k} \tag{6.49b}$$

$$\hat{\hat{x}}_{k+1|k+1} = \hat{\hat{x}}_{k+1|k} - \hat{\mathbf{K}}_{k+1}\left(u_{k+1} + \mathbf{L}^*_{k+1}\hat{\hat{x}}_{k+1|k+1}\right). \tag{6.49c}$$

These equations are the result of the application of a Kalman filter to estimate $\hat{x}_{k|k}$, the state estimate at the controller, based on the received control sequence up to that moment. The estimate of $\hat{x}_{k|k}$ is denoted $\hat{\hat{x}}_{k|k}$. Due to the manner in which the estimated variable is transmitted to the output, with the output, normally denoted y_k, is in this case u_k, and the output matrix, normally denoted $\mathbf{C}$, is in this case $-\mathbf{L}^*_k$.

Note that the matrix $\hat{\mathbf{K}}_{k+1}$ (with a $\hat{\cdot}$) to distinguish it from the Kalman gain used in the controller) differs from traditional definitions by a minus sign, that results from the three minus signs coming from the $\mathbf{L}^*_{k+1}$.

The sampling error auto-covariance matrix, normally denoted $\mathbf{R}_k$ is equal to $\mathbf{0}_{r\times r}$ because the output can be read perfectly, since it is delivered in a message that is assumed that whenever it is received, it is received without errors. Note that usual sources of errors, such as the quantization error, do not contribute to $\hat{\hat{x}}_{k|k}$ because the quantization errors happened at measuring y_k, which is the output of the classical Kalman filter, whereas the output of the current filter, u_k is read without any quantization error, i.e., when it is received, it is equal to the value sent by the controller. Summarizing, there is a need to take into account reading errors in the classical Kalman filter because the measurement are not a perfect representation of the output; it is not necessary to do the same in this circumstances because the readings are equal to the system output.

The matrix $\hat{\mathbf{Q}}_k$ is, according to the definitions of Kalman filters applied to this instance, the auto-covariance matrix of $\hat{x}_{k|k} - \hat{x}_{k|k-1}$, which can be readily computed.

Evidently, the controller system that results from the application of a Kalman filter to compute $\mathbf{M}_k^* z_{-1}$ is governed by a different set of output equations, that is

$$
\begin{aligned}
z_k &= \theta_k u_k - (1-\theta_k)\mathbf{L}_k^* \hat{\hat{x}}_{k|k} && \text{(6.50a)}\\
&= -\theta_k \mathbf{L}_k^* \hat{x}_{k|k} - (1-\theta_k)\mathbf{L}_k^* \hat{\hat{x}}_{k|k} && \text{(6.50b)}\\
&= -\mathbf{L}_k^* \left(\theta_k \hat{x}_{k|k} + (1-\theta_k)\left(\hat{\hat{x}}_{k|k} + \hat{x}_{k|k} - \hat{x}_{k|k}\right)\right) && \text{(6.50c)}\\
&= -\mathbf{L}_k^* \left(\hat{x}_{k|k} + (1-\theta_k)\left(\hat{\hat{x}}_{k|k} - \hat{x}_{k|k}\right)\right) && \text{(6.50d)}\\
&= u_k - (1-\theta_k)\mathbf{L}_k^* \left(\hat{\hat{x}}_{k|k} - \hat{x}_{k|k}\right) && \text{(6.50e)}
\end{aligned}
$$

in which the first equality stems from the definition of the applied control value in the absence of a new control message. Similarly, the second equality follows by the application of the control law to the control value produced by the controller. The third equality follows from grouping the terms as to leave $\mathbf{L}_k$ in the left side and adding/subtracting $\hat{x}_{k|k}$ to in the actuator estimation group. The fourth equality stems from the grouping of two terms in $\hat{x}_{k|k}$, one with θ and another with $1-\theta$. The last equality follows by applying the control law to the controller estimate.

Due to the use of optimal estimators, such as the Kalman filter that was employed, the last term of Equation (6.50e) has zero mean and, more importantly, is not correlated with the first. This property, when combined with the certainty equivalence and separation principles implies that $\mathbf{L}_k^*$ is equal to the value controller gain matrix that would be computed on a centralized network. This fact sets the stage for a possible UDP-like implementation of this type of control, since in this type of system the certainty equivalence and separation principles hold.

6.8 Comparison with the Block Transmission Architecture

On one hand, in the block transmission architecture, as presented on Chapters 4 and 5, all future values were computed by the controller and sent to the actuator. This had the advantage of requiring only one computationally intensive network element. It had a possible disadvantage of requiring too many network resources, but as was pointed out before, it is becoming less important with the emergence of fieldbuses with large minimum payloads.

On the other hand, the use of an optimal controller over a lossy network with a generalized linear output, such as the scheme presented in this chapter, allows the actuator to obtain estimates of the controller variable even in the absence of a new control message, which was one of the early motivations for the use of the architecture. The controller approach also has the advantage of not needing to rely on the sizes of the minimum allowed payload of the fieldbus. Obviously, this approach imposes higher computational loads on actuators.

However, as referred above, in a sense the values applied in the controller approach are an estimate of the values applied when the architectural approach is used. This fact is an indication that the former approach produces a control qualitatively inferior to the latter. Other aspects of interest include, for example:

- the fact that a full implementation of this type of control would require an actuator with estimation and control capabilities. In fact, the various network elements behave hierarchically, with the controller as the top element and the sensor and actuator acting as lower level equipments.

- the control strategy emerged out of successive generalizations and optimizations of the types of outputs, starting with the zero *vs* hold discussion and ending with the actuator also holding a state variable, though strictly speaking, the hold strategy already has a state variable associated with the output, i.e., z_k. The new state variable is associated with the system state and is an estimate of the state estimate that is held by the controller.

- it cannot be avoided the discussion regarding the possibilities of either having the controller sending its state estimate instead of control values and the possibility of having the sampler send its readings directly to the actuator, i.e. a controller embedded in the actuator, thereby eliminating the need for this type of controller. The *pros* and *cons* of each possibility may be the object of study of a future work.

- in a related note, based on the system at hand, questions regarding resources, for example, the computational capabilities of the actuator, may dictate what is the best approach to pursue, i.e., a zero vs hold, an optimal linear generalization with a varying $\mathbf{M}_k^*$ or with a constant steady state optimal matrix $\mathbf{M}_{ss}^*$, with $\mathbf{M}_{ss}^* = \mathbf{M}_\infty^*$, or with a state based realization plus a Kalman filter.

6.9 Evaluation

This section evaluates the controller proposed in this chapter. To this end, an unstable system was chosen at random. The cost function, defined according to cost matrices that are defined below, are evaluated for a set of linear control laws during a finite amount of time. The controller gain matrix that minimizes the cost function is compared to the theoretically derived steady state controller gain matrix. The agreement between these values confirms the optimality of the proposed control law.

Note, however, that it was only tested for $\mathbf{M}_k = \mathbf{I}_r$. This choice was made because testing the optimality of the choice of $\mathbf{M}_k$ would require performing many tests similar to the one presented in this section. However, due to timing constraints it will not be possible to include such results on this Thesis.

6.9.1 A First Order System

The system had a state-space representation

$$x_{k+1} = [1.5]\, x_k + [1]\, u_k \tag{6.51}$$

$$y_k = [1]\, x_k \tag{6.52}$$

which means that it is unstable with a discrete pole at $z = 1.5$. The cost function was

$$J_k = x_{300}^2 + \sum_{k=0}^{299} \left(x_k^2 + 0.7 u_k^2 \right). \tag{6.53}$$

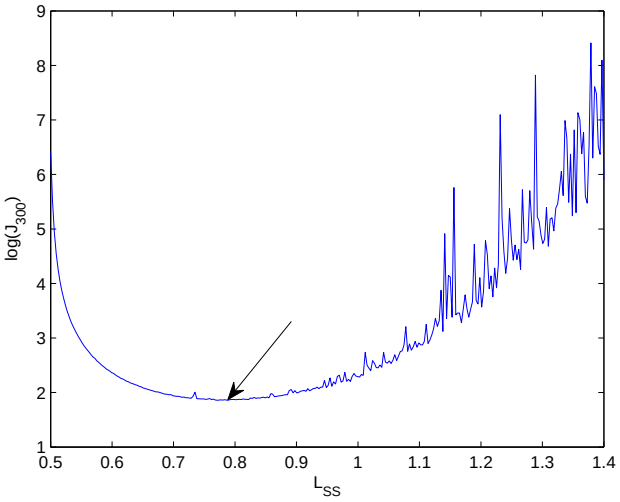


Figure 6.1: Controller Gain versus Cost.

The controller $\mathbf{L}_k$, as in $u_k = -\mathbf{L}_k x_k$, was chosen from 300 linearly spaced points in the interval $[0.5\ , 1.4]$. To smooth out the inherent variance of θ, $5 \cdot 10^6$ simulations were made for each value of $\mathbf{L}_k$. Each simulation took 300 steps. All simulations were initialized with $x_0 = 1$. The simulations were made with $\bar{\theta} = \frac{3}{4}$. All simulations were performed without any type of noise or perturbation. Figure 6.1 depicts the results of such simulations. A logarithmic scale was used in the y-axis to allow a proper visualization of the many orders of magnitude that are spawned by the graph.

There is an asymmetry between the error behavior on the left side, i.e., close to $\mathbf{L}_{ss} = .5$, and the behavior on the right side, i.e., $\mathbf{L}_{ss} = 1.4$, regarding the variance of the curve, since, as can be seen, in the former case, the curve is rather well behaved, whereas in the later case the curve is rather jagged, having many peaks and valleys. This behavior stems from the fact that when $\mathbf{L}_{ss} = .5$ the closed-loop system has a pole close to z = 1, which in turn implies that the control value changes slowly. Hence, the various simulations that differ only by the message error sequence have similar cost, contributing to the observed low variance. Such effect is not present on the right end of the graph.

As can be seen, there is a perfect agreement between theory (i.e., $\mathbf{L} = 0.78$) and practice (the simulation results).

6.10 Conclusion and Future Work

The problem of optimal linear control as well as the problem of control in networks that output zero in the absence of a new control value has already been solved and can be found in the literature. The present chapter solved the problem of (linear) control in systems in which the current output value is computed by a linear function of the last applied control value, in case the present control value is unknown.

A solution for the optimal controller given such generalized output scheme was derived and for each possible output scheme it is generated an appropriate optimal controller. Such optimal controller left open the question of what is the linear function that yields the smallest cost.

A necessary and sufficient condition for the optimality of the linear function was also presented. Moreover, a mechanism to compute such optimal controllers was devised. Such controller was then compared with the controller of the architecture presented in earlier chapters.

Simulation results are in perfect agreement with the optimal control gain computed theoretically.

Chapter 7

Oscillation Free Controller Change

7.1 Introduction and Motivation

The extraordinary development of microprocessors not only made ubiquitous the use of control networks on control systems as discussed in previous chapters, but it also made digital control the dominant control systems approach. When compared with conventional analog controllers, digital controllers of a given quality are usually less expensive, present higher flexibility and adaptability to different applications, are more scalable and can perform several functions concurrently, not all necessarily related to control. Analog compensators, on the other hand, are usually developed in an application specific manner and can hardly be reused for other purposes.

Control systems are typically subject to strong, and often conflicting, constraints, for example, relating to their jitter, message delivery, etc. On top of that, there is the fact that digital control processes are usually not the only task being executed by the respective processor, which often executes a multitude of tasks. As seen on Chapter 3, in order to guarantee that all tasks meet their requirements at each and every moment, these systems are designed according to their worst case requirements. However, designing systems according to the worst-case requirements of each task may lead to expensive and inefficient designs, for a myriad of reasons. For example, there are cases in which it is possible to guarantee that the tasks never enter in overload simultaneously, hence the requirements of such tasks is smaller than the sum of the requirements of each individual task.

A number of approaches that make a more efficient use of system resources, in which task sets are not designed for the worst case requirements were put forward in the literature. Instead, each task runs at a given level. These level are set such that under normal conditions they lead to an underloaded system. Whenever a task requires more resources, it is given enough resources to successfully complete its current job, as discussed in Section 7.2. Obviously, this assumes that tasks will not simultaneously make a request for more resources.

In the specific case of control systems, many such approaches rely on controller changes, taking advantage of the monotonically increasing relationship between control performance and resource utilization, as discussed on Sections 2.3.3 and 2.3.4 and references therein. One of the techniques that has been extensively investigated consists in using several controllers, designed to have different resource needs, and consequently, to exhibit different performance levels. Therefore, less resource demanding, albeit less efficient, controllers are used whenever the system error is small, while more resource demanding and more efficient controllers are

selected when the system experiences a large error. The last comment was one example of a situation that triggers a controller switch. Others examples include an overload situation (possibly caused by the switching of the remaining tasks) or because the controller is at the moment the plant that is being controlled has a different importance to the global system.

However, the contributions found in the literature have been essentially focused on the rules for triggering the controller switching, like some of the approaches for resource optimization presented in Chapter 3, but neglected the full extent that such changes have in the system. That is, notwithstanding the aforementioned advantages of using controllers that can commute, the controllers in question are designed for static operation.

In particular, switching controllers often causes output oscillations that may negate the potential performance gains. More concretely, in the controller change methodologies hitherto proposed in the literature, controller transitions are almost always followed by an output oscillation. At best, these oscillations degrade the quality of control. However, it may easily happen that such oscillations in turn cause another controller change, which itself triggers yet another oscillation and so on and so forth. One example of this effect is provided in Section 6.9. Consequently, systems with multiple controllers tend to oscillate between two or more controllers, even when the system conditions would otherwise be stationary, thus degrading the quality of the control and wasting resources.

In this chapter, firstly, the cause of oscillations in the presence of period changes is presented. Then, it is presented a solution, based in a change of basis matrix. An evaluation section is also presented and it shows that important performance gains are achieved in key control performance indicators such as overshoot, settling time and error.

This chapter intends to provide control system designers with the necessary theoretical tools to predict and avoid oscillations that occur in systems that employ controller changes.

7.2 Related Work

The study of switching controllers began decades ago, hence it enjoys a vast body of literature. However, the study of oscillation free transitions remained a fringe topic between both the control and scheduling communities. This fact rendered its associated literature not so vast. This section provides an overview of the scientific contributions that are both close and relevant to the work presented in this chapter.

In [APM05] it is discussed the use of several controllers, each tuned to a given period. In their approach, the scheduler then chooses a controller of a given period according to the error level. This idea was somehow complemented in [CVMC10], with a study of optimal transition error levels for each controller, though the former study was centered around distributed systems whereas the latter was focused on centrally scheduled systems. Further work in this area includes the investigation of when a number of pre-calculated controllers outperform dynamically chosen ones, in terms of computational load, allowed periods, response time and related metrics.

Several studies, for example [CMVC06] and more prominently [AT09], discussed whether the controller changes should be based in instantaneous or in filtered/accumulated measures of error. Other studies, such as [CMVC06], favor instantaneous measures, based in a number of experiments that show that the advantages of filtering do not compensate the additional computational burden. In [AT09] it is presented a similar experiment, although with different systems, and concludes that filtering was worthwhile. This apparent contradiction certainly

deserves a deeper study, which, however, it is out of the scope of this Thesis.

In [CE00] and more recently in [CEBÅ02] the authors present a rather different approach. Co-design is (re)introduced with a new set of goals and paradigms. Each controlled process has a given weight and the goal of the scheduler is to assign the periods in a way that minimizes the sum of the weights times the squared errors. More formally, let $f(\tau_j) = \sum_{k \in \tau_j} w_k e_k^2$ be the total error, given that it was chosen the schedule τ_j, i.e., the j possible parameter attribution of the task set and under such parameter assignment, the k^{th} process has error e_k and weight w_k. Then the chosen schedule is:

$$\tau = arg \min \left(f(\tau_j) \right). \tag{7.1}$$

However, in [BVM07] it is proposed to minimize a slightly different quantity which the authors call the quality of control. Nevertheless, the mechanism is somewhat similar to its predecessors, hence it may produce similar parameter assignment to the tasks of the task set, and it may also present itself to be more difficult to minimize.

Using the CAN protocol, [AT09] discusses the advantages of period switching at the network level, thus approaching a co-design perspective. In [LMVF08b] and references therein, a discussion regarding feedback scheduling is presented. Feedback scheduling can be loosely defined as a scheduling methodology in which the task's current characteristics (e.g., period, worst-case execution time) are defined based on current operating conditions.

The problem of scheduling such processes has also made significant advances, one of the most relevant ones being the elastic task model [BLCA02]. Drawing from an analogy of strained springs, in which the sum of the displacements is equal to the total displacement and the displacement of each string is inversely proportional to its elastic coefficient, in this method the task's utilization may be adjusted to have a given utilization level. Each task has an individual weight that controls its relative level of adaptivity. This model is general enough to be ported to the control scenario without major modifications. Nonetheless, it was further generalized in [CHL06] for encompassing other types of minimization. It must be pointed out that in all such studies it is assumed that there is a mechanism for correctly/optimally choose the task's weights. However, to the best of our knowledge, there are no studies addressing this issue.

In [VMF+10] non-periodic controllers are presented. The controllers are executed and then, based on the current state and error, the next activation instant is selected, i.e., it is self-triggered. However, this method has a few drawbacks and limitations. One of the most important ones is that it implies a centralized architecture, since the controller must have a new sample at a moment that, in a distributed system, is not available. Additionally, even though a number of simulations were presented, it is not clear that there is no periodic controller that achieves the same quality of control. Furthermore, its erratic activation pattern, which was further discussed in [MVB09], renders its associated schedulers far more complex.

7.3 Controller Adaptation through Period Switch

7.3.1 Types of Controller Change

In general, controller changes are performed whenever the current controller, according to some criteria, is not the most suitable one to deal with the current situation. Different controller characteristics can be modified. To systematize the presentation, the type of controller adaptations are categorized into three non-simultaneously exclusive classes: period

change, order change and complexity (type) change. The number of quantization bits can also be changed in order to change the quality of control. However, it is not included in this analysis because current computational systems does not allow to perform much of a change in this regard. For example, 7-bit and 8-bit quantizations are likely to use the same amount of resources. Any controller change technique should fit in at least one of these classes.

A **period change** is the most basic type of change and possibly the most common, as verified by the authors in the literature survey and references presented therein. One of the main reasons for such widespread use is the fact that this type of change leads to systems in which the schedulability is relatively easy to analyze using common schedulability tests. In this type of controller change, there is a change in the sample and actuation period of the controller, accompanied by a change in the controller's parameters. However, usually nothing is done regarding the system state at the switching instants, which, as will be seen later, may cause oscillations. A typical example is the change from a PID controller at a relatively high sample rate, during instants of high system dynamics, to a relatively low sample rate during periods of low system dynamics.

An **order change** is typical of systems with singularities in regions of the $z-$plane that at certain times have a low or negligible effect in the output. This situation happens, for example, in a system with a poorly canceled pole $-$ zero pair. Another example is industrial machinery with two functional modes, one of them including a high frequency, low response time, output. In principle, when such machinery is in a *slower* mode it could use a lower order model that does not take into account the higher frequency singularities.

A **complexity change** in the controller is also possible. An example would be changing between a PID and a state space controller with an observer and a full-state feedback controller. Note that what is being changed is the controller itself. This is by far the option with fewer references in the literature.

It is important to stress that these categories are not disjoint. A single controller change may belong to more than one category. For example, a change from a PID at 17ms to a state-space (Kalman filter plus full-state feedback) at 11ms is both a complexity and a period change.

As mentioned above, period change is arguably the controller change class most commonly reported in the literature, hence, it is logical to start the subject of oscillation free controller change with this approach. There are other, more subtle, reasons for starting with this approach, namely for the case of order change, the concept of similar state after a state transition, especially when the order is increased, is not well defined.

7.3.2 Period Switch

During its normal operation, discrete-time PID or state-space controllers save a number of past inputs or a linear combination of them, which are normally state variables. These state variables are used to derive the control value. In fact, as discussed on Section 2.2, the state variables are invariant under similarity transformations, which, in essence, transforms one representation of a given system state into another representation of the same system state. Note that system state denotes the physical state, whereas the state variables denote a series of variables, i.e., numbers, that are stored in the control system to represent the physical state.

Upon a controller change, though neither the system state nor the state variable change, it is possible that a similarity transformation takes place. Or, stating it more bluntly, the

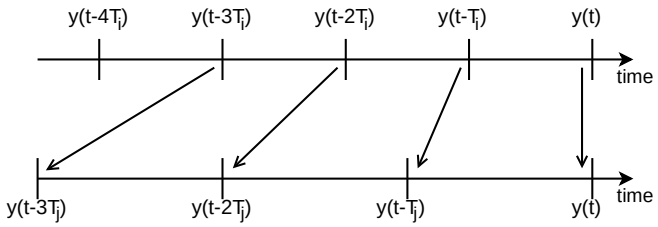


Figure 7.1: State in classical controller change.

mapping from the physical domain to the representation may vary even if neither the physical domain nor the representation do vary. Obviously, if that happens, the state variable after the controller change will correspond to a system state that is not necessarily equal to the physical state, which is the core of the problem.

To illustrate the problem, consider a controller with a period T_i and N state variables. At a given instant t there is a period change and the new period is T_j. In the former case the state includes estimates of $[y(t)\ y(t-T_i)\ \cdots\ y(t-(N-1)T_i)]$. However, when the controller transition occurs, the state representation does not automatically change to point to $[y(t)\ y(t-T_j)\ \cdots\ y(t-(N-1)T_j)]$ (see Fig. 7.1). Since the controller is now tuned for the latter state, it will produce sub-optimal control values until all old state values are replaced by new ones, spaced apart by T_j.

It is elucidative to note that a controller that saves the values of the last N positions will oscillate whenever the controller changes, whereas a controller that saves both the positions and its $N-1$ successive derivatives, i.e., first derivative, second derivative, etc..., does not oscillate upon a controller change, since at the switching instant the state already points to the right place. For example, in the previous example, if successive derivatives were used, then all of them would already point to the current sample after the controller change.

To overcome the identified problem, at the switching instants the state must be updated in order to point to the corresponding values in the new time base. One possible way to carry out this conversion is to move forward/backward in time, using the system's continuous time equations. Obviously, some previous input values would have to be stored in order to make possible to perform such computations. Despite being conceptually simple, this approach implies a relatively high processing overhead, which may be a serious problem, since most digital control systems are normally subject to stringent resource constraints. To overcome such high computational load problem, it was developed a new methodology presented in Subsection 7.3.4, which requires less computations at runtime.

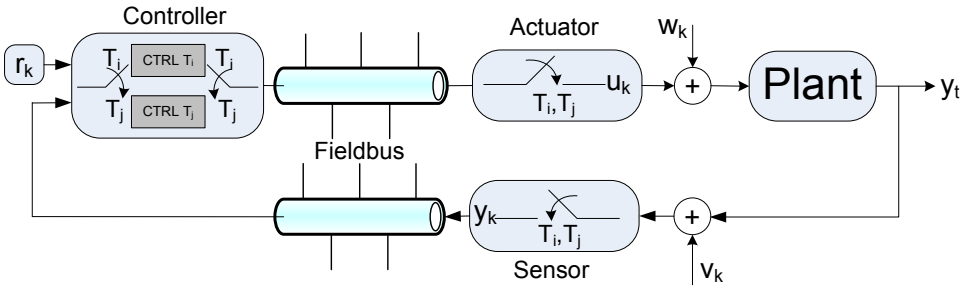


Figure 7.2: Control architecture.

Γ	feedback controller
k	discrete-time variable
t	continuous-time variable
T	period of a function
r	reference signal
u	input variable
w	input noise
v	sampling noise
x	state space variable
y	output variable
$\boldsymbol{A}$	state transition matrix
$\boldsymbol{B}$	input matrix
$\boldsymbol{C}$	output matrix
n	system order
$\boldsymbol{I}_n$	identity matrix, i.e., n × n matrix with 1 in the diagonal entries and 0 elsewhere
$\lambda\{\boldsymbol{X}\}$	set of eigenvalues of matrix $\boldsymbol{X}$
m_i	multiplicity of the i^{th} eigenvalue of a matrix
$\boldsymbol{P}$	change of basis matrix
SYS	a representation of he system (plant) state
[vec]	matrix vectorization operator

Table 7.1: Notation (this chapter only)

7.3.3 Problem Formulation

This chapter addresses the problem of feedback control systems, possibly distributed, as depicted in Figure 7.2. It is assumed that the control system incorporates more than one controller for each physical system. Each such controller is further assumed to have been designed in order to minimize an arbitrary cost function (e.g. energy, CPU utilization, network utilization) and that it uses a state-space representation. Each controller Γ_i has an associated sampling rate (T_i), to which corresponds a state-space representation SYS_i. Table 7.1 presents the notation used in the chapter. Some elements of Table 7.1 had been previously defined, however, for consistency reasons, all variables used in this chapter are (re)defined.

Let $\Gamma_i \equiv \{\{\text{SYS}_i, T_i\}, i \in \{1, 2, ..., n\}\}$ be a set of feedback controllers for SYS. It is assumed that all controllers Γ_i are properly designed and can stabilize system SYS. Furthermore, any controller Γ_i may be used during the system lifetime, although at any time instant one and only one controller is active. The problem of selecting the best controller, selecting the optimum switching instants or designing the controllers, are outside of the scope of this chapter. Methods described in the literature (e.g. [APM05], [CVMC10]) can be used to this end.

Without loss of generality, consider that at an arbitrary time instant t it is requested a change of controller from Γ_i to Γ_j. The associated state-space representations, SYS_i and SYS_j, are respectively,

$$x_i(k+1) = \boldsymbol{A}_i x_i(k) + \boldsymbol{B}_i u(k) \tag{7.2a}$$

$$y(k) = \boldsymbol{C}_i x_i(k) \tag{7.2b}$$

and

$$x_j(k+1) = \boldsymbol{A}_j x_j(k) + \boldsymbol{B}_j u(k) \tag{7.3a}$$
$$y(k) = \boldsymbol{C}_j x_j(k) \tag{7.3b}$$

The problem thus consists in devising a mechanism to switch from SYS_i to SYS_j without oscillation, i.e., finding the value x_j in the space of SYS_j that corresponds to the state x_i in SYS_i at any arbitrary time.

As an additional requirement, the online part of such mechanism must be lightweight, both in terms of processing and memory, since the state conversions must be performed during runtime, upon controller changes, in resource-constrained hardware.

7.3.4 Proposed Solution

The solution that is proposed in this chapter consists, firstly, in defining an auxiliary representation. The application of the similarity transformation of such auxiliary representation, first on the transition matrix and then on the input/output matrix, leads to a Sylvester and to a simpler linear equation, respectively. The resulting Sylvester equation is homogeneous, thus classic solving methods are inadequate, leading to the use of the Kronecker product. From this process it is extracted an ensemble of matrices that verify the transition matrix condition. Furthermore, any non-singular linear combination of such matrices is also a solution. Finally, it is searched for a linear combination of the matrix ensemble that also verifies the input/output matrix similarity equation. The remaining of this section presents, in detail, each one of these steps.

To carry out the state conversion, first, consider an auxiliary representation SYS_{ja}, which is sampled at rate T_j but its continuous time version has a state equal to the state of SYS_i. Such representation can be found, for instance, by discretizing the continuous-time version of SYS_i with period T_j. By construction, SYS_{ja} can be written as

$$x_{ja}(k+1) = \boldsymbol{A}_{ja} x_{ja}(k) + \boldsymbol{B}_{ja} u(k) \tag{7.4a}$$
$$y(k) = \boldsymbol{C}_i x_{ja}(k) \tag{7.4b}$$

The introduction of SYS_{ja} turns the original problem into a simpler one, i.e., a change of basis problem. Since, by design, $x_i = x_j a$ and both SYS_{ja} and SYS_j are sampled at the same rate, there should be a matrix $\boldsymbol{P}$ such that $\text{x}_j = \boldsymbol{P} x_{ja}$. The existence and uniqueness of such matrix is a standard result [dM10].

To find $\boldsymbol{P}$, it can be used the Sylvester equation [1]: $\boldsymbol{A}_{ja} = \boldsymbol{P}^{-1}\boldsymbol{A}_j\boldsymbol{P}$, or

$$\boldsymbol{P}\boldsymbol{A}_{ja} - \boldsymbol{A}_j\boldsymbol{P} = \boldsymbol{0} \tag{7.5}$$

However, classical efficient approaches, e.g. [BS72, GNVL79], fail to give satisfactory solutions to this particular problem, since they always return the trivial solution, i.e., $\boldsymbol{P} = \boldsymbol{0}$. It is noteworthy that this equation also appears in robust pole placement techniques, as attested by [Var00] and references therein. In fact, this mechanism is so widespread that it is even used in standard commercial products such as the `Matlab`® function `place`. However, the mechanisms used to solve it in that context are not easily portable to this problem.

[1] $\boldsymbol{A}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{B} = \boldsymbol{C}$ in which $\boldsymbol{A} = -\boldsymbol{A}_j$, $\boldsymbol{B} = \boldsymbol{A}_{ja}$ and $\boldsymbol{C} = \boldsymbol{0}$.

Therefore, it was opted to use the Kronecker product approach. In this approach, Equation (7.5) is transformed into:

$$\left(\boldsymbol{A}_{ja}^{T} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}_j\right) vec(\boldsymbol{P}) = \boldsymbol{0} \tag{7.6}$$

where vec($\boldsymbol{P}$) is a vectorized version of the matrix $\boldsymbol{P}$ and $\boldsymbol{I}_n$ is the $(n \times n)$ identity matrix.

The vectorization operation as described by the vec($\boldsymbol{X}$), is a matrix operation that stacks column wise, i.e., in order of the column number, the columns of matrix $\boldsymbol{X}$ into a single column, i.e.

$$vec([x_1\, x_2\, \cdots\, x_{n-1}\, x_n]) = \left[x_1'\, x_2'\, \cdots\, x_{n-1}'\, x_n'\right]' \tag{7.7}$$

in which $\mathrm{x_i}$ is the i^{th} column of $\boldsymbol{X}$.

Equation (7.6) is an eigen problem, associated with the eigenvalue 0. Eigen problems are the problems of finding the vectors that when applied to a given function, return an amplified (by a factor equal to the corresponding eigenvalue) version of the vector in question.

The solution of the eigen problem induced by Equation (7.6) is a series of vectors that, when passed back into the initial matrix space (the inverse of vec), gives rise to a series of eigen matrices to which any non-singular linear combination is also a solution of Equation (7.5). This stems from a known result of linear algebra, which states that any linear combination of eigenvectors associated with the same eigenvalue is also an eigenvector associated with the eigenvalue in question.

This principle is combined to the fact that any eigenvector associated with the eigenvalue 0 is a solution of Equation (7.5), implying that any linear combination of a solution of Equation (7.6) is also a solution of Equation (7.5).

Regarding the dimensionality of the space of eigen matrices that are solutions of Equation (7.6), naïvely it could be said that Equation (7.6) in general does not have any eigen matrix associated with the eigenvalue 0. However, it is well known that the eigenvalues of $(\boldsymbol{A} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{B})$ (with $\boldsymbol{A}, \boldsymbol{B}$ square matrices of size n) are $\lambda_i(\boldsymbol{A}) - \lambda_j(\boldsymbol{B})$, where $\lambda_i(\boldsymbol{X})$ is the i^{th} eigenvalue of matrix $\boldsymbol{X}$. In this particular case, $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same eigenvalues due to the fact that they are each other's transposes, hence each eigenvalue of $\boldsymbol{A}$ produces m_i^2 zero eigenvalues in the matrix $\boldsymbol{P}$, where m_i is the multiplicity of the i^{th} eigenvalue. Thus, $\boldsymbol{P}$ has an eigenvalue of 0 with multiplicity $\sum_{i=1}^{q} m_i^2 \geq n$, where q is the number of distinct eigenvalues of $\boldsymbol{A}$.

Due to the similarity transformation, matrix $\boldsymbol{P}$ must verify (note that there was no particular reason to apply the similarity transformation to the input matrices, i.e., it could as well be applied to the output matrices)

$$\boldsymbol{P}\boldsymbol{B}_{ja} = \boldsymbol{B}_j. \tag{7.8}$$

Equations (7.5) (or (7.6)) and (7.8) define matrix $\boldsymbol{P}$, though, as is discussed below, they may not define a single matrix. More precisely, the first equation defines a rank three (3-dimensional) tensor, that is conveniently written as a number of eigen matrices, as pointed out above, and the product of such matrix by matrix $\boldsymbol{B}_{ja}$ can be written as:

$$\boldsymbol{P}\boldsymbol{B}_{ja} = \left(\sum_i \omega_i \boldsymbol{P}_i\right) \boldsymbol{B}_{ja}, \tag{7.9}$$

where $\boldsymbol{P}_i$ are the eigen matrices as computed from Equation (7.6) and ω_i are unknown coefficients. Determining the unknown coefficients would implicitly solve the problem. The

unknown coefficients are determined using classical linear methods, which are only applicable if the known tensor has rank 3, i.e., is a matrix. Therefore, define now $s_i \equiv \text{vec}(\boldsymbol{P}_i \boldsymbol{B}_{ja})$. Then (7.8) and (7.9) can be rewritten as a classical linear problem as

$$vec(\boldsymbol{B}_j) = [s_1\, s_2 \cdots s_z] \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_z \end{bmatrix} \tag{7.10}$$

or simply, $\text{vec}(\boldsymbol{B}_j) = \boldsymbol{S\Omega}$, were $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ are implicitly defined. The previous equation provides the values of $\boldsymbol{\omega}$ that give the actual solution, i.e., solution of both Equations (7.5) and (7.8). Furthermore, $\boldsymbol{P} = \sum_i \omega_i \boldsymbol{P}_i$.

