UNIVERSITÀ DEGLI STUDI DI BOLOGNA
FACOLTÀ DI INGEGNERIA

Dottorato di Ricerca in
Automatica e Ricerca Operativa

XX Ciclo
Settore Scientifico-Disciplinare ING-INF/04 AUTOMATICA

Motion Control and Real-Time Systems: an Approach to Trajectory Rebuilding in Non-Deterministic Networks

Ph.D. Thesis

Manuel Spera

Coordinatore
Prof. Claudio Melchiorri

Tutor
Prof. Carlo Rossi

Preface

Industrial automation is a wide research area studying methodologies and technologies which allow the flow control of energies, materials and information in order to realize production processes without, or with minimal, human effort. It is the result of the know-how in different engineering sectors: automation control as methodological basis; digital electronics, telecommunications and computer engineering for data processing and communication; signal electronics for data acquisition; power electronics, electronics and mechanics, or mechatronics, for actuation. Consequently, an automatic system is nowadays a complex system whose control is of fundamental importance.

Motion control is a sub-field of automation, in which the position and/or velocity of machines are controlled using some type of device such as a hydraulic pump, linear actuator, or an electric motor, generally a servo. In motion control the position, velocity, force, pressure, etc., profiles are designed in such a way that the different mechanical parts work as an harmonious whole in which a perfect synchronization must be achieved.

First facilities prevalently based their development on mechanical parts. The synchronization among the different axes was reached by means of mechanical synchronization. There was a unique source, working at constant velocity, for the motion and the motion itself was distributed by means of kinematic chains to the other parts of the system. This kind of approach was surely robust and dependable, but it was lacking in flexibility.

The rapid growth of electronics and computer science, together with the low cost of this kind of solution, led the companies to change their approach and develop new solutions based on the new technologies. But new problems came out adopting the new technologies. The mechanical systems, controlled by computers, became more similar to computer networks, and then to distributed systems, and communications resulted to be the key point to deal with.

The real-time exchange of information in the distributed system that is nowadays an industrial plant plays an important role in order to achieve always better performance, better effectiveness and better safety. The network for connecting field devices such as sensors, actuators, field controllers such as PLCs, regulators, drive controller etc., and man-machine interfaces is commonly called *fieldbus*. There exist many different fieldbus standards since there are many end-user companies working in different sectors and the possible hosts to be connected (the variety of sensors, of actuators, of controllers) are numerous. Usually different standards are incompatible.

The introduction of real-time networks brings in flexibility into the plant design. There is no more need of a single axis as source of motion, but every axes, called slaves, have

their own source of motion and profile to follow. Then, the synchronization is achieved by means of the control software that, by means of one master axis, generates a synchronization function to which the slaves must be synchronized. In this way every alteration in the relative motion between master and slave is translated into a software change.

Since the motion transmission is now task of the communication system, and not more of the kinematic chain, the communication protocol must assure that the desired profiles, and their properties, are correctly transmitted to the axes then reproduced or else the synchronization among the different parts is lost with all the resulting consequences (loss in productivity in the best case, safety in the worst case).

In this thesis, the problem of trajectory reconstruction in the case of an event-triggered communication system is faced.

In the first part, a brief review of the basic concepts is given: computer networks and their characteristics (Chapter 2), real-time systems and the importance of time as global variable in distributed systems (Chapter 3), and the so-called phase-locked loops, used to develop the proposed solution (Chapter 4).

The second part is the core of this work. It is divided into three chapters. Starting from the basic system composed by one master and one slave (Chapter 5) and passing through systems made up by many slaves and one master (Chapter 6) or many masters and one slave (Chapter 7), the problems in the profile reconstruction, and subsequently the synchronization of different profiles in network adopting an event-triggered communication system, have been shown. These networks are characterized by the fact that a common knowledge of the global time is not available. Therefore they are non-deterministic networks. Each topology is analyzed and the proposed solution based on phase-locked loops adopted for the basic master-slave case has been improved to face with the other configurations.

At last, in Appendix A a concise overview of different fieldbuses is presented, while in Appendix B and C the work developed in collaboration during the first two years of the doctorate school, relative to power electronics and dealing with the current control on magnets in synchrotron machines, is shown.

Questo lavoro rappresenta la fine di un capitolo della mia crescita personale e professionale. Esso, per ovvi motivi, racchiude in sè soltanto la parte tecnica di questi tre anni di dottorato, e non può contenere quella che probabilmente è la parte più importante, quella umana. Desidero pertanto ringraziare in queste poche righe coloro che mi hanno dato l'opportunità di affrontare questo impegno e tutti quelli che, in vari modi, mi hanno accompagnato in questo periodo. Innanzitutto devo ringraziare il Prof. Rossi, che prima mi ha suggerito di tentare il concorso d'ammissione e poi mi ha permesso di svolgere il dottorato sotto la sua supervisione. Ringrazio quindi tutta la mia famiglia, gli amici, ed i compagni del LAR, Giovanni, Gianluca, Alessandro, Davide, Alberto, Manuel T., Roberto. Infine, ringrazio colei che mi ha supportato e sopportato in questi anni di Università e Dottorato. Grazie Stefania.

Bologna, *Manuel Spera*
January, 2008

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motion Control and Fieldbuses

In industrial automation, the search for a communication system's common standard is still an open question. Many protocols and standards (IEC 61158 [1][2][3][4][5], IEC 61784 [6][7][8][9], IEC 62026 [10][11][12], EN 50170 [13][14][15], EN 50254, EN 50325 [16][17][18][19]) were designed and developed in the last 20 years [20], owing to the need for a technology identified by a number of different end-user companies in a number of different sectors and owing to the variety of possible hosts (i.e. sensors, actuators and controllers, sometimes provided with on-chip signal conversion, data and signal processing and communication functions) to be connected. Anyway, as for any communication system, their fundamental mission is the information transfer inside the automation system among nodes.

Field area networks, or fieldbuses [21], are, in general, the networks connecting field devices such as sensors and actuators with field controllers (for instance PLCs) as well as man-machine interface. Usually, field area networks have low data rates and a small size of data packets, and typically require real-time capabilities which demands determinism of data transfer, but sometimes they have to handle other kind of traffic, like best effort one. In fact, the network does not only transport process data, but also configurations or parameters data.

For what concern motion control, the most important feature that a real-time communication system must have is the preservation of the following temporal and spatial properties [22]:

- *Absolute temporal consistency*. It refers to the difference in time between the current time and the time at which the information has been acquired. This is the age of information. Most data are no longer useful when they are too old. The applications should be able to decide if the information they handle is too old or not. Consider two state variables a and b. Let $[t, t_a, v_a]$ and $[t, t_b, v_b]$ be their internal representations where t_a and t_b indicate the instants at which the values v_a and v_b of a and b have been acquired. At instant t, v_a is said to be absolutely consistent if and only if

$$t - t_a \leq A_a \tag{1.1}$$

  where A_a is the absolute consistency threshold for a.
  At instant t, v_b is said to be absolutely consistent if and only if

$$t - t_b \leq A_b \tag{1.2}$$

where A_b is the absolute consistency threshold for b.
Absolute temporal consistency requires the knowledge of the temporal relationship between the time of sampling (or the instant of occurrence of the event) and time of use of the sample (or the event). Event ordering is a special case in which the temporal relationship must be known between two or more events.

- *Relative temporal consistency*. It applies when samples from different signals must be correlated in time. It refers to the temporal delay between the sampling instants on each signal. Two samples of two signals are said to be temporally consistent if their sampling instants differ less than a given duration, called the relative consistency threshold. Many applications assume temporal consistency of their input signals. For instance, a robot controller based on the time-triggered approach will read the positions of the joints to calculate the absolute position of the extremity of a robot arm. If the positions are not sampled at the same time the calculation will be wrong and the controller will determine a wrong position for the arm.
  Using the definition of the internal representation of state variables, v_a and v_b are said to be relatively consistent if and only if

$$|t_a - t_b| \leq R \tag{1.3}$$

  where R is the relative consistency threshold.
  Relative temporal consistency requires that input signals at different nodes should be sampled within strict temporal bounds (relative consistency threshold). Sampling at different nodes should hence be synchronized.
- *Spatial consistency*. It applies when the same information is copied at different locations. The replicas are said to be spatially consistent if they correspond to the same sampling instant or more generally are identical. In the robot controller example given above, the position of the arm may be used by different distributed units: one to control the position of the arm, another to prevent collisions with another robot. If, at a given time, the values are different, collision may occur. Optionally, consistency may be guaranteed.

1.2 Topics of this work

This thesis deals with the preservation of the temporal properties of a profile transmitted over a non-deterministic network.

The first part, Chapters 2, 3, 4, presents the fundamental concepts related to computer networks, distributed systems and phase locked loops.

The importance that computer networks have nowadays is discussed and their characteristics are shown from hardware and software viewpoint. For what concerns hardware architecture, a classification based on the topologies and the size of the network is given. For what concerns the software architecture, the standard ISO-OSI is presented in order to describe the desired functionalities of a network. The most important problem of the access to the physical medium is discussed presenting some solutions. Depending on the modalities with which the physical medium is accessed, a first distinction between deterministic and non-deterministic network can be defined.

In Chapter 3 an introduction to the real-time distributed systems is given. Some definitions and concepts are presented related to real-time environment and the system architecture presented. Event-triggered and time-triggered communication systems are described and the fundamental notions concerning a global time basis and its influence on event-ordering and event time-positioning by means of clock synchronization is explained.

Since phase-locked loops are the essential part of the approach proposed in this thesis, Chapter 4 offers a deep analysis of this technique, that is able to efficiently solve design problems like jitter reduction, skew suppression, frequency synthesis or clock recovery.

The second part of this thesis faces with the problem of the temporal consistency property preservation in the typical topologies of a generic non-deterministic network. Nowadays, such kind of network is of interest for the most increasing attention of industries towards networks based on the Ethernet standard [23]. Since the proposed approach is a solution based on a generic non-deterministic network, it can be implemented not only for a specific fieldbus, but also for all those protocols that have a non-deterministic access to the physical medium when periodic transmission are considered, for example on a CAN-bus with periodic data transmission.

Chapter 5 [24] faces the problem in the simplified, and fundamental, case of a system made-up of two nodes: a master node, that transmits the sampled trajectory on a connection oriented transmission channel, and a slave node, that receives data and rebuilds the trajectory. The two nodes are not synchronized and they are subject to drift. Moreover, even if the master generates real-time periodic traffic, data transmission is subject to jitter. Both smaller latencies on the two nodes and bigger latencies are handled by the proposed solution.

The aim is to minimize the jitter and drift effects on the transmitted data in order to let the system preserve the absolute temporal consistency property of information. Usually, its conservation is achieved by marking with a time-stamp the information inside the transmitter node. So, the receiver, somehow synchronized with the transmitter, is able to evaluate the temporal information provided with the data. This is quite simple if the protocol is based on a Time Division Multiple Access (TDMA) scheme, which is a time-triggered protocol with a common timebase. It's more difficult dealing with an event-triggered system when the time-stamping is not available. Sometimes algorithms are implemented also to estimate the average transmission delay as in the Network Time Protocol [25] or in the IEEE 1588 [26].

Since it is not possible to reduce or avoid the jitter during a transmission, a control algorithm able to estimate the aging of the received information is implemented. The algorithm is based on a phase-locking technique which gives rise to the so-called Phase-Locking Loop (PLL). Normally, this kind of solution is implemented in the lower level of a communication system, for example, with reference to the ISO OSI model, in the physical layer to recover heavily deteriorated data. In this work, the phase-locked loop, even if it is based on hardware components like a counter, works at a higher layer of abstraction than the physical one, directly on the received data packets.The PLLs' basic idea is to locally reproduce the clock of the transmitting node by means of a counter measuring the phase of the incoming signal and usually designing the controller as a low-pass filter. Using a phase-locked loop is advantageous because thanks to its phase-locking property it allows to face problems like the presence of the clock's drift and, generally speaking, all the problems concerning the variation between the nominal and the real

value of the adopted clocks. Besides, by means of the control algorithm, another desirable property in communication systems is achieved: data flow control. Thanks to a correct estimate of the data generation timing, buffer overflow and buffer underrun situations can be prevented, avoiding their dreadful consequences on the temporal properties.

Once the aim of the absolute temporal consistency property is achieved, the next step is the preservation of the relative temporal consistency property. This problem is dealt when the master-multislave and multimaster-slave topologies are exploited. In both cases, different signals must be correlated in time. In the first case, the source of the profiles is the same and data are transmitted to the other nodes. The question is how every node can know when to actuate the received values in a synchronous way with the other nodes if they are not able to know the data arrival time to the other nodes and the positioning along the timeline of the received data on the other nodes. A possible answer to this question is exploited in Chapter 6. In the second case, analyzed in Chapter 7, the source of the signals is different, since the profiles are generated on different nodes and transmitted to the same node. In this situation the relation among data arrival time is possible since the same node learns the single arrival time using the same counter, but the data sending time is unknown. Therefore, the receiving node must align the profiles received by different nodes. It will be shown that also in this case the jitter on the transmission plays an important role.

At last, in Appendix A a short history of the fieldbus standard is presented accompanied by a concise overview of different fieldbuses. In particular the modalities of access to the physical medium adopted by different fieldbuses are pointed out in order to show the variety of approaches to the problem.

Part I

Basic Concepts Review

2

Computer Networks

In this chapter a brief introduction to computer networks is given. Firstly, a description of the hardware architecture is presented, showing the characteristics of typical topologies like broadcast networks, as bus and ring networks, and point-to-point networks. Then the software architecture is described. The concepts of software layering, protocols and services are introduced and the ISO-OSI reference model is described. At last, the very important problem of the access to the bus in broadcast networks is shown [27],[28],[29].

2.1 Introduction

The convergence between computer and communication had, and still has, a deep influence about the computer structure. The old model of only one computer satisfying all the computation requirements has been replaced by another model in which the tasks are carried out by a multitude of single interconnected computers. These systems are called *computer networks*.

Two computers are interconnected when they are able to exchange information. The connection can be realized adopting different technologies and the obtained networks can have different sizes and topologies. There is a distinction between a computer network and a distributed system. The main difference is that in a distributed system the ensemble of single computers appears as a single coherent system to the users. Usually it has a single model or paradigm shown to the users, typically realized by a software layer above the operative system called *middleware*. An example of distributed system is the World Wide Web, in which the model is the web page. In a computer network there are no coherence, no model and no software. The users see the single computers and the system does not try to show or act the calculators in a coherent way. If the PCs have different hardware or operative system, this is visible to the users. The software is the key difference between computer networks and distributed systems. The software gives to the system a high degree of cohesion and transparency. Consequently, the difference between a network and a distributed system is in the software rather than in the hardware. Anyway, the two arguments are superimposed. For example, both computer networks and distributed system require file transfers. The difference is in who executes the operation, the system or the user.

At last, computer networks are born fundamentally for one reason: the sharing of resources and information among different users independently by the position of the resources and users.

2.2 Hardware Architecture

One universal way of classifying computer networks is not defined, but usually two parameters on which this classification is based are the transmission *topology* and the *scale*.

2.2.1 Topologies

The topologies are fundamentally two: broadcasting and point-to-point connections.

Broadcast Network

It is a network composed by one communication channel shared by all the computers connected to the network. Generally, local area networks adopt this kind of network. Short messages, usually called packets, are sent by each computer to the others. The destination address is enclosed in the packet. When the host receives the packet, the destination address is checked: if the message is addressed to the host itself, the message will be processed, otherwise it will be discarded. Usually on broadcast networks, the hosts have the possibility to address one message to all the processors using a special code in the address field. This modality is called *broadcasting*. Moreover, some networks allow the addressing of one message to a subset of computer sharing the network. This modality is called *multicast*.

Typical topologies of broadcast networks are:

- *Bus Network*: it is a network architecture in which a set of clients are connected via a shared communication line, called bus (Fig. 2.1-a). At every time instant, the data transmission is allowed only to one host, the master; the other hosts must abstain from the transmission. When two or more hosts want to transmit at the same time on the same bus problems arise. Therefore a bus arbitration mechanism is required in order to solve conflicts for the access to the network.
- *Ring Network*: it is a network topology in which each node connects to exactly two other nodes, forming a circular pathway, a ring (Fig. 2.1-b). Usually, the single bit travels faster than the packet, that is a bit covers the full ring before the end of the packet is transmitted. Also in this case, an arbitration procedure must be adopted in order to regulate the access to the network.

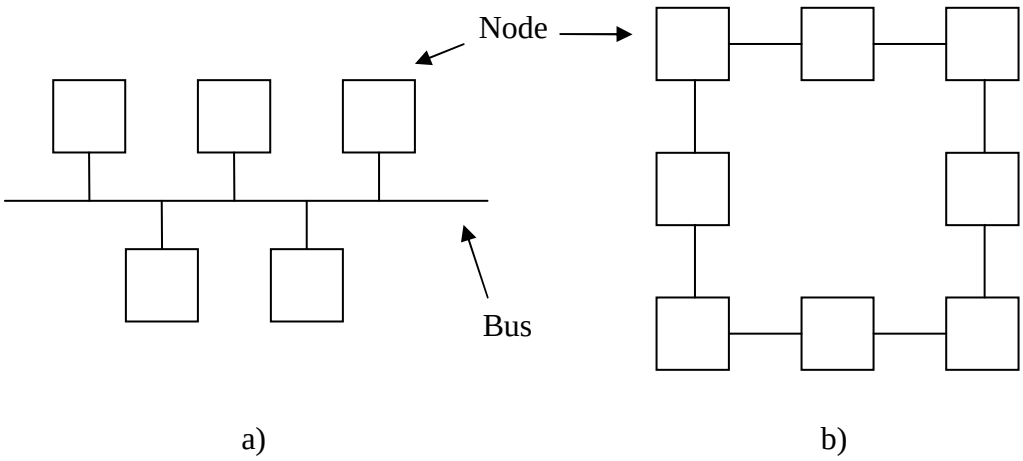

Fig. 2.1. Typical topologies: bus network (a) and ring network (b)

The broadcast network may be further divided into *static* or *dynamic* networks, depending on the way the communication channel is allocated. Again, dynamic networks can be divided into a *centralized* or *distributed* control. This topic will be discussed later in Section 2.5.

Obviously, different broadcast networks can be connected by a broadcast network to improve performance, generating hybrid configurations. In this case, frequently a connecting element is introduced to provide a conversion service among the nets. Such object is called *gateway*.

Point-to-Point Network

On the other hand, in point-to-point networks (Fig. 2.2), generally adopted for wide networks to connect local net, the connections are only between couples of computers. One message generated by a source computer usually passes through intermediate nodes before it arrives to the addressed destination. During the transmission, every node works as a *router*. So, every packet is stored by the intermediate node than forwarded to the next one. Routers and transmission lines compose the *communication subnet*.

The existence of many different ways from the source to the destination is an important topic because usually the best, not necessarily the shortest, way is of interest in order to optimize the net performance. Moreover, since many paths can exist from one point to another point, many packets related to the same message can pass through different ways and be received in a different order from the generation one. So, also the receiving sequence must be handled.

2.2.2 Scale

- *Local Area Network (LAN)*: it is a private network covering a small geographic area, like a home, office, or group of buildings, with typical lengths from a few meters to some Kms. Control systems generally are implemented on LAN.
- *Metropolitan Area Network (MAN)*: it is a large computer network usually spanning a city.
- *Wide Area Network (WAN)*: it is a computer network that covers a broad area. The largest example is the Internet. WANs are used to connect LANs and other types of

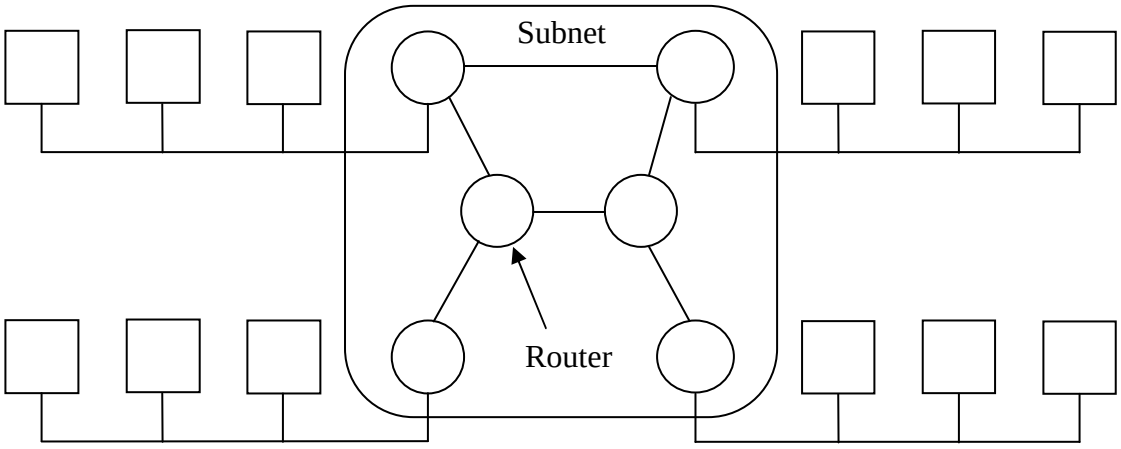

Fig. 2.2. Point-to-point network

networks together. The connections are realized by means of transmission line and switching elements, like routers. So, the adopted type of connection is the point-to-point connection which defines a subnet to link different LANs.

2.3 Software Architecture

2.3.1 Layering

The network software architecture is highly structured. It is developed as a *stack* of *layers*, in which each layer is built over the previous one. Each layer is designed to provide services to the upper layers, hiding them the implementation details of the services.

The n-th layer directly communicates with the n-th layers of the other computers connected to the net by means of a *protocol*, which is a set of communication rules and agreements. The elements forming the same layer on different computers are called *peer*. Every layer can add information to the original message to communicate with the corresponding peer on other nodes. The added information is not transmitted to the upper layers but, obviously, more data are added, less the transmission is efficient.

An *interface* is defined between contiguous layers. It determines the basic operations and services provided by the $(n-1)$-th layer to the upper n-th layer in order to implement the protocol layer. So, the list of protocols adopted by a node is called *protocol stack* (Fig. 2.3).

The combination of layers and protocols defines the *network architecture*.

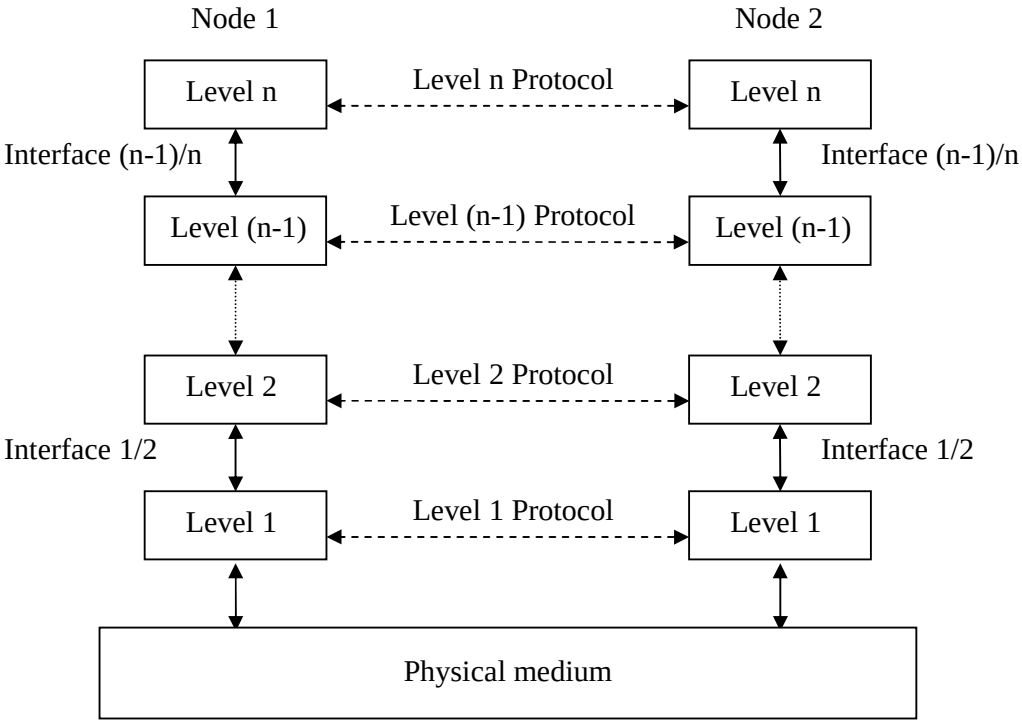

Fig. 2.3. Protocol stack

2.3.2 Services

Every layer provides two different kind of services to the upper layers: connection-oriented or connectionless services.

The *connection-oriented service* works as a dedicated channel: messages inserted at one end directly arrive to the other end. In order to use a connection-oriented channel, the user must establish a connection with the receiver. During this phase, usually transmitter, receiver and subnet negotiate the transmission parameters. When the connection is established, it can be used for the data transmission then released at the end of the operations. By means of the connection-oriented service the data receiving sequence agrees with the data sending sequence.

On the contrary, with the *connection-less service* every message contains the destination address and it is handled by the system independently from the other messages. In this case, the data receiving sequence may not agree with the data sending sequence.

Every service is also classified by means of the provided *quality of service*. A dependable service guarantees the message arrival to the destination usually via an acknowledge mechanism.

At last, every layer develops its own service typology independently from the lower layers.

2.4 ISO-OSI Reference Model

In 1983, revised in 1995, the ISO-OSI (*International Standards Organization - Open System Interconnection*) model has been developed. It was the first step towards an international standardization of the layer protocols, but even if today it is fallen into disuse it is still an excellent model in order to discuss the functionalities and services that each layer should implement.

The ISO-OSI model is constituted by seven layer (Fig. 2.4) that will be presented in the following.

1. *Physical Layer*. The physical layer is always necessary. Essentially, its aim is the bit transmission on the physical medium. So, it defines the mechanical and electrical interfaces and the timing of the network and strongly depends on the physical medium characteristics. Also the bit coding is a physical layer's task.
2. *Data Link Layer*. Aim of the data link layer is to recognize and correct the transmission errors. In order to achieve this task, the sender divides data into data frames sequentially transmitted introducing control bits. Then the receiver checks the frames. Moreover, if the service is reliable, the receiver sends an acknowledgment frame to the sender to confirm the correct reception. Another typical problem faced by the data link layer is the data flow control, that is it must be avoided that a faster sender saturates a slower receiver. At last, the broadcast networks have another problem: how to regulate the access to the common channel. This problem is faced by a special sublayer of the data link layer: the *medium access control* (MAC) layer. The MAC implementation is a critical task in particular in real time applications, because the adopted approach may have considerable reflections on the transmission delays.
3. *Network Layer*. It manages the communication subnet. The packet's routing, the network congestion, the interface among different networks are all problems faced by the network layer.

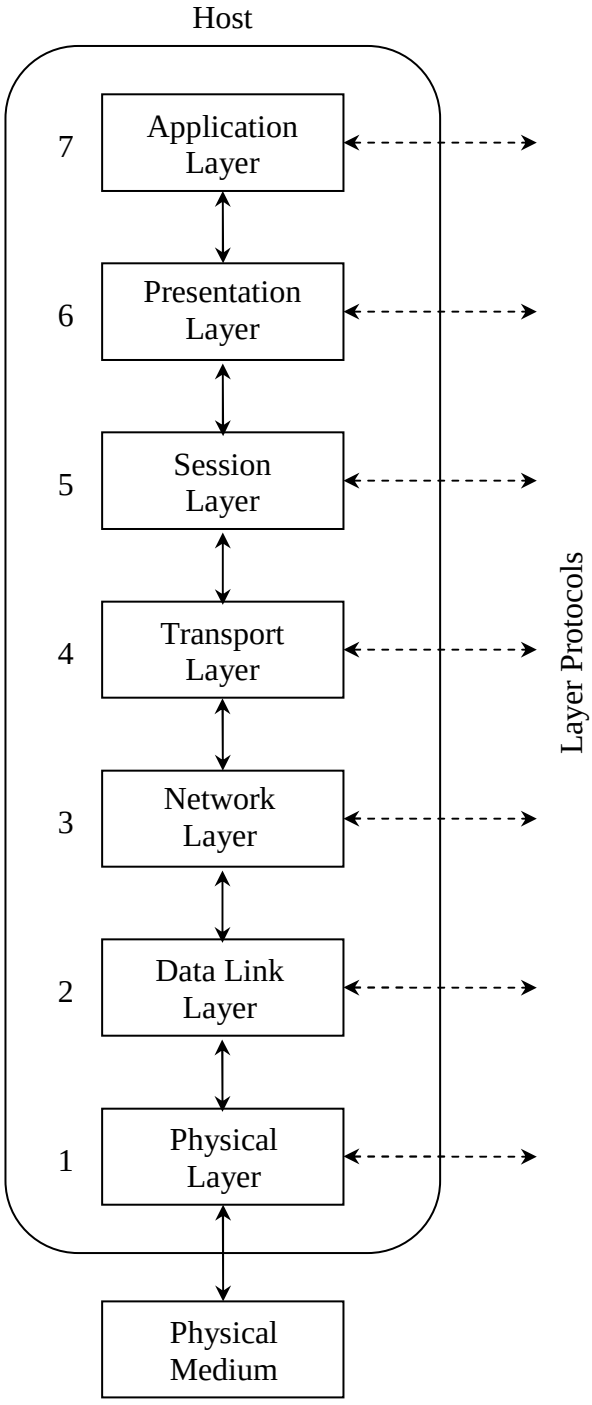

Fig. 2.4. ISO-OSI Reference Model

4. *Transport Layer*. The transport layer is introduced in the OSI model for providing end-to-end control of the exchanges between two end stations, without considering the underlying mechanisms (routing, data link protocol, physical wiring, etc.). To achieve this aim, the transport layer of the sending node cuts the messages into small packets which are transmitted separately from one point to another until they reach the transport layer of the receiving node. Then they are reassembled to reconstitute

the initial message. There is also a mechanism to control the proper reception and possible retransmission.
5. *Session Layer*. It allows to establish a session among different users. Some provided services are: communication tracking, token management and synchronization of long transmissions.
6. *Presentation Layer*. It is responsible for the syntax and semantics of the transmitted data. It allows the data exchange between stations with different internal and local sintaxes, managing the data representation.
7. *Application Layer*. The application layer includes a variety of protocols depending on the applications provided to the users (for example file transfer, e-mail, etc...)

2.5 The MAC Layer

As introduced in the layers description, the MAC implementation is a critical task. It defines the arbitration mechanisms for the access to the physical medium in broadcast networks and therefore it is related to the physical layer and the physical medium.

Depending on the way the communication channel is assigned, we can talk about static or dynamic allocation.

In *static allocation*, the bandwidth is a priori divided and assigned to the users. Obviously if a station has no data to broadcast, the corresponding reserved bandwidth is wasted. Examples of this kind of approach are the *Frequency Division Multiplexing* (FDM) and *Time Division Multiplexing* (TDM) techniques. FDM is a form of signal multiplexing where multiple baseband signals are modulated on different frequency carrier waves and added together to create a composite signal. In TDM, the time is divided into several recurrent timeslots of fixed length, one for each sub-channel. In this case all the nodes must know the transmitting order or they must be synchronized to a global clock. Therefore in TDM it is guaranteed that if one station is transmitting data on the channel the other stations will not transmit their messages but they will wait for their turn. So there will not be any collisions among messages, that is the transmitted messages will not be disrupted.

Dynamic allocation can be divided into *centralized* or *decentralized* control. In the first case there exist one master node that decides which will be the next transmitting node time after time. In the second case the transmission can be still coordinated in some way (usually achieved by passing a token, that is a special data frame, from one station to the next one, like in the IBM token ring/bus) or it can be based on collisions, occurring when two or more stations try to transmit on the channel at nearly the same time instant. In this case some protocols allows to the stations to verify the absence of other traffic before transmitting on a shared physical medium: every node can listen to the line and if the line is free, the data transmission starts, otherwise if the line is busy, the transmission is delayed. These protocols are called pure *Carrier Sense Multiple Access* (CSMA). Moreover, if two ore more stations start the transmission nearly at the same time instant, a collision happens and there are many solutions depending on the implemented protocols. Some protocols only detect the collisions and stop transmitting immediately, backing off for a random amount of time before trying again (*Collision Detection*, CSMA/CD), like Ethernet. Other protocols try ro resolve the collisions (*Collision Resolution*, CSMA/CR), for example assigning to all of the nodes sharing the line an identification number or priority code: when a collision occurs, one of the nodes that are attempting to send at

the same time will be given priority to transmit according to its identification number or priority code.

The adopted approach to the problem of channel access, the characteristics of the physical medium, in particular of the transmission delay that defines the above expressions ”nearly at the same time instant”, and the chosen topology strongly reflects on the type of data frame, the minimum length of messages but especially on the determinism of the message transmission, that is an important topic in real time systems. So, it is clear that an approach based on a TDM technique is more predictable than one based on a CSMA/CD protocol.

3

Real-Time Distributed Systems

In this chapter, the basic definitions and concepts of a real-time system are given. The system architecture is faced describing the hardware architecture. A particular emphasis is given to the Communication-Network Interface and the communication system, event-triggered or time triggered. The important concept of time in a real-time distributed system and the consequent notion of global time and internal clock synchronization are given. At last two algorithms for central master and distributed clock synchronization are briefly described [30],[31].

3.1 Introduction

A *real-time computer system* is a computer system in which the correctness of the system behavior depends not only on the logical results of the computations but also on the physical instant at which these results are produced.

A real-time computer system is always part of a larger system called *real-time system*. The real-time system can be decomposed into a set of subsystems called clusters: the *controlled object*, the real-rime computer system and the *human operator* (Fig. 3.1). The interface between the human operator and the real-time computer system is called the *man-machine interface*, and the interface between the controlled object and the real-time computer system is the *instrumentation interface*.

A real-time computer system interacts with the controlled object and must provide reactions to the controlled object within a specified time instant, called *deadline*. If the produced result has utility even after the deadline has passed, the deadline is called *soft*, otherwise it is *firm*. If a catastrophe could result if a firm deadline is missed, the deadline is *hard*.

