2 Fundamentals

This chapter provides a general state of the art analysis that is relevant for this work. Section 2.1 gives an overview about the production systems and provides deep insights into NCSs build upon TTEB systems. Furthermore, in section 2.2 requirements in terms of real-time and reliability that have to be fulfilled by INS are discussed and summarised. Afterwards, in section 2.3, existing concepts to achieve deterministic and reliable real-time communication are described. Section 2.4 elaborates on the engineering process and all steps that have to be performed in order to bring an INS into work. Finally in section 2.6 the problem statement of this dissertation is formulated and in section 2.7 the solution approach is briefly described.

2.1 Production Systems

Current production systems are complex structures composed of entities coming from different technological areas such as: mechanical engineering, electrical engineering and information technology. They are continuously evolving to fit current market circumstances to allow production companies to remain competitive. The short evolution of the industry, including four major evolutionary steps is illustrated in figure 2.1. The first industrial revolution (Industry 1.0) was started in 18th century with the invention of mechanical machines and steam engine, which was was supplying machines with the power. This significantly improved efficiency of production processes, allowed to produce goods that were not possible before and had a positive influence on working conditions of human beings. Next, the second industrial revolution (Industry 2.0) was introduction of the paradigm of mass production. It dramatically changed the production systems by massively increasing production output at the same time lowering the price. Further evolution of mechanical and electrical engineering allowed building much more sophisticated production lines, at the same time increasing wideness of the production assortment and shortening the production cycle time. The next significant step in improving production systems and its third revolution (Industry 3.0) was appliance of the information technology (IT). It allowed further evolution of production systems with the possibility for the system components to communicate among each other and with the operators (e.g. through Human Machine Interfaces (HMI)). The current trend in production systems is increasing connectivity of the "things" (robots, machines, etc. and also human beings) participating in the production processes. The main idea is to increase accessibility and reduce complexity of the network configuration and integration of new things into the existing environment. This is currently an important research topic, called Internet of Things (IoT), which has been addressed

© Springer-Verlag GmbH Germany 2017
L. Wisniewski, *New methods to engineer and seamlessly reconfigure time triggered Ethernet based systems during runtime based on the PROFINET IRT example*, Technologien für die intelligente Automation 6,
DOI 10.1007/978-3-662-54650-5_2

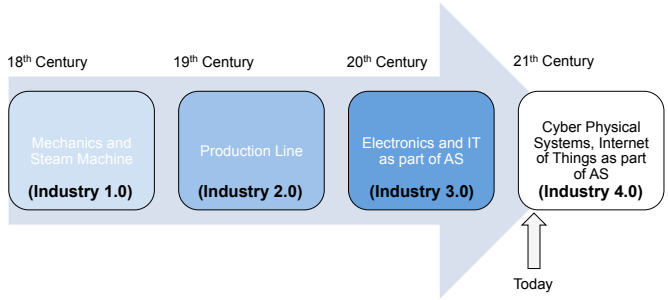


Figure 2.1: Industrial evolution (4 significant revolutions)

in several EU projects, such as IoT-a [36], IoT@Work [37] and many others that can be found in the European Research Cluster of the Internet of Things [38]. The other important aspect is to make "things" smart or "intelligent", e.g. through self diagnosis, self optimisation, etc. This can be achieved by extending the "physical world" (devices, machines, tools, etc.) with algorithms and methods that run in a "cyber world" and interact with the "physical world", thus forming a new concept of Cyber Physical System (CPS) [39][40]. All these concepts are part of the so called fourth industrial revolution (Industry 4.0) [41][42][43].

2.1.1 Automation Systems

In today's factories, there are machines, robots, controllers, communication devices, cameras, etc. that, if properly configured (engineered), can operate 24 hours per day, sometimes without or only minimal intervention of a human being. Typically, when rearrangements are necessary, human (engineer/expert) interaction is needed. Nowadays, such system spans the whole factory, including different departments that are locally or even globally distributed. This has been mainly achieved through the use of appropriate communication technologies. An **automation system (AS)** can be illustrated using well known automation pyramid (AP) illustrated in figure 2.2. According to Sauter [44], through implementation of internet technologies into the AS, the pyramid shrunk from six defined at the time of Computer-Integrated Manufacturing (CIM) down to only three levels. Starting on top from the Company level including Enterprise Resource Planning (ERP) systems [45], going down through the cell level that includes Supervisory Control and Acquisition (SCADA) system [46] and Manufacturing Execution System (MES) [47] ending on the field level, where sensors and actuators coupled with the controller directly interacts with the technical process (TP) [48]. The **TP**, as defined in the VDI guideline 2206 – Design methodology for mechatronic systems [49], is a physical system, composed of mechanical, electric, hydraulic, pneumatic or any combination of these components. As shown in figure 2.2 there is a need for an information exchange in the automation system among different layers defined as horizontal communication as well as across the levels defined as vertical communication. However, the important fact is that these data flows have different requirements in terms of time-

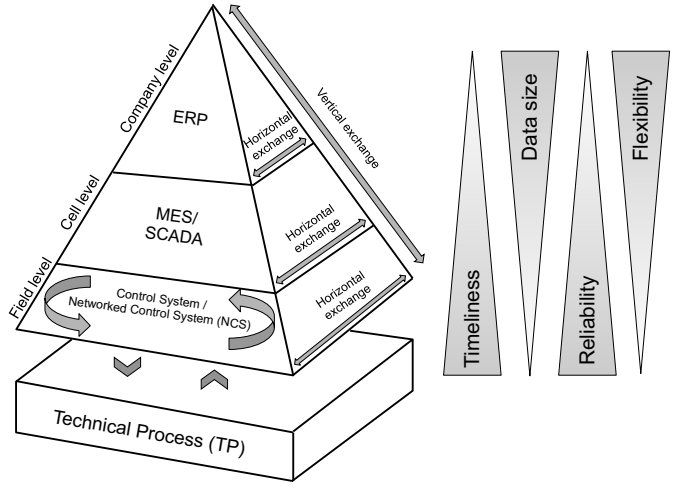


Figure 2.2: Automation pyramid

liness, amount of data sent in one data packet or communication reliability. Level of available **flexibility** is also different, which in this work is measured as the effort related to integration of a new network component with the existing network infrastructure. In case of vertical exchange, the most crucial aspect is the interoperability between different systems located at different levels.

The focus in this dissertation is the lowest level of the automation pyramid that directly interacts with the TP. It has the most demanding requirements in terms of short latency and jitter (defined as timeliness and described in section 2.2.1) and communication reliability (described in section 2.2.2). However the level of flexibility at this level is rather limited.

2.1.2 Networked Control System

At the bottom of the automation pyramid, we can find the following units: sensors, actuators, Input/Output (IO) devices as well as controllers that are directly interacting with the TP [48]. A brief description about each of these components is given below:

- **Sensors** are units responsible for monitoring of variables of the TP in time, converting the physical sizes, such as: speed, torque, temperature, pressure, etc. into a electrical signal, thus creating a “digital picture” of the TP.

- **Controllers** are sophisticated computer systems, designed to operate in a harsh industrial environment to control the TPs. The control is performed by collecting data about the current status of the TP from multiple digital and analogue sensors, processing it with the internal control logic software and producing output signals for actuators. This process is typically performed periodically, where the **cycle time** used to collect sensor data and update actuators with the new data, de-

pends on the dynamics of the TP. For instance printing machines, having a high process dynamics (motion control application), require short cycle times, i.e. below $1ms$ with the jitter less than $1\mu s$ [50], while processes such as beer brewing (process control application), require cycle times in the range of $10ms$ [50]. In the industrial environment, the role of controller is typically performed by a specially designed Programmable Logic Controllers (PLCs) and therefore from now on, this terms will be used to described such control units.

- **Actuators** are responsible to interact with the TP by changing digital signal into energy or force using hydraulics, pneumatics, electric current, magnetics or mechanics in order to control it.

- **IO devices (IOD)** also called remote IO, are currently widely used to communicate with multiple sensors and actuators, which are distributed in the field. In comparison to older centralised systems, in which sensors and actuators are directly connected with a PLC, they allow to save a lot of wiring efforts, simplify the system, thus save time and money. Communication between PLC and sensors and actuators is performed using one physical interface (e.g. Ethernet) and one communication protocol (e.g. EtherCAT).

These components, typically exchange small amount of data by using defined cycles times [51] forming a **Control System (CS)**. Two major classes of control systems can be distinguished. These are open loop control and closed loop control systems, as shown in figure 2.3.

