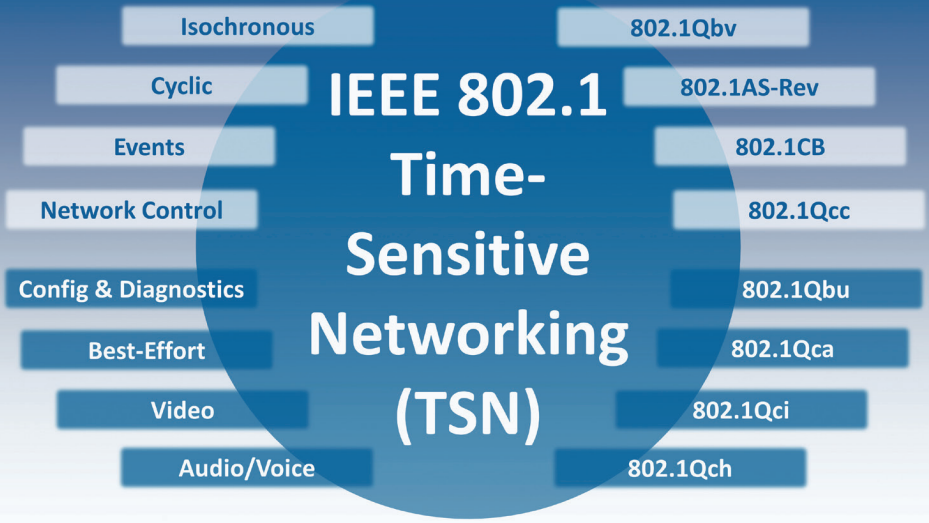
STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

FREDERICK PRINZ

Wandelbare, echtzeitfähige Kommunikationsinfrastruktur für Cyber-Physische Produktionssysteme



Universität Stuttgart

Fraunhofer IPA

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 125

Frederick Prinz

Wandelbare, echtzeitfähige Kommunikationsinfrastruktur für Cyber-Physische Produktionssysteme

FRAUNHOFER VERLAG

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 07 11/9 70-11 01
info@ipa.fraunhofer.de; www.ipa.fraunhofer.de

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl[1,2]
Univ.-Prof. Dr.-Ing. Kai Peter Birke[1,4]
Univ.-Prof. Dr.-Ing. Marco Huber[1,2]
Univ.-Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer[1,5]
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl[3]
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper[1,2]

[1] Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
[2] Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart
[3] Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) der Universität Stuttgart
[4] Institut für Photovoltaik (IPV) der Universität Stuttgart
[5] Institut für Energieeffizienz in der Produktion (EEP) der Universität Stuttgart

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de abrufbar.

ISBN: 978-3-8396-1743-4

D 93

Zugl.: Stuttgart, Univ., Diss., 2021

Druck und Weiterverarbeitung:
Fraunhofer Verlag, Mediendienstleistungen

Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Nobelstraße 12
70569 Stuttgart
verlag@fraunhofer.de
www.verlag.fraunhofer.de

als rechtlich nicht selbständige Einheit der

Fraunhofer-Gesellschaft zur Förderung
der angewandten Forschung e.V.
Hansastraße 27 c
80686 München
www.fraunhofer.de

Dissertation

Wandelbare, echtzeitfähige Kommunikationsinfrastruktur für Cyber-Physische Produktionssysteme

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik der Universität Stuttgart zur Erlangung der Würde eines Doktoringenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Frederick Prinz

aus Mechernich

Hauptberichter:	Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Mitberichter:	Prof. Dr.-Ing. Ludwig Leurs
Tag der mündlichen Prüfung:	12. April 2021

Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart

2021

Vorwort des Autors

Die vorliegende Arbeit entstand während meiner Tätigkeit als Doktorand am Forschungscampus der Robert Bosch GmbH in Renningen. Zeitgleich war ich an der Universität Stuttgart zur Promotion eingeschrieben.

Ich bedanke mich ganz herzlich bei meinem Doktorvater, Herrn Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl, für die Betreuung der Arbeit und die Übernahme des Hauptberichts.

Herrn Prof. Dr.-Ing. Ludwig Leurs danke ich herzlich für seine Bereitschaft, den Mitbericht zu übernehmen.

Für die wertvolle Unterstützung während der Promotionszeit und für die kritische Durchsicht meiner Arbeit danke ich meinem Betreuer Michael Schöffler, meinem Mentor Christian Kircher und meinem Gruppenleiter Matthias Meier recht herzlich.

Darüber hinaus bedanke ich mich bei meinen Arbeitskollegen der Robert Bosch GmbH und der Bosch Rexroth AG, mit denen ich in den drei Jahren zusammenarbeiten durfte. Besonderer Dank gilt dabei Stefan Benkner und Andreas Eckhardt für den wertvollen Austausch und die sehr gute Zusammenarbeit.

Zum Schluss gilt ein ganz besonderer Dank meinen Eltern, die mich zu jeder Zeit bedingungslos unterstützt haben und mir mit Rat und Tat zur Seite standen.

Frederick Prinz

Kurzfassung

Heutzutage agieren Unternehmen in einem immer volatileren Umfeld mit einem unmittelbaren Einfluss auf die Produktionssysteme. Um die Wandelbarkeit von zukünftigen Produktionssystemen zu verbessern wird daher eine Automation basierend auf Cyber-Physischen Systemen (CPS) angestrebt. Diese CPS-basierte Automation zeichnet sich durch eine einheitliche, wandelbare und echtzeitfähige Kommunikationsinfrastruktur aus.

Stand heute existieren verschiedene Lösungsansätze mit unterschiedlichen Technologien zur Etablierung einer echtzeitfähigen Kommunikationsinfrastruktur. Diese Lösungen erfüllen allerdings die Anforderungen an eine zukünftige Kommunikationsinfrastruktur nur teilweise und stellen daher keine ganzheitliche Lösung dar. Insbesondere der Aspekt der Wandelbarkeit mit einer dynamischen Konfiguration von Echtzeitverbindungen zur Laufzeit wird heutzutage nur bedingt unterstützt. Dadurch wird die Vision einer CPS-basierten Automation und die Wandelbarkeit von zukünftigen Produktionssystemen allgemein eingeschränkt.

Als Grundlage für eine CPS-basierte Automation wird in dieser Arbeit eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur für zukünftige Produktionssysteme vorgestellt. Diese Kommunikationsinfrastruktur ermöglicht eine deterministische Kommunikation mit geringer Latenz und erfüllt die harten Echtzeitanforderungen in Produktionssystemen. Gleichzeitig wird die dynamische Konfiguration von Echtzeitverbindungen zur Laufzeit ermöglicht. Mit dem Fokus auf einer ganzheitlichen Lösung basiert die vorgestellte Kommunikationsinfrastruktur auf den drei wesentlichen Aspekten: Echtzeitkommunikation, Parametrierung und Netzwerkkonfiguration.

Die Echtzeitkommunikation basiert auf der neuen IEEE Technologie Time-Sensitive Networking (TSN, IEEE 802.1). TSN ermöglicht eine einheitliche, herstellerunabhängige Kommunikationsbasis für den echtzeitkritischen und bestmöglichen Datenverkehr im Netzwerk. Dazu werden die erforderlichen TSN Mechanismen in die echtzeitfähigen Assets integriert und in einem TSN SDK (Software Development Kit) zusammengefasst.

Zur einheitlichen Parametrierung der Echtzeitkommunikation wird das Konzept der echtzeitfähigen I4.0 Komponente vorgestellt. Das neue Konzept beschreibt ein echtzeitfähiges Asset mit einer sogenannten Verwaltungsschale, d.h. einer virtuellen digitalen Repräsentanz des Assets.

Diese Verwaltungsschale spezifiziert die individuellen Fähigkeiten des Assets und beinhaltet zusätzliche Konfigurationsparameter für die Echtzeitkommunikation.

Zur automatischen Konfiguration der Echtzeitverbindungen im Netzwerk wird ein zentraler SDN (Software-defined Networking) Controller erweitert. Dieser Controller etabliert dynamische, redundante und komplexe TSN Verbindungen zwischen verschiedenen echtzeitfähigen Assets zur Laufzeit. Dabei werden sowohl die TSN Switches in der Netzwerkinfrastruktur als auch die echtzeitfähigen Assets konfiguriert.

Schließlich wird die Machbarkeit der neuen wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur durch eine prototypische Implementierung gezeigt. Darauf aufbauend wird die Performance des neuen TSN SDKs evaluiert und die Integration ins Engineering beschrieben.

Schlagwörter: Industrie 4.0, Cyber-Physisches Produktionssystem, Time-Sensitive Networking, I4.0 Komponente, Software-Defined Networking

Short Summary

Nowadays, manufacturing companies operate in a more and more volatile environment with immediate consequences for the production systems. Therefore, future production systems have to become more advanced in terms of reconfigurability. Within the context of the fourth industrial revolution an automation based on cyber-physical systems (CPS) is striven for. This CPS-based automation is characterized by a uniform, reconfigurable, and real-time capable communication infrastructure.

As of today, there are different technologies to establish a real-time capable communication infrastructure. However, the existing approaches only partially fulfill the future requirements and do not provide a holistic solution. In particular, the dynamic configuration of real-time connections during runtime is only supported to some degree. Thus, the vision of a CPS-based automation and the reconfigurability of future production systems in general is limited.

This doctoral thesis contributes towards the goal of a CPS-based automation. More precisely, a reconfigurable and real-time capable communication infrastructure for future production systems is introduced. The presented approach enables a deterministic communication with bounded low-latency and satisfies the hard real-time requirements within production systems. Simultaneously, it enables the dynamic configuration of real-time connections during runtime. With the focus on a holistic solution the presented approach is separated into three main aspects: real-time communication, asset parameterization, and network configuration.

The real-time communication is based on the new IEEE technology Time-Sensitive Networking (TSN, IEEE 802.1). TSN yields as single vendor-independent communication basis for real-time as well as best-effort traffic within the same network. The required TSN mechanisms are integrated into real-time assets and summarized within a TSN SDK (Software Development Kit).

To enable a consistent parameterization of the real-time communication, the concept of real-time I4.0 components is introduced. This concept describes real-time assets with a so-called asset administration shell, i.e. a standardized virtual representation of the asset's capabilities. This asset administration shell contains the individual capabilities as well as additional parameters to establish a real-time communication with others.

To enable an automatic network configuration, a central SDN (Software-defined Networking) controller is extended. This controller establishes dynamic, redundant, and complex TSN connections between multiple real-time assets during runtime. Thereby, it configures the TSN switches in the network infrastructure as well as the real-time assets in the production system.

Finally, the feasibility of the presented approach is shown. Therefore, the reconfigurable and real-time capable communication infrastructure is prototypically implemented and the performance of the newly introduced TSN SDK is evaluated. Moreover, the seamless integration into an existing engineering method is described.

Keywords: Industrial Internet of Things, Industry 4.0, Cyber-Physical Production System, Time-Sensitive Networking, I4.0 Component, Software-Defined Networking

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
App	Anwendung
AVB	Audio-Video Bridging
BPMN	Business Process Model and Notation
CNC	Central Network Configuration
CoAP	Constrained Application Protocol
CPPS	Cyber-Physisches Produktionssystem
CPS	Cyber-Physisches System
CUC	Central User Configuration
DAN	Double-Attached Node
DDS	Data Distribution Service
DetNet	Deterministic Networking
FlexE	Flexible Ethernet
GCL	Gate Control List
HSR	High-Availability Seamless Redundancy
HTTP	Hypertext Transfer Protocol
I4.0	Industrie 4.0
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPC	Industrie-PC
IT	Information Technology
JSON	JavaScript Object Notation
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
MC	Motion Control
MIB	Management Information Base

MQTT	Message Queuing Telemetry Transport
NETCONF	Network Configuration Protocol
NIC	Netzwerkadapter
NTP	Network Time Protocol
OPC UA	OPC Unified Architecture
OSI	Open Systems Interconnection
OT	Operation Technology
PCP	Priority Code Point
PRP	Parallel Redundancy Protocol
PTP	Precision Time Protocol
PubSub	Publish/Subscribe
QoS	Quality of Service
RAP	Resource Allocation Protocol
RSTP	Rapid Spanning Tree Protocol
RT-App	Echtzeitanwendung
RTPS	Real-time Publish-Subscribe Protocol
SDK	Software Development Kit
SDN	Software-defined Networking
SMT	Satisfiability Modulo Theories
SNMP	Simple Network Management Protocol
SRP	Stream Reservation Protocol
SSH	Secure Shell
TAS	Time-Aware Shaper
TCP	Transmission Control Protocol
TSN	Time-Sensitive Networking
TTS	Time-Triggered Send
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VLAN	Virtual Local Area Network
VT	Virtuelle Topologie
VWS	Verwaltungsschale
YANG	Yet Another Next Generation

Abbildungsverzeichnis

Kapitel 1

Einleitung und Problemstellung

1.1 Einleitung

Produzierende Unternehmen agieren heutzutage in einem volatilen Umfeld mit verschiedenen Einflussfaktoren (Bild 1.1). Diese Einflussfaktoren werden auch als **Wandlungstreiber** bezeichnet, wobei man zwischen internen und externen Treibern in Bezug auf das Unternehmen unterscheidet (ElMaraghy et al. 2009; Wiendahl et al. 2014; Löffler 2011). Darüber hinaus lässt sich die Vielzahl von Wandlungstreibern den wesentlichen Bereichen Märkte, Produkte und Produktionstechnologien zuordnen (Löffler 2011).

Märkte Globalisierte Märkte mit komplexen Abhängigkeiten vereinen mehrere wirtschaftliche und politische Wandlungstreiber (Koch 2016). Dabei sind insbesondere die Internationalisierung der Märkte und die Finanzierung am weltweiten Kapitalmarkt hervorzuheben (Koch 2016). Des Weiteren existieren politische Rahmenbedingungen (z.B. Arbeits-/Umweltgesetze) und geopolitische Risiken (z.B. Handelskonflikte) (IndustryWeek 2016; PwC 2020). Die Folge sind starke und vermehrt kurzfristige Absatzschwankungen, die für produzierende Unternehmen nur schwer vorherzusehen sind (Spath et al. 2013).

Produkte Die Produktentwicklung ist geprägt durch eine steigende Nachfrage nach individualisierten und personalisierten Produkten (Wang et al. 2017). In Kombination mit den weiterhin hohen Absatzzahlen stehen die produzierenden Unternehmen vor der Herausforderung der kundenindividuellen Massenproduktion (Tseng et al. 2014). Darüber hinaus führt die Schnelllebigkeit von Technologien und das volatile Umfeld allgemein zu verkürzten Produktentwicklungszyklen (Nyhuis et al. 2010).

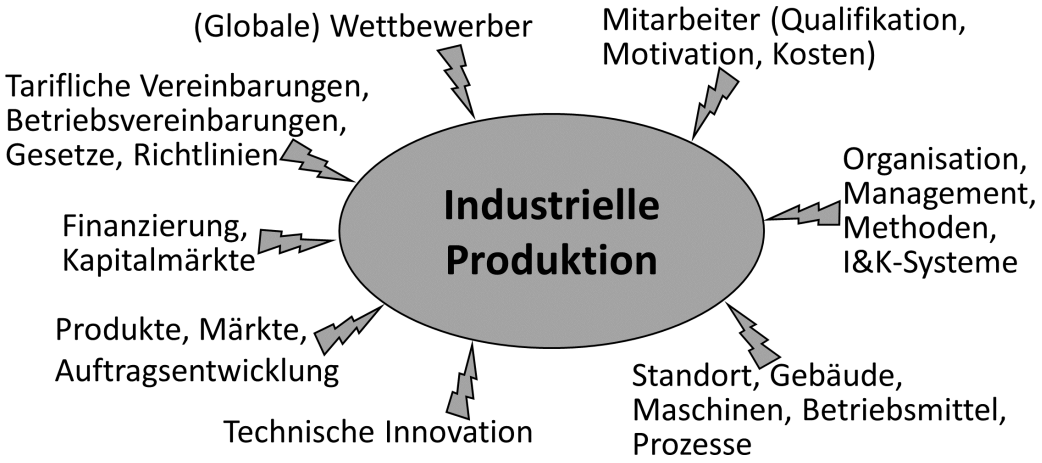


Abbildung 1.1: Wandlungstreiber in der industriellen Produktion (Westkämper et al. 2008).

Produktionstechnologien Die Schnelllebigkeit von Technologien ist auch ein Wandlungstreiber für die Produktionstechnologien innerhalb einer Fabrik. Technische Innovationen ermöglichen eine effizientere bzw. ökonomischere Produktion und einen damit verbundenen Wettbewerbsvorteil gegenüber anderen Unternehmen. Dafür ist allerdings eine kontinuierliche Weiterentwicklung der einzelnen Assets, der Produktionsinfrastruktur, der organisatorischen Struktur und des standortübergreifenden Produktionsnetzwerks erforderlich.

Die verschiedenen Wandlungstreiber aus den drei wesentlichen Bereichen Märkte, Produkte und Produktionstechnologien haben einen unmittelbaren Einfluss auf die Produktionssysteme innerhalb eines Unternehmens (Löffler 2011). Daher ist die **Wandelbarkeit** des Produktionssystems für Unternehmen in der Zukunft von entscheidender Bedeutung (Spath et al. 2013; Steegmüller et al. 2014; Heinen et al. 2010).

Grundlage für die Wandelbarkeit von Produktionssystemen bildet eine umfassende Kommunikationsinfrastruktur. Diese Infrastruktur ermöglicht die Vernetzung von Assets zum Datenaustausch, zur Datenaggregration und zur Synchronisation. Stand heute hat sich eine hierarchische Kommunikationsinfrastruktur entsprechend der **Automatisierungspyramide** etabliert (Bild 1.2, links) (Meudt et al. 2017; Norm IEC 62264-3). Diese Kommunikationsinfrastruktur unterteilt sich insbesondere in ein sogenanntes IT- (Information Technology) und OT-Netzwerk (Operation Technology) mit unterschiedlichen Anforderungen (Schriegel et al. 2018; Lennvall et al. 2017).

IT-Netzwerk Das Ethernet-basierte IT-Netzwerk dient zur Vernetzung von Assets auf den höheren Ebenen der Automatisierungspyramide. Dieses Netzwerk umfasst die gesamte Fabrik mit dem Produktionssystem und verfügt darüber hinaus über eine Anbindung an

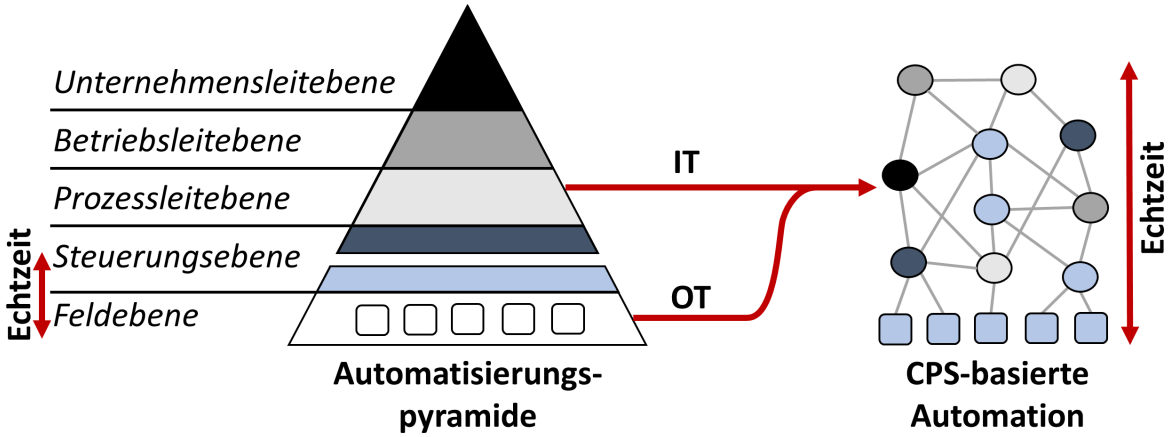


Abbildung 1.2: Wandel von der hierarchischen Automatisierungspyramide hin zu einer CPS-basierten Automation (VDI/VDE 2013).

das Internet. Dabei liegt der Fokus der etablierten Kommunikation auf der Skalierbarkeit, Informationsintegration und Security (Givehchi et al. 2013).

OT-Netzwerk Im Gegensatz dazu ermöglicht das OT-Netzwerk die Anbindung und Steuerung von Assets auf der Feldebene. Die resultierenden Subnetze sind lokal begrenzt und gewährleisten eine deterministische Kommunikation mit geringer Latenz. Stand heute haben sich verschiedene, allerdings untereinander inkompatible Industrial Ethernet Protokolle, wie z.B. Sercos, EtherCAT oder Profinet, für das OT-Netzwerk etabliert (Danielis et al. 2014). Bei diesen Protokollen liegt der Fokus auf der Echtzeitfähigkeit, Verfügbarkeit und Safety (Lennvall et al. 2017).

In Zukunft ist es fraglich, ob eine hierarchische Kommunikationsinfrastruktur entsprechend der Automatisierungspyramide den Anforderungen des Produktionssystems an die Wandelbarkeit genügt (Jahn 2017; Dumitrescu et al. 2015). Im Kontext der vierten industriellen Revolution, kurz Industrie 4.0 (I4.0) (Wollschlaeger et al. 2017; Roth 2016; Hermann et al. 2016), wird daher der Wandel von der etablierten Automatisierungspyramide hin zu einer sogenannten **CPS-basierten Automation** angestrebt (Schriegel et al. 2018; Kagermann 2017; Jeschke et al. 2017; Vogel-Heuser 2017; Drath et al. 2014). Dabei beschreibt der Begriff CPS (Cyber-Physisches System) allgemein die Vernetzung und Interaktion von physikalischen Assets mit integrierten softwareintensiven Systemen (Geisberger et al. 2012; Lee 2008). Mit dem Fokus auf der industriellen Produktion wird auch der Begriff Cyber-Physisches Produktionssystem (CPPS) verwendet (Monostori 2014). Ein wesentliches Merkmal der CPS-basierten Automation ist die sogenannte Netzwerkkonvergenz. Dabei konvergieren die bisherigen IT- und OT-Netzwerke zu einer einheitlichen Kommunikationsinfrastruktur mit einer flachen Hierarchie

(Bild 1.2, rechts) (Jahn 2017; VDI/VDE 2013; Lipnicki et al. 2018). Die resultierende Kommunikationsinfrastruktur umfasst das gesamte Produktionssystem in einer Fabrik und muss daher durchgehend echtzeitfähig sein.

1.2 Problemstellung

Ziel dieser Arbeit ist die Entwicklung einer **wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur** für zukünftige Produktionssysteme. Dabei ergeben sich Stand heute folgende drei Forschungsfragen (Bild 1.3):

Echtzeitkommunikation Ein wesentliches Merkmal der CPS-basierten Automation ist die durchgehende Echtzeitfähigkeit der gesamten Kommunikationsinfrastruktur. Daher muss eine geeignete Kommunikationstechnologie gewählt und die erforderlichen Kommunikationsmechanismen in die Netzwerkkomponenten und Assets integriert werden.

Wie erfolgt die herstellerunabhängige und gleichzeitig durchgängig echtzeitfähige Kommunikation zwischen verschiedenen Assets im Produktionssystem?

Parametrierung Durch die Entwicklung hin zu einem umfassenden CPS rücken in sich abgeschlossene, herstellerspezifische Systeme in den Hintergrund. Stattdessen muss die Integration und Kombination von Assets verschiedener Hersteller ermöglicht werden.

Wie erfolgt die einheitliche und herstellerunabhängige Parametrierung der erforderlichen Kommunikationsmechanismen in den echtzeitfähigen Assets?

Netzwerkkonfiguration In einem durchgängig vernetzten CPS ist im Betrieb mit kontinuierlichen Anpassungen und Änderungen zu rechnen. Daher muss die dazugehörige Kommunikationsinfrastruktur dynamisch zur Laufzeit konfiguriert werden.

Wie erfolgt die dynamische Konfiguration der Echtzeitverbindungen in immer komplexeren und sich stetig ändernden Netzwerken zur Laufzeit?

Basierend auf den existierenden Lösungsansätzen wird dabei eine ganzheitliche Lösung mit einer einheitlichen Kommunikationsinfrastruktur für das gesamte Produktionssystem angestrebt. Diese Lösung ermöglicht insbesondere die Integration von echtzeitfähigen Assets und die dynamische Konfiguration der erforderlichen Echtzeitverbindungen zur Laufzeit. Dadurch leistet diese Arbeit einen wesentlichen Beitrag zu einer CPS-basierten Automation. Darüber hinaus bildet die neue Kommunikationsinfrastruktur die Grundlage für die Wandelbarkeit von echtzeitfähigen Assets und zukünftigen Produktionssystemen allgemein.

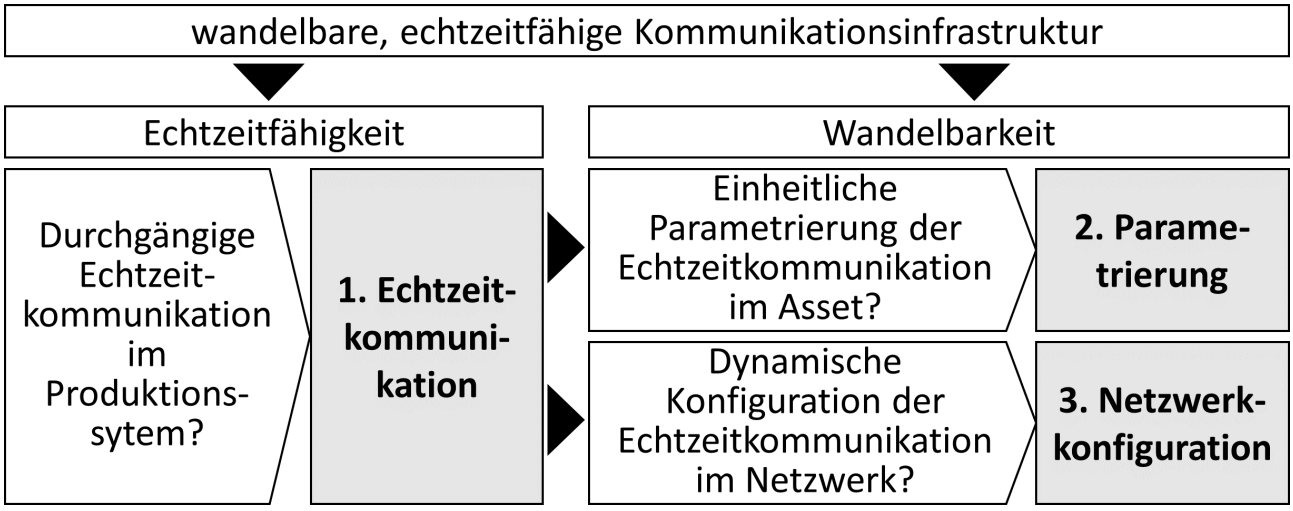


Abbildung 1.3: Die drei wesentlichen Aspekte einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur basierend auf den drei Forschungsfragen.

1.3 Struktur der Arbeit

Basierend auf der vorgestellten Zielsetzung (Kapitel 1) werden zunächst die erforderlichen Begriffe definiert (Kapitel 2). Anschließend werden die Anforderungen an eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur abgeleitet (Kapitel 3). Darauf aufbauend wird der Stand der Technik mit relevanten Lösungsansätzen detailliert vorgestellt und basierend auf den identifizierten Anforderungen bewertet (Kapitel 4). Die darauffolgenden Kapitel 5, 6, 7 repräsentieren den Hauptteil der vorliegenden Arbeit und beinhalten den eigenen Lösungsansatz für eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur. In Kapitel 8 folgt die Bewertung und Validierung des vorgestellten Lösungsansatzes basierend auf einer prototypischen Implementierung. Schließlich werden die Neuerungen gegenüber dem Stand der Technik zusammengefasst und zukünftige Forschungsschwerpunkte abgeleitet (Kapitel 9).

1.4 Veröffentlichungen

In der vorliegenden Arbeit wird eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur für zukünftige Produktionssysteme präsentiert. Dabei wurden verschiedene Teilkonzepte der vorgestellten Lösung bereits publiziert (Prinz et al. 2019b; Prinz et al. 2019a; Prinz et al. 2019c; Prinz et al. 2018a; Prinz et al. 2018b):

A Novel I4.0-enabled Engineering Method and its Evaluation, F. Prinz, M. Schoeffler, A. Lechler, A. Verl, *The International Journal of Advanced Manufacturing Technology*, S. 1-19, 2018

Dynamic Real-time Orchestration of I4.0 Components based on Time-Sensitive Networking, F. Prinz, M. Schoeffler, A. Lechler, A. Verl, *Procedia CIRP*, Jg. 72, S. 910–915, 2018

End-to-end Redundancy between Real-time I4.0 Components based on Time-Sensitive Networking, F. Prinz, M. Schoeffler, A. Lechler, A. Verl, *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, S. 1083-1086, 2018

Configuration of Application Layer Protocols within Real-time I4.0 Components, F. Prinz, M. Schoeffler, A. Eckhardt, A. Lechler, A. Verl, *2019 17th IEEE International Conference on Industrial Informatics (INDIN)*, S. 971-976, 2019

Virtual Network Topologies for Real-time I4.0 Components based on Time-Sensitive Networking, F. Prinz, M. Schoeffler, A. Lechler, A. Verl, *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, S.1388-1391, 2019

Kapitel 2

Begriffsdefinitionen

Zum Verständnis der vorliegenden Arbeit sind folgende Begriffe relevant.

Produktionssystem

Im industriellen Umfeld beschreibt der Begriff Produktion den "Prozess der zielgerichteten Kombination von Produktionsfaktoren (Input) und deren Transformation in Produkte (Erzeugnisse, Output)" (Alisch et al. 2013). Dementsprechend bezeichnet ein Produktionssystem ein System, das aus verschiedenen Inputs in einem Produktionsprozess einen Output erzeugt (Corsten et al. 2012). Da sich ein Produktionssystem auf verschiedene Produktionsebenen beziehen kann, umfasst der Begriff in dieser Arbeit alle produzierenden Assets innerhalb einer Fabrik. Ein Asset bezeichnet dabei einen physikalischen oder virtuellen Gegenstand, der "einen Wert für eine Organisation hat und der auf Grund dessen individuell verwaltet wird" (BMWi 2019).

Wandelbarkeit

Der Begriff der Wandelbarkeit basiert auf dem Begriff der Wandlungsfähigkeit. Dabei beschreibt Wandlungsfähigkeit allgemein die Fähigkeit eines gesamten Produktionssystems reaktive Anpassungen und antizipative Eingriffe vornehmen zu können (Westkämper et al. 2000; Wiendahl et al. 2007). Im Gegensatz zur Umschaltmöglichkeit und Rekonfigurierbarkeit bezieht sich die

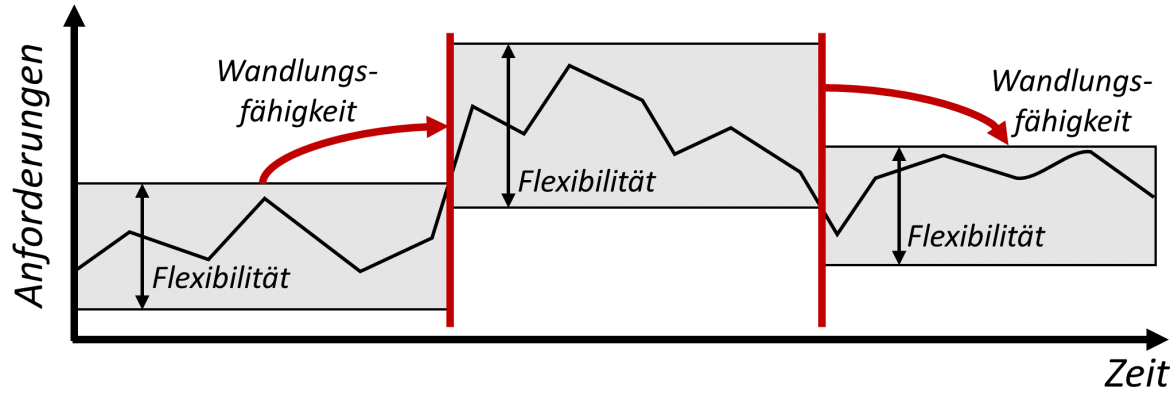


Abbildung 2.1: Abgrenzung von Wandlungsfähigkeit gegenüber Flexibilität (Zäh et al. 2005).

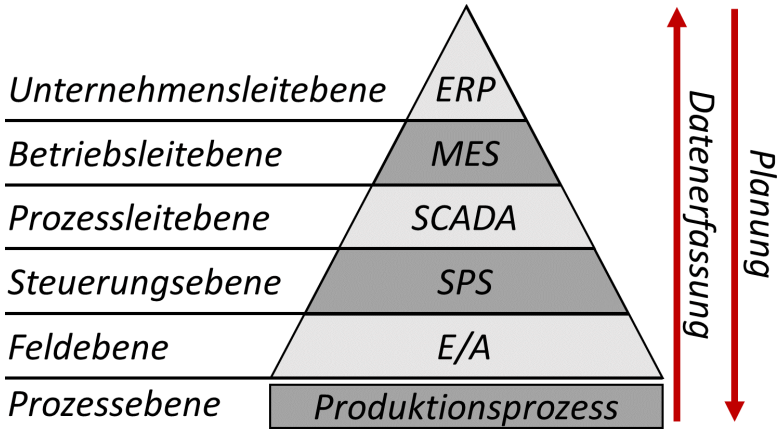


Abbildung 2.2: Die klassische Automatisierungspyramide mit 6 Ebenen (Siepmann et al. 2016).

Wandlungsfähigkeit dabei auf das gesamte Produktionssystem innerhalb einer Fabrik (Wiendahl et al. 2007). Außerdem unterscheidet sich die Wandlungsfähigkeit von der Flexibilität (Bild 2.1). Flexibilität bezeichnet die schnelle Anpassungsfähigkeit in einem bestimmten, vorab definierten Korridor (Abele et al. 2006). Im Gegensatz dazu ermöglicht Wandlungsfähigkeit auch das Verschieben des Korridors und damit verbundene unvorhergesehene Anpassungen (Zäh et al. 2005). Darüber hinaus wird der Begriff Wandlungsfähigkeit gegenüber der Wandelbarkeit abgegrenzt. Wandelbarkeit bezieht sich ausschließlich auf die technischen Aspekte im Produktionssystem, während Wandlungsfähigkeit zusätzlich die sozialen Aspekte mit einbezieht (Westkämper et al. 2000).

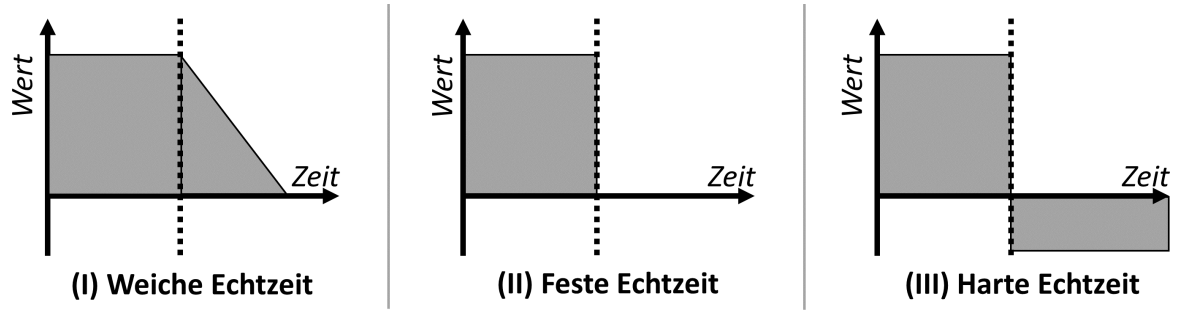


Abbildung 2.3: Drei verschiedenen Arten von Echtzeit, die sich beim Wert einer Information in Abhängigkeit von der Zeit unterscheiden (Wörn 2006).

