Methods of Evaluation and Improvement on Cascaded Wired and Wireless Real-Time Communication Networks for Factory Automation

Von der Fakultät MINT - Mathematik, Informatik, Physik,
Elektro- und Informationstechnik
der Brandenburgischen Technischen Universität Cottbus-Senftenberg
genehmigte Dissertation
zur Erlangung des akademischen Grades eines

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Steven Dietrich

geboren am 21.08.1989 in Finsterwalde

Vorsitzender: Prof. Dr. rer. Nat. Peter Langendörfer
Gutachter: Prof. Dr.-Ing. Rolf Kraemer
Gutachter: Prof. Dr.-Ing Theo Vierhaus
Gutachter: Prof. Dr.-Ing Ludwig Leurs
Tag der mündlichen Prüfung: 15.03.2021

DOI: https://doi.org/10.26127/BTUOpen-5483

To Nathan and Bernhard.

Time is what we want most,

but what we use worst.

- William Penn

Abstract

In the midst of the fourth "Industrial Revolution", factory automation applications require a communication system that concurrently supports them with determinism, reliability, short communication cycles, and precise synchronization as well as mobility and extensibility. However, neither the currently employed wired nor the available wireless communication networks support all of these requirements at once. On the one hand side, wired networks lack mobility and extensibility. Slip rings and cable carriers can be used for some movements, but are expensive and require continuous maintenance. On the other hand side, wireless networks do not support reliability or determinism due to the stochastic nature of wireless channels. Moreover, none of the available wireless networks enable transmission latencies of 1 ms and below. Therefore, they cannot support the required short communication cycles. However, various research initiatives and the currently developed *5G* standard promise to overcome these limitations at least partially.

Nevertheless, even a corresponding wireless network will not directly replace all wired connections, but will initially be used for subsections of the data transmission, where e.g. mobility is of utmost importance. Accordingly, cascaded communication networks consisting of hierarchically ordered, wired and wireless subnetworks will emerge. In this thesis, we investigate the effects and dependencies of such a cascading for the real-time communication of factory automation applications. Therefore, we have to find methods to interconnect arbitrary wired and wireless communication networks such that we can maintain the reliable real-time performance of wired subnetworks and the mobility of wireless subnetworks without, or at most with minimal performance degradation. Ideally, with a cascaded network, no hardware or software adjustments will be necessary compared to the individual networks deployed today. This would allow the overall application to remain unchanged.

Based on a delimiting review of different application areas and a dedicated use case analysis of typical factory automation applications, we derive the most challenging requirements that have to be fulfilled by cascaded communication networks in order to gain wide acceptance. According to this review, we present several methods to reduce the overall transmission latency in cascaded networks. Starting with analyzing different interface concepts and continuing with frame conversion methods, we achieve a decision-making basis to select dedicated subnetworks for a cascaded network that can be used for closed-loop control applications. Additionally, we develop methods to optimize the

parametrization of the individual subnetworks with respect to the overall communication, such that the end-to-end transmission latency can be drastically reduced. For verification and validation, we design a generic model for analyzing the timing behavior of arbitrary communication networks and verify it based on an analysis of a machine tool application scenario. Additional measurements based on a real demonstrator implementation corroborate that, in industrial application, cascaded communication networks can provide under certain assumptions a more competitive performance compared to the currently used wired networks.

Kurzfassung (German Abstract)

Mitten in der vierten „Industriellen Revolution“ benötigen Anwendungen der Fabrikautomatisierung ein Kommunikationssystem, das gleichzeitig Determinismus, Zuverlässigkeit, kurze Kommunikationszyklen und präzise Synchronisation sowie Mobilität und Erweiterbarkeit ermöglicht, um damit eine intelligente, hochflexible Fertigung zu realisieren. Allerdings unterstützen weder die derzeit eingesetzten drahtgebundenen, noch die verfügbaren drahtlosen Kommunikationsnetzwerke alle diese Anforderungen auf einmal. Auf der einen Seite mangelt es den drahtgebundenen Netzwerken an Flexibilität und Skalierbarkeit. Schleifringe und Schleppketten ermöglichen zwar eine eingeschränkte Mobilität, sind jedoch teuer und erfordern kontinuierlich Wartung. Auf der anderen Seite unterstützen drahtlose Netzwerke aufgrund der stochastischen Natur der drahtlosen Kanäle weder Zuverlässigkeit noch Determinismus. Darüber hinaus ermöglicht keines der verfügbaren drahtlosen Netzwerke Übertragungslatenzen von 1 ms und darunter. Somit können sie die erforderlichen kurzen Kommunikationszyklen nicht ermöglichen. Verschiedene Forschungsinitiativen und der derzeit entwickelte Standard *5G* versprechen jedoch, diese Einschränkungen zumindest teilweise zu überwinden.

Auch ein entsprechendes drahtloses Netzwerk wird nicht alle drahtgebundenen Verbindungen direkt ersetzen, sondern zunächst nur in Teilbereichen der Datenübertragung eingesetzt werden können, in denen z. B. Mobilität von größter Bedeutung ist. Dabei werden kaskadierte Kommunikationsnetze entstehen, welche aus hierarchisch geordneten, drahtgebundenen und drahtlosen Teilnetzen bestehen. In dieser Arbeit wird untersucht, welche Auswirkungen eine solche Kaskadierung auf die Echtzeitkommunikation von Anwendungen der Fabrikautomatisierung hat und welche Abhängigkeiten dabei entstehen. Im Idealfall erfordert die Verwendung kaskadierter Netzwerke im Vergleich zu den heute eingesetzten individuellen Netzwerken keine Hardware- oder Software-Anpassungen, sodass die Gesamtanwendung unverändert bleiben kann. Um dies zu ermöglichen, wird untersucht, wie sich beliebige drahtgebundene und drahtlose Kommunikationsnetze miteinander kombinieren lassen, sodass sowohl die drahtgebundene zuverlässige Echtzeitfähigkeit, als auch die drahtlose Mobilität möglichst ohne Einschränkungen aufrechterhalten werden kann.

Hierfür wird mit einer abgrenzenden Betrachtung verschiedener Anwendungsbereiche und einer dedizierten Use-Case-Analyse typischer Anwendungen der Fabrikautomatisierung begonnen. Davon werden die anspruchsvollsten Anforderungen abgeleitet, welche die kaskadierten Kommunikationsnetzwerke erfüllen müssen, um möglichst effizient ein-

gesetzt werden zu können. Basierend auf dieser Analyse werden verschiedene Methoden vorgestellt, welche insbesondere zusätzlich auftretende Übertragungslatenzen in kaskadierten Netzwerken reduzieren. Hierbei werden verschiedene Schnittstellenkonzepte und Methoden der Nachrichtenkonvertierung zwischen den einzelnen Teilnetzwerken analysiert. Damit wird eine Entscheidungsgrundlage zur Auswahl dedizierter Teilnetze für ein kaskadiertes Netzwerk geschaffen, welches die geforderten Echtzeit-Regelungen ermöglicht. Zusätzlich werden Methoden entwickelt, mittels derer die Parametrisierung der einzelnen Teilnetze in Bezug auf die Gesamtkommunikation optimiert wird, um damit die Ende-zu-Ende-Übertragungslatenz drastisch zu reduzieren. Zur Verifikation und Validierung der vorgestellten Methoden wird ein generisches Modell zur Analyse des Zeitverhaltens beliebiger Kommunikationsnetzwerke entwickelt und anhand eines Anwendungsszenarios aus dem Bereich der Werkzeugmaschinen ihr Effekt bewertet. Zusätzliche Messungen an einer realen Demonstrator-Implementierung bestätigen die entwickelten Methoden und zeigen, dass kaskadierte Kommunikationsnetzwerke für industrielle Anwendungen unter bestimmten Annahmen eine konkurrenzfähigere Leistung im Vergleich zu den derzeit verwendeten drahtgebundenen Netzwerken bieten können. Somit ermöglichen sie die Realisierung der gewünschten intelligenten und hochflexiblen Fertigung.

Contents

List of Figures

List of Tables

Abbreviations

5G 5th Generation Mobile Networks.

APP Application Layer.

BER Bit Error Rate.
BPSK Binary Phase-Shift Keying.

CDD Code Division Duplexing.
CDM Code Division Multiplexing.
CDMA Code Division Multiple Access.
CM Condition Monitoring.
CNC Computerized Numerical Control.

DL Downlink.
DLL Data Link Layer.

FA Factory Automation.
FDD Frequency Division Duplexing.
FDM Frequency Division Multiplexing.
FDMA Frequency Division Multiple Access.
FPGA Field Programmable Gate Array.

GSP Global Sampling Point.

I/O Input/Output.
I4.0 Industry 4.0.
IE Industrial Ethernet.
IEC International Electrotechnical Commission.
IoT Internet of Things.

MAC Media Access Control.

NRT Non-Real-Time.

OFDMA Orthogonal Frequency Division Multiple Access.
OH Overhead.
OSI model Open Systems Interconnection model.

PA Process Automation.
PDF Probability Density Function.
PER Packet Error Rate.
PHY Physical Layer.
PLR Packet Loss Rate.
PSO Particle Swarm Optimization.
PSSS Parallel Sequence Spread Spectrum.
PTP Precision Time Protocol.

RE Resource Element.
RT Real-Time.

SDM Space Division Multiplexing.
SDMA Space Division Multiple Access.

TCP/IP Transmission Control Protocol/Internet Protocol.
TDD Time Division Duplexing.
TDM Time Division Multiplexing.
TDMA Time Division Multiple Access.
TSN Time-Sensitive Networking.

UDP User Datagram Protocol.
UL Uplink.

WDM Wavelength Division Multiplexing.

Nomenclature

N_i	subnetwork i
S_j	slave j
s_{Max}	maximum number of slaves
L_k	level k
l_j	level on which S_j is located
D	amount of data in bit
d_j	individual data packet of S_j
R	amount of data in number of resource elements
C	cost function
g	weighting factor
$A_j(k)$	adjacency list to get representing N_i of S_j on L_k
V_i	list of d_j to be transmitted in N_i
t_{DL_i} / t_{UL_i}	offset of DL / UL frame of N_i
t_{DLRx_j} / t_{ULRx_j}	DL / UL data reception time of S_j
$t_{\mathrm{DLTx}_{i,j}}$ / $t_{\mathrm{ULTx}_{i,j}}$	DL / UL data reception time of S_j in N_i
$t_{\mathrm{DLAlo}_{i,0}}$ / $t_{\mathrm{ULAlo}_{i,0}}$	earliest allocation time for DL / UL real-time data in N_i
t_{DLAct_j}	DL activation time of S_j
t_{ULAcq_j}	UL acquisition time of S_j
t_{DLReply_j}	reply time on UL data of S_j
t_{GSP}	timing of the GSP
$\Delta t_{\mathrm{Cycle}_i}$	cycle time of N_i
$\Delta t_{\mathrm{DLData}_i}$ / $\Delta t_{\mathrm{ULData}_i}$	transmission duration for DL / UL data of N_i
$\Delta t_{\mathrm{DLFrame}_i}$ / $\Delta t_{\mathrm{ULFrame}_i}$	transmission duration for DL / UL frame of N_i
$\Delta t_{\mathrm{DLOH}_i}$ / $\Delta t_{\mathrm{ULOH}_i}$	transmission duration for DL / UL overhead of N_i
$\Delta t_{\mathrm{DLRTOfs}_i}$ / $\Delta t_{\mathrm{ULRTOfs}_i}$	DL / UL real-time data offset of N_i
$\Delta t_{\mathrm{DLAlo}_{i,j}}$ / $\Delta t_{\mathrm{ULAlo}_{i,j}}$	offset to specifically allocated DL / UL RE of S_j in N_i
$\Delta t_{\mathrm{DLInact}_i}$ / $\Delta t_{\mathrm{ULInact}_i}$	DL / UL inactive time of N_i
$\Delta t_{\mathrm{ConvLn}_i}$ / $\Delta t_{\mathrm{ConvLp}_i}$	conversion time to next / previous N_i
$\Delta t_{\mathrm{DLData}_{i,j}}$ / $\Delta t_{\mathrm{ULData}_{i,j}}$	transmission duration for DL / UL data of S_j in N_i
$\Delta t_{\mathrm{DLTran}_{i,j}}$ / $\Delta t_{\mathrm{ULTran}_{i,j}}$	transmission duration for DL data to / UL data from S_j in N_i
$\Delta t_{\mathrm{DLProc}_j}$ / $\Delta t_{\mathrm{ULProc}_j}$	DL / UL data processing time of S_j
$\Delta t_{\mathrm{DLComp}_j}$	computation time of DL reply for S_j

PER	packet error rate
MTTF / MTTF_d	mean time to (dangerous) failures
$B_{10\text{d}}$	average time until 10 % of a component type fail
n_OP	number of duty-cycles per year
d_OP	number of operation days per year
h_OP	number of operation hours per day
P_pa	packet per year rate
σ	standard deviation

Chapter 1

Introduction

Industrial plants and applications are continuously developed and improved. New technologies and increased requirements led to major changes through the history of industry. The first "Industrial Revolution" introduced mechanization reducing manual labor. Within the second "Industrial Revolution", mass production was established. This was extended by the digitalization and introduction of computers and automated processes into the industrial environment during the third "Industrial Revolution".

Since this time, communication networks have been the essential basis of most industrial applications. These networks are required to transfer data and process them at the appropriate place. Thus, they have to be permanently modified and adapted to the applications they are designed for.

Now, the fourth "Industrial Revolution" is underway. Concepts inspired by the "Internet of Things" (IoT) and cyber-physical systems are influencing industrial automation processes. The objective of these concepts is a transparent data exchange between any physical devices as well as their remote monitoring and control. At the same time, it should be possible to easily transfer parameter settings from one production to another. This enables automation supported by artificial intelligence, a higher degree of connection between heterogeneous devices, and the shifting of data collecting and processing to cloud-based applications [12–14]. These concepts also form the term "Industry 4.0" (I4.0), which was originally introduced in Germany and has became a global buzzword [15, 16]. The terms "Connected Industries" or "Industrial Internet of Things" are also often used synonymously. I4.0 further includes the evolution towards distributed production organizations with low-energy processes, collaborative robots, and self-organizing manufacturing by goods that are (virtually) connected to each other and integrated in the whole manufacturing and logistics chain.

In order to exploit the potential of these concepts, the underlying communication networks are confronted with new requirements, like increased connectivity. The currently used communication systems already enable a reliable data exchange with high data rates. However, since they are mainly based on wired connections, they offer only a small degree of flexibility and scalability. Thus, some of the most important challenges are to increase their flexibility and mobility while reducing the cost of installation and maintenance [17]. One possible approach to fulfilling these requirements is the deployment of wireless communication networks.

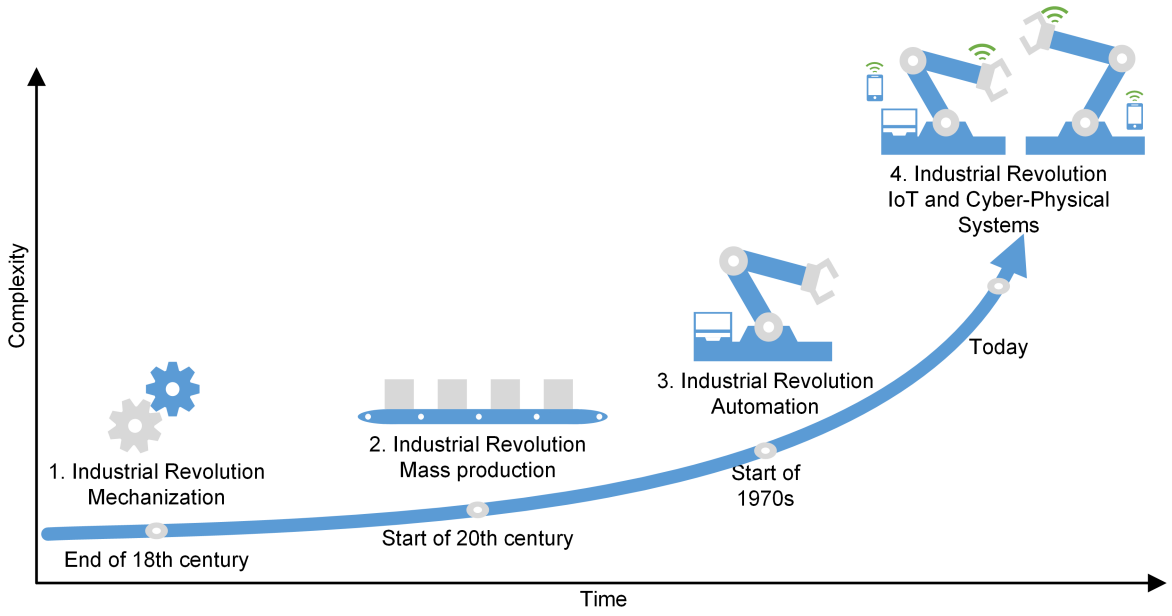

Figure 1.1: *Industrial Revolutions.*

Wireless communication, being almost ubiquitous, is part of many different applications in private and office domains. Compared to these domains, industrial communication represents a relatively new market but less substantial for wireless systems. However, due to the potential of this market, the needs of industrial applications are also considered in newly developed wireless networks, like the *5th generation mobile networks (5G)*. In the remainder of this chapter, we will further introduce these requirements.

First, we start with a classification of industrial communication according to the different levels of the automation pyramid and the scope of their applications in Section 1.1. Next, we introduce fundamentals and important relations of industrial communication in more detail in Section 1.2. In Section 1.3 we briefly describe the development of industrial communication networks and present some typically employed wired and wireless communication protocols and their performance. In Section 1.4 we propose cascaded communication networks as an approach for overcoming standing problems and explain the corresponding challenges. We introduce related work in Section 1.5 and present the contribution of this work towards solving open issues in Section 1.6. Finally, in Section 1.8, we present the structure of this work.

1.1 Classification of Industrial Communication Systems

The first classification of industrial communication systems results from the different areas they are used in. Fig. 1.2 depicts the automation pyramid, which divides these areas into six different levels [18, 19]:

- management level,

- planning level,
- supervisory level,
- control level,
- field level, and
- production level.

Even though the boundaries between the individual levels are becoming increasingly blurred today, we will present them briefly below to give an initial overview.

The management level forms the top level of a factory. It contains the plant's entire planning and organization. This includes a rough production planning as well as the supervision of the ordering process. At this level, the communication network must enable the internal exchange of large amounts of data as well as provide an external connection.

At the planning level, the organization of the production itself is refined. Material and human resources are assigned to individual productions, and the operation of specific applications is scheduled. Moreover, production quality is also controlled. Regarding the communication network, a limited number of devices must handle large data volumes.

At the supervisory level, the production of single products is specifically planned and their construction is monitored. Human-machine interfaces, therefore, provide control and warning messages for the operators. Depending on the number of individual production lines, the number of devices connected to the communication system varies. At the same time, the amount of data that is transmitted per device is further reduced, compared to previous levels.

The control level forms the interface to the technical production itself. Here, control algorithms for the individual production steps are executed, and data are exchanged between several controller devices. Therefore, the transmission of a limited amount of data per device is required. However, this transmission has to be fast and errors shall not influence the production.

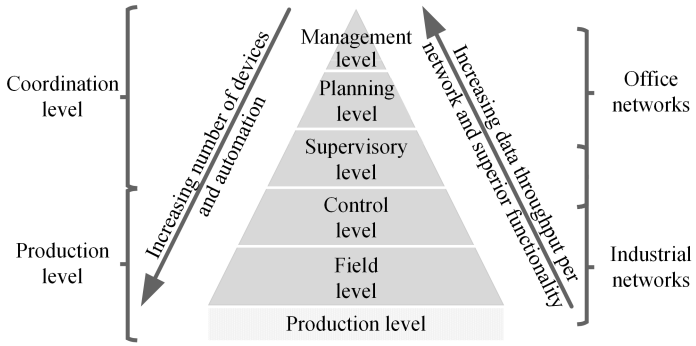

Figure 1.2: *Automation pyramid.*

The field level comprises all the sensor and actuator devices of a production. It contains all relevant processing information in form of dedicated input and output signals. This often only requires the transmission of simple raw data (electrical signals, voltage levels, etc.) without contextual information, but for a huge number of devices.

The production level, finally, represents the actual production process and the products themselves. In particular, information on product properties and production steps are provided here. Basically, this level contains only limited interconnections. The production level is often included in the field level or not even considered at all, since it is not really relevant for the actual automation. However, for the sake of completeness it is listed here.

While the upper levels can be served with standard communication networks from the office sector, the lower levels require special networks for industrial purpose in order to fulfill their requirements. These lower levels and their communication networks are particularly important for industrial applications. Therefore, this work will focus on their analysis and potential for improvement.

At these levels, industrial networks can be further classified by the scope of applications they are used for. These applications could be subdivided into three classes [20–22]:

- condition monitoring (CM),
- process automation (PA), and
- factory automation (FA).

These classes are briefly described below and compared in more detail in Sec. 2.2.

In the class for CM, the state of applications is observed by regular measurements and analyses of physical quantities like vibrations, oscillations, or changes in temperature. This allows a better planning of maintenance and therefore reduces downtimes. Additionally, it can increase safety and efficiency of the corresponding applications. For an effective CM, a large number of different sensors is spread over wide areas. However, the recorded measurements have no direct influence on the actual production process and are not included in the control. An update of these values is required only with a granularity of several hundreds of milliseconds [21]. More important than the data update rate is the energy consumption of the (sensor-) devices and a shared network transmission. Due to the large spatial extent, the individual sensors are typically self-contained. This includes their own energy supply, for example by using batteries.

Therefore, in the CM application class a communication network has to enable a connection of a high number of devices spread over a wide area. A connection from source to destination over these wide distances would either require high transmission powers or signal amplification. As a consequence, messages are typically not transmitted directly from their source to their destination, but forwarded over several network nodes with shorter distances, thus creating a meshed wireless sensor-network. This allows different transmission paths, which also enable newly added or defective devices to easily be included or compensated for. Accordingly, these meshed network structures (Fig. 1.3 C) offer high flexibility and large spatial coverage. Concerning the requirements of CM, these networks are optimized for low energy consumptions, including enhanced

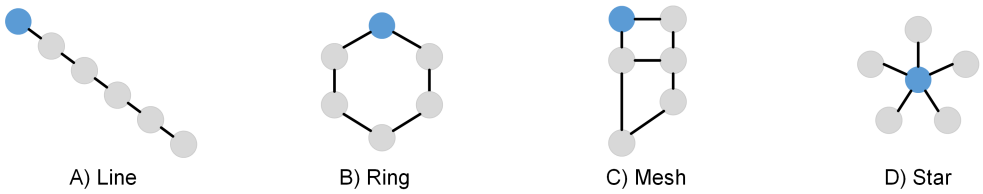

Figure 1.3: *Typical topologies of industrial communication networks.*

data sensing and transmitting as well as optimal routing strategies (e.g. [23, 24]). The disadvantages of these networks are large transmission latencies caused by the variety of involved devices along with their respective processing time to forward each request.

The class for PA deals with continuous production processes. Here, large quantities of a specific product are produced and processed. This includes for example the mixture of liquids or chemicals, or the production of quasi-endless products like wires or cables. These processes are relatively slow and uniform. Once such applications are in operation, only minor adjustments are required. Accordingly, the command data are typically exchanged in the range of several tenths of milliseconds. In addition to the command data, data for quality assurance have to be transmitted as well. This can require up to thousands of Bytes of data per device and communication cycle [22].

Correspondingly, the PA application class requires a communication network that enables high data throughput and a cyclic data exchange. The command data between a controller and its sensors and actuators are typically exchanged in a logical star topology. Therefore, all actuators can be controlled individually. As a variation, physical mesh or ring network topologies (cf. Fig. 1.3) are used in order to reduce the connection effort. However, the network extent has also to correspond to the large applications that could extend over several hundred meters. Moreover, different parallel processes require that communication systems coexist to a certain degree.

The class for FA mainly concerns small discrete productions or processing procedures. Individual products are for example manufactured, assembled, tested, or packaged in this application class. Several different actuators interact with each other and with the workpiece while additional sensors monitor them. This requires a very precise synchronization. In order to optimize these dynamic applications, closed-loop control algorithms are used that enable fast feedback and process adaption. This means that a fast data exchange with short cycle times in the range of sub-milliseconds is required between all devices [22]. Each of the transmissions could contain important data for the control algorithm, such that already small disturbances in the data exchange might result in malfunctions. The spatial extent of each individual FA application is relatively small. However, many are closely placed and work in parallel.

Communication networks for the FA application class have to support a high degree of coexistence, such that the data exchange does not interfere with neighboring systems. Typically, wired networks with a physical line or ring structure are used for this purpose. They facilitate high determinism and their interconnections require low complexity. Additionally, low latencies and a fast forwarding of data from device to device are

needed to meet the required fast cyclic data exchange. Therefore, special hardware components are often implemented. The required reliability is achieved by redundancy and robust error correction, that ensures low error rates.

In terms of timing requirements, FA applications are the most demanding industrial applications. Hence, only a limited number of communication networks are currently available for these applications. Today, these networks exclusively exist in a wired state. Accordingly, they are only partially useful to fulfill the requirements resulting from the I4.0 concepts. As a consequence, this work will focus on the analysis and improvement of FA communication networks. This shall support their development regarding new trends and enable access to a large market potential. Therefore, in the next section, industrial communication will be described in more detail.

1.2 Fundamentals of Industrial Communication

Typical FA applications consist of a controller unit that operates several sensors and actuators. According to this structure, industrial communication networks mainly implement a master-slave related data exchange. In general, the master represents the controller that provides data for all its slaves. These slaves are the sensors and actuators that provide their current status and measurements for the master. The transmission direction from master to slave is called downlink (DL), while the opposite direction is called uplink (UL).

Basically, three different types of data can be differentiated [25]:

- cyclic real-time data,
- acyclic real-time data, and
- non real-time data.

Cyclic real-time (RT) data are required for the control loops of the application. This is often referred to as application data. They consist of the so-called command data transmitted by the master in DL direction and the so-called actual data transmitted by the slaves in the UL direction. According to the control algorithm, these data have to be transmitted in predefined periodic cycles. Such a communication cycle is defined by the time interval from the start of the transmission of one particular data packet until the transmission of the next subsequent update of this packet begins. The cycle time and the size of the cyclic RT data are typically constant during the runtime of an application. Usually, the requirement of very low cycle times is met using an unacknowledged transmission of these data [26]. In order to compensate for missing acknowledgment mechanisms, delay sensitive data are employed and a watchdog on the application level monitors their omission. Accordingly, they have to be transmitted before an upper latency bound is reached that is defined by the application. If the transmission is not possible before their respective deadline, they become obsolete. As a consequence, these data are typically not buffered or transmitted repeatedly but updated with the next communication cycle. If new data are not received before their deadline, they

are interpolated from older values until a certain omission degree is exceeded. Such an transgression will result in a failure of the application.

Acyclic RT data are for example alarm or interrupt signals. These are often referred to as signaling data. They are also used to indicate events like plugging in or unplugging slaves or user-defined events like stop signals. Due to their sporadic and unpredictable occurrence, their transmission cannot be monitored by a watchdog. Therefore, the reception of these data normally has to be acknowledged on the application layer. Like cyclic RT data, they are also delay sensitive, but typically more tolerant regarding their transfer. Their transmission partially allows a certain number of omissions and retransmission might be possible. However, a stop of the application can be caused if the data are still not transmitted correctly. The number of acyclic RT data is usually relative low for industrial application and they are considered subordinated to cyclic RT data [27]. Therefore, dedicated individual signaling transmission channels are typically not sufficient. Since RT channels are already available anyway, it has become industrial practice to integrate the acyclic RT data into the RT data exchange by means of a few information bits. This saves the overhead for additional signaling channels, guarantees a fast transmission and limits the number of unused resources while no acyclic RT data have to be exchanged. In case of a signaling, the (typically device-specific) information can be exchanged via non-real-time (NRT) channels after the application has been able to react directly to the signaling in RT. As a consequence, in the following we will not further differentiate between cyclic and acyclic RT data. Accordingly, we will not consider individual data channels for application data and signaling data.

In addition to RT data, industrial applications partially require a transmission of NRT data. These are often referred to as management data. They are used for setup, monitoring or configuration purposes, as well as for the initialization of the RT communication. This includes, for example, the discovery of new slaves and their basic services or the setup and readout of their configurations during maintenance. NRT data are not time critical, and a best-effort transmission approach can be used to forward them. They are typically allocated to spare capacities that are not used for RT data.

In order to enable the required precise cooperation of all network nodes, the transferred data must have a temporal relation to each other. Therefore, a so-called global sampling point (GSP) is typically implemented as a temporal reference. The GSP usually occurs once per communication cycle and has a static relation to the beginning of a communication cycle. Exactly at the GSP or related to this time, all slaves have to activate their command data and capture their actual data. Thus, the activation of command data that are received later than this time will be delayed about up to one cycle. Moreover, actual data transmitted earlier than this time might be acquired up to one cycle earlier. Both might have a negative influence on the control loop. Accordingly, the timing of the GSP has a major influence on the overall temporal behavior of an application.

In order to be transmitted, the data have to be extended by a certain context information. Therefore, they are encapsulated into frames that contain, for example, preamble information, type identifiers, source and destination addresses and check sequences. These context information are also referred as overhead (OH), which is required for ev-

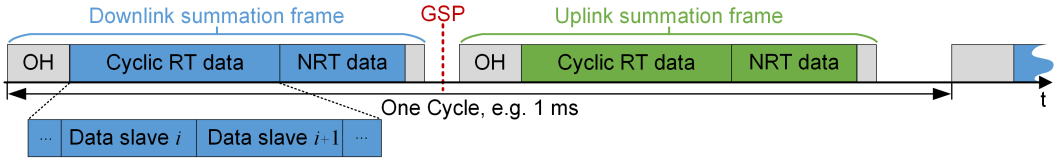

Figure 1.4: *Typical structure of an industrial communication cycle.*

ery single transmission. For the minimization of this overhead, common communication protocols for industrial applications transmit so-called summation frames. These frames aggregate several specific data, such that the context information has to be transmitted only once. This measure increases the data throughput of the network. Summation frames that combine all RT and NRT data of either DL or UL are schematically depicted in Fig. 1.4.

Most industrial applications are not based on a joint clock since the network nodes are spatially distributed. Therefore, additional procedures are implemented to synchronize their own individual clocks on each other. These procedures enable all slaves to satisfy the given communication schedule, like the joint GSP. The synchronization must compensate communication latencies and jitters and has a direct influence on the process accuracy of the system. The precision time protocol (PTP) [28] is one of the most popular synchronization protocols used in industrial communication networks [29]. It obtains a synchronization accuracy in the sub-microsecond range by using precise timestamps that are acquired in hardware. This requires additional transmitted data, such as synchronization telegrams. Due to individual clock drift of the different devices this transmission must be repeated periodically. However, a higher repetition rate could increase the transmission latency of the control and actual data. Therefore, synchronization accuracy and transmission latency respectively cycle time must be precisely matched. Otherwise, individual data could no longer be transferred in time or individual devices could no longer operate synchronously with the rest of the application.

As an alternative to the transmission of additional synchronization telegrams, which can delay the transmission of other data, it is also possible to use some specific (timing) features of the underlying communication network to generating the required synchronization. The master generates and maintains the reference time and distributes it to all its slaves. Therefore, the corresponding synchronization information are integrated into the frames for the date transmission itself. The slaves then use this information in addition to context information, for example from the time the message was received or their position within the communication hierarchy, to derive their synchronization. This requires a clear hierarchy of the slaves and precisely known transmission latencies. This concept enables a readjustment of the synchronization every communication cycle. Moreover, it also enables the synchronization with an external time source, if the master synchronizes to this time. However, it requires that the slaves have specific hardware installed to forward the frames with precise timing constrains.

The following section introduces the implementation of the previously described principles in some of the communication networks typically used for FA.

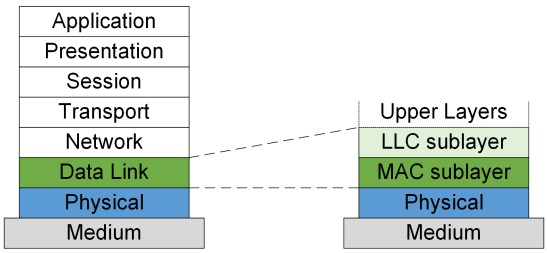

Figure 1.5: *OSI reference model (left) and IEEE 802-2014 specification (right) [11].*

1.3 Industrial Communication Networks

First industrial communication networks were built on individual point-to-point connections between the controller and all its connected sensors and actuators. The necessary wiring efforts resulted in high costs. Moreover, a large variety of different proprietary data transmission protocols emerged. Dedicated automation networks, called fieldbus systems, were developed to overcome these limitations and unify the data exchange. These systems also partially enable a better connection to higher levels of the automation pyramid [30]. As a large number of different manufacturers and operators developed individual fieldbus systems, no uniform standard could be established. Many different systems have been established with individual physical layers that are not or only partially compatible with each other. Examples are *INTERBUS* or *PROFIBUS*.

Over the last few years, Internet technologies based on standardized protocol stacks have become both more and more popular and part of our everyday life. The widespread consistent use of these technologies also ensures a further harmonization of industrial communication networks. Typical methods and implementations of wires and wireless networks are briefly described below.

1.3.1 Wired Industrial Networks

In order to achieve a common standard, wired industrial networks based on Ethernet as defined in IEEE 802.3 [31] were developed. This standard specifies the physical layer (PHY) and data link layer (DLL) of the Open Systems Interconnection (OSI) model. Moreover, it subdivides the DLL into media access control (MAC) layer and logical link control (LLC) layer, as shown in Fig. 1.5. Additionally, an implementation of the session and convergence layer is often moved into the application layer. This results in a narrowed layer model, also known as the Transmission Control Protocol / Internet Protocol (TCP/IP) model. The resulting Industrial Ethernet (IE) networks enabled the use of standard components from the office domain which had become inexpensive compared to the specific components required previously. However, standard Ethernet does not provide all of the real-time capabilities for industrial applications. Therefore, the standard has been adapted again by dedicated networks protocols to achieve the required performance [32].

Typical wired industrial communication networks used today are standardized in

the IEC 61784-2 [33]. They mainly use individual RT extensions for the Ethernet-standard [17]. As a result, they enable reliable cyclic communication with minimal variance in the data reception. The deviation from the timing is also referred to as jitter. As shown in Fig. 1.6, three different RT classes can be distinguished by their achieved timing behavior, which also results in different implementations according to the OSI model or respectively TCP/IP model [34, 35]:

- soft real-time,
- hard real-time, and
- isochronous real-time.

Soft RT protocols are used for applications that accept relaxed timing boundaries. These timing requirements are seen as guidelines. Instead of tight deadlines, they consider average times or statistical criterions. As long as the data are received with a certain tolerance, they are still considered as valid. Typical applications exchange their data cyclically in the range of 10 ms to 100 ms. Therefore, soft RT protocols realize their RT services mainly on the application layer of the OSI model. The actual data transmission is realized with best-effort. Unpredictable delays of lower layer network functionalities can partially retard this transmission. This limits the performance of soft RT protocols and hardly any timing guarantees can be made. The advantage of this RT class is that the networks can still be realized using standard Ethernet hardware. Additionally, the use of standard transport layer protocols like the TCP/IP or User Datagram Protocol (UDP) enable an easy integration into higher layers of the automation pyramid. Typical examples for soft RT protocols are *MODBUS TCP* or *EtherNet/IP*.

Hard RT protocols are used for applications that require a data transmission with a strict timing. A missed deadline will not immediately cause a system error, but the transmitted data might be discarded. Typical cycle time requirements for these applications are in the range of 1 ms to 10 ms. Therefore, hard RT protocols require modified data processing functions, including special timing methods. These functions are directly linked to the Ethernet MAC. NRT TCP/IP communication is still possible, but prioritization ensures that its impact on RT data transmission remains low. Typical examples for hard RT protocols are *Ethernet POWERLINK* or *PROFINET RT*.

Isochronous RT protocols are used for applications that require a very precise timing. The data have to be delivered in an exactly defined order and before strict deadlines. Data that are not received on time might cause immediately an error in the application. Although interpolation might be partially possible up to a certain extent, it deteriorates the quality of the control. Thus, a synchronized communication with highest precision in terms of jitter is required. It enables a completely deterministic data exchange and reduces stall or waiting times. Typical cycle time requirements of the corresponding applications are ≤ 1 ms. Additionally, the jitter of the cyclic data transmission and reception should be $\leq 1\,\%$ of the cycle time. In order to achieve these values, a special RT Ethernet controller is required in combination with modified data processing functions. A specific MAC layer has to ensure a strict communication scheduling. Furthermore,

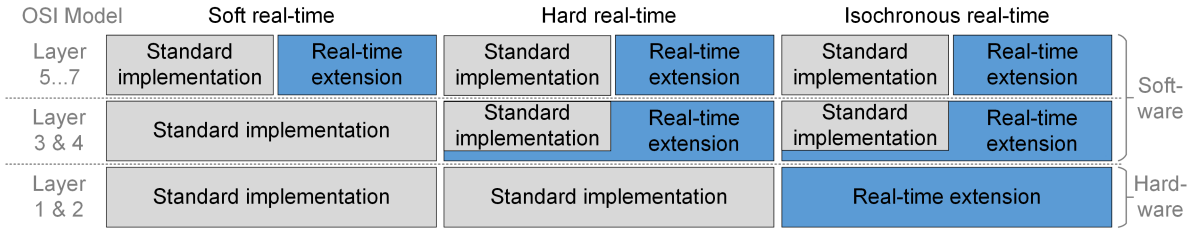

Figure 1.6: *Comparison of different real-time classes.*

isochronous RT protocols implement a precise clock synchronization that requires special hardware components. Today, this class of RT protocols is mainly represented by the protocols *Sercos*, *EtherCAT*, and *PROFINET IRT*. They build the basis for most isochronous FA applications. In Subsec. 3.2.1, we describe the *Sercos* protocol in more detail and use it exemplarily in the remainder of this work.

1.3.2 Wireless Industrial Networks

As introduced before, there is an increasing demand for wireless networks to fulfill new requirements in addition to the improvement of currently deployed wired communication technologies. These wireless networks have several advantages compared to wired networks [36]:

Wireless technologies offer more flexibility since they are not restricted to cables. Therefore, they enable a faster building and commissioning of applications and increase their productivity and reconfigurability. Furthermore, they can be used in harsh environments as well as for limited accessibility and moving application components. This could also increase the level of automation of an application or plant. They have lower installation costs since among other things expensive wear parts like slip rings or cable carriers are no longer needed. They allow easier retrofitting and extending of existing applications, which might reduce downtimes and increase overall quality. Additionally, the safety of an application can be improved since dangerous components can be controlled by wireless remote control instead of local operation units. Moreover, they allow a broadcast data transmission. This also enables entirely new possibilities of automation and manufacturing and a rethinking of the established concept of networking.

However, wireless technologies come with the disadvantage of a non-deterministic wireless transmission channel [37]. This transmission channel could be affected actively or passively and depends on environmental conditions. Active influences are for example caused by the communication system itself or by interfering systems. The robustness of the systems or its ability to correct errors may strongly influence the data exchange. Passive conditions are mainly the wireless propagation conditions. A possible line-of-sight, moving components, or many metallic objects in close vicinity might improve or reduce the quality of the received data. These influences are very difficult to determine or predict and dependent on location and time [38–40].

Accordingly, one of the biggest challenges for the use of wireless communication net-

works in the industrial sector is to enable the same high level of reliability as with the currently available wired networks. This also includes coexistence management with other applications and communication protocols. Another challenge is to ensure the interoperability of the different protocols. In addition, the easier data access of wireless technologies due to the used transmission medium increases the risk of information theft and manipulation of exchanged information. As a consequence, security and safety mechanisms of the applications have to be revised. All these challenges of improving the wireless communication represent large fields of research on their own and are already partially addressed by several scientific and industrial projects. Therefore, they are only marginally considered in this work, and we assume methods that enable a mostly error free, secure and timely wireless transmission.

Despite these existing challenges, some wireless communication networks are already in use in industrial applications [41]. They mainly comprise standard devices for consumer networks due to a wider market acceptance. Some common examples are *IWLAN* [42] (based on IEEE 802.11 [43]), *WSAN* [44] (based on IEEE 802.15.1 [45]), *WirelessHART* [46] (based on IEEEE 802.15.4 [47]), or *IO-Link-Wireless* [48] (as extension of the IEC 61131-9 [49]). However, the RT and reliability requirements of FA applications are not fulfilled by these protocols since they do not support the required data rates and latencies or provide the necessary robustness and parallelism.

Accordingly, the development of wireless communication protocols that fulfill all the requirements of FA applications at the same time, is a vibrant research area. One possible technology is *5G*, which was first addressed by the Beyond 4G project of the UK-China Science Bridges [50]. The standardization is done by the 3rd Generation Partnership Project (3GPP) [51] and influenced by various projects (e.g. METIS 2020 [52]) and organizations (e.g. 5G-ACIA [53]). During its development, the requirements of industrial applications are explicitly taken into account. Therefore, it can be assumed that a *5G* network will enable cyclic communication with low cycle times and high reliability. However, the actual performance that can be achieved and the use cases for which this technology will be sufficient cannot yet be fully predicted. Its earliest commercial introduction in the industrial area is not expected to happen before 2020.

Another promising wireless network, is *ParSec*. It is being developed as part of the ParSec project [54], funded by the German Federal Ministry of Education and Research (BMBF) [55]. The corresponding methods are currently in the final state of design. This network protocol is being especially adapted to FA requirements and particularly easy to integrate into existing wired applications. Therefore, several parts of this work were considered for the development of the current protocol. Some exemplary characteristic performance figures of the *ParSec* protocol are described in more detail in Subsec. 3.2.2. However, its usability in the industrial context, under real environmental conditions still needs to be proven.

1.4 Problem Statement

Ideally, data can be exchanged between any components of an industrial application in an arbitrary way without any substantial limitations. Thus, applications with such an

ideal date exchange would be free of limitations due to cable-based boundaries. Mobile components could easily be integrated into the real-time data exchange required for closed-loop control, as could static components. Applications could arbitrarily be adapted in order to be prepared for shrinking product life cycles and continuously intensifying competition. Data transmission via a attrition-free medium would reduce the operational costs associated with maintenance and service especially with moving components. At the same time, material costs could be saved during setup, installation would be simplified and requires less time and space requirements will be reduced, which in turn reduces expenses.

However, until today there are only wire-based communications networks capable of fulfilling the requirements of FA applications with respect to determinism, reliability, low transmission latencies, precise synchronization. Unfortunately, they lack of flexibility and extensibility. Accordingly, expensive and high maintenance-prone cable carriers or slip rings have to be used so that moving components can be connected. This reduces the flexibility of industrial applications. Additional components for application extensions can only be integrated into the control loop by means of additional wiring. This results in high costs both during installation and operation.

Therefore, in this work we investigate the integration of a wireless communications system in order to replace error-prone wired cable connections. This would allow retrofitting and expanding existing applications more cost-effectively and with less effort. Operational costs could be reduced by eliminating the need for maintenance on worn cables. Installation costs and the required installation space for new applications could also be reduced. Wireless communication would give the industrial applications additional flexibility and extensibility that is required to implement modern automation concepts.

This requires a wireless communication network that is capable of fulfilling all the requirements of FA applications at once. Despite all completed and ongoing work in this area, such a network still needs to be developed and fully elaborated. Moreover, in order to replace cable carriers or slip rings and extend wired networks, such a wireless communication network must also be seamlessly integrated into existing and currently used FA networks technologies. Especially at the beginning of the introduction of such a wireless network such an integration leads to a cascaded communication system with hierarchically ordered, wired and wireless subnetworks as schematically depicted in Fig. 1.7. However, the effects and dependencies of such cascading have not yet been fully investigated either. Which key performance indicators does such a cascaded communication system needs to fulfill? Which interfaces and gateways between the individual subnetworks are required? How do the subnetworks have to be designed in order to achieve no or at most only limited performance degradation due to the cascading? How do the effects scale according to the number of subnetworks and levels of the system? How can the overall performance be increased? How can the parameterization for such a cascaded network be derived according to the application? Can these considerations be extended to form a cascaded communication network with arbitrary subnetworks?

In the following sections we give an overview of related work and present the contribution of this work to these open questions.

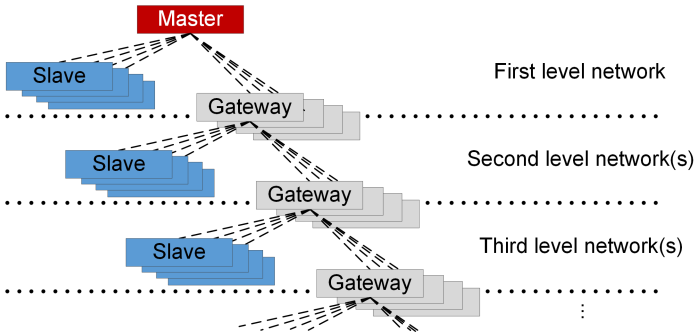

Figure 1.7: *Schematic representation of a cascaded communication network.*

1.5 Overview of Related Work

The currently employed wired communication protocols are mostly based on IE. Its standard Ethernet physical layer itself does not provide the required real-time capabilities. Individual extensions of the standard have been developed to solve this problem. However, the resulting network protocols currently used provide only limited interoperability. In order to overcome these compatibility problems, a recent research field deals with the extension of the Ethernet-standard with relevant real-time properties. One typical example is the development of the Time-Sensitive Networking (TSN) standards and their integration into the industrial communication, as for example shown in [17, 56, 57]. Another example is represented by the Time-Triggered Ethernet (TTEthernet, TTE) standard [58–60].

However, these extensions will not improve the inherent lack of flexibility of Ethernet due to the wired implementation. Therefore, further research directs to the development of a suitable wireless communication protocol for FA applications. Promising technologies such as *ParSec* or *5G* are still in progress, as introduced earlier.

Nevertheless, FA as well as industrial communication in general is a conservative domain with typical application operation times $\geq$10 years [36]. Therefore, for the near future, wireless communication networks are considered more as an extension than as a replacement of the currently employed wired networks [61, 62].

So far, there is only limited actual research concerning cascaded communication systems in industrial automation. Some of this research only deals with the interconnection of either wired (e.g. [63–65]) or wireless communication networks (e.g. [66]). Others combine typical industrial wired protocols with a standard wireless communication network (e.g. [67–70]). Due to the limitations of the individual subnetworks, these investigations do not provide a solution that fulfills all the requirements of FA applications. In [61] and [62], cascaded communication networks for industrial applications were investigated in a more general way. They point out that the efficiency and performance of the overall communication depend on the quality of integration of the individual subnetworks. Maintaining the RT performance over the entire communication chain requires special attention. In order to achieve this, precise determinism and extreme low transmission latencies are particularly important for industrial cascaded communication

systems [29, 71, 72].

The timing behavior of cascaded networks is addressed in [73–75]. The authors of [73] show an experimental evaluation of the service time of an access point in a network of standard Ethernet and WLAN. These networks do not provide the required RT behavior for industrial applications. Moreover, an analytical method is needed to create a universal approach for evaluating the timing behavior. The authors of [74] analyze the delay of individual network devices. However, individual delays also have an effect on subsequent transmissions, particularly in cascaded networks. As a consequence, not only the delay of a single device but the entire communication chain must be analyzed. The authors of [75] present an analytical approach to calculating latencies in heterogeneous distributed control systems and a method to schedule time critical applications. Therefore, the wireless subnetwork is split up into several equal parts in order to reduce the individually scheduled transmissions and related latencies. However, according to closed-loop applications, splitting up individual subnetworks can be very complex. A certain number of slaves with their data and performed actions might belong together and might hardly be divided.

Accordingly, due to the focus on different applications or communication networks, a lack of research could be seen in the investigation of arbitrary cascaded wired and wireless communication networks. In particular, universal approaches for analyzing their timing behavior have yet to be developed. Furthermore, methods to approach an optimal network integration and reduce transmission latencies still need to be found, especially regarding the requirements of FA applications.

1.6 Contribution of This Work

In the following, we present an outlook on how we contribute to solving the aforementioned problems.

Firstly, we perform a detailed analysis of factory automation applications and a comparison to other areas such as condition monitoring and process automation. This comparison enables the identification of requirements that cannot be fulfilled today and therefore require special attention in the development of cascaded communication networks. We specify the requirements with a dedicated use case analysis of typical machine tool applications, as well as printing and packaging machines. Based on this, we can derive whether a cascaded communication network can be used for all application classes or at least for a subset of them.

Secondly, we develop a generic network model representing arbitrary communication networks. This enables the validation and verification of their timing behavior and can be used for a better planning of cascaded networks. According to this model, we develop an algorithm that calculates the most important reference times for all slaves in a cascaded network, so their ability to fulfill application requirements can be analyzed.

Next, we compare different interface and frame conversion methods with respect to their viability. Both can be applied when planning a cascaded network for factory automation application. For an interface, we compare a tunnel based and a proxy based concept in detail and derive their strengths and weaknesses. These depend mainly on

the transmitted data and realized network structure. The frame conversion depends on the assumed subnetworks and may result in various different latencies. Therefore, our comparison can be used to choose dedicated components for the cascaded network that reduce additionally emerging latencies right from the beginning.

Finally, we develop methods to further improve the timing behavior of arbitrary cascaded communication networks. Based on the parameterization of the individual subnetworks, we come up with an analytical and a heuristic method to approach the optimum of their offset parameters and show that the overall transmission latency can drastically be reduced. The analytical approach provides slightly more accurate results but is more complex, especially for larger cascaded networks. On the other hand, the heuristic approach is less complex but requires a dedicated cost function. According to the complexity of the analyzed cascaded network, the use of either one or both methods might be sufficient; whereas, for the latter, the heuristic method can be used to establish an initial starting point for the analytical optimization. Additionally, we perform a detailed analysis of different resource allocation methods and compare their advantages and disadvantages in congruence to the radio channel available for a potential wireless transmission. As a result, we develop an algorithm that provides a latency reduced resource allocation for cascaded communication networks. It considers the size of individual data packets as well as the number of subnetworks they have to pass from source to destination. Moreover, we indicate several more potential methods for a further reduction of transmission latencies, which are not in the direct scope of this work and therefore left for future analysis.

For a proof of concept, we employ, on the one hand side, an exemplary machine tool application scenario that we derive from our use case analysis and, on the other hand side, a demonstrator implementation of a cascaded wired and wireless communication network to perform real measurements. Using the machine tool application scenario, we can show that overall transmission latency can drastically be reduced when applying our proposals and apporaches to reach the optimum. The measurements on the demonstrator implementation verify the usability of cascaded networks for future industrial applications, with only slightly worse performance compared to actually used wired networks but highly increased flexibility and scalability.

1.7 List of Own Publications

In scope of this thesis I have authored or co-authored the following publications:

Journal papers

- Steven Dietrich, Gunther May, Johannes von Hoyningen-Huene, Andreas Mueller, and Gerhard Fohler. *Frame conversion schemes for cascaded wired / wireless communication networks of factory automation.* Mobile Networks and Applications, vol. 22, no. 1, pp. 1-11, May 2017. [1]

- Johannes von Hoyningen-Huene, Holger Heeren, Lisa Underberg, Steven Dietrich, Philipp Kroos, Armin Wulf, Oliver Wetter, and Ruediger Kays. *Timing Evaluation of Cascaded Industrial Communication Networks.* IEEE Transactions on Industrial Informatics, vol. 15, no.10, pp. 5497-5504, 2019. [2]

Conference and refereed workshop papers

- Johannes von Hoyningen-Huene, Andreas Müller, Steven Dietrich, and Gunther May. *Comparison of wireless gateway concepts for industrial real-time-communication.* Proceedings of 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), Berlin, Germany, September 2016. [3]

- Steven Dietrich, Gunther May, Johannes von Hoyningen-Huene, Andreas Müller, and Gerhard Fohler. *Latency in Cascaded Wired/Wireless Communication Networks for Factory Automation.* Proceedings of 2nd EAI International Conference on Industrial Networks and Intelligent Systems (INISCOM 2016), Leicester, Great Britain, October 2016. [4]

- Steven Dietrich, Gunther May, Oliver Wetter, Holger Heeren, and Gerhard Fohler, *Performance indicators and use case analysis for wireless networks in factory automation.* Proceedings of 22nd IEEE International Conference on Emerging Technologies And Factory Automation (ETFA 2017), Limassol, Cyprus, September 2017. [5]

- Steven Dietrich, Gunther May, Johannes von Hoyningen-Huene, Andreas Müller, and Gerhard Fohler. *Modeling Offset Adaptations for Latency Minimization in Cascaded Communication Networks for Factory Automation.* Proceedings of 26th IEEE International Conference on Information, Communication and Automation Technologies (ICAT 2017), Sarajevo, Bosnia and Herzegovina, October 2017. [6]

- Steven Dietrich, Gunther May, Johannes von Hoyningen-Huene, Andreas Müller, and Gerhard Fohler. *Anforderungsanalyse und Optimierungen kaskadierter Netzwerke für die Fertigungsautomatisierung.* Proceedings of 8th Kommunikation in der Automation (KommA 2017), Magdeburg, Germany, November 2017. [7]

- Steven Dietrich, Lisa Underberg, Gunther May, Rüdiger Kays, and Gerhard Fohler. *Optimized Resource Allocation for Cascaded Communication Networks in Factory Automation.* Proceedings of 19th IEEE International Conference on Industrial Technology (ICIT 2018), Lyon, France, February 2018. [8]

- Lisa Underberg, Steven Dietrich, Rüdiger Kays, and Gerhard Fohler. *Towards Hybrid Wired-Wireless Networks in Industrial Applications.* Proceedings of 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2018), Saint-Petersburg, Russia, May 2018. [9]

- Lisa Underberg, Steven Dietrich, and Rüdiger Kays. *Optimierung eines Funksystems für hybride kaskadierte Netzwerke in der Fertigungsautomation.* Proceedings of 9th Kommunikation in der Automation (KommA 2019), Lemgo, Germany, November 2018. [10]

1.8 Organization of This Work

The remainder of this thesis is structured into the following chapters:

Chapter 2 - Analysis of Factory Automation Applications
In this chapter we perform an extensive analysis of FA applications. We describe the differences to CM and PA applications and specify the requirements of FA applications by presenting a use case analysis of typical applications. Moreover, we use the obtained requirements profile to derive comprehensive performance indicators of communication networks in order to prepare applicable capability profiles.

Chapter 3 - Modeling Industrial Communication
In this chapter we introduce the importance of modeling for the validation, development, and verification of communication networks. We introduce our developed generic network model to represent arbitrary communication networks. Using this model, we design an algorithm for the timing analysis of cascaded networks based on the data transmission of individual slaves. We use this algorithm to analyze our introduced application scenarios and reveal possible sources of additional transmission latencies.

Chapter 4 - Latency Optimized Interface Between Heterogeneous Subnetworks
In this chapter we present the importance of an adequate interface between individual subnetworks for the development of cascaded networks. We specify the terms and functionalities of the different interconnection devices and reveal missing performance evaluations of existing interface concepts. Therefore, we perform such an evaluation considering the real-time requirements of industrial applications. We calculate the timing behavior of three different interface concepts and discuss their usability for specific network structures.

Chapter 5 - Frame Conversion Schemes for Cascaded Networks
In this chapter we analyze the frame conversion from one subnetwork to a subsequent subnetwork as one additional source of latency in the data transmission of cascaded communication networks. Therefore, we derive possible frame conversion schemes that could emerge according to the used network protocols. We determine the transmission latency and jitter of these schemes and their subcategories and compare them by using two exemplary network protocols. Moreover, we employ a simulation of a cascaded communication network to verify our analysis.