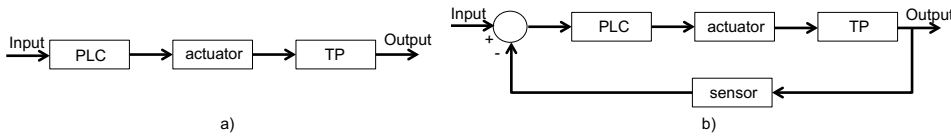


Figure 2.3: Control System a) Open Loop System, b) Closed Loop System with Feedback

Open loop system is controlling TP by setting selected parameters, without considering its current status, see figure 2.3a. Therefore, it can be used to control simple non-critical (rather not industrial) processes, such as drying clothes in washing machine. Here, after setting a particular program (defined by drying time), the drying process starts. Since drying time has been defined based on a simple model, such as 1 kg of clothes should be dried in 0.5h, there is no guarantee that all clothes will be dried after this time elapses. Another important characteristic of this control systems is that whenever there are some disturbances of the process, such as air leakage, the quality of this control loop may be significantly decreased. **Closed Loop Systems with feedback** (figure 2.3b) consider the current status of the TP (e.g. using sensors) to much more precisely control its behaviour. Since the status of the TP system is constantly monitored, closed loop control systems can be used to control diverse industrial processes, such as chemical reactions, motion control applications, etc. As opposed to the open loop, closed loop control system can handle external disturbances that may influence

the TP.
Today, components such as PLCs, IODs, sensors and actuators are highly distributed among the production systems, thus there is a need to span them together with the communication network. Such control systems are called **Networked Control System (NCS)**, where network becomes a part of the control loop, see figure 2.4. It is important

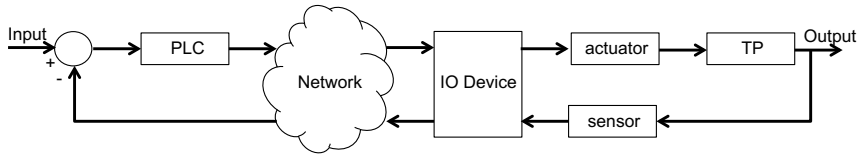


Figure 2.4: Networked Control System

to mention that the network used in the control loop has to fulfil certain requirements in terms of timelines and reliability. Several proprietary and standardised network technologies and protocols have been developed [52] that can be used to build NCSs. They were explicitly designed to fulfil requirements of a large variety of industrial applications. A detailed description about the requirements is given in the following section. Discussion about the way how protocols were designed in order to fulfil these requirements is given in section 2.3.

2.2 NCS Communication Requirements

There is a large variety of different processes (applications) that have to be controlled by a NCS. A simple example could be a boiling process (e.g. in a brewery), where temperature of a liquid must be kept in a predefined temperature range. Such system can be composed of the following components (see figure 2.5): a boiler that holds the liquid (it is the TP), a heater (an actuator) and a temperature sensor (thermometer) responsible for measurements. Sensor and actuator are connected together via IODs responsible for exchanging information with the PLC that controls the boiling process. The HMI

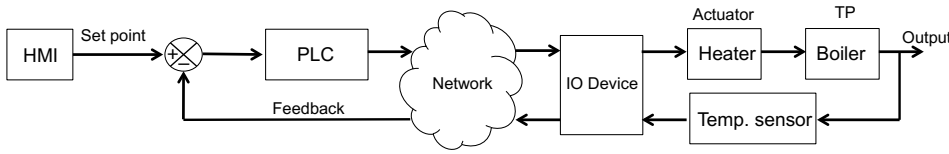


Figure 2.5: Boiling process controlled by a control loop with feedback

enables to set a reference temperature in the boiler. It is a typical example of a control loop that uses feedback to control the process (in this case heater turns on/off). Looking at the process from a higher level, task of the control system is to keep temperature of the boiler at some predefined temperature range. From the control system point of view, the control process can be divided into several smaller tasks that have to

be performed cyclically. In order to be able to control the boiler in real-time, all tasks performed by the NCS are time constrained. It means that tasks, such as data collection from sensors, data processing by the PLC and data distribution to the actuators have to be performed in a defined time window, see figure 2.6. This time window defines its

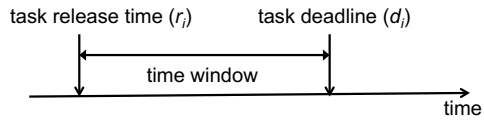


Figure 2.6: Task with the real–time requirements

release time (r_i) for each task, e.g. in case of data collection, defines the earliest time when the sensor data is ready to be sent. Each task have also its deadline (d_i), which is defined as the latest time, where a task has to be accomplished. In case of data collection, it would describe the latest time a sensor data has to be delivered to the PLC. Only in such a way, it is possible to achieve appropriate reaction time of NCS over the TP. The following subchapters will discuss the most important characteristic parameters that has to be considered in designing of NCS. The focus is put on the INS that has to allow to fulfil certain real-time as well as communication reliability requirements as a part of the NCS. The next chapter discusses the timing requirements of the NCS in more detail.

2.2.1 Timeliness in NCS

In order to be able control the industrial process, it is important that the communication is performed in real-time. Figure 2.7 illustrates a sample of communication exchange between an IOD (sender) and a PLC (receiver), and transmission latency T_d (can be t_{di} or t_{do}) caused by the network system. All messages are delayed by the network

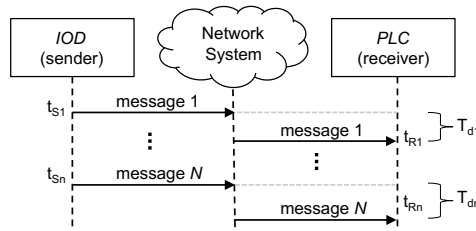


Figure 2.7: Transmission time parameters

system. The transmission latency T_d can be calculated by taking two time stamps, one on the sender side at the time of sending the message (t_{S_i}) and one at the reception time of the message (t_{R_i}). By the subtraction of t_{R_i} and t_{S_i} we obtain the T_d. It is not possible to exclude the network latency completely from the data delivery, however it is important to keep the latency bounded. The parameter that describes the time difference between the worst case ($T_{d_{max}}$) and the best case transmission latency is defined

as latency jitter. The latency jitter can be calculated using eq. 2.1-2.4.

$$T_{d_i} = T_{R_i} - T_{S_i} : i \in \mathbb{N} \tag{2.1}$$

$$T_{d_{max}} = max(T_{di}, ..., T_{d_N}) : 1 \leq i \leq N, N \in \mathbb{N} \tag{2.2}$$

$$T_{d_{min}} = min(T_{di}, ..., T_{d_N}) : 1 \leq i \leq N, N \in \mathbb{N} \tag{2.3}$$

$$Jitter = T_{d_{max}} - T_{d_{min}} \tag{2.4}$$

Latency and jitter are the key performance parameters that defines the real-time capabilities of the communication system [53]. In order to achieve high real-time performance, it is required to keep them as low as possible. Table 2.1 summarises different application areas and their real-time communication requirements [54] [50].

Table 2.1: Summary of real time requirements

Applications	Latency	Jitter	RT performance	RT class
Building automation	~ 100ms	-	Low	1
Process control	≤10ms	≤1ms	Medium	2
Motion control	≤1ms	≤1μs	High	3

Data Collection and Data Distribution

The status of the TP may change, and the dynamics depends on type of the process it is performing. There are processes, where the status of a defined variable changes rarely, e.g. some defined threshold value of physical unit has been exceeded and some action have to be done to return to the allowed value. On the other hand, there are processes that are in constant move and additionally have to run synchronously, e.g. motion control applications. An example can be control of a printing process, where many drives that are involved in a printing operation have to work synchronously. According to authors in [55], among the control systems, two basic communication paradigms to monitor and control the object can be distinguished, these are event-triggered and time-triggered. Figure 2.8a presents the **time-triggered** communication paradigm, where an IOD exchanges data with the PLC through a network system. Lets assume that a control loop maintains temperature of an object that has been defined to be 18 degrees. To realise it, time-triggered paradigm define a cycle time that will be used to do both, collect data about the TP and distribute control data to the actuators interacting with the TP. Typically, both actions (data collection and delivery) are started at the same time, therefore, the earliest time the PLC can react on the TP status observed in Cycle N is the Cycle $N+1$ (see figure 2.8a). The biggest advantage of this approach is that the communication characteristics, such as amount of data to be sent, cycle time, etc. are known in advance and remain constant. Thus the necessary network resources can be precisely defined and reserved in advance e.g. by preparing TDMA schedule. Important is that the actuality of the digital image about the TP depends on the cycle that was defined to poll the information regarding its status. Another advantage is that since the information is exchanged cyclically, the PLC can detect problems that can be e.g. caused

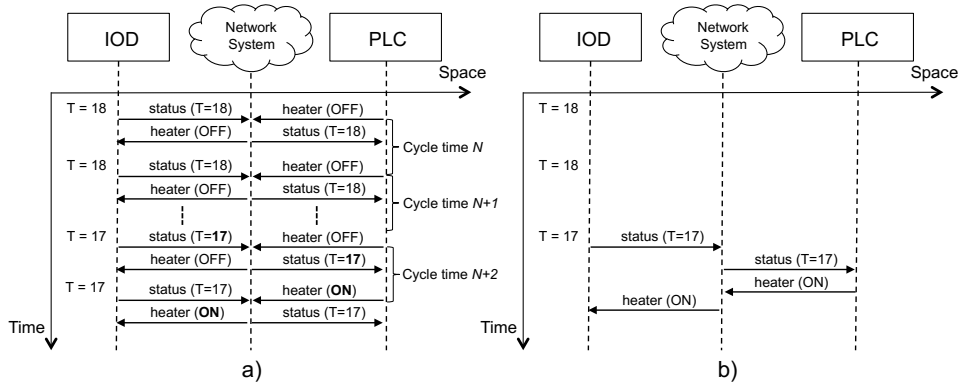


Figure 2.8: Communication system paradigm a) time triggered system b) event triggered system

by broken cable when there will be lack of reply from the IOD. This information can be used to perform emergency stop. The biggest disadvantage of this approach is the resource consumption. It is due to the fact that a defined parameter is always exchanged according to the defined in advance cycle time, also if it remains unchanged (see figure 2.8a).