Algorithm 1 provides a summarized description of the procedure that determines the change of basis matrix. The operation inv_vec used in the algorithm is the inverse of *vec*. $\boldsymbol{S}^{-1}$ may not exist, as discussed above. Since this happens only if the system of equations that define $\boldsymbol{P}$ is over-determined, a proper pseudo-inverse may be used instead. The case of multiple inputs (or outputs if matrix $\boldsymbol{C}$ is used instead) is handled seamlessly by the algorithm, since in fact the pair $(\{\boldsymbol{A}_{ja}, \boldsymbol{B}_{ja}\}, \{\boldsymbol{A}_j, \boldsymbol{B}_j\})$ are different representation of the same system (at the same rate), in which case it is guaranteed that $\text{vec}(\boldsymbol{B}_j)$ is on the space spanned by $\boldsymbol{S}$. Nonetheless, this situation must also be dealt with using the pseudo-inverse of $\boldsymbol{S}$.

It follows that if $\boldsymbol{P}$ switches from SYS_i to SYS_j, then $\boldsymbol{P}^{-1}$ switches from SYS_j to SYS_i. This fact implies that only one of these matrices need to be computed by Algorithm 1. The other one can be computed by matrix inversion.

$$x_j = \boldsymbol{P} x_i, \tag{7.11a}$$

$$x_i = \boldsymbol{P}^{-1} x_j. \tag{7.11b}$$

These formulae apply to a transition between any two periods. In systems with N periods that would need to be $N-1$ such matrices. Each of these would be from a given period, say T_0, for any other period T_i. Its $N-1$ instead of N, because the matrix that changes from a given period (in this case T_0) to itself is the Identity matrix ($\boldsymbol{I}_n$). The matrix for changing between any two periods T_i and T_j can be found by first changing from T_i to T_0, i.e., left-multiplying by $\boldsymbol{P_i^{-1}}$, and then changing from T_0 to T_i, i.e., left-multiplying by $\boldsymbol{P}_j$. Hence, for the global change, it is multiplied by $\boldsymbol{P}_j \boldsymbol{P}_i^{-1}$.

The derivation that culminated with Algorithm 1 did not consider the size of $\boldsymbol{B}$ ($\boldsymbol{C}$), more precisely, the number of inputs (outputs). However, varying such number does not add

Algorithm 1 Algorithm for computing the change of basis matrix

K $\leftarrow \left(\boldsymbol{A}_{ja}^T \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}_j\right)$
$\boldsymbol{ns} \leftarrow$ nullspace(K)
for all Columns of $\boldsymbol{ns}$ **do**
  $\boldsymbol{S}_i \leftarrow vec\,(inv\_vec(\boldsymbol{ns}_i)\boldsymbol{B}_{ja})$
end for
$\boldsymbol{\Omega} \leftarrow \boldsymbol{S}^{-1} vec(\boldsymbol{B}_j)$
$\boldsymbol{P} \leftarrow \boldsymbol{ns} \times \Omega$

to the complexity of the solution, since the use of the *vec* operator already converts matrices of all sizes into a single framework, of which, the solution is presented in the aforementioned algorithm.

It is important to stress that a vector of ones may be an eigenvector of $\boldsymbol{P}$ associated to the eigenvalue 1. For example, if the steady-state of a variable-phase representation is in the space spanned by such vector, i.e., if the state variable is comprised of precisely (current and) past values of the outputs, then whenever the output is stable, a change of period will not change the state variable. Whenever this happens, there is no need to perform these operations, since $\boldsymbol{P}x = x$. This can be generalized to other situations, which is the prime motive why the systems considered in, for example, [VMB08] did not oscillate, even though no compensation was made.

7.3.5 Complexity of the methodology

The proposed methodology comprises two complementary steps. Firstly, once the controllers are designed, $\boldsymbol{P}$ is computed using Algorithm 1. This step is carried out off-line, during system design and most likely in a non-constrained hardware.

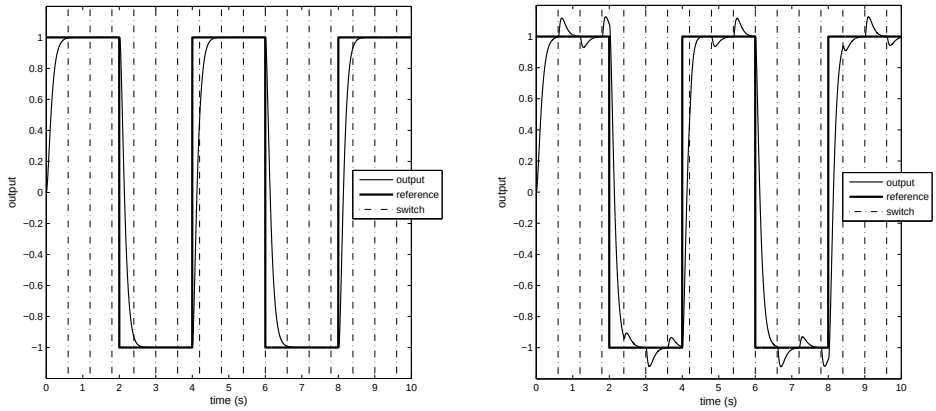
The complexity of Algorithm 1 is dominated by the complexity of the Singular Value Decomposition algorithm, which is used to compute the null space of $\left(\boldsymbol{A}_{ja}^T \otimes \boldsymbol{I}_n - \boldsymbol{A}_j \otimes \boldsymbol{I}_n\right)$. This step can also be done using the eigenvector decomposition instead. However, due to its inherent numerical instability and to the fact that these computations are carried out off-line, as discussed above, the eigenvector decomposition option was not adopted. SVD has a complexity $\mathrm{O}(m^3)$, where m is the number of rows (columns) of the matrix, which in this case is square. Thus, as the system order is n, the overall complexity becomes $\mathrm{O}(n^6)$. The situation can be ameliorated e.g. with the use of QR-decomposition (with column pivoting due to the singularity of the matrix). The algorithm's execution time was measured in an Acer Desktop PC, featuring an Intel Core 2 Quad Q6600 CPU at 2.40 GHz. For all systems reported in Section 6.9, the execution time was found to be smaller than the system's clock resolution (3.9 ms).

The actual change of basis, triggered by controller changes, is performed online, using Equations (7.11a) and (7.11b). This operation implies only a simple matrix product involving a square matrix and a column vector of corresponding size. Hence, the change of basis computation is lightweight, both in terms of processing power and memory, being implementable even in resource-constrained hardware.

7.3.6 Orthogonality with respect to control and scheduling

In both control systems and real-time communities, there are many different methodologies addressing a myriad of specific problems. Frequently, such methodologies are incompatible with others, thus restricting their domain of application.

The controller adaptation methodology proposed in this chapter is agnostic with respect to control, scheduling and system architecture aspects. For example, moving from a centralized to a distributed architecture poses no complications as far as the controller correctly receives sensor data and is allowed to send actuation data — the computations remain exactly the same. Regarding the scheduling discipline, the only requirement is that one additional task, which is the controller switching algorithm, must be executed within one period upon a controller change. No assumptions are made about the scheduling policy, priority scheme or



(a) proposed (b) classical

Figure 7.3: Response of the first system to a square-wave.

any other scheduling-related aspects. Finally, the presented method also does not pose any constraints on the nature or in the dynamics of the system, being suitable for any system for which a set of stable controllers can be designed, though some slight modification may have to be made if assumptions such as linearity are not present.

7.4 Evaluation of Proposed Solution

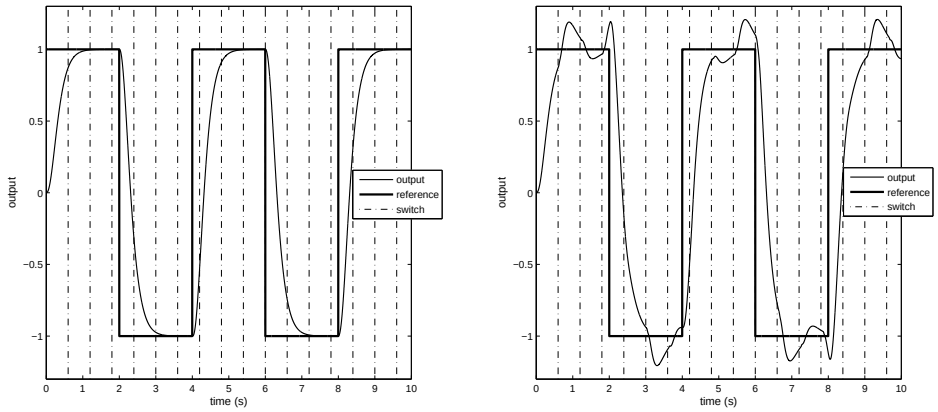
This section presents evaluation results obtained from three different systems. The systems are represented using the *variable phase* (the state transition matrix is a *companion* matrix and the input matrix has one single 1 and $n-1$ zeros). This choice was made due to its popularity in control practice, which results from the fact that it can be readily obtained from the physical characteristics of the systems.

The simulations were made using a square wave as reference signal. This wave changed between ± 1 with a duty cycle of 50% and a period of $4s$. The controllers used the standard regulators theory to drive the system according to the reference signal. Pole-placement was used to place the poles regularly spaced in the interval $[0.85\, 0.9]$.

Unless stated otherwise, all simulations last for 10 s. The controller is changed periodically with a period of 600 ms. The sub-parts of the control system (i.e., sensor, controller, actuator) communicate through a network, as depicted in Figure 7.2, with a delay of 2 ms and a jitter with a Poisson distribution with mean arrival time also of 2 ms. These values of delays and jitter were chosen to reflect values usually found in fieldbuses without isochronous transfer support.

The proposed method was tested in two systems. The first system was sampled at 10 ms and 15 ms. At 10 ms the plant has the following state space representation:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.82 & 1.8 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$



(a) proposed (b) classical

Figure 7.4: Response of second system to a square-wave

At 15 ms the plant's state space equations are:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.6109 & 1.5694 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y = \begin{bmatrix} 0.3009 & 1.7783 \end{bmatrix} x(k)$$

Figure 7.3 a) depicts the system behavior when the oscillation control technique proposed in this chapter is used, while Figure 7.3 b) depicts the results obtained in identical conditions, with the exception that the oscillation control technique is absent. By simple observation it can be concluded that the proposed approach behaves as expected, exhibiting no oscillations, whereas in the classical approach controller switchings result in output oscillations.

The second system commutes between sampling periods 16 ms and 20 ms, with a plant that has the following representations at each respective period:

$$x(k+1) = \begin{bmatrix} 0 & 1.0000 & 0 \\ 0 & 0 & 1.0000 \\ 0.3150 & -1.4300 & 2.1000 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(k)$$

and

$$x(k+1) = \begin{bmatrix} 0 & 1.0000 & 0 \\ 0 & 0 & 1.0000 \\ 0.2360 & -1.1990 & 1.9373 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} -0.0172 & 0.3045 & 1.4277 \end{bmatrix} x(k)$$

Figure 7.4 shows the simulation results of this system. Once again, the classical approach caused the system to oscillate, whereas with the approach proposed in this chapter the output does not oscillate.

Several measurements were made to quantify the improvements shown in Figures 7.3 and 7.4. They are summarized in Table 7.2. The first row of this table shows the rise-time of the

different systems. The proposed and the classical methods have rather similar values because during the rise time window there were no controller transitions.

The second row of Table 7.2 shows the overshoot values. The proposed mechanism does not exhibit any overshoot because it does not oscillate despite the controller changes, a property that classical approaches lack. A similar situation is seen in the settling time, since classical approaches, under controller change, never reach a state that can be considered steady (5% error band).

$$ISE = \int_0^T (y(t) - r(t))^2 \, \mathrm{d}t \tag{7.12}$$

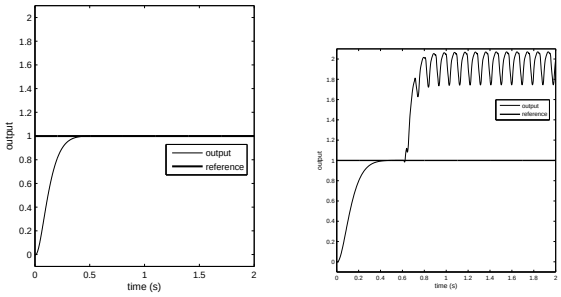
The ISE presented in the next to last row of Table 7.2 is computed according to Equation (7.12), in which T is the duration of the experiment, $y(t)$ is the output signal and $r(t)$ is the reference signal. The improvement in the ISE does not seem to be substantive, mostly due to the fact that the controllers are changed with a relatively low frequency. Hence, the ISE is dominated by the time that the systems spend tracking the input signal. Note that the impact of controller-change induced oscillations is higher on the slower system, since the associated recovery time is higher. This effect appears distinguishably when comparing the ISE of the first (faster dynamics) and second (slower dynamics) systems. The faster system experiences an ISE reduction of 3.1%, while the slower system has an ISE reduction of 9.8%.

To highlight the controller change's impact on the ISE, the *corrected* ISE was also computed, shown in the last row of Table 7.2. This modified ISE value is computed only for points in which both the reference and output signals are slowly varying, hence removing the impact of the reference signal changes, and thus the tracking error. With this metric the performance difference becomes much more evident. The corrected ISE is essentially null for both systems, when the oscillation control technique is applied. Note that the absolute value of the corrected ISE is lower for system two because its output is stable during a shorter amount of time, thereby reducing the time interval during which the corrected ISE is computed.

The final simulation presents an example of a system in which a controller change causes an oscillation, that in turn causes another controller change, so on and so forth. The system changes between a sample rate of 10 ms and 15 ms, according to the output error. There are two threshold values for the estimate of the output error, to induce hysteresis. The higher threshold is set to 0.80 and the lower threshold is set to 0.10. The system remains at the higher rate as long as the lower threshold is not crossed. Similarly, it remains at the lower sampling rate as long as the higher threshold is not crossed. At the highest sampler rate, i.e.,

	SYS1 prop	SYS1 classic	SYS2 prop	SYS2 classic
Rise time	247ms	256ms	525ms	585ms
Overshoot	0%	13.63%	0%	20.9%
Settling time (5%)	439ms	undefined	887ms	undefined
ISE	1584	1633	4051	4448
corrected *ISE*	1.74×10^{-6}	14.19	2.93×10^{-6}	3.59

Table 7.2: Metrics comparison



(a) proposed (b) classical

Figure 7.5: Example of system locked in oscillations

10ms, the system is represented by:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.6400 & 1.5217 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

At the sampling rate of 15 ms it has the representation:

$$x(k+1) = \begin{bmatrix} 0 & 1 \\ -0.5120 & 1.2751 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y = \begin{bmatrix} 1.5950 & 0.4073 \end{bmatrix} x(k)$$

Figure 7.5 depicts the behavior of such system. The *continuous* controller switching pattern is evident. It is also evident that the presented approach completely avoids it.

7.5 Conclusion

Dynamically switching controllers in order to always have active the controller that uses the lowest possible amount of resources, while guaranteing a minimum quality of control, has been a field of intense research. However, in existing approaches controller changes are often followed by output oscillations, which may degrade the quality of control and waste resources.

This chapter investigated the origin of such output oscillations, which are, in essence, due to the state variables not being in agreement before and after the controller switchings. To solve this problem, a novel mechanism, which finds a change of basis matrix that turns a state variable under a given representation into another one, was developed. During runtime, this mechanism only requires a simple matrix product, thus being implementable even in resource-constrained hardware, as those typically found in control system applications.

Simulations of diverse distributed systems, subject to network contention, were carried out. The obtained results are in perfect accordance with the expectations. The same systems, subject to the same stimulus, experienced oscillations when the classical approaches were employed, while such oscillations did not occur when the adaptation method presented in this chapter was used. Quantitatively, the absence of such oscillations results in the nullification of the overshoot and ISE due to controller changes.

Chapter 8

Toward Deterministic Implementations of (m,k)–firm Schedulers

8.1 Introduction

From encoded video files that can withstand the loss of certain packets to control messages that can be dropped without a serious deterioration of the quality-of-control to control applications that can loose some messages without a significant deterioration of the quality of control, it is somewhat common to have tasks that tolerate a number of packet misses, and one possibility is the use of the so called (m, k)–firm tasks.

In recent years, a number of advances were made concerning the implementation of schedulers capable of handling this type of tasks. However, most of the hitherto implemented schedulers are of the so called **probabilistic** (m, k)–firm type, which means that though the scheduler may try to reduce the amount of time in which the (m, k)–firm guarantees are not met, the primary goal is to ensure that **on average** m out of k job executions are met. Such approach may have deleterious effects on the scheduled tasks. For example, in the video case above, the decoder may not be capable of correctly decode certain frames if some other particular frames are missing.

The other type of (m, k)–firm schedulers is called **deterministic** and it ensures that for any group of k consecutive jobs at least m are executed. Nevertheless, all deterministic schedulers that have been presented in the literature suffer from the same illness, namely they are schedulable if they are also EDF-schedulable, a point that will be made more clear throughout this chapter. This fact renders them of little use, since one could simply use EDF instead and execute all jobs from such tasks.

This chapter presents a new scheduler that can schedule tasks with utilizations that would otherwise be greater than one, a feat that, as will be seen, cannot be achieved by any other (m, k)–firm scheduler. Such task sets are scheduled by dropping jobs that do not substantially contribute to the overall quality of the system.

Both a static and a dynamic scheduler will be presented. The scheduler is dynamic, in the sense that whether a given job is executed does not depend only on the priority with which its associated task generates it, i.e., the priorities are associated to jobs as opposed to tasks. The guiding lines for the construction of such schedulers are presented. It is shown that this new

approach can schedule task sets with higher utilizations. In particular, all task sets schedulable by classical static schedulers are also schedulable by the dynamic scheduler presented herein, however the converse is not true. Furthermore, when the task set is schedulable both statically and dynamically, in general the dynamic scheduler achieves utilizations greater than the static one. In this chapter, in addition to the dynamical scheduler, a novel static scheduler is presented, which is derived from the dynamic scheduler, that similarly can schedule task sets with utilizations, that if all tasks were set to (k, k)–firm would be greater than one.

8.1.1 On the Use of (m,k)-firm Schedulers for Control Purposes

As discussed in Section 3.4 and in references therein, in control applications it is possible to significantly reduce the number of jobs that are executed, i.e., the number of control values that are computed/delivered, while incurring only a small increase of the cost function. Evidently, this reduction of the number of computed control values must be accompanied by a change of the control law.

Deterministic (m, k)–firm schedulers are best suited for used in control applications, because probabilistic schedulers do not preclude the possibility of long periods without any execution. Such long periods without service could bring about some catastrophic outcomes to the controlled system.

Hence, the obvious choice of (m, k)–firm schedulers for use in control systems are the deterministic ones. In fact, most of the existing strategies, as seen in Section 3.4, use a repeating pattern of executions and non executions. However, as will be shown in this chapter, the use of these repeating patterns may jeopardize their own schedulability.

For this reason, it is proposed in this chapter, a dynamic (m, k)–firm scheduler that can be used in control settings. Note that the efficient use of network resources in control settings is the main drive for the use of this type of schedulers in this thesis. Notwithstanding this observations, the proposed scheduler can be used for the scheduling of CPU tasks without any modification whatsoever.

This scheduler can be used, for instance, in the implementation of the networked control architecture presented in Chapter 4.

8.2 Static (Circular) (m,k)-firm Schedulers

This section generalizes some theorems in the literature. The first theorem generalizes the theorem proved in [Ram99] that states that upper mechanical (m, k)–firm frames generate (m, k)–firm sequences that always meet the respective guarantees. The theorem is generalized for any frame with the respective size and number of mandatory jobs. The second theorem helps to understand the periodic nature of (m, k)–firm sequences. In [QH00] is proven that the use of the Chinese reminder algorithm can be used to schedule (m, k)–firm task sets. The theorem is generalized to allow any algorithm that handles periodically repeating activation patterns. This is of uttermost importance because the Chinese reminder algorithm is known to be NP-hard, i.e., it cannot be solved in polynomial time.

These theorems will in turn be fundamental at establishing the improvements of the dynamical scheduler proposed in this chapter. But before that, a number of basic definitions are in order.

Definition. *An* (m, k)–firm *frame is any group of k bits in which m and only m bits are ones.*

Definition. *A job is optional if the rightmost bit of the* (m, k)–firm *frame associated with its task is zero, otherwise the job is mandatory.*

Definition. *An* (m, k)–firm *sequence is any sequence of job executions in which for any group of k consecutive job activations, at least m are executed.*

Furthermore, in this analysis it is made a number of assumptions:

- An optional job is executed if and only if its execution does not cause the miss of a deadline of a mandatory job.
- An optional job is executed if and only if it will be able to complete its execution within its respective deadline.
- Whenever a job is released, regardless of being executed or not, unless some pattern breaking event occurs, the whole (m, k)–firm frame associated with it is rotated one bit to the left.
- Tasks release jobs periodically, regardless of the job's priority.

The next theorem helps to understand the name (m, k)–firm frame

Theorem 1. *Any* (m, k)–firm *frame generates an* (m, k)–firm *sequence.*

Proof. The job execution sequence is encoded into the (m, k)–firm frame. So, if the sequence is rotated one bit to the left, then the bit sequence will be exactly the job execution sequence, where the newer job corresponds to the rightmost and the older one to the second from the right. Hence, in the worst case, when only mandatory jobs get executed, there will be exactly m jobs executed out of k, corresponding to the number of ones in the frame. □

This theorem removes the distinctiveness of any special sequence, such as the upper mechanical word used in [Ram99] and an intuition of it was the basis of the work in [QH00].

The next theorem helps to understand the static nature of (m, k)–firm frame.

Theorem 2. *k is a period of any task whose job sequence is generated by an* (m, k)–firm *frame. Or restating, the jobs activated at time t_i and at time $t_i + k * T$, with T the period of the job in question, are either both mandatory or both optional.*

Proof. Given the way in which jobs are set as mandatory or optional, to prove the theorem, it suffices to prove the periodicity of the (m, k)–firm sequence. Now, since each job activation triggers one rotation of the (m, k)–firm frame, then k activations will trigger k rotations of the (m, k)–firm frame. Since the (m, k)–firm frame is k bits long, k rotations put it back in its original state. □

Note, however, that k is not necessarily the smallest period, since the (m, k)–firm frame may have an *internal period*. Notwithstanding, if the frame does have an internal period, let n be the number of periods within the frame, then it is easy to prove that a $(\frac{\mathrm{m}}{\mathrm{n}}, \frac{\mathrm{k}}{\mathrm{n}})$–firm frame built with the last (or any consecutive) $\frac{k}{n}$ bits of the initial frame generates a job sequence with mandatory/optional jobs at the same places. Moreover, this new frame sequence has a

minimum period equal to its size. For example, the sequence [1 0 1 0] generates an $(1, 2)$–firm sequence, but its smallest period is not four, since the $(1, 2)$–firm frame [1 0] generates the same sequence, and does not have any internal periods. Due to the periodicity, the phase is a number between 0 and $k - 1$.

This last point has a strong bearing into the size of the search space. Since, if one (m, k)–firm frame can be rotated into another one, then they are the same (m, k)–firm frame, but with different phases. Thus, not testing the rotated version of a frame reduces the number of tests. Nevertheless, since tasks sets normally have more than one task, it may happen that a given (m, k)–firm frame phase is feasible but some other phase of the same (m, k)–firm frame is not.

Now, consider the following definition, which helps to understand the ensemble of frames generated by the task set.

Definition. *An hiperframe is the shortest set of all job activations (mandatory/optional) and their respective order from all* (m, k)–firm *frames, between two distinct instants, such that in both instants all* (m, k)–firm *frames are in the same phase.*

The period of an hiperframe can be determined by recording all (m, k)–firm frames phases. Then analyze its evolution until it goes back to its initial state. The whole history in between is the hiperframe.

Similarly to the case of (m, k)–firm frames, one can also define a phase of the hiperframe which helps to identify its state at any given point of the period. Evidently, the search for a feasible (m, k)–firm schedule can be reduced to the search of a feasible hiperframe.

8.3 Dynamic (m,k)-firm Schedules

The last section showed that classical static schedulers repeat the schedules of each of their (m, k)–firm tasks.

However, the best (m, k)–firm schedule does not necessarily belong to the set of schedules that can be generated by repeating (m, k)–firm frames, as is assumed, for example, in [QH00]. It is possible that the best schedule has the task set repeating itself with the periodicity of the hiperframe, as is shown in the examples presented in Section 8.4. For this reason, a different approach is proposed — Dynamic (m, k)–firm Schedulers.

In dynamic (m, k)–firm schedulers, at every instant each (m, k)–firm task tries to find the (m, k)–firm frame that best suits its ongoing conditions. This section presents a dynamic scheduler that constantly changes the characteristics of (m, k)–firm frames in such a way that it avoids to put the task set in a situation in which it is not possible to meet the deadlines of all mandatory jobs. This section proposes the changes to the frame of each task that achieves this goal. These constant frame changes is what renders this schedule dynamic. Such changes to the (m, k)–firm frame must respect two conditions, namely:

- not violate any (m, k)–firm guarantees,
- improve the future schedulability of the remaining tasks.

To achieve the first point, it suffices to make transformations that 1) maintain the number of mandatory jobs (m) within the (k) bits of the frame, even if the new sequence is not a possible shift of the original one, and 2) it is done during an optional job that got executed.

As shown by the next theorem, under these two conditions, the (m, k)–firm schedules of the task subject to the change of frame does not get violated. The transformation, would be to maintain the rightmost bit in its position and rotate one bit to the left the resulting leftmost (m, k)–firm frame, as opposed to a left rotation of one bit after a regular execution of a mandatory job. For example, consider the following transformation: $[b_{n-1}\, b_{n-2} \cdots b_1\, b_0] \rightarrow [b_{n-2}\, b_{n-3} \cdots b_1\, b_{n-1}\, b_0]$.

Before presenting the next theorem, it should be remarked the difference between an (m, k)–firm frame and an (m, k)–firm sequence: the frame is used to generate the sequence. The next theorem deals with situations in which the frame is modified in a way that maintains its size and number of ones. But, maintaining these characteristics does not imply that the sequence that is generated will maintain its guarantees, even if each of the original frames have do generate an appropriate sequence. Hence, it is necessary to find frame transformations that guarantee that the generated sequence will provide the respective guarantees. A similar issue arises at the hiperframe level, in which certain transformations cause several tasks to keep generating unfeasible set of simultaneous mandatory jobs.

Theorem 3. *Let there be an* (m, k)–firm *task, belonging to a schedulable set of* (m, k)–firm *tasks. Then, if after the execution of an optional job from a given task, the task in question maintains the most significant bit of its* (m, k)–firm *frame optional and rotates the remaining* $k-1$ *leftmost bits one bit to the left, then the resulting* (m, k)–firm *sequence will maintain its* (m, k)–firm *schedulability state.*

Proof. From the execution point of view, an optional task that gets executed is analogous to a mandatory job in the same point of the frame. In so being, consider Figure 8.1, in which a given frame is sampled. By assumption bit $b_0 = 0$, i.e., it is optional. However, in this execution sequence it was changed to $b_0 = 1$ since it was actually executed. Since, by assumption, the initial sequence respected the (m, k)–firm conditions, then after changing the last bit the initial k consecutive jobs contain $m+1$ executions. If the optional job does not get executed at the second pass, i.e., the second instance of b_0 then the following k consecutive jobs (starting at the first occurrence of b_{n-2}) get m executions from bits b_{n-2} to (the second) b_0 (if it also does not get executed). From there on, the (m, k)–firm guarantee would be met because it would be a static frame as presented before. □

$$\overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{\text{m+1 − executions}}$$
$$[\cdots b_{n-1}\, b_{n-2} \cdots b_1 \underbrace{\overbrace{b_0}^{rep\rightarrow}}_{b_0=0} b_0\ b_{n-2}\, b_{n-3} \cdots]$$

Figure 8.1: State of (m, k)–firm frames during the execution of an optional job

It must be stressed the importance of the last theorem, because it allows for the execution of mandatory jobs without jeopardizing the (m, k)–firm guarantees. Once again, the execution of optional jobs was not considered in static (cyclic) (m, k)–firm and some version actually demanded that they did not take place at all, such as the upper mechanical and the deeply red sequences, whereas in probabilistic approaches there is more of a grey line in this issue due to the inherent differences in semantics. For example, instead of being optional or mandatory, jobs are/can be classified according to their distance to failure.

8.3.1 Dynamic Scheduling of Optional Executions

As previously stated, optional jobs are executed whenever their execution does not cause a deadline miss of a mandatory job. However, no mechanism or algorithm that achieves this goal was found in the literature. Hence, Algorithm 2 was developed and it is a suboptimal algorithm that allows to determine if a given optional job can be safely executed.

To develop such algorithm first, it should added to the ready queue (that already contains the set of all already released mandatory jobs and optional jobs successfully inserted in the queue) the set of yet to be released mandatory jobs that could potentially be denied the use of resources due to insertion of the new job. This contradicts with traditional ready-queues that only accommodate jobs that have already been released.

Following this line of reasoning, first an array with jobs currently being *executed* and *close* future mandatory jobs is maintained. *Closeness* is defined as the horizon in which the loading effect of the optional job disappears. The loading effects appear when the execution of an optional job increases the tardiness of a mandatory job, which in turn increases the tardiness of yet another mandatory job, so on and so forth.

As for the algorithm *per se*, first, let there be another *field* on the elements of the execution queue, namely, the worst case time of job completion, f. The algorithm works by simulating an introduction of the job in question.

The test is based on a concept that has some similarities with the concept of *level i busy period*. A level i busy period is the amount of time in which there are jobs with priority higher than i that are ready for execution. Once the optional job is inserted it will have a priority equal to the priority assigned by the queue's scheduling policy. Hence, even though optional jobs may start with a lower priority, they can preempt a mandatory if they are allowed into the ready queue and their priority is higher than that of the mandatory job in question. For each job that is being tested, the future execution time is divided into a series of busy periods of levels either higher than the job's, and jobs with a busy period of levels lower than it. If the sum of the computational time in which the processor has a busy period of a level not higher than the job's priority, as to allow for its execution, reaches the job's worst case execution time before its respective deadline, then the job in question is deemed as being schedulable. Otherwise it is deemed not schedulable.

It should be remarked that upon the introduction of even a single job, this test is performed for several other jobs that would end later. The purpose of the use of the concept of level i busy is to avoid testing for the jobs that are executed with a priority higher than the job that is being tested. Such tests should be avoided because by definition, the respective jobs will not change their timeliness.

Algorithm 2 has three chained repeat cycles. The outermost moves through the various jobs, the middle one, moves through each preemption parts of the respective job, and the last cycle moves through the busy period that are the cause of the preemption taken into consideration in the middle cycle.

The outermost cycle starts by setting the current time equal to the release time of the current process, if no other process is active, otherwise, it is set to the finish time of the last job. The variable c' stores the amount of CPU time spent by the current process and is initialized with zero. The outermost cycle uses the middle cycle to determine if the job is schedulable. If it is not then it terminates the algorithm returning failure. Otherwise, it tests the next job until the loading effects are dissipated, i.e., until a job does not load the following one. If such tests end without any job's deadline is violated, then it returns success.

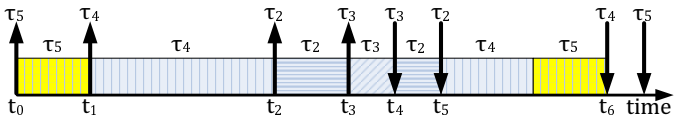


Figure 8.2: Example of execution a test of a single job as described by Algorithm 2.

The middle cycle starts by determining the amount of time till the start of the next busy period level i, which is an amount of time that will be available for the execution of τ_i. It adds this amount to the c' until it becomes equal to $\tau_i.c$, but not greater since in this case the job would finish. It updates the current time and checks for deadline violations. Then it uses the innermost cycle to determine the duration of the next busy period level i, which it uses to skip those instants.

The innermost cycle simple repeats itself for as long as there are higher priority jobs in the ready queue, in order to determine the finish time of this busy period.

Due to the multiple cycles of Algorithm 2, it may prove itself to have high computational requirements in online scheduling scenarios. However, in spite of the dynamic nature of the algorithm, it can be scheduled offline since all the information necessary for the scheduling will be already available. Such offline scheduling produces a scheduling table that can be used online to choose the jobs that get executed and it does not have an high online computational requirement.