Chapter 6 - Parameter Adaptations for Latency Reduction

In this chapter we investigate the parameterization of the individual subnetworks in a cascaded network in order to reduce the overall transmission latency. We have to adapt this parameterization according to the entire communication chain. Accordingly, we derive two methods for a specific offset adaptation and a latency optimized resource allocation of the individual subnetworks. Using these adaptations, we can drastically reduce the transmission latency compared to an unadapted parameter set, as exemplified in our machine tool application scenario.

Chapter 7 - Timing Evaluation of Demonstrator Implementation

In this chapter we use a demonstrator implementation of a cascaded network to validate the methods developed in this work. The demonstrator network represents a typical setup as expected for future applications. We describe several implementation aspects and link them to the corresponding methods for network integration and latency reduction. On the one hand side, we use the demonstrator to measure the synchronization accuracy that could be achieved for all slaves within the cascaded network. On the other hand side, we also measure the end-to-end transmission latency for different devices at different positions within the network. Accordingly, these measurements represent a proof of concept for the results presented so far in this work.

Chapter 8 - Outlook on Additional Methods for Improving Cascaded Networks

In this chapter we introduce and discuss several additional methods to improve cascaded communication network for industrial applications. These methods are intended to further reduce the transmission latency, improve the efficiency or facilitate the integration into existing systems. As they are mainly out of the scope of this work, this chapter represents more an outlook to future work.

Chapter 9 - Conclusions

In this chapter we draw the conclusions and summarize the major contribution of this thesis. Therefore, we evaluate the results of our work and put them in relation to the problem statement once again. Concluding remarks and an outlook to future work complete this chapter.

Chapter 2

Analysis of Factory Automation Applications

Industrial applications can be characterized in various ways, such as size, cost, throughput or functionality. In order to adapt the underlying communication, especially the spatially distributed functions, such as data acquisition, parameter measurement, machine operation, or process control, are of importance. These functions can be used to derive special requirement profiles. Comparing these requirement profiles with the ability profiles of communication networks, we are able finding an adequate communication network for the corresponding application.

However, on the one hand side, the requirements of industrial applications are very diversified and vary from implementation to implementation. Accordingly, it is very inefficient to assess each industrial application individually. Moreover, as introduced in Chapter 1, newly developed methods and advances demand extended functionalities, which further increase the complexity of the requirement profiles.

On the other hand side, there is no common metric to compare the abilities of communication networks with respect to the requirements of industrial applications. They are mainly characterized by parameters such as bit rate, bit error probability, or network range. Several of these parameters are of limited interest for industrial applications and cannot easily be compared with their actual requirements.

Therefore, in this chapter, we derive performance indicators for communication networks from real applications that enable a more reliable comparison of requirement and capability profiles. Additionally, a detailed requirements analysis will improve the development of a communication network for closed-loop control applications.

We start with an introduction of typically used performance indicators for industrial communication networks in Section 2.1. In Section 2.2, we review the requirements of industrial applications in general and identify the main differences of factory automation (FA) applications in comparison with other classes. In Section 2.3, we further specify the requirements of FA applications in a use case analysis of typical implementations. Based on this use case analysis, we introduce a specific application scenario for further investigations and validation in Section 2.4. In Section 2.5, we adapt typical performance indicators to the requirements of FA applications derived from our use case analysis and provide comparable capability profiles for cascaded communication networks. Finally, we summarize our results and draw our conclusions in Section 2.6.

2.1 Typical Performance Indicators of Industrial Networks

Industrial communication networks differ from communication networks in the office sector. According to their requirements, it is difficult to describe them using typical metrics such as bit error rate or average throughput. Consequently, special performance indicators are required to validate their usability. The discussion about these performance indicators and industrial requirements is a process that has been going on over the last decades. A categorization of industrial applications, as introduced in Sec. 1.1 results from industrial practice, but is not necessarily congruent with the view of communication technologies or standardization. As a result, several different approaches to describe the performance of an industrial communication network coexist (e.g. [76–79]). Therefore, in this section, we start with an extended overview about typically used performance indicators for industrial communication networks as we have presented in [5]. Later, we can use these performance indicators to create more general capability profiles, which we can finally map to the application's requirements profiles.

One of the most important lists collecting such performance indicators is introduced by the International Electrotechnical Commission (IEC) within the IEC 61784 and their subordinate standards. Nine parameters are introduced especially for industrial communication networks. These are [76]:

- delivery time,
- time synchronization accuracy,
- non-time-based synchronization accuracy,
- number of terminal slaves[1],
- throughput of real-time data,
- non-real-time data rate[2],
- basic network topology,
- number of switches between terminal slaves, and
- redundancy recovery time.

Delivery time is the basic parameter for evaluating the availability and real-time (RT) capability of an industrial communication network. It represents the time between the transmission of the first atomic component of a user data packet to the reception of the last atomic component of the same user data packet from a source to its destination. Within the IEC 61784, this time is particulary related to the application. Therefore,

[1] We exchanged the term *end nodes*, originally used within the standard, with *terminal slave* in order to continue the nomenclature used so far.

[2] The standard uses the term *bandwidth* instead of *data rate*. However, the description of the corresponding performance parameter is more related to the *data rate*. Since this term is also used more commonly in communications engineering, we will use it here as well.

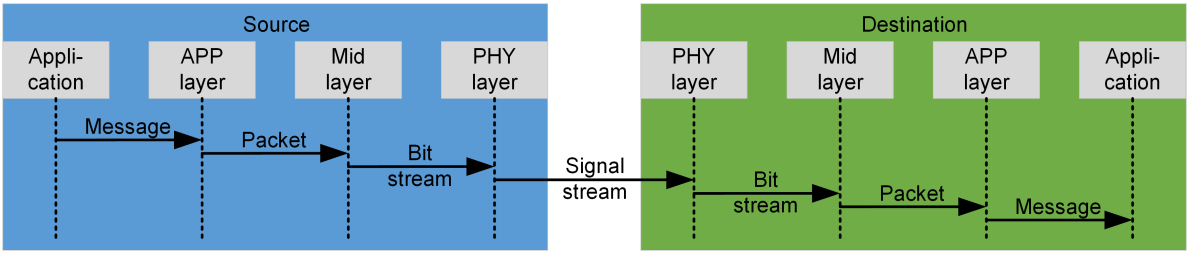

Figure 2.1: *Schematic representation of a successive DL message transmission.*

the delivery time has to be measured at the interface between the application and the communication system (cf. Fig. 2.1), including the necessary latencies for data processing and error correction. Moreover, the delivery time is specified to be unidirectional for either downlink (DL) or uplink (UL). The maximum delivery time shall be indicated for an error-free transmission, as well as an transmission with one lost frame and its recovery.

The time synchronization accuracy indicates the maximum deviation between any two node clocks. It is defined as the variance between the local clocks of the network nodes. Like the delivery time, the time synchronization accuracy is measured at the application layer.

The non-time-based synchronization accuracy does not refer to the local clocks but to a trigger of a periodical event like the global sampling point (GSP). Such events are for example used to generate a cyclic information exchange. The non-time-based synchronization accuracy indicates the maximum jitter of the cyclic behavior of any two network nodes. It can further be used to specify the coherency of information or the synchronization accuracy of events within the application. Therefore, this parameter is of high importance for FA applications.

The number of terminal slaves denotes maximum number of terminal slaves supported by the communication protocol and is related to the PHY layer of the network. This parameter can be used to validate the use of a communication network for a specific application. Moreover, it can be used to evaluate the coexistence of individual applications. However, this coexistence is of special importance to industrial applications. This should be considered in greater detail and by employing a special performance indicator.

The throughput of RT data indicates the maximum number of time critical information (in Byte) that could be transmitted per second on one link. It specifies the maximum number of dedicated transmission resources that the communication network provides to the application. According to the IEC 61784, it is measured unidirectionally. In cyclic communication systems, the throughput of RT data could be calculated by the quotient of RT payload length and transmission time interval on average. Therefore, it is correlated to the cycle time of the industrial closed-loop control applications. However, since the cycle time is of high importance for these applications, it needs to be addressed by a more precise parameter, rather than an average estimation.

The non-real-time (NRT) data rate specifies the fraction of the total data rate that is available for the transmission of NRT information. Due to finite transmission resources

per frame or cycle, it is related to the amount of RT data and therefore to the previous performance indicator. In addition to this indicator, the total link data rate shall be characterized.

The basic network topology defines the physical communication structure that is supported by the communication network. Per definition of the IEC 61784, only star, ring, or line structures are supported. Other network structures shall be built up by combining any of these topologies (cf. Fig. 1.3). However, the physical topology does not necessarily represent the logical communication structure. Moreover, the feasibility of either slave-to-slave or master-to-master communication using a certain communication network is not indicated by the standard.

The number of switches between terminal slaves refers to the number of interconnection devices between two slaves, only counting these switches with a relation to the application. They affect the transmission performance and communication relation. This factor corresponds to the network structure as well. It can also be used to derive the network expansion and range.

Finally, the redundancy recovery time indicates the maximum time between the occurrence of a failure and the recovered unrestricted operation. In other words, it denotes the required time for an application recovery. This indicator can be used to derive the RT capability of a communication network, especially considering temporal disturbances. It correlates to the maximum downtime of an application. Further, it is related to the reliability of a communication network although does not define this directly, as it is inherently assumed for wired networks. However, as reliability is of particular importance in industrial applications, it should be addressed in more detail separately, especially with regard to wireless networks.

Another list of performance indicators is introduced by the IEC 62657-2 [77], which mainly concerns wireless communication in industrial areas. Some of the presented factors are similar to those from the IEC 61784, albeit less related to the application. For example, data rate is only defined generally, without any consideration of RT or NRT data. However, several specific parameters for wireless data transmissions are introduced, highlighting gaps in the IEC 61784. Some important examples are [77]:

- availability,
- packet loss rate (PLR),
- frequency channel,
- power spectral density,
- security level, and
- distance between wireless slaves.

However, especially concerning the wireless data transmission, environmental parameters also have to be taken into account. The IEC 61784 for example indicates characteristics of the area of operation, relative movements of slaves, and regional radio

regulations. Further factors used for the performance evaluation of communication networks are mentioned in many other standards. Some examples are the IEC 61158, IEC 61850, IEC 62601, or IEC 62734.

Accordingly, the list of performance indicators employed to generate an capability profile of a communication network could be continued almost indefinitely. However, not all of these performance indicators are actually relevant for FA applications. For these applications, individual parameters are typically of limited interest as long as overall requirements can be achieved. Therefore, the performance indicators need to be more closely related to the actual requirements.

For example, an evaluation of RT Ethernet networks is performed in [78]. Therefore, the performance indicators described in the IEC 61784-2 are extended by the more application related parameter cycle time and jitter. This enables a good comparison of wired networks. However, for wireless networks, several important factors are missing.

Another attempt to relate the performance indicators to the corresponding applications is presented in [79]. Here, as a result of a case study, only cycle time, RT throughput, and the time it takes to poll a wireless node are used for a performance evaluation. This allows wireless networks to be evaluated by the same set of parameters. However, in order to refine this evaluation, the list of factors needs to be extended by more requirement driven elements, such as reliability issues.

The authors of [41] introduce a requirement analysis of example field level applications, which they uses to evaluate current solutions of wireless communication networks for industrial applications. However, in order to extend this validation to cascaded communication networks, the requirements analysis must be refined to derive the corresponding and comprehensive capability profiles.

For such a refinement, we will introduce a detailed use case analysis of closed-loop control applications of the FA class in the subsequent section.

2.2 Requirements of Industrial Applications

As introduced in Sec. 1.1, the classification of industrial applications into the classes condition monitoring (CM), process automation (PA), and factory automation (FA) has been established in industrial practice. Due to the application-oriented nature of this work, we will retain this perspective in the following. However, the requirements of specific applications of these classes can partially overlap. Accordingly, a large number of requirement definitions can be found in different variations. Subsequently, we summarize the most important ones as we have published in [5] and extend the created requirements profiles.

A basic description of a CM use case can be found in [80]. It describes the support of an average automotive plant by a wireless sensor network. Therefore, parameters such as the number of slaves, slave density, and a message loss rate are specified.

An extended overview of requirement parameters is given in [22]. Here, a distinction between all three application classes is made, while CM is further specified in the use cases of building automation, logistics, and infrastructure plants such as wastewater treatment plants. A large set of parameters is provided, whereas cycle time, amount

of user data, reliability in terms of a bit error rate (BER) and spatial extent of the application represent the most important ones for industrial practice. The necessity of the first three parameters results from the general structure of industrial processes presented in Sec. 1.2. Information on spatial extent is important due to the area of an industrial site that the communication system has to cover. At the same time, it can be used as an indicator of whether several applications can coexist in limited space available in industrial factories.

A further quantitative differentiation of FA and PA use cases is done in [21], especially focusing on coexistence. The application classes are differentiated into individual applications ("local") and a set of identical applications in a production hall ("global"). Additionally, a monitoring and diagnostics use case is specified, for both classes, which corresponds to the class of CM. For these use cases, the maximum transmission delay, update time, and telegram loss rate are specified.

In [34] a differentiation according to the RT requirements is presented. CM applications are assigned to the classes non- and soft RT, PA applications to soft and hard RT and FA applications to isochronous RT (cf. Fig. 1.6). According to the resulting latency requirements from the RT classes, demands for a synchronization accuracy of the master and its slaves are presented in terms of jitter.

Another requirements analysis that shows the scalability from a manufacturing cell up to the plant level is provided in [36]. This is derived by the investigation of various use cases for wireless networks, such as a robotic arm, a rotary storage, a high rack warehouse, and a refinery. Among other things, the required number of slaves are defined, as well as spatial dimensions and typical physical network structures. Additionally, the required number of parallel networks and locally parallel slaves are indicated.

In [41] a more detailed differentiation of closed-loop control applications is done, differentiating between machine tool, printing machines, and packaging machines. Although the use cases are not specified in detail, requirements for a certain cycle time, jitter, number of slaves and average amount of data per slave, and cycle are identified.

Finally, in [81], a questionnaire for a uniform use case analysis is elaborated, indicating sixteen requirement parameters. However, the evaluation of some applications shows that not all of these parameters are important or can be specified for every application.

The previously indicated literature shows that there is a variance in the introduced requirements, even within the same application class. Additionally, most of the example specifies only a subset of the industrial relevant parameters. Therefore, in order to generate a credible requirements profile of the three application classes CM, PA, and FA, we have revised the literature and added additional information from machine manufacturers and operators. The maximum requirements resulting from our study are shown in Fig. 2.2 and described in more detail in the following.

As introduced in Sec. 1.1, CM applications comprise state observation and regular measurements for an increase in efficiency, safety, and maintenance planning. The observed condition parameters are typically not included in the production control loop. Therefore, an update cycle of 100 ms is sufficient for most of the applications. Even the failure of some data transmissions does not necessarily influence the operation of the corresponding application. Accordingly, a typical required packet error rate (PER)

is estimated by 10^{-3}. Typical CM applications use up to 1000 slaves, which might be spread over the entire industrial site of up to $1\,km^2$. However, in order to be able relating the collected data to each other, all slaves must be precisely synchronized to each other. The maximum allowable deviation shall typically be 5 ms.

Since PA deals with relatively slow and uniform processes which change only little over time, control loops with an update cyclic time of 10 ms are normally sufficient. Precise synchronization is of minor interest for most of the use cases, and deviations of up to 100 ms are typically allowed. Also, the spatial extent, of up to $10\,000\,m^2$, and the typical number of 200 slaves are rather small as compared to CM applications. However, for PA applications, high amounts of data per slave and communication cycle are particularly important. For quality assurance and maintenance tasks, up to 1500 Bytes per slave might be transmitted, which strongly affects the requirements for cyclic data traffic. PA applications typically have a very long life span of $\geq$ 20 years, to which the communication system must also be designed. This also always results in the requirements of the seamless integration of new systems into existing ones. Due to the quasi-endless processes, for example in chemical production, an interruption would result in a big economic damage. Therefore, the communication system has to ensure very high robustness and reliability for the data exchange.

FA applications typically have a much smaller dimension compared to CM and PA applications. They typically require an area of $100\,m^2$. Moreover, the number of slaves and their respective amount of data are relatively small. For their dynamic control loops, they require a fast cyclic data exchange. Therefore, cycle times of down to $\leq 100\,\mu s$ can be found, whereas 0.5 ms are required on average. Additionally, all slaves need to be precisely synchronized with a maximum deviation of 0.25 µs. Furthermore, it must be ensured that no data is lost during transmission. Accordingly, a PER $\leq 10^{-9}$ is required.

More details of the literature review are presented in appendix A.1. As a result, Fig. 2.2 depicts the requirement profiles for the application classes CM, PA, and FA. These can be clearly distinguished from each other. CM applications have the highest requirements in terms of spatial extent and number of slaves. PA applications require especially large amounts of data to be transmitted, while synchronization is typically less important. Finally, FA applications are the most time critical applications and require the most accurate reliability.

With respect to the underlying communication, these RT and reliability requirements of FA applications are the most challenging. They are especially difficult to obtain for wireless and cascaded communication networks in an industrial environment. Accordingly, within this work, we will focus on the class of FA applications. Within the next section, we will refine the requirements by introducing a use case analysis of several specific applications.

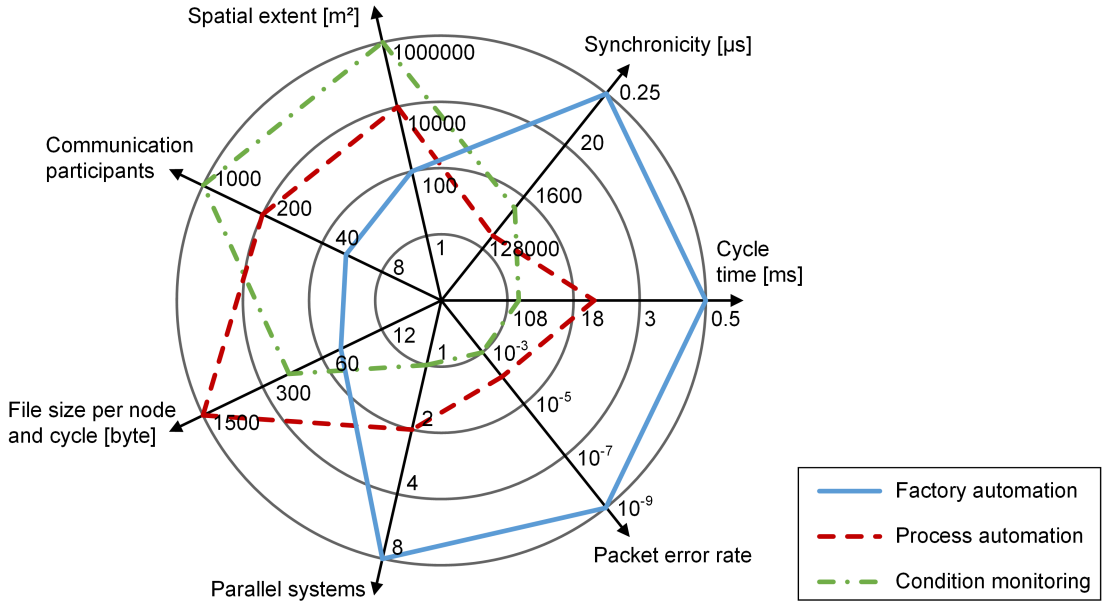

Figure 2.2: *Consolidated average requirements of application classes condition monitoring, process automation, and factory automation.*

2.3 Use Case Analysis of Factory Automation Applications

FA applications are very challenging for any communication network. As shown above, there are no uniform requirements within this application class due to the high complexity. In order to provide a further specification of the corresponding requirement profiles, we extend our dedicated use case analysis of specific applications presented in [5] and [7]. For this purpose, we focused on applications from the following areas:

- machine tools,
- printing machines, and
- packaging machines.

These applications form a good cross-section of most FA applications. We interviewed both users and manufacturers to get a comprehensive picture of the parameters actually required in practice. Our use case analyses shall enable an improved development of cascaded networks which are specially adapted to the actual requirements of industrial applications.

2.3.1 Machine Tools

Common examples of FA applications are machine tools. They process discrete products with the help of special tools (manipulators) like drillers or rotary cultivators. Typically, they are implemented as computerized numerical control (CNC) systems. The most important components of such a systems are controller, rack, tool gripper, and

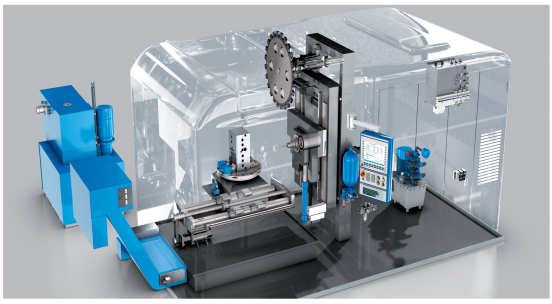


Figure 2.3: *Exemplary machine tool applications: drilling machine (left) and high speed milling machine (right)*[3].

assembly unit. Tool grippers and assembly units have to be flexible and can move or rotate in several directions. Some machine tool applications also contain several of these components cooperating on a single product. This requires precise motion control of different actuators simultaneously. Fig. 2.3 depicts typical examples of machine tool applications.

So far, the moving components are connected by cable carriers or slip rings. These are very expensive and subject to great mechanical stress. Accordingly, they require high maintenance efforts. Therefore, such applications will clearly benefit from wireless connectivity, if their timing requirements can be fulfilled.

Typical products of machine tools are manufactured with precision requirements in the micrometer range. This requires precisely synchronized motion control. For example in a high speed milling machine, a tool must be moved in several axes simultaneously. At a speed of 60 m/min, a dimensional accuracy of 1 µm requires a synchronization accuracy of 1 µs. Therefore, in motion control the deviation of all slaves from a global timing, such as for example represented by the GSP, shall not be higher than 1 µs. Otherwise, defects or inaccuracies in the processing may arise due to incorrectly cooperating components.

Due to high cutting speeds of up to 50 m/s and very quick changes of movement direction, special control algorithms with a fast parameter feedback are required. Therefore, command data and actual data are usually exchanged every 0.5 ms. Specific applications can contain up to 20 slaves and can transmit an average of 50 Bytes of data per slave and cycle in DL and UL direction each.

The frequent change of movement directions also results in high reliability requirements. Since each command from the master could result in a new direction of motion, not even a single data packet should be lost. Consequently, a PER of $\leq 10^{-9}$ is typically required.

Most machine tools are enclosed systems. They have a housing of metal and safety glass with a thickness of up to 2.5 mm. Their typical spatial extent is $5 \times 5 \times 2$ m, and several systems can be grouped up to an extent of $30 \times 20 \times 2$ m. This means that at least 24 parallel systems must be covered by the communication system and partially orchestrated with each other.

[3]Source: Bosch Rexroth Image Pool.

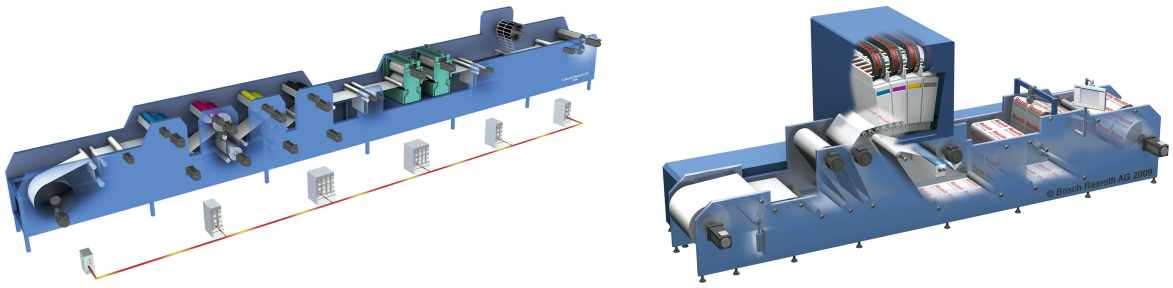

Figure 2.4: *Exemplary industrial printing systems: flexography printing (left) and injector printing machine (right)*[4].

2.3.2 Printing Systems

The second type of FA application presented here is represented by industrial printing machines. Printing machines can be further subdivided into flexography and injector printing machines. Fig. 2.4 depicts two typical examples.

For flexography printing, several rotating cylinders are used to print an image. For each color, a cylindrical mask is dyed and unrolled evenly onto a continuous passing paper. The overlay of different colors results in the final image.

The printing cylinders and the paper feed have to be precisely synchronized. In modern newspaper printing systems, the paper is unrolled at a speed of up to 20 m/s. On the final image, the individual layers of color provided by printing cylinders must not differ from each other by more than 5 µm. Otherwise individual colors will not be precisely aligned and the image will appear blurred. Accordingly, the worst case deviation in synchronization between the printing cylinders and the paper feed should not exceed 0.25 µs.

Due to the continuity of the printing process in flexography printing systems, a cyclic data exchange every 5 ms is sufficient. Furthermore, sporadic data packet losses are acceptable. Therefore, the required PER of 10^{-7} is slightly larger than for machine tools.

Flexography printing systems are large machines and can have a spatial extent of $25 \times 25 \times 3$ m. Moreover, several systems can also be combined into a cluster and even extent over several floors. The average number of slaves is about 100. However, modern systems with up to 250 slaves are currently being developed.

Another very important aspect for these printing machine applications is the paper exchange. When one role of paper is empty, it must be exchanged during the printing process. Therefore, so-called exchangers are used to replace the paper role, glue the new paper to the remaining end, and cut off the supernatant residues. Here, many process steps are mounted rotationally and are synchronized with the rest of the printing process via slip rings or cable carriers.

Injector printing systems, as second type of industrial printing machines, can consist of up to 30 printing heads with at least two sensors or actuators. The resulting average

[4]Source: Bosch Rexroth Image Pool.

Figure 2.5: *Exemplary industrial packaging machines: bottle filling line (left) and pick-and-place systems (right)*[5].

number of slaves is about 80. These are moving with a speed of up to 10 m/s and print different areas of an image. The range is up to 10 m, and movements can overlap. Again, this requires a very precise synchronization, although with a permitted maximum deviation of 1 µs, it is slightly less demanding than flexography printing systems.

Due to the more complex movements, a fast exchange of the RT data is required. Command and actual data are typically exchanged every 2 ms. The average data size for each direction is up to 40 Bytes.

2.3.3 Packaging Machines

The third class of FA applications presented here are packaging machines. Packaging machines are applications that complete stages of the packaging process. This includes for example filling, sorting, sealing, wrapping, or labeling. This class of application has the greatest variance according to the product, the machine operator, and the machine manufacturer. Two typical examples are depicted in Fig. 2.5.

Like machine tools and printing machines, packaging machines consist of several moving components that need to be coordinated and synchronized. One typical example are filling machines. Here, up to 15 bottles are clamped into a so-called revolver, filled and often directly sealed. At least three drives are rotationally mounted to clamp, fill, and close each bottle, resulting in at least 45 slaves per application. The revolver has a typical diameter of 6 m and rotates with a constant speed of up to 20 m/s. The movement of the drives must be coordinated with the rotation as well as with the bottle feeding and removal pipe. Due to the continuous rotation, a cyclic data exchange every 5 ms with a synchronization accuracy of up to 10 µs is usually sufficient. The average data size is 80 Bytes per slave and cycle in both DL and UL direction.

Another kind of packaging machine is pick-and-place systems. Here, unsorted products are taken from a supply by picker arms and dropped at a target position. The picker arms are often mounted on mobile carriages, which often have to cover distances of up to 4 m. Due to the unsorted product feeding, this requires shorter cycle times and better synchronization compared to bottle filling systems. Typical cyclic update intervals are provided every 1 ms. Some applications consist of up to 16 pickers with at least two drives for various movements, which requires a support of at least 32 slaves. However,

[5]Source: Bosch Rexroth Image Pool.

since the picker arms are mostly ordered subsequently, their movements cannot interfere with each other. Therefore, the drives of the moving machine components must only be synchronized with the supply and removal units. Accordingly, the maximum allowed deviation for the global time base can be up to 5 µs.

2.3.4 Consolidated Requirements

As shown before, the requirements of typical FA applications differ from each other. The consolidated requirements of the exemplified machine tools, printing machines, and packaging machines are shown in Fig. 2.6. A clear distinction can be made between all three application classes.

On the one hand side, machine tools require the shortest communication cycles and the lowest PER. On the other hand side, they contain only a small number of slaves compared to printing and packaging machines. Also, the spatial extent is lower compared to the other two application classes.

Printing machines can be clearly characterized by their required synchronization accuracy. At the same time, they require the largest spatial extent and number of slaves per application. In contrast, they require less feedback for their control loops due to the continuous printing process. This corresponds to lowest cycle time and PER requirements.

Packaging machines can be mainly differentiated by their required amount of data per slave that needs to be exchanged. On the other hand, their synchronization requirement is the smallest compared to machine tools and printing machines. The remaining parameters are mainly in between the other two application classes.

With this use case analysis it is now easier to validate the capability profile of a communication network with the specific requirements of FA applications. Additionally, it enables a differentiation whether all or at least a subset of the requirements can be met and for which class of applications it is most suitable.

In Sec. 2.5, we used this use case analysis to derive performance indicators for cascaded communication networks that are actually relevant for industrial applications.

2.3.5 Review of the Packet Error Rate Requirement

As shown in our use case analysis, the PER is an important performance indicator for industrial communication networks. Based on this use case analysis, a PER of at least $\leq 10^{-9}$ has to be provided in order to fulfill the requirements of all applications. This value can be frequently found in the literature and is usually claimed without further argumentation or explanation. However, underlying standards (e.g. [22,31]) quote values for a BER of 10^{-5} to 10^{-8}, which hardly fit to this PER specification. Moreover, the term PER itself is misleading for industrial communication networks that manly use summation frames, which encapsulate individual data packets to transmit information cyclically (cf. Sec. 1.2). However, it is usually assumed that an entire summation frame is corrupted and discarded as faulty as soon as a single one of its data packets is faulty.

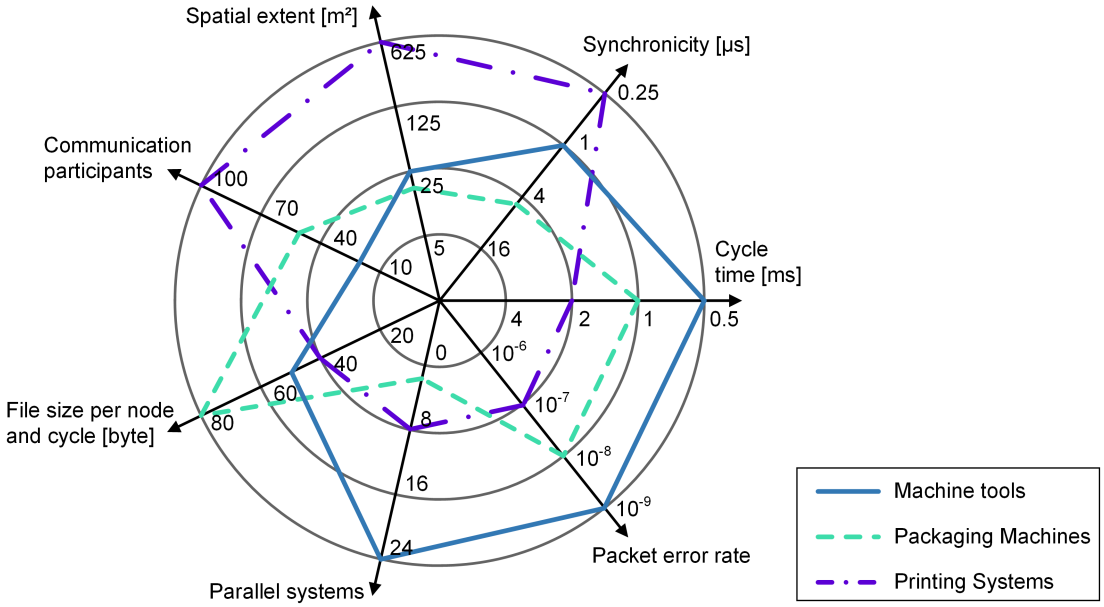

Figure 2.6: *Use case analysis of typical factory automation applications.*

Accordingly, the terms packet and packet error are usually used synonymously for frame and frame error for industrial communication networks.

Especially for the wireless and cascaded communication networks considered in this work, this requirement represents a particular obstacle. Therefore, we performed a bottom up analysis of an industrial application with an underlying cascaded communication network in order to derive more reliable PER requirements, presented in [5]. In the following, we will introduce and refine this analysis.

First, we have to calculate the failure probability of the investigated application. Here, we can use the ISO 13849 [82] to set the PER of the communication system in relation to its uptime. According to the ISO 13849, we have to differentiate between safe and dangerous faults. Safe faults cause the application to switch to a safe state, but reduce the availability. Dangerous faults could lead to undefined states and therefore have be avoided. Accordingly, the standard defines a mean time to failures (MTTF) and a mean time to dangerous failures ($\mathrm{MTTF_d}$). As a conservative assumption, 50 % of all faults are assumed to be dangerous, if not otherwise stated in the application or subcomponent description. Accordingly:

$$2 \cdot \mathrm{MTTF} = \mathrm{MTTF_d} \tag{2.1}$$

Moreover, the ISO 13849 describes three methods of gaining further information about the $\mathrm{MTTF_d}$. These are (in decreasing accuracy):

- using manufacturer specifications,
- considering average values for components listed in the appendix of the ISO 13849, or
- assuming a component life time of ten years.

Since a manufacturer of a corresponding component cannot know the exact environmental conditions and usage of its component (e.g. application, surrounding condition, etc.), they can hardly provide appropriate $\mathrm{MTTF_d}$ values. Therefore, the ISO 13849 lists the average number of actions for several electric and pneumatic components, until 10 % of them fail ($B_{10\mathrm{d}}$). We can now calculate the $\mathrm{MTTF_d}$ values with:

$$\mathrm{MTTF_d} = \frac{B_{10\mathrm{d}}}{0.1 \cdot n_{\mathrm{op}}} \tag{2.2}$$

Here, n_{op} is the number of duty-cycles per year and can be calculated with:

$$n_{\mathrm{op}} = \frac{d_{\mathrm{op}} \cdot h_{\mathrm{op}} \cdot 3600\,\mathrm{s/h}}{\Delta t_{\mathrm{Cycle}}} \tag{2.3}$$

where d_{op} is the number of days per year and h_{op} the number of hours per day during which the system is operated.

Now, the $\mathrm{MTTF_d}$ of the entire application can be calculated considering the total number of components N:

$$\frac{1}{\mathrm{MTTF_d}} = \sum_{i=1}^{N} \frac{1}{\mathrm{MTTF_{d,i}}} \tag{2.4}$$

Finally, as defined in the IEC 61784, the influence of the communication network as failure source of the application should not exceed 1 %. Accordingly, with an appropriate packet per year rate P_{pa} we get the PER that causes the application to fail ($\mathrm{PER_f}$) with:

$$\mathrm{PER_f} \leq 0.01 \cdot \frac{1}{\mathrm{MTTF_d} \cdot P_{\mathrm{pa}}} \tag{2.5}$$

For cascaded communication networks, we need to find the $\mathrm{PER_f}$. Therefore, we have to consider the information about the messages and framing within the considered subnetworks, as well as the entire network structure with its overall and partial communication flows.

In order to exemplify this, we assume a cascaded communication network with three levels (cf. Fig. 1.7). On each level, there is only one subnetwork. On the levels one and three we suppose a typical wired Industrial Ethernet (IE) subnetwork, and on level two a wireless subnetwork. This concept describes the easiest extension of a wired application by wireless features. As typical for all industrial communication networks, we assume that all the subnetworks use summation frames that encapsulate individual data packets for the slaves. Moreover, according to the common communication behavior, we assume discarding the entire data of these summation frames and the respective communication cycle if one data packet is invalid. Accordingly, we will use the term packet and packet error synonymously for frame and frame error in the following, as written before.

In DL direction, a data packet addressed to a slave on level three has to pass level one and two until it is received in level three. In UL direction, the transmission takes place in reverse order. The overall PER can be calculated with:

$$\begin{aligned} \mathrm{PER} = 1 - &(1 - \mathrm{PER_{L1D}}) \cdot (1 - \mathrm{PER_{L2D}}) \cdot (1 - \mathrm{PER_{L3D}}) \\ &\cdot (1 - \mathrm{PER_{L3U}}) \cdot (1 - \mathrm{PER_{L2U}}) \cdot (1 - \mathrm{PER_{L1U}}) \end{aligned} \tag{2.6}$$

For simplification, we assume the same wired networks on level one and three and equal PER values for their DL and UL transmission:

$$\mathrm{PER_{L1D}} = \mathrm{PER_{L3D}} = \mathrm{PER_{L3U}} = \mathrm{PER_{L1U}} = \mathrm{PER_{IE}} \tag{2.7}$$

Moreover, we want to analyze the requirements of the application and do not intend to evaluate the communication channels. Therefore, we can further equate the DL and UL PER of the wireless network on level two:

$$\mathrm{PER_{L2D}} = \mathrm{PER_{L2U}} = \mathrm{PER_{W}} \tag{2.8}$$

Accordingly, we get:

$$\mathrm{PER} = 1 - (1 - \mathrm{PER_{IE}})^4 \cdot (1 - \mathrm{PER_{W}})^2 \tag{2.9}$$

As introduced in Sec. 1.2, isochronous RT protocols do not retry erroneous transmissions. Instead, missing data are interpolated to a certain degree. Most applications typically allow a sequence S_k of $(k-1)$ consecutive packet failures to prevent the interpolation from already becoming too inaccurate. If we do not allow a single invalid data packet ($k = 1$), the failure probability p_f depends on the number of packets m and their error probability p:

$$p_\mathrm{f} = 1 - (1 - p)^\mathrm{m} \tag{2.10}$$

For $k > 1$, p_f has to be calculated by a concatenated sum of failure probabilities over the packet sequence length. Here, all possible sequences have to be analyzed, which represents a huge computational expense and is not appropriate for the following analyses. Alternatively, assuming $k \ll m$, we can conservatively estimate that $(1-p)\cdot m$ valid packets occur and that each might precede S_k. Accordingly, we can estimate p_f with:

$$p_\mathrm{f,k} \approx (1 - p) \cdot m \cdot p^\mathrm{k} \tag{2.11}$$

In order to get the general PER with allowed packet losses, we have to divide Eq. (2.11) by m:

$$\mathrm{PER_k} = (1 - \mathrm{PER}) \cdot \mathrm{PER}^\mathrm{k} \tag{2.12}$$

Assuming that $\mathrm{PER} \ll 1$, we can calculate the PER for an individual data packet with:

$$\mathrm{PER} = \sqrt[k]{\frac{\mathrm{PER_k}}{1 - \mathrm{PER}}} \approx \sqrt[k]{\mathrm{PER_k}} \tag{2.13}$$

Thus, we can now determine the appropriate $\mathrm{PER_{f,k}}$ that causes the application to fail with:

$$\mathrm{PER_{f,k}} = 1 - (1 - \mathrm{PER_{IE,k}})^4 \cdot (1 - \mathrm{PER_{W,k}})^2 \tag{2.14}$$

Using Eq. (2.14) in combination with Eq. (2.5), we can now derive a more reliable PER of the individual subnetworks for FA applications.

For a first comparison to the PER requirements from the use case analysis, we want to calculate the PER for an exemplary machine tool. The parameters and components

of this exemplary machine tool are extrapolated from our use case analysis and represent typical values. The considered components list can be found in appendix A.2. We assume the same cascaded network structure as described previously. Moreover, assuming 16 h of operation per business day and using Eq. (2.4), we get an overall $\mathrm{MTTF_d} = 2.1$ years and according to Eq. (2.1), $\mathrm{MTTF} = 1.05$ years. With an overall communication frequency of 1 kHz, we get $P_{\mathrm{pa}} = 3.15 \cdot 10^{10}\,\mathrm{year}^{-1}$. According to Eq. (2.5), we get $\mathrm{PER_f} \leq 3.02 \cdot 10^{-13}$.

Further, we assume that the commonly used wired networks fulfill the indicated PER requirement of FA applications. Therefore, we suppose a PER of 10^{-9} for the wired subnetworks of our application example. Accordingly, we can derive the PER requirement for the wireless subnetwork using Eq. (2.14) and Eq. (2.13). Finally, for our extrapolated machine tool example, this is $\mathrm{PER} \approx 3.88 \cdot 10^{-7}$.

With this example, we could show that PER requirement of 10^{-9} indicated by our use case analysis might not necessarily be true for all types of communication networks. Using the conservative safety standards, we could calculate an upper bound for the PER of a wireless communication network within a cascaded communication system that is notable smaller and probably more realistic.

2.4 Machine Tool Application Scenarios

For a validation of the outcome of the remaining work, we introduce an extrapolated application scenario from our use case analysis, as shown in Fig. 2.7. This represents a typical industrial application with average parameterization.

In order to especially validate the latency and reliability in a practice-oriented manner, we consider a possible drilling machine tool that requires short, reliable communication cycles. This application consists of six components. The first component is the global control. It contains the controller unit, production feed and removal, a water cooling system, and several sensors for monitoring. The overall number of slaves for this component is 20.

The second component is the workbench. It can move forward and backward and can be tilt in two axes. Therefore, we estimate three slaves.

The third component is the tool gripper. It can pick up different tools, move in all three dimensions, and tilt the tool in one axis. This would require at least five slaves.

The fourth component is a product gripper. It can take the product, lift it, move it forwards and backwards and rotate it. It is used for the feed and removal as well as to optimally position of the product. Therefore, we estimate four more slaves.

The last component is a tool holder. It keeps several tools that are required for the production on a rotating plate. Thus, the tools can be exchanged automatically. Here, only one slave is required.

Therefore, the overall number of slaves of this application is 33. A simple sensor typically transmits and receives less than 10 Bytes of date, whereas an actuator requires on average approximately 50 Bytes. Accordingly, for simplification, we assume the transmission of an average amount of 20 Bytes of command and actual data for each individual slave, every communication cycle. This also corresponds to the findings

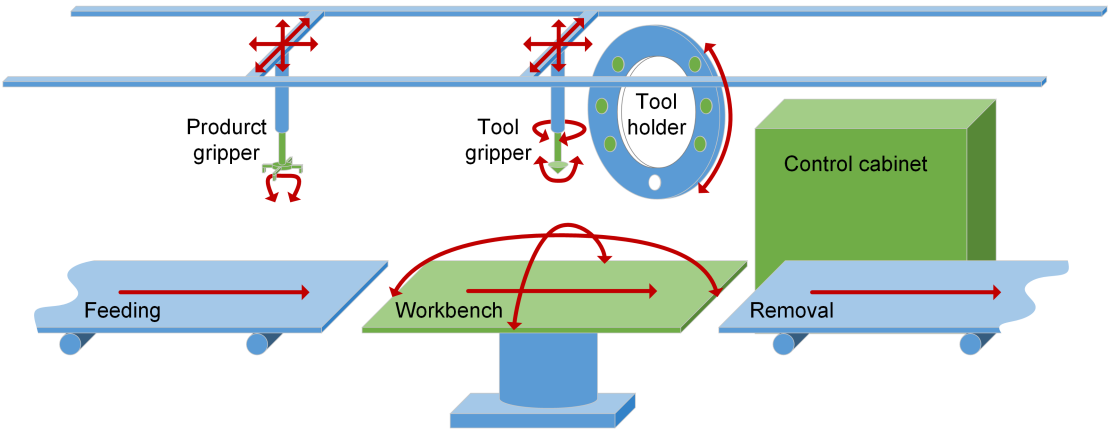

Figure 2.7: *Schematic example of a possible drilling machine tool.*

of our use case analysis. The cycle time is assumed to be 1 ms in order to enable the corresponding control algorithm.

This application scenario represents a typical industrial machine that would benefit significantly from cascaded wired and wireless communication networks. The variety of moving components is currently interconnected via slip rings or cable carriers. A replacement by wireless interconnections would decrease downtimes due to maintenance or breakdown and the costs for the expansive spare parts. Moreover, the flexibility and range of action could be increased. For example, the tool holder could probably support adjacent applications as storage for tools as well. However, this requires that the cascaded communication system still maintains the required performance profile.

2.5 Adapted Performance Indicators for Cascaded Industrial Networks

In order to validate an capability profile with the requirement profiles of industrial applications, a common list of performance indicators for industrial communication networks is required. However, as introduced in Sec. 2.1, there is no such list containing all relevant parameters. In the following, we will create such a list taking into account the results of our use case analysis. We start with the parameters indicated in the IEC 61784, as they represents a good basis (cf. Sec. 2.1).

The first performance indicator specified is the delivery time. In contrast to the time that is required for a single data transmission, industrial applications require the accomplishment of a global timing. Cycle time is an especially crucial parameter for the underlying control loop of the applications. The delivery time can then be used to derive the required number of communication cycles it takes for the master to transmit command data and the slaves to respond with the corresponding actual data.

While time synchronization and non-time-based synchronization are not directly differentiated in industry, both have a strong influence on the corresponding application.

In frame based communication networks, time synchronization is required for data reception and transmission. It determines when the master or a slave has to expect new data and can be used to detect missing data. This is especially crucial for wireless communication networks, e.g. to enable energy saving as well. Without precise synchronization, the start of a new data frame might not be detected correctly. This will result in a loss of data or the discarding of an entire frame. Also, in wireless networks the actual data of all slaves need to be transmitted according to a specific schedule. Otherwise, overlapping transmissions might interfere with each other.

Parameter non-time-based synchronization accuracy is more related to the application itself than to the communication. The evaluated synchronization ensures that all slaves activate their command data and capture their actual data exactly at a predefined time, such as the GSP. Accordingly, this parameter is very important to their cooperation. Accordingly, it is used to specify the maximum amount of time by which a communication participant may deviate from a joint schedule. However, the non-time-based synchronization accuracy also partially depends on the accuracy of the time synchronization as well. In industrial use cases jitter is often indicated as a value for the deviation from a global timing in order to specify the synchronization accuracy.

The next parameter introduced by the IEC 61784 is the possible number of terminal slaves. This parameter is also very important for industrial applications since it limits the number of slaves within an application. Usually, the number of terminal slaves is limited by the amount of data per slave and the maximum allowed frame length containing this data rather than any kind of addressability. Therefore, it would be more useful to identify the frame length and the number of frames permitted for DL and UL. The maximum number of slaves can then be derived from their actual data size requirements. Moreover, not only the number of terminal slaves should be specified, intermediate devices, such as gateways, are also required for cascaded networks. These devices might also require some signaling or management data to be transmitted every communication cycle. Accordingly, their payload also needs to be considered, since typical industrial communication networks do not provide dedicated signaling data channels.

Industrial applications typically do not define an average unidirectional throughput of required RT data. Usually, they are described by an amount of command and actual data that needs to be transmitted per communication cycle. However, if necessary, the RT throughput could be derived using the number of slaves and their corresponding amount of either control or actual data per communication cycle.

Furthermore, a required NRT data rate is rarely indicated for industrial applications. The quantity and frequency of their occurrence are very different and dependent on various circumstances. Moreover, they are transmitted with the lowest priority, and only remaining resource capacities are provided. Accordingly, a comparison of the actual required RT throughput and the overall throughput available can be used to gain an estimation of the available data rate for NRT data.

The basic network topology is also rarely provided as a requirement from any application. However, it still is of a certain importance. On the one hand side, the master needs to control all slaves directly with the lowest latency possible. On the other hand side, a direct communication from slave-to-slave or certain reliability might also be re-

quired. These considerations have a direct impact on the network topology, especially for cascaded communication systems. Here, certain combinations of different network protocols can be used to either enable or exclude specific communication relations or to enable redundancy for some connections.

The number of switches between terminal slaves is also only indirectly required. Similar to the basic network topology, it has an impact on communication relations. Switches or any intermediate devices additionally introduce a certain extra transmission latencies. Moreover, the number of these devices influences the number of subnetworks within an application and, indirectly, its spatial extent.

The last parameter defined by the IEC 61784 is the redundancy recovery time. A fast reboot after an error is important for any application. However, it is much more important to ensure an accurate operation so that no error occurs. Therefore, instead of requiring only a certain recovery time, the applications require assurance about the reliability of the communication network regarding its data delivery.

These reliability aspects are more addressed by the performance parameters introduced in the IEC 62657-2. According to this list, a communication network should be indicated by its packet loss rate (PLR). However, in addition to the loss of data packets, erroneously received data packets might also result in a loss of information. This is especially true for wireless transmission, where fading, multipath spreading, or noise sources influence the transmission performance and increase the probability of an packet error. Therefore, the identification of a communication network's PER of is more crucial for an industrial application.

In addition to these reliability aspects, the IEC 62657-2 defines availability as a performance indicator. Again, this parameter is not explicitly indicated as requirement by any application, but is still mandatory for their functionality. Using the PER and redundancy recovery time of a communication system, a relationship to its availability can be established.

The security level of a communication network is another performance parameter of growing importance these days. Industrial applications can be attacked in various ways to disrupt functionality or steal information. With the introduction of wireless and cascaded communication networks to industrial applications, the possibility of attack has increased even further. Therefore, security aspects have to be implemented immediately right from the beginning of the network planning.

The last important performance indicator we will mention here, is parallelism. Parallelism is essential for most industrial applications which have to coexist on a factory floor. Especially concerning wireless communication, neighboring systems shall not interfere each other. The IEC 62657-2 therefore specifies several parameters that have an influence on the parallelism of several communication networks. The most important ones are the used frequency channels, power spectral density and spatial distance between the slaves. A combination of several parameters can then be used in order to determine whether adjacent communication systems can run in parallel without interruption.

In Tab. 2.1 we present a list of performance indicators that enables an understanding between the requirements profile of an application and the capability profile of a communication network. Starting with the parameters revealed by our use case analysis,

we have assigned the corresponding indicators for a communication network. Because the required parameters are specified by both the application and the communication network, this comparison now enables an comprehensive validation of the respective systems.

Table 2.1: *Performance indicators to compare a requirement profile with an capability profile.*

Application's requirements profile	Network's capability profile
Cycle time	Delivery time Number of slaves RT data throughput
Synchronization accuracy	Time synchronization Non-time-based synchronization
Spatial extent	Network topology Number of interconnection devices Distance between slaves
Number of communication participants	Number of slaves RT data throughput
File size per slave and cycle	RT data throughput NRT data rate
Reliability	PER PLR Redundancy recovery time Security level Availability
Parallelism	Frequency channels Power spectral density Distance between slaves

2.6 Discussion: Universality and Usability

The requirements of FA applications are differentiated by the different application classes. Additionally, they can be distinguished by the individual products they are used for, environments they are used in, machine operators that run them, and manufacturers that build them. Therefore, it is not possible to represent every single available application here. Accordingly, our extended literature review as well as our use case analysis can only provide an overview without any claim to completeness.

Very often, within individual applications, not all slaves have the same requirements. Partially, they are used for the closed-loop control of the processing, but some are also for monitoring purposes only. As a result, a clear distinction between FA and CM is not always possible and it might be possible that a communication network can still be used for a certain application without any problems, even if it does not cover all maximum requirements. Theoretically, it might be possible to cluster individual application segments into several subnetworks with different requirement profiles. However, due to the abilities of the currently available IE networks for FA, such a segmentation is typically not designed so far, as described for the acyclic RT data in Sec. 1.2. From a communications point of view, all RT data of slaves are treated the same way. They have to be provided in every communication cycle and must be transmitted according to the corresponding reliability and timing requirements. A differentiation according to the scope that allows data losses or lower cycle times can only be made on the application level.

However, such a distinction would also be reasonable on the communication level, especially concerning wireless and cascaded communication networks. In this case, individual subnetworks could be planned according to the scope of applicative functionality. For example, a wireless communication network could be used more efficiently without negatively influencing the overall communication due to its inherent transmission characteristics. This would already allow the integration of wireless networks that do not yet meet all FA requirements. For example, 5G is intended to provide a possibility of implementing different subnetworks with individual transmission characteristics and service quality via network slicing. However, in order for an application to be consistently and completely subdivided, a corresponding wireless communication network is anyhow required that meets all requirements regarding real-time capability. Otherwise, limitations will already occur again.

Moreover, our presented use case analysis is mainly based on the experience of machine operators and manufacturers. Since they mainly operate the IE networks commonly available today, we cannot preclude that their information requirements are partly derived or influenced by these IE networks. This could be seen in particular at the required PER, which corresponds exactly to the specifications of the corresponding IE protocols for the applications.

Our bottom up analysis therefore represents a more realistic approach to deriving actual PER requirements. However, we had to assume independent and uniformly distributed packet errors. Otherwise, it would not be possible to relate the PER to the operation time of an application as defined by the different standards. Therefore, this analysis indicates a best-case assumption only. Unfortunately, bit errors and thus packet errors are not uniformly distributed in reality and therefore usually occur in bursts. Accordingly, specific models of industrial radio channels, including burst errors, must be analyzed to establish actual PER values of specific applications. So far these channel models are not available and even a completely analytical solution would probably not be possible under these circumstances. Moreover, the IEC 61784 we use to derive the 1 % upper bound of failure probability a communication should add to the entire communication actually belongs to safety concepts. These standards assume a black

communication channel that already includes error correction methods where only remaining errors shall not exceed 1 % of the overall error probability. However, since we do not consider safety aspects or a black channel here, a direct use of this limit might not be completely correct and has to be further evaluated. Nevertheless, our method represents an improved first approach compared to simply adopting the requirements of existing wired networks.

Accordingly, within this chapter, we were able to present an extensive analysis of FA applications, which can significantly improve the development of communication networks required for this class. In the following chapters, some of the requirements are repeatedly addressed in order to verify the necessity of different optimizations.

Chapter 3

Modeling Industrial Communication

Before an industrial application is deployed, it is offline designed, parameterized and tested. This requires corresponding models of all controllers, sensors, and actuators. Furthermore, the data transmission must also be parameterized and verified according to the application. For this purpose, appropriate models of the corresponding communication network are required, which completely map their characteristics.

In this work, we focus on communication networks for factory automation (FA) applications. Cascaded communication networks will become more and more interesting for these applications in the near future, especially in order to implement new "Industry 4.0" (I4.0) concepts. The concatenation of the individual subnetworks represents a new challenge for data transmission. The parameterization of one subnetwork may influence the transmission of the subsequent subnetwork. In order to map this, models are required, which also enable an evaluation of the entire communication chain in addition to individual parameterization.

Since cascaded networks have hardly been considered for the industrial sector so far, there are hardly any models that could be used to evaluate the parameterization of the individual subnetworks and verify their use for a corresponding application. Therefore, we develop an appropriate modeling of industrial real-time (RT) communication. Moreover, this model will be used to reveal optimization possibilities and verify the respective methods.

We start with a comparison of currently available network models with our requirements in Section 3.1. Subsequently, in Section 3.2, we develop a generic network model which is specially adapted to the performance analysis of industrial communication networks and we demonstrate the mapping of two network protocols using this model. In Section 3.3, we derive a method for evaluating the timing behavior of arbitrary cascaded networks based on the timing of the individual slaves. We use an evaluation of our previously introduced application scenarios to reveal sources of transmission latencies that need to be minimized in Section 3.4. Finally, we summarize our results and draw our conclusions in Section 3.5.

3.1 Usability of Existing Models

The evaluation of cascaded communication networks requires a network model that considers the entire concatenated communication. Several protocols with individual parameterization might be combined and influence each other. Moreover, the typical behavior of industrial communication networks have to be considered. This includes, among others, static RT data size, strict cycle times, and summation frames, which are rather untypical for networks from the office area. Only if these characteristics are adequately reflected by a corresponding model, we can use the model to verify the consistency of the network specifications and validate its usability for the corresponding FA applications.

The most common models for depicting communication networks are the Open Systems Interconnection (OSI) model and the more specialized IEEE model (cf. Fig. 1.5). They subdivides any communication network into at most seven respectively five different layers, each with a precisely defined set of tasks. This enables a comparison of different protocols without the detailed knowledge of the underlying internal functionalities. Network protocols of the same layer with standardized interfaces are easily interchangeable.

