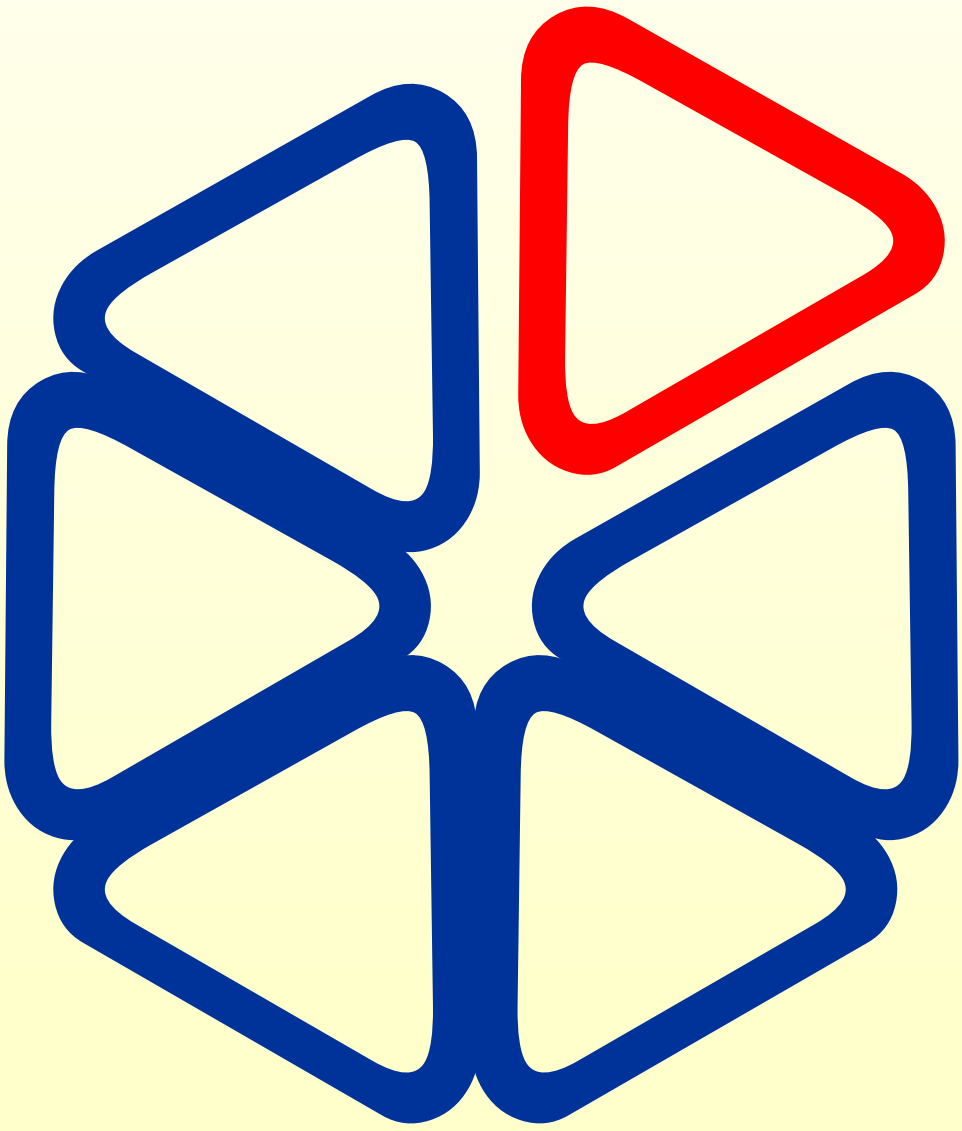
Active Fault-Tolerance in Wireless Networked Control Systems

Cristian I. Chihaia



Active Fault-Tolerance in Wireless Networked Control Systems

Von der Fakultät für Ingenieurwissenschaften
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Dissertation

von

Dipl.-Ing. Cristian Ionuț Chihaia

aus

Iași, Rumänien

Referent: Prof. Dr.-Ing. Steven X. Ding
Korreferent: Prof. Dr.-Ing. Andreas Czylwik
Tag der mündlichen Prüfung: 24.06.2010

This page intentionally left blank,
and it could be used for writing notes.

Acknowledgements

Ich danke Ihnen aus vollem Herzen!

Thank you from all my heart!

谢谢!

Muchas gracias!

دِلِی شُکرِیَہ

बहुत धन्यवाद

شُکرًا جَزِیلًا

ขอบคุณมากครับ

از صمیم قلب متشکرم

Vă mulțumesc din toată inima!

This page intentionally left blank,
and it could be used for writing notes.

Contents

This page intentionally left blank,
and it could be used for writing notes.

List of Figures

This page intentionally left blank,
and it could be used for writing notes.

List of acronyms

P_{lm} Probability of a lost sensor measurement

ACK Acknowledgement

API Application Programming Interface

ARQ Automatic Repeat Request

BER Bit Error Rate

BPSK Binary Phase Shift Keying

CAN Controller Area Network

CAP Contention Access Period

CFD Correct Fault Detection

CSMA Carrier Sense Multiple Access

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CSMA/CD Carrier Sense Multiple Access with Collision Detection

DCF Distributed Coordination Function

DD Detection Delays

DECT Digital Enhanced Cordless Telecommunication

DSSS Direct Sequence Spread Spectrum

ECT Earliest Completion Time

FAR False Alarm Rate

FD Fault Diagnosis

FDD Frequency Division Duplex

FDI Fault Detection and Isolation

FDR Fault Detection Rate

FEC Forward Error Correction

FHSS Frequency-Hopping Spread Spectrum

FMEA Failure Modes and Effects Analysis

FRR Flexible Retransmission Request

FTC Fault-Tolerant Control

GPOS general-purpose operating system

GSM Global System for Mobile Communication

GTSs Guaranteed Time Slots

IAE Integral of the Absolute magnitude of the Error

IFD Incorrect Fault Detection

ISM Industrial, Scientific and Medical

ISO International Organization for Standardization

LAN Local Area Network

LCM least common multiple

LS List Scheduling

MAC Media Access Control

MADWiFi Multiband Atheros Driver for WiFi

MDR Missed Detection Rate

MFD Missed Fault Detection

MTTF Mean Time To Failure

NCS Networked Control System

OFDM Orthogonal Frequency Division Multiplex

OS Operating System

OSI Open Systems Interconnection

OSPF Open Shortest Path First

PCS Personal Communications Service

PLR Packet Loss Rate

QoC Quality of Control

QoS Quality of Service

RF Radio Frequency

RSSI Received Signal Strength Indication

RTOS real-time operating system

RTS/CTS Request To Send/Clear To Send

SOHO Small Office Home Office

TCP Transmission Control Protocol

TDM Time Division Multiplexing

TDMA Time Division Multiple Access

TSMP Time Synchronized Mesh Protocol

UDP User Datagram Protocol

UWB Ultra-Wideband

WAN Wide Area Network

WISA Wireless Interface for Sensors and Actuators

WLAN Wireless Local Area Network

WNCS Wireless Networked Control System

WPAN Wireless Personal Area Network

WSAN Wireless Sensor and Actuator Network

WSN Wireless Sensor Network

This page intentionally left blank,
and it could be used for writing notes.

Abstract

With the control systems becoming more and more complex from both the communication and computer science perspectives, there is an increasing need for methods and analysis tools that are able to quantify the intricate relationships between the communication/computer design and runtime parameters (e.g. data rate, bit error rate) and the control/monitoring specific parameters (e.g. control performance, false alarm rate, missed detection rate). Parallel to the complexity, also the demand for safer, more reliable and more environmentally friendly systems had grown, which led to the emergence of fault-tolerant control and monitoring systems.

In a Wireless Networked Control System (WNCS), several nodes or components of the system may communicate over the common network that connects them together. Thus, there may be communication taking place between the sensors and the controller nodes, among the controllers themselves, among the sensors themselves, among the actuator themselves, and between the controller and the actuator nodes. The purpose of this communication is to improve the performance of the control system. The performance may be a measurable quantity defined in terms of a performance criterion, as in the case of optimal control or estimation, or it may be a qualitative measure described as a desired behavior.
Each node of the WNCS may act as a decision maker, making control as well as communication decisions. The presence of a network brings in constraints in the design of the control system, as information between the various decision makers must be exchanged according to the rules and dynamics of the network. Our goal is to quantify some of these constraints, and design the control system together with the communication system so as both do their best given the constraints. This work in no way attempts to suggest the best way to design a communication network that suits the needs of a particular control system, but some of the results obtained here may be used in conjunction with other results in forming an understanding as to how to proceed in the design of such systems in the future.

The work proposes a novel real-time communication protocol based on the Time Division Multiple Access (TDMA) strategy, which has built-in tolerance against the network-induced effects like lost packets, assuring a highly deterministic and reliable behavior of the overall networked control system, thus allowing the use of classical control design methods with WNCS. Determinism in the transmission times, for sending and for receiving, is assured by a communication schedule that is dynamically updated based on the conditions of the network and the propagation environment.

An advanced experimentation platform has been developed, called WiNC, which demonstrates the efficiency of the protocol with two well-known laboratory benchmarks that have very different dynamics, namely the three-tank system and the inverted pendulum system. Wireless nodes belonging to both systems are coordinated and synchronized by a master node, namely

the controller node.

The WiNC platform, having the main features of a flexible rapid control prototyping system, offers the possibility also to other research groups to experiment, test and optimize new monitoring and fault tolerant control designs for the WNCSs, all without any manual programming. The code corresponding to the Matlab/Simulink models is automatically generated and then integrated into the WiNC application.

The WiNC platform uses only open source software and general-purpose (commercial, off-the-shelf) hardware, thus making it with a minimal investment (low cost) a flexible and easily extendable research platform for WNCS. And considering the general trend towards the adoption of Linux as a real-time operating system for embedded system in automation, the developed concepts and algorithms can be ported with minimum effort to the industrial embedded devices which already run Linux.

Several Fault Diagnosis (FD) methods have been successfully included and tested on the WiNC platform, for both benchmark systems, demonstrating that from theory to practice it is just a small step.

1 Introduction

Progress is the road from the primitive via the complicated to the simple.

K. Biedenkopf

With the revolutionary changes in information and communication technologies from the last years, the roles that embedded computing and networked communication play in control systems engineering are becoming increasingly important. The number of networked embedded devices deployed in practice has been far more than that of various general-purpose computers including, e.g. desktop PCs and laptops. These devices are used in many application areas, including automation, process control and monitoring. In this context, networked embedded control and monitoring systems or, in short, Networked Control Systems (NCSs) have become unprecedentedly popular.

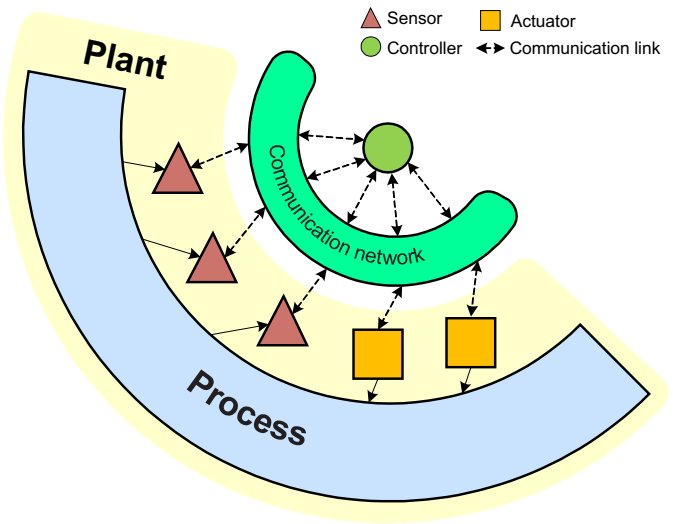


Figure 1.1: Example of a Networked Control System (NCS).

As illustrated in Figure 1.1, in an NCS, several nodes/components of the system may communicate over the common network that connects them together. Thus, there may be communication taking place between the sensors and the controller nodes, among the controllers themselves, among the sensors themselves, among the actuator themselves, and between the controller and the actuator nodes. The purpose of this communication is to improve the performance of the control system. The performance may be a measurable quantity defined in terms of a performance criterion, as in the case of optimal control or estimation, or it may be

a qualitative measure described as a desired behavior. Each node of the NCS may act as a decision maker, making control as well as communication decisions. The presence of a network brings in constraints in the design of the control system, as information between the various decision makers must be exchanged according to the rules and dynamics of the network. Our goal is to quantify some of these constraints, and design the control system together with the communication system so as both do their best given the constraints. This work in no way attempts to suggest the best way to design a communication network that suits the needs of a particular control system, but some of the results obtained here may be used in conjunction with other results in forming an understanding as to how to proceed in the design of such systems in the future.

Even though wireless networks have the potential to help industry use energy and materials more efficiently, lower production costs, and increase productivity, presently only wired networks are employed in the real-time communication infrastructure. The reason is that the industrial applications often have stringent requirements on timing and reliability, which are significantly more difficult to meet by the wireless networks, due to the adverse properties of the radio channels. The properties translate to very high Bit Error Rate (BER) or Packet Loss Rate (PLR), and heavily varying transmission delays (latencies), which negatively affect the control performance and stability of an NCS.

Our research focuses on the integrated design of control/monitoring and communication parts, including deterministic medium access schemes, cross-layer design and robust coding techniques. The key issue is achieving timely and reliable transmissions despite channel errors, and be able to address industrial automation applications belonging to class 1 and class 0 (as established by the ISA-SP100 committee and shown in Figure 1.2). This can be realized either by adapting existing wireless technologies, like Wireless Local Area Network (WLAN), to the requirements of control systems, or by developing from scratch a new customized wireless solution.

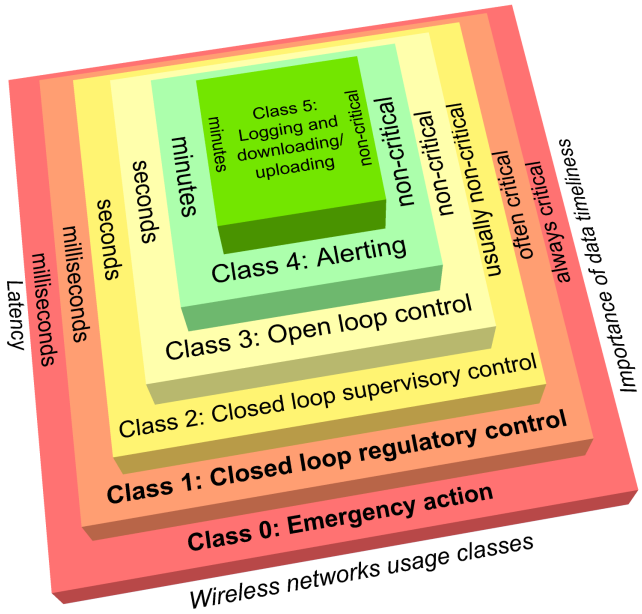


Figure 1.2: Classes of wireless industrial applications, as established by ISA-SP100 Committee.

With the control systems becoming more and more complex from both the communication and computer science perspectives, there is an increasing need for methods and analysis tools that are able to quantify the intricate relationships between the communication/computer design and runtime parameters (e.g. data rate, bit error rate) and the control/monitoring specific parameters (e.g. control performance, false alarm rate, missed detection rate). Parallel to the complexity, also the demand for safer, more reliable and more environmentally friendly systems had grown, which led to the emergence of fault-tolerant control and monitoring systems.

Our society depends strongly upon the availability and correct function of complex technological processes. For example, manufacturing systems consist of many machine tools, robots and transportation systems, all of which have to correctly satisfy their purpose in order to ensure an efficient and high-quality production. Economy and every-day life depend of the function of large power distribution networks and transportation systems, where faults in a single component could have major effects on the availability and performance of the system as a whole [16]. An example are the safety-critical systems, like aircrafts, spacecrafts and satellites, where a fault occurrence can trigger a system failure that can produce loss of human lives and huge economical losses. Mobile communication constitutes another example where networked components interact so heavily that component fault can have far reaching consequences.

1.1 Faults and failures

In the general sense, a *fault* is something that changes the behaviour of a system such that the system does no longer satisfy its purpose. It may be an internal event in the system, which stops the power supply, breaks an information link, or creates a leakage in a pipe. It may be a change in the environmental conditions that causes an ambient temperature increase that eventually stops a chemical reaction or even destroys the reactor. It may be a wrong control action given by the human operator that brings the system out of the required operation point, or it may be an error in the design of the system, which remained undetected until the system comes into a certain operation point where this error reduces the performance considerably. In any case, the fault is the primary cause of changes in the system structure or parameters that eventually leads to a degraded system performance or even the loss of the system function [16].

Broadly speaking, by term *fault* we mean failures, errors, malfunctions or disturbances in the functional units that can lead to undesirable or intolerable behavior of the system [20].

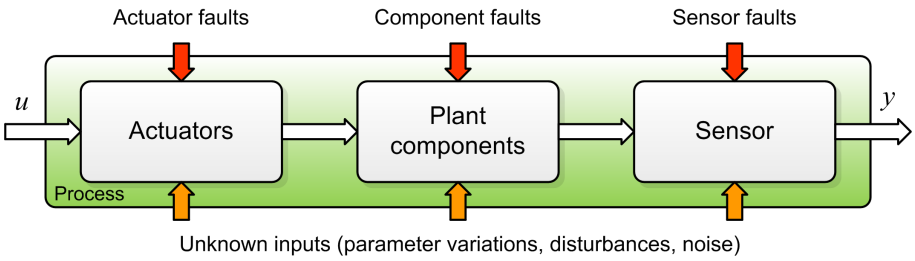
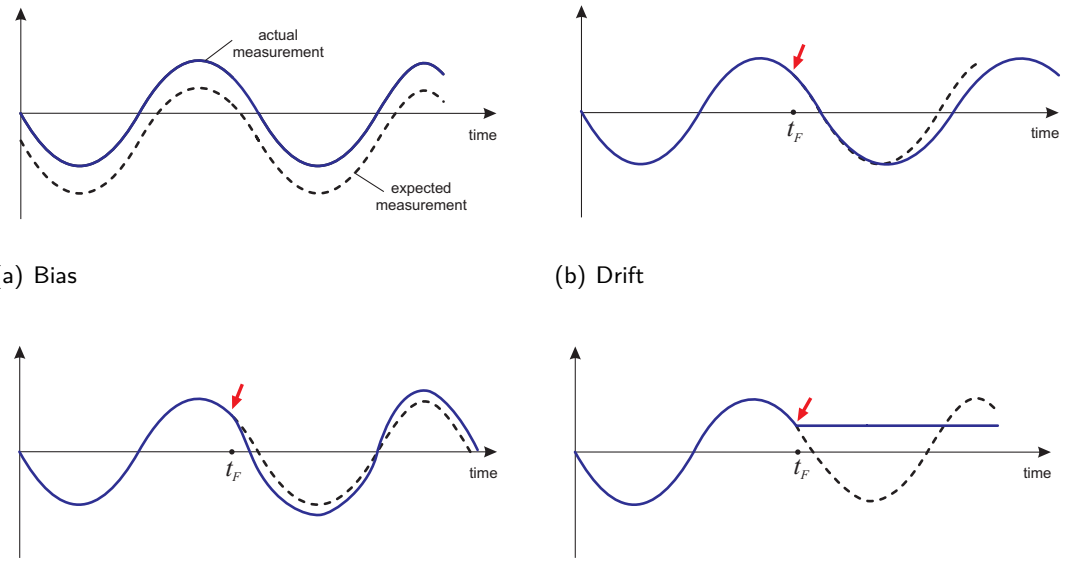


Figure 1.3: Fault types in a control system.

The faults are often classified as follows (see Figure 1.3):

- Component or process faults – such faults change the dynamical I/O properties of the system.
- Sensor faults – sensor readings have substantial errors, but the plant properties are not affected. Common types of sensor faults are shown in Figure 1.4.
- Actuator faults – the influence of the controller on the plant is interrupted or modified, but the plant properties are not affected. Common types of actuator faults are shown in Figure 1.5.



(a) Bias (b) Drift

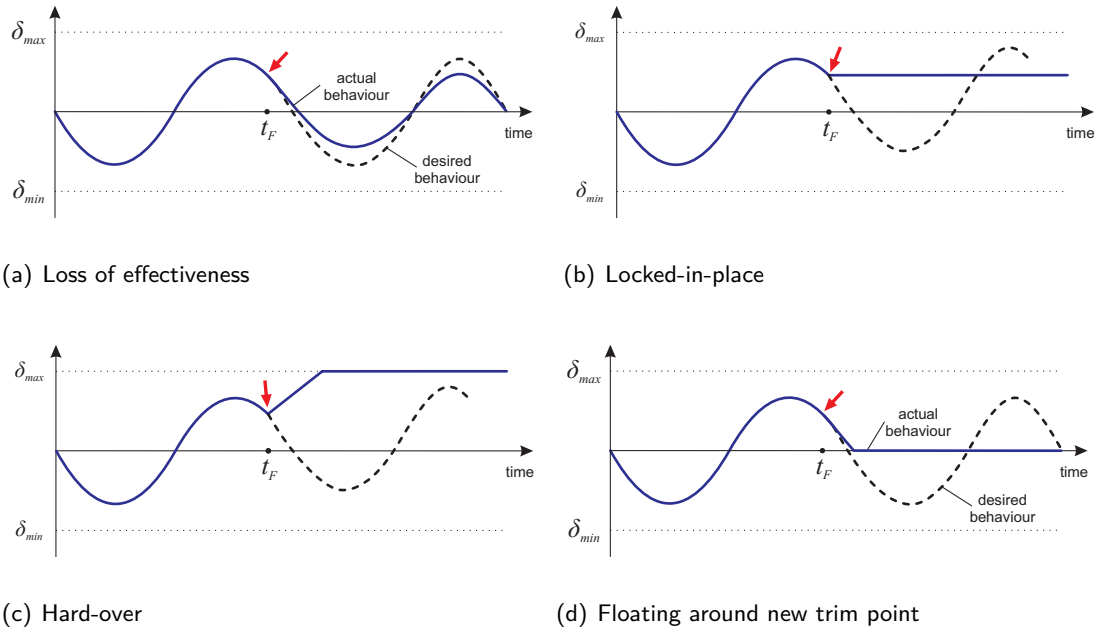
(c) Calibration error or loss of accuracy (d) Freezing

Figure 1.4: Common types of sensor faults [20, 36].

In other words, a fault in a dynamical system is a deviation of the system structure or the system parameters from the nominal situation. It can generally be classified using the following criteria:

- Fault nature
  - value faults – an incorrect value is provided;
  - timing faults – the value is presented outside the specified time interval.
- Fault persistence
  - transient faults – the fault exists only for a short period of time and disappears by itself;
  - permanent faults – the fault exists until an explicit repair action takes place.

A *failure* can be defined as a permanent interruption of a system's ability to perform a required function under specified operating conditions [61]. Resulting from one or more faults, a failure is therefore an event that terminates the functioning of a device/unit in the system.



(a) Loss of effectiveness

(b) Locked-in-place

(c) Hard-over

(d) Floating around new trim point

Figure 1.5: Common types of actuator faults [20, 36].

Considering this classification, we can treat a lost measurement as a transient fault in the sensor, and a lost command can be considered a transient fault in the actuator. Thus, disturbances/faults in the communication network are masked as disturbances/faults in the control system, and so strategies from the Fault-Tolerant Control (FTC) field can be applied, which can handle the above situations and allow to evaluate the stability and performance of the control system.

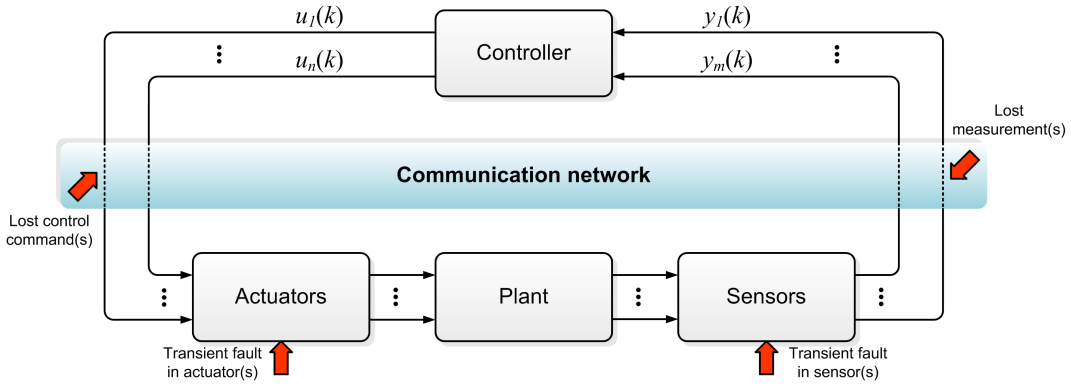


Figure 1.6: Lost measurements and control commands can be treated as transient faults in sensors and actuators, respectively. $u(k) \in \mathbb{R}^n$ denotes the system input vector (the control commands transmitted by the controller to the actuator nodes), and $y(k) \in \mathbb{R}^m$ denotes the system output vector (the measurements sent by the sensor nodes to the controller).

A diagnosis system is capable of identifying the faults of a system by investigating the observable effects (or symptoms). The system is referred to as *intelligent* if it can provide similar diagnosis results as a human expert could do for the same fault. The intelligent system categorizes the fault into a diagnosis class and identifies a probable cause based on the supplied

fault symptoms.
The development of diagnosis classes is done by making use of previous experience, knowledge or information within an application area. The necessary information used may come from several sources of knowledge, such as from system analysis.

1.2 Fault-tolerance

In large-scale systems, every component has been designed to accomplish a certain function and the overall system works satisfactory only if all components provide the service they are designed for. Therefore, a fault in a single component usually changes the performance of the overall system.

In order to avoid production deteriorations or damage to machines and humans, faults have to be found as quickly as possible and decisions that stop the propagation of their effects have to be made. These measures should be carried out by the control equipment. Their aim is to make the system *fault tolerant*. If they are successful, the system function is satisfied also after the appearance of a fault, possibly after a short time of degraded performance. The control algorithm adapts to the faulty plant and the overall system satisfies its function again, albeit with a possible decrease in the performance.

Generally, the way to make a system fault-tolerant consists of two main steps, as shown in Figure 1.7 and described in the following.

1. *Fault diagnosis* – The existence of faults, which break the corresponding control loop and it could render the whole system inoperable, has to be detected and the faults have to be identified. The fault is usually found in three stages:

- Fault detection - determine if a fault has occurred.
- Fault isolation - determine where the fault has occurred.
- Fault estimation - determine how strong the fault is.

2. *Remedial actions* – The control structure has to be reconfigured or adapted to the faulty situation so that the overall system continues to satisfy its goal. Reconfiguration could involve closing the control loop via redundant signal path. The redundant signal path can be an implicit property of the communication network employed within the control system, and can be optimally exploited in the communication protocol.

Both steps must be designed in an integrated way, so that the reliability of the fault-tolerance scheme is kept high. Actually, the most important part remains the fault diagnosis, which basically drives the performance of the reconfiguration step.

There are two main methods for reconfiguration, they are the redesign and fault-hiding. Redesign refers to the controller that is adapted to the faulty situation so that it keeps the plant stable. Since there are no "perfect" diagnostic algorithms, i.e. which can guarantee a zero percent for the False Alarm Rate (FAR) or a hundred percent for the Fault Detection Rate (FDR) or a zero percent for the Missed Detection Rate (MDR), care must be taken in the reconfiguration step. In safety-critical processes, usually the fault detection and isolation are

based on hardware redundancy – voting strategy together with consistency checking of sensor measurements, including plausibility check.

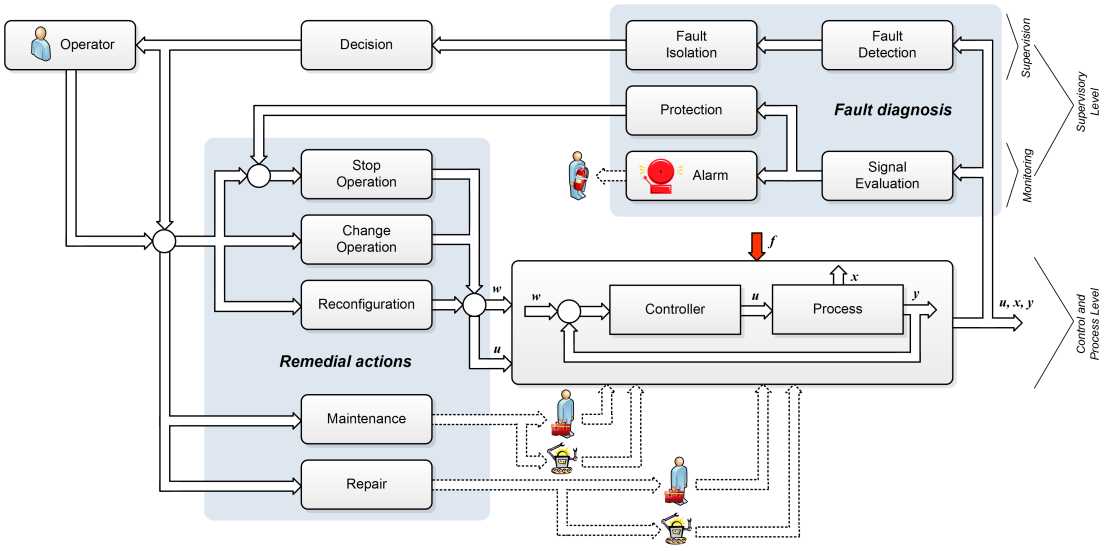


Figure 1.7: Diagram of an Integrated Fault Management System [61]. Signal u represents the control commands, w are the reference signals, y are the sensor measurements, x represents the states of the process, and f symbolizes the faults and disturbances affecting the control system.

An integrated fault management system, as shown in Figure 1.7, has the task of preventing faults or failures from occurrence (preventive action) and preventing their recurrence (corrective action). These steps are not carried out by the usual feedback controller, but by a supervision system that prescribes the control structure and selects the algorithm and parameters of the feedback controller.

Traditional methods for fault diagnosis include limit-checking, spectral analysis of selected signals, which allow for the detection of specific faults. In the case of faults, the controller switches to a redundant component. For example, important elements of an aircraft use this principle with a threefold redundancy. On the other hand, these means for fault tolerance can only be applied to safety-critical or life-critical systems, since for a more general use they are unnecessarily complicated and too expensive for two main reasons. First, the traditional methods for FD assume that for every fault to be detected there is a measurable signal that indicates the presence of a fault by, for example, the violation of a threshold or by changing its spectral properties. In complex systems with many possible faults, such a direct relationship between a fault and an associated signal does not exist or it is too expensive to measure all such signals. Second, this kind of fault tolerance is based on *physical redundancy* or *replication*, where important components are implemented more than once. This is in contradiction with the goal of cost reduction set by the industry, which cannot afford to use this kind of redundancy on a large scale.

Another solution that can enhance the dependability of a system is the *analytical redundancy* or so called model-based FTC, where an explicit mathematical model is used to perform the two steps. The fault is diagnosed by using the information included in the model and in the on-line measurements from the sensors. Then the model is adapted to the faulty situation and the controller is re-designed so that the closed-loop system including the faulty plant satisfies the given specifications.

In conclusion, FTC aims at changing the control law so as to cancel the effects of the faults or to reduce them to an acceptable level.

1.2.1 Model-based fault diagnosis

This technique replaces the hardware redundancy by a process model which is implemented in the software form. A process model is a quantitative or a qualitative description of the process dynamic and steady behavior, which can be obtained using the well-established process modeling techniques. In this way, we are able to reconstruct the process behavior on-line, which, associated with the concept of hardware redundancy concept, is called software redundancy concept or analytical redundancy [33]. In other words, a process model will run in parallel to the real process and it will be driven by the same inputs. The difference between the measured process variables and the estimated ones is called *residual*.

An FD system, as illustrated in Figure 1.8, is in fact a dynamic system whose inputs are the measurements and control input signals of the process under observation, and outputs are represented by the alarm signals (fault information) [6, 33]. This system can be divided into two subsystems: a so called residual generator and a residual evaluator.