An example of the execution of Algorithm 2 in which the queue is EDF sorted is provided in Figure 8.2. In it, it is tested the schedulability, after the introduction of τ_1. τ_4 is tested because the loading effects of τ_1 are still present. In $t = t_1$ τ_4 is released and it finds out that the executing task has a priority lower than its own, so the value of r is set to $t_2 - t_1$ which is the amount of time until the start of the next busy period of a level than its own. This busy period lasts until t_3, with another job with a priority higher than that of τ_4's being released in it. At t_3, the last job with priority higher than τ_4 ends. τ_4 recomputes r and this time its value is $t5 - t3$. However, it completes its execution at $t = t_4$.

τ_4 is the only job whose schedulability is being tested, which corresponds to one execution of the outermost cycle. By construction, all jobs with priority higher than τ_4 are schedulable, whereas the jobs with priority lower than τ_4 will have their schedulability tested later.

8.3.2 Precedence Rules

In the last sections it was shown that 1) frames can be changed dynamically without triggering a miss of the (m, k)–firm guarantees and 2) an algorithm to insert an optional job into the execution queue was presented. However, if all tasks that can independently enter the execution queue without causing a deadline miss do so simultaneously, it might happen that the new execution queue/hiperframe is not schedulable. In fact, in order to make changes if and only if it will not cause a mandatory job to fail its deadline, it is not sufficient to test for a single job trying to enter in the queue, it is necessary to test for the whole hiperframe.

At this point a number of new concepts must be introduced to help understand when a deadline might be failed. First, as mentioned above, all jobs that get executed are put in a single execution queue. This is a significant departure from approaches that use two queues, one for mandatory jobs and another for optional jobs, with jobs in the second queue being executed if, and only if, the first queue is empty. The reason for a single queue is that 1) it completely excludes the possibility of an optional task starting its execution but missing

Algorithm 2 Inserting an optional job into an ready queue

$T \leftarrow$ set of all jobs inside the execution queue
τ_j %%j^{th} job in T ordered according to the inverse of the priority
$\tau_j.\{f, d, r, c\}$ %%j^{th} job's finish, deadline, release and WCET times
$m \leftarrow SYS_{TIME}$ %%current time
$i \leftarrow$ index of inserted job
repeat
  $m \leftarrow \max\{\tau_i.r, \tau_{i-1}.f\}$ %%initializes current time
  $c' \leftarrow 0$ %%amount of CPU time hitherto used by the current job
  repeat
    $r \leftarrow \min\{\tau_j.r | (\tau_j.f > m \wedge j \leq i)\} - m$ %%length of current busy period
    $c' \leftarrow \min\{\tau_i.c, c' + r\}$ %%add amount of time that the job can execute before being preempted
    $m \leftarrow \min\{\tau_i.c - c', r\} + m$ %%update current time
    if $m > \tau_i.d$ **then**
      return FAIL %%if deadline is past, then return fail
    end if
    if $c' == \tau_i.c$ **then**
      $\tau_i.f \leftarrow m$ %%the job's finish time is equal to the current time
    else
      $T_1 \leftarrow \{\forall_k \tau_k : (\tau_k.r == m) \wedge (\tau_k.d < \tau_i.d)\}$ %%set of all tasks that are released at this instant and belong to level i busy period
      repeat
        $c_- \leftarrow \sum T_1.c$ %%total amount of time of the current busy period
        $T_2 \leftarrow \{\forall_k : m < \tau_k.r < m + c_- \wedge \tau_k.d < \tau_i.d\}$ %%set of tasks released during the last busy period
        $m \leftarrow m + c_-$ %%update current time
        if $m > \tau_i.d$ **then**
          return FAIL %%if deadline is past, then return fail
        end if
      until $T_2 == \{\}$ %%until the queue only have lower priority processes
    end if
  until $c' == \tau_i.c$ %%until the completion of the job
  $i \leftarrow i + 1$ %%test next job
until $\tau_{i-1}.f \leq \tau_i.r$ %%the last job finishes before the start of the current
return SUCCESS

its deadline and 2) it will never happen that an optional task could be scheduled if it were executed before a mandatory task but it is not schedulable if executed afterwards (think *Earliest Deadline First* (EDF) or *Least Laxity First* (LLF)), thus decreasing the number of executed optional tasks.

Moreover, EDF scheduling policy is used due to its high utilization, relatively lower computational demand when compared to other protocols that use dynamic priorities. Furthermore, as with other dynamic priorities protocols, EDF can be considered a static priority assignment policy at the job level, meaning that any two jobs in an hiperframe, with phase relations that allow them to compete for resources, maintain their priority relationship. This renders dy-

namic schedulers such as EDF a natural choice for dynamic scheduling that involves changes to the (hiper)frame. Other aspect that favor EDF is its associated large body of knowledge.

Thus, whenever an optional job is released, it checks the state of the execution queue using the EDF rules in order to determine if it can enter the queue without causing any job that is already in the queue to miss its deadline. Nevertheless, as stated before, there may be more than one job that performs this action and concludes that it is eligible to enter in the queue. In this situation, though the entry of only one task will not cause a deadline miss, the entry of several might. Hence, to solve this problem, first there is a mechanism that introduces precedences, and second only are inserted in the execution queue the highest precedence jobs that do not cause a deadline miss of jobs already inside the queue.

The precedence mechanisms are only used when there is a *draw* concerning which optional job enters in the execution queue. Good candidates for precedence mechanisms are those that would lead to choices of optional jobs to be executed that increase the overall schedulability, or simply that increase the number of future optional jobs that will be executed. However, for the best of the author's knowledge, there is no computationally light algorithm that allows to perform such type of ordering in real-time. Hence, a simple ordering mechanism that consists in choosing the job from the task set that lost the last arbitration.

Note that the precedence rules are internal to the scheduler. For example, it would be tempting to choose, in a control setting with a draw concerning to optional jobs, the job associated with a task that had the highest value of some monotonically increasing function of the error. However, 1) by definition these draws are rare and 2) as seen from the discussion above if the scheduler has to choose the loser of the previous draw in order to maintain schedulability. It is in this sense that these draws are internal to the scheduler. The only draws that do not have these characteristic are the first to happen. However, also by definition, these are even more scarce, too scarce to have a significant impact.

8.3.3 Shifting Mandatory Jobs

As presented in Section 8.3.1, frame changes were bound to happen only in optional jobs that actually got executed. However, there are scheduling sets in which:

- the task set is schedulable,
- it is impossible to stop jobs from all tasks from being placed as mandatory and released simultaneously,
- whenever jobs from all tasks are released simultaneously at least one deadline is missed.

Such condition can happen, for example, at system startup. Under such conditions, there is no possible shift of the optional jobs that could possible save the system from missing an important deadline. A possible solution is to do with mandatory jobs what is also done to optional ones. Namely,

- whenever mandatory jobs are released, check if all mandatory jobs can be executed,
- if not, sort them according to a predefined set of parameters and the top jobs (according to the referred order) are executed as mandatory, the rest act as optional jobs,
- if a mandatory job is not executed, the leftmost bits of its (m, k)–firm frame are rotated.

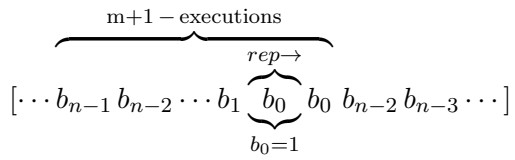


Figure 8.3: State of (m, k)–firm frames during the non execution of a mandatory job

---

The second point above should be done with a certain amount of caution, since, unlike in the optional case, if done repeatedly will lead to a dynamic failure. To overcome this problem, an initial schedulability analysis should be performed. In that way it would be guaranteed that all such (m, k)–firm frame transformations would lead to systems that are still (m, k)–firm schedulable. Nonetheless, an online approach, which still require an initial schedulability analysis is provided, based on the following theorem.

Theorem 4. *A given* (m, k)–firm *task, that belongs to a set of schedulable* (m, k)–firm *tasks, keep its rightmost bit fixed while rotating one bit to the left the remaining bits, after the non-execution of mandatory job, if and only if its previous* $k-1$ *job releases had at least* m *executions.*

If part. As in the case of optional jobs, execution-wise, a mandatory job that did not get executed is akin to an optional job that also did not get executed. Therefore, after changing the last bit from $0 \rightarrow 1$, and shifting the leftmost $k-1$ bits, the job execution will appear as presented in Figure 8.2. The sequences before the transition will be scheduled to the (m, k)–firm schedulability assumption, the sequence of k consecutive bits starting at b_0 has m execution due to the assumption that its previous $k-1$ job release (from b_0 to b_{n-1}) had at least m executions. [**Only if part**] Note that without this assumption the (m, k)–firm guarantee would not be met in this execution. Moreover, the execution of the following set of k consecutive bits is bound to meet the (m, k)–firm guarantees if the initial one has done so. In fact, since bit b_0 was exchanged for a 0, if $b_0 = 0$ then it would not change the number of executed jobs (which by assumption are at least m). On the other hand, if $b_0 = 1$ the number of executed jobs would actually increase. Subsequent activations of the task in question would meet the task's (m, k)–firm guarantees because they correspond to shifts of a static (m, k)–firm frame. □

However, it is noteworthy that the main condition for the previous theorem is met if, and only if, the task had executed an optional job in its recent past, i.e., within the last $k-1$ jobs, since otherwise the number of jobs activations within $k-1$ jobs would be $k-1$ jobs. Another noteworthy aspect of the last theorem is the fact that it applies only to one task. The case of a group of tasks will be slowly developed in the following sections.

As in the optional job case, precedences can/must also be defined, which would allow the scheduler to choose which mandatory jobs, among the mandatory jobs that can be turned into optional ones, would be *left out* in case of such temporary overloads. The use of a precedence rule similar to the one used for the case of optional jobs, i.e., mandatory jobs that have lost in the last arbitration have higher priorities, proved itself always successful in a series of tests performed by the author. Some of these tests are presented in Section 8.4. In fact, this is a result that the author has not been able to prove: the need for shifting a mandatory job only

occur in the first jobs of a task, either at the beginning of the executions or when a new task is introduced in the task set.

The last three paragraphs suggest an alternative view of the dynamic algorithm presented herein, namely

1 initialize the execution set with k bits in which m bits are set,

2 if the non-execution of a job would jeopardize the (m, k)–firm guarantees (among the last $k-1$ jobs only $m-1$ jobs were executed), set it as mandatory k and set it as optional otherwise,

3 optional jobs compete for the execution in unused *spaces*,

4 if a draw occur, when the precedence rule is applied, the task that won the last arbitration loses the current one,

5 if a draw remains, choose jobs for execution randomly.

8.3.4 hiperframe Initialization

Dynamic (m, k)–firm scheduling has a number of peculiarities that are nonexisting in static (m, k)–firm scheduling. For instance, in the static case the hiperframe is known *a priori*, i.e., before the activation of the first job. A naïve approach would be to initialize the dynamic scheduler with a statical hiperframe that was known to have reasonable scheduling characteristics. But the knowledge of such hiperframe implies some sort of static scheduling, which defeats the purpose of using a dynamic scheduler.

In spite of this aspect, the dynamic approach is still of use, because if the task set is schedulable then the dynamical approach converges to an hiperframe that is schedulable regardless of the initial hiperframe. However, it is possible that there is more than one possible schedule that verifies of (m, k)–firm conditions, some of them differing only in their phases. Moreover, different initial hiperframes may converge to different final hiperframes.

This is a positive aspect of this scheduler, and of dynamical schedulers in general, namely there is no need to find an offline schedule. Hence, it suffices to guarantee that the task set is schedulable.

Regarding the initialization *per se*, as stated before, it is not known *a priori* any schedulable hiperframe. More precisely, for operations like the postponement of a mandatory job, as previously discussed, it is vital to already know the (m, k)–firm frame of the job in question. Therefore, measures are necessary to ensure that such initial lack of knowledge will not turn into a dynamic failure in an otherwise (m, k)–firm schedulable set.

One way of achieving such goal is to assume that all previous jobs, of tasks that have not yet activated any jobs, were successfully executed. This ensures that at the beginning of the hiperframe all possible transformations are allowed. However, from an execution point-of-view, being able to execute such jobs even while having to change the initial hiperframe is akin to start from a correct hiperframe and execute the same jobs. Hence, by assuming that all jobs before the beginning of the hiperframe were executed the hiperframe will automatically act as or converge to a *correct* hiperframe.

The strategy presented in the last paragraph is, in a sense, optimal but not unique. For example, the phases of the hiperframe can be used to develop several variations of the presented strategy.

8.4 Examples of Dynamic Scheduling

This section presents a few examples of the execution of the dynamic scheduler presented in this chapter. These examples are intended, primarily, to give an impression of how the scheduler works. The first group of examples can be scheduled by a static scheduler but it was included because it is helpful to understand how the scheduler works. The second group is more complex, and presents task sets that are not schedulable by any static scheduler hitherto in the literature, though, subsequent sections of this chapter present a static (m, k)–firm scheduler that is capable of scheduling these task sets. Furthermore, these examples show some emerging properties of the dynamic scheduler.

Whenever necessary, (m, k)–firm tasks will be described as (C, T, m, k) were the first two variables designate the *worst case execution time* and the *period* respectively.

First, consider a system with two tasks, with the same period and execution time, which in turn are equal to each other ($C = T$) and with a $(1, 2)$–firm frame. This is a system that is not schedulable by the scheduler presented in [Ram99], though there are static schedulers that are able to find a suitable schedule, for example [QH00]. As pointed out in the previous section, since it is schedulable, the system will converge to an hiperframe that meets all (m, k)–firm guarantees. In so being, lets assume that both tasks start with the (m, k)–firm frame $[0\,1]$, which leads to a non schedulable hiperframe. At the start both tasks generate what supposedly would be mandatory jobs. Moreover, both jobs can be turned into optional ones, since it is assumed that jobs generated before the start of the hiperframe were all executed. Hence, failing this particular job would still meet the (m, k)–firm guarantee, with the past assumed as have been executed and the current as failed. Using the rules for draws, one job would be chosen randomly for execution and it would maintain its mandatory status, whereas the other would switch to optional as previously described. For the sake of the presentation, lets assume that the first task kept its first job as mandatory. It is easy to show that these tasks would never change their (m, k)–firm frames again, repeating themselves as $\tau_1 : [0\,1\cdots]$ and $\tau_2 : [1\,0\cdots]$. Note that the other possible (m, k)–firm hiperframe, according to the random choice that is made, differs from the presented choice only in its phase.

For the second example consider also a system with two tasks, namely $(1, 1, 1, 2)$ and $(1, 1, 1, 3)$. Assume that they would start with the (m, k)–firm frames $[0\,1]$ and $[0\,0\,1]$ respectively, which are the frames that are generated by the scheduler in [Ram99]. A summarized

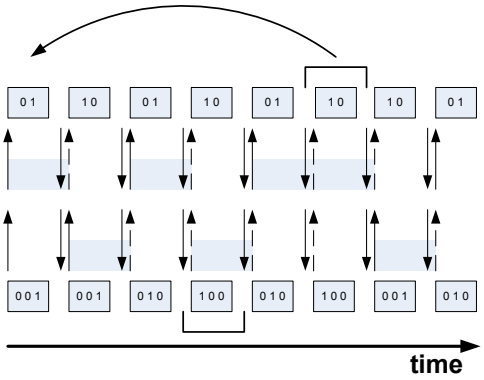


Figure 8.4: A possible schedule of example 2 generated by the dynamic scheduler proposed in this document.

execution history of the described task set can be found in depiction Figure 8.4, whereas Figure 8.5 provides a step-by-step view of its execution. Once again there will be a tie according to the execution rules presented above, and for the sake of simplicity, lets choose that the first task maintains its job as mandatory. So, after the first execution the first task gets shifted to [1 0] whereas the second maintains the one in the rightmost position and shifts the other two bits, which causes no effect since the last two bits are equal. The second period is executed by the second task, since it is the only task with a mandatory job, leaving the system frames in $\tau_1 : [0\,1]$ and $\tau_2 : [0\,1\,0]$. In the third period, the first task executes again because it is the only task with a mandatory job, hence the system frames are in the state $\tau_1 : [1\,0]$ and $\tau_2 : [1\,0\,0]$. At this point, both tasks have optional jobs which are competing for execution. It is important to notice that the first tie involved two mandatory jobs whereas the second one involves two optional jobs, which according to the precedence rules are dealt with separately. Furthermore (this point will become more evident at the end of this paragraph) the first tie was not a tie in the hiperframe to which the system is converging, but it was an *artificially* generated tie stemming from the fact that a feasible hiperframe was not known at the beginning, hence some care must be exerted when applying this rule. Continuing, since this is the first occurrence of a tie of optional jobs, any choice can be made. Once again, let the job from the second task be randomly set as mandatory, leaving the system frames in $\tau_1 : [0\,1]$ and $\tau_2 : [0\,1\,0]$. The system state at the beginning of the fifth period is akin to the system state at the beginning of the third, with the major difference that whatever task was assumed to have won (the first task in this example) the last arbitration in the third period, now will lose it. Moving over, the job from the first task gets executed for obvious reasons, bringing the system frames into $\tau_1 : [1\,0]$ and $\tau_2 : [1\,0\,0]$, which is another tie that this time in this instance is won by task 1, leading to $\tau_1 : [1\,0]$ and $\tau_2 : [0\,0\,1]$. On this situation, the job from the second task gets executed, leading to $\tau_1 : [0\,1]$ and $\tau_2 : [0\,1\,0]$. At this point the repetition is self evident, the first state system frames in $\tau_1 : [0\,1]$ and $\tau_2 : [0\,0\,1]$ is equivalent to the sixth state $\tau_1 : [1\,0]$ and $\tau_2 : [1\,0\,0]$ since they produce the same output and a transition to the same state (hiperframe phase). It is noteworthy that this task set with $(m, k) = \{(1, 2), (1, 3)\}$ generated an *effective* $(m, k) = \{(3, 5), (2, 5)\}$ generated by the hiperframes [1 0 1 0 1] and [0 1 0 1 0] respectively, which could not be generated by any classical static scheduler with the initial (m, k)–firm values. Note also that it achieves an utilization of one, while distributing the (m, k)–firm utilization in a manner consistent with their initial values (multiplied/increased by the same amount).

The next example is a bit more complex, because the tasks do not have the same period. The task set is $\{(5, 6, 1, 3), (4, 5, 1, 2)\}$. This schedule is represented at Figures 8.6 and 8.7.Once more, static (m, k)–firm schedulers would have problems scheduling this simple task set since regardless of the initial phases (both of the tasks and of their hiperframes) there would be an instant in which both tasks would generate mandatory jobs. Once more, the initial hiperframes are unknown, hence it is assumed that all previous activations were executed. Let both jobs start as mandatory in instant $t = 0$ and let the job from τ_2 be chosen for execution. It will end its execution at instant $t = 4$. At $t = 5$ the second job from τ_2 is activated and it is set as optional, however, its execution would jeopardize the execution of the second job of τ_1, which is mandatory. Hence, this optional job does not get executed even tough the execution queue is currently empty. At $t = 6$ the second job of τ_1 is activated and executed. At $t = 10$, the third job of τ_2 is activated as mandatory and starts its execution at $t = 11$ after the completion of the job from τ_2. At $t = 12$, an optional job of τ_1 is released, which is also not executed because it would not meet its deadline due to interference of the

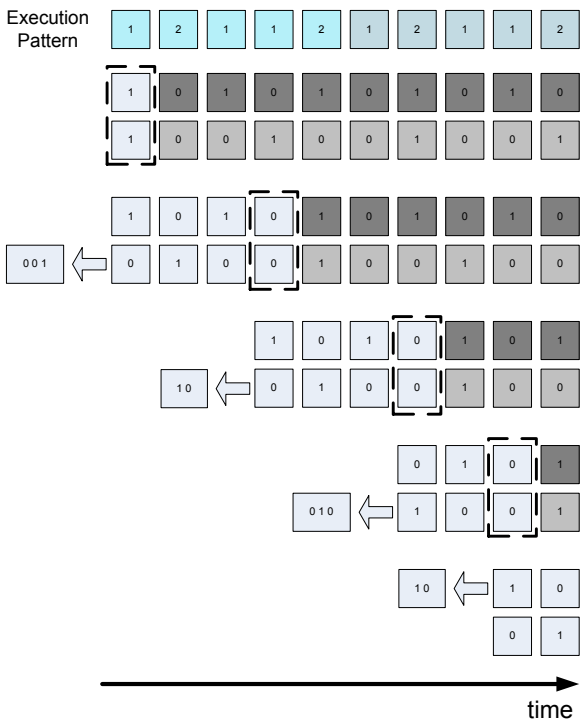


Figure 8.5: Full execution of a possible schedule of example 2 generated by the dynamic scheduler proposed in this document.

---

ongoing job. At $t = 15$ another optional job from τ_2 is activated and executed. At $t = 18$, τ_2 activates an optional job that gets executed because the next job from τ_1 is also optional (that would be activated at $t = 20$) and only one of them can be activated. Using the rules defined above, there is a draw and τ_2 executed its last. Continuing, the next job activation (τ_2) is at $t = 24$, which is an optional job that cannot be executed because at $t = 25$ a mandatory job (τ_1) will be activated. The later job ends its execution at $t = 29$ and at $t = 30$ the whole process repeat itself. Note, once more, that if τ_1 had (m, k)–firm frame of [0 1 0 1 0] and τ_2 [1 0 1 1 0 1] or simply [1 0 1] since there is an internal period within the (m, k)–firm frame. This schedule is represented at figures 8.6 and 8.7. Note also that if in the moment at which the two optional jobs were competing for execution ($t = 18$) τ_1 were chosen for execution then the (m, k)–firm frames would be τ_1 [0 1 0 0 1] and τ_2 [1 0 1 1 1 0]. There is also another possibility that emerges by starting the scheduling by choosing the job from the first task as the first job to be executed, which would lead to: τ_1 [1 0 0 1 0] and τ_2 [0 1 1 1 0 1] or τ_1 [1 0 1 0 0] and τ_2

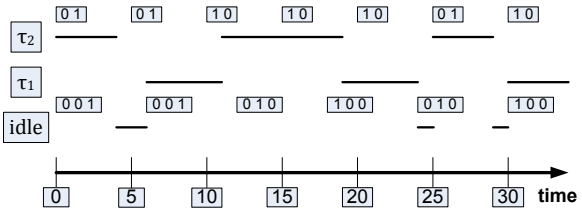


Figure 8.6: First half of a possible schedule of example 3 generated by the dynamic scheduler proposed in this document

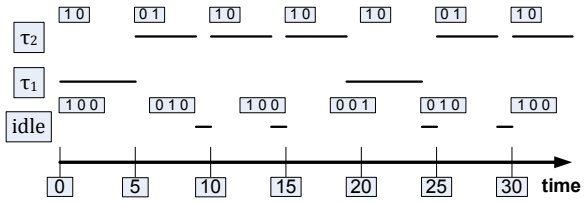


Figure 8.7: Second half of a possible schedule of example 3 generated by the dynamic scheduler proposed in this document

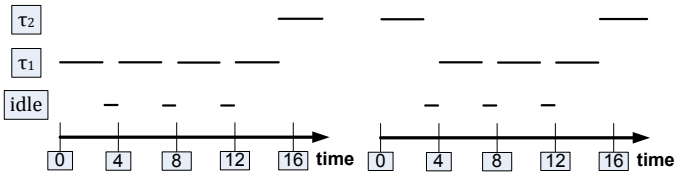


Figure 8.8: Schedule of task set of example 4 generated by GDPA — left starts with job from τ_1, right starts with job from τ_2

[0 1 1 0 1 1] (or [0 1 1]), according to a particular choice of optional job. Once again, the second hiperframe is comprised of rotated (m, k)–firm frames of the first hiperframe. In particular the first set can be generated by starting the sequence shown in this example at $t = 6$.

The following example shows that the scheduler presented herein is able to schedule task sets that GDPA is not.

Consider the task set $\{(3, 4, 1, 2), (3, 5, 1, 2)\}$. On this task set, GDPA, regardless of the choice of the first task to be executed would cause τ_2 to have a dynamic failure since τ_1 would, on several occasions, terminate the execution of one of its jobs, wait one time unit and then restart, since jobs from τ_2 would not be in execution. Hence, the execution queue would be empty. On the other hand, several jobs from τ_2 are activated while a job from the first task is in execution. Thus, the execution queue would be full. Figure 8.8 presents all possible schedules that the GDPA can generate on this task set.

The same task set can be scheduled using the dynamic scheduler presented in this chapter and such schedule is presented in Figure 8.9. In particular, for example, in $t = 4$ when the second job of τ_1 is activated, it does not enter in the ready queue due to a blocking from the mandatory job from τ_2 (which will be activated at $t = 5$). The rest of the schedule is similar to previous schedules. More specifically, it converges to a suitable hiperframe at $t = 8$, with $\tau_1 : [1\,0\,1\,0\,1]$ and $\tau_2 : [0\,1\,0\,1]$ or $\tau_2 : [0\,1]$ due to the internal period. Once again, starting with a job from τ_2 would only cause a phase difference in the hiperframe.

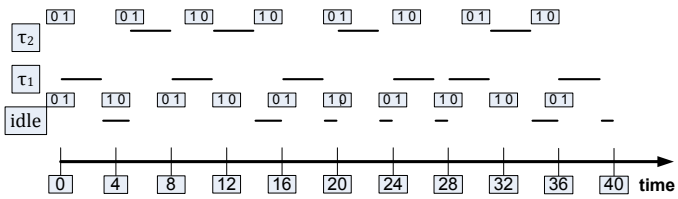


Figure 8.9: A possible schedule of example 3 generated by the dynamic scheduler proposed in this document

8.5 Comparison with Previous Solutions

The work presented herein improves upon previous schedulers, as described in Section 3.3.1, because it is not bound to EDF-schedulable task sets. The presented work is an improvement even over the work in [QH00], which searched over all possible static frames of each task with the corresponding values of m and k, i.e., for each $(\mathrm{m_i}, \mathrm{k_i})$–firm task it was searched for all possible k_i bits long frame which contained m_i ones. This implies that the presented algorithm outperforms any static algorithm of this family. The following theorem provides a formal proof of this fact.

Theorem 5. *The set of* (m, k)–firm *task sets that can be scheduled by classical static schedulers in which* m *and* k *of the* (m, k)–firm *frames are equal to the respective values of the* (m, k)–firm *task, is a proper subset of the set of* (m, k)–firm *task sets that can be scheduled by the dynamic scheduler presented herein.*

Proof. The theorem is true if all (m, k)–firm task sets that can be scheduled by a static scheduler can also be scheduled by the novel scheduler, i.e., the meaning of the *subset* part, and there is at least one task set that is schedulable by the proposed scheduler but it is not schedulable by any static scheduler, implying *proper subset* part. The *subset* part of the proof can be established if one considers that 1) if there is a static (m, k)–firm schedule that meets all (m, k)–firm guarantees using (m, k)–firm frames with values of m and k equal to the respective values of the (m, k)–firm, then if the dynamic scheduler is initialized with the aforementioned schedule (hiperframe), in the worst case scenario, i.e., no optional job is executed, the dynamic scheduler will send exactly the same sequence of jobs into the execution queue. And 2) as discussed above, regardless of the initial schedule (hiperframe), if the task set is schedulable, then the hiperframe converges into a schedulable (fixed) hiperframe, possibly the static schedule in question. From what follows that if a classical static scheduler can meet all guarantees, so can the dynamic scheduler. For proving the *proper* part it suffices to present an example of a task set that is schedulable by the dynamic scheduler but is not schedulable by any classical static scheduler. To this end, consider a task set with two tasks, for which the classical utilization, i.e., $U = \sum_i \frac{C_i}{T_i}$, is greater than one, with periods that are not multiples of one another and with implicit deadlines. Consider also, that their (m, k)–firm properties are chosen in such a that it would allow the dynamic scheduler to meet its deadlines, for instance $m_i = 1$ and $k_i = 2\left\lceil \frac{T_j}{T_i} \right\rceil$, where j represents the other task. Due to the fact that the periods are not multiples of each other, when scheduled by a stationary scheduler, there will be an instant in which both tasks are activated as mandatory. At such instant at least one of the tasks will fail its deadline since the combined utilization of both tasks is greater than one. Therefore no static scheduler can guarantee that the (m, k)–firm constraints will be met. On the other hand this task set is schedulable by the dynamic scheduler by construction. This completes the proof. □

An important remark is that classical static (m, k)–firm schedulers can only schedule task sets that are EDF schedulable (see discussion at the beginning of this section) and all EDF schedulable task sets are schedulable by the dynamical algorithm proposed here, since the execution queue is itself EDF scheduled. Furthermore, besides the point raised by the last theorem about the set of schedulable task sets, it is important to realize that even among the task sets for which there is a classical static schedule, the proposed scheduler achieves

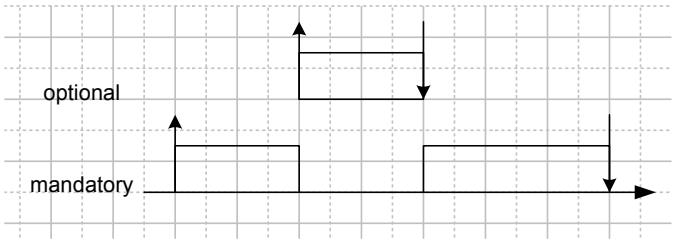


Figure 8.10: Example of an advantage of a single queue.

utilizations that are never inferior. In fact, they are usually higher than those achieved by classical statical schedulers.

A major difference, which is also an improvement, between the approach presented here and classical static (m, k)–firm schedulers, is the fact that in the classical approaches the schedulers have two queues, namely: queue 1, devoted to mandatory jobs, and all jobs that are in it get scheduled; queue 2, is used to schedule optional jobs, and these jobs are only executed if queue 1 is empty. Each queue is scheduled according to classic scheduling policies, such as EDF. In the approach proposed in this chapter, it is used only one queue. This in turn may increase the number of optional jobs that are executed. For example, consider Figure 8.10, in which the mandatory job has an absolute deadline that is rather far away, whereas there is an optional job that has a deadline that is relatively near. In the scheme that is used in classical static/dynamic (m, k)–firm schedulers, the mandatory job would be in the highest priority queue, hence it would be executed first. Thus, the optional job would not be executed right away because the mandatory job was already in execution, leading to a deadline miss of the optional job. On the other hand, in the proposed scheduler, after certifying itself that the optional job would not cause a deadline miss, would execute the optional job, as shown in Figure 8.10, hence meeting the deadline of both jobs.

But there are a few downturns. The first is related to the high computational complexity of the proposed dynamic scheduler. A related negative aspect, which is also common to all dynamic algorithms, deals with the loss of predictability under overload conditions. By overload it is meant the moments in which the task set is not schedulable. The loss of predictability occurs in the sense that in scheduling algorithms such as those presented in [Ram99], whenever there is a violation of an (m, k)–firm guarantee, the tasks with the longest periods suffer the violation. Since on such schedulers the initial instant is a critical instant, due to the fact that in such instant all tasks activate a job set as mandatory, and jobs in the execution queue are served in a Rate Monotonic manner. Furthermore, when a job misses a mandatory job it also violates its (m, k)–firm guarantee. On the other hand, for the case of dynamic (m, k)–firm schedulers, it is not possible to make such predictions beforehand.

The work presented in this chapter improves upon the work of DBP and its variants because it does not have any of the downturns presented in the section 3.3.1. Examples of such downturns include a disregard for the deadlines of the tasks, the fact that the algorithm tends to schedule tasks with a short distance to failure close to each other, a total disregard for the relation of the periods of the tasks (this point will become clearer in the next section) and a disregard for the relations between the time to failure of the various tasks.

A third negative aspect of the use of dynamic (m, k)–firm schedulers, relates to the fact that some applications require that beside the usual (m, k)–firm guarantees, tasks must have a mandatory job in a pre-determined part of the (m, k)–firm frame. Examples of these ap-

plications include the JPEG frame transmission, as discussed in the Section 3.3.1. Under such conditions, a dynamical scheduler, as presented so far, can not give all guarantees due to the loss of predictability that was already discussed. Nevertheless, it is possible to make some modifications in order to properly accommodate these requirements, such as splitting the task, giving different values of m and k to each of the subtasks, one or more behaving as a classical task, i.e., executing all of its jobs. Or modify the (m, k)–firm definitions as to have three levels of jobs instead of the current two, i.e., mandatory and optional.

8.6 Static (m,k)-firm frames (Revisited)

In the previous section it was established that the dynamic (m, k)–firm scheduler presented here can schedule tasks sets that otherwise cannot be scheduled by classical (m, k)–firm schedulers. This may seem paradoxical since, for example, the approach presented in [QH00] (theoretically) searches through all possible (m, k)–firm (meta) frames.