Therefore, the both models are of special importance for the development of cascaded communication networks since they can be used to analyze the interoperability between the individual subnetworks. Furthermore, we can use these models derive the interfaces between the individual communication layers in order to achieve a generic communication system. However, it does not allow the easy deduction of the performance parameter of the entire network across all layers and subnetworks.

Accordingly, we need a method to analyze the traffic flow within the communication system. Queueing theory is one method for doing this. It is a subdomain of probability theory and can be used to predict the stability of a communication system, the number of messages within the system, and their respective transmission times. Therefore, a network node like a slave or gateway is represented as a node with n queues and m server processing the queues, which results in a specific service time. Accordingly, we can calculate the time that a message of a certain size requires to be forwarded from the master to a specific slave if the path of the message transmission and probability of interfering messages are known.

Network calculus [83] as specific form of queueing theory has become a typical method of modeling and analyzing communication systems over the past few years. It enables the expression of constraints such as link capacity and background traffic with well-defined metrics. Thus, specific performance guarantees can be evaluated with standard methods and made comprehensive. Accordingly, we can use this approach for computing upper bounds on transmission latencies or message backlog of a certain communication network.

However, in typical wired industrial communication systems for FA applications, only the master initiates new data transmission (cf. Sec. 1.2). Moreover, the use of summation frames and strict cycle times clearly defines the number of packets transmitted per time unit. Apart from statistical transmission errors, which are manageable in

wired communication networks, both results in a deterministic communication relation between the master and its slaves. Accordingly, the probability analysis of typical Industrial Ethernet (IE) networks turns into a countable problem, at least for the respective RT data.

For wireless networks, the probability is of greater significance. Even if basic methods of IE networks are adopted and a deterministic communication relation is established, the transmission channel remains unpredictable due to external influences. So far there are no realistic channel models for an exact evaluation of industrial wireless transmissions. At the same time methods for an prediction of the wireless transmission are required to increase the reliability. Both represent large and complex fields of research on their own that are only marginally considered in this work. For our analyses of an improved integration and performance, we will therefore assume an error-free transmissions at first. Accordingly, modeling data flow in wireless networks also becomes a simplified problem.

As a result, the methods of *network calculus* and queueing theory are overdesigned for our analyses of cascaded network for FA applications. Due to their high complexity, they are not efficient for the investigation of the corresponding data flow. The same applies to other models that are designed for more complex probability analyses, such as those presented by Lange et al. [65], Greifeneder et al. [84], Song et al. [85], and Liu et al. [86]. Therefore, we develop a simpler model in the following section.

3.2 Development of a Generic Network Model

Since the investigation of typical RT communication networks for FA applications could be simplified to a countable problem, we could focus on the development of a generic network model for arbitrary communication protocols according to our performed requirements analysis. With respect to the OSI model, each communication network puts some application data into a certain frame with additional context information and adapts this frame to specific physical resources before transmitting it (cf. Fig. 2.1). Since the conversion of an individual application data to a data packet by the networks application layer (APP) and its adaptation to a specific bit stream by some mid layer is very protocol dependent, we develop our model from a physical layer (PHY) perspective. Subsequently, we will derive our model, that we first have introduced in [4] and evolved in [1] and [6].

Each communication network has only a limited resource space for information transmission. Typically, this medium is shared by several slaves. Therefore, multiplexing strategies are used to combine different signals for the transmission over this medium. There are four most general possibilities for a resource multiplexing:

- space division multiplexing (SDM),
- frequency division multiplexing (FDM) or wavelength division multiplexing (WDM),
- time division multiplexing (TDM), or

- code division multiplexing (CDM).

With SDM a multiple use of a transmission medium is achieved by a spatial distribution of the individual nodes. For a wired communication, it represents the simplest method of multiplexing. Individual wires are used for separate point-to-point connections for each transmission link between the master and its slaves. In wireless networks, this is achieved by an appropriate distribution and orientation of the transmitters and receivers. In order to influence the spatial distribution of the wireless signal propagation, directed antennas can also be used to achieve SDM.

In FDM, several distinct carrier frequencies are used for individual links. The transmitted signal then consist of the entire summarized frequency range. The terminology FDM is typically used for radio carriers, while WDM is commonly applied to optical carriers.

TDM is one of the most commonly used multiplexing approach for wired industrial communication networks. Here, time slots of individual and fixed length are assigned to individual data transmissions. These assignments repeat cyclically within the frame structure.

In CDM,a data stream is deliberately spread with a specific spreading code in the frequency domain, which results into a signal with a wider bandwidth. By using different spreading codes, several data streams can be transmitted on the same carrier frequency. The individual data can then be distinguished by demodulation with the appropriate code.

Since all basic multiplexing techniques are orthogonal to each other, they can be used together in any combination. Thus, they form the basic method for most virtualization techniques and enable the combined transmission of different streams of data with limited bandwidth. If these links originate from multiple slaves or address several of them, one talks about multiple access solutions. Multiple access requires an appropriate data link layer (DLL) respectively more precisely media access control (MAC) and logic link control (LLC) sublayer. The multiple access strategies result from extension of the respective multiplexing method and address, for example, the problem of spatially distributed slaves. Common methods are space division multiple access (SDMA), frequency division multiple access (FDMA), time division multiple access (TDMA), and code division multiple access (CDMA).

According to these methods, the limited resource space could be segmented into individual resource element (RE). The use of one multiple access strategy results in a one-dimensional segmentation. However, also different approaches could be combined in a two-dimensional segmentation. A common example is Orthogonal Frequency Division Multiple Access (OFDMA), which combines TDMA and FDMA.

With respect to the transmission of command data in downlink (DL) and actual data uplink (UL), we have to be aware of different duplexing methods. Half-duplex systems allow only one communication at a time. Accordingly, the transmission of command and actual data must take place one after the other. Full-duplex communication system on the other hand enable a two-way communication without limitations. Here, command data and actual data can be transmitted simultaneously. Basically any of the aforementioned Multiplexing techniques can be used also for duplexing. the most typical

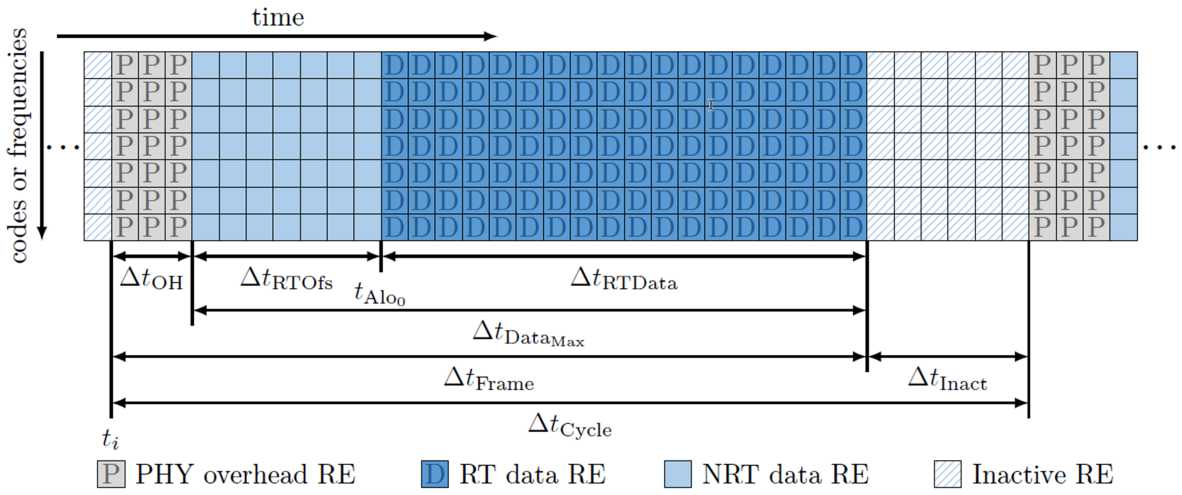

Figure 3.1: *Proposed generic frame model for cyclic industrial communication systems.*

ones are time division duplexing (TDD), and frequency division duplexing (FDD). Also code division duplexing (CDD) can be used as a duplexing method, but has not yet been implemented in practice. Therefore, the model has to provide a distinguishable DL and UL transmission and potentially multidimensional segmented resources for each direction.

Moreover, we have to differentiate according to the required data structure for the transmission within industrial networks. According to Fig. 1.4, we have to differentiate between physical layer overhead data, RT data, and non-real-time (NRT) data. Additionally, we have to model potential blank spaces between sequential frames. Therefore, we will use inactive REs. While this allows only a rough granularity in the representation of blank spaces, it allows an easy modeling and consideration of their influence. A possible divergence depends on the representation of the RE.

With our model we want to analyze the general timing behavior of a network. Therefore, we assume that no transmission errors occur. For typical wired industrial communication networks, this is achieved by sufficiently strong error correction codes, interpolation of an application dependent number of erroneous or missed data (cf. Subsec. 2.3.5), or redundancy. Although this is much more complex, we also have to make this assumption for industrial wireless communication networks or they would not fulfill the requirements of the respective application and could not be used. Such an error reduced transmission could be achieved for example by using robust modulation and coding schemes as well as diversity or redundancy techniques, such as multiple reception systems, transmission antenna systems, frequency diversity, or time diversity. However, the reliability of wireless networks is an independent research area out of the scope of this work.

Fig. 3.1 depicts our resulting generic frame model that could be used for a timing analysis of arbitrary cyclic industrial communication networks. As introduced, the individual RE are created over time and probably also by coding and multiple access strategies. Accordingly, one RE comprises a certain amount of data n_{BpRE} that can

be transmitted at a specific time interval Δt_{RE}. The RE transmission rate T_{RE} also depends on the individual network implementation.

Within each frame, a fixed number of REs is reserved for the transmission of the physical layer overhead, denoted as R_{OH}. This contains, among others, the frame preamble and delimiter, as well as address and synchronization information. This results in an overhead transmission duration of $\Delta t_{\mathrm{OH}} = R_{\mathrm{OH}}/T_{\mathrm{RE}}$.

A fixed number of sequential REs is available, denoted by R_{Data}, for the payload of the frame. These resources are assigned to the respective RT and NRT data. However, only the RT data are of interest for the timing analysis. Accordingly, we model the arbitrary allocation of the RT data within the frame by an respective offset $\Delta t_{\mathrm{RTOfs}}$, whereas the earliest possible allocation time for these data is denoted by $t_{\mathrm{Alo_0}}$. The transmission of a maximum number of REs for these RT data R_{RT} requires a time of $\Delta t_{\mathrm{RTData_{Max}}} = R_{\mathrm{RT}}/T_{\mathrm{RE}}$. The remaining spare capacities $R_{\mathrm{Data}} - R_{\mathrm{RT}}$ can then be used for the NRT data transmission, as is typical in industrial communication networks. The resulting frame duration is equal to $\Delta t_{\mathrm{Frame}} = \Delta t_{\mathrm{OH}} + \Delta t_{\mathrm{Data_{Max}}}$. Its start is denoted by t_i.

The number of sequential inactive REs and the corresponding time $\Delta t_{\mathrm{Inact}}$ must be larger than or equal to the required guard interval between successive frames defined by the protocol. The frame transmission is repeated cyclically with an cycle time $\Delta t_{\mathrm{Cycle}} \geq \Delta t_{\mathrm{Frame}} + \Delta t_{\mathrm{Inact}}$.

In order to analyze the entire communication chain of DL and resulting UL transmissions, we have to use at least two, possibly concatenated, of these models. The parameter for DL and UL can be set individually, whereas they are repeated with the same $\Delta t_{\mathrm{Cycle}}$. Also, the global sampling point (GSP) must be the same for the entire network. Each cycle it occurs exactly at t_{GSP}. Once set, the parameterization and frame structure is not changed during the operation of the network.

Subsequently, we use two specific network implementations to show how to map both wired and wireless communication networks with this model.

3.2.1 Mapping Wired Communication Networks

As introduced in Subsec. 1.3.1, FA applications require isochronous RT communication networks. Only a few network protocols are capable of fulfilling the respective requirements. Here, we present the mapping of the *Sercos* protocol specification [87]. The representation of *EtherCAT* or *PROFINET IRT* would be similar.

Sercos is built on IE with a raw data rate $T_{\mathrm{RE}} = 100\,\mathrm{Mbit/s}$. It implements a master-slave communication relation with a TDMA strategie. Here, we model one RE to contain one Byte. Therefore, the duration of one RE is $\Delta t_{\mathrm{RE}} = 80\,\mathrm{ns}$.

Individual summation frames for DL and UL are used to encapsulate the different data packets for all slaves. Thus, the overhead could be reduced and a better control of the communication flow could be achieved. The amount of overhead per frame is 26 Bytes, such that $R_{\mathrm{OH}} = 26$. Accordingly, it takes $\Delta t_{\mathrm{OH}} \approx 2.1\,\mu\mathrm{s}$ to transmit it.

As defined in the Ethernet standard 802.3 [31], one frame could comprise up to 1500 Bytes, equivalent to $R_{\mathrm{Data_{Max}}} = 1500$ of payload. Therefore, the maximum trans-

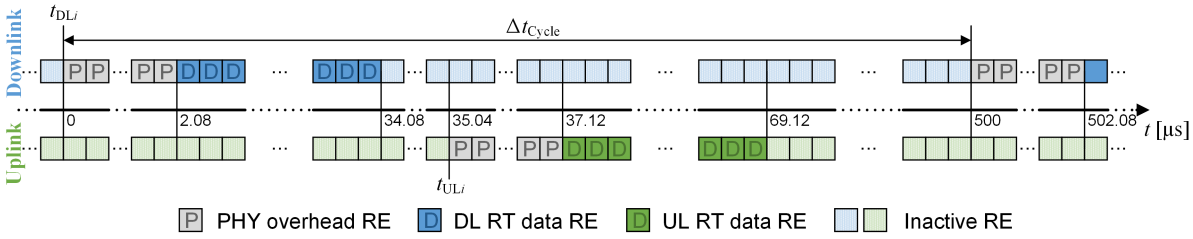

Figure 3.2: *Mapped Sercos networks with one frame for DL and UL each.*

mission duration is $\Delta t_{\mathrm{Data_{Max}}} \leq 120\,\mu s$. The protocol requires an guard interval between successive frames of at least 12 Bytes, equivalent to $\Delta t_{\mathrm{Inact}} \geq 0.96\,\mu s$.

Sercos supports standard Ethernet full duplex transmission. The two parallel channels are used to create complete connection redundancy. Per channel DL and UL frames are transmitted sequentially. Special to *Sercos* is that up to four separate frames can be transmitted per cycle for both DL and UL. Each of these frames can have its individual payload length, which is constant during runtime. The start of one frame t_0 is concatenated to the length of the previous frame and the corresponding inactive time.

The *Sercos* protocol supports a number of 511 slaves. They are typically connected with a line or ring topology. The minimum achievable cycle time is $\Delta t_{\mathrm{Cycle_{Min}}} = 32.25\,\mu s$. In this case, the maximum payload is 365.125 Bytes, specifically $R_{\mathrm{Data_{Max}}} = 365.125$. The achievable synchronization deviation is $< 1\,\mu s$.

Fig. 3.2 depicts an exemplary *Sercos* network implementation. For this example, wa assume one frame for DL and UL and 400 Bytes ($R_{\mathrm{Data}} = 400$) for each frame. The cycle time is $\Delta t_{\mathrm{Cycle}} = 0.5\,ms$. Between both frames, the minimum guard interval of $0.96\,\mu s$ is assumed. However, due to the subsequent transmission of DL and UL frame, the inactive time for both frames is $\Delta t_{\mathrm{Inact}} = 465.2\,\mu s$.

3.2.2 Mapping Wireless Communication Networks

As introduced in Subsec. 1.3.2, there are currently various ongoing activities towards the development of novel wireless systems for closed-loop control applications. Here, we will further introduce and map the current working hypothesis of the *ParSec* network, developed in the ParSec project [54].

The *ParSec* system operates in the 5.8 GHz ISM band with a bandwidth of 150 MHz. The transmission power is limited to $\leq 14\,dBm$[1]. Thus, no Listen Before Talk is required in order to prevent interferences with other systems. However, it reduces the transmission range to $\leq 20\,m$, which is sufficient for most FA applications. Specific security mechanisms were implemented in order to secure the data access and transmission. These are designed for the typically short data packets and provide a key

[1]The limited transmission power results from the conditions of use of the ISM bands regarding possible coexistence. Currently a private band for industrial applications is under discussion in which a transmission power of up to 30 dBm would be allowed.

generation based on local characteristics of the communication channel.

For the *ParSec* network, a completely new physical, media access control and application layer were developed in order to fulfill the requirements of FA applications. Therefore, a combination of a TDMA and CDMA-like approach, called Parallel Sequence Spread Spectrum (PSSS) [88,89], is used to generate a two dimensional resource space. Here, 255 orthogonal codes are used to generate 255 parallel REs with an transmission duration of $\Delta t_{\mathrm{RE}} \approx 14.3\,\mu\mathrm{s}$ each.

Moreover, the use of FDD enables a separation between an individual DL und UL frame, that could be sent in parallel. The duration of these frames $\Delta t_{\mathrm{Frame}}$ mainly depends on the quality of the wireless. If the channel properties change, it might be possible that individual REs will no longer be clearly distinguishable from each other. A channel estimate and channel correction is therefore determined for the channel coherence time in order to reduce transmission errors. Furthermore, a uniform time base among all communication participants is also necessary in order to be able to distinguish individual REs correctly. Since the distributed clocks of the individual slaves might drift over time, this requires a resynchronization in regular time intervals. These time intervals depend on the accuracy of the built-in oscillators. In *ParSec* the same time interval was selected for the synchronization as for the channel coherency. Thus, synchronizations and channel information can be transmitted collected in one preamble.

In order to maintain an channel estimate and derive the corresponding channel correction, the current implementation consequently contains a measuring of the considered wireless channels within every DL preamble. For this measurement, the network master transmits its DL overhead on not only on the corresponding DL frequency band but also the parallel UL frequency band. Accordingly, no UL data of any slave are allowed to be sent during the time required to transmit the DL overhead transmission, since it would interfere with the measurement. Therefore, the start of the UL frame is delayed until the end of the DL overhead (preamble) transmission: $t_{\mathrm{UL0}} = \Delta t_{\mathrm{DLOH}}$.

The duration of the physical layer overhead is assumed to be $\Delta t_{\mathrm{OH}} = 153\,\mu\mathrm{s}$ for each direction. With the currently assumed binary phase-shift keying (BPSK) modulation, one RE contains one bit, such that 255 bit can be transmitted parallel for DL and UL each. Therefore, the gross data rate is about 17.9 Mbit /s for each link. However, this would be further reduced, since each data packet contains additional methods of security and error correction to achieve the required reliability. In the current working hypothesis, approximately $\frac{2}{3}$ of each transmitted symbol are considered for these methods. Therefore, about 10 Bytes remain for the actual data transmission per symbol.

Fig. 3.3 shows an exemplary *ParSec* networks implementation with a cycle time $\Delta t_{\mathrm{Cycle}} = 1\,\mathrm{ms}$. Due to the overhead and a RE duration of $\Delta t_{\mathrm{RE}} \approx 14.3\,\mu\mathrm{s}$, $\approx 6.3\,\mu\mathrm{s}$ have to remain inactive at the end of each cycle. The maximum amount of data that can be transmitted per DL frame is 590 Bytes ($R_{\mathrm{DLData}} = 590$). For the UL frame, the inactive time is $\Delta t_{\mathrm{ULInact}} = 163\,\mu\mathrm{s}$ since no UL transmission is allowed while the DL overhead is being transmitted. Accordingly, a maximum amount of 480 Bytes of data can be transmitted within each UL frame ($R_{\mathrm{ULData}} = 480$).

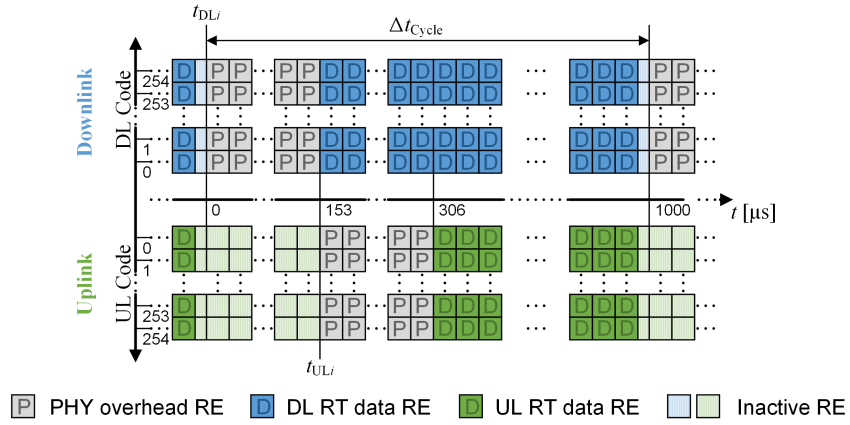

Figure 3.3: *Mapped ParSec networks with 1 ms cycle time.*

3.2.3 Timing Methods

As mentioned above, most closed-loop control applications implement a GSP within every communication cycle. The timing of this GSP is essential for the applications, since it defines the activation time of the command data and the acquisition time of the actual data. Therefore, it is also related to the positioning of the frames and the corresponding data within the frames.

Regarding an individual communication network, the timing of the GSP has to be set according to the requirements of local slaves and the application. However, in cascaded communication systems, its timing is influenced by all slaves on all levels of the cascade. In order to analyze the influence on the overall timing behavior, we subsequently introduce currently employed timing methods of closed-loop control applications and relate them to our network model.

Basically, there are four different timing methods for the overall network and the GSP with respect to the application requirements:

- command data optimized timing,
- actual data optimized timing,
- closed-loop optimized timing, and
- cycle time optimized timing.

Subsequently, we consider the timing from the point of view of the controller, since this is primarily relevant for the control itself. From the point of view of the application, the perspective could differ but does not change the four different cases.

If the application requires a command data optimized operation, then the controller has to be able to compute new command data based on the latest actual data. This enables a fast reaction on deviations of the control algorithm. Therefore, the UL data are transmitted as soon as possible following the DL data. Accordingly, the GSP has to be timed so that all actual data are already acquired and processed before they

have to be transmitted. Therefore, the individual UL processing time $\Delta t_{\mathrm{ULProc}}$, which a slave requires to prepare the acquired actual data for transmission, has to be used to schedule the GSP. Moreover, the transmission of the subsequent DL frame must be set according to the time that the master needs to compute the new command data $\Delta t_{\mathrm{DLComp}}$ following the reception of the latest actual data. Thus, this command data contains a direct reaction to the actual data. However, it cannot be guaranteed that all command data are received and processed for activation $\Delta t_{\mathrm{DLProc}}$ before the GSP when using this timing method. An exemplary command data optimized timing is schematically depicted in Fig. 3.4.

For an application that requires an actual data optimized operation, it is most important that the actual data contain a direct reaction to the latest command data. Thus, the influence of the control can be seen as quickly as possible. Therefore, its behavior is the complete opposite of the command data optimized timing. Here, the time between the DL and the subsequent UL transmission must be long enough that all individual slaves can receive and activate their DL data and provide their corresponding actual data processing for the UL transmission. The GSP is timed according to the last DL processing. Here, it cannot be guaranteed that the time between reception of the UL data and the transmission of the next subsequent DL data is long enough for the controller to compute new command data based on the current actual data. A response therefore cannot be determined before the next cycle. Fig. 3.5 schematically shows an exemplary actual data optimized timing.

Closed-loop optimized timing presents a combination of the previous operation modes. This timing is required when, within one cycle, DL data must be transmitted to the slaves that provide immediate responses to these data and when the next DL data should contain a reaction to the corresponding UL data. Therefore, as for the actual data optimized timing, the time between DL and subsequent UL frame must be large enough, such that the corresponding data for all slaves can be received and processed for activation as well as acquired and processed for transmission. The GSP is again timed according to the last DL data processing. Moreover, the time between the UL and DL frame of the next cycle must also be appropriate for the computation of the respective DL data. Accordingly, for this operation mode, all three processing times ($\Delta t_{\mathrm{DLProc}}$, $\Delta t_{\mathrm{ULProc}}$ and $\Delta t_{\mathrm{DLComp}}$) have to be considered for the timing and resource allocation in order to fulfill the requirements. However, this requires longer cycle times compared to the previous two approaches. An ideal closed-loop optimized timing is depicted in Fig. 3.6.

The cycle time optimized operation represents a counter-draft to the longer cycle times of the closed-loop optimized timing. Therefore, all frames are ideally transmitted immediately after each other. Accordingly, the cycle time is the shortest possible. However, it cannot be guaranteed that the UL data contain an immediate reaction to the latest command data or the new DL data are computed as result of the latest actual data. In this case, the transmission and processing of a data packet from master to slave and back again with the generation of a new data packet will take up to three cycles. Nevertheless, since the cycle time is much smaller than for the previous approaches, this timing method might result in shorter round trip data transmissions. It is schematically

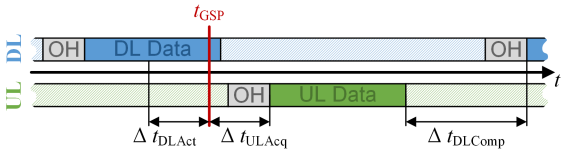

Figure 3.4: *Command data optimized network timing.*

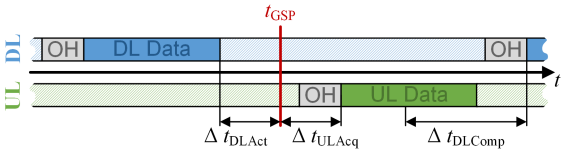

Figure 3.5: *Actual data optimized network timing.*

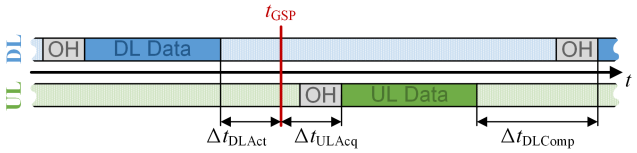

Figure 3.6: *Closed-loop optimized network timing.*

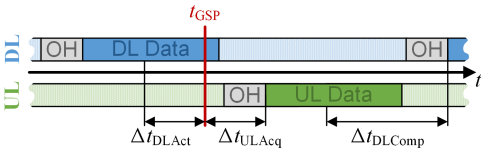

Figure 3.7: *Cycle time optimized network timing.*

represented in Fig. 3.7.

Since these timing methods have a large influence on the overall timing behavior of a communication network, in the remainder of this work we use them to derive parameter boundaries or verify their applicability.

3.3 Timing Analysis of Cascaded Networks

Using our previously introduced model, we are now able to analyze the timing behavior of arbitrary cascaded networks consisting of multiple wired and wireless protocols. Therefore, we have to combine a certain number of frames with their relative parameterization and investigate the respective data transmission, as we have shown in [6].

A cascaded network consists of a maximum number levels l_{Max} and a maximum number of subnetworks n_{Max} (cf. Fig. 1.7). It can be represented as an undirected rooted tree. Here, each vertex represents one of the individual subnetworks N_i, $i \in \{1, \ldots, n_{\mathrm{Max}}\}$, and each edge represents a connection between two subnetworks. For example, Fig. 3.8 depicts an exemplary cascaded network according to the application scenario we introduced in Sec. 2.4. Here, the number of levels is $l_{\mathrm{Max}} = 3$, whereas the overall number of subnetworks is $n_{\mathrm{Max}} = 6$.

Each subnetwork N_i is defined by an individual set of parameters according to the network model we introduced above. The start of the individual DL and UL frame transmission of N_i (t_{DL_i} and t_{UL_i}) is related to the start of the application cycle t_0. Moreover, for each edge, we have to define a computation time to convert the data for the corresponding subnetwork of the next level ($\Delta t_{\mathrm{ConvLn}}$) or the previous level ($\Delta t_{\mathrm{ConvLp}}$).

The root of the tree represents the subnetwork that includes the controller unit that is

the overall master of the communication. All slaves in the cascaded network are related to this master. However, since the other subnetworks typically implement a master-slave relation, they require their own (virtual) master. We investigate this mapping and the special interfaces as interconnections between the individual subnetworks in more detail in Chapter 4.

For simplification, we do not consider the data transmission time within one subnetwork in this work. Due to the high propagation speed of electromagnetic waves, this is hardly significant compared to other delays. Even along a copper cable, this still is $c_{\text{copper}} = 2 \cdot 10^8\,\text{m/s}$. Therefore, a transmission over the largest distance of 25 m of the considered applications would require only 125 ns.

The cascaded network contains an overall number of s_{Max} slaves. Here, only real slaves are considered. Therefore, (virtual) slaves required for the emulation of higher level subnetworks do not contribute to this value. Each S_j, $j \in \{1, \ldots, s_{\text{Max}}\}$ is actually located on a specific level L_{l_j}, whereas $l_j \leq l_{\text{Max}}$. However, the data for this slave have to pass all levels until L_{l_j}. Therefore, within each of these level L_k ($\forall k | k \in \{1, \ldots, l_j\}$) the slave S_j is assigned to a specific subnetwork N_i that provides or forwards its data. We can represent this with an adjacency list $A_j(k)$.

Within one subnetwork N_i, we assume a transmission time $\Delta t_{\text{DLTrans}_{i,j}}$ for the DL data from the (virtual) master to $S(j)$ and $\Delta t_{\text{ULTrans}_{i,j}}$ for its UL data transmission back to the (virtual) master. The start of the allocated RE for the corresponding data transmission is indicated by an offset $\Delta t_{\text{Alo}_{i,j}}$, which corresponds to the corresponding frame start. These time specification can be used to calculate the start of the data transmission of each slave for the DL ($t_{\text{DLTx}_{i,j}}$) and UL ($t_{\text{ULTx}_{i,j}}$).

Using these assignments and the parameterization of the individual subnetworks, we are now able to analyze their corresponding timing behavior. However, several different subnetworks might be combined in cascaded networks. Due to their individual parameterization, they influence each other's individual timing. Therefore, the complexity of the timing analysis of the overall cascaded network increases, requiring a concatenated investigation of the subnetworks.

One method of investigating this, is to calculate the relevant timing of the individual slaves. There are five significant times for analyzing the usability of a specific cascaded network for a specific application:

1. the DL data reception time at each slave t_{DLRx_j},
2. the DL data activation time of each slave t_{DLAct_j},
3. the UL data acquisition time of each slave t_{ULAcq_j},
4. the reception time of the UL data of each slave at the controller t_{ULRx_j}, and
5. the DL reply time of the controller to each slave t_{DLReply_j}.

In order to calculate these times for an arbitrary cascaded network, we introduce an analytical algorithm presented in Alg. 1.

Algorithm 1: Mathematical description for analyzing the time behavior of cascaded networks.

Input: Configuration of the cascaded network
Output: Relevant timing of all slaves

begin
  for $j \leftarrow 1$ **to** s_{Max} **by** $+1$ **do**
    1. Calculate DL data reception time for S_j:
    $t_{\mathrm{DLRx}_j} \leftarrow 0$
    for $k \leftarrow 1$ **to** l_j **by** $+1$ **do**
      $i \leftarrow A_j(k)$
      $t_{\mathrm{DLTx}_{i,j}} \leftarrow t_{\mathrm{DL}_i} + \Delta t_{\mathrm{DLOH}_i} + \Delta t_{\mathrm{DLRTOfs}_i} + \Delta t_{\mathrm{DLAlo}_{i,j}}$
      $t_{\mathrm{DLRx}_j} \leftarrow t_{\mathrm{DLRx}_j} + \Delta t_{\mathrm{DLData}_{i,j}} + \Delta t_{\mathrm{DLTran}_{i,j}} + \Delta t_{\mathrm{ConvLn}_i} + \left(t_{\mathrm{DLTx}_{i,j}} - t_{\mathrm{DLRx}_j}\right)_{\mathrm{mod}\,\Delta t_{\mathrm{Cycle}_i}}$
    2. Calculate DL data activation time for S_j:
    $t_{\mathrm{DLAct}_j} \leftarrow t_{\mathrm{DLRx}_j} + \Delta t_{\mathrm{DLProc}_j}$
    $t_{\mathrm{DLAct}_j} \leftarrow t_{\mathrm{DLAct}_j} + \left(t_{\mathrm{GSP}} - t_{\mathrm{DLAct}_j}\right)_{\mathrm{mod}\,\Delta t_{\mathrm{Cycle}_i}}$
    3. Calculate UL data acquisition time for S_j:
    $t_{\mathrm{ULAcq}_j} \leftarrow \begin{cases} t_{\mathrm{DLAct}_j} & \text{for actual data or closed-loop optimized timing} \\ 0 & \text{for command data or cycle time optimized timing} \end{cases}$
    $t_{\mathrm{ULAcq}_j} \leftarrow t_{\mathrm{ULAcq}_j} + \left(t_{\mathrm{GSP}} - t_{\mathrm{ULAcq}_j}\right)_{\mathrm{mod}\,\Delta t_{\mathrm{Cycle}_i}}$
    4. Calculate UL data reception time for S_j at the controller:
    $t_{\mathrm{ULRx}_j} \leftarrow t_{\mathrm{ULAcq}_j} + \Delta t_{\mathrm{ULProc}_j}$
    for $k \leftarrow l_j$ **to** 1 **by** -1 **do**
      $i \leftarrow A_j(k)$
      $t_{\mathrm{ULTx}_{i,j}} \leftarrow t_{\mathrm{UL}_i} + \Delta t_{\mathrm{ULOH}_i} + \Delta t_{\mathrm{ULRTOfs}_i} + \Delta t_{\mathrm{ULAlo}_{i,j}}$
      $t_{\mathrm{ULRx}_j} \leftarrow t_{\mathrm{ULRx}_j} + \Delta t_{\mathrm{ULData}_{i,j}} + \Delta t_{\mathrm{ULTran}_{i,j}} + \Delta t_{\mathrm{ConvLp}_i} + \left(t_{\mathrm{ULTx}_{i,j}} - t_{\mathrm{ULRx}_j}\right)_{\mathrm{mod}\,\Delta t_{\mathrm{Cycle}_i}}$
    5. Calculate DL reply time for S_j:
    $t_{\mathrm{DLReply}_j} \leftarrow \begin{cases} t_{\mathrm{ULRx}_j} + \Delta t_{\mathrm{DLComp}} & \text{for command data or closed-loop optimized timing} \\ \Delta t_{\mathrm{Cycle}_i} & \text{for actual data or cycle time optimized timing} \end{cases}$
    $t_{\mathrm{DLTx}_{i,j}} \leftarrow t_{\mathrm{DL}_i} + \Delta t_{\mathrm{DLOH}_i} + \Delta t_{\mathrm{DLRTOfs}_i} + \Delta t_{\mathrm{DLAlo}_{i,j}}$
    $t_{\mathrm{DLReply}_j} \leftarrow t_{\mathrm{DLReply}_j} + \left(t_{\mathrm{DLTx}_{i,j}} - t_{\mathrm{DLReply}_j}\right)_{\mathrm{mod}\,\Delta t_{\mathrm{Cycle}_i}}$

In Alg. 1, we sequentially calculate all significant times for the individual slaves S_j. Therefore, we start with t_{DLRx_j}. Assuming S_j is actually located on the root subnetwork on level L_1, this reception can be calculated by the sum of the corresponding start of the allocated resources, $\Delta t_{\mathrm{DLAlo}_{1,j}}$; the required time for the data transmission, $\Delta t_{\mathrm{DLData}_{1,j}}$; and the subnetwork internal transfer time from the controller to S_j $\Delta t_{\mathrm{DLTran}_{1,j}}$. However, for slaves located at a different level than L_1, the respective data have to pass one or more subnetworks, before they reach their destination. Therefore, we have to calculate t_{DLRx_j} iteratively at each level until L_{l_j} (Lines 3 to 8). On each level L_k, $\forall k < l_j$, this reception is related to the interface, that forwards the data to the associated subsequent subnetwork. Accordingly, we have to add the reception time at former interfaces to the data transmission time of the current subnetwork. Moreover, since the data can be received at any time relative to their allocated RE for subsequent transmission, we have to add this time difference as well (Line 8). In the worst case, this would be an entire communication cycle of the corresponding subnetwork ($\Delta t_{\mathrm{Cycle}_i}$).

Using the calculated reception time t_{DLRx_j} of the DL data, we can continue to calculate their earliest possible activation time t_{DLAct_j} (Line 10). However, since the actual data activation is relative to the GSP, we have to add the time difference until the next GSP after reception and processing (Line 11). Equivalently, we can also calculate the UL acquisition time t_{ULAcq_j} (Line 14). Here we have to differentiate according to the required timing method of the application (Line 13, cf. Subsec. 3.2.3). For actual data and closed-loop optimized timing, the UL data shall contain an immediate reaction on the latest command data. Therefore, it is necessary that they are not acquired before the DL data are received and activated, which corresponds to $t_{\mathrm{ULAcq}_j} \geq t_{\mathrm{DLAct}_j}$. This condition does not necessarily have to be fulfilled for the command data or cycle time optimized timing.

Subsequently, we can calculate the UL data reception time at the controller t_{ULRx_j} similarly to the DL data reception time at the slave t_{DLRx_j}. Therefore, we have to go through the different levels L_k in reverse (Line 15 to 20). Finally, we can calculate when the controller can provide new command data (Line 21 to 24). Therefore, we have to differentiate according to the applied application timing (Line 22).

The algorithm is assumed to be completely correct (cf. appendix A.3). Accordingly, we can use it to evaluate the timing behavior of arbitrary cascaded networks.

3.3.1 Machine Tool Application Scenario

In order to verify our introduced algorithm and for a better understanding, in this section we evaluate our introduced machine tool application scenario (cf. Sec. 2.4). It consists of an overall number of $s_{\mathrm{Max}} = 33$ slaves, divided into five components. Four of these components are moving machine parts and should therefore have a wireless connection. The controller is assumed to be in the static machine part. This application represents a very good example of a wireless extension of a wired machine.

We can represent this application with the cascaded network shown in Fig. 3.8. The subnetwork N_1 on L_1 is the root subnetwork containing the master and 20 slaves ($S_{1...20}$). For this subnetwork, we assume a wired *Sercos* communication. The subnetwork N_2 on

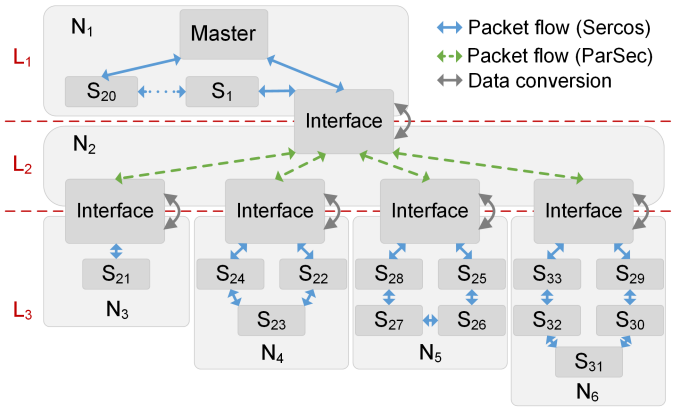

Figure 3.8: *Exemplary cascaded network for machine tool application scenario.*

level L_2 is used as extension of the wired application. Therefore, we design it as intermediate subnetwork only by omitting local slaves and connecting the moving machine components to the root subnetwork. For this subnetwork, we assume the wireless *ParSec* communication described earlier. The remaining four subnetworks $N_{3...6}$ on level L_3 represent the moving machine components. For these subnetworks, we again assume the wired *Sercos* communication.

The application shall have a cycle time $\Delta t_{\text{Cycle}} = 1\,\text{ms}$ and requires 20 Bytes RT data per slave for DL and UL each. The parameterization of the individual subnetwork $N_{1...6}$ can be found in Tab. 3.1. Their frame resources are allocated according to the order of the slaves whose data they contain. The resulting frame structure of the individual subnetworks is depicted in Fig. 3.9. The GSP is set in the middle of the communication cycle at $t_{\text{GSP}} = 500\,\mu\text{s}$. We selected this parameter in order to represent the standard parameterization for the corresponding subnetworks introduced in Subsec. 3.2.1 and Subsec. 3.2.2. As can be seen for the *Sercos* subnetworks, the UL frame is transmitted with lowest possible gap to the DL frame. Moreover, we do not assume any NRT data and RT data offset $\Delta t_{\text{RTOfs}} = 0$. Accordingly, the command or actual data are transmitted immediately after the overhead, such that $\Delta t_{\text{Frame}} = \Delta t_{\text{OH}} + \Delta t_{\text{RTData}}$. The same applies to the *ParSec* network. We set the cycle time of the individual subnetworks for the standard parameterization equal to the application cycle time. The effects of different cycle times and different frame lengths is analyzed in more detail in Chapter 5.

For the analysis, we implemented our algorithm and the network structure in *MATLAB (R2014b)*. The results can be seen in Tab. 3.2. Independent of the application operation mode, the result shows, that some slaves (the slaves on L_1) already receive their command data in the first communication cycle, as $t_{\text{DLRx}_{\text{Min}}} < 1000\,\mu\text{s}$. However, due to the forwarding of the data through the subnetworks, the slaves on L_3 receive their DL data one cycle later. For the command data and cycle time optimized timing we see that the acquisition of the actual data does not require the previous activation of the DL data ($t_{\text{ULAcq}_j} \leq t_{\text{DLAct}_j}$). The same applies for the actual data and cycle

Table 3.1: *Basic parameterization of exemplary cascaded network for machine tool application scenario.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	55.9	153.0	4.7	7.9	9.5	11.1
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]	1000.0					
t_{GSP} [µs]	500.0					

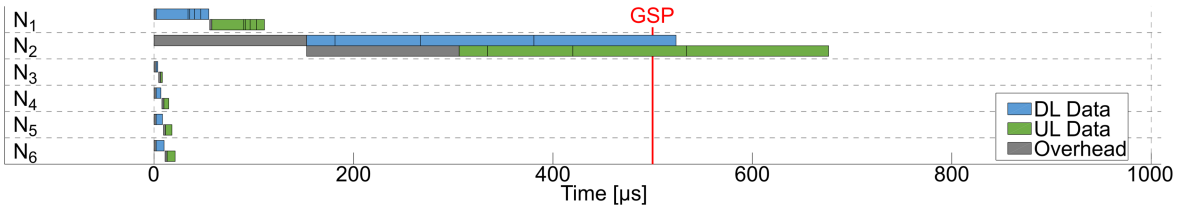

Figure 3.9: *Frame structure of exemplary cascaded network for machine tool application scenario using the parameterization of Tab. 3.1.*

Table 3.2: *Timing behavior of exemplary cascaded network for machine tool application scenario.*

		t_{DLRx_j}	t_{DLAct_j}	t_{ULAcq_j}	t_{ULRx_j}	t_{DLReply_j}
Command data optimized	Max [µs]	1010.1	1500.0	500.0	2110.8	3054.9
	Avg [µs]	407.8	894.0	500.0	1480.0	2424.2
	Min [µs]	3.7	500.0	500.0	1059.6	2003.7
Actual data optimized	Max [µs]	1010.1	1500.0	1500.0	3110.8	1054.9
	Avg [µs]	407.8	894.0	894.0	1874.0	1030.3
	Min [µs]	3.7	500.0	500.0	1059.6	1003.7
Closed-loop optimized	Max [µs]	1010.1	1500.0	1500.0	3110.8	4054.9
	Avg [µs]	407.8	894.0	894.0	1874.0	2818.1
	Min [µs]	3.7	500.0	500.0	1059.6	2003.7
Cycle time optimized	Max [µs]	1010.1	1500.0	500.0	2110.8	1054.9
	Avg [µs]	407.8	894.0	500.0	1480.0	1030.2
	Min [µs]	3.7	500.0	500.0	1059.6	1003.7

time optimized timing regarding the reception of the UL data at the controller and the transmission of new command data.

Generally, the results further show, that due to the wireless extension of the wired network, complete communication from master to slave is not possible with standard parameterization. Therefore, in the remainder of this work, we will identify additional sources of latencies and introduce methods to reduce them. Therefore, we will use the results shown in Tab. 3.2 as a benchmark.

3.4 Identification of Sources of Latencies in Cascaded Networks

As introduced in Chapter 1, cascaded communication networks will play an important role in future industrial communication. They are necessary for fulfilling the increasing requirements of FA applications since wired networks only do not provide the required flexibility and scalability, and industrial applications will not change into completely wireless systems within the next few years. However, our presented timing analysis of the exemplary cascaded network shows that typical cycle times $\leq 1\,\mathrm{ms}$ are already no longer applicable for small applications using unadapted and uncoordinated subnetworks.

Therefore, we have to implement special adjustments in order to combine the advantages of the individual subnetworks without compromising the overall performance. One of the biggest challenges here is the integration of arbitrary subnetworks into a combined cascaded communication system. The cascading causes additional latencies that delay the data transmission and make it impossible to meet the hard RT requirements of FA

applications. Accordingly, we will reveal some of the largest sources of these latencies before analyzing them in the remainder of this work.

Reviewing the application scenarios analyzed above, we can identify the interface between two subnetworks as an additional source of latencies. Each subnetwork other than the root subnetwork can only be reached by such an interface. Two-way communication between master and slave requires the corresponding interconnections to be passed twice, greatly influencing the overall timing behavior. Moreover, the interface represents the link between probably different network protocols. This requires additional conversion time within the interface and increases the latency. It is responsible for the conversion of different frames and data structures. Therefore, an optimized network interconnection is most important for cascaded communication systems.

Even with an optimized network interface, additional latencies might emerge due to different data rates of the individual subnetworks. However, the overall performance of the cascaded network cannot easily be evaluated by the subnetwork with the lowest transfer rate. Since various number of slaves could be distributed over all subnetworks and levels, the influence of the single subnetworks performance on the overall communication is difficult to predict. Not only the number of slaves, but also the way in which the different protocols and subnetworks are adapted to each other are of crucial importance here.

One possible approach, to improve the overall timing could be to adapt the cycle time of the individual subnetworks. Although the application requires a certain cyclic data exchange for closed-loop control, the individual subnetworks do not necessarily have to employ this specific cycle time or even use the same one. However, the cycle time is also dependent on a large number of factors, like the amount of data to be transmitted compared to previous or subsequent subnetworks or the corresponding frame structure. Therefore, possible adjustments and their effects have to be further investigated.

In addition to changing the cycle time, other parameters of the individual subnetworks can also be adapted for minimizing transmission latencies. However, it is not sufficient to optimize a subnetwork on its own. Therefore, a generic parameter optimization that takes the entire communication structure into account has to be found. According to the large number of possible network protocols and corresponding combination possibilities, this is a particular challenge.

Not only the parameterization, but also basic strategies of the individual protocols, such as frame structures and resource allocation, might have an influence on the overall transmission performance as well. Since this is measured among others by the transmission duration of individual data packets, it is very important where and how these data packets are transmitted within their frames. Again, a generic approach must be found that can be implemented independent of the different network protocols.

These and corresponding optimizations will be analyzed in the remainder of this work.

3.5 Discussion: Restrictions and Remarks

Within this chapter, we have addressed the importance of an adequate model for the development of cascaded communication networks for FA applications. Such a model is

required to validate existing networks, design of new network methods, reveal optimization possibilities, and verify specifications.

We have shown that several models for analyzing communication networks already exist. Here, we have introduced and reviewed among others the OSI model and network calculus. However, there was no model that is explicitly designed for the analysis of isochronous RT communication or cascaded networks. Although some could certainly be used for this purpose, it would further increase the complexity to investigate the concatenated networks we address in this work by models not intended for this purpose. Therefore, we decided to develop our own model, which can easily represent different networks with regard to their timing, in order to increase the replicability.

We have developed and presented our model which is optimized to our requirements regarding the timing analysis of cascaded networks. By representing individual parameterizable frames, which repeat cyclically, we can reflect most of the timing behavior of the typically used industrial communication networks. However, some assumptions limit the total universality of our model.

First of all, we focused on the transmission of RT data by statically assigned resources only. As introduced in Subsec. 1.3.1, it is a common assumption for isochronous RT networks, that NRT data are of limited importance and are only assigned to leftover resources. Nevertheless, there are network protocols, that implement the transmission of RT data on demand or random assignment. Such assignment could be done by reservation or by polling mechanism that might be negotiated every communication cycle. However, for industrial communication networks with mainly RT data transmission we can assume that these approaches perform less efficiently than static assignments due to the more complex allocation process and signaling or transmission collisions.

Moreover, our model mainly assumes a sequential TDMA based resource allocation, since this is the state of the art of multiple access strategy applied in most of the currently available wired communication networks for FA applications. Nevertheless, wireless networks in particular might also implement other strategies. For example, individual data packets may be assigned to individual frequencies or codes that are only active when these data are actually transmitted. This might significantly change the transmission duration and calculation of the overall frame length, since data transmission might start and end at the same time. However, this could also partially be mapped with our model by using same allocation times and transmission duration for specific data packets.

This is also an advantage of our algorithm introduced to evaluate the timing of the application. Since we analyze the data transmissions successively but independently of previous transmissions, we can also evaluate data transmission with the same start and duration. Moreover, mostly only the last transmitted or received data packet is of special importance for the application. Therefore, it is of limited interest if one or several transmissions end with the last available resource element.

Finally, in this chapter we have shown that cascaded networks can offer great benefits for industrial communication. However, this requires special adaptations of the individual subnetworks. Therefore, we have introduced some important aspects that we will further analyze. Here, we focus only on specific characteristics and do not claim for completeness. Especially for the complex network structure we consider in this work,

such a completeness would be very hard to achieve and to prove.

In the following chapter we will therefore start with an evaluation of the first mentioned optimization method: the interface between individual subnetworks.

Chapter 4

Latency Optimized Interface Between Heterogeneous Subnetworks

A network interface is the defined connection of a functional unit, like a piece of equipment or protocol layer, which can be connected to other functional units in order to exchange data. It is a complete description of the electrical, physical and semantic parameters which partly also contains functional specifications. However, in this work, we will focus solely on the influence on information transmission and consider other characteristics only marginally. A communication network consist of several interfaces, such as the interface to the application or the physical media. In this work, we will focus on the interfaces between subnetworks in a cascaded communication systems for industrial purposes.

In general, an interface must meet certain requirements and guarantee appropriate properties to enable the connection. Regarding cascaded communication networks for factory automation (FA) applications, this includes combining several different network protocols and converting the corresponding data without any information loss and with lowest possible latency. Above all, we have to enable the transition from wired to wireless subnetworks with all necessary adaptations here.

The challenge is to develop a universal interface approach that enables the connection of as many different networks protocols as possible while maintaining their individual performances. In particular, the different requirements of the various applications, as well as the large number of currently used communication protocols must be harmonized. Especially the real-time transmission mechanisms differ, sometimes considerably, which makes it difficult to find a mapping that is as uniform as possible. Therefore, within this chapter, we analyze several different options for interconnections and reveal their advantages and disadvantages regarding our specific demands.

In order to do so, we start with classification of basic interconnection devices in Section 4.1. In Section 4.2 we present a review of related gateway concepts available in the literature. Based on this review, we further analyze and compare two specific gateway concepts regarding their usability for industrial cascaded networks in Section 4.3. Finally, in Section 4.4 we summarize this chapter.

Table 4.1: *Comparison of different interconnection devices.*

OSI-Layer	Hub	Bridge	Switch	Router	Gateway
5 - 7					×
4				(×)	×
3			(×)	×	×
2		×	×		×
1	×				×

4.1 Classification of Network Interfaces

Using two or more different communication protocols within one application requires a conversion of the data in order to transmit them within those individual networks. Generally, the interconnection between two physical entities could be realized by an interconnection device that realizes the interface. These devices are so-called backbone devices. However, not all of them could be use for data conversion. Accordingly, we have to investigate the different classes in more detail. We can differentiate between five different classes of interconnection devices (cf. Tab. 4.1)[1]:

- hubs,
- bridges,
- switches,
- routers, and
- gateways.

Hubs operate typically on the physical layer of the Open Systems Interconnection (OSI) model. They enable the construction of star topologies. Incoming data is typically not evaluated, but only processed and forwarded to all other ports. Therefore, they can be used as a signal amplifier. The network data rate is shared among all connected slaves. Hubs do not allow the division into different collision domains. Accordingly, they are usually unable to avoid collisions and data losses, as long as the protocol to be interconnected does not provide corresponding mechanisms.

Bridges operate on the data link layer of the OSI model. Unlike hubs, they allow you to divide the network segments into different collision domains. This leads to a reduction of data traffic in the entire network and fewer collisions of transmissions. Therefore, they can differentiate between various links and forward data specifically.

[1]Note that repeaters are not listed separately here. For wired networks they do not provide any network functionality besides signal amplification. For wireless networks, however, they can be compared to hubs because of their functionality.

They create address tables from network nodes connected to the corresponding ports and update them continuously.

As hubs, switches typically operate on the data link layer of the OSI model. They create address table by assigning MAC addresses to the corresponding port they are connected to. They also allow a segmentation of the network into different collision domains. Therefore, switches typically provide more ports than bridges. Additionally, they can prioritize specific transmissions and offer the possibility to define the data rate for individual links. Swichtes are equipped with buffers to store data before forwarding. However, they provide several options for this forwarding. In cut-through mode, they immediately forward the data after they decode the corresponding destination address. This allows a fast data transmission. However, in this case, they do not check the data for correctness, leading to unnecessary data traffic. In store-and-forward mode, the data are first received in total and only forwarded after successful verification. Furthermore, combination modes are also possible, in which the switch changes from cut-through to store and forward after transmission errors occure. There are also switches that work on layer 3 of the OSI model. These are called multi-layer switches and forward data based on IP addresses.

Router operate on the network layer of the OSI model. It is independent of layer 2 and can therefore connect different layer 2 networks (e.g. Ethernet and Token Ring). They are used to interconnect networks based on IP addresses. They derive the most favorable way to forward data from source to destination. Therefore, they create routing table and update them continuously. These tables contain information on the next hop to the destination, error rates and transmission rate. Router allow a data encryption and often provide own firewalls. Additionally, they partially enable the filtering of data. However, they can only forward connection-oriented data transmissions.

Finally, gateways can link completely different (heterogeneous) networks with each other. They represent a common (virtual) node that belongs to both networks and handles cross-network data traffic. It performs conversion functions of, among other things, structure, addresses and physical transmissions. Therefore, the gateway consists of special hardware and software components and may operate across all layers of the OSI model. However, their complexity increases with increasing number of layers included. Moreover, increased processing also causes larger latencies.

With respect to cascaded communication networks, we need an interconnection device that enables the use of different network protocols and the conversion from one format to another one. Moreover, we need the ability to map a virtual master in subnetworks at higher levels or virtual slaves at lower levels. Only a gateway offers this flexibility at application layer. Therefore, we subsequently focus on their implementation in order to identify low latency concepts fulfilling the requirements of FA applications.

4.2 Related Gateway Concepts

As introduced before, gateways may operate on all layers of the OSI model. We can differentiate between media converter (layer 1 and 2), protocol converter (layer 3 and 4), and application converter (layer 5 - 7). Media converter are particularly important for

the transition between wired and wireless communication networks. Further important functionalities include for example conversion of logical links and port numbers.

Partially, we might also have to perform modifications on the application data. This could for example concern context information about the communication system, the composition of the data, or the mapping of virtual devices. In this case, an application converter is essential. Only this allows us to analyze and change the data content.

Since the wired Industrial Ethernet (IE) networks we consider in this work mainly operate on layer 1, 2 and 7 only to transmit real-time (RT) data, a protocol converter is of miner interest for our investigations. However, layer 3 and 4 are important for non-real-time (NRT) data transmission. Accordingly, their conversion should not be completely neglected, but is only marginally considered, since the real-time requirements must first be fulfilled.