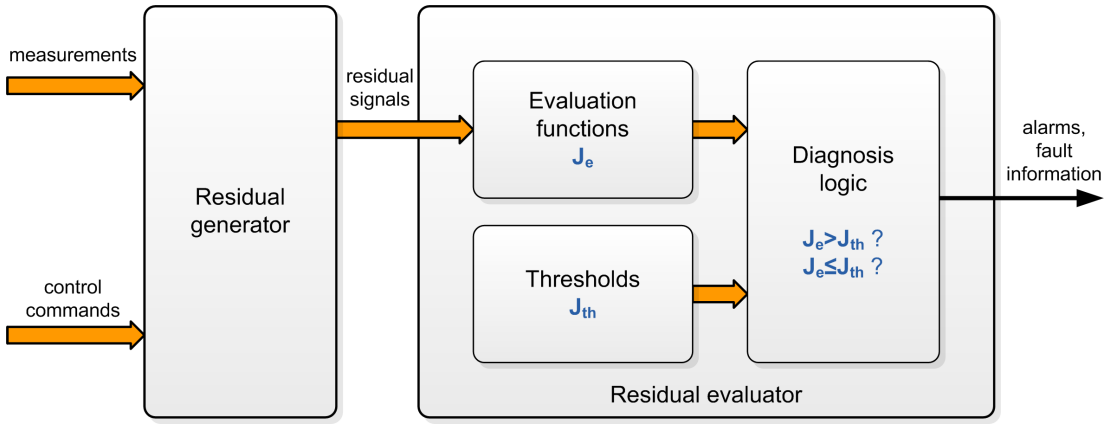


Figure 1.8: Components of a Fault Diagnosis (FD) system.

The procedure of creating the estimates of the process outputs and building the difference between the process outputs and their estimates makes up the *residual generator* module. The procedures of extracting the information about the fault of interests from the residual signals by means of post-processing of the residuals builds up the *residual evaluator*.

Residual evaluation

The peak value of the k_R residual signals is defined by (1.1), where $r(k) \in \mathbb{R}^{k_R}$ and N is a finite time window.

$$J_{peak} = \left(\sum_{i=k-N}^{k} r^T(i) r(i) \right)^{1/2} \tag{1.1}$$

The average value evaluation of the residual is defined by (1.2). It is often used in practice to overcome the noise problem.

$$J_{average} = \frac{1}{N}\left(\sum_{i=k-N}^{k} r(i)\right) \tag{1.2}$$

The root mean square (RMS) value is defined by (1.3). Its physical meaning can be interpreted as a measure of the average energy level of $r(k)$ over a time interval $(k, k+N)$.

$$J_{RMS} = \left(\frac{1}{N}\sum_{i=k-N}^{k} r^T(i)r(i)\right)^{1/2} \tag{1.3}$$

Based on the residual evaluation functions given in 1.1-1.3, the threshold can be realized.

1.2.2 Fault-tolerant control

Conventionally, fault-tolerant control systems are achieved and ensured through hardware redundancy, that is, by including redundant actuators and sensors in the system. The control and measurement channels are generally made duplicated or triplicated in hardware. The main disadvantage of physical redundancy is the additional cost and the corresponding increase in complexity of operation. Moreover, the weight of the system and the maintenance requirements are subsequently increased. Consequently, analytical redundancy approach, which makes use of the mathematical model of the system and relationships between sensor outputs and actuator inputs, has been proposed and are increasingly being employed in complex control systems [100].

FTC describes techniques, classified as shown in Figure 1.9, for adapting control loops to faulty plants by suitable use of the available redundancy. It aims at preventing component faults, component failures or subsystem faults from causing system failures. *Passive FTC*, such as robust control, denotes techniques to let the controller tolerate a set of possible faults. However, the set of faults that can be tolerated without active controller re-adjustment is usually limited. *Active FTC* denotes techniques to achieve fault tolerance by changing the control loop after a fault has occurred. Fault Detection and Isolation (FDI) seeks to find out whether the plant is subjected to a fault and to identify the fault. The fault diagnosis step is followed by a controller adjustment step, called control reconfiguration [16].

Fault-tolerant systems and associated control designs have inherently wide and diverse engineering applications, divided in four main categories as follows [124]

- *safety-critical systems* such as aircraft, helicopters, spacecraft and automobiles, nuclear power and hazardous chemical plants;
- *life-critical systems* such as tele-robots for surgery, implanted heart monitors, nanoscale diagnostic instruments, digital protheses and other medical devices, as well as ground traffic control and automated highway systems;
- *mission-critical systems* such as avionics and air traffic control systems, defense systems, spacecraft and space stations, autonomous aerial/space/underwater vehicles, robots used in industrial processes, and communication networks;

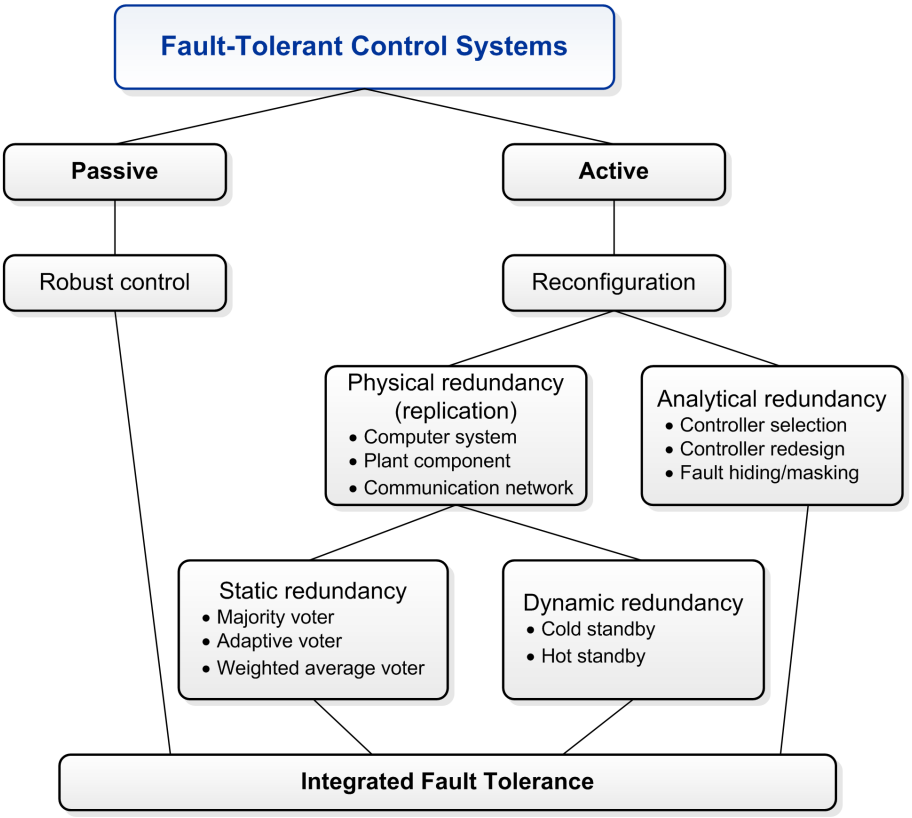


Figure 1.9: Classification of Fault-Tolerant Control Systems.

- *cost-critical systems* such as large-scale space structures, drive-by-wire automobiles, distributed process control, computing and communication networks.

An FTC system has the property that faults do not develop into a failure of the closed-loop system. The system is said to be *fail-operational* when it remains in operation after the occurrence of a fault and its performance stays the same. When its performance decreases, the system is called *fail-graceful* or *fail-passive* [16]. If a fault triggers a situation when the system cannot be kept up, the system is brought into a safe state and switched off. We call it a *fail-safe* system.

1.3 Real-time control systems

A classical real-time control system, represented in Figure 1.10, is composed of four main elements, as follows: the industrial process that must be controlled, one or more controllers, sensors and actuators. The process measurements $y(t)$ are gathered with sensors and sampled $y(k)$ at the controller's side by an A/D convertor. The sampled measurements are then further scaled to physical units and used by the control algorithms for computing the control commands $u(k)$. The control commands are then converted to analog signals and delivered to the actuators.

An NCS, be that wired or wireless, has the functionalities of a classical real-time control system distributed between several computing/network nodes that are interconnected by a

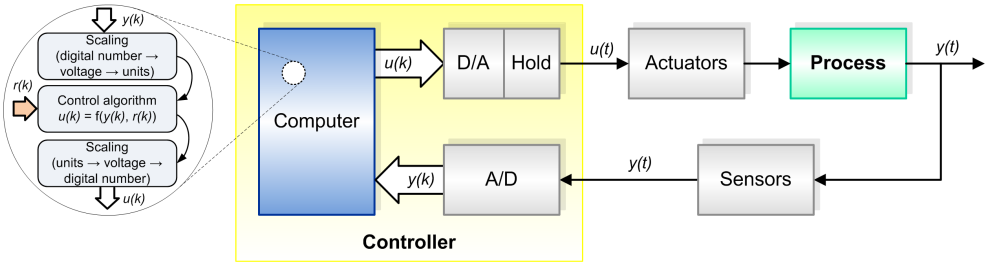


Figure 1.10: A classical real-time control system.

communication network, as shown in Figure 1.11. Compared to a classical real-time control system, in an NCS all the operations previously mentioned are executed on spatially distributed computing nodes, which communicate with each other through a wireless network.

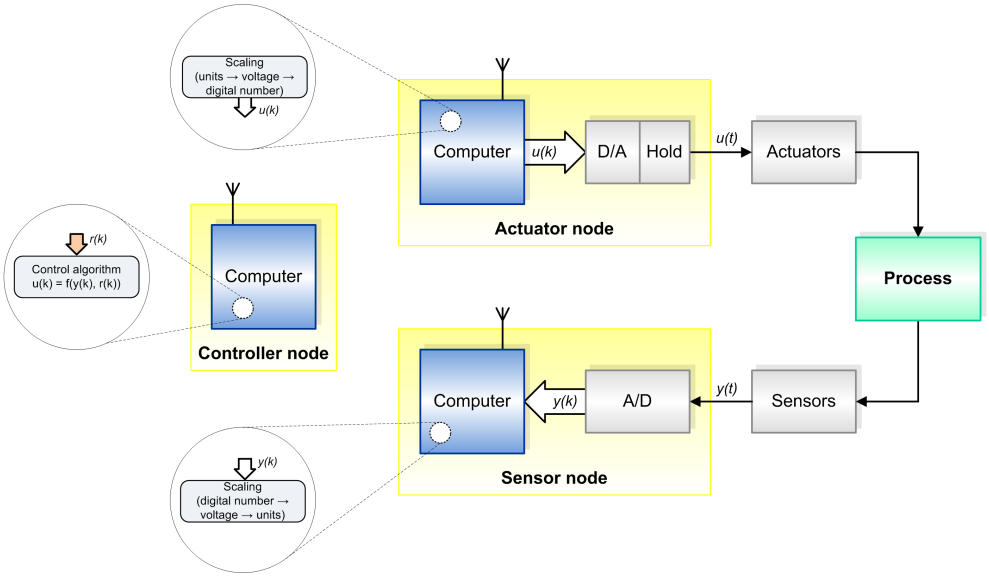


Figure 1.11: Wireless networked control system.

According to DIN 44300 (1972 resp. 1985), real-time operation is the operating mode of a computer system in which the programs for the processing of data arriving from the outside are permanently ready, so that their results will be available within predetermined periods of time. The arrival times of the data can be randomly distributed or be already a priori determined depending on different applications. Thus, real-time systems must keep pace with their environments, in which they are embedded, resulting in the fundamental requirements timeliness, i.e., no tardiness is permitted, *simultaneity*, *predictability*, and *dependability* to hold under worst-case conditions [48]. Real-time systems are (almost) always safety-related, and there is no functional correctness without correct timing. Preconditions for dependable behaviour are functional correctness, robustness, also with respect to inappropriate handling, and permanent readiness. The latter is implied by the definition of real-time operation and calls for fault-tolerance and graceful degradation, i.e., predictable reduction, of performance in cases of error. Thus, predictability of system behaviour is the central concept for the design of real-time systems, since it essentially implies the other three requirements .

1.4 Quality of service

1.4.1 Stability and control performance

Stability is the qualitative property of a system, which can be studied without a direct recourse to solve the differential equations that describe the system. Stability is often the most important property of a dynamical system. In control engineering, one needs to make sure that the closed-loop system is stable (or at least stabilizable) before we can consider the design of a regulatory tracking control device [77].

The performance of a control loop can be measured in several ways. In servo control problems the main task is to follow a desired setpoint trajectory as closely as possible. Here measures associated with the time-domain response to a step change in the setpoint are commonly used, e.g., rise time, peak time, overshoot, and settling time. Alternatively, one can use frequency domain measures such as cross-over frequency, closed-loop bandwidth, amplitude margin, or phase margin.

In regulatory control problems the main task is to keep the controlled variable at a constant value in the presence of disturbances. For these types of control problems variance-type measures are natural. This is often expressed in terms of a quadratic cost function, i.e.

$$J = \lim_{T\to\infty} \int_0^T (x^T(t)Qx(t) + u^T(t)Ru(t))dt \tag{1.4}$$

where $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^p$ the control signal, and Q and R quadratic matrices.

Alternatively one can use as performance index the Integral of the Absolute magnitude of the Error (IAE) [34]

$$IAE = \int_0^T |e(t)|\, dt \tag{1.5}$$

where $e(t) = w(t) - y(t)$ is the system error, $w(t)$ the reference signal and $y(t)$ the system output. The discrete form of the index can be written as

$$IAE = \sum_{i=1}^{N} T_s\, |e(i)| \tag{1.6}$$

where N is a finite time window and T_s is the control or sampling period of the system.

Since the index IAE must be minimized in order to get a better control performance, a more intuitive criterion would be the Quality of Control (QoC) defined as

$$QoC = \frac{1}{IAE} \tag{1.7}$$

and interpreted in this way: the smaller the system errors, the better the quality of control.

The above performance measures mostly concern single control loops. In industrial applications it is more natural to consider application-specific performance measures. These typically concern the performance of the overall system, involving multiple cooperating controllers implemented as one or several tasks. Hence, the performance is related to the goal or mission of the system rather than the performance of the individual loops. For example, in an industrial

robotics control loop the main concern is the position and movement of the robot hand holding some type of tool. The performance of this is a complex function of the performance of the link controllers for each of the robot's individual links [13].

In analyzing a control system, we must examine transient-response behavior and steady-state behavior. The transient response of a control system exhibits damped oscillations before reaching steady state [88]. The common transient-response characteristics of a control system (to a unit step) are as follows and shown graphically in Figure 1.12.

- Delay time, t_d: the time required for the response to reach half of the final value the very first time.
- Rise time, t_r: the time required for the response to rise from 10% to 90%, 5% to 95%, or 0% to 100% of its final value.
- Peak time, t_p: the time required for the response to reach the first peak of the overshoot.
- Maximum (percent) overshoot, M_p: the maximum peak value of the response curve measured from the final steady-state value of the response.
- Settling time, t_s: the time required for the response curve to reach and stay within a range about the final value of size specified by absolute percentage of the final value (usually 2% or 5%).

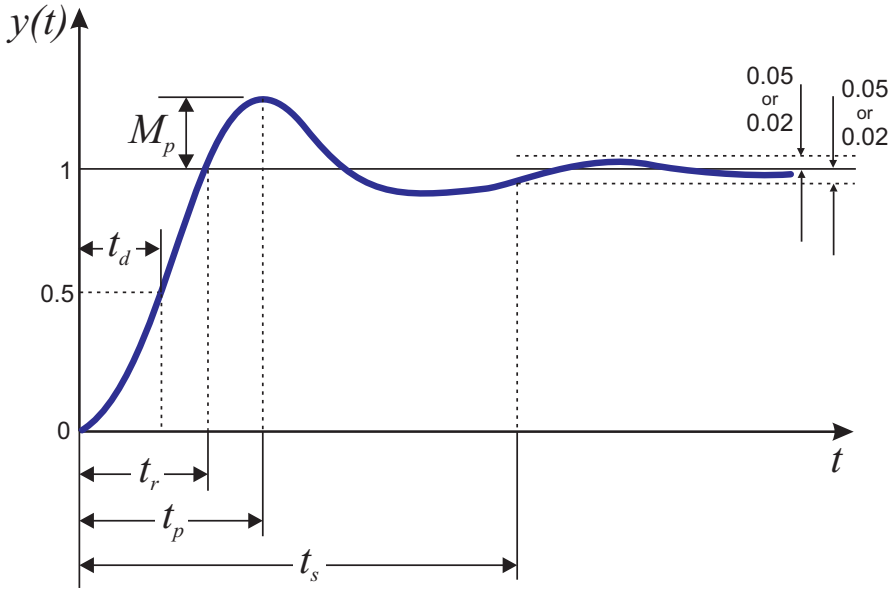


Figure 1.12: Transient-response characteristics of a control system to a unit step input.

1.4.2 Performance measures in FDI

To evaluate the performance of a FDI system, several performance indices have been defined and used, such as Correct Fault Detection (CFD), FAR, Missed Fault Detection (MFD), Incorrect Fault Detection (IFD), and Detection Delays (DD) [77].

The major emphasis in quantitative model-based FDI has been placed upon methods of detecting and isolating faults rapidly and accurately. A good FDI process should posses a certain degree of robustness in its decisions, i.e. the FDI process should have high CFD, and low FAR, MFD, IFD and DD.

A probability based performance index is considered here. The idea is to study the probabilities of the naturally defined events [87]:

- no alarm – not detect a fault given that no fault has occurred,
- *false alarm* – detect a fault given that no fault has occurred,
- positive alarm – detect fault j given that fault j has occurred,
- *missed detection* – not detect a fault j given that fault j has occurred.

These four events can be modeled as a Markov chain. The first and the second event are the complement of each other. The same is true for the third and the fourth event. Therefore, without loss of generality, the considered two performance indices are the probability of the second event – so called FAR, and the fourth event – MDR. The goal would be to minimize the probabilities of these two events.

1.5 Integration of control, communication and computing

The advantages of networking in control systems are so strong that any systems above some minimal level of complexity are likely to utilize networking. However, networking in control has operational requirements that are quite different from networking in a general computing environment. Not only is the selection of network type important, but also the design and configuration of the network are crucial for achieving satisfactory performance.

A *network* in the computational sense is a configuration of computers that share access to a common medium (electrical, optical, radio spectrum/frequency), encapsulate information into packets, and share a common understanding of the meaning of the information (protocol) [53].

A Local Area Network (LAN) uses a single, common medium to connect a set of computer nodes. All of the nodes are energetically connected to each other. As the name implies, LANs cover relatively compact areas. A primary issue in LANs is controlling access to the medium by each of the nodes in order to maintain order, make efficient use of the medium, and to establish some notion of "equity", that is, making sure each computer node has appropriate access to the network.
A Wide Area Network (WAN) is made by interconnecting LANs. The most prominent example of a WAN is the Internet. In the course of traversing a WAN, the information is typically carried over several different physical media along the way.

Traditionally, control algorithms are designed assuming unlimited computing and communication resources. In embedded systems, these resources are often shared between many applications and the environmental conditions are changing. Thus, the development of the real-time control algorithms must be realized through an integration of computing, communication and

control. In this way, a fundamental process of dynamic interaction is established between control systems, communication technology and computer science, which contributes to the overall system performance optimization.

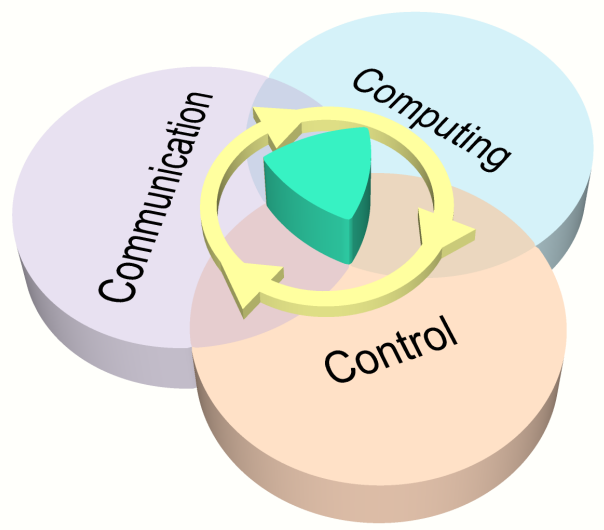


Figure 1.13: Integration of Control, Communication and Computing.

The Medium Access Control (MAC) and the link layers are exposed most to the error characteristics of wireless channels and they should do most of the work to improve the quality and reliability of the wireless channel. Specifically for real time communications the MAC layer is a key component; if the delays on the MAC layer are not bounded, the upper layers cannot compensate for this [13]. In other words, the MAC and link-layer protocol is a key issue for performing hard real-time communications, and if these layers are not able to give tight time- and reliability guarantees (medium access time), this can be hardly corrected by other protocol layers [111].

In wireless media the reliability can be ensured using different mechanisms such as retransmissions, packet duplications, multi-carrier modulation schemes (e.g. Orthogonal Frequency Division Multiplex (OFDM)), spread spectrum modulations (e.g. Direct Sequence Spread Spectrum (DSSS), Frequency-Hopping Spread Spectrum (FHSS)) or error correcting codes (e.g. Forward Error Correction (FEC)) [13].

This page intentionally left blank,
and it could be used for writing notes.

State of the art in Wireless Networked Control Systems

2

The future success of wireless technology rests upon it becoming as overlooked as electricity.

Dean A. Gratton

Wireless technologies in general are not developed with automation application requirements in mind and consequently the focus is shifted more towards developing wireless devices for bigger markets such as home and office. The unprecedented success in the Small Office Home Office (SOHO) markets has led to the emergence of wireless communication in the industrial setting as well. Increasingly more applications are becoming viable, and more industrial wireless products are becoming available.

Application areas for wireless technology in industry have a wide range of requirements and subsequently require more than one wireless solution. Most of them require low data rate and involve sensing of one form or another, that is why the networks supporting them have been called Wireless Sensor Networks (WSNs). WSNs rely now upon distributed sensing and central control, but they will evolve towards the so-called Wireless Sensor and Actuator Networks (WSANs) or more generally WNCSs, where also the control is performed in a distributed fashion by active devices. The main difference between purely sensor networks and sensor-actor networks is the capability of handling real-time critical operations, which could clearly enlarge the application scope of such networks [93]. There are numerous radios and protocols that have relevance to industry including both standards-based and proprietary technologies operating at a number of different frequencies [83].

Due to the general tendency towards standardization and the fact that cheap, commercial SOHO wireless technologies are available, it seems logical to investigate these for their suitability in industrial deployment. Of particular interest for industrial environments are technologies that do not require any sort of frequency licensing, i.e. the Industrial, Scientific and Medical (ISM) band. These technologies include the Wireless Personal Area Network (WPAN) technologies such as IEEE 802.15.1 (Bluetooth) and IEEE 802.15.4 (ZigBee) as well as the WLAN technologies from IEEE 802.11 family [13].

The broad variety of options provides hope that wireless may be effectively applied to any number of connectivity needs by aligning the most appropriate technology with the specific requirements of the application. Understanding the taxonomy of the most prominent technologies requires a quick overview of their major qualities and functions.

2.1 Overview

WLAN, or Wi-Fi, is based on the IEEE 802.11 standard and it was originally designed to provide wireless Ethernet connectivity for computers, eliminating the need for wired connections. Later revisions of the standard have focused on increased data rates (802.11g) and operation at a higher frequency (802.11a) to improve network performance in electrically noisy environments. The high degree of standardisation and low costs of the technology, coupled with broad availability of enabled products and an increasingly sophisticated level of security were factors making it to be adapted for industrial and embedded applications [80].

Bluetooth was originally developed for low-cost wireless short range communication with desktop personal computers and laptops (e.g. wireless keyboard, mouse) as well as for wireless audio transmission with telecommunication devices such as fixed and mobile phones (e.g. wireless headsets).
The specification was developed by the members of the Bluetooth Special Interest Group (SIG) and the lower layers (physical and MAC layers) were introduced into the IEEE 802.15.1 standard. The first version of the standard can be considered as mature, and the number of products for home and office applications is increasing [82]. Bluetooth became interesting also for automation applications in recent years.

ZigBee is a wireless specification controlled by the ZigBee Alliance and it was designed from the ground up for device connectivity and is targeting low cost, low power and low data rate communication mainly for sensor networks. The intended application area includes building automation (heating, lighting, air-conditioning), remote control units, metering and also industrial control systems. It uses the physical and link layers of the IEEE 802.15.4 standard, supporting star and peer-to-peer topologies with so-called superframes that are divided into a CDMA based Contention Access Period (CAP) and a period with contiguous Guaranteed Time Slots (GTSs) [62]. Both ZigBee and IEEE 802.11 have very similar connection ranges and are deployed worldwide in the unlicensed ISM bands at 2.4 GHz and 5 GHz (IEEE 802.11a only) [82]. Additionally IEEE 802.15.4 may also utilize the 868-868.6 MHz and the 905-928 MHz band.

Peer-to-peer topology allows formation of mesh networks that can scale to hundreds and thousands of devices and all will communicate using the best available path for reliable message delivery. ZigBee networks are self-forming and self-healing, meaning that if one path stops working, a new path is automatically discovered and used without stopping the system operation. This mesh networking formation makes it well suited for products which are installed in large numbers throughout a facility. The standard is not viewed as "industrial strength" by most of the large instrumentation vendors and the inconsistency of long battery life with low latency makes a one-size-fits-all approach very difficult [17].

Z-Wave is a proprietary wireless RF-based communications technology designed by Zensys for residential and light commercial control and status reading applications such as meter reading, lighting and appliance control, HVAC, access control, intruder and fire detection, and also entertainment control and digital home healthcare devices [120]. The Z-Wave Alliance is a consortium of leading independent manufacturers who have agreed to build wireless home control products based on the Z-Wave. The competition among ZigBee, Z-Wave and similar mesh networking schemes will lead to lower chip and system prices, which will stimulate the rapid adoption of wireless mesh networking technology overall.

Wireless Interface for Sensors and Actuators (WISA) is a proprietary architecture based on the physical layer of IEEE 802.15.1 and is used by ABB. It was developed for providing real-time communication between several sensors/actuators and a master station. The media access is a tailor-made solution of ABB and combines FHSS and Frequency Division Duplex (FDD) with TDMA. For WISA a maximum message transmission delay of 15 ms is specified [82].

Time Synchronized Mesh Protocol (TSMP) is a proprietary networking protocol pioneered by Dust Networks that forms the foundation of highly reliable, ultra low-power wireless sensor networking. All node-to-node communication in a TSMP network is transacted in a specific time window. The media access method used is TDMA for which a critical component is time synchronization – all nodes must share a common sense of time so that they know precisely when to talk, listen, or sleep. This is especially critical in power-constrained applications like wireless sensors networks where battery power is often the only option and changing batteries can be costly and cumbersome. A common sense of time enables many network virtues: bandwidth can be pre-allocated to ensure extremely reliable transmission and zero self-interference; transmitting nodes can effectively change frequencies on each transmission and the receiving node can keep in lock-step; bandwidth can be added and removed at will in a very predictable and methodical way to accommodate traffic spikes [95].

Ultra-Wideband (UWB) wireless technology is capable of transmitting data at very high data rates at short range, over a wide frequency spectrum, with very low power consumption and it has applications in radar imaging, precision positioning and tracking technology. UWB technology's unique characteristics make it a more suitable solution than narrow-band wireless systems (such as Bluetooth and IEEE 802.11a/b/g) for meeting wireless needs in the industrial environment. Since it is still developing, several issues remain unresolved, especially for industrial applications [49].

Despite the advantages a single wireless network might offer on the factory floor, it will be often required to run multiple wireless networks in parallel in different or overlapping regions of the plant. Because of this fact, the *coexistence* of multiple networks of either the same or varying types needs to be assured [114].

Beyond the necessity of ensuring coexistence, *interoperability* also becomes an issue as wireless finds use as extension to legacy wired bus protocols – *hybrid wired/wireless* networks. For example, the HART Communications Foundation (HCF) is highly focused on delivering a wireless specification which will allow wireless nodes to act as any HART enabled device. More broadly, the SP100 committee within ISA is working to develop security and policy definitions for industrial technologies for application in the sensor and instrumentation realm. The risk is that such an overarching framework is often a terrific proposition but intractably difficult to develop through a standards-based process. This further highlights the issue of *security*, widely seen as the most significant barrier to industrial wireless adoption [80].

2.2 Standards and their organizations

Industry standards are critical to the broad adoption of wireless networking technology.

The **ISA SP-100** committee is tasked to develop standards and best practices for wireless in

industrial automation applications.

WirelessHART developed by HART Communications Foundation (HCF) is the first open and interoperable wireless communication standard designed to address the critical needs of the process industry for reliable, robust and secure wireless communication in real world industrial plant applications. A backward-compatible extension to the HART Protocol, WirelessHART provides the same user experience, ease-of-use, flexibility and friendliness that users expect from HART-enabled products. The HART 7 specifications protect the globally installed base of an estimated 25 million HART devices and support the entire range of HART-enabled measurement, control and automation systems products, both wired and wireless [50].

WINA (Wireless Industrial Networking Alliance) is a coalition of industrial end-user companies, technology suppliers, industry organizations, software developers, system integrators, and others interested in the advancement of wireless solutions for industry [115].

The IEEE leads the way in developing open, leading-edge consensus standards for Wireless Local Area Networks (Wireless LANs), Wireless Personal Area Networks (Wireless PANs), and Wireless Metropolitan Area Networks (Wireless MANs) [125].

In the context of wireless industrial communication there are many interesting solutions and standards as following

- Wireless telecommunication standards like GSM, GPRS, EDGE, UMTS, DECT (cordless phones);
- Lower Layer standards (IEEE 802.11 and 802.15) as basis for wireless local networks (WLAN, Pico networks, sensor and actuator networks);
- Higher Layer standards (application layers above 802.15.4, e.g. IEEE 802.11, Bluetooth and Zigbee);
- Proprietary Protocols as for example WISA (Wireless Interface for Sensors and Actuators) used by ABB;
- The coming radio communication technologies like UWB (Ultra Wideband).

As more companies begin looking at wireless sensor and control networks there is a growing awareness that the wireless protocols fall short in key areas such as peer-to-peer control, local field device access and others, and thus the ISA's SP100 committee done the first formal study in this direction.

The ISA-SP100 committee was formed to help establish a standard for industrial wireless networking that will permit easy specification and use of wireless sensors and controls in process automation. One of the group's early projects has been to outline the key requirements that a wireless protocol must have in order to effectively support industrial applications such as monitoring and control. Members of the committee have begun releasing preliminary results from their research, and it is the first study that identifies each of the key requirements that a wireless protocol needs to meet in order to properly support industrial automation and control; and provides quantitative analysis of how well (or poorly) each of the leading protocols do in meeting those requirements. The committee also proposed a classification of inter-device industrial wireless communications as depicted in Figure 2.1 [59].

Category	Class	Application	Description
Safety	0	Emergency action	(always critical)
Control	1	Closed loop regulatory control	(often critical)
	2	Closed loop supervisory control	(usually non-critical)
	3	Open loop control	(human in the loop)
Monitoring	4	Alerting	Short-term operational consequence (e.g., event-based maintenance)
	5	Logging and downloading/uploading	No immediate operational consequence (e.g., history collection, sequence-of-events, preventive maintenance)

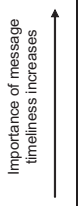


Figure 2.1: ISA-SP100 Usage classes for wireless communication.

Additionally, ISA-SP100 has identified examples of the following wireless automation classes [59]:

- Class 5: Monitoring without immediate operational consequences
  This class includes items without strong timeliness requirements. Some, like sequence-of-events logs, require high reliability; others, like reports of slowly-changing information of low economic value, need not be so reliable since loss of a few consecutive samples may be unimportant.
- Class 4: Monitoring with short-term operational consequences
  This class includes high-limit and low-limit alarms and other information that might instigate further checking or dispatch of a maintenance technician. Timeliness for this class of information is typically low (slow), measured in minutes or even hours.
- Class 3: Open loop control
  This class includes actions where an operator, rather than a machine, "closes the loop" between input and output. Such actions could include taking a unit offline when conditions so indicate. Timeliness for this class of action is human scale, measured in seconds to minutes.
- Class 2: Closed loop supervisory control
  This class of closed-loop control usually has long time constants, with timeliness of communications measured in seconds to minutes. Examples are batch unit and equipment selection.
- Class 1: Closed loop regulatory control
  This class includes motor and axis control as well as primary flow and pressure control.
- Class 0: Emergency action
  This class includes safety-related actions that are critical to both personnel and plant. Most safety functions are, and will be, performed through dedicated wired networks to limit both failure modes and susceptibility to external events or attack. Examples are safety interlock, emergency shutdown, and fire control.

2.3 Benefits and requirements of wireless communication

The three main benefits of deploying wireless communication in automation industry are summarized in the following:

- Increased robustness (reduced maintenance costs): no cable wear and tear and no connector failure;
- Increased (location) flexibility: no mechanical design limitations, allows mobile/moving/rotating production equipment, fast commissioning and reconfiguration;
- Reduced system cost: reduced wiring, terminals, I/O blocks, cabinets, reduced design and installation work, easier expansion and relocation, reduced maintenance [60].

The general requirements of an wireless communication system for networked control are summed up in the following list, and depending on the area of application, requirements might be added or reduced.