Automatisierungspyramide

Die klassische Automatisierungspyramide ist ein hierarchisches Modell zur Unterteilung der Produktion in verschiedene Ebenen. In dieser Arbeit wird die Definition von Siepmann angewendet, die zwischen 6 verschiedenen Ebenen unterscheidet (Siepmann et al. 2016). Basierend auf der Prozessebene werden die Feld-, Steuerungs-, Prozessleit-, Betriebsleit- und Unternehmensleitebene definiert (Bild 2.2). Ähnliche Definitionen finden sich in der IEC-Norm 62264 und in der Literatur von Heinrich (Norm IEC 62264-3; Heinrich et al. 2015).

Echtzeit

In der DIN-Norm 44300 (DIN ISO/IEC 2382) wurde der Begriff Echtzeit wie folgt definiert (Norm DIN 44300; Norm ISO/IEC 2382): “Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen”. Darüber hinaus werden drei Arten von Echtzeit unterschieden (Wörn 2006). Diese Arten unterscheiden sich im Wert einer Information in Abhängigkeit der Zeit (Bild 2.3). Auf Grund der harten Echtzeitanforderungen von Assets auf der Feldebene, wird der Begriff Echtzeit in dieser Arbeit synonym zur harten Echtzeit verwendet wird.

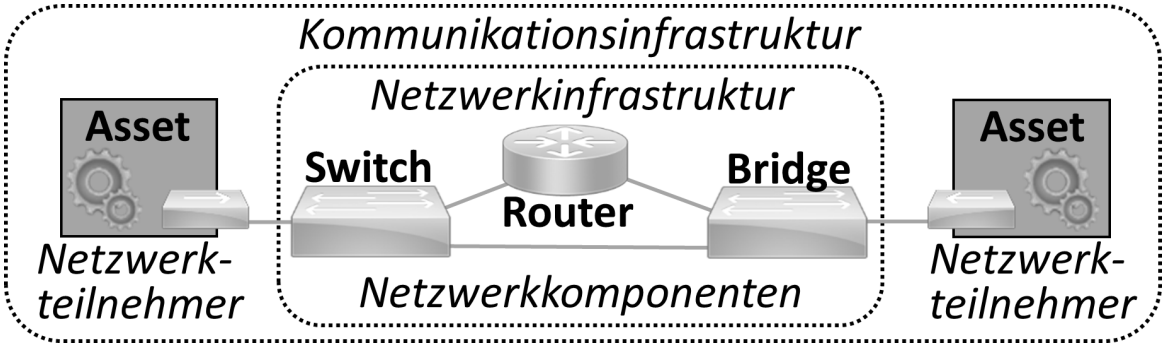


Abbildung 2.4: Die verwendeten Netzwerkbegriffe in dieser Arbeit.

Netzwerkterminologie

Eine (physikalische) Netzwerkinfrastruktur besteht aus Netzwerkkomponenten, insbesondere Switches bzw. Bridges (Norm IEEE 802.1D), die untereinander verbunden sind. Die daran angeschlossenen Assets werden als Netzwerkteilnehmer bezeichnet. Die erweiterte Netzwerkinfrastruktur mit den Netzwerkteilnehmern wird in dieser Arbeit auch als Kommunikationsinfrastruktur bezeichnet (Bild 2.4). Darüber hinaus bildet sich aus der Kombination der Netzwerkkomponenten und Netzwerkteilnehmer die sogenannte Netzwerktopologie (z.B. Linie-, Ring-, Stern- oder Mesh-Topologie), d.h. eine Anordnung der Assets im Netzwerk (Alisch et al. 2013).

Netzwerkkonfiguration

Als Netzwerkkonfiguration wird der Vorgang zur Konfiguration einer Netzwerkinfrastruktur bzw. Kommunikationsinfrastruktur bezeichnet. Dabei werden sowohl die Netzwerkkomponenten (z.B. Switches) als auch die Netzwerkteilnehmer (d.h. Assets) konfiguriert. Die Netzwerkkonfiguration resultiert in einer formalen Beschreibung, der sogenannten Netzwerkkonfigurationsdatei.

I4.0 Komponente

Das Konzept der I4.0 Komponente wird von der “Plattform Industrie 4.0” spezifiziert und standardisiert (BMWi 2020). Eine I4.0 Komponente beschreibt ein Asset mit einer Verwaltungsschale (VWS), d.h. einer virtuellen digitalen Repräsentanz des Assets (BMWi 2019). Diese

Verwaltungsschale beinhaltet insbesondere eine Selbstbeschreibung in Form von Teilmodellen. In dieser Arbeit wird hauptsächlich der Begriff Verwaltungsschale verwendet. Die Begriffe “Digitaler Zwilling” oder “Digitaler Schatten” werden häufig synonym zur Verwaltungsschale verwendet und beschreiben ähnliche Konzepte (Wagner et al. 2017).

Engineering

Der Begriff Engineering ist allgemein als die Anwendung von Wissen zum Design bzw. zur Konstruktion von Strukturen, Prozessen, Maschinen oder ganzer Systeme definiert (Collins 2020). Dieser Begriff wird in verschiedenen Domänen (u.a. Maschinenbau, Elektrotechnik, Chemie) und in einem unterschiedlichen Kontext verwendet (Collins 2020). In dieser Arbeit bezeichnet der Begriff Engineering den Vorgang zur Erstellung eines Arbeitsablaufs im Produktionssystem. Der resultierende Arbeitsablauf spezifiziert die zeitliche Abfolge des Zusammenwirkens von Assets zur Realisierung eines gemeinsamen (Teil-)Produkts (Norm DIN EN ISO 6385). Eine Engineering-Methode ist folglich eine Methode zur Erstellung von Arbeitsabläufen.

Kapitel 3

Anforderungen

Ziel dieser Arbeit ist die Entwicklung einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur für zukünftige Produktionssysteme. Basierend auf den drei Forschungsfragen unterteilt sich die Entwicklung in die wesentlichen Aspekte: Echtzeitkommunikation, Parametrierung und Netzwerkkonfiguration. Im Folgenden werden die einzelnen Aspekte im Detail betrachtet und die daraus resultierenden Anforderungen an eine Lösung abgeleitet.

3.1 Echtzeitkommunikation

Die Etablierung der Echtzeitkommunikation ist der erste wesentliche Aspekte für eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur. Dabei müssen neben der Netzwerkinfrastruktur die echtzeitfähigen Assets um die notwendigen Kommunikationsmechanismen erweitert werden. Diese Mechanismen gewährleisten eine deterministische Kommunikation mit geringer Latenz zwischen echtzeitfähigen Assets bzw. den jeweiligen Echtzeitanwendungen. Ausgangspunkt der Entwicklung sind die heutigen Anforderungen von echtzeitfähigen Assets an die Kommunikation, die auch von einer zukünftigen Lösung erfüllt werden müssen. Mit dem Fokus auf der Wandelbarkeit von zukünftigen Produktionssystemen ergeben sich weitere Anforderungen. Die einzelnen Anforderungen an die Echtzeitkommunikation werden im Folgenden detailliert erläutert (Bild 3.1).

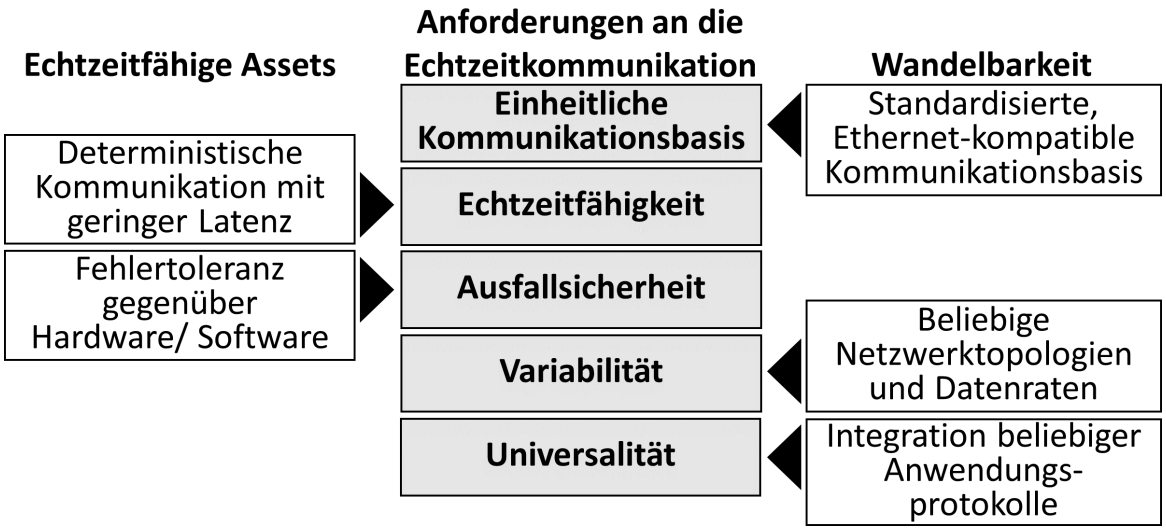


Abbildung 3.1: Die Anforderungen an die Echtzeitkommunikation in einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur.

Einheitliche Kommunikationsbasis Im Kontext der CPS-basierten Automation umfasst eine zukünftige Kommunikationsinfrastruktur das gesamte Produktionssystem innerhalb einer Fabrik. Die damit verbundene Netzwerkkonvergenz erfordert eine einheitliche und standardisierte Kommunikationstechnologie als Basis. Eine solche Technologie ermöglicht insbesondere die Integration von Netzwerkkomponenten und Assets beliebiger Hersteller. Darüber hinaus hat sich heutzutage Ethernet (IEEE 802.3(IEEE 2020b)) als Basis für die drahtgebundene Kommunikation in dem IT-Netzwerk der Produktionssysteme bewährt und etabliert. Daher muss auch eine zukünftige Kommunikationstechnologie Ethernet-kompatibel sein, d.h. die Übertragung von standardkonformen Ethernet Frames unterstützen.

Echtzeitfähigkeit Ein wesentliches Merkmal der CPS-basierten Automation ist die durchgehende Echtzeitfähigkeit der gesamten Kommunikationsinfrastruktur. Die resultierenden Echtzeitverbindungen ermöglichen eine deterministische Kommunikation mit geringer Latenz zwischen verschiedenen Assets. Der Maßstab für die Echtzeitkommunikation sind dabei die echtzeitfähigen Assets auf der Feldebene (z.B. Motion Control), die über die höchsten Anforderungen bezüglich Jitter und Latenz verfügen (Bild 3.2). Folglich muss die verwendete, einheitliche Kommunikationstechnologie auch harte Echtzeitanforderungen unterstützen und eine entsprechende Echtzeitkommunikation gewährleisten.

Ausfallsicherheit Trotz regelmäßiger Wartung können sowohl die Netzwerkinfrastruktur als auch einzelne Assets im Produktionssystem von Fehlern betroffen sein. Dabei können sowohl

Echtzeit-klasse	Anwendung	Anforderungen	
		Latenz	Jitter
1	Steuerung-zu-Steuerung, Visualisierung	10-100ms	-
2	Steuerung-zu-dezentraler-Peripherie	1-10ms	< 1ms
3	Synchronisierte Bewegungs-abläufe (Motion Control)	< 1ms	< 1µs

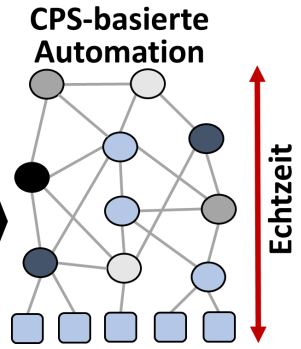


Abbildung 3.2: Klassifizierung der Echtzeitanforderungen (Jasperneite 2005).

Hardwarefehler (z.B. Kabelbruch) als auch Fehler in der Software (z.B. Systemabsturz) auftreten. Echtzeitfähige Assets und insbesondere sicherheitskritische Systeme erfordern daher eine fehlertolerante, ausfallsichere Echtzeitkommunikation. Auf Grund der harten Echtzeitanforderungen muss dabei eine redundante Datenübertragung ohne Frameverlust im Falle eines Fehlers gewährleistet werden.

Variabilität Bei der Vernetzung von mehr als zwei echtzeitfähigen Assets im OT-Netzwerk haben sich heutzutage Linien- oder Ring-Topologien etabliert. Im Gegensatz dazu erfolgt die Vernetzung von gewöhnlichen Assets im IT-Netzwerk häufig in Stern- bzw. Mesh-Topologien und mit höheren Datenraten (>100 MBit/s). Auf Grund der Konvergenz beider Netzwerke muss die zukünftige Kommunikationsinfrastruktur beliebige physikalische Netzwerktopologien und eine skalierbare Datenrate unterstützen.

Universalität Basierend auf der einheitlichen Kommunikationsbasis existiert ein breites Spektrum an Anwendungsprotokollen für verschiedene Szenarien. Auch in der Echtzeitkommunikation haben sich mehrere Anwendungsprotokolle etabliert, wie z.B. diverse Industrial Ethernet Protokolle, DDS oder OPC UA. Die etablierte Echtzeitkommunikation muss daher generisch sein und die Integration von beliebigen echtzeitfähigen Anwendungsprotokollen ermöglichen.

3.2 Parametrierung

Die einheitliche Parametrierung der Echtzeitkommunikation in den Assets ist der zweite wesentliche Aspekt einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur. Die Para-

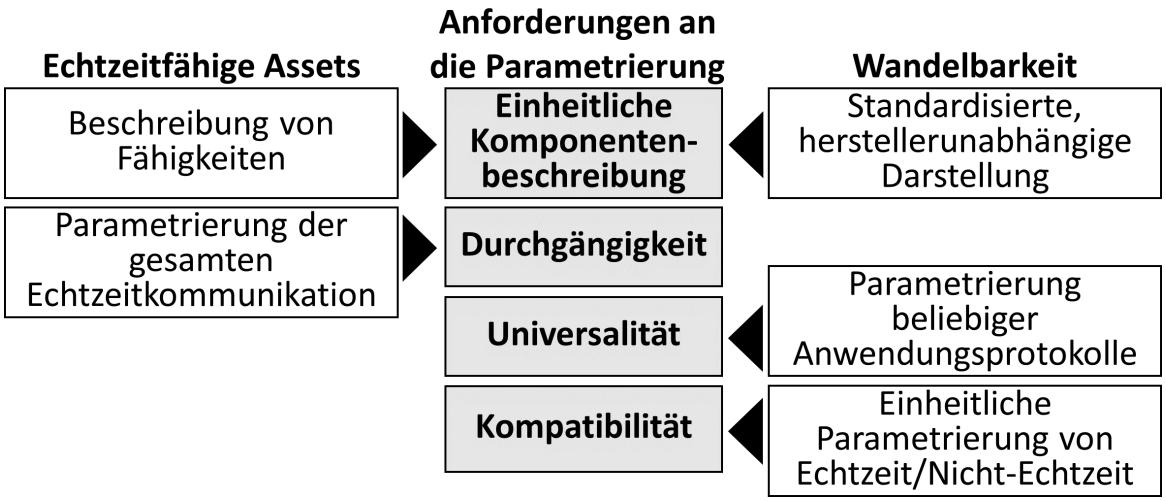


Abbildung 3.3: Die Anforderungen an die Parametrierung der Echtzeitkommunikationen in den Assets.

metrierung ermöglicht die Anpassung an das jeweilige Asset, die einheitliche Konfiguration des Gesamtsystems sowie dynamische Änderungen zur Laufzeit. Dabei ergeben sich folgende Anforderungen an eine Lösung (Bild 3.3).

Einheitliche Komponentenbeschreibung Zur Parametrierung der echtzeitfähigen Assets ist eine herstellerunabhängige und standardisierte Komponentenbeschreibung erforderlich. Diese Beschreibung ermöglicht die Darstellung von verfügbaren Konfigurationsparametern und Statusinformationen für die Echtzeitkommunikation. Darüber hinaus werden neben den spezifischen Parametern für die Echtzeitkommunikation auch die individuellen Fähigkeiten eines Assets beschrieben. Diese Fähigkeiten basieren im Wesentlichen auf den (Echtzeit-)Anwendungen der Assets.

Durchgängigkeit Für die Echtzeitkommunikation ist eine durchgängige Parametrierung der Kommunikationsmechanismen in den echtzeitfähigen Assets erforderlich. Dabei werden sowohl Konfigurationsparameter als auch Statusinformationen für die Kommunikationsbasis, das Anwendungsprotokoll und die eigentliche Echtzeitanwendung spezifiziert. Für eine einheitliche Parametrierung im Produktionssystem ist eine konsistente Darstellung aller Parameter erforderlich.

Universalität Basierend auf der einheitlichen Kommunikationsbasis umfasst die Parametrierung auch die darüberliegenden Anwendungsprotokolle. Dabei existiert eine Vielzahl von

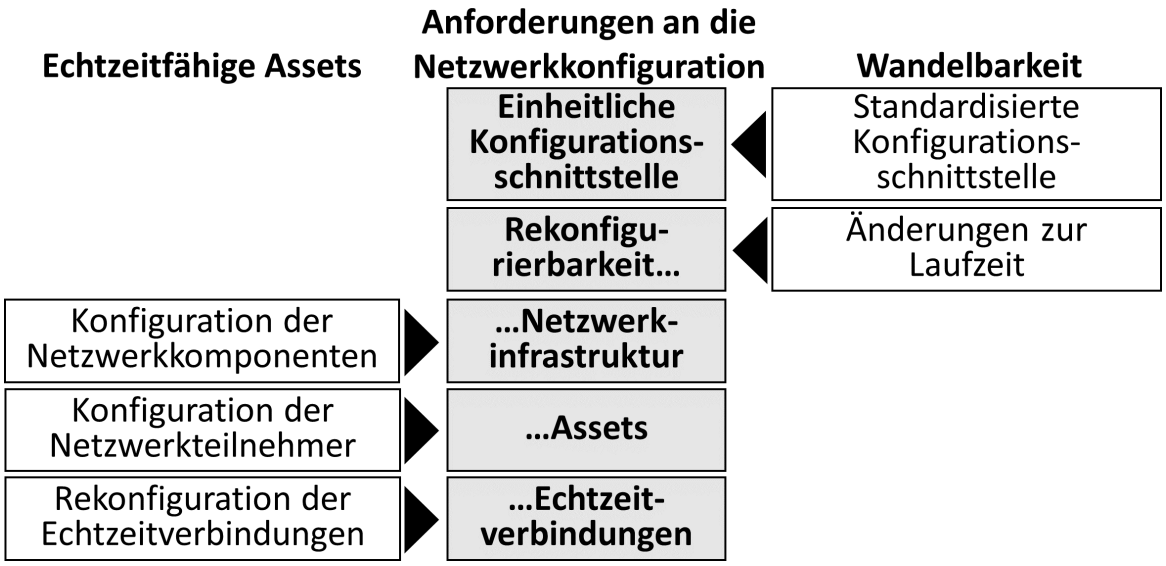


Abbildung 3.4: Die Anforderungen an die Konfiguration der Echtzeitkommunikation im Netzwerk.

(echtzeitfähigen) Anwendungsprotokollen für verschiedene Szenarien im Produktionssystem. Daher ist eine konsistente Darstellung der Konfigurationsparameter und Statusinformationen unabhängig von einem konkreten Anwendungsprotokoll erforderlich.

Kompatibilität Die angestrebte Netzwerkkonvergenz führt zu einer Koexistenz von echtzeitfähigen Assets und gewöhnlichen Assets ohne Echtzeitanforderungen in der Kommunikationsinfrastruktur. Folglich muss eine einheitliche Parametrierung von echtzeitfähigen und gewöhnlichen Assets ermöglicht werden. Diese Parametrierung umfasst sowohl die (Echtzeit-spezifischen) Kommunikationsfähigkeiten als auch die individuellen Fähigkeiten der Assets.

3.3 Netzwerkkonfiguration

Der dritte wesentliche Aspekt der wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur ist die dynamische Netzwerkkonfiguration. Dabei werden insbesondere die notwendigen Echtzeitverbindungen zwischen verschiedenen echtzeitfähigen Assets zur Laufzeit etabliert. Mit dem Fokus auf der Wandelbarkeit von zukünftigen Produktionssystemen ergeben sich folgende Anforderungen an die Netzwerkkonfiguration (Bild 3.4).

Einheitliche Konfigurationsschnittstelle Grundlage für die Netzwerkkonfiguration bildet eine einheitliche Konfigurationsschnittstelle für die Echtzeitkommunikation. Diese Schnittstelle kann von echtzeitfähigen Assets, einer grafischen Weboberfläche oder einem Engineering-Tool genutzt werden um eine Echtzeitverbindung zu initiieren.

Rekonfigurierbarkeit Netzwerkinfrastruktur Die Netzwerkinfrastruktur bildet die Grundlage für die Echtzeitkommunikation im Produktionssystem. Mit dem Fokus auf der Wandelbarkeit ist dabei eine dynamische Konfiguration der einzelnen Netzwerkkomponenten (z.B. Switches) erforderlich. Dadurch wird eine Anpassung der Kommunikationsmechanismen an die aktuellen Anforderungen im Produktionssystem ermöglicht.

Rekonfigurierbarkeit Assets Neben der Netzwerkinfrastruktur verfügen auch die Netzwerkteilnehmer (d.h. echtzeitfähigen Assets) über spezifische Mechanismen für die Echtzeitkommunikation. Mit dem Fokus auf der Wandelbarkeit ist daher eine dynamische Konfiguration der echtzeitfähigen Assets erforderlich. Dabei wird insbesondere die Kommunikationsbasis einheitlich zur Netzwerkinfrastruktur konfiguriert.

Rekonfigurierbarkeit Echtzeitverbindungen Die dynamische Konfiguration der Netzwerkkomponenten und der Netzwerkteilnehmer ermöglicht die Konfiguration der erforderlichen Echtzeitverbindungen zur Laufzeit. Dabei werden zum einen neue Echtzeitverbindungen zur Laufzeit etabliert. Zum anderen werden nicht mehr benötigte Echtzeitverbindungen beendet und die verfügbare Bandbreite im Netzwerk möglichst effizient genutzt. Darüber hinaus ermöglicht die dynamische Konfiguration der Echtzeitverbindungen eine kontinuierliche Anpassung an die aktuelle Netzwerktopologie und die Anforderungen der angeschlossenen echtzeitfähigen Assets.

3.4 Zusammenfassung

Bild 3.5 fasst die Anforderungen an eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur zusammen. Die identifizierten Anforderungen bilden die Grundlage für die Bewertung von existierenden Lösungsansätzen im folgenden Kapitel.

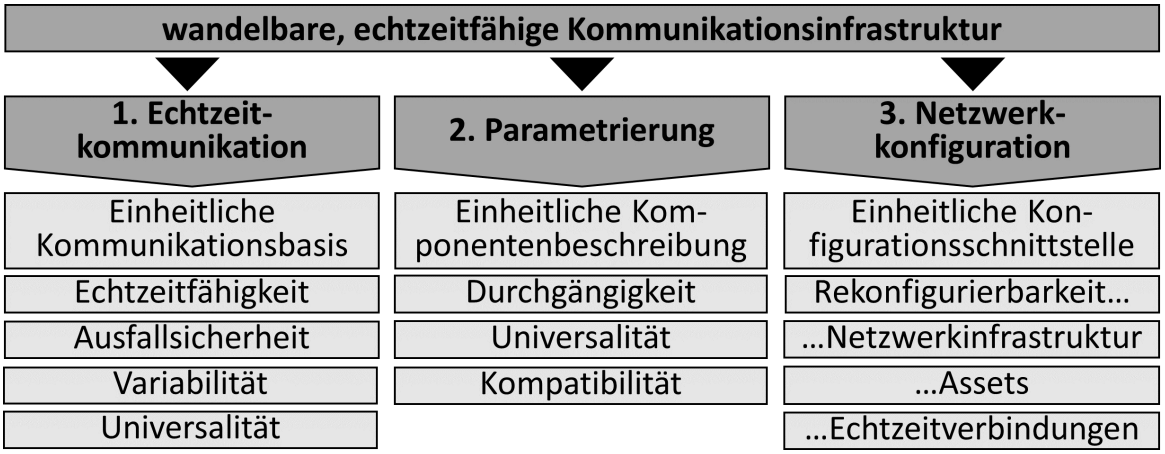


Abbildung 3.5: Die Anforderungen an eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur.

Kapitel 4

Stand der Technik

Im vorherigen Kapitel wurden die Anforderungen an eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur identifiziert. Darauf aufbauend wird im Folgenden der Stand der Technik betrachtet. Dabei werden die existierenden Lösungsansätze mit Bezug zur Echtzeitkommunikation in das OSI-Referenzmodell eingeordnet und bewertet.

4.1 OSI-Referenzmodell

Zur Strukturierung der (Echtzeit-)Kommunikation und existierender Technologien hat sich das OSI-Referenzmodell (Open Systems Interconnection) etabliert. Dieses Modell definiert insgesamt 7 Kommunikationsschichten (Bild 4.1) (Zimmermann 1980):

Bitübertragungsschicht Die unterste Schicht spezifiziert die Übertragung von Bits über ein Medium, wie z.B. Kupferkabel, Lichtwellenleiter oder Funk. Dabei wird jedes Bit in ein elektrisches Signal, optisches Signal oder elektromagnetische Wellen umgewandelt.

Sicherungsschicht Die zweite Schicht definiert den Zugriff auf das Übertragungsmedium und gewährleistet eine möglichst zuverlässige Datenübertragung. Dazu werden die Daten segmentiert und in sogenannte Frames verpackt. Ein Frame besteht aus einem Header, den eigentlichen Daten (Payload) und einer Prüfsumme zur Fehlererkennung.

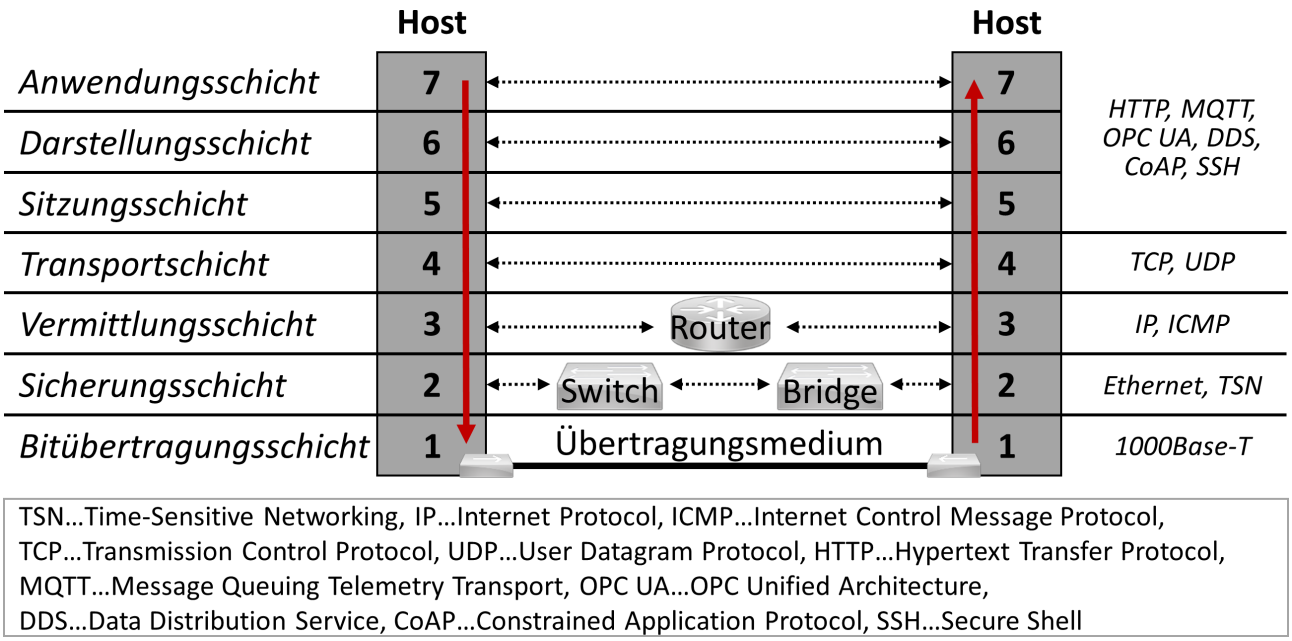


Abbildung 4.1: Das OSI-Referenzmodell zur Strukturierung der (Echtzeit-)Kommunikation (Zimmermann 1980).

Vermittlungsschicht Die dritte Schicht spezifiziert die Übertragung von Datenpaketen zwischen dem Sender und Empfänger. Die Vermittlung der Datenpakete zwischen verschiedenen Subnetzen wird auch als Routing bezeichnet und basiert auf einer eindeutigen Adressierung im Netzwerk.

Transportschicht Die vierte Schicht realisiert die Ende-zu-Ende Übertragung von Datensegmenten. Dabei wird zwischen einer verbindungsorientierten und verbindungslosen Übertragung unterschieden. Außerdem werden die Datensegmente einer Anwendung im Sender bzw. Empfänger zugeordnet.

Sitzungsschicht Die fünfte Schicht verwaltet eine logische Verbindung (Sitzung) zwischen dem Sender und Empfänger. Dazu zählt insbesondere das Starten und Beenden einer Verbindung sowie die Synchronisation.

Darstellungsschicht Die sechste Schicht definiert die Codierung der übertragenen Daten. Dabei werden die anwendungsspezifischen Daten in ein für das Netzwerk geeignetes Format umgewandelt.

Anwendungsschicht Die oberste Schicht bildet die Schnittstelle zu den eigentlichen Anwendungen. Dabei werden die anwendungsspezifischen Daten in Nachrichten verpackt (Sender) und eingehende Nachrichten verarbeitet (Empfänger).

Heutzutage spezifizieren Anwendungsprotokolle (z.B. HTTP, MQTT) üblicherweise die obersten drei OSI-Schichten. Daher werden diese Schichten im Folgenden zusammengefasst und nicht weiter unterschieden. Darüber hinaus werden die mittleren OSI-Schichten 3-4 bei der Echtzeitkommunikation aus Performancegründen häufig ausgelassen. Daher werden im Folgenden die existierenden Lösungsansätze für die Kommunikationsbasis (OSI-Schicht 1-2) und die verschiedenen Anwendungsprotokolle (OSI-Schicht 5-7) betrachtet.

4.2 Ethernet

Im IT-Netzwerk der Produktionssysteme hat sich heutzutage Ethernet als Kommunikationsbasis auf den OSI-Schichten 1-2 etabliert. Ethernet wird von der weltweit anerkannten IEEE ("Institute of Electrical and Electronics Engineers") standardisiert und in der Arbeitsgruppe IEEE 802.3 weiterentwickelt (IEEE 2020b). Die dazugehörigen Standards spezifizieren eine bestmögliche Datenübertragung in einer Switch-basierten Netzwerkinfrastruktur.

Echtzeitfähigkeit Der Standard IEEE 802.1Q ermöglicht das priorisierte Weiterleiten von Ethernet Frames im Netzwerk (Norm IEEE 802.1Q). Dazu definiert der Standard insgesamt 8 Prioritäten zwischen 0 (Best-Effort) und 7 (höchste Priorität). Die Priorität eines Frames ist im Ethernet Header codiert, d.h. im PCP-Feld (Priority Code Point) des VLAN-Tags (Virtual Local Area Network) spezifiziert (Norm IEEE 802.1Q). Beim Weiterleiten werden zuerst die Frames mit der höchsten Priorität in einer Netzwerkkomponente (z.B. Switch) berücksichtigt. Allerdings kann die Übertragung von anderen Frames nicht unterbrochen werden. Dadurch werden zeitkritische Frames blockiert und die harten Echtzeitanforderungen verletzt. Daher eignet sich Standard-Ethernet nicht für eine deterministische Übertragung von echtzeitkritischen Frames mit geringer Latenz.

Ausfallsicherheit Für eine ausfallsichere Kommunikation verwenden Ethernet Netzwerke standardmäßig das Rapid Spanning Tree Protocol (RSTP, IEEE 802.1w (Norm IEEE 802.1w)). Dieses Protokoll konfiguriert eindeutige, azyklische Netzwerkpfade basierend auf der aktuellen physikalischen Netzwerkinfrastruktur. Bei einem Fehler werden alternative Netzwerkpfade berechnet und aktualisiert. Dazu benötigt RSTP allerdings eine Rekonfigurationszeit und ist daher für die unterbrechungsfreie Echtzeitkommunikation ungeeignet.

Variabilität Ethernet ermöglicht eine skalierbare Kommunikation mit beliebigen Netzwerktopologien, insbesondere Stern- und Mesh-Topologien. Dabei werden verschiedene Datenraten zwischen 10 MBit/s und bis zu 400 GBit/s unterstützt (IEEE 2020b).

Komponentenbeschreibung und Netzwerkkonfiguration Zur Konfiguration von Ethernet Netzwerken hat sich das Simple Network Management Protocol (SNMP (Case et al. 2002)) bewährt. Dieses Protokoll basiert auf der sogenannten Management Information Base (MIB) zur Beschreibung von Netzwerkkomponenten, wie z.B. Switches. Dabei existieren sowohl standardisierte MIBs als auch individuelle, herstellerspezifische Erweiterungen. Eine MIB spezifiziert jeweils Konfigurationsparameter und Statusinformationen, die über SNMP gesetzt bzw. abgerufen werden.

Vor dem Hintergrund immer komplexerer Ethernet Netzwerke wurde das Network Configuration Protocol (NETCONF (Enns et al. 2011)) standardisiert (Schönwälder et al. 2010). Dieses Protokoll basiert auf einer XML-basierten Beschreibung von Konfigurationsparametern und Statusinformationen, den sogenannten YANG-Modellen (Yet Another Next Generation (Bjorklund 2010)). Sowohl die IEEE als auch die einzelnen Switchhersteller arbeiten aktiv an der Standardisierung von YANG-Modellen (Norm IEEE 802.3.2; Norm IEEE 802.1Qcp). Dabei liegt der Fokus allerdings bisher auf den Netzwerkkomponenten und den dazugehörigen Kommunikationsfähigkeiten.

Darüber hinaus haben sich einige Protokolle als De-facto-Standard in Ethernet Netzwerken etabliert. Dabei sind insbesondere die folgenden Protokolle für die vorliegende Arbeit relevant.

Zeitsynchronisation Zur Zeitsynchronisation in Ethernet Netzwerken haben sich zwei Protokolle etabliert. Das Network Time Protocol (NTP (Mizrahi et al. 2016)) ermöglicht die Synchronisation mit der koordinierten Weltzeit (UTC). Dabei wird die lokale Systemzeit über hierarchische NTP Server mit einer hochpräzisen Atomuhr synchronisiert. Die resultierende Genauigkeit liegt im Millisekundenbereich (<1 s).