The flow of real-time can be modeled by a directed time line that extends from the past into the future. Any occurrence that happens at a cut of this time line is called an *event*. An *interval* on the time line is defined by two events, the *start event* and the *terminating event*. The *duration* of the interval is the time of the terminating event minus the time of the start event. Any property of a real-time entity that remains valid during a finite duration is called a *state attribute*. A change of state is thus an event. An *observation* is an event that records the state of a real-time object at a particular instant, the *point of observation*. A digital clock partitions the time line into a sequence of equally-spaced durations, called the *granules* of the clock which are bounded by special periodic event, the *ticks* of the clock. A *trigger* is an event that causes the start of some action. Depending on

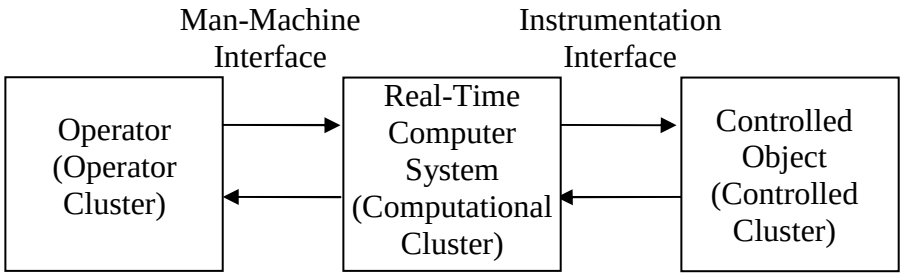

Fig. 3.1. Real-time system

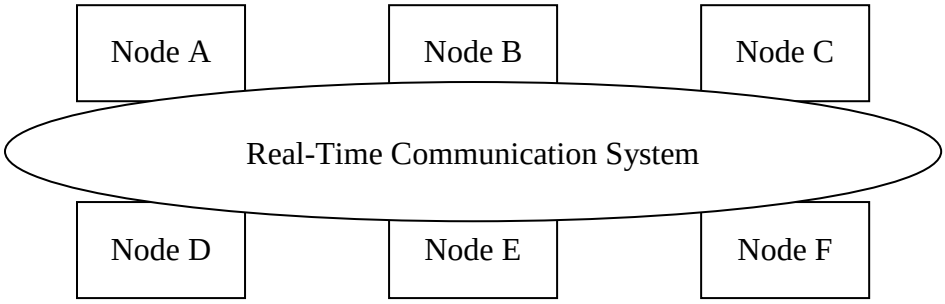

Fig. 3.2. Distributed computer system

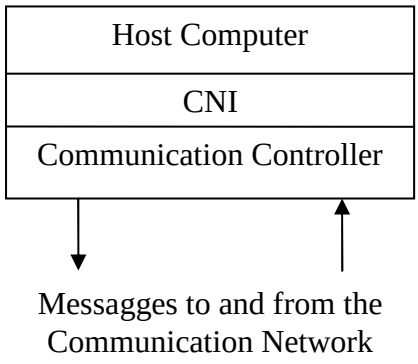

Fig. 3.3. Structure of a node

the triggering mechanisms for the start of the communication and processing activities, two different approaches to the design of real-time computer applications can be defined. In the *event-triggered (ET)* approach, all communication and processing activities are initiated whenever a significant change of state is noted. In the *time-triggered (TT)* approach, all communication and processing activities are initiated at predetermined points in time. In an ET system, the signaling of significant events is realized by the interrupt mechanism. In a TT system, all activities are initiated by the progression of time.

3.2 System Architecture

3.2.1 Hardware Structure

If the real-time computer system is distributed, it consists of a set of nodes (computers) interconnected by a *real-time communication system* (Fig. 3.2). Each node can be considered composed by at least two subsytems: the *local communication controller* and the

host computer (Fig. 3.3). The set of all the communication controllers of the nodes within a cluster, along with the physical interconnection medium, forms the real-time communication system of the cluster. The interface between the communication controller within a node and the host computer of the node is called the *communication-network interface* (CNI). The CNI is located at the transport level of the OSI reference model.

3.2.2 The Communication-Network Interface

The purpose of the real-time communication system is to transport messages from the CNI of the sender to the CNI of the receiver node within a predictable time interval, with a small latency, and with high reliability. From the point of view of the host computer, the details of the protocol logic and the physical structure of the communication network are hidden behind the CNI.

Two types of message processing are distinguished at the CNI of the receiver depending on the information contained in the message: *occurrence of an event* or *value of the state*. In the first case, every event is significant and messages must not be lost. If one message is lost, also the synchronization between sender and receiver could be lost. So, this kind of messages must be queued at the receiver and removed by the queue once they have been read. In fact, also the processing of the same message twice or more could lead to the loss of sender-receiver synchronization. Moreover, the order in the queue should be the temporal order of event occurrence, that not always corresponds to the delivery order. In the second case, if the message contains state information, usually the new information is of interest, so the old one can be overwritten.

Depending on who takes the decision about the sending time instant of the message, two kind of control strategy are distinguished. If the decision is taken by the host computer, the control is *external*. Otherwise, if the decision is taken by the communication system, the control is *autonomous*. In the case of external control, the execution of a ”send” command in the host computer causes the transfer of a control signal across the CNI and initiates the transmission of a message by the communication system. Similarly, at the receiver side, a control signal from the communication system crosses the CNI and unblocks a ”receive” command in the receiving host computer when the message arrives. In the case of autonomous control, no control signals cross the CNI from or to the host computer, so the communication system autonomously decides when to send the next message and when to deliver the message at the CNI of the receiver. Usually autonomous control is time-triggered and the communication system adopts a transmission time-table.

The possible combinations given by the semantics, events and state, and by the adopted control strategy, external or autonomous, both at the sender and the receiver are sixteen. Two of them are of special significance and represent event messages and state messages. *Event messages* combines event semantics with external control both at the sender and the receiver. Every arriving event message is queued at the receiver. Event messages require one-to-one synchronization between the sender and the receiver, otherwise the queue will overflow, or the receiver will be blocked. This mechanism is typical in non real-time systems. *State messages* combine state-value semantics with autonomous control. In this case the one-to-one synchronization between sender and receiver is not required because the receivers can read a state value many times or not at all.

3.2.3 The Communication System

There are a number of different topologies available for the design and implementation of the communication service: point-to-point, bus, ring etc.

The communication system is a critical resource of a distributed system, since the loss of communication results in the loss of all global system services.

3.2.4 Gateways

The purpose of a gateway is to exchange relative views between two interacting clusters. Sometimes, the structure of the messages and the representation of the information is not identical in both the clusters. Thus, the gateway host must transform the data formats of one cluster to those expected by the other cluster.

3.2.5 Event-Triggered Communication System

If a communication system transports event messages, i.e. it is a communication system event-triggered ET, the temporal control is external to the communication system. It is task of the host computer to decide when a message must be sent. If the communication system uses a single shared channel that serializes the traffic, then a conflict for gaining the access is unavoidable. The problem in an ET communication system is that the temporal control at the CNI is not defined by an ET protocol. Temporal control in an ET system is thus a global issue, depending on the behavior of the application software in all nodes of the distributed system. From the point of view of temporal behavior, ET systems are not composable.

3.2.6 Time-Triggered Communication System

In a time-triggered communication system, temporal control resides within the communication system, and is not dependent on the application software in the nodes. State messages are transported from the sender CNI to the receiver CNI at predetermined points in time which are stored in message scheduling tables within the communication controllers. The host computers have no opportunity to influence the temporal behavior of the communication system. The CNI is strictly a data-sharing interface without any control signals crossing the interface. It thus acts as a temporal firewall, isolating the temporal behavior of the host computer from the temporal behavior of the communication system.

3.3 Global Time

In a typical real-time application, the distributed computer system performs a multitude of different functions concurrently, normally executed at different nodes. To guarantee a consistent behavior of the entire distributed system, it must be ensured that all nodes process all events in the same consistent order, preferably in the same temporal order in which the events occurred. A global time base helps to establish such a consistent temporal order on the basis of the timestamps of the events.

It is possible to distinguish three kind of different orders:

- *Temporal Order*. A sequence of events is temporally ordered. Events are only partially ordered, since simultaneous events are not in the order relation. Eventually, events can be totally ordered if another criterion is introduced to order events that occur simultaneously.
- *Causal Order*. In many real-time applications, the causal dependencies among events are of interest. In a chain of events, a cause-event necessarily happens before the corresponding effect-event. The temporal order of two events is necessary, but not sufficient, for their causal order. Causal order is more than temporal order.
- *Delivery Order*. The communication system guarantees that all host computers in the nodes see the sequence of events in the same delivery order. This delivery order is not necessarily related to the temporal order of event occurrences or the causal relationship between events.

3.3.1 Clocks

- *Physical clock*. It is a device for measuring time. It contains a counter and a *physical oscillation mechanism* that periodically generates an event to increase the counter. The periodic event is called the *microtick* of the clock. The duration between two consecutive microticks is the *granularity* of the clock. The granularity of any digital clock leads to a digitalization error in time measurement. In the following, microtick i of clock k is denoted by $microtick_i^k$
- *Reference Clock*. Assume an omniscient external observer. This observer possesses a *unique reference clock z* with frequency f^z. $\frac{1}{f^z}$ will be the granularity g^z of clock z. Whenever the omniscient observer perceives the occurrence of an event e, it will instantaneously record the current state of the reference clock as the time of occurrence of this event e, and, will generate a *timestamp* for e. *Clock(event)* denotes the timestamp generated by the use of a given clock to timestamp an event. *z(e)* is called the *absolute timestamp* of the event e. The temporal order of events that occur between any two consecutive microticks of the reference clock, i.e. within the granularity g^z, cannot be reestablished from their absolute timestamps.
- *Clock Drift*. The drift (Fig. 3.4) of a physical clock k between microtick i and microtick $i+1$ is the frequency ratio between this clock k and the reference clock, at the instant of microtick i. The drift is determined by measuring the duration of a granule of clock k with the reference clock z and dividing it by the nominal number n^k of reference clock microticks in a granule:

$$drift_i^k = \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{n^k} \tag{3.1}$$

Because a good clock has a drift that is very close to 1, the notion of a *drift rate* ρ_i^k is introduced as

$$\rho_i^k = \left| \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{n^k} - 1 \right| \tag{3.2}$$

A perfect clock will have a drift rate of 0. Real clocks have a varying drift rate that is influenced by environmental conditions. Within specified environmental parameters, the drift rate of a resonator is bounded by the maximum drift rate, which is documented in the data sheet. Typical maximum drift rates are in the range of 10^{-2} to 10^{-7} sec/sec.

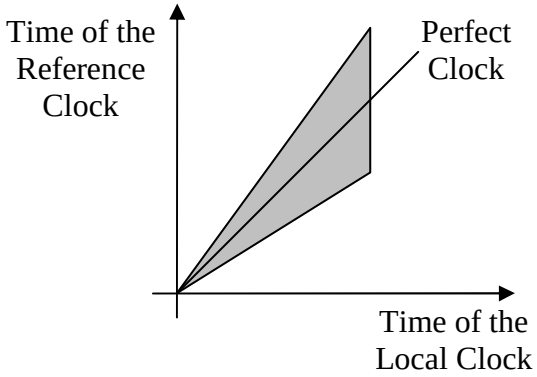

Fig. 3.4. Clock Drift: a good clock with a bounded drift rate stays in the shaded area

- *Offset*. The offset at microtick i between two clocks j and k with the same granularity is defined as
$$offset_i^{jk} = |z(microtick_i^j) - z(microtick_i^k)| \tag{3.3}$$
The offset denotes the time difference between the respective microticks of the two clocks, measured in the number of microticks of the reference clock.
- *Precision*. Given an ensemble of clocks 1,2,...,n, the maximum offset between any two clocks of the ensemble
$$\Pi_i = \max_{\forall 1 \le j,k \le n} offset_i^{jk} \tag{3.4}$$
is called the precision Π_i of the ensemble at microtick i. The maximum of Π_i over an interval of interest is called the precision Π of the ensemble. Because of the drift rate of any physical clock, the clocks of an ensemble will drift apart if they are not resynchronized periodically. The process of mutual resynchronization of an ensemble of clocks to maintain a bounded precision is called *internal synchronization*.
- *Accuracy*. The offset of clock k with respect to the reference clock z at microtick i is called the $accuracy_i^k$. The maximum offset over all microticks i that are of interest is called the $accuracy^k$ of clock k. The accuracy denotes the maximum offset of a given clock from the external time reference during the time interval of interest. To keep a clock within a bounded interval of the reference clock, it must be periodically resynchronized with the reference clock. This process of resynchronization is called *external synchronization*.

3.3.2 Global Time

Suppose a set of nodes exists, each one with its own local physical clock k that ticks with granularity g^k. Assume that all of the clocks are internally synchronized with a precision Π, i.e. for any two clocks j, k and all microticks i

$$|z(microtick_i^j) - z(microtick_i^k)| < \Pi. \tag{3.5}$$

It is then possible to select a subset of the microticks of each local clock k for the generation of the local implementation of a global notion of time. We call such a selected local microtick i a *macrotick* of the global time. A global time is thus an abstract notion that is approximated by properly selected microticks from the synchronized local physical clocks.

The global time t is called *reasonable* if all local implementations of the global time satisfy the condition

$$g > \Pi. \tag{3.6}$$

This reasonable condition ensures that the synchronization error is bounded to less than one macrogranule.

3.3.3 Internal Clock Synchronization

The purpose of internal clock synchronization is to ensure that the global ticks of all correct nodes occur within the specified precision Π, despite the varying drift rate of the local real-time clock of each node. Every node of a distributed system has a local oscillator. A subset of the local oscillator's microticks are interpreted as the global time ticks at the node. These global time ticks increment the node's local global time counter. Depending on the way the synchronization is achieved, it is possible to distinguish between *central master synchronization* or *distributed synchronization*.

- *Central Master Synchronization*. A unique node, the central master, periodically sends the value of its time counter in synchronization messages to all other nodes, the slave nodes. As soon as a slave node receives a new time value from the master, the slave records the state of its local-time counter as the time of message arrival. The difference between the master's time, contained in the synchronization message, and the recorded slave's time of message arrival, corrected by the latency of the message transport, is a measure of the deviation of the two clocks. The slave then corrects its clock by this deviation to bring into agreement with the master's clock. The latency, called *latency jitter*, is the difference between the fastest and the slowest message transmission to the slave nodes of the ensemble.
- *Distributed Synchronization Algorithm*. Distributed synchronization typically proceeds in three distinct phases. In the first phase every node acquires knowledge about the state of the global time counters in all the other nodes by exchange of messages among the nodes. In the second phase, every node analyzes the collected information to detect errors, and executes a function to calculate a correction value for the local global time counter. Finally, in the third phase, the local time counter of the node is adjusted by the calculated correction value.

At last, depending on the way the correction term is applied, it is possible to talk about *state correction*, if the value is applied to the local-time value immediately, or *rate correction*, if the rate of the clock can be modified so that the clock speeds up or slows down during the next resynchronization interval to bring the clock into better agreement with the rest of the ensemble. State correction is simple to apply but has the disadvantage of generating a discontinuity in the time base. It is therefore advisable to implement rate correction with a bound on the maximum value of the clock drift so that the error in interval measurements is limited. The resulting global time base then maintains the chronoscopy property despite the resynchronization.

4

Phase-Locked Loops

In this chapter the phase locking technique is presented. It is a powerful technique able to solve problems like jitter reduction, skew suppression, frequency synthesis and clock recovery. The classic approach is introduced and the basic topology, with the corresponding linear model and tracking behavior, presented. At last, an analysis of the phase-locking loops evolution from continuous-time to discrete-time and advanced topologies are discussed.

4.1 Introduction

Phase-locking is a powerful technique that can provide elegant solutions in many applications and that has done its first appearance in a paper in 1922 by Appleton [32]. It is used in many areas like communications, wireless systems, digital circuits and disk drive electronics because it is able to efficiently solve design problems like:

- *Jitter reduction*: when a signal travels through a communication channel, it experiences a variable delay in the transmission that it is seen by the receiver as a variation of the period of the waveform (Fig. 4.1).

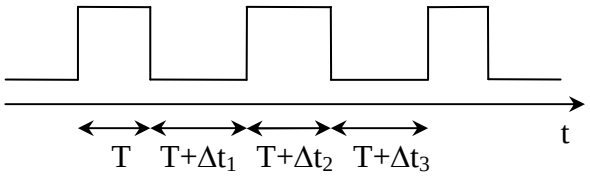

Fig. 4.1. Jitter Reduction

- *Skew suppression*: having two clocks, CK_m and CK_s, with the same period, it is required to align them reducing the phase difference Δt (Fig. 4.2).
- *Frequency synthesis*: many applications require frequency multiplication of periodic input signals.
- *Clock recovery*: in many systems, data is transmitted without timing reference so the timing information must be recovered from the data at the receiving end.

In the next sections, a brief review of the concepts useful for approaching to PLLs will be given [33].

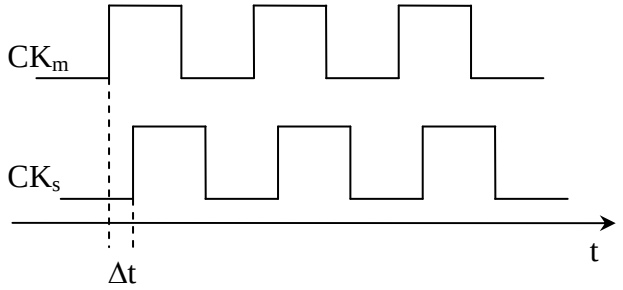

Fig. 4.2. Skew suppression

4.2 Time- and frequency-domain characteristics

The PLL's behavior is mostly nonlinear, but it can be considered linear in steady state and during slow transient. So, it is necessary a characterization of the interesting signals. Assuming to be in analog domain, the typical signals encountered in PLLs are either strictly periodic, as $x(t) = Acos(\omega_c t)$, or phase-modulated, for example,

$$x(t) = Acos[\omega_c t + \varphi_n(t)] \tag{4.1}$$

The second case is the most general one and it will be considered during the rest of the treatment. The total phase and the total frequency of this signal are defined as

$$\varphi_c(t) = \omega_c t + \varphi_n(t) \tag{4.2}$$

$$\Omega_c(t) = \frac{d\varphi_c(t)}{dt} = \omega_c + \frac{d\varphi_n(t)}{dt}. \tag{4.3}$$

PLLs usually operate on the excess components of φ_c and Ω_c, that is $\varphi_n(t)$ and $\frac{d\varphi_n(t)}{dt}$ respectively. The presence of these excess components leads to a slight deviation from a strictly periodic behavior of the signal because there will be a little difference between consecutive periods of the waveform. This kind of signal will be called *almost periodic*. Now, it is possible to define two important parameters:

- *cycle-to-cycle jitter*: difference between every two consecutive periods of an almost periodic waveform;
- *absolute jitter*: phase difference between the same waveform and a periodic signal having the same average frequency.

In the frequency domain, counterparts of the jitter are *sidebands* and *phase noise*.

The first ones are deterministic components that do not have an harmonic relationship with the main component ω_c , the carrier, and that are usually specified with their frequency and magnitude relative to that of the carrier (for example, supposing $\varphi_n(t) = \varphi_m sin(\omega_m t)$, $|\varphi_m| \ll 1$, there will be two sidebands located respectively at $\omega = \omega_c + \omega_m$ and $\omega = \omega_c - \omega_m$).

In contrast to sidebands, phase noise arises from random frequency components.

Anyway, if uniformity of zero crossing is critical, both of them are undesirable.

4.3 Signal generator: voltage-controlled oscillator

In classic literature, the sinusoidal signal is generated by means of a *voltage-controlled oscillator* (VCO). The frequency produced is a linear function of a control voltage V_{cont}:

$$\omega_{out} = \omega_{FR} + K_{VCO} V_{cont} \tag{4.4}$$

where ω_{FR} is the *free-running frequency* and K_{VCO} is the gain of the VCO. Since the phase is the time integral of frequency, the output of a sinusoidal VCO can be expressed as

$$y(t) = Asin(\omega_{FR} t + K_{VCO} \int_{-\infty}^{t} V_{cont} dt). \tag{4.5}$$

In practical VCOs, K_{VCO} exhibits some dependence on the control voltage and eventually drops to zero as $|V_{cont}|$ increases.

Considering the VCO as a linear time-invariant system, with the control voltage as the system's input and the excess phase of the output signal as the system's output, it is possible to write down:

$$\varphi_{out}(t) = K_{VCO} \int_{-\infty}^{t} V_{cont} dt \tag{4.6}$$

that corresponds to the input/output transfer function:

$$\frac{\varphi_{out}(s)}{V_{cont}(s)} = \frac{K_{VCO}}{s}. \tag{4.7}$$

It is important to notice that if no other input is available to set the VCO's phase, we must first change the frequency and let the integration take place to obtain the desired output phase. It means the output phase of a VCO cannot be determined only from the present value of the control voltage, but it depends on its history.

4.4 PLL's basic topology

A phase-locked loop is a feedback system operating on the excess phase of nominally periodic signals. It consists of a phase-detector, a regulator (usually a low-pass filter) and a VCO (Fig. 4.3).

The phase-detector detects the phase error, $\Delta\varphi$, that has to be minimized between the input and output signals. This value is passed to the low-pass filter, which generates the control action for the VCO.

The loop is considered *locked* when $\Delta\varphi$ is constant in time, meaning the input and output frequencies are equal. When the locked condition is reached, all the signals are in

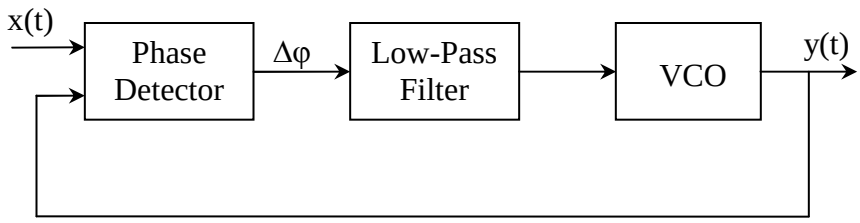

Fig. 4.3. PLL's basic topology.

steady state. Anyway, if the frequencies are equal but $\Delta\varphi$ is not the desired one, the loop must continue the transient, temporarily making the frequencies unequal again. In other words, both *frequency acquisition* and *phase acquisition* must be completed.

4.4.1 Linear model in locked-state

Transient response of phase-locked loops is generally a nonlinear process that cannot be formulated easily. Nevertheless, as with other feedback systems, a linear approximation can be obtained. The linear model is presented in Fig. 4.4 and it can be considered valid if the following hypotheses hold:

1. the loop is frequency locked;
2. the time-domain is continuous;
3. the VCO is an ideal integrator;
4. the phase-detector is a linear adder.

Its open-loop transfer function is therefore equal to

$$H_O(s) = K_{PD}R(s)\frac{K_{VCO}}{s} \tag{4.8}$$

yielding the following closed-loop transfer function:

$$H(s) = \frac{\varphi_{out}(s)}{\varphi_{in}(s)} = \frac{K_{PD}K_{VCO}R(s)}{s + K_{PD}K_{VCO}R(s)} \tag{4.9}$$

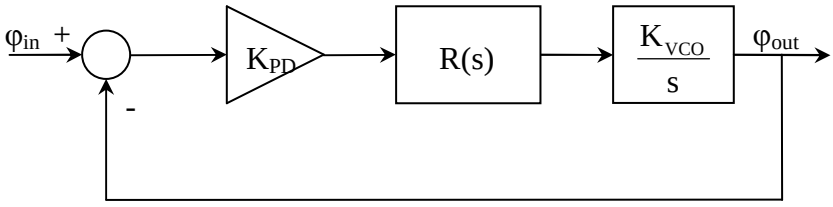

Fig. 4.4. Linear model in locked state.

4.5 Tracking behavior

A PLL is able to track the input frequency minimizing the phase error. Now, assume the loop is locked. To analyze the features of the tracking behavior it is possible to consider two extreme cases:

1. the input frequency varies slowly (*static tracking*);
2. the input frequency is changed abruptly (*dynamic tracking*). In this case it is possible to talk about *acquisition of lock*.

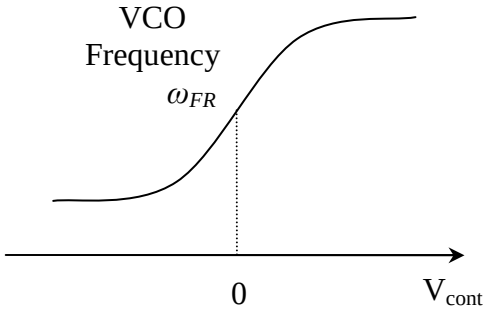

Fig. 4.5. VCO characteristic.

4.5.1 Static Tracking

Suppose, starting from the VCO free-running frequency, the input frequency varies slowly such that the difference between ω_{in} and ω_{out} always remains much less than the cut-off frequency of the regulator. The PLL will track the incoming signal until it will remain bounded in the *tracking range*. This one mainly depends on the boundaries set by the implementation of the phase detector and the VCO. For example, in the analog domain, the VCO frequency typically has a limited range, out of which its gain drops sharply (Fig. 4.5). Also, in a typical phase detector, the characteristic becomes non-monotonic for a sufficiently large input phase difference, at which point the PLL fails to maintain lock.

4.5.2 Dynamic Tracking

This is the case of two similar situations:

1. a loop initially locked at ω_{FR} experiences a large input frequency step $\Delta\omega$;
2. a loop initially unlocked and free-running (ω_{out}=ω_{FR}) must lock onto an input frequency given by $|\omega_{in} - \omega_{FR}| = \Delta\omega$.

In both cases, the loop must acquire lock.

As in the static tracking, it is possible to define the tracking range, called *acquisition range* or *capture range*, as the maximum value of $\Delta\omega$ for which the loop locks. This value essentially depends on the loop gain at $\Delta\omega$, that usually drops as $\Delta\omega$ increases. In fact, the regulator, which is fundamentally a low-pass filter, has to pass the $\Delta\omega$ component and its output must have a continuous component strong enough to drive the VCO to the demanded frequency.

Another important feature about the dynamic tracking is the *acquisition time*. This parameter, together with the *settling time*, can be fundamental in some applications. Anyway, it is difficult to analytically calculate them owing to the nonlinearities of the system even if it is possible to obtain simplified formulas assuming a linear system.

4.6 Regulator

In its simplest form, the controller is a low pass filter implemented as:

$$R_{LPF}(s) = \frac{1}{1 + \frac{s}{\omega_{LPF}}}. \tag{4.10}$$

In this way, the closed loop transfer function, also called *jitter transfer function*, is a classical second order system, with one pole contributed by the VCO and another one by the LPF. The definition of the parameters is usually a trade-off between the desired frequency response and the boundaries imposed by sidebands or noise suppression. So, it's not possible choosing ω_{LPF} and the loop gain independently.

Moreover, such a regulator is able to drive the phase error to zero if the input phase has a step variation, thanks to the presence of integration in the VCO. Instead, this is not true if the step variation concerns the frequency. The phase error will be bounded, but not zero, due to the linear relation between phase and frequency (the phase is the integral of the frequency, so the input of the closed loop will have a ramp variation).

The most used regulator is a PI controller. It allows obtaining a phase error equal to zero even in presence of a frequency step variation and, thanks to the introduction of the zero in the closed loop transfer function, relaxing the trade off between ω_{LPF} and the loop gain making them independent. However, there's a drawback: the magnitude of the jitter transfer function of such a system exceeds unity over some range of frequencies because of the presence of the closed-loop zero at a frequency lower than that of the poles. This gain in excess of unity is known as *jitter peaking* and it could be a problem if there's jitter/noise in that range of frequencies because it will be copied and amplified on the output signal. The traditional solution is to reduce the spacing between the zero and the lowest frequency pole, i.e. increasing the damping ratio, but it often has side-effects on the frequency acquisition speed.

In principle, the controller can include more poles to achieve sharper cut-off characteristics, a desirable property in many applications. However, such applications are difficult to stabilize.

4.7 From continuous-time to discrete-time

In the previous sections, an s-domain analysis was performed. Nowadays, it is more interesting a digital approach to the PLL design. In the years, it is possible to do a kind of classification of the PLL technology. Starting from the classic analog PLL (APLL), in which both signals and components are analog, and arriving to all-digital solutions (ADPLL), in which both signals and components are digital, several intermediate models have been developed. In fact, sometimes a digital signal, like a square waveform, is considered as input of an APLL. This is the case, for example, of [34], in which, to estimate the correct behavior of an APLL in presence of a digital input, the s-domain is transformed to the z-domain via the impulse invariant transformation, and [35], that presents an accurate mathematical model of the discrete and nonlinear behavior of the APLL.

Other times, hybrid-solutions using digital filter and VCO were presented.

The classical solution to pass from APLL to ADPLL is the substitution of all the analog components (analog phase-detectors, filters and VCOs) with the corresponding digital ones (digital phase detectors and filters, Digitally Controlled Oscillator DCO or adder, with the presence of time-to-digital converters) [36].

It's interesting to note that digital implementations introduce new design problems: quantization, aliasing and the non-continuous knowledge of the input signal. In particular, the last problem is sometime faced trying to estimate its value when it is not available from the knowledge of the also estimated frequency, as in [37].

Finally, in [38], a mathematical description and operational details of the phase-domain ADPLL are presented.

4.8 PLL's advanced topology

In the literature, a lot of papers deal with PLLs, both in analog and digital domain, and their improvement, in particular concerning aided frequency acquisition, frequency multiplication, acquisition time and reliable tracking.

Sometimes, the frequency capture range or the acquisition time can be inadequate. The problem can be faced adding a frequency detection loop. In Fig. 4.6 a conceptual diagram of this kind of solution is presented . Here, the system utilizes a frequency detector and a second low-pass filter, LPF_2, whose output is added to that of LPF_1. If the difference between ω_{in} and ω_{out} is large, the VCO is mainly driven by LPF_2 because the continuous component of the Phase Detector output is negligible. When ω_{out} is getting close to ω_{in}, the output of the Phase Detector increases while the one of the Frequency Detector decreases and becomes null when $\omega_{in} = \omega_{out}$. In this way, the loop gain can be considered relatively constant all over the frequency range and not decreasing.

A PLL can also be used in those applications that require an output frequency that is a multiple of the input one. In this case, it suffices introducing a frequency divider, for example ÷M, in the feedback loop. The drawbacks are that also the loop gain is divided by M and the input jitter is amplified.

It has been shown jitter peaking is a problem linked to the order of the closed loop. A family of solutions tries to solve it reducing the number of poles. That is how a close relative of PLLs, the delay-locked loops (DLLs), works. The idea is that if a periodic input is delayed by an integer multiple of the period, then its phase shift can be considered zero (Fig. 4.7). Thus, the phase detector drives the loop so that the phase difference between the input and the output is an integer multiple of the period of the input signal. So, the VCO and its integral action can be replaced with a delay-line, i.e. a gain, and the order of the system reduced by one, also relaxing the trade-offs among gain, bandwidth and stability. Another advantage is the signal is not more generated but delayed, introducing much less jitter in the output signal. For example, a similar solution is implemented in [39] in which a DLL is added to a PLL to take advantage from both of them. In this way,

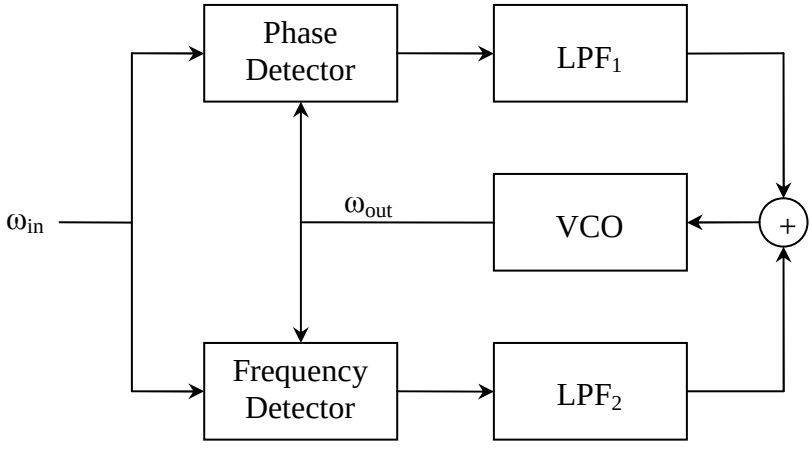

Fig. 4.6. Advanced topology: phase and frequency detector.

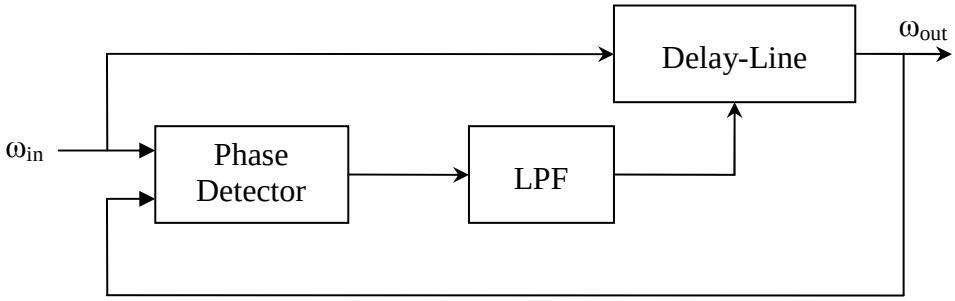

Fig. 4.7. Advanced topology: delay-locked loop.

the acquisition time mainly depends on the DLL, while the filtering properties depend on the PLL. Moreover, the input-output transfer function doesn't produce zeros but only poles in the closed loop, substantially reducing the jitter peaking.

As many applications require both fast acquisition in the beginning and reliable tracking with jitter reduction in steady state, many papers, like [40] and [41], approach the problem using a variable bandwidth. In particular, the first one implements a first order digital PLL using a variable gain as loop filter. This gain is arranged so that the loop gain is large at the beginning of a signal for rapid acquisition, and decreases to a small value for reliable tracking. A finite state machine formalism is used and the performances are analyzed by means of Markov chain techniques. The second one presents a bandwidth adaptive algorithm, based on the recursive least squares criterion, for a second order digital PLL with frequency detection.

Part II

Topologies

5

Master-Slave

In this chapter the reconstruction of a profile transmitted on a non-deterministic network is exploited in the simple master-slave case. The problem of preserving the absolute temporal consistency property is analyzed and the effects of jitter and drift on the transmission explained. Then the controller is designed adopting a phase-locking technique. Finally simulations show the results of this kind of approach to the problem.

5.1 Introduction

In this chapter one node, called master, transmits data on a connection oriented channel to another node, the slave. Therefore the problem of preserving the *absolute temporal consistency property* is faced in the simple master-slave case. Its preservation is an important objective: it reflects many implicit signal properties related to the right data positioning on the time line. Indeed, supposing the trajectory be the position profile of an actuator, a wrong reconstruction has negative effects also on the signal's derivatives, i.e. velocity and acceleration profile, as well as on the signal's spectrum and bandwidth. Firstly, the problem is analyzed in Section 5.2 and all the hypotheses and the considered entities given. Also an explanation of the importance of this property is given by means of an example with considerations in the case the system does not comply with the property. Then the control design begins (Section 5.3). The real and desired system behavior are deeply analyzed and the control algorithm is developed into two steps: the regeneration of the transmitting master clock period locally to the slave and the creation of data when requested by the slave node. At last, simulations are reported (Section 5.4).

5.2 Problem analysis

A distributed real-time system, for simplicity made up of two nodes, is considered. The first node, called master, generates and broadcasts data to the second one, called slave, which collects and elaborates the received data. Each node works at its own frequency. Let $f_m = 1/T_m$ and $f_s = 1/T_s$ be respectively the operating frequencies of the master and the slave nodes and T_m and T_s the corresponding periods. The two nodes are connected by means of a bus. In this case, since there are only two nodes and only the master has the permission to transmit data on the bus, there are no problems for the access to the physical medium.

The communication system is event-triggered, since the temporal control is assumed to be external and it is task of the host computer to decide when a message must be sent (Sec. 3.2.5).

The information transmitted from the master to the slave is a trajectory $x(t)$, i.e. a curved path which is a temporal function. Therefore a certain value is associated to every time instant and vice versa.

At every master tick, $x(t)$ is sampled. Considering the time instant of the first sample as the time origin, a set X_M of elements is created:

$$x_{M,k} = x((k-1)T_m) \tag{5.1}$$

where the subscripts $k = 1, ..., \infty$ and M define the $k-th$ sample generated by the master node. As for the original trajectory, a certain time instant is implicitly coupled with a sampled value, in particular the $k-th$ sample is coupled with the $(k-1)T_m$ time instant. So, every sample has not only a dimensional meaning related to its value, but also a temporal significance linked to the sampling time instant, and in particular to the temporal distance among samples. This is the absolute temporal consistency property.