- Node density (scalability): hundreds within one cell, thousands within a factory;
- Bandwidth: moderate bit rate;
- Latency: bounded and deterministic transmission delays;
- Power consumption: low consumption to enable wireless powering;
- Operating frequency: license-free bands above 1 GHz or 860/905 MHz;
- Interference: industrial environment, electromagnetic interference (EMI), other wireless systems.

2.4 Wireless communication issues

Along with the benefits of using wireless technology come several issues that must be considered in the design phase and also during deployment.

In realistic environments, the wireless channel suffers from phenomena like path loss and shadowing, frequency selectivity (multipath propagation) and thermal noise. On the digital level, these phenomena can lead to bit errors or even to the total loss of packets due to a receivers failure to acquire carrier or symbol synchronization. The precise error characteristics depend on the specific wireless technology, the carrier frequency, the distance and the environment between transmitter and receiver, and other factors. Measurements of the error characteristics in industrial environments have been presented in a number of studies and for different types of wireless technologieslike Bluetooth, IEEE 802.11b, IEEE 802.11a or IEEE 802.15.4 [113].

The wireless channel is often time-varying, with variations caused by mobility of wireless stations or in the propagation environment. The signal strength at the receiver can vary over time, thus creating a fading channel. The channel variations can happen on different timescales.

Fast fading is caused by a combination of mobility and multipath propagation, since movements give rise to quick changes in the mutual interference created at the receiver by differently delayed and phase-shifted copies of the same transmitted waveform. Fast fading typically happens on timescales of milliseconds to tens of milliseconds. On the other hand, when mobility leads to situations where obstacles are placed or removed from wave propagation paths between transmitter and receiver, additional attenuation results, also referred to as *shadowing*. The timescales involved are typically on the order of seconds to tens of seconds. Furthermore, when the distance between transmitter and receiver changes significantly, the attenuation coming from path loss can also change significantly. Given pedestrian-speed mobility, this often happens on the timescale of tens of seconds to minutes. This type of time-variation is also referred to as *slow fading* [113].

Another very important property of practical wireless channels is their *location-dependency*. With multipath propagation even a slight movement of the receiver suffices to change the absolute value and the statistics of the received signal strength.

In summary, the wireless channel can be considered location-dependent, time-varying and random, and often significant error rates can be observed [113].

2.5 Channel access mechanisms

MAC protocols solve a seemingly simple task: they coordinate the times where a number of nodes access a shared communication medium. A large number of protocols have emerged in more than thirty years of research in this area. They differ, among others, in the types of media they use and in the performance requirements for which they are optimized [62].

MAC protocol is the first protocol layer above the Physical Layer (PHY) and consequently MAC protocols are heavily influenced by its properties. The fundamental task of any MAC protocol is to regulate the access of a number of nodes to a shared medium in such a way that certain application-dependent performance requirements are satisfied. Some of the traditional performance criteria are delay, throughput, fairness, energy conservation and determinism.

Within the OSI reference model, as presented in Figure 2.6, the MAC is considered as a part of the Data Link Layer (DLL), but there is a clear division of work between the MAC and the remaining parts of the DLL. The MAC protocol determines for a node the points in time when it accesses the medium to try to transmit a data, control, or management packet to another node (unicast) or to a set of nodes (multicast, broadcast). Two important responsibilities of the remaining parts of the DLL are error control and flow control. Error control is used to ensure correctness of transmission and to take appropriate actions in case of transmission errors and flow control regulates the rate of transmission to protect a slow receiver from being overwhelmed with data.

Traditionally, the most important performance requirements for MAC protocols are throughput efficiency, stability, fairness, low access delay (time between packet arrival and first attempt to transmit it), and low transmission delay (time between packet arrival and successful delivery), as well as a low overhead. The overhead in MAC protocols can result from per-packet overhead (MAC headers and trailers), collisions, or from exchange of extra control packets. Collisions can happen if the MAC protocol allows two or more nodes to send packets at the same time.

Collisions can result in the inability of the receiver to decode a packet correctly, causing the upper layers to perform a retransmission. For time-critical applications, it is important to provide deterministic or stochastic guarantees on delivery time or minimal available data rate [62].

The operation and performance of MAC protocols is heavily influenced by the properties of the underlying physical layer. Since WNCSs use a wireless medium, they inherit all the well-known problems of wireless transmission. One problem is time-variable, and sometimes quite high, error rates, which is caused by physical phenomena like slow and fast fading, path loss, attenuation, and man-made or thermal noise. Depending on modulation schemes, frequencies, distance between transmitter and receiver, and the propagation environment, instantaneous bit error rates in the range of $10^{-3} \ldots 10^{-2}$ can easily be observed.

The main job of the MAC (Medium Access Control) protocol is to regulate the usage of the medium, and this is done through a channel access mechanism. A channel access mechanism is a way to divide the main resource between nodes, the radio channel, by regulating the use of it. It tells each node when it can transmit and when it is expected to receive data. The channel access mechanism is the core of the MAC protocol and it can be classified in four main types: TDMA, Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA) and Space Division Multiple Access (SDMA).

TDMA

A specific node, the base station, has the responsibility to coordinate the nodes of the network. The time on the channel is divided into time slots, which are generally of fixed size. Each node of the network is allocated a certain number of slots where it can transmit. Slots are usually organised in a frame, which is repeated on a regular basis.

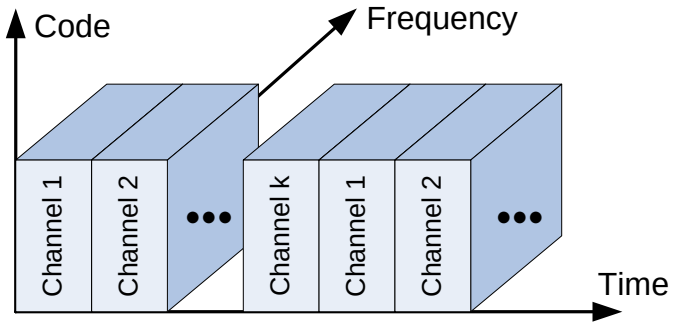


Figure 2.2: Time-division multiple access (TDMA)

The base station specify in the beacon (a management frame) the organisation of the frame. Each node just needs to follow blindly the instruction of the base station. Very often, the frame is organised as downlink (base station to node) and uplink (node to base station) slots, and all the communications goes through the base station. A service slot allows a node to request the allocation of a connection, by sending a connection request message in it (see chapter 5.2.4). In some standards, uplink and downlink frames are one different frequencies, and the service slots might also be a separate channel.

TDMA suits very well phone applications, because they have very predictable needs (fixed and identical bit rate). Each handset is allocated a downlink and a uplink slot of a fixed size (the size of the voice data for the duration of the frame). This is no surprise why TDMA is used into all cellular phone standards (Global System for Mobile Communication (GSM) in Europe, Personal Communications Service (PCS) in Canada, Mexico and the USA) and cordless phone standards (Digital Enhanced Cordless Telecommunication (DECT) in Europe). TDMA is also very good to achieve low latency and guarantee of bandwidth.

TDMA is not well suited for data networking applications, because it is very strict and inflexible. IP is connectionless and generates bursty traffic which is very unpredictable by nature, while TDMA is connection oriented (so it has to suffer the overhead of creating connections for single IP packets). TDMA uses fixed size packets and usually symmetrical link, which doesn't suit IP that well (variable size packets).

TDMA is very much dependant of the quality of the frequency band. In a dedicated clean band, as it is the case for cellular phone standard, TDMA is fine. But, because of its inflexibility, and because it doesn't really take care of what is happening on the channel, TDMA can not cope and adapt to the bursty interference sources found in the unlicensed bands, unless a retry mechanism is put on top of it [89].

CSMA/CA

CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) is the channel access mechanism used by most wireless LANs in the ISM bands.

The basic principles of CSMA/CA are listen before talk and contention. This is an asynchronous message passing mechanism (connectionless), delivering a best effort service, but no bandwidth and latency guarantee. It's main advantages are that it is suited for network protocols such as TCP/IP, adapts quite well with the variable condition of traffic and is quite robust against interferences.

CSMA/CA is derived from CSMA/CD (Collision Detection), which is the base of Ethernet. The main difference is the collision avoidance: on a wire, the transceiver has the ability to listen while transmitting and so to detect collisions (with a wire all transmissions have approximately the same strength). But, even if a radio node could listen on the channel while transmitting, the strength of its own transmissions would mask all other signals on the air. Therefore, the protocol can not directly detect collisions like with Ethernet and it only tries to avoid them.

The protocol starts by listening on the channel (this is called carrier sensing), and if it is found to be idle, it sends the first packet in the transmit queue. If it is busy (either another node transmission or interference), the node waits the end of the current transmission and then starts the contention (wait a random amount of time). When its contention timer expires, if the channel is still idle, the node sends the packet. The node having chosen the shortest contention delay wins and transmits its packet. The other nodes just wait for the next contention (at the end of this packet). Because the contention is a random number and done for every packet, each node is given an equal chance to access the channel (on average - it is statistic).

As we have mentioned, we can not detect collisions on the radio, and because the radio needs time to switch from receive to transmit, this contention is usually slotted (a transmission may start only at the beginning of a slot: 50 μs in 802.11 FH and 20 μs in 802.11 DS). This makes

the average contention delay larger, but reduces significantly the collisions [89].

FDMA

In FDMA, the available frequency band is subdivided into a number of subchannels and these are assigned to nodes, which can transmit exclusively on their channel, as depicted in Figure 2.3. This scheme requires frequency synchronization, relatively narrowband filters, and the ability of a receiver to tune to the channel used by a transmitter. Accordingly, an FDMA transceiver tends to be more complex than a TDMA or CDMA transceiver.

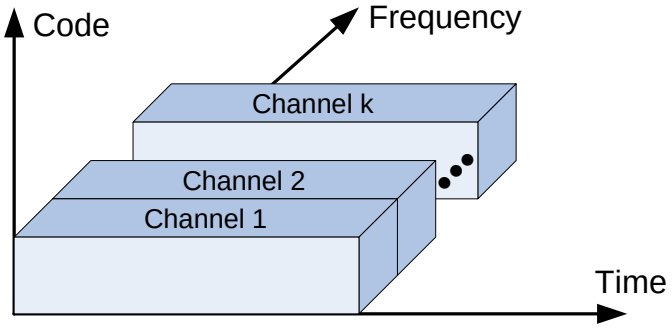


Figure 2.3: Frequency-division multiple access (FDMA)

The channels often have guard bands between them to compensate for imperfect filters, adjacent channel interference, and spectral spreading due to Doppler effect [43].

CDMA

In CDMA schemes, the nodes spread their signals over a much larger bandwidth than needed, using different codes to separate their transmissions, as depicted in Figure 2.4. The receiver has to know the code used by the transmitter; all parallel transmissions using other codes appear as noise. Crucial to CDMA is the code management.

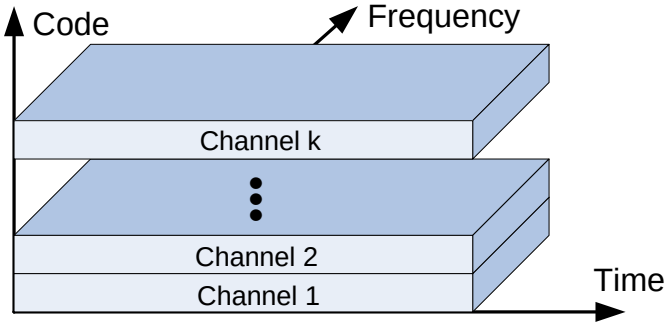


Figure 2.4: Code-division multiple access (CDMA)

CDMA has advantages compared with TDMA and FDMA, especially with respect to flexibility,

while it also has some drawbacks, like complexity. It is thus obvious to combine CDMA with other multiaccess methods in order to obtain the "best of both worlds".

SDMA

In SDMA, the spatial separation of nodes is used to separate their transmissions. In other words, SDMA uses direction (angle) as another dimension in signal space, which can be channelized and assigned to different users. This is generally realized with directional antennas, as depicted in Figure 2.5. SDMA requires arrays of antennas and sophisticated signal processing techniques and cannot be considered a candidate technology for WNCSs.

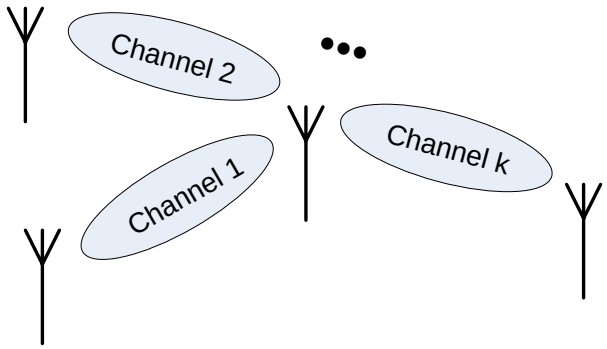


Figure 2.5: Space-division multiple access (SDMA)

In practice, SDMA is often implemented using sectorized antenna arrays, in which the 360° angular range is divided into a number of sectors. There is high directional gain in each sector and little interference between sectors. Either FDMA or TDMA is used to channelize users within a sector. For mobile nodes, the node must be handed off to a new sector when it moves out of its original sector [43].

2.6 Modeling of faults induced by the communication network

In a network, link failures cause the information flow between the controller and the plant to be disrupted, which results in command and/or measurement packets being lost. Packets may also be lost due to congestion. Link failures, on the other hand, may occur due to the unreliable nature of the links, such as in the case of wireless networks. Whatever the reason, this disruption of communication has a deteriorating effect on the networked control system performance. Therefore, it is important to develop an understanding of how much loss the control system can tolerate before the system becomes unstable, or in the case of estimation before the estimation error becomes unbounded.

If the statistical description of the link failure process is given a priori, a problem of interest is to determine the optimal control and estimation policies under the link failure constraints [58].

Packet losses may occur on the way from sensor to controller, and on the way from controller

to actuator. We will focus next on the latter case.

One has to decide what happens when an actuation packet is not received by actuator within a given time. There are two potential actions for the actuator in this case:

- First one is to apply "zero control" and can be justified by observing that zero control would cost the least amount of energy among all possible actions.
- Second one is to apply the "latest available command", which is equivalent to the zero-order hold (ZOH) action in continuous time.

The unreliable nature of the links are modeled by a Bernoulli process, where links fail or packets are lost. The plant is described by the following discrete-time dynamics

$$x(k+1) = Ax(k) + \alpha(k)Bu(k),\ k = 0, 1, ... \tag{2.1}$$

where $x(k) \in \mathbb{R}^n$ denotes the system state vector, and $u(k) \in \mathbb{R}^p$ denotes the system input vector. Matrices A and B are real constant matrices with appropriate dimensions. The stochastic process $\{\alpha(k)\}$ models the unreliable nature of the link from the controller to actuator. Basically, $\alpha(k) = 0$ when this link fails, i.e. the control packet is lost, and $\alpha(k) = 1$, otherwise. This definition corresponds to the "zero control" action by the actuator. We let $\{\alpha(k)\}$ be an independent and identically (i.i.d.) Bernoulli process with

$$P[\alpha(k) = 0] = \alpha,\ P[\alpha(k) = 1] = 1 - \alpha \tag{2.2}$$

The links from sensors to controller are prone to faults as well, but with a different probability (we assume that $\{\alpha(k)\}$ and $\{\beta(k)\}$ are also independent). Thus, controller has access to states intermittently and the measurement equation is

$$y(k) = \beta(k)Cx(k),\ k = 0, 1, ... \tag{2.3}$$

where $y(k) \in \mathbb{R}^m$ denotes the system output vector. Matrix C is a real constant matrix with appropriate dimensions. Here the process $\beta(k)$ is an i.i.d. Bernoulli process with parameter β, i.e.

$$P[\beta(k) = 0] = \beta,\ P[\beta(k) = 1] = 1 - \beta \tag{2.4}$$

2.7 Cross-layer design

Building a distributed control system supported by a wireless network is a challenging task that requires a new design approach to both systems. Control systems and communication networks are typically designed using very different principles. Traditional control theory requires the feedback data to be accurate, timely and lossless. Conversely, random delay and packet loss are generally accepted in communication network design. Moreover, this delay and loss is much more pronounced in wireless networks than in wired networks due to limited spectrum and power, time-varying channel gains, and interference [72].

Layering in network technology separates functional elements of the network into layers and then very carefully defines how each layer interacts with the layers above and below it. The

classic work on layered design of networks is the Open Systems Interconnection (OSI) model from the International Organization for Standardization (ISO). Although this model is not used directly in commercial networking software, it forms the basis for design and construction of many networking technologies [53].

As with any modular scheme, however, a major limitation is that any optimization that crosses layers cannot be implemented without breaking the layered model. These modules, which supply basic functionality, are called *layers* in network technology because information goes through them in sequence. Information that starts in an application goes down through the layers in the originating computer until it reaches the lowest level, where it enters the network and is transmitted to the target computer. The information then goes up through the layers until it gets to the application on the target computer that needs the data. Implementations of layered network systems are generally called *stacks* for that reason.

The full layered model is only required for WANs. Not all of the middle layers are required for LANs because each node on a local network is physically connected to all other nodes and so sees all traffic on the network. It only reacts to packets intended for it, ignoring the others. A network technology intended only for local area use can thus be simpler than one intended for both local and wide area use.
Even if the solution is presented in a layered manner, the actual implementation may not follow the layering. In fact, in many cases, for efficiency reasons, layering is partially violated in the implementation. This violation of design modularity principle is not uncommon in other areas of system design where, as in control systems, performance is extremely important. However, it should not be done lightly because portability, interoperability and maintainability are often sacrificed in order to achieve improved performance [53].

Current WSN and WSAN protocol designs are largely based on layered approach. However, the suboptimality and inflexibility of this paradigm result in poor performance for WSANs, due to constraints of low latency and low energy consumption. Therefore, instead of having individual layers, we may need cross-layering design where layers are integrated with each other [5].

In distributed control systems, the network design objective should be to optimize the control performance. Furthermore, there is a tradeoff between communication and controller performance. From the control perspective, the more the controller knows about the system, the better the control performance is. This can be done by increasing the number of sensors or sending sensor measurements more frequently. However, this increases the communication burden on the network and the network may become congested. The congestion results in longer delays and more packet losses, which degrade the control performance. Therefore, a *joint design of the network and control* is necessary. Joint design of control and communication is two-fold: the controller design needs to be robust and adaptive to the communication faults such as random delays and packet losses, while the network should be designed with the goal of optimizing the end-to-end control performance. Joint design of control and communication has received little attention [72]. A *cross-layer framework* for the joint control and communication problem allows each layer of the network protocol stack to be optimized relative to the end-to-end controller performance.

The general design rule is thus that time critical operations in networked control systems are usually confined to well-designed LANs. Wide area usage in control is usually limited to obtaining diagnostic information, very high level supervisory access, etc [53].

The layered architecture is central to data network design. Layering provides the network design with modularity that facilitates standardization and implementation. An international standard of an OSI model includes seven layers from top to bottom as in Figure 2.6.

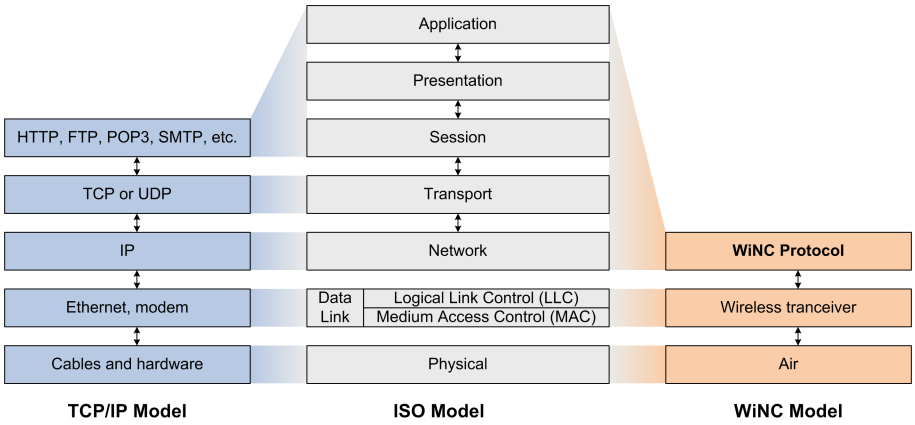


Figure 2.6: OSI layers of a communication protocol.

Traditionally, each layer is designed separately with control messages passing between adjacent layers. The idea of cross-layer design is to jointly design these different layers. Cross-layer design can imply a joint design across all network layers simultaneously, which is highly complex. Alternatively, it can entail choosing parameters or protocols at different network layers from existing designs in a joint fashion. The goal of cross-layer design is to provide the best possible end-to-end performance of the application. Application examples include voice, web browsing, email, etc. In a joint control and network design, the application is control [72]. This kind of optimization is very complex since the performance would be a complicated function of the packet delay distribution, the probability of packet loss and the data resolution associated with the network, thus the effort should be invested in assuring a bounded network delay, i.e. a bounded layer crossing times.

2.8 Determinism in industrial communication

A protocol is a well-defined set of rules for interactions between entities. In networked systems, protocols operate on each network layer. The advantage of using a protocol at any given layer is that only the interaction with the partner network node needs to be considered – from a functional point of view, the other layers are completely transparent. The performance will depend entirely on how the other layers are implemented but not the function [53].

Using networking for control systems requires either using existing protocols or writing new ones. Most protocols written for control systems are at the highest layer - the application layer. Designing a protocol involves all of the usual problems of software design, with the addition of interactions with other computers, lost information and delays [53].

Sensor and actuator data will traverse wired (e.g., optical, twisted-copper) and wireless networks in which bit errors, packet loss, and delay could adversely affect the quality and timeliness of the ultimate result. The goal then is to choose a network and processing architecture to ameliorate the deleterious effects of data loss and network delay in the data collection/fusion process. Due to the costs associated with developing, deploying, and maintaining a fixed ter-

restrial infrastructure, as well as inventing wholly new modulation methods and standards for wireless and terrestrial signaling, it is cost-effective and expedient for the automation technology to ride the "commercial wave" of technical investment and progress in communication technologies.

With a fixed network infrastructure consisting primarily of commercial components, combating data loss and delay in terrestrial networks involves choosing the right protocols so that the network can enforce Quality of Service (QoS) demands; in wireless networks, this involves coding, modulation, and "lightweight" flow control for efficient bandwidth utilization. With sufficient complexity and bandwidth, it is possible with today's IP-based protocols to differentiate high-priority data to impart the mandated QoS for time-critical applications [56].

Unlike terrestrial networks, flow control and routing in mobile wireless sensor networks must contend with potentially long point-to-point propagation delays (e.g., satellite to ground) as well as a constantly changing topology. In a traditional terrestrial network employing link-state routing (e.g., OSPF), each node maintains a consistent view of a (primarily) fixed network topology so that a shortest path algorithm can be used to find desirable routes from source to destination. This requires that nodes gather network connectivity information from other routers [56].

Traditional flow control mechanisms over terrestrial networks that deliver reliable transport (e.g., TCP) may be inappropriate for wireless networks because, unlike wireless networks, terrestrial networks generally have a very low BER on the order of 10^{-10}, so errors are primarily due to overfull queues. Packet loss occurs in heavily congested networks when an ingress or egress queue of a switch or router begins to fill, requiring that some packets in the queue be discarded. This condition is detected when acknowledgments from the destination node are not received by the source, prompting the source's flow control to throttle back the packet transmit rate. In a wireless network in which BERs are four to five orders of magnitude higher than those of terrestrial networks, packet loss due to bit errors can be mistakenly associated with network congestion, and source flow control will mistakenly reduce the transmit rate of outgoing packets [56].

A great deal of press has been generated in the past several years about real-time operating systems; however, the distinction between soft real-time and hard real-time operating systems is seldom discussed. Hard real-time systems guarantee the completion of tasks in a deterministic time period, while soft real-time systems give priority to critical tasks over other tasks but do not guarantee the completion of tasks in a deterministic time period [56].

2.8.1 Common constraints

- Notification of errors: errors may occur in transfers, and usually an automatic retransmission is triggered until a certain maximum number of transmissions is reached. Whether the protocol uses retries or not, the application layer will be notified about the errors and decide what to do based on the network protocols.
- Temporal consistency: this is realized with a synchronization message sent by multicasting to all network nodes by coordinator.
- Event ordering: is provided with the help of a sequence identification number, which is actually the current network cycle.

- Amenability to analysis: given the requirements of the control application, one expects to find out if the given network will be able to support it, and at the same time to analyze the worst-case scenario.

Because network media is shared, there must be a mechanism to ensure that all participating nodes have "equitable" access, not necessarily equal as might be implied.

The nature of shared medium is that all nodes of the local network can listen simultaneously, but only one at a time can be transmitting. The challenge, then, is to figure out a method that gives all nodes on the network appropriate access for transmitting packets.

There are two primary methods of controlling access to the network by a computer node: token passing and collision detection. A number of network architectures use combinations of these techniques. For example, the Controller Area Network (CAN) uses collision detection combined with bit arbitration. A variant of CAN, called CANopen, and Profibus use a master-slave architecture in which one node controls all communication on the network.

There are four basic media access principles used in the most common communication systems in automation: Time Division Multiplexing (TDM), Token Bus, Token Ring and Carrier Sense Multiple Access (CSMA). Of these four principles the CSMA is commonly considered to be non-deterministic.

Time Division Multiplexing - TDM protocols reserve time (a time slot) for each node on the medium. Each node then transmits its data in its reserved time and the maximum amount to be transmitted is given by the size of the time slot assigned to the particular node. The maximum delay to access the media is determined by the number and size of time slots assigned to the nodes. We propose a master-slave architecture (also called primary-secondary) in which one node coordinates all communication on the network and synchronizes all nodes [53].

Token Passing Protocols - Token bus and token ring systems circulate a so-called "token". The token is a message that grants the right to transmit on the medium. A node is allowed to hold the token for a restricted time. The access to the network is bounded by the maximum latency of the token to circulate the network. A weaknesses of token passing protocols is possible non-deterministic error recovery when a node is disconnected (the token-passing sequence has to be updated) or when a token is lost. Most protocols use a random (non-deterministic) method to recover tokens.

CSMA-based protocols - In the CSMA based systems a node that intends to transmit at first listens to the network. If no ongoing communication is detected then the node starts its transmission. The CSMA principle has many variants (e.g., non-persistent CSMA, persistent CSMA, CSMA/CD, CSMA/CA, etc.). The CSMA is fully deterministic if used as a Master-Slave system, however in such case the advantages of CSMA are lost. CSMA cannot be deterministic in peer-to-peer systems unless some deterministic algorithms are implemented above the CSMA. However, if the collision resolution algorithms are properly implemented and tuned according to the intended application, the nondeterministic CSMA-based protocol can be well suited for many control applications [82].

2.9 Real-time classes

Within the automation domain the real-time requirements are focused on the response time behavior of data packets. Thus, there are three real-time classes guaranteeing response time [82]:

- *Class 1*: soft real-time (scheduling of data traffic, normally on top of UDP/TCP): scalable cycle time; used in factory floor and process automation.
- *Class 2*: hard real-time (scheduling of data traffic, normally on top of MAC): cycle time between 1 ms and 10 ms; used for control.
- *Class 3*: isochronous real-time (with time/clock synchronisation and routing with time schedule): cycle time between 250 μs and 1 ms; jitter less than 1 μs; used for motion control.
- *Class 4*: additionally, there is a class "non real-time".

2.10 Hybrid wired/wireless networks

In a hybrid wired/wireless fieldbus system, both wired stations (with transceivers being attached to a cable) and wireless stations (having wireless transceivers) should be able to communicate with each other.

In master/slave protocols, for example, a wired (wireless) master should be able to send a request to a wireless (wired) slave and get a response back. Often, there are legacy wired fieldbus systems running in a plant. The addition of wireless stations should not require any modifications to the protocols and applications running on these wired stations. The idea behind such an interconnection is to cut the cabling of a wired fieldbus system at some point and insert a wireless link between the two ends (cable replacement).

A *segment* is defined as a set of stations that are attached to a common medium, run the same protocols, agree on transmission parameters, and thus are able to directly communicate. A wired segment consists entirely of wired stations, while a wireless segment consists entirely of wireless stations. When two wireless segments overlap in space, they need to be separated by some means, for example, by the use of non-overlapping frequencies. Wired and wireless segments are coupled through the use of a single or multiple *coupling devices*. It is possible for these devices to work on different layers of the OSI reference model. The main classes of coupling devices in the realm of fieldbus systems are *repeaters* (working on the physical layer), *bridges* (working on the data link layer) and *gateways* (application level).

For real-time communications, the delay introduced by these coupling devices is important. In repeaters and certain types of bridges, packets are forwarded from one segment to another with either no modifications to their contents or with only minor ones (for example to translate between different addressing formats). Here, the *forwarding delay* can be defined as the time difference between the time instant at which the last bit of a packet is received on the input segment and the time instant at which the last bit of the packet was transmitted on the output segment [114].

2.11 Multi-layer communication structure

Control systems may be represented in a layered manner. At the bottom is the controlled process. Next comes the execution layer, in which each control loop acts on a single variable of the process. On top of this layer comes the coordination and supervision layer that has the role to coordinate the actions on two or more control loops of the first layer. The coordination may use feedback from the first layer or may be independent of any feedback (feedforward). A third layer - management layer - may then coordinate the actions of the coordinators from the second layer (see Figure 2.7). Again, this can be done in a feedback or feedforward manner. This cascading of feedback loops is sometimes referred to as multi-rate control.

Even if the presentation given above seems hierarchical, we do not assume any hierarchy. In practice, coordination between two or more entities at a given layer may either be ensured by exchanging messages between the entities - *decentralized case*, or by using an entity of the next higher layer - *hierarchical case*. Coordination can be ensured in two possible ways:

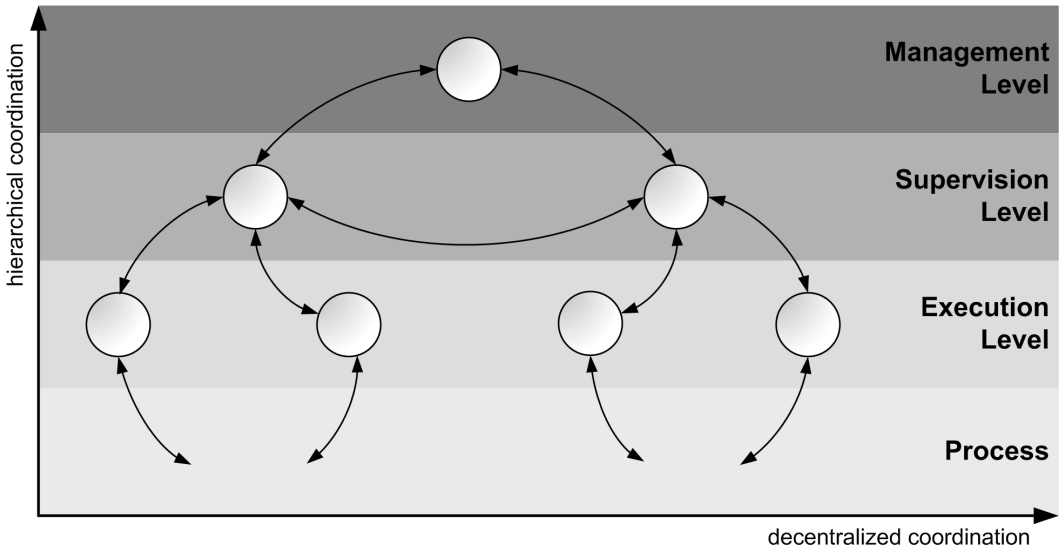


Figure 2.7: Levels of control

- *explicit* - the higher level periodically sends a synchronization message to the lower levels it manages, announcing the lower levels that a new system cycle just begun. It is specific to hierarchical coordination.
- *implicit* - it is ensured through a common sense of time. The local clocks are synchronized over the network using a clock synchronization algorithm. It can be used within decentralized and hierarchical architectures.

2.12 Diversity schemes

A diversity scheme refers to methods for improving the reliability of a message signal. Diversity plays an important role in combating fading and co-channel interference and avoiding error bursts. It is based on the fact that individual channels experience different levels of fading

and interference. Multiple versions of the same signal may be transmitted and/or received and combined in the receiver. Diversity techniques may exploit the multipath propagation, resulting in a diversity gain, often measured in decibels [116].

The following classes of diversity schemes can be identified:

- *Time diversity*: Multiple versions of the same signal are transmitted at different time instants. Alternatively, a redundant forward error correction code is added and the message is spread in time before it is transmitted. By means of bit-interleaving, error bursts are avoided, which often improves the error correction. If the code is not able to correct errors, it might instead be able to detect errors. Automatic Repeat Request (ARQ) schemes, which organize the retransmission of packets, can be applied.
  The main advantage is an increased reliability obtained without additional hardware. The disadvantages are the reduction of bandwidth and the delay caused by multiple transmissions.