Für eine genauere Synchronisation im Nanosekundenbereich (<1 µs) wird das Precision Time Protocol (PTP, IEEE 1588 (Norm IEEE 1588)) verwendet. Dabei definiert ein sogenannter PTP Grandmaster die lokale Netzwerkzeit und sendet zyklische Nachrichten mit den aktuellen Zeitinformationen. Jeder Slave empfängt die PTP Nachrichten, berechnet die Zeitdifferenz zum Master und passt die interne Systemzeit entsprechend an (Bild 4.2). Der PTP Grandmaster wird

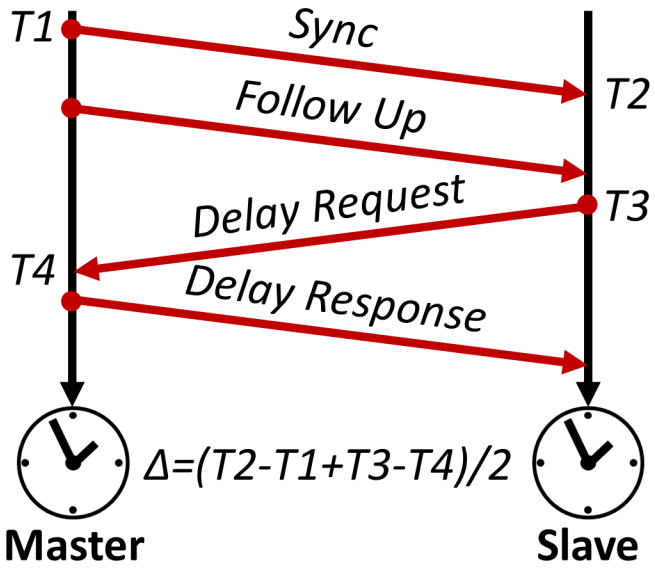


Abbildung 4.2: Zeitsynchronisation basierend auf den zyklischen PTP Nachrichten (Norm IEEE 1588).

zur Laufzeit durch den Best-Master-Clock Algorithmus bestimmt. Dabei wird eine spezifizierte Priorität, die Genauigkeit der internen Uhr und eine PTP-ID bei der Auswahl berücksichtigt.

PTP bildet die Grundlage für eine präzise Zeitsynchronisation und bietet umfangreiche Konfigurationsmöglichkeiten. Daher existieren sogenannte PTP Profile mit ausgewählten Konfigurationsparametern für spezifische Anwendungsfälle. Für echtzeitkritische Netzwerke wird insbesondere das Profil aus dem Standard IEEE 802.1AS angewendet (Norm IEEE 802.1AS; Stanton 2018).

Netzwerkdiscovery Ethernet Netzwerke basieren auf einer Switch-basierten Netzwerkinfrastruktur mit einer beliebigen Topologie (z.B. Stern- oder Mesh-Topologie). Zur Discovery der aktuellen physikalischen Netzwerktopologie hat sich das Link Layer Discovery Protocol (LLDP (Norm IEEE 802.1AB)) etabliert. Bei diesem Protokoll versendet jede Netzwerkkomponente zyklische LLDP Nachrichten mit ihren lokalen Informationen. Gleichzeitig werden eingehende LLDP Nachrichten von anderen Netzwerkkomponenten ausgewertet und die resultierenden (Port-)Informationen zur Verfügung gestellt.

Zusammengefasst ermöglicht Ethernet eine einheitliche, herstellerunabhängige Kommunikationsbasis und hat sich in den IT-Netzwerken der Produktionssysteme etabliert. Auf Grund der fehlenden Echtzeitfähigkeit ist Standard-Ethernet allerdings nicht für eine zukünftige Kommunikationsinfrastruktur geeignet.

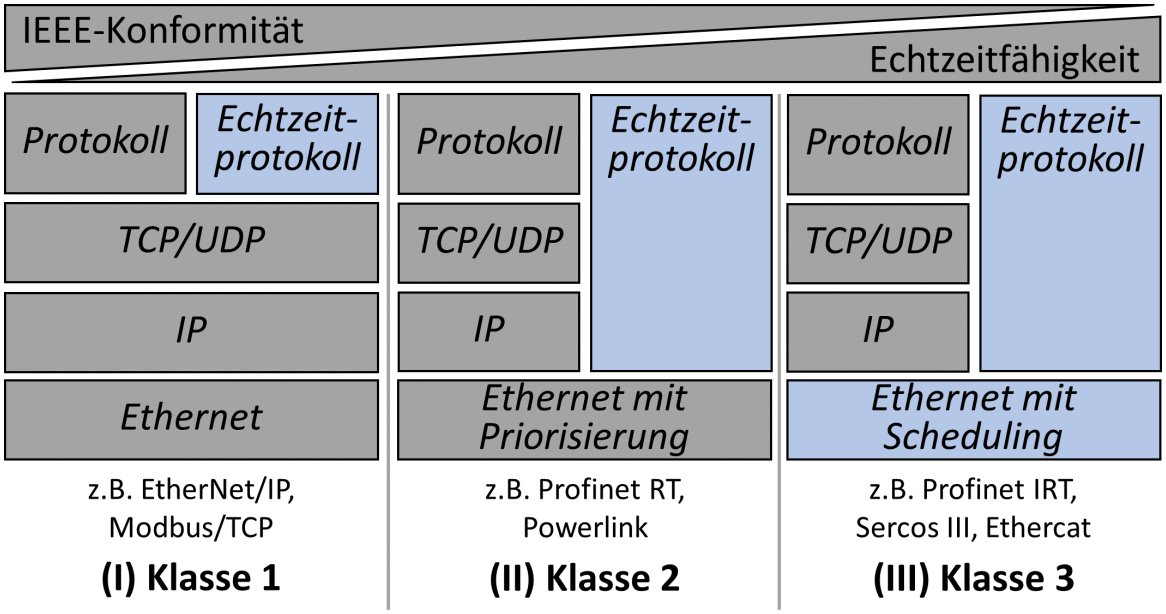


Abbildung 4.3: Klassifizierung von Industrial Ethernet Protokollen (Felser 2005; Jasperneite et al. 2007)

4.3 Industrial Ethernet

Während sich Standard-Ethernet als Kommunikationsbasis im IT-Netzwerk etabliert hat, erfüllt es nicht die harten Echtzeitanforderungen im OT-Netzwerk. Daher wurden verschiedene Ethernet-basierte Lösungen für eine deterministische Übertragung von echtzeitkritischen Frames entwickelt (Danielis et al. 2014). Diese Lösungen werden unter dem Begriff Industrial Ethernet oder Echtzeit-Ethernet zusammengefasst (Norm IEC 61784-2; Sauter et al. 2011). Dabei werden drei verschiedene Klassen von Industrial Ethernet Protokollen unterschieden (Bild 4.3) (Felser 2005; Jasperneite et al. 2007):

Klasse 1 Die erste Klasse basiert auf einem gewöhnlichen TCP/IP Protokollstack mit Standard-Ethernet. Die Echtzeitmechanismen werden ausschließlich in das Anwendungsprotokoll integriert. Daher erreicht diese Klasse Zykluszeiten von ungefähr 100 ms (Jasperneite et al. 2007).

Klasse 2 Die zweite Klasse basiert unmittelbar auf Standard-Ethernet und umgeht damit aus Performancegründen die OSI-Schichten 3-4. Darüber hinaus wird die Priorisierung von Ethernet Frames entsprechend dem Standard IEEE 802.1Q verwendet (Norm IEEE 802.1Q). Dadurch werden Zykluszeiten von ungefähr 10 ms für die echtzeitkritischen Frames gewährleistet (Jasperneite et al. 2007).

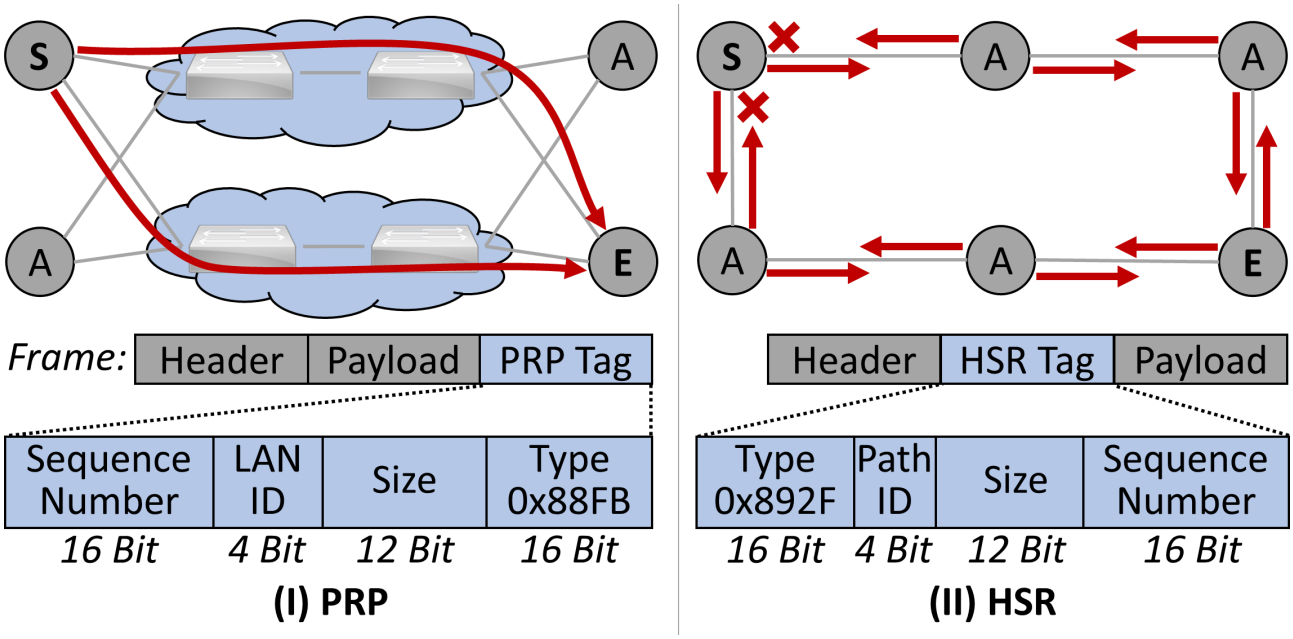


Abbildung 4.4: Redundante Übertragung von Ethernet Frames basierend auf den Standards PRP und HSR (Norm IEC 62439-3).

Klasse 3 Die dritte Klasse modifiziert Standard-Ethernet und realisiert ein zeitliches Scheduling auf der OSI-Schicht 2. Dadurch werden Zykluszeiten unter 1 ms ermöglicht. Auf Grund der proprietären Ethernet-Erweiterungen ist allerdings eine spezifische Hardware notwendig.

Echtzeitfähigkeit In dieser Arbeit liegt der Fokus auf den Industrial Ethernet Protokollen der Klasse 3. Dazu zählen insbesondere die etablierten Protokolle Sercos, EtherCAT und Profinet (Norm IEC 61784-2; Schemm 2004; Rostan et al. 2010; Pigan et al. 2008). Diese Protokolle ermöglichen Zykluszeiten von bis zu 31,25 µs und genügen damit den höchsten Echtzeitanforderungen auf der Feldebene (Danielis et al. 2014). Wesentlicher Nachteil sind allerdings die inkompatiblen Ethernet-Erweiterungen der verschiedenen Protokolle und die spezifische Hardware.

Ausfallsicherheit In (Prytz 2006) werden verschiedene Redundanzimplementierungen für Industrial Ethernet Protokolle beschrieben. Basierend auf diesen teilweise proprietären Lösungen wird die Notwendigkeit einer standardisierten Implementierung hervorgehoben. Dabei spezifiziert der IEC Standard 62439 Stand heute zwei wesentliche Redundanzprotokolle (Norm IEC 62439-3).

Das High-Availability Seamless Redundancy Protokoll (HSR) ermöglicht eine redundante Übertragung von Ethernet Frames ohne Rekonfigurationszeit im Falle eines Fehlers. Dabei basiert das Protokoll auf einer Ring Topologie mit sogenannten Double-Attached Nodes (DANs) (Kirrmann et al. 2011). Ein DAN verfügt über zwei spezielle Netzwerkadapter und sendet einen Frame in beide Richtungen entlang des Rings (Bild 4.4). Zur Identifikation der redundanten Frames wird ein HSR-Tag mit einer Sequenznummer vor dem eigentlichen Payload eingefügt. Wesentlicher Nachteil von HSR ist allerdings die fixe Ringtopologie ohne Switches.

Das Parallel Redundancy Protocol (PRP) ermöglicht ebenfalls eine redundante Frame-Übertragung ohne Rekonfigurationszeit und basiert auf zwei physisch getrennten Netzwerken (Kirrmann et al. 2007). Analog zu HSR verfügt ein Double-Attached Node (DAN) über zwei Netzwerkadapter, die jeweils mit einem Netzwerk verbunden sind. Folglich wird ein Ethernet Frame redundant über zwei vollständig disjunkte Netzwerkpfade übertragen (Bild 4.4). Die Codierung der Redundanzinformationen im Frame erfolgt über einen zusätzlichen Trailer nach dem eigentlichen Payload. Während PRP die Verwendung von Switches unterstützt, ist allerdings eine doppelte physikalische Netzwerkinfrastruktur erforderlich.

Variabilität Die etablierten Industrial Ethernet Protokolle (z.B. Sercos, EtherCAT, Profinet) erfüllen die Anforderungen an die Variabilität und Skalierbarkeit einer zukünftigen Kommunikationsinfrastruktur nur bedingt. Zum einen wird heutzutage fast ausschließlich Fast Ethernet unterstützt, d.h. eine maximale Datenrate von 100 MBit/s (Danielis et al. 2018; IEEE 2020b). Zum anderen ist die Netzwerktopologie und die Teilnehmeranzahl je nach Protokoll beschränkt (Danielis et al. 2014; Schemm 2004).

Komponentenbeschreibung Im Kontext der Industrial Ethernet Protokolle haben sich verschiedene Applikationsprofile zur Beschreibung der echtzeitfähigen Assets etabliert. Insbesondere existieren standardisierte Profile, wie z.B. CiA 402, CIP Motion, PROFIdrive oder SERCOS (Norm IEC 61800-7-1). Diese Profile spezifizieren Funktionen und Parameter für eine gewisse Klasse von Assets und ermöglichen einen herstellerunabhängigen Zugriff. Darauf aufbauend existieren Ansätze um die protokollspezifischen Profile zu vereinheitlichen (Norm IEC 61800-7-1; Lechler 2011). Diese Idee der protokollübergreifenden Parametrierung wird in der vorliegenden Arbeit aufgegriffen und auf eine standardisierte Darstellung übertragen.

Netzwerkkonfiguration Die Konfiguration der Industrial Ethernet Protokolle ist durch einen hohen, manuellen Engineering-Aufwand geprägt (Dürkop et al. 2015). Daher existieren

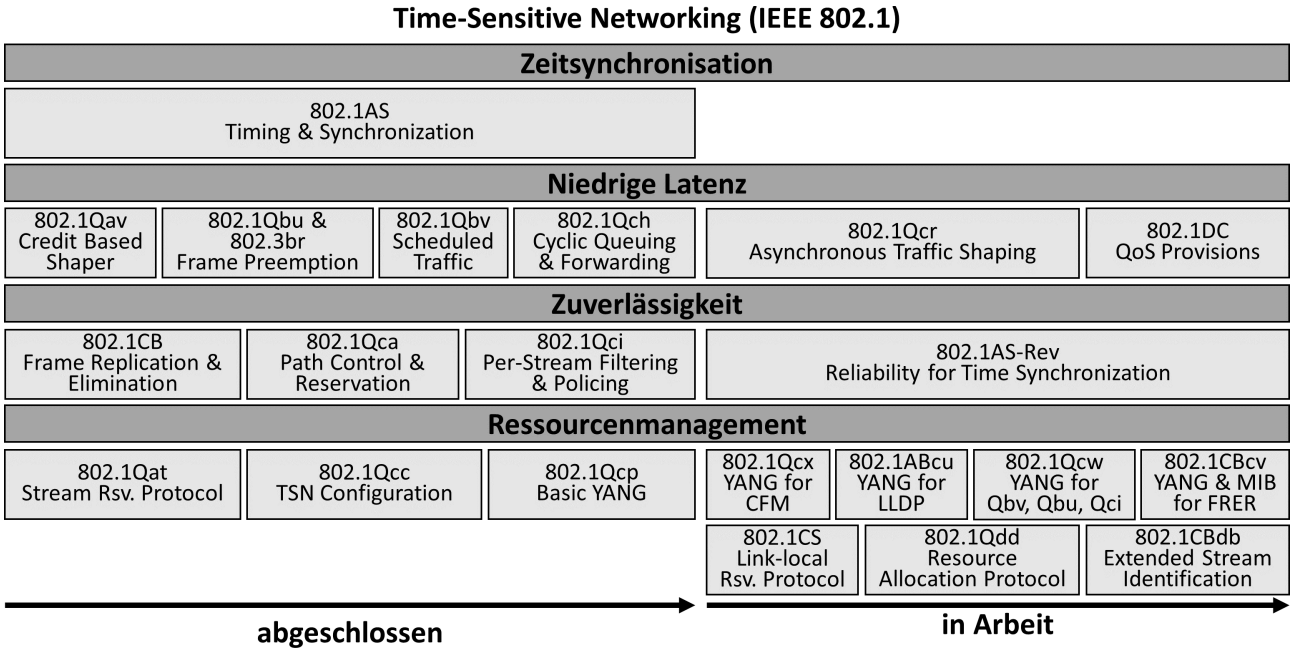


Abbildung 4.5: Fortschritt der TSN Standardisierung (IEEE 2020; Farkas 2018).

verschiedene Ansätze für eine automatisierte Konfiguration einzelner Protokolle, wie z.B. Profinet (Dürkop et al. 2013; Wisniewski 2017) oder Ethernet Powerlink (Veichtlbauer et al. 2017; Reinhart et al. 2010). Darauf aufbauend existieren erste Ansätze für eine automatisierte Konfiguration unabhängig von einem konkreten Protokoll (Dürkop et al. 2015; Imtiaz et al. 2008). Allerdings wird eine dynamische Konfiguration, insbesondere das Hinzufügen und Entfernen von echtzeitfähigen Assets zur Laufzeit, nur bedingt unterstützt (Danielis et al. 2018; Dürkop et al. 2015).

Zusammengefasst erfüllen die verschiedenen Industrial Ethernet Protokolle die höchsten Echtzeitanforderungen und ermöglichen eine durchgängige Echtzeitkommunikation. Wesentlicher Nachteil sind allerdings die Vielzahl existierender Protokolle und die Inkompatibilität der Lösungen.

4.4 Time-Sensitive Networking

Die neue Kommunikationstechnologie Time-Sensitive Networking (TSN, IEEE 802.1 (IEEE 2020)) spezifiziert eine echtzeitfähige und gleichzeitig IEEE-konforme Ethernet-Erweiterung. Dabei ist das Ziel die verschiedenen, existierenden Ethernet-Erweiterungen (Abschnitt 4.3) durch eine einheitliche, standardisierte Lösung zu ersetzen.

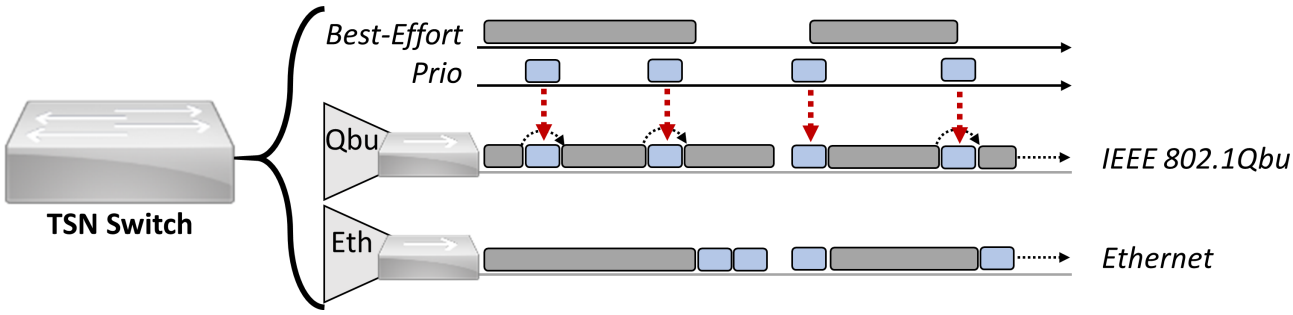


Abbildung 4.6: Frame-Preemption entsprechend dem TSN Standard IEEE 802.1Qbu (Norm IEEE 802.1Qbu).

TSN basiert auf der Technologie Audio-Video Bridging (AVB (Norm IEEE 802.1BA)) und wird inzwischen von der IEEE 802.1 TSN Task Group weiterentwickelt (IEEE 2020). Diese Arbeitsgruppe spezifiziert eine Reihe von Standards, die sich in insgesamt vier Teilaspekte unterteilen lassen (Bild 4.5) (Farkas 2018):

Zeitsynchronisation Der erste Teilaspekt gewährleistet ein einheitliches Zeitverständnis im Netzwerk und bildet die Grundlage für eine TSN-basierte Kommunikation.

Niedrige Latenz Der zweite Teilaspekt spezifiziert die erforderlichen Kommunikationsmechanismen zum Weiterleiten von (echtzeitkritischen) TSN Frames im Netzwerk.

Zuverlässigkeit Der dritte Teilaspekt gewährleistet eine zuverlässige und insbesondere ausfallsichere Übertragung von (echtzeitkritischen) TSN Frames im Netzwerk.

Ressourcenmanagement Der vierte Teilaspekt beschreibt die Konfiguration der TSN-basierten Kommunikation und der dazugehörigen Echtzeitverbindungen.

Ausgehend von den vier Teilaspekten werden die für die vorliegende Arbeit wesentlichen TSN Standards im Folgenden erläutert (IEEE 2020; Finn 2018).

IEEE 802.1AS-Rev Die Zeitsynchronisation in TSN Netzwerken basiert auf dem etablierten Precision Time Protocol. Dazu definiert der Standard IEEE 802.1AS-Rev ein TSN-spezifisches PTP Profil (Norm IEEE 802.1AS-Rev). Dieses Profil erweitert den existierenden Standard IEEE 802.1AS um eine zuverlässige Zeitsynchronisation (Norm IEEE 802.1AS). Dabei werden insbesondere redundante Grandmaster im Netzwerk etabliert und der Ausfall eines Grandmasters kompensiert.

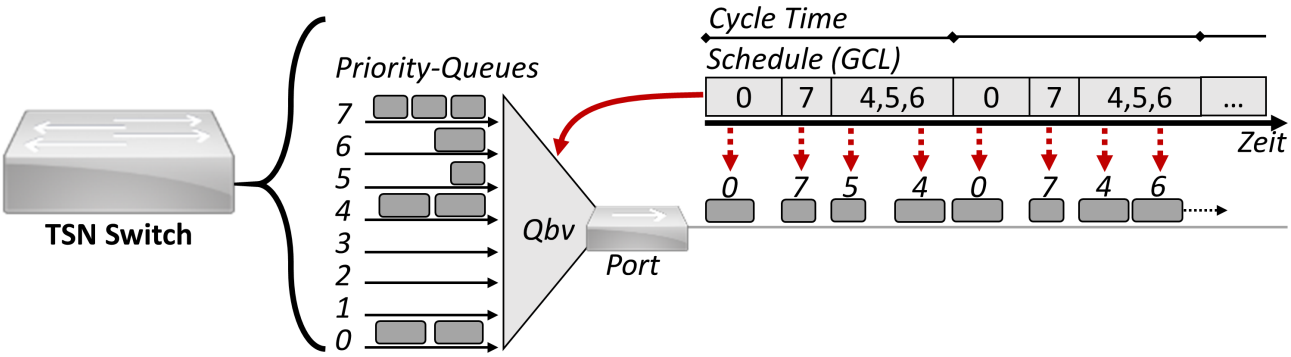


Abbildung 4.7: TSN Scheduling entsprechend dem Standard IEEE 802.1Qbv (Norm IEEE 802.1Qbv).

IEEE 802.1Qbu Beim Weiterleiten von echtzeitkritischen TSN Frames im Netzwerk werden zwei technische Verfahren unterschieden. Das erste Verfahren wird als Frame-Preemption bezeichnet und ist im TSN Standard IEEE 802.1Qbu definiert (Norm IEEE 802.1Qbu). Voraussetzung für Frame-Preemption ist die Unterscheidung zwischen priorisierten, echtzeitkritischen Frames und gewöhnlichem Datenverkehr. Darauf aufbauend wird die Übertragung eines gewöhnlichen Frames zugunsten von priorisierten Frames unterbrochen (Bild 4.6). Dabei muss allerdings die minimale Ethernet-Framegröße von 64 Byte gewährleistet bleiben. Zudem erhöht der Wechsel zum echtzeitkritischen Frame die Gesamtlatenz der Übertragung. Daher eignet sich Frame-Preemption insbesondere für Latenzen zwischen $1 - 10$ ms und den Einsatz auf der Steuerungsebene (Nsaibi et al. 2017b).

IEEE 802.1Qbv Für die Feldebene (Latenzen <1 ms) wird das TSN Scheduling als zweites technisches Verfahren spezifiziert. Dieses Scheduling basiert auf dem Time-Aware Shaper (TAS) aus dem TSN Standard IEEE 802.1Qbv (Norm IEEE 802.1Qbv). Grundlage für das TSN Scheduling ist die Klassifizierung des Datenverkehrs in insgesamt 8 VLAN-Prioritäten (IEEE 802.1Q (Norm IEEE 802.1Q)). Dabei haben gewöhnliche Frames die Priorität 0 und mit steigender Priorität erhöhen sich auch die (Echtzeit-)Anforderungen der Frames. Darauf aufbauend beschreibt das TSN Scheduling das explizite Reservieren von Zeitslots für bestimmte Prioritäten. Diese Zeitslots sind in einer sogenannten Gate Control List (GCL) definiert und wiederholen sich entsprechend einer ebenfalls definierten Zykluszeit. Durch das explizite Reservieren von Zeitslots wird die Verzögerung von echtzeitkritischen Frames auf Grund von anderem, gewöhnlichen Datenverkehr ausgeschlossen. Folglich eignet sich TSN Scheduling für eine deterministische Übertragung mit minimaler Latenz und genügt den höchsten Echtzeitanforderungen (Nsaibi et al. 2017b).

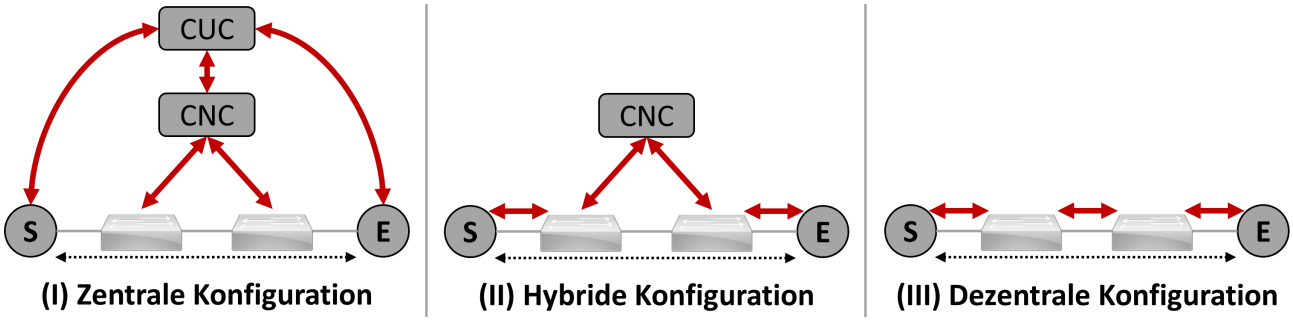


Abbildung 4.8: Drei Varianten zur Konfiguration von TSN Netzwerken (Norm IEEE 802.1Qcc).

IEEE 802.1CB Der TSN Standard IEEE 802.1CB ermöglicht eine ausfallsichere Übertragung von TSN Frames im Netzwerk (Norm IEEE 802.1CB). Dazu definiert der Standard insgesamt vier Redundanzmechanismen:

Frame Duplizierung Der erste Mechanismus spezifiziert die Duplizierung von TSN Frames und ermöglicht die parallele Übertragung über disjunkte Netzwerkpfade.

Sequenz De-/Codierung Eine Sequenznummer dient zur Wiederherstellung der Frame-Reihenfolge und zur Erkennung redundanter TSN Frames. Zur Sequenzcodierung spezifiziert der TSN Standard das sogenannte Redundancy Tag mit dem Ethertype “0xF1C1”.

Frame Identifizierung Eine Identifizierungsfunktion ordnet eingehende TSN Frames einer aktiven (redundanten) TSN Verbindung zu. Dazu verwendet die Funktion bestimmte Felder im Header des TSN Frames oder Protokollinformationen auf den höheren OSI-Schichten.

Frame Eliminierung Der vierte Mechanismus spezifiziert einen Algorithmus, der redundante TSN Frames erkennt und verwirft.

Die standardisierten Redundanzmechanismen können sowohl in die Switches als auch in die echtzeitfähigen Assets integriert werden. Unter der Voraussetzung von zwei disjunkten Netzwerkpfaden wird dadurch eine ausfallsichere Übertragung von echtzeitkritischen TSN Frames gewährleistet. Insbesondere wird durch die parallele Übertragung der Verlust von echtzeitkritischen Daten im Falle eines Fehlers vermieden.

IEEE 802.1Qcc Der TSN Standard IEEE 802.1Qcc bildet die Grundlage für die Konfiguration von TSN Netzwerken. Dabei spezifiziert der Standard drei Konfigurationsvarianten:

Zentrale Konfiguration Die erste Variante basiert auf zwei zentralen Netzwerkkomponenten, der sogenannten CUC (Central User Configuration) und der CNC (Central Network Con-

figuration). Die CUC dient als Schnittstelle zu den echtzeitfähigen Assets und übergibt die TSN-spezifischen Kommunikationsanforderungen an die CNC. Darauf aufbauend realisiert die CNC die Konfiguration der TSN Mechanismen in der Netzwerkinfrastruktur.

Dezentrale Konfiguration Bei der zweiten Variante verwenden die echtzeitfähigen Assets ein dezentrales Reservierungsprotokoll zur Konfiguration der TSN Verbindungen im Netzwerk. Dazu wird beispielsweise das Stream Reservation Protocol (SRP (Norm IEEE 802.1Qat)) bzw. das Resource Allocation Protocol (RAP (Norm IEEE 802.1Qdd)) verwendet und für die TSN-spezifischen Mechanismen erweitert.

Hybride Konfiguration Bei der dritten Variante verwenden die echtzeitfähigen Assets ebenfalls ein dezentrales Reservierungsprotokoll. Die Anfrage wird anschließend vom nächstgelegenen Switch an eine zentrale CNC weitergeleitet. Diese CNC realisiert die eigentliche Konfiguration der TSN Verbindungen in der Netzwerkinfrastruktur.

Weitere TSN Standards Neben den vorgestellten Standards existieren noch weitere TSN Standards, wie z.B. IEEE 802.1Qca, IEEE 802.1Qci, IEEE 802.1Qch oder IEEE 802.1Qcr (IEEE 2020; Bello et al. 2019; Nasrallah et al. 2018). Darüber hinaus entwickelt die TSN Task Group die verschiedenen Standards kontinuierlich weiter und definiert neue Standards (IEEE 2020).

Echtzeitfähigkeit Das TSN Scheduling entsprechend dem Standard IEEE 802.1Qbv ermöglicht eine deterministische Übertragung von echtzeitkritischen Frames mit geringer Latenz (Norm IEEE 802.1Qbv). Insbesondere genügt das TSN Scheduling den höchsten Echtzeitanforderungen auf der Feldebene. Zum Nachweis der Echtzeitfähigkeit existieren zum einen theoretische Berechnungen basierend auf dem Network Calculus (Zhang et al. 2019; Zhao et al. 2018). Zum anderen sind die TSN Mechanismen in verschiedenen Simulationsumgebungen realisiert (Jiang et al. 2019; Pahlevan et al. 2018a; Falk et al. 2019; Nasrallah et al. 2019). Dabei wird das Verhalten eines einzelnen TSN Switches simuliert und die benötigte Zeit zur Übertragung von TSN Frames analysiert. Darauf aufbauend liegt der Fokus in dieser Arbeit auf einer konkreten Implementierung um die Machbarkeit des Lösungsansatzes zu demonstrieren. Dafür existieren bereits prototypische TSN Switches, die ein echtzeitfähiges, Port-basiertes TSN Scheduling unterstützen (Belden 2020b; Bruckner et al. 2018). Bei diesen Prototypen ist die Konfiguration allerdings noch proprietär und erfordert ein manuelles, herstellerspezifisches Engineering. Daher wird in dieser Arbeit eine einheitliche, herstellerunabhängige Konfiguration der TSN Switches in der Netzwerkinfrastruktur ermöglicht.

Ausfallsicherheit Der TSN Standard IEEE 802.1CB ermöglicht eine ausfallsichere TSN Kommunikation (Norm IEEE 802.1CB). Ein detaillierter Vergleich zu anderen existierenden Redundanz-Standards ist in (Kehrer et al. 2014) aufgeführt. Dabei wird insbesondere das (manuelle) Engineering von redundanten TSN Verbindungen als wesentlicher Aspekt für die Akzeptanz einer zukünftigen Lösung hervorgehoben (Kehrer et al. 2014). Daher wird in dieser Arbeit die automatische Konfiguration von redundanten TSN Verbindungen betrachtet und der manuelle Engineering-Aufwand verringert.

In (Álvarez et al. 2017) wurden einige Defizite des heutigen TSN Standards identifiziert und eine redundante Konfiguration mit mehreren CNCs vorgestellt. Darauf aufbauend wurde in (Álvarez et al. 2018a; Álvarez et al. 2019) der Aspekt der zeitlichen Redundanz betrachtet. Dabei wird ein TSN Frame nicht nur über disjunkte Netzwerkpfade übertragen (räumliche Redundanz), sondern auch mehrfach über den gleichen Pfad (Álvarez et al. 2018b). Stand heute sind diese Verbesserungsvorschläge allerdings proprietär und es ist fraglich, ob der TSN Standard entsprechend erweitert wird.

Variabilität TSN erweitert Standard-Ethernet und ermöglicht ebenfalls eine Switch-basierte Netzwerkinfrastruktur. Die zusätzlichen TSN Mechanismen modifizieren die Sicherungsschicht, sodass die Bitübertragungsschicht weiterhin verschiedene Datenraten unterstützt. Konkret wird Stand heute eine Datenrate von 100 MBit/s und insbesondere 1 GBit/s unterstützt (Nsaibi et al. 2018). Für die Zukunft wird bereits an einer Übertragung mit 10 MBit/s bzw. 10 GBit/s gearbeitet (IEEE 2020). Dadurch wird eine skalierbare TSN Kommunikation gewährleistet.

Komponentenbeschreibung Analog zu Ethernet verwendet TSN YANG-Modelle zur Beschreibung und Konfiguration der Kommunikationsmechanismen im Netzwerk (Norm IEEE 802.1Qcp). Dabei sind insbesondere neue YANG-Modelle für die TSN-spezifischen Kommunikationsmechanismen geplant (z.B. IEEE 802.1Qcw (Norm IEEE 802.1Qcw) und IEEE 802.1CBcv (Norm IEEE 802.1CBcv)). Diese YANG-Modelle befinden sich allerdings noch in der Standardisierungsphase.

Darüber hinaus ist TSN eine Grundlagentechnologie und bietet umfangreiche Kommunikations- und Konfigurationsmechanismen für verschiedene Anwendungsgebiete. Dabei werden nicht alle Standards für einen spezifischen Anwendungsfall benötigt bzw. teilweise schließen sich die TSN Standards gegenseitig aus. Daher werden sogenannte TSN Profile standardisiert (z.B. (Norm IEC/IEEE 60802; Norm IEEE 802.1DG)). Diese Profile beschreiben die Anwendung von TSN und die notwendigen TSN Mechanismen für das jeweilige Anwendungsgebiet.

Traffic Class	Perio-dicity	Period	Sync to Network	Data Delivery Requirements	Tolerance to Interference	Tolerance to Loss	Application Data Size (Bytes)	Criticality
Isochronous	periodic	< 2ms	yes	Deadline	0	none	30-100 fixed	high
Cyclic	periodic	2-20ms	no	Latency	<=Latency	1-4 Frames	50-1000 fixed	high
Events	sporadic	-	no	Latency	-	yes	100-1500 variable	high
Network Control	periodic	50ms-1s	no	Bandwidth	yes	yes	50-500 variable	high
Config & Diagnostics	sporadic	-	no	Bandwidth	-	yes	500-1500 variable	medium
Best-Effort	sporadic	-	no	-	-	yes	30-1500 variable	low
Video	periodic	Frame Rate	no	Latency	-	yes	1000-1500 variable	low
Audio/Voice	periodic	Sampling Rate	no	Latency	-	yes	1000-1500 variable	low

Abbildung 4.9: Definition der Verkehrsklassen im "Industrial Automation" Profil (Ademaj 2019).