Today, gateways are already used in some industrial applications. So-called bus couplers are exploited to interconnect various wired IE protocols. They enable the use of specific sensors or actuators that are designed only for a particular communication protocol. This increases the flexibility of an application regarding their components and expenses. The bus coupler automatically assigns specific input and output data of the preceding subnetwork to the data of the subsequent subnetwork. This requires the complete knowledge of the communication structure of the specific protocols. Therefore, bus couplers are typically not universal, and specific devices are necessary for specific network combinations. Moreover, they are designed only for wired communication networks so far. With respect to our target applications, we also need an interface to wireless networks.

A more general gateway approach is introduced in [90]. The authors analyze the vertical integration of communication networks from the production level to higher level networks of the automation pyramid. Moreover, they address the usability of a gateway in order to subdivide different subnetworks and increase their security via access limitations. Two different basic gateway implementations are introduced. Either data of one subnetwork are tunneled through the other subnetwork, or they have to be converted and restructured. Both concepts are explained in more detail later on. The vertical communication across the different levels of the automation pyramid is typically not time critical. Therefore, the different concepts are not further evaluated or compared regarding their timing. Instead, the general disadvantage of gateways due to performance degradation is introduced. As a consequence, we have to evaluate the approaches regarding their timing behavior. Additionally, we will review their universality.

A gateway for a horizontal integration as we need it is analyzed in [63]. The authors specifically investigate the connection of the wired *PROFIBUS* and *HART* protocols. Both protocols are typically used for the area of process automation (PA) and usually not suitable for isochronous RT applications. The introduced gateway concept performs the conversion on layer seven of the OSI model. Therefore, they use a dual core device, where each core runs one of the specific protocols, and a shared memory. However, since the developed concept is not intended for time critical applications, no timing evaluation is shown.

An even more detailed gateway approach also intended for FA applications is shown

in [91]. The authors analyze the extension of a wired IE by a wireless communication protocol. Therefore, the gateway requires an IE physical and data link layer, as well as a wireless physical and data link layer. These individual layers are covered by a joint application layer, because the functionalities of the layers in between are not needed. For the individual IE subnetworks the gateway has to represent either a slave or a master in order to perform the required communication tasks. Moreover, it initiates the corresponding wireless data transmissions whenever data from the wired subnetworks arrive. For the assumed point-to-point communication, this is sufficient. However, for our considered point to multi-point communication additional adaptation for the wireless transmission are required. Also with respect to parallel systems, we have to further review and evaluate this.

Since most of the currently available gateway concepts are not intended for closed-loop control application from the area of FA or do not provide any performance evaluation, we are going to review them again in the next section. The challenge here is especially represented by the numerous different communication protocols that could be integrated. Therefore, we have to consider the trade-off between universality and a possible performance degradation. Moreover, in case of a network extension, the intermediate subnetwork should ideally be completely transparent. This means that, ideally, neither the controller nor its slave will recognize their possible distribution over several subnetworks. Otherwise, their hardware and software would probably have to be adjusted. This kind of network virtualization is an important but challenging concept especially for cellular networks. It requires an abstraction of the performance parameters of the networks in order to reduce the influence of aspects resulting from the implementation. Accordingly, the abstraction creates a necessary semantic distance. Only a sufficient semantic distance will enable a mapping of abstract properties to actual network implementations and therefore the required universality.

4.3 Adapted Gateway Concepts for Isochronous Industrial Networks

Future industrial applications require the combination of several heterogeneous communication networks in order to fulfill their requirements. As introduced before, the gateway as interface between these subnetworks has a high impact on the overall performance. It has to act as smart entity controlling the communication flow and representing the underlying automation.

In order to avoid changes on the application, the communication flow ideally should remains unchanged. Neither the master nor the slaves should be aware of their distribution over different subnetworks. Therefore, the gateway has to transparently forward downlink (DL) data from a lower level subnetwork (considering the master to be on level one) to a higher level subnetwork and uplink (UL) data the other way around.

For this forwarding, the gateway has to maintain the master-slave relation typically implemented by industrial communication networks. For the lower level subnetwork it has to act as a virtual slave, receiving command data and providing actual data. Virtual

here indicates that it is not the real master or slave hardware device and therefore probably has conditional functionalities. However, this functionality depends on the implementation and can be equal to that of real slaves. For the higher level subnetwork, it has to act as virtual master in order to initiate the DL data transmission.

The conversion of the data to the corresponding subnetwork format could be done on various layers of the OSI model. Since we investigate the RT data flow considering typical industrial networks, we have to focus on the data link layer or the application layer. Accordingly, we have to differentiate between a tunnel and a proxy concept for the gateway. In [3] we have started to investigate these concepts. In the following, we will continue on examining them, presenting clear decision criteria for practical use based on a simulative comparison.

4.3.1 Tunnel-Based Gateway Concept

In the tunnel-based gateway realization, data are converted on the data link layer. The received frames of a preceding subnetwork are encapsulated into the payload of the frame structure of the subsequent subnetwork. Therefore, no knowledge about the frame structure of the preceding subnetwork is required. Moreover, the individual data does not have to be examined on the application level. However, the additional overhead of the subsequent subnetworks extends the overall overhead. Fig. 4.1 depicts a communication layer overview of this concept.

Ideally, the real frame structure of later subnetworks behind the tunneling is equal to the one of the root subnetwork. In this case, the additional frame overhead of the intermediate subnetworks could be removed. The corresponding frame could be transmitted in its original structure as generated by the controller (cf. Fig. 4.1). Therefore, the controller can communicate directly with the respective slaves and the intermediate transmissions remain invisible. However, in this approach, either no slaves are allowed in the intermediate subnetwork or they are adapted to be able to decapsulate the data on their own.

Since the corresponding data are received and encapsulated without modifications, the data flow must follow the original network topology. Therefore, the slaves in the individual subnetworks have to be addressed sequentially and not in parallel. Especially with several subnetworks on the same level, this has to be emulated by the corresponding gateways. One means to achieve this is a direct communication between gateways on the same level. This is equal to slave-to-slave communication. Otherwise, the gateway on the common lower level has to coordinate the data exchange. This results in large data traffic but better control of the traffic flow.

One method to reduce the amount of data that have to be transmitted is the clustering of information. For this optimized tunnel concept, the master has to generate several individual frames according to the number of subnetworks. In this case, only the necessary data have to be forwarded through the individual subnetworks, as depicted in Fig. 4.2. However, it requires larger data overhead and the consideration of several guard intervals compared to one large summation frame. As a result, the transmission in the root subnetwork will take longer. However, since only the actually required data have

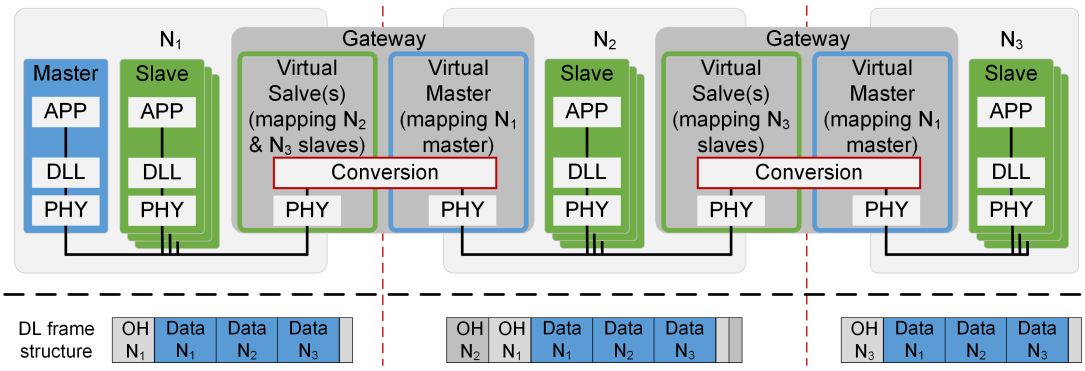

Figure 4.1: *Overview of a tunnel-based gateway with resulting frame structure.*

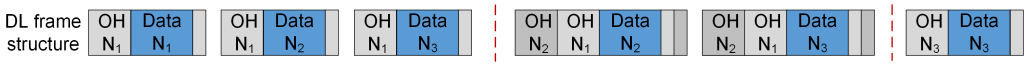

Figure 4.2: *Frame structure for optimized tunnel-based gateway corresponding to Fig. 4.1.*

to be transmitted in the subsequent subnetworks, the total transmission time might be significantly reduced. Since the application layer is not included in the conversion, the data flow has to be rebuild and remain unchanged. Nevertheless, a substantial improvement can be expected due to the lower amount of data. However, the transmission of individual frames probably requires changes in the controller configuration. This slightly reduces the universality of the optimized tunnel-based gateway concept compared to the basic tunnel concept.

4.3.2 Proxy-Based Gateway Concept

In the proxy-based gateway implementation, data are converted on application layer, as depicted in Fig. 4.3. The received frames are completely resolved, and the individual data are extracted. For the subsequent subnetwork, a completely new frame with the corresponding structure is rebuilt. The respective frame structure might differ significantly in each subnetwork. Therefore, a comprehensive knowledge of the corresponding protocols is required. However, this limits the universality of the proxy-based gateway. Furthermore, only the pertinent data need to be exchanged. Accordingly, data traffic could be reduced. Moreover, the proxy-based gateway offers a high degree of flexibility and is versatile. It enables the use of different protocols in all individual subnetworks, allowing a higher degree of flexibility for the application.

A proxy-based gateway implementation also represents an interruption of the direct communication links between master and slave due to the resolving of the frame structures. Therefore, the links have to be maintained virtually within the gateway at the application layer. This requires the emulation of behavior of subsequent subnetworks and the provision of a virtual representation of all following slaves. This virtual rep-

resentation also enable the required segmentation of the received frames. There are basically three different possibilities for the mapping, presented in Fig. 4.4:

- all real slaves are condensed into one single virtual slave with extended functionality,
- one virtual slave for all real slaves of an individual subsequent subnetwork, or
- one-to-one mapping from subsequent real slaves to virtual slaves.

In the first emulation, the master recognizes the gateway only as single slave with extended functionality and slightly different timing behavior. It can control all its functionalities by transmitting one big application data packet. These application data are subdivided and forwarded to the relevant real slaves within the corresponding application layer of the gateway. This approach reduces the overhead at the master that is required to address individual slaves. A disadvantage is that certain data, which do not belong to the corresponding merged RT data, can no longer be transmitted for each slave individually generated. For example, in *Sercos* there is one data block of fixed size for signaling data per slave that does not belong to the RT data. If several real slaves are combined in one virtual slave, they cannot receive individualized signaling data anymore, unless these data can be restored in the corresponding subnetworks.

The emulation with multiple slaves for the individual subnetworks enables more control. It represents an intermediate solution between an all-to-one or one-to-one representation. The master will recognize the gateway as a number of individual slaves, according to the number of subsequent subnetworks. The greatest added value is obtained when this representation is maintained across all levels of the cascaded network. Therefore, it can control the subnetworks individually, with the corresponding context information and the respective virtual master. However, this representation requires a more complex representation and more processing resources than with the single slave emulation.

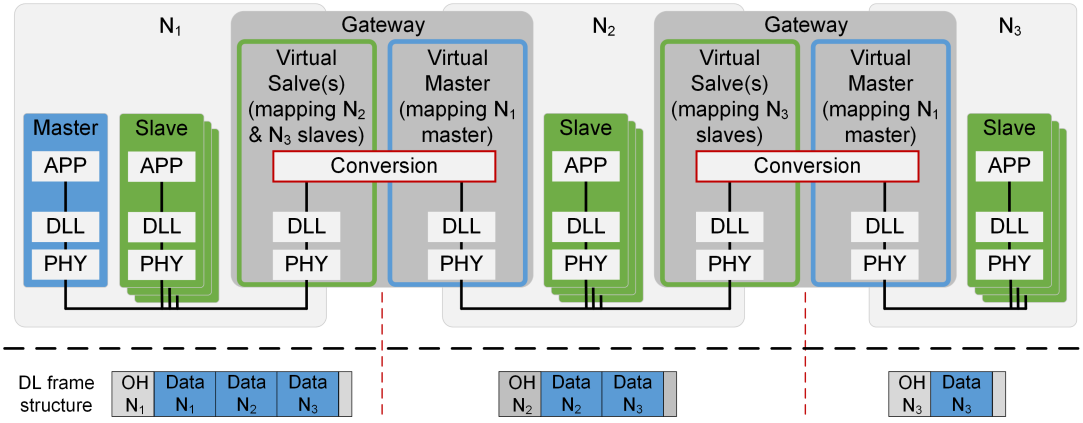

Figure 4.3: *Overview of a proxy-based gateway with resulting frame structure.*

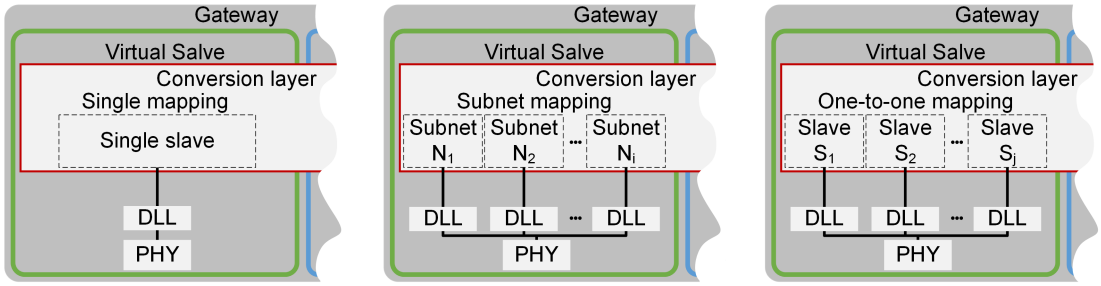

Figure 4.4: *Schematic slave representation in a proxy-based gateway with single slave mapping (left), subnetwork slave mapping (center) or one-to-one mapping (right).*

The one-to-one representation of the distributed slaves enables the best control of each real slave. The master can virtually communicate with all distributed slaves directly. Therefore, no individual information need to be summarized or discarded (referring to the *Sercos* above), since all slaves are still addressed individually. In terms of seamless integration, this approach is preferable. However, this implementation requires the highest processing performance since the gateway has to emulate the largest number of slaves in parallel.

Since the slaves are virtually represented in the preceding subnetwork, the original data flow does not need to be maintained in the subsequent subnetworks. For example, this enables a parallel transmission of data to these subnetworks. Moreover, due to the increased flexibility, the proxy-based gateway implementation could also be used to divide the application into individual subtasks. Individual slaves with specific requirements could be grouped into individual logical subnetworks (that are not related to the physical topology) with specific performances. However, this requires an adaptation of the application and is therefore not the focus of this thesis.

4.3.3 Comparison of Gateway Concepts

All presented gateway concepts have individual advantages and disadvantages. They are summarized in Tab. 4.2. A general statement which of these variants is more appropriate for our applications cannot be made. Therefore we have to differentiate on a case by case basis.

The basic tunnel-based gateway implementation is ideally used when data should be forwarded through only one intermediate subnetwork. If the preceding and subsequent subnetworks are based on the same network protocol, the additional overhead for the encapsulation is required only for the intermediate transmission. No slaves should be directly connected to the intermediate subnetwork since they would require adaptation that enables them to decapsulate the data. Due to the universality of this gateway approach, any arbitrary communication protocol could be used as intermediate subnetwork. Moreover, this gateway concept requires limited processing in the gateways themselves, since no slaves have to be emulated and the conversion is made only up to the data link layer. This allows the use of less expensive components which can be used

Table 4.2: *Comparison of different gateway concepts.*

	Basic Tunnel	Optimized Tunnel	Proxy
Universality	✓	✓	×
Required application-specific changes	✓	◯	×
Direct communication	✓	✓	◯
Required processing resources	✓	◯	×
Minimization of data traffic	×	◯	✓
Minimization of end-to-end latency	×	◯	✓

✓- good × - insufficient ◯ - partial

in larger quantities.

The optimized tunnel-based gateway concept represents an improvement of the basic tunnel implementation. Therefore, its advantage could be seen for the same network structure as previously described. Moreover, the optimized tunnel approach enables a selection of the data to be transmitted. Accordingly, it could be used reducing the amount of data that have to be forwarded through the individual subnetworks. This also reduces the necessary transmission time and enables its utilization for larger cascaded networks that also include a point-to-multipoint communication. However, the required adaptation of the controller for the transmission of several individual frames is a disadvantage of this gateway concept. It slightly reduces its universality since, ideally, the existing network structure and components should remain unchanged.

The proxy-based gateway concept is ideally used for large cascaded networks and low transmission latencies. It does not require any changes on the controller or its slaves. Due to the emulation of all subsequent subnetworks in the corresponding gateway, the entire network is represented in the root subnetwork. Therefore, the timing of subsequent subnetworks does not influence the data flow of the controller. At most, it only negatively affects the response of some individual slaves. However, the proxy approach requires exact knowledge of the frame structure of the connected subnetworks. Therefore, it could only be used for the specific conversions it is adapted to and is not universal.

For an FA application, low latencies are especially important. Therefore, we analyzed the three concepts in more detail with respect to the time required for a transmission from master to the slaves and back again. In order to do so, we assume a cascaded network with three levels. While we anticipate one subnetwork on level one and two each, the number of subnetworks on level three is variable. The subnetwork on level one contains 20 slaves. The subnetwork on level two is considered as intermediate subnetwork only and does not contain any slaves. Each subnetwork on level three is assumed to contain four slaves.

We use the same parameterization for all individual subnetworks, since we want to investigate the influence of the gateway concept on the transmission. Accordingly, we

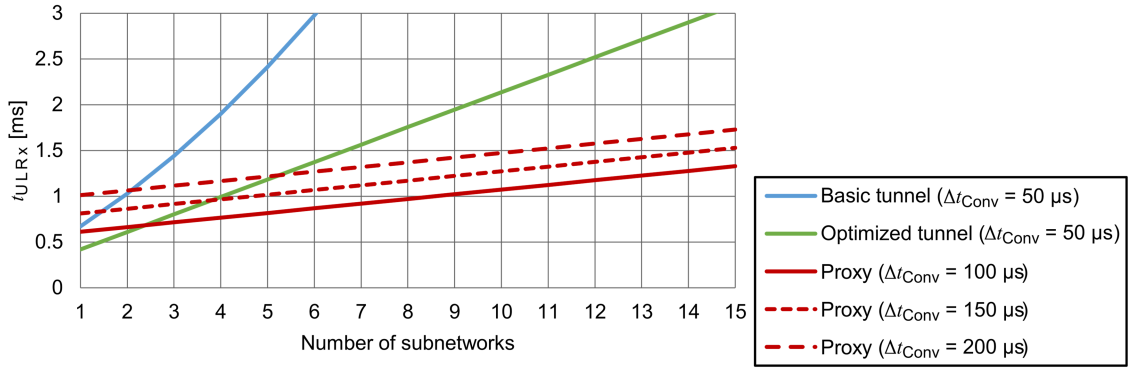

Figure 4.5: *Calculated best case uplink data reception time.*

assume a frame overhead of $R_{\mathrm{OH}} = 16\,\mathrm{Bytes}$, a guard interval equal to $R_{\mathrm{IFG}} = 12\,\mathrm{Bytes}$, and a payload of $R_{\mathrm{Data}} = 20\,\mathrm{Bytes}$ per slave for DL and UL each. We suppose a data rate of $T_{\mathrm{RE}} = 100\,\mathrm{Mbit/s}$. The exact conversion time for each gateway concept is not exactly known and strongly depends on the corresponding hardware and software implementation. Therefore, we estimate $\Delta t_{\mathrm{Conv_{Tunnel}}} = 50\,\mu\mathrm{s}$ for each the basic and optimized tunnel approach and $\Delta t_{\mathrm{Conv_{Proxy}}} = 100\,\mu\mathrm{s}$ to $200\,\mu\mathrm{s}$ for the proxy concept.

Using these parameters, we can calculate the minimum time required for the transmission from master to the corresponding slaves and back again. Therefore, we have to consider the data flow according to the analyzed gateway concept as well as the corresponding amount of data. For a variable number subnetworks on level three the resulting best case UL reception time is depicted in Fig. 4.5.

In Fig. 4.5, we show that the required transmission time for the basic tunnel-based gateway implementation increases very rapidly with higher numbers of subnetworks. Accordingly, we can confirm that this approach should ideally be used only for point-to-point data transmission and conversions. The optimized tunnel-based gateway concept results in lowest latencies for up to two subnetworks on level three of the investigated cascaded network. For more than two subnetworks on level three of the cascaded network, the proxy-based gateway with relatively short conversion times already shows the lowest latencies. The additional overhead for both the individual frames instead of the summation frame and the intermediate transmission results in a worse timing behavior of the optimized tunnel concept compared to the proxy approach. However, the difference to the optimized tunnel and the proxy realization is relatively small for up to two subnetworks on level three. Therefore, it is questionable whether the increased effort of adapting the controller in order to transmit several individual frames is really reasonable. Even for a very conservative estimation of the conversion time of the proxy-based gateway implementation it enables lower transmission latencies for more than five subnetworks on level three. Since we anticipate the conversion time lower than four times the conversion time of the tunnel based approach, its advantage could be seen for even lower numbers of subnetworks. However, the hardware and software costs are significantly higher for this implementation.

4.4 Discussion: Performance Evaluation

Within this chapter, we investigated the importance of an adequate interface between individual subnetworks for the development of cascaded networks. Such an interface enables the data exchange across existing network boundaries and has a strong impact on data flow and timing behavior of cascaded networks.

We revised five different interface classes according to their abilities. This review was intended to clarify the terms and functionalities of the different technologies. We have shown that the corresponding metrics are partly overlapping and cannot always clearly be distinguished from each other. However, for the connection of heterogeneous subnetwork we have to focus on a gateway technology. This is the only one allowing the connection of arbitrary communication protocols and an influence on the application layer to map master or slave characteristics.

Accordingly, we reviewed existing gateway concepts for their use in cascaded communication networks in the area of FA. Some promising gateway technologies have already been introduced in the literature so far. However, they are mostly designed for specific networks and not intended for universality. Moreover, there are no detailed performance evaluations or comparisons with other concepts. In particular, their corresponding timing received little investigation. Therefore, we presented a more detailed analysis of the different concepts regarding their usability for industrial cascaded networks.

We derived the advantages and disadvantages of a tunnel-based and a proxy-based gateway concept. The tunnel concept is more universal but requires larger data traffic. This will cause additional latencies, especially for larger cascaded networks since the data flow has to be maintained. Therefore, we presented an optimized tunnel concept that reduces the amount of data to be transmitted and the corresponding latencies. However, this approach requires adaptations of the controller that reduces the universality. In contrast, a proxy-based gateway does not provide any universality at all. It has to be adapted to specific network protocols in order to convert their frame structure. However, this approach reduces the necessary data throughput to a minimum. As a consequence, we can achieve small latencies even for large cascaded networks.

Beside the use of a proxy based gateway, there might be other concepts to reduce the data traffic and achieve low latencies, that have not be discussed so far. For example, it might be possible to compress data or reduce redundant transmissions. Especially in industrial automation, several information might be transmitted repeatedly. Instead of transmitting this data completely every cycle, it might be possible to signal only its repetition. In Chapter 8 we further investigate such a concept and discuss advantages and disadvantages. However, it requires adaptation of the application and increases the probability of errors. Therefore, it is not in focus of our gateway evaluation.

For the proxy based gateway concept, we investigated different mapping possibilities of real slaves to virtual slaves. We did not investigate the mapping of the master, since we assumed that the virtual master must map the real controller almost completely to the individual subnetworks. However, the real controller could also be virtualized to a greater extent and its functionality could be more effectively distributed to the gateways. In this case, the individual subnetworks could be more individualized and controlled

independently. The real controller would then only have a superior control functionality and would enable communication between the individual virtual masters. However, this would also require extensive changes to the application and would also have a significant impact on the vulnerability to errors and the timing behavior. Therefore, we have not further examined this concept here.

In order to better evaluate of the temporal behavior of the different gateway concepts we have presented, we calculated the required transmission time for the master to the slaves and back again. Therefore, we used anticipated conversion times and an exemplary cascaded network with a variable number of subnetworks. This calculation confirms that the tunnel-based gateway concepts are useful only for a small amount of subnetworks in the cascaded network. Therefore, we have to accept the higher level of implementation efforts in order to achieve the low latencies, especially for the RT requirements of FA applications.

So far we could only show an analytical evaluation. Therefore, we made certain assumptions about the conversion time and implementation effort. Moreover, we considered an optimistic data flow without any stall or waiting times and an immediate response from the slaves. This is sufficient for our intended basic analysis of the individual gateway concepts. However, it does not allow a complete validation. In order to do so, we would have needed to do a real implementation the different approaches to fully assess the corresponding effort as well as their actual performance. Unfortunately, this was not yet possible due to the required time and resource effort. Such a detailed analysis would be especially beneficial for cascaded networks with a small number of subnetworks. However, one of the biggest advantages of cascaded networks results from the segmentation of a network into several individual subnetworks. Therefore, we also focused on the evaluation of additional latency minimization methods, as shown in the remainder of this work.

In order to overcome the assumption of the optimized data flow, we will analyze the influence of the different frame structures and cycle times of the individual subnetworks on the data forwarding and the overall timing in the following chapter.

Chapter 5

Frame Conversion Schemes for Cascaded Networks

In the previous section, we evaluated the different interface concepts based on an optimistic data flow. We assumed an immediate data conversion and forwarding without any stall or waiting times. This helped to avoid other possible timing related influences while assessing the different gateway approaches. However, it does not necessarily represent a realistic data flow.

Data forwarding in industrial communication networks is related to the corresponding resource allocation as introduced in Sec. 3.2. These resources correspond to a specific frame structure. Accordingly, data can only be transmitted at specific time-slots and not arbitrarily. In a cascaded communication system, these time-slots are defined by the parameterization of the individual subnetworks.

Each subnetwork in a cascaded communication system could have its own, independent parameterization. This results in individual frame structures, cycle times, and transmission durations. Accordingly, the parameterization strongly influences the corresponding data forwarding. In this chapter, we evaluate this influence by analyzing individual frame conversion schemes.

We start with an overview of related frame conversion methods implemented or introduced in the literature so far in Section 5.1. Subsequently, in Section 5.2 we introduce details about possible frame conversion schemes and their different subcategories. In Section 5.3 we compare these different approaches with respect to the transmission latency and jitter they introduce and evaluate them for their usability in cascaded communication networks. Finally, in Section 5.4 we summarize this chapter.

5.1 Related Frame Conversion Approaches

As introduced in Sec. 1.5, there is only limited research focusing on cascaded communication networks for factory automation (FA) applications. They are often based on standard network technologies and not intended for a latency minimization. The cascaded networks for example introduced in [67–70] use a best-effort data forwarding from one subnetwork to the subsequent subnetwork. Therefore, they assume regularly transmitted frames and independent cyclic transmission within each subnetwork. This results in additional stall or waiting times that have to be considered and reduced for the use in FA applications.

A more detailed analysis of the data forwarding in heterogeneous networks is presented in [65]. The authors investigate a combination of *CAN* [92] and *FlexRay* [93] for automotive communication networks. Therefore, they introduce a queueing based reference model to calculate the worst case response time of data transmission from master to its slaves. Additionally, they investigate the schedulability of data forwarding in individual subnetworks. Although this analysis also takes individual frame structures into account, it still only considers regular frames and best-effort data forwarding. Moreover, the assumed queueing of data that cannot be forwarded is unsuitable for transmission in FA applications since these data will typically be discarded rather than stalled.

A comparison of the data forwarding for synchronous and asynchronous network nodes that could be extended for individual subnetworks is introduced in [94]. It addresses the problem of reducing jitter by introducing synchronization mechanisms. However, since these synchronization mechanisms introduce additional latencies, a balancing between jitter reduction and higher latencies has to be found. Therefore, the authors compare three scenarios with different numbers of synchronized and not synchronized networks nodes. Their evaluation shows that the average transmission latency is about three times lower for synchronized network nodes. In addition, the jitter of the data transmission is significantly increased for asynchronous nodes. Their scenario with only a few asynchronous nodes performs worse with respect to latency and jitter as compared to the scenario with only asynchronous nodes. A comparable result could be expected when extending this analysis to cascaded communication networks. However, the corresponding assessment would be more complex since the individual subnetworks differ not only in the respective cycle time but also in the individual frame structures.

Therefore, we give a detailed overview of possible frame conversion schemes with different synchronization and frame methods in the remainder of this chapter. Moreover, we provide an evaluation of their performance with respect to the additional latency and jitter of the data transmission introduced by the frame conversion.

5.2 Introduction of Different Frame Conversion Schemes

Cascaded communication networks may consist of a high quantity of different subnetworks. Each individual subnetwork might have its own independent parameterization. According to the implemented protocol, the resources for the data transmission within the subnetwork can be allocated arbitrarily. The resulting transmission patterns could

vary greatly and influence the overall transmission latency and the jitter of the end-to-end data reception. Especially for the transition and respective frame conversion between different subnetworks, additional stall or waiting times may emerge. These influence the entire subsequent data transmission and can mutually reinforce each other. As a result, not every combination of subnetworks can reasonably fulfill the real-time (RT) requirement of FA applications.

For a more detailed analysis, as we have presented it in [4], we investigate two consecutive subnetworks N_h and N_i and the corresponding subsequent data transmission within N_i. The result can be extrapolated for all transitions within the cascaded network. Depending on the resource allocation in N_i, six basic relation to the data transmission in N_h are possible:

- Static resource allocation in N_i:
  - synchronized frame transmission,
  - asynchronous frame transmission with shorter frames,
  - improved asynchronous frame transmission with shorter frames, and
  - asynchronous frame transmission with larger frames.
- Stream based resource allocation in N_i:
  - stream based transmission with limited preemption, and
  - stream based transmission with full preemption.

We will review their corresponding transmission patterns and the consequences for the data forwarding in the following. We start deriving the transmission latency and jitter in the data forwarding for each scheme and continue with an analytical comparison and a simulative evaluation.

5.2.1 Static Resource Allocation

The static resource allocation requires a static frame structure for every communication cycle. According to this frame structure, one or several frames are required to transmit the corresponding amount of data. Within these the resource elements are statically assigned to specific data and cannot be used otherwise.

Generally, data have to be forwarded in correct order. If a data packet is received later than the allocated resource elements are transmitted, this and all subsequent packets will be stalled until the next cyclic repetition of their resources. This can cause reserved resource elements to remain idle or to be used for the transmission of non-real-time (NRT) data.

In order to identify such unused resources or identify the affiliation of data, we have to have to assume a RT data signaling for the transmission in a subsequent subnetwork. We assume this signaling to be transmitted in addition to the payload for the individual slaves in each cycle. It is independent of the signaling for network control (which is usually transmitted as acyclic RT data as introduced in Sec. 1.2). In typical wired Industrial Ethernet (IE) systems, such signaling is usually not necessary, since the

affiliation of data and resources is defined offline and does not change during runtime. However, in cascaded communication systems, the allocation can vary depending on the frame length and resource allocation schemes of the individual subnetworks. Therefore, we assume an additional signaling overhead D_{S} per slave that is transmitted in addition to the RT data and frame overhead.

In case of a static resource allocation, such additional RT data signaling could be very small. The assignment could be negotiated during the network initialization and distributed from master to its slaves on the management plane. During runtime, the resource allocations do not change and the slaves only need to observe the corresponding resource elements within the frame.

Within a subnetwork N_i, the number of resource elements n_{RE} that have to be transmitted to forward the corresponding RT data D_{RT} to a certain number of slaves n_{Slave} can be calculated with:

$$n_{\mathrm{RE}_i} = \left\lceil \frac{D_{\mathrm{RT}_i} + n_{\mathrm{Slaves}_i} \cdot D_{\mathrm{S}_i}}{n_{\mathrm{BpRE}_i}} \right\rceil \tag{5.1}$$

With an available number of RT data resource elements R_{RT} per frame, the required number of frames n_{Frames} for the data transmission can be calculated with:

$$n_{\mathrm{Frame}_i} = \left\lceil \frac{n_{\mathrm{RE}_i}}{R_{\mathrm{RT}_i}} \right\rceil \tag{5.2}$$

For simplification, we assume that no NRT data have to be forwarded. Therefore, the transmission time for the corresponding data resource elements is $\Delta t_{\mathrm{Data}_i} = \frac{n_{\mathrm{RE}_i}}{T_{\mathrm{RE}_i}}$, where T_{RE_i} is the resource element transmission rate of N_i. The transmission time for all the data and overhead in N_i must be shorter than the cycle time of the previous subnetwork N_h:

$$\Delta t_{\mathrm{Cycle}_h} \geq \Delta t_{\mathrm{Data}_i} + n_{\mathrm{Frame}_i}(\Delta t_{\mathrm{OH}_i} + \Delta t_{\mathrm{Inact}_i}) - \Delta t_{\mathrm{Inact}_i} \tag{5.3}$$

Otherwise new data will arrive before all previously received data can be forwarded unless we change the application and improve the forwarding (which is not considered here). As a result, data congestion occurs, and they can never be forwarded completely. Such a combination of subnetworks would be unsuitable for application.

We can further subdivide the static resource allocation corresponding to the synchronization of the frame transmission of N_i to the cycle time of N_h. Accordingly, we can differentiate between a synchronous and an asynchronous frame transmission.

Synchronized frame transmission: In case of a static resource allocation with synchronized frames, the frame transmission in N_i starts cyclically and exactly at the same time relative to the start of a cycle in N_h. This requires that either the parameterization of N_i is adapted to the cycle time of N_h or vice versa. It follows: $n_{\mathrm{Frame}_i}(\Delta t_{\mathrm{Frame}_i} + \Delta t_{\mathrm{Inact}_i}) = \Delta t_{\mathrm{Cycle}_h}$. Using such a static and synchronized resource allocation, there will never be a displacement between data arrival and the transmission of corresponding resource elements. Therefore, there will be no jitter in data transmission in N_i caused by the frame conversion. The worst case transmission latency $\Delta t_{\mathrm{Tx}_i}^{\mathrm{Sync}}$ within N_i using a synchronized resource allocation can be calculated with:

$$\Delta t_{\mathrm{Tx}_i}^{\mathrm{Sync}} = \Delta t_{\mathrm{Data}_i} + (n_{\mathrm{Frame}_i} - 1)(\Delta t_{\mathrm{OH}_i} + \Delta t_{\mathrm{Inact}_i}) + \Delta t_{\mathrm{Cycle}_h} \tag{5.4}$$

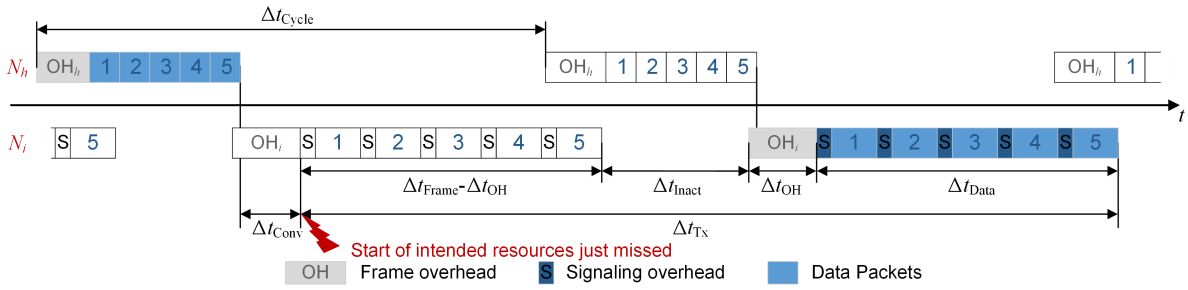

Figure 5.1: *Schematic representation of the worst case transmission latency caused by the frame conversion for synchronized frame transmission.*

Fig. 5.1 schematically depicts the worst case transmission latency caused by the frame conversion for the static synchronized resource allocation. For this example, data packets have to be forwarded from subnetwork N_h to five individual slaves in subnetwork N_i. Therefore, the transmission of each data packet in N_i requires additional signaling overhead. For this example, we assume a lower data rate of N_i as compared to N_h. This is especially reasonable for the considered wireless subnetworks since they require stronger error correction, improved security mechanism, etc.

The transmission latency of received data depends on the offset of the corresponding frames to their reception and conversion. Assuming a sufficiently good synchronization, this offset will be constant for every communication cycle. In worst case, the converted data will be ready for forwarding right after the transmission of their intended resource elements has already begun. Accordingly, their transmission is delayed until the next subsequent intended resource elements. In this case, the latency equals almost the cycle time of the preceding subnetwork N_h. However, due to the static synchronization it could be significantly reduced by an optimized offset adaptation such that the intended resources are scheduled exactly when the data are ready. We will further analyze such an optimization as additional approach for minimizing the transmission latency in Chapter 6.

Asynchronous frame transmission with shorter frames: For subnetworks, where frame duration and cycle time cannot be adapted to each other, we will get an asynchronous frame conversion. Therefore, the time between the data reception and the transmission of the corresponding frames varies during run time. This results in a certain jitter of the data transmission and a higher number of unused resources compared to the synchronous static frame conversion.

According to the relation of the frame length in N_i and the cycle time of N_h we can further specify the conversion. Fig. 5.2 depicts the worst case transmission latency for shorter frames. In this example, $n_{\text{Frames}} = 2$ frames are required to transmit the data packets and the corresponding signaling overhead. Due to the static allocation, the structure of all frames is predefined and their order has to be maintained. As a result, in worst case all n_{Frames} frames might remain unused for the transmission of RT data if they are received later than transmission of the first allocated frame.

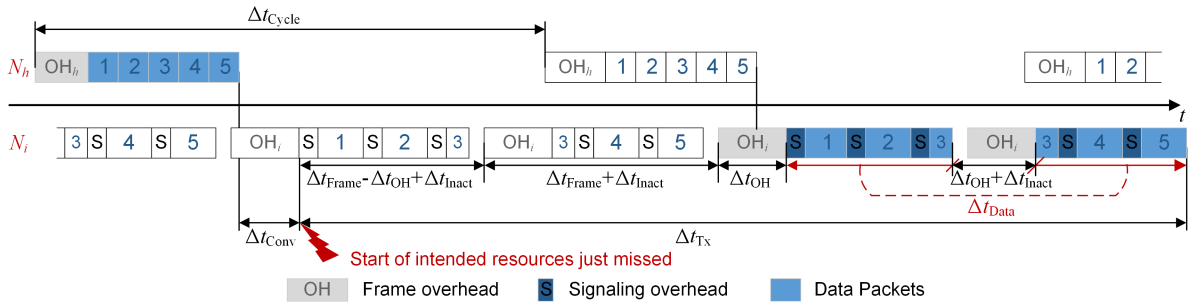

Figure 5.2: *Schematic representation of the worst case transmission latency caused by the frame conversion for an static asynchronous frame transmission with shorter frames.*

The corresponding worst case transmission latency that result from the frame conversion for a static asynchronous resource allocation with shorter frames than the cycle time of N_h can be calculated with:

$$\Delta t_{\mathrm{Tx}_i}^{\mathrm{Async,Short}} = \Delta t_{\mathrm{Data}_i} + n_{\mathrm{Frame}_i}(\Delta t_{\mathrm{Frame}_i} + \Delta t_{\mathrm{OH}_i} + 2 \cdot \Delta t_{\mathrm{Inact}_i}) - \Delta t_{\mathrm{OH}_i} - \Delta t_{\mathrm{Inact}_i} \quad (5.5)$$

The corresponding maximum jitter of the data transmission equals the difference between worst case and best case latency:

$$t_{\mathrm{TxJitter}_i}^{\mathrm{Async,Short}} = n_{\mathrm{Frame}_i}(\Delta t_{\mathrm{Frame}_i} + \Delta t_{\mathrm{Inact}_i}) - \Delta t_{\mathrm{Inact}_i} \quad (5.6)$$

Asynchronous frame transmission with improved shorter frames: With previously described frame conversion scheme, all of the required frames cannot be used for RT data transmission if the corresponding data are not available in time. An improvement of the resulting worst case latency can be achieved when at least some of the n_{Frames} frames remain available. One possibility to do this is to mark the frame whose deadline could not be met as empty and to repeat the allocation in the following frame. Thus only one frame is missed for the RT data transmission and the jitter in the data transmission could be reduced. However, since this improvement softens the static frame allocation, the signaling overhead would be higher compared to the previously introduced frame conversion.

Fig. 5.3 depicts the worst case transmission latency caused by the improved asynchronous frame transmission with shorter frames. frame conversion for the improved static asynchronous resource allocation with short frames. Compared to Fig. 5.2, only one of two required frames remains unused. After transmission of the frame that is marked unused, the same frame structure is repeated with the corresponding data that has arrived in the meantime.

The worst case transmission latency for this improved frame conversion approach can by calculated with:

$$\Delta t_{\mathrm{Tx}_i}^{\mathrm{Async,Imp}} = \Delta t_{\mathrm{Data}_i} + n_{\mathrm{Frame}_i}(\Delta t_{\mathrm{OH}_i} + \Delta t_{\mathrm{Inact}_i}) + \Delta t_{\mathrm{Frame}_i} - \Delta t_{\mathrm{OH}_i} \quad (5.7)$$

The corresponding maximum jitter of the data transmission can be calculated with:

$$t_{\mathrm{TxJitter}_i}^{\mathrm{Async,Imp}} = \Delta t_{\mathrm{Frame}_i} + \Delta t_{\mathrm{Inact}_i} \quad (5.8)$$

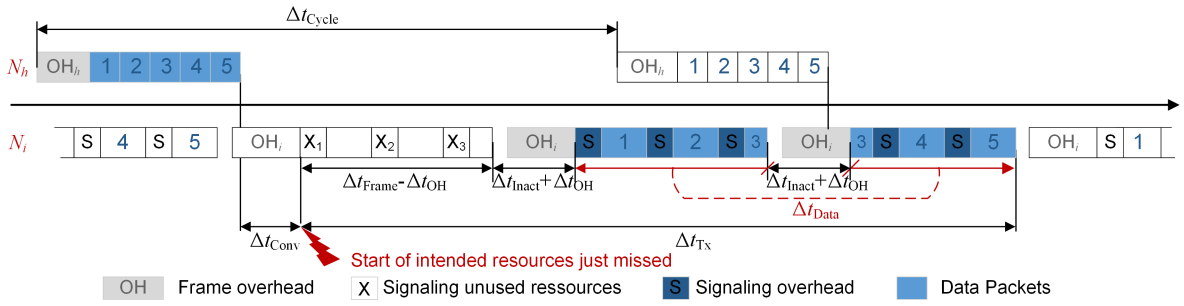

Figure 5.3: *Schematic representation of the worst case transmission latency caused by the frame conversion for an improved asynchronous frame transmission with shorter frames.*

Asynchronous frame transmission with larger frames: Instead of having shorter frames in N_i compared to the cycle time of N_h, we could also have a subnetwork transmitting larger frames that are statically allocated. Accordingly, the number of required frames to forward the RT data is $n_{\text{Frame}_i} = \frac{n_{\text{RE}_i}}{R_{\text{RT}_i}} < 1$. This frame conversion scheme comes with the advantage of less frame header to be transmitted. A disadvantage is that if the synchronization or channel estimation depend on this header, they must be maintained according to the longer frame duration. Moreover, as long as all data to be forwarded do not fit into the larger frames more than once ($0.5 < n_{\text{Frame}_i} < 1$), the individual frames vary in the resource allocation. This would increase the signaling effort and thus reduce the advantage of fewer frame headers to be transmitted. Therefore, in the following, we assume for this conversion scheme only frame sizes that allows the data to be fully allocated at least twice ($n_{\text{Frame}_i} \leq 0.5$) in order to take advantage out of this approach. This also reduces the worst case transmission latency and forwarding jitter, since frame header (Δt_{OH_i}) and inactive time ($\Delta t_{\text{Inact}_i}$) have to be considered at most once.

The worst case data forwarding for this frame conversion approach is depicted in Fig. 5.4. Since no changes in the frame structure have to be announced, we assume a signaling overhead comparable to the previous static asynchronous resource allocation. The resulting worst case transmission latency can be calculated with:

$$\Delta t_{\text{Tx}_i}^{\text{Async,Large}} = 2 \cdot \Delta t_{\text{Data}_i} + \Delta t_{\text{OH}_i} + \Delta t_{\text{Inact}_i} \tag{5.9}$$

The corresponding maximum jitter of the data transmission can be calculated with:

$$\Delta t_{\text{TxJitter}_i}^{\text{Async,Large}} = \Delta t_{\text{Data}_i} + \Delta t_{\text{OH}_i} + \Delta t_{\text{Inact}_i} \tag{5.10}$$

5.2.2 Stream Based Resource Allocation

Besides the static resource allocation approach, we could have subnetworks that use a streaming based resource allocation. Here, individual resource elements are not statically

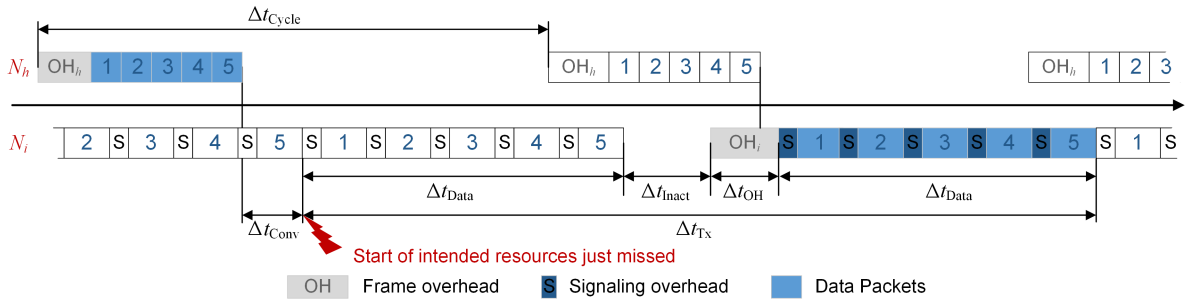

Figure 5.4: *Schematic representation of the worst case transmission latency caused by the frame conversion for a static asynchronous resource allocation with large frames.*

allocated but can be assigned as required and available. For this approach, we do not necessarily have to distinguish between individual frames but can consider the resource space as a continuous stream of resource elements without inactive times.

In order to maintain the correct reception of individual resource elements during run time, it is required that all nodes operate synchronously. Most probably a synchronization on physical layer will be implemented. Therefore, a preamble has to be transmitted at regular time intervals. Together with additional context information, the preamble represents an overhead comparable to the frame overhead for individual frames. It must be transmitted at sufficient intervals to maintain stream synchronization. These intervals shall not be delayed by data transmission.

The necessary number of resource elements n_{RE_i} that is required to forward the RT data could be calculated equally to Eq. (5.1):

$$n_{\text{RE}_i} = \left\lceil \frac{D_{\text{RT}_i} + n_{\text{Slaves}_i} \cdot D_{\text{S}_i}}{n_{\text{BpRE}_i}} \right\rceil \tag{5.11}$$

The transmission time for the corresponding data resource elements is $\Delta t_{\text{Data}_i} = \frac{n_{\text{RE}_i}}{T_{\text{RE}_i}}$, where T_{RE_i} is the resource element transmission rate of N_i. Assuming the synchronization could be maintained over a duration of Δt_{Sync_i} and a preamble duration of Δt_{OH_i}, we can calculate the corresponding number of preambles as follows:

$$n_{\text{Pmbl}_i} = \frac{\Delta t_{\text{Data}_i}}{\Delta t_{\text{Sync}_i} - \Delta t_{\text{OH}_i}} \tag{5.12}$$

In order to prevent data from piling, the transmission time for all data and overhead in N_i must be shorter than the cycle time of the previous subnetwork N_h:

$$\Delta t_{\text{Cycle}_h} \geq \Delta t_{\text{Data}_i} + n_{\text{Pmbl}_i} \cdot \Delta t_{\text{OH}_i} \tag{5.13}$$

As introduced before, we can differentiate between a stream based transmission with limited and full preemption. Here, preemption refers to whether the transmission of a single data packet can be split arbitrarily, for example if its transmission interferes with a scheduled preamble transmission.

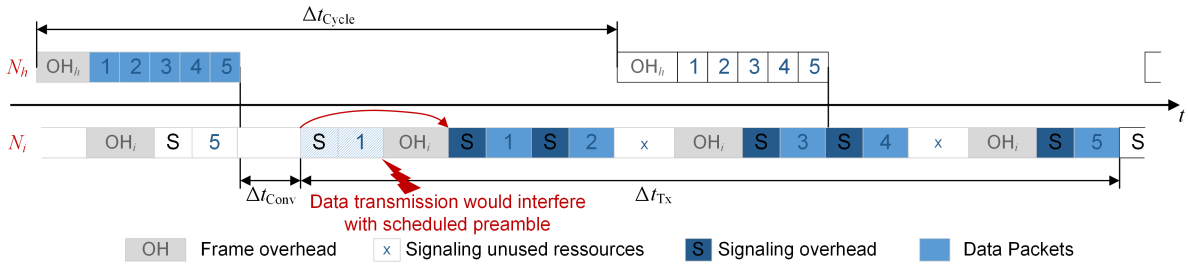

Figure 5.5: *Schematic representation of an exemplary transmission latency caused by the frame conversion for an streaming based resource allocation with limited preemption.*

Stream based transmission with limited preemption: In case of a stream based transmission with limited preemption, we assume that a single data packet can only be transmitted as a whole in subsequent resource elements. Accordingly, its transmission cannot be split to transmit other information in between. This might be the case, for example, if individual communication participants can only evaluate entire data blocks. Furthermore, a higher robustness can be assumed if the temporal distribution of a transmission is as small as possible. As a result, if the transmission of a data packet and its signaling overhead interferes with the scheduled transmission of a preamble, we would postponed the transmission, until the preamble is send. The corresponding resource elements that are available before the preamble will remain unused for the transmission of RT data to maintain their order.

Fig. 5.5 depicts an exemplary transmission latency caused by frame conversion for the streaming based resource allocation with limited preemption. However, the shift of one data packet might influence resource allocation of subsequent data packets as well. These probably would have to be stalled and shifted too. Such a chain reaction might cause Eq. (5.13) to be violated, since the required time for data transmission Δt_{Data_i} could be significantly increased.

Accordingly, a calculation of worst case transmission latency and corresponding jitter is not possible in a general manner. The data forwarding has to be analyzed at any possible position within the stream to verify the schedulability and determine the worst case performance. Moreover, we assume that a signaling overhead for such an approach might significantly be increased. Due to this substantial disadvantages, we will not further consider the frame conversion with a continuous stream of resource elements and limited preemption in the remainder of this chapter.

Stream based transmission with full preemption: The previous description has shown that limited preemption would cause several resource elements to remain unused for transmitting RT data. This can only be avoided if individual data packages can be split arbitrarily to resource elements as required. In this case, no chain reactions will occur due to data shifts as long as Eq. (5.13) is fulfilled. Accordingly, the stream based transmission with full preemption allows us to use the full capacity of the stream to forward corresponding data.

Neglecting resource elements required for preamble transmission, the data are allo-

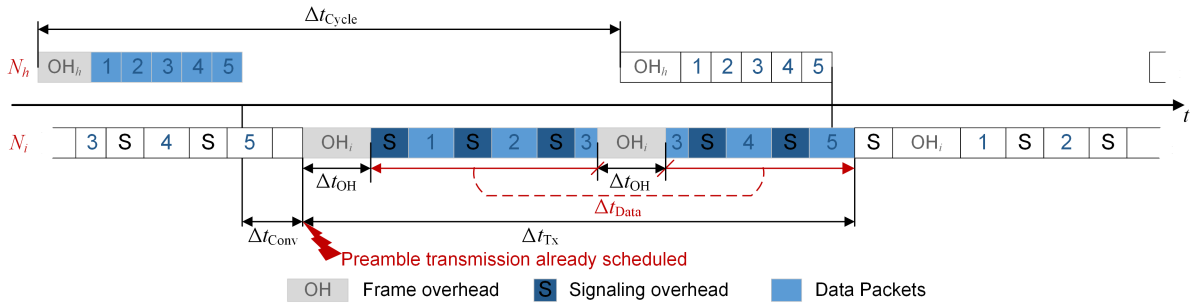

Figure 5.6: *Schematic representation of the worst case transmission latency caused by the frame conversion for an streaming based resource allocation with full preemption.*

cated at regular distances within the stream since they statically arrive at equidistant intervals. Therefore, data positioning within the stream could also partially be predefined, which reduces the individual signaling overhead. Nevertheless, as we cannot yet quantify the exact complexity of signaling for a stream based data forwarding, we assume that the signaling overhead for this approach would be higher than for the static resource allocation. Especially if the cycle time of the incoming data t_{Cycle_h} is not an integer multiple of the resource element transmission rate T_{RE_i}, we have to renew the resource allocation of individual data packets or the entire resource plan from time to time. This must be signaled to the slaves during run time.

Fig. 5.6 depicts the worst case transmission latency caused by frame conversion for the streaming based resource allocation with full preemption. The resulting worst case transmission latency can be calculated with:

$$\Delta t_{\text{Tx}_i}^{\text{Stream}} = \Delta t_{\text{Data}_i} + n_{\text{Frame}_i} \cdot \Delta t_{\text{OH}_i} \tag{5.14}$$

The corresponding maximum jitter of the data transmission can be calculated with:

$$\Delta t_{\text{TxJitter}_i}^{\text{Stream}} = \Delta t_{\text{OH}_i} \tag{5.15}$$

After introducing the corresponding transmission patterns of the five relevant forwarding schemes, we will analyze and compare their resulting transmission latencies in the following section. We especially investigate their usability for cascaded communication networks for FA applications.

5.3 Comparison of Different Frame Conversion Schemes

In the previous section, we introduced static and stream based frame conversion schemes with their corresponding subcategories. Within this section, we want to verify their performance regarding transmission latency in more detail, as presented in [4] and [1]. Therefore, we perform an analytical analysis which shows the differences of the methods more clearly. Additionally, we will evaluate their usability for cascaded networks of FA applications by implementing a simulation based verification.

Table 5.1: *Parameters for analytical comparison based on ParSec.*

	Synchronous	Asynchronous			Stream	
		Short	Improved	Large		
D_{S}	1 Byte	2 Bytes	3 Bytes	2 Bytes	10 Bytes	20 Bytes
$\Delta t_{\mathrm{Cycle}_i}$	1 ms	0.6 ms	0.6 ms	2.4 ms	0.6 ms	0.6 ms
$\Delta t_{\mathrm{Inact}_i}$	6.3 µs	5.3 µs	5.3 µs	9.8 µs	5.3 µs	5.3 µs

5.3.1 Analytical Comparison

In order to provide a more detailed comparison of the different frame conversion schemes, we calculate the corresponding transmission latencies using an suitable application scenario. We assume data conversion from a subnetwork N_h to a subnetwork N_i. The subnetwork N_h provides new data every $\Delta t_{\mathrm{Cycle}_h} = 2\,\mathrm{ms}$. After conversion, the data have to be forwarded to the corresponding (virtual) slaves within N_i.

In order to evaluate the influence of the required signaling overhead for frame conversion schemes, we assume that each slave is addressed individually. The corresponding signaling overhead D_{S_i} for the allocation of resources to the slaves depends on the respective resource allocation scheme. The exact number of bytes required has not been evaluated yet and will probably vary with the corresponding implementation. Therefore, we conservatively estimate values for D_{S_i} in order to represent its basic influence.

We calculate worst case transmission latency for a variable number of 1 to 100 slaves. The amount of data per slave is 20 Bytes. Initially, we assume a wireless *ParSec* network for N_i. For the calculation of the worst case latency we use the configuration of the current working hypothesis introduced in Subsec. 3.2.2. The different frame conversion schemes result from different cycle times and different frame repetition rates of N_h compared to N_i. Here we use exemplary values to represent the individual schemes. The variable parameters are summarized in Tab. 5.1. The calculated worst case transmission latency is shown in Fig. 5.7.