Another paradigm shown in figure 2.8b is **event-triggered** communication. Here, the information containing status information of the defined parameter of the TP is only sent when it changes or a defined threshold value is exceeded (push method). It is clear that this method is much more resource efficient. However, since it is not clear when the event happens, it is harder to reserve resource in advance for an event to be happened. It is much more difficult to keep the transmission latency and jitter at the low level. In case of problems like broken links, the PLC cannot detect them without additional mechanisms observing the network system status. Due to all these facts, the event-based communication paradigm is not a good candidate to control industrial processes, especially motion control applications where both, timeliness and reliability are the most important requirements. The event-triggered based approaches are good candidates for building automation applications. Table 2.2 summaries both communication paradigms and theirs applicability.

Table 2.2: Summary of communication paradigms

Comm. paradigm	Building autom.	Process control	Motion control
Event-triggered	Yes	No	No
Time-triggered	[1)] Yes	Yes	Yes

[1)] in most of the cases, the event-triggered approach is sufficient for building automation applications. Although, time-triggered communication approach can be used in applications such as energy monitoring [56], but using much bigger cycle times than in the industrial applications [57].

NCS System Reaction Time

An important characteristic, especially in case of hard real-time systems is the *IO Response Time* T_{IOR} [58]. It describes the time difference between a physical event (observed by a sensor) at the TP and the time where an action of an actuator has been taken. That means, it is a system metric telling how fast a NCS can react on a physical event that occurred at the TP (preciseness of control). The IO response time T_{IOR} is illustrated as a time-space diagram in figure 2.9, it includes the following time components [59]:

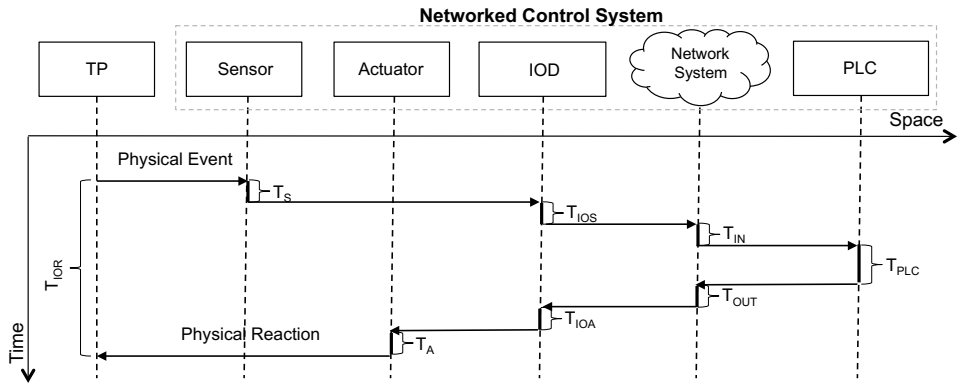


Figure 2.9: IO Response Time T_R (ger. Klemme-Klemme-Reaktionszeit)

Sensor time (T_S) – it is the time that is needed to convert the physical signal into a digital one. This time is mostly dependent on the hardware components of the sensor module.

Actuator time (T_A) – it is the time that is needed to convert a digital signal into a physical one. This time is mostly dependent on the actuator type, i.e. different reaction time will have a hydraulic cylinder in comparison to a electric motor.

IOD sensor time (T_{IOS}) – it is the time needed to process sensor data by the IOD and prepare it for the transmission. Since, sensor data may need to wait some time before it is transmitted, this waiting time is also considered in t_{IOS}. In case of modular IOD, some additional time to transport data internally to the head-end station has to be considered [59].

IOD actuator time (T_{IOA}) – it is the time difference between the time, when the actuator data has been received by the IOD and the time, when the actuator received this data from the IOD. In case of modular IOD, some additional time to transport data internally from the head-end station has to be added [59].

Transmission delay inbound (T_{IN}) – it is the time necessary to transport sensor data from the network interface of the IOD to the PLC. This time is dependent on several

factors like: communication system (half duplex, full duplex), communication protocol type, medium access method, network topology, data length, network load, etc.

Transmission delay outbound (T_{OUT}) – it is the time necessary to transport information from the network interface of the PLC to the IOD. It depends on the same factors like the T_{IN}.

PLC time (T_{PLC}) – it is the time needed by the PLC to process input data, generate output data and prepare it for the delivery to the network. It includes time related to pass data through the communication stack and processing it by the control logic [59].

In order to precisely define all timing parameters defined above, it is of major importance to precisely define the interfaces that were used in the measurements. This topic has been deeply investigated by Hoeme in his dissertation [60]. Since both T_S, T_A and T_{IOS}, T_{IOA} reaction times are mainly dependent on their hardware and in case of T_{IOS}, T_{IOA} also software stacks, it is possible to measure and define the worst case reaction times of these components. Similarly in case of T_{PLC}, it is possible to estimate the worst case execution time of a PLC that is needed to produce an output based on defined input information. The remaining two components, namely T_{IN} and T_{OUT} can be influenced by several factors, therefore it is much more difficult to define the total duration, especially for the INS which are using shared communication network [61]. So the main requirement for the INS is to allow deterministic exchange of data, while keeping certain real-time requirements (latency and jitter) independently on the physical network structure. Since, more than one node shares the same network system, it has to be guaranteed that the real-time flows will have no influence on each other, and cyclic communication between the nodes is possible, for instance, by defining a suitable medium access control. So the main requirement for the network system is to allow deterministic exchange of data, while keeping certain real-time requirements (latency and jitter) independently on the physical network structure.

Deterministic Communication

NCS systems are called real-time systems, since "*the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical, at which these result are produced*" [22]. Although, there is a vast number of applications that may differ in terms of required real-time communication details. There are processes that allow longer latencies or even sporadic missing deadline of some tasks, these are **soft real-time** requirements. In such applications, missing a deadline do not have any catastrophic consequences (e.g. explosion of some over pressured vessel). Although missing a deadline is allowed, the overall system performance will suffer [62]. There are, however, methods to deal with missed deadlines, i.e. by monitoring systems and oversampling techniques [63]. Whenever a control system has to control a continuous process e.g. some chemical process, the reaction on the exceeded temperature would be to stop the process or take some other action. If the action taken by a control system will be too slow (deadline exceeded), the process can have dramatical consequences and may even cause even harm to the life of a human being, in this case these

are so called **hard real-time** requirements. In order to be able to meet the deadlines, all components of a NCS have to provide deterministic behaviour.

Deterministic behaviour according to Kopetz "*is a property of a computation that makes it possible to predict the future result of a computation, given that the initial state and all timed inputs are known*" [22]. For the NCS it means that collecting the sensor data, processing it and setting the output signal should always be done within a defined time window, bounded with a deadline. Deterministic behaviour of the NCS greatly depends on the INS. It includes aspects such as deterministic channel access and real-time network behaviour, which has been deeply investigated by Jasperneite in [54], and aspects such as reliability and availability discussed in [64].

In the industry, there is a lot of machinery that cooperates together performing a common task. It means that some components, e.g. motors, drives, etc., have to run synchronously with each other. There are three terms related to deterministic communication systems, which describe a special type of operation required by such systems, these are synchronous, asynchronous and isochronous operation.

Synchronous operation means that the sampling operation (reading actual status of the TP) of defined sensors or actuation (sending signal to actuators) of defined actuators is performed according to pre-defined time instance [65]. Such operation is only possible if there is a common notion of time (clock synchronisation) in the system.

Asynchronous operation means that some actions are performed in a stochastic way. This can be e.g. generation of an alarm message that has to be transmitted from an IOD to the PLC or the need to transport some configuration or monitoring messages.

Isochronous operation means that periodic operations are performed in a equidistant time. Hanzalek *et al.* in [30] defines isochronous communication as an approach, where *"individual data frames have to be transmitted at time instants, which are equidistant with respect to the consecutive communication cycles"*. This can be observed in communication protocols such as PROFINET IRT, where IRT data is transmitted according to the TDMA schedule in an equidistant way. Such communication approach was used also by Trsek *et al.* in [66] to deterministically access wireless communication medium.