Beispielweise wird für den Einsatz in Produktionssystemen das TSN Profil "Industrial Automation" entwickelt (Norm IEC/IEEE 60802). Das "Industrial Automation" Profil beschreibt typische industrielle Anwendungsfälle und leitet daraus insgesamt 8 verschiedene Verkehrsklassen ab (Bild 4.9). Diese Verkehrsklassen haben ansteigende Kommunikationsanforderungen und sollen in der Zukunft auf die spezifischen TSN Mechanismen abgebildet werden (Ademaj 2019). Stand heute befindet sich das "Industrial Automation" Profil allerdings noch in der Standardisierungsphase (Norm IEC/IEEE 60802; Ademaj 2019).

Netzwerkkonfiguration Der TSN Standard IEEE 802.1Qcc definiert die Rahmenbedingungen zur Netzwerkkonfiguration (Norm IEEE 802.1Qcc). Darauf aufbauend wurde insbesondere die zentrale Konfigurationsvariante mit einer CNC/ CUC konkretisiert und erweitert. Beispielsweise wurde in (Böhm et al. 2019b; Álvarez et al. 2018a) eine Konfiguration mit mehreren TSN-Subnetzen und CNCs betrachtet. Ein weiterer Ansatz präsentierte selbstlernende TSN Netzwerke, in denen die CNC den aktuellen Datenverkehr analysiert und entsprechende TSN Schedules generiert (Gutiérrez et al. 2017a). Die Machbarkeit und die Qualität der generierten Schedules, insbesondere für größere TSN Netzwerke, wurde allerdings nicht weiter evaluiert. Darüber hinaus liegt der Fokus der bisherigen Ansätze auf der Konfiguration von einzelnen TSN Verbindungen. Darauf aufbauend werden in dieser Arbeit auch redundante bzw. komplexere TSN Verbindungen betrachtet.

Ein wesentlicher Teilaspekt der Netzwerkkonfiguration ist das TSN Scheduling, d.h. die Berechnung der einzelnen Schedules basierend auf dem Standard IEEE 802.1Qbv (Norm IEEE 802.1Qbv). Das TSN Scheduling ist in der Theorie NP-vollständig und skaliert mit der Anzahl Switches und TSN Verbindungen im Netzwerk (Bello et al. 2019; Pahlevan et al. 2018b).

Daher sind möglichst effiziente Verfahren zur Lösung des (mathematischen) Problems Gegenstand aktueller Forschung (Bello et al. 2019). Etablierte Verfahren basieren beispielsweise auf Satisfiability Modulo Theories (SMT (Barrett et al. 2018)) und werden bereits in anderen Echtzeitprotokollen eingesetzt (z.B. TTEthernet (Finzi et al. 2019)). Der Einsatz von SMT für das TSN Scheduling wird in (Craciunas et al. 2016; Oliver et al. 2018) detailliert erläutert. Neben SMT existieren noch weitere Verfahren zur Berechnung der TSN Schedules, insbesondere verschiedene Heuristiken (Dürr et al. 2016; Raagaard et al. 2017; Pahlevan et al. 2018b; Pop et al. 2018; Ansah et al. 2019a). Darauf aufbauend wird in der vorliegenden Arbeit ebenfalls eine Heuristik für das dynamische TSN Scheduling zur Laufzeit verwendet. Die optimale Lösung für das TSN Scheduling ist auf Grund der mathematischen Komplexität nicht Gegenstand dieser Arbeit.

Zusammengefasst bildet die neue Kommunikationstechnologie TSN eine vielversprechende Basis für eine zukünftige Kommunikationsinfrastruktur. Wesentlicher Vorteil ist die Ethernet-Kompatibilität in Kombination mit den IEEE-konformen Echtzeiterweiterungen. Basierend auf TSN existieren weitere Lösungsansätze mit anderen Technologien (Schriegel et al. 2017). Diese Ansätze werden in den folgenden Abschnitten detailliert betrachtet.

4.5 TSN und Industrial Ethernet

Die etablierten Industrial Ethernet Protokolle (z.B. Sercos, EtherCAT, Profinet) verfügen Stand heute über nicht IEEE-konforme und insbesondere inkompatible Ethernet-Erweiterungen. Die einzelnen Nutzerorganisationen der Protokolle verfolgen daher das Ziel, die individuellen Ethernet-Erweiterungen durch IEEE-konforme TSN Mechanismen zu ersetzen (Sercos 2020; Weber 2019; PNO 2017). Davon werden sich beispielsweise reduzierte Kosten durch standardisierte Hardware sowie eine bessere Skalierbarkeit auf Grund höherer Datenraten versprochen (Sercos 2020).

Echtzeitkommunikation Die Industrial Ethernet Protokolle der Klasse 3 realisieren ein zeitliches Scheduling auf der Sicherungsschicht. Dieses proprietäre Scheduling soll in der Zukunft durch ein IEEE-konformes TSN Scheduling entsprechend dem Standard IEEE 802.1Qbv ersetzt werden (Norm IEEE 802.1Qbv). Dazu existieren bereits erste Konzepte und Simulationen für die einzelnen Protokolle (Schriegel et al. 2017; Nsaibi et al. 2018; Szancer 2017; Nsaibi et al. 2017a).

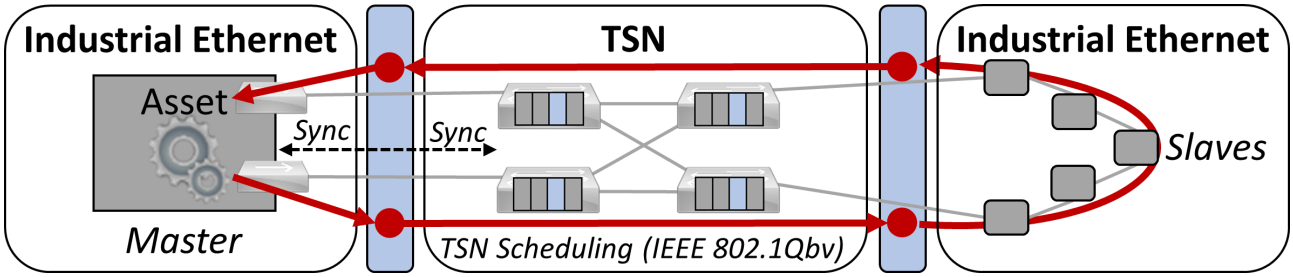


Abbildung 4.10: Übertragung von Industrial Ethernet Frames über ein TSN Netzwerk (Weber 2019).

Voraussetzung für das Scheduling ist ein einheitliches Zeitverständnis im Netzwerk. Daher wird eine Zeitsynchronisation (IEEE 802.1AS-Rev (Norm IEEE 802.1AS-Rev)) zwischen dem protokollspezifischen Master und den TSN Switches in der Netzwerkinfrastruktur etabliert. Darauf aufbauend werden exklusive Zeitslots mit einer gewissen Priorität für die Industrial Ethernet Frames reserviert (IEEE 802.1Qbv (Norm IEEE 802.1Qbv)). Diese Zeitslots gewährleisten eine echtzeitfähige Übertragung durch das TSN Netzwerk bis zu den Slaves (Bild 4.10). Die Slaves, d.h. echtzeitfähigen Assets, verarbeiten eingehende Frames wie gewohnt und verfügen über keine TSN-spezifischen Mechanismen (Nsaibi et al. 2018). Da die Umstellung der Industrial Ethernet Protokolle auf TSN ausschließlich die OSI-Schichten 1-2 betrifft, bleiben die darüberliegenden Protokollschichten unverändert.

Komponentenbeschreibung und Netzwerkkonfiguration Zur Beschreibung der echtzeitfähigen Assets werden die existierenden Applikationsprofile der jeweiligen Protokolle verwendet. Basierend auf diesen Profilen werden die Anforderungen an die TSN-Kommunikation abgeleitet (z.B. Zykluszeit, Framegröße) und das TSN Scheduling im Netzwerk konfiguriert. Die Koexistenz verschiedener Echtzeitprotokolle und die dynamische Konfiguration der gesamten Netzwerkinfrastruktur mit weiteren TSN Standards ist Stand heute nicht im Fokus.

Die Kombination aus den etablierten Industrial Ethernet Protokollen und TSN ermöglicht eine durchgängige Echtzeitkommunikation zwischen den Assets. Dabei liegt der Fokus auf der Integration der erforderlichen TSN Mechanismen in das jeweilige Protokoll und der Verwendung von standardkonformer Hardware. Eine protokollunabhängige und dynamische Konfiguration von TSN zur Laufzeit wird nicht weiter betrachtet.

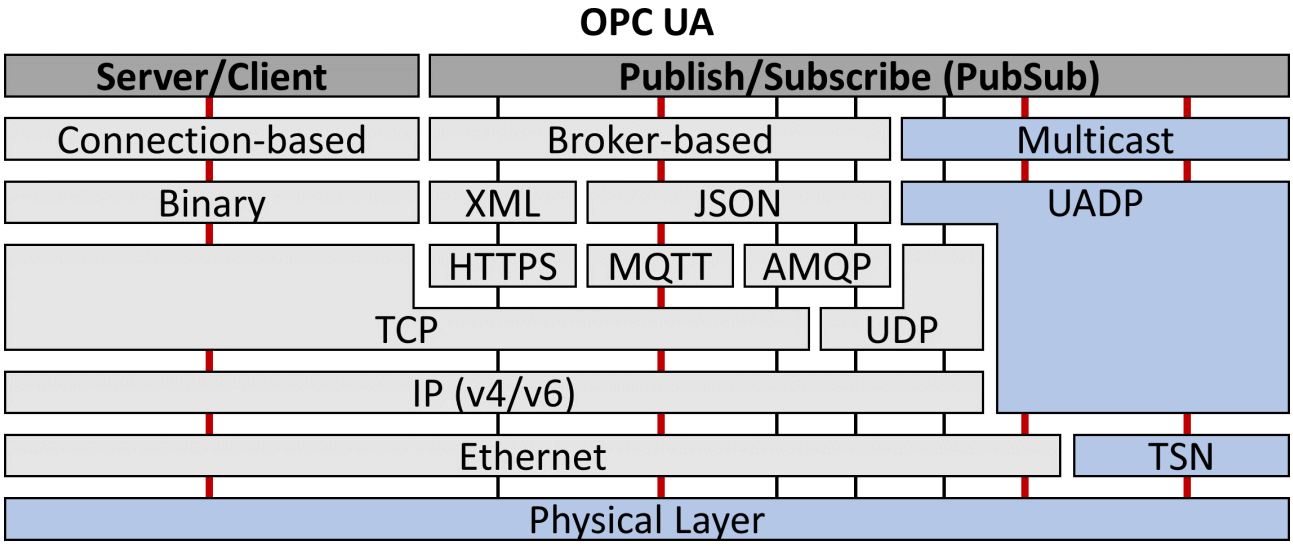


Abbildung 4.11: Echtzeitkommunikation basierend auf TSN und OPC UA PubSub (Eckhardt et al. 2018).

4.6 TSN und OPC UA PubSub

Im Kontext der vierten industriellen Revolution hat sich unter anderem OPC Unified Architecture (UA) als Anwendungsprotokoll etabliert (Zezulka et al. 2018). OPC UA wurde ursprünglich als verbindungsorientiertes Protokoll mit einer Client/Server Architektur spezifiziert (Norm IEC 62541), was den Einsatz für die Echtzeitkommunikation verhinderte (Eckhardt et al. 2018). Daher wurde OPC UA Publish/Subscribe (PubSub) als Erweiterung für die Echtzeitkommunikation im OT-Netzwerk standardisiert (Norm IEC 62541-14). Dieses Protokoll basiert auf einer Publish/Subscribe Architektur mit verschiedenen Transportprotokollen (Bild 4.11). Für die deterministische Kommunikation mit harten Echtzeitanforderungen wird dabei insbesondere natives TSN verwendet.

Echtzeitkommunikation Die Kombination aus OPC UA PubSub und TSN wurde in (Bruckner et al. 2019; Bruckner et al. 2018) detailliert vorgestellt. Diese Kombination ist insbesondere für die Echtzeitkommunikation auf der Feldebene (OT-Netzwerk) geeignet (Eckhardt et al. 2018). Dabei wurde die Echtzeitfähigkeit basierend auf verschiedenen prototypischen Implementierungen nachgewiesen (Eckhardt et al. 2019; Pfrommer et al. 2018; Bruckner et al. 2018).

Darüber hinaus wurde auch die Kombination von OPC UA Client/Server und TSN evaluiert (Gogolev et al. 2018; Gogolev et al. 2019). Dabei ermöglichen die TSN Mechanismen, insbe-

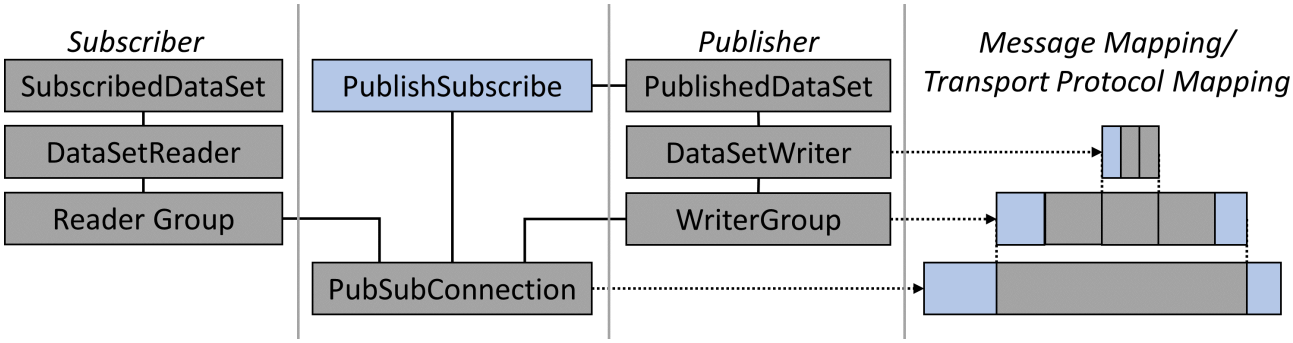


Abbildung 4.12: Das Informationmodelle für OPC UA PubSub basiert auf den verschiedenen ObjectTypes (Norm IEC 62541-14).

sondere das TSN Scheduling, eine deterministische Übertragung der OPC UA Nachrichten. Allerdings genügen die resultierenden Latenzen ($\approx$ 2 ms) nicht den harten Echtzeitanforderungen von Assets auf der Feldebene (<1 ms) (Eckhardt et al. 2018).

Komponentenbeschreibung OPC UA (PubSub) ermöglicht eine umfangreiche Modellierung von Assets und ihren Fähigkeiten in Form von Informationsmodellen (Norm IEC 62541; Zezulka et al. 2018). Dabei existieren sowohl OPC UA-eigene Informationsmodelle, erweiterte Informationsmodelle von anderen Standardisierungsorganisationen (Companion Specifications) als auch individuelle Informationsmodelle (Norm IEC 62541; Bruckner et al. 2019). Diese Informationsmodelle eignen sich insbesondere zur einheitlichen Beschreibung von echtzeitfähigen und nicht-echtzeitfähigen Assets im Produktionssystem.

Im Rahmen der Standardisierung von OPC UA PubSub wurde ein spezifisches Informationsmodell definiert (Bild 4.12). Dieses Modell besteht aus mehreren sogenannten ObjectTypes, die in (Norm IEC 62541-14; Prinz et al. 2019a) detailliert beschrieben sind. Ein ObjectType repräsentiert eine Teilfunktionalität des Protokolls und spezifiziert entsprechende Konfigurationsparameter und Statusinformationen. Dadurch wird die Konfiguration einer OPC UA PubSub Verbindung in den echtzeitfähigen Assets ermöglicht.

Netzwerkkonfiguration Stand heute erfolgt die Konfiguration einer OPC UA PubSub Verbindung und der dazugehörigen TSN Verbindung getrennt und weitestgehend statisch. Basierend auf den standardisierten OPC UA PubSub Informationsmodell werden zum einen die Konfigurationsparameter in den echtzeitfähigen Assets gesetzt. Zum anderen wird die TSN Verbindung im Netzwerk etabliert. Parallel zu der vorliegenden Arbeit entstanden erste Ansätze zur automatischen Konfiguration von OPC UA PubSub und TSN (Pop et al. 2018; Gutiérrez

et al. 2017a). Dazu wurde ein sogenannter OPC UA Connection Broker (CUC) (Gutiérrez et al. 2017a) bzw. Configuration Agent (CUC/CNC) (Pop et al. 2018) als Konfigurationsschnittstelle zwischen beiden Technologien etabliert. Der Fokus liegt dabei auf dem TSN Scheduling und der Konfiguration von einzelnen TSN Verbindungen in der Netzwerkinfrastruktur. Darüber hinaus wird in dieser Arbeit auch die dynamische Konfiguration von redundanten und komplexeren TSN Verbindungen zur Laufzeit betrachtet.

Zusammengefasst ermöglicht die Kombination aus TSN und OPC UA PubSub als Anwendungsprotokoll eine durchgängige Echtzeitkommunikation zwischen verschiedenen Assets. Auf Grund der Komplexität beider Technologien ist die automatische und dynamische Konfiguration zur Laufzeit eine wesentliche Herausforderung für die Zukunft (Bruckner et al. 2019).

4.7 TSN und Software-defined Networking

Software-defined Networking (SDN) basiert auf der Virtualisierung der physikalischen Netzwerkinfrastruktur (McKeown 2009; Mijumbi et al. 2016). Daraus resultiert eine Unterteilung in die sogenannte Data Plane und Control Plane (Bild 4.13). Auf der Data Plane realisieren die physikalischen Netzwerkkomponenten (z.B. Switches) die Übertragung von Frames zwischen dem Sender und Empfänger. Die Entscheidung über den Netzwerkpfad wird dabei auf der Control Plane getroffen und von einem zentralen SDN Controller umgesetzt. Dieser SDN Controller verfügt über zwei wesentliche Schnittstellen. Zum einen werden die Anforderungen von den Anwendungen über die Northbound-API kommuniziert. Zum anderen wird die Southbound-API zur Konfiguration der Netzwerkkomponenten verwendet. Für die Southbound-API werden die Protokolle SNMP (Case et al. 2002), NETCONF (Enns et al. 2011) und insbesondere die SDN-spezifische Entwicklung OpenFlow (Karakus et al. 2017) eingesetzt.

Ursprünglich lag der Fokus von SDN auf der Konfiguration von Ethernet Netzwerken mit entsprechenden Switches (McKeown 2009). Darauf aufbauend wurden weitere (drahtlose) Kommunikationstechnologie betrachtet (Wan et al. 2016). Inzwischen ist auch die Kombination von SDN und TSN Gegenstand aktueller Forschung (Silva et al. 2019; Ehrlich et al. 2018).

Echtzeitkommunikation und Komponentenbeschreibung SDN ermöglicht die Konfiguration einer TSN-basierten Echtzeitkommunikation. Dabei liegt der Fokus auf dem TSN Scheduling entsprechend dem Standard IEEE 802.1Qbv und der Konfiguration von Netzwerkkomponenten (Norm IEEE 802.1Qbv). Zur Beschreibung der Netzwerkkomponenten werden

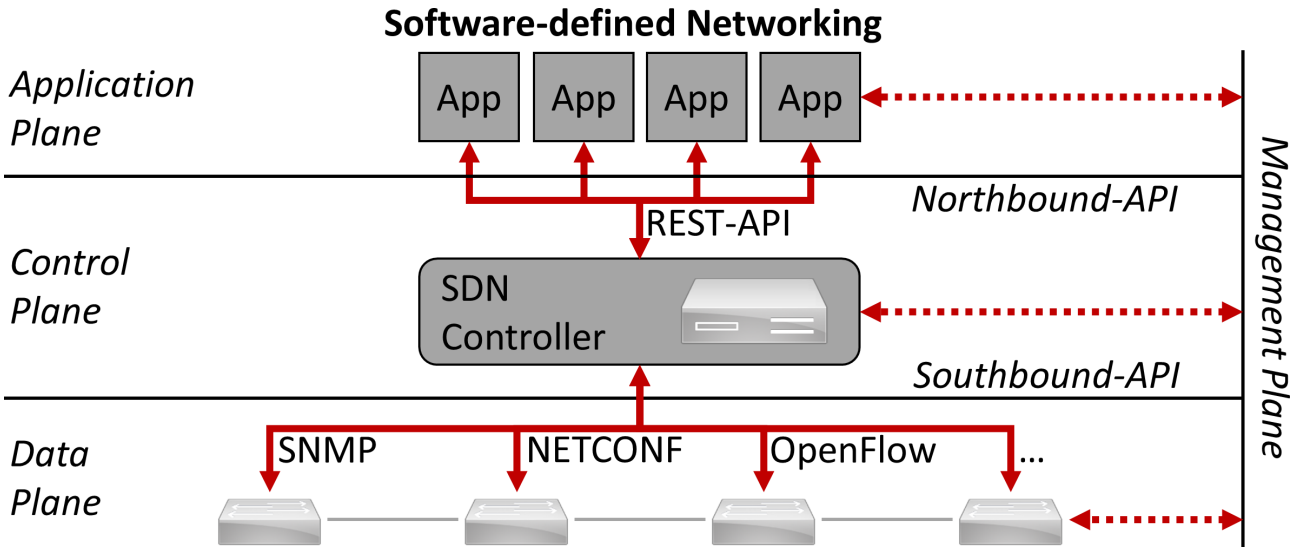


Abbildung 4.13: Architektur basierend auf Software-defined Networking (SDN) (Ehrlich et al. 2018; Ventre et al. 2018).

die TSN-spezifischen YANG-Modelle verwendet (Norm IEEE 802.1Qcw; Norm IEEE 802.1CB-cv), wobei die Standardisierung Stand heute noch nicht abgeschlossen ist. Die Beschreibung der Asset-spezifischen Fähigkeiten und die Integration von Anwendungsprotokollen wird nicht weiter betrachtet.

Netzwerkkonfiguration SDN ermöglicht eine zentrale Konfiguration von TSN Netzwerken entsprechend dem Standard IEEE 802.1Qcc (Norm IEEE 802.1Qcc). Dabei werden die Konzepte von CUC und CNC in einem zentralen SDN Controller vereint (Gerhard et al. 2019). Auch wenn in den meisten Veröffentlichungen ein zentraler SDN Controller angenommen wird, lässt sich das Konzept auf mehrere Subnetze und SDN Controller übertragen (Oktian et al. 2017; Yin et al. 2012; Phemius et al. 2014).

Ein SDN Controller konfiguriert die Netzwerkinfrastruktur und etabliert Echtzeitverbindungen mit einer gewissen Dienstgüte (Quality of Service, QoS) (Karakus et al. 2017). Dieser Vorgang wird auch als Network Slicing bezeichnet (Ansah et al. 2019b; Kalør et al. 2018). Dabei existieren unabhängig von TSN verschiedene Ansätze für das Scheduling von echtzeitkritischen Frames im Netzwerk (Karakus et al. 2017; Schweissguth et al. 2016; Nayak et al. 2016; Nayak et al. 2017). Parallel zur vorliegenden Arbeit wurden weitere TSN-spezifische Ansätze entwickelt und das TSN Scheduling (IEEE 802.1Qbv (Norm IEEE 802.1Qbv)) realisiert (Gerhard et al. 2019; Silva et al. 2019; Böhm et al. 2019a; Said et al. 2019). Darüber hinaus wird in dieser

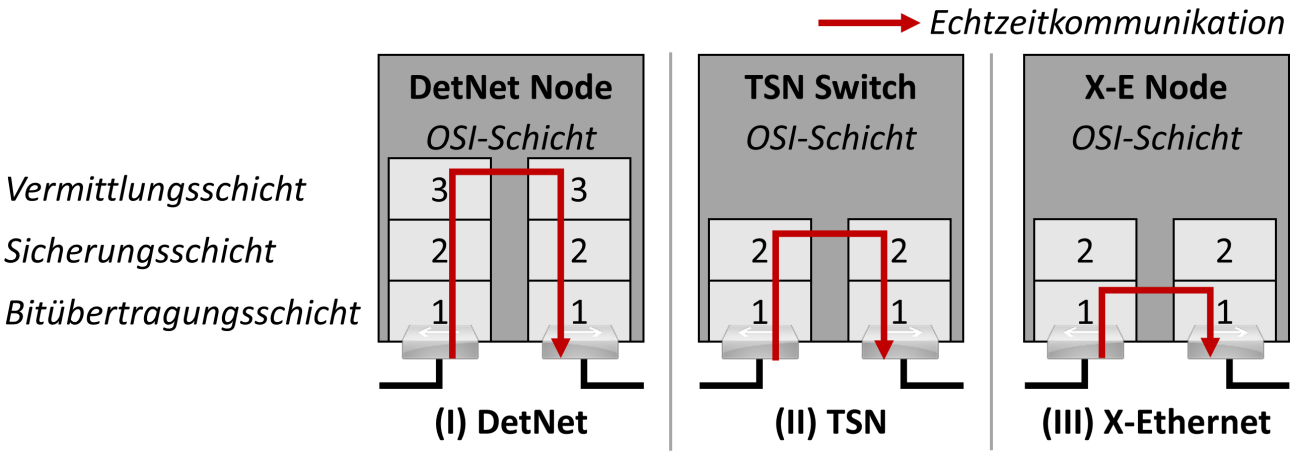


Abbildung 4.14: Vergleich der Netzwerkkomponenten zur Echtzeitkommunikation.

Arbeit die dynamische Konfiguration von redundanten und komplexeren TSN Verbindungen in der gesamten Kommunikationsinfrastruktur betrachtet.

Zusammengefasst ermöglicht SDN eine zentrale Konfiguration von TSN Netzwerken und realisiert insbesondere das TSN Scheduling. Dabei liegt der Fokus auf der Konfiguration von TSN-fähigen Switches in der Netzwerkinfrastruktur. Die Parametrierung von echtzeitfähigen Assets und ihren spezifischen Fähigkeiten wird nicht weiter betrachtet.

4.8 Weitere Ansätze

DetNet Die neue Kommunikationstechnologie TSN spezifiziert die OSI-Schichten 1-2 und ermöglicht eine deterministische Übertragung von Ethernet Frames mit geringer Latenz. Darauf aufbauend wurde die IETF Deterministic Networking (DetNet) Working Group gegründet (IETF 2020). Diese Arbeitsgruppe erweitert die OSI-Schicht 3, d.h. insbesondere das Internet Protocol (IP (Deering et al. 2017)), um zusätzliche Echtzeitmechanismen (Bild 4.14). Ziel ist die Standardisierung einer echtzeitfähigen, ausfallsicheren Übertragung von IP-Paketen über mehrere TSN-Subnetze (Nasrallah et al. 2018). Stand heute befindet sich die Technologie allerdings noch in der Standardisierungsphase.

DDS Neben OPC UA hat sich der Standard Data Distribution Service (DDS) als Anwendungsprotokoll im industriellen Umfeld etabliert (Marcon et al. 2017; Maruyama et al. 2016). DDS ermöglicht den Austausch von echtzeitkritischen Nachrichten basierend auf einer verteilten, datenzentrierten Architektur (Pardo-Castellote 2003). Dabei wird das Real-time Publish-

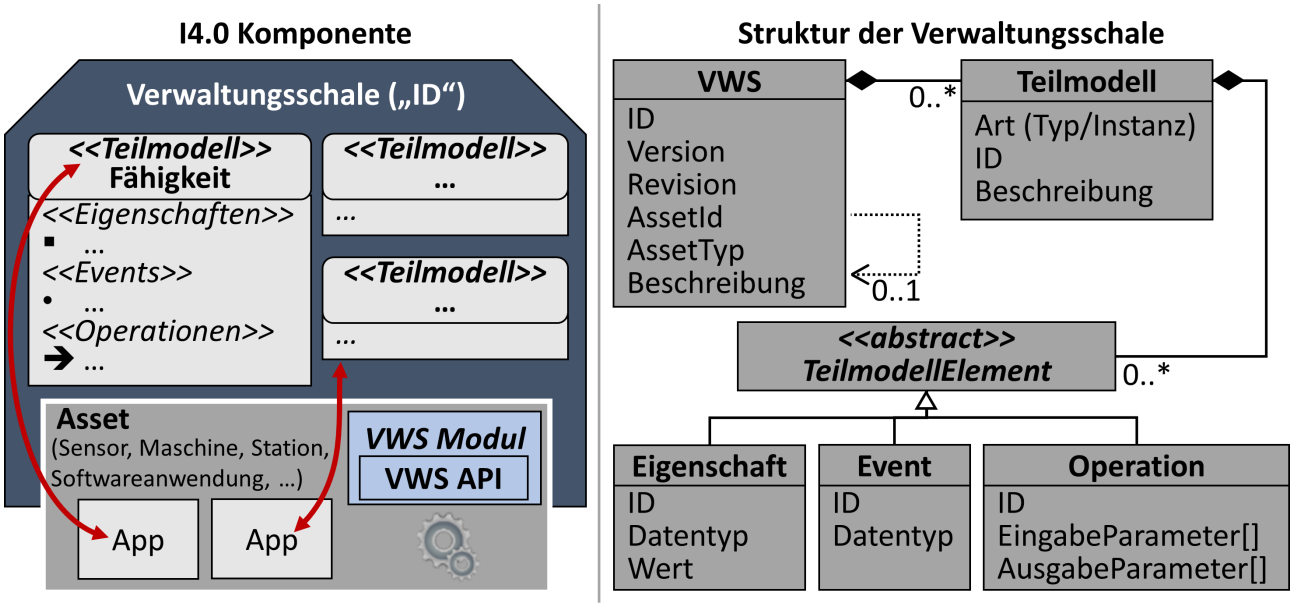


Abbildung 4.15: I4.0 Komponente mit Verwaltungsschale und der dazugehörigen Struktur (BMWi 2018a).

Subscribe Protocol (RTPS (Felser 2005)) als Transportprotokoll für die Nachrichten verwendet. Darauf aufbauend wird die Integration von TSN und eine TSN-basierte DDS Komunikation in der Zukunft angestrebt (Agarwal et al. 2019).

X-Ethernet Die neue Kommunikationstechnologie X-Ethernet ermöglicht analog zu TSN eine echtzeitfähige Übertragung von Ethernet Frames (Li et al. 2017). Grundlage für die neue Technologie bildet Flexible Ethernet (FlexE (OIF 2018)). FlexE spezifiziert die Link Aggregation bzw. Segmentierung von Ethernet Verbindungen (z.B. 1/10/100 GBit/s) und ermöglicht flexiblere Datenraten (z.B. 5/25/50 GBit/s). Darauf aufbauend modifiziert X-Ethernet die Bitübertragungsschicht in den Netzwerkkomponenten (Bild 4.14). Dadurch wird das Ethernetkonforme Store-and-Forward Switching auf der Sicherungsschicht umgangen und die Latenzen beim Weiterleiten signifikant verringert (Li et al. 2017). Allerdings ist X-Ethernet Stand heute noch proprietär und steht in Konkurrenz zu der IEEE Technologie TSN.

I4.0 Komponente Im Kontext der vierten industriellen Revolution wurde das Konzept der I4.0 Komponente etabliert (Hoffmeister 2015). Dieses Konzept wird von der “Plattform Industrie 4.0” standardisiert und ermöglicht eine herstellerunabhängige Beschreibung von Assets im Produktionssystem (BMWi 2018a). Allgemein beschreibt eine I4.0 Komponente ein Asset mit einer Verwaltungsschale, d.h. einer virtuellen digitalen Repräsentanz des Assets (Bild 4.15)

(BMWi 2019). Dabei können sowohl physikalische Assets (z.B. Sensoren, Maschinen, Stationen) als auch Software-basierte Assets (z.B. Anwendungen, Algorithmen) als I4.0 Komponente modelliert werden. Die entsprechende Verwaltungsschale zeichnet sich durch die folgenden wesentlichen Merkmale aus:

Identifikation Die Verwaltungsschale verfügt über eine eindeutige ID zur Adressierung und zur Identifikation des Assets.

I4.0-konforme Kommunikation Die Verwaltungsschale realisiert eine I4.0-konforme Kommunikation (z.B. über HTTP oder OPC UA (Koziolek 2017; Deppe et al. 2019; Fuchs et al. 2019)) und stellt eine entsprechende API zur Verfügung. Diese API ermöglicht den Zugriff auf die Selbstbeschreibung.

Selbstbeschreibung Die Verwaltungsschale beinhaltet eine Selbstbeschreibung des Assets. Diese Selbstbeschreibung umfasst den gesamten Lebenszyklus und basiert auf einem Metamodel mit Teilmodellen (Bild 4.15) (BMWi 2018a). Ein Teilmodell beschreibt jeweils eine Fähigkeit des Assets und spezifiziert dazugehörige Eigenschaften, Events sowie verfügbare Operationen.

Referenzierbarkeit Die Verwaltungsschale kann auf andere Verwaltungsschalen verweisen.

In (Ye et al. 2019) werden existierende Ansätze mit Bezug zur I4.0 Komponente und der Verwaltungsschale zusammengefasst. Dabei wird insbesondere die Notwendigkeit zur Spezifikation und Standardisierung von anwendungsspezifischen Teilmodellen hervorgehoben. Einzelne Teilmodelle sind bereits in (BMWi 2019; Lang et al. 2019; BMWi 2018b) beschrieben. Dadurch wird auch die Grundlage für ein einheitliches, herstellerunabhängiges Engineering basierend auf der Verwaltungsschale geschaffen (Terzimehić et al. 2019; Prinz et al. 2019c).

Zusammengefasst verfolgt das neue Konzept der I4.0 Komponente das Ziel die Wandelbarkeit von Assets in zukünftigen Produktionssystemen zu verbessern. Allerdings liegt der Fokus bisher auf der Beschreibung von gewöhnlichen Assets ohne spezifische Echtzeitanforderungen. Daher wird das Konzept in der vorliegenden Arbeit für die Echtzeitkommunikation erweitert.

4.9 Bewertung Stand der Technik

Das Bild 4.16 fasst die wesentlichen Lösungsansätze basierend auf dem Stand der Technik zusammen. Diese Lösungsansätze erfüllen die Anforderungen an eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur nur teilweise und stellen daher keine ganzheitliche Lösung dar.

		Produkte		Forschung			
Netzwerk		IT	OT	IT/OT			
Kommunikationsbasis		Ethernet	IE	TSN	TSN + IE	TSN + OPC UA	TSN + SDN
Echtzeit-kommunikation	Einheitliche Kommunikationsbasis	●	◑	●	●	●	●
	Echtzeitfähigkeit	○	●	●	●	●	●
	Ausfallsicherheit	○	●	●	●	●	●
	Variabilität	●	○	●	●	●	●
	Universalität	●	○	◑	○	○	◑
Parame-trierung	Einheitliche Komponentenbeschreibung	●	◑	●	◑	●	●
	Durchgängigkeit	◑	●	◑	●	◑	◑
	Universalität	○	◑	○	◑	◑	○
	Kompatibilität	○	○	○	○	●	○
Netzwerk-konfiguration	Einheitliche Konfigurationsschnittstelle	◑	◑	◑	◑	●	●
	R. Netzwerkinfrastruktur	●	●	●	●	●	●
	R. Assets	○	●	○	●	◑	◑
	R. Echtzeitverbindungen	○	◑	○	◑	○	◑

IT…Information Technology, OT…Operation Technology, IE…Industrial Ethernet, SDN…Software-defined Networking, R…Rekonfigurierbarkeit

Abbildung 4.16: Zusammenfassung vom Stand der Technik.