- *Frequency diversity*: The signal is transferred using several frequency channels or spread over a wide spectrum. In this way the signal is more robust against fading which can often be assumed as frequency selective .

- *Space diversity*: The signal is transferred over several different propagation paths. In the case of wired transmission, this can be achieved by transmitting via multiple wires. In the case of wireless transmission, it can be achieved by antenna diversity using multiple transmitter antennas (transmit diversity) and/or multiple receiving antennas (diversity reception).

- *Code diversity*: Several wireless nodes can transmit at the same time using different codes.

- *Cooperative diversity*: This technique takes advantage of the broadcast nature and inherent spatial diversity of the channel. Groups of wireless nodes benefit by relaying messages for each other to propagate redundant signals over multiple paths (or routes) in the network. This redundancy allows the ultimate receivers to essentially average channel variations resulting from fading, shadowing, and other forms of interference [64].

The time diversity scheme was implemented for the path from controller to actuators by means of multiple transmissions of the control command signals. On the actor side a simple voting algorithm decides which version of the received signal will be applied to the plant.

2.13 General-purpose control systems

General-purpose control systems can be defined as systems with nodes featuring sensing, actuation, computation, and wireless or wireline/wired communication, that are not biased toward a particular control application, but designed to support many applications such as home automation and control, manufacturing, transportation, robotics, etc. General-purpose computing has proliferated because it has many uses. Proliferation drives down cost, which enables innovation, and encouraging further proliferation. We anticipate that general-purpose

control systems will follow a similar trend. One attribute of general-purpose systems is that they use commonly available computer equipment, networks, components and services [45].

A specific issue of importance for control applications is that general-purpose communication networks exhibit random latency, i.e., delay and jitter, packet loss. Knowledge of delay can be helpful, i.e. it can be used to stabilize the system and improve performance.

Embedded systems have a choice of general-purpose operating system (GPOS) and real-time operating system (RTOS). The choice for either type depends on the functional and timing requirements of the application and of the capabilities of the hardware. However, contemporary GPOSes and RTOSes have strengths and weaknesses where the strength of one is often the weakness of the other.

Complex networked control applications require a feature rich platform combined with third party hardware and software components and has real-time requirements. A solution to this problem has been available for some years. It is possible to host a GPOS as a thread inside an RTOS (e.g. RTAI, Xenomai). The number of combinations is limited, but several solutions are available. However this type of solution has drawbacks. The two main drawbacks are that the support for these solutions is limited and that the flexibility of software becomes limited since the RTOS and GPOS environments differ (Application Programming Interfaces (APIs) and development tools for example). This forces early deployment choices during the development phase.

Real-Time Linux

Currently most embedded applications are using real-time embedded operating systems with small overhead, but as performance and resources improve in the embedded market, GPOS like Linux look more and more attractive for actors in the embedded market.

A number of Linux kernel developers rejected the idea of the hosted GPOS and have become convinced that the Linux kernel can be modified to support real-time applications, and they worked together on changing the kernel and adding features that are common in RTOS kernels. They have created the so called PREEMPT_RT patch that can be applied on top of the standard Linux kernel, which effectively transforms the Linux kernel into an RTOS, but without negatively impacting any of the GPOS functionality. This means that the Linux kernel can be setup to combine the strengths of GPOS and RTOS, but also consolidates API and development environments, meaning that software and development flexibility increases. This creates a very compelling argument for using Linux in many embedded environments. The patch mainly addresses the kernel and over time its features are integrated into the standard kernel.

Linux is a royalty-free, open source operating system. It is based on robust solutions which anyone can view, criticize or improve on. This makes it possible to develop your own extensions to suit special needs, which may often be the case in embedded development. Because of the high number of developers, the GNU/Linux operating system goes through a constant improvement. New functionality is added and bugs are fixed very quickly. In addition, the Linux community is very active and offers support through forums and mailing lists.

The GNU/Linux operating systems is a full-fledged Operating System (OS) with support for a variety of network and file handling protocols, which are very important features for an

embedded system. The operating system is modular, which makes it easy to slim down. This property makes the OS very suitable for systems with limited resources.

Linux also supports a variety of hardware platforms, which decreases the hardware dependencies. With the Linux operating system, the task to move the system to another hardware platform is relatively simple.

Using Linux as the host operating system, you can benefit from the advantage of developing on the same platform as the application should run on. For time independent testing, there is no need for cross-compilers and downloading the software to the target hardware. Linux also provides a very good environment for software development, including a rich set of development and debugging tools.

This page intentionally left blank,
and it could be used for writing notes.

Communication protocol for Wireless Networked Control Systems

3

What would a user do with a gigabit data link (The Internet)?
We are not very good at predicting uses until the actual service becomes available. I am not worried; we will think of something when it happens.

Bob Lucky (1989)

In many industrial sectors like process industry, power plants, manufacturing, traffic and transport systems, different processes can run sequentially or simultaneously on the same plant. A whole process can be divided in *subprocesses* (partial processes) that are fulfilling a specific functionality and run on a set of plant components (sensors, actuators, etc.) and processing units, as shown in Figure 3.1. These modules are considered as *resources* allocated to the partial process, and when a module fails, a subprocess possibly can be maintained by allocating different resources to it.

Due to several constraints like limited or missing communication link, limited processor power, frequently not all partial processes can continue in their full normal mode and reconfiguration is needed. Some less important could be stopped, others could switch to a degraded mode with reduced resource needs or with a different set of resources. These decisions are made in the management layer based on resource states, the urgency of subprocesses in different modes, and their resource needs.

The procedure of identification of reconfigurable subprocesses or functionalities is, in general, strongly application dependent even though some attempts have been done in recent literature to provide systematic tools useful to approach the problem [16][102]. In general, the identification of the reconfigurable functionalities requires a very deep knowledge about the controlled plant (in terms of physics, of interaction between the different parts of the system, etc.) and about the effects which particular reconfiguration actions may have on the system. The crucial step of identifying reconfigurable functionalities is clearly affected by the specific controlled system, as the capability of reconfiguring a specific functionality strongly depends on the degree-of-freedom available to the designers in terms of controllability and observability parts of the system (actuators, sensors, etc.). On the other hand, it also affects, in principle, the design of the controlled system, as the fact of making a specific functionality reconfigurable may be a criterion by which the designers enrich the system with controllable/observable parts and redundancy [11].

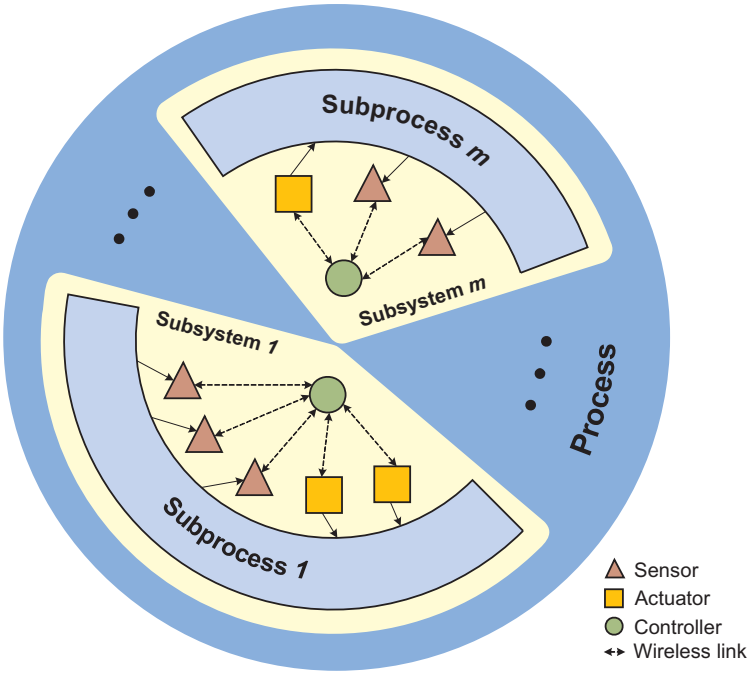


Figure 3.1: Structure of a Networked Control System (NCS).

Useful tools for the design of both diagnostic and reconfiguration algorithms are the fault and functionality tree, which represent well-established methods in the study and synthesis of advanced engineering applications. *Functionality Tree* allows to highlight the functionality map linked to each partial process and is a graphical tool by which the global functionality of the system, representing the root of the tree, is expressed in terms of sub-functionalities, representing intermediate nodes of the tree, which are instrumental for the achievement of the main functionality [11].

Strictly related to the functionality tree is the so-called *Fault Tree* which shows the map of loss of functionalities as a consequence of faulty conditions [11][16]. The fault tree can be interpreted as a complementary version of the functionality tree. Losses of functionalities at each level of the tree are seen as caused by loss of functionalities sitting at lower levels until the main elementary cause, given by the physical fault, is reached. The root of the fault tree represents the loss of the main functionality, namely the inability of fulfilling the main functionality linked to the partial process.

3.1 Definitions

A *task* is defined as the smallest schedulable element in a real-time application. Hence a task is indivisible in the sense that it must be allocated to one processor only. For example, sampling a sensor, updating a state, writing to an actuator, computing a control command. Similarly, a message transmission or reception through a communication medium (electrical, radio spectrum/frequency, optical) is considered a task that is indivisible and non-preemptive.

A general classification of the tasks is shown in Figure 3.2, where the highlighted boxes are

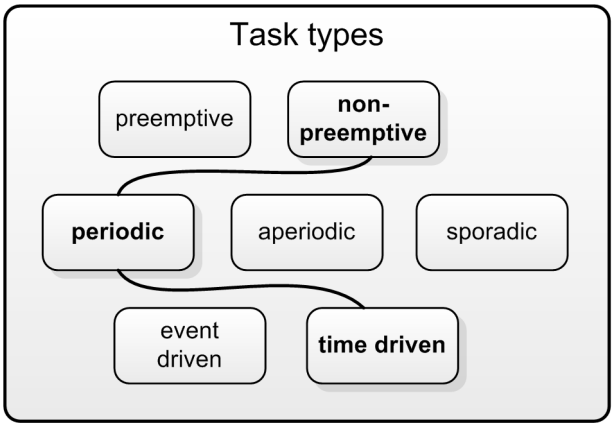


Figure 3.2: Types of tasks.

the types/classes that will be considered further.

The typical classification of tasks is done by behavior in terms of their arrival to the system workload. The traditionally identified types of tasks are *periodic* (sensor readings, control loops), *aperiodic* (synchronization) and *sporadic* (alarms).

3.2 Preemptive versus non-preemptive scheduling

In a *non-preemptive schedule*, a task that has been started is executed until it finishes or gets blocked over a resource, and returns the control of the CPU itself. No other task is allowed to interrupt an executing task, no matter its importance or priority. In a *preemptive schedule*, a task that is executing can at any time be preempted and replaced by a higher priority task. The preempted task is later resumed. Hence a task that is very critical can be guaranteed to be allowed to start executing when it is triggered.

A preemptive schedule is in general preferable since it increases the probability of the existence of a feasible schedule. It should be noted however, that for algorithm complexity reasons, non-preemptive scheduling can be preferable when a static schedule is to be generated. This is because the solution space to search through for a schedule grows tremendously if preemptions are allowed. Furthermore, a system with a scheduler which uses preemption is more difficult to test and verify due to the large set (infinitely large in some cases) of execution cases.

Deriving a feasible communication schedule for a task set while satisfying different types of constraints and dependencies is a problem that belongs to the computation complexity class NP-complete, i.e. the time required to solve it using any currently known alorithm increases very quickly as the size of the problem grows. The algorithms used for solving this problem have many attributes of the branch-and-bound class of search algorithms. The computation-intensive nature of the algorithms favors the off-line computation of the schedule [53].

3.3 Types of task constraints

Typical *constraints* that can be specified on real-time tasks are of three classes: timing constraints, precedence relations, and mutual exclusions constraints on shared resources.

Real-time systems are characterized by computational activities with stringent *timing constraints* that must be met in order to achieve the desired behavior. A typical timing constraints that must be met in order to achieve the desired behavior. A typical constraint on a task is the deadline, which represents the time before which it should complete its execution without causing any damage to the system. Depending on the consequences of a missed deadline, real-time tasks are usually distinguished in two classes [19]:

- *Hard*. A task is said to be hard if a completion after its deadline can cause catastrophic consequences on the system. In this case, any instance of the task should be guaranteed a priori in the worst-case scenario.
- *Soft*. A task is said to be soft if missing its deadline decreases the performance of the system but does not jeopardize its correct behavior.

In the control applications there are *precedence constraints* or execution dependencies, i.e. the tasks cannot be executed in arbitrary order, but they have to respect some *precedence relations* defined at the design stage.

From a task point of view, a resource is any software structure or hardware device that can be used by the process to advance its execution. A resource that can be used by more tasks is called *shared resource*. To maintain data consistency, many shared resources do not allow simultaneous accesses but require mutual exclusion among competing tasks. By analogy, in communication the transmission medium (e.g. air for a wireless network) is the resource that must be shared by all network nodes without disturbing each other.

3.4 Analogy between task and communication scheduling

Organizing the network access in a predictable manner is realized through a schedule – time is divided into slots and each network node has a predefined slot, when it can use the shared medium to transmit. A common approach to the communication scheduling problem is to handle it in the same way as the task scheduling problem, viewing communication links as processors and the communications as tasks. The analogy between network and CPU scheduling is summarized in table 3.1.

The primary parameters of a task are graphically shown in Figure 3.3, where a_i represents the activation time, s_i is the start time, C_i is the computation or completion time, D_i and d_i are the relative and the absolute deadline, respectively.

Scheduling of communications can be performed either by a scheduler before the system starts its execution – *off-line scheduling*, or by a scheduler (e.g. being part of an operating system or a network stack) during the execution – *on-line scheduling*. Off-line scheduling results in a cyclic executive – a table of transmission times of all tasks – and has the advantage that it can give guarantees in advance on the communication start and finish times, and satisfaction

	CPU scheduling	*Network scheduling*
$task\ i$	task execution	packet transmission
T_i	task period	transmission period
C_i	computation or completion time	transmission time
d_i	task deadline	transmission deadline
$preemption$	preemptive	non-preemptive

Table 3.1: Processor and network scheduling analogy

of mutual exclusion requirements. Off-line schedulers are well suited for periodic hard real-time systems, but in order to handle sporadic tasks the scheduling has to be defensive or the hard real-time requirements must be relaxed. Purely aperiodic tasks with hard real-time requirements are usually not allowed at all [97].

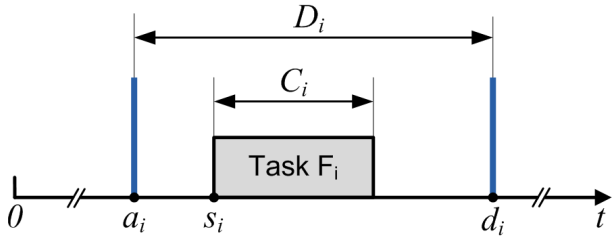


Figure 3.3: Task model parameters

Scheduling algorithms can be grouped in two classes, namely static and dynamic algorithms. A *static* scheduling requires that the complete information about the scheduling problem (number of tasks, deadlines, periods, etc.) is known a priori. Thus, the scheduling problem is solved before the schedule is executed. If at run-time the feasibility can be determined and changes in the configuration may be carried out, then we have a *dynamic scheduling* [40]. Static schedules must always be planned off-line. Dynamic schedules can be planned either off-line if the complete scheduling problem is known a priori but with an on-line implementation (i.e. the configuration is changed at run-time), or on-line if the future is unknown or ignored. Advantage of off-line scheduling is its determinism and the disadvantage its inflexibility. On the contrary, an on-line scheduling is very flexible but poor in determinism.

Figure 3.4 presents a classification of scheduling policies for uniprocessor systems. The highlighted boxes are the policies and the properties considered further for communication scheduling.

There are basically four scheduling paradigms as follows, static table-driven scheduling, static priority-driven preemptive scheduling, dynamic planning-based scheduling, and dynamic best-effort scheduling [53].

Static table-driven scheduling is most suitable for scheduling a set of periodic tasks or operations. A system that deploys this scheduling algorithm has the highest degree of predictability, which is why it is the most practiced approach in real-time systems. A schedule table guarantees that all operations are executed within the given period while satisfying the execution

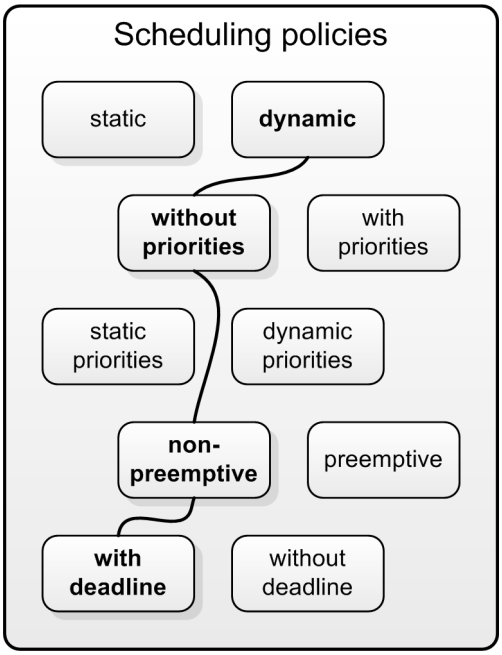


Figure 3.4: Scheduling policies.

dependencies, also known as precedence constraints. On the other hand, this approach can produce a suboptimal schedule in the presence of non-periodic operations.

Timeline scheduling, also known as *cyclic scheduling*, is one of the most used approaches to handle periodic tasks in defense military systems and traffic control systems. The method consists in dividing the temporal axis into slices of equal length – so called *time slots*, in which one or more tasks can be allocated off-line for execution, in such a way to respect the periods or frequencies derived from the application requirements [19].

The scheduling of network transmissions is performed off-line and results in a *system schedule* – a table of transmission times of all tasks. Table-driven scheduling is most suitable for scheduling a set of periodic tasks or operations, giving to the system that deploys it the highest degree of predictability, therefore well suited for periodic hard real-time systems. On the other hand, this approach can produce a suboptimal schedule in the presence of non-periodic operations [97].

The following assumptions are made about the wireless network:

(1) The network is fully linked, i.e. each node is within transmission range of every other node. This is especially important for the network coordinator or high level controller that must synchronize all other nodes.

(2) All nodes have knowledge about current network schedule, which is generated offline and it is provided by the network coordinator at the first system startup or as often is needed.

Given a task set with N periodic tasks F_i, each task having a period T_i, the least common multiple (LCM) of periods is defined as the smallest value possible such that there exist N

integers and the following equation satisfied:

$$T = n_i \cdot T_i,\ 1 \leq i \leq N \tag{3.1}$$

where T represents the schedule length and it is called *system cycle* or *schedule horizon*, and n_i is the number of instances of task F_i within one system cycle. Therefore, a feasible schedule derived for the system cycle is a feasible schedule for the lifetime of the system.

A necessary condition for the existence of a valid schedule and which guarantees the system schedulability is given below:

$$\sum_{i=1}^{N} \frac{T}{T_i}(N_{i,sensors} + N_{i,actuators}) \leq \frac{T}{T_{slot}} \tag{3.2}$$

where $N_{i,sensors}$ and $N_{i,actuators}$ represent the number of sensors and actuators belonging to subsystem i, and T_{slot} is the scheduling time slot. When all control commands are sent together in one message, then $N_{i,actuators} = 1$.

Numerical example. We consider the example of a system divided in two subsystems ($N = 2$) characterized by periods $T_1 = 50$ ms and $T_2 = 30$ ms; composed of $N_{1,sensors} = 3$ and $N_{2,sensors} = 2$ respectively, $N_{1,actuators} = 2$ and $N_{2,actuators} = 1$ respectively. According to (3.2), scheduling the two subsystems requires a $T_{slot} \leq 150/27$ ms $= 5.56$ ms. For a $T_{slot} =$ 1 ms, the considered system has a total of 123 (150-27) free slots left in a cycle, which can be used for retransmissions.

3.5 Modeling of the scheduling system

Modeling of a communication scheduling system involves three main steps that will be described in the following paragraphs, namely the scheduling problem definition, scheduler type definition and the scheduling algorithm definition.

Scheduling problem definition

The scheduling problems and algorithms are categorized by notation $(\alpha|\beta|\gamma)$ proposed in [44], which is widely used in scheduling community, greatly facilitating the presentation and discussion of scheduling problems. Let $\circ$ denote the empty symbol.

(1) The first part $\alpha = \alpha_1\alpha_2$ describes the machine environment. Shortly

- $\alpha_1 = \circ$ and $\alpha_2 = 1$: single machine environment;
- $\alpha_1 = P$ and $\alpha_2 = m$: identical m parallel machines environment.

In our case, the communication medium is shared between all nodes hence $\alpha = 1$.

(2) The second part $\beta \in \{\beta_1, ..., \beta_6\}$ describes the job characteristics of the scheduling problem as precedence constraints, or release times, defined as follows:

- $\beta_1 = \circ$: no preemption is allowed;
- $\beta_2 = res1$: the presence of only one resource is assumed;
- $\beta_3 = prec$: a precedence relation between the jobs is specified. It is derived from a directed acyclic graph G with vertex set $1, ..., n$. If G contains a directed path from task j to task k, we write $F_j < F_k$ and require that F_j is completed before F_k can start.
- $\beta_4 = r_j$: release times that differ per job are specified.

(3) The last part ($\gamma \in \{f_{max}, \sum f_i\}$) denotes an optimality criterion. This field denotes what the scheduler is supposed to optimize. Given a schedule, we can compute for each task F_i the following parameters:

- the completion time C_i;
- the lateness $L_i = C_i - d_i$;
- the tardiness $T_i = max\{0, C_i - d_i\}$;
- the unit penalty $U_i = 0$ if $C_i \leq d_j$, else 1.

The optimality criteria most commonly chosen involve the minimization of

$$f_{max} \in \{C_{max}, L_{max}\}$$

where $f_{max} = max_i\{f_i(C_i)\}$ with $f_i(C_i) = C_i$, L_i respectively, or

$$\sum f_i \in \left\{\sum C_i, \sum T_i, \sum U_i, \sum w_i T_i, \sum w_i U_i\right\}$$

Scheduler type definition

For our communication system the following facts hold true

- the communication medium is shared between all nodes, therefore $\alpha = 1$;
- tasks have different activation times and there are precedence relations between them, hence $\beta = r_j, prec$;
- the sum of completion times C_i is minimized, thus $\gamma = \sum C_i$.

Consequently, the overall scheduling problem is defined as

$$\left(1 \mid r_j, prec \mid \sum C_i\right) \tag{3.3}$$

Scheduling algorithm definition

The algorithm that is able to solve the scheduling problem defined by (3.3) is the so called List Scheduling (LS) algorithm. It is a heuristic algorithm in which tasks are taken from a pre-specified list. Whenever a machine becomes idle, the first available task on the list is scheduled and consequently removed from the list. The availability of a task means that the task has been released. If there are precedence constraints, all its predecessors have already been processed. The algorithm terminates when all the tasks from the list are scheduled [104].

Earliest Completion Time (ECT) first, intended to solve $(P \mid \circ \mid \Sigma C_j)$ problem, is a strategy for the LS algorithm in which the tasks are arranged in the order of increasing completion time C_j in each iteration of the algorithm. The time complexity of ECT is equal or better than $O(n^2 \cdot log(n))$.

3.6 Description of the implemented protocol

In the communication schedule that is generated for a WNCS, there still might be time slots that are not allocated to any node, so they are available for measurement retransmissions or for transmitting other network management messages. In Figure 3.5 it is shown how the free time slots can be used for retransmission requests.

Two crucial requirements of the WNCS have to be assured by the communication protocol. First, all sensor nodes should sample their measurements at the same time instance. And second, control commands have to be simultaneously applied by the actuators before the end of the current control period. The length of the control period determines if there are free time slots left for retransmissions.

The communication is initiated by the controller node that transmits during the first time slot a multicast SYNC message to all sensors and actuators (see Figure 3.5).

The controller knows exactly, according to the communication schedule, during which time slot it should have the first set of sensor measurements (see Figure 3.5). If any measurement is missing from the set, then the controller needs to decide if it requests the retransmission of lost measurements. This decision is based on the number of slots that are left in the current system cycle. The controller can request several sensors to retransmit their measurement with a single multicast management message. This NACK message contains information about which sensors and the order they have to retransmit their measurements.
If the controller decides that there are not enough slots for retransmission requests, it starts with the second phase, i.e. the control commands have to be calculated with the measurements estimated by the observer (see Figure 3.6).

The WiNC protocol ensures tolerance against network-induced effects (like packet loss or time-delayed packets) by employing a voting scheme, i.e. multiple copies (at least three) of the same measurement are simultaneously sent by sensors to the controller, and respectively multiples copies of the control commands are simultaneously sent by the controller to the the actuators. On the communication side there are employed several robust coding/decoding algorithms, thus leading to a multi-layered or multi-level fault tolerance.

In the second phase of the protocol, the control commands are sent for multiple times in the

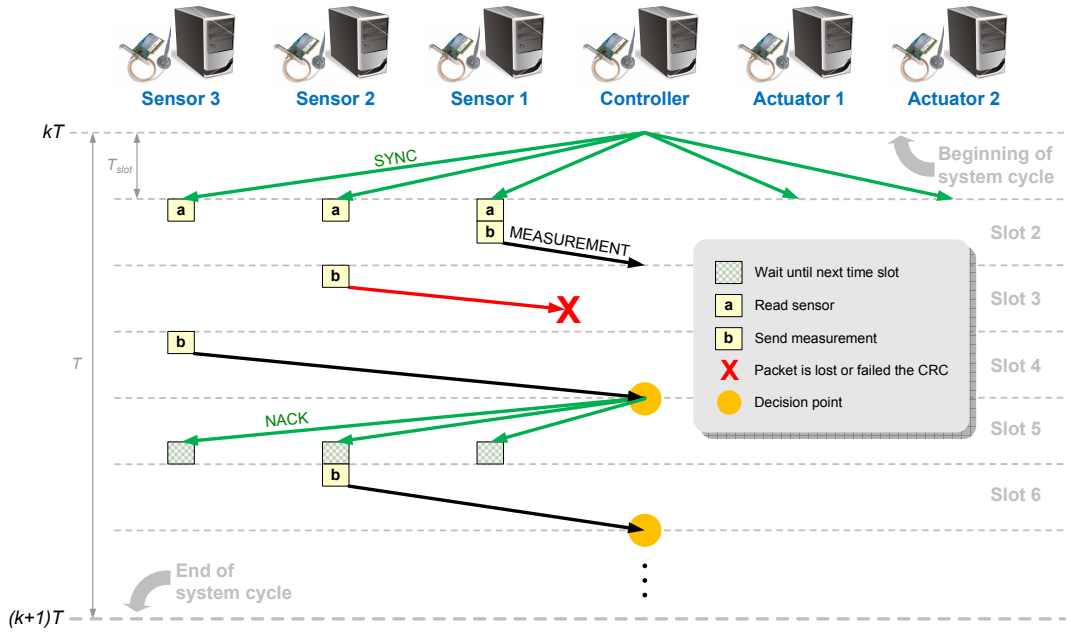


Figure 3.5: Sequence diagram for the first phase of the WiNC protocol

remaining free slots. For the controller it would be important to know if the current control command has actually reached all actuators, and this would require that each actuator replies to the controller with an acknowledgment message, but this would imply one more reserved time slot for each actuator. Otherwise the controller would not be directly aware of a possible failure of the actuator node.

The last time slot in the system cycle is reserved for sending management messages that are announcing a change of the communication schedule, for example in case that nodes are removed or added to the network.

3.7 Addressing scheme

The designed addressing scheme fulfills the following requirements (see also Figure 3.7):

R1. Allows to send a message to all nodes of the network, e.g. a `SYNC` or `SETTINGS` message.

R2. Allows to send a message only to a certain type of nodes, e.g. a `NACK` message to sensor nodes.

R3. Allows to send a message to nodes of a certain subsystem.

R4. Allows to send a message to nodes of certain type belonging to a certain subsystem. This requirement is optional.

In other words, the new addressing scheme must support multicast, i.e. deliver a message to a group of nodes simultaneously, and be able to encode the following information:

1. Node type (profile, category), e.g. sensor, actuator, or controller.

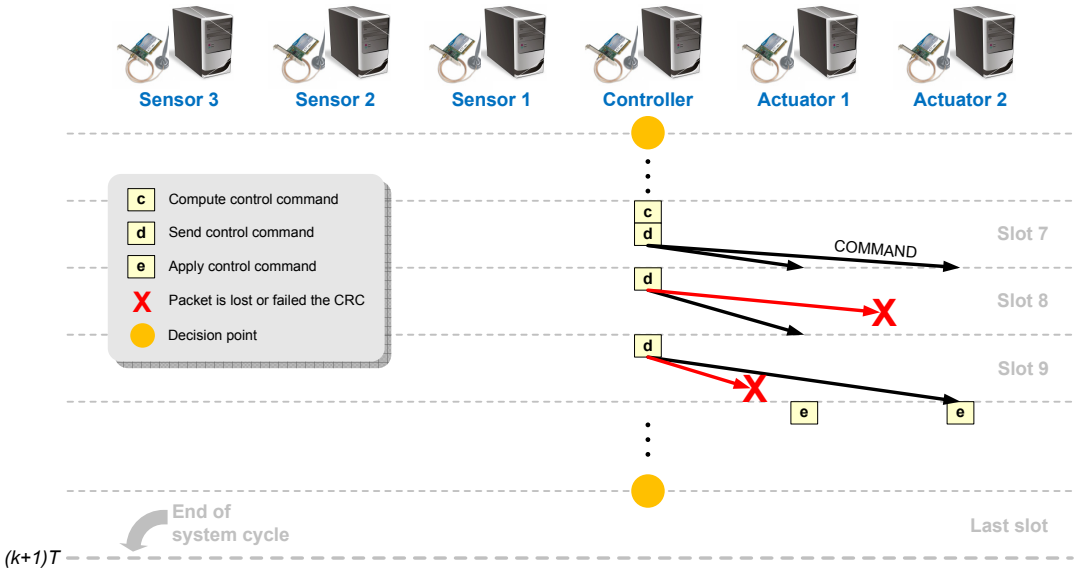


Figure 3.6: Sequence diagram for the second phase of the WiNC protocol

2. The subsystem to which the node belongs, e.g. subsystem #10.

3. The identification number of each node type within a subsystem, e.g. sensor #0, #1 and #2, together with actuator #0 and #1 belong to subsystem #1.

The new address format consists of two bytes separated by a dot: `Byte1.Byte2`.

Byte2 encodes both the node type and subsystem identification number to which the node belongs:

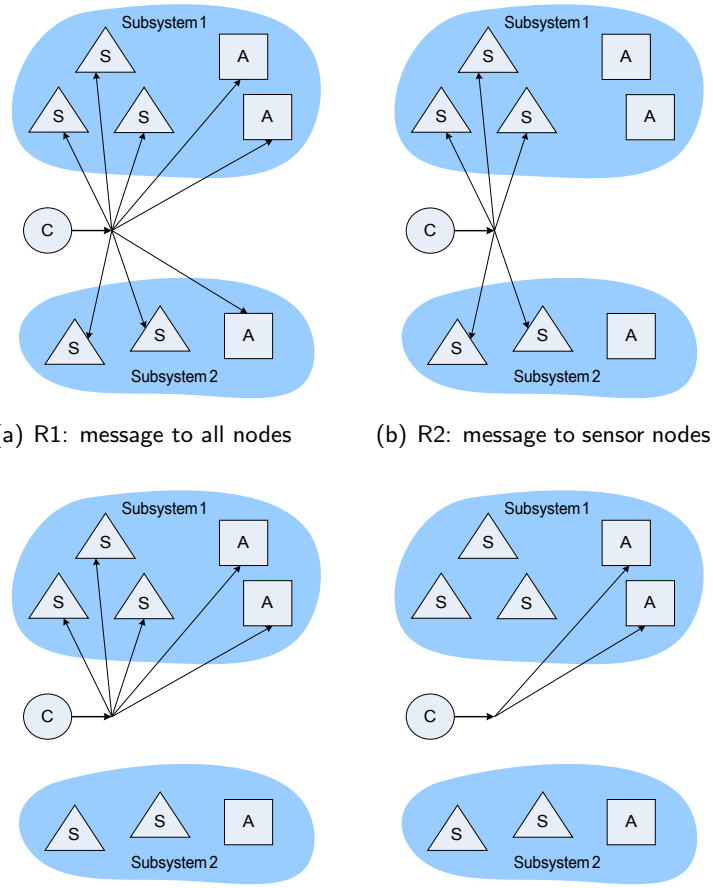
$$Byte2 = \begin{cases} 0, \text{ controller node(s)} \\ 2n-1, \text{ sensor nodes belonging to subsystem } n,\ 0 < n \leq 127 \\ 2n, \text{ actuator nodes belonging to subsystem } n,\ 0 < n \leq 127 \\ 255, \text{ reserved for defining multicast groups} \end{cases} \tag{3.4}$$

Byte1 represents the node identification number within a subsystem:

$$Byte1 = \{m\ , 0 \leq m \leq 255\} \tag{3.5}$$

3.8 Reliability and safety model construction

A successful system or reliability and safety evaluation depends on the process used to define the model for the system. Knowledge of the proper system operation is essential. But more important is an understanding of system operation under failure conditions. What happens when a sensor fails? What happens if the controller fail with its output energized? What happens if the valve jams open? One of the best tools to systematically answer these questions is the Failure Modes and Effects Analysis (FMEA) [42].