For the given payload, the synchronous frame conversion and resource allocation scheme results in the highest worst case transmission latency for up to 44 slaves. Since in this case we assume that the conversion is only completed after the transfer of the corresponding resources has already begun, the forwarding must be delayed here by one complete cycle (cf. Fig. 5.1). However, due to the static relation between data reception form N_h and forwarding in N_i that applies for this frame conversion concept, a corresponding offset adaptation could drastically reduce the latency.

For the asynchronous frame transmission with shorter frames in N_i we could see a discontinuous increase of the forwarding latency for 45 and 90 slave, since from there on two respectively three frames are required to forward the corresponding payload. The discontinuous increase is largest for this approach compared to the others, as we assume

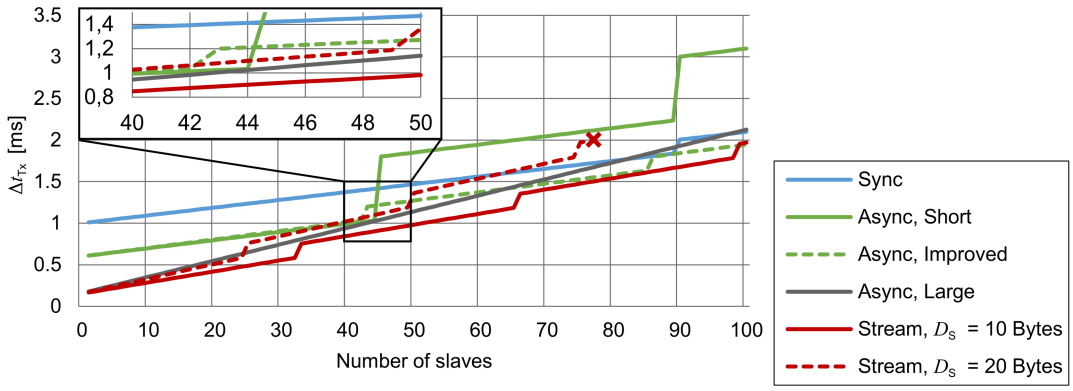

Figure 5.7: *Calculated worst case transmission latency for a ParSec network*

the entire number of required frames to remain unused if the data are not available in time for transmission. Accordingly, for more than 44 slave the overall highest forwarding latency could be seen for this approach. Moreover, the forwarding latency increases faster compared to the synchronous frame conversion and resource allocation approach, since we assume a higher signaling overhead.

For the asynchronous frame transmission with improved shorter frames, the forwarding latency for less than 45 slaves is slightly higher compared to the unimproved approach, since we assume an even higher signaling overhead. Accordingly, a second frame is already required for more than 42 and a third for more than 85 slaves. However, the discontinuous increase for this approach is much lower, since not the entire number of frames remains unused. Therefore, the latency for the improved resource allocation with asynchronous shorter frames remains lower if more than one frame is required in the unimproved approach.

For asynchronous frame conversion with static resource allocation and large frames, only one frame is required to forward the data over the total number of slaves considered. Accordingly, the transmission latency for this approach does not show any discontinuities. However, the forwarding latency increases proportional to the amount of data to be forwarded, since in worst case the transmission is delayed by the duration of the initially allocated resources and therefore takes twice the time. As a result, the increase of the latency according to the number of slaves is strongest in this approach. Never the less, for up to 63 slave, the forwarding latency for this approach is lower than for asynchronous frame transmission with improved shorter frames, since no entire frames including frame overhead have to be awaited and since we assume less signaling overhead.

Stream based transmission with full preemption, we investigated two scenarios: with a signaling overhead of 10 Bytes and 20 Bytes. As introduced before, we cannot yet quantify the exact complexity of signaling for a stream based data forwarding. Therefore, we assumed particularly high values that are intended to represent the influence of an increased payload. In the case of lower signaling overheads, the latency would be reduced accordingly. Despite the significantly higher signaling overhead, both scenarios result in lower forwarding latency compared to the other forwarding approaches for less than 25

Table 5.2: *Parameters for analytical comparison based on Sercos.*

	Synchronous	Asynchronous		Stream	
		Short	Improved	$D_S = 10$ Bytes	$D_S = 20$ Bytes
D_S	1 Byte	2 Bytes	3 Bytes	10 Bytes	20 Bytes
$\Delta t_{\text{Cycle}_i}$	0.1 ms	0.08 ms	0.08 ms	0.08 ms	0.08 ms

slaves. For 25 slaves and more, a second preamble transmission has to be considered for $D_S = 20$ Bytes and a third for more than 49 slaves. For $D_S = 10$ Bytes these limits are 32 respectively 65 slaves. The increased amount of data due to signaling overhead results in a faster increase of the forwarding latency for the streaming based frame conversion. Accordingly, with $D_S = 20$ Bytes only up to 77 slaves can be supported, since otherwise Eq. (5.13) is violated and new data would accumulate. However, for $D_S = 10$ Bytes the transmission latency for up to 100 slaves is always lower compared to the other forwarding approaches.

For a further evaluation, we adopt a wired *Sercos*-like network for N_i in a second example. Basically, *Sercos* is intended for statically allocated resources and constant frame sizes. Therefore, we only assume its basic parameterization introduced in Subsec. 3.2.1 and map the corresponding resource allocation to it. We assume the same signaling overhead for the individual frame conversion approaches as for the previous example. However, in order to be able to visualize any differences due to a variable payload, we assume extremely small cycle times of $\Delta t_{\text{Cycle}_i} = 0.1$ ms respectively 0.08 ms corresponding to the forwarding approach for this example. Otherwise, even the discontinuities resulting from additionally required frames would not have been visible due to the high data rate. Tab. 5.2 summarizes the considered variable parameter, still assuming that new data arrive every 2 ms form N_h.

We assume that there are always four frames transmitted per cycle. The resources of these frames are allocated according to the individual frame conversion approaches. Due to the static maximal frame size of the IE frames, it is not possible to adopt asynchronous frame conversion with longer frames in N_i compared to the cycle time of N_h.

The calculated results for the *Sercos* example are shown in Fig. 5.8. We can see a similar behavior of the different frame conversion approaches. The synchronous frame conversion still shows the largest transmission latency in the worst case. This approach requires the cycle time of N_i to be an integer divisor of the cycle time of N_h. Due to the high data rate of this example, the resulting transmission latency remains higher compared to the other approaches, even for a large amount of data. On the other hand, streaming frame conversion still results in the lowest transmission latency, with only a signaling overhead of $D_S = 10$ Bytes. Moreover, the high data rate renders the small increase of transmission latency through increased data transmission as almost negligible However, for streaming frame conversion and $D_S = 20$ Bytes we can observe that a

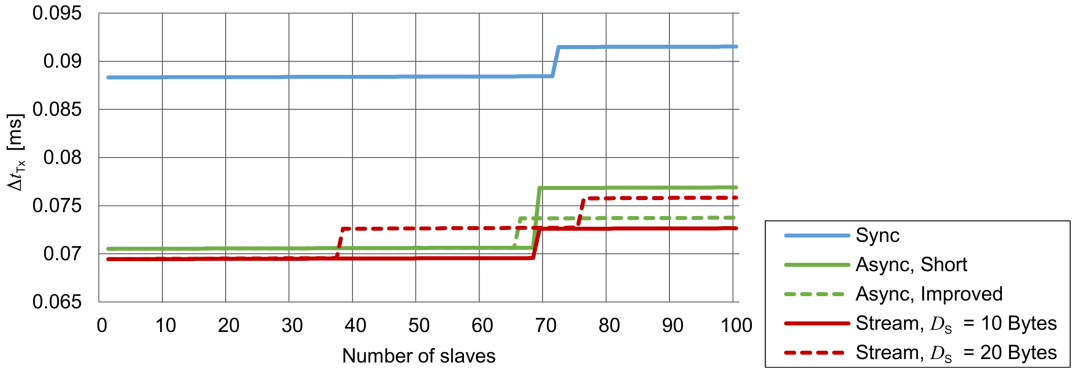

Figure 5.8: *Calculated worst case transmission latency for a Sercos network.*

second frame is required much earlier for data forwarding. Therefore, the latency for more than 38 slaves is larger compared to the static frame conversion with asynchronous resource allocation.

According to the two examples presented, the lowest worst case transmission latency that emerges due to the frame conversion could be achieved using a subnetwork that allows a streaming based resource allocation. However, for this approach we have to consider limited payload due to the probably increased signaling overhead. Also, the effort required to achieve the signaling of the allocated resources would be higher. In this regard, a static allocation would be preferable.

Moreover, for FA applications we also have to consider the resulting jitter in data forwarding. Especially for cascaded networks, this jitter could negatively influence the transmission in subsequent subnetworks. Accordingly, we have to reduce the forwarding jitter as well or the latency might increase exponentially.

In order to further analyze the forwarding jitter and to verify our previously discussed results, we subsequently present a simulative evaluation.

5.3.2 Simulative Evaluation

A further evaluation of the different frame conversion schemes according to their use within a cascaded network requires a more detailed analysis. Therefore, we implemented a cascaded network in OMNEST[1]. Again, we use the protocols *Sercos* and *ParSec* to create a cascaded communication network with three levels.

The root subnetwork on level one is based on *Sercos*. It contains the master, two slaves and a gateway, represented as virtual slave. This gateway enables a connection to the *ParSec* subnetwork on level two, where it represents the master virtually. Again, we consider this subnetwork an intermediate subnetwork. Therefore, it does not contain any real slaves. Instead, it contains a variable number of gateways, where each gateway is again connected to a *Sercos* subnetworks on level three. Each of these subnetworks

[1]OMNEST is a C++ based simulation framework that enables the analysis of computer networks and communication protocols. It provides a generic framework for modeling arbitrary network communication.

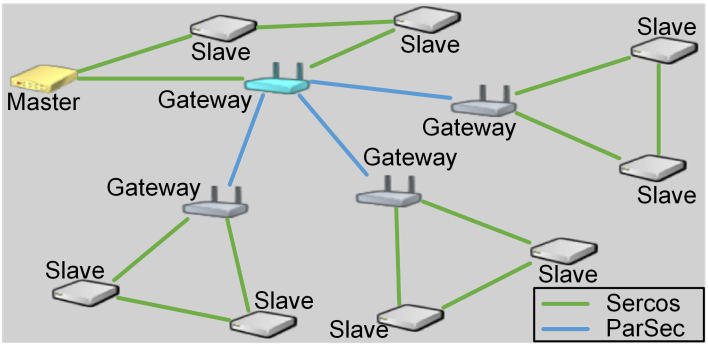

Figure 5.9: *Modeled cascaded communication network in the OMNEST simulation environment.*

again contains two slaves. Fig. 5.9 shows the network model for three subnetworks on level three.

For evaluation of the latency introduced by frame conversion, we first capture the time, when data arrive at the gateway between levels one and two. After frame conversion and transmission, we capture the time when the corresponding last data packet arrives at the gateway between levels two and three. When we subtract the constant transmission time within the *ParSec* network, we get the latency of the data transmission introduced by frame conversion from *Sercos* to *ParSec*.

The master provides new data every 1.0 ms. Due to static resource allocation and constant frame size of the *Sercos* protocol, these data arrive in a constant time interval at the gateway. Related to industrial applications, we assume a data packet size of 32 Bytes per slave within each communication cycle for both downlink (DL), and uplink (UL) transmission. According to the *ParSec* parameters presented in Subsec. 3.2.2, we can forward data for up to twelve slaves through this subnetwork. Accordingly, we simulate a varying number of one to six subnetworks on level three, where each subnetwork receives a data packet of 64 Bytes plus the corresponding signaling overhead.

We run the *ParSec* subnetwork with both static and streaming based resource allocation, with four different frame lengths: 0.7 ms, 0.8 ms, 0.9 ms and 1.0 ms[2]. Lower frame durations are not reasonable due to the required frame overhead. Accordingly, a *ParSec* cycle is either shorter or equal to the cycle time of the data providing *Sercos* subnetwork. Therefore, we do not simulate asynchronous frame conversion with larger frames. Moreover, we only need one frame to forward data to the subnetworks on level three. Thus, we do not distinguish between improved and unimproved asynchronous frame transmission with shorter frames. Independent of the resource allocation, the frame length of 1.0 ms always results in a synchronous frame conversion due to the static relation between data arrival and frame transmission. We capture the transmission latency

[2]In order to provide a sufficiently large variance in the displacement of data reception and corresponding frame transmission, we have to simulate values of 0.701 ms, 0.801 ms and 0.901 ms. Otherwise we will measure only a limited number of different transmission latencies according to the smallest integer multiple of the frame duration and the cycle time.

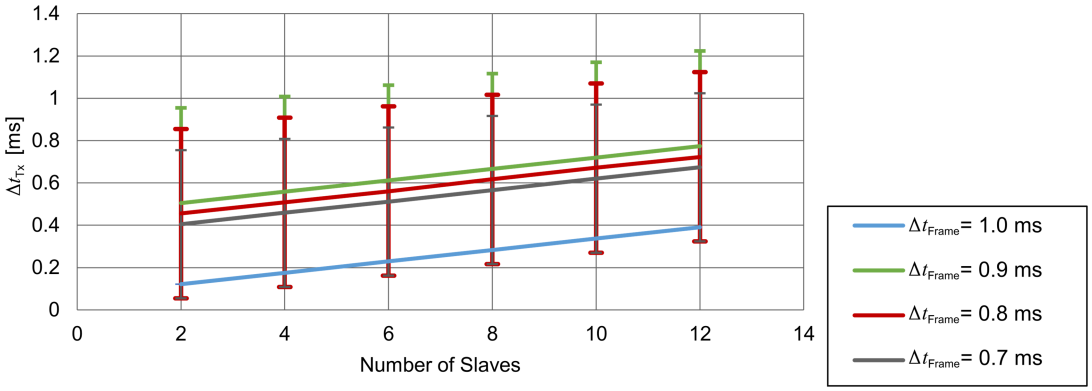

Figure 5.10: *Simulated average transmission latency with error indicators for frame conversion with static resource allocation.*

for a runtime of 1.0 s, which equals to 1000 frame conversion.

Fig. 5.10 depicts the average transmission latency resulting from frame conversion for a static resource allocation. Here, the synchronous approach with a frame duration of $\Delta t_{\text{Frame}} = 1.0\,\text{ms}$ shows the lowest average transmission latency. As expected, this frame conversion scheme does not produce any additional jitter in data transmission. In this case, the latency mainly depends on the offset between the data arrival and the transmission of the corresponding frame. The asynchronous frame conversion approaches show higher average transmission latencies. The worst case transmission latency increases according to our simulated frame duration, since this is one of the main influence factors for the worst case assumption. Therefore, larger frames also cause an increased average transmission latency. However, the resulting best case transmission latency is equal for all approaches, independent of frame duration. In this case, arriving data have been directly forwarded after their conversion. This also indicates the best achievable transmission latency for the synchronous frame conversion with an optimized offset adaption.

As presented, the average transmission latency and the corresponding forwarding jitter for an asynchronous frame conversion with a static resource allocation increases with frame duration. Therefore, having a subnetwork with such a resource allocation scheme, the frame duration should be reduced. However, this also limits the amount of data that can be transmitted. Moreover, with shorter frames, the ratio of usable resources to frame overhead deteriorates, reducing the effective data rate.

Fig. 5.11 presents simulation results for frame conversion with streaming based resource allocation. The resulting average transmission latency differs little for the four different frame durations. Since we assume the same signaling overhead for all of them, even the synchronous frame conversion is not easily distinguishable. However, for a real implementation, the signaling overhead for a synchronous frame conversion would be lower. Therefore, also the increase in transmission latency with an increasing amount of data to be transmitted would be lower. Moreover, this approach again does not show any jitter in the data transmission. The forwarding jitter for the streaming based

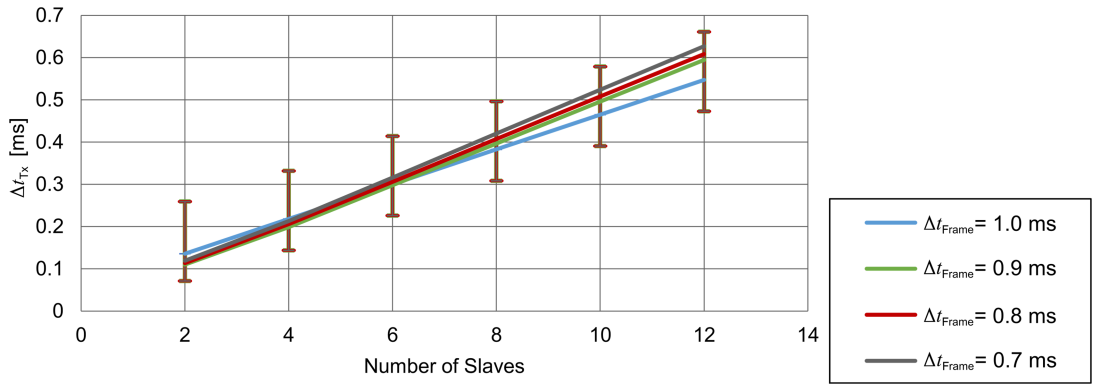

Figure 5.11: *Simulated average transmission latency with error indicators for frame conversion with streaming resource allocation.*

frame conversion approaches is equal, independent of the frame duration and amount of data. It only depends on the time required to transmit the frame overhead, preamble and inactive time. However, with an increased amount of data to be transmitted, the average transmission latency shifts to the worst case transmission latency. An increased payload increases the probability that a preamble has to be transmitted during data transmission. Therefore, an increase of the average transmission latency towards the worst case is faster for shorter frames.

As a consequence, for a subnetwork with a streaming based resource allocation, the time interval of the preamble transmission should be increased as much as possible, which equals increasing the frame duration. However, this might lead to an even further increased signaling overhead that is required to inform the slave about the allocated resources.

Finally, in Fig. 5.12 we compare the synchronous frame conversion approach with the best frame conversion approach of asynchronous static ($\Delta t_{\text{Frame}} = 0.7\,\text{ms}$) and streaming based resource allocation ($\Delta t_{\text{Frame}} = 0.9\,\text{ms}$). Over all simulations, asynchronous frame conversion results in lowest forwarding latencies in best case. For up to six slave, streaming based frame conversion could achieve lower latencies compared to the synchronous frame conversion as well. For more than six slaves, this is no longer true, even in the best case, due to the higher assumed signaling overhead. However, in worst case, both asynchronous and streaming based frame conversion show higher latencies compared to the synchronous frame conversion. For the simulated number of slaves the worst results are obtained for asynchronous frame conversion. However, the latency increases faster for the streaming based approach. Comparing the forwarding jitter of asynchronous and streaming based frame conversion, the latter show lower variance since only the preamble duration and not an entire frame will delay the transmission.

As a result of our simulative evaluation with respect to the forwarding jitter of the data transmission, a subnetwork that enables a synchronous frame conversion is always preferred. Considering only transmission latencies, the other approaches might perform

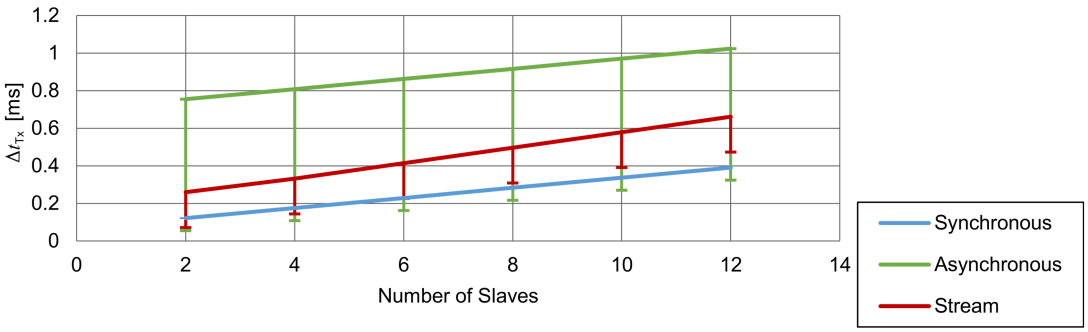

Figure 5.12: *Comparison of simulated worst case transmission latency with error indicators for the best allocation approaches.*

better in the best cases. However, for industrial application worst case transmission latency must always be considered, since they will define the deadlines in control loops. Accordingly, if no synchronization is possible, a streaming based resource allocation would be preferable over static resource allocation for asynchronous frame conversions.

This analysis reveals once more that the individual subnetworks within a cascaded communication system should not only be revised individually. Instead, parameters such as the individual frame offset, have to be adapted with respect to the overall data transmission. Therefore, within the next chapter we will investigate such a concatenated parameter adaptation.

5.4 Discussion: Related Limitation

Within this chapter, we focused on the transmission latency that is introduced during forwarding from one subnetwork to a subsequent subnetwork. The arbitrary network protocols that build a cascaded communication system use different frame structures and resource allocation schemes to forward corresponding data. Therefore, an incoming data packet cannot simply be forwarded.

The conversion of the frame causes a certain latency. This latency depends on the time interval between incoming data, the length of the corresponding frames for their forwarding, the time interval between successive frame transmissions, and the resource allocation method of these frames. Therefore, we introduced a differentiation of the various frame conversion schemes. The main difference results from frame duration and repetition rate compared to the interval of data arrival. The frame conversion is synchronous if the frame duration and corresponding inactive time is an integer multiple of the cycle time of the preceding subnetwork. Otherwise, we have an asynchronous frame conversion. According to the resource allocation of the corresponding frames, we can further differentiate between a static and a streaming based resource allocation. Depending on the amount of frames required to forward the data, several subcategories exist.

In order to evaluate their usability, we have compared these different frame conversion schemes with their individual subcategories analytically and by simulation. Our inves-

tigation revealed that with respect to the introduced forwarding jitter, a synchronous frame conversion is preferable. With a corresponding adaptation of the frame transmission offset to the reception time of data, this approach would also lead to lowest transmission latencies. However, only limited existing network protocols will support such a frame conversion. The IE networks considered in this work have strict boundaries for their parameterization and cannot be adapted arbitrarily. This applies in particular to necessary adjustment of the frame length and cycle time. Such parameter adaptations might also result in further challenges, like the relation of the payload compared to the overhead or the transmission within the wireless channel, for wireless protocols. Moreover, it requires a very precise clock synchronization of the individual clocks.

The second best frame conversion scheme is the asynchronous stream based resource allocation. As we showed, this approach performs best for large frames, which equals large preamble repetition intervals. Again, especially with respect to the wireless transmission channel this might result in further challenges. Moreover, the required signaling overhead for this approach has yet to be evaluated. Independent of the relatively conservative assumptions we have made here, the actual implementation effort for this approach is likely to be significantly higher compared to the other approaches. Since resources are not statically allocated, it will be difficult to achieve the required determinism.

Asynchronous frame conversion with statically allocated resources results in the highest worst case transmission latency and forwarding jitter. Here, one or several frames have to remain unused if the corresponding data arrive later than the frames are transmitted. The described improvement, which reduced the number of unused frames to only one, significantly reduced the transmission latency for higher amounts of data. However, this approach changes the frame order within a communication cycle. This has to be compensated for in order to maintain the determinism.

Since low latency and forwarding jitter are of special importance for industrial applications, we will focus on synchronous frame conversion within the remainder of this work. Although this also has some disadvantages, it has the highest potential in terms of requirements. Particularly with regard to the development of a wireless communication protocol for FA applications, a further development of this approach can be very useful towards enabling the required level of integration.

Chapter 6

Parameter Adaptations for Latency Reduction

Cascaded communication networks can contain arbitrary network protocols. Each of these networks can have its own individual parameterization in order to provide an specific quality of service. However, in cascaded communication networks, complex dependencies emerge due to both limits of individual subnetworks regarding their parameterization and specific requirements of the applications. As a consequence, not only the individual subnetworks themselves but the entire communication chain has to be optimized. Once again, we will focus on latency reduction, since this is one of the most important factors for fulfilling the requirements of industrial applications.

In Chapter 5 we concluded that a synchronous frame conversion to be the most promising approach for developing a cascaded communication network for industrial application. Compared to other frame conversion schemes, it does not introduce any additional jitter to data transmission. However, its worst case transmission latency is very high since an unadapted data forwarding might result in additional delays of entire communication cycles. Therefore, we assess the parameterization of the individual subnetworks within a cascaded communication system as a whole in this chapter. This is intended to improve the overall communication chain, making it easier to fulfill the requirements of industrial applications.

We start with an overview of related methods for parameterization of industrial communication networks in Section 6.1. In order to derive a new parameterization concept for cascaded communication networks, in Section 6.2 we reveal specific parameter boundaries that result from both: application requirements and characteristics of the considered communication networks. Taking into account the revealed parameter boundaries, we start analyzing the offset parameter of individual subnetworks in a cascaded communication network in order to reduce the overall transmission latency. Therefore, in Section 6.3 we derive an analytical offset adaptation. Due to the complexities that have to be considered, we propose an alternative approach for finding an aligned parameterization by using metaheuristics in Section 6.4. In addition to the adaptation of the offset parameter, we further analyze specific resource allocation schemes as an additional method for compensating for forwarding and conversion latencies in cascaded communication systems in Section 6.5. Finally, we conclude this chapter in Section 6.6.

6.1 Related Parameterization Methods

On the one hand side, industrial applications are mostly designed top-down according to the communication system [95]. The control actions are calculated for the worst case data transmission and the controller is adapted to it. This conservative assumption could result in a performance degradation. On the other hand side, commonly used industrial communication networks are designed bottom-up with respect to the Open Systems Interconnection (OSI) model. They are especially adapted to the conditions of the physical layer in order to provide certain real-time (RT) capabilities. This specification often means that only a part of the application requirements are completely fulfilled and compromises have to be accepted for others. This can result in disadvantages in form of further performance degradations, especially when applications are expanded or adapted to new production processes.

Consequently, optimal performance is mainly reliant on co-designing the application and communication networks. Several dependencies between application, communication and physical conditions increase the complexity of such a co-design. Therefore, the authors of [95] formulate the design as an multi-parameter optimization problem to achieve dedicated characteristics. They described network specific constraints and cost functions specially focused on the optimization of energy consumption of application and communication. Using an integer real optimization, they were able to improve the design of two exemplary application and network combinations.

A similar optimization approach is presented in [96]. The authors implemented a mixed integer linear programming optimization in order to improve a network virtualization. Therefore, they introduce mathematical descriptions of virtual network capacities, as well as data flow, latency, and node sharing constraints. Their optimization reduced the number on necessary virtual network functions while only minimally compromising latency and link utilization.

Both optimization approaches show an increased performance according to the intended characteristics. Therefore, it seems beneficial to transfer such an optimization approach to the factory automation (FA) applications and their required low latencies considered in this work. This includes not only the co-design of one application and one communication network, but should cover several applications by cascaded communication networks. The additional dependencies in the data transmission increase the complexity of such an optimization even further.

Cascaded networks so far are mainly based on combinations of static network protocols with only limited adaptability. Examples, such as shown in [67–70], use a best effort data forwarding that is sufficient for the intended applications. Therefore, a combined parameter optimization was of limited interest. However, due to additional latencies and reduced determinism, these cascaded networks are unsuitable for closed-loop applications considered in this work. Especially in combination with more flexible subnetworks, such as *ParSec*, an adequate co-design of the cascaded network and the intended application could further improve the overall performance. Therefore, in the remainder of this chapter we will investigate such a aligned parameter adaptation of cascaded networks based on our previously introduced network model. For this purpose, we will

first reveal boundary conditions that result from applications and the networks under consideration.

6.2 Parameter Boundaries

According to our presented communication model (cf. Fig. 3.1), data transmission within a subnetworks depends on several parameters. For a subnetwork N_i these parameters are the individual frame starts related to the start of the application cycle t_0 (t_{DL_i} and t_{UL_i}), the corresponding downlink or uplink frame length ($\Delta t_{\mathrm{DLFrame}_i}$ and $\Delta t_{\mathrm{ULFrame}_i}$), the offset of the RT data within the frames ($\Delta t_{\mathrm{DLRTOfs}_i}$ and $\Delta t_{\mathrm{ULRTOfs}_i}$), the allocation time of an individual data packet j ($\Delta t_{\mathrm{DLAlo}_{i,j}}$ and $\Delta t_{\mathrm{ULAlo}_{i,j}}$), and the inactive time between two consecutive frame transmissions ($\Delta t_{\mathrm{DLInact}_i}$ and $\Delta t_{\mathrm{ULInact}_i}$).

Some of these parameters are directly depending upon each other. For example, the cycle time is comprised of frame duration and inactive time. Therefore, increasing one of them would decrease the other. Another example is the offset of the RT data within the frame. As long as the frame length does not increase likewise with a RT data offset, it reduces the total amount of RT data that can be transmitted within the frame.

Moreover, in cascaded networks the parameter of one subnetwork and the corresponding data transmission can have a direct influence on subsequent data transmissions. Therefore, the parameterization of one subnetwork also influences other subnetworks. As a result, we have to analyze the overall communication chain for an optimized data transmission.

The variability of the individual parameters influences the adaptation towards specific requirements. The fewer limitations they have, the higher the probability of an optimized parameterization solving a specific problem. However, not all parameters can be set completely independently. Some of them are restricted by protocol defaults or application requirements. These restrictions could either statically specify a parameter or limit it between a restricted lower and upper bound within its defined range.

Protocol based restrictions can be individual for each subnetwork according to protocol based predefinitions and network behavior. They are mainly based on physical limitations or implementation aspects, such as fixed frame size and structure. For the wired *Sercos* network we consider in this work, some parameter boundaries result from its Industrial Ethernet basis. The most important examples are the maximum frame duration $\Delta t_{\mathrm{Frame}_{\mathrm{S,Max}}} \approx 122.1\,\mu\mathrm{s}$, the minimum inactive time, which is related to the minimum guard interval between successive frames, $\Delta t_{\mathrm{Inact}_{\mathrm{S,Min}}} \approx 1.0\,\mu\mathrm{s}$ and the required frame overhead transmission time $\Delta t_{\mathrm{OH}_{\mathrm{S}}} \approx 2.1\,\mu\mathrm{s}$. Other boundaries result from the intended transmission structure itself, where downlink (DL) and uplink (UL) data are transmitted sequentially. Accordingly, both transmissions have to fit into one communication cycle:

$$\Delta t_{\mathrm{DLFrame}_{\mathrm{S}}} + \Delta t_{\mathrm{ULFrame}_{\mathrm{S}}} + 2 \cdot \Delta t_{\mathrm{Inact}_{\mathrm{S,Min}}} \leq \Delta t_{\mathrm{Cycle}} \tag{6.1}$$

Thus, for the DL we get the following boundaries:

A DL frame could start earliest right after the guard interval that follows the UL frame of the previous communication cycle:

$$t_{\mathrm{DL_{S,Min}}} = t_{\mathrm{UL_S}} + \Delta t_{\mathrm{ULFrame_S}} + \Delta t_{\mathrm{Inact_{S,Min}}} - \Delta t_{\mathrm{Cycle}} \tag{6.2}$$

The latest possible start is determined by the scheduled start of the UL frame and the time required for the DL frame transmission and a guard interval:

$$t_{\mathrm{DL_{S,Max}}} = t_{\mathrm{UL_S}} - \Delta t_{\mathrm{DLFrame_S}} - \Delta t_{\mathrm{Inact_{S,Min}}} \tag{6.3}$$

Between two successive DL frames must be enough time to transmit a UL frame. The corresponding minimum inactive time for the DL results as:

$$\Delta t_{\mathrm{DLInact_{S,Min}}} = \Delta t_{\mathrm{ULFrame_S}} + 2 \cdot \Delta t_{\mathrm{Inact_{S,Min}}} \tag{6.4}$$

The maximum inactive time for the DL results of the cyclic repetition and the time required to transmit the frame:

$$\Delta t_{\mathrm{DLInact_{S,Max}}} = \Delta t_{\mathrm{Cycle}} - \Delta t_{\mathrm{DLFrame_S}} \tag{6.5}$$

The maximum offset to schedule the RT data within the DL frame results from its maximum frame duration and the time required to transmit the frame overhead and RT data:

$$\Delta t_{\mathrm{DLRTOfs_{S,Max}}} = \Delta t_{\mathrm{DLFrame_{S,Max}}} - \Delta t_{\mathrm{DLOH_S}} - \Delta t_{\mathrm{DLRTData_S}} \tag{6.6}$$

Similarly, we get for the UL the following boundaries:

$$t_{\mathrm{UL_{S,Min}}} = t_{\mathrm{DL_S}} + \Delta t_{\mathrm{DLFrame_S}} + \Delta t_{\mathrm{Inact_{S,Min}}} \tag{6.7}$$

$$t_{\mathrm{UL_{S,Max}}} = t_{\mathrm{DL_S}} + \Delta t_{\mathrm{Cycle}} - \Delta t_{\mathrm{DLFrame_S}} - \Delta t_{\mathrm{Inact_{S,Min}}} \tag{6.8}$$

$$\Delta t_{\mathrm{ULInact_{S,Min}}} = \Delta t_{\mathrm{DLFrame_S}} + 2 \cdot \Delta t_{\mathrm{Inact_{S,Min}}} \tag{6.9}$$

$$\Delta t_{\mathrm{ULInact_{S,Max}}} = \Delta t_{\mathrm{Cycle}} - \Delta t_{\mathrm{ULFrame_S}} \tag{6.10}$$

$$\Delta t_{\mathrm{ULRTOfs_{S,Max}}} = \Delta t_{\mathrm{ULFrame_{S,Max}}} - \Delta t_{\mathrm{ULOH_S}} - \Delta t_{\mathrm{ULRTData_S}} \tag{6.11}$$

The wireless *ParSec* protocol we consider in this work was specially developed as extension for wired Industrial Ethernet (IE) networks. Therefore, it offers more flexibility in its parameterization compared to the *Sercos* network. This should enable easier adaptation and integration. However, due to implementation reasons, there are some parameter limitations as well. Those result, among others, from the wireless transmission and the necessary channel estimation. The corresponding most important parameter boundaries are:
The start of the UL frame is fixed to the end of the preamble transmission of the DL frame due to channel estimation reason:

$$t_{\mathrm{UL_P}} = t_{\mathrm{DL_P}} + \Delta t_{\mathrm{DLOH_P}} \tag{6.12}$$

In order to achieve a synchronous relation between the individual subnetworks, at least a minimum inactive time between two successive DL frames is required, if the cycle time

is not an integer multiple of the time required to transmit an *ParSec* resource element (RE):

$$\Delta t_{\mathrm{DLInact_{P,Min}}} = \Delta t_{\mathrm{Cycle}} - \left\lfloor \frac{\Delta t_{\mathrm{Cycle}}}{\Delta t_{\mathrm{RE_P}}} \right\rfloor \cdot \Delta t_{\mathrm{RE_P}} \tag{6.13}$$

In addition to the inactive time that is required to achieve the synchronous relation, in UL direction there is no transmission allowed while the DL preamble is transmitted. The minimum UL inactive time increases accordingly:

$$\Delta t_{\mathrm{ULInact_{P,Min}}} = \Delta t_{\mathrm{DLInact_{P,Min}}} + \Delta t_{\mathrm{DLOH_P}} \tag{6.14}$$

This also limits the maximum UL frame duration:

$$\Delta t_{\mathrm{ULFrame_{P,Max}}} = \Delta t_{\mathrm{DLData_P}} + \Delta t_{\mathrm{DLInact_P}} \tag{6.15}$$

For both, DL and UL the RT data offset within the frame may have a maximum duration that still allows the transmission of all RT data:

$$\Delta t_{\mathrm{DLRTOfs_{P,Max}}} = \Delta t_{\mathrm{DLFrame_{P,Max}}} - \Delta t_{\mathrm{DLOH_P}} - \Delta t_{\mathrm{DLRTData_P}} \tag{6.16}$$

$$\Delta t_{\mathrm{ULRTOfs_{P,Max}}} = \Delta t_{\mathrm{ULFrame_{P,Max}}} - \Delta t_{\mathrm{ULOH_P}} - \Delta t_{\mathrm{ULRTData_P}} \tag{6.17}$$

More parameter restrictions result from the requirements of the application. The most important one is the definition of the cycle time $\Delta t_{\mathrm{Cycle}}$. As shown in Chapter 5, the cycle time is ideally the same in all subnetworks. Therefore, this restriction has a high influence on the overall network performance. Other parameter boundaries result from the required timing of the application. The different timing methods introduced in Subsec. 3.2.3 result each in a different set of parameter restrictions.

For the command data optimized timing, the controller has to compute new command data based on the latest actual data. Therefore, the UL has to be transmitted as early as possible. Hence, the global sampling point (GSP) must be set considering the UL processing time at the slave $\Delta t_{\mathrm{ULProc}}$. After the reception of the actual data at the master, there must be enough time to process them and compute new command data for the subsequent communication cycle $\Delta t_{\mathrm{DLComp}}$. Considering only one communication network, the following parameter boundaries emerge (cf. Fig. 3.4):

$$t_{\mathrm{UL_{Min}}} = t_{\mathrm{GSP}} + \Delta t_{\mathrm{ULProc}} - \Delta t_{\mathrm{ULOH}} - \Delta t_{\mathrm{ULRTOfs}} \tag{6.18}$$

$$t_{\mathrm{DL_{Max}}} = t_{\mathrm{UL}} + \Delta t_{\mathrm{ULFrame}} + \Delta t_{\mathrm{DLComp}} - \Delta t_{\mathrm{DLOH}} - \Delta t_{\mathrm{Cycle}} \tag{6.19}$$

For the actual data optimized timing, the requirements are reversed compared to the command data optimized timing. The actual data have to contain an immediate reaction to the latest command data. Therefore, the time between the DL and subsequent UL transmission must be long enough that the command data can be completely received and processed. Moreover, there must be enough time to activate these command data and transfer them into new actual data. For this timing, the GSP has to be set between DL and UL transmission. Consequently, the following parameter boundaries emerge (cf. Fig. 3.5):

$$t_{\mathrm{GSP_{Min}}} = t_{\mathrm{DL}} + \Delta t_{\mathrm{DLFrame}} + \Delta t_{\mathrm{DLProc}} \tag{6.20}$$

$$t_{\mathrm{UL_{Min}}} = t_{\mathrm{GSP}} + \Delta t_{\mathrm{ULProc}} - \Delta t_{\mathrm{ULOH}} - \Delta t_{\mathrm{ULRTOfs}} \tag{6.21}$$

The closed-loop optimized timing method combines both previous timing methods. The actual data shall contain an immediate reaction to the latest command data, and the subsequent command data shall be based on this reaction. The inactive time between both DL and UL, as well as UL and subsequent DL, has to be long enough for all processing and computation to be performed. The GSP has to be set between DL and UL. Only with this timing method can a complete control loop be covered by only one communication cycle (cf. Fig. 3.6). However, this requires further delimitation of the parameters:

$$t_{\mathrm{GSP_{Min}}} = t_{\mathrm{DL}} + \Delta t_{\mathrm{DLFrame}} + \Delta t_{\mathrm{DLProc}} \tag{6.22}$$

$$t_{\mathrm{UL_{Min}}} = t_{\mathrm{GSP}} + \Delta t_{\mathrm{ULProc}} - \Delta t_{\mathrm{ULOH}} - \Delta t_{\mathrm{ULRTOfs}} \tag{6.23}$$

$$t_{\mathrm{DL_{Max}}} = t_{\mathrm{UL}} + \Delta t_{\mathrm{ULFrame}} + \Delta t_{\mathrm{DLComp}} - \Delta t_{\mathrm{DLOH}} - \Delta t_{\mathrm{Cycle}} \tag{6.24}$$

$$\Delta t_{\mathrm{DLInact_{Min}}} = \Delta t_{\mathrm{DLProc}} + \Delta t_{\mathrm{ULProc}} + \Delta t_{\mathrm{DLComp}} + \Delta t_{\mathrm{ULRTData}} - \Delta t_{\mathrm{DLOH}} \tag{6.25}$$

$$\Delta t_{\mathrm{ULInact_{Min}}} = \Delta t_{\mathrm{DLProc}} + \Delta t_{\mathrm{ULProc}} + \Delta t_{\mathrm{DLComp}} + \Delta t_{\mathrm{DLRTData}} - \Delta t_{\mathrm{ULOH}} \tag{6.26}$$

The cycle time optimized timing offers the highest flexibility regarding the timing requirements. It is the only timing method that does not generate any further parameter boundaries in addition to the network based boundaries. This flexibility has the disadvantage that several communication cycles might be required for both DL and UL transmission (cf. Fig. 3.7). However, in this cases, the cycle time itself is a parameter that has to be adapted in order to achieve comparable results.

According to the investigated requirements, there could be several more parameter boundaries. The boundaries introduced above represent only a subset required for our further latency analysis. The network specific and application specific parameter boundaries partially overlap or refine each other. This again represents the complexity of optimizing a cascaded communication system.

However, only such a combined optimization of individual subnetworks allows to maintain their advantages causing mostly only limited performance degradation. Therefore, within the subsequent section we will introduce a gradual adaptation of the individual parameterization according to the considered requirements and conditions.

6.3 Analytical Offset Adaptation

As introduced above, the design of a cascaded communication network is a very complex process that requires the consideration of several dependencies. Its optimization is required in order to achieve the desired performance of the overall communication system. In this section, we derive a general analytical parameter optimization, as introduced by us in [6]. Therefore, we consider once again the machine tool application example introduced in Sec. 2.4 and Subsec. 3.3.1. The corresponding cascaded network structure is depicted in Fig. 3.8.

We assume a proxy-based gateway concept between the individual subnetworks (cf. Chapter 4). According to Chapter 5, we further assume synchronized cyclic communication with a cycle time of $\Delta t_{\mathrm{Cycle}} = 1\,\mathrm{ms}$ in every subnetwork. Moreover, we suppose a processing and computation time of command and actual data of $\Delta t_{\mathrm{Proc}} = \Delta t_{\mathrm{Comp}} =$

50 µs each. Also for the conversion time between subnetworks we assume a duration of $\Delta t_{\mathrm{Conv}} = 50\,\mu\mathrm{s}$. The amount of command and actual data for each of the $s_{\mathrm{Max}} = 33$ slaves is 20 Bytes. The transmission duration for this 20 Bytes equals $\Delta t_{\mathrm{Data}_{i,\mathrm{S}}} = 1.6\,\mu\mathrm{s}$ within the *Sercos* network and $\Delta t_{\mathrm{Data}_{i,\mathrm{P}}} = 28.5\,\mu\mathrm{s}$ within the *ParSec* network. The order of data transmission corresponds to the sequence of the slaves. These assumptions represent a typical industrial average.

The standard parameterization of the individual subnetworks results in an overall timing behavior presented in Tab. 3.2. In order to improve this timing, we have to adapt the parameters of the individual subnetworks with respect to the overall communication and their parameter boundaries. Considering the network based parameter boundaries, the following parameters remain for an adaptation:

- the individual DL offset of each subnetwork t_{DL_i},
- the individual UL offset of each subnetwork t_{UL_i},
- the offset of the RT data within the individual DL frames $\Delta t_{\mathrm{DLRTOfs}_i}$,
- the offset of the RT data within the individual UL frames $\Delta t_{\mathrm{ULRTOfs}_i}$, and
- the timing of the GSP t_{GSP}.

In this example, the total number of subnetworks is $n_{\mathrm{Max}} = 6$. Since the GSP has to be the same for all subnetworks, there are 25 parameters to be aligned. Due to the cascaded communication structure, all of them have a concatenated influence on the overall timing behavior. Consequently, changing one of them to achieve a better intermediate timing (e.g. reduce the reception time at a certain gateway) might result in a worse overall timing if another parameter is not changed as well.

According to our frame model, having a real-time data offset $\Delta t_{\mathrm{RTOfs}}$ would increase the corresponding frame duration. Especially for the considered *Sercos* network, this would decrease the flexibility of the UL frame offset since DL and UL cannot be transmitted at the same time and the UL follows the DL. Instead, we rate the influence of the total offset of a frame (t_{DL_i} and t_{UL_i}) to be higher than the offset of the data within the corresponding frame. Therefore, for *Sercos* based subnetworks N_1 and N_3 to N_6, we adapt the real-time data offset for each DL and UL frame to $\Delta t_{\mathrm{RTOfs}_S} = 0$. Consequently, we can fully adapt the UL frame offset t_{UL_S}.

Considering the parameter boundaries resulting from the network structure and the application, we can now find an improved parameterization that reduces the transmission latency. In the following, we derive the improved parameterization assuming that the application requires a closed-loop optimized timing. For all other timing requirements the approach would be similar but less constrained. First, we have to find a timing for the GSP such that $\forall S_j \mid j \in \{1, \cdots, s_{\mathrm{Max}}\}$:

$$t_{\mathrm{GSP}} \geq \max_j \left\{ t_{\mathrm{DLRx}_j} + \Delta t_{\mathrm{DLProc}_j} \right\} \tag{6.27}$$

Accordingly, we have to find the slave in the cascaded network that completes the processing of its command data last. This depends on the network structure, the transmission rates of the individual subnetworks, and the number of slaves in the individual

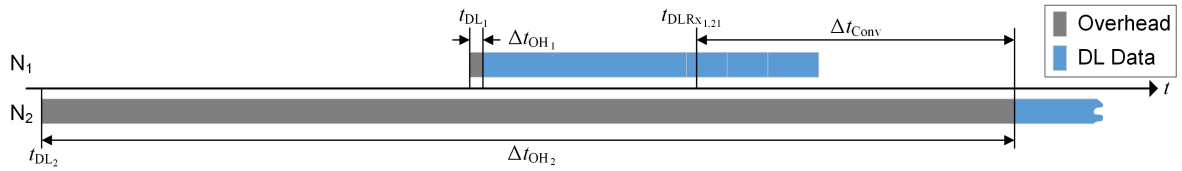

Figure 6.1: *Exemplary adaptation of the DL offset of a subsequent subnetwork.*

subnetworks. In our example, the *ParSec* subnetwork N_2 on L_2 has a lower transmission rate than the *Sercos* subnetworks. Moreover, the amount of data that have to be forwarded through N_2 is that large that remaining data transmission on L_1 will be completed early. Therefore, in our example, slave S_{33} in subnetwork N_6 on level L_3 receives its command data last.

We can now reduce this reception time by analyzing the path of the corresponding data. These data are forwarded from N_1 on L_1 to N_2 on L_2 and N_6 on L_3. On this path, additional latencies will emerge, if the data have to be stalled until they can be forwarded. Therefore, we can reduce the transmission latency by adapting the DL frame offset of subsequent subnetworks such that incoming data can directly be forwarded:

$$t_{\mathrm{DLRx}_{i,j}} + \Delta t_{\mathrm{ConvLn}_i} \stackrel{!}{=} t_{\mathrm{DL}_{(i+1)}} + \Delta t_{\mathrm{DLOH}_{(i+1)}} + \Delta t_{\mathrm{DLRTOfs}_{(i+1)}} + \Delta t_{\mathrm{DLAlo}_{(i+1),j}} \tag{6.28}$$

However, due to the individual subnetworks and their performance, it is not always possible to adapt the transmission for every individual data packet. Thus, we have to consider different transmission rates of individual subnetworks. For a subnetwork with a lower transmission rate than the incoming data rate, we have to adapt its offset to the first received data packet. Assuming sequential data transmission, this guarantees that all remaining data packets will then be received and processed before their allocated resource elements will be transmitted.

In our example, the *ParSec* subnetwork N_2 has a lower transmission rate than the other subnetworks. Therefore, we can adapt the offset of its DL frame according to the first received data packet from N_1 that has to be forwarded, as shown in Fig. 6.1. Using Eq. (6.28), we can calculate the offset as follows:

$$t_{\mathrm{DL}_2} = t_{\mathrm{DLRx}_{1,21}} + \Delta t_{\mathrm{ConvLn}_1} - \Delta t_{\mathrm{DLOH}_2} - \Delta t_{\mathrm{DLRTOfs}_2} - \Delta t_{\mathrm{DLAlo}_{2,21}} \tag{6.29}$$

The calculation of the DL reception time $t_{\mathrm{DLRx}_{1,21}}$ could be derived from our mathematical description presented in Sec. 3.3:

$$t_{\mathrm{DLRx}_{1,21}} = t_{\mathrm{DL}_1} + \Delta t_{\mathrm{DLOH}_1} + \Delta t_{\mathrm{DLRTOfs}_1} + \Delta t_{\mathrm{DLAlo}_{1,21}} + \Delta t_{\mathrm{DLData}_{1,21}} + \Delta t_{\mathrm{DLTran}_{1,21}} \tag{6.30}$$

As described in Sec. 3.3, we can neglect the time required to transmit the data from slave to gateway $\Delta t_{\mathrm{DLTran}_{1,21}}$ because is it negligibly small. Furthermore we set the RT data offset in N_1 to $\Delta t_{\mathrm{DLRTOfs}_1} = 0\,\mathrm{s}$. The DL frame of N_1 starts at $t_{\mathrm{DL}_1} = 0\,\mathrm{s}$. Considering the *Sercos* parameterization, the overhead transmission time equals

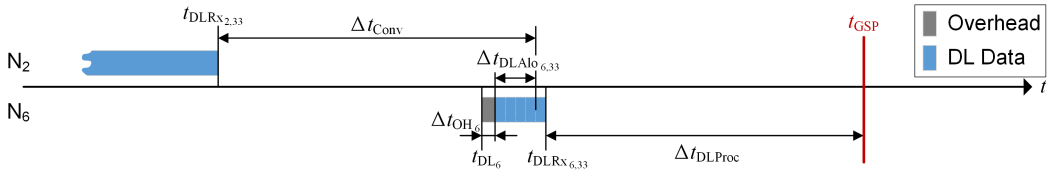

Figure 6.2: *Exemplary adaptation of GSP based on the last data reception.*

$\Delta t_{\mathrm{DLOH}_1} \approx 2.1\,\mu\mathrm{s}$ and it take $\Delta t_{\mathrm{DLData}_{1,21}} = 1.6\,\mu\mathrm{s}$ to transmit 20 Bytes. Accordingly, we get $t_{\mathrm{DLRx}_{1,21}} \approx 35.7\,\mu\mathrm{s}$ for the reception time of data packet for slave S_{21} at the gateway between N_1 and N_2 using Eq. (6.30). In N_2 this data packet is scheduled first, since it is the first one that has to be forwarded ($\Delta t_{\mathrm{DLAlo}_{2,21}} = 0\,\mathrm{s}$). As a result, with the *ParSec* frame overhead of $\Delta t_{\mathrm{DLOH}_2} = 153\,\mu\mathrm{s}$ and a conversion time of $\Delta t_{\mathrm{ConvLn}_1} = 50\,\mu\mathrm{s}$ we get from Eq. (6.29) the offset of $t_{\mathrm{DL}_2} \approx -67.4\,\mu\mathrm{s}$. This negative offset results from the high overhead of *ParSec*. It could be realized considering the cyclic communication behavior. According to $\Delta t_{\mathrm{Cycle}} = 1\,\mathrm{ms}$, it is equal to $t_{\mathrm{DL}_2} \approx 932.7\,\mu\mathrm{s}$.

Similarly, we can calculate the offset of the DL frames for the subnetworks N_3 to N_6. However, for these subnetworks, the incoming data rate is lower than the transmission rate. Therefore, we have to adapt the offset according to the last allocated data packet (S_{21}, S_{24}, S_{28}, respectively S_{33}) of the corresponding frame using Eq. (6.28). Fig. 6.2 depicts this exemplarily for the adaptaion of t_{DL_6}. The calculated parameter can be found in Tab. 6.1. With this adaptation of DL offsets, we have reduced the latest data reception of slave S_{33} in subnetwork N_6 down to $t_{\mathrm{DLRx}_{6,33}} \approx 507.8\,\mu\mathrm{s}$.

Having t_{DLRx} reduced for all slaves, we can calculate a latency adapted GSP using Eq. (6.27):

$$t_{\mathrm{GSP}} = t_{\mathrm{DLRx}_{6,33}} + \Delta t_{\mathrm{DLProc}_{33}} \tag{6.31}$$

With an processing time of $\Delta t_{\mathrm{DLProc}_{33}} = 50\,\mu\mathrm{s}$, this results in an earliest possible GSP at $t_{\mathrm{GSP}} \approx 557.8\,\mu\mathrm{s}$ (cf. Fig. 6.2).

Now, we have adapted the DL transmission such that all slaves receive their command data before an earliest possible GSP. In order to enable the closed-loop optimized timing, we have to adapt the UL transmission as well. Therefore, we search for actual data that the master receives last and investigate this communication chain.

For our example, this is again the data packet of slave S_{33}. Its data are first transmitted in subnetwork N_6. We can reduce the transmission latency in this subnetwork by adapting its UL such that processed data can directly be transmitted. In this example, level L_3 is the highest level of the cascaded network. Therefore, for subnetworks on this level we can directly adapt the offset of the corresponding UL frame to the GSP since we are independent of the data arrival of higher levels. Since we assume equal processing times for all slave ($\Delta t_{\mathrm{ULProc}} = 50\,\mu\mathrm{s}$), we get:

$$t_{\mathrm{UL}_{3\ldots6}} = t_{\mathrm{GSP}} + \Delta t_{\mathrm{ULProc}} - \Delta t_{\mathrm{ULOH}_{3\ldots6}} \tag{6.32}$$

As all subnetworks on L_3 are based on *Sercos* with $\Delta t_{\mathrm{OH}} \approx 2.1\,\mu\mathrm{s}$, we can set the

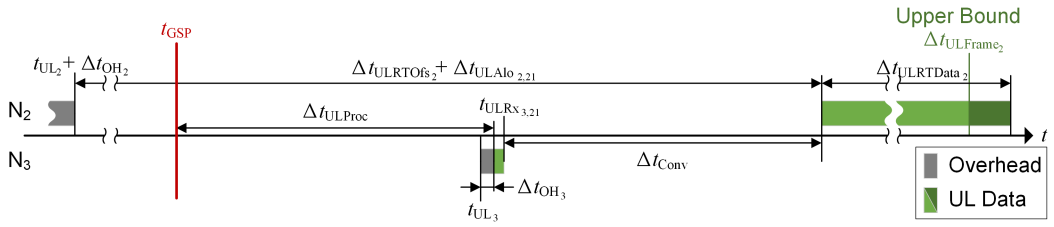

Figure 6.3: *Exemplary adaptation of UL offset based on GSP and earliest data reception.*

offset of their UL frame equally to $t_{\mathrm{UL}_{3\ldots6}} \approx 605.7\,\mu\mathrm{s}$ (cf. Fig. 6.3).

After transmission on L_3, data are forwarded through N_2 on L_2. Accordingly, we have to adapt its UL frame to corresponding data reception. This depends again on the resource allocation order and the data transmission rate compared to the incoming data rate. Since in our example, the *ParSec* network on L_2 has the lower data rate, we can ideally adapt the frame according to the first data packed that has to be forwarded. Based on the allocation corresponding to the slave number we have to consider the data reception of slave S_{21} for the adaptation:

$$t_{\mathrm{ULRx}_{3,21}} + \Delta t_{\mathrm{ConvLp}_3} = t_{\mathrm{UL}_2} + \Delta t_{\mathrm{ULOH}_2} + \Delta t_{\mathrm{ULRTOfs}_2} + \Delta t_{\mathrm{ULAlo}_{2,21}} \tag{6.33}$$

As described in Sec. 6.2, within *ParSec* the UL offset is statically related to the end of the preamble transmission of the DL frame (cf. Eq. (6.12)). Accordingly, the UL frame offset equals $t_{\mathrm{UL}_2} \approx 85.7\,\mu\mathrm{s}$. As a result, we have to adapt the corresponding RT data offset as compensation in order to adapt the overall data forwarding. From Eq. (6.33) follows:

$$\Delta t_{\mathrm{ULRTOfs}_2} = t_{\mathrm{ULRx}_{3,21}} + \Delta t_{\mathrm{ConvLp}_3} - t_{\mathrm{UL}_2} - \Delta t_{\mathrm{ULOH}_2} - \Delta t_{\mathrm{ULAlo}_{2,21}} \tag{6.34}$$

The UL reception time at the gateway between N_3 and N_2 could be calculated similar to Eq. (6.29). In our example, it equals $t_{\mathrm{ULRx}_{3,21}} \approx 609.4\,\mu\mathrm{s}$. Again, since the data packet of slave S_{21} is the first one allocated, the allocation time equals $\Delta t_{\mathrm{ULAlo}_{2,21}} = 0\,\mathrm{s}$. Accordingly, with the *ParSec* overhead time $\Delta t_{\mathrm{ULOH}_2} = 153\,\mu\mathrm{s}$ and assuming again equal conversion times of $\Delta t_{\mathrm{ConvLp}} = 50\,\mu\mathrm{s}$, we get from Eq. (6.34) an ideal RT data offset of $\Delta t_{\mathrm{ULRTOfs}_2} = 420.7\,\mu\mathrm{s}$.