Insbesondere eignen sich die etablierten Kommunikationstechnologien im IT- und OT-Netzwerk nicht für eine zukünftige Kommunikationsinfrastruktur. Im Gegensatz dazu stellt eine TSN-basierte Echtzeitkommunikation einen vielversprechenden Ansatz dar. Dabei wird allerdings die einheitliche Integration und umfassende Parametrierung von (echtzeitfähigen) Assets nur teilweise betrachtet. Darüber hinaus wird der Aspekt der Wandelbarkeit mit einer dynamischen Konfiguration von Echtzeitverbindungen zur Laufzeit nur bedingt unterstützt. Dadurch wird die Vision einer CPS-basierten Automation für zukünftige Produktionssysteme eingeschränkt.

4.10 Lösungsansatz

Basierend auf den identifizierten Anforderungen wird in den folgenden Kapiteln eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur als ganzheitliche Lösung präsentiert (Bild

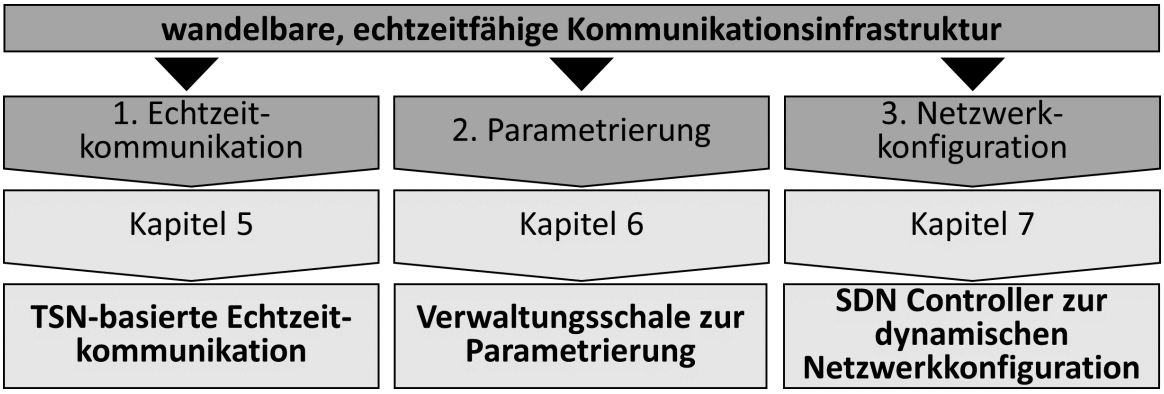


Abbildung 4.17: Lösungsansatz für eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur.

4.17). Dabei wird zunächst eine TSN-basierte Echtzeitkommunikation vorgestellt und die notwendigen Kommunikationsmechanismen in die echtzeitfähigen Assets integriert (Kapitel 5). Darauf aufbauend wird die einheitliche Parametrierung basierend auf dem Konzept der I4.0 Komponente mit Verwaltungsschale vorgestellt (Kapitel 6). Schließlich wird ein zentraler SDN Controller für die dynamische Konfiguration der Echtzeitverbindungen erweitert (Kapitel 7).

Kapitel 5

TSN-basierte Echtzeitkommunikation

In diesem Kapitel wird die Echtzeitkommunikation für die neue wandelbare, echtzeitfähige Kommunikationsinfrastruktur erarbeitet. Dadurch wird eine deterministische Kommunikation mit geringer Latenz zum Datenaustausch und zur Synchronisation zwischen echtzeitfähigen Assets ermöglicht. Grundlage für die Vernetzung von Assets ist eine einheitliche und gleichzeitig echtzeitfähige Netzwerkinfrastruktur. Darauf aufbauend werden die erforderlichen Kommunikationsmechanismen herausgearbeitet und in die echtzeitfähigen Assets integriert.

5.1 Einheitliche Kommunikationsbasis

Die neue Kommunikationstechnologie Time-Sensitive Networking (TSN) stellt eine vielversprechende Basis für eine CPS-basierte Automation dar. Wesentlicher Vorteil gegenüber existierenden Lösungen ist die Echtzeitfähigkeit in Kombination mit einer IEEE-konformen Ethernet-Erweiterung. Daher wird TSN auch als einheitliche Kommunikationsbasis für die neue Kommunikationsinfrastruktur verwendet. Dabei spezifiziert TSN allgemein die beiden untersten OSI-Schichten und bestimmt damit maßgeblich die drahtgebundene Netzwerkinfrastruktur. Gleichzeitig müssen auch die (echtzeitfähigen) Assets in der Kommunikationsinfrastruktur eine TSN-konforme Kommunikation unterstützen.

TSN-fähige Netzwerkinfrastruktur Die TSN-fähige Netzwerkinfrastruktur besteht aus Switches, die über Netzwerkkabel miteinander verbunden sind. Neben den gewöhnlichen Mechanismen zur Weiterleitung von Frames unterstützen die Switches insbesondere die TSN-spe-

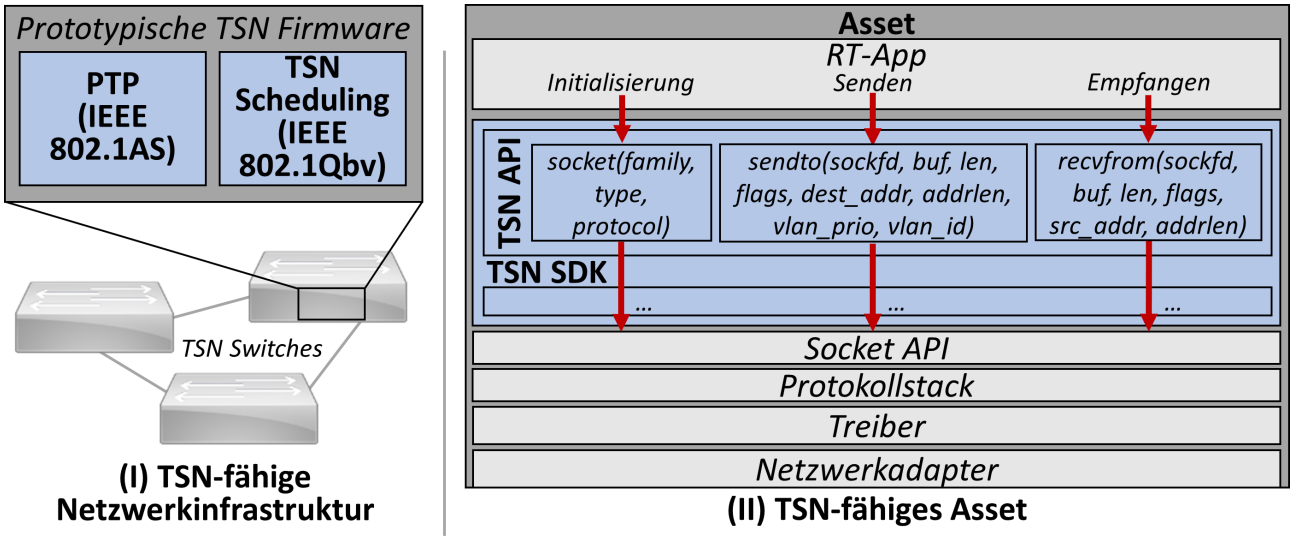


Abbildung 5.1: Die Kombination aus TSN-fähiger Netzwerkinfrastruktur und TSN-fähigen Assets bildet die Basis die vorgestellte Echtzeitkommunikation.

zifischen Kommunikationsmechanismen. Im Folgenden werden diese Switches als TSN Switches bezeichnet (Bild 5.1). Stand heute existieren TSN Switches mit einer prototypischen Firmware von verschiedenen Switchherstellern. Diese Firmware unterstützt zum einen PTP mit dem spezifischen Profil IEEE 802.1AS zur Zeitsynchronisation im Netzwerk (Norm IEEE 802.1AS). Zum anderen wird das Port-basierte TSN Scheduling entsprechend dem TSN Standard IEEE 802.1Qbv unterstützt (Norm IEEE 802.1Qbv). Diese prototypischen TSN Switches sind für die in diesem Kapitel vorgestellte Echtzeitkommunikation bereits ausreichend. Daher liegt der Fokus im Folgenden auf der TSN-konformen Kommunikation in den (echtzeitfähigen) Assets.

TSN-fähige Assets Die (echtzeitfähigen) Assets im Produktionssystem sind an die TSN-fähige Netzwerkinfrastruktur angebunden. Dazu verfügt jedes Asset über mindestens einen Netzwerkadapter, der mit einem Port eines TSN Switches verbunden ist. Für eine durchgängige Echtzeitkommunikation müssen die angeschlossenen Assets ebenfalls die TSN-spezifischen Kommunikationsmechanismen unterstützen. Stand heute fehlen diese TSN Mechanismen allerdings in den echtzeitfähigen Assets. Daher werden in diesem Kapitel die erforderlichen TSN Mechanismen standardkonform in die Assets integriert und in einem sogenannten TSN Software Development Kit (SDK) zusammengefasst (Bild 5.1). Dieses SDK erweitert den Protokollstack des Betriebssystems und stellt eine API als Schnittstelle für die (Echtzeit-)Anwendungen zur Verfügung. Damit erfüllen die erweiterten Assets auch harte Echtzeitanforderungen und bilden in Kombination mit der TSN-fähigen Netzwerkinfrastruktur die Basis für eine durchgängige Echtzeitkommunikation.

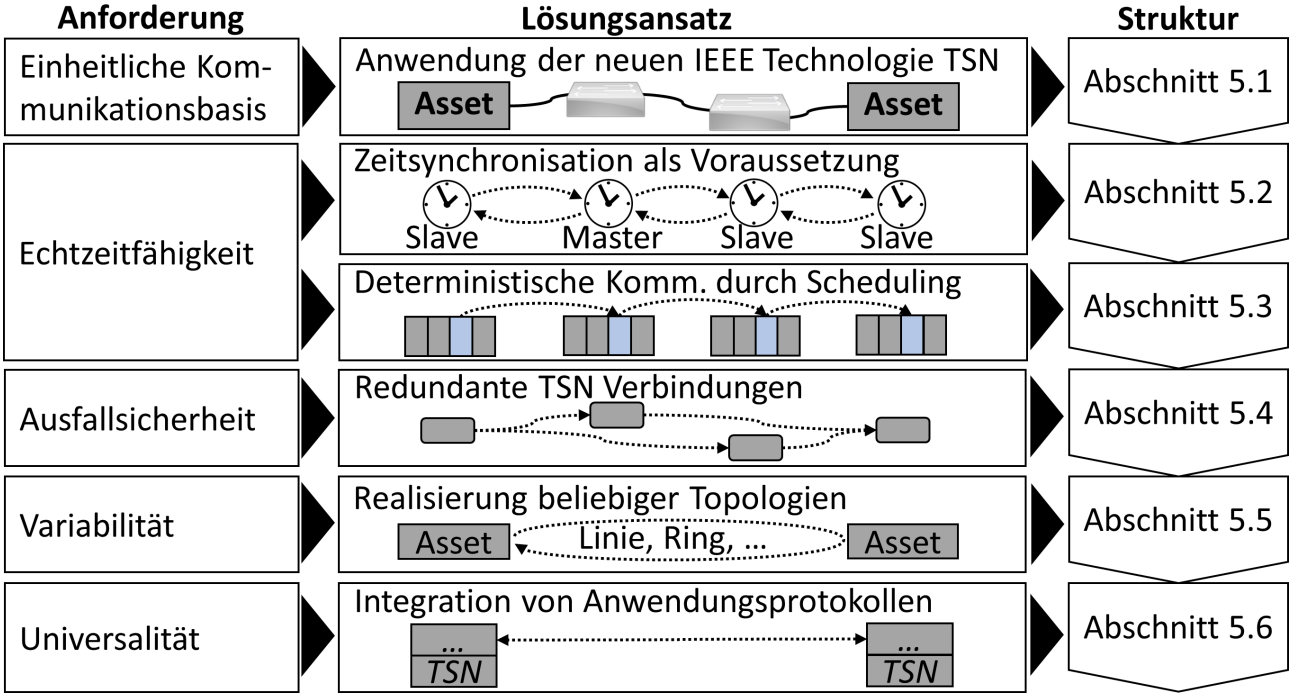


Abbildung 5.2: Die Teilaspekte der Echtzeitkommunikation basierend auf den Anforderungen.

TSN API Die von dem TSN SDK bereitgestellte API basiert auf dem weit verbreiteten Linux Betriebssystem und der dazugehörigen Socket API (Linux 2020). Dabei wurde versucht, die Änderungen gegenüber der etablierten Socket API so gering wie möglich zu halten (Linux 2020). Dadurch sind bei der Umstellung auf eine TSN-konforme Kommunikation nur minimale Anpassungen an den existierenden (Echtzeit-)Anwendungen notwendig. Die resultierende TSN API beinhaltet drei grundlegende Funktionen zur Initialisierung sowie zum Senden und Empfangen von TSN Frames (Bild 5.1). Dabei spezifiziert die Funktion “sendto” mit der VLAN-ID und Priorität zwei zusätzliche Parameter für die TSN-basierte Echtzeitkommunikation. Diese Parameter werden beim Erstellen des TSN Frames verwendet und ermöglichen das Prioritätenbasierte TSN Scheduling.

TSN SDK Im Folgenden wird das neue TSN SDK für die echtzeitfähigen Assets im Detail vorgestellt. Dabei werden insbesondere die erforderlichen TSN Mechanismen herausgearbeitet und die Interaktion mit der TSN-fähigen Netzwerkinfrastruktur erläutert. Durch die Verwendung der TSN Technologie wird die Anforderung an eine standardisierte, Ethernet-kompatible Kommunikationsbasis erfüllt. Basierend auf den weiteren Anforderungen an die Echtzeitkommunikation, gliedert sich die vorgestellte Lösung in den echtzeitfähigen Assets in fünf Teilaspekte (Bild 5.2):

Zeitsynchronisation Die harten Echtzeitanforderungen der Assets auf der Feldebene erfordern das explizite Reservieren von Netzwerkbandbreite zu gewissen Zeitpunkten. Dazu ist eine Zeitsynchronisation zwischen allen TSN Switches und den echtzeitfähigen Assets in der Kommunikationsinfrastruktur notwendig.

Dynamische TSN Verbindungen Basierend auf der Zeitsynchronisation wird das TSN Scheduling entsprechend dem Standard IEEE 802.1Qbv in den echtzeitfähigen Assets realisiert (Norm IEEE 802.1Qbv). Dieses Scheduling gewährleistet eine deterministische Übertragung von TSN Frames mit geringer Latenz. Dadurch wird die Grundlage für dynamische TSN Verbindungen zwischen verschiedenen Assets in der Kommunikationsinfrastruktur geschaffen.

Redundante TSN Verbindungen Echtzeitfähige Assets und insbesondere sicherheitskritische Systeme erfordern neben der Echtzeitfähigkeit auch eine ausfallsichere Kommunikation. Dazu werden die erforderlichen Redundanzmechanismen in die echtzeitfähigen Assets integriert und redundante TSN Verbindungen ermöglicht.

Virtuelle Netzwerktopologien Basierend auf den einzelnen TSN Verbindungen wird die Vernetzung von mehr als zwei echtzeitfähigen Assets betrachtet. Dazu werden sogenannte virtuelle Netzwerktopologien präsentiert, die von der physikalischen TSN Infrastruktur abstrahieren und Linien- oder Ring-Topologien ermöglichen. Die erforderlichen Kommunikationsmechanismen werden ebenfalls in das TSN SDK der echtzeitfähigen Assets integriert.

Anwendungsprotokolle Die TSN Verbindungen bilden die Basis für eine echtzeitfähige Kommunikation zwischen verschiedenen Assets im Netzwerk. Darauf aufbauend wird die Integration von Anwendungsprotokollen in die echtzeitfähigen Assets erläutert.

Die Kombination der fünf Teilaspekte ermöglicht eine durchgängige Echtzeitkommunikation zwischen verschiedenen echtzeitfähigen Assets im Produktionssystem.

5.2 Zeitsynchronisation

TSN bildet die Basis für die Echtzeitkommunikation in der neuen Kommunikationsinfrastruktur. Dabei wird insbesondere das TSN Scheduling entsprechend dem Standard IEEE 802.1Qbv

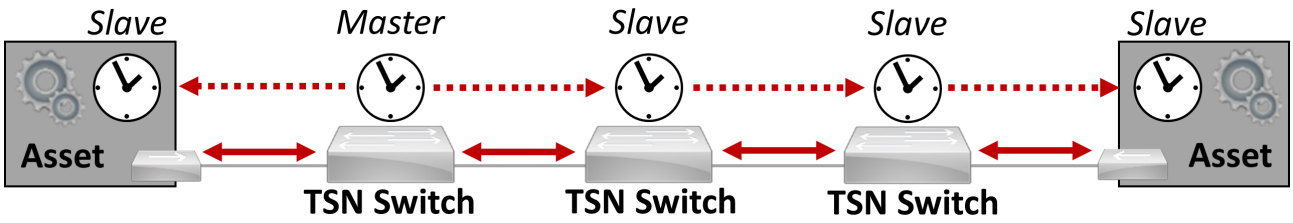


Abbildung 5.3: Zeitsynchronisation zwischen TSN-fähiger Netzwerkinfrastruktur und den echtzeitfähigen Assets.

angewendet (Norm IEEE 802.1Qbv). Dieses Scheduling basiert auf der Reservierung von Netzwerkbandbreite zu gewissen Zeitpunkten und gewährleistet eine deterministische Datenübertragung mit geringer Latenz. Dadurch werden im Gegensatz zu Frame Preemption (IEEE 802.1Qbu (Norm IEEE 802.1Qbu)) die harten Echtzeitanforderungen von Assets auf der Feldebene erfüllt. Voraussetzung für das Prioritäten-basierte TSN Scheduling ist ein einheitliches Zeitverständnis zwischen den echtzeitfähigen Assets und der Netzwerkinfrastruktur mit den TSN Switches (Bild 5.3).

Protokoll zur Zeitsynchronisation In der Zukunft wird voraussichtlich der TSN Standard IEEE 802.1AS-Rev mit dem dazugehörigen Profil die bevorzugte Lösung zur Zeitsynchronisation werden (Norm IEEE 802.1AS-Rev). Dieser Standard befindet sich Stand heute allerdings noch in der Standardisierungsphase und wird von den prototypischen TSN Switches noch nicht unterstützt. Daher wird das Precision Time Protocol (PTP) mit dem Profil IEEE 802.1AS für die Zeitsynchronisation in der neuen Kommunikationsinfrastruktur verwendet (Norm IEEE 802.1AS). Diese Kombination wird von den prototypischen TSN Switches bereits unterstützt und gewährleistet eine für die Echtzeitkommunikation ausreichende Zeitsynchronisation. Für eine durchgängige Zeitsynchronisation werden die erforderlichen PTP Mechanismen im Folgenden in die echtzeitfähigen Assets integriert.

TSN SDK Für die PTP-basierte Zeitsynchronisation werden insgesamt zwei Softwarebausteine zum TSN SDK der echtzeitfähigen Assets hinzugefügt. Der erste Softwarebaustein realisiert die gesamte PTP-spezifische Kommunikation zur Zeitsynchronisation. Konkret werden die eingehenden PTP Nachrichten des Masters verarbeitet und die eigenen Zeitinformationen zurückgesendet. Basierend auf den ausgetauschten Nachrichten wird eine interne PTP Uhr bestmöglich mit der globalen Netzwerkzeit synchronisiert. Der zweite Softwarebaustein im TSN SDK dient zur Synchronisation der PTP Uhr mit der internen Systemzeit des echtzeitfähigen

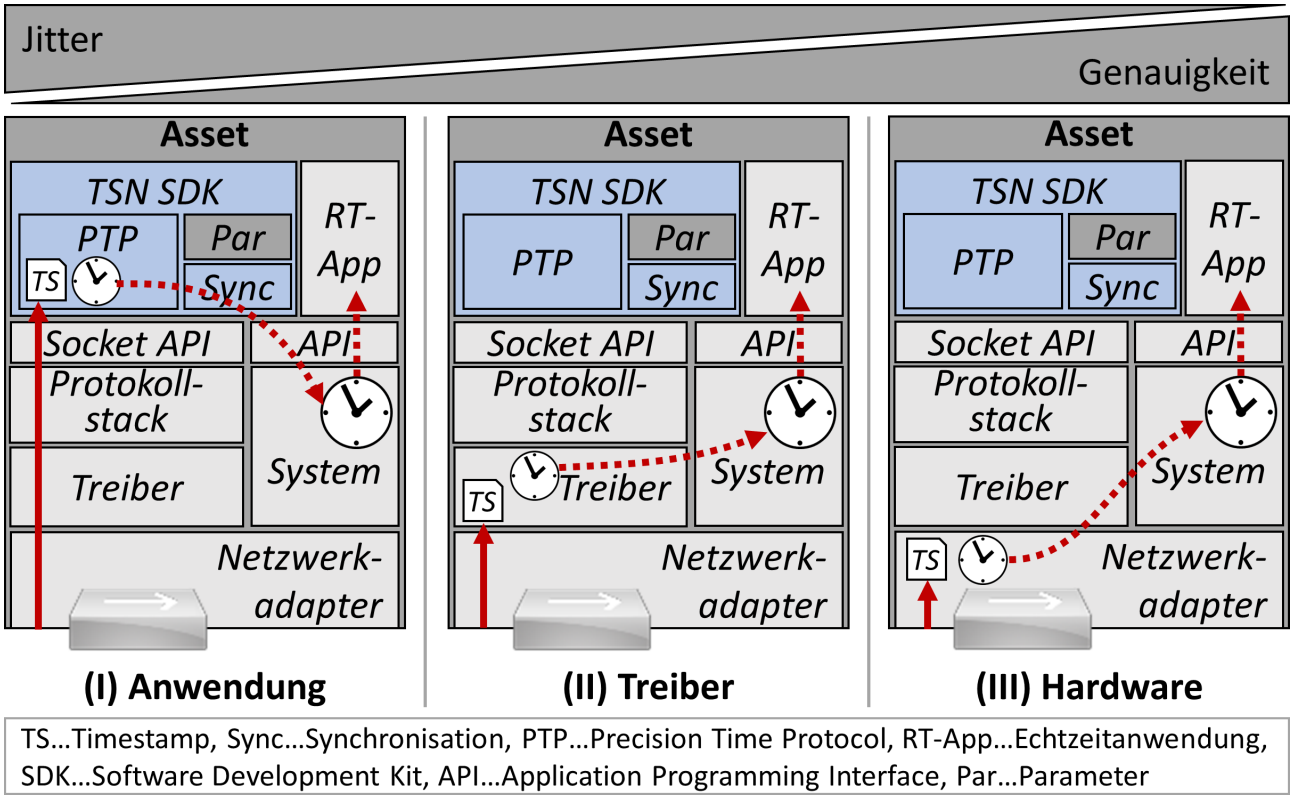


Abbildung 5.4: Drei Varianten zur Integration der Zeitsynchronisation in den echtzeitfähigen Assets.

Assets. Dadurch wird ein einheitliches Zeitverständnis im Betriebssystem und den eigentlichen (Echtzeit-)Anwendungen gewährleistet.

Die PTP-spezifischen Mechanismen in dem TSN SDK umfassen das Verarbeiten eingehender PTP Nachrichten und insbesondere das Setzen des Zeitstempels für jeden Frame. Diese Zeitinformation ist beim PTP Algorithmus maßgeblich für die Berechnung der Übertragungslatenz und die daraus resultierende Genauigkeit der Zeitsynchronisation verantwortlich. Dabei existieren drei verschiedene Varianten zum Setzen der Zeitstempel in den echtzeitfähigen Assets (Bild 5.4):

Anwendung Das Setzen des Zeitstempels in der PTP-Anwendung selbst impliziert den größten Jitter und damit die geringste Genauigkeit bei der Zeitsynchronisation. Der Jitter entsteht insbesondere durch das Durchlaufen des nicht-deterministischen Netzwerkstacks und dem Kontextwechsel zur Anwendung.

Treiber Eine PTP-Unterstützung im Treiber verringert den Jitter im Vergleich zur vorherigen Variante, da der Zeitstempel im Netzwerkstack gesetzt wird und somit der Kontextwechsel zur Anwendung entfällt.

Hardware Das sogenannte Hardware-Timestamping garantiert die bestmögliche Genauigkeit bei der Zeitsynchronisation. Dabei werden eingehende Frames direkt in dem Netzwerkadapter mit einem Zeitstempel versehen und dadurch der Jitter minimiert. Voraussetzung ist allerdings, dass der verwendete Netzwerkadapter in dem echtzeitfähigen Asset das Hardware-Timestamping unterstützt.

Die ersten beiden Varianten implizieren eine größere Abweichung zwischen der globalen Netzwerkzeit und der internen Systemzeit des Assets, was später beim Senden eines TSN Frames kompensiert werden muss. Daher wird die dritte Variante in den echtzeitfähigen Assets verwendet und eine bestmögliche Zeitsynchronisation für die TSN-basierte Echtzeitkommunikation gewährleistet. Folglich müssen allerdings alle echtzeitfähigen Assets in der neuen Kommunikationsinfrastruktur das Hardware-Timestamping im Netzwerkadapter unterstützen.

Zusammengefasst wurden die erforderlichen PTP Mechanismen zur Zeitsynchronisation in das TSN SDK der echtzeitfähigen Assets integriert. Dadurch wird eine Zeitsynchronisation mit der TSN-fähigen Netzwerkinfrastruktur ermöglicht und die Grundlage für das TSN Scheduling geschaffen.

5.3 Dynamische TSN Verbindungen

Dynamische TSN Verbindungen zwischen echtzeitfähigen Assets ermöglichen eine deterministische Kommunikation mit geringer Latenz zum Datenaustausch und zur Synchronisation. Grundlage für diese Verbindungen ist das TSN Scheduling. Dieses Scheduling basiert auf dem sogenannten Time-Aware Shaper und ist im TSN Standard IEEE 802.1Qbv spezifiziert (Norm IEEE 802.1Qbv). Die prototypischen TSN Switches in der Netzwerkinfrastruktur unterstützen diesen Standard bereits. Für eine durchgängige, TSN-konforme Echtzeitkommunikation müssen die echtzeitfähigen Assets ebenfalls das Prioritäten-basierte TSN Scheduling unterstützen (Bild 5.5). Daher werden die erforderlichen Kommunikationsmechanismen im Folgenden in die echtzeitfähigen Assets integriert.

TSN SDK Im vorherigen Abschnitt wurde die Zeitsynchronisation in den echtzeitfähigen Assets ermöglicht und damit die Voraussetzung für das TSN Scheduling geschaffen. Darauf

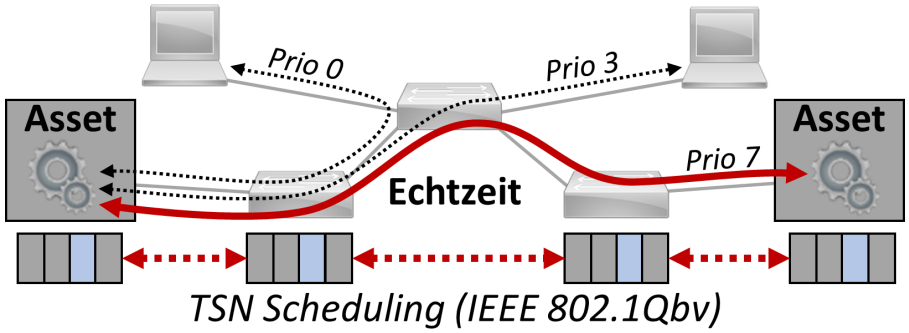


Abbildung 5.5: Das TSN Scheduling in den echtzeitfähigen Assets und der TSN-fähigen Netzwerkinfrastruktur bildet die Grundlage für die dynamischen TSN Verbindungen.

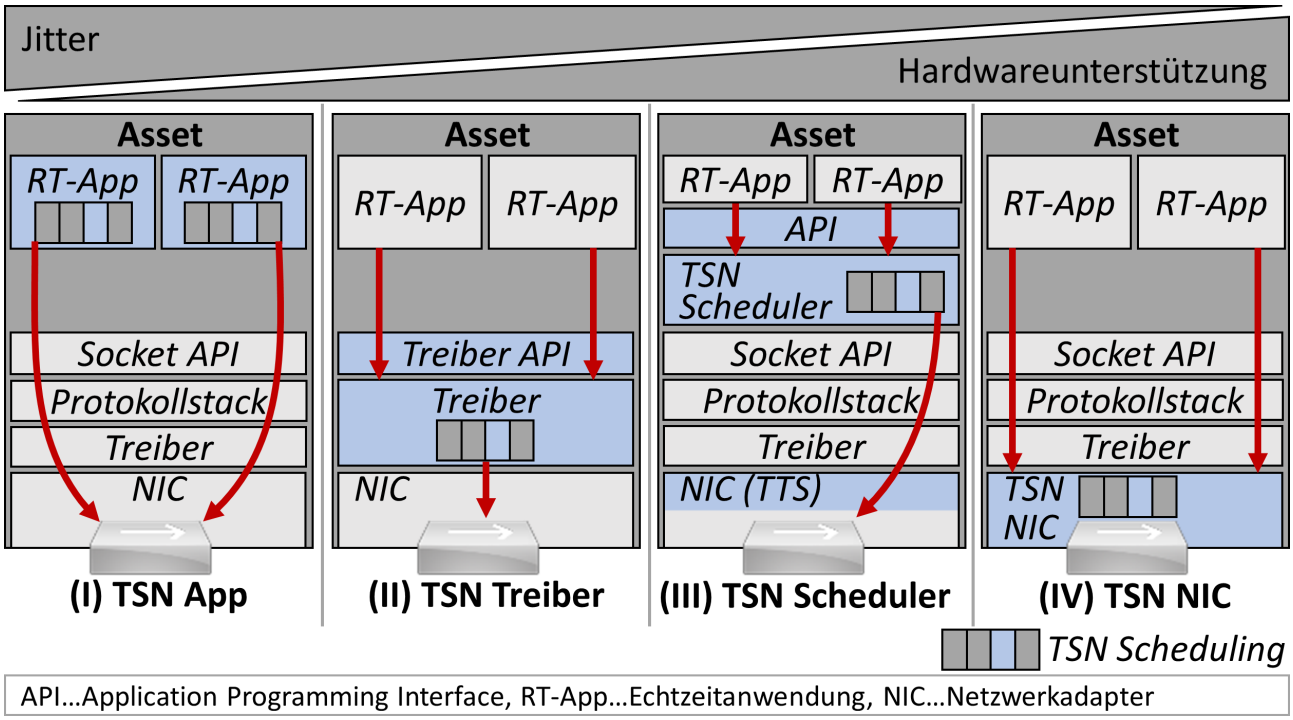


Abbildung 5.6: Vier Varianten zur Integration des TSN Schedulings in die echtzeitfähigen Assets.

aufbauend existieren grundsätzlich vier verschiedene Varianten um das TSN Scheduling in die echtzeitfähigen Assets zu integrieren:

TSN App Die erste Variante beschreibt das TSN Scheduling in der eigentlichen Echtzeitanwendung unabhängig von den anderen Anwendungen und dem Betriebssystem. Dazu müssen die erforderlichen TSN Mechanismen in jede Echtzeitanwendung integriert und parametriert werden. Nachteil dieser Variante ist der sich wiederholende Implementierungsaufwand sowie die Kompatibilität zwischen den verschiedenen Implementierungen. Darüber hinaus verursacht das Durchlaufen des gesamten Netzwerkstack ohne Echtzeitgarantien zusätzlichen Jitter für jeden Frame.

TSN Treiber Bei der zweiten Variante erfolgt das TSN Scheduling im Treiber des jeweiligen Netzwerkadapters. Dadurch wird der nicht-deterministische Netzwerkstack des Betriebssystems umgangen und der Jitter des TSN Frames reduziert. Nachteil ist allerdings die Hardware-spezifische Schnittstelle des Treibers sowie die notwendigen Anpassungen der Echtzeitanwendungen an die Treiber API.

TSN Scheduler Die dritte Variante kombiniert einen Software-basierten TSN Scheduler mit einer Hardwareunterstützung im Netzwerkadapter, dem sogenannten Time-Triggered Send (TTS). Dabei berechnet der TSN Scheduler den genauen Sendezeitpunkt für jeden TSN Frame entsprechend des parametrierten Schedules. Anschließend wird der TSN Frame mit einem Zeitstempel versehen und vom Netzwerkadapter zu dem spezifizierten Zeitpunkt übertragen. Vorteil dieser Variante ist der reduzierte Jitter auf Grund der Hardwareunterstützung im Netzwerkadapter. Die zusätzliche API des TSN Schedulers erfordert allerdings eine Anpassung der Echtzeitanwendungen.

TSN NIC Bei der vierten Variante erfolgt das gesamte TSN Scheduling im Netzwerkadapter, wobei eine entsprechende Hardwareunterstützung vorausgesetzt wird. Die zu sendenden TSN Frames werden automatisch der entsprechenden Prioritätswarteschlange im Netzwerkadapter zugeordnet und in dem designierten Zeitintervall übertragen. Neben dem bestmöglichen Jitter sind auf Grund der gleichbleibenden Socket API auch keine Anpassungen an den Echtzeitanwendungen notwendig.

Für die echtzeitfähigen Assets in der neuen Kommunikationsinfrastruktur wird im Folgenden die dritte Variante verwendet. Diese minimiert den Jitter durch die Hardwareunterstützung im Netzwerkadapter und genügt damit höchsten Echtzeitanforderungen. Gleichzeitig ermöglicht der TSN Scheduler eine einheitliche Integration der TSN Mechanismen unabhängig von der spezifischen Hardware, dem Betriebssystem und den Echtzeitanwendungen. Die vierte Variante

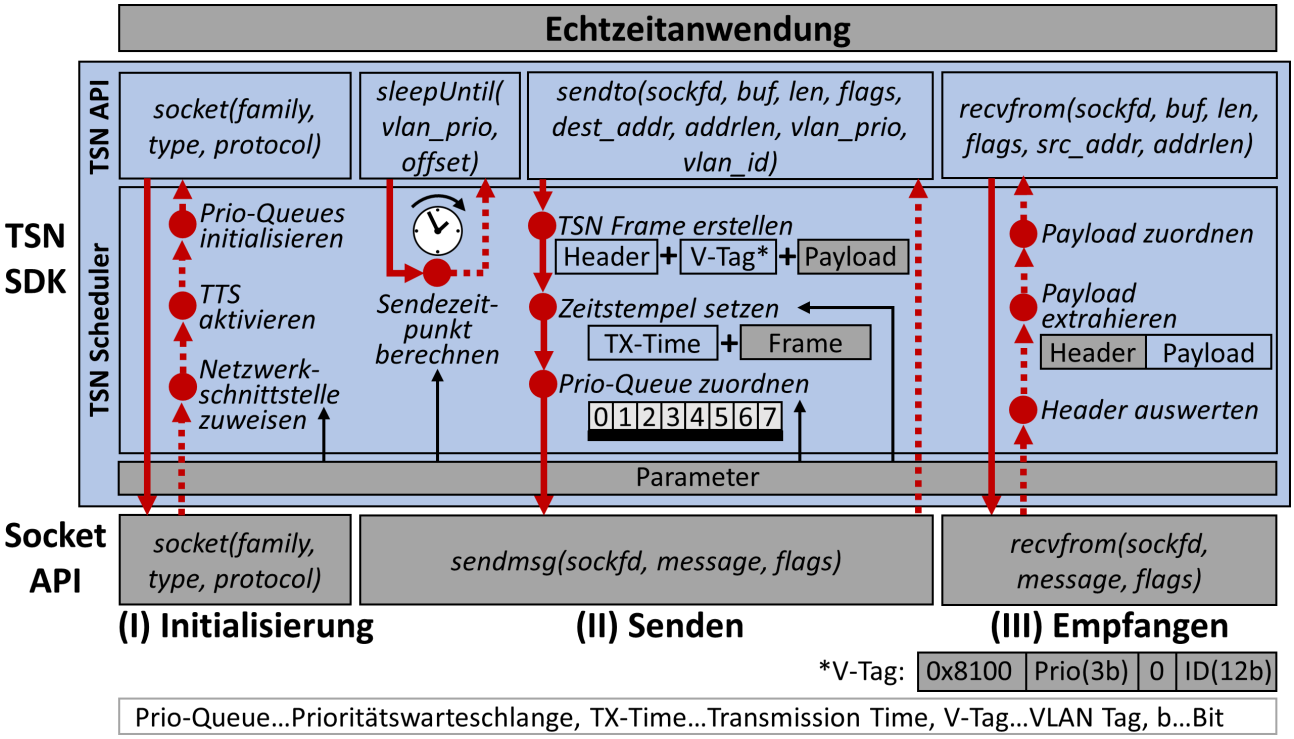


Abbildung 5.7: Die Softwarearchitektur des TSN Schedulers mit der dazugehörigen TSN API für die Echtzeitanwendungen.

bietet neben der Hardwareunterstützung in der Zukunft weitere Vorteile, allerdings existieren Stand heute nur erste Protoptypen für die vollständig TSN-fähigen Netzwerkadapter. Im Gegensatz dazu sind bereits etablierte, marktreife Netzwerkadapter mit TTS-Unterstützung verfügbar, wie z.B. die “Intel I210 NIC” (Intel 2020).