Figure 2.10 illustrates an example, where all defined above operation modes can be observed. The example is based on the PROFINET IRT protocol that runs in isochronous mode. Starting from the top, we can distinguish the following levels:

- **Communication Protocol level**, represents the cyclic communication pattern, where two phases, namely real-time (RT) and best effort (BE) can be distinguished.

- **PLC level**, shows processing time of input data delivered in each communication cycle.

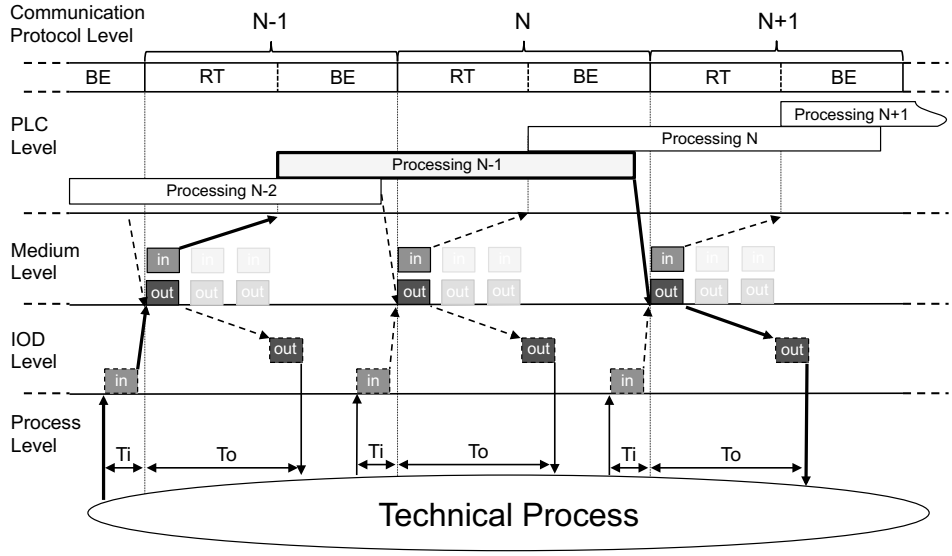


Figure 2.10: PROFINET IRT communication example

- **Medium level**, illustrates (bi-directional) communication at the communication medium, where square in represents inbound and out outbound data.
- **IOD level**, presents activities (i.e. preparation of the input data) that are done at the IODs.
- **Process level**, illustrates how IOD interacts with the technical process (TP).

The **synchronous operation** can be observed at the Process Level, where data collected from sensors by the IOD is done according to the defined time Ti. This time is relative to the beginning of the communication cycle. It is the same case for passing the data from IOD to the actuators. Time point used to perform these operations synchronously is defined as To and is also relative to the beginning of the communication cycle. **Asynchronous operation** can be observed at the Communication Protocol Level in the slot dedicated for the best effort traffic. This is the communication channel for asynchronous communication in the INS. **Isochronous operation** is visible at the Medium Level, where the real-time data (marked with in and out) is sent equidistantly in the consecutive cycles. Processing of input data by the PLC is also equidistant, see PLC and Medium Level in figure 2.10. Here, the sensor data is send in the cycle $N-1$, the processing by the PLC is done in cycles $N-1$ and N, and delivery of actuator data to the Medium Level is done in cycle $N+1$. This is defined as a three cycle model, as described in [59].

A typical group of applications requiring determinism and described above operations are industrial printing machines. These are typical motion control applications, where

several motors have to run synchronously. They require synchronous data collection and data distribution to control synchronously running motors, isochronous data transmission as well as asynchronous channel to transfer monitoring to configuration messages. An example of web-fed offset printing machine is shown in figure 2.11. The print-

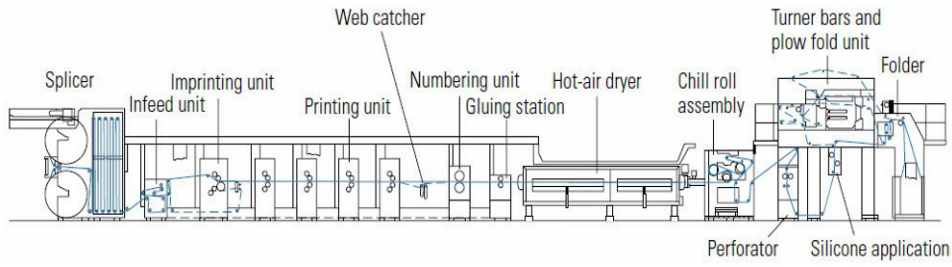


Figure 2.11: Web-fed offset printing machine [67]

ing is done by pulling the paper width through the printing machine constantly, with the paper flow that goes up to 15m/s. Here, a sheet of paper goes with the high speed through a series of rolls that has to operate synchronously in order to perform printing and additional related operations (drying, cooling, etc.), and avoid rupture of the paper. If rupture would happen, there is a need to stop the process, which leads to production loss and high costs.

2.2.2 Communication Reliability in NCS

Another important characteristics and requirements for building NCS are communication reliability and network availability [22]. Only by using reliable network technologies and taking care of the network failures, it is possible to build real-time capable and deterministic NCSs.

Communication reliability can be defined as a probability with which a message will be delivered from its source to the destination [68]. Kopetz in [22] defines reliability as a *"probability that a system will provide the specified service until time t"*. This means that the transmission time of the message should always be less than t to achieve 100% of reliability. He also suggests how real-time communication reliability can be improved, i.e. by *"robust channel encoding, the use of error-correcting codes for forward error correction, or the deployment of diffusion based algorithms, where replicated copies of a message are sent on diverse channels (e.g., frequency hopping in wireless systems), possibly at different times, are the techniques of choice for improving the communication reliability"*. Another way to improve reliability is by utilisation of data redundancy, where the critical data is duplicated and sent over different paths to its destination [18]. However, this method is tightly coupled with network redundancy, since it depends on the redundant physical connections that are used to transport redundant data.

Network availability is defined in [69] as *"the ratio of (a) the total time a functional*

unit is capable of being used during a given interval to (b) the length of the interval. Note 1: An example of availability is 100/168 if the unit is capable of being used for 100 hours in a week. Note 2: Typical availability objectives are specified in decimal fractions, such as 0.9998".

In order to improve the availability, network needs to take care of failure instances, which may depreciate the network availability. Among other ways, the redundancy is the most common way to improve the availability within the network. In this context, redundancy means that some or all parts of a network are duplicated, what assures that in case of a failure of a network entity (e.g. link, device, etc.), the redundant entity will be able to perform the functionality of the failed one, and the communication will be restored within a short period of time, where zero time is considered optimum. Industrial requirements are based on the characteristics of industrial TP that are controlled by NCSs. Therefore, it is vital to know when a TP can continue its operation after a network entity failure. Under normal operations, a network entity failure does not have catastrophic consequences. It is due to the fact that during the layout of a plant, an intrinsic protection against such failures is planned. However, with the use of intrinsic protection system, the industrial production has to bear with the production loss. In order to prevent a loss in production, the system should resume its operations before the plant takes actions. The time a plant can allow for recovery, before taking any action is known as the **grace time** (t_G). Kirmann *et al.* in [70] defines it as the *"time a plant can sustain without shutting down or suffering damages"*. The **recovery time** (T_R) can be defined as a time difference between the time where a fault event happened and the time when the network regains its required communication function in the presence of fault, see IEC62439 [71]. So it is important to keep the recovery time always smaller than the grace time, $T_R < T_G$. In case of industrial applications, control applications running at the PLC typically monitor the real-time data streams using special timer, called **watchdog**. They can be set from $0s$ to any other value that is appropriate for a particular application. A default value e.g. for protocols such as PROFINET is equal to 3 multiplied by the cycle time. The working principle of a watchdog is very simple. When a particular a time sensitive message has not been received in defined time (by the watchdog time), the application shuts down. This fact should be considered while designing recovery methods of the network system. Therefore, the recovery time T_R for the industrial systems should be in the range between $0s$ and watchdog time.
Table 2.3 summarise characteristics, relevant to the reliability of the network system considering different application fields. The term reliability level was introduced to stress the time required to recover the network error.

Table 2.3: Summary of real time requirements

Applications	Cycle time	Recovery time	Reliability level
Building automation	$\sim 100ms$	[1)] Watchdog	[1)] Low
Process control	$\sim 10ms$	Watchdog	Medium
Motion control	$\leq 1ms$	[2)] Watchdog	High

[1)] this is only in case of life-safety applications, such as fire alarms. Otherwise building

automation have rather relaxed requirements in terms of reliability [57].
[2)] Some time critical applications might not have a Watchdog or it can be set to 0. Therefore for some motion control application, the recovery time has to be equal to $0s$.
To achieve availability and reliability of a system as defined in table 2.3, proper planning of the redundant infrastructure is necessary as well as efficient algorithms responsible to find the best paths for the redundant communication. Especially in cases, when the network system is only partially duplicated, thus allowing to find only possibly disjoint paths.