However, there is no paradox. In fact, the apparent contradiction stems from the set of assumptions that were made about the space of (m, k)–firm frames that meet a given (m, k)–firm guarantee. That is, at the beginning of this chapter it was shown that any repetition pattern, with k bits of which (at least) m, i.e., frame, generate a mandatory job sequence that fulfilled the respective (m, k)–firm guarantees. The contradiction appears when it is assumed that this is the *only* way of generating sequences that provide (m, k)–firm guarantees in question. In fact, there are other repeating sequences that also generate (m, k)–firm sequences with similar guarantees.

For example, recalling the sequence that was given in the first example, i.e., an $(1, 2)$–firm task had the generating (m, k)–firm frame [1 0 1 0 1] which always meet the $(1, 2)$–firm guarantee.

Thereby, the space of all possible (m, k)–firm frames that give a certain (m, k)–firm guarantee does not only include all possible (m, k)–firm frames with m and k having values equal to those of the respective (m, k)–firm, but it includes all $(\mathrm{m}', \mathrm{k}')$–firm frames that when repeated (indefinitely) generate a sequence in which any subsequence of k consecutive activations contains at least m jobs that are executed. In the remaining of this section a number of properties of such sequences will be presented, but first an important definition is in order.

Definition. *An infinite sequence of* (m, k)–firm *job activations is tight if for every set of* k *consecutive activations there are exactly* m *executed jobs.*

The concept of tight sequences, though simple, helps to visualize other concepts related with (m, k)–firm frames and sequences. In fact, in a nutshell, the concept of tight sequences encapsulates most of the contributions in the literature regarding static (m, k)–firm scheduling. But more importantly, it helps to give a view of other possible (m, k)–firm schedulers.

Theorem 6. *An* (m, k)–firm *sequence is tight if and only if it exists an* (m, k)–firm *frame with the same values of* m *and* k *that can generate it.*

Proof. The basics for the proof was already given in previous discussions. The *if part*, which seeks to establish that using the aforementioned type of frames generates the sequences in question, is easy to establish: In fact, it was proved in Theorem 1. The *only if* part, which aims at proving that this type of sequences can be produced only by the claimed type of frames, is proven because if a sequence is tight then whenever the last k^{th} job is/was executed

(or not) so will the current one, since otherwise it would either fail its guarantees or not longer be tight. Therefore, k is a period of the sequence, hence there is a (m, k)–firm frame that can generate it. Note, however, that (m, k)–firm frames may have an internal period, which, in turn, means that there is a smaller frame that generates that same sequence. Nevertheless, the theorem stands. □

Evidently, a tight (m, k)–firm sequence can only be generate by an (m, k)–firm frame with the same values of m and k, as stated by the theorem above. However, (m, k)–firm frames may have internal periods which may make them appear to have a different size.

A corollary of this theorem is that (m, k)–firm sequences with certain guarantees can be generated from (m, k)–firm frames, i.e., tight sequences, but nothing prevents these same guarantees to be provided by non-tight sequences, as was believed in classical schedulers. Moreover, given the theorem, (m, k)–firm sequences not generated by an (m, k)–firm frame have *optional* executions on its midst.

This might seem like a new paradox, since it was started with (m, k)–firm sequences generated by (m, k)–firm frames (thus tight), that could not be scheduled by certain processes, and ended up with (m, k)–firm sequences generated in a manner that execute more jobs than the *necessary* to provide the (m, k)–firm guarantees, yet the set of both types can schedule more task sets than the former alone, as if the introduction of redundancy is helpful to schedule an already overloaded task set.

The apparent paradox disappears if it is noted that these *optional* executions (on critical cases) happen before a critical moment, thus liberating one of the tasks from producing a mandatory job in a critical moment that would lead to a deadline miss.

Another important aspect is that the point raised above implies that the (m, k)–firm frames of each of the tasks on a (m, k)–firm task set must have the same period, i.e., $k_i T_i = constant$, forming within themselves an hiperframe that has the property that the points in which the tasks generate the highest loads have a phase difference, thereby all tasks are in *phase resonance*. Otherwise, it would happen that after a number of periods, the mentioned critical moment would occur. This is a point that classical static (m, k)–firm schedulers do not respect.

The last paragraphs implies that there is a necessity to study (m, k)–firm guarantees that are provided by (m, k)–firm frames of different sizes. For example, a $(4, 6)$–firm guarantee, can be implemented using a $(9, 12)$–firm frame. Algorithm 3 allows to perform a test of such compatibility.

Algorithm 3 provides a test to whether a certain frame respects a given guarantee. The first three lines of the algorithm 3 define the variables that are use in the algorithm. Line four computes and stores the number of mandatory jobs among the first k activations, it has that particular form because the array that stores the (m, k)–firm frame may be of a size, k_f, smaller than k. Otherwise it would simply be $\sum_{j=0}^{k-1} mkf(j)$. Line five determines if the condition is initially verified. Then it comes a *while cycle* that starts at line six with the initialization of the counting variable. The cycle has two conditions, as shown in line seven, one that will be debated in a moment and another regarding the fact that if the test failed at any given point then the whole test is considered a failure. Line eight computes the number of activations in the current set of k consecutive bits, by adding the current number status (zero for optional and one for mandatory) and subtracting the status of the job from k executions ago. This is due to the fact that any set of k executions differs from the previous only by this two jobs. The number computed in line eight is used in line nine to test for a violation

Algorithm 3 Test for compatibility of a (m, k)–firm sequence with a given guarantee

1: n %number of mandatory jobs in the current set
2: mkf %array of k_f entries with the (m, k)–firm frame
3: m, k %the (m, k)–firm guarantee
4: $n \leftarrow \sum_{j=0}^{k-1 \bmod k_f} mkf(j) + \left\lfloor \frac{k}{k_f} \right\rfloor \sum_{j=0}^{k_f - 1} mkf(j)$
5: *compatible* $\leftarrow$'$n \geq m$'
6: $i \leftarrow 0$
7: **while** $i < k_f$ **and** *compatible* **do**
8: $\quad n \leftarrow n + mkf((i+k) \bmod k_f) - mkf(i)$
9: $\quad$ *compatible* $\leftarrow$'$n \geq m$'
10: $\quad i \leftarrow i + 1$
11: **end while**
12: **return** *compatible*

of the (m, k)–firm guarantee. In line ten the cycle counter is increased. The counter counts up to k_f, which due to the periodicity of the sequence generated by the respective frame, is sufficient to test all possibilities that can be generated.

The counter is updated by noting that two consecutive groups of k bits differ only at their extremities. Hence, in the sum of their respective mandatory jobs, it is removed from the new sequence the bit that belonged to the previous sequence and no longer belongs to this sequence and is added the bit that belongs to the new sequence but was not present in the previous one.

Definition. *An induced* $(\mathrm{m}_1, \mathrm{k}_1)$–firm *frame of a given* $(\mathrm{m}_2, \mathrm{k}_2)$–firm *guarantee is a frame that verifies* $(\mathrm{m}_2, \mathrm{k}_2)$–firm *guarantees and is obtained by composing a smaller* $(\mathrm{m}', \mathrm{k}')$–firm *frame until it has size* k_1.

Note that, 1) the set of (m, k)–firm frames that can provide another $(\mathrm{m}', \mathrm{k}')$–firm guarantees is not limited to $k > k'$. For example, consider internal periods, and 2) not all $(\mathrm{m}_1, \mathrm{k}_1)$–firm frames constructed by running a smaller frame, meet the $(\mathrm{m}_2, \mathrm{k}_2)$–firm guarantees. In fact, those that do meet the guarantees are called minimum induced frames.

The study of these induced frames, as opposed to the application of Algorithm 3, allows for the generation of frames with certain guarantees, as opposed to the testing of frames to see if they provide the guarantees in question.

In essence an induced frame has the format

$$F_1 = \left\{ f' \overbrace{F' \; F' \cdots F' \; F'}^{n} \right\}$$

where $n = \left\lfloor \frac{k_2}{k'} \right\rfloor$, F' is the bit sequence of $(\mathrm{m}', \mathrm{k}')$–firm frame, and f' are the rightmost $k_2 \bmod k'$ bits of the same sequence. Due to its format, if a given $(\mathrm{m}_1, \mathrm{k}_1)$–firm frame is a minimal induced frame of another $(\mathrm{m}, \mathrm{k}_2)$–firm frame, i.e., meet all its guarantees, for $n > 1$ then will also be for $n = 1$. Therefore, assuming that the sequence formed by a pure concatenation of F' blocks satisfy the $(\mathrm{m}_2, \mathrm{k}_2)$–firm guarantees, then it suffices to study the case $n = 1$. In the following, it is discussed first the conditions that allow for a repeating

frame to satisfy certain guarantees and second the conditions as is to some extent done in the following set of theorems. The following theorem can also be used in other circumstances, namely to make it easier to find new frames that meet some guarantees, given that another frame that meets the same guarantees is known.

Theorem 7. *Any* $(\mathrm{m}_1, \mathrm{k}_1)$–firm *frame generated by the repetition of another* $(\mathrm{m}', \mathrm{k}')$–firm *frame is an induced frame that satisfies some* $(\mathrm{m}_2, \mathrm{k}_2)$–firm *guarantees, with* $k_1 > k_2$ *if and only if 1)* $(\mathrm{m}', \mathrm{k}')$–firm *frame satisfies* $(\mathrm{m}_2, \mathrm{k}_2)$–firm *and 2) every right-going sequence of, up to* $k_1 \bmod k_2$*, consecutive bits starting at bit* $k_1 \bmod k_2$ *counted from the right does not have less mandatory jobs than a right-going sequence of the same size, starting from the leftmost bit. The case* $k_1 \equiv 0 \bmod k_2$ *is also contained in the last sequence: it would imply a comparison of two zero length sequences that cannot have different number of mandatory activations, hence it suffices that* $(\mathrm{m}', \mathrm{k}')$–firm *frame satisfied the* $mkfm_2k_2$ *guarantees.*

Proof. The first k_1 activations (mandatory/optional) of the $(\mathrm{m}_1, \mathrm{k}_1)$–firm frame are exactly the activations of its generating ($(\mathrm{m}', \mathrm{k}')$–firm) frame, by construction. Therefore, if any miss of the $(\mathrm{m}_2, \mathrm{k}_2)$–firm guarantees occur it will happen during the shift of the $k_1 \equiv 0 \,(\bmod k_2)$, i.e., when the set of last k_2 consecutive bits have at its left $p\,(p < k_2)$ bits of which where the rightmost bits of the $(\mathrm{m}', \mathrm{k}')$–firm frame and at its right $k_2 - p$ also from the rightmost bits from the $(\mathrm{m}_2, \mathrm{k}_2)$–firm frame. Upon such shifts, the set of the last consecutive bits will have a number of the rightmost bits of the initial $(\mathrm{m}', \mathrm{k}')$–firm. Since the initial sequence was tight (by construction), it will maintain its guarantee if and only if the bits that replaced the original bits of the sequence do not have less ones than the original. □

A corollary of the previous theorem is that, if at any point the number of ones of the replacing sequence, i.e., the sequence that starts at bit $k_1 \bmod k_2$ (counted from the right), is greater than that of the sequence being replaced, then at such point, the respective sequence of last k_2 consecutive bits will have more than m_2 bits, thus its sequence that is generated will not be tight. Conversely, if it is smaller, then at the point in question the respective sequence will be in a dynamic failure.

For example, borrowed from the second example above, assuming that $m = \lceil \frac{k}{2} \rceil$, the frame [1 0] only generates minimum induced frames if the new frames have an even k, since otherwise the leading zero would conflict with the trailing one. On the other hand, the frame [0 1] always generates a minimum induced frame. In particular the solution presented at the time (generated by the dynamic scheduler) is not unique. Another solution is τ_1:[0 1 0 1 0 1] and τ_2:[1 0 1 0 1 0] (and a shift of one bit of the hiperframe), which is generated for $k_1 = 2$ the initial frames τ_1:[0 1] and τ_2:[1 0]. Note that for $tau_2, k_1 < k_2$.

It should be remarked that a shifted version of an induced frame that satisfies a given (m, k)–firm guarantee, also satisfies the guarantees in question. This point becomes of uttermost importance when building static hiperframes, in which each of individual frame was built from a smaller (m, k)–firm, but the frames must have a phase relation between themselves as discussed above. Nevertheless, regarding the phase of inducing frame, i.e., the smaller one, its phase, also as discussed above, does change whether or not the resulting frame provides the intended guarantees.

The following is another corollary of the previous theorem, which is essential to understand the relationship between static and dynamically (m, k)–firm frames. But due to its importance, it will be stated as a theorem.

Theorem 8. *If a given* $(\mathrm{m}_1, \mathrm{k}_1)$–firm *frame satisfies the guarantees of another* $(\mathrm{m}_2, \mathrm{k}_2)$–firm*, then after extending the repeating* $(\mathrm{m}_1, \mathrm{k}_1)$–firm *frame until it has a size that is a multiple of* k_2*, the number of mandatory jobs will be no smaller than* $m_2 \cdot \gcd(k_1, k2)$*, with* gcd *being the 'greatest common divider'.*

Proof. The proof is rather simple, since if the number of mandatory jobs were smaller than $\mathrm{m}_2 \cdot \gcd(\mathrm{k}_1, \mathrm{k2})$, then it would imply that at some point of the extended frame it would violate the $(\mathrm{m}_2, \mathrm{k}_2)$–firm guarantees, which contradicts the assumptions that $(\mathrm{m}_1, \mathrm{k}_1)$–firm frame satisfies the guarantees of $(\mathrm{m}_2, \mathrm{k}_2)$–firm. □

Note that being non-smaller implies that either their is a classical (m, k)–firm frame, i.e., in which $k_1 = \mathrm{k}_2$, or there will be more executions than necessary, thus implying the execution of optional jobs, as it was postulated beforehand and now is proven.

Summarizing, the dynamic scheduling algorithm presented in previous sections, if left running without any unforeseen events, generates an execution pattern that is similar to the execution pattern generated by static frames discussed in this section.

8.6.1 Optimal Statical (m,k)-firm Scheduling

This section provided a number of tools that allows one to speak about optimal (m, k)–firm scheduling. In special, now searches of all possible hiperframes that respect a given set of (m, k)–firm guarantees are possible. This section is intended to provide rules to guide such a search.

The minimum induced $(\mathrm{m}_1, \mathrm{k}_1)$–firm frame is, as implied by its name, the frame of size k that has the minimum number of mandatory jobs and is generated by a given $(\mathrm{m}', \mathrm{k}')$–firm frame, while meeting a given $(\mathrm{m}_2, \mathrm{k}_2)$–firm guarantees. Due to such minimality, if there is any possible schedule then, there is a way of scheduling it using only minimum induced (m, k)–firm frames.

The hiperframe has a period that is also a period of all of its tasks. Thereby, it only makes sense to test the (m, k)–firm (each task) in which k is equal to the hiperframe, that is, of common multiples of all tasks periods. Furthermore, only initial $(\mathrm{m}', \mathrm{k}')$–firm frames that generate appropriated minimum induced (m, k)–firm frames need to be tested, even though, for each set of minimum induced frames all possible phases will have to be tested.

8.7 Conclusion

(m, k)–firm schedulers have a great potential to improve the performance of many real-time applications. However, for many application, improvements cannot occur until it is possible to provide some deterministic guarantees regarding the performance of the schedulers.

Two avenues for the development of this type of (m, k)–firm schedulers were discussed. The first approach concerned dynamic schedulers, which traditionally has been mostly comprised by probabilistic schedulers of the DBP family. However, the proposed dynamic scheduler hinged upon the fact that it is possible to dynamical change an (m, k)–firm frame while maintaining its respective guarantee. In fact, there are task sets in which only is possible to provide the respective guarantees if the respective frame is changed at some point.

A second scheduler was discussed, in this case a static one, and it was based on the fact that the frame that a given (m, k)–firm task used to schedule a given task did not need to

be k bits long with m ones. In fact, it was shown that it is possible to provide the same guarantees using frames of different sizes. Furthermore, there are task sets that cannot be scheduled, in the deterministic sense, using classical approaches, i.e., with frames of sizes equal to the respective (m, k)–firm values, but that can be scheduled by this generalization of static schedulers.

Notwithstanding the novelty of these two new schedulers, some points remain to be optimized. For example, the algorithm for the insertion of a new job on the dynamic schedulers' execution queue, which in its current state is relatively computationally intensive, thereby increasing the time required to perform a context switch. Another point that needs some improvement is the schedulability test, since so far it has only been put forth a mechanism that detects if a given schedule verifies the guarantees but has not been put forth an algorithm to test if it is possible to find a schedule that provides the guarantees.

Chapter 9

Aggregation of Duplicate Sensitive Summaries

9.1 Introduction

Previous decades brought about a revolution in radio and microprocessor technology that made possible a plethora of new applications. In the midst of such development, Wireless Sensor Networks (WSN) have recently emerged as a synergy of two related technologies, i.e. radio and microprocessors. Both of them had exponential improvements in cost, size and functionality in recent years. In particular, the possibility of using many inexpensive sensor nodes interconnected by WSN for a number of ends, which made possible a novel set of applications so diverse as air monitoring, forest fire detection, structural monitoring, water monitoring, etc. Furthermore, such improvements are expected to keep their pace in the near future, thus WSN should become even more prevalent.

This potential of WSN drew the attention of the research community. This research is further motivated by the fact that many aspects of WSN remain as open problems. Problems such as the fact that WSN are usually heavily resource-constrained. Of particular relevance is energy efficiency since, in many applications, nodes have to operate during long periods without an external energy source, which made energy conservation one of the most studied aspect of WSN.

The literature on this topic reveals many techniques to improve energy efficiency, such as the use of in-network aggregation, the use of clustering (with or without cluster head rotation), exploitation of the inherent spatial correlation of readings among neighbor nodes, variation of transmission energy, sleeping during long periods of inactivity, exploitation of temporal correlation of the signals (data caching, estimation, system identification and so on), among others.

One approach to data transmission from common nodes to the root, which is of particular relevance for the present discussion, is to have all nodes transmit their data integrally to the root and at the root perform the computations on the collected data. However, this approach is energy-wise inefficient due to the rather high number of messages that are sent. A more efficient approach is the gradual aggregation of values as they are transmitted upstream, which is called **in-network aggregation**.

On the other hand, WSN links are *fragile*. Particularly, they can be temporarily unavailable, be subject to relatively high error rates or be asymmetric, among other issues. Hence,

the use of multi-path routing has been proposed [CLKB04, MYH06, MNG05, WW11] to lessen its effects.

However, the problem of aggregation of duplicate sensitive summaries (e.g. sum, average, histogram, etc.) in multi-path routing networks is not fully resolved. In fact, the use of multi-path routing may cause errors in the aggregation of duplicate-sensitive functions. A function is said to be duplicate-insensitive if its result does not change upon the introduction of a duplicate argument. For example, the min and max functions are duplicate-insensitive. A function is said to be duplicate-sensitive if its output changes with the addition of a duplicate argument. For example the sum function, in which introducing a non-zero argument more than once causes different results, is duplicate-sensitive.

Many attempts have been made to solve this issue, for example, by giving approximate answers, by forcing single path routes, by searching for aggregate-insensitive versions or by decomposing these functions in a series of duplicate-insensitive functions and then query the WSN for each of the new functions, among others.

None of the already introduced solutions satisfy the initial goals, since they either have an error by nature (approximations based approaches), or use single routes, which does not offer any redundancy, or considerable increase the number of messages that are required to compute the aggregate (alternative functions approaches), thus defeating the original purpose of the aggregation, which is reduce the number of messages to save energy.

This chapter proposes algorithms that address this problem by sending redundant aggregated information through different paths, so the data can be reconstructed to obtain the best possible summary, given the available paths. Two algorithms are presented, one better suited for networks dominated by link errors and another better suited to networks where the predominant error source is node failure. The algorithms are light during normal network operation, with the most intensive processing performed during the initialization phase.

The approach presented herein outperform previous solutions found in the literature in two key aspects: complete topology independence and aggregation depth independence. The algorithms are topology independent in the sense that there is no need to manually configure the topology of the network into the nodes nor is necessary to put the nodes in a pre-specified distribution. The algorithms are aggregation depth independent in the sense that they work properly regardless of the number of levels of the networks.

The remaining of this chapter is organized as follows. Section 9.2 presents an overview of the related work. Section 9.3 describes the count summary that is used in this chapter. Section 9.4 presents the methodology proposed in this chapter to carry out the aggregate of duplicate-sensitive data in multi-path networks. Section 9.5 discusses the applicability of the prosed algorithms. Section 9.6 presents simulation results carried out to assess the correctness of the algorithms and to evaluate its performance. Finally, section 9.7 concludes the chapter.

9.1.1 On the Use of In-Network Aggregation of Duplicate Sensitive Summaries for WSN Control Applications

The use of WSN for control processes is booming, due to the ever decreasing cost-performance ratio of the WSN. Furthermore, there are certain industries, e.g. the metallurgic, in which the use of wires is not a viable option for the sensor-controller communication.

However, existent WSN lack a number of functional, and of more relevance for the current thesis, the error rate guarantees needed to ensure the proper function of a control system. Moreover, these control applications require many spatially distributed, though temporally

dense sensor readings.

These facts imply that the current WSN need to be improved in order to deliver such improvements. This chapter presents one such improvement that has the potential to make the goal of using WSN for control applications a reality.

9.2 Related Work

Due the size of the literature of the topic, as can be attested by some reviews as in [CGKL11], the field of WSN is too vast to be covered in a related work topic of a single chapter. Therefore, this section is intended to give a perspective of the field by presenting only contributions closely related to the problem addressed in this chapter.

The use of aggregation in WSN excels in metrics such as energy expenditure and network lifetime, as shown in [KEW02]. Furthermore, in the same reference it is also shown that the denser the network the higher the benefits of aggregation. [KDN02] corroborates with these findings. [WMG04] presents a meta-level view of aggregation and argues for a co-design of the integrating parts of the network.

TAG [MFHH02] was one of the first contributions to tackle the problem of aggregation in WSN and under it nodes build the network by setting the level of the parent as the lowest level that it received and set the sending node(s) as the parent, set its own level as the level of the parent(s) plus one and broadcast its own address plus its level. [WDAA09] builds a tree similar to Shared Resource Tree (SRT) used in TAG (which the authors of TAG call TinyDB), however it is augmented by the use of an index table to decide the aggregation value, plus a common value (cv) agreement, which is a value that nodes with the same parent are supposed to verify $|node_{reading} - cv| < \delta$, cv is computed as the average of the children reading and is sent back to them. If a children verifies the last condition then it does not send its data. max / min are also based on cv, thus may lead to erroneous values. In fact, this aspect is common to approaches that use a cluster head and spatial correlation. Under this approach, nodes with more than one parent alternate between sending messages to each of its parents.

In [ZGE03] it is introduced the direct diffusion of digests (aggregates), in which nodes compute their value as a function of their current value and the value received from neighbor nodes. They start sending this new value piggy-backed in a periodic beacon of the underlying communication layer, for example, with a heartbeat message. It takes at most the diameter of the network (in number of hops) to diffuse such digests. Obviously, in this form it only can compute digests of exemplary functions. Exemplary functions are those that can be computed as the result of an aggregate of previous values and one single new value. In fact, they have this name because their result depend on only one value, for example max and min summaries. For non-exemplary functions, the authors propose that, first, a direct digest that is always *won* by the sink is performed, second, each node would memorize the node from which it received the winning diffusion (parent) until a tree is formed, third, a diffusion is sent along the tree that emerged in the process. All the same, the authors do not show any difference between their proposal and the regular diffusion. Notwithstanding, the paper presents an interesting study of link asymmetry in WSN, which led the authors to propose a mechanism to switch parents whenever a certain link is asymmetric.

In the same line, [NGSA04] provides a formalization of the concept of diffusions. Three operations are considered: synopsis generation, fusion and evaluation, that work as suggested

by their names. The paper also provides a number of, so called, necessary and sufficient conditions for correctness, though, it also presents a situation that verifies all such conditions but does not produce the correct value, implying that the presented conditions are not sufficient (their necessity may also be under dispute). [CMV08] uses a fuzzy logic approach to compute exemplary functions. Each node has a *fuzzifier* that decides whether to send the data. A comparison with the 'no aggregation scenario' was made, though no comparison with classical aggregations was presented. It was also used the sleep approach.

In [MYH06] it is proposed to solve the problem of multi-path aggregation of duplicate sensitive nodes by keeping nodes from aggregating data if there is a possibility of duplication. To this end, upon the construction of the multi-path tree, nodes send their's and their parents addresses along with their hop count to the root. Nodes that join the network, would know each of their parents and their parent's parents. Based on this information, if the node as more than one parent, then it chooses the parent's parent with most paths from it as the aggregation point, otherwise it chooses its only parent as the aggregation point. Only two levels of the WSN are searched for, therefore it might happen that the link of the chosen parent's parents up the tree may fail while there is another parent's parents link which is operational. The fact that the lost of one (up tree) link/node can cause the loss of information of many of children nodes puts in question the very use of multi-path (redundancy). This approach point is explored in the approach advocated in this chapter.

In [MNG05] it is used an hybrid approach to routing. Using a tree approach closer to the leaves and a multipath approach closer to the sink. This helps to leverage the advantages of trees (low latency, low messaging) with the advantages of multipath (increased robustness). Furthermore, it is paramount to have redundancy in nodes that transport the aggregates of several other nodes, nonetheless, the authors approach to the transport of multipath traffic is highly inefficient, since it uses no aggregation. The authors justify their choice by noting the problems that arise with the use of aggregation in multipath routing, which are the same problems that this chapter is set to solve.

The above presented approaches to WSN try to reduce the energy expenditure of WSN by changing the way in which data is routed. But, as discussed in Section 9.1, there are other approaches, such as the use of approximate queries, which require less data to compute. Among the approximate queries (in which SOD based approaches are arguably a member), [GGP+03] uses a wavelet based approximate aggregation technique to store the results of several queries at different network hierarchies. Nodes at different hierarchies store results with different precisions. Whenever a query with a certain precision is done, it *drills down* the network until it finds a node with enough precision to answer the query. Due to the high volume of data generated by this approach, an aging scheme was employed.

Considine *et al* [CLKB04] present the FM-SKETCH (introduced by Flajolet and Martin), which approximate the number of distinct elements of a superset by seeing only its initial part. In turn, such sketches are used to estimate the number of distinct nodes in a network — approximate count summary — which, the authors argue, is well suited for multi-path networks. The paper generalizes FM-SKETCH to approximate sum summaries by having each node producing *val* (*val* is the value that the node sensed) distinct elements into the superset. Evidently, the number of distinct elements of this superset is equal to the sum summary. The sum summary is approximated by using the FM-SKETCH on the superset. A number of optimizations are provided. All the same, this approach that supposedly reduces the amount of data by computing approximates, actually for typical relative errors (0.85-0.95), it uses about the same amount of network resources (since it has no aggregation) and

computer resources (memory and CPU) as traditional exact summaries (with aggregation). The authors compared their approach to TAG and to the LIST approach, i.e., each node sends the value of the aggregate and the set of nodes used to compute the aggregate, which is similar to the approach provided herein, however, their LIST approach is considerable less optimal. First, nodes always send a huge set, second the node that is receiving may not be capable of finding a way to properly aggregate the received data due to duplications in the received data.

The use of SOD in order to reduce the amount of data that is sent, i.e., the exploitation of temporal correlation, has also been explored in a number of articles, more prominently in [MA01] and later at [MA02] the authors use a hybrid periodic\asynchronous model, in which data is sent periodically, however, rapid transitions are responded to by immediately sending data asynchronously. [SBLC03] aims at similar goals, using data caching and aggregation at each level of the tree.

Al-Karaki *et al* [AKUMK04] present both an exact and an approximate algorithms to find optimal routes for aggregation in WSN. The exact algorithm is stated as an integer linear programming problem, which the authors argue to be too complex to be solved in WSN. Hence, they propose an approximate genetic algorithm. However, the genetic algorithm is itself computationally heavy and involves many message exchanges, which is exacerbated by being performed in several rounds, during the normal function of the WSN, thereby consuming the very same energy that the algorithm aims at saving.

Approximate aggregations for the histogram (HIST) summary have also been proposed, as is the case of quantile tracking [CGMR05, SBAS04], top-k estimation [WXTL06, WXTL07] though the later group tend to be more exact. [AN11] is an example in which an approximation of an histogram is used to compute queries in WSN, however, the algorithm used is exact if the histogram is exact, but the authors do not provide any algorithm to compute exact histograms.

Aggregation in sparse networks is also an active research topic, motivated the energy saving approaches that put a significant fraction of networks on *sleep mode*. One such approach is described in [GGMH07], in which a dense WSN with a small number of hotspots is studied. A mechanism to find suitable routes is presented.

Another class of approaches is the bio inspired one, which is epitomized in [LKF07] (nonetheless, a similar approach is pursued in [OK09] and in references therein,) leverages the (ant) bio-inspired path finding algorithm, to build a routing tree. In this algorithm, all source nodes explore all paths to the sink, leaving a certain amount of *pheromone* at each link. Several factors (e.g. path size) determine the amount of pheromone left on each link. The amount of pheromone on each link is the sum of the amount left by on each paths that goes through the link. The higher a path's pheromone levels, the more likely it will be active. This family of algorithm has the downside of not being computationally and energetically light, even during normal operation. But have the upside of being extremely optimal at finding the paths and at reacting to topology changes, though not necessarily fast.

Another type of aggregates that gain a certain moment in the community is the gossip based aggregation [DSW06, KDG03, SZG07, FLS06, BGPS06, CPX06], in which each node 1) read its own value, 2)aggregates it with messages that it receives and 3) sends a message to a random neighbor with a fraction of its current value while maintaining another fraction of it. After an algorithm-dependent number of rounds, the aggregates converge to the correct value. Usual objections include the apparent lack of benefits for the common WSN, higher latencies and increased number of data exchanges.

Other types of optimizations to WSN that are not related to the main contribution of this chapter have been proposed, such as the exploitation of temporal correlation in [OJW03, OLW01, OW02, DKR04, SBS+02, OW00]. And spatial correlation, such as [RKK04, GBT+04, GNDC05, HCB00, DGM+04, JP04, OK09, GGMH07, VA06, YS07].

9.3 Count Summary

The main contribution of this work is independent of the intricacies of the underlying count summary. Nevertheless, it requires the execution of a count summary prior to carrying out an aggregation of a duplicate-sensitive function, or at least, to have the results of a count summary in cache. Therefore, it is paramount to use an efficient count summary and that it should be performed as infrequently as possible.

The following conditions reduce the need to perform count summaries: 1) parent nodes trigger a count summary action only if a given child does not communicate for a given period of time (T_{alive}), 2) all children send a message only during initialization or when they notice that have a different parent node. Additionally, at least one message (of whatever type) is sent with a given time window (T_{alive}). The optimal value of T_{alive} is a compromise: on the one hand, it must be as big as possible to save energy, but, on the other hand, large T_{alive} impairs network reactivity.

A possible count summary is to have: nodes use a bitmap addressing, in which each node address corresponds to a bit in a address array. Addresses can be pre-programmed prior to the deployment of the network. Leaf nodes put themselves in the count summary by sending a message with their bit set. Aggregations of the count summary are performed by implementing a bitwise OR of partial results. The number of distinct nodes that are offspring of a given node is equal to the number of set bit on its bitmap array (minus itself, depending on the working definition of offspring). Evidently, the number of nodes in the network is equal to the number of offspring of the root node plus one (the root itself). Due to the simple nature of this aggregate, a formal proof of its correctness will not be provided.

It should be remarked that a similar counting mechanism was proposed in [NGSA04]. However, their approach started with the bitmap addressing but at the higher levels used the FM-SKETCH [CLKB04], which conditions their approach to provide approximate results, and uses an high amount of memory [1]. Hence, there approach both used many resources and only provided an approximate answer. Beside this advantage, this count summary was chosen because it allows each node to know who its offsprings are.

9.4 Multi-path Aggregation

This chapter considers mesh-like networks, where each node can reach only a limited subset of other network nodes, normally the nearest neighbors. Node's data should be forwarded to a particular node, called sink. Links that connect nodes are subject to errors, either transient or permanent. It is assumed that the underlying communication protocol provides error detection capabilities, discarding erroneous frames. Nodes are also subject to errors and are fail-silent, i.e., they either operate correctly or do not send any information. Furthermore, the following definitions apply:

[1] approximate queries are used primarily to reduce the amount of time and memory used to perform a given action, whereas exact queries are used when the correctness of the results have primacy

- The WSN consists of $N_i, i = 0...K$ nodes.
- Without loss of generality, in the remainder of this chapter N_0 designates the sink node;
- Each node is connected to one or more neighbour nodes;
- Each link between two nodes (N_i, N_j) is designated by $L_{i,j}, i, j \in 0...K$;
- T_i is a bitmap representing the addresses of the node itself and its offspring
- A_k=[a,b,c, ...] denotes the aggregation of values a, b, c, ...;
- M_i designates a message sent by node N_i to its parent;
- A message can contain several aggregates. Aggregates sent in the same message are connected by a "+" symbol.