TSN Scheduler Der in Variante drei beschriebene TSN Scheduler wird als zusätzlicher Softwarebaustein in das TSN SDK der echtzeitfähigen Assets integriert. Dieser Scheduler nutzt die gewöhnliche Socket API des Betriebssystems und fügt die TSN-spezifischen Mechanismen hinzu. Die dazugehörige API ermöglicht den einheitlichen Zugriff für alle Echtzeitanwendungen und ist identisch zu der bereits spezifizierten TSN API des TSN SDKs (Bild 5.1). Die resultierende Softwarearchitektur ist im Bild 5.7 skizziert. Im Folgenden wird das Vorgehen bei der Initialisierung sowie beim Senden und Empfangen von Frames im TSN Scheduler detailliert beschrieben.

Initialisierung Zur Initialisierung ruft die Echtzeitanwendung die Funktion “socket” in der TSN API auf. Der Funktionsaufruf wird vom TSN Scheduler entgegengenommen und ein ent-

sprechender Socket mit Hilfe der API des Betriebssystems erstellt. Außerdem wird die TTS-Option für diesen Socket aktiviert und damit das zeitgesteuerte Senden von TSN Frames ermöglicht. Die dazu notwendigen Prioritätswarteschlangen für die verschiedenen VLAN-Prioritäten werden ebenfalls im Netzwerkadapter bzw. dem Betriebssystem initialisiert. Als Rückgabewert der Funktion wird die ID des neuen Sockets an die Echtzeitanwendung für spätere Aufrufe zurückgegeben.

Senden Beim Senden ruft eine Echtzeitanwendung die Funktion "sendto" in der TSN API auf und übergibt den gewünschten Payload. Neben den gewöhnlichen Parametern der Socket API umfasst der Funktionsaufruf auch die VLAN-ID und die Priorität des zu sendenden TSN Frames. Basierend auf diesen Informationen erweitert der TSN Scheduler den Payload um den Ethernet Header und das VLAN-Tag. Außerdem wird der frühestmögliche Sendezeitpunkt des TSN Frames vom Scheduler berechnet und als Zeitstempel zum Frame hinzugefügt. Der berechnete Zeitpunkt basiert auf dem TSN Schedule und entspricht dem Startzeitpunkt des nächstmöglichen Zeitslots für die spezifizierte Priorität. Anschließend wird der zu sendende TSN Frame der entsprechenden Prioritätswarteschlange im Netzwerkadapter zugeordnet und über die Socket API übergeben.

Empfangen Im Gegensatz zum Senden eines TSN Frames sind beim Empfangen keine TSN-spezifischen Mechanismen notwendig. Daher wird der Aufruf "recvfrom" unverändert an die Socket API des Betriebssystems weitergeleitet. Der Rückgabewert entspricht einem empfangenden TSN Frame, wobei der Payload extrahiert und an die Echtzeitanwendung übergeben wird.

Neben den drei wesentlichen Funktionen in der TSN API wird eine weitere Funktion "sleepUntil" zur Verfügung gestellt. Diese Funktion ermöglicht den Echtzeitanwendungen die Synchronisation mit der globalen Netzwerkzeit und dem zyklischen TSN Schedule. Dabei wird eine Anwendung bei einem Funktionsaufruf blockiert bis der nächstmögliche Zeitslot im TSN Schedule beginnt. Dieser Zeitslot basiert auf der Priorität, die als erster Parameter übergeben wird. Damit der Payload der Anwendung rechtzeitig an das TSN SDK übergeben wird, muss zusätzlich die Worst-Case-Ausführungszeit berücksichtigt werden. Diese Ausführungszeit wird als zweiter Parameter übergeben und von dem designierten Sendezeitpunkt abgezogen. Der resultiere Zeitpunkt bestimmt das Ende des blockierenden Funktionsaufrufs ("sleepUntil"). Dadurch wird das rechtzeitige Senden eines TSN Frames gewährleistet und eine Synchronisation der Echtzeitanwendung mit dem konfigurierten TSN Schedule ermöglicht.

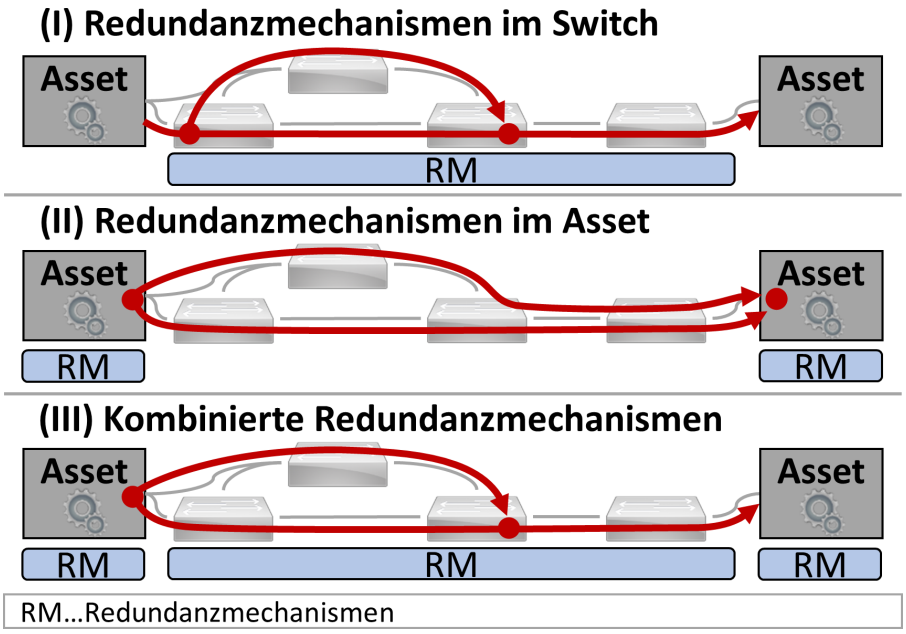


Abbildung 5.8: Drei Varianten zur Etablierung redundanter TSN Verbindungen zwischen echtzeitfähigen Assets.

In diesem Abschnitt wurde die Grundlage für die dynamischen TSN Verbindungen in der neuen Kommunikationsinfrastruktur geschaffen. Dazu wurde das TSN Scheduling in die echtzeitfähigen Assets integriert und das TSN SDK um einen Softwarebaustein erweitert. Dieser Softwarebaustein beinhaltet einen TSN Scheduler, der das standardkonforme Senden und Empfangen von TSN Frames realisiert. Dadurch wird in Kombination mit der Hardwareunterstützung im Netzwerkadapter eine echtzeitfähige, TSN-konforme Kommunikation gewährleistet.

5.4 Redundante TSN Verbindungen

Im vorherigen Abschnitt wurde das TSN SDK der echtzeitfähigen Assets erweitert und die Grundlage für dynamische TSN Verbindungen zwischen zwei echtzeitfähigen Assets geschaffen. Trotz regelmäßiger Wartung können sowohl die Assets als auch die TSN Verbindungen von unvorhersehbaren Hardware- oder Softwarefehlern betroffen sein. Hardwarefehler umfassen zum Beispiel Kabelbrüche oder Fehler in den Netzwerkadaptern. Softwarefehler betreffen beispielsweise das Betriebssystem der echtzeitfähigen Assets oder die Firmware der TSN Switches. Echtzeitfähige Assets und insbesondere sicherheitskritische Systeme erfordern daher eine ausfallsichere Echtzeitkommunikation, d.h. redundante TSN Verbindungen.

Redundanz-Varianten Die erforderlichen Redundanzmechanismen in den echtzeitfähigen Assets hängen maßgeblich von der Integration der Redundanz in die neue Kommunikationsinfrastruktur ab. Dabei werden grundsätzlich drei verschiedene Varianten unterschieden (Bild 5.8):

TSN Switches Die Integration der Redundanzmechanismen in die TSN Switches kompensiert Fehler in der Netzwerkinfrastruktur und ist unabhängig von den echtzeitfähigen Assets. Dabei dupliziert ein TSN Switch den gesendeten TSN Frame und leitet ihn über möglichst disjunkte Netzwerkpfade weiter. Sobald der Netzwerkpfad zum Empfänger eindeutig wird, eliminiert ein TSN Switch den redundanten Frame wieder. Nachteil dieser Variante ist die fehlende Absicherung auf dem letzten Netzwerksegment zu den echtzeitfähigen Assets und den dazugehörigen Netzwerkadaptern.

Echtzeitfähige Assets Die Integration der Redundanzmechanismen in den echtzeitfähigen Assets ermöglicht eine Ende-zu-Ende Redundanz ohne spezifische Mechanismen in den TSN Switches. Sowohl das Duplizieren von TSN Frames als auch das Eliminieren von redundanten Frames ist ausschließlich in den echtzeitfähigen Assets möglich. Nachteil ist die ineffizientere Nutzung der verfügbaren Bandbreite, da redundante TSN Frames auch über eindeutige Netzwerkpfade doppelt übertragen werden.

Kombinierte Variante Die Integration der Redundanzmechanismen sowohl in den TSN Switches als auch den echtzeitfähigen Assets vereint die beiden vorherigen Varianten. Zum einen wird eine bestmögliche Redundanz zwischen zwei echtzeitfähigen Assets gewährleistet, zum anderen wird eine vollständige Kontrolle über den Netzwerkpfad ermöglicht. Nachteil dieser Variante ist der doppelte Implementierungsaufwand und die komplexe Konfiguration der redundanten TSN Verbindungen im Netzwerk.

Im Folgenden wird die zweite Variante für die redundanten TSN Verbindungen verwendet und die echtzeitfähigen Assets erweitert. Dadurch können auch Komponentenhersteller unabhängig vom Fortschritt der Switchhersteller redundante TSN Verbindungen zwischen ihren echtzeitfähigen Assets etablieren. Gleichzeitig wird der höhere Implementierungsaufwand und die komplexere Konfiguration im Netzwerk gegenüber der dritten Variante vermieden. Die erforderlichen Redundanzmechanismen werden im Folgenden in die echtzeitfähigen Assets integriert.

TSN SDK Die Kommunikationsmechanismen für die redundanten TSN Verbindungen werden in das TSN SDK der echtzeitfähigen Assets integriert. Diese Mechanismen spezifizieren die Frame Duplizierung, die Sequenz De-/Codierung, die Frame Identifizierung und die Frame

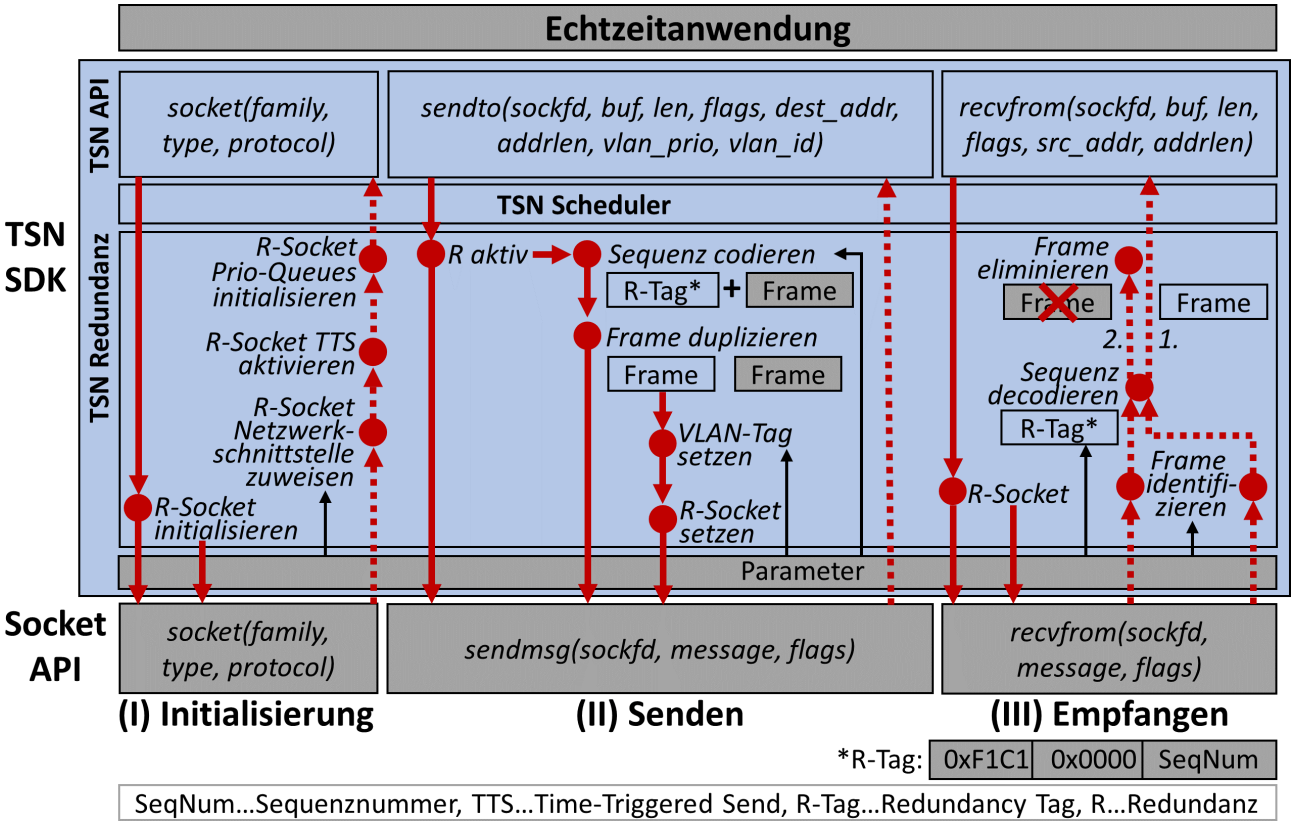


Abbildung 5.9: Die Integration der vier erforderlichen Redundanzmechanismen in das TSN SDK der echtzeitfähigen Assets.

Eliminierung basierend auf dem TSN Standard IEEE 802.1CB (Norm IEEE 802.1CB). Der resultierende Softwarebaustein im TSN SDK ist im Bild 5.9 dargestellt. Das genaue Vorgehen bei der Initialisierung sowie dem Senden bzw. Empfangen von redundanten TSN Frames wird im Folgenden erläutert.

Initialisierung Bei der Initialisierung wird neben dem primären TSN Socket ein weiterer Socket für die redundante TSN Verbindung erstellt. Dieser zweite Socket ist transparent für die eigentliche Echtzeitanwendung und wird von dem neuen Softwarebaustein verwaltet. Analog zum primären Socket wird auch die TTS-Option für den redundanten Socket aktiviert und die Prioritätswarteschlangen initialisiert.

Senden Beim Senden eines TSN Frames wird der Payload von der Echtzeitanwendung über die TSN API zunächst an den TSN Scheduler übergeben. Nachdem der primäre TSN Frame erstellt wurde und das TSN Scheduling erfolgt ist, durchläuft der TSN Frame den Redundanz-spezifischen Softwarebaustein. Dabei wird zunächst das sogenannte "Redundancy-Tag" zwischen dem Ethernet-Header und dem Payload hinzugefügt. Dieses Tag beinhaltet eine fortlaufende Sequenznummer für jeden TSN Frame und dient zur Decodierung der Redundanz auf der Empfängerseite. Der erweiterte TSN Frame wird anschließend dupliziert und zum eigentlichen Senden an die beiden Netzwerkadapter weitergeleitet. Dabei wird der primäre TSN Frame über den gegebenen Socket gesendet, während der redundante TSN Frame über den zusätzlichen redundanten Socket übertragen wird. Außerdem wird, sofern notwendig, die VLAN-ID der redundanten TSN Frames für den disjunkten Netzwerkpfad angepasst.

Empfangen Ein echtzeitfähiges Asset empfängt sowohl den primären als auch den redundanten TSN Frame über möglicherweise verschiedene Netzwerkadapter. Unabhängig davon werden beide TSN Frames direkt an den Redundanz-spezifischen Softwarebaustein im TSN SDK weitergeleitet. Da ein echtzeitfähiges Asset über mehrere redundante TSN Verbindungen verfügen kann, wird der eingehende TSN Frame zunächst identifiziert und einer Verbindung, d.h. einem Sender, zugeordnet. Die technischen Merkmale zur Frame Identifizierung sind in der Redundanz Parametrierung spezifiziert. Nach der Frame Identifizierung wird die Sequenznummer aus dem "Redundancy Tag" ausgelesen. Basierend auf dieser Sequenznummer werden die eingehenden TSN Frames sortiert und redundante Frames auf Grund der wiederkehrenden Sequenznummer erkannt. Ein redundanter Frame wird schlussendlich verworfen und nur der zuerst eintreffende

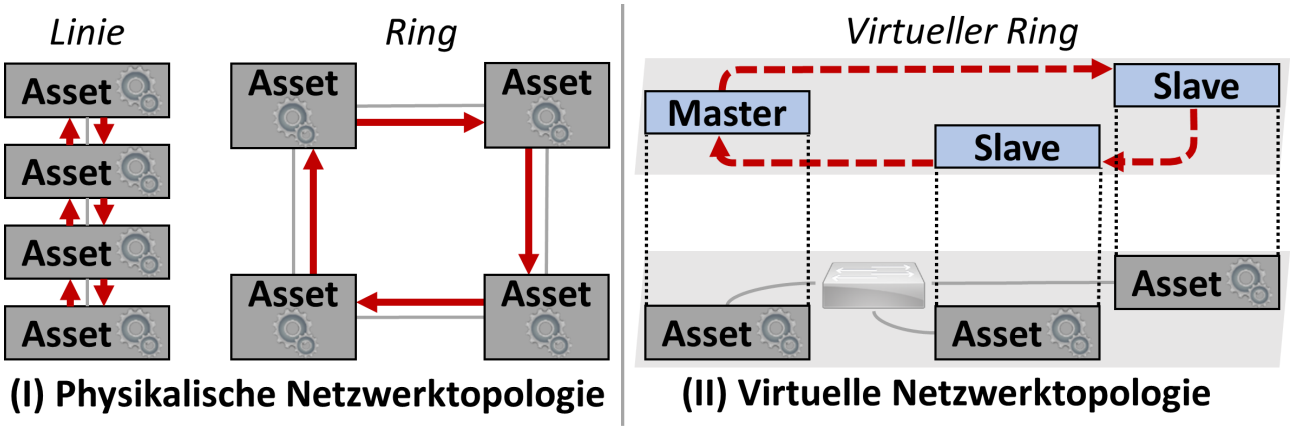


Abbildung 5.10: Virtuelle Netzwerktopologie verbinden mehrere echtzeitfähige Assets und ermöglichen gleichzeitig die Abstraktion von der physikalischen Netzwerkinfrastruktur.

Frame an den TSN Scheduler weitergeleitet. Dadurch ist die redundante Übertragung für die darüberliegenden Softwarebausteine und insbesondere die Echtzeitanwendung transparent.

Zur Etablierung redundanter TSN Verbindungen wurden die vier erforderlichen Redundanzmechanismen in die echtzeitfähigen Assets integriert. Dabei wurde ein weiterer Redundanz-spezifischer Softwarebaustein zum TSN SDK hinzugefügt. Die Netzwerkkonfiguration der redundanten TSN Verbindungen und insbesondere die Konfiguration der disjunkten Netzwerkpfade wird im Kapitel 7 genauer betrachtet.

5.5 Virtuelle Netzwerktopologien

In den vorherigen Abschnitten wurden dynamische TSN Verbindungen zwischen zwei echtzeitfähigen Assets etabliert und die Ausfallsicherheit durch redundante Verbindungen ermöglicht. Im Folgenden wird die TSN-basierte Echtzeitkommunikation zwischen mehr als zwei Assets in der neuen Kommunikationsinfrastruktur betrachtet.

Virtuelle Netzwerktopologien Zur Vernetzung von mehreren echtzeitfähigen Assets werden sogenannte virtuelle Netzwerktopologien vorgestellt. Diese Topologien abstrahieren von der physikalischen Netzwerkinfrastruktur und realisieren eine logische Linien- oder Ring-Topologie zwischen den Assets (Bild 5.10). Dadurch wird eine beliebige physikalische Netzwerkinfrastruktur mit den TSN Switches ermöglicht, d.h. insbesondere die Vernetzung in einer Mesh-

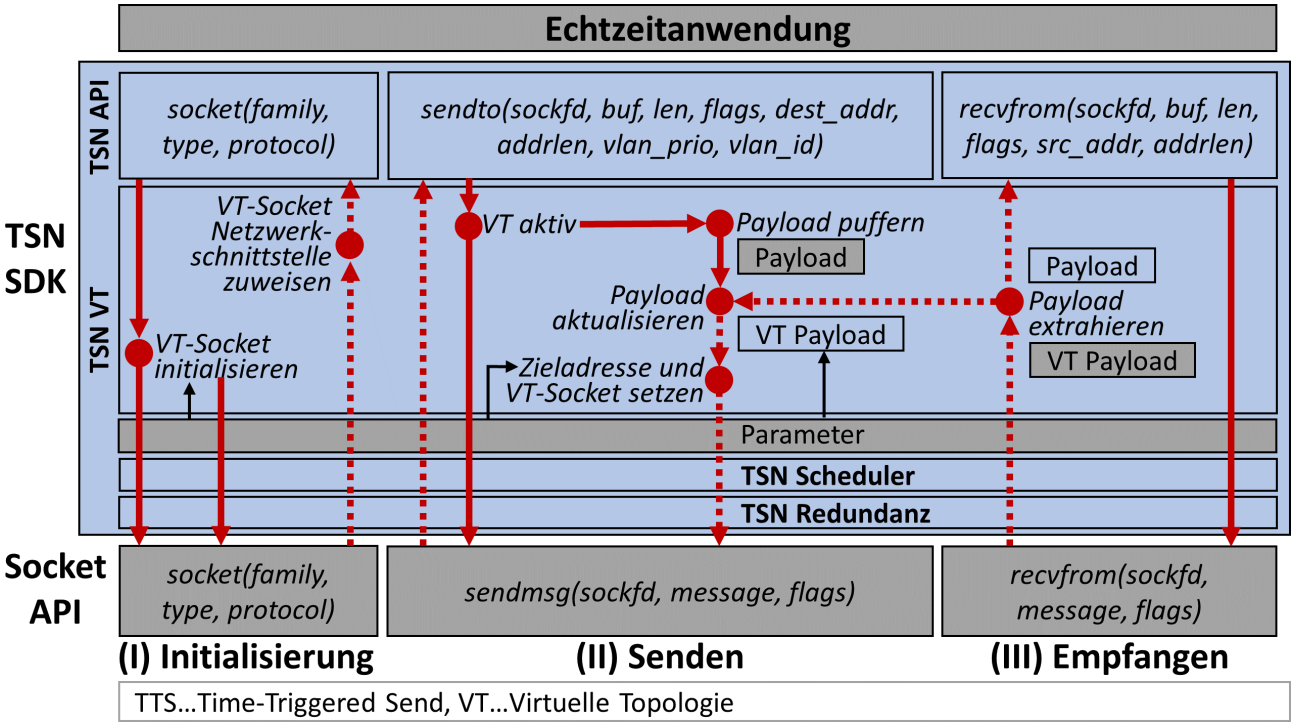


Abbildung 5.11: Der zusätzliche, VT-spezifische Softwarebaustein in dem TSN SDK der echtzeitfähigen Assets.

Topologie. Gleichzeitig werden die heutzutage weit verbreiteten Linien- und Ring-Topologien in den OT-Netzwerken auch in der neuen Kommunikationsinfrastruktur unterstützt.

Eine virtuelle Netzwerktopologie definiert die logische Reihenfolge der echtzeitfähigen Assets im Netzwerk. Daraus lässt sich ein entsprechender Pfad in der physikalischen Netzwerkinfrastruktur ableiten. Basierend auf diesem Netzwerkpfad werden jeweils zwei aufeinanderfolgende echtzeitfähige Assets mit Hilfe einer TSN Verbindung verbunden. Daher kann eine virtuelle Netzwerktopologie als Verkettung von einzelnen dynamischen TSN Verbindungen (Abschnitt 5.3) interpretiert werden. Die Konfiguration der einzelnen TSN Verbindungen ist dabei aufeinander abgestimmt und gewährleistet die zyklische Übertragung eines echtzeitkritischen Frames.

TSN SDK Für die Teilnahme in einer virtuellen Netzwerktopologie benötigen die echtzeitfähigen Assets neben den TSN-spezifischen Mechanismen weitere Kommunikationsmechanismen. Diese Mechanismen spezifizieren die Initialisierung sowie das Senden und Empfangen von TSN Frames in einer virtuellen Netzwerktopologie. Der resultierende Softwarebaustein ist im Bild 5.11 skizziert und wird ebenfalls in das TSN SDK der echtzeitfähigen Assets integriert. Da die virtuellen Netzwerktopologien auf einzelnen TSN Verbindungen basieren, verwendet der

neue Softwarebaustein direkt den vorgestellten TSN Scheduler. Darüber hinaus bleibt die TSN API als Schnittstelle zur eigentlichen Echtzeitanwendung unverändert. Im Folgenden werden die zusätzlichen Kommunikationsmechanismen für die virtuellen Netzwerktopologien im Detail erläutert.

Initialisierung Zur Teilnahme in einer virtuellen Netzwerktopologie werden zwei Sockets benötigt, ein Socket zum Empfangen und ein Socket zum Weiterleiten eines TSN Frames. Der zusätzliche Socket zum Weiterleiten wird im Folgenden als "VT-Socket" bezeichnet. Dieser "VT-Socket" wird von dem neuen Softwarebaustein verwaltet, wobei die Initialisierung identisch zum primären Socket ist.

Senden und Empfangen Das Senden und Empfangen in einer virtuellen Netzwerktopologie ist zusammengefasst, da eingehende TSN Frames verarbeitet und anschließend direkt weitergeleitet werden. Konkret wird der Payload eines eingehenden TSN Frames extrahiert und an die Echtzeitanwendung übergeben. Dies geschieht analog zu der Socket API des Betriebssystems über die Funktion "recvfrom" in der TSN API. Gleichzeit wird der empfangene Payload aktualisiert, d.h. mit den aktuellen Daten der Echtzeitanwendung überschrieben. Der neue Payload wird dabei bis zum nächsten zyklischen TSN Frame in der virtuellen Netzwerktopologie gepuffert und schließlich in den TSN Frame kopiert. Das Senden des aktualisierten Payloads erfolgt über den zusätzlichen "VT-Socket" und wird von dem TSN Scheduler übernommen. Dabei wird der TSN Frame an das darauffolgende echtzeitfähige Asset in der virtuellen Netzwerktopologie gesendet.

Zusammengefasst ermöglicht der vorgestellte Softwarebaustein echtzeitfähigen Assets die Teilnahme in einer virtuellen Netzwerktopologie und die Interaktion mit mehreren echtzeitfähigen Assets. Dazu werden eingehende TSN Frames transparent für die Echtzeitanwendung verarbeitet, gegebenenfalls aktualisiert und anschließend weitergeleitet. Die resultierende virtuelle Netzwerktopologie etabliert eine logische Linien- oder Ring-Topologie zwischen mehreren Assets unabhängig von der darunterliegenden physikalischen Netzwerkinfrastruktur. Die dazugehörige Netzwerkkonfiguration der erforderlichen TSN Verbindungen wird im Kapitel 7 betrachtet.

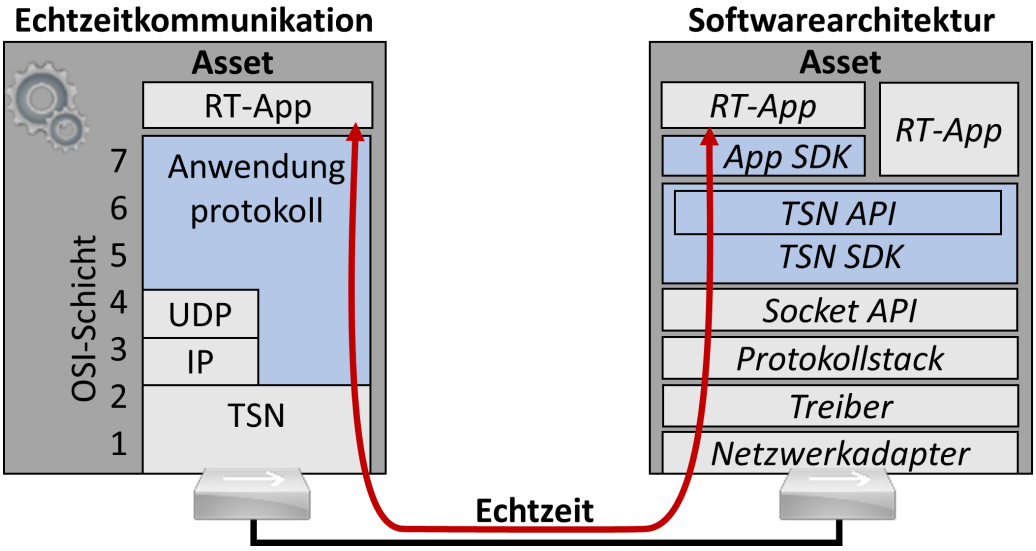


Abbildung 5.12: Die Kombination eines echtzeitfähigen Anwendungsprotokolls mit einer TSN Verbindung ermöglicht die durchgängige Echtzeitkommunikation.

5.6 Anwendungsprotokolle

Während TSN die einheitliche Kommunikationsbasis für die neue Echtzeitkommunikation spezifiziert (OSI-Schicht 1-2), nutzen echtzeitfähige Assets Anwendungsprotokolle zur Übertragung von anwendungsspezifischen Daten (OSI-Schicht 5-7). Die dazwischenliegenden OSI-Schichten 3 und 4 werden bei der Echtzeitkommunikation aus Performancegründen häufig ausgelassen. Stand heute haben sich verschiedene echtzeitfähige Anwendungsprotokollen etabliert, wie z.B. Sercos, EtherCAT, Profinet, OPC UA PubSub oder DDS. Auf Grund der Vielzahl existierender Protokolle ist es unwahrscheinlich, dass sich ein bestimmtes Anwendungsprotokoll in der Zukunft durchsetzen wird. Stattdessen ist die Verwendung von einem beliebigen, für den individuellen Anwendungsfall optimalen, Anwendungsprotokoll erstrebenswert. Daher wird basierend auf dem vorgestellten TSN SDK die Integration von beliebigen Anwendungsprotokollen in die echtzeitfähigen Assets erläutert. Dadurch wird eine durchgängige Echtzeitkommunikation zwischen zwei echtzeitfähigen Assets und ihren Echtzeitanwendungen ermöglicht (Bild 5.12).

App SDK Basierend auf dem vorgestellten TSN SDK erfolgt die Integration eines Anwendungsprotokolls durch ein weiteres SDK (Bild 5.12). Dieses SDK wird im Folgenden als App SDK bezeichnet und beinhaltet die spezifischen Kommunikationsmechanismen für das jeweilige Protokoll. Die dazugehörige App API stellt eine protokollspezifische Schnittstelle für die Echtzeitanwendungen zur Verfügung. Für die etablierten echtzeitfähigen Anwendungsprotokolle

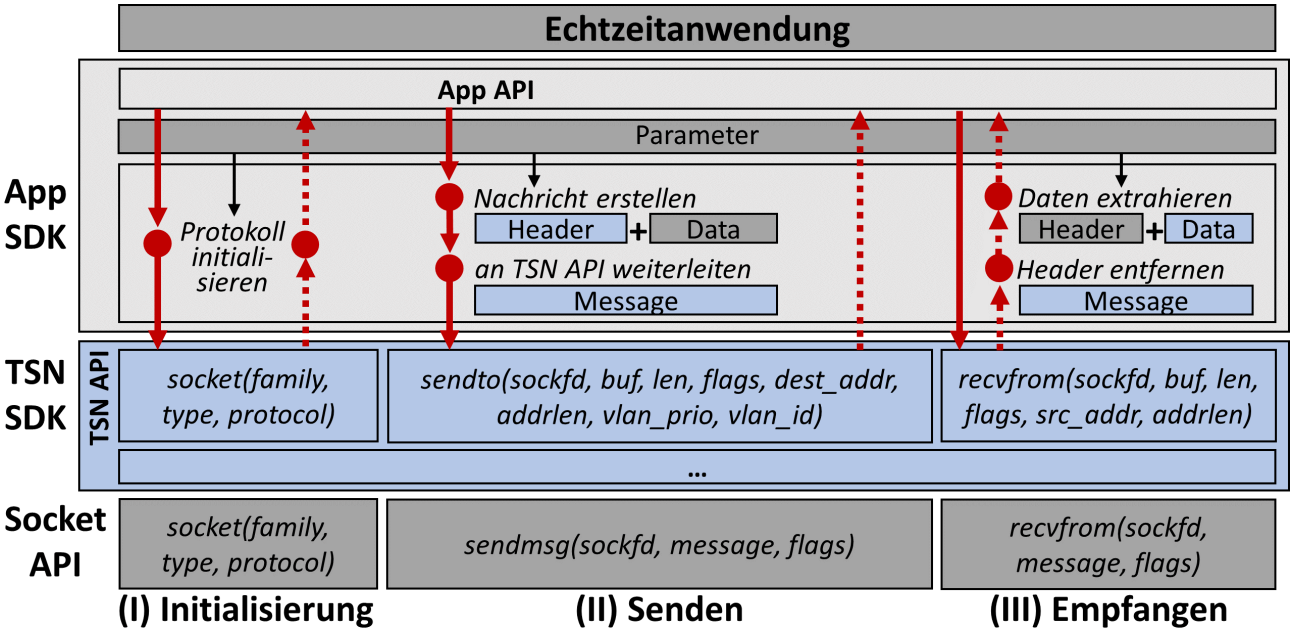


Abbildung 5.13: Die Integration eines App SDKs in ein echtzeitfähige Assets mit TSN SDK.

existieren bereits App SDKs, die einmalig an die neue TSN API angepasst werden müssen. Dazu werden alle Funktionsaufrufe an die gewöhnliche Socket API des Betriebssystems an das TSN SDK umgeleitet und die Parameter entsprechend angepasst. Dadurch wird eine TSN-basierte Echtzeitkommunikation mit einem Anwendungsprotokoll ermöglicht. Das genaue Vorgehen bei der Initialisierung des Anwendungsprotokolls sowie beim Senden und Empfangen einer echtzeitkritischen Nachricht wird im Folgenden erläutert.