Unfortunately, as shown in Fig. 6.3, this conflicts with the upper bound of the UL frame duration. Since the UL data transmission requires $\Delta t_{\mathrm{ULRTData}_2} = 13 \times 28.5\,\mu\mathrm{s} = 370.5\,\mu\mathrm{s}$, the entire UL frame transmission would require $\Delta t_{\mathrm{ULFrame}_2} = 944.2\,\mu\mathrm{s}$, whereas according to Eq. (6.15) a maximum of $\Delta t_{\mathrm{ULFrame}_{2,\mathrm{Max}}} = \Delta t_{\mathrm{DLData}_2} + \Delta t_{\mathrm{ULInact}_2} = 847.0\,\mu\mathrm{s}$ would be allowed. As a result, the UL frame transmission would overlaps with the DL frame of the next communication cycle for 97.2 µs, which is not allowed within *ParSec*. Therefore, the RT data offset within the UL frame equals 323.5 µs for this example in order to stay within the corresponding parameter bounds. As a consequence, data packets for slave S_{21} to S_{24} will not be available before their allocated resources will be transmitted. A delay of one communication cycle emerges for these data. In order to minimize the transmission latency for the remaining data, we can use Eq. (6.34) to

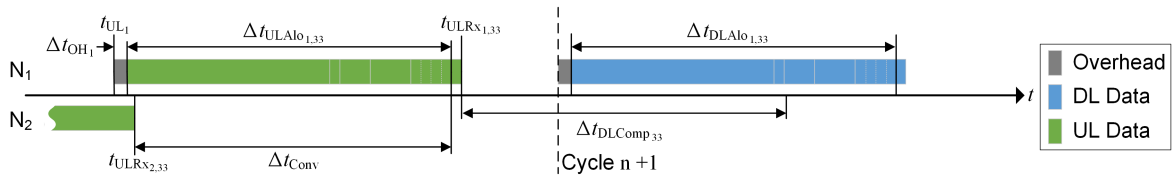

Figure 6.4: *Exemplary adaptation of UL offset based on latest data reception and verification of intended timing.*

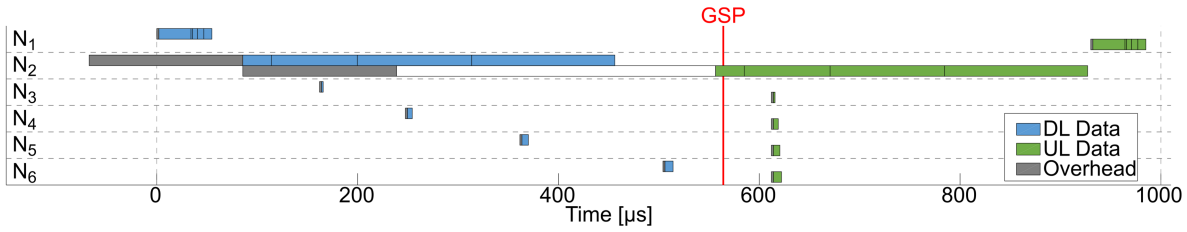

Figure 6.5: *Frame structure of exemplary cascaded network for machine tool application scenario using the parameterization of Tab. 6.1.*

directly adapt the RT data offset of N_2 to the forwarding of the UL data packet of salve S_{25}. With $t_{\mathrm{ULRx}_{2,25}} \approx 609.4\,\mu\mathrm{s}$ and $\Delta t_{\mathrm{ULAlo}_{2,25}} = 114\,\mathrm{s}$, the resulting RT data offset equals $\Delta t_{\mathrm{RTOfs}_2} = 306.7\,\mu\mathrm{s}$.

Having the UL transmission adapted until N_2, we finally have to adapt the offset of the UL frame of N_1 according to the data reception from N_2 (cf. Fig. 6.4). Again, we have to consider the order of the allocated data and the different transmission rates of the individual subnetworks. Accordingly, we have to adapt the offset to the data reception of slave S_{33} for our example. Adjusting Eq. (6.33) we get:

$$t_{\mathrm{ULRx}_{2,33}} + \Delta t_{\mathrm{ConvLp}_2} = t_{\mathrm{UL}_1} + \Delta t_{\mathrm{ULOH}_1} + \Delta t_{\mathrm{ULRTOfs}_1} + \Delta t_{\mathrm{ULAlo}_{1,33}} \tag{6.35}$$

Considering again a RT data offset of 0 s, we get with the corresponding overhead duration of $\Delta t_{\mathrm{ULOH}_1} \approx 2.1\,\mu\mathrm{s}$ and allocation offset of $\Delta t_{\mathrm{ULAlo}_{1,33}} = 51.2\,\mu\mathrm{s}$ the adapted UL frame offset of $t_{\mathrm{UL}_1} = 912.6\,\mu\mathrm{s}$. With this offset and the corresponding frame duration of $\Delta t_{\mathrm{ULFrame}_1} \approx 54.9\,\mu\mathrm{s}$, all data are received by the master at $\approx$967.5 µs.

In order to validated the closed-loop optimized timing, there must be enough time for computing new command data after receiving actual data (cf. Fig. 6.4):

$$t_{\mathrm{ULRx}_{1,i}} + \Delta t_{\mathrm{DLComp}} \leq t_{\mathrm{DL}_1} + \Delta t_{\mathrm{DLOH}_1} + \Delta t_{\mathrm{DLRTOfs}_1} + \Delta t_{\mathrm{DLAlo}_{1,i}} \tag{6.36}$$

In our example, Eq. (6.36) hold for all 33 data packets. Therefore, there are no further conflicts with any parameter boundaries. As a result, this is the best achievable parameter set for our investigated example for the closed-loop optimized timing approach. The corresponding frame structure for one entire communication cycle is depicted in Fig. 6.5.

Table 6.1: *Analytically optimized parameterization of the machine tool application scenario with closed-loop optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	932.7	162.1	244.4	356.8	497.7
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	912.6	85.7	605.7	605.7	605.7	605.7
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	306.7	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]	1000.0					
t_{GSP} [µs]	557.8					

The corresponding parameter sets for the other timing approaches can be found in the appendix in Tab. A.5 to A.7. We calculate them in the same way as the parameter for the closed-loop optimized timing. However, for the cycle time optimized timing approach, we have to adapt the cycle time itself as well. Since for the synchronous data conversion the cycle time has to be identical in all subnetworks, we have to adapt it to the subnetwork with the longest transmission time. In our example, minimum cycle time results from the UL transmission within the *ParSec* subnetwork and equals $t_{\mathrm{Cycle_{Min}}} = \Delta t_{\mathrm{ULOH}_2} + \Delta t_{\mathrm{ULRTData}_2} + \Delta t_{\mathrm{ULInact}_2} = 676.5\,\mu\mathrm{s}$.

In order to evaluate the resulting timing behavior of the application based on the adapted parameter sets, we again used our analytical algorithm presented in Alg. 1. The corresponding results are shown in Tab. 6.2. Compared to the timing behavior that results from an unadapted parameter set (cf. Tab. 3.2), we achieved clearly a reduction of transmission latency.

For the closed-loop optimized timing, we could achieve a reduction of the maximum DL reception time by about 1/2 with our adapted parameter set ($\Delta t_{\mathrm{DLRx_{Max}}} \approx 514.2\,\mu\mathrm{s}$) compared to the unadapted parameters ($\Delta t_{\mathrm{DLRx_{Max}}} \approx 1010.1\,\mu\mathrm{s}$). With our parameter set, all slaves receive their command data within the same cycle the master generates them. Moreover, the parameter adaption resulted in a reduction of 2/3 in the DL activation and UL acquisition time. Whereas the standard parameter set requires 1.5 cycles for activation and acquisition ($\Delta t_{\mathrm{DLAct_{Max}}} = 1500.0\,\mu\mathrm{s}$), it takes little more than 0.5 communication cycles with the adaptation ($t_{\mathrm{DLAct_{Max}}} \approx 564.2\,\mu\mathrm{s}$). However, even with our parameter adaptation it is not possible to schedule the transmission such that the master receives all actual data within the same cycle it provides the corresponding command data. Due to the limitation of the *ParSec* network and the corresponding parameter boundaries, a second cycle is required to transmit the actual data of four slaves to the master. Nevertheless, this is still a reduction of two communication cycles

compared to the unadapted parameter set. The same applies to the transmission of the subsequent command data.

Regarding the command data optimized timing, our parameter adaption results in a worse DL reception and DL activation time compared to the standard parameter set. However, for this approach an optimization of the UL transmission is most important. With our improved parameter set, it is possible to transmit all actual data to the master within the same cycle they were acquired. Moreover, the master is able to transmit an immediate response to these actual data within the subsequent communication cycle. Compared to the standard parameterization ($t_{\mathrm{DLReply_{Max}}} \approx 3054.9\,\mu\mathrm{s}$), we were able to reduce the maximum DL reply time by 2/3 ($t_{\mathrm{DLReply_{Max}}} \approx 1054.9\,\mu\mathrm{s}$).

The adapted parameterization for the actual data optimized timing is similar to the exemplified adaptation for the closed-loop optimized timing. However, in this case only the DL has to be adapted. With our improved parameter set, we were able to reduce the maximum DL reception by 2/3 from $t_{\mathrm{DLRx_{Max}}} \approx 1010.9\,\mu\mathrm{s}$ to $t_{\mathrm{DLRx_{Max}}} \approx 514.2\,\mu\mathrm{s}$. Therefore, all slaves can activate their command data and acquire the corresponding actual data within the same cycle. Since the subsequent DL reply time is of minor interest for this optimization, this timing remains unchanged compared to the standard parameterization. Additionally, this reply does not necessarily include a direct reaction to the latest actual data as it is not required for this timing.

The same applies to the results of the parameter adaptation for the cycle time optimized timing. In this case, neither are all actual data based on latest command data, nor do the subsequent command data contain a reaction to this actual data. However, at least for the slaves of the root subnetwork, the transmission latency is very low due to the minimized cycle time. For the slaves in the subnetworks on L_3, two communication cycles are required to transmit the corresponding data from the master to these slaves and back again. Therefore, with this adaptation, the maximum DL reply time for a closed communication loop would be $t_{\mathrm{DLReply}_j} \leq 1.5\,\mathrm{ms}$. This is even better than a 1 ms cycle time with the adapted parameter set for the closed-loop optimized timing. Although a shortening of the cycle time would not be feasible in practice because of the assumed subnetworks, this example shows that it would be partially reasonable to consider this timing method as well.

Independent of the timing method required by the application, we could show that our analytical parameter adaptation results in an improved overall timing with reduced transmission latencies. With our introduced adaptation method it will be possible to improve the timing behavior for each communication network. However, due to the variety of dependencies resulting from the application requirements and the considered network structure, this adaptation is very complex. Parameter boundaries in the parameter adaptation of the last subnetwork in the communication cascade might require a complete revision of previous parameter adaptations. Consequently, our analytical adaptation might require much time and effort. Therefore, we would like to introduce a heuristic parameter optimization in the remainder of this chapter.

Table 6.2: *Analytically optimized timing behavior of the exemplary network for the machine tool application scenario.*

		t_{DLRx_j}	t_{DLAct_j}	t_{ULAcq_j}	t_{ULRx_j}	t_{DLReply_j}
Command data optimized	Max [µs]	514.2	1420.1	420.1	950.2	1054.9
	Avg [µs]	161.5	571.6	420.1	925.5	1030.2
	Min [µs]	3.7	420.1	420.1	899.0	1003.7
Actual data optimized	Max [µs]	514.2	564.2	564.2	1739.8	1054.9
	Avg [µs]	161.5	564.2	564.2	1109.1	1030.2
	Min [µs]	3.7	564.2	564.2	688.6	1003.7
Closed-loop optimized	Max [µs]	514.2	564.2	564.2	1970.7	2040.5
	Avg [µs]	161.5	564.2	564.2	1081.7	1151.5
	Min [µs]	3.7	564.2	564.2	933.9	1003.7
Cycle time optimized	Max [µs]	514.2	760.6	84.1	863.0	731.4
	Avg [µs]	161.5	350.6	84.1	428.6	706.7
	Min [µs]	3.7	84.1	84.1	135.7	680.2

6.4 Heuristic Offset Adaptation

As described in [6], we propose mapping the parameter adjustment as a general optimization problem as additional method to improve the overall timing. Optimization problems are a branch of theoretical computer science and part of the computational complexity theory. They are intended to find the minimum or maximum for a given cost function over all possible combination in the solution space, even if the cost function is not continuously differentiable. There are several different complexity classes according to the effort that is required to solve them. For example simple sorting algorithms have quadratic runtime dependency on the number of objects to be sorted [97]. A typical class of optimization problems are NP (nondeterministic polynomial time) problems. Here, polynomial time refers to how quickly the number of operations needed by an algorithm grows relative to the size of the problem.

Well known examples of NP optimization problems are the travelling salesman problem and the knapsack problem [98]. The traveling salesman problem concerns finding the optimal order of visiting different places such that no place is left out or visit twice, start and end of the journey are at the same place and the traveled path is as short as possible. The size of the search space depends exponentially on the number of locations, since in each step all remaining locations can be visited. Within the knapsack problem, objects with a certain value and weight should be packed in such a way that a total weight is not exceeded, but the value is as high as possible. Again, there is a exponential dependence of complexity on the number of possible objects.

The parameter adjustment we presented in the previous section has a similar complex-

ity as the two problems presented before. The dependencies in the adaptation increase with the number of subnetworks, layers, and distributed slaves. Even for our small example, 25 parameter have to be adapted with respect to the overall communication, application requirements, and individual parameter boundaries. In addition, the individual parameters and their relationships with the entire communication chain can influence each other. Even if these parameters took values only in the binary range, there would be already 625 different parameter combinations possible. Considering the five slave timings (DL reception, DL activation, UL acquisition, UL reception, and DL reply) and the 33 slaves in our example, there are in total already 103 125 results requiring comparison. However, instead of binary values, we have to consider real numbers for the parameters resulting in a corresponding higher number of possible combinations. Without giving a particular mathematical proof, which would exceed the scope of this work, we consider the parameter adaptation of cascaded networks as similar to NP-equivalent.

For NP problems there is currently no algorithm known to find an optimal solution in polynomial time. Accordingly, heuristic methods are used in order to achieve sufficiently good results in a relatively short runtime. There are different heuristics, which for example calculate a possible solution or improve given solutions if possible. So-called metaheuristics combine several of these methods and thus enable a combined search for local and global optima. In order to reduce the danger of getting stuck in a local optima, some of these metaheuristics are inspired by natural or evolutionary processes. They employ a set of candidate solutions to find a good result using a trial and error method and then re-adjust the set for another search. By adapting each candidate with respect to the overall best solution found so far, local optima can be left. This may lead to a worse result for a single solution, but the overall solution may approach the global optimum over several iterations. Different metaheuristics pursue different approaches how individual candidate solutions are adapted and changes in the result are permitted. However, quality and runtime of metaheuristics depends on the actual definition and implementation of the individual steps and iterations. Accordingly, they could be used finding sufficiently good solutions, but might be arbitrarily bad compared to an optimal solution. Some heuristics also allow to specify a (previously theoretically determined) optimum and stop accordingly, if the algorithm reaches a certain threshold around this optimum. Thus the quality of the optimization can be improved and defined.

Despite the disadvantage that metaheuristics do not necessarily find the absolute optimum, we will show below that they can be used to simplify parameter adjustment. The selection of the most suitable metaheuristic and the optimization of the procedure itself is less in focus than the consideration of the principle approach.

As example, we will consider the evolutionary metaheuristic Particle Swarm Optimization (PSO) for the parameter adaptation. Compared to other other evolutionary algorithms, it shows a slightly better performance and faster convergence, for some well-known NP optimization problems [99, 100]. PSO imitates the biological swarming behavior for example of a bird flock or fish school, which relay on swarm-intelligence to find an optimal solution. It was initially introduced in [101] and has been continuously discussed and reviewed for NP optimization problems (e.g. [100, 102]). It is defined as

follows:

The metaheuristic consists of a swarm of particles that moves around the defined search space. The search space is given by the parameters to be adapted. A cost function C defines the optimization to be found. The swarm has a certain population size n_{Pop}. Each particle i has a position $\vec{x_i}$ within the search space that represents a candidate solution for the optimum to be found. Thus $C(\vec{x_i})$ represents a result of the optimization, which could be compared over all particles.

Moreover, a particle has a velocity in a certain direction $\vec{v_i}$ that is driven by its own known best position achieved so far $\vec{p_i}$ and the overall best position over all particles of the swarm $\vec{g}$. These parameter influence the velocity and direction $\vec{v_i}$ of the particle, such that it moves to another position with each iteration and slows down as closer it gets to $\vec{g}$.

On its path, a particle i may find an improved position within the search space according to $C(\vec{x_i})$ that correspondingly influences $\vec{p_i}$ and $\vec{g}$. The performance of PSO is controlled by some specific PSO parameterization that defines how $\vec{p_i}$ and $\vec{g}$ influence the velocity $\vec{v_i}$ of an particle. The choice of these parameters is widely analyzed in literature (e.g. [103–106]). The quality and run time of PSO strongly depends on the opulation size n_{Pop} of the swarm and on the cost function C itself. A more detailed descriptions of PSO is shown in appendix A.6.

In the following, we will use PSO with the extended constriction coefficients presented by Clerc et. al. [105] to describe a method for adapting the parameterization of a cascaded network. However, our principle approach could also be adapted to other metaheuristics designed for NP problems. The improvement of our approach would be part of future research.

As analytically shown in Sec. 6.3, the adaptation is feasible at least for small problems. Therefore we should be able to find a parameter set with a close proximity to our calculated adaptation using PSO. In order to do so, we have to find a suitable cost function C that should be minimized by the algorithm. Regarding the overall timing behavior of the cascaded network, one intuitive optimization is the minimization of the last DL reply time t_{DLReply_j}. This timing is based on all previous timings (t_{DLRx_j}, t_{DLAct_j}, t_{ULAcq_j} and t_{ULRx_j}), so that these must also be adjusted to get a good result at the end of the chain. Therefore, a simple cost function could be represented by:

$$C = \max_j \left\{ t_{\mathrm{DLReply}_j} \right\}, \quad \forall j | j \in \{1, \cdots, s_{\mathrm{Max}}\} \tag{6.37}$$

However, because of the cyclic communication behavior and the statically allocated transmission resources, this cost function is a discontinuous function and shows similar results even for larger changes of the timings in between. This lack of graduality increases the likelihood that the optimization algorithm will get stuck in a local minima instead of converging to the global minima. Moreover, such a simple cost function would not represent the importance of the intermediate timing results that are the basis for the individual timing methods an application may require. Therefore, it is also reasonable to extend C by the other timing parameters that it is build on. An extended cost function

could then look as follows:

$$\begin{aligned} C =& g_1 \cdot \max_j \left\{ t_{\mathrm{DLRx}_j} \right\} + g_2 \cdot \max_j \left\{ t_{\mathrm{DLAct}_j} \right\} + g_3 \cdot \max_j \left\{ t_{\mathrm{ULAcq}_j} \right\} \\ & + g_4 \cdot \max_j \left\{ t_{\mathrm{ULRx}_j} \right\} + g_5 \cdot \max_j \left\{ t_{\mathrm{DLReply}_j} \right\}, \quad \forall j | j \in \{1, \cdots, s_{\mathrm{Max}}\} \end{aligned} \tag{6.38}$$

Here, the parameter g could be used for a proper weighting of the individual timings in order to emphasize the specific application requirements.

Nevertheless, the cost function will remain discontinuous, even if the discontinuities are lower. Therefore it might be necessary to further simplify the problem in order to find a sufficiently good solution with a heuristic optimization. This could be achieved by combining or reducing variables that have to be adapted. It would reduce the search space and enable a faster convergence of the search algorithms.

By considering the data transmission trough the cascaded network, we could for example simplify the adaptation of the DL offset time of subnetworks on the highest level. Instead of heuristically searching for individually adapted offsets, we could directly link their offset to the GSP such that the last transmitted data packet could just be processed right before the GSP (cf. Eq. (6.27)). This represents the largest possible offset time for all DL transmission that enable the corresponding date to be received and processed before and activated right at the GSP. If data are still unavailable for these allocated resources, they need to be stalled until the next subsequent GSP anyhow. Accordingly, the DL activation time t_{DLAct_j} remains unchanged. Therefore it would not change the overall latency reduction if the offset for the corresponding subnetworks would be different. Even though this might increase the DL reception time t_{DLRx_j} for some slaves, we could reduce the number of parameters to be adapted by the corresponding number of subnetworks on the highest level of the cascaded network.

Similarly, instead of having a flexible UL offset for the subnetworks on the highest level, we can directly link this offset to the GSP as well. By adapting the UL offset according to Eq. (6.18), we can guarantee for these subnetworks that UL data will be transmitted at the earliest time after they were acquired. This allows again a reduction of the search space according to the number of subnetworks on the highest level of the cascaded network.

In order to test the feasibility of the heuristic optimization for our parameter adaptation, we implemented PSO in *MATLAB*. We run PSO with a swarm size of $n_{\mathrm{Pop}} = 500$ particles and 1000 iterations. The start position of any particle was randomly generated over the entire search space. We implemented PSO with the extended constriction coefficients φ_1, φ_2 and κ presented in [105] to iteratively derive the velocity of the particles and empirically set them to $\varphi_1 = 1.55$, $\varphi_2 = 1.25$ and $\kappa = 1$. We used the cost function shown in Eq. (6.38) and the above mentioned search space reduction. The individual timing parameters within the cost function are weighted according to the analyzed required timing behavior of the application. The computation was done on a standard notebook without any parallelization or performance improvements. Running the algorithm under the given condition takes approximately 30 min.

Tab. 6.3 presents the best parameterization for our machine tool application scenario requiring the closed-loop optimized timing behavior. For this we ran the algorithm ten

Table 6.3: *Heuristically optimized parameterization of the machine tool application scenario with closed-loop optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	933.9	5123.0	509.8	508.2	506.6
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	945.2	86.9	614.6	614.6	614.6	614.6
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	323.5	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			1000.0			
t_{GSP} [µs]			566.7			

times, where the results were relatively similar each time. The resulting timing behavior is represented in Tab. 6.4. The heuristic optimization for the closed-loop timing is very similar to the analytical optimization. With this parameterization, the last DL data are received ≈2.5 µs later. In fact, the average DL data reception time is even ≈52.7 µs later. This results from the direct adaptation of the DL offset of subnetworks on the highest level to the GSP. However, all data are still received in time. Since the GSP is also scheduled ≈2.5 µs later, this results in a little performance degradation because it influences the DL activation and UL acquisition time. The UL offset of N_1 is also slightly larger. Therefore, the UL reception time is ≈15.0 µs later compared to the analytical optimization. The resulting frame structure is very similar, with minimal degradation compared to Fig. 6.5. As we are more focused on the feasibility of the heuristic optimization, the small performance degradation is quite acceptable, especially since the overall timing behavior remains the same as analytically found.

The heuristically optimized parameter sets for the remaining timing methods can be found in Tab. A.8 to A.10, whereas the resulting timing behavior is presented in Tab. 6.4 as well. With this parameter adaptation, their performance is also very similar compared to our analytical adaptation presented in Tab. 6.2. For all timing methods, the DL reply time t_{DLReply} is identical to our analytical adaptation. This shows that the heuristic adaptation results in the same cyclic behavior regarding the responses from the master to the actual data of its slaves. Also the DL activation and UL acquisition time (t_{DLAct} and t_{ULAcq}) are quite similar compared to the results of our analytical adaptation. This indicates that the heuristic approach is capable of finding a similar GSP as our analytical adaptation.

A general degradation can be seen for the DL reception time t_{DLRx} for all application

Table 6.4: *Heuristically optimized timing behavior of the exemplary network for the machine tool application scenario.*

		t_{DLRx_j}	t_{DLAct_j}	t_{ULAcq_j}	t_{ULRx_j}	t_{DLReply_j}
Command data optimized	Max [µs]	1374.8	1424.5	424.5	955.6	1054.9
	Avg [µs]	309.6	576.0	424.5	930.9	1030.2
	Min [µs]	3.7	424.5	424.5	904.4	1003.7
Actual data optimized	Max [µs]	520.2	570.2	570.2	1748.0	1054.9
	Avg [µs]	215.5	570.2	570.2	1117.2	1030.2
	Min [µs]	3.7	570.2	570.2	696.8	1003.7
Closed-loop optimized	Max [µs]	516.7	566.7	566.7	1985.6	2040.9
	Avg [µs]	214.1	566.7	566.7	1096.6	1151.5
	Min [µs]	3.7	566.7	566.7	948.8	1003.7
Cycle time optimized	Max [µs]	715.5	765.5	89.0	1000.0	731.4
	Avg [µs]	292.4	355.5	89.0	565.4	706.7
	Min [µs]	3.7	89.0	89.0	272.3	680.2

timings. Especially for the command data optimized timing the heuristic adaptation results in a maximum of $t_{\mathrm{DLRx}_{\mathrm{Max}}} \approx 1374.8\,\mu\mathrm{s}$, which is $\approx$860.6 µs later than with our analytical adaptation. However, this results from our approach to reduce the search space by linking the DL offset of the subnetworks on the highest level of the cascaded network directly to the GSP. Therefore, the heuristic could not individually adapt these offsets. As a result, the command data for the slaves in subnetwork N_6 were transmitted with one cycle delay in our example. In our analytical adaptation, we set the DL offset for this subnetwork behind the GSP. As it is not of importance for this timing method if not all DL data are available at the GSP, this does not influence the overall required timing behavior. Instead, it enables the corresponding DL data to be transmitted within the same cycle. Never the less, due to the minor importance of the DL reception time for the command data optimized timing, we can estimate the result of the heuristic as sufficiently good.

The strongest overall degradation compared to our analytical adjustment could be seen for the cycle time optimized timing behavior. Even though PSO also finds the shortest possible cycle time of $t_{\mathrm{Cycle}} = 676.5\,\mu\mathrm{s}$, other parameters are not ideally adapted. As a result, the maximum DL reception time equals $t_{\mathrm{DLRx}} = 715.5\,\mu\mathrm{s}$ and is $\approx$201.4 µs later compared to our analytical adaptation. Again, this results from our approach of linking of the DL offset for some subnetworks to the GSP, since some DL data are transmitted with one cycle delay. In addition, a degradation of the UL reception time t_{ULRx} shows also a less favorable adaptation of the corresponding UL offset. However, only the UL offset for subnetwork N_1 is affected. For all other subnetworks similar parameters were found as with our analytical adaptation. Unfortunately, the subnetwork

N_1 has the greatest overall influence on the timing, since all data packets are equally affected by a delay here. This degradation might results from the unbounded relations of the cycle time optimized timing. Since none of the individual intermediate results are based upon each other (for this method it is not necessary that either command or actual data have to be received and processed before the subsequent transmission), it is possible that larger local optima exist causing the heuristic to get stuck. Accordingly, it might be necessary to further adapt cost function in order to consider all individual values equally, but also to take the overall timing into account. Nevertheless, it can be assumed that sufficient good results can be found with the heuristic, since, despite the slight deterioration, the overall temporal behavior with respect to the cyclic data exchange is comparable to that of the analytical adaptation.

The scalability of the heuristic approach for larger cascaded networks has to be further analyzed. Since it requires already 30 min to run PSO under the described conditions for our example, this approach might not necessarily be appropriate for larger cascaded networks. However, it might be used to generate a parameterization that is at least better than an unadapted parameter set and could further be used as first start for the analytical approach and thus reduce the corresponding effort. Moreover, it might be also possible to further improve the heuristic approach itself. For example, the iterative calculation of the particle positions can be parallelized, which would significantly reduce the required time. Other improvements, such as specific advancement of the cost function, more appropriate parameter weighting, and a more advanced PSO parameterization, might further improve the performance of the heuristic adaptation. It is also possible that other metaheuristics are more suitable for our parameter adaptation and shorten the required calculation time. These possibilities indicate that a heuristic optimization with suitable adaptations might also be useful for larger cascaded networks. However, the actual proof is beyond the scope of this work.

Regarding our machine tool application example, both the analytical and the heuristic optimization reveal the importance of adequate adapted offset parameters for the individual subnetworks according to the overall communication. However, both methods are based on static resource allocation of the corresponding DL and UL frames. These resources are so far allocated according to the numbering of the slaves within each individual subnetwork. In the reminder of this chapter we will further analyze this resource allocation method as additional approach to further reduce transmission latencies.

6.5 Latency Reduced Resource Allocation

As mentioned above, industrial communication networks typically employ static resource allocation for transmission of RT data. These resources are typically scheduled offline for the corresponding DL and UL frames and the schedule is distributed between the master and all the slaves. According to the network protocol, the resources can be allocated arbitrarily for individual data packets within specific boundaries. Since cascaded communication networks have received little consideration in the literature so far, there are basically no specific adaptations that consider the entire communication flow. Accordingly, individually composed frames are transmitted in each subnetwork.

This might increase the overall transmission latency, as introduced earlier. Therefore, in this section we will propose a latency reduced resource allocation scheme for arbitrary cascaded communication networks, as we have introduced in [8].

As presented in Chapter 3, typical wired industrial communication networks provide a one-dimensional resource space for data transmission. DL and UL frames are typically transmitted consecutively. This limits possible optimizations. One of the advantages of wireless networks is that they typically provide a wider resource spaces due to multi-dimensional resource allocation. A multiplexing between several communication participants and duplexing between DL and UL typically provides a two-dimensional resource space for individual data transmissions.

The duplexing method used for the considered subnetworks can influence the transmission latency. The most commonly used methods so far are frequency division duplexing (FDD) and time division duplexing (TDD). FDD enables a simultaneous transmission and reception on separate channels. Therefore, the latency might be reduced due to a parallel DL and UL transmission. However, a guard band is required to avoid interferences between transmitter and receiver that reduced the spectral efficiency. Moreover, a diplexer and filtering is required to separate the two signals, which increases the effort for the radio modules. TDD switches between DL and UL on the same channel. Therefore, channel propagation is the same in both directions which enables transmit and receive to use on set of parameters. The switching causes additional latencies since it has to be ensured by guard intervals between the individual transmissions. As the guard bands for FDD, these guard intervals reduces the efficiency. In contrast, the hardware costs are lower, since the same frequencies are used for DL and UL transmission. In TDD, an unbalanced amount of DL or UL traffic could be easily adjusted by the transmission intervals. In FDD, this is only possible to a limited extent. TDD requires accurate synchronization with respect to DL and UL transmission times. Otherwise, cross slot interferences may occur if neighboring base stations use different assignments while sharing the same channel.

Both duplexing strategies provide different advantages and disadvantages. The preferred method has to be chosen, among other things, according to the application requirements, implementation convenience and feasibility, environmental conditions, and available hardware. However, since latency is one of the key aspects for industrial applications, we propose using a wireless network that employs FDD for duplexing DL and UL in a cascaded network. This reduces transmission latencies and allows higher adaptability to other subnetworks. code division duplexing (CDD) also enable a parallel transmission of DL and UL. However, as described in Sec. 3.2, this has not yet been implemented in practice and is therefore not considered here in detail.

The two-dimensional resource space that is created by using both time and frequency or code domain could be used in various ways to transmit individual data packets. For this purpose, we have to assign the data to the different resource elements using a mapping algorithm. Depending on the assignment, the individual data packets get different properties, which are particularly expressed in their vulnerability to errors. There are four different resource allocation schemes possible, as discussed in [8] and [107]. These are:

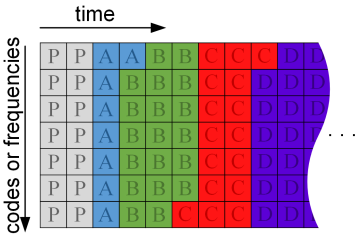

Figure 6.6: *Sequential resource allocation scheme.*

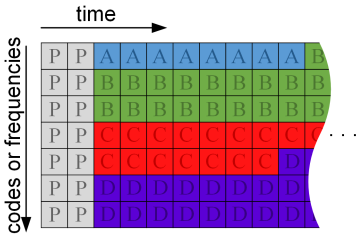

Figure 6.7: *Parallel resource allocation scheme.*

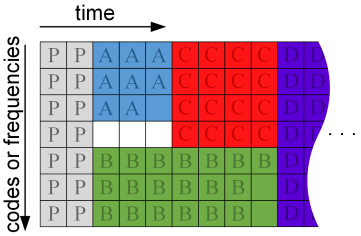

Figure 6.8: *Rectangular resource allocation scheme.*

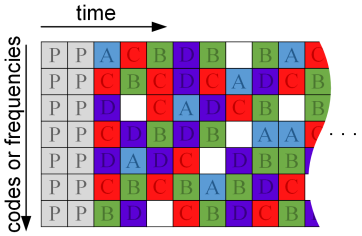

Figure 6.9: *Pseudo-random resource allocation scheme.*

- sequential,
- parallel,
- rectangular, or
- pseudo-random.

In Fig. 6.6 to 6.9 we illustrate them for four data packets A, B, C and D with different sizes (reflected by the different number of REs required). In the following, we will analyze the allocation schemes with respect to general aspects and technology restrictions.

Sequential: In the sequential resource allocation scheme, individual data packets are allocated consecutively following the time domain. All REs for frequency or code domain are deployed until the data packet is fully allocated. Therefore, only a sequential processing at the transmitter and receiver is required. However, a receiver has to analyze the entire code or frequency range.

Parallel: In the parallel resource allocation scheme, all data packets are assigned to different frequencies or codes until they are fully allocated. Therefore, they can be transmitted at the same time. A receiver only has to observe a certain code or frequency range after the detection of the preamble. This might be beneficial for power saving options. However, at least the master requires simultaneous processing units for the parallel processing of the data.

Rectangular: The rectangular resource allocation scheme partially combines parallel and sequential resource allocation. This enables an individual combination of transmission duration and required codes or frequencies for each data packet. However, unused resource elements might emerge with an unfavorable distribution over the different domains (exemplified for data packed A and B in Fig. 6.8).

Pseudo-random: In the pseudo-random resource allocation, data are distributed arbitrarily over the resource space using a pseudo-random assignment. This results in partially disjoint REs per data packet spread over each domain. This can make unwanted spying more difficult, which represents a certain inherent security.

All allocation schemes have their individual advantages. We qualitatively compare nine central properties by three grades (good, insufficient, and partial) in Tab. 6.5. Since we consider the two-dimensional resource space mainly for wireless networks, this comparison also contains an extract of their dependency on the physical layer (PHY). Even though this is not the focus of this work, we need to consider this dependency in order to choose a suitable network for specific applications.

One of the most important aspects is the robustness for symbol and frequency or code errors. This represents the influence of transmission error on PHY level. If disturbances occur only in a small frequency range but for long time, a sequential resource allocation is preferable. In this case, only a small number of REs of individual data packets are corrupted that could probably be restored with sufficient error correction. Parallel resource allocation would not be advantageous for such conditions, since individual data packets would be affected completely or in large parts. This scheme is preferable, if disturbances occur over large frequency or code ranges. Since the rectangular scheme combines both parallel and sequential allocation, its susceptibility to errors depends on the actual implementation. The pseudo-random allocation if affected by both types of errors in the same way, since the REs for data packets are equally distributed over all domains. Accordingly, the preferable allocation scheme strongly depends on the actual environmental conditions of the PHY layer. However, the evaluation of the PHY in industrial environments is an individual field of research on its own and not further evaluated in this work. Exemplary measurements are for example presented in [108] for Orthogonal Frequency Division Multiple Access (OFDMA) and in [109,110] for a Parallel Sequence Spread Spectrum (PSSS) based system. They reveal uniformly distributed bit errors in both domains and present the likelihood of burst errors. Both could vary depending on the respective scenario and environmental conditions. Therefore, related to the PHY based criteria, each applications has to be analyzed individually. Accordingly, no particular allocation scheme will always be most efficient.

Besides robustness, transmission latency is one of the most important aspect for industrial communication. The shortest transmission latencies can be achieved using the sequential resource allocation. Since data packets are spread over a maximum of frequencies or codes, the temporal spread is minimal. The opposite applies for the parallel and pseudo-random resource allocation, which results in maximized transmission latencies for each data packet. The rectangular resource allocation allows a certain adaptation and depends on the actual implementation.

In addition to the robustness against specific errors and individual transmission laten-

cies, flexibility in data transmission is another key aspect. Especially considering future industrial communication that requires transmission of different traffic types in parallel. This requires the individual adaptation of each transmission. Here, the rectangular allocation scheme offers the highest flexibility in individually adapting the transmission. A data packet can be spread over any domain, allowing long transmission times using only few codes or frequencies respectively short transmissions using a maximum of frequencies or codes. With the sequential resource allocation, at least the order of the data packet transmission could be chosen, allowing at least some flexibility. With the parallel and pseudo-random allocation, each transmission is spread over a maximum of time. Accordingly, there is no possibility to individually adapt the transmission time.

Another important factor is the resource utilization. As transmission resources are limited, it should be preferable to fast as few as possible. This is the case for the sequential and parallel resource allocation, since for both the corresponding REs are filled sequentially without gaps. In case of rectangular resource allocation, individual REs might remain unused according to the data packet size and the number of REs of the corresponding rectangular (cf. Fig. 6.8). The resource utilization for the pseudo-random allocation again depends on the actual implementation. However, since the resource are assigned randomly, an efficient assignment could not be guaranteed.

Since individual REs belong to different data packets for different receivers, it is important to distribute the allocation schedule. For the sequential and parallel resource allocation this requires only little overhead. In both cases only the first allocated RE and the number of required REs has to be shared. Based on the maximum spread over the corresponding domain, the remaining allocated REs can be determined. For the rectangular allocation the exact extent of the rectangle over the corresponding time and frequencies or codes has to be shared in addition to the start RE. Thus, the signaling overhead is slightly higher. For the pseudo-random approach, the entire mapping function has to be shared. This requires the largest overhead compared to the other allocation schemes.

The pseudo-random resource allocation is the only scheme offering a certain security functionality by providing inherent encryption. As long as the mapping function is handled secretly, an attacker could hardly identify individual data packets. However, data integrity and message authentication still has to be checked independently.

Besides functional aspects, we can also compare implementation aspects that correspond to certain hardware expanses. Using a sequential resource allocation, all data packets are received sequentially. Thus, they can be processed sequentially using a single processing unit and a buffering would not necessary be required. For all other schemes, two or more data packets are transmitted and received in parallel. These have to be either buffered in order to be processed sequentially or a number of processing units is required that correspond to the maximum number of data packets arriving at the same time. In both cases, additional hardware is required, which increases the corresponding expanses.

Most of the aspects described before strongly depend on the actual PHY conditions on premise. However, we have to focus on low transmission latencies as this is one

Table 6.5: *Comparison of two-dimensional resource allocation schemes.*

Allocation scheme	Sequential	Parallel	Rectangular	Pseudo-random
Symbol error robustness	×	✓	◯	✓
Code or frequency error robustness	✓	×	◯	✓
Individual latency adaption	✓	×	◯	×
Individual cycle time adaption	◯	×	✓	×
Flexible resource allocation	◯	×	✓	×
Resource utilization efficiency	✓	✓	×	×
Schedule distribution efficiency	✓	✓	◯	×
Inherent security	×	×	×	✓
Hardware expense	✓	×	×	×

✓- good × - insufficient ◯ - partial

of the most important aspects for the industrial closed-loop application we focus on in this work. Consequently, in the following, we develop a latency optimized resource allocation for an individual network and extend this resource allocation for subnetworks in a cascaded network, as we presented it in [8].

As introduced in our offset adaptation for the closed-loop timing, we achieve the lowest possible latency when the time between the start of the data transmission and the complete reception and processing of the last data is minimized for both DL and UL. Moreover, regarding an individual network, the GSP should ideally be scheduled immediately after the DL transmission and the processing of the corresponding command data. In this case, all slaves can acquire and process their actual data with minimal delay after they activate their command data and the pending UL transmission can be initiated immediately.

With respect to our introduced network model, the DL data reception time of an individual command data packet d_{DL_j} is calculated with:

$$t_{\mathrm{DLRx}_j} = t_{\mathrm{DLTx}_j} + \Delta t_{\mathrm{DLData}_{i,j}} + \Delta t_{\mathrm{DLTran}_{i,j}} + \Delta t_{\mathrm{DLProc}_j} \tag{6.39}$$

Here, the transmission time t_{DLTx_j} can be calculated with:

$$t_{\mathrm{DLTx}_j} = t_{\mathrm{DLAlo}_0} + \Delta t_{\mathrm{DLAlo}_j} \tag{6.40}$$

As introduced in Sec. 3.3, we can neglect the transmission time due to its low influence with respect to the short distances. The earliest possible allocation time t_{DLAlo_0} within a network N_i strongly depends on its respective offset t_{DL_i} and the corresponding RT data offset $\Delta t_{\mathrm{DLRTOfs}_i}$. Both were subjects of our previously introduced offset adaptation and are therefore considered as minimized and constant during run time in the following. Moreover, t_{DLAlo_0} depends on the protocol dependent overhead Δt_{OH_i} that is

also constant. Accordingly, the latency of an individual data packet d_{DL_j} depends on:

$$t_{\mathrm{DLRx}_j} \sim \Delta t_{\mathrm{DLAlo}_j} + \Delta t_{\mathrm{DLData}_{i,j}} + \Delta t_{\mathrm{DLProc}_j} \tag{6.41}$$

Generalized, the overall DL latency results in particular from the last command date reception time. Therefore, we have to find a schedule, that minimizes $\max_j \left\{ t_{\mathrm{DLRx}_j} \right\}$ over all slaves S_j in the corresponding network.

Referring to Eq. (6.41), the transmission time $\Delta t_{\mathrm{DLData}_{I,j}}$ mainly depends on the implemented PHY allocation scheme. Especially for the two-dimensional resource space, the different allocation schemes result in different timing behavior as described above. With the parallel and pseudo-random allocation, the transmission of an individual data packet is distributed over the entire DL frame duration. Accordingly, t_{DLRx_j} is always maximized such that also the overall DL latency is maximized. With a sequential resource allocation, an individual data packet is distributed over the maximum number of codes or frequencies and within the minimum time. Therefore, the individual data packet transmission time is minimized and enables a minimization of the overall DL latency. The rectangular allocation scheme also enables minimizing $\max_j \left\{ t_{\mathrm{DLRx}_j} \right\}$ and transferring individual data packets with fewer codes or frequencies but over longer times. However, this might result in a larger number of unused resource elements. Therefore, considering especially industrial closed-loop application where all RT data share the same latency requirements, the sequential resource allocation is to be preferred.

The same applies for the one-dimensional resource allocation. It can similarly be shown that a sequential resource allocation results in an lower transmission duration for individual data packets compared to other allocation schemes, such as round-robin or pseudo-random.

In addition to the PHY based behavior, $\Delta t_{\mathrm{DLData}_{i,j}}$ and probably also $\Delta t_{\mathrm{DLProc}_j}$ depend on the size of the corresponding data packet. The larger a data packet is, the more resource elements are required for the transmission and the more processing is required for the activation. Therefore, the transmission latency of larger data packets is automatically higher compared to shorter data packets. We can compensate this by allocating larger packets as early as possible in the DL frame. Hence, the last transmitted data packet requires the lowest number of resource elements and processing, which reduces the overall DL transmission latency. With respect to Eq. (6.41), we have to postulate that the DL allocation offset should be ($\overset{!}{\sim}$) inversely proportional to data packet size: $\Delta t_{\mathrm{DLAlo}_j} \overset{!}{\sim} \mathrm{size}(d_{\mathrm{DL}_j})^{-1}$.

Unlike the command data in the DL transmission, all actual data for the UL transmission are acquired in parallel at the GSP by all slaves. After acquisition, they need to be processed before they can be transmitted. This influences the corresponding resource allocation:

$$t_{\mathrm{GSP}} + \Delta t_{\mathrm{ULProc}_j} \leq t_{\mathrm{ULTx}_j} = t_{\mathrm{ULAlo}_0} + \Delta t_{\mathrm{ULAlo}_j} \tag{6.42}$$

Moreover, the processing time again depends on the packet size. Therefore, we can schedule smaller packets already for transmission, while larger packets are still being processed. Accordingly, we can postulate that the UL allocation time should be proportional to the data packet size: $\Delta t_{\mathrm{ULAlo}_j} \overset{!}{\sim} \mathrm{size}(d_{\mathrm{UL}_j})$.

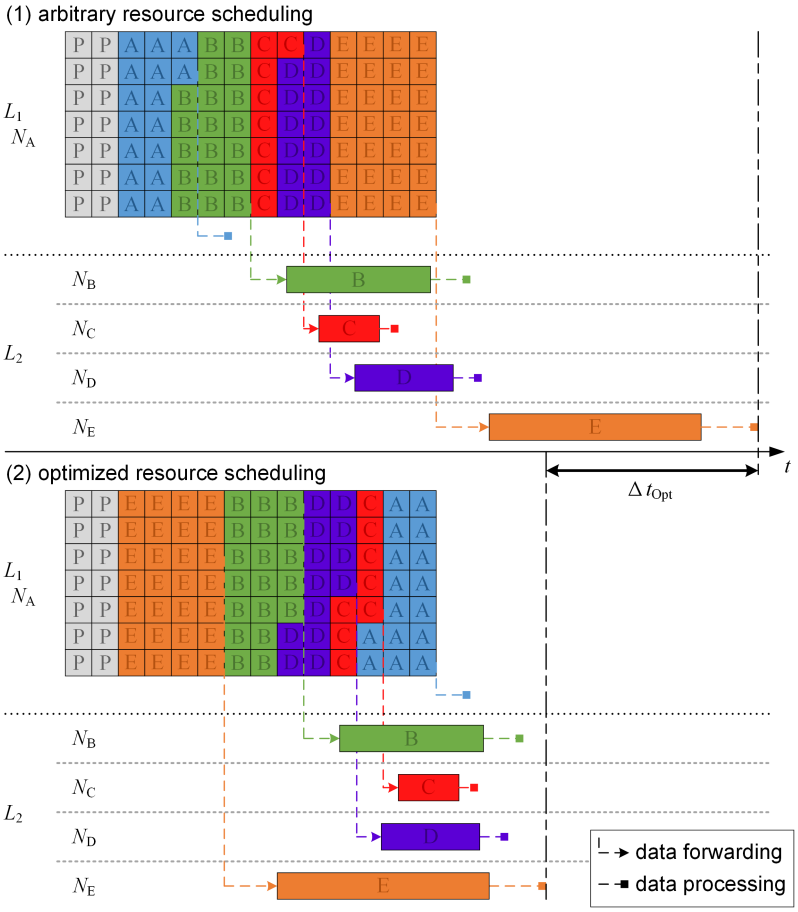

Figure 6.10: *Comparison of (1) arbitrary and (2) optimized resource scheduling for DL transmission in an exemplary cascaded network with two levels.*

Finally, according to Eq. (6.42), we have to schedule the GSP as early as possible. A discussion on the timing of the GSP can be found in our discussion of offset adaptation (cf. Sec. 6.3).

This scheduling thus represents the latency optimized resource allocation for an individual communication network in an industrial closed-loop application. For cascaded networks, we now have to generalize this analysis for the data transmission through several subnetworks. Therefore, in addition to the individual packet size, we have to take the destination of a packet also into account.

In previous sections, we exemplified possible DL data flows in cascaded networks. In each subnetwork, either individual data packets have to be forwarded to other subnetworks or they belong to slaves located in this subnetwork. The transmission latency of data packets that have to be forwarded are increased by additional conversion and transmission times. Therefore, these data will have larger latencies compared to data packets for slaves within the corresponding subnetwork. Consequently, we should allocate DL data packets that have to be forwarded first.

This is exemplified in Fig. 6.10. Here, we compare an arbitrary resource scheduling and our described optimized scheduling for a cascaded network with two levels. We illustrate the transmission of the five data packets A to E according to the five subnetworks N_A to N_E. All data packets are transmitted in subnetwork N_A on level L_1. The packets B to E are forwarded to L_2, and only the data packet A can be directly processed after being transmitted. The time interval Δt_Opt shows the potential latency reduction that results from our optimized resource scheduling.

Moreover, we can distinguish data packets that we have to forward by the level of the cascaded networks their corresponding slaves are located in. The higher this level, the more often the respective data packet has to be forwarded. Therefore, we can postulate for the DL transmission that data packets should always be allocated according to higher destination level. For slaves on the same level, we have to consider the previously introduced order according to the packet size. Consequently, we developed the latency reduced scheduling algorithm presented in Alg. 2, which is assumed to be completely correct (cf. appendix A.4).

Algorithm 2: Latency reduced DL resource allocation for arbitrary subnetworks N_i in cascaded network.

Input: List V_i of slaves S_j thats DL packets d_{DL_j} have to be scheduled in N_i
Output: Latency optimized schedule for N_i
Condition: $\sum_j \mathrm{size}(d_{\mathrm{DL}_j}) \leq n_{\mathrm{RE}_i}, \quad \forall j | j \in \{V_i\}$

```
1  begin
2      Δt_Alo ← 0
3      c_off ← 0
4      while V_i ≠ {} do
5          L_x ← max_j {L(S_j)},   ∀j|j ∈ {V_i}
6          Create list V_i' with slaves S_j of V_i on level L_x
7          while V_i' ≠ {} do
8              Find j ∈ {V_i'} such that max_j {size(d_DL_j)}
9              Allocate all REs for d_DL_j starting at (t_Alo_i,0 + Δt_Alo, c_off)
10             Increase (Δt_Alo, c_off) to the next unused RE
11             Remove S_j from V_i and V_i'
```

For the UL transmission, on the other hand, we can use the resource allocation to compensate larger transmission and processing latencies. Data packets that are received from subnetworks on higher levels in order to be forwarded to the master in the root subnetwork might be available later than data that are generated within the corresponding subnetwork. Therefore, for the UL transmission, we can postulate that data packets should always be allocated according to lower source level. For this purpose, the algorithm presented in Alg. 2 could be easily adapted to generate an UL schedule by changing the maximum search in Line 4 and 7 to a minimum search.

In order to verify our previously found results, we deploy Alg. 2 to our machine tool application scenario introduced in Sec. 2.4. As a result, in the DL frame of N_1 we start with the transmission of the data packets for the subnetworks N_3 to N_6 in reverse

Table 6.6: *Resource scheduling and offset adapted parameterization of the machine tool application scenario for closed-loop optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	907.1	478.5	450.0	364.5	250.5
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	898.2	60.1	580.1	580.1	580.1	580.1
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	311.5	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]	1000.0					
t_{GSP} [µs]	532.2					

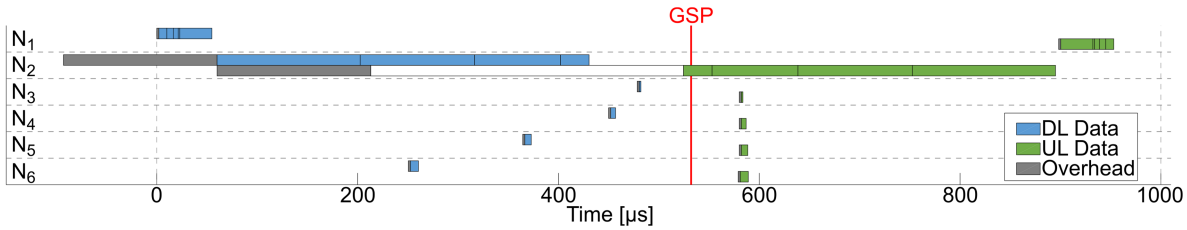

Figure 6.11: *Frame structure of exemplary cascaded network for machine tool application scenario using the parameterization of Tab. 6.6.*

order. After that, we allocate the data packets for the slaves S_1 to S_{20}. Since all of these packets are of the same size, we allocate them according to the number of the slaves. The DL frame of N_2 is only used to forward the data packets belonging to the subnetworks N_3 to N_6. These corresponding resources are again allocated in the reverse order. The resource allocation for the DL frames of N_3 to N_6 is not changed by our algorithm since all individual data packets that have to be transmitted are of the same size. The UL frames of all subnetworks are allocated in the inverse order compared to the DL frames.

Due to the changed frame structure, we have to readapt the offset parameterization once again in order to provide comparable results. We used the analytical optimization method presented in Sec. 6.3. The parameterization that results from the offset adaptation with latency optimized resource allocation can be found in Tab. 6.6. The corresponding frame structure is depicted in Fig. 6.11, whereas the timing behavior is presented in Tab. 6.7.

Table 6.7: *Comparison of timing behavior resulting from different parameter sets for the machine tool application scenario considering requiring closed-loop optimized timing.*

		t_{DLRx_j}	t_{DLAct_j}	t_{ULAcq_j}	t_{ULRx_j}	t_{DLReply_j}
Offset adapted parameterization	Max [µs]	514.2	564.2	564.2	1970.7	2040.5
	Avg [µs]	161.5	564.2	564.2	1081.7	1151.5
	Min [µs]	3.7	564.2	564.2	933.9	1003.7
Resource scheduling and offset adapted parameterization	Max [µs]	482.2	532.2	532.2	1938.7	2022.9
	Avg [µs]	164.0	532.2	532.2	1049.7	1151.5
	Min [µs]	24.7	532.2	532.2	901.9	1010.2

For comparison, we have also listed the timing behavior that we found for offset adapted parameterization with an arbitrary resource allocation once again. Changing the order of the transmitted data packets according to their destination level results in a further latency reduction of 32 µs on average. Especially due to the earlier scheduled GSP, it is possible to activate all command data and to acquire all actual data earlier. This continues for UL reception at the master without any further latency savings. In fact, the improved timing behavior results already from the improved DL schedule. The 32 µs equals exactly the time required to transmit data packets for the slaves on L_1 that are shifted to the end of the frame. Nevertheless, especially the minimum and average DL reception time t_{DLRx_j} is higher than without the optimized resource allocation. This corresponds to the 20 slaves on L_1 and their contribution to the average DL reception time.

Changing the resource allocation order according to the size of the individual data packets does not improve the timing behavior any further for our machine tool application scenario. On the one hand side, we assumed the same conversion and processing time for all data packets independent of their size as sufficient computation performance will probably be available. In case of an actual dependency, the optimized resource allocation according to the data packet size would also further contribute to an improvement of the timing behavior. On the other hand side, our considered intermediate subnetwork on L_2 has a significantly lower data rate compared to the subnetworks on L_1 and L_3. Therefore, incoming data packets have to be stalled partially before we can forward them with the corresponding resource elements. The latency caused by these retards exhausts the savings due to the optimized resource schedule.