The system undergoes three sequential phases: physical topology discovery, virtual topology set up and, finally, the data collection, which is the normal state. Permanent errors may be considered as topology changes and thus this whole process may be repeated as often as necessary, although eventually only over specific parts of the network, i.e., those affected by errors. This can be achieved, for example, by having the nodes that note that they added/removed offsprings start broadcasting their new count summary.

In the first phase, all reachable nodes and their connections are identified. More concretely, each node discovers which nodes are its offsprings. This is achieved, for example, by using a count summary as the one described in section 9.3.

Once the topology is identified, phase 2, which consists in the virtual topology set-up, is started. At this stage each parent knows all the paths to each one of its offspring. Based on this information, each parent sends a message to each one of its children indicating if and how data should be aggregated, thereby creating several groups of aggregates. The decision about which data should be aggregated is taken primarily with a focus on minimizing the total number of messages.

Two different aggregation strategies are proposed in this chapter, one more suitable to handle node errors and another more efficient in the presence of link errors. The difference between these strategies resides primarily in which messages (nodes readings) are aggregated together. In the former case, messages are aggregated such that if all messages that come from a given child are lost the aggregates can still be *reconstructed* to the best value of the remaining nodes, whereas in the latter case the focus is on messages lost on a given link, in which the information that was lost in a given link can still be recovered for as long as there is at least one alternative path. The best strategy to use in a particular WSN should be selected according to the most frequent error type, though, as it will be seen in section 9.6, the link case algorithm has significant correction capabilities for errors resulting from node failures.

Finally, after phase 2 is complete, the system enters the normal operation phase, in which the data is collected. During this phase each parent node receives messages from its offspring, eliminates or reconstructs data, depending on the existence of errors, and forwards the aggregates defined during phase 2 to its parent nodes. Data recovery requires only a few table look-ups and simple algebraic operations on the received aggregates, thus during normal operation the processing overhead is small.

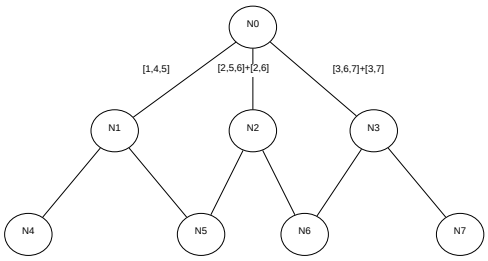


Figure 9.1: Example 1: Node Error Strategy

9.4.1 Multipath Aggregation of Duplicate Sensitive Summaries — Node Error Case

The strategy proposed to deal with node failures consists in setting one of the children to aggregate its own data with the data of its offspring, while the other children send several aggregates. These aggregates consist in its own data aggregated with the data of its offspring, followed by aggregates that contain the data that do not intersect with previous siblings in the same level.

To illustrate this strategy, consider the simple WSN depicted in figure 9.1.

As depicted in figure 9.1, node N_1 sends a message composed only by one field, which aggregates its own data and the data of its offspring (M_1 = [1,4,5]). Node N_2 sends one message with two fields, one aggregating its own data with its offspring (A_1=[2,5,6]) and another field that contains the data that does not intersect T_1, i.e., $T_2 \setminus T_1$; A_2 = [2,6]. Thus, M_2={[2,5,6]+[2,6]}. A similar procedure is applied to node N_3. Table 9.1 presents the aggregates conveyed by messages M_1 to M_3.

In case there are no errors, the sink receives aggregates $A_i = \{[1,4,5],[2,5,6],[2,6],[3,6,7],[3,7]\}, i \in 1 \ldots 5$, respectively contained in messages M_1, M_2 and M_3. It can be trivially verified that the correct value can be recovered by taking the last field of each one of these messages. Therefore, this strategy meets the first goal addressed in this paper, which is the ability to remove duplicates in the presence of redundant paths.

Lets now consider the case in which one of the nodes fails, e.g., node N_1. Observing figure 9.1, it can be seen that data from node N_1, which failed, and from node N_4, which has no redundant path to the sink, will be lost. However, data from all the other nodes should be recoverable. In fact, the sink node receives aggregates {[2,5,6] + [2,6]} and {[3,6,7] + [3,7]}, respectively contained in messages M_2 and M_3. Data from all accessible nodes can be recovered by combining the first field of M_2 with the last field of M_3. If the failing node is node N_2, the sink node receives aggregates [1,4,5] and [3,6,7] + [3,7]. Data from all accessible nodes can be obtained by combining aggregates [1,4,5] and [3,6,7]. A similar reasoning could be carried out regarding the failure of any node in the network. Therefore, the proposed strategy meets the second goal of the algorithm, which is the ability to recover data that has redundant paths to the source in the presence of node errors.

This idea can be extended to a larger number of children, as shown in table 9.1, i.e. each children first aggregate all its offspring, then aggregate all of its offspring minus the nodes that are reachable by nodes above it in the table, then do the same with minus sets from two nodes in above, then three and so on.

It can be seen that this approach may lead to a relatively high number of messages that must be sent by nodes further down the table, since the worst case number of aggregates

child index	message to send
1	$\{T_1\}$
2	$\{T_2\} + \{T_2 \backslash T_1\}$
3	$\{T_3\} + \{T_3 \backslash T_1\} + \{T_3 \backslash T_2\} + \{T_3 \backslash \{T_1 \cup T_2\}\}$
$\vdots$	$\vdots$

Table 9.1: Example of aggregates construction for the smallest node numbers in node errors case.

grows as a power of two. Methods to dramatically reduce the number of entries of such table are addressed latter on.

Analytical Proof of Reconstruction Capabilities

The last example showed that the node failure case algorithm provides some error correction capabilities. This subsection will establish 1) the basics for the reconstruction algorithm and 2) a formal proof[2] that such reconstruction algorithm always achieves its intended goals. The conjunction of 1) and 2) imply that the node failure case algorithm works as advertised.

The algorithm is based on Table 9.1 and it is as follows: whenever data from a given offspring is not received, all references from the offspring in question are removed from the table; the summary is computed as the aggregate of the rightmost elements of the resulting table.

For the proof of the reconstruction capabilities, note that the resulting table is exactly the table that would be generated if the node in question only had the offspring that were successfully received. This can be seen by a direct inspection of their structure. Therefore, to prove the point in question, it suffice to prove that in a table such as table 9.1 the aggregate of the rightmost elements of each line equals to the aggregate of the whole table.

The last proposition is proven by induction. In the first case, with only one offspring the proposition follows by definition. Computing the aggregate recursively, the value of the aggregate at the n^{th} line, is equal to the value of the aggregate of the first $n-1$ lines (the induction) plus the rightmost element of the n^{th}. The rightmost element of the n^{th} line is by construction the element that is necessary to be aggregated to the summary of the first $n-1$ lines in order, concluding the proof.

9.4.2 Multipath Aggregation of Duplicate Sensitive Summaries — Link Error Case

The link error case requires that whenever a link fails, the parent (or the sink) node receives all the data necessary to compute all aggregates that are computable, given the link failures in question.

The algorithm proposed to achieve this goal consists in having each node sending two aggregates, one containing the data related to its descendants that are reachable only by itself, and another aggregate composed by the data pertaining to nodes that communicate both with itself and with its siblings. The algorithm resembles the one presented for the node error case, except that in the error node case the operation was a bitwise complement,

[2]that was not written in the form of Theorems due to its rather verbose nature

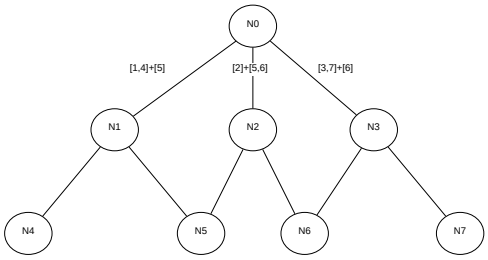


Figure 9.2: Example 1: Link Error Strategy

whereas in this case (link error) the sets are disjoint in relation with the other children's children count summaries.

To illustrate this strategy, consider the simple WSN depicted in Figure 9.2. Node N_1 sends a message composed by two fields, one aggregating its own data and the data of descendants that are reachable only by itself (node N_4, in the present case) and another aggregate with the data of descendants shared with each one of its siblings (only node N_5, in the present case), therefore $M_1 = \{[1,4]+[5]\}$. Node N_2 sends one message with three fields, one aggregating its own data only, since all its descendants are shared with its siblings, and two other aggregates with data of descendants shared with each one of its siblings, therefore $M_2 = \{[2]+[5]+[6]\}$. Following a similar reasoning, $M_3 = \{[3,7]+[6]\}$.

In this simple case it can be checked, by exhaustion, that this is a solution to the problem. In the absence of errors, the sink receives aggregates $A_i = \{[1,4],[5],[2],[5],[6],[3,7],[6]\}, i \in 1 \ldots 7$. The exact value can be recovered by adding A_1, A_2, A_3, A_5 and A_6. Thus, without errors the exact value can be recovered even in the presence of redundant paths, by sequentially adding the aggregates that have values not added before. By simple inspection of the aggregate set received by the sink, it can also be observe that each value appears in as many aggregates as the number of distinct paths to the sink. E.g. node N_5 that has two links appears in sets A_2 and A_4, while node N_4 appears only in one aggregate, since it has no redundant link. Thus, the redundancy is visible at the sink and it should be possible to recover the exact aggregate value even in the presence of errors. E.g. if link $L_{5,1}$ fails, the sink receives aggregates $A_i = \{[1,4],[\phi],[2],[5],[6],[3,7],[6]\}$. The exact value can be obtained e.g. by taking A_1, A_3, A_4, A_5 and A_6. The same reasoning can be applied to other sets to confirm that it is possible to recover the exact value of an aggregate function, even in the presence of link errors, by simple algebraic manipulation of the aggregates received by the sink, provided that there at least one alternative path.

Analytical Proof of Reconstruction Capabilities

As in the node error case, the proof will be established by providing a reconstruction algorithm and then proving that such algorithm can always find the (best possible) value.

Such an algorithm can be seen in Algorithm 7, which aggregates data from a number of offspring and then computes the respective aggregate. In a simplified manner, upon the reception of the aggregate of disjoint sets of offsprings (of which disjointness is ensured by the algorithm that generates the aggregation lists), the algorithm divides them according to offspring, i.e. source. From this information it is built a table in which each column has the data from each offspring.

For each aggregate that a node computes, it first divides it into a collection of parts or

sub-aggregates according to the count summary of each offspring. At run time it looks at each offspring column and if there are entries (sub-aggregates) that a) are necessary for the computation of the aggregate and b) are not yet in the list of aggregated entries then it 1) reads the value in question then 2) add the entry in question to the list of entries (sub-aggregates) that were already aggregated.

For the proof of the correctness of this algorithm its noteworthy that due to the way in which the table is built 1) elements in the same row are disjoint and 2) elements of different rows are either equal or disjoint. Note that given the way in which the data is stored at the receiver, 1) deals with the intra-node aggregates relationship whereas 2) deals with internode aggregates relationship. Both points follow directly from an analysis of the aggregate generating algorithm, i.e. Algorithm 5. that are equal because there the use of multipath routing may introduce duplicates at the receiver.

The proof follows by noting that a) there are no duplications due to the fact that each segment is introduced in the aggregate only once and b) whenever there is sufficient data received form the offspring to compute the aggregate it will find at least one entry from each sub-aggregate.

Notwithstanding, there are two aspects that deserve to be pointed out. The first pertains to the assumption that nodes are *fail silent* implying that all nodes that do transmit, transmit the right message. This assumption plus the assumption that all erroneous messages are discarded imply that all messages with a given sub-aggregate have the same value. Therefore, its value can be retrieved by reading the value of only one such sub-aggregate, which motivated the find-and-discard aspect of algorithm 5. Note, however, that if this assumption was not in place the value of each sub-aggregate could be decided based on some function of the values of the various copies.

Another related topic is what is done when there are not enough sub-aggregates to compute a particular aggregate. A naïve approach would be to compute the aggregate with the available partial sub-aggregates, however, if two or more nodes had to compute the same partial aggregate based on different sets of sub-aggregates, they would arrive at different values, implying that the assumption that all nodes that have a copy of a sub-aggregate have the same value would no longer be true. Note, however, that this situation is relatively uncommon, but whenever it happens it is impossible to guarantee that the various sub-aggregates will be equal. In fact, it can be seen that among the algorithms that aggregate at that particular depth, there is none that can guarantee that if there is a path to the sink, then it will be correctly aggregated.

At this point it is paramount to (formally) introduce the concept of aggregation depth. Aggregation depth is the distance in tree levels from the point of aggregation to the point in which the data is generated. For example, it can be said that all node aggregate their own data with a depth of zero, and their direct parents (if they aggregate) aggregate with a depth of one, and a node *grandparent* aggregate with a depth of two.

The main issue with link case algorithms is that most of them (the one in this paper included) aggregate with relatively small depth, leaving the possibility that the path to the point of aggregation may be inaccessible but that there is another path to a higher level node that is still accessible, hence not fulfilling their intended goals. Therefore, the only way of insuring that 1) there will be aggregation, 2) it will be possible to remove duplicates, 3) be resilient to link failures, while still ensuring the best possible aggregation at the sink, is if and only if each data is not aggregated until one level above the first level in which the information goes through one and only one node. Let us call the nodes in question *first*

common progenitors of a given data.

Such an optimal approach raises the issue of resource utilization. Since it may mean that a value may cross an entire network without being aggregated (if the node's first common progenitor is the sink). This calls for a compromise between the maximum number of hopes that a value is allowed to make without being aggregated and the (extra) energy needed to transport such value. At the low energy-near aggregation end is the option of choosing one parent, send a message only to it and see it aggregated, which essentially is the single path approach. At the high energy-high precision end is the option of choosing to aggregate only at the first common progenitor.

Note a relationship with the approach in [MYH06], in which the authors used their intuitions, without any formal analysis proposed a link error type algorithm with an aggregation level of at most two.

The link error type explored in this chapter is somewhat in the middle of such scale. In this approach two or more values are aggregated at a given level (by all of their parents at such level) if and only if all parents of a given sub-aggregate (value) are also parents of all the other sub-aggregates. Note also that it would be trivial to change the link error case algorithm as to always aggregate at the first common progenitors.

9.4.3 Aggregate Generation and Data Reconstruction Algorithms

In previous sections, two algorithms to generate aggregates were introduced and illustrated in simple scenarios. Algorithms 4 and 5 describe how the aggregates can be generated for arbitrarily large WSN. In all presented algorithms, $\#S$ denotes the cardinality of set S.

Even for the simple cases presented before, it was obvious that a rather high number of message exchanges would be necessary. In fact, in both cases the worst-case number of messages grew as a power of two with the number of siblings. However, it should be noted that some of the entries of the aggregation tables are empty sets, while others are repeated. The empty sets occurs, for example, if all offspring that a node can reach can also be reached by at least one of its siblings that are above it in the siblings table. And a repetition may happen if, for example, the set of offspring that a node can reach minus the set of nodes from a given sibling is equal to a similar set excluding another sibling's offspring. Therefore, the algorithms herein presented also include an optimization section, in which the generated aggregate set is pruned of duplicate and empty sets, thus reducing the number and size of exchanged messages.

Eliminating redundancies (duplicates and empty sets) allows to perform a significant reduction of messages and aggregates. In sparse networks (logically sparse, not necessarily physically sparse) this optimization would not make much difference, since it is likely that each node would have a rather distinct set of offspring. This is not a problem since under this circumstances each node would send only one message with the aggregate of its own value and all its children, i.e. sparse networks have less paths in their multipath. However, as the network density grows, there would be more nodes with rather similar offspring tables, thereby the use of this improvement tend to become significant. Nonetheless, it must be stressed that node density control is out of the scope of this chapter.

Algorithm 6 describes how the aggregates can be recovered from the received messages in the case of node errors. For each aggregate that a node manages (Q_k), it must inspect the aggregates sent by each of its children (R_i). Then, all the entries that failed are removed from u. Finally, the correct aggregate value is obtained by taking the rightmost element of

Algorithm 4 Node Error Case: Parent Generated Transmission Lists

```
T_i ← Set of nodes reachable by offspring i
P← {} (Nodes that have been processed)
for {i = 1; i ≤ number of offsprings; i = i + 1} do
  M_i ← {}
  for {j = 0; j < i; j = j + 1} do
    Z ← Permutation of P taken j by j
    for all Z_l (elements of Z) do
      M_i ← M_i ⋃ { T_i \ ⋃_{x∈Z_l} T_x }
    end for
    j ← j + 1
  end for
  P← P ⋃{i}
  \\Remove duplicates and empty sets
  for j = 1; j < 2^{i−1} − 1; j = j + 1 do
    if M_i(j) = {} then
      remove M_i(j)
    else
      for k = j + 1; k < 2^{i−1}; k = k + 1 do
        if M_i(k) =M_i(j) then
          remove M_i(k)
        end if
      end for
    end if
  end for
end for
```

u that has been received from each descendant. As can be verified, the algorithm is not computationally intensive. The operations carried out are relatively simple and the number of iterations depends on the number of aggregates and on the number of children, which are frequently relatively low values.

Algorithm 7 describes how the aggregates can be recovered from the received messages in the case of link errors. The process consists in taking the aggregates sent by each children sequentially, and merge them if they have not been already included from a previous child. To keep track of which values have already been processed, it is used an exclusion list (E). In each step this list is appended with the index of all values that have been correctly received ($RC_i = 1$) but not yet merged.

In terms of computational complexity, the algorithm is similar to the previous one.

9.5 Application of the Algorithm

The first aspect of the application of the algorithms proposed herein is that the link error type of algorithms inherently can cope with node type errors, whereas node type algorithms are more limited in the link error corrections. The node error correction capabilities of link error type algorithms stem from the fact that a node failure is (under the assumption made

Algorithm 5 Link Error Case: Parent Generated Transmission Lists

```
T_i ← Set of nodes reachable by offspring i
N_i ← number of offsprings
for {i = 1; i ≤ N ; i = i + 1} do
  M_i ← {}
  Z ← Arrangements of {0, 1} with repetition taken N−1 by N−1
  for all Z_l (elements of Z) do
    M_i ← M_i ⋃ { T_i ⋂_{Z_l(x)=1} T_x ⋂_{Z_l(x)=0} T̄_x }
  end for
end for
\\Remove duplicates and empty sets
for j = 1; j < 2^{i−1} − 1; j = j + 1 do
  if M_i(j) = {} then
    remove M_i(j)
  else
    for k = j + 1; k < 2^{i−1}; k = k + 1 do
      if M_i(k) =M_i(j) then
        remove M_i(k)
      end if
    end for
  end if
end for
```

herein), in essence, a failure of all of its links.

However, no similar relation can be established between node type algorithms and link failures. In fact, whenever there is a link error the assumptions made for the node error algorithm no longer hold.

Nevertheless, the node error type algorithms, when applied to networks in which node failures are the dominant type of errors, requires less sub-aggregates to be computed (unless a number of simplification of empty and duplicated sets dictates that it requires the same number). Hence, it uses less messages, which in turn is translated into a longer battery life.

With these information at hand it is now possible to make an informed decision about the best algorithm for each approach. Obviously, if there is previous information regarding the dominant type of error in the network, then it should be chosen a matching algorithm. Otherwise, considerations of energy, computational capacity acceptable error rate should define the final choice. Similar trade-offs are to be considered when choosing between the algorithms presented in this article and classical approaches.

9.6 Evaluation

This section presents simulation results to assess the effectiveness of the approaches proposed in this paper. The simulations were carried out in the Matlab® software, with a standard error model, in which the error probability is a function of the distance $P_r = P_e d^{-\alpha}$, with $\alpha = 2$, transmission power was equal in all the nodes, and the probability of error as $(1 - erf(P_r/N_o))/2$.

Algorithm 6 Node Error Case: Aggregate Computation From Children Messages

```
Q_k ← Set of aggregates that a node manages
R_i ← Set of aggregate values received from child i
RC_i ← Binary vector. 1 if received message form child i, 0 otherwise
U_k ← Set of reconstruction path to k^th aggregate
A← {} \\ A_k ← value of aggregate k
for {k = 1; k ≤ #[Q_k]; k = k + 1} do
  A_k ← self
  u← U_k
  for {i = 1; i ≤ #[R_i]; i = i + 1} do
    if not RC_i then
      for {j = i; j ≤ #[U_j]; j = j + 1} do
        remove instances of u related to node j
      end for
    end if
    A_k ← A_k+R_i(arg max u(i) = 1)
  end for
end for
```

In addition to the two algorithms presented in this paper, this section also presents simulations of TAG and the DAG approaches, described in Section 9.2. The TAG approach is the simplest of all. It does not have any type of redundancy, using only simple aggregation, thus it will be used as the base line. The DAG approach has two level deep link error correction capability, hence it will be used to perform a comparison with the link error approach presented herein. To the best of the authors knowledge, there is no approach that can be used to make a direct comparison with the node error case.

The simulation was made with a network with a varying number of nodes, starting from 4 nodes, moving upwards with steps of 4, up to 32 nodes. The nodes were placed randomly

Algorithm 7 Link Error Case: Aggregate Computation From Children Messages

```
Q_k ← Set of aggregates that the node handles
R_i ← Set of aggregate values received from child i
RC_i ← Binary vector. 1 if received message form child i, 0 otherwise
U_k ← Set of reconstruction path to k^th aggregate
A← {} \\ A_k ← value of aggregate k
for {k = 1; k ≤ #[Q_k]; k = k + 1} do
  A_k ← self
  E← {} \\set of excluded aggregates
  for {i = 1; i ≤ #[R_i]; i = i + 1} do
    u← U_k \ U_k(RC_i = 1)
    u← u \ E
    A_k ←aggr{A_k,R_i(u)}
    E← E ⋃ {u}
  end for
end for
```

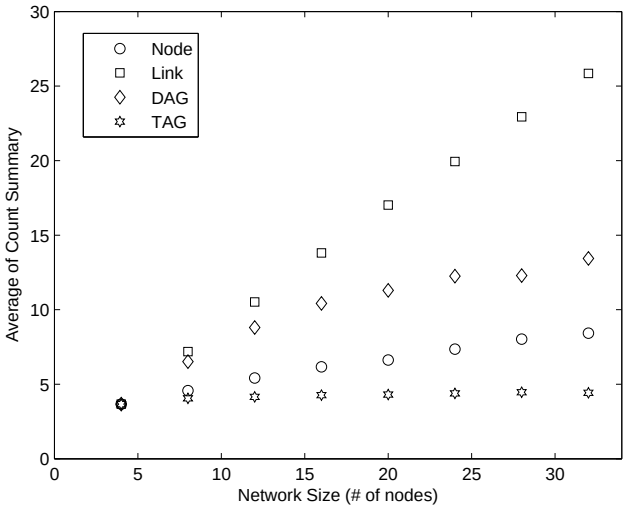


Figure 9.3: Link Error Case.

with a uniform distribution into a square region with an area that provided a constant density 0.81 nodes per square meter. This was done in order to avoid putting all nodes in a small area, which becomes more important as the number of nodes grow, because it makes all nodes able to communicate with each other. This in turn, contradicts with one of the goals behind increasing the number of nodes, i.e., to increase the number of levels of the network. Networks in which, due to the random nature of the placement of the nodes, one or more nodes were not reachable by the sink were discarded.

For each network configuration, 100 counting cycles were performed. There were chosen 100 network configurations for each number of nodes. So in total, there were made 10 thousand cycles for each number of nodes, for each protocol type. The values displayed below are the result of the average taken both over the network configurations and counting cycles.

The sink was chosen randomly, therefore, not being necessarily in one of the edges of the network. Communication parameters, including the average distance between nodes, were tuned to ensure a communication range of about $2m$ with an error probability of 0.1. All nodes were programmed to act as if they had read a 1 from their respective sensor (including the sink) which allowed to have a global view into the number of nodes that were successfully aggregated. Recall that TAG is single path, hence in all case the nodes that were aggregated, were done so once.

Two groups of simulations were performed, one with failures in the link and another with failure in the node, which are depicted in Figure 9.3 and Figure 9.4, respectively.

In Figure 9.3, the link error algorithm behaved as expected having most of its occurrences with the correct reception of all nodes. The DAG approach also presented a reasonable/similar behavior. The node error algorithm performed worst than the DAG approach, in the link error case. Thats due to the fact that, as pointed out in Section 9.4.2, the DAG approach has a two level deep link error correction capabilities and since the network was small it could correct most of the errors. The TAG approach behaved in a poorly manner.

In Figure 9.4, as expected, the node error algorithm had the best performance. The remaining cases also behaved as expected. Moreover, the proposed algorithms showed primacy in their respective target scenarios.

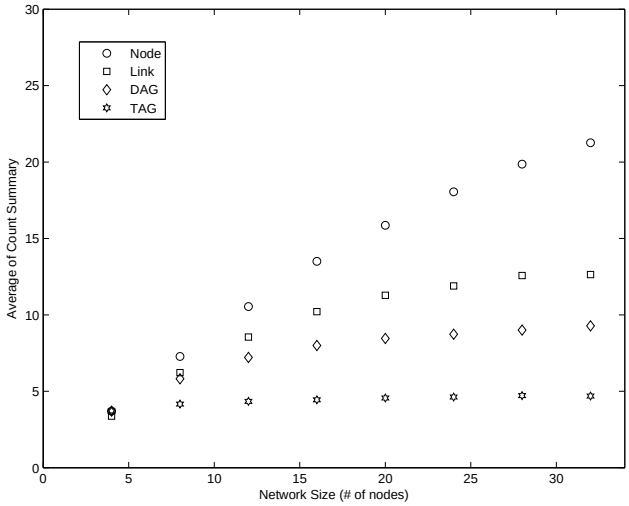


Figure 9.4: Node Error Case.

It should be stressed, however, that although they are indicative of the type of gains that are expected from the use of these techniques, these particular results are highly dependent on the simulation parameters.

9.7 Conclusions

WSN present data transmission/reception reliability issues, which can be dealt with by the use of multi-path routing. Nonetheless, this solution introduces another problem, namely aggregate reliability in the presence of duplicate-sensitive summaries. This chapter presented a mechanism that ensures that the value of such aggregates will be as close as possible to the actual value, namely by taking advantage of redundant paths in the presence of errors and removing duplicates.

The mechanism is centered in the partition of the summaries into several messages that are recomposed to form the best possible message in their path to the sink. Two such algorithms were devised, one best suited for networks in which the dominant failure mode is link failure and another in which the dominant failure mode is node failure.

Both scenarios were simulated and compared to two standard approaches in the literature, having demonstrated a superior performance in their respective failure mode scenarios.

Chapter 10

Conclusions and Future Lines of Study

This work started with the proposal of a novel control architecture for achieving a more efficient use of network resources. The proposed architecture includes the use of estimators and of bulk data transmissions to either attenuate the effects of packet losses or to reduce the overall number of sent packets. Bulk transmissions can be employed at a low or no extra bandwidth cost due to the fact that current fieldbuses have a minimum data length that is significantly larger than the size of a single control value.

As for the extra computational load incurred for the computation of the estimates, its effects can be reduced if most of the computing is done at the controller, which usually has more computational resources. This fact, i.e., all *heavy lifting* done at the controller, limited a number of possibilities of the architecture. The architecture has other advantages, with the most outstanding ones being, 1) the possibility of having the sensor and controller working at different rates. This is important, for example, in systems that due to the low precision of the available sensors, it is necessary to make a rather large number of samples to achieve a reasonable estimate, but have a dynamics that do not require a rapid change in actuation value. And 2) this approach already gives more bandwidth to control tasks in which the respective controlled processes are in a high error state, a property sought after by the co-design community.

The architecture was tested in a bandwidth reduction scenario in which the claims made about were proven to be true.

The remaining set of characteristics where tested in subsequent developments, namely, when the architecture was used in a scenario with a lossy network. Though, only the controller actuator side of the architecture was used, it already demonstrated the capabilities that were argued that the architecture had. Moreover, it was studied the optimal implementation of such scenario.

The study included two possible families of protocols: TCP-like characterized by the presence of a packet reception acknowledgement and UDP-like characterized by the absence of such acknowledgment. For both cases, it was devised the optimal noise filtering algorithm, which for the UDP-like case, it was a modified Kalman filter, whereas for the TCP-like case it was the classical Kalman filter.

As for the controller under this architecture, the certainty equivalence and separation principles still hold, implying that the optimal controller is equal to the optimal classical

controller. The certainty equivalence and separation principles, in essence, state when the state estimate can be used with a separately derived controller, can be used in the place of state variable, while maintaining the optimality.

Tests of this configuration revealed, as in the previous case, what was expected from theoretical considerations. However, this approach still needed a particular underlying architecture. The following case began by looking for controller approaches that were applied to the general architecture. However, the simplest cases were already solved.

The simplest forms of controller design that are architecture independent is a combination of TCP-like vs UDP-like and the output zero vs hold. In the output zero scheme, whenever a message is missing, the actuator applies a zero, whereas in the hold approach, whenever, there is a missing message, the actuator maintains the previous value. As for the possible generalizations, the generalizations of TCP-like versus UDP-like into a spectrum, i.e., the probabilistic reception of acknowledgement messages, is solvable, using modern mathematical tools only for the case in which the acknowledgement is received with probability one, i.e., TCP-like case, because in the remaining cases the certainty equivalence and separation principles do not hold.

The logical choice was to extend the zero vs hold case into a situation in which the actuation value, in periods that no control message is received, it was applied a value that was equal to the last applied value times a matrix that have no eigenvalue greater than one. Under this approach, and under TCP-like it was derived the optimal control value. This produced a cost function that was itself a function of the matrix used in the actuator to compute the next control value.

The minimum of the cost function over the actuator transition matrix revealed that such optimal implies that the actuator ought to keep an estimate of the controller's state estimate. Such actuator estimate is updated whenever the actuator receives a controller message and in periods that there are no controller messages it uses a control value computed based on its state estimate.

This approach differs significantly from the starting point, which was the architectural approach. Similarities and differences between them were explored, with both having advantages and disadvantages.

A different line of inquiry was pursued with the goal of permitting the already existing co-design to perform on a more control optimal manner, namely, to ensure that upon the application of one of the various techniques to improve the resource optimization that include a controller change, the output does not oscillate. The theoretical work culminated with the discovery of an algorithm that allowed to perform the controller change without output oscillation, and it was based on the fact that at the switching instant, the state does not automatical change in order to represent the new situation.

Most of the computation carried out to implement this strategy is done offline, with the only online operation being a matrix-vector multiplication. Hence, the algorithm can be implemented even in systems with relatively low computational capabilities. As for the task model of the underlying real-time system, it does not need to be changed in order to accommodate the task responsible to the change of basis. Though, such a new task needs to be executed before the controller in the respective periods in which there is a controller change.

The new method was tested against several systems and in all of them it performed as expected.

Mechanism for introducing and/or tolerating missing jobs (messages) executions (trans-

missions) from a real-time perspective were also investigated. In particular, the study had a focus into (m, k)–firm systems. However, it was noted that the existing types of (m, k)–firm systems are not the best fit to the type of application investigated in this Thesis, because the existing systems are either 1) probabilistic, which mean that there is the possibility of occurring a relatively long sequence without any controller execution, or 2) its deterministic with a low utilization.

The (m, k)–firm work done in Thesis was concerned with deterministic (m, k)–firm systems with a high utilization. It was shown that the approaches that were heralded as being optimal are, in fact, suboptimal. Furthermore, it was presented a mechanism that outperforms all hitherto presented (m, k)–firm schedulers.

The study of transformations of current Wireless Sensor Networks into networks more conducive for control purposes was also explored. However, even though contemporary wireless networks possess the relevant characteristics to allow for the use on small control networks, they lack the characteristics that would allow them to deal with wireless networks with a larger number of sensors and actuators. In this Thesis it was proposed a mechanism that improves some aspects of wireless networks that are germane for a more precise reading of multi-sensor values. The method allows for the joint use of two other strategies that have already been proposed, i.e., multipath routing and in-network aggregation. However, these two strategies cannot be naively integrated. An evaluation of the proposed scheme showed an improvement over all previous schemes that attempted to achieve similar goals.

10.1 Future Work

Regarding the architecture presented in Chapter 4, there are some benefits of its use that were not experimentally proven, such as the capacity to online and autonomously giving higher utilizations to control loops with higher errors, or its use in multi-rate system. As for the application of the architecture in lossy scenarios, as presented in Chapter 5, only the controller to actuator case was explored. Though, the other case is very similar to the one that was explored, it could be interesting to know if any new behavior emerges when both cases are used simultaneously.

As for the optimal generalizations of the output types, it would be very informative to future engineers to know exactly the computational requirements as well as the type of performance gains of each possible version. Moreover, it might also be interesting to explore possibilities that mix the actuator based compensation, as was done in Chapter 6, with controller based compensation, as in Chapter 5.

A similar line can be followed to augment the work presented in Chapter 7, in which the work done can benefit from a validation in a scenario that used an actual co-design mechanism, as opposed to the performed test that consisted in the introduction of signals in the places that ideal co-design schedulers would introduce.

In the same integration theme, there is a room for the integration of the (m, k)–firm schedulers presented in Chapter 8, with the control techniques presented in this Thesis. However, some challenges may present themselves since, contrary to the classical deterministic (m, k)–firm schedulers, the presented (dynamic) scheduler (when used online) executes jobs probabilistically.