Initialisierung Bei der Initialisierung wird die Anwendungsprotokoll-spezifische Logik im App SDK ausgeführt und die Voraussetzung zum Senden und Empfangen von Nachrichten geschaffen. Dabei werden auch die erforderlichen Sockets für die TSN-basierte Echtzeitkommunikation initialisiert. Diese Sockets werden analog zu der bisherigen Socket API des Betriebssystems über die neue TSN API erstellt.

Senden Das Senden einer Nachricht wird von der Echtzeitanwendung initiiert und die gewünschten Daten an das App SDK übergeben. Das App SDK realisiert anschließend die Anwendungsprotokoll-spezifischen Mechanismen und erstellt die Nachricht inklusive dem Anwendungsprotokollheader. Anschließend wird die resultierende Nachricht über die TSN API an das TSN SDK übergeben und gesendet. Dabei wird statt dem bisherigen Aufruf an die Socket API die neue Funktion “sendto” mit zwei zusätzlichen Parametern in der TSN API verwendet.

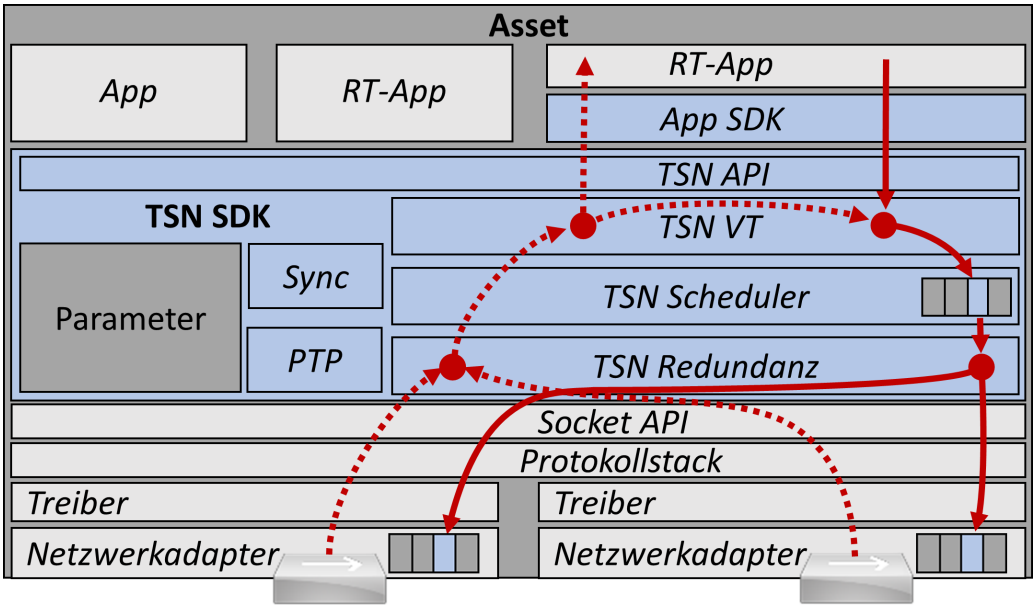


Abbildung 5.14: Die vorgestellten Softwarebausteine für die Echtzeitkommunikation in einem echtzeitfähigen Assets.

Empfangen Das Empfangen erfolgt analog zum Senden einer Nachricht in umgekehrter Reihenfolge. Ein eingehender TSN Frame wird von dem TSN SDK verarbeitet und die Nachricht an das App SDK übergeben. Anschließend verarbeitet das App SDK die eingehende Nachricht und entfernt den Anwendungsprotokollheader. Schließlich werden die extrahierten Daten an die eigentliche Echtzeitanwendung übergeben.

Basierend auf dem vorgestellten TSN SDK wurde in diesem Abschnitt die Integration von beliebigen Anwendungsprotokollen in die echtzeitfähigen Assets erläutert. Dazu wurde ein App SDK mit dazugehöriger API integriert und an die neue TSN API angepasst. Dadurch wird eine TSN-konforme Echtzeitkommunikation mit Anwendungsprotokoll zwischen verschiedenen Assets im Produktionssystem ermöglicht.

5.7 Zusammenfassung

In diesem Kapitel wurde der erste wesentliche Aspekt einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur betrachtet und eine TSN-basierte Echtzeitkommunikation vorgestellt. Dabei wurden die erforderlichen Kommunikationsmechanismen im Detail erläutert und in die echtzeitfähigen Assets integriert. In Kombination mit der TSN-fähigen Netzwerkinfrastruktur wurde dadurch eine einheitliche und standardisierte Kommunikationsinfrastruktur ermöglicht.

Integration in echtzeitfähige Assets Die erforderlichen TSN-spezifischen Kommunikationsmechanismen für die echtzeitfähigen Assets wurden in einem TSN SDK zusammengefasst. Dieses SDK besteht aus verschiedenen Softwarebausteinen basierend auf den Anforderungen an die Echtzeitkommunikation (Bild 5.14). Die PTP Implementierung zur Zeitsynchronisation ist die Voraussetzung für den TSN Scheduler. Dieser Scheduler realisiert das Senden bzw. Empfangen von TSN Frames entsprechend dem Standard IEEE 802.1Qbv und bildet die Grundlage für die dynamischen TSN Verbindungen im Netzwerk (Norm IEEE 802.1Qbv). Darauf aufbauend wurden die erforderlichen Kommunikationsmechanismen für die redundanten TSN Verbindungen sowie die virtuellen Netzwerktopologien integriert. Schließlich wurde das neue TSN SDK mit einem beliebigen App SDK kombiniert und eine durchgängige Echtzeitkommunikation inklusive Anwendungsprotokoll ermöglicht.

Parametrierung der Echtzeitkommunikation Die verschiedenen Teilaspekte der Echtzeitkommunikation und die dazugehörigen Softwarebausteine im TSN SDK benötigen allerdings eine Parametrierung. Diese Parametrierung umfasst die Konfiguration des Gesamtsystems (z.B. Zeitsynchronisation im Netzwerk) und die Anpassung an das konkrete echtzeitfähige Asset (z.B. Netzwerkschnittstellen). Außerdem werden einzelne Parameter erst zu Laufzeit festgelegt, wie z.B. der Schedule für die dynamischen TSN Verbindungen. Daher ist die einheitliche Parametrierung der Echtzeitkommunikation in den echtzeitfähigen Assets der zweite wesentliche Aspekt der neuen Kommunikationsinfrastruktur und wird im folgenden Kapitel detailliert betrachtet.

Kapitel 6

Verwaltungsschale zur Parametrierung

Im Kapitel 5 wurde die TSN-basierte Echtzeitkommunikation für die neue wandelbare, echtzeitfähige Kommunikationsinfrastruktur erarbeitet und die echtzeitfähigen Assets erweitert. Dabei wurde insbesondere ein TSN SDK mit den erforderlichen Kommunikationsmechanismen in die Assets integriert. Darauf aufbauend wird im folgenden Kapitel die Parametrierung des TSN SDKs bzw. der Echtzeitkommunikation allgemein vorgestellt. Die Parametrierung ermöglicht die einheitliche Konfiguration des Gesamtsystems, die Anpassung an das konkrete echtzeitfähige Asset sowie dynamische Änderungen zur Laufzeit. Grundlage für die Parametrierung der echtzeitfähigen Assets ist eine einheitliche Komponentenbeschreibung. Basierend auf dieser Beschreibung wird die Parametrierung der einzelnen Teilaspekte der Echtzeitkommunikation vorgestellt.

6.1 Einheitliche Komponentenbeschreibung

Eine Komponentenbeschreibung ermöglicht allgemein die Darstellung von Eigenschaften und Fähigkeiten eines Assets im Produktionssystem. Dabei werden insbesondere die (Echtzeit-)Anwendungen beschrieben und entsprechende Konfigurationsparameter bzw. Statusinformationen spezifiziert. Neben den individuellen Fähigkeiten des Assets werden in der vorgestellten Lösung auch die Kommunikationsfähigkeiten beschrieben. Dadurch wird die Parametrierung der TSN-basierten Echtzeitkommunikation und dem dazugehörigen TSN SDK in den echtzeitfähigen Assets ermöglicht.

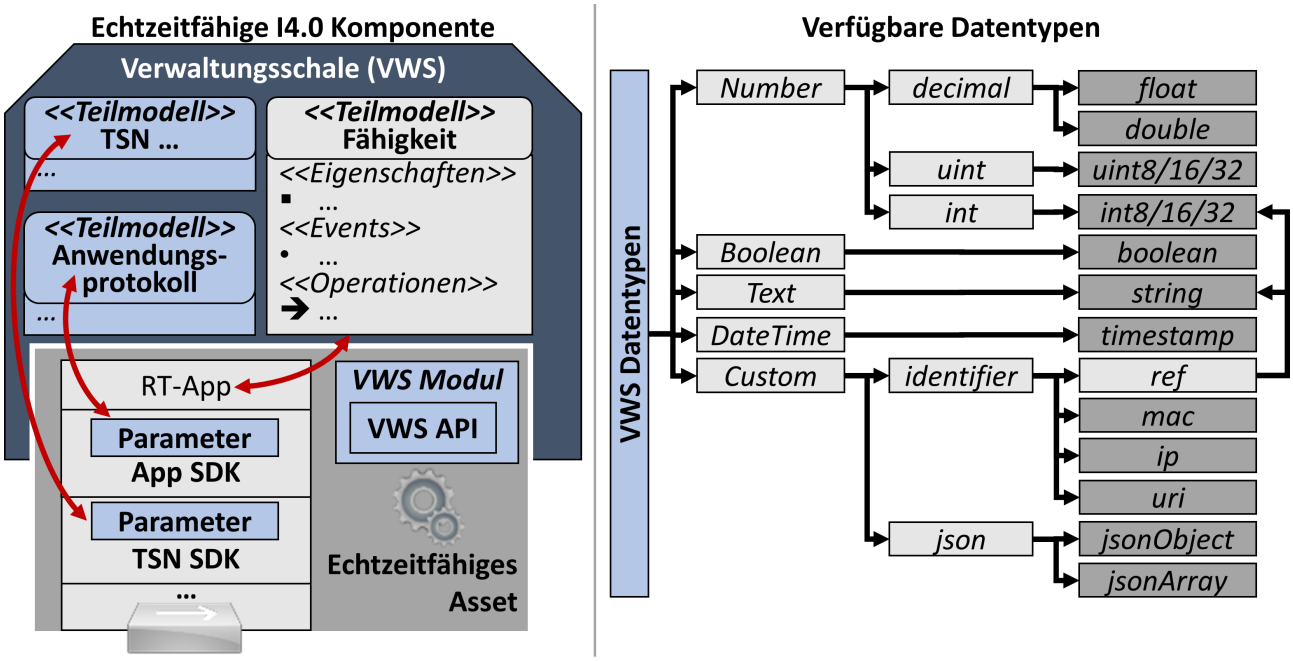


Abbildung 6.1: Eine echtzeitfähige I4.0 Komponente mit Verwaltungsschale und die verfügbaren Datentypen in der Verwaltungsschale.

I4.0 Komponenten Zur einheitlichen Parametrierung der Assets im Produktionssystem ist eine herstellerunabhängige und standardisierte Komponentenbeschreibung essentiell. Im Kontext der vierten industriellen Revolution wurde daher das Konzept der I4.0 Komponente mit Verwaltungsschale entwickelt. Dabei werden die einzelnen Fähigkeiten der Assets in Form von Teilmodellen in der Verwaltungsschale beschrieben. Bisher wurden I4.0 Komponenten allerdings hauptsächlich zur Beschreibung von gewöhnlichen Assets ohne Echtzeitanforderungen verwendet. Daher wird das Konzept der I4.0 Komponente im Folgenden für die neue Kommunikationsinfrastruktur erweitert und die Beschreibung von echtzeitfähigen Assets ermöglicht.

Echtzeitfähige I4.0 Komponenten Zur Parametrierung von echtzeitfähigen Assets und der dazugehörigen Echtzeitkommunikation wird das Konzept der echtzeitfähigen I4.0 Komponente vorgestellt (Bild 6.1). Dieses Konzept beschreibt ein echtzeitfähiges Asset mit einer Verwaltungsschale. Die Verwaltungsschale beinhaltet neben den Asset-spezifischen Teilmodellen zusätzliche Teilmodelle für die TSN-basierte Echtzeitkommunikation. Dadurch wird die einheitliche Parametrierung von echtzeitfähigen Assets und insbesondere dem TSN SDK ermöglicht. Gleichzeitig bleibt durch die Definition zusätzlicher Teilmodelle die Kompatibilität zu den gewöhnlichen I4.0 Komponenten im Produktionssystem erhalten.

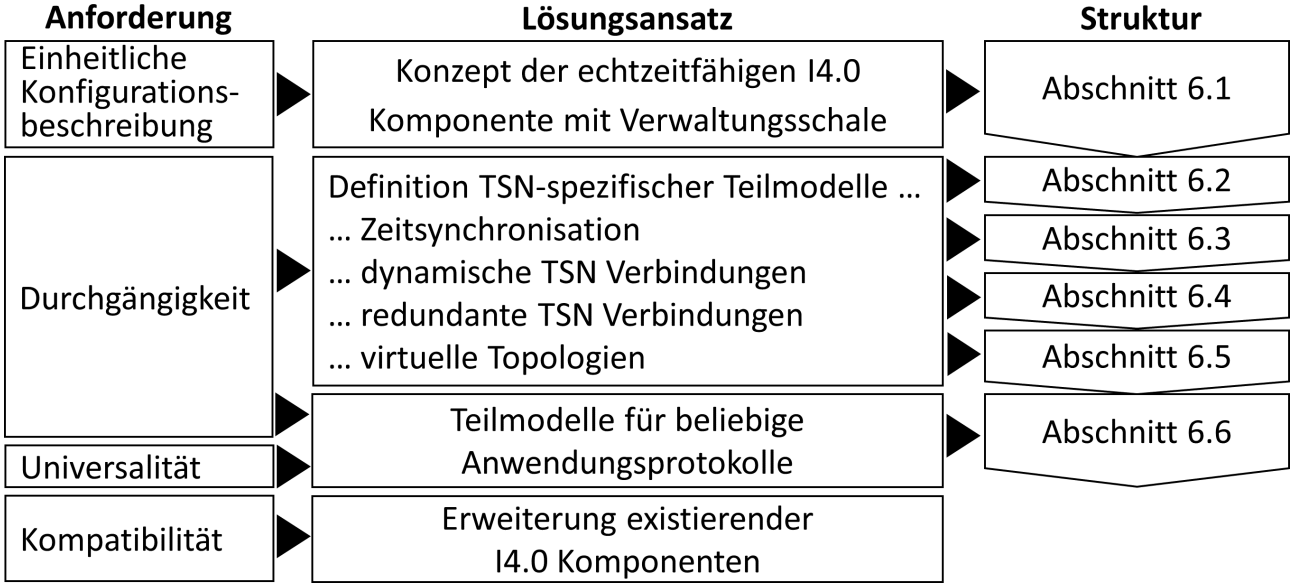


Abbildung 6.2: Der vorgestellte Lösungsansatz basierend auf den Anforderungen an die Parametrierung der echtzeitfähigen Assets.

Teilmodelle Die Verwaltungsschale einer echtzeitfähigen I4.0 Komponente beinhaltet neben den Asset-spezifischen Teilmodellen zusätzliche Teilmodelle für die TSN-basierte Echtzeitkommunikation. Dabei basieren alle Teilmodelle auf dem standardisierten Beschreibungsformat und spezifizieren Eigenschaften, Events und Operationen für die jeweilige Fähigkeit. Mit dem Fokus auf der Echtzeitkommunikation werden dabei insbesondere Konfigurationsparameter und Statusinformationen als Eigenschaften beschrieben. Die verwendeten Datentypen für die Eigenschaften im Teilmodell sind im Bild 6.1 dargestellt.

Teilmodelle für die Echtzeitkommunikation Im Folgenden werden die zusätzlichen Teilmodelle für die TSN-basierte Echtzeitkommunikation im Detail vorgestellt. Dabei unterteilt sich die Parametrierung der echtzeitfähigen I4.0 Komponenten in die fünf Teilaspekte des vorgestellten TSN SDKs (Bild 6.2). Für jeden Teilaspekt werden die erforderlichen Konfigurationsparameter und Statusinformationen herausgearbeitet und entsprechende Teilmodelle definiert. Die resultierenden Teilmodellen werden von den echtzeitfähigen I4.0 Komponenten bereitgestellt und bilden die Grundlage für eine einheitliche Konfiguration der Echtzeitverbindungen (Kapitel 7).

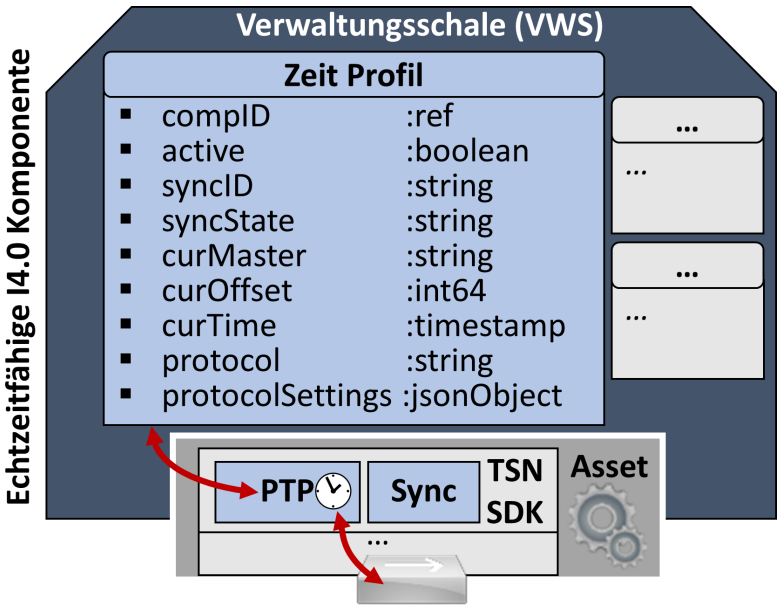


Abbildung 6.3: Das neue Teilmodell zur Parametrierung der Zeitsynchronisation in den echtzeitfähigen I4.0 Komponenten.

6.2 Zeitsynchronisation

Die Zeitsynchronisation im Netzwerk ist der erste Teilaspekt der TSN-basierten Echtzeitkommunikation und bildet die Grundlage für das TSN Scheduling in den echtzeitfähigen I4.0 Komponenten. Dabei wird die lokale Systemzeit in den echtzeitfähigen I4.0 Komponenten mit einer globalen Netzwerkzeit synchronisiert. Die vorgestellte Zeitsynchronisation basiert auf dem weit verbreiteten Precision Time Protocol (PTP) mit dem erweiterten Profil IEEE 802.1AS (Norm IEEE 802.1AS). Dabei wurden zwei Softwarebausteine zum TSN SDK der echtzeitfähigen I4.0 Komponenten hinzugefügt (Bild 5.4). Im Folgenden wird ein neues Teilmodell zur einheitlichen Parametrierung der beiden Softwarebausteine vorgestellt.

Teilmodell zur Zeitsynchronisation Zur Parametrierung der Zeitsynchronisation in den echtzeitfähigen I4.0 Komponenten wird das Teilmodell “Zeit Profil” zur Verwaltungsschale hinzugefügt. Dieses Teilmodell beinhaltet die Konfigurationsparameter und Statusinformationen für das verwendete Precision Time Protocol (PTP) bzw. die Zeitsynchronisation allgemein (Bild 6.3). Dabei werden konkret die folgenden Eigenschaften im Teilmodell spezifiziert:

compID Die ID ermöglicht die eindeutige Zuordnung des Teilmodells zu einer echtzeitfähigen I4.0 Komponente.

active Dieses Flag aktiviert oder deaktiviert die Zeitsynchronisation in der echtzeitfähigen I4.0 Komponente.

syncID Diese Eigenschaft entspricht der ID der internen PTP Uhr, die zur protokollspezifischen Identifikation im Netzwerk dient.

syncState Diese Statusinformation beschreibt den aktuellen Zustand der Zeitsynchronisation.

curMaster Diese Eigenschaft zeigt die ID der PTP Uhr des aktuellen Masters im Netzwerk. Die entsprechende Uhr definiert die globale Netzwerkzeit und dient als Synchronisationsgrundlage. Sollte die Eigenschaft nicht spezifiziert sein, agiert die echtzeitfähige I4.0 Komponente selbst als Master.

curOffset Diese Statuseigenschaft zeigt die aktuelle Zeitdifferenz zwischen der internen Systemzeit in der I4.0 Komponente und der globalen Netzwerkzeit an. Diese Differenz wird in Nanosekunden angegeben und kann sowohl positive als auch negative Werte annehmen. Sofern der absolute Wert kleiner als 30 Nanosekunden ist, werden beide Uhren üblicherweise als synchronisiert betrachtet.

curTime Diese Eigenschaft zeigt die aktuelle Systemzeit der I4.0 Komponente als Information bzw. zum Debugging an.

protocol Diese Eigenschaft ermöglicht die Auswahl des verwendeten Protokolls zur Zeitsynchronisation. Dabei ist PTP die bevorzugte Lösung und wird als Teil des TSN SDKs in den echtzeitfähigen I4.0 Komponenten unterstützt.

protocolSettings Basierend auf PTP existieren verschiedene Profile für die spezifischen Anwendungsfälle. Diese Profile, wie z.B. das Profil IEEE 802.1AS (Norm IEEE 802.1AS), werden als JSON-Objekt spezifiziert und über die Eigenschaft im Teilmodell gesetzt.

Das neue Teilmodell in der Verwaltungsschale ist mit dem PTP-spezifischen Softwarebaustein im TSN SDK synchronisiert (Bild 6.3). Folglich wird eine Änderung eines Konfigurationsparameters im Teilmodell entsprechend umgesetzt und die Statusinformationen zur Verfügung gestellt. Dabei muss jede echtzeitfähige I4.0 Komponente in der neuen Kommunikationsinfrastruktur das vorgestellte Teilmodell in der Verwaltungsschale bereitstellen. Dadurch wird eine einheitliche Parametrierung der Zeitsynchronisation gewährleistet.

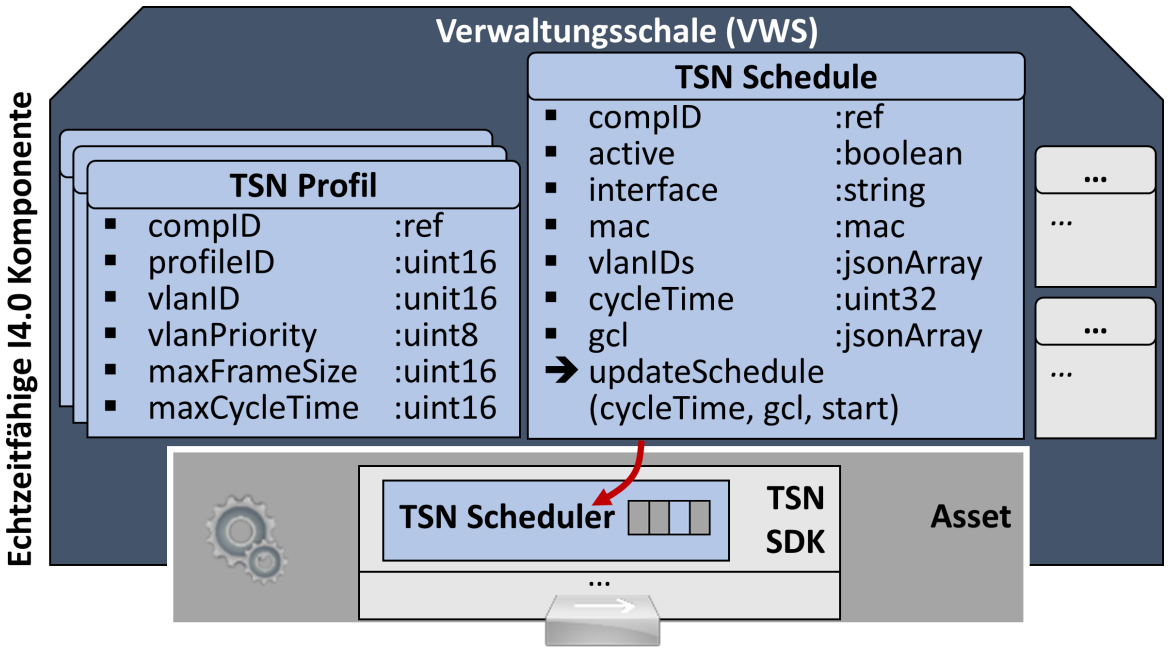


Abbildung 6.4: Zwei zusätzliche Teilmodelle zur Parametrierung des TSN Schedulings in den echtzeitfähigen I4.0 Komponenten.

6.3 Dynamische TSN Verbindungen

Für die dynamischen TSN Verbindungen wurde das TSN SDK in den echtzeitfähigen I4.0 Komponenten um einen TSN Scheduler erweitert (Bild 5.7). Dieser Scheduler realisiert ein Prioritäten-basiertes Scheduling entsprechend dem TSN Standard IEEE 802.1Qbv (Norm IEEE 802.1Qbv). Zur Parametrierung des TSN Schedulers und der dynamischen TSN Verbindungen allgemein werden zwei zusätzliche Teilmodelle in den echtzeitfähigen I4.0 Komponenten definiert. Dabei beschreibt das erste Teilmodell die TSN-spezifischen Echtzeitanforderungen der jeweiligen I4.0 Komponente. Das zweite Teilmodell ermöglicht die einheitliche Parametrierung des TSN Schedulers zur Laufzeit.

Teilmodell mit TSN Anforderungen Für die einheitliche Beschreibung der TSN-spezifischen Kommunikationsanforderungen wird ein sogenanntes “TSN Profil” als Teilmodell definiert (Bild 6.4). Dieses Profil beinhaltet die Anforderungen einer echtzeitfähigen I4.0 Komponente an eine dynamische TSN Verbindung. Konkret werden die folgenden Anforderungen als Eigenschaften im Teilmodell spezifiziert:

compID Die ID ermöglicht die eindeutige Zuordnung des Teilmodells zu einer echtzeitfähigen I4.0 Komponente.

profileID Jedes Profil verfügt über eine eindeutige ID in der Kommunikationsinfrastruktur.

vlanID Diese Eigenschaft spezifiziert die VLAN-ID für die übertragenen TSN Frames und die TSN Verbindung in der Kommunikationsinfrastruktur. Dabei ist ein Wertebereich zwischen 0 (Standardwert) und 4095 möglich.

vlanPriority Dieser Konfigurationsparameter spezifiziert die Priorität der echtzeitkritischen TSN Frames. Der Wert variiert zwischen der höchstmöglichen Priorität 7 und der gewöhnlichen Priorität 0.

maxFrameSize Dieser Parameter spezifiziert die maximale Größe der echtzeitkritischen TSN Frames, wobei der maximal zulässige Wert 1522 Byte beträgt. Die maximale Framegröße bestimmt die Zeit zum Senden und Empfangen des TSN Frames und wird zur Berechnung des Schedules für die TSN Verbindung verwendet.

maxCycleTime Neben der maximalen Framegröße ist die Zykluszeit der TSN Frames entscheidend. Der dazugehörige Konfigurationsparameter spezifiziert die Taktzeit in Mikrosekunden, d.h. in welchen Zeitabständen ein TSN Frame gesendet wird. Die Taktzeit richtet sich üblicherweise nach der jeweiligen Echtzeitanwendung im Asset.

Das “TSN Profil” wird für jede TSN Verbindung von der echtzeitfähigen I4.0 Komponenten definiert. Folglich kann das entsprechende Teilmodell mehrfach in der Verwaltungsschale spezifiziert sein. Zur Unterscheidung der verschiedenen Profile in der I4.0 Komponente bzw. der Kommunikationsinfrastruktur allgemein dient die eindeutige “Profile-ID”. Darüber hinaus sollten auch gewöhnliche I4.0 Komponenten ohne harte Echtzeitanforderungen ihre Kommunikationsanforderungen mit dem “TSN Profil” charakterisieren und insbesondere die gewünschte Priorität der Datenübertragung spezifizieren. Dadurch kann jede I4.0 Komponente in der Kommunikationsinfrastruktur bestmöglich bei der Netzwerkkonfiguration (Kapitel 7) berücksichtigt und die zur Verfügung stehende Netzwerkbandbreite effizient genutzt werden.

Teilmodell für TSN Scheduler Neben den Kommunikationsanforderungen ist die Parametrierung des TSN Schedulers in den echtzeitfähigen I4.0 Komponenten erforderlich. Dabei wird insbesondere der aktuelle Schedule für die TSN-fähige Netzwerkschnittstelle der I4.0 Komponente spezifiziert. Zur einheitlichen Parametrierung wird ein weiteres Teilmodell “TSN Schedule” definiert, das mit dem Scheduler im TSN SDK synchronisiert ist (Bild 6.4). Konkret beinhaltet das Teilmodell die folgenden Konfigurationsparameter als Eigenschaften und eine zusätzliche Operation:

compID Die ID ermöglicht die eindeutige Zuordnung des Teilmodells zu einer echtzeitfähigen I4.0 Komponente.

active Dieses Flag aktiviert oder deaktiviert das TSN Scheduling in der echtzeitfähigen I4.0 Komponente.

interface Bei echtzeitfähigen I4.0 Komponenten mit mehreren Netzwerkadaptern spezifiziert dieser Konfigurationsparameter die Schnittstelle zum Senden und Empfangen der echtzeitkritischen TSN Frames.

mac Dieser Parameter entspricht der MAC-Adresse der verwendeten Netzwerkschnittstelle in der echtzeitfähigen I4.0 Komponente. Diese Adresse bestimmt die Quelladresse bei gesendeten Frames und die Zieladresse für andere I4.0 Komponenten.

vlanIDs Über die TSN-fähige Netzwerkschnittstelle einer echtzeitfähigen I4.0 Komponente werden Frames mit unterschiedlichen VLAN-IDs und Prioritäten übertragen. Dieser Konfigurationsparameter spezifiziert dabei ein Array mit allen zulässigen VLAN-IDs. Folglich werden TSN Frames mit anderen VLAN-IDs von dem TSN Scheduler verworfen.

cycleTime Die aktuelle Zykluszeit bestimmt die Dauer eines TSN Schedules und stimmt gewöhnlich mit dem globalen Takt im Netzwerk überein.

gcl Die sogenannte Gate-Control-List (GCL) beschreibt den eigentlichen TSN Schedule für die TSN-fähige Netzwerkschnittstelle in der echtzeitfähigen I4.0 Komponente. Dazu werden aufeinanderfolgende Zeitslots mit einer gewissen Dauer und Priorität definiert. Die Gesamtdauer aller Zeitslots entspricht üblicherweise der spezifizierten Zykluszeit, sodass sich die GCL zyklisch wiederholt.

scheduleUpdate(neueZykluszeit, neueGcl, startZeit) Diese Operation ermöglicht die Aktualisierung des TSN Schedules zur Laufzeit. Dazu wird sowohl die neue Zykluszeit als auch die neue GCL als Parameter übergeben. Basierend auf der Zeitsynchronisation bestimmt der dritte Parameter die Startzeit des neuen TSN Schedules und ermöglicht eine zeitsynchrone Aktualisierung in der Kommunikationsinfrastruktur.

Zusammengefasst wurden die zwei Teilmodelle “TSN Profil” und “TSN Schedule” für die Verwaltungsschale der echtzeitfähigen I4.0 Komponenten definiert. Diese Teilmodelle ermöglichen die einheitliche Beschreibung der TSN-spezifischen Kommunikationsanforderungen sowie die Parametrierung des TSN Schedulings. Dabei ermöglicht das zweite Teilmodell insbesondere eine Aktualisierung des TSN Schedulers im TSN SDK zur Laufzeit. Dadurch wird die Grundlage

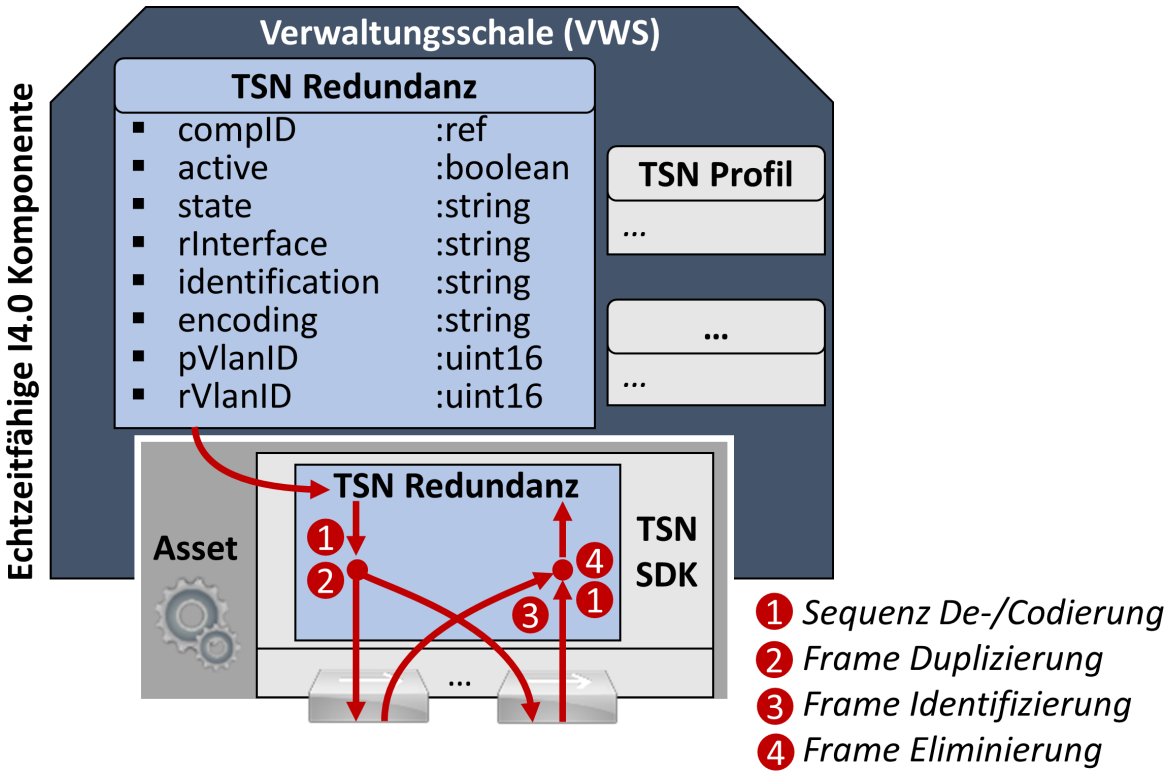


Abbildung 6.5: Das Teilmodell "TSN Redundanz" ermöglicht Parametrierung der redundanten TSN Verbindungen in den echtzeitfähigen Assets.

für die Konfiguration der dynamischen TSN Verbindungen in der Kommunikationsinfrastruktur geschaffen.

6.4 Redundante TSN Verbindungen

Im vorherigen Abschnitt wurde die Parametrierung des TSN Schedulers im TSN SDK der echtzeitfähigen I4.0 Komponenten vorgestellt. Darauf aufbauend wird die Parametrierung der zusätzlichen Redundanzmechanismen betrachtet. Diese Mechanismen ermöglichen die redundante Übertragung eines TSN Frames über eine weitere Netzwerkschnittstelle in der echtzeitfähigen I4.0 Komponente (Bild 5.9). Zur Parametrierung der Redundanzmechanismen wird im Folgenden ein zusätzliches Teilmodell definiert.