2.2.3 Industrial Communication Requirements - Summary

The question: which real-time INS should be used to build a NCS to control a particular process, depends on the requirements of a particular application. The most demanding requirements in terms of real-time and reliability performance is required by the motion control applications. Table 2.4 summarises all NCS requirements described in chapter 2.2.

Table 2.4: Summary of NCS communication requirements

Features	Building autom.	Process control	Motion control
Latency	$100ms$	$\leq 10ms$	$\leq 1ms$
Jitter	-	$\leq 1ms$	$\leq 1\mu s$
RT performance	Low	Medium	High
RT class	1	2	3
Asynchronity	Yes	Yes	Yes
Synchronity	No	No	Yes
Isochronity	No	No	Yes
Event-triggered	Yes	No	No
Time-triggered	[1)]Yes	Yes	Yes
Recovery time	[2)]Watchdog·Cycle time	Watchdog·Cycle time	[2)]Watchdog·Cycle time
Reliability level	[3)]Low	Medium	High

[1)], see Table 2.2 reference [1)]
[2)], see Table 2.3 reference [1)]
[3)], see Table 2.3 reference [2)]
How the currently available industrial communication systems deal with these requirements is briefly described in the next section. Detailed description about functionality details of selected real-time communication systems is provided in chapter 3.

2.3 Industrial Network Systems

The NCS should be able to exchange process data through the INS in real-time, while keeping certain level of reliability. Nowadays, INSs implement highly sophisticated technologies, allowing exchange of high amount of information in a tiny amount of time. In this section, the evolution process of the industrial communication system will be given in order to show most important breakthroughs and contributions in this field.

In the early days of automation, a vast number proprietary communication technology solutions were developed to allow exchange of information in real-time. This situation demanded variety of experts with the knowledge about each highly complex solution. The evolution of these systems was described in detail by Sauter in [72]. However, in this work the most important milestones will be pointed out.

The main difference of the technologies available at that time was the proprietary way to satisfy diverse industrial requirements. The most important aspect is the Medium Access Control (MAC) mechanism that is mainly involved in achieving real-time performance. Three main approaches to achieve deterministic medium access had been defined at that time and they are valid till today [72], these are:

- Polling – it is typical master-slave approach, where the master device is giving slaves opportunity to send them a piece of data by polling it, using a polling message. In this way, access to the medium is controlled by the master device, thus there is no problem of another users competing to access the medium causing effects such as collisions or unexpected queuing delays. This approach allows to achieve real-time communication and determinism. However, it requires a proper scheduling approach that defines order in which master device is polling slave devices. This category is represented by the protocols, such as: Ethernet Powerlink, PROFIBUS (DP and PA), MODBUS.

- Token passing – this method uses a special message as a virtual token that is passed typically in a ring topology (or a virtual ring as in case of EtherCAT) to the neighbouring devices. The device holding a token is the only device having access to the medium. After data had been transferred, the token is passed to the next device. This group is represented by the protocols like PROFIBUS DP, INTERBUS and also EtherCAT, in which the summation frame works as a token passed in a "virtual" ring.

- Time division multiple access (TDMA) – in this approach, communication is carried out using well defined time slots, where a particular device gets the opportunity to transfer defined amount of data. This method requires precise clock synchronisation to avoid communication conflicts caused by differences in local time of devices. This communication approach is used today by the protocols like: SERCOS III, TTEthernet or PROFINET IRT.

The natural evolutional step to achieve broader acceptance of a communication technology was to create a standardised solution. Two concurrent solutions have been developed in the middle of 80's namely PROcess FIeld BUS (PROFIBUS) [73] and Factory Instrumentation Protocol (FIP) [74], which were trying to get the status of the universal standard. Finally, none of these protocols became a universal standard, and many other field bus protocols were developed in the next years. Another big breakthrough in the industrial communication world has been made by the idea of putting Ethernet standard that was already well know and widely used in the Home and Office environment [75]. Since Ethernet was developed for totally different requirements, the main question was how to use this technology for demanding in terms of timeliness and re-

liability industrial applications. A brief introduction of the approaches of fitting and adapting Ethernet standard to the industrial requirements is given in the next section.

2.3.1 Industrial Ethernet based Protocols - Real-Time

As described in chapter 2.2, there is a wide range of applications having different requirements in terms of timeliness and reliability. To satisfy them, many communication protocols have been developed in the past years. Starting from the fieldbus system technologies ending at real-time Ethernet based approaches. According to [76], there are currently more than 20 different Ethernet based real-time INSs. Since there is a lot of overlapping, and there are many protocols that satisfy the same or similar requirements, real-time base Ethernet systems can be divided into three main protocol groups according to [50][3], which satisfy requirements of different applications, see also figure 2.12:

- Class 1 – the real-time communication is achieved above the transport layer. The achievable cycle time is around $100ms$ [3]. Protocols which belongs to this class are: ModBus/IDA, Ethernet/IP, FF HSE.

- Class 2 – the real-time communication is realised above the MAC layer, by approaches such as prioritisation. The achievable cycle time using fast Ethernet as a basis lays in the range of $10ms$ [3]. An example protocol in this group is PROFINET RT.

- Class 3 – is the most stringent class, with the real-time communication capabilities built in the MAC layer and strict communication scheduling. The achievable cycle time lies below $1ms$ [3]. Examples of the protocols from this group are: PROFINET IRT, EtherCAT, TTEthernet, SERCOS III, Ethernet and Powerlink.

In this work, the focus is on the most demanding applications, therefore protocols that are part of the Class 3 will be briefly described in the following paragraphs. Detailed description can be found in chapter 3.

Class 3 – Polling

High real-time performance and determinism is achieved here by scheduling of the communication nodes and then polling them one after the other by sending polling messages. This approach is used by the Ethernet Powerlink protocol [10]. It is an open source communication protocol that achieves cycle time down to 100 μs and jitter below $1\mu s$. Here the managing node (MN) manages the communication cycle, which is divided into 3 phases. The first phase is used for synchronising controlled nodes (CN) in the topology. In the second isochronous phase, MN polls each CN to allow them to send time sensitive data to all other nodes (broadcast). The last (asynchronous) phase is dedicated for best effort traffic. The protocol is fully compliant with the Ethernet standard.

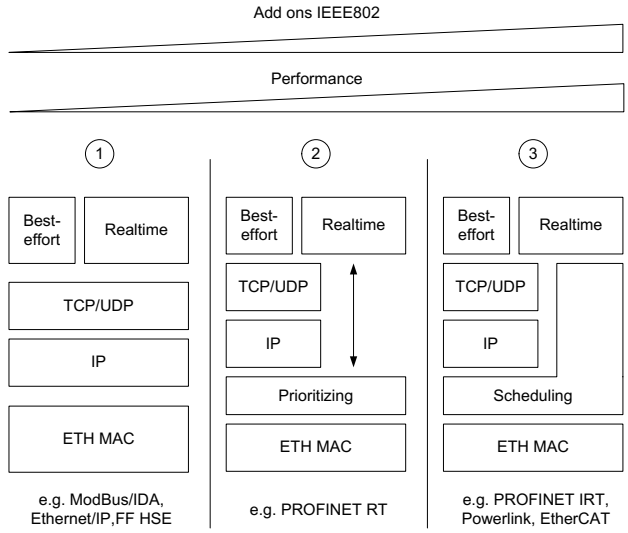


Figure 2.12: Ethernet based protocols taxonomy [3]

Class 3 – Token Passing

Totally different approach is followed by EtherCAT [6][7][8] protocol. The communication is organised using summation frame that is provided by the controller and traverses through all devices in topology. While receiving it, slave devices can include and/or extract the process in/from the summation frame. EtherCAT achieves high performance, down to 12.5μs cycle times [17]. Before bringing a setup to run, an engineering process is mandatory. Similar approach is also followed by the SERCOS III protocol. Here also summation frame (telegram) issued by the master device is used as a token, defining transmission opportunity of each slave device. Cycle time that can be achieved by SERCOS III is 31.25μs [77]. In case of both EtherCAT and SERCOS III, no scheduling mechanism is necessary to plan the real-time communication.