(a) R1: message to all nodes

(b) R2: message to sensor nodes

(c) R3: message to nodes of Subsystem 1

(d) R4: message to sensor nodes of Subsystem 1 (optional requirement)

Figure 3.7: Requirements for the addressing scheme

A series of steps, including an FMEA, can be used to help ensure the construction of an accurate reliability and safety model. The following steps are usually followed:

1. Define the faults/failures.
2. Complete a system-level FMEA.
   - Identify and list all system components.
   - For each system component, identify all failure modes and system effects.
3. Classify faults/failures according to effect.
4. Determine the level of model detail.
5. Develop the model.
   - List all failure rates.
   - Build a model that accounts for all failure rates.

Safety Integrity Level (SIL)	Failure probability per hour	Operating hours per failure	Operating years per failure
1	$10^{-6}...10^{-5}$	$10^{5}...10^{6}$	$10...10^{2}$
2	$10^{-7}...10^{-6}$	$10^{6}...10^{7}$	$10^{2}...10^{3}$
3	$10^{-8}...10^{-7}$	$10^{7}...10^{8}$	$10^{3}...10^{4}$
4	$10^{-9}...10^{-8}$	$10^{8}...10^{9}$	$10^{4}...10^{5}$

Table 3.2: Safety Integrity Levels (SILs) from IEC61508 [42, 61].

6. Solve the needed reliability and safety measurements.

Several standards play an important role in the safety and reliability evaluation of the control systems. An important standard is IEC61508, which describes the safety integrity level (SIL) concept, with four levels as shown in table 3.2.

We consider the failure rate and reliability for each wireless link within our NCS, that is the links between sensors and controller, and from controller to actuators.

The failure rate λ relates the number of failures per time interval to the number of survived elements, and it is defined below [61]

$$\lambda(t) = \frac{1}{\text{number of functioning elements}} \frac{\text{number of failures}}{\text{time interval}} \tag{3.6}$$

The reliability function $R(t)$ describes the number of survived elements relative to the initial number of elements, and it is described by the equation below

$$R(t) = e^{-\lambda t} \tag{3.7}$$

Another measure for reliability is the Mean Time To Failure (MTTF) that represents the average failure-free (correct) operation until a failure occurs

$$MTTF(t) = \frac{1}{\lambda(t)} \tag{3.8}$$

Considering the previous definitions of the failure rate, reliability function and MTTF, we reformulate them for a wireless link as following

$$\lambda(t) = \frac{1}{N_{rp}(t)} \frac{N_{lp}(t)}{t} \tag{3.9}$$

$$R(t) = e^{-N_{lp}(t)/N_{rp}(t)} \tag{3.10}$$

$$MTTF(t) = \frac{N_{rp}(t)}{N_{lp}(t)} t \tag{3.11}$$

where N_{rp} represents the number of received packets and N_{lp} is the number of lost and not received within the deadline packets. Thus, reliability translates to the probability of receiving a packet.

Reliability analysis requires the evaluation of connected elements. This is based on so called reliability networks, representing the kind of connection also called combinational modeling. For series connection of elements, which may fail statistical independently from each other, it holds for the overall reliability that all elements operate correctly for constant failure rates λ_i

$$R_{system}(t) = \prod_{i=1}^{m} R_i(t) = e^{-\sum_{i=1}^{m} \lambda_i(t)} = e^{-\lambda_{system}(t)} \tag{3.12}$$

$$\lambda_{system}(t) = \sum\nolimits_{i=1}^{m} \lambda_i(t) \tag{3.13}$$

$$\mathrm{MTTF}_{system} = \left[\sum_{i=1}^{m} (\mathrm{MTTF}_i)^{-1}\right]^{-1} \tag{3.14}$$

In order to reach a small overall failure rate, all elements should be of similar small failure rate λ_i. Otherwise the largest λ_i will dominate and determine λ_{system}.

If the elements are arranged in parallel connection, they are redundant and the reliability of this type of connection is then

$$R_{system}(t) = 1 - \prod_{i=1}^{m} (1 - R_i(t)) = 1 - \prod_{i=1}^{m} \left(1 - e^{-\lambda_i(t)}\right) \tag{3.15}$$

$$\lambda_{system} = \frac{\lambda}{\sum_{i=1}^{m} \frac{1}{i}} \tag{3.16}$$

$$\mathrm{MTTF}_{system} = \frac{1}{\lambda} \sum_{i=1}^{m} \frac{1}{i} \tag{3.17}$$

assuming the same $\lambda_i = \lambda$ for all elements.

The overall reliability of a WNCS is given by the reliability of the control strucure/system and the reliability of the communication network.

3.9 Schedule for the benchmark system

The benchmark system is composed of a set of ten periodic tasks that involve network transmissions: a subset $\{F_1,\ F_2,\ F_3,\ F_4,\ F_5,\ F_6\}$ of six tasks ($N_{1,\mathrm{tasks}} = 6$) belonging to the first subsystem with periods of $T_1 = 50ms$ and another subset $\{F_7,\ F_8,\ F_9,\ F_{10}\}$ of four tasks ($N_{2,\mathrm{tasks}} = 4$) belonging to the second subsystem with periods of $T_2 = 30ms$, as shown in table 3.3.

Each task F_i is characterized by the following parameters, where $1 \leq i \leq (N_{1,\mathrm{tasks}} + N_{2,\mathrm{tasks}})$:

Task name	*Task description*	*Task executed on*
F_1, F_2, F_3	Send measurement to controller	Level sensor 1, Level sensor 2, Level sensor 3
F_7, F_8		Position sensor, Angle sensor
F_4, F_9	Compute control commands and send them to actuators	Controller
F_5, F_6	Acknowledge to controller after receiving	Pump 1, Pump 2
F_{10}	and applying the control command	Servomotor

Table 3.3: Task definitions for the benchmark system.

- **Activation** (or **arrival**, or **release**) **time**: $a_i = (InstanceNumber(F_i) - 1) \cdot T_n$, where T_n represents the period of subsystem to which the task belongs.
  Construction $InstanceNumber(F_i)$ gives the current instance number of task F_i within the system cycle.

- **Computation** (or **completion**) **time**: $C_i = T_{slot}$
  Control commands might be sent several times and then the corresponding number of time slots must be allocated, for example $C_i = 3T_{slot}$.

- **Relative deadline**: $D_i = T_n$,
  where T_n represents the period of subsystem to which the task belongs.

- **Start time**: s_i is the time at which the function begins its execution, i.e. data transmission. It is the information that actually represents the system schedule.

- **Precedence constraints**: $s_i < s_j$
  Constraints are represented by the natural order in which tasks must be executed, e.g. measurements must be sent before computing the control commands.
  These constraints are captured in the *adjacency matrix* that defines a graph were nodes correspond to tasks and edges are precedence constraints between these tasks. Inequality above reads that task F_i must be completed before task F_j can be started, and represents an edge in the graph.

This page intentionally left blank,
and it could be used for writing notes.

4 Resource and reconfiguration management

If a system is designed to be tolerant to a set of faults, there will always exist an idiot so skilled to cause a non tolerant fault.

Green's Law

In a dictionary sense, the term *reconfigurability* of a system can be defined as its ability to adapt to a new task by altering its configuration. In the context of industrial automation, reconfigurability can refer to the ability of the control system to adapt to the faults or the disturbances in production operations (e.g., fault in a system component, failure of a process unit, loss or raw-material or utility supplies, even changes in market demands or prices) or to emerging changes (e.g., introduction of new products, processes, raw-materials, utilities, technologies) [25].

The recently increasing complexity of man-made systems increases their vulnerability for faults and malfunctions. At the same time, requirements for system dependability are surging as a consequence of, for example, tightening environmental regulations. Maintaining system dependability at required levels by improving individual components is challenging and expensive. Feedback control is an ideal technology for increasing the system dependability. Control reconfiguration denotes a class of solutions to the fault-tolerant control problem, where the closed-loop structure and the controller dynamics are actively adjusted in response to component malfunctions. The goal of the control reconfiguration consists in preventing component-level faults and failures into system-level failures.

In the following chapter, a modeling and analysis approach for networked control systems, which is adaptable and dynamically reconfigurable, is presented.

We model the overall networked control system according to the following assumptions:

- The resources that are present in the system are grouped into *categories*, where a category consists of the resources of a certain type;
- Each function (partial process or task) is executed according to a *schedule* (plan) that specifies how much time (estimated time) the execution may take and what resources or what category of resources are needed;
- The resources that are engaged in execution of a function can be used at the same time to execute any other function only if the resources can be *shared*;
- There are function-resource and resource-resource dependencies considered as *constraints* in optimization.

A resource and reconfiguration manager would have the following main roles:

- it gets and evaluates information regarding operational state of all the resources,
- it supervises resource needs and urgencies of all functions in their different working modes,
- it decides on resource allocation and reconfiguration, i.e. how best to use available resources in order to achieve the most urgent functions (the ones with the highest priority) – *scheduling*,
- it receives requests from a human operator.

Distributed intelligent control involves matching the control model more closely with the physical system. This is particularly relevant to manufacturing control systems that are required to control widely distributed devices in an environment that is prone to disruptions. With this model, control is achieved by the emergent behavior of many simple, *autonomous*, and cooperative entities that are based on the principles of object-oriented and agent-based systems [18].

Using object-oriented approaches, which are known from computer science, we can regard partial processes (respectively their corresponding control functions) and their used resources as objects with the relation *uses*. Switching into a different working mode means exchanging some used resource objects. That means the object model is dynamic, i.e. it varies with time. Because we need a dynamic model, the allocation of functions (software) to resources (hardware) should be done at runtime (on-line).

Dynamic object oriented model (DOOM) is a system that represents classes, attributes, relationships and behavior as metadata. The system is a model based on instances rather than classes. In the engineering phase, users have to describe resources, control functions and their relationships using metadata (an object model). These can be changed to reflect changes in the domain. Consequently, the object model is adaptable; when the descriptive information is modified, the system reflects those changes. The test implementation uses a text input in XML syntax for the engineering data and stores it as an XML file (this replaces the engineering data base). Of course, commercial applications would require an additional graphical user interface as part of the engineering tool.

The engineering tool interprets the XML model off-line and generates the corresponding C++ code. This code is translated and loaded into the target platform. It contains plant specific resource and function objects, i.e. special objects inherited from the metaobject classes `MResource` and `MFunction` (see Figure 4.1).

4.1 Definitions

A *resource* is an entity representing the smallest part (unit) of the plant whose unavailability (faulty or already allocated to a function) causes a change in the behavior of a function (plant partial process). Its main roles are as follows

- update its properties/attributes (e.g. the read value from a sensor);

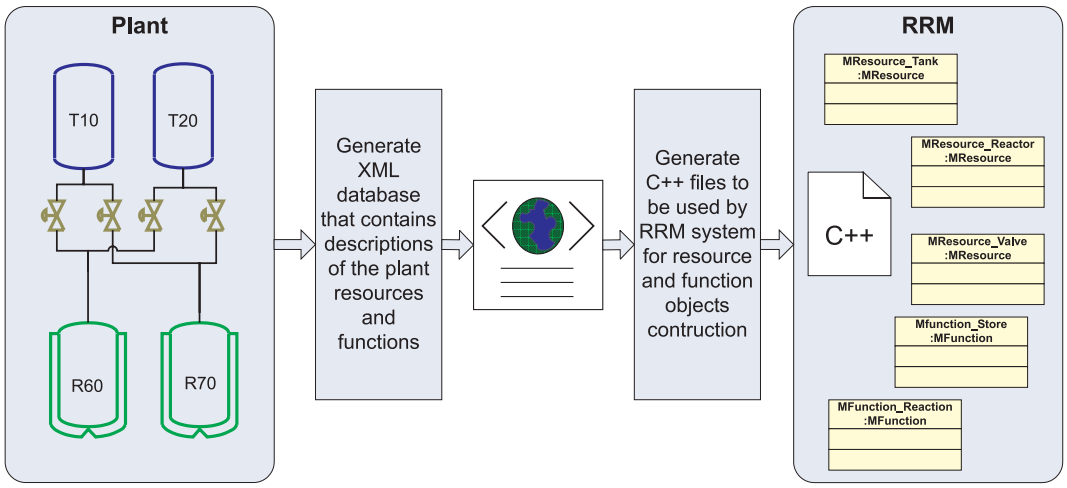


Figure 4.1: Steps of code generation.

- request the state of physical resource from hardware layer, resource monitors, or diagnostic modules;
- respond to application layer (operator) with resource state.

Resources can be grouped together and form a *resource group*, which will be treated as one resource and considered a new category of resources (see Figure 4.2).

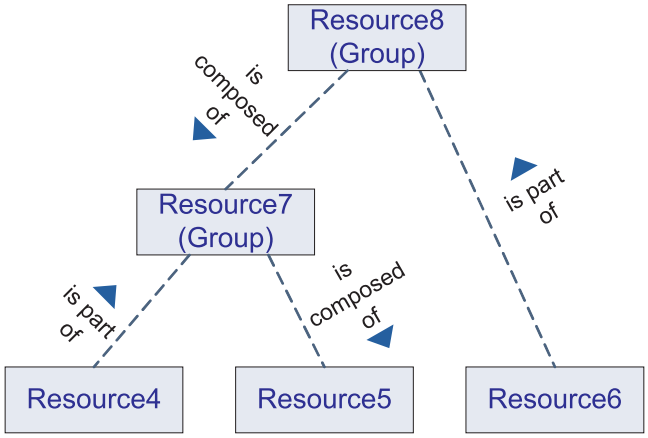


Figure 4.2: Resource groups.

A *function* is an entity performing a useful task, job or service. Its main roles are as follows

- search for available resources;
- process the values received from resources;
- coordinate the scheduling of other functions – it has the role of local scheduler for its sub-functions.

Differentiating the system into resources and functions simplifies the architecture and our task in defining the relationships between them (see Figure 4.3). This helps in reflecting the

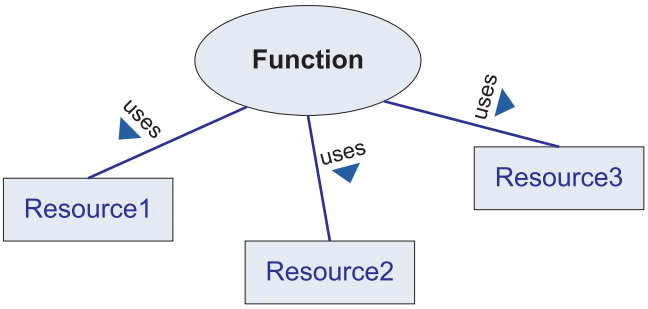


Figure 4.3: Resources allocated to a function.

abstract model deeper into the software structure, giving way to semantic concept to be still valid at any level. There are different types of plants, and each plant has different categories of resources and functions to be performed, therefore many types of resource and function objects must be considered. This is achieved using specialization by inheritance.

4.2 Hierarchy of functions

A process is subdivided into partial processes (functions). Functions can be represented by other higher-level functions. This results in ordering functions in a hierarchy. We call *super-function* the highest-level function. Lower level functions are named *sub-functions* or *child-functions* relative to their *parent-functions* (see Figure 4.4).

Each super-function comprises one or more sub-functions distributed in one or more levels. The super-function represents the overall process or task to be performed, requested from the application layer (operator). The hierarchy must be modeled in a way that the integrity within a function will be the highest and interdependence among functions the lowest, so that we will have a low communication between separate functions.

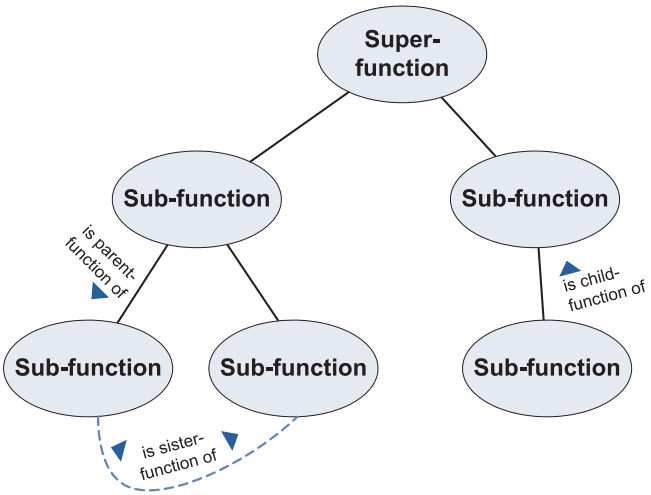


Figure 4.4: Hierarchy of functions.

The following factors shape the modeling of the function behavior:

- We need a system that administers the usage of resources in the most optimal way;
- The importance of running functions is time varying (in case of discontinuous functions) and dependent on the usage of resources;
- Running functions must be fault-tolerant, i.e. able to detect faults and self-heal (self-repair).

Every system has a limited number of resources, which are not always operational or available. Each function needs one or more operational and available resources. To model an architecture that handles all these, we have to consider the worst-case scenarios.

Resources are always limited in availability and furthermore, the usefulness of a particular resource to some particular function may also be limited. Because of the scarcity of the resources, we need a smart scheduler and a decision making factor for the allocation of resources. Performing a process using a set of resources attributes a utility or added-value to the function. In other words, for a function to run, we need to reserve a number of resources to that function. This will decrease the available resources in our system, increasing the difficulty for the following functions to find necessary resources for running.

4.3 Communication mechanism

In order to fulfill the goal of relocatability, all objects are implemented independently from the underlying hardware and used technology. The independence is achieved by providing the *virtual functional bus (VFB)* – an abstract communication level – as a means for a virtual hardware. All function and resource objects communicate with each other through VFB (see Figure 4.5).

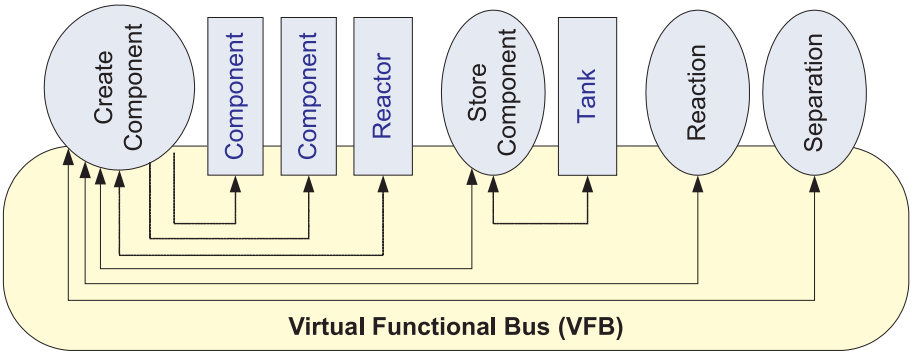


Figure 4.5: Virtual Functional Bus (VFB).

Mapping the virtual functional bus to a set of processing and input/output nodes is done now manually by the control engineer. This operation must be automatically done by a tool that is fed with

- description of function and resource objects;
- description of processing and I/O nodes;

- description of system constraints,

and offers in exchange a set of optimal mappings. These descriptions should be provided in a standard format, as depicted in 4.6.

Function descriptions should include information about each function, like interfaces, behavior, direct hardware interfaces (I/O), requirements on run-time performance (memory, computing power, throughput, timing/latency, etc.).

Resource descriptions (including processing nodes and I/O nodes) should include information like hardware interfaces, hardware attributes (memory, processor, computing power, etc.), connections and bandwidths.

System constraints description includes information about the overall system: bus systems, protocols, communication matrix and attributes (e.g. data rates, timing, etc.), function clustering, function deployment (distribution to processing and I/O nodes).

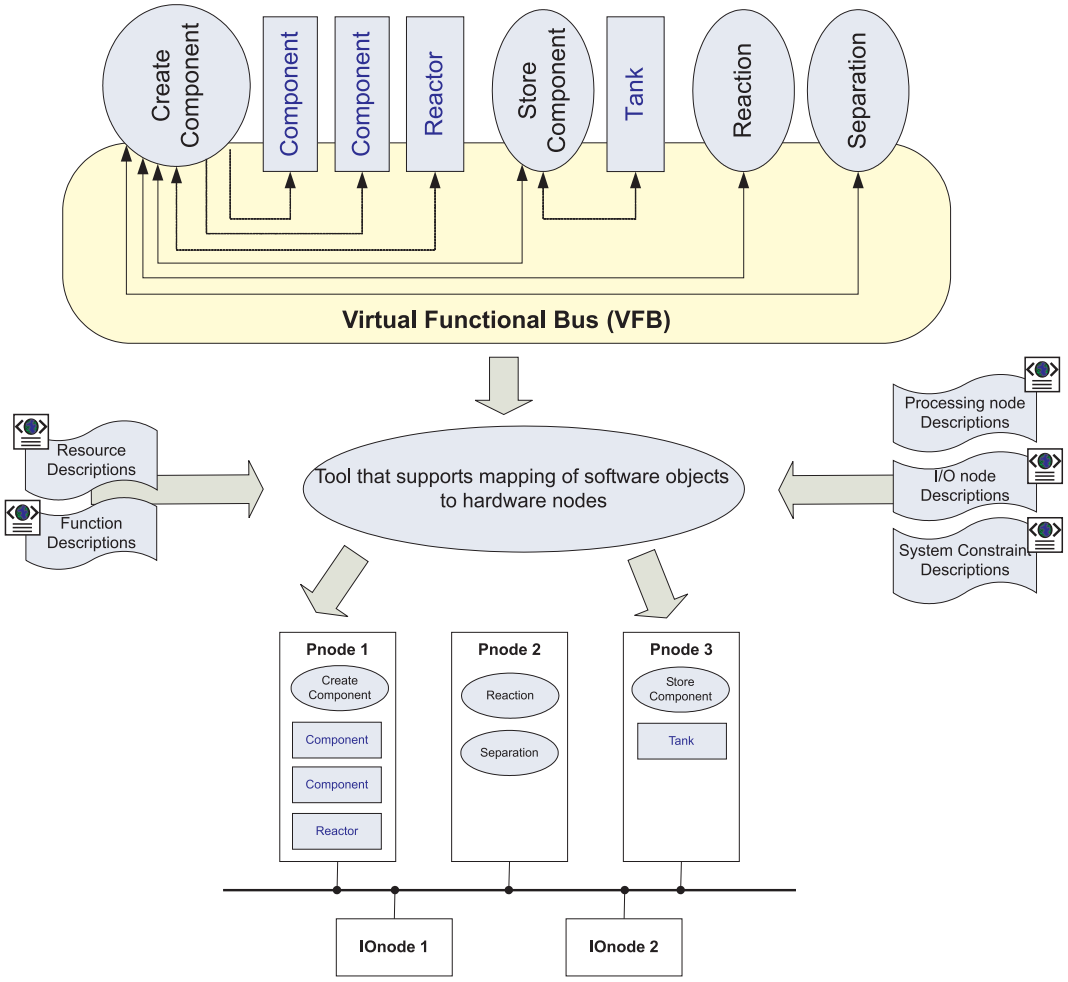


Figure 4.6: Mapping of software objects to hardware.

4.4 Global and local fault tolerance – Tolerance layers

Having the function relationships structured in a hierarchy, super-functions aggregating sub-functions, it is necessary to analyze how the fault tolerance behavior is reflected by this structure. Fault tolerance is provided at many levels in this system – hierarchical failure masking, as depicted in Figue 4.7. Each layer implicitly uses the fault tolerance provided by lower levels, but may need to intervene explicitly if a fault occurs and cannot be recovered at a lower level.

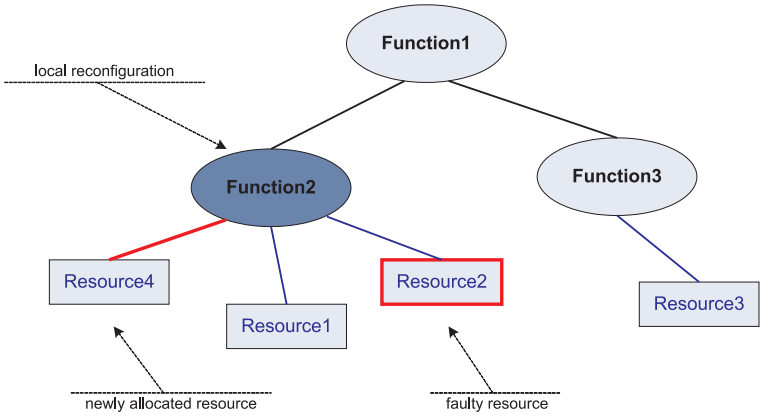


Figure 4.7: Function layers – Tolerance layers.

By local fault tolerance we understand the reallocation of resources managed by sub-functions (lower level functions) – lower cost, without the intervention of super-function (higher level functions). This prevents from propagating the failure to higher level functions.

On the other hand, if the fault cannot be recovered locally, global reconfiguration is needed, involving higher level functions – higher cost.

Function granularity adds another degree of freedom in designing the controller of a plant. The proposed architecture allows us to partition or divide the functions arbitrarily, based on the needs.

4.5 Actual implementation

Each resource will have its properties represented by two types of data: a string of characters or a numerical value; or by an array of one of the two types (see Figure 4.8).

For each function that will run on the plant, we define its *Parameters*:

- `InputParameters` are the parameters received from higher-level functions (parent-functions) or operator. They can represent numerical values (e.g. volume of liquid that should be stored in a tank, temperature that has to be reached in order for a reaction to take place) or names of the resources that will be used by the function.
- `OutputParameters` are the parameters sent to the low-level functions (child-functions) and they are the result of searching resources needed by the function, specified within `NeededResources` tags.

```
<Resource>
    <PlantResource>
        <Tank>
            <T10>
                <sources type="string">V91;V101</sources>
                <sinks type="string">V11;V12;V13</sinks>
                <contains type="string">LIAL10</contains>
                <level type="double">0</level>
            </T10>
            :
            <T30>
                <sources type="string">V93;V103</sources>
                <sinks type="string">V31;V32;V33</sinks>
                <contains type="string">LIAL30</contains>
                <level type="double">0</level>
            </T30>
        </Tank>
        <Reactor>
            <R60>
                <sources type="string">V61;V62;V11;V21;V31</sources>
                <sinks type="string">V63;V64</sinks>
                <contains type="string">TIC61;LISH62</contains>
                <level type="double">0</level>
            </R60>
            :
        </Reactor>
        <Level_Sensor>
            <LIAL10>
                <partOf type="string">T10</partOf>
                <controlledBy type="string">nodeIO1;nodeIO2</controlledBy>
                <data type="double">0</data>
            </LIAL10>
            :
        </Level_Sensor>
    </PlantResource>
    <ControlResource>
        <nodeIO>
            <nodeIO1>
                <controlledBy type="string">nodeCPU1</controlledBy>
            </nodeIO1>
            :
        </nodeIO>
        <nodeCPU>
            <nodeCPU1/>
            <nodeCPU2/>
        </nodeCPU>
        <CPU>
            <CPU1>
                <load type="double">0</load>
                <partOf type="string">nodeCPU1</partOf>
            </CPU1>
            :
        </CPU>
        <Memory>
            <MEM1>
                <free_space type="double">3000</free_space>
                <partOf type="string">nodeCPU1</partOf>
            </MEM1>
            :
        </Memory>
    </ControlResource>
</Resource>
```

Figure 4.8: Structure of resources in XML file.

The child-functions together with their input parameters are specified within `NeededFunctions` tags. The rules or constraints for selecting the needed resources are specified within `constraint` tags. Input and output parameters of a parent function and its child functions are linked by parameter's name (see Figure 4.9 and Figure 4.10).

Resource allocation steps: First, from the defined plant and control resources are selected the ones from the same category and that are fulfilling the constraints specified within `constraint` tags. They compose the selected resources set. Second, from selected set are chosen the

```
<Function>
    <CreateComponent>
        <InputParameters>
            <inputp type="string" name="liquid_type"/>
            <inputp type="double" name="needed_volume"/>
        </InputParameters>
        <OutputParameters>
            <outputp type="string" name="reactor_destination"/>
            <outputp type="string" name="liquid_type1"/>
            <outputp type="string" name="liquid_type2"/>
            <outputp type="double" name="needed_volume1"/>
            <outputp type="double" name="needed_volume2"/>
        </OutputParameters>
        <NeededResources>
            <nres name="res1">
                <res name="liquid_type1" category="Component">
                    <constraint name="componentOf" type="string">liquid_type</constraint>
                    <constraint name="volume" type="double">80%</constraint>
                </res>
            </nres>
            <nres name="res2">
                <res name="liquid_type2" category="Component">
                    <constraint name="componentOf" type="string">liquid_type</constraint>
                    <constraint name="volume" type="double">20%</constraint>
                </res>
            </nres>
            <nres name="res3">
                <res name="reactor_destination" category="Reactor"/>
            </nres>
        </NeededResources>
        <NeededFunctions>
            <function name="StoreComponent">
                <parameter type="string" name="liquid_type1"/>
                <parameter type="double" name="needed_volume1"/>
                <parameter type="string" name="reactor_destination"/>
            </function>
            <function name="StoreComponent">
                <parameter type="string" name="liquid_type2"/>
                <parameter type="double" name="needed_volume2"/>
                <parameter type="string" name="reactor_destination"/>
            </function>
        </NeededFunctions>
    </CreateComponent>
    .
    .
    .
</Function>
```

Figure 4.9: Structure of functions in XML file.

resources (faultless and not allocated) that will be allocated to the function. They compose the allocated resources set.

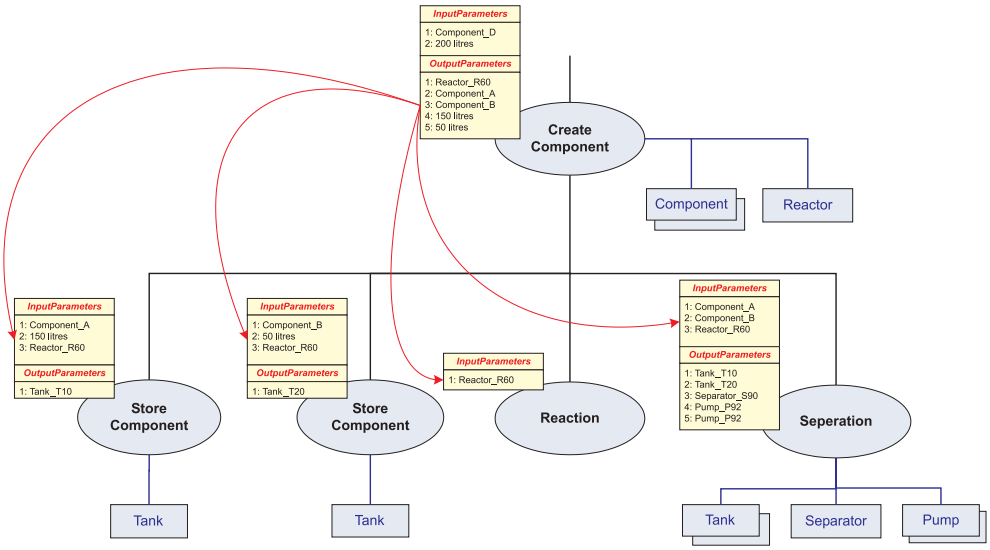


Figure 4.10: Example of Input/OutputParameters exchange between parent- and child-functions.

4.5.1 Hierarchical control pattern

It distributes complex control algorithms among its various pieces. It uses two types of interfaces: control interfaces that monitor and control how the behaviors are achieved and functional interfaces, which provide the services controlled by the other set of interfaces. The control interfaces set the quality of service, such as fidelity, accuracy, and so on, as well as selected policies that govern how the execution proceeds. The functional interfaces execute the desired behavior using the quality of service and policies set via the control interface. In our case, we consider as control interface the `Input/Output Parameters` and as functional interface the `Input/Output Variables`.

4.6 Classes hierarchy

The classes are determined through the following considerations. With embedded systems, the hardware that the system executes on can be an excellent source of classes. In fact, it is often the case that the best software architecture tends to be a close match with the physical hardware architecture [12]. In our case, we have the active class (active objects are instances of active classes) `MSystem` that manages creation and destruction of `MResource` and `MFunction` active objects (coordinates subsidiary threads which are modeled as active objects). We need multi-threading (multi-tasking), as `MResource` objects representing for example the sensors must continuously monitor/read the corresponding physical sensor. The reason that this pattern is so common is that in concurrent systems, you need some way to start, stop, and synchronize the concurrent threads of execution in a controlled manner. Having a primary control object is a simple way to achieve this [12].