Then, sampled data are broadcast by the master at the instant of their definition by means of an interrupt, which is a sending-data event.

Therefore the type of traffic generated by the master is essentially periodic. This will be an advantageous condition in order to solve the problem.

Let the transmission channel be connection oriented (Sec. 2.3.2), that is the delivery order is always respected (i.e. the receiving order agrees with the sending order). When a datum arrives to the slave node, a receiving-data event is generated. The received value is identified by:

$$x_{S,k}(t_{a,k}) = x_{M,k} \tag{5.2}$$

where $t_{a,k}$ corresponds to the receiving time of the $k-th$ sample. The subscript S identifies the data received by the slave. The set made up of these elements will be X_S.

The data transfer is not instantaneous because the time interval between the sending operation and the receiving one is not equal to zero. Moreover, it is not constant. Assume that the generic receiving time of the $k-th$ sample is:

$$t_{a,k} = (k-1)T_m + \Delta_r + jitter(k) \tag{5.3}$$

where:

- Δ_r is the average constant delay of the transmission channel;
- $jitter(k)$ is the stochastic variable having average value equal to zero and maximum bounded value $jitter_{max}$. The jitter corresponds to the deviation from the average constant delay. Obviously, $jitter_{max}$ must be less than Δ_r to let the system be causal, otherwise it means there is the possibility the message arrives before its sending event. The sources of jitter can be very different. There can be smaller latencies on the two nodes, due to the managing of an event-triggered systems (access times, priorities of other active tasks...) and bigger latencies as data retransmission in case of transmission error (or even in case of loss of arbitration if many nodes are connected to the bus and an access mechanism is required, based for example on a priority system).

It's worth to note that only the value of the sampled data is transmitted and not the information concerning the sampling time instant or the master period. The data temporal

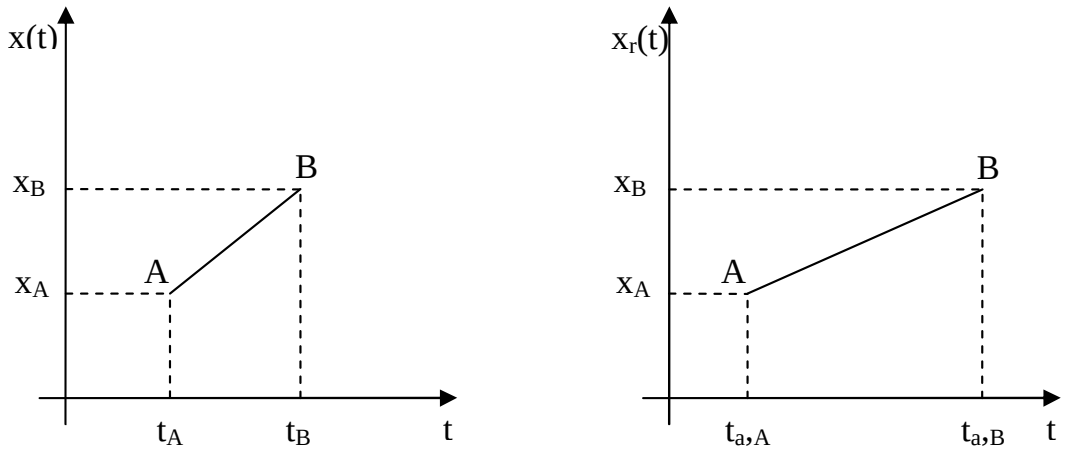

Fig. 5.1. Original trajectory (a) and a possible reconstruction (b)

meaning, i.e. its temporal relation with the other data, is lost during the transmission. For example, let's consider Fig. 5.1-a in which a trajectory $x(t)$ is depicted on the time-space plane. The ending points A and B are respectively described by the couples (t_A, x_A) and (t_B, x_B), but the master will transmit only x_A and x_B. So, the slave will not have any information about their reciprocal temporal positioning. If their arrival time $t_{a,A}$ and $t_{a,B}$ are considered as the time instants in which the slave has to position data, a possible result is depicted in Fig. 5.1-b: x_A and x_B are preserved, but the new trajectory is a different waveform, in particular with a different slope.

The original waveform results as more distorted as data positioning is wrong, that is the time interval between data is different from the original one.

Another possibility could be the placing of arrived data in such a way that the temporal distance among data is the nominal sampling value at the master node. Obviously this is a wrong solution too because the nominal master sampling time does not correspond to the effective sampling time. But even in the very lucky, and impossible, case in which these values are equal, another problem is not considered: the drift (Sec. 3.3.1). In the long period, owing to the drift, the nominal and real values will be different again. In both cases, not considering the drift, another problem arises. The master produces more, or less, data than the number consumed by the slave. So, there will be a filling, or emptying, of the slave buffer. Therefore a mechanism to manage data flow must be introduced (Sec. 2.4).

The right data positioning along the time line would be easier if a global time basis would be available. In this way, there would be a consistent synchronization between the clocks of master and slave and the temporal information related to each sample could be easily regenerated. This is the approach used in deterministic time-triggered networks, typical when periodic transmission are considered. But what we are interested in are non-deterministic networks, in which there is no time-triggered approach.

Finally, master and slave work at different frequencies, but the slave needs a new datum at each own tick. Owing to this condition, missing data have to be rebuilt.

All the analyzed steps are shown in Fig. 5.2.

Aim of the control system is to supply the right timing to the arrived data, estimating the period T_m and reducing the effect of the jitter, and to re-sample the re-timed trajectory by means of the slave's period.

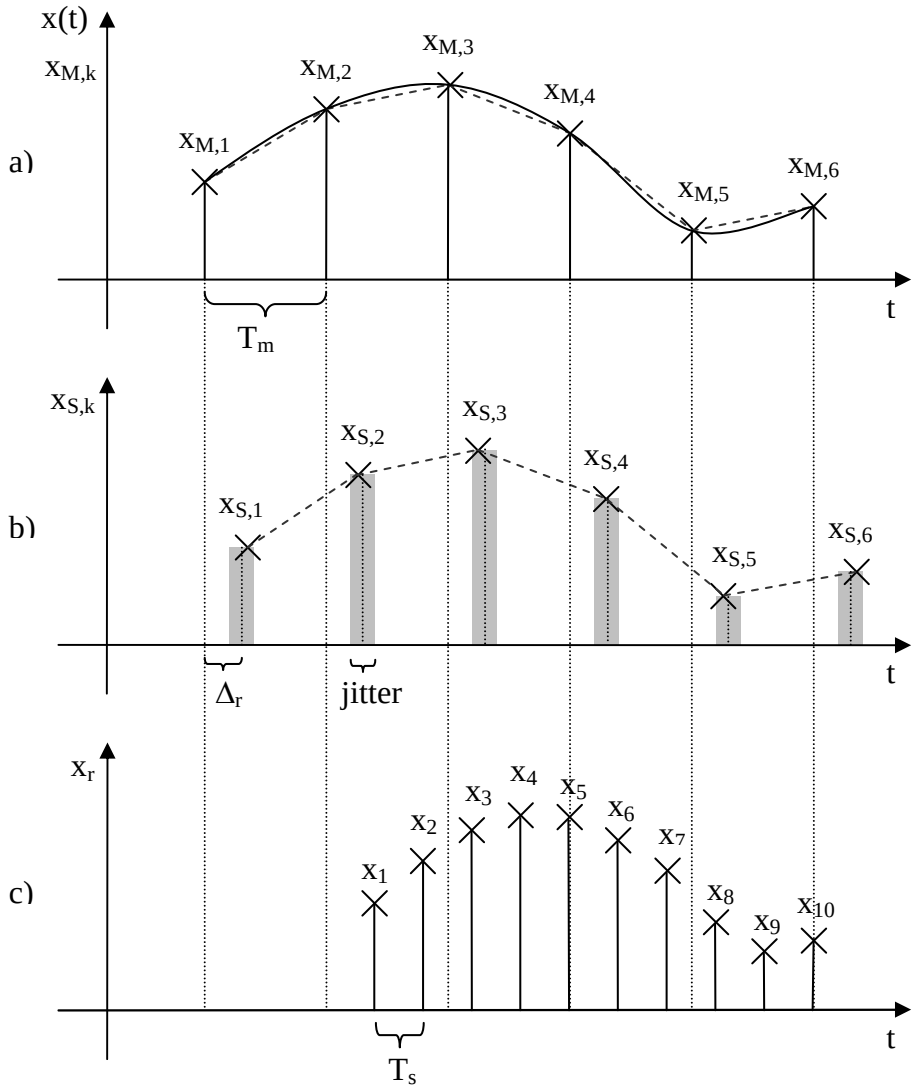

Fig. 5.2. Trajectory sampling (a), data transmission (b) and trajectory rebuilding (c)

5.3 Control design

The reconstruction of the original trajectory $x(t)$ develops into two steps:

1. the regeneration of the transmitting master clock period locally to the slave;
2. the creation of data when requested by the slave node.

In Fig. 5.3 the system's logical architecture is presented:

- every T_m seconds, the master node broadcasts data to the slave node by means of the transmission channel;
- when data arrive, the communication system generates an interrupt and the slave's counter tick value is stored (a counter has been introduced in the slave configuration. Its task will be explained later in Sec. 5.3.1). Then, the control algorithm used for the master clock regeneration is updated;
- when the slave asks for data, i.e. every T_s seconds, the data creation algorithm computes new data using the information provided by the previous step.

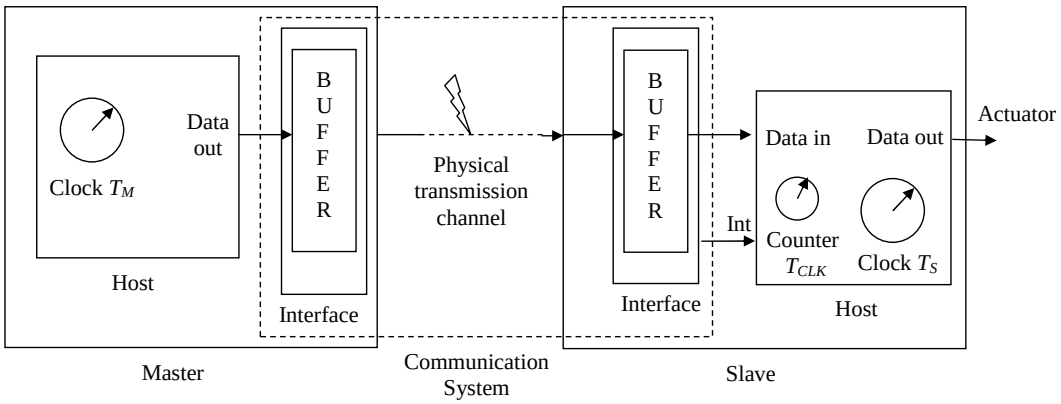

Fig. 5.3. System's logical architecture

In practice, the control algorithm tries to assign the correct temporal distance between received data by estimating the master clock period. In order to estimate the correct value and to be aligned with the transmitting period, a phase-locking technique has been adopted and a phase-locking loop created (Chap. 4). Using this kind of procedure no information, like the nominal value of the master clock period, are introduced in the slave node.

At last, the control algorithm resamples the obtained trajectory by means of the slave clock period.

5.3.1 Signal analysis

In the following, the basic concepts of the proposed solution will be presented neglecting the drift of the clocks, for sake of simplicity. So, all the reflections will be done assuming the clock's nominal periods equal to their effective periods. In any case, the proposed approach works even in presence of different drifts in both nodes because the slave, thanks to the phase locking property, is able to adapt to the sampling rate of the master, even if it is drifting.

In order to proceed with the master clock regeneration, an analysis of the real and desired behavior is performed assuming $f_s > f_m$ (Fig. 5.4).

The master node generates periodic interrupts. Each one can be represented by the rising edge of a square waveform (Fig. 5.4-a) having period equal to T_m. The transmission of the $k-th$ datum is subject to a variable delay, as seen in Sec. 5.2. So, the interrupts generated on the slave by data arrival can be represented by the rising edge of an almost periodic waveform, phase-modulated by the jitter (Fig. 5.4-b). This is the real behavior of the system.

Now, the desired behavior will be analyzed.

If the transmission delay would be constant, the reconstructed master clock would correspond to the one reported in Fig. 5.4-c. Obviously, the signal's period would be constant, too, and equal to T_m. What is really desired, yet, is a reconstructed clock subject to a constant phase shift αT_m, $0 < \alpha < 1$, as regards to the ideal arriving signal (Fig. 5.4-d). It means $\Delta\varphi$ phase reached by the shifted ideal reconstructed clock is always the same when data arrive in absence of jitter. This phase shift is required to avoid that data are asked by the slave before their arrival. In fact, if the slave would be perfectly aligned with the ideal waveform in Fig. 5.4-c, the jitter could give rise to such a delay to make

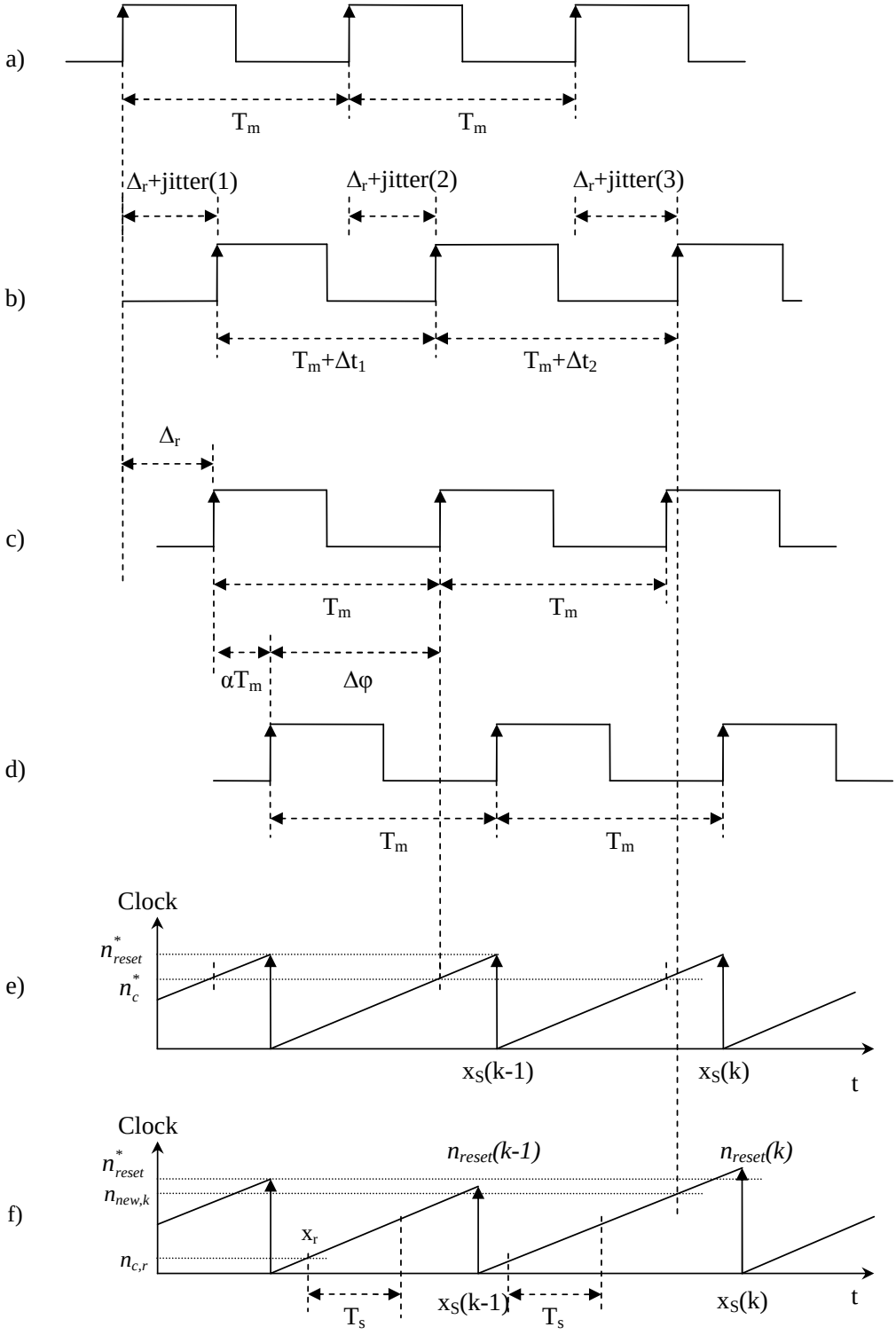

Fig. 5.4. Master clock (a), waveform generated by the incoming data on the slave (b), ideal received waveform (without jitter) (c), desired regenerated clock (d), desired counter's progress (e) and real counter's progress (f)

data not available when required. For this reason, phase shift αT_m has to be greater than $jitter_{max}$.

Besides having two clocks, master and slave, a high-frequency counter, $f_{clk} = 1/T_{clk}$, $f_{clk} >> (f_s, f_m)$, is introduced to measure and reproduce the time interval between two consecutive interrupts on the slave node and to be able to know the counter's tick corresponding to the time instant in which the slave asks for data.

The ideal condition for a perfect reconstruction of the original trajectory is that T_m, T_s, T_{clk} are commensurable, that is the ratios T_m/T_{clk} and T_s/T_{clk} must be integer numbers. In this way, perfect alignment between clocks and counter is possible. Clearly, this situation is unreachable: the presence of uncertainties (nominal clock vs. real clock, drift, aging of components...) does not allow to arbitrarily set and know the previous ratios so a sort of quantization effect is introduced.

In the ideal situation, the above mentioned condition of phase shifting calls for the same counter's tick number (n_c^* in Fig. 5.4-e) each time one datum arrives to the slave in absence of jitter. Moreover, the time interval between two consecutive data arrivals is constant, equal to T_m and equivalent, on counter's tick basis, to n^*_{reset}, the counter's resetting number.

In conclusion, aim of the controller is the reconstruction of a waveform the most possible similar to the one shown in Fig. 5.4-e. This goal is achieved controlling the resetting value of the counter. The controller has to make up for phase if the regeneration is slower than the real signal or, vice versa, to lose phase if the regeneration is faster than the real signal by respectively decreasing and increasing the resetting value in accordance with the control law. The effective waveform will result like the one in Fig. 5.4-f. Once the clock has been recovered, the positioning of x_S data is allowed (see also Fig. 5.2-b).

At last, trajectory rebuilding, as seen in Fig. 5.2-c, will be based on the regenerated master clock, as shown in Fig. 5.4-f.

Remark 5.1. The properties of a trajectory are preserved only if the right time spacing among data is guaranteed, in particular with reference to the absolute temporal consistency property. They do not depend on the shifting in time of the complete trajectory. So, the αT_m phase-shift ($> jitter_{max}$) can be chosen at will.

5.3.2 Master clock regeneration

A phase-locking technique is implemented in order to recover the master clock period (Fig. 5.5).

The basic ideas of a PLL scheme are briefly summarized in the following:

- a digital counter is used to physically regenerate the master clock;
- the control loop reference is the measured phase of the incoming signal;
- the controller is designed as a low-pass filter.

At the same time, this approach allows to handle the jitter as a high frequency noise and to follow the clock drift.

Reference

The reference is the measure of the phase of the incoming signal. It is available only when data arrive. This situation makes the knowledge of the signal's phase in a time instant between two consecutive interrupts not possible.

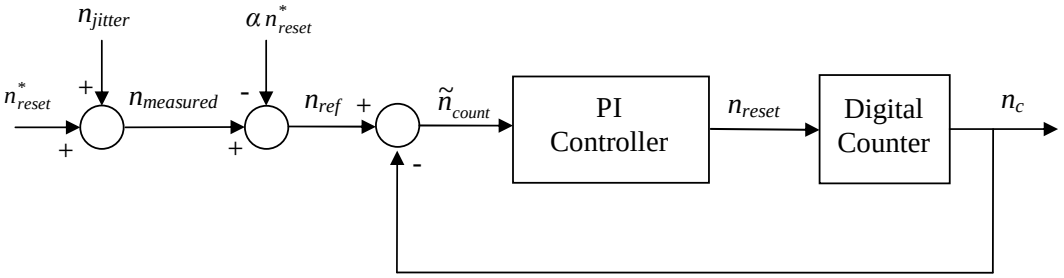

Fig. 5.5. PLL scheme adopted for master clock regeneration

The phase is measured counting the counter ticks between two consecutive interrupts. Calling $n_{jitter}(k)$ the amount of ticks owing to the jitter that change the real value of the phase and considering the desired phase-shift αT_m, also expressible in ideal condition as αn^*_{reset}, the ticks number of reference is:

$$n_{ref} = n^*_{reset} + n_{jitter}(k) - \alpha n^*_{reset} \tag{5.4}$$

Obviously $n_{jitter}(k)$ depends not only on the jitter of the k-th data arrival, but also on the jitter of the $(k-1)$-th data arrival since both the jitters modulate the phase of the signal. This consideration is more clear looking at Fig. 5.2-b, where $n_{jitter}(k)$ corresponds to the entity Δt_i expressed on the counter tick basis, neglecting the granularity of the counter.

Plant

The system is a digital counter controlled by means of the resetting input n_{reset}. Therefore it can be represented by a discrete integrator. Obviously, the output of the counter is the counter's actual value n_c:

$$n_c(z) = \frac{1}{1 - z^{-1}}(n^*_{reset} - n_{reset}(z)). \tag{5.5}$$

Controller

Its aim is twofold:

1. to provide the trajectory rebuilder with the average resetting value $\bar{n}_{reset}$
2. to compute the next resetting value n_{reset} of the digital counter so that the rebuilt master clock follows the ideal waveform presented in Fig. 5.4-e.

The difference equations describing the controller are:

$$\begin{cases} \bar{n}_{reset}(k) = a * \bar{n}_{reset}(k-1) + (1-a) * [n_{reset}(k-1) - n_c(k-1) + n_c(k)] \\ n_{reset}(k) = \bar{n}_{reset}(k) - gain * \tilde{n}_{count}(k) \end{cases} \tag{5.6}$$

where:

- a is the coefficient of the first-order low-pass filter described by the first equation. The quantity inside the square brackets represents the phase of the incoming signal, so a is a weight for the previous average resetting value and $(1-a)$ is a weight for the actual measure.

- $\tilde{n}_{count}$ is the counter's error, i.e. the difference between the reference n_{ref} and the actual counter's value n_c;
- $gain$ is the error's weight in the $\bar{n}_{reset}$'s correction.

Processing the controller equations, the relation between counter's error and controlled variable in z-domain is given by:

$$\frac{N_{reset}(z)}{-\tilde{N}_{count}(z)} = (1 - a + gain) * \frac{1 - (\frac{1-a+a*gain}{1-a+gain})z^{-1}}{1 - z^{-1}} \tag{5.7}$$

that is a classical PI controller.

a and $gain$ parameters have to be set to reduce the closed loop's bandwidth and to cut the jitter, modeled as high frequency noise, off.

Remark 5.2. It's worth to note that, from a logical viewpoint, this algorithm does not periodically work because it works only when new data arrive, that is almost-periodically. Anyway, a periodic implementation can be achieved.

Remark 5.3. For simplicity, a PI controller is developed since aim of this work is to demonstrate the effectiveness of this kind of approach. More complex controllers able to obtain better performance are analyzed in PLL's literature as discussed in Sec. 4.8.

5.3.3 Trajectory rebuilding

The algorithm adopted for the trajectory rebuilding is analyzed.

The reconstruction is based on the information provided by the regenerated master clock, i.e. the counter's average resetting value $\bar{n}_{reset}$ and the counter's actual value n_c, and, obviously, on the data stored in the buffer.

In order to start the algorithm, the initial conditional that must be satisfied is the waiting for two data from the master node. Once two data are available inside the slave buffer, then the algorithm starts at the first slave's data request. After that, each time the slave asks for data, that is every T_s seconds, a new value has to be produced and the algorithm periodically works.

The procedure is the following:

1. the counter's actual value n_c is read;
2. a. if a counter's reset has not yet happened, the data stored in the buffer still remain and the buffer is not updated;
   b. if a counter's reset has happened, the data stored in the buffer are updated adopting a FIFO policy, that is the first data is discarded since too old and the second and third data stored in the buffer respectively becomes the first and the second data;
3. a linear interpolation, based on the actual n_c and $\bar{n}_{reset}$, is computed using the data stored in the buffer, i.e. the first two data, as extreme points.

Looking at Fig. 5.4-e, it is clear that a new counter's resetting value is computed before the bounds of the trajectory interval are updated. Calling $n_{new,k}$ the counter's value in which the computation takes place, the adopted linear interpolation formula is:

$$x_r = \begin{cases} \dfrac{n_{c,r}}{\bar{n}_{reset}(k-1)}(\hat{x}_{S,k} - \hat{x}_{S,k-1}) + \hat{x}_{S,k-1} & \text{if } n_{c,r} < n_{new,k} \\ \dfrac{n_{c,r}}{\bar{n}_{reset}(k)}(\hat{x}_{S,k} - \hat{x}_{S,k-1}) + \hat{x}_{S,k-1} & \text{if } n_{c,r} \geq n_{new,k} \end{cases} \tag{5.8}$$

where $n_{c,r}$ and x_r respectively represent the counter's value and the new interpolated data identifying the $r-th$ slave's data request. Obviously, more complex interpolation formulas can be adopted to achieve better estimated values.

In conclusion, the new set of x_r elements constitutes the rebuilt trajectory.

Remark 5.4. The cooperation between the master clock recovery algorithm and the buffer updating process assures data flow control without the need of any other kind of controller or acknowledgement system, that would complicate the overall system and reduce performances. In this way buffer overflows or underruns can be simply avoided thanks to the phase locking property: the alignment of the regenerated master clock with the original master period assures a precise data positioning so that the buffer updating process is able to judge if data in buffer are too old with respect to the actual time and then if a data refresh is required.

5.4 Simulations

The system model has been developed using Matlab 7.0 and Simulink 6.0 and the presented approach has been tested.

In Fig. 5.6 the simulation scheme is presented. It is possible to distinguish the single components of the simulation: the trajectory generated by the master with the corresponding generation signal on the left, the slave on the center, the buffer on the right and the counter and the controller on the bottom.

Fig. 5.7 shows that the trajectory is sampled by means of the master clock period and both the sampled value and the corresponding sampling event are transmitted and subject to a constant delay plus a variable one.

Fig. 5.9 and Fig. 5.8 show the architecture of the counter and the controller.

Fig. 5.10, Fig. 5.11 and Fig. 5.12 represents the rebuilding algorithm. It is divided into two sections, master and slave. The first one is responsible for filling the buffer with the arrived data, while the second one takes care for the rebuilding of the trajectory and the buffer updating.

The simulations have been performed using the parameter values presented in Table 5.1 and in nominal conditions both with $jitter_{max,1}$ and $jitter_{max,2}$. Such choices allow to consider something more than one data retransmission as sources of the jitter, that is assumed as a stochastic variable having uniform distribution. The choice of the controller's parameters is the result of the compromise between the need of reducing the bandwidth, in such a way to cut off the jitter, and the velocity of the phase acquisition. Such a choice leads to a cut-off frequency of 32.3 rad/sec and allows to reduce all the components over 300 rad/sec of at least 20 dB.

In Fig. 5.13 and Fig. 5.15, the system behavior in nominal condition, respectively with $jitter_{max,1}$ and $jitter_{max,2}$, is presented. Since the rebuilding error is very small, 0.15% and 0.3%, the original and the rebuilt trajectories are almost overlapped. Moreover, a correct estimate is performed (Fig. 5.14 and Fig. 5.16). It is possible to appreciate how the algorithm reduces the entity of the oscillations, due to the jitter, of the signal's measured phase.

In Fig. 5.17, the system behavior when the real T_m is the 95% of the nominal T_m and $jitter_{max,1}$ is used is presented. The rebuilt trajectory is late because data are initially used slower than their arrival. In fact, the regenerated master clock is late too and it has

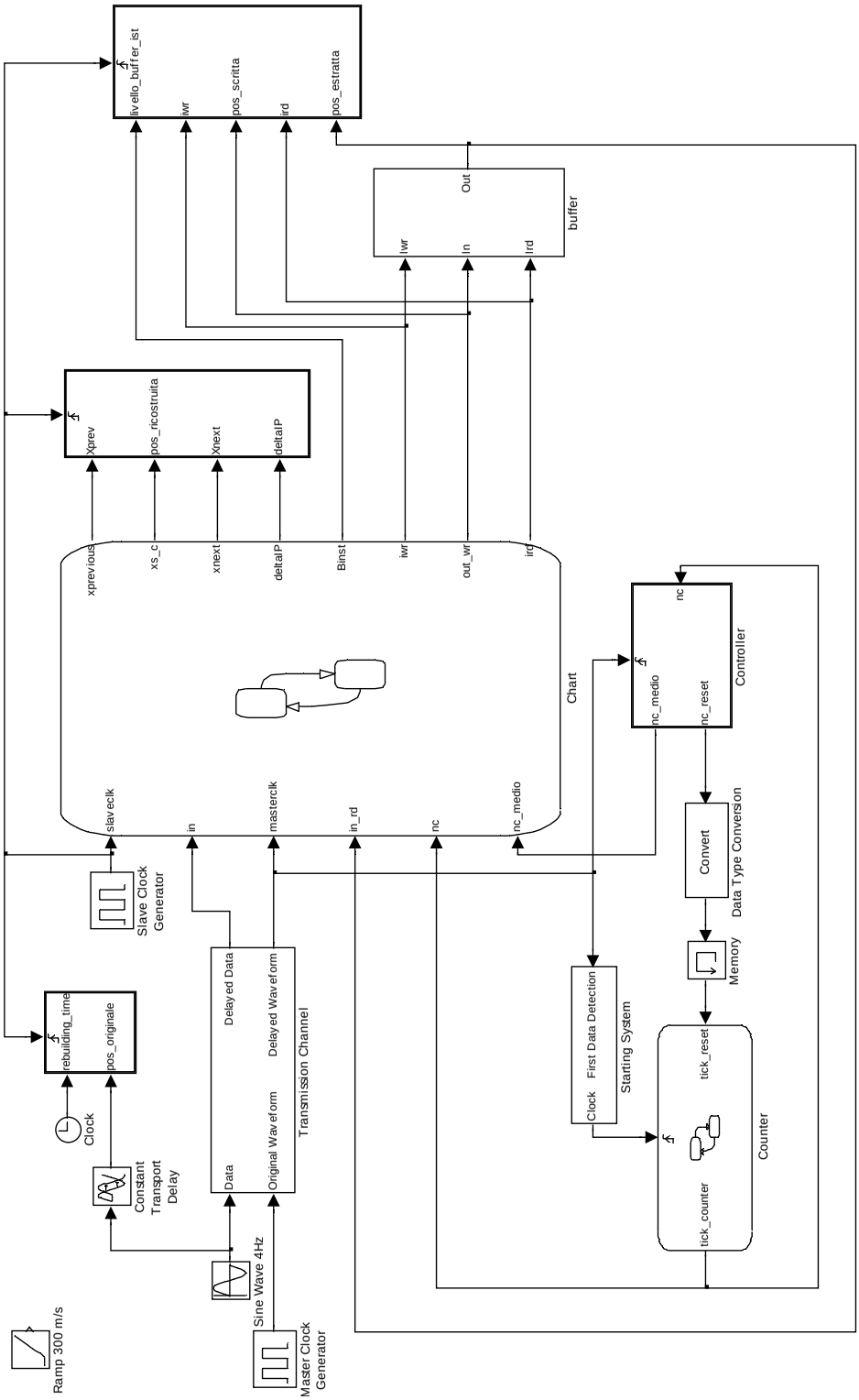

Fig. 5.6. Simulation scheme

to make up for the phase delay. This is performed by the control algorithm as it is shown in Fig. 5.18. The n_{reset} value initially goes below the real n_{reset} value, 4750, to recover the phase delay, and then it realigns itself to the real value. According to this situation,

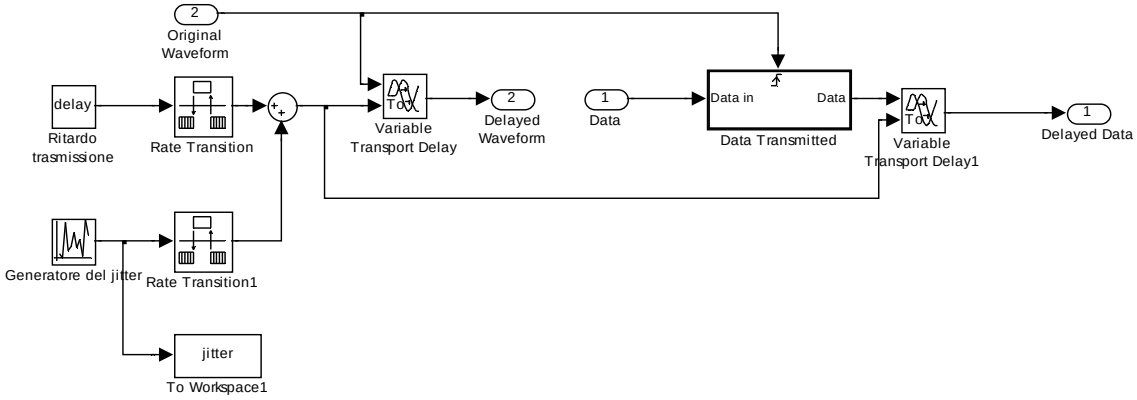

Fig. 5.7. Data transmission

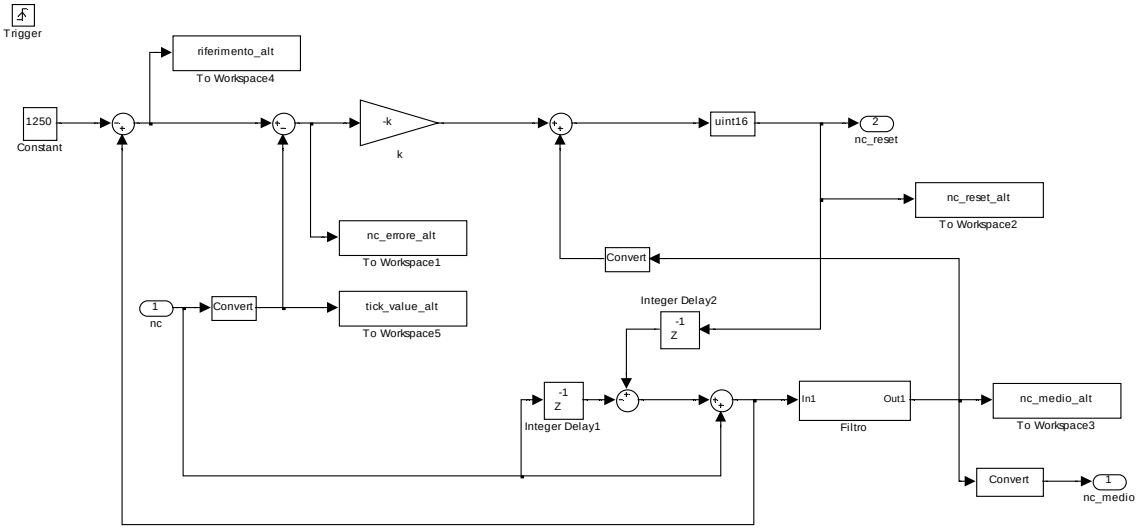

Fig. 5.8. Controller

Parameter	Value
trajectory $x(t)$	$sin(8\pi t)$
nominal T_m	2 msec
nominal T_s	1 msec
nominal T_{clk}	400 nsec
Δ_r	0.5 msec
$jitter_{max,1}$	0.2 msec
$jitter_{max,2}$	0.4 msec
a	0.96907
$gain$	0.032334

Table 5.1. Simulation parameters

the rebuilding error becomes smaller. In particular, the above mentioned choice of the controller's parameters allows to reach an error less than 1% in about 0.15 sec. So, the phase-locking property allows the locking to the real master clock period even when it is different from the nominal value. The direct consequence of this situation is that when this value is locked the slave node will always adapt to the sampling rate of the master,

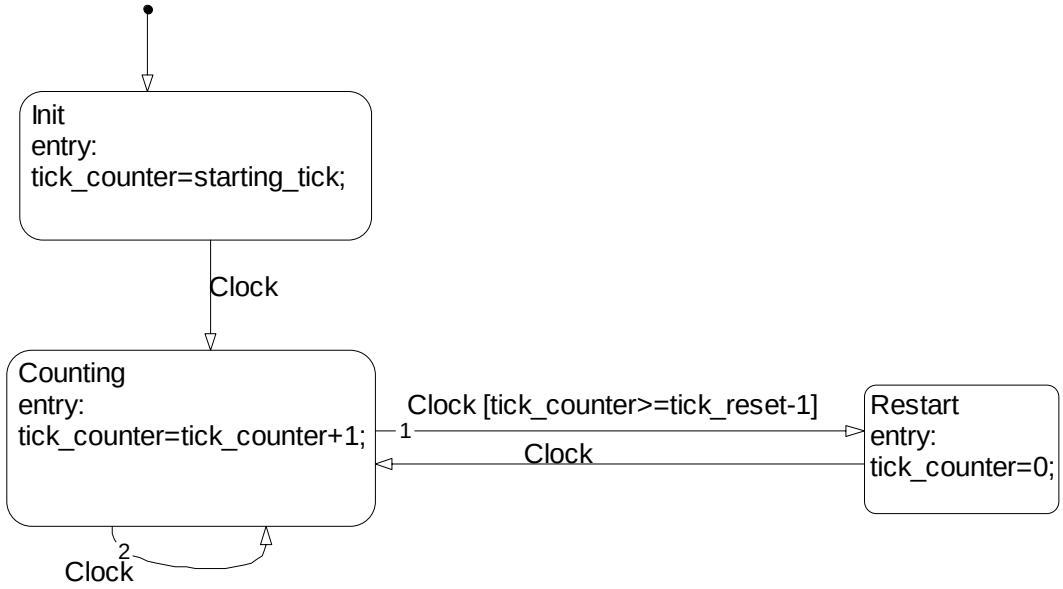

Fig. 5.9. Counter

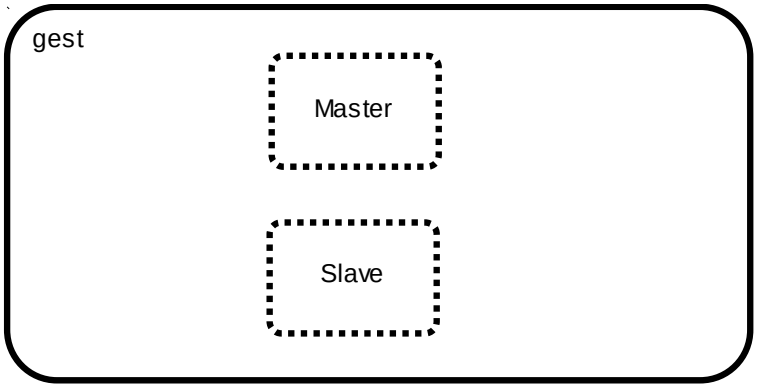

Fig. 5.10. Rebuilding algorithm: overview

as in every PLL-based controller. In this way, the master clock variations due to the drift can be followed too.