As a consequence, the overall communication behavior of our application scenario could not be further improved such that more data packets could be transmitted from the master to the slaves and back again within the assumed cycle time of 1 ms. Still, we were able to show the advantages of a resource allocation that is adapted according to the data flow in cascaded networks. The *ParSec* network we assumed as intermediate subnetwork represents a bottleneck that does not allow further latency reduction by the considered optimization methods under the given boundary conditions. Nevertheless,

it allows much better real-time performance compared to other commercially available wireless systems at the moment. Therefore, we present a demonstrator implementation of a cascaded network build on *Sercos* and *ParSec* in the subsequent chapter.

6.6 Discussion: Complex Dependencies

In this chapter, we focused on the parameterization of individual subnetworks in a cascaded network in order to reduce transmission latency. Due to their concatenation within the data transmission process, each subnetwork should not only be adapted on its own. The parameterization of one subnetwork influences data transmission of subsequent or previous subnetworks. Therefore, an adapted parameter set with respect to the overall communication must be found.

Several of these parameter cannot be chosen arbitrarily. Parameters might be restricted by protocol defaults or application requirements. They can statically define a specific parameter or set upper and lower bounds for it. These boundaries have to be known and taken into account in order to find a well-adapted parameter set.

We present several parameter boundaries related to our considered subnetworks *Sercos* and *ParSec*. Thus we have shown that the boundaries can be very different according to the chosen network protocols. Moreover, we have also identified several parameter boundaries resulting from application requirements. Some of them specify protocol specific parameter boundaries even further. However, there are probably several more parameter boundaries and dependencies than those presented by us. Therefore, because of the already huge complexity, it is necessary to focus on those parameters and their respective parameter boundaries that are required for the specific goal of adaptation.

Since we focus on latency reduction in cascaded data transmission, we could specify the offset parameter of individual subnetworks as some of the most important factors to be adapted. Therefore, we presented two methods to adapt them using both our introduced networks model (cf. Fig. 3.1) and the corresponding algorithm for the timing analysis (cf. Alg. 1). The first method was an analytical parameter adaption. Therefore, we adapt the individual communication flows successively according to the application requirements. This ensures at least a partial improvement. However, if parameter boundaries are violated, previously adapted parameters have to be reviewed iteratively. Therefore, this method is quite complex already for small cascaded networks.

For our machine tool application example, we have shown that the timing behavior of the cascaded network could be improved massively. Regarding our example with the closed-loop application timing, we were able to reduce the end-to-end transmission latency from master to the slaves and back again by 50 %. Although the most accurate results can be obtained by this method, it is easy to overlook potential improvements because of the complexity involved. This is particularly to be expected for larger networks, since it is not always directly obvious whether a potential improvement actually has a positive effect on all subsequent parameters.

As second approach to adapt the offset parameterization we have introduced a heuristic method to find an adequate parameter set. Therefore, the adaptation has to be considered as a minimization problem. We have proposed using a combination of the weighted data reception and processing times of the individual slaves to create a dedicated cost function. This could subsequently be solved by some well established metaheuristics. We have demonstrated this adaptation using a Particle Swarm Optimization (PSO) implemented in *MATLAB*. It results mostly in very similar results compared to

our analytical adaptation. Therefore, it seems suitable for finding an advanced parameter set for a latency reduced data transmission within a cascaded network. However, this result strongly depends on the chosen cost function and the parameterization of the optimization algorithm itself.

Regarding the cost function, we have introduced the problem of discontinuities due to the cyclic communication behavior and the statically allocated transmission resources. Since it requires a constantly differentiable function to analytically find an optimum, this justifies again the use of a heuristic approach in order to find an adapted parameterization. Moreover, we have shown that the discontinuities could be reduced by taking more intermediate timing results for the cost function into account. As a result, in our example we have to use a cost function with a dedicated parameter weighting for each of the four application timing requirements. Regarding future work, it might be beneficial for a comprehensive analysis to have a universal cost function for all timing requirements. However, we have focused on a feasibility analysis in this work. In this scope the presented results could be considered as sufficient.

The same applies for the parameterization of the PSO algorithm. This parameterization is an individual field of research on its own. Finding the best PSO internal parameterization is even partially solved heuristically for some specific problems. Therefore, in this work we have focused on an empirical set of constriction coefficients presented in [105]. Nevertheless, an ideal parameterization of the metaheuristic might further improve on the adaptation result such that it would be identical with the analytical adaptation.

Also the performance of the implementation could probably be improved. The calculation of the cost function for the individual particles could be parallelized. This would reduce the runtime such that more particles or iterations could be simulated in the same time, which might also improve the final result. It is also possible, that other metaheuristics than PSO might be better suited for the problem of the parameter adaption. However, such an evaluation was outside the scope of this work and therefore left for future research.

In addition to the offset adaptation, we also analyzed the resource allocation as an additional method for reducing transmission latencies. Therefore, we analyzed different duplexing strategies according to their advantages and disadvantages and proposed FDD as preferred duplexing method. Subsequently, we compared different resource allocation schemes. For the two-dimensional resource allocation, there are four different approaches possible. We compared their individual strengths and also considered their susceptibility to errors in physical transmission. Since the radio channel strongly depends on local conditions, from a PHY perspective it was not possible to highlight one specific allocation scheme as universally best. However, being outside the scope of this work, PHY related considerations remain brief.

Consequently, we compare the resource allocation according to the application related timing. We started with a review of an individual network and extended the results for the data transmission in cascaded communication systems. We have shown that the data reception time and therefore the transmission latency mainly depends on the allocation offset of the individual data packet, the actual transmission time of this data packet,

and its corresponding processing time.

Since the allocation offset predominantly depends on the allocation scheme, we have identified the sequential resource allocation as the preferred scheduling method with respect to the transmission latency. However, this does not take the PHY related considerations into account. Especially for an environment with broadband interference, a sequential resource allocation might remain unfavorable even if we could reduce the transmission latency for correctly transmitted data packets. This again shows the dependency on the specific wireless channel and the need for more accurate wireless channel models for industrial areas.

Moreover, we have proposed a resource scheduling that compensates the latency that results for the individual data packet size. We have shown that larger data packets should always be scheduled first for the DL, but at the end of an individual frame for the UL. By extending this resource allocation scheme, we have to consider both the destination and source level of the individual data packets to be forwarded. In DL direction, data packets for slaves in higher levels of the cascade should be scheduled first in order to compensate for additional forwarding and conversion latencies. For the UL, the scheduling should be vice versa. Finalizing these considerations, we were able to introduce a latency reduced resource allocation algorithm for arbitrary subnetworks in a cascaded communication system.

In order to verify our results, we applied them to our machine tool application scenario. As a result, we were able to reduce the overall transmission latency by a further 32 µs. However, we could also identify the *ParSec* subnetwork as bottle neck for the use case. Due to its lower data rate compared to the other subnetworks, we cannot fully exploit the potential of our latency reduced resource allocation. This has shown that the adaptation might not always result in a significant latency reduction due to the complex dependencies in the data transmission in cascaded communication networks. Accordingly, the actual improvement has to be evaluated for each implementation individually.

Chapter 7

Timing Evaluation of Demonstrator Implementation

Regarding communication networks, industrial communication is a conservative domain due to long application operation times. New concepts have to be tested and verified explicitly before they find their way to the market. Therefore, one of the goals of the ParSec project was the development of a demonstrator implementation to verify the performance and usability of the *ParSec* network. In the context of this work, a specific demonstrator development for the validation of cascaded networks was supported. This setup consist of hierarchically ordered *Sercos* and *ParSec* networks and is specially intended for closed-loop control applications. Therefore, several of the concepts we presented in this work have been deployed, unless limited by hardware and software restrictions. Accordingly, we use this demonstrator implementation here as a practical verification of our introduced concepts.

The following project partners were involved in the development of the demonstrator: Robert Bosch GmbH, Bosch Rexroth AG, TU Dortmund, IHP, and FH Bielefeld. The main contribution can be summarized as follows: We presented our latency reduced network integration methods on behalf of Bosch Rexroth and supported the hardware adaptation and software implementation of the other partners. The hardware was assembled and adapted by Robert Bosch GmbH, who also implemented the corresponding software. The TU Dortmund developed and implemented the specific *ParSec* media access control (MAC) methods. A link emulator that was used instead of the actual *ParSec* physical layer (PHY) was provided by the IHP. The FH Bielefeld supported the demonstrator with dedicated measurement hardware and software. The overall integration was done mainly by the Robert Bosch GmbH with the support of the Bosch Rexroth AG and TU Dortmund. The exact and detailed contribution of the individual partners is described in the ParSec project internal reports of the individual contributors (cf. [2, 54, 55]).

In the context of this work, we support the demonstrator setup explicitly with our introduced integration methods for reducing the overall transmission latency. Therefore, we supported the implementation, the final measurements, and their evaluation. The results are presented as follows: In Section 7.1 we start with a detailed description of the implemented demonstrator setup and link it to the concepts introduced in this

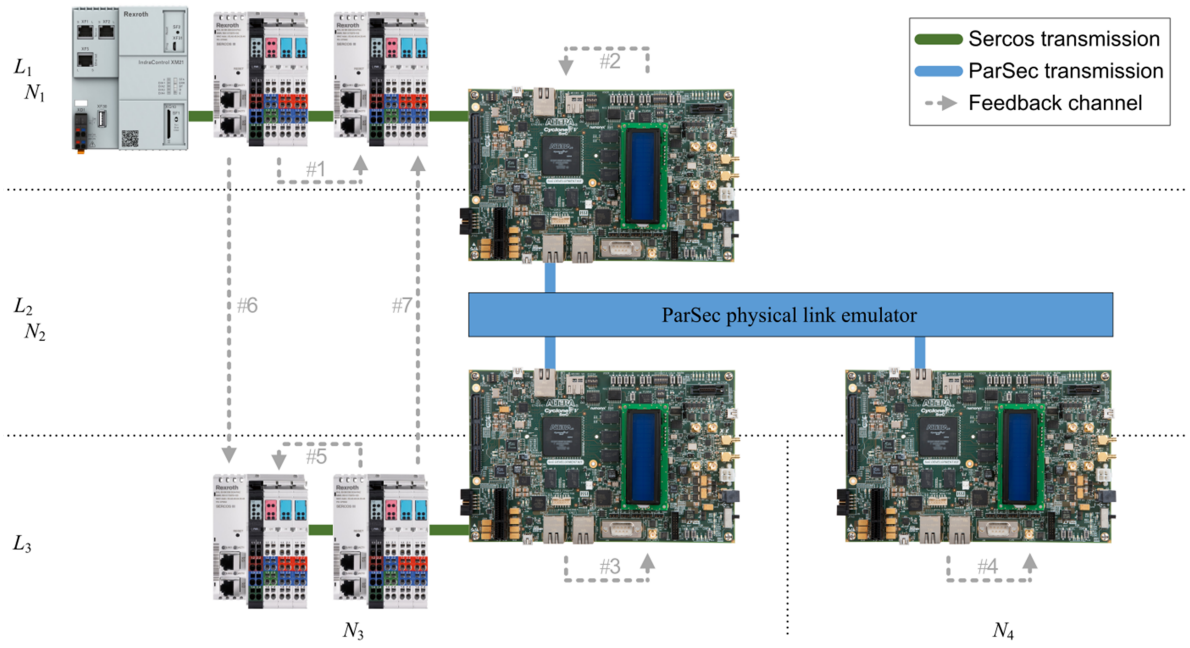

Figure 7.1: *Schematic representation of the assembled demonstrator setup.*

work. In Section 7.2 we measure the achieved synchronization accuracy and compare it to the requirements of industrial closed-loop control applications. Subsequently, we present techniques for measuring the latency of the end-to-end data transmission from the master to the slaves and back again in Section 7.3. Finally, Section 7.4 concludes this chapter.

7.1 Detailed Demonstrator Setup

The demonstrator was intended to validate the usability of the *ParSec* network for industrial closed-loop applications. Therefore, we demonstrated a cascaded communication network as expected for future applications, where a wireless network replaces dedicated parts of the wired communication. Fig. 7.1 depicts the basic setup, consisting of four subnetworks (N_1 to N_4) distributed over three hierarchy levels (L_1, L_2, and L_3). Tab. 7.1 describes the structure in more detail.

For the subnetwork N_1 on L_1, we used *Sercos* as example for a typically used Industrial Ethernet (IE) protocol for factory automation (FA) applications. This subnetwork represented the root-subnetworks. Correspondingly, it contained the overall communication master. This was represented by an XM21 embedded controller from Bosch Rexroth. It was directly connected to two digital input/output (I/O) modules (IP20 by Bosch Rexroth) and a gateway to L_2, implemented on a Cyclone V SX SoC development board from Intel. Another two I/O modules have been virtually implemented on the development board in order to increase the transmission payload. For the controller,

Table 7.1: *Detailed description of the assembled demonstrator setup.*

Level	Subnetwork	Protocoll	Cycle time [ms]	Controller	Number of I/O modules (virtual)	Gateways
L_1	N_1	*Sercos*	1	1	4 (2)	1
L_2	N_2	*ParSec*	1	0	0 (0)	3
L_3	N_3	*Sercos*	1	0	4 (2)	1
	N_4	*Sercos*	1	0	2 (2)	1

the gateway and the I/O modules were representing *Sercos* slaves. The subnetwork N_2 on L_2 represented the intermediate wireless communication. Therefore, we used the wireless *ParSec* protocol. This subnetwork included two more gateways to N_3 and N_4 on L_3. The subnetwork N_3 contained four more *Sercos* slaves: two IP20 digital I/O modules and two virtual I/O modules. N_4 contained only two virtual I/O modules.

The development boards for implementation of the gateways and virtual I/O modules consisted of an field programmable gate array (FPGA) and a dual-core ARM processor. The processor was running a Linux operating system patched with real-time (RT) preemption. The time critical tasks for data forwarding, such as cut-through data insertion and extraction, time synchronization, the interface to PHY, and parts of MAC processing were implemented in the FPGA of the development board. More complex but less time critical tasks, such as the state machine or the actual data forwarding, were realized in the gateway application layer (APP) and executed in the processor. One of the processor cores was used to continuously poll a specific register indicating new data arrival at the FPGA. Although this generated a core load of up to 100 %, the latency ($\leq 5\,\mu s$) was drastically reduced compared to an interrupt based notification method ($\approx 100\,\mu s$).

According to our gateway concepts we introduced in Chapter 4, all gateways had to fulfill a double role. The gateway between N_1 and N_2 thus was representing a single modular *Sercos* slave with eight digital I/O function blocks to the *Sercos* master on N_1 and a *ParSec* master to N_2. The gateways between N_2 and N_3 respectively N_2 and N_4 were representing a *ParSec* slave to N_2 and a *Sercos* master to N_3 respectively N_4. All of them were implemented in a proxy-based concept (cf. Subsec. 4.3.2) for latency reduced data forwarding.

Concerning our introduced resource allocation methods (cf. Sec. 6.5), a sequential resource allocation was implemented for the *ParSec* communication. Unfortunately, the corresponding *ParSec* PHY was not fully available at the time of these measurements. Therefore, we had to use a wired PHY link emulator for the actual data transmission. It was implemented on a Virtex II development board from Xilinx and represented the expected interfaces to the MAC. It was broadcasting received downlink (DL) transmission to all *ParSec* slaves and multiplexing individual uplink (UL) transmissions in order

to forwarded them to the *ParSec* master. For forwarding, the expected transmission delay including baseband processing and propagation delay were emulated. However, it was not representing the actual error behavior of a wireless communication channel, nor the synchronization deviation resulting from not perfectly synchronized devices. Reproducing the exact error behavior would have been highly complex, since geometry of the setup, alignment of the antennas and several other parameters would have had to be included. Accordingly, we did not consider bit and frame errors for the following measurements as 1) they were not explicitly required for our intended first proof of concept and timing evaluation, and 2) we might assume corresponding error correction methods that provide the required robustness in future wireless communication systems.

According to the intended closed-loop applications, precise end-to-end synchronization was required over the entire cascaded network. Therefore, the subnetworks on higher levels were subordinated to the root subnetworks. In order to achieve this, the transmission of new DL frames in a subsequent subnetwork was only triggered if a DL frame from the preceding subnetwork was received at the gateway. This was implemented directly within the FPGA of the gateway to achieve the required accuracy. Accordingly, all subnetworks were implementing the same cycle time (cf. Chapter 5) and the jitter in the cycle start was only depending on the sampling rate of the trigger signal. This jitter was further reduced by a phase-locked-loop that generates an average frame trigger according to a certain number of past trigger signals. However, the frame transmission could be shifted by a specific offset. These offsets were set according to our optimization presented in Sec. 6.3.

The actual data forwarding was realized in software on the processor. This allowed more flexibility in adaptation of different data structures as for *Sercos* and *ParSec*, but introduced additional processing latencies. An implementation in FPGA would reduce this latencies, but limit the set of supported data structures.

Each I/O module provided eight digital inputs and eight digital outputs. In the demonstrator setup, we used some of them as electrical feedback for timing measurement. Therefore, output ports of some digital I/O modules were directly connected by wires to input ports of other digital I/O modules (cf. feedback channel #1, 5, 6, and 7 in Fig. 7.1). As a result, the UL signal was directly following the DL signal, and time difference between a signal change could be measured.

The cycle time implemented in the demonstrator equals $\Delta t_{\mathrm{Cycle}} = 1\,\mathrm{ms}$. The offset parameter follow our optimization presented in Chapter 6. However, in contrast to our theoretical analysis, we had to consider further restrictions due to the actual implementation. For example, in the *ParSec* subnetwork all data had to be fully received before the transmission of the frame has started. Otherwise, they could not be allocated for the corresponding resources. Moreover, in the *Sercos* subnetwork on level L_1 of the cascaded network the UL transmission was statically allocation at the end of the communication cycle and could not be scheduled earlier. As a result, adapting the offset parameter for a closed-loop optimized timing was not possible. Instead, we implemented two different frame structures: one that allows an actual data optimized timing and another one for a command data optimized timing.

Both required measuring the specific conversion times within the gateways and the

Table 7.2: *Offset parameter for subnetworks N_1 to N_3 of the demo setup to enable actual data respectively command data optimized timing.*

	t_{DL_1}	t_{UL_1}	t_{DL_2}	t_{UL_2}	t_{DL_3}	t_{UL_3}	t_{GSP}
Actual data	0 µs	963 µs	109 µs	163 µs	567 µs	767 µs	765 µs
Command data	0 µs	963 µs	500 µs	554 µs	0 µs	295 µs	270 µs

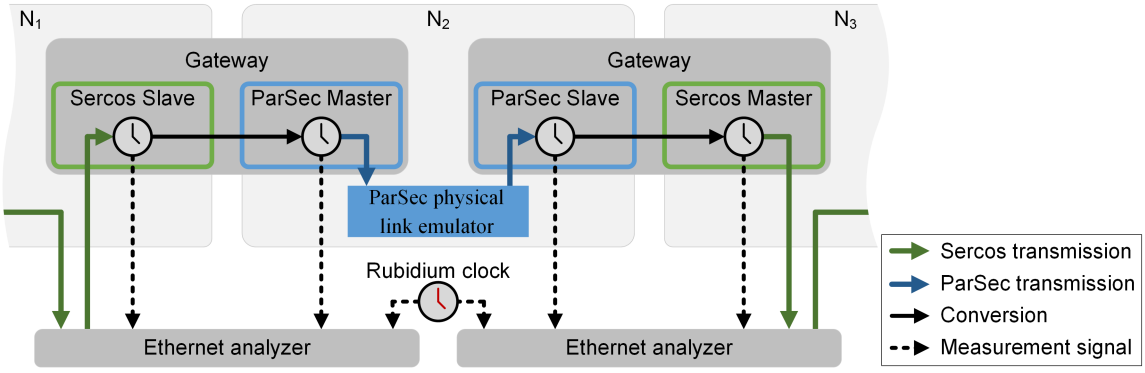

Figure 7.2: *Schematic representation of measurement setup for timestamps at various reference points.*

processing times of the slave in the actual demo setup. Converting the DL data for forwarding in N_2 took about 100 µs, while the conversion for forwarding in N_2 took only about 63 µs. Processing of DL data took about 180 µs. Acquiring the UL data took about 10 µs. In the UL direction, it took about 113 µs to process the data for the transmission in N_2, respectively 38 µs for the transmission in N_1.

According to these parameters, we could adapt the offset parameters for the corresponding timing method. In both cases, it was not necessary to implement an additional RT data offset within any frame. The resulting offset parameters for our specific demo setup are presented in Tab. 7.2.

7.2 Synchronization and Jitter Measurements

As introduced before, accurate synchronization with minimal deviation is one of the most important aspects in factory automation. Therefore, we start with an evaluation of the achieved synchronization accuracy and jitter of the demonstrator setup using the parameter set for the actual data optimized timing. The corresponding measurement setup is depicted in Fig. 7.2.

We have used this setup to record timestamps in end-to-end transmission from master to its slave at specific reference points. Therefore, we used several Ethernet analyzer

netANALYZER NANL-B500G-RE of Hilscher. They were able to record two bidirectional Ethernet non-delay forwarding Terminal Access Points (TAP) and up to four electrical signals with a resolution of 10 ns. They were connected to a standard computer via Ethernet. This computer was running *MATLAB* to control and evaluate the measurement.

Each analyzer had an individual internal crystal for clock generation. However, the individual clocks of multiple devices may slightly vary during measurement. In order to avoid this measurement error, we used an additional rubidium clock GRClok-1500 of Spectratime to synchronize the individual Ethernet analyzers. This rubidium clock provided a precise and temperature stable 1 Hz trigger signal as time reference.

For our measurement, we recorded the timestamps of the exchanged *Sercos* telegrams, the timing of the global sampling point (GSP), and the *ParSec* frame trigger. We measured these signals once at the gateway between N_1 and N_2 and once at the gateway between N_2 and N_3. Accordingly, we could derive the individual DL and UL offsets of N_1 to N_3. The Ethernet analyzers recorded the time between two successive events. In our demonstrator setup, these were the received Ethernet telegrams or the rising edges of the digital signal.

The actual measurement was initiated using *MATLAB*. However, this was leading to a small offset between the individual Ethernet analyzer. Accordingly, we had to assimilate the number of signal samples after the recording. Moreover, the post-processing also included the correction of the different internal clocks according to the rubidium clock. Therefore, we calculated the difference between an ideal 1 Hz signal and the measured 1 Hz signal of the rubidium clock. We used this difference to correct all other measured signals by a linear interpolation. After these adaptations, we could assume the measurements being synchronized with a maximum deviation of 10 ns resulting from the measurement resolution.

For our first measurement, we captured 340 s runtime while the system was completely initiated and running. With a cycle time of $\Delta t_{\mathrm{Cycle}} = 1\,\mathrm{ms}$ this equals 340 000 communication cycles. In Tab. 7.3 and Fig. 7.3 we present the measured timestamps. Additionally, we depict their individual probability density function (PDF) in Fig. 7.4. Since the communication was initiated with the DL transmission of the controller, we normalized all other measured timestamps to the DL offset of N_1 (t_{DL_1}). For a clear representation, we do not depict the UL offset of N_1 or N_3 in Fig. 7.3 and Fig. 7.4. The transmission of the UL frame was directly linked to the preceding DL frame for the *Sercos* based subnetworks. Hence, the corresponding graphs are very similar apart from possibly different offsets.

The measurement revealed that all recorded timestamps remained relatively constant during runtime. Their standard deviation was $\sigma < 50\,\mathrm{ns}$. As expected, the jitter of the timestamps increased with the forwarding from N_1 to N_2 and from N_2 to N_3. Since the individual components were deriving the synchronization of their asynchronous clocks from the incoming signal, inaccuracy was added with each additional data forwarding. Nevertheless, the jitter remained below the typically required synchronization deviation

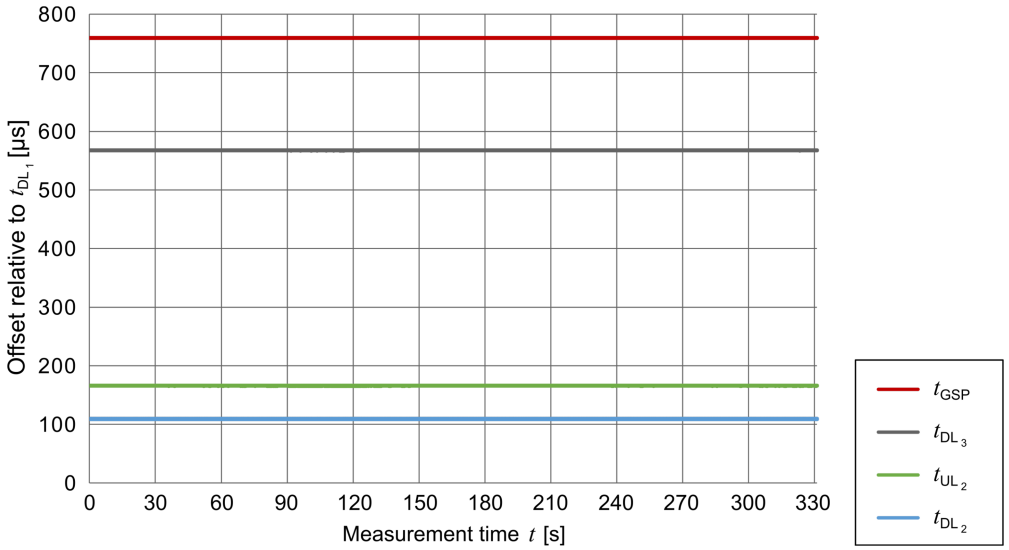

Figure 7.3: *Measured reference times of demonstrator setup normalized to* t_{DL_1}.

Table 7.3: *Measured reference times of demonstrator setup normalized to* t_{DL_1}.

Time	Mean [µs]	Standard deviation [ns]	Jitter [ns]
t_{DL_1}	0.00	0.0	0.0
t_{UL_1}	962.60	23	250
t_{DL_2}	109.18	29	240
t_{UL_2}	162.82	45	358
t_{DL_3}	567.42	48	401
t_{UL_3}	765.40	48	403
t_{GSP_1}	759.18	12	50
t_{GSP_3}	756.06	46	376

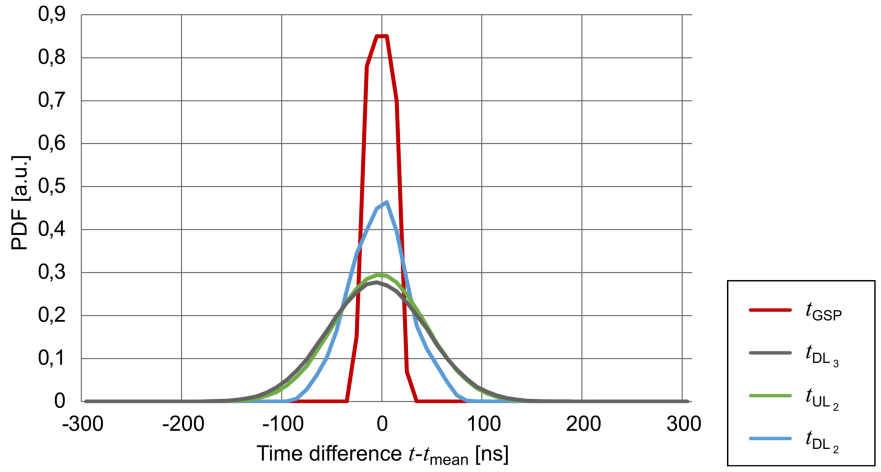

Figure 7.4: *Probability density function (PDF) of measured reference times related to corresponding average.*

of 1 µs (cf. Chapter 2).

Especially for the GSP, a deviation of $\approx 3.13\,\mu s$ could be seen. However, the individual slaves were deriving the GSP according to their command data reception and an additional offset that was provided by the master. This offset must compensate for all delays in the data forwarding. Therefore, its calculation included several parameters that remain to be defined precisely. The measurements reveal that we actually overcompensated the expected latencies, since the timestamps for the GSP in N_1 were recorded later than those in N_3. As the jitter indicated that the deviation was relatively constant, it could be easily compensated for by adapting the corresponding offset. Accordingly, the measurements revealed that the demonstrator setup provides the required synchronicity for typical closed-loop control applications.

7.3 Latency Measurements

The previously described exchange of synchronization information was performed directly in the FPGA. Therefore, it was not influenced by additional delay due to the operation system or the application software that are both executed in the processor. However, the transmission of the actual RT data was influenced by those delays. Therefore, we measured the end-to-end transmission latency from the master to the slaves and back again to evaluate the timing behavior of the cascaded network. Additionally, we measured direct feedback from each slave separately. This measurement could be used to evaluate the corresponding latency introduced by the individual gateways. Moreover, we could derive on which level this latency was added.

We measured the latency directly at the controller. Therefore, we adapted the application and implement seven different feedback channels (cf. Fig. 7.1). In order to measure the latency of the three individual gateways, we used a 4 bit counter value for the corresponding control and actual data. We could use the difference between the

Table 7.4: *Measured transmission latency for different feedback paths using the actual data respectively command data optimized timing.*

#	Feedback path	Required communication cycles	
		Actual data	Command data
1	Slave $N_1 \rightarrow$ Slave N_1	2	2
2	Gateway $N_{1/2} \rightarrow$ Gateway $N_{1/2}$	2	2
3	Gateway $N_{2/3} \rightarrow$ Gateway $N_{2/3}$	3	3
4	Gateway $N_{2/4} \rightarrow$ Gateway $N_{2/4}$	3	3
5	Slave $N_3 \rightarrow$ Slave N_3	3	3
6	Slave $N_1 \rightarrow$ Slave N_3	3	2
7	Slave $N_3 \rightarrow$ Slave N_1	2	3

values to derive their end-to-end latency on order of communication cycles. For the I/O modules, we implemented four electrical 1 bit feedback channels (#1, 5, 6, and 7). As the controller inverted the signal level at each reception, we could measure the latency as the time until the inverted signal was again received at the controller. The feedback channel #1 was used to evaluate only the data transmission in N_1. The feedback channel #5 represented the data exchange between slaves in N_3, including the forwarding of DL and UL over N_2. Within the feedback channel #6 we implemented a direct shortcut of the command data of N_1 to the actual data of N_3. The corresponding UL included forwarding over N_2. Finally, on the feedback channel #7 the UL was bridged from N_1 to N_3, whereas the DL included forwarding over N_2.

We evaluated the timing for actual data optimized parameter set, as well as for the command data optimized parameter set. The measurement results are presented in Tab. 7.4. Here we represent latency in the number of communication cycles required for end-to-end data exchange. According to the cyclic behavior of industrial communication, this value is more meaningful than the exact latency. Moreover, at application level the recorded timestamps were influenced by inaccuracy due to processing in parallel to the operation system. This would further reduce the informative value of exact latency.

We used feedback channel #1 as benchmark for the other measurements. The data transmission included only *Sercos* components. Therefore, timing was not affected by any additionally introduced latencies due to cascading and represented the best case. The measurement revealed, that one communication cycle was required for the exchange of the command and actual data between master and corresponding slaves, including processing at the slave. One additional cycle was required for processing at the master.

The measurement of the second feedback channel represented the evaluation of our *Sercos* slaves implementation in the gateway. The latency was measured only at the gateway without any forwarding to other subnetworks. As expected, the result equals the

benchmark measurement. Accordingly, no significant latency was added by our gateway implementation. Since this result was still independent of the *ParSec* transmission schedule, the latency was equal for both timing methods.

The third and fourth feedback channel included data forwarding over the *ParSec* subnetwork N_2. We measured the latency at the *ParSec* slave implementation of the gateways. Accordingly, it was equal to the delay added by the first forwarding. Compared to the benchmark, one additional communication cycle was required for end-to-end data transmission. In case of the DL optimized schedule this additional cycle was required to transmit actual data, whereas command data were not affected. For the UL optimized schedule, the timing was behaving the other way around.

We used the feedback channel #5 to represent the data exchange between the master and a slave in N_3. The measured latency included the data forwarding over the *ParSec* subnetwork for both DL and UL. The required number of communication cycles was equal to the feedback directly from the *ParSec* slave. Hence, no additional significant latency was added by data transmission in N_3. As before, one additional communication cycle was required for both optimizations for the same reason.

We recorded the feedback channels #6 and #7 explicitly to represent the difference of both different scheduling methods. In the measurement of feedback channel #6, only the *ParSec* UL was included. This measurement revealed that it was possible to achieve the benchmark result with UL optimized scheduling. However, for DL optimized scheduling the latency increased such that one additional communication cycle was required for end-to-end data transmission. For the feedback channel #7, the timing behaved exactly opposite.

All in all, our latency measurement shows that with the implementation of our developed latency minimization concepts the demonstrated cascaded network performs nearly as well as the communication networks so far. In the worst case, only one additional communication cycle was required for end-to-end data transmission from master to its slaves and back again. Moreover, even with an intermediate wireless data transmission, either DL or UL can be optimized to fulfill the corresponding timing requirements. Therefore, we could validate that the demonstrated cascaded network is suitable for a large quantity of industrial closed-loop control applications. Only those applications with tightest latency requirements might not be supported sufficiently.

7.4 Discussion: Required Measurements

In this chapter we presented a demonstrator implementation of a cascaded network for closed-loop control application. It was created as part of the ParSec project and intended for the timing evaluation of the entire communication system. However, at the time of measurement, the corresponding PHY of the *ParSec* network was not available. Therefore, the measurements were based on a specific *ParSec* link emulator. This emulates at least the expected transmission and processing time of the *ParSec* PHY. However, the expected error behavior of a wireless industrial communication channel has still to be considered. Accordingly, our validation represents only one aspect of the evaluation. More measurements with the real PHY are required in order to validate the

entire system, especially error behavior and interferences to parallel systems.

During development of the demonstrator, several of the methods introduced in this work were considered in order to minimize the overall transmission latency. The gateway implementation equals our introduced proxy-based gateway concept, where each gateway represents all subsequent slaves as one single slave with extensive functionality. Frame conversion to individual subnetworks was implemented with a static synchronized resource allocation in order to reduce transmission jitter. Offset and resource allocation were also adapted in order to minimize transmission latency. However, due to the *ParSec* implementation, several parameters differ from our assumptions used earlier in this work. In particular, the fact that all data have to be available for transmission before the corresponding frame is initiated results in a larger latency degradation then predicted. Accordingly, this implementation aspect should be changed for future applications by eliminating individual data copying between the protocol layers.

In order to validate synchronization accuracy and latency requirements, we have performed two different measurements. For synchronization accuracy measurements, we have adapted the demonstrator setup using several Ethernet analyzers to record precise time stamps of various reference times that are used to derive the synchronization information. Our measurement revealed that a good synchronization with a deviation $\leq 403\,\text{ns}$ could be achieved for all slaves within the cascaded network. Therefore, synchronization requirements of industrial application could be fulfilled sufficiently. However, the demonstrator setup represents only a minimalistic application. For larger applications including a cascaded communication network with more than three levels, the deviation will continue to increase with each level. Therefore, further measurements might be required in order to derive the inaccuracy that is added with each level and up to which size the required accuracy of $\leq 1\,\mu\text{s}$ will still be achieved.

For latency evaluation, we have implemented seven feedback channel to measure the latency of different signal paths. Our measurements revealed that the timing behavior of the cascaded network is slightly worse compared to a typically used communication network so far. The *ParSec* subnetwork adds one additional communication cycle delay to end-to-end data transmission between master and corresponding slaves. However, here we have to mention again that the demonstrator setup represents only a minimalistic example. For a cascaded subnetwork with higher network load, the latency will probably increase further. Accordingly, additional measurements might be required. Moreover, since the *ParSec* network represents the previously described bottleneck, better results might be found with alternative networks, such as *5G*.

Nevertheless, the demonstrator implementation and its evaluation serves as additional proof of the methods for network integration and latency reduction introduced in this work. According to these methods, we are able to achieve a similar timing behavior with the cascaded network as with a normal network. Anyhow, there are more possible optimization methods to reduce the latency even further such that the remaining deviation can probably be overcome as well.

Chapter 8

Outlook on Additional Methods for Improving Cascaded Networks

In this work we presented several methods for an optimized interconnection between different communication networks and a reduction of additionally emerging latencies. The verification based on our machine tool application scenario and the ParSec demonstrator both prove our presented methods and reveal their improvements compared to arbitrary communication networks. Nevertheless, we were still unable to exchange all command and actual data within just one communication cycle of 1 ms for either the machine tool application scenario or the ParSec demonstrator. Regarding the *ParSec* communication, too many additionally latencies emerge. However, previously attempted methods also could not compensate for or minimize this effect.

Based on these results, we developed additional ideas for a latency reduced communication or improved network integration within various collaborations. However, these might require significant changes to applications, network protocols, or hardware. Since this is partly outside the scope of this thesis, we have not yet examined these ideas in detail. Therefore, Their following short discussion serves primarily as an outlook for future work. More information can be found in appendix A.8

Shifting the Overhead

The first idea concerns latency resulting from the transmission of the overhead required by the communication system. Even though common communication protocols for industrial applications use summation frames in order to minimize this overhead, it is required for each frame transmission and delays the transmission of the corresponding data. Therefore, the basic idea was to transmit the overhead required by the communication system when data cannot otherwise be transmitted anyhow.

This concept is already included indirectly in our offset adaptation (cf. Sec. 6.3). Here, we shift the frame transmission such that processed data packets can be directly transmitted and do not have to be stalled. As a consequence, the required overhead is transmitted while the application is still busy processing data for transmission.

The idea specially targets the time interval before and after the global sampling point (GSP). During this time, the application is typically busy processing the command data or acquiring new actual data. Accordingly, no new data are provided for the transmission during this time interval. Therefore, it might be beneficial to utilize this time to exchange the necessary overhead information. Of course, the type of overhead obviously has an important role to be taken into account. For example, overhead for management of the system could be easily shifted to dedicated frames and be transmitted with best effort only, while preamble information, for example for frame delimiters, have to be transmitted with every frame.

However, the shift of the offset transmission might not be limited to the time slot around the GSP. It might be possible to find additional or other, more useful time slots for each subnetwork, or to use small channels parallel to the real-time (RT) transmission, which are dedicated only for management information.

By applying this method to our machine tool application scenario, we will be able to start the downlink (DL) transmission in the root subnetwork earlier. Additionally, we can improve the adaptation of the uplink (UL) RT offset of the *ParSec* subnetwork. So far, we are limited due to the successive DL overhead transmission. Shifting this overhead relaxes the corresponding parameter boundaries. As a consequence, we might be able to transmit the data of one more slave from the master to this slave and back again within the cycle time of 1 ms.

This example proves that the concepts require further evaluation in future work.

Reducing the Amount of Data

This idea targets traffic reduction and therefore also the reduction of transmission latencies. Industrial closed-loop control applications are mostly based on cyclically exchanged command data and actual data. The actual transmission finally takes place independent of the content. Therefore, it might happen that we repeatedly transmit redundant data without any changes if the application does not require any modifications. Especially for shared wireless resources networks, this might decrease the efficiency of the communication. A reduction of redundancies or compression of data might release allocated resources for other transmissions, reduce interferences, and decrease the amount of energy and costs that is required for the transmission.

Thus, the basic concept behind this idea is to reduce the amount of data that have to be transmitted. In particular, we should avoid any form of redundant transmission,

e.g. repeated control or actual data, or status information. One approach to do so is to implement a content prediction for each communication participant. Before we transmit new data, we compare their content with the predicted content. If the predictions are correct at the transmitter, we can assume that the prediction at the receiver will also be correct. Therefore, we do not have to transmit these data. Only if the prediction is incorrect due to external interventions or unpredicted errors do we have to transmit the data explicitly.

For our presented examples, such a content prediction will only be feasible and useful at the gateways, since we cannot influence the software of the master or slaves. In addition to spare resources, we might also reduce the transmission latency if less or no data have to be transmitted and this is signaled fast enough. In order to achieve this, the introduced concept requires more investigation and detailed development of specific prediction methods.

Advanced Resource Allocation of Non-Real-Time Data

This idea targets an advanced resource allocation for non-real-time (NRT) data that has not been considered in this work so far. It is based on concepts is presented in [111] and aims to reduce scheduling effort in cascaded communication networks.

NRT data typically appear sporadically and differ in size, source, and destination. Therefore, they cannot be scheduled offline as RT data and have to be allocated for each communication cycle individually. The authors of [111] describe a methods of allocating dedicated resources for NRT data only that are not statically assigned but shared between all devices. The authors describe this method with a direct interconnection between the DL and UL resources for NRT data. Only those slaves addressed within the NRT DL resources are allowed to transmit NRT UL data within the corresponding resources.

Complementing this method, we would like to introduce two possible extensions that might further improve the usability. The first extension is to introduce a notification about available NRT data for the slaves to the master. Such a notification, for example within the RT data, would probably further reduce the number of unused resource elements. Therefore, for example, only one single signaling bit could be used. Even slaves that do not transmit any RT data can probably transmit such a signaling bit easily. With this notification, it might even be possible to decouple the relation between DL and UL resources for NRT data. The second improvement might be the consideration of a successive polling. Since the NRT resources are limited, the corresponding data packets probably have to be segmented and transmitted over several communication cycles. Accordingly, the probability that a slave has more than one NRT data packet to transmit might be higher than the probability that another slave will also have to transmit NRT data. It is also conceivable that a slave will inform the master within the NRT data whether it has any further data to transmit or not. As a result, the efficiency of the scheduling could be increased even further.

The necessary implementation effort and effectiveness still need to be investigated further.

Chapter 9

Conclusions

Industrial applications are currently more challenging and diverse than ever before. Ever increasing quality requirements coupled with greater competition demands constant adjustment and optimization. New concepts from the internet of things and cyber-physical systems enable connected production processes and increased efficiency. Data mining and machine learning accelerate automation. At the same time, shorter production development cycles require more flexibility. The fourth Industrial Revolution is on.

Continued development is only possible with support from corresponding communication technologies. However, factory automation is considerably more challenging than other domains because of its closed-loop control applications. In addition to being characterized by low latencies, high reliability, precise synchronization and determinism, communication between corresponding applications is plagued by security and safety requirements. Moreover, the success of future applications also depends on innovations in flexibility, scalability, and maintenance reduction.

Currently, neither wired nor wireless communication networks are capable of concurrently meeting all requirements of industrial application. The typically used wired communication networks based on Industrial Ethernet (IE) lack flexibility and scalability. Current wireless technologies, however, cannot meet reliability and latency requirements. Newly developed systems such as *ParSec* and *5G* should overcome this problem. However, breaking into the industry and being adopted as industry standard will require time since industrial applications typically last more than 10 years. The required communication network therefore has to be universally applicable and easy to integrate.

9.1 Overview of Conclusions

In this thesis, we present cascaded communication networks consisting of hierarchically ordered wired and wireless communication networks as a possible communication concept for future applications. The combination of wired and wireless communication networks offers the potential of both technologies. Moreover, existing applications based on wired communication networks can be easily extended by wireless networks, enabling cascaded systems to emerge. A required proper interconnection of individual subnetworks will enable adopting the advantages of both wired and wireless networks. Our dedicated requirements analysis has shown that we need to focus on interconnection methods that minimize additionally emerging latencies resulting from the combination of arbitrary heterogeneous communication networks. We presented four different approaches which allow retaining the advantages of individual subnetworks with minimal or no performance degradation when used in combination. We provided an evaluation of these methods and gave an outlook on future work.

9.1.1 Specification of Requirements

Using a detailed requirements analysis of industrial applications, we specified the performance that cascaded communication networks have to provide. Our introduced comparison of three application classes condition monitoring (CM), process automation (PA), and factory automation (FA) as well as the specific use case analysis of FA applications can be used as indication of the most important performance figures for each area and as benchmark for cascaded networks that are intended to be used there. We revealed that applications of class of FA are most demanding with respect to communication timing. These applications need a cyclic data exchange with cycle time down to 0.5 ms, a synchronization deviation of up to 0.25 µs, and reliability, where missing more than two successive data packets cannot be tolerated.

9.1.2 Timing Evaluation

Fulfilling the hard timing requirements is particularly challenging, even with wired communication networks. We showed that there is only a limited number of wired-only networks offering such a performance and that we can probably only achieve the above stated requirements and gain wide acceptance when consider them as part of cascaded network. Nevertheless, additional latencies will emerge due to cascading, especially when considering heterogeneous subnetworks and wireless data transmission. In order to quantify these latencies, we developed a simple model that enables timing analysis of both individual communication networks and complex cascaded networks. Based on this model, we also presented an algorithm for evaluation of most relevant timing parameter (such as command data reception time and actual data acquisition time) for all slaves connected to the controller. This algorithm concerns also the required timing behavior of the application and enables an assessment of the network parameterization and potential optimization methods.

9.1.3 Latency Reduction Methods

In this thesis, we manly focused on four methods intended to reduce additional latencies within data transmission that emerge by cascading heterogeneous communication networks. These methods enable the combination of wired and wireless subnetworks where communication advantages of wired timing behavior meets the flexibility of wireless networks.

Interface Concepts

The first method for reducing additionally emerging latencies within cascaded communication networks is based on the interface between individual subnetworks. We analyzed different interface concepts across arbitrary subnetworks. We concluded that a proxy based gateway concept results in the lowest transmission latencies unless in case of point-to-point communication only. At the same time, this concept is less flexible. Therefore, we expect larger conversion times when forwarding data from one subnetwork to a subsequent subnetwork than for a tunnel based gateway. Once the exact conversion time is known, our analysis is useful to decide whether it is preferable to implement a proxy or a tunnel based gateway. We showed that the conversion time of the tunnel based gateway approach has to be several orders of magnitude (depending on the amount of data and number of subnetwoks) lower than for the proxy based approach in order to compensate for the significantly higher amount of data to be transmitted.

Frame Conversion Schemes

The second method for reducing additionally emerging latencies corresponds to data conversion between arbitrary subnetworks. As typical wired communication networks use frame structures for transmitting corresponding data, we compared different frame conversion methods according to their resulting timing behavior. Depending on whether a possible subsequent subnetwork also uses frames or implements a streaming procedure, different latencies can emerge. Our analysis revealed that it is most beneficial if a subsequent subnetwork uses an similar frame concept with the same cycle time as the preceding subnetwork. This not only enables the shortest possible latencies for subsequent transmission, it also prevents additional jitter in data forwarding.

Offset Adaptation

An adapted parameterization of individual subnetworks is a third method for reducing additionally emerging latencies within cascaded communication networks. Accordingly, we presented an overall parameter adaptation. We stated that it is not sufficient optimizing just each individual subnetwork. The entire communication chain has to be considered in order to reduce end-to-end latency in data transmission form master to its slaves and back again. Therefore, we started providing an analytical and a heuristic method for finding an improved offset parameterization for the entire cascaded network. Both methods found a parameter set that significantly reduces the overall transmission latency compared to an unadapted parameter set. Based on an analyzed example, we

showed that the overall transmission lasency could be reduced by 50 %. The analytical method therefore performs slightly better but is much more complex, especially for larger cascaded networks. The heuristic optimization strongly depends on the used cost function. In order to get closer to the results of the analytical adaptation, the cost function has to be specifically adapted to fully represent the optimization target.

Resource Allocation

Besides offset parameterization, a dedicated resource allocation represents a fourth method for reducing additionally emerging latencies within cascaded communication networks. We compared different allocation schemes and described their advantages and disadvantages. In order to reduce transmission latencies we proposed using a sequential resource allocation. This enables arranging individual data packets to compensate for longer transmission time. Accordingly, we presented an algorithm to generate a latency reduced resource scheduling for each individual subnetworks that optimizes transmission latency of the entire cascaded network. Together with an adequate readjustment of the offset parameter, even lower latencies can be achieved.

9.2 Evaluation

According to the first two described methods for a latency reduction in cascaded networks, the results presented in this thesis contribute a basis for decision-making when investigation the most suitable subnetworks for building a cascaded network for a specific application. The remaining two methods we presented can then be used for setting up and optimizing such a cascaded network. All methods can be combined with each other. Therefore, the results of this thesis can be used supporting industrial communication networks for time critical application from planning to implementation.

Based on an application scenario example derived from our use case analysis, we were able verifying our methods for latency reduction. Moreover, we presented a validation based on a demonstrator implementation that was assembled within the ParSec project. Most of the methods developed in this work were key to the demonstrator setup. Thus, based on a practical example, we showed that cascaded communication networks are a very good alternative for future industrial applications that offer real-time performance and flexibility at once.

9.3 Future Work

In addition to latency reduction methods that we analyzed in detail, we also identified several other methods to further reduce transmission latencies. However, most of them require significant changes to the application, communication protocols, or hardware. Therefore, we considered them being outside the scope of this thesis, since we focused on combining integration and latency reduction ensuring both the application and the hardware remaining viable. Accordingly, we do not fully elaborate on these methods but presented them as open issues for future work.

Furthermore, using a more efficient wireless communication network for the cascaded communication, e.g. *5G*, might further improve data transmission by lowering latencies, reducing jitter, or increasing data rates. Again, their universality ensures that we will also be able to use our presented methods for such implementations. However, actual improvements still require further evaluation in additional simulations and measurements and should be the focus of future work.

Appendix A

Extended Results

A.1 Requirements of Industrial Application Areas

The subsequently presented tables represent the extended summary of our requirements analysis. Tab. A.1 shows the range of requirements of the different application areas that could be found in literature [21,22,34,36,41,80,81,112,113]. In Tab. A.2 we specify these indications by average requirements that were revealed by our use case analysis and that led to Fig. 2.2.

Moreover, Tab. A.3 presents the specific requirements for the machine tools, packaging machines and printing systems that result from our use case analysis. They are also presented inFig. 2.2.

Table A.1: *Consolidated requirements of industrial application areas.*

	FA	PA	CM
Cycle time	$\leq$1 ms	1 ms to 5000 ms	10 ms to 10 000 ms
Synchronization deviation	$\leq$1 µs	1 ms to 1000 ms	1 ms to 1000 ms
User data length	30 to 100 Bytes	30 to 1500 Bytes	30 to 1523 Bytes
Slaves per system	2 to 50	100 to 300	120 to 1000
Parallel systems	2 to 25	1 to 5	1 to 5
PER	10^{-9}	10^{-4} to 10^{-9}	10^{-3} to 10^{-9}
Spatial extent	$10 \times 10 \times 2$ m	$100 \times 100 \times 10$ m	$1000 \times 1000 \times 50$ m

Table A.2: *Average requirements of industrial application areas.*

	FA	PA	CM
Cycle time	0.5 ms	10 ms	100 ms
Synchronization deviation	0.25 µs	100 ms	5 ms
User data length	50 Bytes	1500 Bytes	250 Bytes
Slaves per system	30	200	1000
Parallel systems	8	2	1
PER	10^{-9}	10^{-4}	10^{-3}
Spatial extent	$10 \times 10 \times 2$ m	$100 \times 100 \times 10$ m	$1000 \times 1000 \times 50$ m

Table A.3: *Specified requirements of factory automation use cases.*

	Machine tools	Packaging machines	Printing systems
Cycle time	0.5 ms	1 ms	2 ms
Synchronization deviation	1 µs	5 µs	0.25 µs
User data length	50 Bytes	80 Bytes	40 Bytes
Slaves per system	20	50	100
Parallel systems	24	2	8
PER	10^{-9}	10^{-8}	10^{-7}
Spatial extent	$5 \times 5 \times 2$ m	$5 \times 4 \times 4$ m	$25 \times 25 \times 3$ m

A.2 Component List of a Typical Machine Tool

In Tab. A.4 we represent a component list of a typical machine tool derived from our use case analysis. This list does not represent a real machine tool. However, the individual components and their numbers are comparable. Timing relevant information are extrapolated for years. Elements of the same cycle time are combined and components, like fuses or emergency stops, with cycle times $\geq 60\,\mathrm{s}$ are ignored, since they have only minimal influence to the machines $\mathrm{MTTF_d}$. The corresponding $\mathrm{MTTF_d}$ indications result from the ISO 13849 in combination with Eq. (2.2).

Table A.4: *Component list of a typical machine tool.*

Component	Quantity	t_{Cycle} in s	n_{op} in a	$\mathrm{MTTF_d}$ in a
Cylinder	8	20	725760	27.55
Cylinder	8	8	1814400	11.20
Valve	8	20	725760	27.55
Valve	8	8	1814400	11.20
Reed-Contact	16	20	725760	27.55
Reed-Contact	16	8	18144000	11.20
Valve-sensor	8	20	725760	27.55
Valve-sensor	8	8	18144000	11.20
Proximity switch	2	20	725760	27.55
Photoel. switch	6	20	725760	27.55
Photoel. switch	4	8	1814400	11.02
Motor drill	5	20	725760	10.00
Motor material entry	1	20	725760	10.00
Motor contactor	6	10	1451520	1.37
PLC	1	-	-	10.00
Mechanic in general	-	-	-	150.00
Exemplified machine tool	-	-	-	2.10

A.3 Short Formal Proof of Algorithm 1

Under the assumption of a correctly parameterized cascaded network with a finite number of levels L_k, subnetworks N_i, and slaves S_j, all for-loops (Lines 1 to 24, 5 to 8, and 17 to 20) are finite as well. Therefore, this algorithm is verified for its finiteness. The partial correctness of producing the correct output value for a valid input value could be proven analytically and is shown simulative in the previous section. Therefore, the algorithm could be verified to be completely correct.

A.4 Short Formal Proof of Algorithm 2

The while-loop in Line 3 ensures that all data packets that have to be transmitted in the corresponding subnetwork N_i are scheduled. Of these data packets, the highest destination level is searched for in Line 4. In Line 5, we deduct a second list for all slaves within the highest destination level. Within the second while-loop from Line 6 to Line 10, we browse the deducted list of data packets and schedule the largest data packet until the list is empty. After scheduling a data packet, we increase the allocation offset according to its size and remove it from both the deducted list and the list of all data packets. The partial correctness of producing correct output values for valid input values is verified exemplarily for our machine tool application example later on. Therefore, the algorithm can be considered as completely correct.

A.5 Analytically Optimized Network Parameterization

In this section we present the analytically optimized parameter set for the command data optimized timing Tab. A.5, the actual data optimized timing Tab. A.6 and the command data optimized timing Tab. A.7. They were generated as presented in Sec. 6.3 and used to generate the results shown in Tab. 6.2.

Table A.5: *Analytically optimized parameterization of the machine tool application scenario with command data optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	932.7	162.1	247.6	361.6	504.1
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	895.3	85.7	468.0	468.0	468.0	468.0
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	283.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			1000.0			
t_{GSP} [µs]			420.1			

Table A.6: *Analytically optimized parameterization of the machine tool application scenario with actual data optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	932.7	162.1	247.6	361.6	504.1
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	684.9	85.7	612.1	612.1	612.1	612.1
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	317.9	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			1000.0			
t_{GSP} [µs]			564.2			

Table A.7: *Analytically optimized parameterization of the machine tool application scenario with cycle time optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	609.2	162.1	247.6	361.6	504.1
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	132.0	85.7	132.0	132.0	132.0	132.0
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			676.5			
t_{GSP} [µs]			84.1			

A.6 Particle Swarm Optimization Algorithm Description

In this section we introduce an Particle Swarm Optimization (PSO) algorithm according to [101]. Let C be a cost function that needs to be minimized. Moreover, let i be a particle out of a swarm with a population size n_{Pop}. Each particle has an position $\vec{x_i}$ in the search space and a velocity $\vec{v_i}$. Let $\vec{p_i}$ be the position of the particle i that minimizes C best and $\vec{g}$ be the overall best position any particle in the swarm reaches so far. Furthermore, T is the a termination criterion of the optimization, $V(\vec{x_i}, \vec{p_i}, \vec{g})$ is a function to update the velocity of a particle and $X(\vec{x_i}, \vec{v_i})$ is a function to update the position of the particle. The performance of the algorithm strongly depends on the update functions V and X. Their input factors are weighted differently based on some PSO specific parameters. Some methods for an advanced parameterization are introduced for example in [103–106].

A basic PSO algorithm than would be as follows:

Algorithm 3: Basic principle of Particle Swarm Optimization.