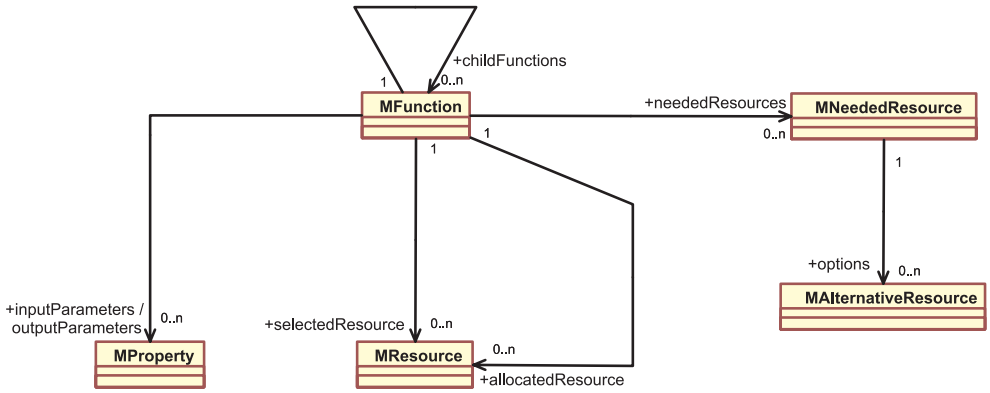


Figure 4.11: Classes hierarchy of RRM.

Functions are distributed among devices (processing nodes) taking into account the following considerations:

a. functional interconnections (interdependencies) between functions;

b. interdependencies between functions and (needed category of) resources;

c. physical interconnections between resources;

d. physical interconnections between nodes/units/devices.

4.7 Decision algorithms

Quality is understood today as the totality of features and characteristics of an item (product or service) that bear on its ability to satisfy stated or implied needs [15].

For comparing different allocation alternatives, a criterion is needed. In order to choose "the best" resource from a category set of resources, we define a quality measure $Q \in [0, 1]$ specific to each resource – *quality of (functional) service* provided by resource. For example, a sensor provides a better service than an observer (estimator), i.e. it will have a larger QoS compared to the observer; or a sensor with 8-bit resolution has a higher QoS compared to a sensor with 4-bit resolution.

The second criterion – *suspending cost* – will help making decisions when comparing possible alternatives with the currently available resources. The general form of suspending cost of a function F_i for $t \in [T_{SF_i}, T_{DF_i}]$ is given below:

$$C_{F_i}(t) = P_{F_i} \cdot [f_{1F_i}(t - T_{SF_i}) + f_{2F_i}(T_{DF_i} - t)] \tag{4.1}$$

where P_{F_i} is the priority among its sister-functions, f_{1F_i} is the suspending function, f_{2F_i} is the deadline function, T_{SF_i} is the starting time, and T_{DF_i} is the deadline time. For a function that runs periodically, the difference between the deadline and starting time represents the period of the function:

$$T_{F_i} = T_{DF_i} - T_{SF_i} \tag{4.2}$$

Function $f_{1F_i}(t)$ takes into account that we have already invested effort into the function, which in some cases is lost or results in lower quality when we relinquish this function. This invested effort often is increasing with the progress of function execution, and could be for example:

- cost of material and energy (as far as they are lost or diminished in their value because of effects like lower product quality);
- cost of not running other functions, because our function had occupied valuable resources.

Function $f_{2F_i}(t)$ might be useful if there is a deadline (that must be respected), when the result of the function is needed. For example in case of a periodic function.
We consider a case when f_{1F_i} and f_{2F_i} have the form below:

$$f_{1F_i}(t) = \sum_{R_j} \left(Q_{R_j} \cdot U_{R_j,F_i} \right) \tag{4.3}$$

$$f_{2F_i}(t) = 0$$

which yields the following form of $C_{F_i}(t)$

$$C_{F_i}(t) = P_{F_i} \sum_{R_j} \left(Q_{R_j} \cdot U_{R_j,F_i} \right) \tag{4.4}$$

where Q_{R_j} is the *quality of service provided by resource* R_j, and U_{R_j,F_i} is the *utility* of resource R_j with respect to function F_i

$$U_{R_j,F_i} = U_{category(R_j),F_i} = U_{Category_k,F_i}$$

for which

$$\sum_{Category_k} U_{Category_k,F_i} = 1 \tag{4.5}$$

where $Category_k$ is the set of resource categories needed by function F_i.

The benefit of executing a task or function may depend not only on the task priority/importance but also on the time at which it is completed. This can be described by means of specific *utility functions*, which describe the value associated with the task as a function of its completion time [19].

Assertion. In a decision between more alternatives, the function with actually higher suspending cost is executed, whereas the function with lower cost is suspended (stopped or not started).

We can also consider C_{F_i} (independent of time) as being a measure of *quality of service provided by function* F_i

$$Q_{F_i} = \sum_{R_j} \left(Q_{R_j} \cdot U_{R_j,F_i} \right) \tag{4.6}$$

The goal would be running as many functions as possible with the highest possible QoS.

Definition. The *use degree* D_{F_i,R_j} necessary to run function F_i represents the percentage of resource R_j needed by function in order to run, and is defined by following equation

$$D_{F_i,R_j} = \begin{cases} 1, \text{ exclusive use} \\ 0 < D_{F_i,R_j} < 1, \text{ shared use} \\ 0, \text{ not used} \end{cases} \tag{4.7}$$

A resource R_j can accept a service request from function F_i only if the actual use degree of resource D_{R_j} satisfies next condition

$$D_{R_j} + D_{F_i,R_j} \leq 1 \tag{4.8}$$

We may also define the quality of service provided by resource R_j as a function of its own use degree D_{R_j}

$$Q_{R_j}(t) = f\left(D_{R_j}(t) \right) \tag{4.9}$$

An example is the QoS provided by a communication network, which offers a lower QoS if the traffic in the network is high, i.e. its actual use degree is high. Or the use degree can be expressed as a function of network's parameters and so we get the following expression

$$Q_{R_j}(t) \quad = \quad f(\text{one-way delay}(t), \text{packet loss}(t), \text{bandwidth})$$

Considering the proposed hierarchy of functions, we define the quality of service provided by super-function F_i as

$$Q_{F_i} = \sum_{k} \left(Q_{SF_{i,k}} \cdot U_{SF_{i,k},F_i} \right) \tag{4.10}$$

where $Q_{SF_{i,k}}$ is the quality of service provided by sub-function $SF_{i,k}$ of super-function F_i, and $U_{SF_{i,k},F_i}$ is the utility of sub-function $SF_{i,k}$ with respect to F_i (i.e. the importance of running $SF_{i,k}$ compared to other sister-functions), and it is defined by the following equation:

$$\sum_{k} U_{SF_{i,k},F_i} = 1 \tag{4.11}$$

where *k* is the number of sub-functions belonging to F_i.

Numerical examples

Example 1
Let us consider the three-tank benchmark. *Decoupling Controller* is defined as super-function and *Measurement1, Measurement2, Measurement3, Command1, Command2* as its sub-functions. Utility of each sub-function with respect to the super-function F_A is $U_{F_{A1},F_A} = U_{F_{A2},F_A} = U_{F_{A3},F_A} = U_{F_{A4},F_A} = U_{F_{A5},F_A} = 1/5 = 0.2$. Utility of each needed resource is expressed as follows:

$$U_{R_1} = U_{Category(R_1),F_{A1}} = U_{C_1,F_{A1}} = 1$$

So the QoS provided by sub-function F_{A1} would be

$$Q_{F_{A1}} = \begin{cases} Q_{R_1} \cdot U_{R_1,F_{A1}} = 1 \cdot 1 = 1, \ (mode1) \\ Q_{R_6} \cdot U_{R_6,F_{A1}} = 0.5 \cdot 1 = 0.5, \ (mode2) \end{cases}$$

and the QoS provided by super-function F_A will have the value within the interval given by

$$\begin{aligned} Q_{F_A} &= Q_{F_{A1}} \cdot U_{F_{A1},F_A} + Q_{F_{A2}} \cdot U_{F_{A2},F_A} + ... \\ &\quad + Q_{F_{A5}} \cdot U_{F_{A5},F_A} = \\ &= \begin{cases} 5(1 \cdot 0.2) = 1 = Q_{F_A,max} \\ 3(0.5 \cdot 0.2) + 2(1 \cdot 0.2) = 0.7 = Q_{F_A,min} \end{cases} \end{aligned}$$

And using a virtual sensor (R_6) instead of the real sensor (R_1) we obtain

$$Q_{F_A} = 0.5 \cdot 0.2 + 4(1 \cdot 0.2) = 0.9$$

Example 2
Let us consider as an example a system composed of four resources (R_1, R_2, R_3 and R_4) and two functions (F_1 and F_1), which has the engineering data defined in table 4.1:

		R_1	R_2	R_3	R_4
Category		C_1		C_2	C_3
QoS		1	0.5	1	1
F_1	Utility	0.6		0.4	0
	Use degree	1		1	0
F_2	Utility	0.3		0	0.7
	Use degree	1		0	1

Table 4.1: Example of engineering data.

In order to run, function F_1 needs resources of category C_1 and C_2; and F_2 needs resources of category C_1 and C_3. The possible running modes of each function are depicted in table 4.2.

F_1	allocated resources	QoS
mode 1	R_1, R_3	1
mode 2	R_2, R_3	0.7

F_2	allocated resources	QoS
mode 1	R_1, R_4	1
mode 2	R_2, R_4	0.85

Table 4.2: Example of two possible running modes for each function

We also assume that both functions F_1 and F_2 must run simultaneously. In this case, both functions cannot run with the maximum QoS, so a decision must be made in such a way that *the loss of QoS is minimized*. That means, function F_1 will run in the first mode with $Q_{F_1} = 1$, and function F_2 in the second mode with $Q_{F_2} = 0.85$.

5 WiNC Platform

Biologically the species is the accumulation of the experiments of all its successful individuals since the beginning.

Herbert George Wells

An advanced experimental platform, called WiNC, has been developed, demonstrating the efficiency of the protocol with two well-known laboratory benchmarks with very different dynamics, namely the three-tank system and the inverted pendulum system.

5.1 Benchmarks description

Three-tank system

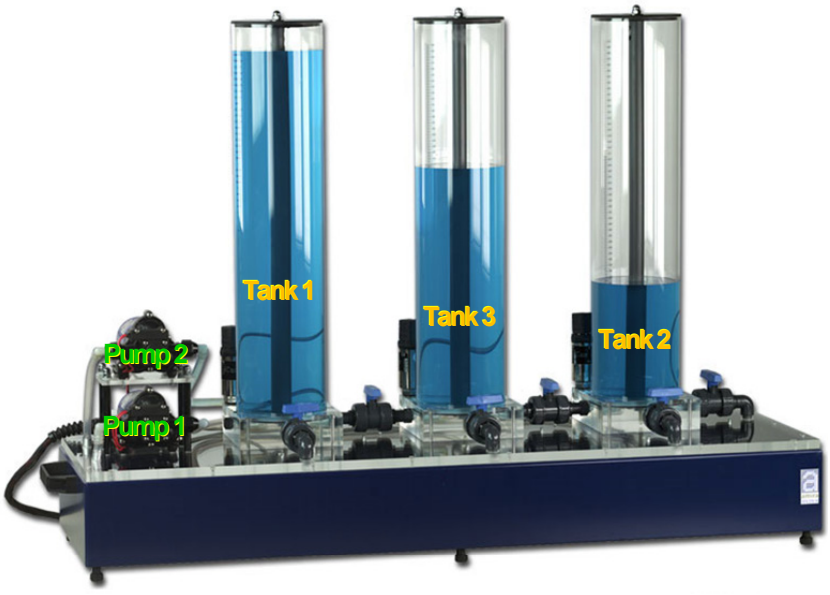


Figure 5.1: Three-tank laboratory setup.

The nonlinear system shown in Figure 5.1 consists of three plexiglas cylinders or tanks, which are interconnected in series by two pipes. The liquid (distilled water) leaving tank 2 is collected in a reservoir from which the two pumps supply it to the tanks 1 and 2. All three tanks are equipped with piezoresistive pressure transducer for measuring the level of the liquid.

The goal is to control the flow rates from the two pumps such that the levels in the tanks 1 and 2 can be independently specified. The level in middle tank is uncontrollable. The connecting pipes and the tanks are additionally equipped with manually adjustable valves and outlets for the purpose of simulating clogs as well as leaks [8].

Inverted pendulum system

The system is represented by a revolving pendulum mounted on top of a moving base. With a DC-motor, a toothed wheel, a toothed belt and a clutch, the moving base can be driven along a guiding bar over a length of approximately 1.5 m.

Figure 5.2: Inverted pendulum laboratory setup.

The goal is to stabilize the pendulum in the upright position at a certain location along the guiding bar. This is realized through a suitable signal that drives the DC-motor, which is based on the measurements represented by the pendulum angle and the cart position (both obtained by incremental encoders) [9].

5.2 Platform description

WiNC is an experimentation platform for wireless networked control, which was built with off-the-shelf hardware and open source software. The platform is composed essentially of an application, a driver and a protocol stack optimized for industrial wireless communication.

Due to the general tendency towards standardization and the fact that cheap, commercial off-the-shelf wireless technologies that do not require any sort of licensing are available, we have been motivated to investigate these for suitability in industrial deployment and adapt them to the real-time and reliability requirements of a networked control system.

After a careful investigation of the available solutions, we opted for wireless cards supporting IEEE 802.11a/b/g standards, having in this way the possibility of choosing different modulation methods and frequency bands, especially to escape the sometimes-crowded 2.4 GHz channels by using the 5 GHz band, which means less interference, therefore potential for more reliable transmission.

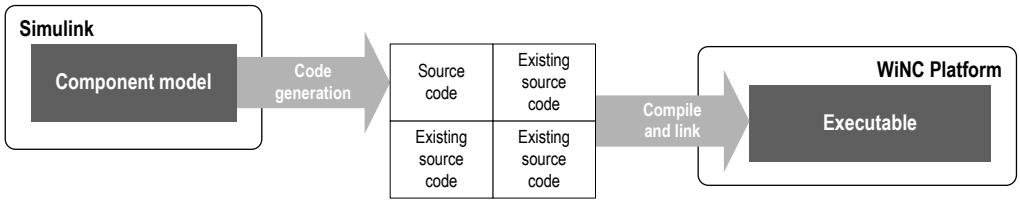


Figure 5.3: Integration of new components into the WiNC platform

In almost every available 802.11 hardware the MAC protocol employs a contention based channel access called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) described by the Distributed Coordination Function (DCF) [1], which lacks essentially deterministic behavior. Therefore, we decided to design a new MAC protocol optimized for industrial wireless communication using the available 802.11 based hardware. This was accomplished by using SoftMAC [85], an extension to the Multiband Atheros Driver for WiFi (MADWiFi) for WLAN cards based on certain Atheros chipsets.

In SoftMAC several measures are taken to provide real-time capability of the 802.11 MAC layer. The result is that the carrier sensing and the backoff procedure can be controlled and no additional packets, like Acknowledgement (ACK) or Request To Send/Clear To Send (RTS/CTS), occur. The nodes do not wait for acknowledgments and are forced to receive all messages that can be decoded by the physical layer, even those that contain checksum errors and would normally be discarded. SoftMAC allows user defined protocol implementation and header definitions.

In SoftMAC, the original 802.11 MAC header has been reduced to only five bytes at the beginning of a packet plus four-byte frame check sequence which is overhead that can not be avoided. Apart from that, the user can implement the whole protocol functionality based on the bit stream that is provided by the wireless card up to higher layers.

The WiNC platform is composed of the application that provides several functionalities like ones depicted in Figure 5.5, a driver and a protocol stack optimized for industrial wireless communication.

The application integrates the communication functionality of organizing the network access by use of a schedule based on time slots and tailored to the needs of the control system. It implements several control-specific functionalities including fault diagnosis and monitoring.

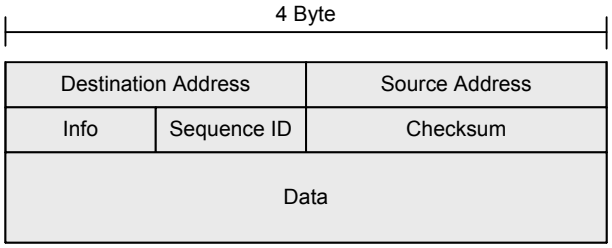


Figure 5.4: WiNC packet structure.

The designed protocol increases the reliability of a wireless network by employing several techniques like:

- reducing to minimum the size of a packet that is actually sent on air;
- retransmitting packets when they do not reach the destination within allocated or expected time;
- time diversity – transmitting critical packets (like control commands) multiple times.

Reducing the size of a data packet is done through a simplified addressing scheme that uses as node address only two bytes instead of four (as depicted in Figure 5.4), and also data structures optimized for the limited resolutions of A/D and D/A converters.

The protocol stack integrates a simplified addressing scheme with a custom packet structure and robust channel coding techniques, thus ensuring deterministic transmission times.

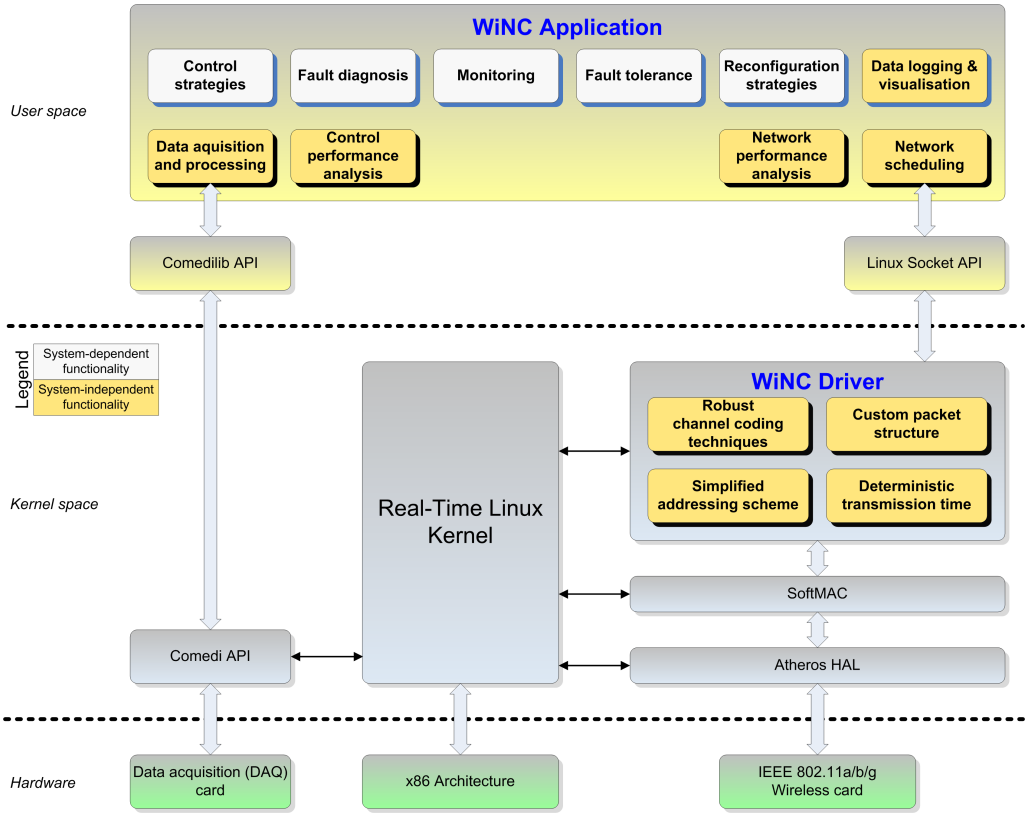


Figure 5.5: Structure of the WiNC platform

The employed hardware consists of desktop computers with x86 architecture microprocessors, equipped with a D-Link DWL-AG530 Tri-Mode Dual-band PCI card and a data acquisition card.

We have used Linux as the operating system (Fedora 6 with RT-Preempt Patch 2.6.20-rt8), the package SoftMAC, and – a collection of drivers for a variety of common data acquisition boards.

The WiNC application comprises several components that are shortly described in the following:

- Controller component – allows to change and tune the controller parameters for each subsystem (see Figure 5.8);

- Diagnosis component – allows to observe the current status of fault diagnosis algorithms (see Figure 5.9);
- Fault Tolerance component – allows to apply and choose between different fault tolerance strategies (see Figure 5.10);
- Network Performance component – offers detailed statistics about the transmitted and received data packets, including received power levels or Received Signal Strength Indication (RSSI) values (see Figure 5.7);
- Control Performance component – offers an overview of the overall system control performance (see Figure 5.6);
- Visualization component – plots the sensor measurements and control commands for each subsystem (see Figure 5.6, 5.8, 5.9 and 5.10);

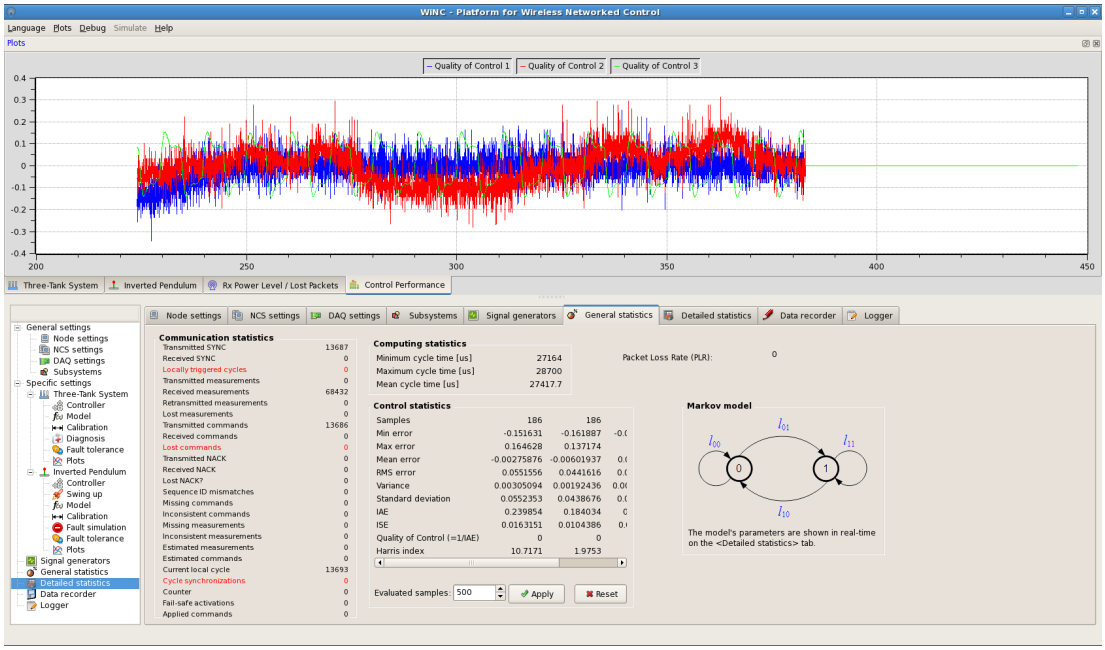


Figure 5.6: WiNC application: Control Performance component.

5.3 Multicast groups

Considering the addressing scheme presented in section 3.7, we define the multicast groups as shown in table 5.1 and Figure 5.11.

Only nodes that joined a certain multicast group can receive messages sent to the group address. Sending a message to a multicast group does not require the sending node to be part of that group.

Several tables, as follows, are defined and they must be initialized with data specific to each control system.

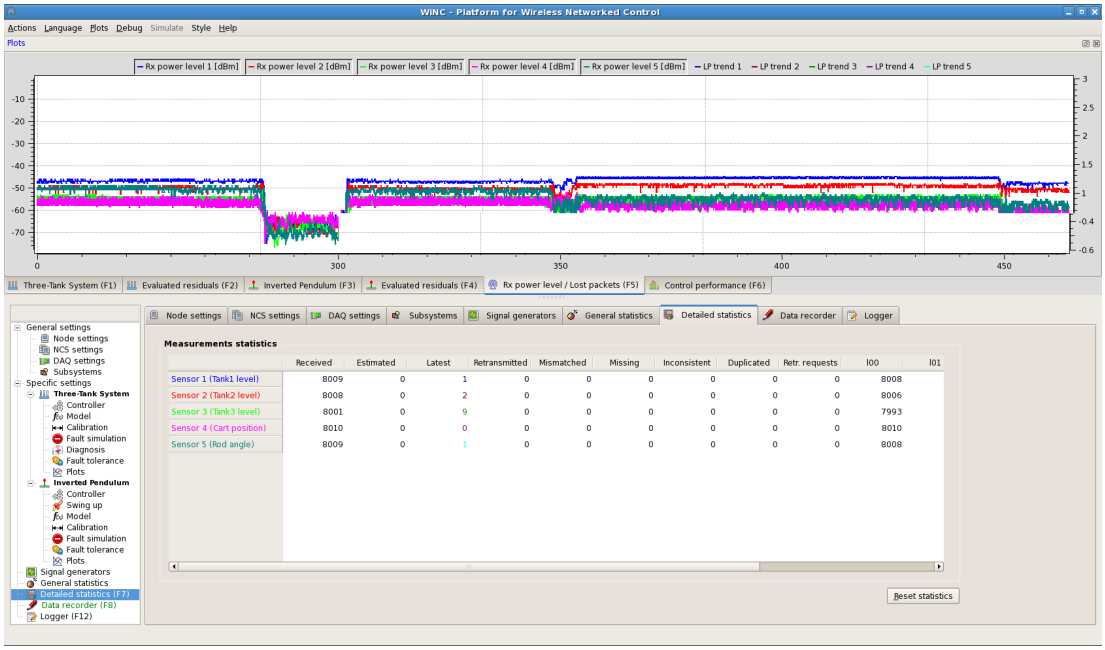


Figure 5.7: WiNC application: Network Performance component.

Group name	*Group address*	*Group nodes*
System group	255.255	all nodes
Subsystem 1 group	255.1	nodes of Subsystem 1
Subsystem 2 group	255.2	nodes of Subsystem 2
Sensor group	255.253	sensor nodes
Actuator group	255.254	actuator nodes

Table 5.1: Multicast groups.

- `AddressToId[Byte2][Byte1]` is a conversion table of a network node address to an unique identification number (UID) within the networked system.
- `IdToAddress[Id][2]` is initialized with data from `AddressToId[Byte2][Byte1]` table and it is used only to speed up the UID lookup process.
- `AddressToResolution[Byte2][Byte1]` stores the resolution of analog inputs and outputs for each sensor and actuator node, respectively.

5.4 Experimental results

We considered the two laboratory setups as subsystems – three-tank as *subsystem 1* and inverted pendulum as *subsystem 2* – belonging to a larger system that will be remotely controlled via wireless network, as it is shown in Figure 5.12. The first subsystem is composed of three level sensors and two pumps as actuators. The second subsystem has two sensors measuring cart position and rod angle, respectively, and one servomotor as actuator.

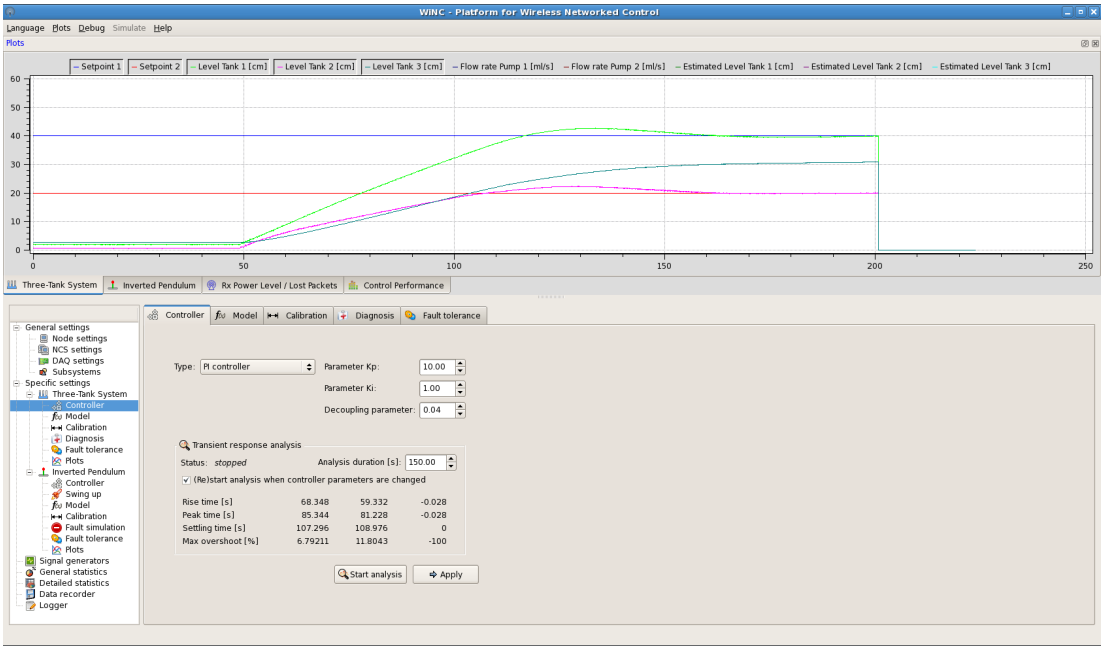


Figure 5.8: WiNC application: Controller component.

Each sensor and actuator constitute an independent wireless node, and so the whole system comprises eight nodes that are controlled and coordinated by a remote controller node.

The benchmark system is composed of a set of ten periodic tasks that involve network transmissions: a subset $\{F_1,\ F_2,\ F_3,\ F_4,\ F_5,\ F_6\}$ of six tasks ($N_{1,\text{tasks}} = 6$) belonging to the first subsystem with periods of $T_1 = 50ms$ and another subset $\{F_7,\ F_8,\ F_9,\ F_{10}\}$ of four tasks ($N_{2,\text{tasks}} = 4$) belonging to the second subsystem with periods of $T_2 = 30ms$, as shown in table 3.3.
The least common multiple of the two periods is $T = 150$ ms and represents the system cycle. Within one system cycle, the first subset of tasks will be executed for $n_1 = 3$ times and the second for $n_2 = 5$ times. The precedence constraints are depicted in Figure 5.13 and represented by the inequalities in (5.1). The first time slot of system cycle is always reserved for the transmission of a synchronization message, and control commands are transmitted to actuators three times in a row.

$$\begin{cases} s_1, s_2, s_3 < s_4 < s_5, s_6 \\ s_7, s_8 < s_9 < s_{10} \end{cases} \tag{5.1}$$

For the sake of simplicity, we will not consider here the tasks F_5, F_6 and F_{10}. Therefore, a total of only 43 tasks need to be scheduled within a system cycle of $T = 150$ ms. For a $T_{slot} = 1$ ms, there will be a total of 106 free time slots left in a system cycle that can be used within the Flexible Retransmission Request (FRR) scheme.

As an example, a system schedule for our benchmark was generated for $T_{slot} = 2.5$ ms with the help of *TORSCHE Scheduling Toolbox* for MATLAB [104], as shown in Figure 5.14. Control commands are sent three times for each subsystem hence the number of time slots allocated for tasks F_4 and F_9.

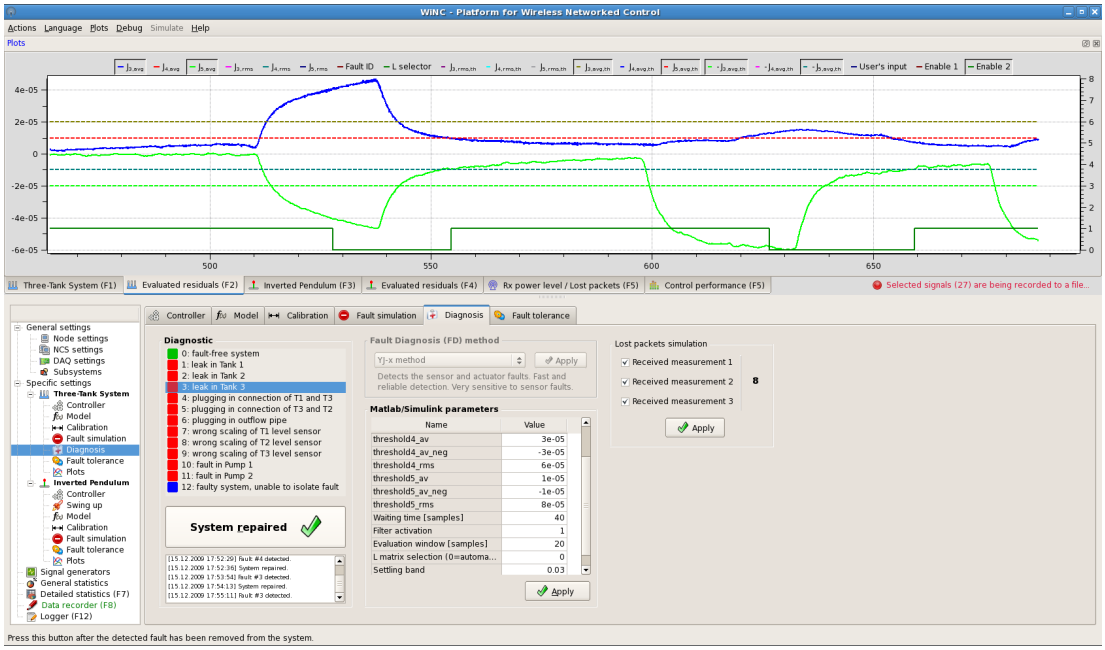


Figure 5.9: WiNC application: Diagnosis component.