The analysis of the spectrum relative to the first period of the rebuilt sine wave in this second situation, i.e. not in nominal condition, highlights the predominance of the $4Hz$ component on all the other spurious components, that are in the order of 10^{-3} (Fig. 5.19).

At last, Fig. 5.20 shows that data flow control is achieved.

5.5 Conclusions

In this chapter, a trajectory rebuilding system developed on a PLL-based synchronization mechanism was discussed. Considering the background of real-time periodic traffic transmitted by means of an event-triggered communication system, the analysis of the considered signals, the ideal and real system's behavior has been dealt. It has been shown that the preservation of the temporal characteristics of a trajectory when it passes on a digital communication channel and it is subject to jitter and clock drift is achievable with a great accuracy. The proposed solution introduces a slight modification of the hardware

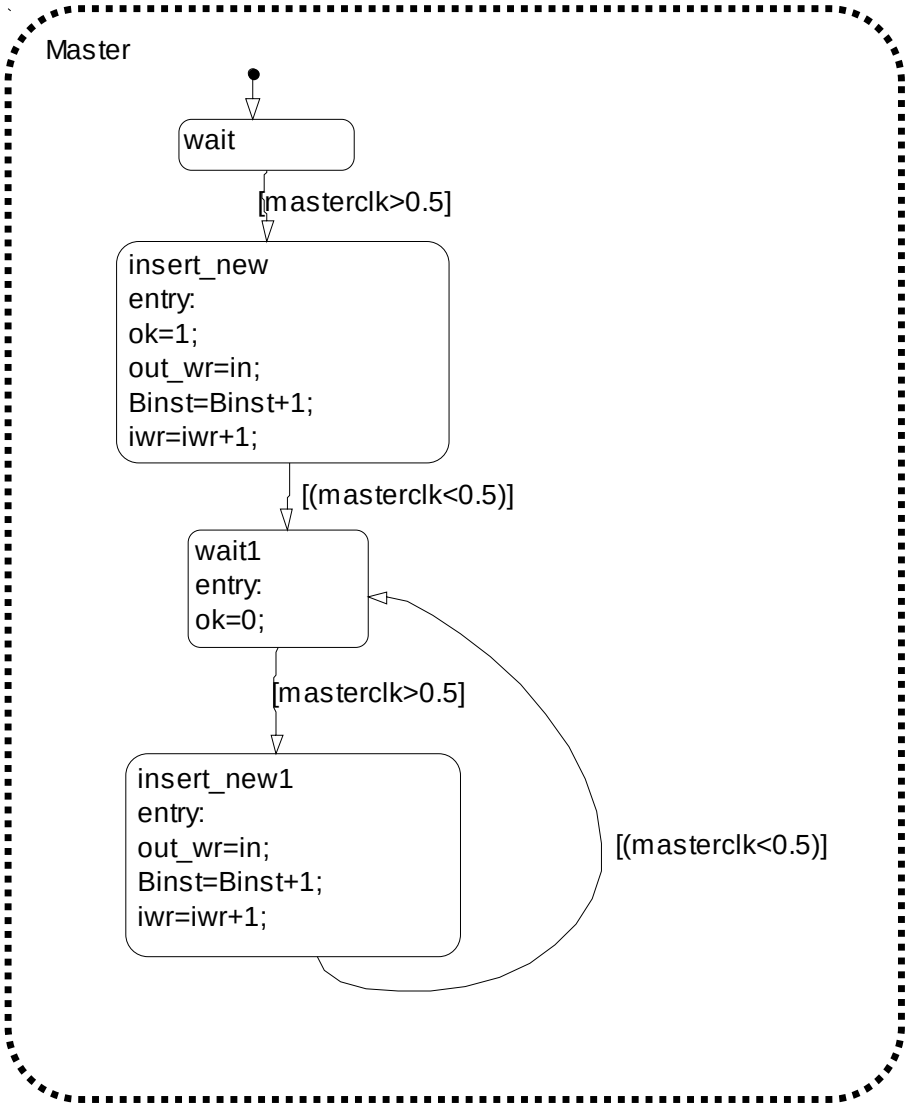

Fig. 5.11. Rebuilding algorithm: handling of data arrival

configuration, that is it adds only a counter, and adopts a simple control strategy, like a PI controller. Moreover, thanks to this kind of solution, data flow control is naturally obtained, with positive consequences on the complexity of the overall system.

At last, the proposed approach has the advantage that the used network bandwidth is reduced with respect to other approaches as time stamping (no expensive agreement protocol is implemented) even if the hardware/processing complexity increases at the slave side. Moreover, at the slave node the a-priori knowledge of the transmitting time instants is not required as in a time-triggered protocol with a global time base since the slave side is able to adapt to the transmitting period. Thanks to this feature, the hot-plug of a new slave node in the net could be eventually allowable.

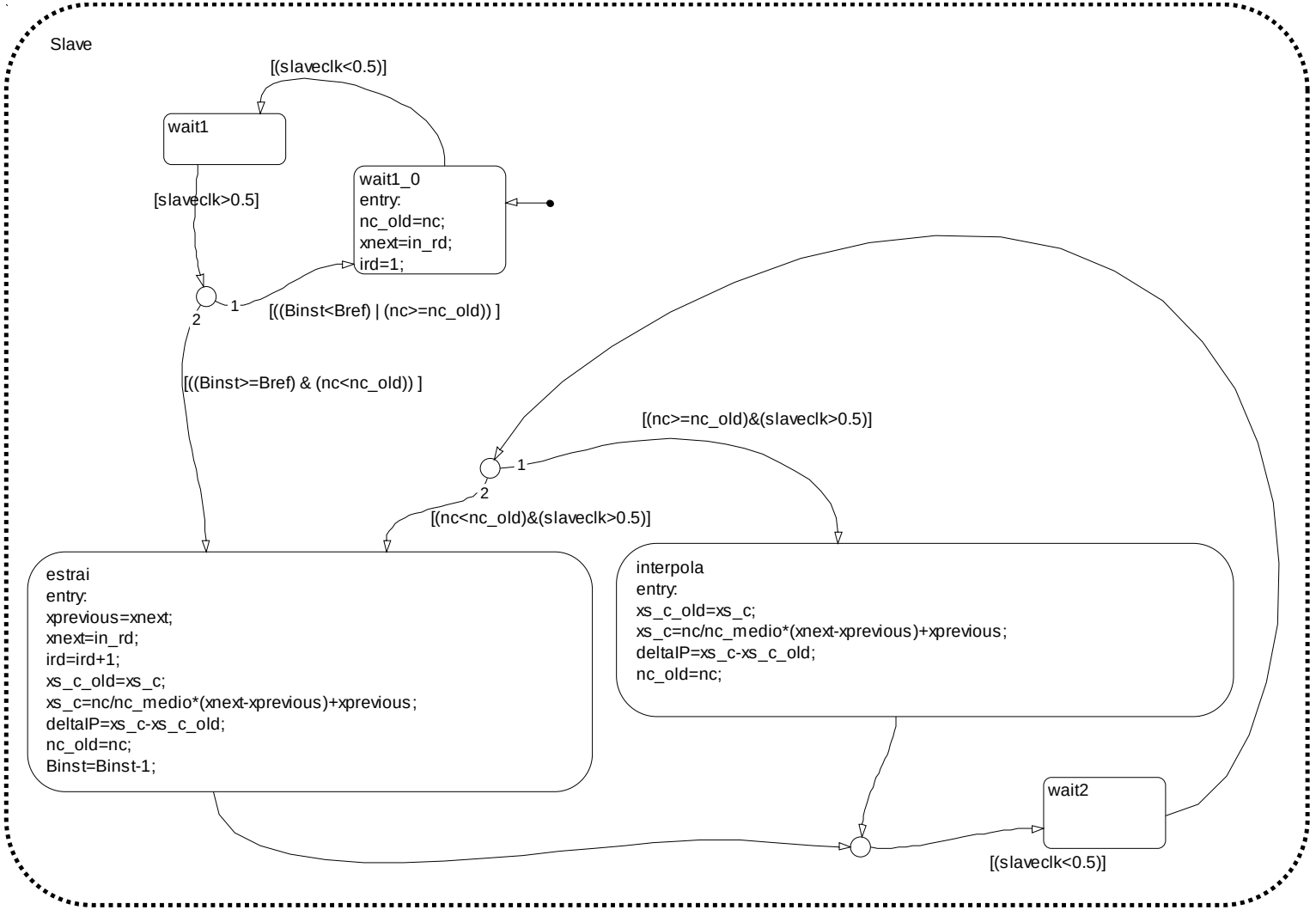

Fig. 5.12. Rebuilding algorithm: buffer updating and trajectory rebuilding

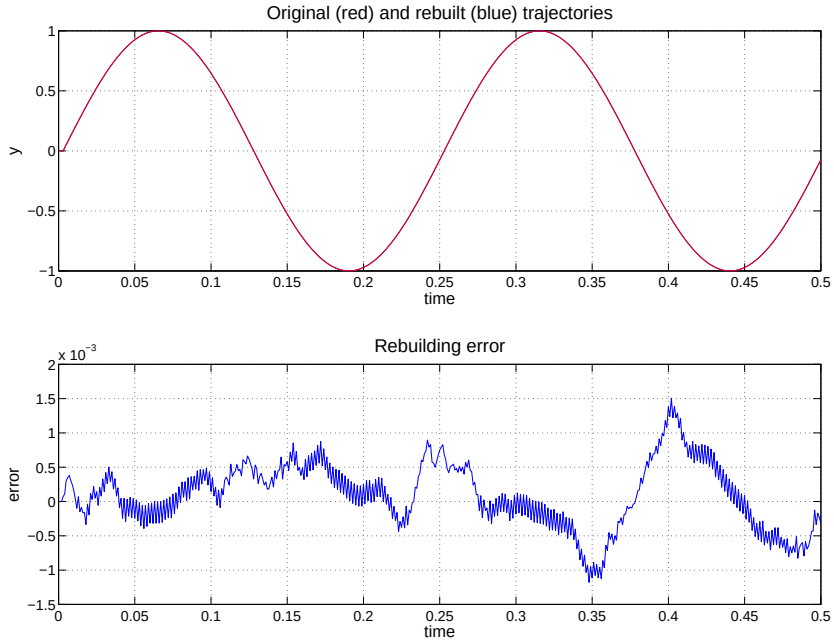

Fig. 5.13. Real T_m=nominal T_m, $jitter_{max}$=0.2 msec: Rebuilt and original trajectories and rebuilding error

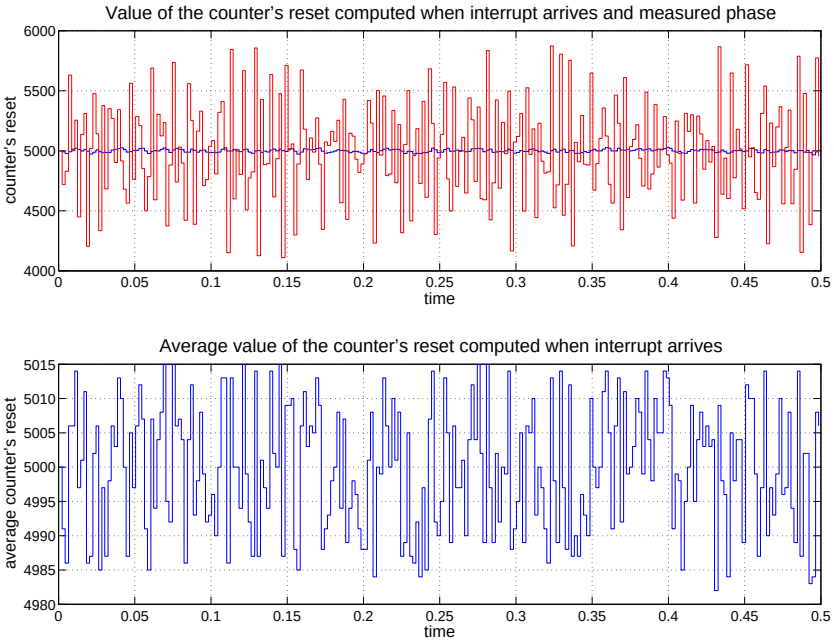

Fig. 5.14. Real T_m=nominal T_m, $jitter_{max}$=0.2 msec: n_{reset} (compared to the measured phase) and $\bar{n}_{reset}$

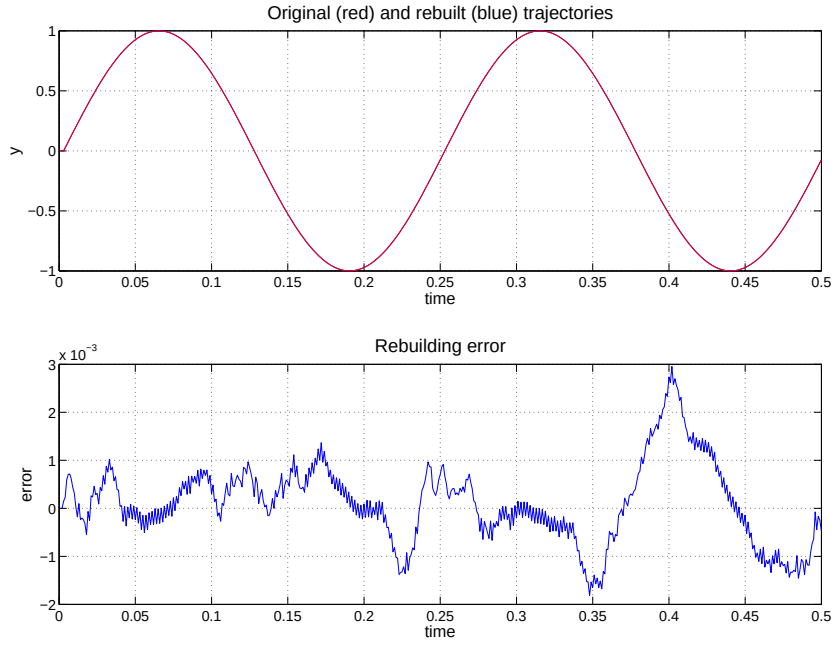

Fig. 5.15. Real T_m=nominal T_m, $jitter_{max}$=0.4 msec: Rebuilt and original trajectories and rebuilding error

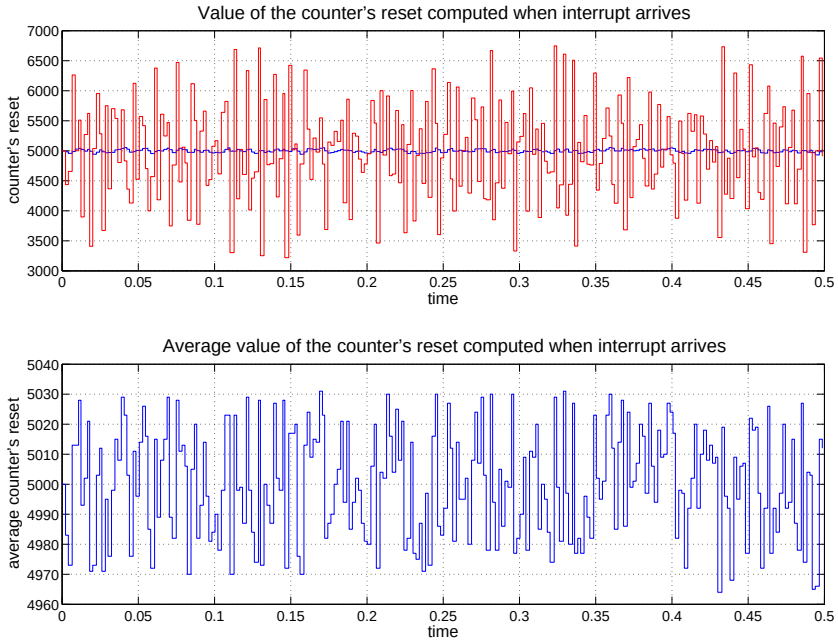

Fig. 5.16. Real T_m=nominal T_m, $jitter_{max}$=0.4 msec: n_{reset} (compared to the measured phase) and $\bar{n}_{reset}$

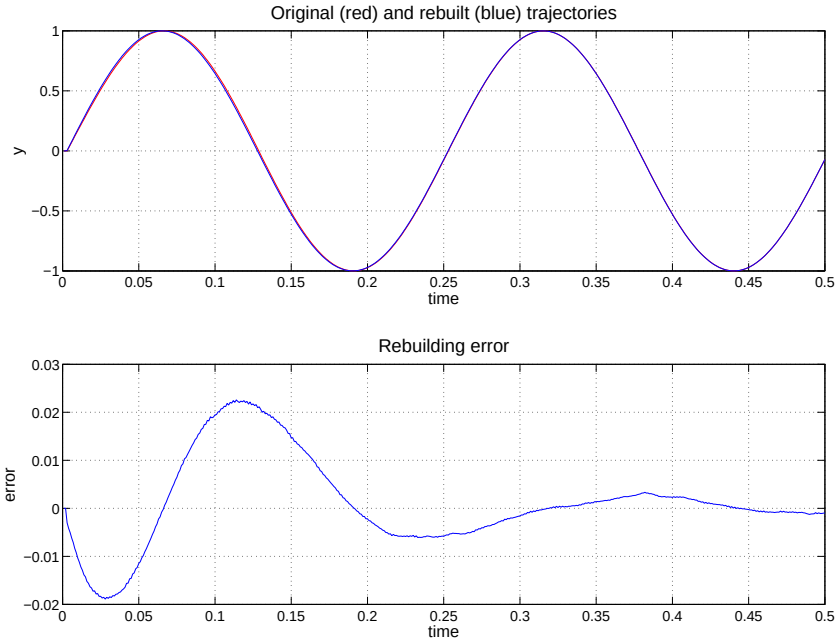

Fig. 5.17. Real T_m=95%nominal T_m: Rebuilt and original trajectories and rebuilding error

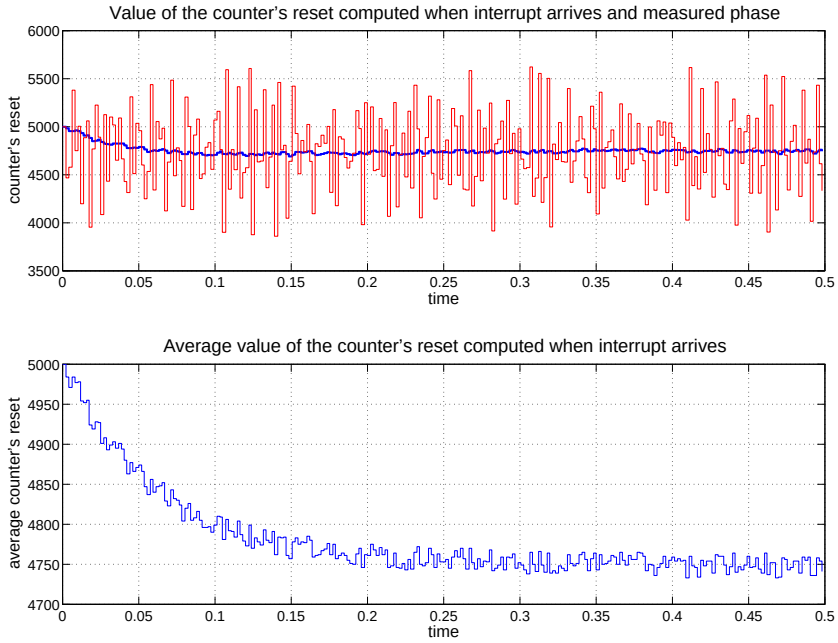

Fig. 5.18. Real T_m=95%nominal T_m: n_{reset} (compared to the measured phase) and $\bar{n}_{reset}$

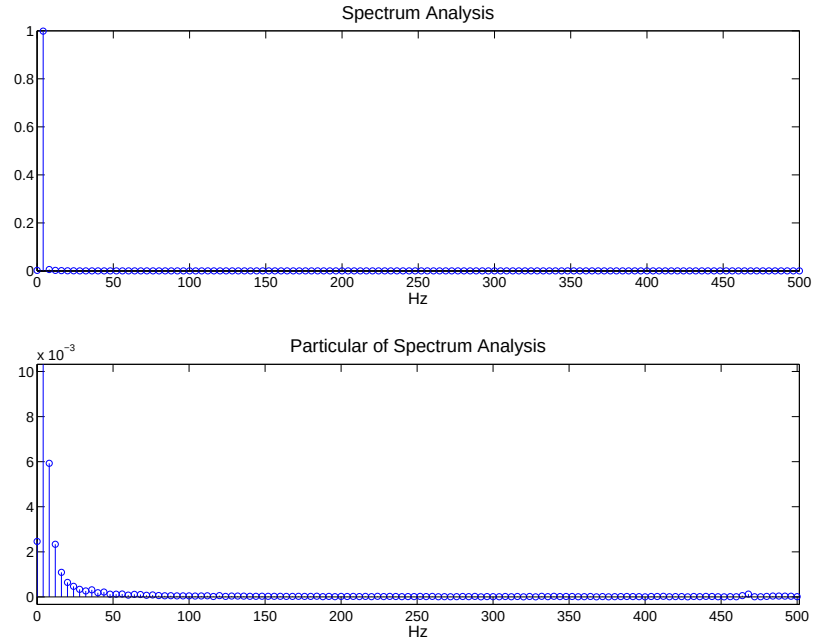

Fig. 5.19. Real T_m=95%nominal T_m: Spectrum analysis

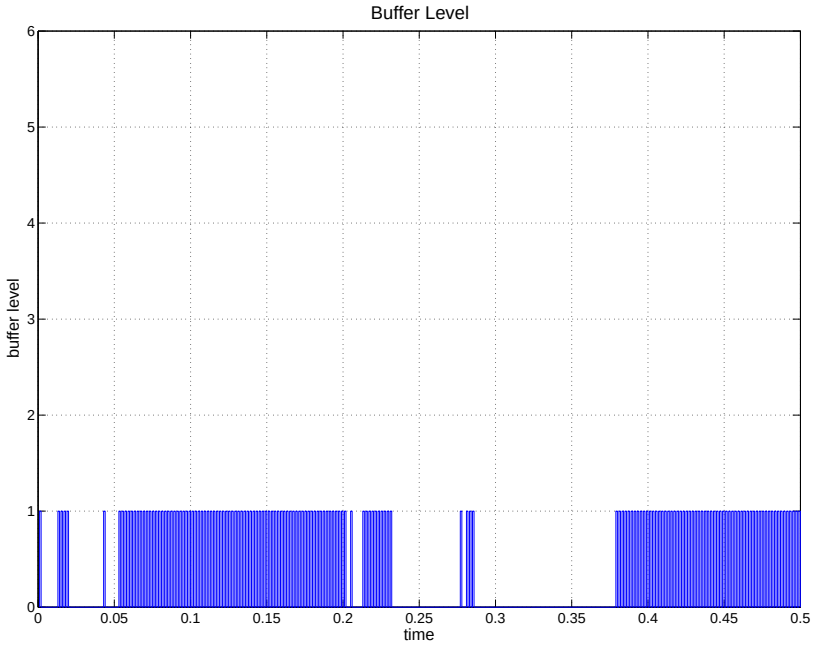

Fig. 5.20. Buffer level

6

Master-Multislave

In this chapter the approach presented in chapter 5 has been improved to deal with the master-multislave topology. The modifications to the original algorithm are presented both at the master and slave side. At last simulations show the effectiveness of the approach.

6.1 Introduction

In chapter 5 the clock synchronization and trajectory rebuilding have been achieved in the case of one master node transmitting one trajectory to one slave node. However modern systems are made up of several nodes having to work as an harmonious whole.

In motion control, a typical situation is a central master node transmits many synchronous trajectories to different nodes in order to obtain synchronous movements among different parts of the distributed system. For example, the axes of a robotic arm must be moved in such a way to reach a precise final position following a defined three-dimensional trajectory. So, a simple master-slave synchronization is not sufficient to achieve this aim and a master-multislave synchronization is required.

While in the first case the data absolute temporal consistency property has to be preserved, now the preservation of the data relative temporal consistency property is needed since there are many synchronous trajectories on different nodes, meaning that samples from different signals must be correlated in time.

In order to solve this problem, the basic ideas are the master sends one Broadcast Message to all nodes and then transmits the sampled data to every single node. In the meantime, the slaves rebuild the original trajectory adopting the previously proposed algorithm running when these Broadcast Messages arrive and not when data arrive.

In the following, the procedure will be explained in details both at the master side and the slave side.

6.2 Master Side Procedure

In the master-slave case, the master periodically samples the trajectory then transmits the samples. In the master-multislave case, many trajectories must be sampled then their samples transmitted.

In order to preserve the relative temporal consistency property among data of different trajectories at the master side, the master node periodically samples all the trajectories at the same time instant. Moreover, another assumption is done: a message sent to all nodes,

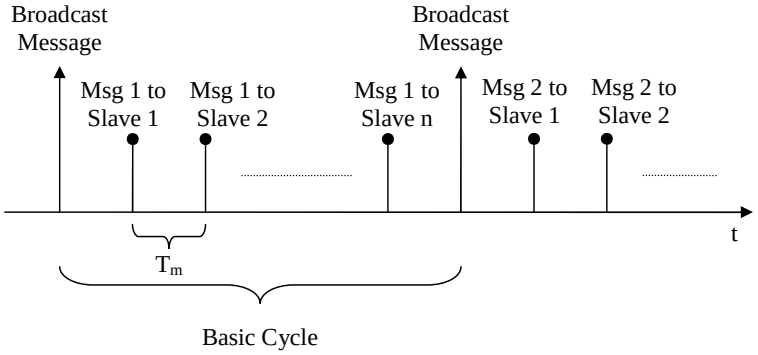

Fig. 6.1. Basic cycle

belonging to the same net layer, arrives to all nodes at the same time instant. This kind of message will be called "Broadcast Message", BM.

The next step of the procedure is the definition of the basic transmission cycle. Each basic cycle is assumed to start with the Broadcast Message. This means that during the BM event all the trajectories are sampled and the BM is sent on the network. Later, each time the master clock ticks, all samples are transmitted to the single slave nodes: at the first tick after the BM, the first sample of the first trajectory is sent to the slave 1; at the second tick, the first sample of the second trajectory is sent to the slave 2, and so on until the last n-th trajectory is sent to the last n-th slave (Fig. 6.1). Then a new cycle begins.

Assuming the master clock period equal to T_m and the number of trajectories to be transmitted equal to n, the Broadcast Message is $(n+1)T_m$ periodic.

6.3 Slave Side Procedure

The control algorithm running at the slave side is essentially the same adopted for the master-slave case with two slight modifications owing to the introduction of the basic transmission cycle.

The fundamental assumption is that if all the trajectories are sampled at the same time instant, that is when BM is sent, the slaves must position the samples in correspondence with the regenerated Broadcast Message event. Moreover, since the BM is supposed to arrive at the same time instant to all nodes, all the slaves will regenerate the same waveform and will position data at almost the same time instant. Obviously, the effective positioning time instant will not be exactly the same at all nodes because of the granularity of the adopted counter and hardware/software implementation. So, in the design phase, this approximation should be as more negligible as possible, as it will be considered in the following simulations.

The first change to the original algorithm concerns the activation time instant of the master clock regeneration algorithm, as previously anticipated. While in the master-slave case the updating process of the algorithm worked at the data arrival time instant, now it works at the BM arrival event.

The second change concerns the phase-shift αT_m required in the master-slave case. Since in the master-multislave case the basic cycle is introduced, the phase-shift is not needed. Indeed, having a connection oriented communication channel, when the new BM arrives to the slaves, the sample of the n-th trajectory is already arrived to the n-th

slave. From a different point of view, the phase-shift could be considered equal to T_m for the n-th trajectory, $2T_m$ for the $(n-1)$-th trajectory and so on.

This modification implies that the reference used for the master clock regeneration algorithm is only the measured phase of the incoming signal, that is:

$$n_{ref} = n^*_{reset} + n_{jitter}(k) \tag{6.1}$$

Thanks to these changes, the Broadcast Message works as a synchronization message sent to all nodes. So, when the slave will regenerate the master clock period, the regenerated waveform will have a period equal to $(n+1)T_m$.

In this way, the preservation of the data relative temporal consistency property is achievable because the samples of different trajectory computed at the same time instant are positioned at the same time instant of the regenerated waveforms.

Moreover, since the master clock is recovered using only the BM, some flexibility is introduced in the definition of the events belonging to the basic cycle. For instance, the data order of delivery could be modified run-time by the master or the basic cycle could be a little bit longer than $(n+1)T_m$ in order to have more time available to transfer other kind of data or satisfy some priorities.

6.4 Simulations

The simulations have been performed in nominal conditions considering a system composed by one master transmitting three trajectories to three slaves.

The adopted parameter values are presented in Table 5.1. The counter of each node has been supposed identical, that is with the same granularity, for each slave.

Parameter	Value
trajectory $x_A(t)$	t
trajectory $x_B(t)$	$sin(8\pi t)$
trajectory $x_C(t)$	$cos(8\pi t)$
nominal T_m	1 msec
nominal $T_{s,A}$	3.5 msec
nominal $T_{s,B}$	2.4 msec
nominal $T_{s,C}$	2.2 msec
nominal $T_{clk,ABC}$	1 μsec
Δ_r	0.5 msec
$jitter_{max}$	0.1 msec
a	0.96907
$gain$	0.032334

Table 6.1. Simulation parameters

Fig. 6.2 shows the simulation scheme: one master, by means of the transmission channel, transmits data to three slaves. In Fig. 6.3 the master scheme is shown. The introduced modifications allow to generate the basic cycle, that is the broadcast signal and the single data transmission of each trajectory, accompanied by an identifier. Obviously at the slave side some modifications to recognize the broadcast message and the data to be received have been implemented, as shown by Fig. 6.4.

In the first plot of Fig. 6.5 the ideal and real arrivals of the Broadcast Message are presented while in the second, third and fourth plot the BM positioning along the timeline

versus the ideal positioning is shown. Since the adopted counters and controller parameters are the same for the three slaves, the regenerated BM is positioned at the same time instant in all the nodes. So, the time error of the BM positioning is the same for every node and it is shown in the fifth plot. In Fig. 6.6, the computed resetting value and the average resetting value, equal for every node, are shown. Then, the rebuilding behavior of the three slaves is similar to the one shown in the master-slave case, as in Fig. 6.7, Fig. 6.8, Fig. 6.9.

Assuming the trajectories as the position profiles on the x-y-z axes of a three-dimensional trajectory, the result is shown in Fig. 6.10 and the corresponding RMS error in Fig. 6.11. In particular, Fig. 6.11 shows that after an initial transient, the steady-state RMS error is in the order of 0.05%. At last, Fig. 6.12 shows that the buffer flow control is achieved in every slave.

6.5 Conclusions

In this chapter the possibility to develop a solution for master-multislave synchronization based on the approach proposed in the previous chapter has been discussed.

Simulations demonstrate that the introduction of a basic cycle, composed by a Broadcast Message and the single data transmissions to the single slaves, and the trajectory rebuilding based on the broadcast message can be a good approach to the problem. Moreover, the modifications required to the original algorithm are very slight.

Also in this topology time stamping and a-priori knowledge of the transmitting time instants is not required, since the messages can be identified by an identifier and the reconstruction is based on the Broadcast Message.

The aim of the preservation of the data relative temporal consistency property has been achieved.