Input: Cost function C, termination criterion T, swarm of particles $i\,\{\vec{x_i}; \vec{v_i}\}$
Output: Best found position $\vec{g}$ to minimize f

```
1  begin
2      for i ← 1 to n_Pop by +1 do
3          Initialize x_i to a random position within the search space
4          p_i ← x_i
5          if C(p_i) < C(g) then
6              g ← p_i
7          v_i ← V(x_i, p_i, g)
8      while T not reached do
9          for i ← 1 to n_Pop by +1 do
10             x_i ← X(x_i, v_i)
11             if f(x_i) < f(p_i) then
12                 p_i ← x_i
13                 if C(p_i) < C(g) then
14                     g ← p_i
15             v_i ← V(x_i, p_i, g)
```

A.7 Heuristically Optimized Network Parameterization

In this section we present the heuristically optimized parameter set for the command data optimized timing Tab. A.8, the actual data optimized timing Tab. A.9 and the command data optimized timing Tab. A.10. They were generated as presented in Sec. 6.4 and used to generate the results shown in Tab. 6.4.

Table A.8: *Heuristically optimized parameterization of the machine tool application scenario with command data optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	935.3	370.8	367.6	366.0	364.4
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	900.7	88.3	472.4	472.4	472.4	472.4
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	285.8	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			1000.0			
t_{GSP} [µs]			424.5			

Table A.9: *Heuristically optimized parameterization of the machine tool application scenario with actual data optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	934.1	516.5	513.3	511.7	510.1
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	693.1	87.1	618.1	618.1	618.1	618.1
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	323.5	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			1000.0			
t_{GSP} [µs]			570.2			

Table A.10: *Heuristically optimized parameterization of the machine tool application scenario with cycle time optimized timing.*

Parameter	N_1	N_2	N_3	N_4	N_5	N_6
t_{DL_i} [µs]	0.0	610.5	35.4	32.2	30.6	29.0
$\Delta t_{\mathrm{DLOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{DLRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{DLRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
t_{UL_i} [µs]	269.0	87.3	137.0	137.0	137.0	137.0
$\Delta t_{\mathrm{ULOH}_i}$ [µs]	2.1	153.0	2.1	2.1	2.1	2.1
$\Delta t_{\mathrm{ULRTOfs}_i}$ [µs]	0.0	0.0	0.0	0.0	0.0	0.0
$\Delta t_{\mathrm{ULRTData}_i}$ [µs]	52.8	370.5	1.6	4.8	6.4	8.0
$\Delta t_{\mathrm{Cycle}_i}$ [µs]			676.5			
t_{GSP} [µs]			89.0			

A.8 Details to Ideas for Future Work

Idea of shifting the overhead:
The idea was initially created in cooperation with Oliver Utz Wetter (Prof. Dr.-Ing., FH Bielefeld, Campus Minden).

As introduced in Sec. 1.2, most communication systems require the transmission of a certain amount of protocol specific information in order to transmit data packets. There are different kinds of overhead generated through all levels of the protocol stack. There is for example overhead for management of the system. This does not have any real-time requirements and could correspondingly be transmitted at any time. Another kind of overhead are control information for the protocol handling. These typically have a strong connection to the specific data packets but could also be partially abstracted and shifted in time. A third type of overhead is typically transmitted in a preamble. Information, such as frame delimiter or automatic gain control can only hardly be abstracted and detached from the frame.

Since we shifted much of the overhead transmission to time slots that cannot be used for data transmission during our offset adaptation, the overall additional gain of this method seems to be low. Nevertheless, we can at least initiate the downlink (DL) data transmission of the root subnetwork earlier if the overhead is transmitted after the data. Moreover, especially for wireless networks this concept might be beneficial to improve channel estimation. The time the application is busy can be used to analyze the channel and if necessary evaluate alternative transmission parameters. Thus, the estimation might be improved for both directions instead of channel measurement only occurring directly before the DL or uplink (UL) transmission.

One disadvantage of this method could be seen in the additional complexity that is added, especially for cascaded networks. Moreover, not every overhead could easily be shifted. Only transmitting additional frames for certain remaining overhead might also delay other transmissions.

Idea of reducing the amount of data:
The idea was developed in cooperation with Andreas Müller (Dr., Robert Bosch GmbH, CR/AEX1) and Monique Düngen (Dr., Robert Bosch GmbH, CR/AEC1). An invention disclosure is in preparation [114]. We target especially the cyclic data exchange of industrial closed loop application. This is easier to predict and has the greatest influence on the overall communication.

In particular, time critical data are transmitted on statically allocated resource elements. These resource elements are allocated offline, and the corresponding transmission schedule is maintained during the entire runtime (cf. Sec. 1.2). This ensures the required determinism and timing behavior as expected by the application.

However, this results in a content independent data transmission and probably wasted resources for redundant information. For the typically used wired communication networks based on IE, this might be sufficient. Regarding their transmission capacity, they usually provide an adequate data rate. Nevertheless, even these could benefit if less data had to be transferred.

In order to prevent for transmitting redundant information, a content prediction could be implemented for each communication participant. This content prediction might be based on the content history or context and has to be the same across all communication participants. Regarding cyclical data transmission, the simplest content prediction might be to always retransmit the latest data.

Several implementations might be possible. In the first, we do not transmit anything when we notice congruence. In this case, the receiver has to know exactly, when new data will theoretically arrive. Considering a short waiting time for probably delayed transmission, the receiver can then generate the corresponding awaited and not received data based on the prediction. In a second implementation, we might transmit a signaling message instead. This might require just one or only a few bits and reduces the payload correspondingly. However, we have to ensure that this signaling message is transmitted and interpreted correctly. Accordingly, in the third implementation, we will transmit a generated checksum of the correct prediction. Receiving this checksum, the receiver can derive the correctness of its own prediction.

Several advantages might result from reducing data transmissions. On the one hand side, the energy efficiency of the communication system will be increased. On the other hand side, the data rate can be used more meaningfully. For example the transmission of non-real-time (NRT) data might be enabled or the robustness could be increased. Moreover, the interferences of adjacent communication systems will be reduced, especially for wireless communication systems.

However, this idea also comes with some disadvantages. A content prediction requires a certain intelligence and computation performance from each communication participant. In particular, slaves without internal processing, such as sensors, might not always meet these requirements. Moreover, dropping information is very uncommon for industrial communication since even the repetition of data contains some additional information, such as information about the healthiness of the application. Additionally, the repetition rate of data is very application specific. Therefore, the saving cannot be generalized and has to be analyzed from case to case. Furthermore, such a prediction might be very complex and requires significant changes to the software. In case of not transmitting anything, we have to be able to distinguish between dropped data and lost data. For the other implementations, the signaling and checksum transmission, this optimization may prove redundant if they require the typical overhead as the reduction would be negligible. Even if the prediction is possible, we have to ensure that the preallocated resource elements are released fast enough that they can be otherwise used. Moreover, this reallocation has to be signaled or recognized rapidly enough.

Idea of advanced resource allocation of non-real-time data:
This idea is based on concepts presented in [111]. It is especially designed for centrally coordinated communication networks with limited transmission resources.

Industrial communication networks are mainly focused on the transmission of cyclic real-time (RT) data. The amount of RT data that is exchanged each communication cycle is basically known in advanced and statically scheduled. However, the transmission of NRT data has to be enabled as well. In contrast, these data are typically not known in advance and have to be allocated for each communication cycle individually.

Generally, we can transmit NRT data without any or at most with minimal timing constraints. However, the required transmission time should be finite. Moreover, their transmission should not interfere with the RT data transmission. Therefore, their transmission should be scheduled by the master as well. This represent a problem especially in the UL transmission direction. Since the master has no knowledge about the intention of the slaves to transmit NRT, this intention has to be signaled somehow in advance.

The authors of [111] describe three methods to schedule NRT data. The first method is to allocate each slave a certain amount of resources for NRT data in addition to those for RT data. However, this would most often result in a waste of resources when there are no NRT data to be transmitted as they cannot be otherwise used. In the second method, resources are temporarily assigned according to need. This requires the exchange of specific signaling messages, resulting in additional traffic and delay. Therefore, a third method is introduced. This method also uses a specific reservation of resources for NRT data. However, these resources are not statically assigned but shared by all communication participants. All devices examine these resources, but only those actually addressed will completely decode it. This requires that the structure of the corresponding resources remains the same for the entire runtime, or examination will not be possible.

The authors describe this method with a direct interconnection between the DL and UL resources for NRT data. Only those slaves addressed within the NRT DL resources are allowed to transmit NRT UL data within the corresponding resources. Again, if these slave do not have any NRT UL data to transmit, the resources remain empty. However, if no NRT DL are available, a polling to the slaves NRT data for the UL transmission is required. Therefore, different methods are introduced and compared. Simulations prove, that the introduced method for scheduling NRT data performs very reliably and results in a balanced utilization of the reserved resources.

The authors assume that the master basically does not know details about pending NRT data on the slaves. A possible notification by the slaves is addressed only marginally.

Therefore our extension is to further investigate this signaling and to introduce an successive polling, which could increase the scheduling efficiency even further.

Our presented extensions might be also useful for transmission of acyclic RT data. These have to be transmitted with lowest possible latency. As a consequence, they are often treated as cyclic RT data and scheduled every communication cycle. As long as they do not have to be transmitted, the corresponding resource elements remain unused. In order to avoid this, we might adapt the method introduced before by some kind of prioritization. Due to this prioritization, acyclic RT data might interrupt the scheduling of NRT data such that only a minimal delay will occur. However, the necessary feasibility analysis will be addressed in future research.

Bibliography

[1] S. Dietrich, G. May, J. von Hoyningen-Huene, A. Müller, and G. Fohler, "Frame conversion schemes for cascaded wired / wireless communication networks of factory automation," *Mobile Networks and Applications*, vol. 22, no. 1, pp. 1–11, 2017.

[2] J. von Hoyningen-Huene, H. Heeren, L. Underberg, S. Dietrich, P. Kroos, A. Wulf, O. Wetter, and R. Kays, "Timing evaluation of cascaded industrial communication networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5497–5504, 2019.

[3] J. von Hoyningen-Huene, A. Müller, S. Dietrich, and G. May, "Comparison of wireless gateway concepts for industrial real-time-communication," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'16)*. IEEE, September 2016.

[4] S. Dietrich, G. May, J. von Hoyningen-Huene, A. Müller, and G. Fohler, "Latency in cascaded wired/wireless communication networks for factory automation," in *Industrial Networks and Intelligent Systems*. Springer, 2016.

[5] S. Dietrich, G. May, O. Wetter, H. Heeren, and G. Fohler, "Performance indicators and use case analysis for wireless networks in factory automation," in *IEEE International Conference on Emerging Technologies And Factory Automation (ETFA'2017)*. IEEE, September 2017.

[6] S. Dietrich, G. May, J. von Hoyningen-Huene, A. Müller, and G. Fohler, "Modeling offset adaptations for latency minimization in cascaded communication networks for factory automation," in *IEEE International Conference on Information, Communication and Automation Technologies (ICAT'2017)*. IEEE, October 2017.

[7] ——, "Anforderungsanalyse und Optimierungen kaskadierter Netzwerke für die Fertigungsautomatisierung," in *8th Kommunikation in der Automation (KommA 2017)*, 2017.

[8] S. Dietrich, G. May, L. Underberg, R. Kays, and G. Fohler, "Optimized resource allocation for cascaded communication networks in factory automation," in *IEEE International Conference on Industrial Technologies (ICIT'2017)*. IEEE, 2017.

[9] L. Underberg, R. Kays, S. Dietrich, and G. Fohler, "Towards hybrid wired-wireless networks in industrial applications," in *IEEE Industrial Cyber-Physical Systems (ICPS'2018)*. IEEE, 2018.

[10] L. Underberg, S. Dietrich, and R. Kays, "Optimierung eines Funksystems für hybride kaskadierte Netzwerke in der Fertigungsautomation," in *9th Kommunikation in der Automation (KommA 2018)*, 2018.

[11] *802 IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, Institute of Electrical and Electronics Engineers (IEEE) Std., Rev. 1.0, 07 2014.

[12] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of Things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 66–73, 2016.

[13] J. Weinman, "The economics and strategy of manufacturing and the cloud," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 6–11, 2016.

[14] Z. Shu, J. Wan, D. Zhang, and D. Li, "Cloud-integrated cyber-physical systems for complex industrial applications," *Mobile Networks and Applications*, vol. 21, no. 5, pp. 865–878, 2016.

[15] R. Drath and A. Horch, "Industrie 4.0: Hit or hype?" *IEEE Industrial Electronics Magazine*, vol. 8, no. 2, pp. 56–58, 2014.

[16] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *IEEE International Conference on Automation, Quality and Testing, Robotics (AQRT'14)*. IEEE, 2014.

[17] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[18] *IEC 62264-1 Enterprise-control system integration - Part 1: Models and terminology*, International Electrotechnical Commission (IEC) Std., Rev. 2.0, 05 2013.

[19] A. Roth, *Einführung und Umsetzung von Industrie 4.0*. Springer Berlin Heidelberg, 2016, ISBN 978-3-662-48504-0.

[20] V. Ç. Güngör and G. P. Hancke, *Industrial wireless sensor networks: Applications, protocols, and standards*. CRC Press, 2013, ISBN 978-1-1380-7620-4.

[21] ZVEI Automation Division, "Coexistence of wireless systems in automation technology," White Paper, 2009.

[22] VDI/VDE Society for Measurement and Automatic control (GMA), "VDI/VDE 2185 Part 1: Radio based communication in industrial automation," Verein Deutscher Ingenieure, Verband Der Elektrotechnik Elektronik Informationstechnik VDI/VDE, Tech. Rep., 2009.

[23] D. Jiang, X. Ying, Y. Han, and Z. Lv, "Collaborative multi-hop routing in cognitive wireless networks," *Wireless Personal Communications*, vol. 86, no. 2, pp. 901–923, 2016.

[24] Y. Zhang, S. He, and J. Chen, "Data gathering optimization by dynamic sensing and routing in rechargeable sensor networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1632–1646, 2016.

[25] J. T. Benra, *Software-Entwicklung für Echtzeitsysteme.* Springer-Verlag, 2009, ISBN 978-3-642-01596-0.

[26] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[27] P. Gaj, J. Jasperneite, and M. Felser, "Computer communication within industrial distributed environment - A survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 182–189, 2013.

[28] *IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Institute of Electrical and Electronics Engineers (IEEE) Std., Rev. 1.0, 07 2008.

[29] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Synchronize your watches: Part II: Special-purpose solutions for distributed real-time control," *IEEE Industrial Electronics Magazine*, vol. 7, no. 2, pp. 27–39, 2013.

[30] T. Sauter, "The three generations of field-level networks - evolution and compatibility issues," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3585–3595, 2010.

[31] *IEEE 802.3 Standard for Ethernet*, Institute of Electrical and Electronics Engineers (IEEE) Std., Rev. 1.0, 03 2016.

[32] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowski, D. Timmermann, and J. Schacht, "Survey on real-time communication via ethernet in industrial automation environments," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'18).* IEEE, 2014.

[33] *IEC 61784 Industrial communication networks - Profiles - Series of standard ISO/IEC*, International Electrotechnical Commission (IEC) Std., 2014.

[34] C. E. Pereira and P. Neumann, *Handbook of Automation.* Springer Berlin Heidelberg, 2009, ch. Industrial Communication Protocols, pp. 981–999, ISBN 978-3-540-78831-7.

[35] M. Felser, "Real-time ethernet-industry prospective," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1118–1129, 2005.

[36] ETSI Technical Committee on Electromagnetic compatibility and Radio spectrum Matters (ERM), "ETSI tr 102 889-2 v1.1.1 (2011-08): Electromagnetic compatibility and radio spectrum matters (erm); system reference document; short range devices (srd); part 2: Technical characteristics for srd equipment for wireless industrial applications using technologies different from ultra-wide band (uwb)," European Telecommunications Standards Institute ETSI, Tech. Rep., 2011.

[37] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 2, pp. 102–124, 2008.

[38] E. Tanghe, W. Joseph, L. Verloock, L. Martens, H. Capoen, K. Van Herwegen, and W. Vantomme, "The industrial indoor channel: large-scale and temporal fading at 900, 2400, and 5200 MHz," *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2740–2751, 2008.

[39] R. Croonenbroeck, A. Wulf, L. Underberg, W. Endemann, and R. Kays, "Parallel sequence spread spectrum: Bit error performance under industrial channel conditions," in *VDE International Conference on OFDM and Frequency Domain Techniques (ICOF'16)*. VDE, 2016.

[40] M. Düngen, F. Hofmann, and H. Schulze, "Bit error rate simulation studies for PSSS with multi-user detection for industrial multipath-fading environments," in *IEEE International Workshop on Factory Communication Systems (WFCS'17)*. IEEE, 2017.

[41] A. Frotzscher, U. Wetzker, M. Bauer, M. Rentschler, M. Beyer, S. Elspass, and H. Klessig, "Requirements and current solutions of wireless communication in industrial automation," in *IEEE International Conference on Communications Workshops (ICC'14)*. IEEE, 2014.

[42] Siemens AG, "IWLAN specification and further information," Latest view: 24.11.2017. [Online]. Available: http://w3.siemens.com/mcms/industrial-communication/en/industrial-wireless-communication/iwlan-industrial-wireless-lan/pages/iwlan.aspx

[43] *IEEE 802.11 Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Institute of Electrical and Electronics Engineers (IEEE) Std., Rev. 1.0, 12 2016.

[44] *WSAN Air Interface Specification Technical Specification*, PROFIBUS User Organisation (PNO) Std., Rev. 1.0, 2012.

[45] *IEEE 802.15.1 Standard for Information technology - Local and metropolitan area networks - Specific requirements - Part 15.1a: Wireless Medium Access Control*

(MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN), International Organization for Standardization(ISO) Std., Rev. 1.0, 6 2005.

[46] *IEC 62591 Industrial networks - Wireless communication network and communication profiles - WirelessHART*, International Electrotechnical Commission (IEC) Std., Rev. 2.0, 03 2016.

[47] *IEEE 802.15.4 Standard for Local and Metropolitan Area Networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANS)*, Institute of Electrical and Electronics Engineers (IEEE) Std., Rev. 1.0, 2011.

[48] R. Heynicke, D. Krush, G. Scholl, B. Kaercher, J. Ritter, P. Gaggero, and M. Rentschler, "IO-Link Wireless enhanced sensors and actuators for Industry 4.0 networks," in *AMA Conferences SENSOR and IRS²*. AMA, 2017.

[49] *IEC 61131-9:2015 Programmable controllers - Part 9: Single-drop digital communication interface for small sensors and actuators (SDCI)*, International Electrotechnical Commission (IEC) Std., Rev. 1.0, 02 2015.

[50] UK-China Science Bridges, "R&D on (B)4G Wireless Mobile Communications (UC4G)," Latest view: 24.11.2017. [Online]. Available: http://www.ukchinab4g.ac.uk/

[51] 3rd Generation Partnership Project (3GPP), "Mobile broadband standard releases 15 and 16," Latest view: 06.05.2018. [Online]. Available: http://www.3gpp.org/release-15

[52] European Commission, "Mobile and wireless communications Enablers for Twenty-twenty (2020) Information Society (METIS)," Latest view: 24.11.2017. [Online]. Available: https://www.metis2020.com/

[53] 5G Alliance for Connected Industries and Automation (5G-ACIA), "Designing 5G for industrial use," Working Party of ZVEI, Latest view: 06.05.2018. [Online]. Available: https://www.5g-acia.org/

[54] German Federal Ministry of Education and Research (BMBF), "A Parallel, Reliable and Secure Radio System for a Latency-Optimized Factory Automation (ParSec)," Latest view: 24.11.2017. [Online]. Available: www.parsec-projekt.de

[55] ——, "INDUSTRIAL RADIO," Latest view: 21.03.2017. [Online]. Available: http://industrialradio.de/Home/Index

[56] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks," in *ACM International Conference on Real-Time Networks and Systems (RTNS'16)*. ACM, 2016.

[57] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of IEEE 802.1 AS in large-scale industrial automation networks," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'17)*. IEEE, 2017.

[58] TTTech Computertechnik AG, "Time-Triggered Ethernet," Latest view: 16.09.2018. [Online]. Available: https://www.tttech.com/technologies/deterministic-ethernet/

[59] L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate-constrained traffic in TTEthernet using network calculus," *Real-Time Systems*, vol. 53, no. 2, pp. 254–287, 2017.

[60] C. Liu, F. Li, G. Chen, and X. Huang, "TTEthernet transmission in software-defined distributed robot intelligent control system," *Wireless Communications and Mobile Computing*, vol. 2018, no. 1, pp. 1–13, 2018.

[61] G. Cena, A. Valenzano, and S. Vitturi, "Hybrid wired/wireless networks for real-time communications," *IEEE Industrial Electronics Magazine*, vol. 2, no. 1, pp. 8–20, 2008.

[62] T. Sauter, J. Jasperneite, and L. L. Bello, "Towards new hybrid networks for industrial automation," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'09)*, vol. 9. IEEE, 2009.

[63] J. Liu, Y. Fang, and D. Zhang, "PROFIBUS-DP and HART protocol conversion and the gateway development," in *IEEE Conference on Industrial Electronics and Applications (ICIEA'07)*. IEEE, 2007.

[64] Y. Zhang, X. Feng, and Y. Guo, "Design of Ethernet-CAN protocol conversion module based on STM32," *International Journal of Future Generation Communication and Networking*, vol. 7, no. 1, pp. 89–96, 2014.

[65] R. Lange, R. S. de Oliveira, and F. Vasques, "A reference model for the timing analysis of heterogeneous automotive networks," *Computer Standards & Interfaces*, vol. 45, no. 3, pp. 13–25, 2016.

[66] E. Hossain, G. Chow, V. C. Leung, R. D. McLeod, J. Mišić, V. W. Wong, and O. Yang, "Vehicular telematics over heterogeneous wireless networks: A survey," *Computer Communications*, vol. 33, no. 7, pp. 775–793, 2010.

[67] D. Miorandi and S. Vitturi, "A wireless extension of Profibus DP based on the Bluetooth radio system," *Ad Hoc Networks*, vol. 3, no. 4, pp. 479–494, 2005.

[68] M. Alves and E. Tovar, "Real-time communications over wired/wireless profibus networks supporting inter-cell mobility," *Computer Networks*, vol. 51, no. 11, pp. 2994–3012, 2007.

[69] J. Kjellsson, A. E. Vallestad, R. Steigmann, and D. Dzung, "Integration of a wireless I/O interface for PROFIBUS and PROFINET for factory automation," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4279–4287, 2009.

[70] L. Seno, S. Vitturi, and C. Zunino, "Analysis of Ethernet Powerlink wireless extensions based on the IEEE802.11 WLAN," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 2, pp. 86–98, 2009.

[71] G. Fettweis and S. Alamouti, "5G: Personal mobile internet beyond what cellular did to telephony," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 140–145, 2014.

[72] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel *et al.*, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.

[73] L. Seno, S. Vitturi, and F. Tramarin, "Experimental evaluation of the service time for industrial hybrid (wired/wireless) networks under non-ideal environmental conditions," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'11)*. IEEE, 2011.

[74] M. Bertocco, C. Narduzzi, and F. Tramarin, "Estimation of the delay of network devices in hybrid wired/wireless real-time industrial communication systems," in *IEEE Instrumentation and Measurement Technology Conference (I2MTC'12)*. IEEE, 2012.

[75] J. Cecilio and P. Furtado, "Planning for time-critical heterogeneous wired plus wireless industrial networks," in *IEEE International Conference on Control Automation Robotics & Vision (ICARCV'14)*. IEEE, 2014.

[76] *IEC 61784-2 Industrial communication networks - Profiles - Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*, International Electrotechnical Commission (IEC) Std., Rev. 1.0, 02 2014.

[77] *IEC 62657-2 Industrial communication networks - Wireless communication networks - Part 2: Coexistence management*, International Electrotechnical Commission (IEC) Std., Rev. 1.0, 02 2015.

[78] L. Seno, S. Vitturi, and C. Zunino, "Real time ethernet networks evaluation using performance indicators," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'09)*. IEEE, 2009.

[79] G. Gamba, L. Seno, and S. Vitturi, "Performance indicators for wireless industrial communication networks," in *IEEE International Workshop on Factory Communication Systems (WFCS'10)*. IEEE, 2010.

[80] G. Scheible, D. Dzung, J. Endresen, and J. E. Frey, "Unplugged but connected - Design and implementation of a truly wireless real-time sensor/actuator interface," *IEEE Industrial Electronics Magazine*, vol. 1, no. 2, pp. 25–34, 2007.

[81] D. Schulze, A. Gnad, and M. Krätzig, "Anwendungen, Anforderungen und Validierung im BMBF-Förderprogramm "IKT 2020 - Zuverlässige drahtlose Kommunikation in der Industrie" (BZKI) - Anforderungsprofile im ZDKI," White Paper, 10 2016.

[82] *ISO 13849 Safety of machinery - Safety-related parts of control systems*, International Organization for Standardization(ISO) Std., Rev. 1.0, 2008.

[83] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer Berlin Heidelberg, 2003, ISBN 978-3-540-42184-5.

[84] J. Greifeneder and G. Frey, "Reactivity analysis of different networked automation system architectures," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'08).* IEEE, 2008.

[85] Y. Song, A. Koubaa, and F. Simonot, "Switched Ethernet for real-time industrial communication: Modelling and message buffering delay evaluation," in *IEEE International Workshop on Factory Communication Systems (WFCS'02).* IEEE, 2002.

[86] L. Liu and G. Frey, "Simulation approach for evaluating response times in networked automation systems," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07).* IEEE, 2007.

[87] *sercos the automation bus - Communication Specification*, sercos Working Group "TWG Communication" Std., Rev. 1.3.1-1.12, 2013.

[88] A. C. Wolf, "Verfahren zum Übertragen eines Daten-Worts," German Demande DE10 301 250 A1, july 29, 2004.

[89] H. Schwetlick and A. Wolf, "PSSS-parallel sequence spread spectrum a physical layer for RF communication," in *IEEE International Symposium on Consumer Electronics (ISCE'04).* IEEE, 2004.

[90] T. Sauter, "The continuing evolution of integration in manufacturing automation," *IEEE Industrial Electronics Magazine*, vol. 1, no. 1, pp. 10–19, 2007.

[91] M. Bauer, G. May, and V. Jain, "A wireless gateway approach enabling industrial real-time communication on the field level of factory automation," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'14).* IEEE, 2014.

[92] *ISO 11898:1-6 Road vehicles - Controller area network (CAN)*, International Organization for Standardization(ISO) Std., Rev. 1.0, 2015.

[93] *ISO 17458:1-5 Road vehicles - FlexRay communications system*, International Organization for Standardization(ISO) Std., Rev. 1.0, 2013.

[94] S. Home, S. Palis, and C. Diedrich, "Design of communication systems for networked control system running on PROFINET," in *IEEE International Workshop on Factory Communication Systems (WFCS'14)*. IEEE, 2014.

[95] C. Fischione, P. Park, P. Di Marco, and K. H. Johansson, "Design principles of wireless sensor networks protocols for control applications," in *Wireless Networking Based Control*. Springer, 2011, pp. 203–238.

[96] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE International Conference on Cloud Networking (CloudNet'15)*. IEEE, 2015.

[97] D. E. Knuth, *The art of computer programming*. Pearson Education, 1997, vol. 3.

[98] B. Vöcking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer, and D. Wagner, *Algorithms unplugged*. Springer Science & Business Media, 2010.

[99] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[100] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, 2016.

[101] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *IEEE International Symposium on Micro Machine and Human Science (MHS'95)*. IEEE, 1995.

[102] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, no. 1, pp. 1–10, 2008.

[103] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *International Conference on Evolutionary Programming (EP'98)*. Springer, 1998.

[104] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *IEEE Congress on Evolutionary Computation (CEC'00)*, vol. 1. IEEE, 2000.

[105] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[106] G. I. Evers, "An automatic regrouping mechanism to deal with stagnation in particle swarm optimization," Master's thesis, University of Texas - Pan American, 5 2009. [Online]. Available: http://www.georgeevers.org/thesis.pdf

[107] L. Underberg, R. Croonenbroeck, A. Wulff, W. Endemann, and R. Kays, "A PSSS approach for wireless industrial communication applying iterative symbol detection," *IEEE Transactions on Industrial Informatics*, 2018.

[108] R. Croonenbroeck, L. Underberg, A. Wulf, and R. Kays, "Measurements for the development of an enhanced model for wireless channels in industrial environments," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'17)*. IEEE, 2017.

[109] L. Underberg, R. Croonenbroeck, R. Kays, and R. Krämer, "ParSec: Wireless industrial communication first PSSS measurements in industrial environment," in *IEEE International Workshop on Factory Communication Systems (WFCS'17)*. IEEE, 2017.

[110] L. Underberg, E. L. Peter, R. Croonenbroeck, and R. Kays, "Bit error performance of a PSSS transmission system in industrial environments," in *IEEE International Workshop on Factory Communication Systems (WFCS'18)*. IEEE, 2018.

[111] J. von Hoyningen-Huene and A. Müller, "Scheduling acyclic traffic in cyclic and deterministic communication systems," in *IEEE International Workshop on Factory Communication Systems (WFCS'17)*. IEEE, 2017.

[112] N. A. Johansson, Y.-P. E. Wang, E. Eriksson, and M. Hessler, "Radio access for ultra-reliable and low-latency 5G communications," in *IEEE International Conference on Communication Workshop (ICCW'15)*. IEEE, 2015.

[113] B. Holfeld, D. Wieruch, T. Wirth, L. Thiele, S. A. Ashraf, J. Huschke, I. Aktas, and J. Ansari, "Wireless communication for factory automation: an opportunity for LTE and 5G systems," *IEEE Communications Magazine*, vol. 54, no. 6, pp. 36–43, 2016.

[114] A. Müller, S. Dietrich, and M. Düngen, "Verfahren und Vorrichtung zur optimierten Datenübertragung in Kommunikationssystemen mit zyklischem Verkehr," Erfindungsmeldung (SP13 001 EM Content Based Scheduling), 2018, German, Robert Bosch GmbH and Bosch Rexroth Internal.

Glossary

Actual Data

Actual data are transmitted by the slaves in order to provide feedback about their current status to the master. They typically have specific real-time to be used within the control loop.

Application

An application is a specific composition of functional units and their configuration that enable a particular process or production. It thus has definite physical properties and specific requirements to a communication network that connects the individual components.

Closed-Loop Control

In comparison to open-loop control, the command action from the controller is dependent on feedback from the process that is measured and returned.

Command Data

Command data are transmitted by the master in order to control its slaves. They typically have specific real-time requirements to be used within the control loop.

Computerized Numerical Control

Computerized numerical control (CNC) is the automation of machines by means of computers executing pre-programmed sequences of machine control commands.

Cyber-Physical Systems

Cyber-physical systems are defined by a combination of electronic and mechanical components along with software and information elements through specific communication networks.

Downlink

In telecommunication, the downlink (DL) describes the data flow direction from the master to its slaves. In industrial networks, this mainly refers to the transmission of command data.

Global Sampling Point

The global sampling point (GSP) is a synchronized point in time for all involved networks nodes that is specified on the application level. It represents a reference time to which all command and actual data are referred to.

Industrial Ethernet

Industrial Ethernet (IE) is a generic term for all communication networks that use the (partially extended or modified) Ethernet-standard IEEE 802.3 to connect industrial network nodes. These networks enable the use of standard components and concepts from the office domain in the industrial context.

Industry 4.0

Industry 4.0 (I4.0) refers to the fourth Industrial Revolution. The terms Connected Industries or Industrial Internet of Things are also often used synonymously. It describes the connection of industrial devices by modern communication systems. It combines concepts of cyber-physical system and the Internet of Things.

Internet of Things

Internet of Things (IoT) refers to the connection of objects to the Internet such that these objects can communicate independently over the Internet and fulfill various tasks.

Jitter

Jitter describes a variation of a certain time of reception or deviation in the synchronization.

Master

In industrial communication networks, the master typically is the control unit on which the control algorithm and application are executed. Alternatively, it could be a similar device that emulates corresponding control tasks. It provides command data for the slaves to be controlled and receives their actual data as feedback for the control loop.

Network Node

A network node describe any possible entity of the communication network. This could be either the master, a slave or any network device like a gateway.

OSI Model

The Open Systems Interconnection (OSI) model is an abstract model defined by the International Organization for Standardization (ISO) to characterize the communication functions of a communication network without detailed understanding of their underlying internal structure and technology.

Precision Time Protocol

The precision time protocol (PTP) is a standardized network protocol to synchronize distributed clocks and applications.

Reliability

Reliability comprisis the probability that an application can perform a required function under stated conditions for a given time interval. It can be evaluated by the success probability of the transmission with a certain data rate and within a certain delay.

Resource Element

A resource element (RE) is the smallest possible physical entity data could be assigned to. It depends on the corresponding physical media, the coding and multiple access strategy among others.

Slave

In industrial communication networks, a slave typically is an application entity that records data or performs actions. Alternatively, it could be a similar device that emulates corresponding tasks. It is controlled by the master by command data and provides actual data as feedback for the control loop.

Time-Sensitive Networking

Time-Sensitive Networking (TSN) is a set of standards of the IEEE 802.1 working group to extend Ethernet communication by real-time features.

Uplink

In telecommunication, the uplink (UL) describes the data flow direction from the slaves to the master. In industrial networks, this mainly refers to the transmission of actual data.

Acknowledgments

Parts of the research leading to the results presented in this work have received funding from the German Federal Ministry of Education and Research (BMBF) under the "Reliable wireless communication in industry (ZDKI)" project *A parallel, reliable and secure radio system for a latency-optimized factory automation* (ParSec, grant agreement no. 16KIS0225).

Summary

Methods of Evaluation and Improvement on Cascaded Wired and Wireless Real-Time Communication Networks for Factory Automation

Future industrial applications require determinism, reliability, short communication cycles, and precise synchronization as well as scalability and flexibility. Yet, no existing communication network can concurrently perform all of these functions. In this thesis, we present the use of cascaded communication networks consisting of hierarchically order wired and wireless subnetworks to overcome this problem. With a dedicated connection between individual subnetworks, cascaded communication networks can offer the advantages of both network technologies. Therefore, we must identify methods that maintain the advantages of heterogeneous subnetworks with little to no performance degradation. In addition, these methods should allow the application and the used hardware to remain ideally unchanged.

Chapter 1 - Introduction

In this chapter, we classify industrial communication and introduce the fundamentals of industrial communication concepts, such as the separation of downlink (DL) and uplink (UL) and the global sampling point (GSP). Additionally, we introduce industrial communication networks. For the wired networks, we focus on Industrial Ethernet (IE) based protocols and provide some example according to their real-time performance. In particular, isochronous real-time is most important for closed-loop applications in the area of factory automation. However, we identify that no wireless networks currently support these requirements. Accordingly, we specify the problem statement of developing and optimizing cascaded communication networks for industrial closed-loop control applications that we will focus on in this work and provide examples for related work. Finally, we give an outlook on the contribution of this thesis according to the problem statement.

Chapter 2 - Analysis of Factory Automation Applications

In this chapter, we present a detailed analysis of factory automation applications. We review typical performance indicators of communication networks as listed for example in the IEC 61784. By summarizing the requirements of industrial application that can be found in literature, we derive specific requirement profiles of the areas of condition monitoring (CM), process automation (PA), and factory automation (FA). Here, we reveal FA applications as the most demanding with respect to latency, synchronization, and reliability. Accordingly, we specify these requirements by providing a dedicated use case analysis of typical machine tool applications, printing machines, and packaging machines. According to our use case analysis, we adapt the list of typical performance parameters by specifying those most important and in line with the requirements of the applications. Therefore, it can be used as actual benchmark for industrial communication networks.

Chapter 3 - Modeling Industrial Communication

In this chapter, we focus on the evaluation of the timing behavior of communication network. Therefore, we introduce the importance of network modeling for their validation and verification and present a generic network model to represent arbitrary communication networks. This model is specified to analyze the corresponding timing and less complex than comparable network models. It is based on the finite resource space of any communication network and maps the overhead and data transmission as well as potential inactive times to it. We used this model to represent two exemplary communication networks and introduce different timing methods that could be required by the applications. Additionally, we present an algorithm for the timing analysis of arbitrary cascaded communication networks based on our introduced model. It is used to calculate the most relevant times within the data transmission for each slave and can be used for any required timing method. Finally, we use this algorithm to calculate the timing behavior of an exemplary cascaded network and reveal possible sources of additional latencies due to the cascading.

Chapter 4 - Latency Optimized Interface Between Heterogeneous Subnetworks

In this chapter, we compare different interface concepts for connecting arbitrary communication networks. Therefore, we start with a classification of network interfaces, revealing that we need gateways for cascaded communication networks in order to convert between arbitrary subnetworks. Accordingly, we review a tunnel and a proxy based gateway concept and present their advantages and disadvantages. A tunnel based gateway simply encapsulates incoming data frames within the frame structure of the subsequent network. This requires low conversion times but a high data rate to transmit the extended data. A proxy based gateway resolves the incoming frame structure and generates new frames according to the subsequent subnetwork. This probably require a longer conversion time but reduces the amount of data that have to be transmitted. According to this comparison, we propose the usage of a proxy based gateway for

cascaded networks in order to achieve low overall transmission latencies, especially for larger cascaded communication networks.

Chapter 5 - Frame Conversion Schemes for Cascaded Networks

In this chapter, we analyze the transition of data packets from one subnetwork to a subsequent subnetwork as one source of additional latencies in cascaded communication networks. In order to forward the data by arbitrary network protocols, the corresponding frames need to be converted to the corresponding subnetwork. According to frame duration, required number of frames, cycle time, and data reception interval, we can basically distinguish between a synchronous and an asynchronous frame conversion. For these basic frame conversion schemes we introduce different subcategories that also depend on the resource allocation. For each subcategory, we derive the corresponding worst case transmission latency and the jitter of the data transmission. We compare the transmission latency for two exemplary network protocols. This comparison reveals that a synchronous frame conversion does not introduce any additional jitter to the data transmission. Moreover, the corresponding latency could be minimized by an offset adaptation of the frame transmission to the corresponding data arrival. Using a simulation of a cascaded network, we were able to verify that the synchronous frame conversion is the most promising approach for the use in industrial communication systems.

Chapter 6 - Parameter Adaptations for Latency Reduction

In this chapter, we focus on the parameter adaption of the individual subnetworks in order to reduce the overall transmission latency in cascaded communication networks. We identify the individual offset parameter as well as the allocation of the transmission resources as specific parameters to be adapted according to the overall communication chain. Considering some introduced parameter bound, we present an analytical offset adaptation for individual subnetworks in order to optimize the data flow through the entire cascaded network. This enables a significant latency reduction compared to an arbitrary parameterization. However, its complexity growths exponentially with the size of the cascaded network. Therefore, we introduce additionally a heuristic offset adaptation method using well-known metaheuristics that provides similar results compared to the analytical adaptation. Moreover, based on a comparison of different allocation schemes, we propose a sequential resource allocation in order to enable further latency reductions. Based on this proposal, we introduce a latency optimized resource allocation algorithm for arbitrary cascaded communication networks. Together with an appropriate offset adaptation, we are able to reduce the transmission latency even further.

Chapter 7 - Timing Evaluation of Demonstrator Implementation

In this chapter we use a demonstrator implementation of the ParSec project to validate the methods developed in this work. The demonstrator represents a cascaded communication with three level, using *Sercos* subnetworks on levels one and three and *ParSec*

on level two. This represents a typical setup as expected for future applications. Since the actual *ParSec* physical layer (PHY) was not available for the measurements, a wire-based link emulator had to be used but did not emulate all aspects of an real wireless transmission. Several of the methods presented in this work are considered within the demonstrator at least partially. We measured the achievable synchronization accuracy and the end-to-end transmission latency to prove its usability for industrial closed-loop control application. The measurements reveal that a precise synchronization with a maximum deviation of 403.1 ns could be achieved with the demonstrator setup. This is suitable for the intended applications requiring a worst case deviation $\leq 1\,\mu s$. However, even with the introduced methods for latency reduction, a minor degradation of the worst-case end-to-end transmission latency emerges due to the cascading. This results from some deviation in the implementation compared to the assumptions made so far. Nevertheless, the measurements serve as additional proof of the concepts for network integration and latency reduction introduced in this work.

Chapter 8 - Outlook on Additional Methods for Improving Cascaded Networks

In this chapter we introduce three additional methods to improve cascaded communication networks for industrial applications. The first method concerns shifting the transmission overhead to timeslots where no data will anyhow be transmitted. This might reduce the transmission latency that results from overhead transmission. The second method describes a method for reducing the amount of data that have to be transmitted. Therefore, a content prediction has to be implemented. As a consequence, interferences to neighboring systems can be reduced while the data rate can be used more meaningfully. Finally, a specific resource allocation of non-real-time (NRT) data is reviewed and extended in order to further improve its performance and introduce a possible adaption for acyclic real-time (RT) data. However, these and other briefly introduced methods are out of the scope of this work and therefore left for future research.

Chapter 9 - Conclusions

In this chapter, we draw the conclusion of our work. We summarize once again the problem statement and evaluate our contribution to it. Finally, we provide remarks to the further usability if our work and review our outlook to future work.

Zusammenfassung (German Summary)

Methoden zur Evaluierung und Verbesserung von kaskadierten kabelgebundenen und kabellosen Echtzeit-Kommunikationsnetzwerken für die Fertigungsautomatisierung

Zukünftige industrielle Applikationen benötigen Determinismus, Zuverlässigkeit, kurze Kommunikationszyklen und präzise Synchronisation sowie auch Skalierbarkeit und Flexibilität. Dies muss von einem entsprechenden Kommunikationsnetzwerk unterstützt werden, welches alle Anforderungen gleichzeitig erfüllt. In dieser Arbeit wird hierfür die Verwendung von kaskadierten Kommunikationsnetzwerken vorgeschlagen, die aus hierarchisch angeordneten kabelgebundenen und kabellosen Teilnetzwerken bestehen, da diese mit einer dedizierten Verbindung der einzelnen Teilnetzwerke die Vorteile beider Netzwerktechnologien bieten können. Daher müssen Konzepte für solch eine Verbindung gefunden werden, welche die Vorzüge der heterogenen Teilnetzwerke ohne oder höchstens mit minimaler Leistungseinbuße erhalten. Darüber hinaus sollten es diese Konzepte ermöglichen, dass die Anwendung und die verwendete Hardware im Idealfall unverändert bleiben.

Kapitel 1 - Einleitung

Dieses Kapitel umfasst eine Klassifizierung industrieller Kommunikation sowie deren fundamentaler Konzepte, wie beispielsweise die strikte Unterteilung in Soll-Daten und Ist-Daten sowie den sogenannten Global-Sampling-Point. Zusätzlich werden Beispiele für verschiedene industrielle Kommunikationsnetzwerke aufgeführt. Dabei liegt bezüglich der kabelgebundenen Netzwerke der Fokus dieser Arbeit auf Industrial-Ethernet-basierten Protokollen. Typische Vertreter hiervon werden entsprechend ihrer Echtzeitfähigkeit spezifiziert. Dabei wird gezeigt, dass für Regelungsanwendungen aus dem Bereich der Fabrikautomatisierung besonders isochrones Echtzeitverhalten von größter Bedeutung ist. Weiterhin wird dargestellt, dass es gerade für diese Anwendungen derzeit noch keine kabellosen Netzwerke gibt, welche ein solches Zeitverhalten unterstützen. Dementsprechend wird in diesem Kapitel die Problemstellung dieser Arbeit bezüglich der Evaluierung und Optimierung kaskadierter Netzwerke für Regelungsanwendungen der Fabrikautomatisierung noch einmal spezifiziert und Beispiele für zugehörige Arbeiten genannt.

Abschließend wird ein Ausblick gegeben, wie diese Arbeit zur Problemlösung beitragen kann und deren weitere Struktur vorgestellt.

Kapitel 2 - Analyse von Anwendungen der Fertigungsautomatisierung

Dieses Kapitel präsentiert eine detaillierte Analyse von Anwendungen aus dem Bereich der Fabrikautomatisierung. Dafür werden zunächst allgemeine Leistungsindikatoren für in diesem Bereich typischerweise eingesetzte Kommunikationsnetze überprüft, wie sie beispielsweise in der IEC 61784 aufgelistet sind. Anschließend werden Anforderungsprofile für die Bereiche Zustandsüberwachung, Prozessautomatisierung und Fabrikautomatisierung aus vorhandener Literatur abgeleitet. Dabei zeigt sich unter anderem, dass Anwendungen aus dem Bereich Fabrikautomatisierung besonders hohe Anforderungen bezüglich der erlaubten Latenz, Synchronisation und Zuverlässigkeit haben. Um dies zu spezifizieren, wird ein gesondertes Anforderungsprofil typischer Anwendungen aus diesem Bereich mittels einer Use-Case-Analyse typischer Werkzeugmaschinen, Druckmaschinen und Verpackungsmaschinen konkretisiert. Entsprechend dieser Analyse wird eine Abbildung von den tatsächlichen Anforderungsprofilen auf die Leistungsindikatoren erstellt, sodass damit eine tatsächliche Bewertung eines Netzwerkes für eine spezifische Anwendung möglich wird.

Kapitel 3 - Modellierung industrieller Kommunikationsnetzwerke

Dieses Kapitel beschreibt eine Methode zur Evaluierung des Zeitverhaltens von Kommunikationsnetzwerken. Hierfür wird zunächst die Notwendigkeit der Modellierung von Kommunikationsnetzwerken zur Validierung und Verifikation beschrieben und ein generisches Modell zur Abbildung beliebiger Kommunikationsnetzwerke vorgestellt. Dieses Modell ist speziell auf die Analyse des Zeitverhaltens angepasst und weißt dabei eine geringere Komplexität als vergleichbare Modelle auf. Es basiert auf dem limitierten Ressourcen-Raum eines jeden Kommunikationsnetzwerkes und bildet darauf sowohl die Datenübertragung als auch potentielle Inaktivität ab. Es wird im Weiteren verwendet, um zwei exemplarische Kommunikationsnetzwerke abzubilden und verschiedene Übertragungsmodi der Applikationen einzuführen. Darüber hinaus wird davon ein Algorithmus zur Evaluierung des Zeitverhaltens von beliebigen kaskadierten Netzen abgeleitet, welcher basierend auf den wichtigsten Referenzzeiten in der Datenübertragung aller Slaves ermittelt, ob ein gewünschtes Zeitverhalten einer Anwendung erfüllt werden kann. Abschließend wird mittels dieses Algorithmus das Zeitverhalten eines beispielhaften kaskadierten Netzwerks berechnet und mögliche Quellen von zusätzlicher Latenz aufgezeigt, welche durch das Kaskadieren heterogener Netzwerke entstehen.

Kapitel 4 - Latenzoptimierte Schnittstelle zwischen heterogenen Teilnetzwerken

In diesem Kapitel wird die Schnittstelle zwischen zwei beliebigen Kommunikationsnetzwerken analysiert. Hierfür wird zunächst in einer Klassifizierung verschiedener Netzwerkschnittstellen gezeigt, dass für kaskadierte Kommunikationsnetzwerke vor allem

Gateways benötigt werden, welche die Umsetzung der Daten von verschiedenen Teilnetzwerken zueinander ermöglicht. Dementsprechend werden anschließend ein Tunnel- und ein Proxy-basiertes Konzept für Gateways miteinander verglichen und deren Vor- und Nachteile aufgezeigt. Bei einem Tunnel-basierten Gateway werden die erhaltenen Daten-Frames vollständig und unverändert im Rahmen des nachfolgenden Teilnetzwerkes verkapselt und versendet, weshalb in diesem Fall die Konvertierungszeit relativ gering ausfällt, die zu übertragende Datenmenge dafür aber umso höher ist. Bei einem Proxy-basiertem Gateway werden die Datenframes hingegen vollständig in einzelne Datenpakete aufgeteilt und ausschließlich die notwendigen Daten im Rahmen des nachfolgenden Netzwerks übertragen. Somit ist hier ein höherer Konvertierungsaufwand zu erwarten, wohingegen die zu übertragende Datenmenge reduziert ist. In Bezug auf Latenz ist daher der Proxy-basierte Ansatz zu bevorzugen, da hiermit vor allem bei größeren kaskadierten Netzen die Gesamtverzögerung minimiert werden kann.

Kapitel 5 - Schemata zu Framekonvertierung für kaskadierte Netzwerke

In diesem Kapitel wird die Datenübertragung von einem in das nachfolgende Teilnetzwerk und die dabei entstehenden Verzögerungen genauer betrachtet. Damit Daten entsprechend weitergeleitet werden können, müssen die einzelnen Übertragungsmethoden der Teilnetzwerke aufeinander abgebildet werden. Abhängig von der Frame-Dauer, der Anzahl an notwendigen Frames, der Zykluszeit und der Datenempfangsrate der unterschiedlichen Netzwerke kann zwischen einer synchronen und asynchronen Konvertierung unterschieden werden. Für diese grundlegenden Konvertierungsschemata werden mögliche Unterkategorien vorgestellt, welche unter anderem auch von der Ressourcen-Zuteilung abhängen und diese entsprechend ihrer maximal möglichen Latenz und dem zugehörigen Jitter in der Datenweiterleitung bewerten. Basierend auf einer Beispielverknüpfung zweier Netzwerke wird ein Vergleich erstellt, welcher aufzeigt, dass eine synchrone Framekonvertierung den auftretenden Jitter minimalisiert und somit mit dieser Methode die Latenz in der Datenübertragung durch entsprechende Optimierung am geringsten gehalten werden kann. Eine zusätzliche Simulation verifiziert, dass die synchrone Framekonvertierung für kaskadierte Kommunikationsnetze am vielversprechendsten ist.

Kapitel 6 - Parameteranpassung zur Latenzminimierung

In diesem Kapitel wird eine Parameteranpassung der einzelnen Teilnetzwerke als weitere Methode zur Latenzreduzierung in kaskadierten Kommunikationsnetzen betrachtet. Hierbei werden besonders die einzelnen Offset-Parameter sowie die Zuteilung der Übertragungsressourcen als spezifische Einflussgrößen hervorgehoben, die im Hinblick auf die Gesamtkommunikation angepasst werden müssen. Bezugnehmend auf mögliche Parametergrenzen wird dann eine analytische Offsetanpassung der einzelnen Teilnetzwerke vorgestellt, welche den Gesamtdatenfluss durch das kaskadierte Netzwerk optimiert und damit eine signifikante Reduktion der Übertragungslatenz im Vergleich zu einer beliebigen Parametrierung ermöglicht. Da allerdings die Komplexität dieses Verfahrens mit der Größe des kaskadierten Netzwerkes exponentiell ansteigt, wird daher im Wei-

teren auch eine heuristische Methode zur Offsetoptimierung vorgestellt. Diese basiert auf bekannten Metaheuristiken und ermöglicht das Erzielen vergleichbarer Ergebnisse wie mit der analytischen Optimierung. Im Weiteren wird zusätzlich, basierend auf einem Vergleich verschiedener Reservierungskonzepte der Übertragungs-Ressourcen, eine sequentielle Zuteilung vorgeschlagen, da diese die Übertragungslatenz weiter reduzieren kann. Entsprechend dieses Vorschlags wird ein Algorithmus zur optimierten Ressourcen-Zuordnung in beliebigen kaskadierten Kommunikationsnetzen vorgestellt, mit dessen Hilfe und einer geeigneten Offsetanpassung die Latenz noch einmal weiter reduziert werden kann.

Kapitel 7 - Auswertung des Zeitverhaltens einer Demonstratorimplementierung

In diesem Kapitel wird eine Demonstratorimplementierung aus dem Rahmen des ParSec-Projektes vorgestellt, welche zur Validierung von in dieser Arbeit vorgestellten Konzepten genutzt werden kann. Dieser Demonstrator repräsentiert ein kaskadiertes Kommunikationsnetzwerk mit drei Hierarchieebenen, bestehend aus den Teilnetzwerken *Sercos* auf den Ebenen eins und drei und *ParSec* auf Ebene zwei. Dies repräsentiert einen typischen Aufbau, wie er für zukünftige und zur Erweiterung bestehender Applikationen zu erwarten ist. Viele der in dieser Arbeit vorgestellten Konzepte wurden bei der Implementierung des Demonstrators zumindest teilweise berücksichtigt. Da der tatsächliche Physical-Layer des *ParSec*-Systems zum Zeitpunkt der Messungen noch nicht verfübar war, wurde dieser möglichst reell mittels eines kabelbasierten Link-Emulators nachgebildet. Basierend darauf wurden zum einen die erreichbare Synchronisationsgenauigkeit und zum anderen die Ende-zu-Ende-Übertragungslatenz gemessen, um die Nutzung eines solchen Netzwerks für Regelungsapplikationen aus dem Bereich der Fabrikautomatisierung zu validieren. Dabei haben die Messungen ergeben, dass in der Demonstratorimplementierung eine präzise Synchronisation mit einer maximalen Abweichung voneinander von $403,1\text{ns}$ erreicht werden kann. Dies wäre ausreichend für die angedachten Applikationen, die eine maximale Abweichung $\leq 1\,\mu\text{s}$ fordern. Allerdings wird trotz der implementierten Konzepte zur Latenzreduktion eine geringe Verschlechterung der Ende-zu-Ende-Latenz gegenüber einer reinen kabelgebundenen Übertragung festgestellt, welche sich aus der Kaskadierung ergibt. Verglichen mit den bisherigen Ergebnissen entsteht diese unter anderem dadurch, dass die reale Implementierung an einigen Stellen von den getroffenen Annahmen, vor allem in Bezug auf das *ParSec*-System, abweicht. Dennoch konnten auch mittels des Demonstrators die in dieser Arbeit vorgestellten Konzepte zur verbesserten Integration und Reduktion der Übertragungslatenz erneut zum Teil verifiziert werden.

Kapitel 8 - Zusätzliche Konzepte zur Verbesserung von kaskadierten Netzwerken

In diesem Kapitel werden hauptsächlich drei weitere Konzepte zur Verbesserung kaskadierter Kommunikationsnetzwerke für industrielle Anwendungen vorgestellt. Das erste Konzept beschreibt eine Verschiebung der Übertragung des Overhead zu einem Zeitpunkt, an dem eine Datenübertragung ohnehin ausgeschlossen ist. Dadurch könnte weitere Latenz eingespart werden, welche sich ansonsten aus einer verzögerten Datenüber-

tragung durch den Overhead ergibt. Das zweite Konzept beschreibt eine Möglichkeit, mittels Vorhersagen die Menge an Daten reduzieren zu können, welche übertragen werden müssen. Als Konsequenz könnten die gegenseitigen Beeinflussungen mit benachbarten Systemen reduziert werden, während freie Übertragungs-Ressourcen anderweitig genutzt werden können. Abschließend wird eine spezifische Reservierung von Ressourcen für Nicht-Echtzeitdaten überprüft und dabei eine mögliche Leistungsverbesserung und eine Erweiterung zur Abbildung von azyklischen Echtzeitdaten vorgeschlagen. Allerdings befinden sich diese und weitere kurz vorgestellte Konzepte nicht mehr im unmittelbaren Fokus dieser Arbeit und werden daher nur kurz andiskutiert und für zukünftige Arbeiten vorgemerkt.

Kaiptel 9 - Fazit

In diesem Kapitel wird das Fazit dieser Arbeit gezogen. Dazu wird erneut die Problemstellung kurz zusammengefasst und der Beitrag dieser Arbeit zu dessen Lösung bewertet. Abschließende Bemerkungen zur zukünftigen Nutzung der vorgestellten Konzepte und eine Bewertung des gezeigten Ausblicks runden die Arbeit ab.