In order to have a compact schedule, we grouped the tasks belonging to each subsystem in *task blocks* which can be regarded as one task, thus resulting a total of eight task blocks that must be scheduled. The defined block for first subsystem is formed of F_1, F_2 and F_3, and the block for second subsystem is composed of F_7, F_8 and F_9. They were scheduled as shown in Figure 5.15. The rest of the 16 free time slots left in a cycle will be used within FRR scheme.

To obtain the results shown in Figure 5.16 we ran the system over night with no people around, so that the wireless transmission channels remain relatively time invariant. We employed a fixed antenna setup with all sensors and actuators being attached to the plants in our laboratory and the controller being placed one floor below in the room beneath the laboratory. We chose the 2.4 GHz band and the Binary Phase Shift Keying (BPSK) modulation with the DSSS spreading technique which is expected to provide the most reliable communication of all available modulation methods of the 802.11a/b/g standards. The schedule shown in Figure 5.15 was applied.
Data was collected for two scenarios, one case without what we call FRR and a second case with FRR. For these scenarios, we varied the transmission power of all cards in the range from 1 dBm to 16 dBm. During all the trials the *probability of a lost sensor measurement* P_{lm} was measured.

According to WiNC protocol, for a sensor measurement to be successfully received, first the synchronization message has to be transmitted from the controller to the sensor and subsequently the sensor has to transmit successfully its packet to the controller. In the case of no retransmission, the P_{lm} corresponds to the probability of two subsequent successful transmissions.

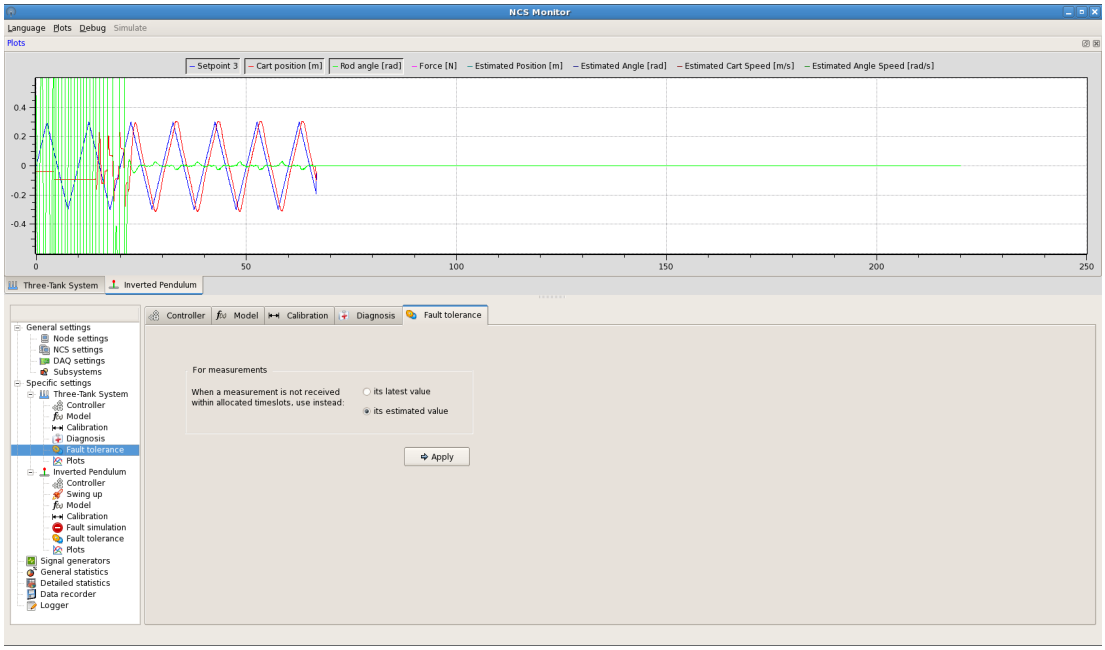


Figure 5.10: WiNC application: Fault Tolerance component.

5.5 Extended solution

The Radio Frequency (RF) environment in an industrial application also must content with a phenomenon called *multi-path*. This is the effect when radio waves strike a very dense object such as metal or stone, and they reflect off as a mirror reflects light. Some of the reflected waves reach the receiving antenna at different times than the intended signal and therefore out of phase. This effect can seriously degrade, and in some cases, completely cancel the received signal.

Since the industrial landscape is typically crowded with large metal structures, multi-path interference is especially problematic. One hardware solution to the multi-path problem is *antenna* or *space diversity*. Multiple antenna ports are on the receiving unit, and an internal circuit selects the antenna that receives the stronger signal [79]. This will be supported by the new IEEE 802.11n standard.

A pure software solution that looks promising is *cooperative diversity*, when nodes can help each other by relaying messages whenever the environment settings change.

In relaying schemes, there are a number of relay nodes that help in the transmission between a neighbored sender-receiver pair. All involved nodes can be single-antenna nodes. The relay nodes possibly receive the senders packet and can for example unconditionally forward their observations to the receiver node, or they could assist the sender with performing retransmissions when the receiver has not received the packet. Relaying can hence be tightly coupled to ARQ protocols.

The incorporation of spatial diversity approaches into error-control for wireless industrial networks is a very promising approach to significantly improve the reliability and real-time properties, and that it is a very interesting area of future research [114]. Relaying schemes can

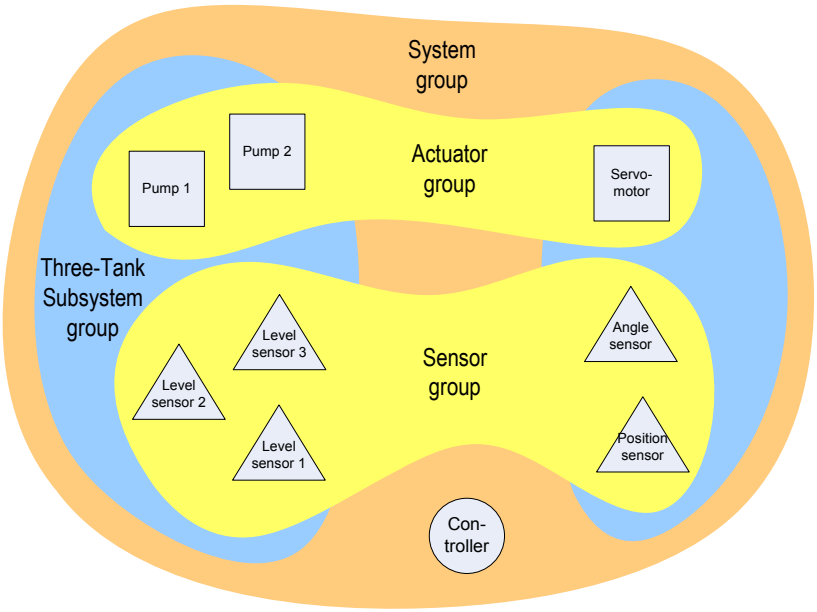


Figure 5.11: Defined multicast groups

give appreciable benefits for industrial communication systems by showing that the success probability can be significantly increased on fading channels as compared to classical error control schemes. In more realistic cases, the broadcast property of the wireless medium (or sleeping activities of sensor nodes) leads to situations where a random number of relayers can be present. The key research issue here is how to coordinate and control the activities of these relayers in a deadline-aware fashion. When relaying is integrated into an ARQ protocol (stated differently: when the activity of the relayers depends on feedback from the destination), then all relayers need a consistent view on the destination feedback. Unfortunately, some relayers can be mutual hidden terminals (i.e., they are not able to receive packets from each other), or a relayer can be hidden from the destination. A second important research issue are rules and heuristics for network planning and deployment that give hints for good positions of relayers. It is intuitively clear that a relayer is most helpful when placed right between source and destination, and that it is harmful when it is even farther away from the destination than the source node. In industrial networks it might be helpful to deploy extra nodes that do nothing else but to help with relaying, and for these nodes good positions need to be identified.

In its simplest form of the protocol, a sensor node can include in its data packet the measurements from few of the neighboring (closely located, within the transmission range) sensor nodes. In this way, the controller will still be able to timely receive all the measurements, in spite of a temporarily interrupted communication link of any sensor node with the controller. Another important advantage of this strategy is the possibility to check measurements for consistency and correct potential transmission errors.

Let us consider the example of a communication schedule for a system with one controller (C) and three sensor nodes (S_1, S_2, S_3), which need to send their measurements (m_1, m_2, m_3) to the controller. Sensor S_1 broadcasts its measurement m_1, then sensor S_2 broadcasts its measurement m_2 together with the previously received measurement from sensor S_1. And finally, sensor S_3 broadcasts its measurement m_3 together with measurement m_2 (see Figure 5.17). Therefore, the controller expects at the end of a system cycle two copies of the

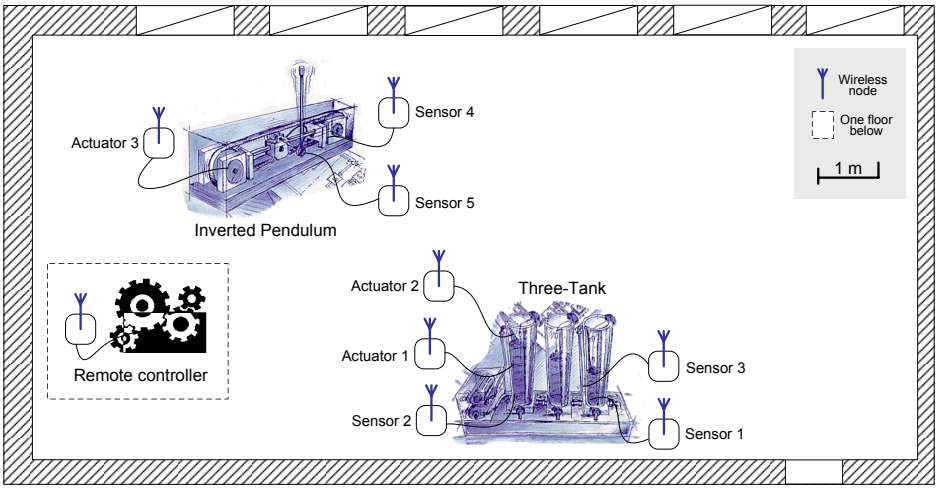
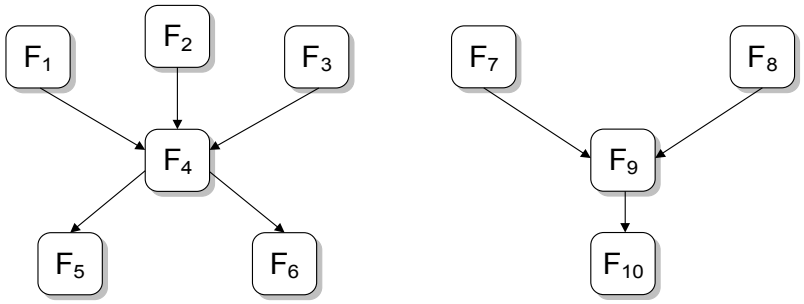


Figure 5.12: WiNC platform test bench. The sensor and actuator nodes are attached to the laboratory setups, and the controller is located one floor below.



(a) Three-Tank subsystem (b) Inverted Pendulum subsystem

Figure 5.13: Precedence graphs for the two subsystems. Nodes correspond to tasks, and edges are precedence constraints between these tasks.

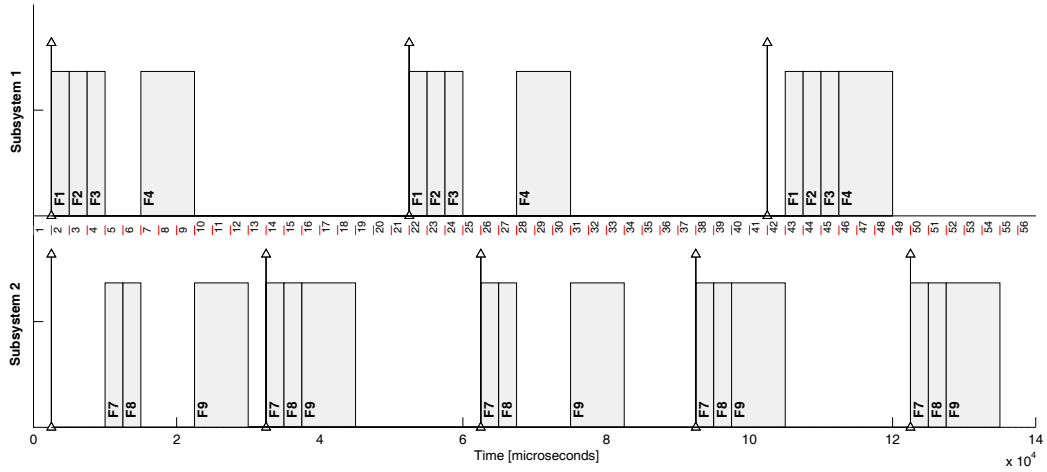


Figure 5.14: Example of generated system schedule for $T_{slot} = 2.5$ ms.

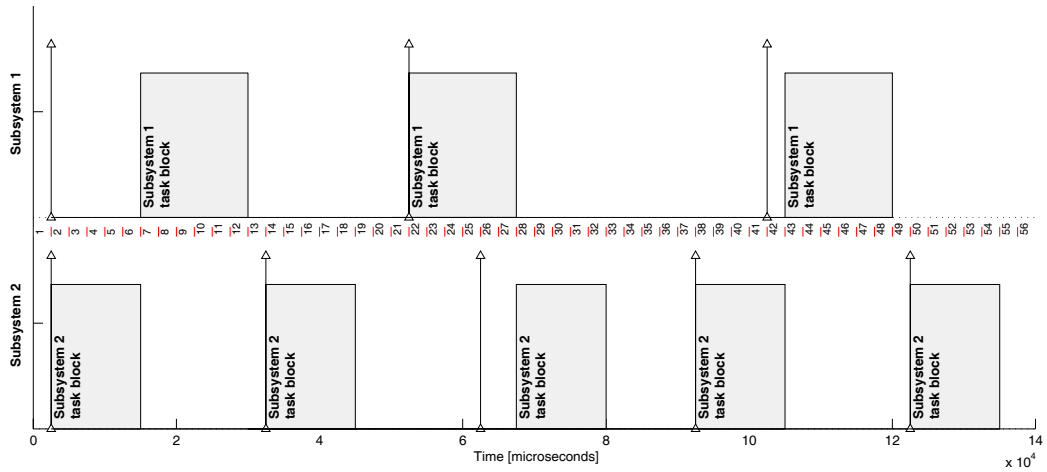


Figure 5.15: Example of generated system schedule with task blocks for $T_{slot} = 2.5$ ms.

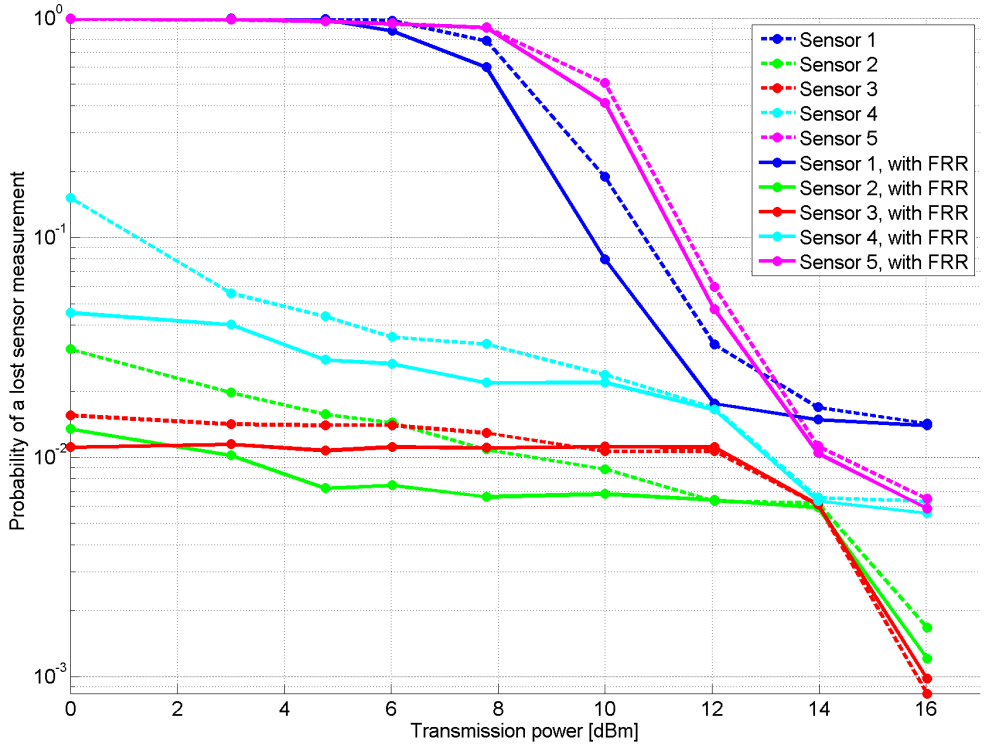


Figure 5.16: The probability of lost measurements P_{lm} for $T_{slot} = 2.5$ ms is displayed over the transmission power of all involved nodes. P_{lm} depends on the used protocol and includes at least the transmission of a synchronization packet and the transmission of a measurement packet. With the use of Flexible Retransmission Request (FRR) scheme, the P_{lm} can be significantly reduced.

measurement m_1, two of m_2 and one of m_3 (see Figure 5.18).

Re-scheduling of the communication would take place at run-time, based on the observed activity of all network nodes. Observed activity would include statistics like the number of

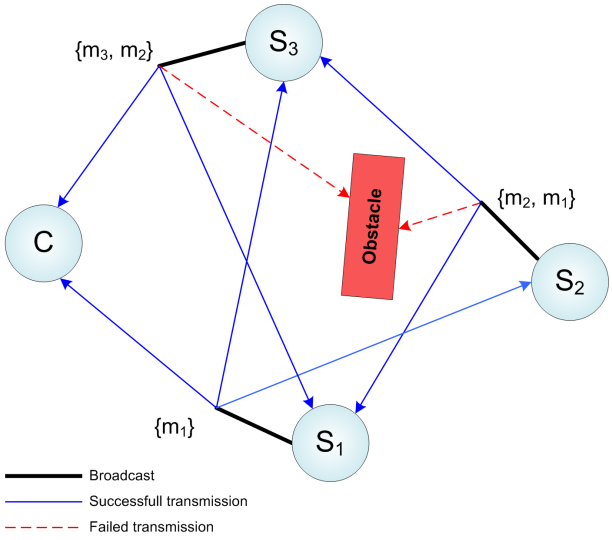


Figure 5.17: Relaying is an example of cooperating diversity.

retransmissions, the received signal power, etc.

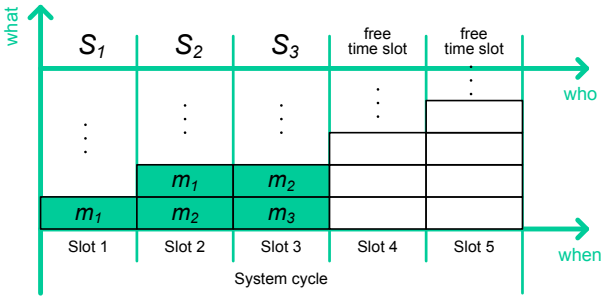


Figure 5.18: Three-parameter network schedule.

In conclusion, by extending our solution also with space and cooperative diversity, we can increase its reliability to a level that would make it employable even for mission-critical control systems.

5.6 Reliability model

To compute the overall failure rate and the reliability for a wireless networked control system, we consider the series connection between the elements, i.e. the nodes that are composing the WNCS. Therefore, the weakest communication link within the system will affect the overall reliability. For example, in Figure 5.21 it is shown the overall reliability and failure rate for the three-tank and inverted pendulum systems, computed with (3.12), (3.13) and (3.14).

The developed FRR scheme allows, as the name says, the retransmission of measurements by the sensor nodes on the request of the controller node. The reliability of this mechanism can be modeled as shown in Figure 5.19, where the overall reliability function is given by equation 5.2 in which m is the number of sensor nodes within the system, and n is the number of measurement retransmissions.

	Fault mode	*Fault cause*	*Fault effect*	*Fault handling*	*Fault detection*	*Criticality*
1	MEASUREMENT message is missing.	Message got lost.	No effect.	Controller asks for a retransmission by sending a NACK message.	<estimated/latest measurements> variable is *occasionally* incremented.	Safe
		Corresponding sensor failed.	Corresponding controlled subsystem might become unstable.	Controller uses instead the estimated measurement.	<estimated/latest measurements> variable is *successively* incremented – user-defined threshold.	Dangerous
2	SYNC message is missing.	Message got lost.	Corresponding controlled subsystem might become unstable.	A new cycle is locally triggered. When the <locally triggered cycles> variable reaches a certain threshold, the sensor/actuator node switches to the fail-safe mode, i.e. wakes up only when a new SYNC message arrives.	<locally triggered cycles> variable is *occasionally* incremented.	Safe
		Coordination node failed.	Whole system might become unstable.	When the <locally triggered cycles> variable exceeds the user-defined threshold, a redundant coordination node takes over the coordinator role.	<locally triggered cycles> variable is *successively* incremented – user-defined threshold.	Dangerous
3	COMMAND message is missing.	Message got lost.	Corresponding controlled subsystem might become unstable.	Actuator uses the latest available command.	<latest commands> variable is *occasionally* incremented.	Safe
		Controller node failed.	Corresponding controlled subsystem might become unstable.	When the <latest commands> variable exceeds the user-defined threshold, the actuator node switches to the fail-safe mode, i.e. wakes up only when a new SYNC message comes.	<latest commands> variable is *successively* incremented – user-defined threshold.	Dangerous

Table 5.2: Communication fault scenarios in a WNCS.

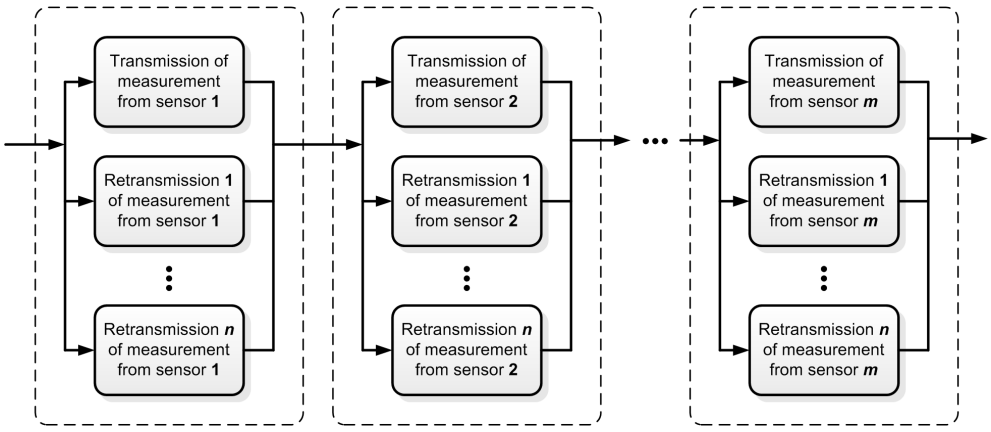


Figure 5.19: Overall system reliability R_{system} for different number of retransmissions.

$$R_{system}(t) = \prod_{1}^{m} \left[1 - \prod_{i=1}^{n+1} (1 - R_i(t)) \right] \tag{5.2}$$

The overall reliability of the WNCS substantially improves when the FRR mechanism is applied. In Figure 5.20 it is shown that the more retransmissions are scheduled, the more reliable the WNCS becomes. It can be noticed that for a link reliability of 0.5, the overall reliability increases sevenfold when only one retransmission per link is used and sixteenfold when two retransmissions are employed.

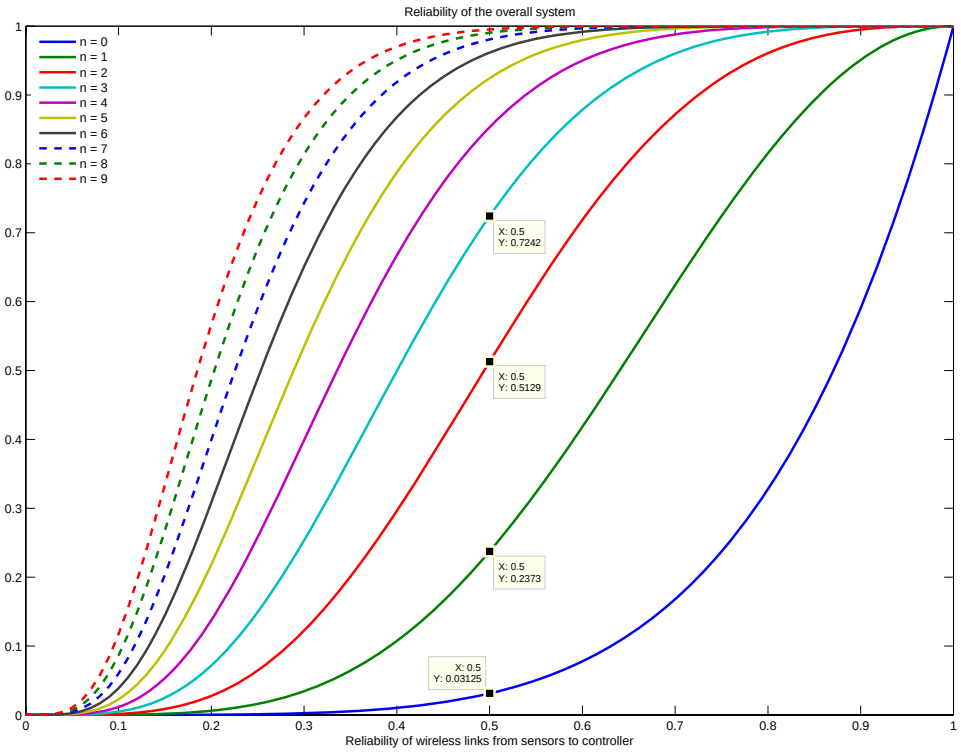


Figure 5.20: Overall system reliability R_{system} for different n number of retransmissions, considering identical reliability values for the wireless links from sensors to controller.

Using only retransmissions does not entirely solve the problem, especially when the antenna of a sensor or actuator node is "badly" located relative to antenna of the controller node. This problem requires to rethink the communication protocol so that it is able to cope with these situations. Closely located sensor nodes could relay information for each other, until it reaches the controller nodes.

The reliability of each wireless link can be further used by the FD module for the adaptation of the thresholds in order to avoid false alarms or missed detections.

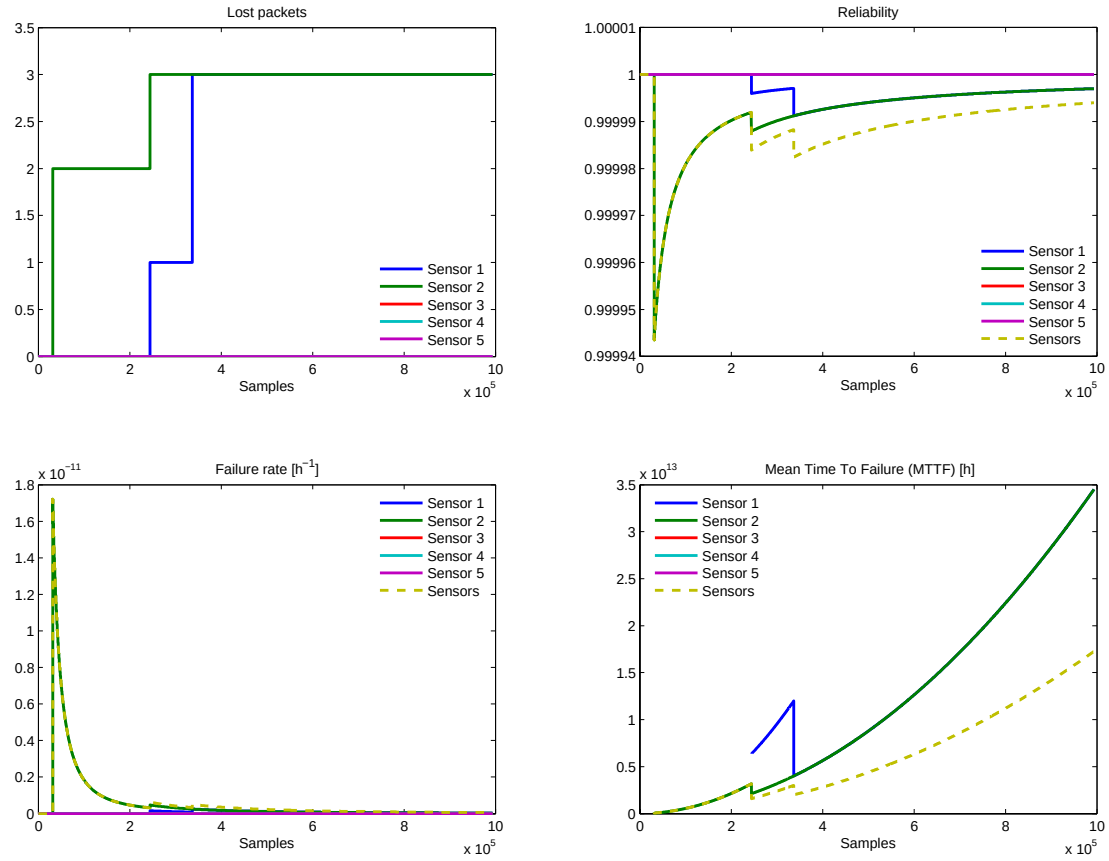


Figure 5.21: Reliability, MTTF and the failure rate of each wireless link of a sensor node relative to the controller node.

Experimental results (see Figure 5.24 and Figure 5.25) show that the power level of the measurement packets received by the controller node is to a great extent influenced by several factors, like moving objects interposed between the the two antennas, or even a small change in the orientation of the controller's antenna. Even though packet loss occurs during such events, the control errors and the stability of the two subsystems are not affected.

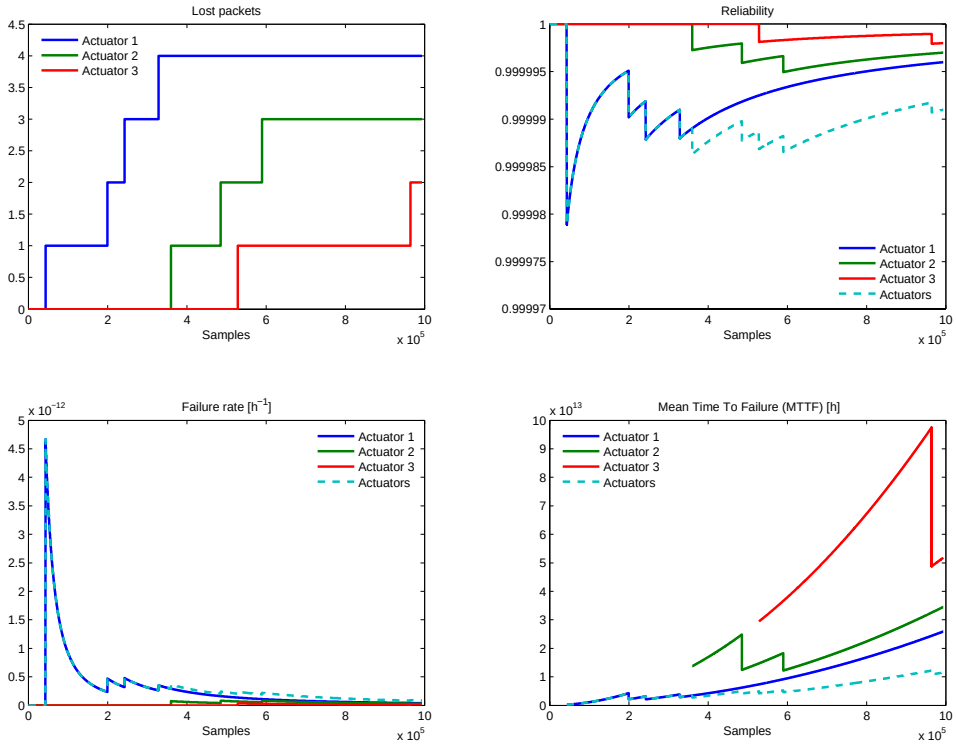


Figure 5.22: Reliability, MTTF and the failure rate of each wireless link of a actuator node relative to the controller node.