As for extensions of the presented work, the possibility of its application under UDP-like protocols, which has driven a significant part of the research community, is still an open issue.

The work on wireless networks has a rather obvious follow up, namely, its use on its intended framework, that is, the transport medium of a large distributed control network.

Appendix A

Available Fieldbuses

A.1 Controller Area Network

With the growing number of microprocessors present in the average vehicle, Bosch GmbH engineers felt a need in 1983 to come up with a simple protocol that could connect all devices, the Controller Area Network (CAN) protocol [Rob91][Cor08]. Currently, the protocol is on its second version and is used in a wide variety of industrial applications, though the apex of its use was in the late 90's and early 2000's.

CAN is an uncontrolled MAC protocol with a Carrier-Sense Multiple Access (CSMA) non-destructive arbitration scheme. In this MAC there are two signaling bit levels: one recessive and one dominant. Normally the dominant bit is the logical zero and the recessive is the logical one, because it simplifies its physical realization, though the norm does not force so. In CAN arbitration, at the beginning of a frame all nodes that are waiting to transmit data attempt to do so. Whenever a node writes a bit in the bus, it waits a constant amount of time and then reads it back. If a node writes a recessive bit but reads a dominant bit, then it considers to have lost the arbitration and it must stop trying to transmit. However, if a node writes a dominant bit and reads a recessive bit, then it considers that there was an error in the bus and shall start the appropriate response to it.

In order to allow for a larger flexibility, the CAN protocol does not define a physical layer. The bit rates of the CAN bus are specified by the norm and the norm refers to an expectation to see speeds up to 1 Mbps (in part B of the latest specification) [Rob91], or in ISO 11898-2 (1Mbps) and ISO 11819-2 (125 Kbps).

In CAN 2.0 there are two frame formats: Standard Format, kept for compatibility reasons, and Extended Format. In the standard format the identifier is 11 bits long while in extended format it is 29 bits. This increase allows a wider range of devices to be connected using the same bus. It also allows a better structuring of the address space since it is used during the arbitration process. However, this address expansion further reduces CAN data efficiency.

In CAN there are several types of frames: Data frames, Remote Frames, Error frames, and overload frames. A data frame comprises seven fields, namely: start of frame, arbitration field, control field, data field, CRC field, ACK field, end of frame. Figure A.1 depicts the data/remote frame format[1].

A single dominant bit, called start of frame, is used to end the interframe space which is comprised only of recessive bits. The format of the arbitration field depends on the version

[1] image taken from the CAN 2.0 standard.

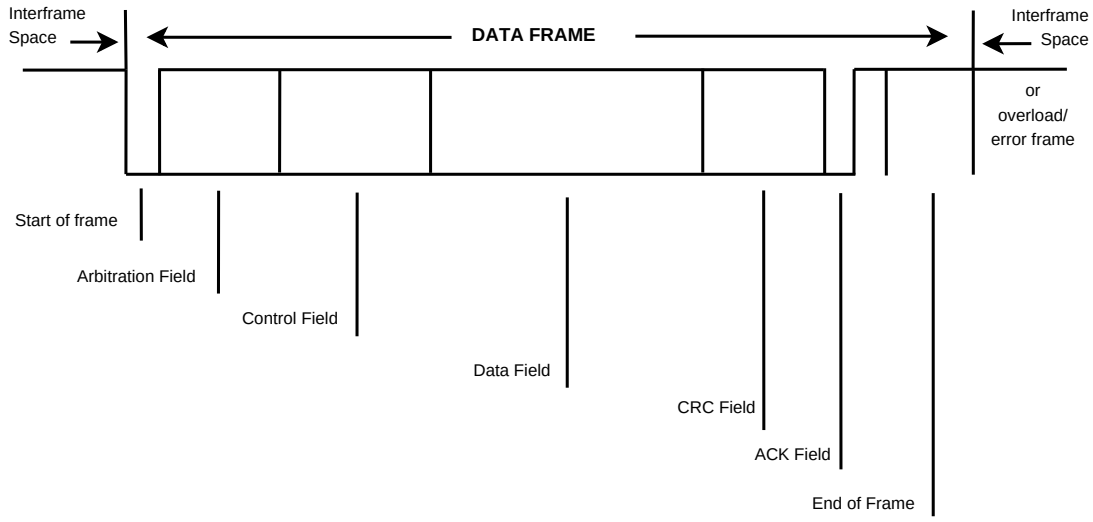


Figure A.1: CAN DATA (Remote) Frames

---

of the CAN protocol. Figure A.2[2] shows the arbitration field of a standard frame.

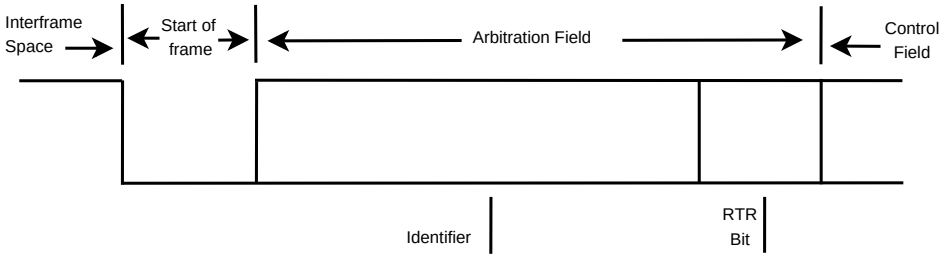


Figure A.2: The arbitration part of CAN DATA frames

The Remote Transmit Request (RTR) bit is used to require the transmission of a data frame by another station. It differs from a data frame by setting the RTR bit as dominant. In such frames the identifier refers to the information that is being requested and the data length field refers to the length of the message that shall be sent in reply.

In the extended format (the arbitration field) the Substitute Remote Request (SRR) bit is placed after the first 11 Most Significative Bits (MSB) and it is as indicated by its name. It is always sent recessive. These first 11 bits are called the base address, whereas the remaining 18 are called the extended address.

The IDentifier Extension (IDE) bit is used to distinguish the version of CAN. In standard format it is sent dominant and in extended format, the IDE bit is sent recessive.

The Control Field comprises six bits. In the standard format the first two are IDE and r0, however in extended format the first two bits are r1 and r0. The reserved bits (r0 and r1) are sent dominant, though the receivers accept them either way. The remaining four bits comprise the data length of the message, with the leftmost bit being the MSB. A data frame can have between zero and eight data bytes.

The Circular Redundancy Check (CRC) field contains the CRC sequence followed by a

[2]image taken from the CAN 2.0 standard

CRC delimiter, which is a single recessive bit. The CRC is computed using a polynomially, i.e, the CRC is the remaining of the division of the destuffed (more on stuffing in this subsection) version of the fields: Start-Of-Frame (SOF), arbitration, control, data (if present) and for the lowest 15 coefficients by zero.

The Acknowledgement field is two bits long and the first bit is always sent recessive, whereas the second bit is sent dominant if the message is error free and recessive otherwise. A transmission is considered to have been an error if no station receives it correctly.

The CAN protocol also includes an error confinement scheme, which consists of two error counters: one for transmission and another for receptions. These counters are initialized with zero and incremented whenever there is an error and decremented whenever a message is successfully transmitted/received. When an error counter achieves a given threshold (128) a node becomes error passive, and another threshold (256) it becomes bus off. In the descending order it is similar, i.e., at 127 a node passes from error passive to error active. Error active is the category for fully functional nodes, error passive is for less functional nodes which cannot generate error frames, but are able to transmit and receive data frames, and they have less privileges. Finally bus off stations cannot use the bus, and future actions of these stations depend on an application reset.

An Error Frame contains two fields, error flags and error delimiter, as shown in Figure A.3. Error flags are superpositions of error flags from multiple stations. Error active node signals the occurrence of an internal error by sending six consecutive dominant bits, which violates the bit stuffing law. Thereby triggering all other error active stations to start transmit their own error flags. This subfield has between 6 and 12 bits. Error passive stations try to signal error conditions by waiting for the bus to be free for six consecutive bits. The Error delimiter is a sequence of eight recessive bits.

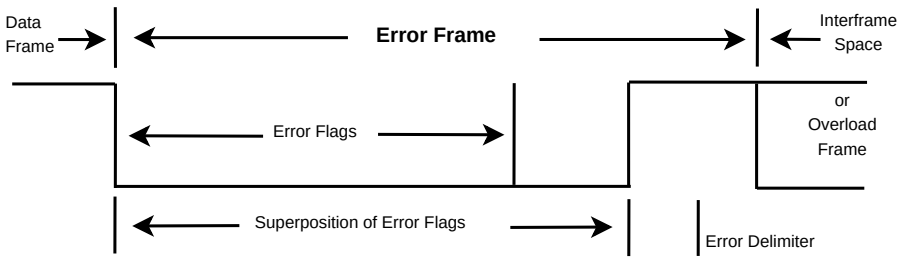


Figure A.3: CAN Error Frame

Data Frames and Remote Frames are preceded by an interframe space. The interframe space consists of an intermission field, and a bus idle at the end. The intermission time is 3 recessive bits long and the only allowed action during this time is the start of an Overload Frame. Error passive stations that have been transmitters of the last message must wait an additional Suspended Transmission time. The suspended transmission time is eight recessive bits long. The bus is said to be idle if after all obligatory interframe space has been fulfilled no station is sending any message.

Overload Frames are sent either if the intermission bits are violated (time between two frames) or a receiver needs to delay either Data or Remote Frames. Overload frames are similar to error frames. Moreover, their fields have similar meanings.

The fields start-of-frame, arbitration, control, data and CRC of the data and remote frames are encoded using the bit stuff encoding scheme, namely: whenever there are five bits

of equally polarity a sixth bit of opposite polarity is inserted with the goal of ensuring that the receiver of the message is capable of extracting the clock information. The remaining fields CRC delimiter, ACK field, and end-of-frame are not encoded in this way. The remaining frames are also not encoded with the bit stuffing scheme. The stuff bits are automatically removed at the receivers.

A.2 Time Triggered CAN (TTCAN)

Though the standard CAN protocol is capable of providing connectivity between multiple nodes, by itself, it does not provide neither jitter nor latency guarantees. Even the highest priority variable (message) in CAN can be blocked when it gets ready to be sent while another message is being sent. To introduce such timing guarantees the TTCAN extension was introduced [FMD+00]. TTCAN was standardized as ISO11898-4.

The TTCAN has two levels: level 1 that provides time-triggered operation based on a reference message broadcasted by a time master, and fault-tolerance; and a level 2 extension, in which a global time base is guaranteed and a continuous drift correction mechanism is ensured.

The most important timing element on TTCAN is the Reference Message. In the level 1 extension this message only contains one byte, whereas at level two extension it contains four bytes used for control, such as the global time information of the current TTCAN master (the concept of time information is explained in this subsection). The amount of time between two successive reference messages is constant (up to jitter) and defined during network power up.

The basic cycle is the time that goes from the start of the transmission of a reference message to the beginning of the next. Within a basic cycle there are three types of windows, namely: exclusive windows which are used to transmit time-triggered messages, arbitration windows which are used to transmit normally arbitrated CAN messages (nodes are not allowed to retry in case of losing the arbitration since this would destroy the basic cycle structure) and free windows. Free windows are used either to allow the expansion of one of the other windows types or to allow for the addition of new nodes, given that the scheduling is done statically.

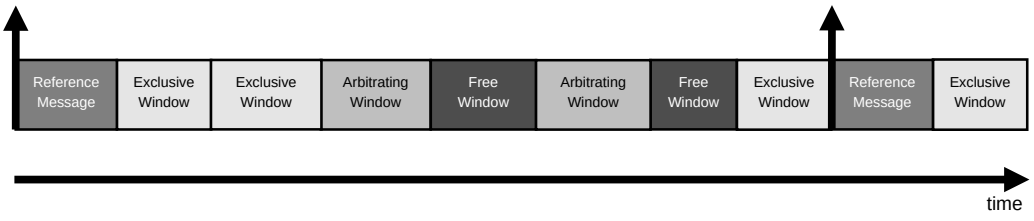


Figure A.4: TTCAN Basic Cycle

TTCAN introduces the System Matrix, depicted in Figure A.5[3], which in essence is a group of basic cycles that is constantly repeated, in which each basic cycle has its own set of messages. Though the messages vary between two basic cycles in the system matrix, the windows appear always in the same order and with the same sizes. These windows are called transmission columns.

[3]Image built based on a figure in [FMD+00]

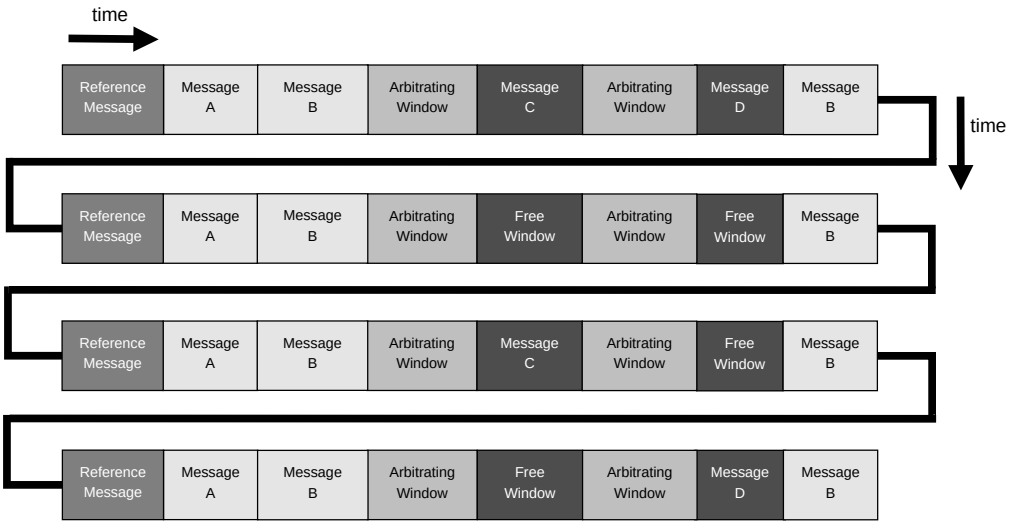


Figure A.5: TTCAN System Matrix

TTCAN also allows merger of two arbitration windows when they come in series. Since in arbitration windows a transmission cannot start if it will end after the end of the corresponding arbitration window (otherwise, it would interfere with the next window), merging two arbitration windows allows to reduce the total amount of guard space.

The basic unit of time in TTCAN is called Network Time Unit (NTU). In level 1 TTCAN the NTU is a bit time, implying that every time difference is measured as the number of bits that could have been sent in that same amount of time. The cycle time is the number of NTUs passed since the reception of the last reference message. This of course, due to clock differences, is very imprecise.

In level 2 TTCAN there is a rather more elaborated mechanism. All stations have an oscillatory circuit that is used to provide the system clock. The oscillatory signal generated by such circuitry goes through a frequency divider that produces the NTU. The frequency divider in turn is fed by a Time Unit Ratio (TUR) that is used for continuous drift correction. TUR is the ratio between the local and network frequencies.

The time master sends its time value (the norm does not state at which moment that should be, it only states that all nodes must be configured to use the same time instant) and all other nodes read that message and compare this time with the time that they had at that moment. The difference between them is the station's local offset. Every time a station wants to know the global time it adds its local time offset to its local time. In addition to the local time offset, stations can also use the aforementioned TUR.

TTCAN expansion has a built-in fault-tolerance mechanism, namely, more than one node can be defined as potential time masters, though at each time only one station will be the master. Potential time masters can become time masters after an arbitration that happens whenever it is noted that the current master is not functioning properly.

The TTCAN also allows the synchronization of the beginning of the basic cycle with an event (as opposed to a time). To this end, in the basic cycle that precedes the synchronization the time master announces that there will be a synchronization so that other masters do not act as if an error has occurred.

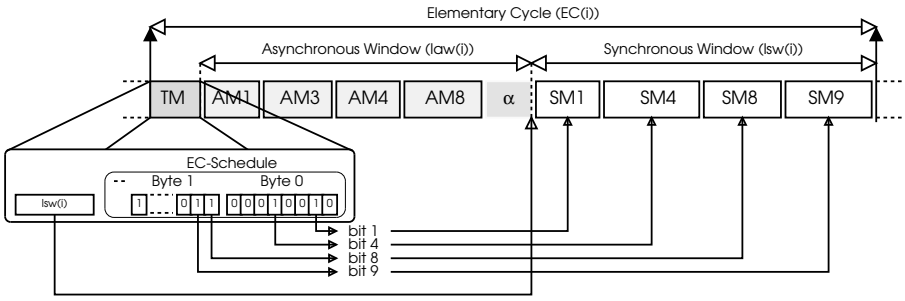


Figure A.6: FTT-CAN Elementary Cycle

A.3 Flexible Time Triggered CAN (FTT-CAN)

FTT-CAN [APF22] introduced a distinction between periodic and asynchronous traffic, between time-triggered and event-triggered messages, and between messages with and without temporal constraints. FTT-CAN does so by these two types of traffic to coexist in the bus. Therefore, FTT-CAN is a step toward the provisioning of QoS guarantees on the CAN bus.

In FTT-CAN time is divided into fixed duration time-slots called Elementary Cycles (EC). Each EC starts with a Trigger Message (TM), followed by an Asynchronous Window (AW) and a Synchronous Window (SW). The TM is sent in the beginning of the EC. The TM is a constant size message that contains the duration of the SW and has a bitmap-coded table of synchronous messages that will be transmitted in the SW.

The AW is a portion of the EC reserved for asynchronous (event-triggered) traffic. In this window the messages are exchanged as in basic CAN, i.e., based in the CAN arbitration previously described. At the end of the AW there is a *guarding window* in which no station can start a transmission, in order to prevent a transmission that would finish after the end of the AW. When this window starts retransmissions are also blocked. The duration of the AW is inferred from the duration of the TM and the SW.

At the end of the EC it is the SW. This window is placed at the end of the EC to allow slower nodes to process the TM. In the SW, all nodes with messages scheduled to be transmitted in the current SW, try to transmit their messages simultaneously. The CAN MAC ensures that only one message is transmitted at a time. The SW is scheduled centrally by a master node, which allows to only schedule messages that fit in SW.

Since the FTT-CAN is a high level protocol, it can be decoupled from its physical layer, i.e., CAN, and applied to a different L1/L2 protocol such as Ethernet, as it was done in FTT over shared Ethernet [PGAB05] and FTT over switched Ethernet [Mar09].

A.4 CANopen

CANopen [Cc00] was created for standardizing the higher layer implementations of CAN and it is a series of protocols of level three and upper of the OSI model. Due to the fact that CANopen is a purely higher level protocol it can be implemented in a different lower level protocol, being used e.g. in Ethernet POWERLINK and EtherCAT.

CANopen devices logically have three main units: the CAN interface (as pointed out above, the interface does not need to be CAN), the device's application, and the interface between the application and the CAN-interface, which is implemented by the Object Dictio-

nary. The object dictionary is a list of parameters and provides access to the capabilities of the device. The CANopen protocol stack ensures the access to the object dictionary of each CANopen device. This protocol stack is usually implemented on the same micro-controller that is used by the application.

The object dictionary is the main component of CANopen. It is basically a grouping of parameters (or objects in the CANopen terminology) that can be accessed through the network in a pre-defined fashion. Each object in the object dictionary has a 16-bit address (index) and a 8-bit sub-index. The object dictionary is divided in index ranges. A single CANopen device can have up to eight device/application profile implementations.

CANopen provides a series of bit timing specifications. These include a bit rate, the bus length, maximum stub length and accumulated stub length. Bit rates vary from 10 kbps to 1 Mbps. CANopen capable devices are required to support at least one of the bit rates. Other bit rates are optional.

CANopen provides two mechanisms to verify the functional integrity of nodes: Heartbeat protocol and the Guarding protocol. In the heartbeat protocol a node periodically broadcasts a message that contains the node's current NMT (network management) status. A node's heartbeat protocol transmission period is adjustable. These messages and all other error related messages are sent using an id obtained by adding 700_h to the ID of the sending node.

All CANopen devices must support the CANopen Management (MNT) slave state machine. MNT is a state machine that sets the communication behavior of CANopen nodes. MNT can be in one of four states: initialization, pre-operational, operational and stopped. The NMT protocol allows CANopen masters to force state transitions in their slaves. A NMT frame has a sender address of zero (highest CAN priority) ; a future state identifier and a slave identifier. If the identifier field is zero, then all slave must transit to the referred state.

The initialization state contains three other substates, namely initializing, reset application and reset communication which work as suggested by their names. Whenever a device is started or is reset it goes to the initializing state. When a device finishes the initialization of its internal hardware, it goes to the reset application substate.

The boot-up protocol is an error handling protocol that is executed every time a node transits from initializing state to pre-operational. In it data fields are set to zero.

Service Data Objects (SDO) provide access to all entries of a CANopen object dictionary, in a confirmed way. It follows a peer-to-peer, client-server model. There are three variants of the SDO protocol: expedited, normal and block; which work as suggested by their names. SDO message transfers are set up during the initialization of the client device.

There are three types of services specified by the protocol. They are, the synchronization (SYNC), emergency object and time stamp object. Beside SDOs, CANopen provides Process Data Objects (PDO) that are used to broadcast status and control messages. PDOs require a set of communication parameters and a set of mapping parameters. These are used, for example, to state the CAN-id that a PDO will use to communicate, and the type of triggering event of the PDO.

There are four types of triggering events: event-(time-)driven, remote request, synchronous transmission (cyclic) and synchronous transmission (acyclic). Event-(time-)driven trigger happens whenever an internal state of a device drives it to send a PDO (a sensor reached a given value or timer went off). A remote request triggered transmission happens whenever a PDO is sent as a response to an RTR. Synchronous (triggered) transmission, both cyclic and acyclic, happen after the transmission of a SYNC message in the bus. The main difference is that cyclic message are sent in every elementary cycle, while acyclic message are trigger by a

device specific cause.

A.5 WorldFIP

WorldFIP [AC98] is a three layer fieldbus protocol centered in the producer-distributer-consumer philosophy. In the physical layer, it is defined as a twisted pair (amended later) and an optical fiber cable.

WorldFIP defines a series of network (first layer) elements, from repeaters, to taps and junction boxes that provide connectivity to a main cable of one or more derivation cables respectively, to diffusion boxes that are active junction boxes, hence, they can create different collision domains. WorldFIP defines two types of stations, namely Locally Disconnected station and Non-Locally Disconnected station.

In the WorldFIP norm there are three transmission speeds for the copper wire: S1 - 31.25 kb/s, S2 - 1 Mb/s, S3 - 2 Mb/s. S2 is the standard speed, S1 and S3 being reserved for special applications. Optical fibers use a speed of 5 Mb/s.

WorldFIP uses the Manchester bit coding scheme, which allows to send clock information along side the data. Besides these two bit coding, there are also used two additional *violations*, called violation− and violation+ respectively, that serve to indicate a special event (or an error) in the bus.

WorldFIP frames are composed of three parts: Frame Start Sequence (FSS), Control And Data (CAD) and Frame End Sequence (FES). The FSS in turn is divided in a Preamble and a Frame Start Delimiter (FSD). Due to this frame coding, 24 bits are added to every transmitted frame. Figure A.7 depicts a WorldFIP frame[4].

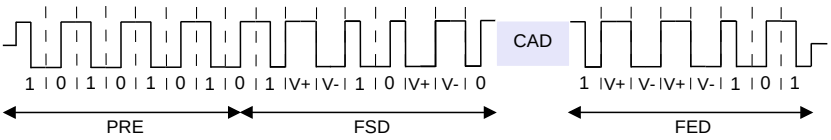


Figure A.7: WorldFIP Frame

The Data Link Layer (DLL) of WorldFIP defines two types of transfers, Variables and Messages. Variables transfer use 16 bit identifiers. One and only one station identifies itself as the producer of a given variable, though many stations can identify themselves as consumers of the same variable. Messages have 24 bits identifiers, and they identify both their source and destination. A station's address (for messaging) includes its unique identifier and that of the bus section that it is in. Messages can be both point-to-point and point-to-multipoint.

WorldFIP supports both cyclical and aperiodic messages. Cyclical messages are pre-configured, i.e., the names and periods of the variables are preset. Aperiodic transfers require an explicit transmission request by the sender every time it has data to send.

The DLL interfaces with the application layer through a series of buffers and System Requests. When an application wants to synchronize the value variable that it produced with the value of the respective variable in the application layer of another node in the bus, it sends a request to the DLL stating the variable that it wants to update and its new value, the DLL responds with a status OK message. This value is stored in an output buffer, that is written by the application and red by the DLL. In turn when the bus arbitrator asks the producer of

[4]figure based on an image in [AC98].

the variable in question for a new value (if it is a periodically transfer, otherwise the station would have to request it), the DLL sends the value that is stored in the corresponding buffer, even if the buffer has not been updated in the last period. After sending the variable to the bus, the DLL sends a signal to the application stating that the variable has been sent.

In case the application layer wants to consume a variable, it indicates so by sending a request to the DLL in which it states which variable it wants to consume. The DLL answers with a status OK message. Whenever there is a variable with the same identifier being sent in the bus, the DLL stores in the corresponding buffer and sends a signal to the application layer informing that the variable in question have been received.

The bus traffic generation follows a different philosophy. Whenever the bus arbitrator detects that it is time to poll for a given variable it sends a request frame asking the producer of the referred variable the send the data. The station that identifies itself as being the producer of the variable sends a response frame with the most recent value of the variable. Symmetrically, the stations that identify themselves as consumers of the variable in question, upon reception of the question frame, prepare to receive the value of the respective variable.

In a WorldFIP bus, only one station can have the function of bus arbitrator, though all stations must be capable of doing so. The bus arbitrator makes a table-based offline static schedule, and it keeps a table of variables, periods, and data types.

The bus arbitrator takes a different procedure for ensuring the exchange of aperiodic variables. First the bus arbitrator broadcasts a request frame for a periodic variable A. If the producer of A has any aperiodic variable waiting to be sent then the producer of A besides responding with the value of A, also requests to send an aperiodic variable. The bus arbitrator puts the producer of variable A in queue of producers that have aperiodic traffic to send. There are two such queues, one for urgent transfer and another for normal transfers.

In a window set for aperiodic traffic, in order to isolate the periodic traffic, the bus arbitrator asks the producer of variable A for a description of the variables that it wants to send. Then the producer of A replies with such information, namely: the identifiers of the variable and their data types; in fact it can reply with up to 64 different variables (assume that one of this variable is called B). In another windows reserved for aperiodic traffic, (now the bus arbitrator uses the same process by which periodic variable are exchanged) the bus arbitrator sends a request frame with the identifier of B. The producer of B responds with the value of B and the consumers of B consume such value.

The producer of a variable is not the only entity capable of starting an aperiodic transfer of the variable in question. They can also be started by other entities, such as a consumer or a third-party.

A.6 FlexRay

FlexRay was created by a consortium of automotive companies interested in having a fieldbus that matched their exact needs. CAN based fieldbuses had low throughput and Ethernet requires a rather complex *middleware* for providing real-time features. FlexRay was created with the purpose of combining the advantages of both. FlexRay is a fieldbus that defines all layers of the OSI model, typically involved in a automotive system, allowing it to be a stand-alone solution to communications.

FlexRay is characterized by a rather high throughput of up to 20 Mbps, if the two channels are used independently. FlexRay accepts both time and event-triggered traffic. It has a

built-in redundancy. It is also fault-tolerant. FlexRay provides mechanisms that improve transmissions determinism.

In FlexRay terminology the communications controller is referred to as the *Electronics Control Unit* (ECU). Each ECU has a local clock, which must have a precision not worse than 1.5%, to ensure that any two ECUs have a relative clock drift of less than 3.0%.

During bit transfers, the sender ECU keeps each bit steady for eight cycles. The receiver ECU keeps a buffer of the last five bits, and the bit logical value is decided based on a majority philosophy.

Each message, called micro-frame in FlexRay, is composed of a Transmission Start Signal (TSS), Frame Start Signal (FSS), *data field*, Frame End Signal (FES) and Transmission End Signal (TES). TSS is logical 0, FSS a 1, FES a 0 and TES is logical 1. Bits that comprise each byte are preceded by two bits: Byte Start Signal 0 and 1 (BSS0 and BSS1) which have the values 1 and 0 respectively.

Receiving ECUs note an end of transmission by detecting the sequence bit0 - bit1 (FES - TES) after the transmission of a byte, and continue receiving bytes if note the sequence bit1 - bit0 (BSS0 - BSS1). Note that each bit correspond to eight cycles.

A bit time is denoted macrotic and is the smallest network time unit. A cycle (of which a bit is composed of eight) is denoted microtic and is used mostly for bit sampling. Nodes synchronize their clocks to the macrotics.

FlexRay frames are structured in header, payload and trailer. The header is 40 bits long and is comprised by the fields Frame Type, Frame ID, Payload Length, Header CRC. The payload field is up to 254 bytes long. The trailer field consists of a 24 bit CRC. Figure A.8 illustrates a FlexRay frame.

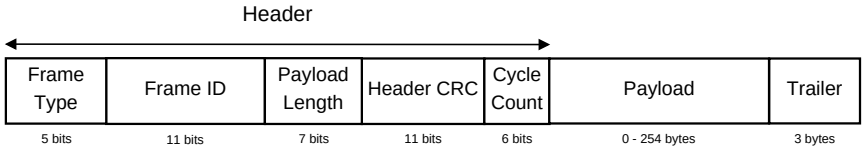


Figure A.8: FlexRay Frame

FlexRay has a network cycle with a offline configurable period, which usually is 1 ms. This cycle is divided in four parts: Static Segment, Dynamic Segment, Symbol Window and Network Idle Time. The static segment is reserved to offline scheduled Time Division Multiple Access (TDMA) communications. The dynamic segment is reserved for event triggered communications and it is subdivided into macrotics. Nodes with higher priorities are assigned macrotics closer to the beginning of the dynamic segment. Whenever a node wants to send data in the dynamic segment it waits until all higher priorities nodes finish sending their data. There is a configurable time-out parameter (in number of macrotic) that tells to lower priority nodes that if a higher priority node did not start sending data during this time-out is because it will not use the dynamic segment in this period. In case the dynamic segment is fully used by high priority nodes, then lower priority nodes will try again in the next network cycle.

The symbol window is where all dynamic configurations take place. This window is reserved to the exchange messages used in network maintenance and to signal special network events. The Network idle time provides a period of known *silence* which is used mostly for cycle synchronization.

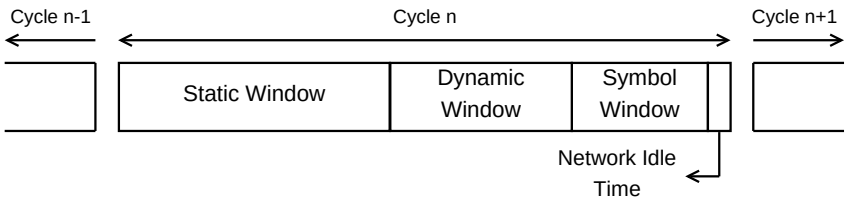


Figure A.9: FlexRay Cycle

FlexRay also defines a network unit called Central Bus Guardian (CBG). CBGs act in ways similar to Ethernet switches, by isolating collision domains or sub-fieldbuses in FlexRay terminology, and all other applications that follow from it. CBG role is mostly policing related, providing resilience to errors such as the babbling idiot effect, i.e., when a (failing) node starts sending frames indiscriminately.

FlexRay has a three level error model similar to the CAN error model.

A.7 Time-Triggered Protocol

The Time-Triggered Protocol (TTP) [KG94] was created as a general, physical and MAC layer independent, way of providing real-time guarantees. Thus, it works both in twisted pair and over a optical fiber, without important assumptions. Though TTP has some features that improve its performance in situations close to those that appear in the automotive industry.

One of the basis of TTP is that offline TDMA scheduling with distributed clock synchronization can be used to reduce the bandwidth used in communications. For example, in TTP, pre-scheduled messages do not require addressing, since all nodes know which message will be exchange at a particular time. Hence, TTP is a protocol with minimal overhead.

TTP supports different modes, which allows the fieldbus to adequate its behavior to the current application conditions. Each mode is cyclic, with messages repeating themselves, after a mode dependent amount of time. TTP has mechanisms for providing fast mode switches which are particularly helpful when changing to an emergency mode. Under TTP, modes are characterized by their constituent messages (format, sending node, receiving mode) and respective schedule/dispatch.

A node is declared to be faulty if in the last TDMA round, no message was sent in the slot reserved for it. A global controller state is maintained. This state contains a list of active and faulty nodes, in its membership field. Whenever there is a discrepancy in the membership field, all nodes participate in a system-wide scrutiny in which for each supposedly fault-node, it is taken the opinion of the majority. Other fields of the controller state are the mode and time fields.

TTP frames comprise a start-of-frame, control field, data field and CRC. Beside these there are interframe spaces. The Control field is 8-bit long, with one initialization bit, three mode change bits and four acknowledgement bits.