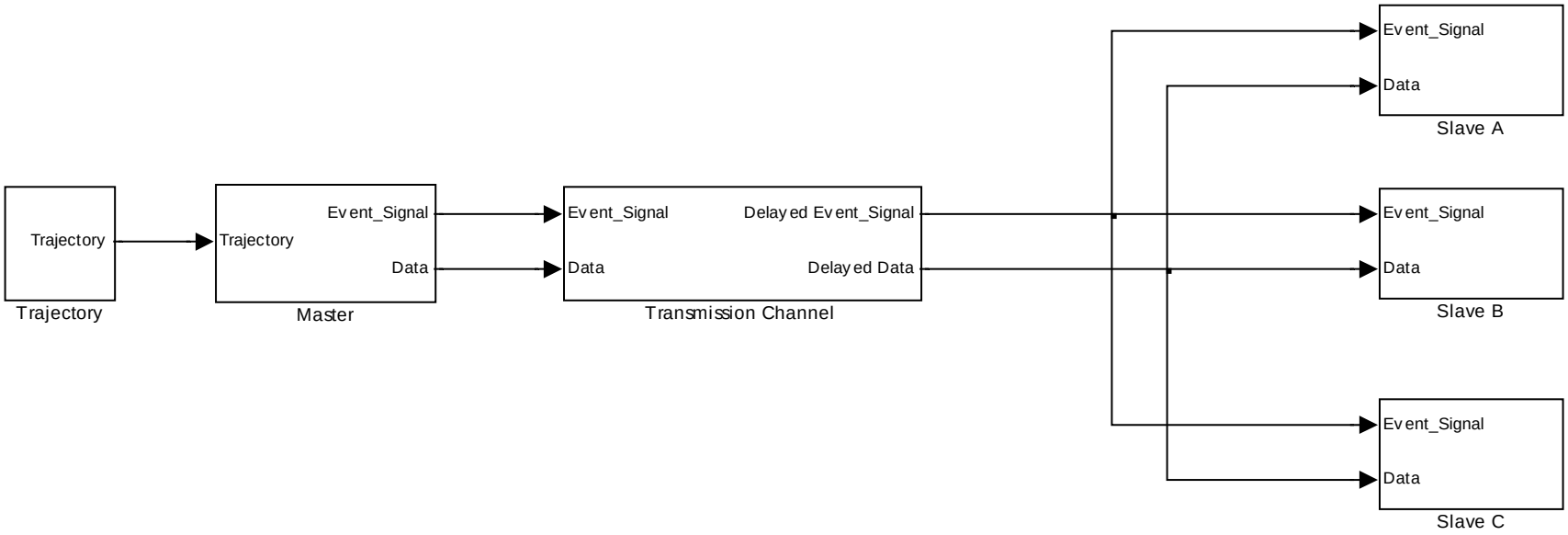

Fig. 6.2. Simulation scheme

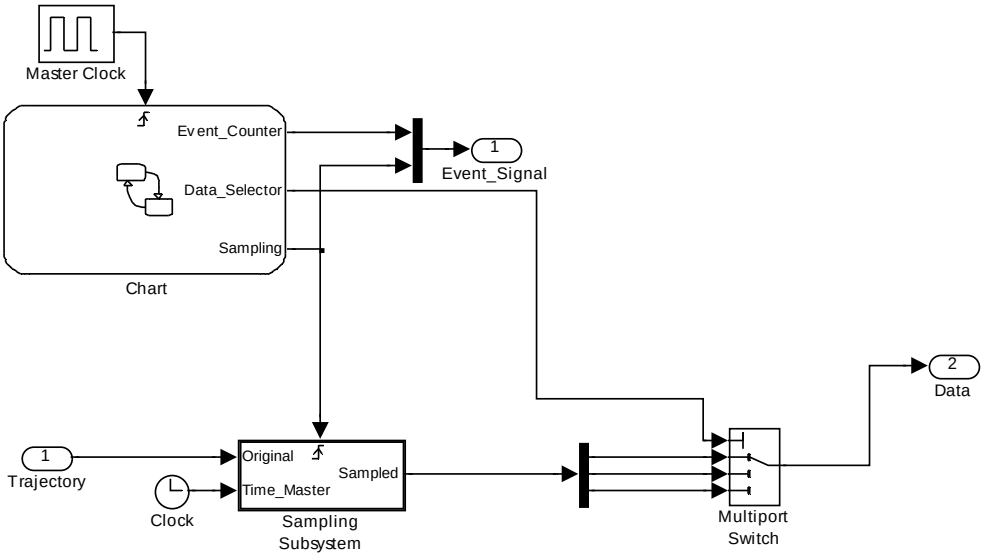

Fig. 6.3. Master node simulation scheme

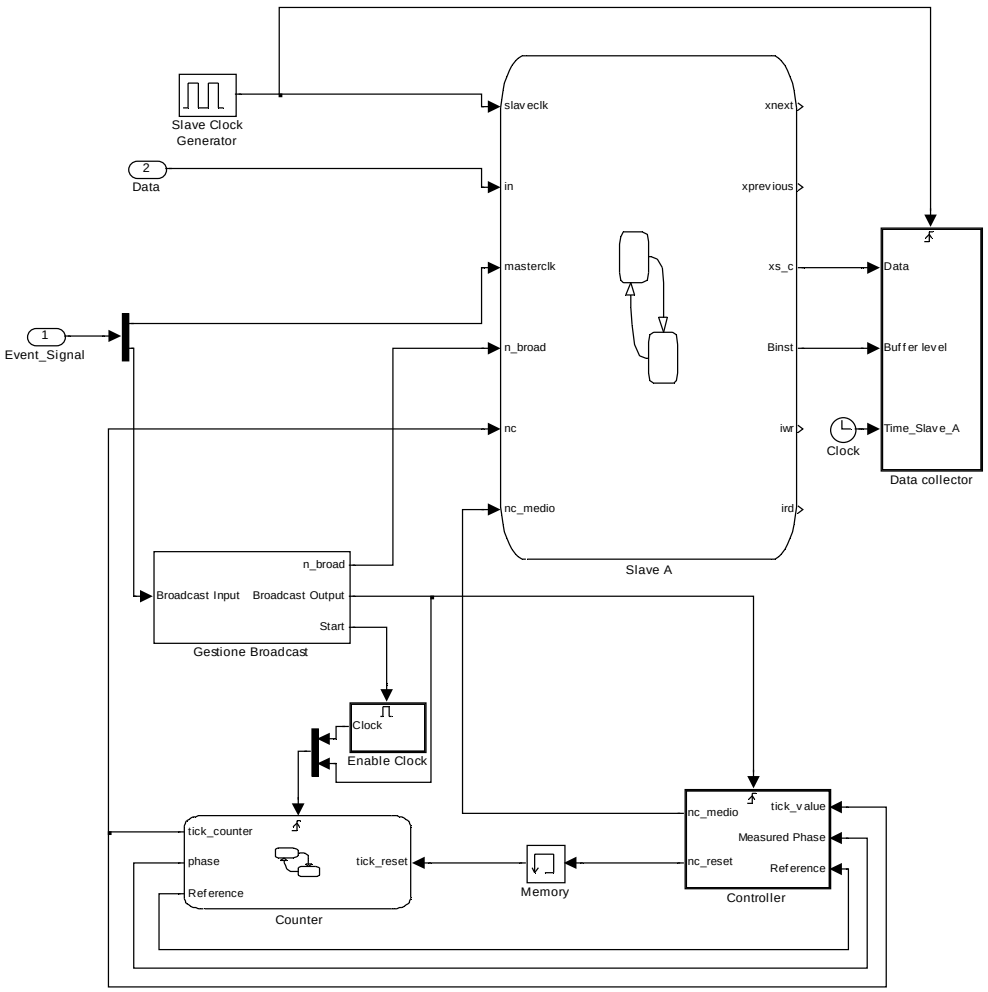

Fig. 6.4. Slave node simulation scheme

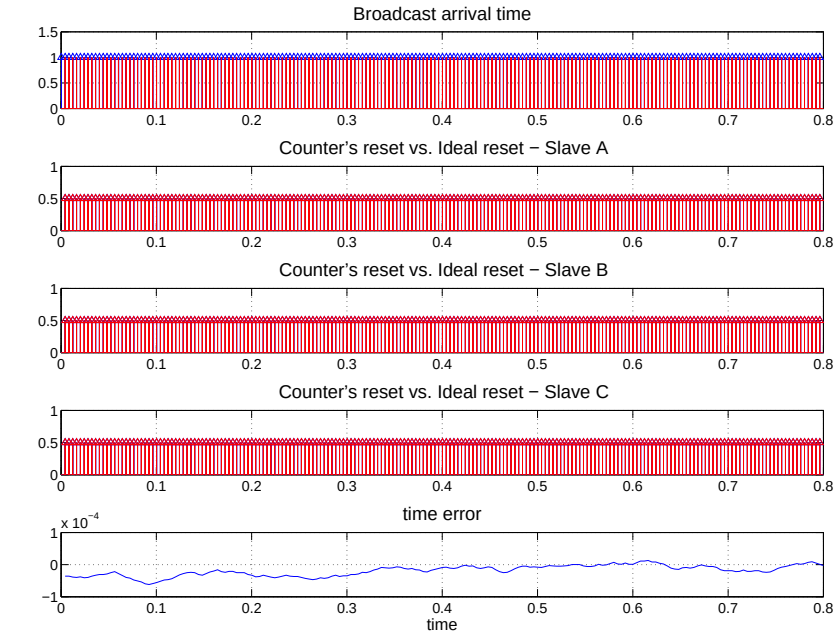

Fig. 6.5. Broadcast Message: ideal vs real arrival, ideal vs rebuilt data positioning in slave A, B, C and time error of data positioning

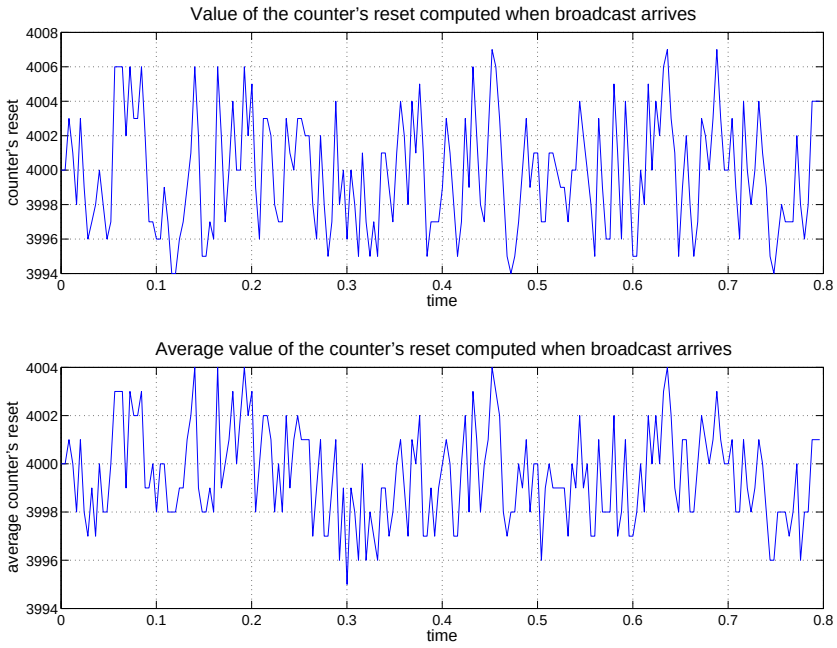

Fig. 6.6. Computed counter's reset and average counter's reset

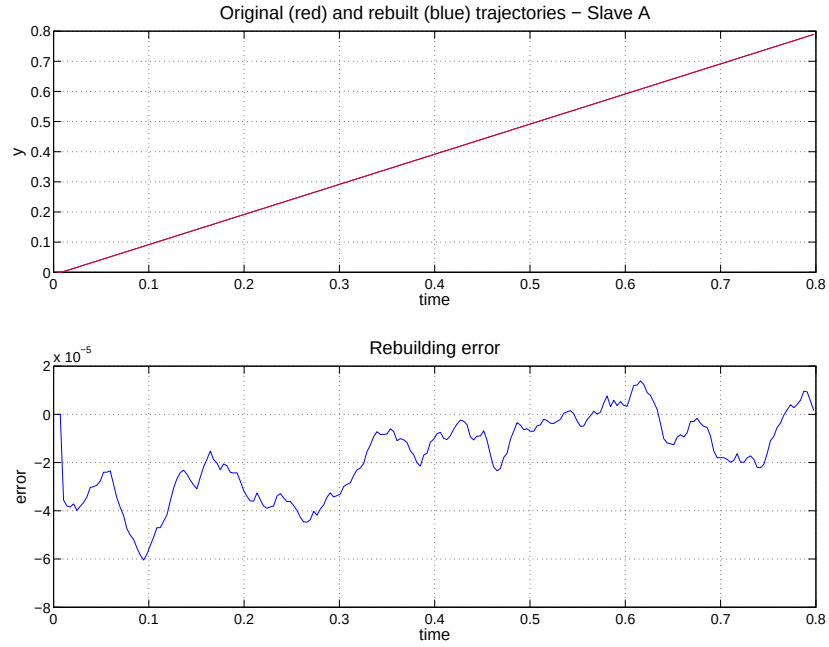

Fig. 6.7. Slave A: rebuilt ramp and rebuilding error

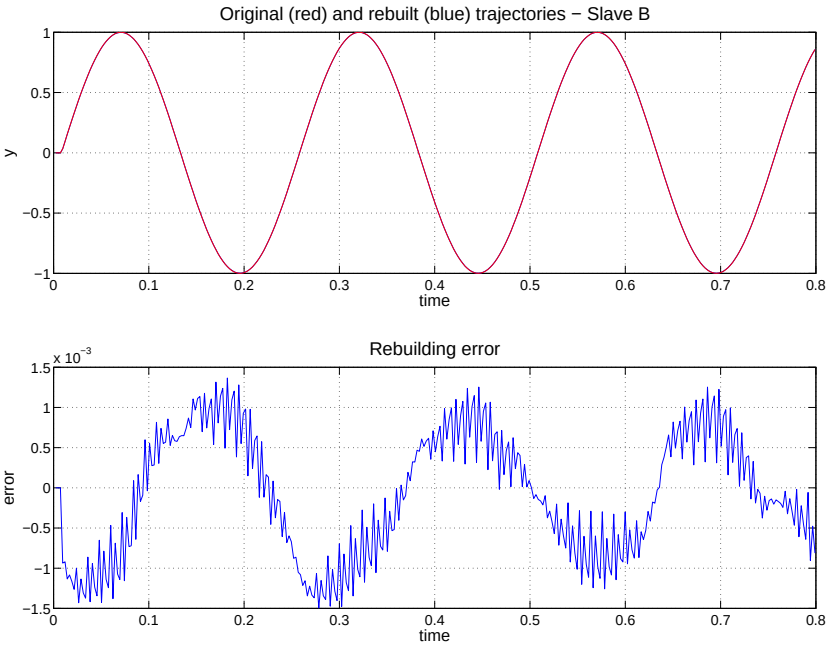

Fig. 6.8. Slave B: rebuilt sine and rebuilding error

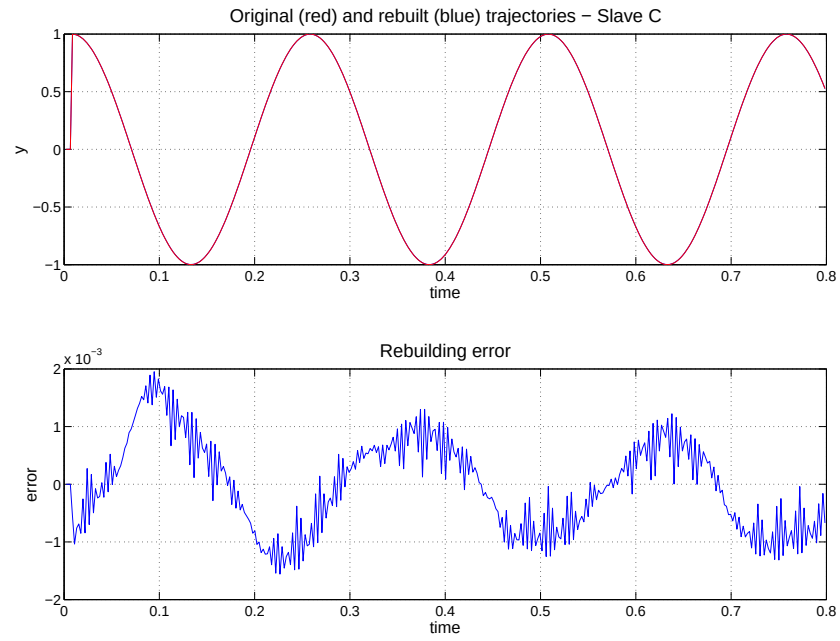

Fig. 6.9. Slave C: rebuilt cosine and rebuilding error

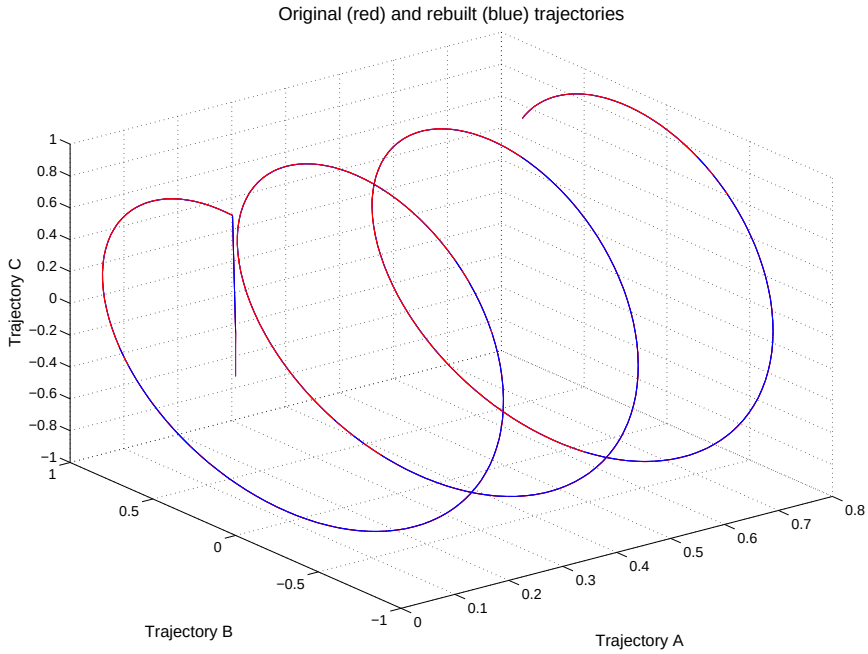

Fig. 6.10. Three-dimensional rebuilt trajectory: ideal vs real

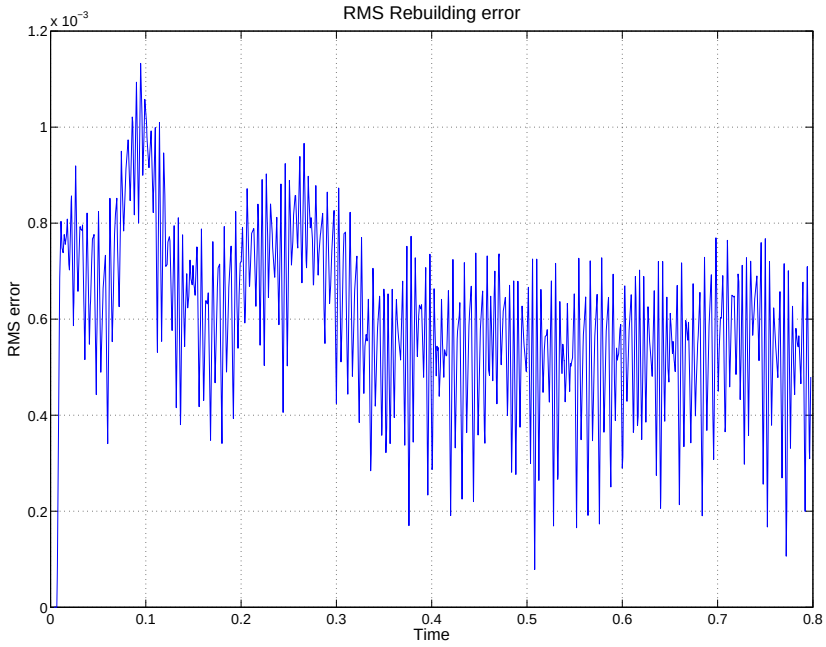

Fig. 6.11. Three-dimensional rebuilt trajectory: RMS error

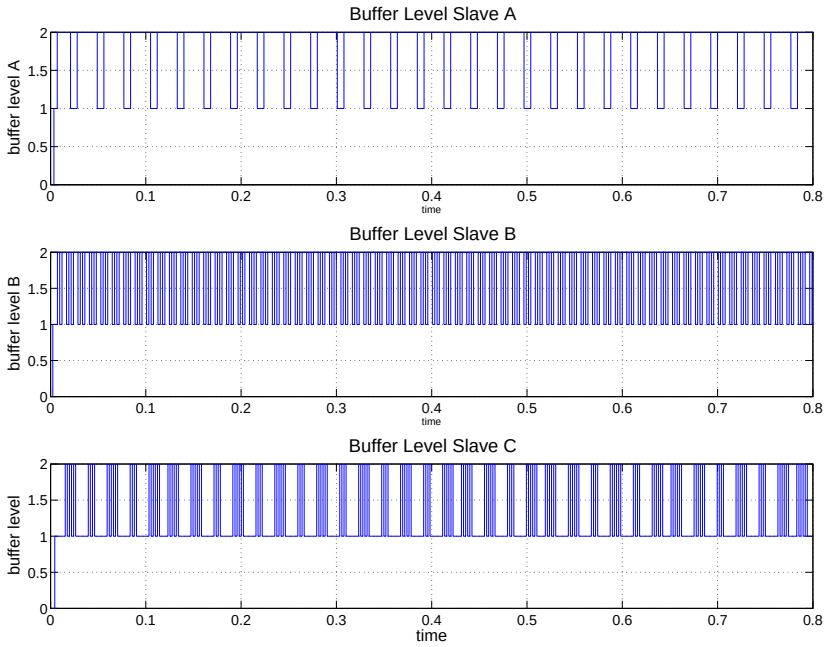

Fig. 6.12. Buffer flow: slave A, B, C

7

Multimaster-Slave

In this chapter the approach presented in chapter 5 has been improved to deal with the multimaster-slave topology in order to realize a gateway. The modifications to the original algorithm are presented accompanied by considerations on both pros and cons of the procedure. At last simulations show the results of the approach.

7.1 Introduction

In the previous chapters, the master-slave and master-multislave topologies have been introduced. The last configuration that must be exploited is the multimaster-slave one, that is the last brick required for the development of nets built on multiple layers.

In this structure, two ore more master nodes transmit their sampled trajectories to one single node: the gateway. A temporal relation exists among the trajectories. This relation is the relative temporal consistency property of data, like in the master-multislave case, since samples from different signals must be correlated in time. So, aim of the gateway node is the synchronization of different trajectories whose samples come from different nodes, the masters.

7.2 Procedure

In order to start with the description of the synchronization procedure, one hypothesis, explained by the following example, must be done.

For simplicity, assume that there are two masters, A and B, transmitting their trajectory to the gateway, as in the higher part of Fig. 7.1. Each node works at its own frequency, f_A and f_B. The figure shows that the A-trajectory first sample, $A, 1\_Data$, is sent by master A at the t_0 time instant, while the B-trajectory first sample, $B, 1\_Data$, is sent by master B at the t_1 time instant, that is after a time delay equal to Δ_{AB}. The fundamental hypothesis is that for the trajectory rebuilding at the gateway the first sample of both the trajectories must be placed at the same time instant, that is $A, 1\_Data$ and $B, 1\_Data$ are simultaneous, as shown in the lower part of Fig. 7.1 where $A, 1\_Data$ and $B, 1\_Data$ are both placed at t_{start} time instant. So, the first sampling time instant (t_0 and t_1) of both the trajectories does not matter for what concerns the trajectory rebuilding and each master can start the transmission at any time. Since a common time basis is not available, by means of which a measure of the Δ_{AB} temporal distance would be possible, this assumption is very important because otherwise an a-priori temporal relation can not

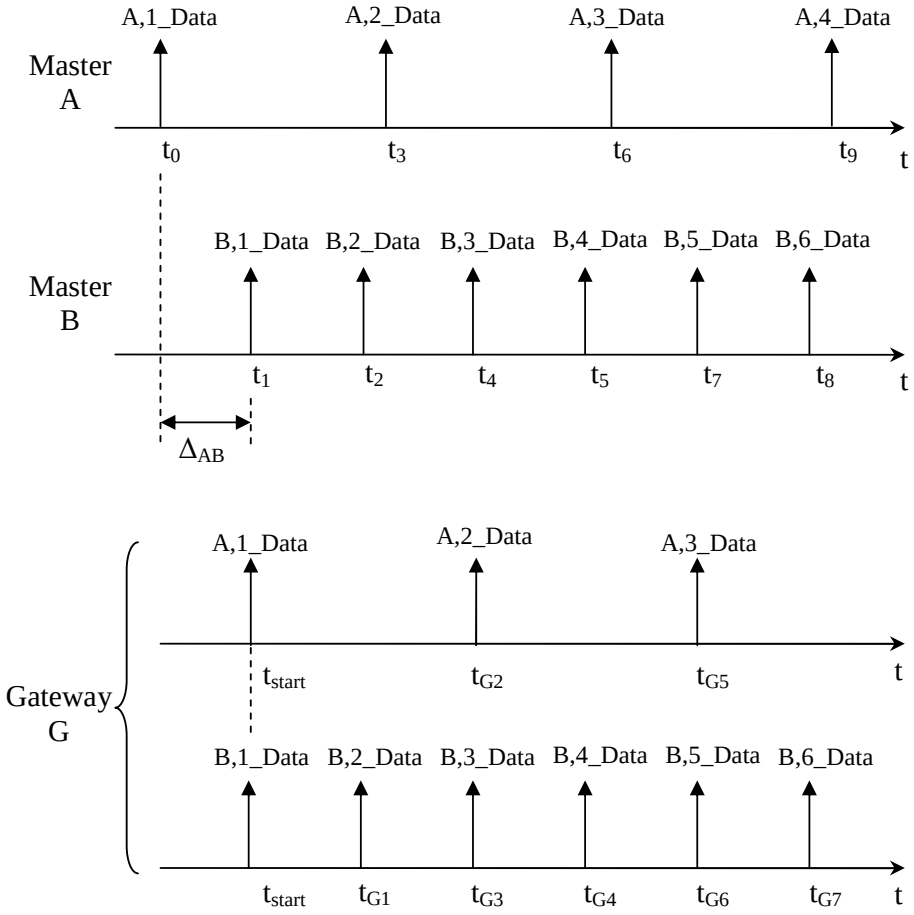

Fig. 7.1. Required hypothesis for trajectory synchronization

be defined among the trajectories and the information for the reconstruction are not sufficient. However the assumption is plausible since different trajectories may be planned during the phase design to start from a common time instant and the Δ_{AB} entity may be assumed as a delay in the transmission.

Now, given the above assumption, the rebuilding procedure will be analyzed.

As presented in the master-slave case, the slave node is able to start with the reconstruction when two samples are available. The same procedure is adopted in the gateway: the data positioning of the single trajectory starts when two samples of this trajectory are available. In particular, the gateway will start the data positioning of each trajectory independently from the other trajectories.

An alternative would be to wait for the time instant in which the second datum of every trajectory is arrived then start with the rebuilding of all the trajectories, but in this way all the information related to the temporal distance, measured by the counter, among the first data of a single trajectory would be lost.

Consequently, by means of the proposed approach, the data positioning of all the trajectories along the time-line is already started when the last second datum arrives. This consideration is shown in the upper part of Fig. 7.2. Suppose to be in ideal condition, that is without jitter and drift. At the gateway, the $t_{A,1}, t_{A,2}, t_{B,1}, t_{B,2}, t_{B,3}$ time instants are the arrival times of the data generating the corresponding $(A, 1)$, $(A, 2)$, $(B, 1)$, $(B, 2)$, $(B, 3)$ events. Then, by means of the clock recovery algorithm, the arrival event is ideally

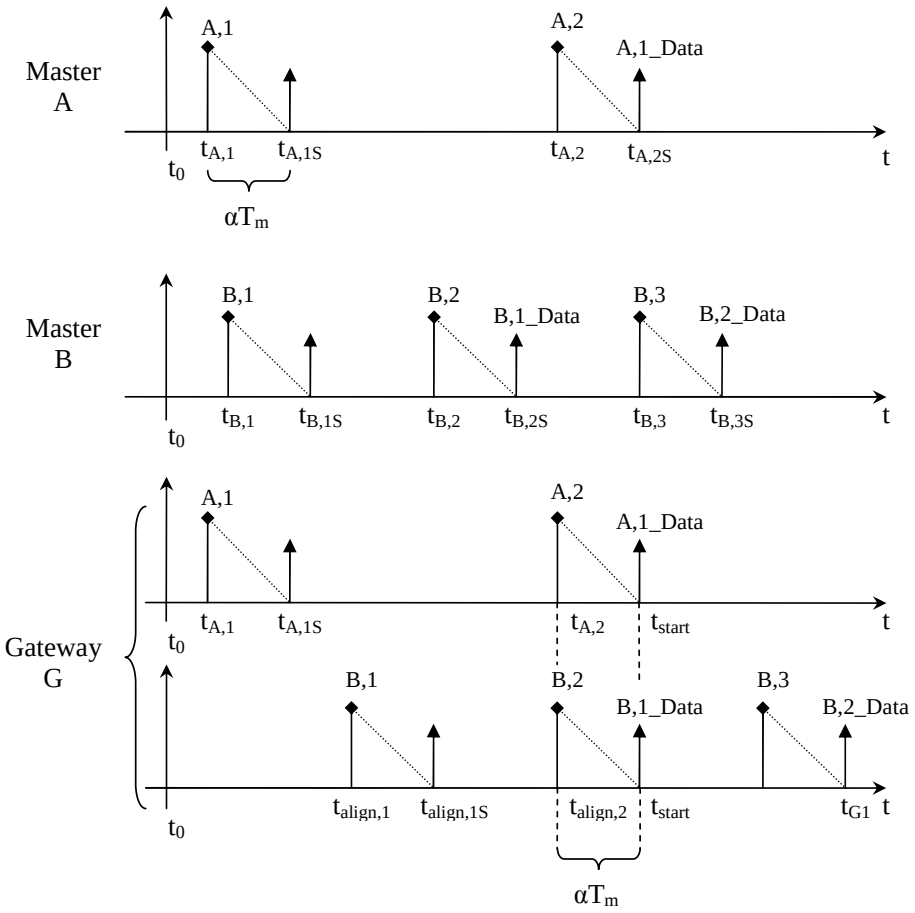

Fig. 7.2. Alignment in ideal conditions

phase-shifted of an amount of time equal to αT_m. Now, this value must be greater than the maximum jitter among the single maximum jitters, that is

$$\alpha T_m \geqslant jitter_{MAX} = max\{jitter_{max,i}\} \qquad i = 1, ..., n \tag{7.1}$$

and equal for every trajectory. So the $(A, 1)$ event is placed at $t_{A,1S}$, $(A, 2)$ at $t_{A,2S}$ and so on. Similarly, since the reconstruction of one trajectory is independent from the other trajectories, the same procedure is applied to the events generated by the B_Data arrivals: $(B, 1)$ event is placed at $t_{B,1S}$, $(B, 2)$ at $t_{B,2S}$ and so on.

Every data arrival is associated to every arrival event, but a distinction between event and transferred data must be done. Even if $(A, 1)$ event declares the $A, 1\_Data$ arrival at the $t_{A,1}$ time instant, data will not be placed at $t_{A,1S}$ but at $t_{A,2S}$ since the reconstruction starts when two samples are available.

Now, the data positioning of every trajectory is started, but there is no synchronization among them. So, in order to obtain the desired alignment, that is the synchronization, these trajectories must be time-shifted. Since data positioning is based on the master clock recovery, the discovery of a common starting point for the reconstruction is desirable and a suggestion is given by the lower part of Fig. 7.2. In ideal condition, that is without jitter or drift, the alignment of the second datum arrival time instants of all the trajectories must be achieved. This means that there will be a time instant in which the reconstruction of all the trajectories but one, the last one, is started. When the second datum of the

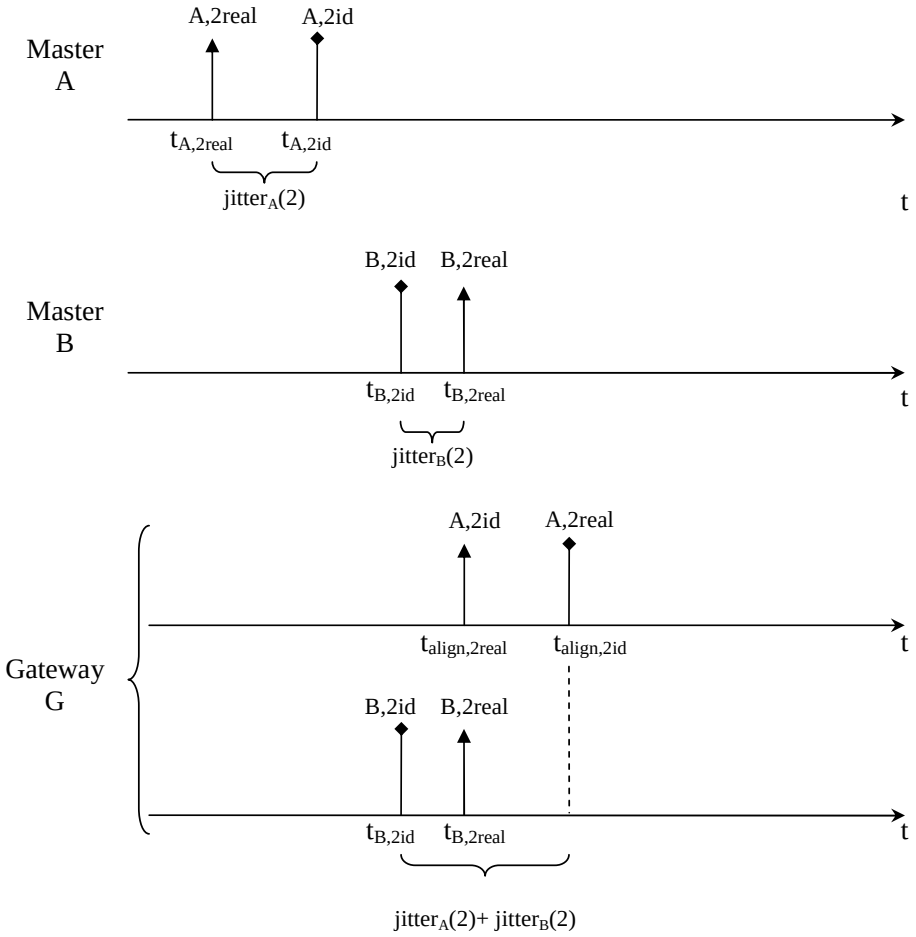

Fig. 7.3. Alignment in non-ideal conditions

last trajectory arrives, all the other trajectories must be time-shifted in such a way to reproduce the already rebuilt master clocks starting from their own second datum arrival time instant. Looking at Fig. 7.2, the last trajectory to rebuild is the A-trajectory and consequently the last second datum arrival time instant is $t_{A,2}$. So the B-trajectory can be synchronized with the A-trajectory aligning its second datum arrival time instant $t_{B,2}$ to $t_{A,2}$ and the synchronized rebuilding of all the trajectories will start from the t_{start} time instant, thanks to αT_m equal for every trajectory.