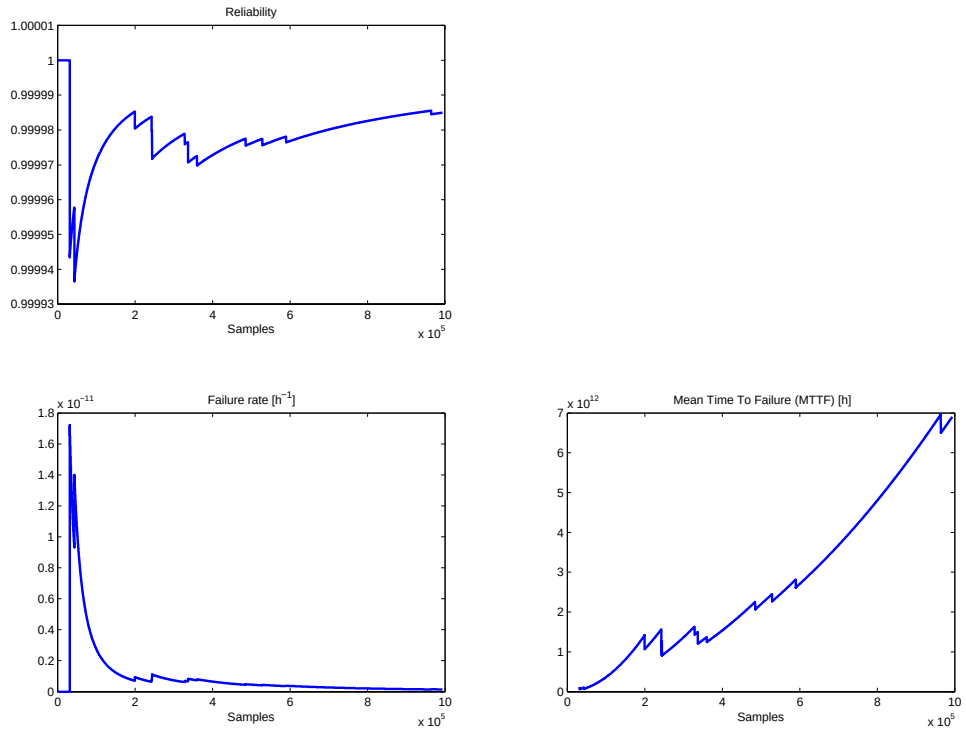


Figure 5.23: Reliability, MTTF and the failure rate of the overall system.

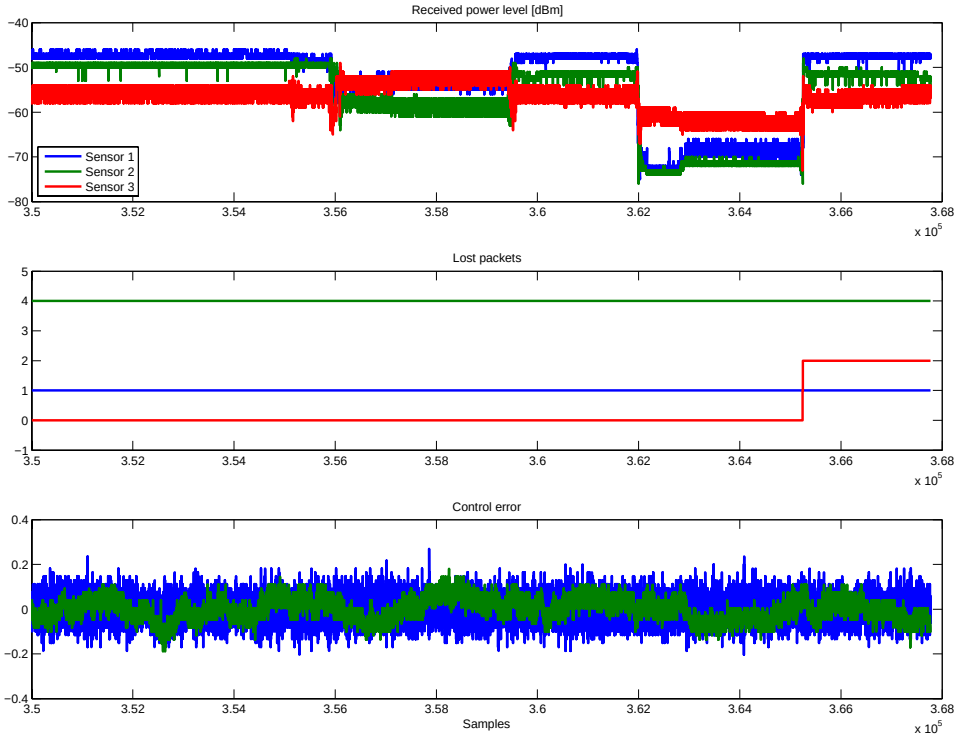


Figure 5.24: The power level of the measurement packets received by the controller node is influenced by several factors.

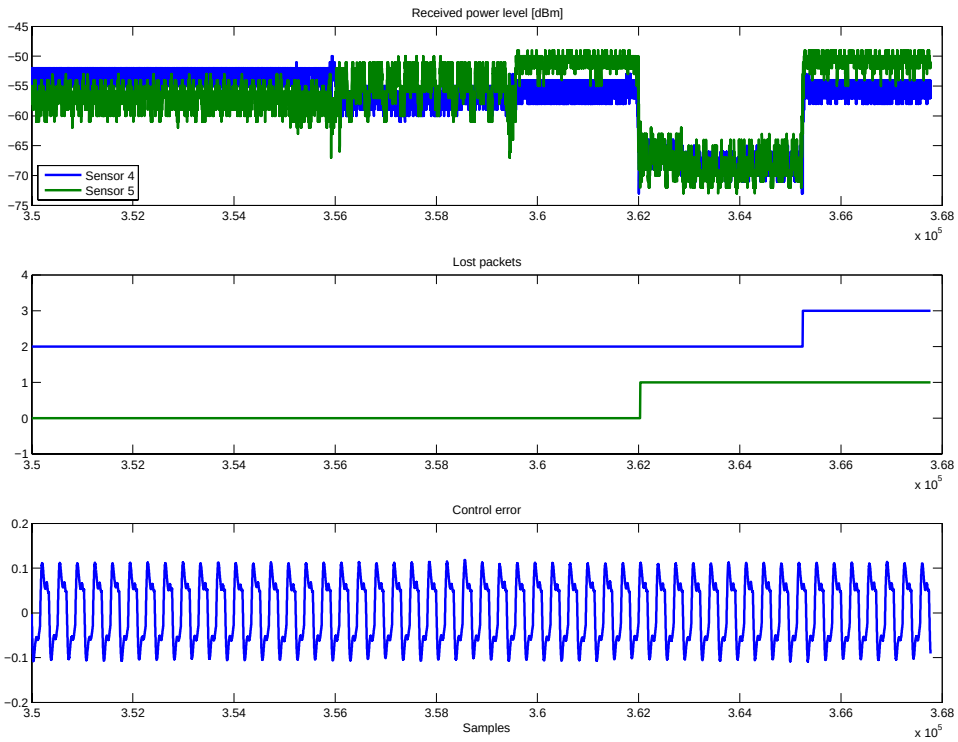


Figure 5.25: The power level of the measurement packets received by the controller node is influenced by several factors.

6 Conclusions and future directions

Here is a thing about the future... every time you look at it, it changes. Because you look at it, and that changes everything else.

Frank Cadillac

This work proposes a novel real-time communication protocol for wireless networked control systems (WNCS). The protocol is based on the TDMA strategy and has built-in tolerance against the network-induced effects like lost packets, assuring a highly deterministic and reliable behavior of the overall networked control system, thus allowing the use of classical control design methods with WNCS. Determinism in the transmission times, for both sending and for receiving, is assured by a communication schedule that is dynamically updated based on the conditions of the network and the propagation environment.

An advanced experimental platform, called WiNC, has been developed, demonstrating the efficiency of the protocol with two well-known laboratory benchmarks with very different dynamics, namely the three-tank system and the inverted pendulum system. Wireless nodes belonging to both systems are coordinated and synchronized by a master node, namely the controller node.
The WiNC platform, having the main features of a flexible rapid control prototyping system, offers the possibility also to other research groups to experiment, test and optimize new monitoring and fault tolerant control designs for the WNCSs, all without any manual programming. The C code corresponding to the Matlab/Simulink models is automatically generated and then integrated into the WiNC application.

The WiNC platform uses only open source software and general-purpose (commercial, off-the-shelf) hardware, thus making it with a minimal investment (low cost) a flexible and easily extendable research platform for WNCS. And considering the general trend towards the adoption of Linux as a real-time operating system for embedded system in automation, the developed concepts and algorithms can be ported with minimum effort to the industrial embedded devices which already run Linux.
Several FD methods (both open-loop and closed-loop), for both benchmark systems, have been designed and then successfully implemented and tested on the WiNC platform, demonstrating that it is just a small step from theory to practice, as long as the right software tools are available.

The RF environment in an industrial application also must content with a phenomenon called *multi-path propagation*. This is the effect when radio waves strike a very dense object such as metal or stone, and they reflect off as a mirror reflects light. Some of the reflected waves

reach the receiving antenna at different times then the intended signal and therefore out of phase. This effect can seriously degrade, and in some cases, completely cancel the received signal.
Since the industrial landscape is typically crowded with large metal structures, multi-path interference is especially problematic. A hardware solution would be *antenna* or *space diversity*. Multiple antenna ports are on the receiving unit, and an internal circuit selects the antenna that receives the stronger signal. This will be supported by the new IEEE 802.11n standard. A pure software solution that looks promising is *cooperative diversity*, when nodes can help each other by relaying messages whenever the environment settings change. By extending our solution also with cooperative and space diversity, its robustness and reliability can be increased to a level that would make it employable even for mission-critical control systems.

Our main focus was the control and monitoring of plants composed of a small number of spatially close located subsystems. Within such a subsystem it was assumed that all nodes are in the transmission range of the controller and the number of sensors and actuators is small enough so that a time slot structure is feasible, i.e. there is a free slot for every necessary transmission. However, in large-scale plants it is likely that the number of subsystems is quite large, within a subsystem there might exist a large number of sensors or actuators which are spread over a large area. Up to a certain degree one may assume that one can control all subsystems like we proposed by assigning different frequency bands to the subsystems and the use of another technology for data exchange between subsystems. But the spectrum is scarce and it is not desirable to use different wireless technologies at the same time. When the plants are huge and the number of subsystems is high, concepts from cellular networks might be applicable to achieve a proper sharing of spectrum. Cellular concepts require usually a wired backbone network that connects the different cells or subsystems. Every subsystem might form a cell in the network that is spatially separated to other cells. This only works under the condition that subsystems are not spatially nearby which is not necessarily the case.

Because of the possible large number of sensors within only one subsystem, there might not be enough time slots for all sensors to transmit their measurements within the control period. A pure TDMA based medium access method would not feasible in this case, considering that the length of a time slot is a physical constraint imposed by the employed wireless technology. In such scenarios, some transmissions have to occur at the same time. Results from multi-antenna communication systems might be applicable here, like spatial multiplexing where several antennas are used to create spatial channels.

For practical systems a very important property that was not dealt with is *data security*. It is important that a potential attacker can not eavesdrop into current communication, so-called packet sniffing. And it is even more important that it is not possible to send fabricated information, so-called packet injection, that might cause system breakdowns. A cooperation with encryption experts might lead to new concepts, considering that in the case of WNCS the payload size is so small that typical encryption mechanisms that rely on long keys may not be applicable here.

Bibliography

[1] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. ANSI/IEEE Std 802.11, 1999 Edition.

[2] X.200 : Information technology - Open Systems Interconnection - Basic Reference Model: The basic model, 1994. ISO/IEC 7498-1:1994.

[3] Quality of Service Definition. Technical report, SEQUIN Project Deliverable D2.1, IST-1999-20841, 2001.

[4] Sherif Abdelwahed, Gabor Karsai, and Gautam Biswas. System Diagnosis using Hybrid Failure Propagation Graphs. *The 15th International Workshop on Principles of Diagnosis*, December 2004.

[5] Ian F. Akyildiz and Ismail H. Kasimoglu. Wireless sensor and actor networks (WSANs): research challenges. *Elsevier Ad Hoc Networks*, 2:351–367, May 2004.

[6] Ibrahim Al-Salami. *Observer-based Fault Detection in Networked Control Systems*. PhD thesis, Universität Duisburg-Essen, 2008.

[7] Gülgün Alpan. *Design and analysis of supervisory controllers for discrete event dynamic systems*. PhD thesis, 2004.

[8] Amira. *Practical instructions for the Inverted Pendulum PS600 laboratory setup*.

[9] Amira. *Practical instructions for the Three-Tank DS200 laboratory setup*.

[10] T. Anderson and P.A. Lee. *Fault tolerance, principles and practice*. Prentice-Hall International, 1981.

[11] J. D. Andrews and T. R. Moss. *Reliability and Risk Assessment*. Professional Engineering Publishing, 2nd edition, 2002.

[12] Jim Arlow and Ila Neustadt. *UML and the Unified Process*. Addison-Weasley, 2002.

[13] ARTIST2 Control for Embedded Systems Cluster. Roadmap on Real-Time Techniques in Control System Implementation. Technical report, EU/IST FP6 Artist2 NoE, 2005.

[14] Héctor Benítez-Pérez and Fabián García-Nocetti. *Reconfigurable Distributed Control*. Springer, 2005.

[15] A. Birolini. *Reliability engineering - Theory and Practice*. Springer, 3rd edition, 1999.

[16] Mogens Blanke, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer Verlag, 2nd edition, 2006.

[17] Bob Blumenscheid. Comparing WLAN and ZigBee for embedded applications. Industrial Control Designline Magazine, April 2007.

[18] Robert W. Brennan, Fletcher Martyn, and Norrie Douglas H. An agent-based approach to reconfiguration of real-time distributed control systems. *IEEE Transactions on Robotics and Automation*, 18(4):444–451, 2002.

[19] Giorgio C. Buttazo. *Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications*. Springer, 2nd edition, 2005.

[20] Fabrizio Caccavale and Luigi Villani, editors. *Fault Diagnosis and Fault Tolerance for Mechatronic Systems*. Springer, 2003.

[21] Edgar H. Callaway. *Wireless Sensor Networks: Architectures and Protocols*. CRC Press, 2004.

[22] Luciano Carotenuto and Paolo Pugliese. A probabilistic approach to the intermittent measurements problem. In *Proceedings of the European Control Conference (ECC) 2007*, pages 591–596, July 2007.

[23] Anton Cervin. *Integrated Control and Real-Time Scheduling*. PhD thesis, Department of Automatic Control Lund Institute of Technology, Lund, Sweden, April 2003.

[24] C. I. Chihaia, O. Bredtmann, E. Goldschmidt, W. Li, S. X. Ding, and A. Czylwik. WiNC - Advanced Experimentation Platform for Wireless Networked Control. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SafeProcess 2009)*, 2009.

[25] Nirav N. Chokshi and Duncan C. McFarlane. *A Distributed Coordination Approach to Reconfigurable Process Control*. Springer London, 2008.

[26] Matjaž Colnaric and Domen Verber. *Distributed Embedded Control Systems*. Springer, 2008.

[27] Francis Cottet, Joelle Delacroix, Claude Kaiser, and Zoubir Mammeri. *Scheduling in real-time systems*. John Wiley & Sons, 2002.

[28] George Coulouris. *Distributed systems - Concepts and design*. Pearson, 2001.

[29] D.Brevi, D.Mazzocchi, R.Scopigno, A.Bonivento, R. Calcagno, and F.Rusina. A Methodology for the Analysis of 802.11a Links in Industrial Environments. *IEEE International Workshop on Factory Communication Systems*, pages 165–174, June 2006.

[30] Dirk Düding. *Ein Beitrag zum Einsatz von echtzeitfähigen Linux-Varianten in der Automatisierungstechnik*. PhD thesis, Universität Dortmund, 2003.

[31] S. X. Ding and P. Zhang. An observer based fault tolerant scheme for distributed networked control systems. *Proc. of 2007 European Control Conference*, pages 3687–3694, 2007.

[32] S. X. Ding, P. Zhang, C. I. Chihaia, W. Li, Y. Wang, and E. L. Ding. Advanced design scheme for fault tolerant distributed networked control systems. *Proceedings of the 17th IFAC World Congress*, pages 13569–13574, Juli 2008.

[33] Steven X. Ding. *Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools*. Springer, 2008.

[34] Richard D. Dorf and Robert H. Bishop. *Modern Control Systems*. Pearson Education International, 10th edition, 2005.

[35] Bruce Powell Douglass. *Real-time Design Patterns - Robust Scalable Architecture for Real-time Systems*. Addison-Wesley Professional, 2003.

[36] Guillaume J.J. Ducard. *Fault-tolerant Flight Control and Guidance Systems - Practical Methods for Small Unmanned Aerial Vehicles*. Springer, 2009.

[37] J. Eker, A. Cervin, and A. Hörjel. Distributed wireless control using Bluetooth. In *Proceeedings of the IFAC Conference on New Technologies for Computer Control 2001*, Hong Kong, 2001.

[38] Tim Enwall. Deploying Wireless Sensor Networks for Industrial Automation & Control, May 2006.

[39] Michael Epstein, Ling Shi, Stefano Di Cairano, and Richard M. Murray. Control Over a Network: Using Actuation Buffers to Reduce Transmission Frequency. In *Proceedings of the European Control Conference (ECC) 2007*, pages 597–602, July 2007.

[40] Adrian Gambier. Real-time Control Systems: A Tutorial. *5th Asian Control Conference*, 2:1024–1031, 2004.

[41] Deepak Ganesan and Alec Woo. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Technical Report UCLA/CSD-TR 02-0013, University of California, Los Angeles (UCLA), Computer Science Department, 2002.

[42] William M. Goble. *Control Systems Safety Evaluation and Reliability*. Instrument Society of America (ISA), 2nd edition edition, 1998.

[43] Andrea Goldschmidt. *Wireless Communications*. Cambridge University Press, 2005.

[44] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, pages 287–326, 1979.

[45] S. Graham and P.R. Kumar. Time in general-purpose control systems: The Control Time Protocol and an experimental evaluation. *43rd IEEE Conference on Decision and Control*, 4:4004– 4009, 2004.

[46] Dean A. Gratton. *Developing Practical Wireless Applications*. Elsevier, 2007.

[47] Dazhang Gu and Lonnie Welch. *Metaheuristics for scheduling in distributed computing environments*, volume 146 of *Studies in Computational Intelligence*, chapter Robust Allocation and Scheduling Heuristics for Dynamic, Distributed Real-Time Systems, pages 61–93. Springer Berlin, 2008.

[48] Wolfgang A. Halang. Simplicity considered fundamental to design for predictability. In Lothar Thiele and Reinhard Wilhelm, editors, *Perspectives Workshop: Design of Systems with Predictable Behaviour*, number 03471 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2004.

[49] Gerhard P. Hancke and Ben Allen. Ultrawideband as an Industrial Wireless Solution. *IEEE Pervasive Computing*, 5(4):78–85, December 2006.

[50] HART Communications Foundation (HCF). http://www.hartcomm2.org.

[51] J. K. Hedrick, M. Uchanski, and Q. Xu. Enhanced AHS Safety Through the Integration of Vehicle Control and Communication. Technical Report UCB-ITS-PRR-2001-28, University of California, Berkeley, 2001.

[52] Joao P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A Survey of Recent Results in Networked Control Systems. In *Proceeeding of the IEEE*, volume 95, pages 138–162, January 2007.

[53] Dimitrios Hristu-Varsakelis and William S. Levine, editors. *Handbook of Networked and Embedded Control Systems*. Birkhäuser, 2005.

[54] Andreas Hörjel. Bluetooth in Control. ISRN LUTFD2/TFRT-5659-SE, Department of Automatic Control Lund Institute of Technology, Lund, Sweden, January 2001.

[55] Nick Hunn. Choosing a Short Range Wireless Technology. White Paper, 2005.

[56] Mohammad Ilyas and Imad Mahgoub, editors. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, 2005.

[57] Orhan C. Imer. *Optimal Estimation and Control under Communication Network Constraints*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[58] Orhan C. Imer, Serdar Yuksel, and Tamer Basar. Optimal control of LTI systems over unreliable communication links. *Automatica*, 42(9):1429–1439, 2006.

[59] ISA. ISA-SP100.11 Call for Proposal - Wireless for Industrial Process Measurement and Control, July 2006.

[60] Alf Isaksson. Wireless networked control - Challenges from Automation Industry. Network of Excellence HYCON (Hybrid Systems Control): Taming heterogenity and complexity of networked embedded systems, June 2006.

[61] Rolf Isermann. *Fault-Diagnosis Systems - An Introduction from Fault Detection to Fault Tolerance*. Springer, 2006.

[62] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2006.

[63] Jan M. Kościelny, Marcin Leszczyński, and Michał Z. Bartyś. Investigations of Fault Tolerant Systems. *First Workshop on Networked Control System and Fault Tolerant Control*, 2005.

[64] J. Nicholas Laneman. *Cooperative Diversity in Wireless Networks: Algorithms and Architectures*. PhD thesis, Massachusetts Institute of Technology (MIT), 2002.

[65] Chris LeBlanc. The Future of Industrial Networking and Connectivity. *Dedicated Systems Magazine*, http://www.dedicated-systems.com, 2000.

[66] Insup Lee, Joseph Y-T. Leung, and Sang H. Son, editors. *Handbook of Real-Time and Embedded Systems*. Chapman & Hall/CRC, 2008.

[67] Joseph Y-T. Leung, editor. *Handbook of scheduling*. Chapmann & Hall/CRC, 2004.

[68] W. Li, P. Zhang, S. X. Ding, and O. Bredtmann. Fault detection over noisy wireless channel. *IEEE Conference on Decision and Control*, 2007.

[69] W. Li, P. Zhang, S. X. Ding, C. I. Chihaia, E. Goldschmidt, O. Bredtmann, and A. Czylwik. Networked Fault Detection Systems with Noisy Data Transmission. *at – Automatisiderungstechnik 56*, 56, 2008.

[70] Wei Li. *Observer-based Fault Detection of Technical Systems over Networks*. PhD thesis, Universität Duisburg-Essen, 2009.

[71] Feng-Li Lian, James Moyone, and Dawn Tilburry. Network design consideration for distributed control systems. *IEEE Transactions on Control Systems Technology*, 10(2):297–307, March 2002.

[72] Xiangheng Liu and Andrea Goldsmith. Wireless Network Design for Distributed Control. *CDC. 43rd IEEE Conference on Decision and Control*, 3:2823–2829, December 2004.

[73] J. Lunze and J. Richter. Control reconfiguration: Survey of methods and open problems. Technical report, 2006.

[74] J. Lunze and J. H. Richter. Reconfigurable Fault-tolerant Control: A Tutorial Introduction. *European Journal of Control*, 5:359–386, 2008.

[75] Jan Maciejowski. Discussion on: Reconfigurable Fault-tolerant Control: A Tutorial Introduction. *European Journal of Control*, 5:387–390, 2008.

[76] J.M. Maciejowski. *Diagnosis of Processes and Systems*. PWNT Gdańsk, 2009.

[77] Mufeed Mahmoud, Jin Jiang, and Youmin Zhang. *Active Fault Tolerant Control Systems*. Springer, 2003.

[78] G. Matos and E. White. Application of dynamic reconfiguration in the design of fault-tolerant production systems. In *Proceedings of Fourth International Conference on Configurable Distributed Systems*, pages 2–9, 1998.

[79] Patrick McCurdy, Ira Sharp, and Nicholas Sheble. *The best Ethernet may be coming*. ISA InTech, August 2008.

[80] Ian McPherson. Industrial Wireless: Hope, help or hype? The Industrial Wirless Book.

[81] Balachandran Natarajan, Chris D. Gill, A. Gokhale, and D. Schmidt. Towards Dependable Real-time and Embedded CORBA Systems. *IEEE Workshop on Dependable Middleware-Based Systems*, 2002.

[82] Virtual Automation Networks. State of the Art and Trends in Safety, Security, Wireless Technologies and Real-time Properties (Deliverable D01.1-1-V1). Technical report, European Project FP6/2004/IST/NMP/2 - 016696 VAN, December 2005.

[83] Virtual Automation Networks. State of the Art and Trends in Safety, Security, Wireless Technologies and Real-time Properties (Deliverable D01.1-1-V2). Technical report, European Project FP6/2004/IST/NMP/2 - 016696 VAN, September 2006.

[84] Virtual Automation Networks. Trend Screening Report on VAN Relevant Technologies (Deliverable D01.3-1-V2). Technical report, European Project FP6/2004/IST/NMP/2 - 016696 VAN, September 2006.

[85] Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth, and Dirk Grunwald. SoftMAC – Flexible Wireless Research Platform. *HotNets*, IV, November 2005.

[86] Johan Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, 1998.

[87] Mattias Nyberg. Design of an complete FDI system based on performance index with application to an automotive engine. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 812–817, 1997.

[88] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, 4th edition, 2002.

[89] Hewlett Packard. The Linux Wireless LAN Howto.

[90] Andrea Paoli. *Fault Detection and Fault Tolerant Control for Distributed Systems: a general framework*. PhD thesis, University of Bologna, 2004.

[91] Tal Pasternak. *Hierarchical control reconfiguration for a class of hybrid systems*. PhD thesis, Vanderbilt University, August 2002.

[92] R.J. Patton, C. Kambhampati, , and F. J. Uppal. Challenges of Networked Control Systems: Autonomy, Reconfiguration and Plug-and-play. *First Workshop on Networked Control System and Fault Tolerant Control*, 2005.

[93] Francesco De Pellegrini, Daniele Miorandi, Stefano Vitturi, and Andrea Zanella. On the Use of Wireless Networks at Low Level of Factory Automation Systems. *IEEE Transactions on Industrial Informatics*, 2(2):129–143, May 2006.

[94] John G. Proakis. *Digital Communications*. McGraw-Hill, 4th edition edition, 2001.

[95] TSMP (Time Synchronized Mesh Protocol). http://www.dustnetworks.com.

[96] Theodore S. Rappaport. *Wireless Communications - Principles and Practice*. Prentice Hall PTR, 2nd edition edition, 2002.

[97] Ola Redell. Global Scheduling in Distributed Real-Time Computer Systems - An Automatic Control Perspective. Technical report, Royal Institute of Technology, Sweden, 1998.

[98] Luca Schenato. To zero or to hold control inputs in lossy networked control systems? In *Proceedings of the European Control Conference (ECC) 2007*, pages 585–590, 2007.

[99] Peter Scholz. *Softwareentwicklung eingebetteter Systeme - Grundlagen, Modellierung, Qualitätssicherung*. Springer Berlin Heidelberg, 2005.

[100] Ehsan Sobhani-Tehrani and Khashayar Khorasani. *Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach*. Springer Berlin / Heidelberg, 2009.

[101] Jianping Song, Aloysius K. Mok, Deji Chen, and Mark Nixon. Challenges of Wireless Control in Process Industry. *Workshop on Research Directions for Security and Networking in Critical Real-Time and Embedded Systems*, April 2006.

[102] M. Staroswiecki, S. Attouche, and M.L. Assas. A graphic approach for reconfigurability analysis. *10th Int. Workshop on Principle of Diagnosis (DX' 99), Loch Awe, Scotland (UK)*, page 250–256, 1999.

[103] Thomas Steffen. *Control Reconfiguration of Dynamical Systems*. Springer Berlin / Heidelberg, 2005.

[104] P. Šůcha, M. Kutil, M. Sojka, and Z. Hanzálek. TORSCHE Scheduling Toolbox for Matlab. In *IEEE Computer Aided Control Systems Design Symposium (CACSD'06)*, pages 1181–1186. Centre for Applied Cybernetics, Department of Control Engineering Czech Technical University in Prague, October 2006.

[105] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock Synchronization for Wireless Sensor Networks: A Survey, March 2005.

[106] Sekhar Chandra Tatikonda. *Control under communication constraints*. PhD thesis, Massachusetts Institute of Technology (MIT), September 2000.

[107] James K. Taylor. Worldwide industrial markets for wireline and wireless ethernet infrastructure components and network software. Whitepaper, Venture Development Corporation, May 2004.

[108] Fei-Yue Wang and Derong Liu, editors. *Networked Control Systems - Theory and Applications*. Springer London, 2008.

[109] Y. Q. Wang, S. X. Ding, P. Zhang, W. Li, H. Ye, and G. Z. Wang. Fault Detection of Networked Control Systems with Packet Dropout. *Proceedings of the 17th IFAC World Congress*, Juli 2008.

[110] Wibree. http://www.wibree.com.

[111] Andreas Willig. *Investigations on MAC and Link Layer for a wireless PROFIBUS over IEEE 802.11*. PhD thesis, Technische Universitaet Berlin, 2002.

[112] Andreas Willig. Redundancy Concepts to Increase Transmission Reliability in Wireless Industrial LANs. *IEEE Transactions on Industrial Informatics*, 1(3):173–182, August 2005.

[113] Andreas Willig. Recent and Emerging Topics in Wireless Industrial Communications: A Selection. *IEEE Transactions on Industrial Informatics*, 4(2):102–124, May 2008.

[114] Andreas Willig, Kirsten Matheus, and Adam Wolisz. Wireless Technology in Industrial Networks. In *Proceedings of the IEEE*, volume Volume 93, pages 1130–1151, June 2005.

[115] The Wireless Industrial Networking Alliance (WINA). http://www.wina.org.

[116] Jörg F. Wollert, Thomas Kruse, and Andreas Vedral. Die Kraft der drei Antennen. *Elektronik Wireless - Fachmagazin für Entwicklungen funkbasierter Systeme*, 2:36–42, October 2007.

[117] Heinz Wörn and Uwe Brinkschulte. *Echtzeitsysteme - Grundlagen, Funktionsweisen, Anwendungen*. Springer Berlin Heidelberg, 2005.

[118] Feng Xia and Youxian Sun. *Control and Scheduling Codesign - Flexible Resource Management in Real-Time Control Systems*. Springer, 2008.

[119] S.H. Yang, X. Chen, L.S. Tan, and L. Yang. Time delay and data loss compensation for internet-based process control system. *Transactions of the Institute of Measurement and Control*, 27(2):103–118, 2005.

[120] Z-Wave. http://www.zen-sys.com.

[121] Dieter Zöbel. *Echtzeitsysteme - Grundlagen der Planung*. Springer Berlin Heidelberg, 2008.

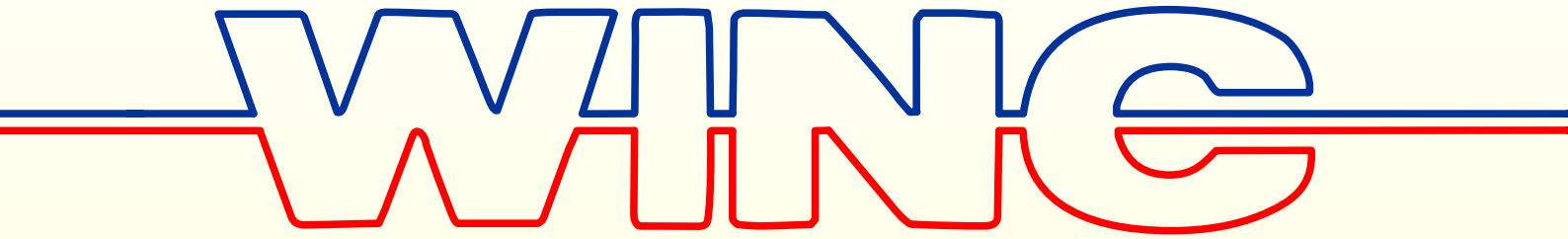
[122] P. Zhang, S. X. Ding, P. M. Frank, and M. Sader. Fault detection of networked control systems with missing measurements. *Proceedings of 5th Asian Control Conference*, 2004.

[123] Wei Zhang. *Stability Analysis of Networked Control Systems*. PhD thesis, Case Western Reserve University, August 2001.

[124] Youmin Zhang and Jin Jiang. Bibiographical review on reconfigurable fault-tolerant control systems. *Proceedings of the 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS '03)*, page 265–276, June 2003.

[125] IEEE Wireless Standards Zone. http://standards.ieee.org/getieee802.

Index


winc