By virtue of being time-triggered, TTP has an implicit sending acknowledge. As for the reception acknowledgement, which regards the correctness of the transmitted data, only one node sends an acknowledgement per message because TTP assumes that situations in which the error occurs to a group of nodes and not to others are very rare. TTP also consider that nodes are *fail silent*, i.e., faulty nodes are not capable of transmitting messages.

The data field is at maximum eight bytes long. It can consist of one or more concatenated application layer message, but it is of fixed length, and this length is declared in the mode. The CRC field is two bytes long. TTP has a number of fault-tolerant features. It also describes four classes of system configuration according with different levels of fault-tolerance.

A.8 Ethernet-based Fieldbuses

In the the 80's and 90's, the fieldbus arena was dominated by protocols that offered relatively small bandwidths. At that time, this was not an inconvenient because the existent applications required low bandwidth and there where no affordable microcontrollers that could process high bit rate messages in real-time.

As time went by, the number of nodes in typical industrial control network surged. Moreover, quality standards improved, thus requiring an improvement of the quality of control, which implies an increase of the fieldbus bandwidth. This led to a demand for higher bandwidth fieldbuses.

Under these conditions, Ethernet appeared as a good option for the communication medium, i.e., physical and MAC, of the next generation of fieldbuses. Due to its high availability, being the *de facto* LAN standard, low cost of implementation, and related advantages. Nonetheless, Ethernet did not provide some essential fieldbus features, e.g. low latency and small and bounded jitter. Thus, an intensive research effort was devoted to adapt Ethernet to the constrains of industrial control systems, giving origin to the so called Real-time Ethernet Protocols (RTE).

The next subsections present an overview of the representative RTE protocols. A more in-depth analysis of the use of Ethernet as a fieldbus, plus some historical perspective, is presented in [ATV01].

A.9 Time-Triggered Ethernet

Time-Triggered Ethernet (TT-Ethernet) [KAGS05] is an Ethernet based fieldbus. It was developed at the Technical University of Vienna, and revolves around two types of traffic: Time-Triggered (TT) and Event-Triggered (ET).

General purpose switches use two different philosophies to forward traffic: *cut-through* and *store-and-forward*. In cut-through a message is forwarded as soon as the switch decodes its destination, i.e., after receiving the initial ID fields. In store-and-forward a message is received, decoded, and stored until it is scheduled to be sent. Cut-through has the advantage of offering lower latencies, however store-and-forward allows switches to detect errors, that can only be detected by checking the CRC field, preventing their propagation.

TT-Ethernet switches handle ET as any other Ethernet traffic, that is: with *store-and-forward* with any possible collision among them being resolved by the Ethernet MAC. The TT traffic is handled with a *cut-through* philosophy, that is: any TT message that arrives preempts any ongoing ET message that may be in transmission.

Collision among TT messages are avoided/eliminated by the use of a TDMA based static scheduling table (dispatcher), that allocates exclusive transmission slots for each message.

Preempted ET message are retransmitted from their beginning. The preemption mechanism reduces the bandwidth efficiency. For tackling this issue it was proposed in [MAR08] that the ET messages were fragmented in a way that they always fit between two TT messages,

thus avoiding the retransmission of the initial bytes. The fragmentation/defragmentaion process is transparent to higher layers.

Beside standard Ethernet components, TT-Ethernet defines a Bus Guardian which is a network element, that may be collocated with the TT-Ethernet Switch, be a stand-alone element or being instantiated inside a host. The Bus Guardian is used in safety-critical application in order to guarantee a proper functioning of the fieldbus.

TT-Ethernet switches distinguish the TT traffic based on the Ethertype pattern (0x88d7), assigned by the Ethernet standard authority of the IEEE.

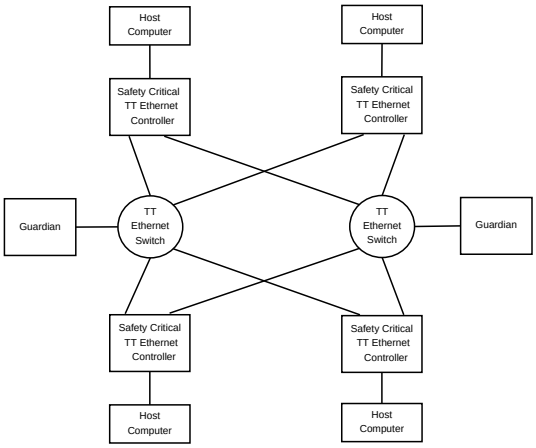


Figure A.10: Safety Critical TT-Ethernet Network

TT-Ethernet support various classes of time-triggered traffic, that allows to a differentiated service. It also has several types of messages, namely : ET message, Free-Form TT message, TT startup message, TT synchronization message, Unprotected TT message and Protected TT message, which work as suggested by their names. Figure A.10 depicts a TT-Ethernet network.

A.10 EtherCAT

Most fieldbuses send many address and control bytes, and relatively a few data bytes. Hence, they have a low data transfer efficiency. This is true even for TT-Ethernet, in which a small control message is normally encapsulated (with padding) in a long Ethernet frame.

EtherCAT [Groa] was created to solve this issue. It does so by encapsulating several messages in a single Ethernet frame. EtherCAT protocol has its own Ethertype. When a EtherCAT node sees the Ethertype in question, it decodes the frame, which consists of various EtherCAT messages, properly addressed and extract (inserts) the data that is addressed to (produced by) it. EtherCAT nodes decode the message while transmitting them simultaneously, in a scheme that resembles the cut-through philosophy, thus requiring special-purpose switches.

This scheme induces an open ring: it starts with the master, which sends a frame with all the requests, goes through the nodes where each node reads either the request for data or an actuation value until the message reaches the end of the ring; in the way back, nodes that received data requests, write down the respective values.

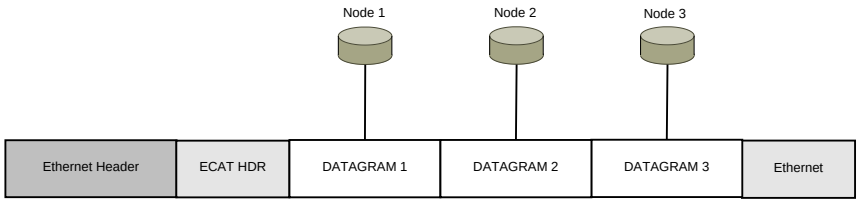


Figure A.11: EtherCAT Frame

Due to its topology, EtherCAT natively supports broadcast. However, this type of broadcast can only be made from a node closer to the master (up the chain) to a node further away from the master (down the chain). To broadcast, or send a regular message, to nodes up in the chain, a node has to send a message to the master identifying itself and the destination node(s).

EtherCAT provides a clock synchronization mechanism that is based in the IEEE 1588 standard [802]. Optionally, slave nodes can time-stamp messages when they are going through them in both direction (from the master and to the master). This allows the master to know how long a message takes to get to a particular node.

EtherCAT also allows the exchange of messages between nodes in different networks, hence going through routers. To achieve this, the EtherCAT message is encapsulated into UDP/IP.

EtherCAT has built-in support for many different topologies, including star, bus or line, and tree, with no limitation to the number of components that can be placed on the topology. Nevertheless, the topology must allow the existence of a central master, which is accomplished by breaking complex topologies into smaller ones.

A.11 Ethernet POWERLINK

Ethernet POWERLINK (EPL) [PSSV09] [Gmb] was developed, in 2001, by Bernecker & Rainer GmbH, which in its essence is the Data Link Layer (DLL) of CANopen on top of Ethernet, hence it is also referred to as *CANopen over Ethernet*. In 2003, version 2.0 was published as an open standard by the EPL Standardization Group (EPSG) [Grob], and was made international standard IEC 61784.

EPL uses the Ethernet stack, thus it inherited Ethernet physical medium, though the 100Base-X Half-Duplex transmission mode is the chosen mode.

In EPL there are two types of nodes, Managing nodes (MN) and Controlled nodes (CN). MNs are usually controllers and have a considerably amount of computing power. CNs are usually sensors or actuators. In a network there must be one and only one active MN.

EPL has a TDMA-based medium access control which is divided in four parts, as depicted in Figure A.12[5]. The first is the Start of Cycle, in which the MN sends a message that signals the start of cycle. Right after it the system enters in the second phase.

The second phase is the isochronous phase. In this phase MN polls one specific CN at a time using PReq (Poll Request). After an elaboration time the polled CN answers with a PRes (Poll Response) that is broadcasted to the whole network. After receiving a PRes the MN waits for a quiet time before sending another PReq. EPL allows the MN not to poll all

[5]Figure partially based on [PSSV09]–Figure 2

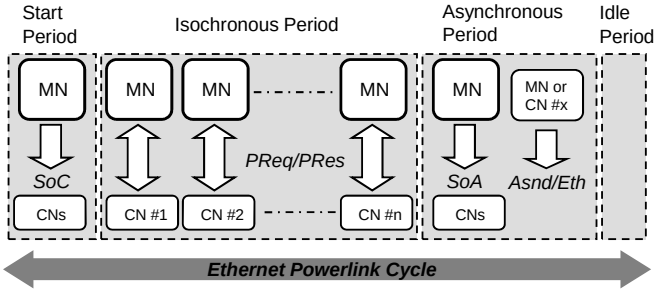


Figure A.12: Ethernet POWERLINK Cycle

CN on each cycles. In fact, in EPL there are two communication classes: continuous which occurs every cycle, and multiplexed which occurs once every n cycles ($n > 1$).

After the Isochronous period, the MN broadcasts the Start of Acyclic (SoA) frame, which informs the CNs of the start of the Asynchronous Period. In this period only one node may transmit (either MN or on of the CN), though during the isochronous period several CNs may send requests to transmit in the next PRes frames. Then, in the asynchronous period the MN schedule them using a predefined mechanism/priorities. The node that is granted the right to transmit is informed by the MN through the SoA frame. There are two possible categories of frames sent in this period, namely EPL AsyncSend and legacy Ethernet.

At the end of a cycle it is located the idle period, which is placed there to guarantee that all pending asynchronous frames end before the start of the next cycle.

As other Ethernet based fieldbuses, EPL has its own Ethertype (0x88AB). EPL frames have four fields. The first three are message type, source node and destination node. All of them are one byte long. The fourth field contains the data. Figure A.13 represents the EPL frame format[6].

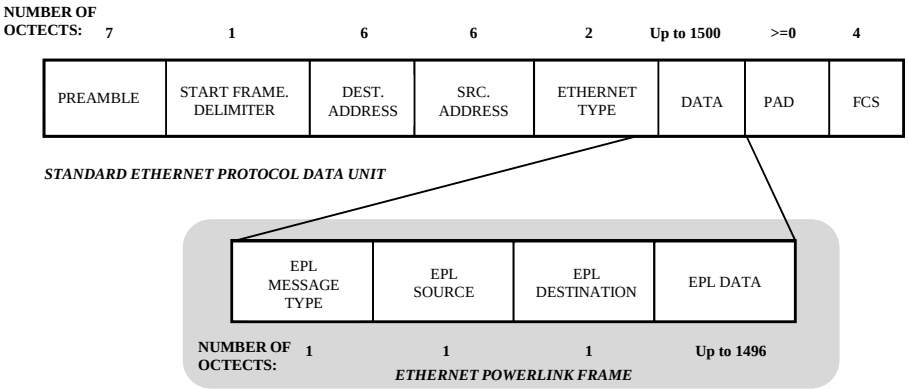


Figure A.13: Ethernet POWERLINK Frame

Even though EPL uses traditional Ethernet hubs and switches, it can be set up in more configuration than bare Ethernet. That is due to the inherent collisions avoidance provided by TDMA, EPL topologies are not limited by the Ethernet maximum Round-Trip-Time (RTT)

[6]Figure based on [PSSV09]–Figure 3

of 5.12 μs.

EPL has two built-in classes of redundancy mechanisms: medium redundancy and MN redundancy. To increase the redundancy of the network EPL uses two cables per node, forming two physical networks, which performs as one single logical network. Another form of medium redundancy is accomplished through the use of the ring topology.

The MN redundancy is obtained by the introduction of several Redundant Managing Nodes (RMNs) of which one is elected to be the Active Managing Node (AMN). Whenever the current AMN fails, the RMNs elect a new AMN and in the following cycle it is already in charge of sending the appropriate messages.

A.12 PROFINET

The main idea of PROFINET is to create a device independent, user-friendly way of supervising an automation network. To this end PROFINET uses a Component Object Model (COM) to identify the different devices in the network, and a Remote Procedure Call (RPC) to execute certain routines in other devices. PROFINET provides a distributed model of the devices that are in the network, through DCOM. In this model, devices use the Interface Definition Language (IDL), that allows them to export a model of their functionalities, thus making their services available to other devices.

PROFINET devices provide a vendor specific XML file with the definition of its capabilities and limitations, allowing the controller to build an image of the network and to correctly initialize them.

In PROFINET there are three types of devices. IO devices are responsible for transducing values to/from the analogue world. Controller devices that receive data from input IO devices, process it, and then send data to output IO devices. Finally, there are supervisor nodes, which provide network engineers with user-friendly access to configuration, maintenance and monitoring.

In PROFINET there are three types of traffic, namely Non-Real-Time (Non-RT), Real-Time (RT) and Isochronous Real-Time (IRT). Non-RT traffic is intended to be used in communications that do not require real-time guarantees, e.g. ftp file transfer. RT traffic is best suited for situations in which real-time guarantees are required, but they are not very restrictive. IRT traffic is best suited for real-time communications with tight time bounds. RT and IRT are also called Class 1 and class 2 traffic respectively.

PROFINET devices uses TDMA for accessing the medium. Time is divided in a series of cycles. Each cycle is composed of three parts. The first one is reserved to IRT traffic, the second is reserved for RT traffic and the last is reserved for NRT traffic.

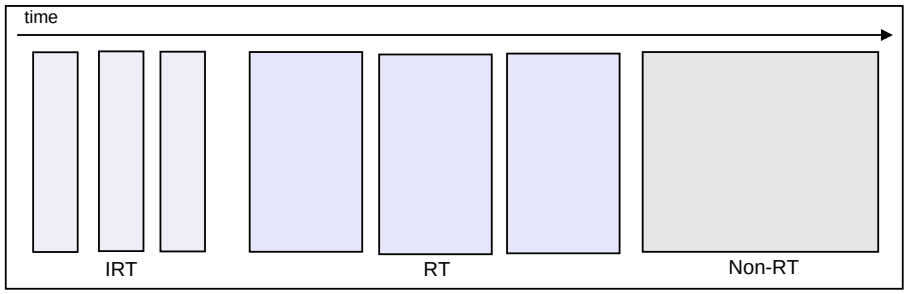


Figure A.14: PROFINET CYCLE

Due to their characteristics each type of traffic has a range of possible/recomended periods and associated jitters. A typical Non-RT traffic has a period in the range of 100 ms and a jitter as large as its period. RT traffic has a period in the order of 10 ms and jitter up to 15% of the period. IRT can have a periods down to 250 μs, though it normally is in the 1 ms and virtually no jitter.

There are two variants of the PROFINET protocol: PROFINET CBA (Component Based Automation) and PROFINET IO. PROFINET CBA emerged first and introduced many of the concepts of PROFINET, device abstraction as suggested by the the name. It provides Non-RT and RT (class 1) type of traffic. PROFINET IO, is more focused in real-time applications, and it was introduced due to a *demand* for motion control, which has tight time constraints. PROFINET IO supports IRT and RT types of traffic.

Acronyms

BIBO Bounded-Input Bounded-Output, page 19

C Worst Case Execution Time, page 54

CBS Constant Bandwidth Server, page 63

CPU Central Processing Unit, page 50

CSMA Carrier Sense Multiple Access, page 66

CSMA-CA CSMA Collision Avoidance, page 66

CSMA-CD CSMA Collision Detection, page 66

CSMA-NBA CSMA Non-destructive Bitwise Arbitration, page 66

D Deadline, page 54

DAC Digital-to-Analog Conversion/Converter, page 16

DMS Deadline Monotonic Scheduling, page 58

EDF Earliest Deadline first, page 60

FCFS First Come First Served, page 59

GPC Generalized Predictive Control, page 40

ICSS Integrated Communication and Control Systems, page 50

ISO International Standard Organization, page 63

LLF Least Laxity First, page 61

LQC Linear quadratic control, page 34

LSF Least Slack First, page 61

LTI Linear Time Invariant, page 11

MAC Medium Access Control, page 66

MIMO Multiple-Input Multiple-Output, page 11

mit Minimum Inter-arrival Time, page 54

MPC Model Predictive Control, page 40

OSI Open Systems Interconnection, page 63

PCP Priority Ceiling Protocol, page 56

PDC producer-distributor-consumer, page 65

PI Proportional Integrator, page 44

PID Proportional Integral Derivative, page 44

PIP Priority Inheritance Protocol, page 56

QoS Quality of Service, page 49

RMS Rate Monotonic Scheduling, page 56

RMS Root Mean Square, page 20

SISO Single-Input Single-Output, page 11

SOD Send-on-Delta, page 43

T Minimum Inter-arrival Time, page 54

TBS Total Bandwidth Server, page 63

U CPU utilization factor, page 57

WCET Worst Case Execution Time, page 54

Bibliography

[802] IEEE 802.3. Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications.

[AB98] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *RTSS '98: Proceedings of the IEEE Real-Time Systems Symposium*, page 4, Washington, DC, USA, 1998. IEEE Computer Society.

[ABRW91] N.C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard real-time scheduling: The deadline-monotonic approach. *Proc. IEEE Workshop on Real-Time Operating Systems and Software*, pages 133–137, 1991.

[AC98] J.De Azevedo and N. Cravoisy. Worldfip specifications. Technical report, Universita di Roma, http://www.dia.uniroma3.it/autom/RetieSistemiAutomazione/PDF/WorldFipProtocol.pdf, 1998.

[AKUMK04] J.N. Al-Karaki, R. Ul-Mustafa, and A.E. Kamal. Data aggregation in wireless sensor networks - exact and approximate algorithms. In *High Performance Switching and Routing, 2004. HPSR. 2004 Workshop on*, pages 241 – 245, 2004.

[AN11] Khaled Ammar and Mario A. Nascimento. On the use of histograms for processing exact aggregate queries in wireless sensor networks. *International Workshop on Data Management for Sensor Networks*, 8, 2011.

[APF22] Lus Almeida, Paulo Pedreiras, and Jos Alberto Fonseca. The ftt-can protocol: Why and how. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 49(6):1189–1201, December 22.

[APLW02] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole. Analysis of a reservation-based feedback scheduler. In *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, pages 71–80, 2002.

[APM05] A. Antunes, P. Pedreiras, and A.M. Mota. Adapting the sampling period of a real-time adaptive distributed controller to the bus load. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, volume 1, pages 4 pp. –1084, september 2005.

[As03] Babak Azimi-sadjadi. Stability of networked control systems in the presence of packet losses. In *in Proceedings of 2003 IEEE Conference on Decision and Control*, pages 676–681, 2003.

[AT09] Adolfo Anta and Paulo Tabuada. On the benefits of relaxing the periodicity assumption for networked control systems over can. In *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, RTSS 09, pages 3–12, Washington, DC, USA, Dec 2009. IEEE Computer Society.

[ATV01] M. Alves, E. Tovar, and F. Vasques. Ethernet goes real-time: a survey on research and technological developments. Technical report, ISEP-IPP, HURRAY-R-2K01, 2001.

[Bat06] S. Battilotti. Robust detectability from the measurements plus state feedback stabilization imply semiglobal stabilization from the measurements. *Automatic Control, IEEE Transactions on*, 51(9):1542 –1547, sept. 2006.

[BB08] D. Bernardini and A. Bemporad. Energy-aware robust model predictive control based on wireless sensor feedback. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3342–3347, 2008.

[BBB03] E. Bini, G.C. Buttazzo, and G.M. Buttazzo. Rate monotonic analysis: the hyperbolic bound. *Computers, IEEE Transactions on*, 52(7):933–942, Jul 2003.

[BC01] G. Bernat and R. Cayssials. Guaranteed on-line weakly-hard real-time systems. In *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, pages 25 – 35, dec. 2001.

[BDSV12] M. Beschi, S. Dormido, J. Sanchez, and A. Visioli. A new two degree-of-freedom event-based pi control strategy. In *American Control Conference (ACC), 2012*, pages 2362–2367, 2012.

[Ber07] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II.* Athena Scientific, 3rd edition, 2007.

[BGPS06] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *Information Theory, IEEE Transactions on*, 52(6):2508 – 2530, june 2006.

[BGSS08] M. Ben Gaid, D. Simon, and O. Sename. A convex optimization approach to feedback scheduling. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 1100–1105, 2008.

[Bla34] H.S. Black. Stabilized feedback amplifiers. *Bell Syst. Tech. J.*, 13:1–18, 1934.

[BLCA02] G.C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *Computers, IEEE Transactions on*, 51(3):289–302, mar 2002.

[Bod40] H.W. Bode. Relations between amplitude and phase in feedback amplifier design. *Bell Syst. Tech. J.*, 10:421–454, 1940.

[BS72] R. H. Bartels and G. W. Stewart. Solution of the matrix equation ax + xb = c [f4]. *Commun. ACM*, 15(9):820–826, Sep 1972.

[BSJ07] Lei Bao, M. Skoglund, and K.H. Johansson. A scheme for joint quantization, error protection and feedback control over noisy channels. In *American Control Conference, 2007. ACC '07*, pages 4905 –4910, july 2007.

[Bur91] A. Burns. Scheduling hard real-time systems: a review. *Softw. Eng. J.*, 6(3):116–128, 1991.

[But97] Giorgio C. Buttazzo. *Hard Real-Time Computing: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[BVDS11] M. Beschi, A. Visioli, S. Dormido, and J. Sanchez. On the presence of equilibrium points in pi control systems with send-on-delta sampling. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 7843–7848, 2011.

[BVM07] Giorgio Buttazzo, Manel Velasco, and Pau Marti. Quality-of-control management in overloaded real-time systems. *IEEE Trans. Comput.*, 56:253–266, Feb 2007.

[CB08] A. Chaillet and A. Bicchi. Delay compensation in packet-switching networked controlled systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3620–3625, 2008.

[Cc00] CAN-cia. Canopen protocol. *http://www.can-cia.org/index.php?id=84*, 3, 2000.

[CCC+10] Wei Chen, Ruizhi Chen, Yuwei Chen, H. Kuusniemi, and Jianyu Wang. An effective pedestrian dead reckoning algorithm using a unified heading error model. In *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pages 340–347, 2010.

[CCP10] Hyeonjoong Cho, Yongwha Chung, and Daihee Park. Guaranteed dynamic priority assignment schemes for real-time tasks with (m, k)-firm deadlines. *ETRI Journal*, 32:422–429, june 2010.

[CE00] A. Cervin and J. Eker. Feedback scheduling of control tasks. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 5, pages 4871–4876, 2000.

[CEBÅ02] Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik Årzén. Feedbackfeedforward scheduling of control tasks. *Real-Time Syst.*, 23:25–53, Jul 2002.

[CeSP11] Milton Armando Cunguara, Tomás António Mendes Oliveira e Silva, and Paulo Bacelar Reis Pedreiras. A loosely coupled architecture for networked control systems. In *Industrial Informatics, IEEE 9th International Conference on*, 2011.

[CF03] Martin Corless and Arthur Frazho. *Linear Systems and Control — An Operator Perspective*. Marcel Dekker, Ink, 2003.

[CF13] G.C. Calafiore and L. Fagiano. Robust model predictive control via scenario optimization. *Automatic Control, IEEE Transactions on*, 58(1):219–224, 2013.

[CGKL11] Suan Khai Chong, Mohamed Medhat Gaber, Shonali Krishnaswamy, and Seng Wai Loke. Energy-aware data processing techniques for wireless sensor networks: A review. *T. Large - Scale Data - and Knowledge-Centered Systems*, 3:117–137, 2011.

[CGMR05] Graham Cormode, Minos Garofalakis, S. Muthukrishnan, and Rajeev Rastogi. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 25–36, New York, NY, USA, 2005. ACM.

[Cha03] C.D. Charalambous. Stochastic nonlinear minimax dynamic games with noisy measurements. *Automatic Control, IEEE Transactions on*, 48(2):261 – 266, feb. 2003.

[CHH85] Gary B. Lamont Constantine H. Houpis. *Digital Control Systems: Theory, Hardware, Software*. McGraw-Hill series in electrical engineering. McGraw-Hill, 1985.

[CHL06] Thidapat Chantem, Xiaobo Sharon Hu, and M.D. Lemmon. Generalized elastic scheduling. In *Real-Time Systems Symposium, 2006. RTSS '06. 27th IEEE International*, pages 236–245, dec 2006.

[CLKB04] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 449 – 460, march-2 april 2004.

[CMV08] Silvio Croce, Francesco Marcelloni, and Massimo Vecchio. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *Comput. J.*, 51:227–239, March 2008.

[CMVC06] Rosa Castane, Pau Marti, Manel Velasco, and Anton Cervin. Resource management for control tasks based on the transient dynamics of closed-loop systems. In *Proceedings of the 18th Euromicro Conference on Real-Time Systems*, pages 171–182, 2006.

[Con10] Online Contributors. Osi model. online: en.wikipedia.org/wiki/OSImodel, 2010.

[Cor08] Steve Corrigan. Introduction to the controller area network (can). Application report, Texas Instruments, July 2008.

[CPA+10] T. Cucinotta, L. Palopoli, L. Abeni, D. Faggioli, and G. Lipari. On the integration of application level and resource level qos control for real-time applications. *Industrial Informatics, IEEE Transactions on*, 6(4):479–491, 2010.

[CPX06] Jen-Yeu Chen, G. Pandurangan, and Dongyan Xu. Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis. *Parallel and Distributed Systems, IEEE Transactions on*, 17(9):987–1000, 2006.

[CQG11] Wei Chen, L. Qiu, and Guoxiang Gu. Stabilization of networked multi-input systems over awgn channels with channel resource allocation. In *Control and Automation (ICCA), 2011 9th IEEE International Conference on*, pages 680–685, 2011.

[CSWS04] Jiming Chen, Yeqiong Song, Zhi Wang, and Youxian Sun. Equivalent matrix dbp for streams with (m,k)-firm deadline. In *Industrial Electronics, 2004 IEEE International Symposium on*, volume 1, pages 675 – 680 vol. 1, may 2004.

[CvdWHN06] M. Cloosterman, N. van de Wouw, M. Heemels, and H. Nijmeijer. Robust stability of networked control systems with time-varying network-induced delays. In *Decision and Control, 2006 45th IEEE Conference on*, pages 4980–4985, 2006.

[CVMC10] Anton Cervin, Manel Velasco, Pau Martí, and Antonio Camacho. Optimal online sampling period assignment: Theory and experiments. *IEEE Transactions on Control Systems Technology*, 19(99):1–9, 2010.

[CZQ12] Wei Chen, Jianying Zheng, and L. Qiu. Linear quadratic optimal control of continuous-time lti systems with random input gains. In *Information and Automation (ICIA), 2012 International Conference on*, pages 241–246, 2012.

[Des85] B. Desarthy. Timing constrains of real-time systems: constructs for expressing them, methods for validating them. *IEEE Transactions on Software Engineering*, 11(1):80–86, 1985.

[DGM+04] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 588–599. VLDB Endowment, 2004.

[DKR04] Antonios Deligiannakis, Yannis Kotidis, and Nick Roussopoulos. Hierarchical in-network data aggregation with quality guarantees. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Bhm, and Elena Ferrari, editors, *EDBT*, volume 2992 of *Lecture Notes in Computer Science*, pages 658–675. Springer, 2004.

[DLG10] Shi-Lu Dai, Hai Lin, and S.S. Ge. Scheduling-and-control codesign for a collection of networked control systems with uncertain delays. *Control Systems Technology, IEEE Transactions on*, 18(1):66–78, 2010.

[dM10] Antnio Pereira de Melo. *Teoria dos Sistemas de Controlo Lineares*. Universidade de Aveiro, 2 edition, 2010.

[DSW06] A.G. Dimakis, A.D. Sarwate, and M.J. Wainwright. Geographic gossip: efficient aggregation for sensor networks. In *Information Processing in Sensor*

Networks, 2006. IPSN 2006. The Fifth International Conference on, pages 69–76, 2006.

[DVF12] M.L. Della Vedova and T. Facchinetti. Feedback scheduling of real-time physical systems with integrator dynamics. In *Emerging Technologies Factory Automation (ETFA), 2012 IEEE 17th Conference on*, pages 1–8, 2012.

[EAFR12] M. Elbes, A. Al-Fuqaha, and A. Rayes. Gyroscope drift correction based on tdoa technology in support of pedestrian dead reckoning. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 314–319, 2012.

[EHA00] Johan Eker, Per Hagander, and Karl-Erik Årzén. A feedback scheduler for real-time controller tasks. *IFAC, Control Engineering Practice*, 8:1369–1378, 2000.

[ERA07] C. Ebenbauer, T. Raff, and F. Allgower. Certainty-equivalence feedback design with polynomial-type feedbacks which guarantee iss. *Automatic Control, IEEE Transactions on*, 52(4):716 –720, april 2007.

[ESM07] M. Epstein, Ling Shi, and R.M. Murray. Estimation schemes for networked control systems using udp-like communication. In *Decision and Control, 2007 46th IEEE Conference on*, pages 3945–3951, 2007.

[ESTM08] Michael Epstein, Ling Shi, Abhishek Tiwari, and Richard M. Murray. Probabilistic performance of state estimation across a lossy network. *Automatica*, 44(12):3046 – 3053, 2008.

[Eva50] W.R. Evans. Control system synthesis by root locus method. *Trans. AIEE*, 69:1–4, 1950.

[Eve10] C. Evequoz. Guaranteeing optional task completions on (m,k)-firm real-time systems. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 1772–1779, 2010.

[FLS06] K.-W. Fan, S. Liu, and P. Sinha. On the potential of structure-free data aggregation in sensor networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, april 2006.

[FMD+00] Thomas Fhrer, Bernd Mller, Werner Dieterle, Florian Hartwich, Robert Hugel, Michael Walther, and Robert Bosch GmbH. Time triggered communication on can (time triggered can- ttcan). Protocol description, Robert Bosch GmbH, 2000.

[FNSLY08] F. Flavia, Jia Ning, F. Simonot-Lion, and Song YeQiong. Optimal on-line (m,k)-firm constraint assignment for real-time control tasks based on plant state information. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 908 –915, sept. 2008.

[Foh93] Gerhard Fohler. *Realizing Changes of Operational Modes with a Pre Run-Time Scheduled Hard Real-Time System*, chapter Dependable computing and fault tolerant system, pages 287–300. Springer Verlag, 1993.

[FU95] N.M. Filatov and H. Unbehausen. Adaptive predictive control policy for nonlinear stochastic systems. *Automatic Control, IEEE Transactions on*, 40(11):1943 –1949, nov 1995.

[FU98] N.M. Filatov and H. Unbehauen. Design of improved adaptive controllers using partial certainty equivalence principle. In *American Control Conference. Proceedings of the 1998*, volume 2, pages 1068 –1072, jun 1998.

[FV92] Simonot-Lion F. and C. Verlinde. Importance d'un cadre de rfrence dans la mise en place d'une dmarche de dveloppement d'un sustme automis de production. *Conference Canadienne sur L'automatisation industrielle*, 1992.

[GBT+04] Carlos Guestrin, Peter Bodik, Romain Thibaux, Mark Paskin, and Samuel Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 1–10, New York, NY, USA, 2004. ACM.

[GGMH07] Jie Gao, Leonidas Guibas, Nikola Milosavljevic, and John Hershberger. Sparse data aggregation in sensor networks. In *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, pages 430–439, New York, NY, USA, 2007. ACM.

[GGP+03] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 89–102, New York, NY, USA, 2003. ACM.

[GHBVdW12] T. M P Gommans, W. P M H Heemels, N.W. Bauer, and N. Van de Wouw. Compensation-based control for lossy communication networks. In *American Control Conference (ACC), 2012*, pages 2854–2859, 2012.

[GJ10] Xiaohua Ge and X. Jiang. A new robust stability criterion of networked control systems. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 2855–2860, 2010.

[Gmb] Bernecker & Rainer Industrie-Elektronik GmbH. Powerlink developers page. http://www.br-automation.com.

[GNDC05] Himanshu Gupta, Vishnu Navda, Samir R. Das, and Vishal Chowdhary. Efficient gathering of correlated data in sensor networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '05, pages 402–413, New York, NY, USA, 2005. ACM.

[GNVL79] G. Golub, S. Nash, and C. Van Loan. A hessenberg-schur method for the problem ax + xb = c. *Automatic Control, IEEE Transactions on*, 24(6):909–913, dec 1979.

[Groa] Ethercat Technology Group. Ethercat protocol. Web page. http://www.ethercat.org/en/technology.html.

[Grob] Ethernet POWERLINK Standardization Group. Powerlink strandard. http://www.ethernet-powerlink.org.