However, this approach has a drawback when jitter is introduced, that is in non-ideal conditions. The explanation of the problem is visible in Fig. 7.3. Suppose that the A-trajectory second event is generated at the $t_{A,2real}$ time instant instead of $t_{A,2id}$ time instant because it is subject to $jitter_A(2)$. Similarly, the B-trajectory second event is generated at the $t_{B,2real}$ time instant instead of $t_{B,2id}$ time instant because it is subject to $jitter_B(2)$. When the alignment is computed, the single recovered clock periods, neglecting the phase-shift αT_m, will be aligned respectively to $t_{align,2id}$ for the A-trajectory and to $t_{B,2id}$ for the B-trajectory. Consequently, there will be an offset in the trajectory synchronization due to the sum of the single jitters, $jitter_A(2) + jitter_B(2)$. Obviously, this offset also reflects on the A-trajectory rebuilding.

The procedure could be improved trying to estimate the second data jitter values and using them to adjust the αT_m value of each trajectory. In this way, every trajectory will

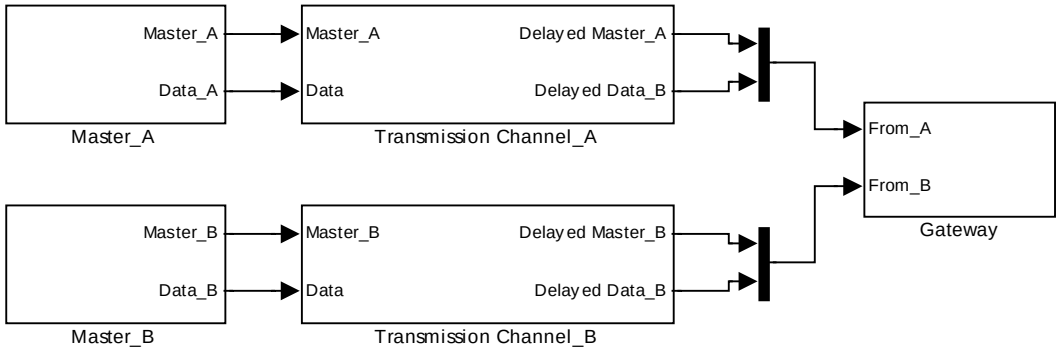

Fig. 7.4. Simulation scheme

not have the same αT_m value, but a custom-made value. Clearly, the lower bound due to the single trajectory $jitter_{max}$ must be satisfied anyway.

At last, the trajectory rebuilding based on the gateway period by means of the interpolation algorithm can start as usually.

7.3 Simulations

The simulations have been performed considering a system composed by two masters transmitting one trajectory each one to the gateway.

The adopted system initialization is shown in Table 7.1. The simulations are performed in nominal conditions. Because of such initialization, the first rebuilt trajectory is A, that will be also the trajectory to shift in time.

Parameter	Value
trajectory $x_A(t)$	t
trajectory $x_B(t)$	$sin(8\pi t)$
nominal T_A	2 msec
nominal T_B	1.5 msec
nominal T_{gate}	1.8 msec
nominal T_{clk}	1 μsec
Δ_{AB}	3.625 msec
master A transmission delay	0.5 msec
master B transmission delay	0.375 msec
$jitter_{max,A}$	0.18 msec
$jitter_{max,B}$	0.1 msec
a	0.96907
$gain$	0.032334

Table 7.1. Simulation parameters

Fig. 7.4 shows the simulation scheme: two masters transmit their trajectories to one gateway. The gateway is shown in Fig. 7.5. It is composed by a section that regenerates the master clock periods, the Control block in the Figure, and a section related to the trajectory rebuilding, the Gateway block in the Figure. The Control block, shown in Fig. 7.6 detects the data arrivals and starts with the single master clock recovery then, when the last second datum arrives, the Data alignment block provide for the alignment of the recovered clocks. The Gateway block is similar to the slave node, but modularity has been

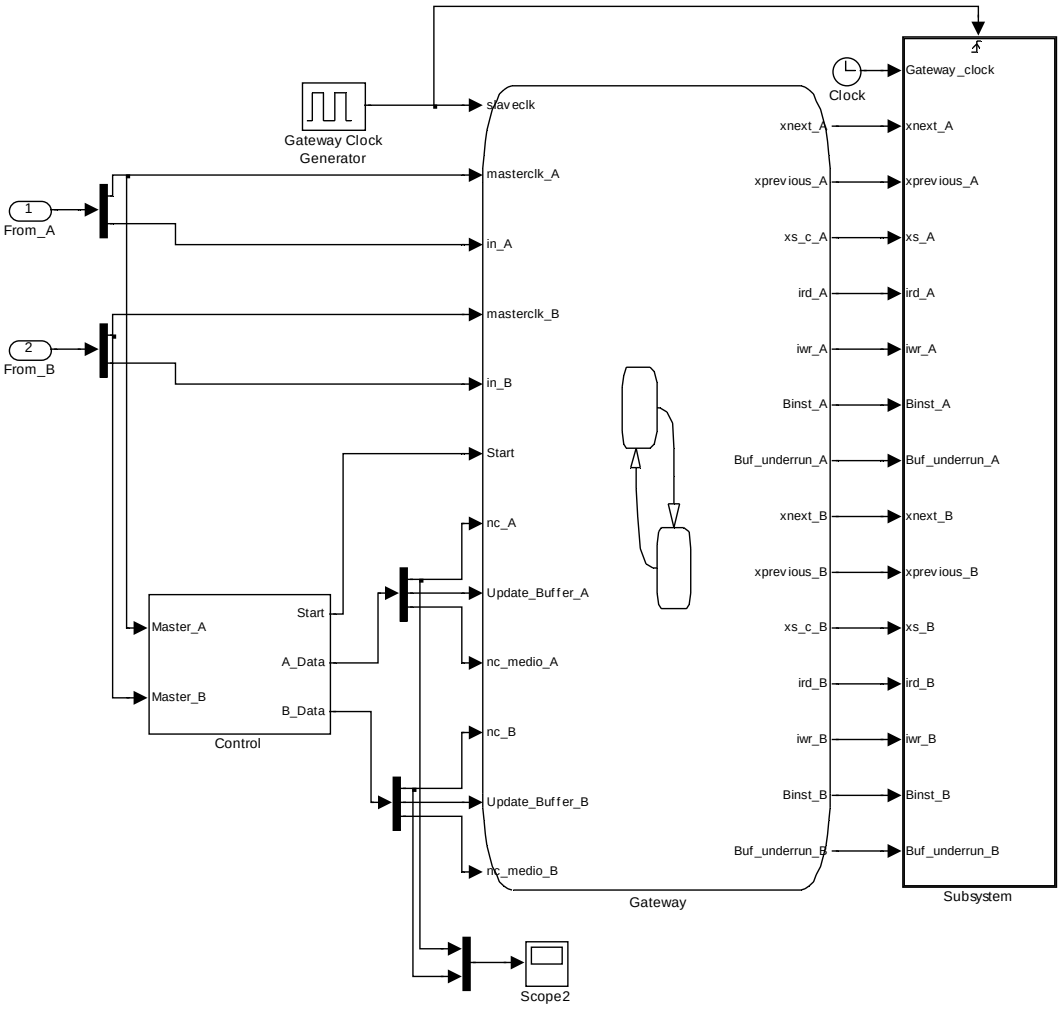

Fig. 7.5. Gateway simulation scheme

improved. As shown in Fig. 7.7, there is always the distinction between the functions assigned to the handling of arrived data (Gestione_arrivo_dati block), that is divided into two blocks in order to manage data from both master A and master B (Fig. 7.8), and to the rebuilding trajectory (Gestione_generazione_dati block), both for master A and master B (Fig. 7.9), but this second block has been divided into two other blocks (Fig. 7.10) to be able to separately handle the updating process in the buffer (Fig. 7.11) and the real trajectory rebuilding (Fig. 7.12).

Fig. 7.13 and Fig. 7.14 show the ideal and real data positioning in the second plot and the corresponding time error in the third one, respectively for trajectory A and B, without taking into account the desired time shift: the positioning works as usually.

Fig. 7.15 shows the effects of the alignment. In the first plot the B-trajectory data positioning is presented. Since there is no need to time-shift this trajectory, the positioning is the same of Fig. 7.14. In the second plot, the effective (blue) and desired (red) A-trajectory data positioning are shown. The effective positioning is obtained by means of the previously described procedure, that is time-shifting the clock recovery when the second datum of the B-trajectory arrives. Obviously, the generic trend is the same of the third plot in the previous figure because the ideal and real time-shifted behaviors are

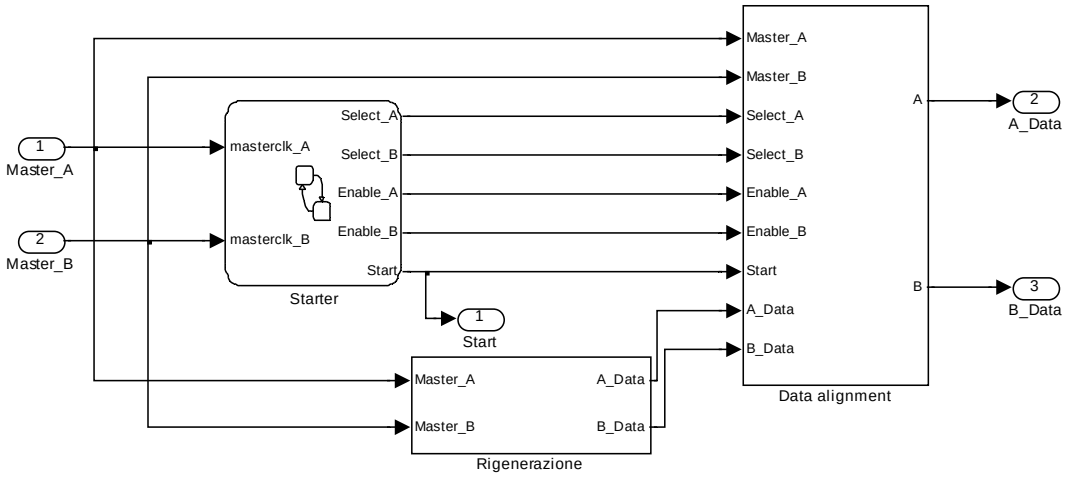

Fig. 7.6. Gateway simulation scheme: Control block

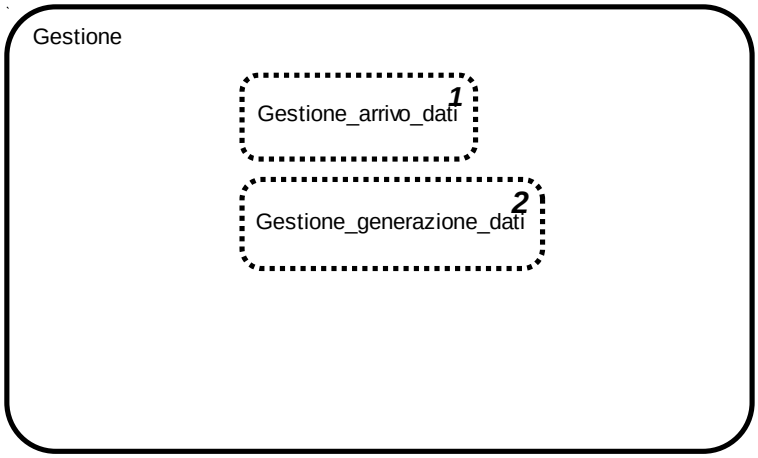

Fig. 7.7. Gateway simulation scheme: Gateway block

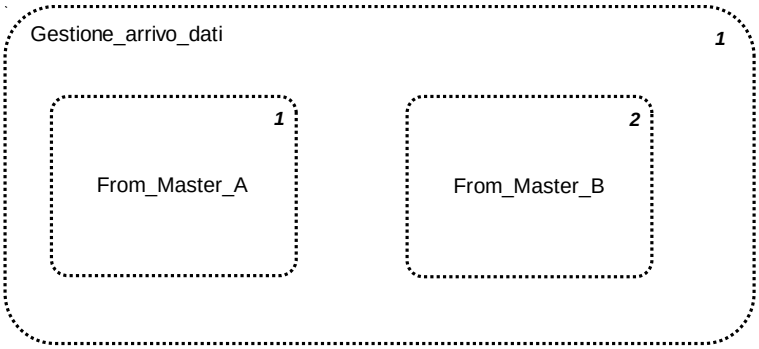

Fig. 7.8. Gateway simulation scheme: Gestione_arrivo_dati block

the same ideal and real not-time-shifted behaviors plus a constant time-shifting. What is interesting is that the first ones are not subject to the same constant time-shifting. This difference generates the synchronization offset shown in the third plot of Fig. 7.15. This offset is due to the jitter, as explained in the previous section, and has effect also on the rebuilt A-trajectory, as shown in Fig. 7.16, while it has no effect on the B-trajectory rebuilding (Fig. 7.17).

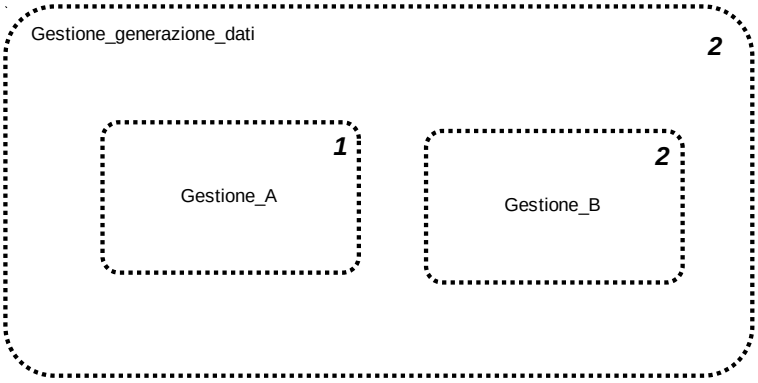

Fig. 7.9. Gateway simulation scheme: Gestione_generazione_dati block

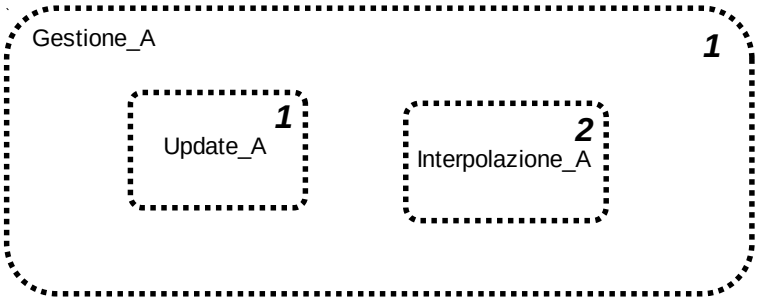

Fig. 7.10. Gateway simulation scheme: Gestione_A block

At last, Fig. 7.18 shows the progress in the buffer levels. Since A-trajectory clock recovery starts before then the B-trajectory clock recovery, more samples arrive from master A than master B, so they must be store until their use. After that, the buffer level is under control again.

7.4 Conclusion

In this chapter, how to realize a gateway able to synchronize different trajectories, each one transmitted by different masters, in order to preserve the relative temporal consistency property of the profiles has been shown. Adopting the presented approach, the most significant problem for the trajectory synchronization results to be the jitter on the second data transfer, as confirmed by both considerations and simulations. The offset generated by the jitter during the reconstruction can be eventually reduced if other information are introduced in the procedure, for example an estimate of the jitter amount on the second data. So, at the moment, the aim of preserving the relative temporal property in the multimaster-slave case is partially reached.

This topology, with the master-slave and master-multislave procedures, concludes the work on the trajectory rebuilding in non-deterministic network. These are the basic configurations with which more complex network on different layers can be built. By means of the proposed approach, the temporal properties of a trajectory can be preserved after their digitalization, transmission and reconstruction. In the future work, the simulations based on the composition of the three topologies should be developed and the problem

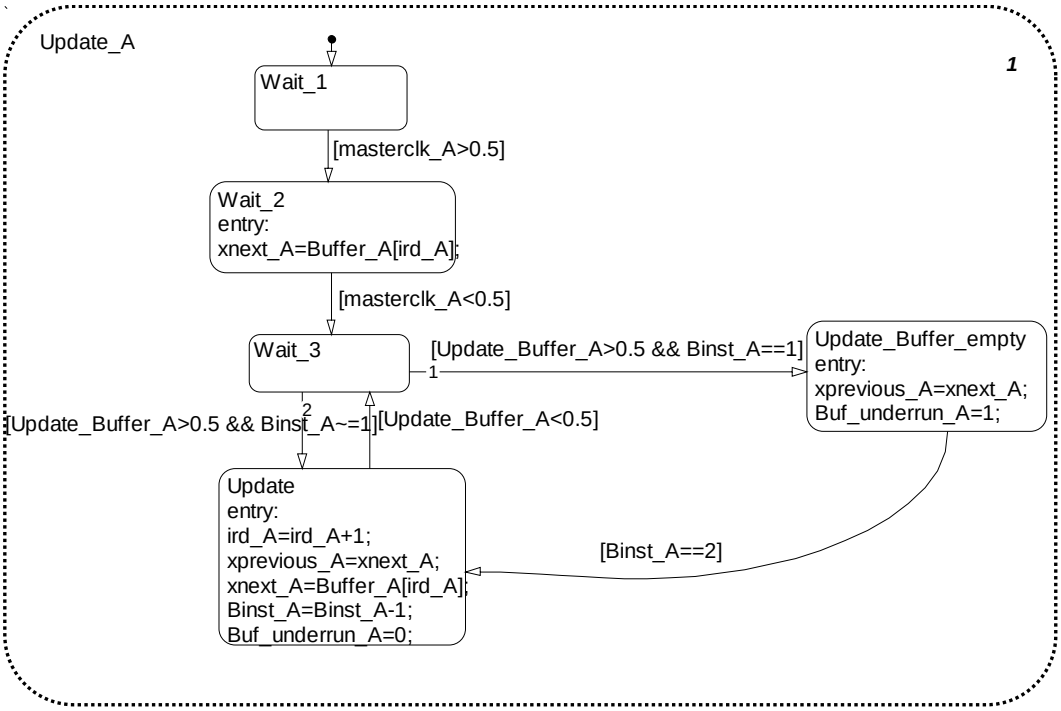

Fig. 7.11. Gateway simulation scheme: Update_A block

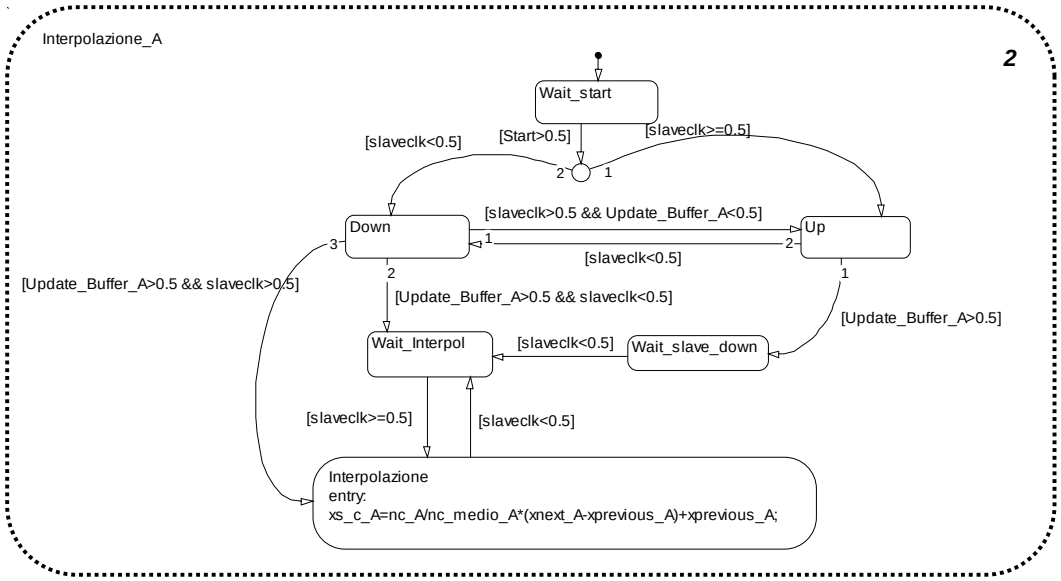

Fig. 7.12. Gateway simulation scheme: Interpolazione_A

of the jitter in the multimaster-slave case should be face in order to improve the current solution.

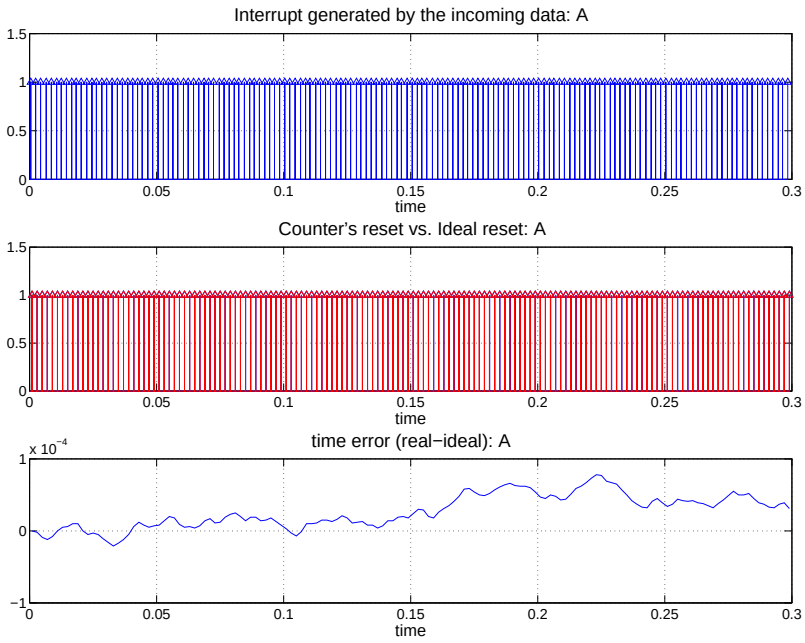

Fig. 7.13. Ideal vs rebuilt data positioning of A-trajectory and corresponding time error

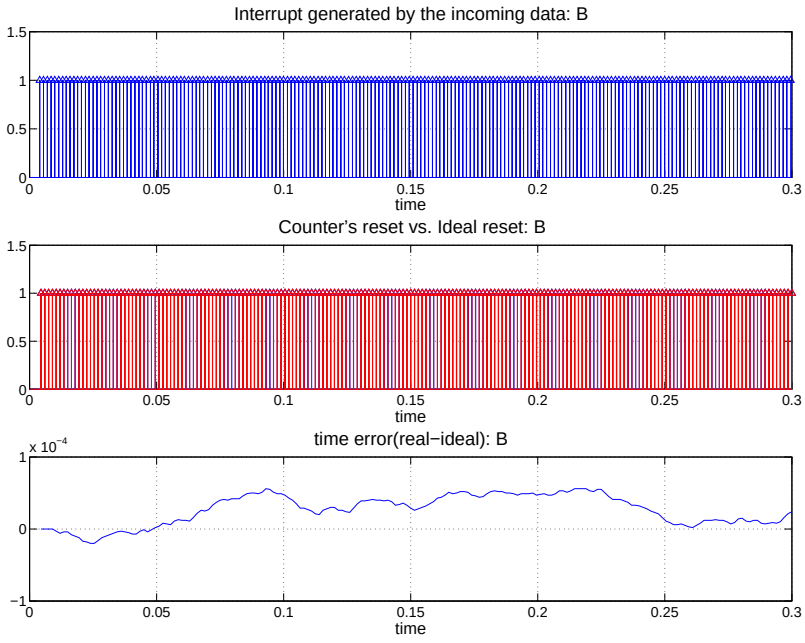

Fig. 7.14. Ideal vs rebuilt data positioning of B-trajectory and corresponding time error

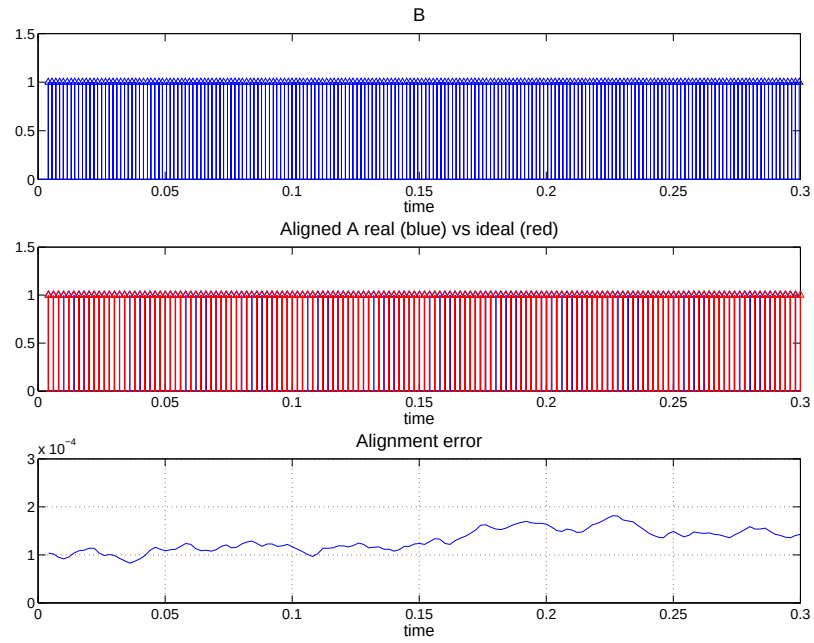

Fig. 7.15. Ideal vs rebuilt time-shifted data positioning of A-trajectory and corresponding time error

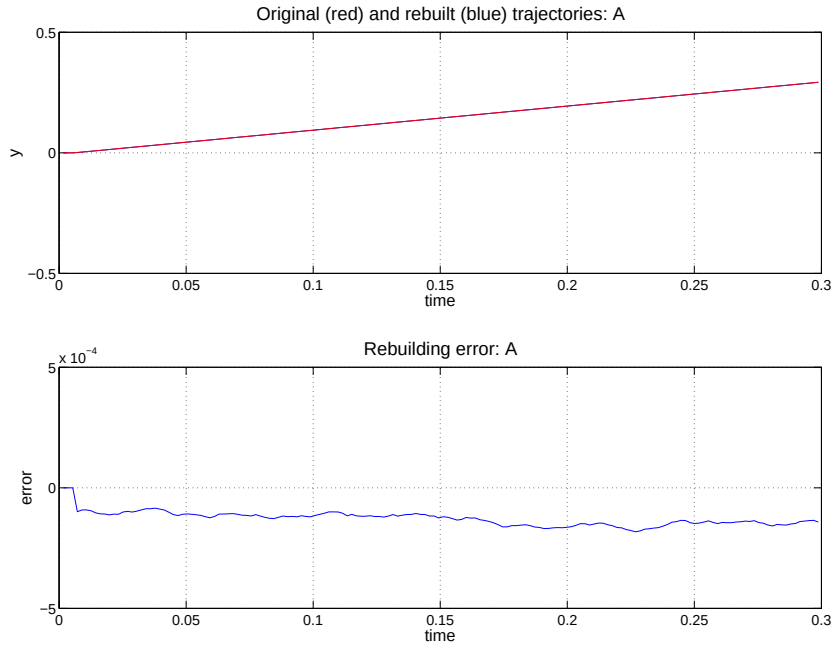

Fig. 7.16. A-trajectory rebuilding and rebuilding error

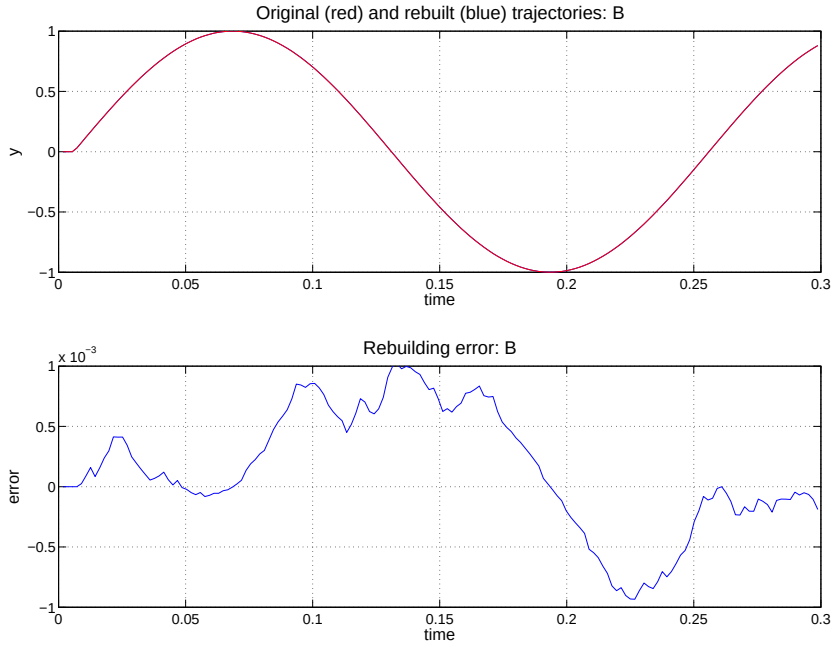

Fig. 7.17. B-trajectory rebuilding and rebuilding error

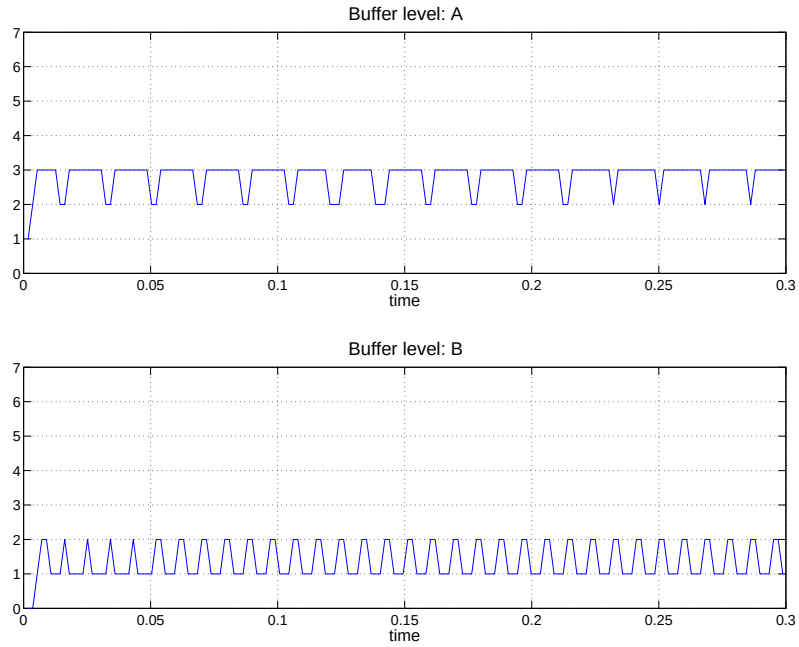

Fig. 7.18. Buffer levels

A

Fieldbuses

A.1 Introduction

Fieldbus [42] is the word widely used to indicate a network for connecting field devices, such as sensors, actuators, field controllers such as PLCs, regulators, drive controllers, etc., and man-machine interfaces.

Fieldbus technology involves a variety of solutions and techniques, developed in more than 20 years to face with similar problems, which are different from each other.

Initially, there was no existing standard so each information technology provider developed their own solutions in a given sector. Since a standardized system gains a competitive edge over its non standardized rivals, a race for the standardization started. Several projects started in Europe: the FIP project in France (1982), subsequently extended and called WorldFIP; the P-Net in Denmark (1983); the PROFIBUS in Germany (1984); the CAN, by Bosch Company, always in Germany (1983).

Even if the standardization on a national level was quite easy, the problems came out when the international standardization was sought. Since the standardization took more than a decade and did not produce a real universal fieldbus, and considering that fieldbus systems had already made their way into the market with enormous amount of money invested in the development of protocols and devices, the standardization process turned from a technical to a political and economical question. Within CENELEC, the national committees found after lengthy discussions a remarkable and unprecedented compromise: all national standards under consideration were simply compiled ”as is” to European standards. Every part of such a multi-part standard is a copy of the respective national standard, which means that every part is a fully functioning system.

In the meantime, several mainly American companies began the definition of a new fieldbus optimized for the process industry: the Foundation Fieldbus. So, the situation had further complicated and new conflicts started in the IEC committee.

Again, the resolution was to create a large and comprehensive IEC standard accommodating all fieldbus systems. However, other than CENELEC, where complete specification had been copied into the standard, the IEC decided to retain the original layer structure of the draft with physical, data link and application layer, each separated into a services and protocols part. The individual fieldbus system specifications had to be adapted to so-called ”types” to fit into this modular structure.

The relation among the standards is presented in Table A.1 [43].

CENELEC standards part	Contained in IEC standard	Brand name
EN50170-1 (Jul. 1996)	IS 61158 Type 4	P-Net
EN50170-2 (Jul. 1996)	IS 61158 Type 1/3/10	PROFIBUS
EN50170-3 (Jul. 1996)	IS 61158 Type 1/7	WorldFIP
EN50170-A1 (Apr. 2000)	IS 61158 Type 1/9	Foundation Fieldbus
EN50170-A2 (Apr. 2000)	IS 61158 Type 1/3	PROFIBUS-PA
EN50170-A3 (Apr. 2000)	IS 61158 Type 2	ControlNet
EN50254-2 (Oct. 1998)	IS 61158 Type 8	INTERBUS
EN50254-3 (Oct. 1998)	(IS 61158 Type 3)	PROFIBUS-DP (Monomaster)
EN50254-4 (Oct. 1998)	(IS 61158 Type 7)	WorldFIP (FIPIO)
EN50325-2 (Jan. 2000)	IS 62026-3 (2000)	DeviceNet
EN50325-3 (Apr. 2000)	IS 62026-5 (2000)	SDS
EN50325-4 (under vote)		CANOpen
EN50295-2 (Dec. 1998)	IS 62026-2 (2000)	AS-Interface

Table A.1. Contents of the CENELEC fieldbus standards

A.2 CAN 2.0

The Controller Area Network (CAN) [44] is a serial communications protocol which supports distributed real time control.

In CAN network, data are transmitted and received using fixed format messages of different but limited length. Transmitted data do not contain addresses of either the source or destination of the message since a CAN node does not make use of any information about the system. Instead, the content of a message is named by an IDENTIFIER which describes the meaning of the data, so that all nodes in the network are able to decide by Message Filtering whether the data is to be acted upon by them or not. As a consequence of the concept of Message Filtering any number of nodes can receive and simultaneously act upon the same message. This mode of operation is known as multicast.

The message IDENTIFIER has also another function: it defines a static message priority during bus access. In fact, CAN is based on a Carrier Sense Multiple Access with Collision Avoidance protocol (CSMA-CA) that avoid the occurrence of collisions adopting bit arbitration. The arbitration logic assumes that a recessive and a dominant state on the communication channel exist such that the dominant state can overwrite the recessive state. Assume that a ’0’ is coded into the dominant state and a ’1’ is coded into the recessive state. Whenever a node wants to send a message, it put the first bit of the message identifier on the channel. Since every node can start a data transmission at any moment, two or more stations could start the transmission at the same time generating a conflict for the bus channel. In this case, the node with a ’0’ in its first identifier bit wins, and the one with ’1’ must back off. This arbitration continues for all bits of the identifier. The mechanism of arbitration guarantees that neither information nor time is lost.