Class 3 – TDMA Based

An important example of protocol, which became a standard in German automotive industry [78], is PROFINET [4], [5]. PROFINET IRT, which is the high performance profile of this protocol, achieves strict and tight deadlines by planning the whole communication in advance. Additionally, the real-time communication has to be prioritised, in order to assure no negative influences from the non real-time traffic. This has been achieved through bypassing the TCP/IP stack and by introduction of a special hardware, which is able to handle real-time traffic according to the prepared in advance communication schedule. The approach to achieve high real-time performance is to divide the communication into phases, in which a particular kind of traffic can be sent. In this way, real-time frames are isolated from the other kinds of traffic, allowing to

avoid queuing delays. This however demands efficient planning algorithm that is able to calculate the communication schedule, and used afterwards by all PROFINET IRT devices in the network. Since the communication schedule is calculated for some particular network topology, whenever there is some topological change, the scheduling needs to be performed again. Another TDMA based protocol is Time Triggered Ethernet (TTEthernet) [9]. Here, for the most demanding traffic flows (e.g. safety critical), a communication schedule is prepared in advance and used by the involved devices in a time triggered way. It is also possible to reserve bandwidth for some time critical data. Additionally, the best effort traffic is also allowed and can be sent according to the IEEE 802.3 rules. Typically TTEthernet is used for safety critical applications in cars, aeroplanes or even space shuttles. Before starting the TTEthernet system, engineering process is required. Currently, there is ongoing work on a new IEEE standard called 802.1 TSN (Time Sensitive Networks), that may have significant influence on the industrial communication world. It offers, already, precise clock synchronisation due to 802.1 AS [12] protocol, possibility to reserve bandwidth thanks to 802.1Qat [15] protocol and real time communication thank to 802.1Qav [16]. The new amendment called times aware shaper defined in 802.1Qbv supposed to support scheduled deterministic communication. After finishing of works on these amendments, it may happen that this standard will replace some classes illustrated in figure 2.12. More about TSN and other Class 3 protocols can be found in chapter 3. In case of all TDMA protocols, efficient scheduling algorithm is needed to organise real-time communication.

2.3.2 Industrial Ethernet based Protocols - Reliability

There are many aspects that can influence communication reliability, and also methods that can deal with the communication problems. These aspects can be correlated with the particular layer of the ISO/OSI model, see figure 2.13a and 2.13b. In case of industrial Ethernet based communication protocols, only three layers are involved in the real-time data transfer, namely physical (1), data link (2) and the application layer (7). The IEEE 802 standard defines the two lowest layers of the ISO/OSI model. The industrial protocol defines the application layer and also modifies parts of the data link layer that allow to achieve real-time capability. Therefore, in the following paragraphs, we will focus only on these three layers. We will discuss what kind of problems may occur and also how to deal with these problems.

Physical Layer

At this layer, two major communication technologies can be distinguished i.e. wired and wireless. Since, the focus of this work are Ethernet based systems, we will focus here only on the physical aspects of the wired communication. Wired technology is far more reliable than wireless, since it is easier to control the communication in a “closed” medium such as copper or optic cable than in the air, where in principle everyone can access the communication medium. Wires are also much more resistant to the electromagnetic field, especially, shielded twisted pair cable or the fiber optic cable. One of the

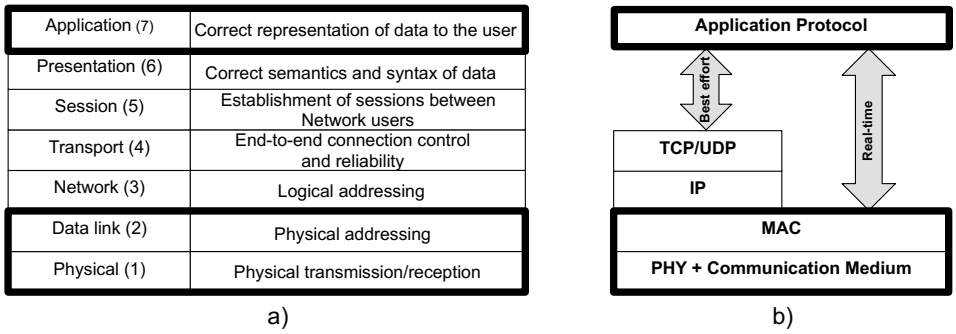


Figure 2.13: a) ISO/OSI model, b) industrial Ethernet based real-time protocol stack

most common methods to improve the communication reliability at this level is physical redundancy of cables or network devices. There are basically two types of network topologies that provide an implicit link redundancy, it is mesh and ring topology, see figure 2.14. In case of full mesh topology 2.14a, all nodes are physically connected with all

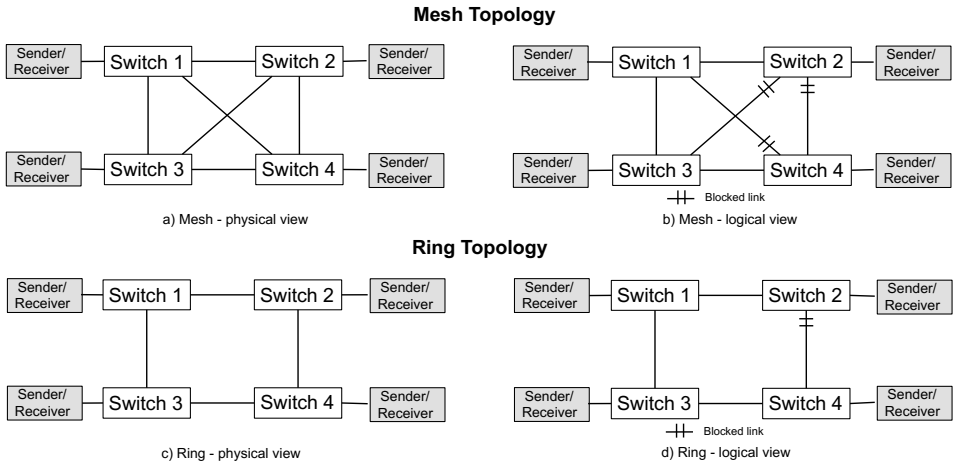


Figure 2.14: Dynamic Redundancy - a) Mesh topology - physical view, b) Mesh topology - logical view, c) Ring topology - physical view, d) Ring topology - logical view

other nodes, providing the needed redundant connection in case of link failure. Since a loop-free network is needed for the proper communication, network redundancy protocol (NRP) is used in order to block redundant connections during normal operation of the network, as shown in 2.14b. In case of link failure, the NRP is responsible to activate redundant link. Advantage of mesh (partial or full) redundancy is robustness and failure tolerance, however it generates high installation costs and maintenance complexity. In contrary, the ring topology shown in 2.14c also allows to deal with the network failures

and it is far simpler. Ring topology is very efficient solution, since it can be build by just adding one connection to the line topology, which is very common network structure in the industrial environments. Whenever a failure occurs, the redundant link is activated by the NRP. A brief description about NRP is given the the paragraph Data Link Layer. For applications with the highest requirements in terms of reliability, it is also possible to duplicate (or partially duplicate) the network system between the communicating entities and use both of them at the same time in the data delivery, as shown in figure 2.15. Such a redundancy method is called seamless redundancy, since whenever there is a problem with one network, the second network is operating in the same time, so the traffic consumer will not miss any data. Such redundancy method can be used in case of safety critical application, but the installation is the biggest drawback of this method. The working principles of seamless redundancy will be described in the paragraph Application Layer.

Data Link Layer

At the data link layer, the IEEE 802 standard offers mechanism of error detection, such as Cyclic Redundancy Checks (CRC) or Checksums [79]. Error detection methods help to process only valid data, and the invalid frames will be simply rejected. E.g. the 32bit version of CRC is able to detect 99,999999977% of errors [80]. There are also approaches to detect and correct errors using error correction codes e.g. Error-Correct Code (ECC) [79]. To recover errors, some additional data have to be added to the "original" data. In case of invalid data, the data sender will not be informed about this fact e.g. to retransmit such frame. End-to-end reliability is realised by TCP protocols using acknowledged transmission at the Transport layer, see figure 2.13. As mentioned in the previous paragraph, to avoid physical failures cable or device redundancy can be used. At the link layer, the NRP takes care of blocking and activating redundant links if necessary. This is typically done by checking status of the ring by sending a frame on one side and receiving it on the other side of the ring. This tasks is typically dedicated to the redundancy master that suppose to activate the blocked port in case a failure happens. Such a redundancy method is called **dynamic redundancy**, meaning the redundant links are activated after a failure happened. The fastest recovery time achievable for this method is $5ms$ [18]. Since hard real-time systems work typically with cycle times that are less than $1ms$, dynamic redundancy methods are not appropriate for this kind of systems.

Application Layer

This layer is mainly involved in the **seamless redundancy**, where duplicated data traverses through the duplicated or partially duplicated network in order to avoid data loss in case of network failure. Example shown in figure 2.15 illustrates the basic functionality principles of the seamless redundancy method. On the left hand side, the traffic source produces two messages with the same content (i.e. one original and one duplicate). These two frames are sent from two different physical ports and traverse using two different (disjoint) paths to the destination. Destination has to reject one of the frames. In the given example, the network system was duplicated so it was possible to

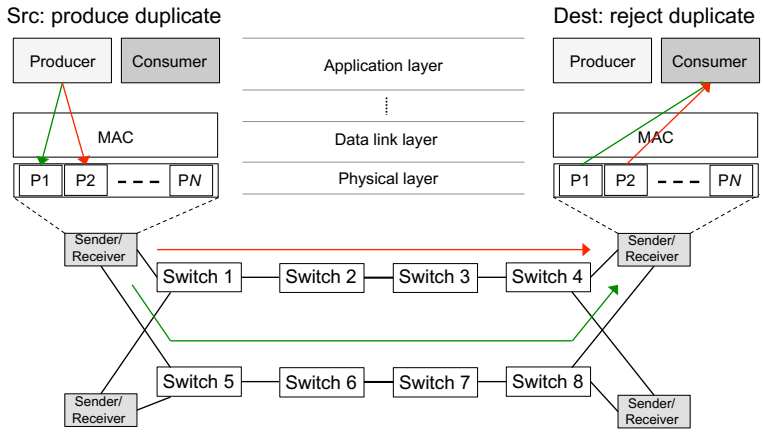


Figure 2.15: Seamless Redundancy

communicate using two disjoint paths. In case of partially duplicated networks that are highly complex, finding such two disjoint becomes a big problem. Therefore, efficient algorithms, which are able to find disjoint or possibly disjoint are needed.