[GRSK11] P. Goyal, V.J. Ribeiro, H. Saran, and A. Kumar. Strap-down pedestrian dead-reckoning system. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1–7, 2011.

[GSC08] E. Garone, B. Sinopoli, and A. Casavola. Lqg control over lossy tcp-like networks with probabilistic packet acknowledgements. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 2686–2691, 2008.

[GSGC12] E. Garone, B. Sinopoli, A. Goldsmith, and A. Casavola. Lqg control for mimo systems over multiple erasure channels with perfect acknowledgment. *Automatic Control, IEEE Transactions on*, 57(2):450–456, 2012.

[GSHM05] Vijay Gupta, Demetri Spanos, Babak Hassibi, and Richard M Murray. Optimal lqg control across packet-dropping links. In *Systems and Control Letters*, pages 360–365. Accepted, 2005.

[HA88] Yoram Halevi and Ray Asok. Integrated communication and control system: Part i—analysis. *Journal of Dynamic Systems, Measurement, and Control*, 110:367–373, Dec 1988.

[Haz34] H.L. Hazen. Theory of servomechanisms. *J. Franklin Inst*, 218:283–331, 1934.

[HBG10] A. Hakiri, P. Berthou, and T. Gayraud. Qos-enabled anfis dead reckoning algorithm for distributed interactive simulation. In *Distributed Simulation and Real Time Applications (DS-RT), 2010 IEEE/ACM 14th International Symposium on*, pages 33–42, 2010.

[HC05] D. Henriksson and A. Cervin. Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 4469–4474, 2005.

[HCB00] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, volume 8 of *HICSS '00*, pages 8020–, Washington, DC, USA, 2000. IEEE Computer Society.

[HGT99] J. Hildebrandt, F. Golatowski, and D. Timmermann. Scheduling coprocessor for enhanced least-laxity-first scheduling in hard real-time systems. In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*, volume 11, pages 208 –215, 1999.

[HHLH10] R.C. Harvey, A. Hamza, Cong Ly, and M. Hefeeda. Energy-efficient gaming on mobile devices using dead reckoning-based power management. In *Network and Systems Support for Games (NetGames), 2010 9th Annual Workshop on*, pages 1–6, 2010.

[HLC12] M. Heidarinejad, Jinfeng Liu, and P.D. Christofides. Distributed model predictive control of switched nonlinear systems. In *American Control Conference (ACC), 2012*, pages 3198–3203, 2012.

[HNX07] João P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A survey of recent results in networked control systems. In *Proceedings of the IEEE*, pages 138–162, 2007.

[HOV02] Joo Hespanha, Antonio Ortega, and Lavanya Vasudevan. Towards the control of linear systems with minimum bit-rate. In *In Proc. of the Int. Symp. on the Mathematical Theory of Networks and Syst*, 2002.

[HR95] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m, k)–firm deadlines. *Computers, IEEE Transactions on*, 44(12):1443 –1451, dec 1995.

[HRS07] Christiaan Heij, Andr Ran, and Freek van Schagen. *Introduction to Mathematical Systems Theory*. Birkhäuser Basel, 2007.

[HS05] Lars Peter Hansen and Thomas J. Sargent. Certainty equivalence and model uncertainty. *Proceedings, Board of Governors of the Federal Reserve System (U.S.)*, pages 17–38, 2005.

[HSJ08] E. Henriksson, H. Sandberg, and K.H. Johansson. Predictive compensation for communication outages in networked control systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 2063–2068, 2008.

[HV01] D. Hristu-Varsakelis. Feedback control systems as users of a shared network: communication sequences that guarantee stability. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 4, pages 3631–3636, 2001.

[HY07] Shawn Hu and Wei-Yong Yan. Stability robustness of networked control systems with respect to packet loss. *Automatica*, 43(7):1243–1248, 2007.

[HY08] Shawn Hu and Wei-Yong Yan. Stability of networked control systems under a multiple-packet transmission policy. *Automatic Control, IEEE Transactions on*, 53(7):1706 –1711, aug. 2008.

[Int87] Interbusclub, www.interbusclub.com/en/doku/pdf/interbusbasicsen.pdf. *Interbus Protocol*, 1987.

[IYB06] Orhan C. Imer, Serdar Yüksel, and Tamer Başar. Optimal control of lti systems over unreliable communication links. *Automatica*, 42:1429–1439, September 2006.

[JJ07] Sirkka-Liisa Jämsä-Jounela. Future trends in process automation. *Annual Reviews in Control*, 31(2):211 – 220, 2007.

[JP86] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.

[JP04] A. Jindal and K. Psounis. Modeling spatially-correlated sensor network data. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 162 – 171, oct. 2004.

[JW08] Sherif Abdelwahed Jian Wu. An efficient finite-input receding horizon control method and its application for the pneumatic hopping robot. *JOURNAL OF COMPUTERS*, 3(9):50–57, September 2008.

[JYQ06] Li Jian and Song Ye-Qiong. Relaxed (m, k)-firm constraint to improve real-time streams admission rate under non pre-emptive fixed priority scheduling. In *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*, pages 1051 –1060, sept. 2006.

[KAGS05] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered ethernet (tte) design. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 22–33, May 2005.

[KBAF11] Markus J. Koegel, Rainer Blind, Frank Allgower, and Rolf Findeisen. Optimal and optimal-linear control over lossy, distributed networks. *Proceedings of the 18th IFAC World Congress*, 18:13239–13244, 2011.

[KC11] Yeonhwa Kong and Hyeonjoong Cho. Guaranteed scheduling for (m,k)-firm deadline-constrained real-time tasks on multiprocessors. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on*, pages 18–23, 2011.

[KDG03] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482 – 491, oct. 2003.

[KDN02] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi. Maximum lifetime data gathering and aggregation in wireless sensor networks. Proceedings of IEEE Networks, 2002.

[KEW02] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCSW '02, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[KF11] Markus Kogel and R. Findeisen. Robust suboptimal control over lossy networks using extended input schemes. In *Computer-Aided Control System Design (CACSD), 2011 IEEE International Symposium on*, pages 210–215, 2011.

[KG94] Hermann Kopetz and Günter Grünsteidl. Ttp-a protocol for fault-tolerant real-time systems. *Computer*, 27(1):14–23, 1994.

[Kim10] Ki-Il Kim. A novel scheduling for (m, k)-firm streams in wireless sensor networks. In *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, pages 553 –556, aug. 2010.

[Kired] Donald E. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering, 2004 (republished).

[KJA97] Bjorn Wittenmark Karl J. Astrom. *Computer Controlled Systems: Theory and Design*. Prentice Hall, 3 edition, 1997.

[KKH04] Kyong Hoon Kim, Jong Kim, and Sung Je Hong. Best-effort scheduling (m, k)-firm real-time tasks based on the (m, k)-firm constraint meeting probability. In *ESA/VLSI'04*, pages 240–248, 2004.

[KKH+08] Xenofon Koutsoukos, Nicholas Kottenstette, Joe Hall, Panos Antsaklis, and Janos Sztipanovits. Passivity-based control design of cyberphysical systems. In *In International Workshop on CyberPhysical Systems - Challenges and Applications (CPSCA08*, 2008.

[KM11] M.R. Kandroodi and B. Moshiri. Identification and model predictive control of continuous stirred tank reactor based on artificial neural networks. In *Control, Instrumentation and Automation (ICCIA), 2011 2nd International Conference on*, pages 338–343, 2011.

[Kop91] H. Kopetz. *Operating Systems of the 90s and Beyond*, volume 563 of *Lecture Notes in Computer Science (LNCS)*, chapter Event-Triggered versus Time-Triggered Real-Time Systems, pages 86–101. Springer Berlin / Heidelberg, 1991.

[Kop97] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*, volume 395 of *The Springer International Series in Engineering and Computer Science*. Springer, April 1997.

[Kr99] Andrzej Krlikowski. Adaptive generalized predictive control subject to input constraints. In *Proceedings of the 7th Mediterranean Conference on Control and Automation (MED99) Haifa, Israel*, volume 7. IEEE, IEEE, june 1999.

[KS95] G. Koren and D. Shasha. Skip-over: algorithms and complexity for overloaded systems that allow skips. In *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, pages 110–117, 1995.

[LB02] B. Lincoln and B. Bernhardsson. Lqr optimization of linear system switching. *Automatic Control, IEEE Transactions on*, 47(10):1701–1705, 2002.

[LC06] J. Lin and A.M.K. Cheng. Maximizing guaranteed qos in (m, k)-firm real-time systems. In *Embedded and Real-Time Computing Systems and Applications, 2006. Proceedings. 12th IEEE International Conference on*, pages 402 –410, 0-0 2006.

[Leh90] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Real-Time Systems Symposium, Proceedings.*, volume 11, pages 201–209, Dec 1990.

[Lev96] William Levine, editor. *The Control Handbook*. CRC Press and IEEE Press, 1996.

[LG96] Marco Spuri Laurent George, Nicolas Rivierre. Preempive and non-preemptive real-time uni-processor scheduling. Research report 2772, Institut National de Recherche, Sep 1996.

[Lin07] Mikael Lindberg. *A Survey of Reservation-Based Scheduling*. Department of Automatic Control, Lund Institute of Technology, October 2007.

[LK12] BiJun Li and Ki-Il Kim. A novel routing protocol for (m, k)-firm-based real-time streams in wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 1715–1719, 2012.

[LKF07] Wen-Hwa Liao, Yucheng Kao, and Chien-Ming Fan. An ant colony algorithm for data aggregation in wireless sensor networks. In *Proceedings of the 2007 International Conference on Sensor Technologies and Applications*, SENSORCOMM '07, pages 101–106, Washington, DC, USA, 2007. IEEE Computer Society.

[LL73] Liu and Layland. Scheduling algorithms for multi-programming in a hard-real time. *Journal of the Association for Computing Machinery*, 20(1), January 1973.

[LL12] Chi-Ying Lin and Yen-Chung Liu. Precision tracking control and constraint handling of mechatronic servo systems using model predictive control. *Mechatronics, IEEE/ASME Transactions on*, 17(4):593–605, 2012.

[LMV10] C. Lozoya, P. Marti, and M. Velasco. Minimizing control cost in resource-constrained control systems: From feedback scheduling to event-driven control. In *Control Automation (MED), 2010 18th Mediterranean Conference on*, pages 267–272, 2010.

[LMVF08a] C. Lozoya, P. Marti, M. Velasco, and J.M. Fuertes. Analysis and design of networked control loops with synchronization at the actuation instants. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, volume 34, pages 2899–2904, November 2008.

[LMVF08b] Camilo Lozoya, Pau Martí, Manel Velasco, and Josep M. Fuertes. Control performance evaluation of selected methods of feedback scheduling of real-time control tasks. In *17th IFAC World Congress*, July 2008.

[LSF+03] Bruno Sinopoli Luca, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, Michael I. Jordan, and Shankar S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49:1453–1464, 2003.

[LSSL06] J. Li, Y. Song, and F. Simonot-Lion. Providing real-time applications with graceful degradation of qos and fault tolerance according to -firm model. *Industrial Informatics, IEEE Transactions on*, 2(2):112–119, may 2006.

[LSST02] Chenyang Lu, John A. Stankovic, Sang H. Son, and Gang Tao. Feedback control real-time scheduling: Framework, modeling, and algorithms*. *Real-Time Syst.*, 23(1/2):85–126, July 2002.

[LW82] J. Y. T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2:237–250, 1982.

[LW12] Bing Li and Junfeng Wu. A new delay-dependent stability criteria for networked control systems. *Journal of Theoretical and Applied Information Technology*, 43(1):127–133, Sep 2012.

[LY08] Li Lanying and Tan Yu. Analysis and improvement of scheduling algorithms based on (m, k)-firm constraint. In *Computer Science and Computational Technology, 2008. ISCSCT '08. International Symposium on*, volume 2, pages 74 –77, dec. 2008.

[LZ10] Tao Li and Ji-Feng Zhang. Decentralized adaptive games for large population coupled arx systems with unknown coupling strength. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3110 –3115, dec 2010.

[LZFA05] Hai Lin, Guisheng Zhai, Lei Fang, and Panos Antsaklis. Stability and h_∞ performance preserving scheduling policy for networked control systems. *Proceedings of the 16th IFAC World Congress*, 16:435–400, 2005.

[MA01] A. Manjeshwar and D.P. Agrawal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 2009 –2015, apr 2001.

[MA02] Arati Manjeshwar and Dharma P. Agrawal. Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, IPDPS '02, pages 48–55, Washington, DC, USA, 2002. IEEE Computer Society.

[MAR08] V. Mikolasek, A. Ademaj, and S. Racek. Segmentation of standard ethernet messages in the time-triggered ethernet. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 392–399, Sept. 2008.

[Mar09] Ricardo Marau. *Real-time communications over switched Ethernet supporting dynamic QoS management*. PhD thesis, Universidade de Aveiro Departamento de Electrnica, Telecomunicaes e Informtica, December 2009.

[MAX68] J.C. MAXWELL. On governors. In *Proceedings of the Royal Society*, number 100, 1868.

[MCH10] Zhongjing Ma, D. Callaway, and I. Hiskens. Decentralized charging control for large populations of plug-in electric vehicles: Application of the nash certainty equivalence principle. In *Control Applications (CCA), 2010 IEEE International Conference on*, pages 191 –195, sept. 2010.

[MFFR02] P. Marti, J.M. Fuertes, G. Fohler, and K. Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, pages 91–100, 2002.

[MFHH02] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36:131–146, December 2002.

[MH09] A. Molin and S. Hirche. On lqg joint optimal scheduling and control under communication constraints. In *Decision and Control, 2009. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 5832 –5838, dec. 2009.

[MLV+08] R. Marau, P. Leite, M. Velasco, P. Marti, L. Almeida, P. Pedreiras, and J.M. Fuertes. Performing flexible control on low-cost microcontrollers using a minimal real-time kernel. *Industrial Informatics, IEEE Transactions on*, 4(2):125 –133, May 2008.

[MNG05] Amit Manjhi, Suman Nath, and Phillip B. Gibbons. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 287–298, New York, NY, USA, 2005. ACM.

[MQF12] R. Muradore, D. Quaglia, and P. Fiorini. Predictive control of networked control systems over differentiated services lossy networks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1245–1250, 2012.

[MSSG12] Yilin Mo, B. Sinopoli, Ling Shi, and E. Garone. Infinite-horizon sensor scheduling for estimation over lossy networks. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 3317–3322, 2012.

[MSZ11] R. Majumdar, I. Saha, and Majid Zamani. Performance-aware scheduler synthesis for control systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 299–308, 2011.

[MVB09] Pau Martí, Manel Velasco, and Enrico Bini. The optimal boundary and regulator design problem for event-driven controllers. In *Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control*, HSCC '09, pages 441–444, 2009.

[MWW12] S.C. McLoone, P.J. Walsh, and T.E. Ward. An enhanced dead reckoning model for physics-aware multiplayer computer games. In *Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on*, pages 111–117, 2012.

[MYH06] S. Motegi, K. Yoshihara, and H. Horiuchi. Dag based in-network aggregation for sensor network monitoring. In *Applications and the Internet, 2006. SAINT 2006. International Symposium on*, pages 8 pp. –299, jan. 2006.

[MZ95] N. Malcom and W. Zhao. Hard real-time communication in multiple-access networks. In Kluwer Academic Publishers, editor, *Real Time Systems*, volume 9, pages 75–107, Boston, USA, 1995.

[NFZE07] G.N. Nair, F. Fagnani, S. Zampieri, and R.J. Evans. Feedback control under data rate constraints: An overview. *Proceedings of the IEEE*, 95(1):108 –137, Jan 2007.

[NGSA04] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 250–262, New York, NY, USA, 2004. ACM.

[NP04] Eduardo Tovar Nuno Pereira, Luis Miguel Pinho. Ethernet-based systems: Contributions to the holistic analysis. *Proceedings of the 16th Euromicro Conference on Real-time Systems (ECRTS'4)*, 4:25–28, july 2004.

[NQ06] Linwei Niu and Gang Quan. Energy minimization for real-time systems with (m,k)-guarantee. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(7):717–729, 2006.

[NS07] Vinh Hao Nguyen and Young Soo Suh. Improving estimation performance in networked control systems applying the send-on-delta transmission method. *Sensors 2007*, 7(10):2128–2138, October 2007.

[NS08] Vinh Hao Nguyen and Young Soo Suh. Networked estimation with an area-triggered transmission method. *Sensors*, 8(2):897–909, 2008.

[NS09] Vinh Hao Nguyen and Young Soo Suh. Networked estimation for event-based sampling systems with packet dropouts. *Sensors*, 9(4):3078–3089, 2009.

[NTZ+10] Trong Duy Nguyen, King-Jet Tseng, Shao Zhang, Shao Zhang, and Hoan Thong Nguyen. Model predictive control of a novel axial flux permanent magnet machine for flywheel energy storage system. In *IPEC, 2010 Conference Proceedings*, pages 519–524, 2010.

[Nyq32] H. Nyquist. Regeneration theory. *Bell Syst. Tech. J.*, 11:126–147, 1932.

[Obe94] R. Obenza. Guaranteeing real-time performance using rma. *Embedded Systems Programming*, pages 26–40, 1994.

[Oga92] Katsuhiko Ogata. *Digital Control Systems* . Saunders College Publishing, 2nd edition, 1992.

[OJW03] Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 563–574, New York, NY, USA, 2003. ACM.

[OK09] Selcuk Okdem and Dervis Karaboga. Routing in wireless sensor networks using an ant colony optimization (aco) router chip. *Sensors*, 9(2):909–921, 2009.

[OLW01] Chris Olston, Boon Thau Loo, and Jennifer Widom. Adaptive precision setting for cached approximate values. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, SIGMOD '01, pages 355–366, New York, NY, USA, 2001. ACM.

[OMT02] P.G. Otanez, J.R. Moyne, and D.M. Tilbury. Using deadbands to reduce communication in networked control systems. In *American Control Conference, 2002. Proceedings of the 2002*, volume 4, pages 3015–3020 vol.4, 2002.

[OW00] Chris Olston and Jennifer Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 144–155, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[OW02] Chris Olston and Jennifer Widom. Best-effort cache synchronization with source cooperation. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD '02, pages 73–84, New York, NY, USA, 2002. ACM.

[PDRPS12] I. Penarrocha, D. Dolz, J. Romero-Perez, and R. Sanchis. Codesign strategy of inferential controllers for wireless sensor networks. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on*, pages 28–33, 2012.

[PGAB05] P. Pedreiras, P. Gai, L. Almeida, and G.C. Buttazzo. Ftt-ethernet: a flexible real-time communication protocol that supports dynamic qos management on ethernet-based systems. *Industrial Informatics, IEEE Transactions on*, 1(3):162–172, Aug. 2005.

[PN97] J. Primbs and V. Nevistic. Constrained finite receding horizon linear quadratic control, technical report cit-cds 97-002,. Technical Report 2, California Institute of technology, Pasadena, CA, 1997.

[PSSV09] Paulo Pedreiras, Stefan Schoenegger, Lucia Seno, and Stefano Vitturi. *Industrial Electronic Handbook: Industrial Communication Systems*, chapter Ethernet POWERLINK. CRC Press, May 2009.

[PVK10] J. Ploennigs, V. Vasyutynskyy, and K. Kabitzsch. Comparative study of energy-efficient sampling approaches for wireless control networks. *Industrial Informatics, IEEE Transactions on*, 6(3):416–424, 2010.

[PY93] E. Polak and T. H. Yang. Moving horizon control of linear systems with input saturation and plant uncertainty - part 1 and 2. *International Journal of Control*, 58(3):613 – 638, September 1993.

[QB96] S. J. Qwin and T.A. Badgwell. An overview of industrial predictive control technology. *Proceedings of the 5th International Conference on Chemical Process Control*, 93(316), 1996.

[QGC13] Li Qiu, Guoxiang Gu, and Wei Chen. Stabilization of networked multi-input systems with channel resource allocation. *Automatic Control, IEEE Transactions on*, 58(3):554–568, 2013.

[QH00] Gang Quan and Xiaobo Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pages 79 –88, 2000.

[Ram97] P. Ramanathan. Graceful degradation in real-time control applications using (m, k)-firm guarantee. In *Fault-Tolerant Computing, 1997. FTCS-27. Digest of Papers., Twenty-Seventh Annual International Symposium on*, pages 132 –141, jun 1997.

[Ram99] P. Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *Parallel and Distributed Systems, IEEE Transactions on*, 10(6):549 –559, jun 1999.

[RK07] C.L. Robinson and P.R. Kumar. Control over networks of unreliable links: Controller location and performance bounds. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007. WiOpt 2007. 5th International Symposium on*, pages 1–8, 2007.

[RKK04] L.A. Rossi, B. Krishnamachari, and C.-C.J. Kuo. Distributed parameter estimation for monitoring diffusion phenomena using physical models. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 460–469, oct. 2004.

[Rob91] Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart. *CAN Specification : Version 2.0*, september 1991.

[RS00] H. Rehbinder and M. Sanfridson. Scheduling of a limited communication channel for optimal control. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 1, pages 1011–1016 vol.1, 2000.

[RSBJ11] C. Ramesh, H. Sandberg, Lei Bao, and K.H. Johansson. On the dual effect in state-based scheduling of networked control systems. *American Control Conference (ACC)*, 29(1):2216–2221, july 2011.

[SB94] Marco Spuri and Giorgio C. Buttazzo. Efficient aperiodic service under earliest deadline scheduling. In *Real-Time Systems Symposium, Proceedings.*, pages 2–11, 1994.

[SBAS04] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 239–249, New York, NY, USA, 2004. ACM.

[SBLC03] Mohamed A. Sharaf, Jonathan Beaver, Alexandros Labrinidis, and Panos K. Chrysanthis. Tina: a scheme for temporal coherency-aware in-network aggregation. In *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, MobiDe '03, pages 69–76, New York, NY, USA, 2003. ACM.

[SBS+02] Shetal Shah, Allister Bernard, Vivek Sharma, Krithi Ramamirtham, and Prashant Shenoy. Maintaining temporal coherency of cooperating dynamic data repositories. In *In Proc. VLDB*, 2002.

[Sch00] B.De Schutter. Minimal state-space realization in linear system theory: an overview. *Journal of Computational and Applied Mathematics*, 121(12):331 – 354, 2000.

[Sch08] L. Schenato. Optimal estimation in networked control systems subject to random delay and packet drop. *Automatic Control, IEEE Transactions on*, 53(5):1311–1317, 2008.

[Sch09] L. Schenato. To zero or to hold control inputs with lossy links? *Automatic Control, IEEE Transactions on*, 54(5):1093–1099, 2009.

[SCH11] Hongyang Sun, Yangjie Cao, and Wen-Jing Hsu. Efficient adaptive scheduling of multiprocessors with stable parallelism feedback. *Parallel and Distributed Systems, IEEE Transactions on*, 22(4):594–607, 2011.

[SCS07] M. Sahebsara, Tongwen Chen, and S.L. Shah. Optimal h2 filtering in networked control systems with multiple packet dropout. *Automatic Control, IEEE Transactions on*, 52(8):1508–1513, 2007.

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379423 & 623656, July 1948.

[SJC+12] Jia Shi, Qingying Jiang, Zikai Cao, Hua Zhou, and Yi Yang. Design method of pid-type model predictive iterative learning control based on the two-dimensional generalized predictive control scheme. In *Control Automation Robotics Vision (ICARCV), 2012 12th International Conference on*, pages 452–457, 2012.

[SJH02] T. Skeie, S. Johannessen, and O. Holmeide. The road to an end-to-end deterministic ethernet. In *Factory Communication Systems, 2002. 4th IEEE International Workshop on*, pages 3 – 9, 2002.

[SJLM11] S. Summers, C.N. Jones, J. Lygeros, and M. Morari. A multiresolution approximation method for fast explicit model predictive control. *Automatic Control, IEEE Transactions on*, 56(11):2530–2541, 2011.

[SKG91] Lui Sha, Mark Klein, and John Goodenough. Rate monotonic analysis. Technical Report CMU/SEI-91-TR-6 ESD-91-TR-6, Carnegie Mellon University, March 1991.

[SKM11] K. Staszek, S. Koryciak, and M. Miskowicz. Performance of send-on-delta sampling schemes with prediction. In *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, pages 2037–2042, 2011.

[SLB08] R. Santos, G. Lipari, and E. Bini. Efficient on-line schedulability test for feedback scheduling of soft real-time tasks under fixed-priority. In *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE*, pages 227–236, 2008.

[SNR07] Young Soo Suh, Vinh Hao Nguyen, and Young Shick Ro. Modified kalman filter for networked monitoring systems employing a send-on-delta method. *Automatica*, 43(2):332–338, February 2007.

[Soc08] IEEE Computer Society. *IEEE 802 Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*. IEEE, 3 Park Avenue, New York, NY 10016-5997, USA, revision 2008 edition, December 2008.

[Spu96] Marco Spuri. Analysis of deadline scheduled real-time systems. Research report 2772, Institut National de Recherche, Jan 1996.

[SR89] John A. Stankovic and K. Ramamritham, editors. *Tutorial: hard real-time systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1989.

[SRL90] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Trans. Comput.*, 39(9):1175–1185, 1990.

[SSF+07] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S.S. Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163 –187, jan. 2007.

[Suh07] Young Soo Suh. Send-on-delta sensor data transmission with a linear predictor. *Sensors*, 7(4):537–547, 2007.

[SZG07] R. Sarkar, Xianjin Zhu, and Jie Gao. Hierarchical spatial gossip for multi-resolution representations in sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 420–429, april 2007.

[TB66] R. Tomovic and G. Bekey. Adaptive sampling based on amplitude sensitivity. *Automatic Control, IEEE Transactions on*, 11(2):282–284, Apr 1966.

[TBS75] E. Tse and Y. Bar-Shalom. Generalized certainty equivalence and dual effect in stochastic control. *Automatic Control, IEEE Transactions on*, 20(6):817 – 819, Dec 1975.

[TBW94] K. W. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Syst.*, 6(2):133–151, 1994.

[Tho98] Jean Pierre Thomesse. A review of the fieldbuses. *Annual Reviews in Control*, 22:35–45, 1998.

[TN89] J.P Thomesse and P. Noury. *ISO TC 184/SC5/WG2-TCCA*, chapter Communication models: Client-server vs Producer-Distributor-Consumer. ISO, October 1989.

[VA06] Mehmet C. Vuran and Ian F. Akyildiz. Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 14:316–329, April 2006.

[Var00] A. Varga. Robust pole assignment techniques via state feedback. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 5, pages 4655–4660, 2000.

[VH09] A. Vakili and B. Hassibi. On the steady-state performance of kalman filtering with intermittent observations for stable systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6847–6852, 2009.

[VK07] V. Vasyutynskyy and K. Kabitzsch. Simple pid control algorithm adapted to deadband sampling. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 932–940, 2007.

[VK10] V. Vasyutynskyy and K. Kabitzsch. A comparative study of pid control algorithms adapted to send-on-delta sampling. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pages 3373–3379, 2010.

[VMB08] M. Velasco, P. Marti, and E. Bini. Control-driven tasks: Modeling and analysis. In *Real-Time Systems Symposium, 2008*, pages 280–290, dec 2008.

[VMF+10] Manel Velasco, Pau Martí, Josep M. Fuertes, Camilo Lozoya, and Scott A. Brandt. Experimental evaluation of slack management in real-time control systems: Coordinated vs. self-triggered approach. *System Architecture*, 56:63–74, January 2010.

[VV06] Klaus Kabitzsch Volodymyr Vasyutynskyy. Implementation of pid controller with send-on-delta sampling. *International Conference Control 2006, Aug. 2006, Glasgow.*, Aug 2006.

[wa] wirelessHART association. http://www.hartcomm.org/protocol/wihart/wireless_technology.html. Online resource.

[WÅÅ02] Björn Wittenmark, Karl Johan Åström, and Karl-Erik Årzén. Computer control: An overview. Technical report, IFAC Professional Brief, January 2002.

[WBK11] H.C. Woithe, D. Boehm, and U. Kremer. Improving slocum glider dead reckoning using a doppler velocity log. In *OCEANS 2011*, pages 1–5, 2011.

[WD07] Ba Wei and Zhang Dabo. A novel least slack first scheduling algorithm optimized by threshold. In *Control Conference, 2007. CCC 2007. Chinese*, pages 264 –268, June 2007.

[WDAA09] Mohamed Watfa, William Daher, and Hisham Al Azar. A sensor network data aggregation technique. *International Journal of Computer Theory and Engineering*, 1(1):19–26, April 2009.

[Wit71] H.S. Witsenhausen. Separation of estimation and control for discrete time systems. *Proceedings of the IEEE*, 59(11):1557 –1566, nov 1971.

[WJ08] Tong Wu and Shiyao Jin. Weakly hard real-time scheduling algorithm for multimedia embedded system on multiprocessor platform. In *Ubi-Media Computing, 2008 First IEEE International Conference on*, pages 320 –325, 31 2008-aug. 1 2008.

[WMG04] Alec Woo, Sam Madden, and Ramesh Govindan. Networking support for query processing in sensor networks. *Commun. ACM*, 47:47–52, June 2004.

[WW11] Dan Wu and Man Hon Wong. Fast and simultaneous data aggregation over multiple regions in wireless sensor networks. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(3):333 –343, may 2011.

[WXTL06] Minji Wu, Jianliang Xu, Xueyan Tang, and Wang-Chien Lee. Monitoring top-k query inwireless sensor networks. *Data Engineering, International Conference on*, pages 143–147, 2006.

[WXTL07] Minji Wu, Jianliang Xu, Xueyan Tang, and Wang-Chien Lee. Top-k monitoring in wireless sensor networks. *IEEE Trans. on Knowl. and Data Eng.*, 19:962–976, July 2007.

[XH04a] Yonggang Xu and J.P. Hespanha. Communication logics for networked control systems. In *American Control Conference, 2004. Proceedings of the 2004*, volume 1, pages 572–577 vol.1, 2004.

[XH04b] Yonggang Xu and J.P. Hespanha. Optimal communication logics in networked control systems. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 3527–3532 Vol.4, 2004.

[XH05] Yonggang Xu and J.P. Hespanha. Estimation under uncontrolled and controlled communications in networked control systems. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 842–847, 2005.

[XLL12] Binqiang Xue, Shaoyuan Li, and Ning Li. Moving horizon state estimation for constrained networked control systems with missing measurements. In *Modelling, Identification Control (ICMIC), 2012 Proceedings of International Conference on*, pages 719–724, 2012.

[XXY+10] Feng Xia, Zhenzhen Xu, Lin Yao, Weifeng Sun, and Mingchu Li. Prediction-based data transmission for energy conservation in wireless body sensors. In *Wireless Internet Conference (WICON), 2010 The 5th Annual ICST*, pages 1–9, 2010.

[YPZ+10] Zheping Yan, Shuping Peng, Jiajia Zhou, Jian Xu, and Heming Jia. Research on an improved dead reckoning for auv navigation. In *Control and Decision Conference (CCDC), 2010 Chinese*, pages 1793–1797, 2010.

[YS07] Sunhee Yoon and Cyrus Shahabi. The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. In *ACM Trans. Sen. Netw.*, volume 3, New York, NY, USA, March 2007. ACM.

[YTS02] J.K. Yook, D.M. Tilbury, and N.R. Soparkar. Trading computation for bandwidth: reducing communication in distributed control systems using state estimators. *Control Systems Technology, IEEE Transactions on*, 10(4):503–518, July 2002.

[YWL+11] Hehua Yan, Jiafu Wan, Di Li, Yuqing Tu, and Ping Zhang. Codesign of networked control systems: A review from different perspectives. In *Cyber*

Technology in Automation, Control, and Intelligent Systems (CYBER), 2011 IEEE International Conference on, pages 84–90, 2011.

[YZL+12] Jinming Yue, Tiefei Zhang, Yannan Liu, Baixin Quan, and Chen Tianzhou. Thermal-aware feedback control scheduling for soft real-time systems. In *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, pages 1479–1486, 2012.

[ZBP01] Wei Zhang, M.S. Branicky, and S.M.; Phillips. Stability of networked control systems. *Control Systems Magazine, IEEE*, 21(1):84 – 99, Feb 2001.

[ZGE03] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 139 – 148, may 2003.

[Zha05] Fuzhen Zhang, editor. *The Schur Complement and Its Applications*. Numerical Methods and Algorithms. Springer US, 2005.

[ZLQ13] Y. Zheng, S. Li, and H. Qiu. Networked coordination-based distributed model predictive control for large-scale system. *Control Systems Technology, IEEE Transactions on*, 21(3):991–998, 2013.

[ZM95] A. Zheng and M. Morari. Stability of model predictive control with mixed constraints. *Automatic Control, IEEE Transactions on*, 40(10):1818–1823, Oct 1995.

[ZWxS04] Ji-ming Chen Zhi Wang and You xian Sun. An integrated dbp for streams with (m,k)-firm real-time guarantee. *JOURNAL OF ZHEJIANG UNIVERSITY*, 5:816–826, 2004.