A.3 TTCAN

TTCAN [45] is based on a time triggered and periodic communication which is clocked by a time master’s reference message. The reference message can be easily recognized by its identifier. Within TTCAN’s level 1 the reference message only holds some control information of one byte, the rest of a CAN message can be used for data transfer. In

extension level 2, the reference message holds additional control information, e.g. the global time information of the current TTCAN time master.

The period between two consecutive reference messages is called the basic cycle. A basic cycle consists of several time windows of different size and offers the necessary space for the messages to be transmitted. The time windows of a basic cycle can be used for periodic state messages and for spontaneous state and event messages. The TTCAN specification allows to use more than one basic cycle to build the communication matrix or system matrix of the systems engineer's needs. Several basic cycles are connected to build the matrix cycle.

Within a basic cycle of TTCAN, the protocol execution is driven by the progression of time. This time is the so called cycle time of TTCAN and is restarted after the reception of every reference message. The necessary link between the cycle time and the system matrix are the so called time marks.

The cycle time of TTCAN is the basic time to guarantee the time triggered operation of the protocol. An important property of such a time information is the granularity of the time. The granularity of any timing information within TTCAN is the network time unit (NTU). So the cycle time is measured in NTU and is based on the nominal CAN bit time in TTCAN level 1 and on the physical second in TTCAN level 2. In level 2, to establish a system wide NTU, the node local relation between the physical oscillator of a TTCAN controller and the system wide NTU has to be established. The node dependent oscillator circuit provides the system clock to a frequency divider. This frequency divider generates the system wide NTU while a node local time unit ratio (TUR) takes care for the correct relation between the system clock and NTU. NTU now can be used to build a local time and to build the global time.

In TTCAN level 2 all nodes take a snapshot of their time values at the frame synchronization pulse. The time master sends its (by definition correct) global time value for this frame synchronization pulse as part of the reference message. After reception each node can build its local offset as the difference between the master global time snapshot value and the own local time snapshot value. During the next basic cycle the node can compute the global time by global time = local time + local offset. If local time and global time have the same speed this ensures that all nodes have a consistent view on the global time. Due to slightly different clock drifts of the different nodes a mechanism has to be introduced to guarantee that local and global time have in fact the same speed. This mechanism is the continuous update of TUR. An initial value of TUR is a priori known node locally by the oscillator specification. During operation, to adapt this value to the correct value determined by the master clock speed the node measures the length between two successive frame synchronization pulses both locally (number of oscillator periods in this interval) and in global time (difference between the two master snapshot values). The quotient of these two values gives the actual TUR (limited only by the precision of the measurement). The achievable precision determines a reasonable choice of the NTU-value in physical seconds. In level 2 the global time values of two nodes will then not differ by more than one NTU. The NTU typically will be in the order of a CAN bit-time.

A.4 WorldFIP

WorldFIP (World Fieldbus Instrumentation Protocol) [46] is a protocol based on the TDMA mechanism for the access to the shared bus in the MAC layer. Consequently it works better with periodic data transmission. As for other fieldbuses, the WorldFIP architecture is made up of three layers: the physical layer, the Data Link Layer and the Application Layer.

The problem for the bus access has been solved assuming that one station will be the Bus Arbitrator (BA). The mechanism is the following:

- the BA has a table containing all the variables of interest and their periodicity to realize a correct scheduling for the transmission cycles;
- the variables are identified by identifiers;
- the BA transmits an identifier on the bus;
- all the stations receives the message, but only one identifies itself as the producer of the variable while one or more stations recognize themselves as consumers of the variable;
- the producer transmits the value of the variable on the bus;
- only the consumers receive the value transmitted by the producer while the other stations discard the message;
- the bus arbitrator looks at the table for the next identifier and the cycle begins again.

Since the elementary cycles may not be saturated by the variable transmissions, free time could be available for aperiodic data transmission.

A.5 PROFIBUS

Different profiles exist of the Profibus standard [46]. The best known Profibus version are Profibus-DP for factory automation, Profibus-PA for process automation, motion control with Profibus for drive technology and PROFIsafe for safety-relevant applications. The features of the Profibus-DP will be now described.

Profibus-DP adopts a master-slave communication typology, in which the station are divided into active station (Masters) and passive stations (Slaves). The masters have the bus access control. The slaves can access to the bus only for answers to requests. All the slaves are identified by a unique address assigned during the initialization.

The access protocol to the medium is based on the token passing. Only the master that holds the token can transmit data on the net. The token is passed to the next master since at the initialization a logic ring has been defined. In fact, each station knows each own address and the addresses of the next and the previous stations. The message transmission is divided into cycles. Each message cycle is composed by an action frame of the master and the associated acknowledge or response frame. The token is passed to the next station following the ascending order of the address (with the exception of the last station). When the next station receives the token from its previous one, it starts its cycle of transmission.

A.6 P-NET

P-NET [47] is a multi-master bus, which can accept up to 32 masters per bus segment. All communication is based on the principle, where a Master sends a request, and the addressed Slave returns an immediate response. Requests can be of a read or write type.

The right to access the bus, is transferred from one P-NET master to another, by means of a token. P-NET uses a method called “virtual token passing”, which does not require messages to be sent over the bus. When a master has finished bus access, the token is automatically passed on to the next master, by a cyclic mechanism based on time. The method used in P-NET differs from that used in other multi-master systems. Other busses such as Profibus for example, use real message telegrams for transferring the token. This results in an increase in master processing time, and reduces the capacity of the bus. The virtual token passing principle also accepts that a master might not even be present. In this situation, all devices, including other masters, will continue performing normally.

Each P-NET master is given a node address (NA), between 1 and the number of masters expected within a system. All masters contain an ”idle bus bit period counter” which increments for each bit period the bus is idle, but is reset to zero when the bus becomes active. Each master also has an access counter, which is incremented when the idle bus bit period counter reaches 40, 50, 60, ... When the access counter in a master is equal to its node address, that master holds the token, and is allowed access to the bus. When the access counter exceeds the maximum number of masters, it is preset to 1. Consequently, P-NET does not require any bus arbitrator functions.

A.7 TTP/C

The TTP/C protocol [48] is a fault-tolerant time-triggered protocol that provides the following services.

1. Autonomous fault-tolerant message transport with known delay and bounded jitter between the CNIs of the nodes of a cluster by employing a TDMA medium access strategy on replicated communication channels.
2. Fault-tolerant clock synchronization that establishes the global time base without relying on a central time server.
3. Membership service to inform every node consistently about the “health-state” of every other node of the cluster. This service can be used as an acknowledgment service in multicast communication. The membership service is also used to efficiently implement the fault-tolerant clock synchronization service.
4. Clique avoidance to detect and eliminate the formation of cliques in case the fault hypothesis is violated.

In TTP/C, the communication is organized into rounds, where every node must send a message in every round. The times of the periodic fetch and delivery actions are contained in the message scheduling table, the message descriptor list (MEDL) of each communication controller. To achieve high data efficiency, the sender name and the message name is derived from the send instant.

The clock synchronization of TTP/C exploits the common knowledge of the send schedule: every node measures the difference between the a-priori known expected and the actually observed arrival time of a correct message to learn about the difference between the sender’s clock and the receiver’s clock. This information is used by a fault-tolerant average algorithm to calculate periodically a correction term for the local clock in order to keep the clock in synchrony with all other clocks of the cluster. The membership service employs a distributed agreement algorithm to determine whether the outgoing link of the sender or the incoming link of the receiver has failed. Nodes that have

suffered a transmission fault are excluded from the membership until they restart with a correct protocol state. Before each send operation of a node, the clique avoidance algorithm checks if the node is a member of the majority clique.

B

CNAO Storage Ring Dipole Magnet Power Converter 3000A / ±1600V

This part will describe the design and simulations of the CNAO Dipole Power Converter rated 3000A / ±1600V. The Power Converter will feed the 16+1 synchrotron bending dipole magnets of the CNAO Storage Ring. The actual design confirms how the choice of a twenty-four pulses, 4 bridges series-parallel connected, active filter, bipolar voltage, meets the stringent requested technical specification (10^{-5} of maximum current for the output current residual ripple and setting resolution). The extensive modelling will also be presented. The design includes the strength of the topology design, component derating and component standardization. As the other CNAO power converters, the Storage Ring Dipole Power Converter uses the same digital controller, under licence from the Diamond Light Source [49].

B.1 Introduction

A synchrotron machine, capable to accelerate either light ions or protons, will be the basic instrument of the CNAO (Centro Nazionale di Adroterapia Oncologica), the medical center dedicated to the cancer therapy, that is under construction in Pavia (Italy). The machine complex consists of one proton-carbon-ion linac that will accelerate the particles till the energy of 7 MeV/u. An injection line will transport them to the synchrotron ring where the injected particles will be accelerated and extracted with an energy ranging from 60 to 250 MeV for protons and from 120 to 400 MeV/u for carbon ions.

Protons and light ions are advantageous in conformal hadrontherapy because of three physical properties. Firstly, they penetrate the patient practically without diffusion. Secondly, they abruptly deposit their maximum energy density at the end of their range, where they can produce severe damage to the target tissue while sparing both traversed and deeper located healthy tissues. Thirdly, being charged, they can easily be formed as narrow focused and scanned pencil beams of variable penetration depth, so that any part of a tumour can accurately and rapidly be irradiated. Thus, a beam of protons, or light ions, allows highly conformal treatment of deep-seated tumours with millimeter accuracy.

This appendix is organized as follows. In the first part Power supply specifications are given. In the second part the system topology is faced, while in the third one control design is described. Finally, in the last part, simulations results are reported.

B.2 Power Supply Specification

The CNAO synchrotron ring is equipped with sixteen bending dipole magnets, plus one off line dipole magnet used for magnetic field measurements. In order to drive the par-

Three phase, 50 Hz input mains voltage	15,000 V ± 10%
Maximum Output Current	3,000 A
Maximum Output Voltage	±1,600 V
Maximum Output Power	> 5 MVA
Load Inductance	199.1 mH
Load Resistance (cables included)	79.24 mΩ
Current Setting and Control Range	0.5 to 100% f.s.
Normal Operating Range (N.O.R.)	0.5 to 100 % f.s.
Current Setting Resolution	$< \pm 5 \times 10^{-6}$
Current Reproducibility	$< \pm 2.5 \times 10^{-6}$ f.s.
Current Readout Resolution	$< \pm 5 \times 10^{-6}$ f.s.
Residual Current Ripple (peak to peak) in N.O.R	$< \pm 5 \times 10^{-6}$ f.s.
Linearity Error $[(I_{set} - I_{out})/I_{set}]$	$< \pm 5 \times 10^{-6}$ f.s.
Ambient Temperature	0° to +40° C
Current Stability ($\Delta I/I_{set}$ over the normal operating range)	$< \pm 5 \times 10^{-6}$

Table B.1. Specification for power supply

ticles to the required energy, the magnets must follow a predetermined cycle (see figure B.1).

It consists of 7 parts:

- a starting bottom level, that is about the 5% of the maximum current level;
- a current/field ramp-up till the injection level, in a fixed time;
- a flat-bottom level (depending on the particle type) during which the particles are injected into the ring;
- a current/field ramp-up till the extraction level, in a fixed time;
- a flat-top level (depending on the particular therapy cycle the patient must be subject to) during which the slow extraction takes place and the particles are extracted from the ring; this level does not necessarily coincides with the maximum current level;
- a ramp-up till the maximum field/current value, for a correct magnet "standardization"; no particles are in the ring during this phase of the cycle;
- a ramp-down to the starting bottom level.

To achieve the above magnets behavior, the power supply has to satisfy some tight constraints. In particular, it has to track very high current references (maximum output current of 3000 A) with tracking error smaller than 5 ppm with respect to full scale (see table C.1 for the complete power supply specification).

B.3 Topology

The stringent specification on CNAO synchrotron ring power supply includes two key requirements: high load current and small ripple and tracking error with respect to the specified reference.

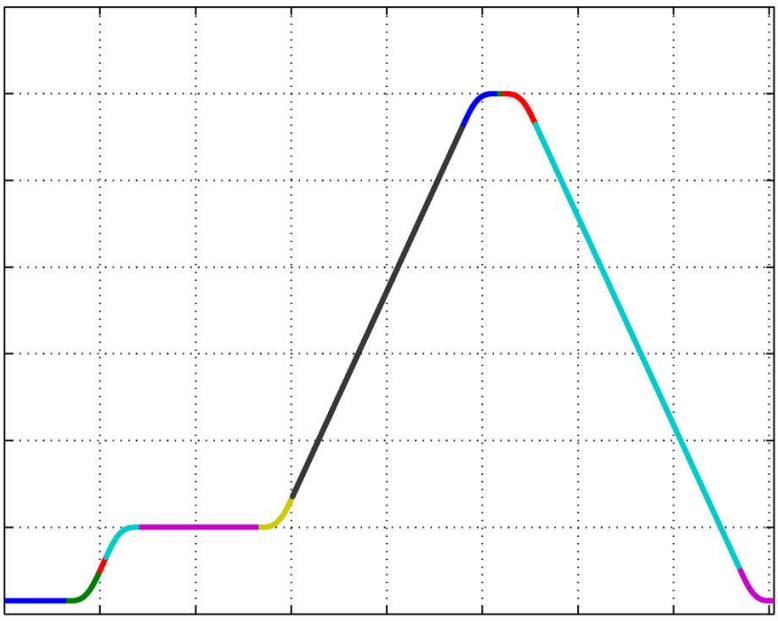

Fig. B.1. Magnets cycle

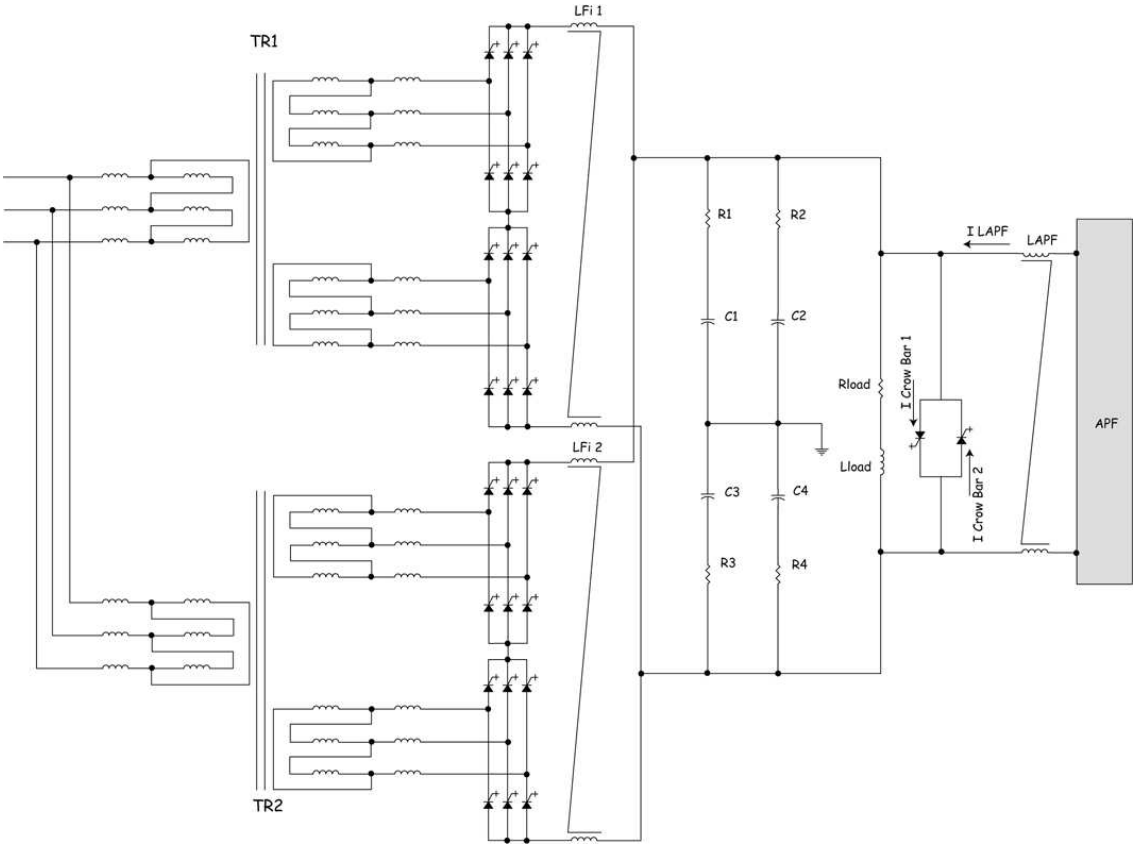

Fig. B.2. Topology of CNAO synchrotron power supply

The high requested current can be supplied by a thyristors-based power converter (in particular a twenty-four pulses SCR rectifier); nowadays, thyristors are the only controllable power device capable to work properly in so high current and voltage conditions. Unfortunately, they introduce high ripple in low load current conditions and their bandwidth is very small. Therefore, the small tracking error requirement cannot be satisfied using a twenty-four pulses SCR rectifier alone. The adopted solution consists in adding an Active Power Filter (APF) which cooperates with the 24-pulses rectifier in order to improve the tracking error capability of the system when the current reference is small or rapidly variable.

A first power converter design was characterized by a series connection between the APF and the 24-pulses rectifier. This choice required the addition of a transformer for the necessary APF DC-link electrical insulation: otherwise in the case of APF not inserted, the APF DC-link would be charged indefinitely. The series solution was soon discarded because the saturation of transformer complicated the control structure. In final power converter topology (fig. C.2) a parallel connection has been preferred for the APF: in this way no additional transformer is needed and control structure is simpler. Moreover, using a suitable reconfigurable control, the parallel connected APF can be disconnected when necessary without mining the system stability.

In summary, the main components present in CNAO power supply topology are: a 24-pulse SCR-rectifier; an IGBT-based Active Power Filter; a digital control system (implemented on DSP and FPGA) controlling the 24-pulses and the APF output currents; a very accurate DCCT sensor (specifically designed for this application); a protection system (crow-bar) to discharge the load stored energy on the load itself.

B.3.1 Twenty-four pulse rectifier

The twenty-four pulse SCR-rectifier is made up of two Δ_{ext}-Δ_{ext}-Δ_{ext} three-phase transformers, four six pulse thyristor bridges and a suitable passive low-pass filter. The primary windings of the transformers are parallel connected, consequently the nominal primary voltage is 15 kV, that is the voltage of medium voltage distribution network that power all the CNAO structure. The secondary windings are series connected. The requested output voltage of each secondary winding can be easily calculated given the maximum output voltage of the power converter V_{max} and the voltage drops in transformers and HV/MV line $V_{linedrop}$. The specification's worst case has been considered, that is a -10% on primary nominal voltage:

$$V_{2(rms)} = \frac{V_{linedrop}}{2} + \frac{1}{0.9}\left(\frac{|V_{max}|}{2} \cdot \frac{\pi}{3\sqrt{2}}\right) \cong 690V$$

The low-pass filter dimensioning is performed to compensate the maximum load voltage ripple that is reached for a firing angle $\alpha = 90^\circ$ of the thyristor bridge. In this case the output voltage waveform is a sawtooth with amplitude peak to peak of 976 V at frequency of 600 Hz. A LPF with resonance frequency of 145 Hz, $A_{db} = -24$ dB at $f = 1200$ Hz and $A_{db} = -35.5$ dB at $f = 2400$ Hz is chosen. The resulting inductors, capacitors and resistors parameters are:

$$L_{Fi1} = L_{Fi2} = 3.2\,\mathrm{mH}$$
$$C_1 = C_3 = 1.2\,\mathrm{mF}$$
$$C_2 = C_4 = 300\,\mu\mathrm{F}$$
$$R_1 = R_3 = 0.7288\,\Omega$$
$$R_2 = R_4 = 25\,\mathrm{m}\Omega$$

B.3.2 Active Power Filter

The APF is built by four modules series connected, each module being a four quadrant full bridge. The main stage of each module is a six pulses IGBT rectifier with a low pass filter whose resonance frequency is 70 Hz.

The sizing of DC-link capacitor is estimated assuming that the output current in the worst case can be approximated to a ramp with a slope of 267A/s for $T_{ramp} = 300$ ms. Hence, balancing the involved energies, the DC-link capacitance is:

$$C_{DC} = 4\frac{2E_{DC}}{(4.4V_{DC} + \Delta V)^2 - (4.4V_{DC})^2} = 15\,\mathrm{mF}$$

where $V_{DC} = 444$V is the nominal DC-link voltage of each module and $E_{DC} = 688J$ is the energy that has to be stored in DC-link in the worst case.

As in twenty-four pulse rectifier, considering the voltage drop on APF lines $V_{linedrop}$, the nominal secondary output rms voltage can be calculated:

$$V_{2APF(rms)} = \frac{\pi V_{DC}}{3\sqrt{2}} + V_{linedrop} = 346\mathrm{V}$$

B.4 System model and Control Design

The aim of the control system design is to develop a closed loop control system suitable to be implemented on a DSP board. The design of the control system in the discrete time plays a fundamental role to satisfy the tight specification on CNAO power supply. To assure enough safety margin on the control system reliability, a sample period T_s=100μs has been chosen, i.e. a frequency of 10 kHz. Moreover, plants and regulators have been discretized using the ZOH method.

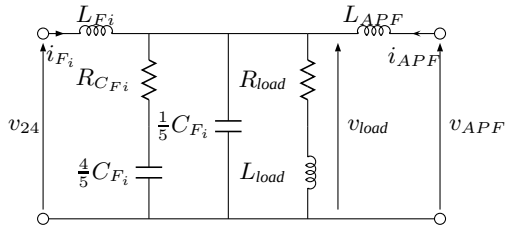

Fig. B.3. Simplified equivalent electrical circuit of the plant

The design of a good control algorithm needs a previous modelling phase. In figure B.3 a simplified electrical equivalent circuit of the plant is presented: from a control

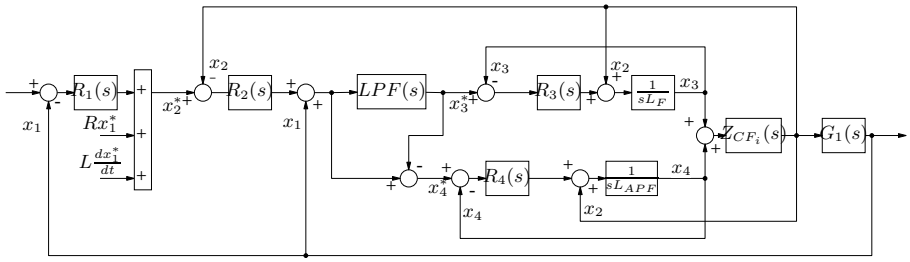

Fig. B.4. Structure of Cascade controller

point of view the series connected dipole magnets constitute a single load with resistance (including cables) $R_{load} = 79.24$ mΩ and inductance $L_{load} = 199.1$ mH.

Assuming the following state variables:

$$\begin{aligned} x_1 &= i \\ x_2 &= v_{load} \\ x_3 &= i_{LF} \\ x_4 &= i_{APF} \\ x_5 &= v_{C1} \end{aligned}$$

and the following controlled variables:

$$\begin{aligned} u_1 &= v_{24} \\ u_2 &= v_{APF} \end{aligned}$$

the state space equation of the system is defined as follows:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

where

$$\begin{aligned} \mathbf{x} &= [x_1, x_2, x_3, x_4, x_5]^T, \\ \mathbf{u} &= [v_{24}, v_{APF}]^T, \end{aligned}$$

$$A = \begin{bmatrix} -\frac{R_{load}}{L_{load}} & \frac{1}{L_{load}} & 0 & 0 & 0 \\ -\frac{5}{C_{Fi}} & -\frac{5}{C_{Fi}R_{CFi}} & \frac{5}{C_{Fi}} & \frac{5}{C_{Fi}} & \frac{5}{C_{Fi}R_{CFi}} \\ 0 & -\frac{1}{L_{Fi}} & 0 & 0 & 0 \\ 0 & -\frac{1}{L_{APF}} & 0 & 0 & 0 \\ 0 & \frac{5}{4C_{Fi}R_{CFi}} & 0 & 0 & -\frac{5}{4C_{Fi}R_{CFi}} \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{L_{Fi}} & 0 \\ 0 & \frac{1}{L_{APF}} \\ 0 & 0 \end{bmatrix}.$$

Denoting with $i^* = x_1^*$ the reference for the current running through the magnets, the goal of the CNAO controller is to generate the right controlling input $\mathbf{u}$ able to track i^* with a maximum error equal to ± 0.015 A.

The developed solution is a cascade controller (see figure B.4). It is composed by three nested loops, with the inner one composed by other two parallel loops, that will be analyzed one by one in the next paragraphs.

B.4.1 Outer loop

The outer loop has to generate a correct reference $v^*_{load} = x_2$ for the intermediate loop when the reference i^* is given and the tracking error is computed. The considered plant, obtained by a simple voltages balance on the load, is:

$$\dot{x}_1 = \frac{1}{L}(x_2 - Rx_1).$$

Since the controller has to track a linearly growing current reference it must contain a double integrator. The controller zeros have been placed to ensure a bandwidth as large as required by the current error requirements with the assigned current references.

The resulting regulator is:

$$R_1(s) = \frac{10^{\frac{93}{20}}}{s^2}(1 + s\frac{L}{R}1.1)(1 + \frac{s}{2\pi 30})$$

The plant $G_1(s)$ and the regulator $R_1(s)$ transfer functions are discretized by means of the zero order hold method with sampling time $T_s = 10^{-4}$ sec. The Bode diagram of the resulting loop function is shown in fig. B.5.

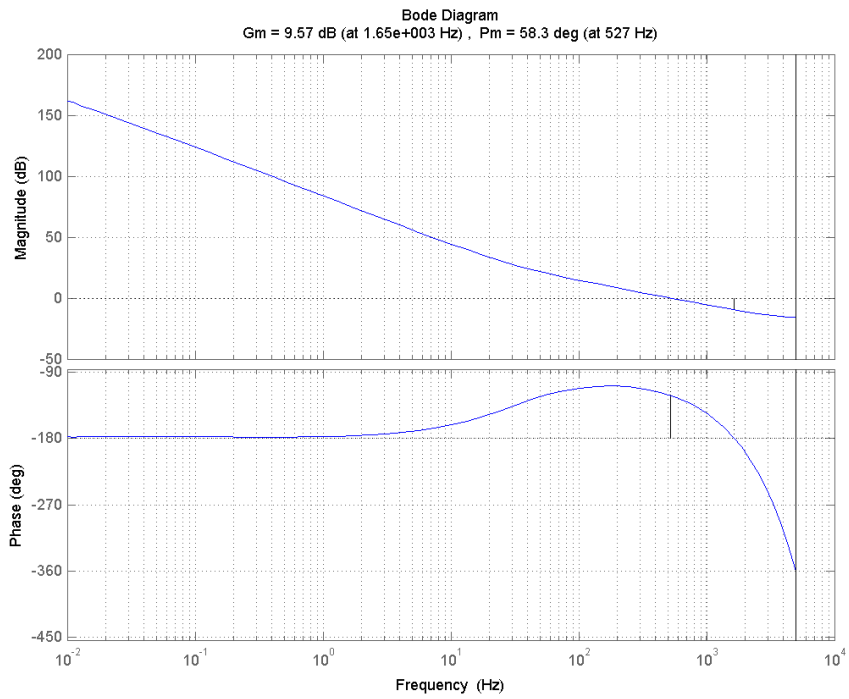

Fig. B.5. Bode diagram of outer loop regulator

As the reference trajectory and the relation between the state variables x_1 and x_2 are well known, performance can be improved by adding a feedforward action, i.e. by adding to the output of $R_1(s)$ the sum $Rx_1^* + L\dot{x}_1^*$.

B.4.2 Intermediate loop

When the reference for the load voltage x_2^* is given, next step is to compute the amount of current that has to be drawn from the 24 pulse rectifier and from the APF. Applying Kirchoff's current law we obtain:

$$x_s = x_1 + x_{ZCFi} = x_3 + x_4$$

where x_{ZCFi} is the current flowing into the two branches in parallel with the load and the value the intermediate controller must generate, as x_1 is given.

So, the plant, in the Laplace domain, is:

$$G_2(s) = Z_{ZCFi}(s) = \frac{X_2(s)}{X_{ZCFi}(s)} = \\ = (R_{CFi} + \frac{4}{5sC_{Fi}})//(\frac{1}{5sCFi})$$

The discretized designed controller is:

$$R_2(z) = 0.89125$$

The Bode diagram of the loop function $R_2(z)G_2(z)$ is reported in fig. B.6.

B.4.3 Inner loops

The sum x_s between the actual value of x_1 and the computed value of x_{ZCFi} is the reference for the inner loop. However this current cannot be entirely supplied only by the 24 pulse rectifier due to its limited bandwidth . So, the separation between low frequency components, that will be tracked by the 24 pulse rectifier, and high frequency ones, that will be tracked by the APF, is required. This result is obtained by lowpassing the reference x_s^* with a 1^{st} order low pass filter, having cut frequency at 70 Hz:

$$LPF(z) = 48.327 * 10^{-5}\frac{(z + 0.9793)}{(z - 0.9691)^2}.$$

The LPF output will be x_3^*. Subtracting it from x_s, x_4^* is given too.

The system to be controlled by the 24 pulse rectifier controller is:

$$\dot{x}_3 = \frac{1}{L_{Fi}}(v_{24} - x_2)$$

Defining $\tilde{x}_3 = x_3 - x_3^*$, the controlling voltage v_{24} is given by:

$$V_{24} = X_2(z) - R_3(z)\tilde{X}_3(z)$$

where:

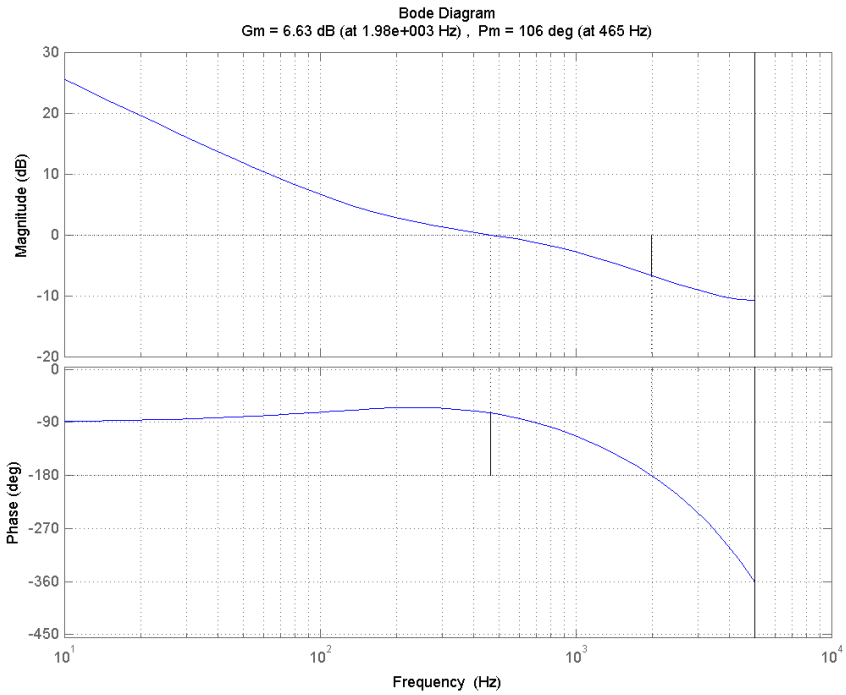

Fig. B.6. Bode diagram of intermediate loop regulator

$$R_3(z) = 1.0042\frac{z - 0.9937}{z - 1}.$$

The Bode diagram of the resulting loop function is shown in fig. B.7.

Similarly, for the Active Power Filter the system is:

$$\dot{x}_4 = \frac{1}{L_{APF}}(v_{APF} - x_2)$$

Defining $\tilde{x}_4 = x_4 - x_4^*$, the controlling voltage v_{APF} is given by:

$$V_{APF} = X_2(z) - R_4(z)\tilde{X}_4(z)$$

where:

$$R_4(z) = 1.3343\frac{z - 0.9813}{z - 0.9969}.$$

The Bode diagram of the resulting loop function is shown in fig. B.8.

B.5 Simulations Results

To test the topology and the adopted control strategies, extensive simulations have been carried out using Matlab and Simulink.

A Simulink model of the system has been implemented using SimPowerElectronics components initialized with parameters of table C.1. The system has been tested with the whole set of current references, each one made up of constants or ramps connected by 5th order polynomial curves with no discontinuities in the first and second derivative (see

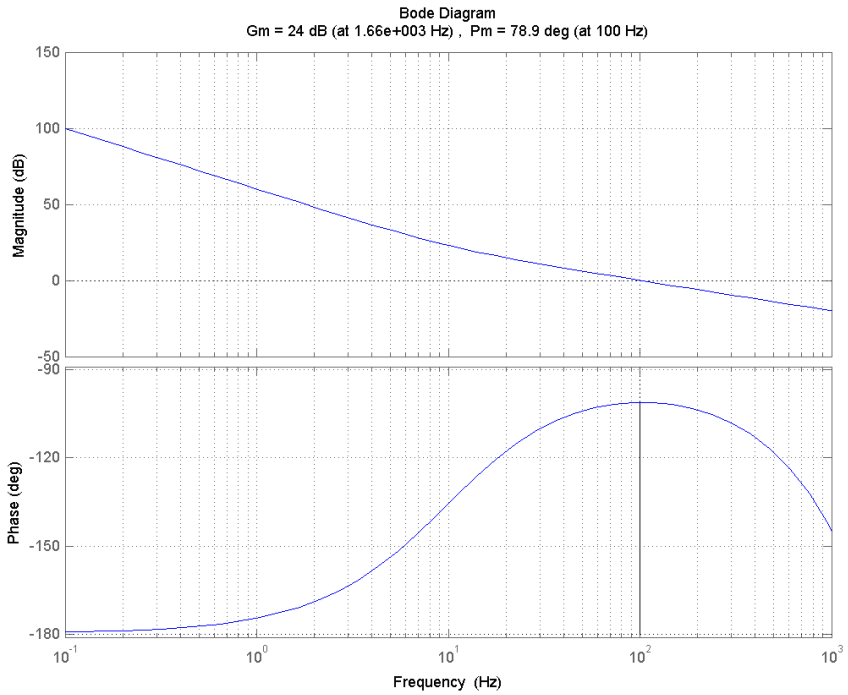

Fig. B.7. Bode diagram of 24 pulse rectifier loop

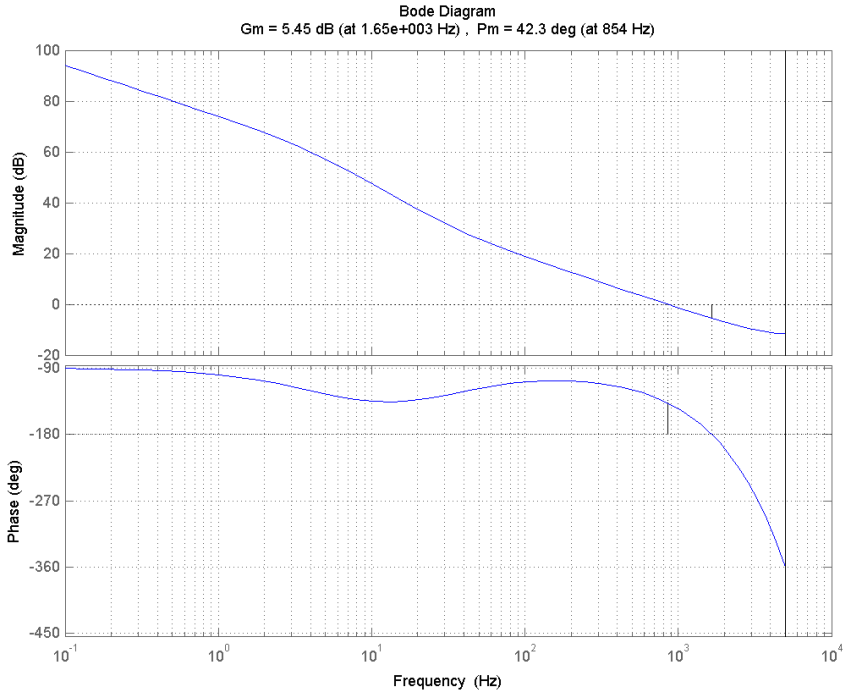

Fig. B.8. Bode diagram of APF loop

fig. B.1). For simulation purposes, these analog signals have been approximated to 100 KHz sampled signals (ten times the digital controller operating frequency).