2.4 Engineering of Industrial Network Systems

All NCSs need some engineering effort in order to bring them to the operational state. This effort is related to engineering of physical installation of hardware components (PLCs, IODs, Sensors, Actuators and the industrial network system) as well as software engineering related actions, such as network configuration, writing a control logic, etc. The focus in this dissertation are INSs, particularly Time Triggered Ethernet based (TTEB) systems that as a part of the NCS also require certain engineering steps in order to bring them to work. Figure 2.16 illustrates a typical software engineering process of an INS build upon TTEB system. Note that the whole process is performed offline, typically on a dedicated PC, using dedicated and vendor specific software framework. Generally speaking, software engineering of such systems can be divided into two main steps, INS engineering and scheduling process. The INS engineering process is performed manually by an expert. The scheduling process is performed automatically by the engineering tool itself. It takes parameters defined in the INS engineering process as an input and calculates communication rules, which in case of TDMA based systems are expressed in a form of a communication schedule. An overview about each of these steps is given below.

The INS engineering process includes manual tasks of setting up the network configuration and mapping process, in which the logical variables of the control program are related with the physical inputs and outputs of the IODs. Mapping process can be obviously performed only when a Control Application has been written. Both these steps together with the Control Application block are represented with three white rectangles

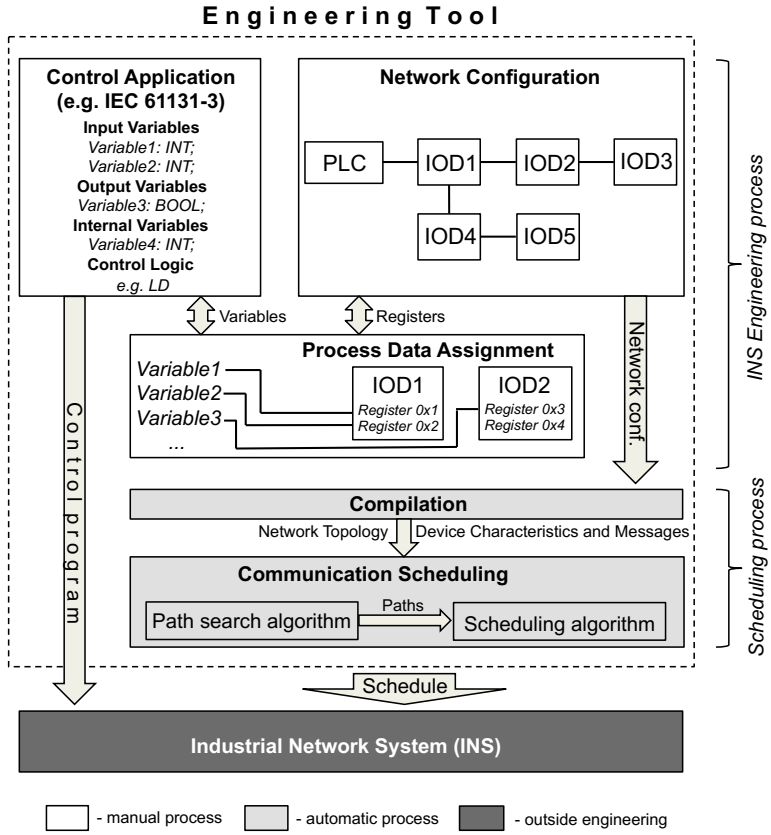


Figure 2.16: Offline engineering of Time Triggered Ethernet based Systems

on top in figure 2.16, and are briefly described in the following lines.
Control application can be written e.g. in one of five available in IEC 61131-3 languages. After uploading it to the PLC, it becomes the heart of the NCS, which is responsible to process the input parameters gathered via INS from distributed in the filed sensors and generate an output for the actuators interacting with the technical process. The first manual operation is the **network configuration**. In this process all parameters needed later on by the scheduling process are defined. According to the engineering guideline of the PROFINET IRT protocol [81], these are:

- **Network topology** – describing how, and which devices are connected with each other, which ports had been used for these connections and what are the cable lengths.

- **Device characteristics** – including communication parameters of network components, such as switch forwarding delays (BridgeDelay), ports delays (TxDelay and RxDelay) and cable delays.

- **Messages** – containing information about the source and destination of a particular message, its length, communication cycle (SendClock) or defining if a message should be send every cycle, every second, third, or so on (in PROFINET IRT it is defined as the ReductioRatio).

Apart from the mentioned above parameters, as described in [26], network addresses have to be assigned to the network devices. It will allow to perform the process of uploading communication schedule, control program and other configuration parameters that are typically done based on IP connectivity.
After completing network configuration, **process data assignment**, i.e. mapping of software variables with a particular input or outputs of an IODs is performed. The input and outputs of the IODs are typically represented in an engineering tool with registers, of which the addresses have to be assigned to a particular input or output variables. This is a manual process, which may take significant amount of time, where its complexity is obviously related to the size of the control program, and the number of sensor and actuators in the setup. This task is the last manual step performed by the engineer. The next process is performed by the engineering tool automatically.
Scheduling process includes two processes, **compilation** and **communication scheduling**. The **compilation** process has been briefly described in [26]. In case of real-time communication system that does not require communication scheduling, it gathers information about the network configuration and produces communication parameters required by the network devices. In case of systems that require calculation of the TDMA schedule, it does the same, gathers information about the network configuration and delivers it to the communication scheduling software.
The **communication scheduling** process takes network topology, device characteristics and messages delivered by the compilation process and in case of TDMA based protocols, uses it to calculate the communication schedule. In case of protocols such as EtherCAT, a virtual ring will be established, where forwarding rules for each network device will be created. The scheduling process also includes the process of finding paths between source and destination of a particular message. In case redundancy is required, the path search mechanism has to consider it. Implemented scheduling algorithm calculates TDMA schedule or defines forwarding rules that will be used by each device in the setup during communication. The communication schedule together with the control logic is uploaded to the PLC afterwards. Thereafter, PLC is distributing the schedule in a vendor specific format to all TTEB network members.

2.5 Re-engineering of Industrial Network Systems

After the engineering process of the INS had been accomplished, the NCS can be started and continuously operate until there is a need to perform maintenance operations. In case of unexpected or expected changes in the setup, such as failures or system upgrades, the NCS may require re-engineering. In this work the focus is on changes related to the INS only. Several possible changes can be distinguished here. They may differ in terms of required engineering operations, and engineering efforts related to them. All possible INS changes are listed in table 2.5. Table 2.6 summarise all necessary

Table 2.5: Summary of possible NCS changes

No.	Change description	Exemplary reason description
I	New IOD added	Production module added
II	IOD replaced	Replacement of a faulty device
III	IOD removed	Production module removed
IV	New switch added	Need to attach more network nodes
V	Switch replaced	Broken switch or need to attach more network nodes
VI	Switch removed	Production module removed
VII	Port(s) swapped	Swapped by a mistake
VIII	Wire replaced	Replacement of faulty wire
IX	PLC replaced	Replacement of faulty PLC

manual engineering operations required by a particular system change, referencing to the changes defined in the table 2.5.

Table 2.6: Summary of required engineering actions

Change	Control Logic	Network Conf.	Proc. Data Assign.	Scheduling
I	Yes	Yes	Yes	Yes
II	[1)] Yes/No	Yes	[1)] Yes/No	[2)] Yes/No
III	[3)] Yes/No	No	No	[4)] Yes/No
IV	No	Yes	No	Yes
V	No	No	No	[5)] Yes/No
VI	No	No	No	Yes
VII	No	No	No	Yes
VIII	No	No	No	[6)] Yes/No
IX	Yes	Yes	No	Yes

[1)] There is no need to update the control program code and its mapping with the IOD registers, whenever replaced device has the same device characteristics and all IOs had been wired with Sensors and Actuators in the same way.
[2)] There is no need to update a schedule, if the replaced device has the same device characteristics and the data to be sent (messages) are the same as well.
[3)] If a module has been removed, all functions related to it should be deactivated. If this function is already offered by the engineering tool, e.g. Simatic Step 7, where a defined code area can be activated or deactivated with a mouse click [27]. It applies also to [1)] in case of adding a module that has been already considered in the control program and has to be only activated after adding the module.
[4)] Whenever an IOD module was at the end of a line or a branch, a schedule does not need to be updated. However, the reserved for that IOD slots will remain reserved, causing decreased performance in terms of resource utilisation.
[5)] Whenever the switch was replaced with one that has the same device characteristics, the old schedule can be used.
[6)] Whenever the replaced wire has exactly the same transmission parameters and length, the schedule does not need to be re-calculated.

As can be seen in table 2.6, all manual engineering steps are not always required. However, in case of scheduling and assuming the worst case, a new schedule is required every time. There are exceptions that are typically related to exchange of unit (IOD, switch or wire) with the same device characteristics or removing a unit completely from the system.

2.6 Problem Definition

The current NCSs build upon TTEB technologies, achieve sufficient, high real-time performance and determinism, allowing communication with cycle times in the range of tens of microseconds, and a jitter below $1\mu s$. This high performance is achieved through a definition of strict communication rules, according to which the real-time communication is organised. These rules can be mapped into medium access approaches, such as the summation frame, where all nodes have to include and/or extract their real-time data to/from a circulating in a logical ring frame. Another way to achieve this high performance is utilisation of a strict TDMA communication schedule, where either one master node, holding a schedule, polls all slave nodes for a real-time data, or the schedule is distributed to all network members and they have to send data according to the time slots assigned in a TDMA schedule. Such strict rules are calculated for a specific setup that is defined through a network topology, involved devices with their individual characteristics as well as communication requirements. Whenever the setup is changed, the communication rules are redefined, and the whole communication system is restarted.
Since, scheduling of dependable tasks with resource constraints are often **hard mathematical problems** [82–84], the time needed to calculate a new schedule can be **unpredictably long** (e.g. in case of scheduling problems out of the family of NP-hard optimisation problems). Fortunately, there are solutions available in the current literature that use approaches like heuristics, genetic algorithms, etc. to find a sub-optimal solution. However, often the **scheduling time** was not taken into account as an optimisation criteria. Additionally, the calculation time may grow exponentially as the number of tasks to be scheduled grows [85]. Moreover, often the assumptions for defining scheduling problem, were not realistic, thus making the scheduling algorithm more generic, but on the other hand more complex. It can be said that the **highly complex algorithms**, which are typically used to calculate the communication schedule, are limiting the flexibility of such systems. These algorithms require **powerful computing hardware** in order to calculate the TDMA schedule. Additionally, the scheduling time can last several hours (depending on the formulation of the scheduling problem and used algorithms), and in that time a new device might have to wait for the integration [30]. Therefore the first research question that has to be answered in this dissertation reads as follows:

1. Research Question - is it possible to formulate the scheduling problem of TTEB systems considering relevant industrial requirements in such a way that it will be solvable in a short and deterministic time and the obtained scheduling result is optimal?

Communication reliability can be achieved through several different approaches. The most common methods to improve reliability among the INSs are **dynamic and static redundancy mechanisms**. In case of dynamic redundancy, the network redundancy manager sweeps the connections with the layer-2 network redundancy protocol (NRP) frames to check the "health" status of the topology. Whenever, NRP frames are not received back by the manager, the redundant link is activated. These methods have many advantages, such as simplicity, cost effectiveness as well as communication resources efficiency. Using them, it is possible to achieve recovery time T_R around $5ms$. This time is mainly affected by a delay caused by **forwarding of NRP** frames by the network devices that grows proportionally to the number of network nodes. According to section 2.2.2, the T_R in TTEB systems should be smaller than the grace time T_G, which in case of TTEB is in the range between $0s$ and $3ms$. This fact excludes the possibility of using dynamic redundancy in TTEB systems. Static redundancy methods use redundant connection all the time. Frames which hold process data are duplicated, and afterwards send using paths to its destination. This allows to achieve seamless redundancy, with the T_R equal to $0s$. However in case of static redundancy methods, the algorithms have to consider two important facts. It is not always possible to find **two different paths** in the INS topology and therefore not possible to use algorithms proposed in [86–89] that are based on this assumption. Therefore, the problem has to be formulated as a problem of searching two **possibly different (disjoint) paths**, where its **complexity has to be kept low**, which do not further increase previously described **scheduling time**. The second research question that has to be answered in this dissertation reads as follows:

2. Research Question - is it possible to consider the redundancy aspects in the scheduling problem of TTEB systems, without affecting the time complexity of the scheduling approach and optimality of the scheduling result?

The engineering process of TTEB systems is rather a **manually oriented task**. It includes activities, such as writing of a control program, manual definition of the TTEB system topology and communication characteristics and mapping of the control variables defined with the appropriate IOs of the IODs. The same steps are required in case the current setup is changed. The only automatic step is the compilation and scheduling [26], which uses the input defined in the previous manual steps. After the engineering process is accomplished, the whole system is typically **restarted** in order to distribute and apply the new schedule [90]. This **manual and complex engineering** process significantly **limits the flexibility** of TTEB systems. The complexity is also related to mastering the engineering software required to engineer the system and the ability to deal with failures caused e.g. by the wrong software versions, the incorrect version of DDFs, etc. Furthermore, manual actions may also cause some **additional errors**, thus significantly prolong engineering process [91]. Dealing with these problems is also necessary, even if only re-scheduling of the communication has to be performed. According to table 2.6 in section 2.5, it is the case in 4 out of 9 possible INS changes. In the best case, 6 out of 9 changes can be re-engineered by just updating the communication schedule. Therefore, if the scheduling process could be performed without the

need of using engineering tool, the software engineering time of INS could be drastically shortened and reconfiguration process would be simplified. This leads to the last research question that has to be answered in this dissertation:

3. Research Question - is it possible to reconfigure a INS build upon TTEB technology at run-time, without disturbing the currently running NCS?

2.7 Solution Approach

The approaches and methods proposed in this dissertation help to overcome the problem of static nature of the TTEB systems. At the same time, keeping the real-time performance and the reliability at the required high level. The flexibility is achieved through partial duplication of the INS engineering tasks, including scheduling process from the engineering tool and placing it directly in field devices, such as PLCs. Hence, whenever specific changes of the INS are applied, a simplified ("light version") engineering of the system can be performed on-line, at the field level, without involving an engineering tool. To cover more use-cases (defined in table 2.6), the proposed approach is build upon the existing in the scientific literature engineering methods, which are developed to automate the manual engineering processes. These are topology detection approaches [92, 93], network auto-configuration mechanisms [26, 94, 95], self-description of entities [96, 97] and automatic assignment between PLC code variables and IOs (sensors/actuators) [98, 99].

The solution approach followed in this work to address the identified problems is split into three main components, which are described in detail in chapters 3, 4 and 5. A short overview about their content is given below:

- Careful analysis of the NCS requirements, in terms of typically used network topologies, communication patterns, cycle times, etc., helped to develop a **scheduling approach**, **tailored** to the real, **industrial needs**. It is based on the **slip-stream effect** [100], enriched by additional features, which support **typical topologies** available in the industrial environment. With the **low complexity algorithm** proposed in this work, it is possible to calculate an **optimal schedule** in a **deterministic short time**, even on a **resource limited hardware** device, such as a PLC. This allows to perform a "light version" of the engineering process at the field level, without the necessity of using an engineering tool installed on a dedicated PC.

- The **reliability approach** proposed in this work tackles both, **dynamic and static redundancy** methods. In case of dynamic redundancy, a special **network bypass** is proposed, which allows to significantly speed-up the forwarding process of network redundancy protocol (NRP) messages. This results in speeding-up of the link failure detection and **shortening of the network recovery time** T_R. Moreover, an approach allowing to find **two disjoint paths** for static redundancy is proposed. The path search process is part of the scheduling process. The proposed algorithm has a **linear time complexity**, thus finding the result is **deterministic** and can also be **performed by the PLC**.

- The last contribution includes an approach allowing **seamless integration** of INS changes into a running system, **without affecting** the NCS in any case. It is enabled by a **flexible PLC architecture** proposal and methods, and algorithms to **automatically detect** network topology, device characteristics and communication requirements. Furthermore, a method for **distribution and synchronous activation** of the newly calculated schedule is proposed. A complementary discussion about the hardware requirements to support the methods proposed in this work will provide support to physically build systems based on the components proposed in this dissertation.

The approach proposed in this work is accompanied by a deep investigation through mathematical analysis, simulation and real measurements. To validate and demonstrate the applicability of the proposed approach in a real industrial TTEB system, the evaluation is based on the PROFINET protocol or parameters that are characteristic for this technology. The reason for selecting this protocol was dictated by the fact that it is one of the most widely used industrial protocols in Europe. It became a de facto standard protocol in the German automotive industry [78]. Furthermore, it allows a simple integration of standard Ethernet devices in the PROFINET setup, without affecting its performance in any case.



http://www.springer.com/978-3-662-54649-9

New methods to engineer and seamlessly reconfigure time triggered Ethernet based systems during runtime based on the PROFINET IRT example

Wisniewski, L.

2017, XIX, 200 p. 113 illus., 43 illus. in color., Softcover
ISBN: 978-3-662-54649-9