All the tests have been performed both in nominal V_{line} conditions and in critical V_{line} conditions when input mains voltage can be either 110% or 90% of nominal value (respectively fig. B.9 and B.10). Finally a test with V_{line} equal to 90% the nominal value and a 5% load derating has been carried out (fig. B.11).

Factory tests are scheduled before the end of 2006.

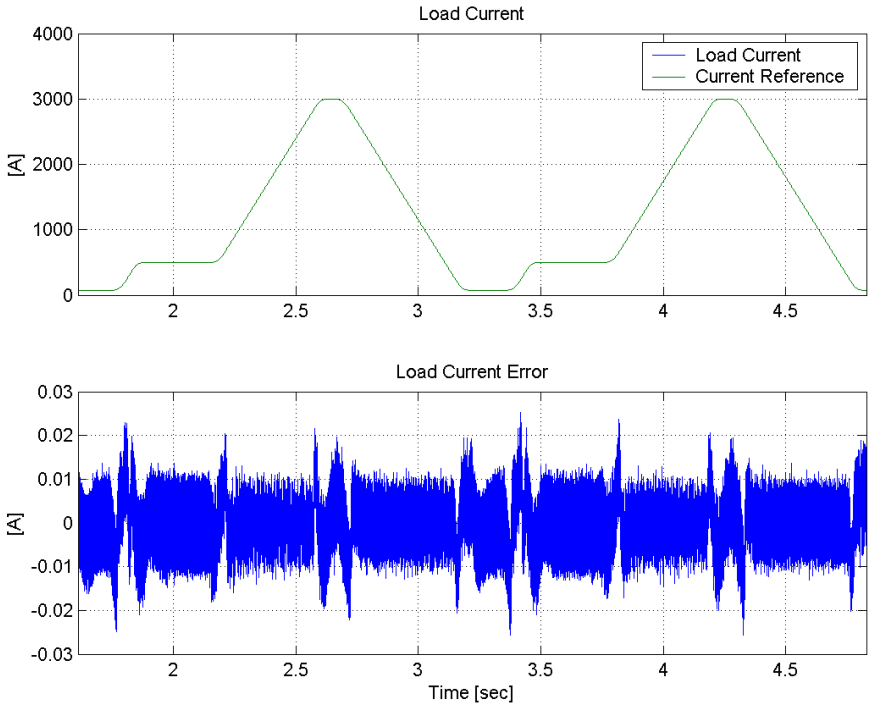

Fig. B.9. Total load current error (ripple and linearity error), case with V_{line} at 110%.

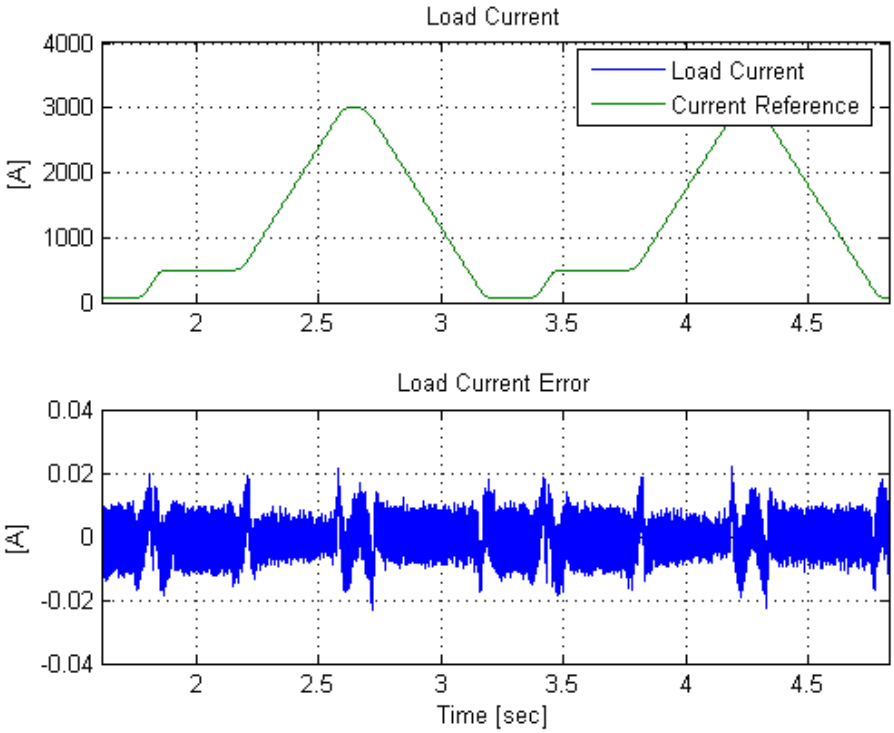

Fig. B.10. Total load current error (ripple and linearity error), case with V_{line} at 90%.

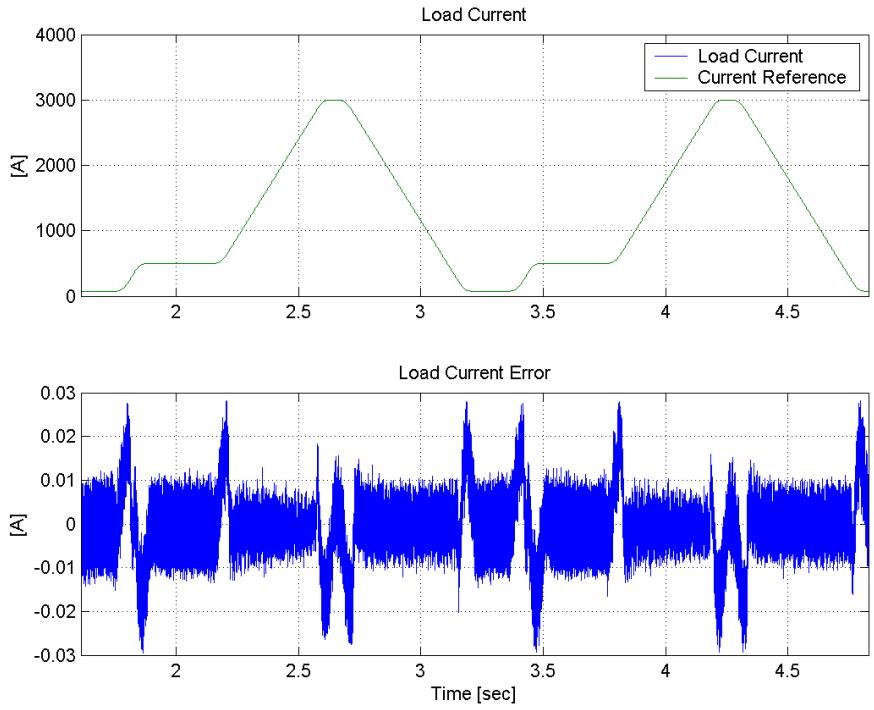

Fig. B.11. Total load current error (ripple and linearity error), case with V_{line} at 90% and 5% load derating.

C

CNAO Resonance Sextupole Magnet Power Converters

The CNAO Resonance Sextupole Magnet Power Converter requirements for the Storage Ring of the CNAO Project are described together with performance and initial operating experience. In particular the achieved performances will be compared with the specification and the extensive modelling that was done during the design phase. Not only the tight required performances were emphasized during the design phase but also particular attention was put on reliability and minimization of the repairing time (MTTR). Some fundamental criteria, like component de-rating and standardisation, have also been taken into account during the component choice phase. All converters adopt the switching technology with full digital control and a common control interface, that, as for the other CNAO power converters, uses the same digital controller, under licence from the Diamond Light Source [50].

C.1 Introduction

A synchrotron machine, capable to accelerate either light ions or protons, will be the basic instrument of the CNAO (Centro Nazionale di Adroterapia Oncologica), the medical center dedicated to the cancer therapy, that is under construction in Pavia (Italy). The machine complex consists of one proton-carbon-ion linac that will accelerate the particles till the energy of 7 MeV/u. An injection line will transport them to the synchrotron ring where the injected particles will be accelerated and extracted with an energy ranging from 60 to 250 MeV for protons and from 120 to 400 MeV/u for carbon ions.

Protons and light ions are advantageous in conformal hadrontherapy because of three physical properties. Firstly, they penetrate the patient practically without diffusion. Secondly, they abruptly deposit their maximum energy density at the end of their range, where they can produce severe damage to the target tissue while sparing both traversed and deeper located healthy tissues. Thirdly, being charged, they can easily be formed as narrow focused and scanned pencil beams of variable penetration depth, so that any part of a tumour can accurately and rapidly be irradiated. Thus, a beam of protons, or light ions, allows highly conformal treatment of deep-seated tumours with millimeter accuracy.

This appendix is organized as follows. In the first part Power supply specifications are given. In the second part the system topology is faced, while in the third one control design is described. Finally, in the last part, simulations results are reported.

C.2 Power Supply Specification

The sextupole is a special magnet of the CNAO synchrotron used to extract the particles from the main ring. It must stay at zero current during particles injection and acceleration,

and ramp up to the specified current, different from cycle to cycle, in an overall time of about 50ms.

The corresponding current reference for the sextupole power supply is presented in figure C.1.

The detailed power supply specification can be found in Table C.1: it's worth to note the short rising time (25 ms) and the small tracking error (less than 50 ppm).

Three phase, 50 Hz input mains voltage	400 V $\pm 10\%$
Maximum Output Current	650 A
Maximum Output Voltage	± 40 V
Maximum Output Power	> 24kVA
Load Inductance (including cables)	3.26 mH
Load Resistance (including cables)	38.65 mΩ
Current Setting and Control Range	0.5 to 100% f.s.
Normal Operating Range (N.O.R.)	0.5 to 100 % f.s.
Current Setting Resolution	$< \pm 1 \times 10^{-4}$ f.s.
Current Reproducibility	$< \pm 5 \times 10^{-5}$ f.s.
Current Readout Resolution	$< \pm 1 \times 10^{-4}$ f.s.
Residual Current Ripple (peak to peak) in N.O.R	$< \pm 1 \times 10^{-4}$ f.s.
Linearity Error [(Iset - Iout)/Iset]	$< \pm 5 \times 10^{-5}$ f.s.
Ambient Temperature	0° to $+40^\circ$ C
Current Stability (.I/Iset over the normal operating range)	$< \pm 1 \times 10^{-4}$
Maximum ramp up/down time in the N.O.R.	25 ms
Maximum first ramp up time	150 ms

Table C.1. Specification for power supply

C.3 Topology

The first proposed solution for the sextupole power supply used a Pulsed Power Supply Topology. It was composed by a Pulse Section (essentially a capacitor with thyristor bridge, resonant with the load inductance) and a Switching Section as regulator converter. After some considerations on the specification the Pulsed solution was abandoned. In fact, the same aim is achievable using a simpler and more standard switching topology.

The adopted solution is composed by an input stage and an output stage (see figure C.2). The input stage is made up of a $\Delta - Y$ transformer, a diode bridge and an input passive filter. The output stage is made up of two parallel connected IGBT full-bridges and an output filter. The load is parallel connected to the output filter.

C.3.1 Input stage dimensioning

The most important parameter to evaluate in the sizing of the input stage is the DC-link capacity. This value is estimated considering that the current drawn from the DC-link in

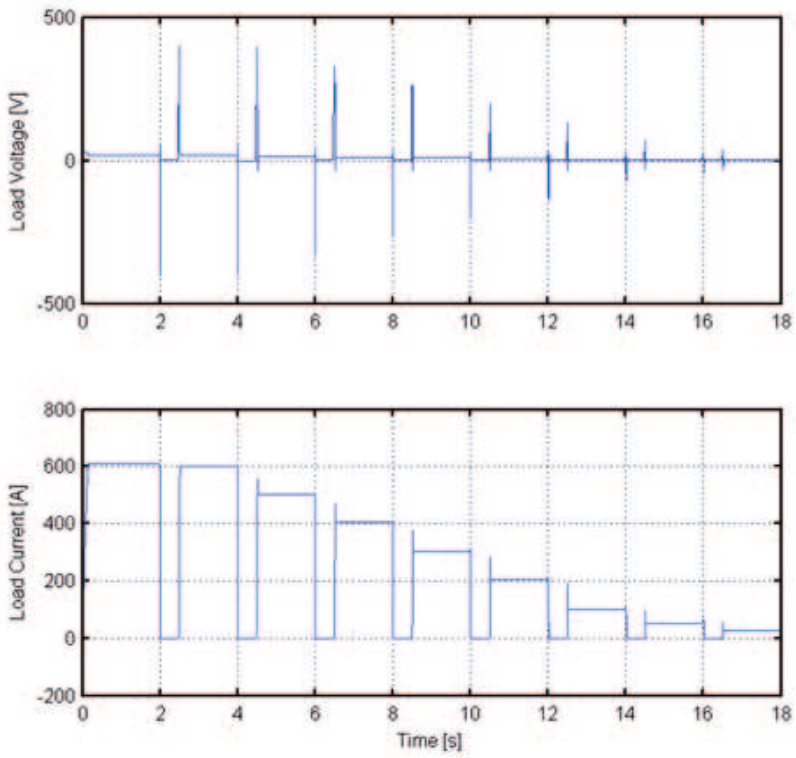

Fig. C.1. Load voltage and current during a treatment plan where the beam energy decrease cycle by cycle

the worst case is a ramp with a slope of 26000 A/s for $T_{ramp} = 25$ ms. Hence, balancing the involved energies and considering the physical limitations of the components, a DC-link capacitance $C_{F_i} = 165$ mF is chosen.

Then, in order to have the input filter resonance frequency at $f_0 = 22.5$ Hz, an inductance value of 300 μH is adopted.

C.3.2 Output stage dimensioning

Each full-bridge module is driven by a PWM signal at a frequency of 10 kHz. Since no phase displacement techniques are used, the resulting voltage ripple has an equivalent frequency of 20 kHz. The maximum current ripple is reached when the modulation index is equal to $\frac{1}{2}$. In order to maintain the ripple under the specification threshold the resulting low pass filter parameters are: $L_{Fumod1} = L_{Fumod2} = 60\mu$H, $C_{Fu1} = 320\,\mu$F, $C_{Fu2} = 80\,\mu$F, $R_{CFu} = 0.66\,\Omega$

C.4 Control Design

The aim of the digital control is twofold: make the power supply satisfy the specification (mostly the small rising time) and make the two output modules work in the same way by drawing the same amount of current from both of them.

Analyzing the electrical circuit made up of output filter and load, the following equations hold:

$$\begin{cases} V_1(s) - sL_{Fumod} I_1(s) = I_s(s) Z_b(s) \\ V_2(s) - sL_{Fumod} I_2(s) = I_s(s) Z_b(s) \end{cases}$$

where:

- V_i is the voltage input of the i-th module;
- I_i is the current flowing in the inductance of the i-th module;
- Z_b is the equivalent impedance corresponding to the parallel of output filter capacitor and load impedance;
- $I_s = I_1 + I_2$;
- $L_{Fumod1} = L_{Fumod2} = L_{Fumod}$.

Performing the change of coordinates

$$\begin{bmatrix} I_s \\ I_d \end{bmatrix} = T \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}, \qquad \text{where} \quad T = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

the following equations are obtained:

$$\begin{cases} I_s(s) = \dfrac{V_1(s) + V_2(s)}{sL_{Fumod} + 2Z_b(s)} = \dfrac{V_s(s)}{sL_{Fumod} + 2Z_b(s)} \\ I_d(s) = \dfrac{V_1(s) - V_2(s)}{sL_{Fumod}} = \dfrac{V_d(s)}{sL_{Fumod}} \end{cases}$$

As the real target of the control is I_{load}, considering the current divider the plant equations are:

$$\begin{cases} I_{load}(s) = \dfrac{1}{sL_{load} + R_{load}} \dfrac{Z_b(s)}{\dfrac{sL_{Fumod}}{2} + Z_b(s)} \dfrac{V_s(s)}{2} = \\ \qquad\qquad = H_{1,2}(s) \dfrac{V_s(s)}{2} \\ I_d(s) = \dfrac{1}{sL_{Fumod}} V_d(s) = H_d(s) V_d(s) \end{cases}$$

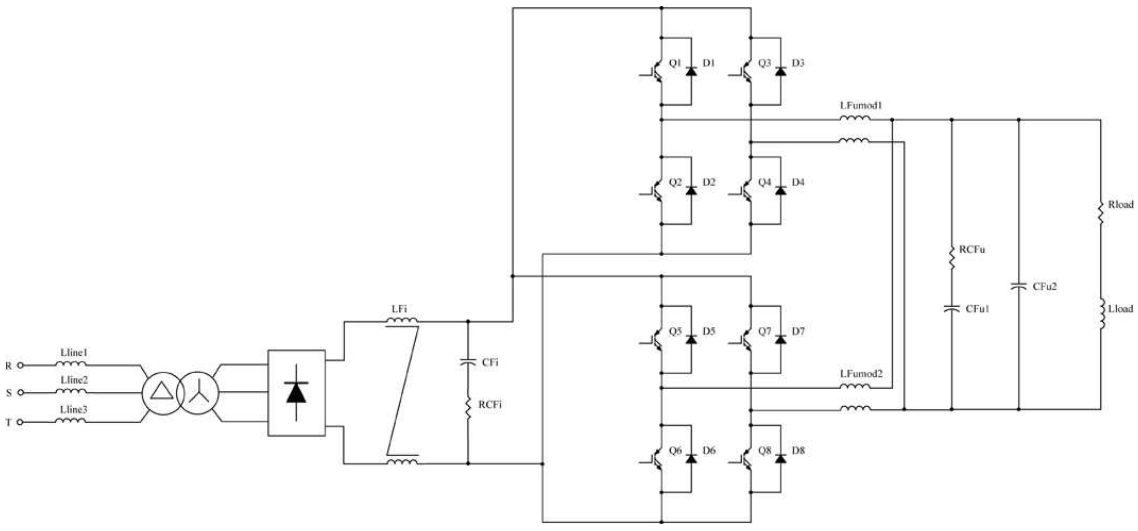

Fig. C.2. Topology of sextupole magnet power supply

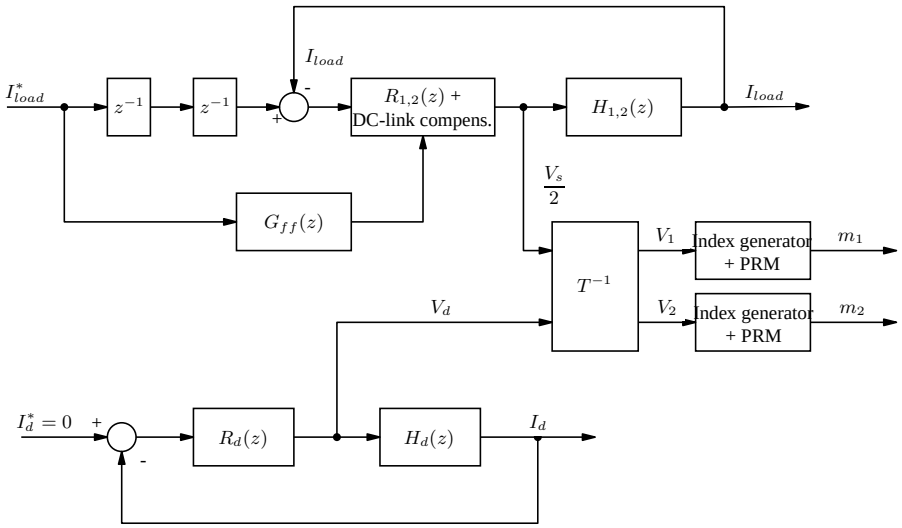

Fig. C.3. Structure of digital controller

The regulator developed for $H_{1,2}(s)$ is a PI controller:

$$R_{1,2}(s) = k_p + \frac{k_i}{s}$$

with $k_p = 7.08$ and $k_i = 7117$ while the regulator for $H_d(s)$ is a simple P controller with very slow dynamics

$$R_d(s) = 0.0562$$

Then, in order to implement controllers on a DSP board, both regulators and plants have been discretized using the ZOH method with a sample time of 100 μs, i.e. a frequency of 10 kHz. The control action is performed by generating appropriate PWM indexes that are calculated from V_s and V_d with a suitable change of base.

The overall performances can be improved by means of the following feedforward action based on the load discretized model inversion.

$$G_{ff}(z) = 32.6193\,\frac{z - 0.9988}{z}$$

Since the relative degree of the controlled system is 2, the digitalized current reference is 2 samples delayed in order to synchronize it with the feedforward action.

Final improvements on the control system are the *DC-link voltage compensation*, introduced to avoid drawing more current than available on the DC-link, and a combination of PWM and PRM (Pulse Repetition Modulation) techniques to increase the accuracy of digitalization.

C.5 Simulations Results

To test the topology and the adopted control strategies, extensive simulations have been carried out using Matlab and Simulink. A Simulink model of the system has been implemented using SimPowerElectronics components initialized with parameters of table C.1. All the tests have been performed both in nominal V_{line} conditions and in critical V_{line}

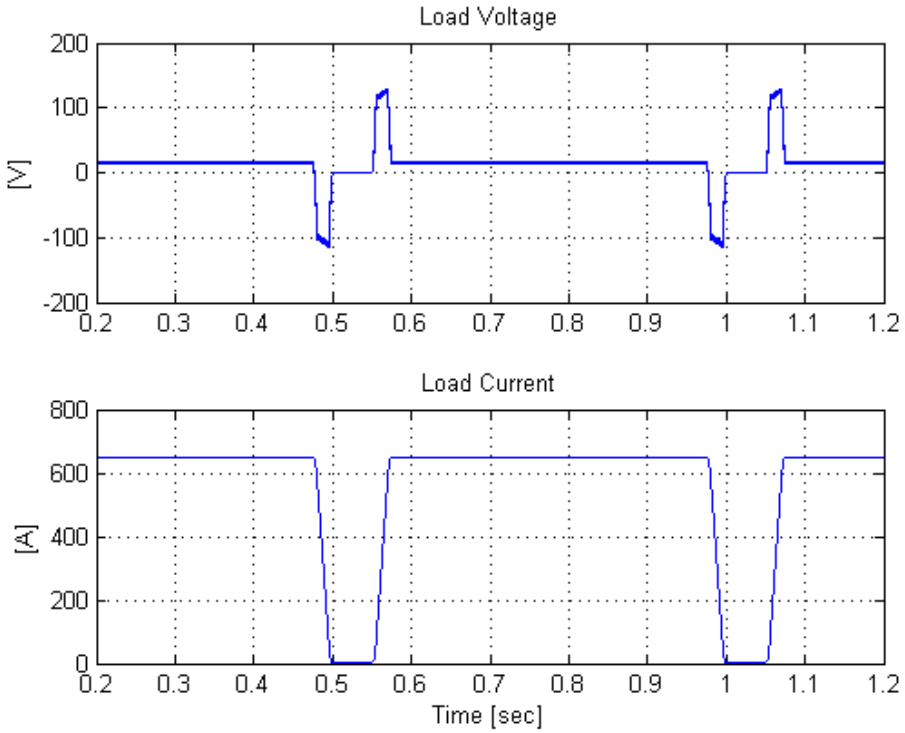

Fig. C.4. Load voltage and current V_{line} at 90%.

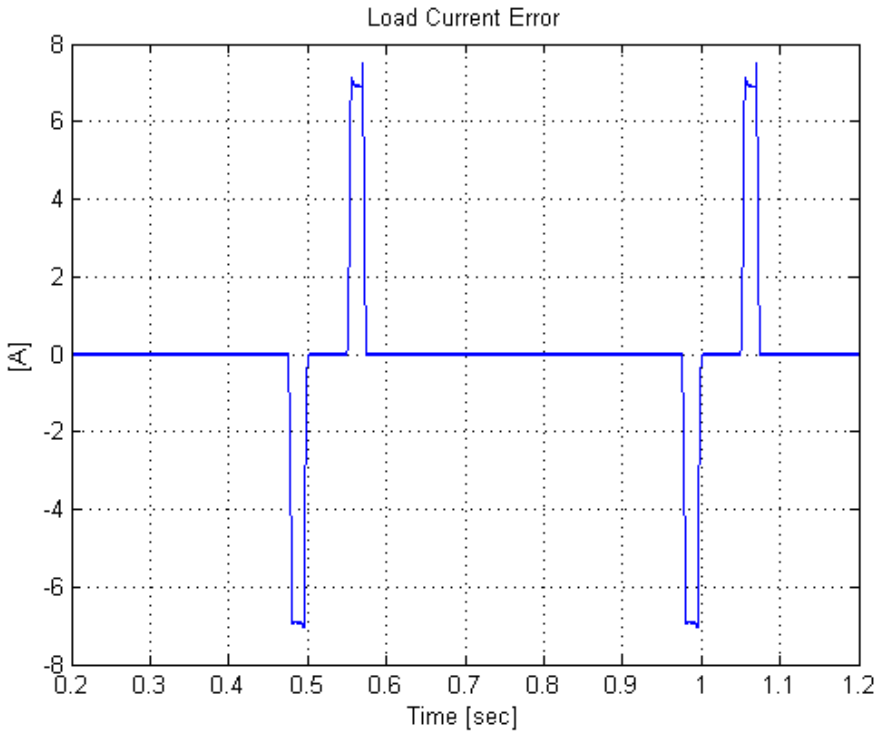

Fig. C.5. Load current error, V_{line} at 90%.

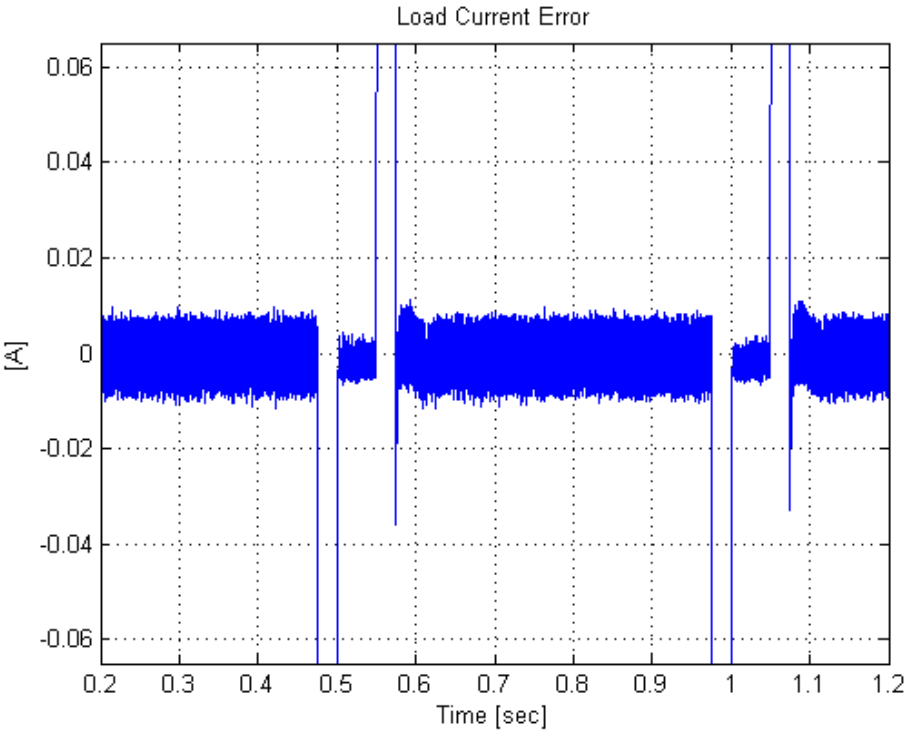

Fig. C.6. Load current error (zoom), V_{line} at 90%.

conditions when input mains voltage can be either 110% or 90% of nominal value (see figures C.4, C.5 and C.6 for 90% case).

Delivery of Sextupole magnet power supply is scheduled for July 2006.

References

[1] *IEC 61158. Digital data communications for measurement and control – Fieldbus for use in industrial control systems. Part 1: Overview and guidance for the IEC 61158 series*. July 2004.

[2] *IEC 61158. Digital data communications for measurement and control - Fieldbus for use in industrial control systems. Part 2: Physical layer specification and service definition*. July 2004.

[3] *IEC 61158. Digital data communication for measurement and control - Fieldbus for use in industrial control systems. Part 3: Data link service definition*. July 2004.

[4] *IEC 61158. Digital data communication for measurement and control - Fieldbus for use in industrial control systems. Part 4: Data link protocol specification*. July 2004.

[5] *IEC 61158. Digital data communication for measurement and control - Fieldbus for use in industrial control systems. Part 5: Application layer service definition*. July 2004.

[6] *IEC 61784. Digital Data Communications for Measurement and Control – Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems*. 2006.

[7] *IEC 61784. Digital data communication for measurement and control - Part 2: Additional profiles for ISO/IEC 8802-3 based communication networks in real-time applications*. 2006.

[8] *IEC 61784. Digital data communications for measurement and control - Part 3: Profiles for functional safety communications in industrial networks*. 2006.

[9] *IEC 61784. Digital data communication for measurement and control - Part 5: Installation profiles for communication networks in industrial control systems*. 2006.

[10] *IEC 62026. Low-voltage switchgear and controlgear - Controller-device interfaces (CDIs) -Part 1: General rules*. 2006.

[11] *IEC 62026. Low-voltage switchgear and controlgear - Controller-device interfaces (CDIs) - Part 2: Actuator sensor interface (AS-i)*. 2006.

[12] *IEC 62026. Low-voltage switchgear and controlgear - Controller-device interfaces (CDIs) - Part 3: DeviceNet*. 2006.

[13] *EN 50170. General Purpose Field Communication System. Part 1: P-NET*. 1996.

[14] *EN 50170. General Purpose Field Communication System. Part 2: Profibus*. 1996.

[15] *EN 50170. General Purpose Field Communication System. Part 3: WorldFIP*. 1996.

[16] *EN 50325. Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces. Part 1: General requirements*. December 2002.

[17] *EN 50325. Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces. Part 2: DeviceNet*. December 2002.

[18] *EN 50325. Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces. Part 3: Smart Distributed Systems (SDS)*. December 2002.

[19] *EN 50325. Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces. Part 4: CANOpen*. December 2002.

[20] M.Felser and T.Sauter. The fieldbus war: History or short brake between battles? *4th IEEE International Workshop on Factory Communications Systems (WFCS2002)*, pages 73–80, 2002.

[21] J.P.Thomesse. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, 93:1073–1101, June 2005.

[22] J.D.Decotignie. Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, 93:1102–1117, June 2005.

[23] M.Felser. Real-time ethernet—industry prospective. *Proceedings of the IEEE*, 93:1118–1129, June 2005.

[24] C.Rossi and M.Spera. A PLL-based approach to clock synchronization for trajectory rebuilding in event-triggered communication systems. In *12th IEEE Conference on Emerging Technologies and Factory Automation*, 2007.

[25] D.Mills. Internet time synchronization: The Network Time Protocol. *IEEE Transactions on Communications*, 39:1482–1493, October 1991.

[26] *1588: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Sistems*. 2002.

[27] A.S.Tanenbaum. *Reti di calcolatori*. Pearson, 2004.

[28] A.Tilli. Slides for the course on "ingegneria e tecnologie dei sistemi di controllo". http://www.lar.deis.unibo.it/people/atilli/Ing-Tecn-LA.html.

[29] C.Bonivento, L.Gentili, and A.Paoli. *Sistemi di automazione industriale: architetture e controllo*. McGraw-Hill, 2006.

[30] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[31] C. Rossi. Slides for the course on distributed control systems. http://www.lar.deis.unibo.it/people/crossi/SCD.html.

[32] E.V.Appleton. Automatic synchronization of triode oscillators. In *Proc. Cambridge Phil. Soc.*, volume 21, page 231, 1922-1923.

[33] B.Razavi. *Monolithic Phase-Locked Loops and Clock Recovery Circuits: Theory and Design*. Wiley-IEEE Press, 1996.

[34] J.P.Hein and J.W.Scott. Z-domain model for discrete-time PLL's. *IEEE Trans.Circuits and Systems*, 35:1393–1400, 1988.

[35] R.S.Co and J.H.MulliganJr. Optimization of phase-locked loop performance in data recovery systems. *IEEE Journal of Solid-State Circuits*, 29:1022–1034, 1994.

[36] T.Olsson and P.Nilsson. A digitally controlled PLL for SoC applications. *IEEE Journal of Solid-State Circuits*, 39, 2004.

[37] N.Kim and I.Ha. Design of ADPLL for both large lock-in range and good tracking performance. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, 46, 1999.

[38] R.B.Staszewski and P.T.Balsara. Phase-domain all-digital phase-locked loop. *IEEE Transactions on Circuits and Systems-II: Express Briefs*, 52, 2005.

[39] T.H.Lee and J.Bulzacchelli. A 155-MHz clock recovery delay-and phase-locked loop. *IEEE Journal of Solid-State Circuits*, SC-27:1736–1746, 1992.

[40] H.Brugel and P.F.Driessen. Variable bandwidth DPLL synchronizer with rapid acquisition implemented as a finite state machine. *IEEE Transactions on Communications*, 42, 1994.

[41] T.H.Kim and B.Kim. Dual-loop digital PLL design for adaptive clock recovery. In *Design Automation Conference 1998. Proceedings of the ASP-DAC '98. Asia and South Pacific*, 1998.

[42] R.Zurawski. Scanning the issue: Special issue on industrial communication systems. *Proceedings of the IEEE*, 93:1067–1072, June 2005.

[43] M.Felser. The fieldbus standards: History and structures. In *Technology Leadership Day 2002, Organised by MICROSWISS Network*, October 2002.

[44] *Bosch CAN Specification—Version 2.0 Part A*. 1991.

[45] T.Führer, B.Müller, W.Dieterle, F.Hartwich, R.Hugel, M.Walther, and R.Bosch GmbH. Time triggered communication on CAN (Time Triggered CAN- TTCAN). http://www.semiconductors.bosch.de/.

[46] O.Mirabella. Slides for the course on "reti di calcolatori". http://reti1.diit.unict.it/.

[47] "The International P-NET User Organization". The p-net fieldbus for process automation. http://www.p-net.dk/fieldbus/fieldbus.html.

[48] H.Kopetz and G.Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91:112–126, January 2003.

[49] F.Ronchi, C.Rossi, M.Spera, and M.Toniato. CNAO storage ring dipole magnet power converter 3000a / +- 1600V. In *10th biennial European Particle Accelerator Conference, EPAC '06*, June 2006.

[50] D.Bagnara, A.Tilli, M.Spera, and M.Toniato. CNAO resonance sextupole magnet power converters. In *10th biennial European Particle Accelerator Conference, EPAC '06*, June 2006.