Teilmodell für Redundanz Das neue Teilmodell "TSN Redundanz" dient zur Parametrierung der spezifischen Redundanzmechanismen im TSN SDK. Dabei ist das Teilmodell optional,

d.h. es wird von der echtzeitfähigen I4.0 Komponente bei Bedarf in der Verwaltungsschale spezifiziert. Konkret beinhaltet das neue Teilmodell die folgenden Konfigurationsparameter und Statusinformationen als Eigenschaften (Bild 6.5):

compID Die ID ermöglicht die eindeutige Zuordnung des Teilmodells zu einer echtzeitfähigen I4.0 Komponente.

active Dieses Flag aktiviert oder deaktiviert die Redundanzmechanismen in der echtzeitfähigen I4.0 Komponente.

state Diese Eigenschaft beschreibt den aktuellen Zustand der redundanten TSN Verbindung. Dabei ist die Ausfallsicherheit der Verbindung gewährleistet, wenn redundante TSN Frames gesendet bzw. empfangen werden.

rInterface Basierend auf der Netzwerkschnittstelle für die primäre TSN Verbindung, wird eine weitere Schnittstelle für die redundante Übertragung der TSN Frames spezifiziert. Für eine bestmögliche Redundanz sollten sich die beiden Netzwerkschnittstellen in der echtzeitfähigen I4.0 Komponente unterscheiden.

identification Da echtzeitfähige I4.0 Komponenten über mehrere parallele TSN Verbindungen verfügen können, muss jeder eingehende TSN Frame einer Verbindung zugeordnet werden. Dazu spezifiziert dieser Konfigurationsparameter die relevanten Felder zur Frame Identifikation, wobei zwei grundlegende Varianten unterschieden werden. Die erste Variante nutzt ausschließlich Informationen aus dem Ethernet Header des eingehenden TSN Frames und ist unabhängig vom restlichen Payload. Die MAC-Adresse des Senders in Kombination mit der VLAN-Priorität wird beispielsweise zur Identifikation verwendet. Im Gegensatz dazu verwendet die zweite Variante Protokollinformationen auf den höheren OSI-Schichten oberhalb von TSN. Die IP-Adresse des Senders mit dem UDP-Port ist ein Beispiel um eine Echtzeitverbindung zu identifizieren. Während die zweite Variante bestimmte Protokolle auf höherer Ebene voraussetzt, erlaubt sie gleichzeitig eine höhere Flexibilität in Bezug auf den eigentlichen TSN Frame.

encoding Dieser Parameter bestimmt die Codierung der Sequenznummer im TSN Frame. Basierend auf dieser Sequenznummer werden doppelte TSN Frames erkannt und eliminiert. Die empfohlene Variante aus dem TSN Standard IEEE 802.1CB ist die Codierung der Sequenznummer im sogenannten Redundancy-Tag (Norm IEEE 802.1CB).

pVlanID Dieser Konfigurationsparameter spezifiziert die VLAN-ID der primären TSN Verbindung, sodass die dazugehörigen TSN Frames redundant übertragen werden. Dadurch

kann die Redundanz auf gewisse, bevorzugt echtzeitkritische, TSN Frames eingeschränkt werden und der restliche Datenverkehr normal übertragen werden.

rVlanID Dieser Parameter definiert die VLAN-ID für die redundanten TSN Frames im zulässigen Wertebereich von 0 bis 4095. Diese VLAN-ID kann von der VLAN-ID der primären TSN Verbindung abweichen und dient zur Konfiguration der redundanten TSN Verbindung im Netzwerk. Voraussetzung ist allerdings, dass das VLAN-Tag im Ethernet Header nicht zur Frame Identifikation genutzt wird und somit nicht identisch zum primären Frame sein muss.

Das vorgestellte Teilmodell "TSN Redundanz" ermöglicht die Parametrierung des entsprechenden Softwarebausteins im TSN SDK der echtzeitfähigen I4.0 Komponenten. Dabei werden die vier erforderlichen Redundanzmechanismen parametriert und die Grundlage für die Konfiguration von redundanten TSN Verbindungen in der Kommunikationsinfrastruktur geschaffen (Kapitel 7).

6.5 Virtuelle Netzwerktopologien

Die virtuellen Netzwerktopologien ermöglichen die Vernetzung von mehreren echtzeitfähigen I4.0 Komponenten in der neuen Kommunikationsinfrastruktur. Dabei werden logische Linien- oder Ring-Topologien unabhängig von der darunterliegenden physikalischen Netzwerkinfrastruktur etabliert. Dazu wurden die erforderlichen Kommunikationsmechanismen in das TSN SDK der echtzeitfähigen I4.0 Komponenten integriert (Bild 5.11). Zur Parametrierung der zusätzlichen Mechanismen wird im Folgenden ein weiteres Teilmodell für die Verwaltungsschale definiert. Dabei erfolgt die Parametrierung in Abhängigkeit der konkreten echtzeitfähigen I4.0 Komponente und der aktuell konfigurierten virtuellen Netzwerktopologie.

Teilmodell für virtuelle Netzwerktopologien Zur Teilnahme in einer virtuellen Netzwerktopologie wird ein weiteres Teilmodell "TSN VT" für die Verwaltungsschale der echtzeitfähigen I4.0 Komponenten definiert (Bild 6.6). Es beinhaltet sowohl die erforderlichen Konfigurationsparameter als auch zusätzliche Statusinformationen:

compID Die ID ermöglicht die eindeutige Zuordnung des Teilmodells zu einer echtzeitfähigen I4.0 Komponente.

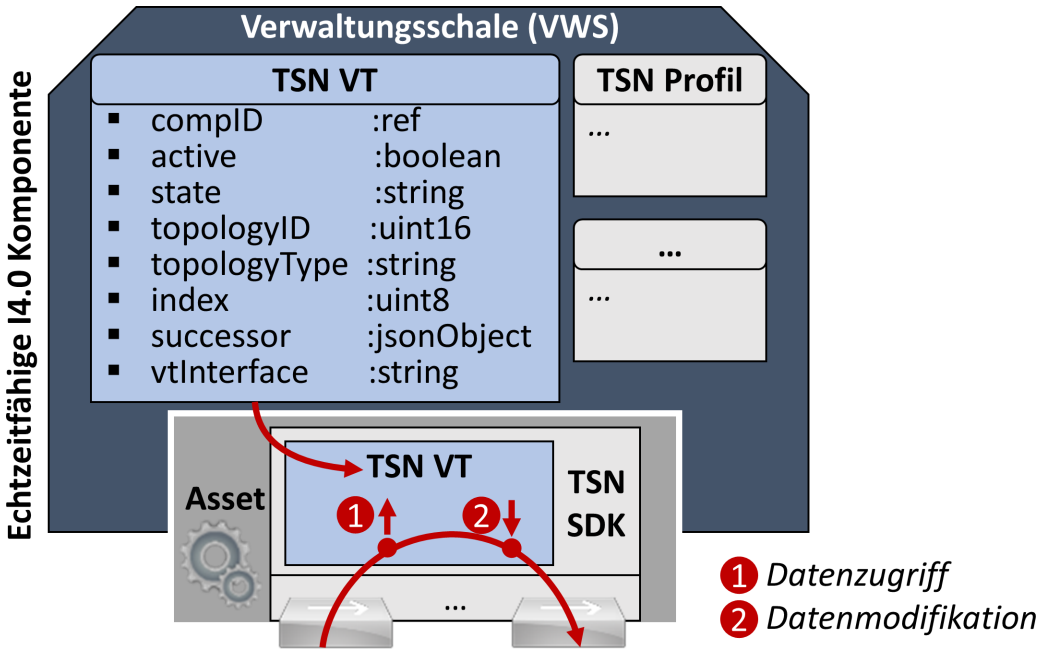


Abbildung 6.6: Das Teilmodell "TSN VT" dient zur Parametrierung von virtuellen Netzwerktopologien in einer echtzeitfähigen I4.0 Komponente.

active Dieses Flag aktiviert oder deaktiviert die Kommunikationsmechanismen für eine virtuelle Netzwerktopologie in der echtzeitfähigen I4.0 Komponente.

state Diese Eigenschaft beschreibt den aktuellen Zustand der virtuellen Netzwerktopologie.

topologyID Jede virtuelle Netzwerktopologie verfügt über eine eindeutige ID um mehrere Topologien in der Kommunikationsinfrastruktur zu unterscheiden. Darauf aufbauend beschreibt dieser Konfigurationsparameter die Zugehörigkeit der echtzeitfähigen I4.0 Komponente.

topologyType Diese Eigenschaft beschreibt die Art der etablierten virtuellen Netzwerktopologie, d.h. Linien- oder Ring-Topologie.

index Der Index ist ein positiver Wert, der die logische Position der echtzeitfähigen I4.0 Komponente innerhalb der virtuellen Netzwerktopologie spezifiziert. Dabei hat der Master einer Topologie immer den Index 0.

successor Dieser Konfigurationsparameter beschreibt die nachfolgende echtzeitfähige I4.0 Komponente innerhalb der virtuellen Netzwerktopologie, an die jeder eingehende TSN Frame weitergeleitet wird. Der Parameter ist ein komplexes JSON-Objekt und beinhaltet unter anderem die Komponenten-ID und den Index. Darüber hinaus dient die MAC-Adresse

des Nachfolgers als Zieladresse für den weitergeleiteten TSN Frame in der virtuellen Netzwerktopologie.

vtInterface Dieser Konfigurationsparameter spezifiziert die Netzwerkschnittstelle zum Weiterleiten der eingehenden TSN Frames, die sich von der primären TSN Schnittstelle unterscheiden kann.

Die vorgestellten Eigenschaften in dem Teilmodell "TSN VT" parametrieren den entsprechenden Softwarebaustein im TSN SDK der echtzeitfähigen I4.0 Komponente. Dadurch wird die Teilnahme in einer virtuellen Topologie ermöglicht und die Grundlage für die Konfiguration der gesamten virtuellen Topologie im Netzwerk geschaffen (Kapitel 7).

6.6 Anwendungsprotokolle

In den vorherigen Abschnitten wurden die TSN-spezifischen Kommunikationsmechanismen in den echtzeitfähigen I4.0 Komponenten betrachtet und das vorgestellte TSN SDK parametriert. Dazu wurden neue TSN-spezifische Teilmodelle mit Konfigurationsparametern und Statusinformationen für die Verwaltungsschale definiert. Darauf aufbauend erfolgt die Parametrierung der (echtzeitfähigen) Protokolle auf den höheren OSI-Schichten, insbesondere den Anwendungsprotokollen.

Teilmodelle für Anwendungsprotokolle Im Gegensatz zu TSN als einheitliche Kommunikationsbasis wird bei den Anwendungsprotokollen kein spezifisches Protokoll vorgeschrieben. Stattdessen kann ein, für den individuellen Anwendungsfall optimales, Protokoll verwendet werden. Daher wird im Folgenden die einheitliche Parametrierung von beliebigen Anwendungsprotokollen in den echtzeitfähigen I4.0 Komponenten betrachtet. Analog zu TSN basiert die Parametrierung auf der Definition zusätzlicher Teilmodelle für das jeweilige Protokoll bzw. das dazugehörige App SDK in der echtzeitfähigen I4.0 Komponente. Dabei werden drei verschiedene Varianten vorgestellt: die monolithische, modulare und generische Parametrierung (Bild 6.7). Diese Varianten bauen aufeinander auf und werden im Folgenden detailliert erläutert.

Monolithische Parametrierung Die erste Variante beschreibt die Parametrierung eines Anwendungsprotokolls mit einem einzelnen Teilmodell in der Verwaltungsschale (Bild 6.8). Diese monolithische Variante eignet sich insbesondere für relativ einfache Anwendungsprotokolle mit wenigen Konfigurationsmöglichkeiten. Dabei werden alle Konfigurationsparameter und

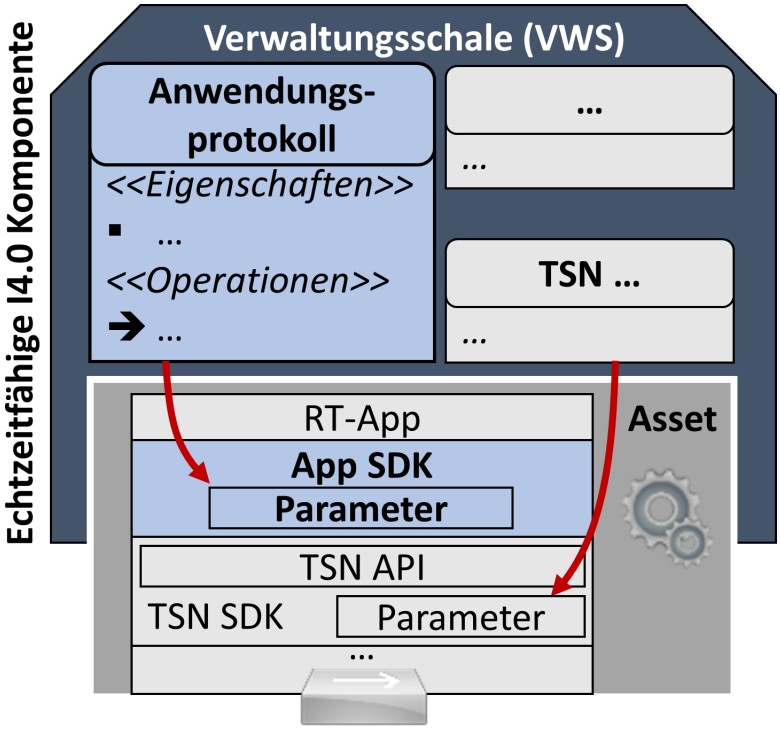


Abbildung 6.7: Die Parametrierung des Anwendungsprotokolls über zusätzliche Teilmodelle in der Verwaltungsschale einer echtzeitfähigen I4.0 Komponente.

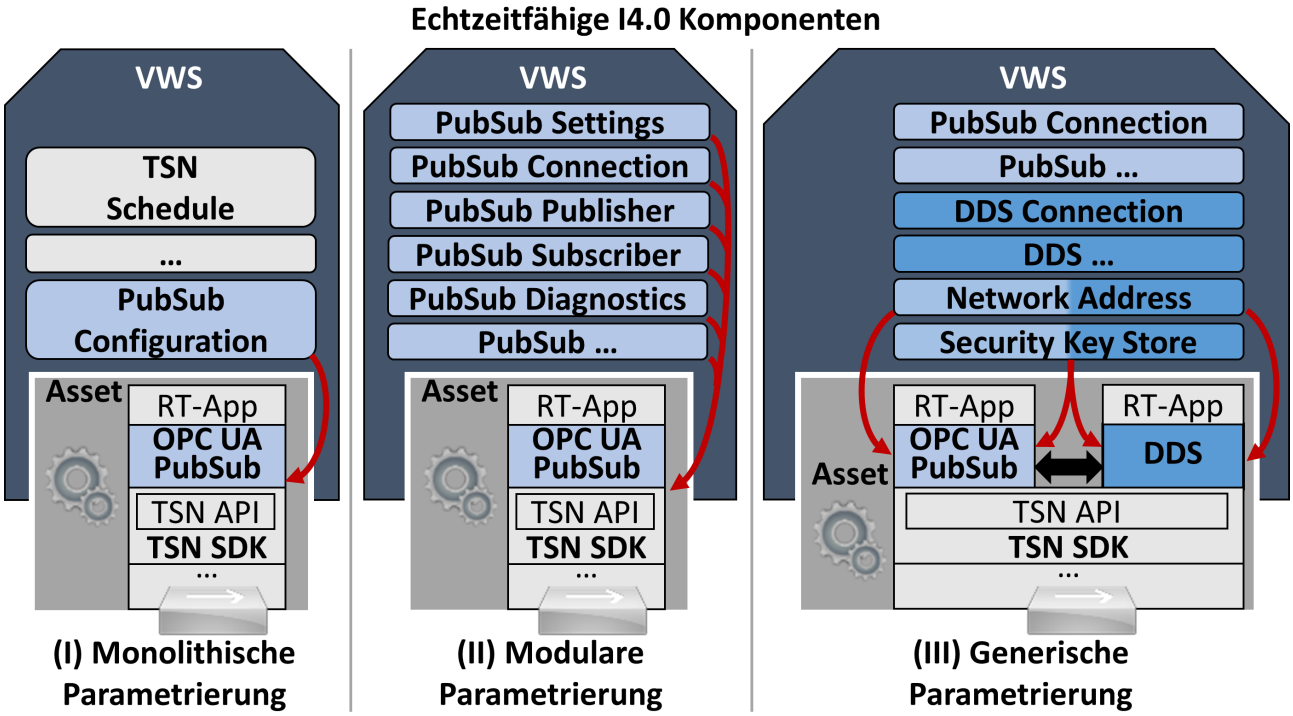


Abbildung 6.8: Drei Varianten zur Parametrierung des Anwendungsprotokolls in einer echtzeitfähigen I4.0 Komponente.

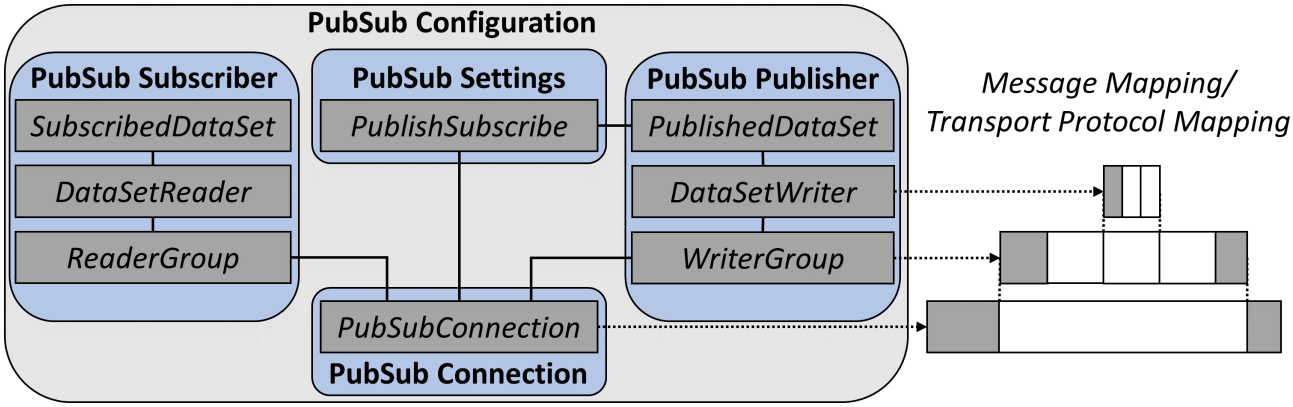


Abbildung 6.9: Monolithische bzw. modulare Parametrierung am Beispiel von OPC UA PubSub.

Statusinformationen als Eigenschaften in einem Teilmodell zusammengefasst. Neben den Eigenschaften kann das Teilmodell auch verfügbare Operationen bzw. Events spezifizieren. Vorteil ist die konsistente Darstellung in der Verwaltungsschale und die einheitliche Parametrierung analog zu den TSN-spezifischen Teilmodellen. Für komplexere Anwendungsprotokolle ist diese monolithische Variante allerdings auf Grund der umfangreichen Konfigurationsmöglichkeiten ungeeignet. Die Vielzahl an Konfigurationsparametern führt zu einem komplexen, unstrukturierten Teilmodell in der Verwaltungsschale. Insbesondere beinhaltet das Teilmodell auch alle optionalen Konfigurationsparameter, obwohl diese nicht zwangsläufig von jeder Echtzeitanwendung benötigt werden. Daher wird für die komplexeren Anwendungsprotokolle im Folgenden eine strukturiertere Variante vorgestellt.

Modulare Parametrierung Die zweite Variante beschreibt die modulare Parametrierung eines Anwendungsprotokolls mit mehreren Teilmodellen in der Verwaltungsschale. Jedes einzelne Teilmodell beschreibt eine Teilfunktionalität des Anwendungsprotokolls und spezifiziert dazugehörige Konfigurationsparameter und Statusinformationen als Eigenschaften. Dabei wird zwischen notwendigen und optionalen Teilmodellen für das jeweilige Anwendungsprotokoll unterschieden. Die optionalen Teilmodelle werden nur bei Bedarf, d.h. je nach Echtzeitanwendung, in der Verwaltungsschale bereitgestellt.

Am Beispiel von OPC UA PubSub wird die zweite Variante verdeutlicht und mehrere Teilmodelle für die verschiedenen Protokollfunktionalitäten definiert. Dabei wird beispielsweise zwischen Publisher- und Subscriber-spezifischen Funktionalitäten unterschieden. Konkret werden vier notwendige Teilmodelle basierend auf den standardisierten Object Types definiert (Bild 6.9):

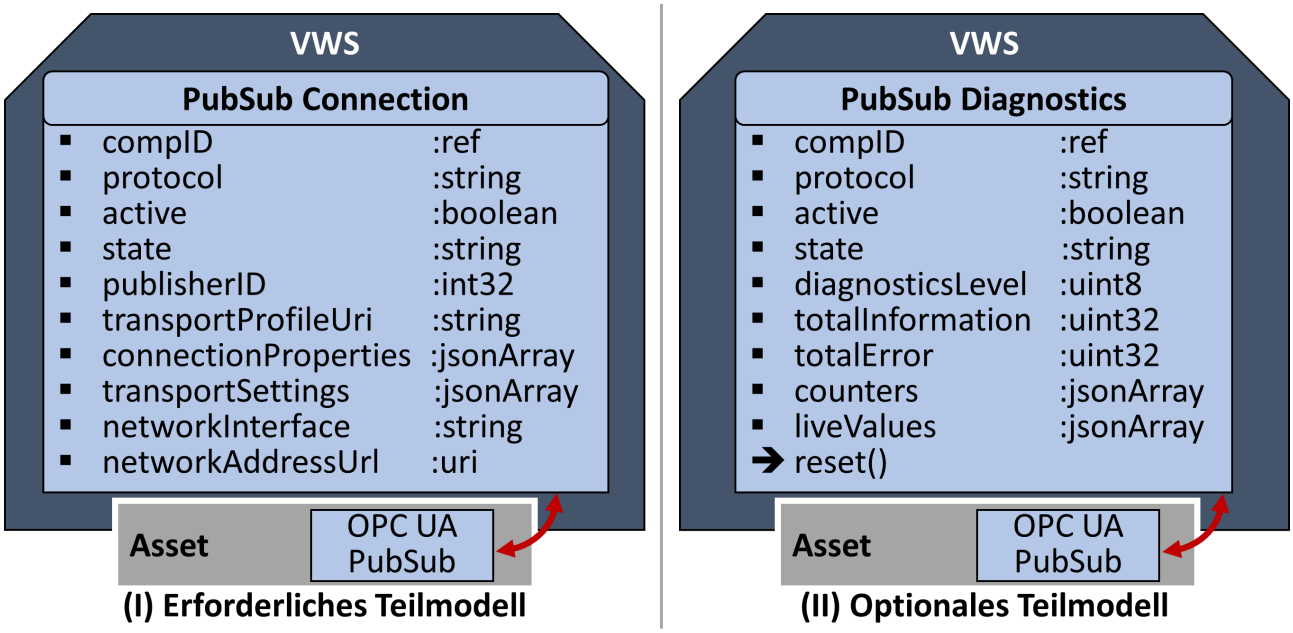


Abbildung 6.10: Ein notwendiges und ein optionales Teilmodell zur Parametrierung des Anwendungsprotokolls in einer echtzeitfähigen I4.0 Komponente.

PubSub Settings Das erste Teilmodell basiert auf dem ObjectType “PublishSubscribe” und stellt allgemeine Konfigurationsparameter und Statusinformationen für eine echtzeitfähige I4.0 Komponente mit OPC UA PubSub zur Verfügung. Beispielsweise spezifiziert das Teilmodell die unterstützten Transportprotokolle sowie die allgemeinen Security-Einstellungen.

PubSub Connection Für jede OPC UA PubSub Verbindung wird ein dazugehöriges Teilmodell “PubSub Connection” spezifiziert. Dieses Teilmodell beinhaltet unter anderem netzwerkspezifische Informationen sowie die Konfigurationsparameter für das verwendete Transportprotokoll.

PubSub Publisher Dieses Teilmodell wird von einem Publisher verwendet, der Daten für andere I4.0 Komponenten zur Verfügung stellt. Es beinhaltet alle Konfigurationsparameter für die bereitgestellten Variablen und die resultierenden Dataset Nachrichten, wie z.B. die Datentypen und das Nachrichtenformat.

PubSub Subscriber Das vierte Teilmodell konfiguriert die Subscriber-spezifischen Funktionalitäten zum Empfangen und Verarbeiten von eingehenden Dataset Nachrichten. Beispielsweise umfasst das Teilmodell die Konfigurationsparameter für das Nachrichtenformat und die Datentypen der relevanten Variablen.

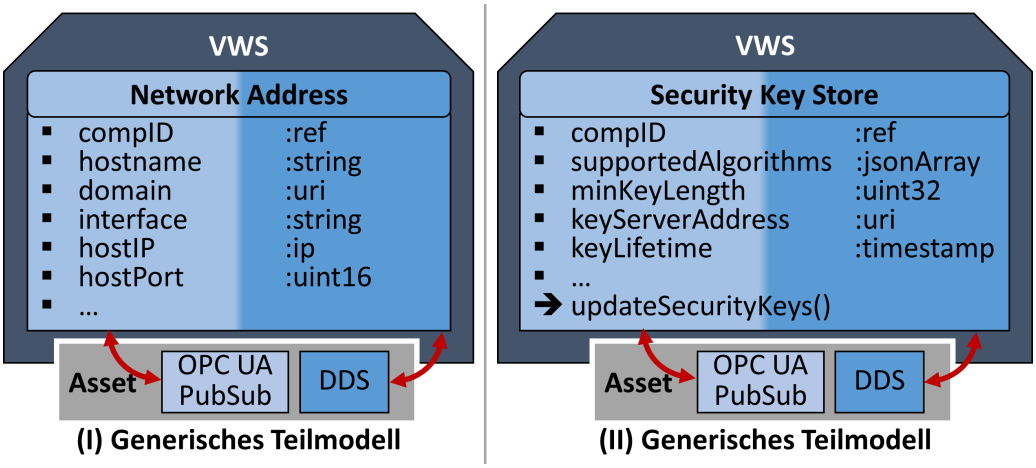


Abbildung 6.11: Zwei generische Teilmodelle, die auf mehrere Anwendungsprotokolle anwendbar sind.

Das Bild 6.10 zeigt das notwendige Teilmodell “PubSub Connection” mit den dazugehörigen Konfigurationsparametern und Statusinformationen als Eigenschaften. Während dieses Teilmodell von jeder echtzeitfähigen I4.0 Komponente mit OPC UA PubSub bereitgestellt wird, existieren weitere optionale Teilmodelle. Ein Beispiel für ein solches optionales Teilmodell ist “PubSub Diagnostics” (Bild 6.10). Dieses Teilmodell beinhaltet verschiedene Diagnoseparameter und Diagnoseinformationen für OPC UA PubSub. Außerdem wird eine Operation zum Zurücksetzen der Fehlerzähler und Diagnosedaten zur Verfügung gestellt.

Zusammengefasst ermöglicht die zweite Variante mit mehreren (optionalen) Teilmodellen eine strukturierte Parametrierung eines Anwendungsprotokolls. Dabei werden insbesondere nur die erforderlichen Teilmodelle in der Verwaltungsschale der echtzeitfähigen I4.0 Komponente bereitgestellt. Allerdings sind die einzelnen Teilmodelle spezifisch für ein bestimmtes Anwendungsprotokoll und lassen sich nicht ohne weiteres auf ein anderes Protokoll übertragen. Zur Parametrierung von echtzeitfähigen I4.0 Komponenten mit mehreren Anwendungsprotokollen, wird im Folgenden eine universellere Variante betrachtet.

Generische Parametrierung Für echtzeitfähige I4.0 Komponenten mit mehreren Anwendungsprotokollen ermöglicht die dritte Variante eine generische Parametrierung. Diese Variante beschreibt die Abstraktion von einem konkreten Anwendungsprotokoll und die Definition sogenannter generischer Teilmodelle für die Verwaltungsschale. Ein generisches Teilmodell spezifiziert allgemeine Konfigurationsparameter und Statusinformationen, die von unterschiedlichen Protokollen verwendet werden. Dadurch wird eine einheitliche Parametrierung von verschiedenen App SDKs über das selbe Teilmodell ermöglicht (Bild 6.8). Darüber hinaus wird der

Wechsel des Anwendungsprotokolls in einer echtzeitfähigen I4.0 Komponente vereinfacht, da die generischen Teilmodelle konstant bleiben.

Am Beispiel von OPC UA PubSub und DDS sind zwei generische Teilmodelle im Bild 6.11 dargestellt. Das generische Teilmodell "Network Address" beschreibt netzwerkspezifische Informationen der echtzeitfähigen I4.0 Komponente, wie z.B. den Hostnamen, die Domäne und die Netzwerkschnittstellen mit den dazugehörigen IP Adressen. Ein App SDK verwendet diese Informationen um beispielsweise die protokollspezifischen Pfade für die Adressierung im Netzwerk und den Datenzugriff abzuleiten. Das zweite generische Teilmodell "Security Key Management" parametriert einen lokalen Key Store um eine Netzwerkverbindung zu verschlüsseln und abzusichern. Dabei werden alle Sicherheitsschlüssel zentral verwaltet und die minimale Schlüssellänge, die Gültigkeit eines Schlüssels sowie die unterstützen Algorithmen zum Erstellen und Austauschen von Sicherheitsschlüsseln spezifiziert. Sofern erforderlich, wird ein Sicherheitsschlüssel von einem App SDK in der echtzeitfähigen I4.0 Komponente abgerufen und für das jeweilige Protokoll verwendet werden.

Zusammengefasst wurde in diesem Abschnitt die Parametrierung von beliebigen Anwendungsprotokollen und den dazugehörigen App SDKs in einer echtzeitfähigen I4.0 Komponente erläutert. Je nach Komplexität des Anwendungsprotokolls wurde dabei eine monolithische bzw. modulare Parametrierung vorgestellt. Darüber hinaus ermöglicht die dritte Variante eine generische Parametrierung für verschiedene Anwendungsprotokolle. Analog zum TSN SDK basieren die drei Varianten auf der Definition von zusätzlichen Teilmodellen für die Verwaltungsschale. Diese Teilmodelle in Kombination mit den TSN-spezifischen Teilmodellen ermöglichen eine einheitliche Parametrierung der gesamten Echtzeitkommunikation in einer echtzeitfähigen I4.0 Komponente.

6.7 Zusammenfassung

Ausgehend von der TSN-basierten Echtzeitkommunikation (Kapitel 5) wurde in diesem Kapitel die dazugehörige Parametrierung in den echtzeitfähigen Assets vorgestellt. Dazu wurde das Konzept der echtzeitfähigen I4.0 Komponente präsentiert, das eine gewöhnliche I4.0 Komponente erweitert. Das neue Konzept beschreibt ein echtzeitfähiges Asset mit einer Verwaltungsschale und zusätzlichen Echtzeit-spezifischen Teilmodellen (Bild 6.12). Diese Teilmodelle basieren auf den verschiedenen Teilaspekten der Echtzeitkommunikation und wurden im Detail vorgestellt.

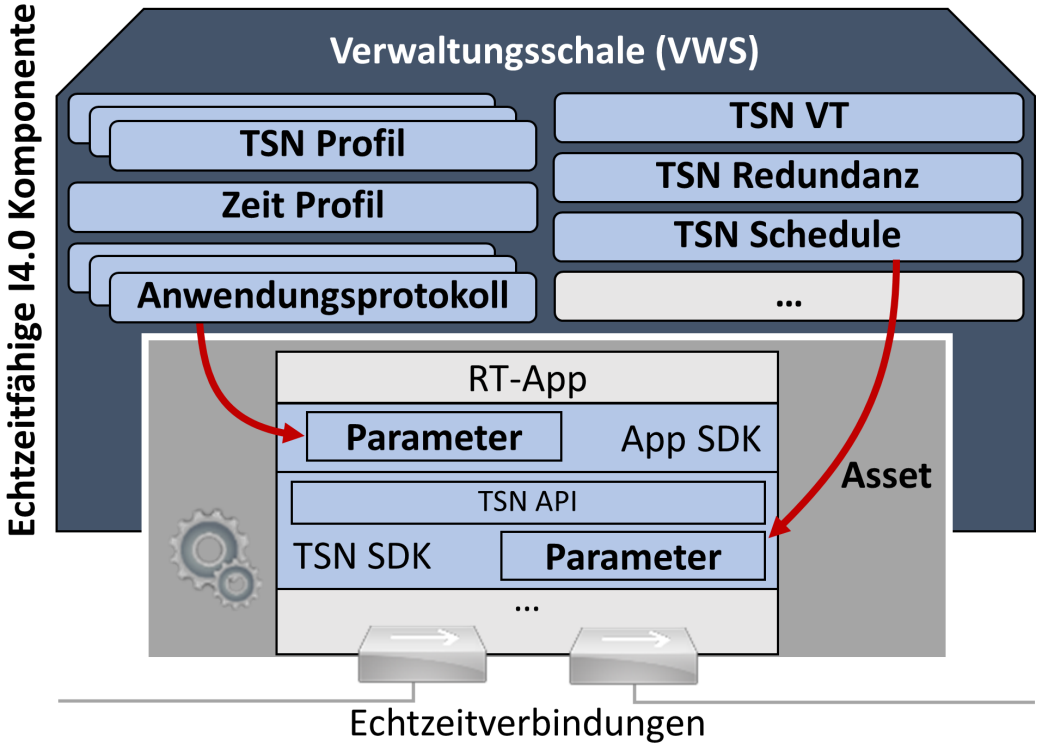


Abbildung 6.12: Eine echtzeitfähige I4.0 Komponente mit Verwaltungsschale und den zusätzlichen Teilmodellen zur Parametrierung der Echtzeitkommunikation.

Dadurch wurde die einheitliche Parametrierung der spezifischen Kommunikationsmechanismen in den echtzeitfähigen I4.0 Komponenten ermöglicht.

Netzwerkkonfiguration der Echtzeitkommunikation Die etablierte Echtzeitkommunikation (Kapitel 5) und die einheitliche Parametrierung (Kapitel 6) in den echtzeitfähigen I4.0 Komponenten bilden die Grundlage für die Konfiguration der Echtzeitverbindungen im Netzwerk. Die erforderliche Netzwerkkonfiguration ist der dritte wesentliche Aspekt der neuen Kommunikationsinfrastruktur und wird im folgenden Kapitel detailliert betrachtet. Dabei wird insbesondere der Aspekt der Wandelbarkeit berücksichtigt und dynamische TSN-basierte Echtzeitverbindungen zur Laufzeit etabliert.

Kapitel 7

SDN Controller zur dynamischen Netzwerkkonfiguration

In den beiden vorherigen Kapiteln wurde die Echtzeitkommunikation in den echtzeitfähigen Assets erarbeitet und parametriert. Dabei wurden die erforderlichen Kommunikationsmechanismen in die Assets integriert und das Konzept der echtzeitfähigen I4.0 Komponente mit spezifischen Teilmodellen vorgestellt. Basierend auf der Parametrierung einzelner echtzeitfähiger I4.0 Komponenten erfolgt in diesem Kapitel die dynamische Konfiguration der Echtzeitverbindungen im Netzwerk. Grundlage für die Netzwerkkonfiguration ist ein erweiterter SDN Controller, der sowohl die TSN Switches als auch die echtzeitfähigen Assets verwaltet. Darauf aufbauend wird die Netzwerkkonfiguration für die fünf verschiedenen Teilaspekte der Echtzeitkommunikation im Detail vorgestellt.

7.1 Einheitliche Konfigurationsschnittstelle

Eine deterministische Kommunikation mit geringer Latenz erfordert die Konfiguration von Echtzeitverbindungen in der neuen Kommunikationsinfrastruktur. Daher wird basierend auf dem TSN Standard 802.1Qcc ein zentraler Ansatz vorgestellt und ein existierender SDN Controller erweitert (Bild 7.1). Dieser SDN Controller konfiguriert sowohl die TSN Switches in der Netzwerkinfrastruktur als auch die echtzeitfähigen I4.0 Komponenten, d.h. er vereint die Konzepte der CNC (Central Network Configuration) und CUC (Central User Configuration) entsprechend dem TSN Standard (Norm IEEE 802.1Qcc).

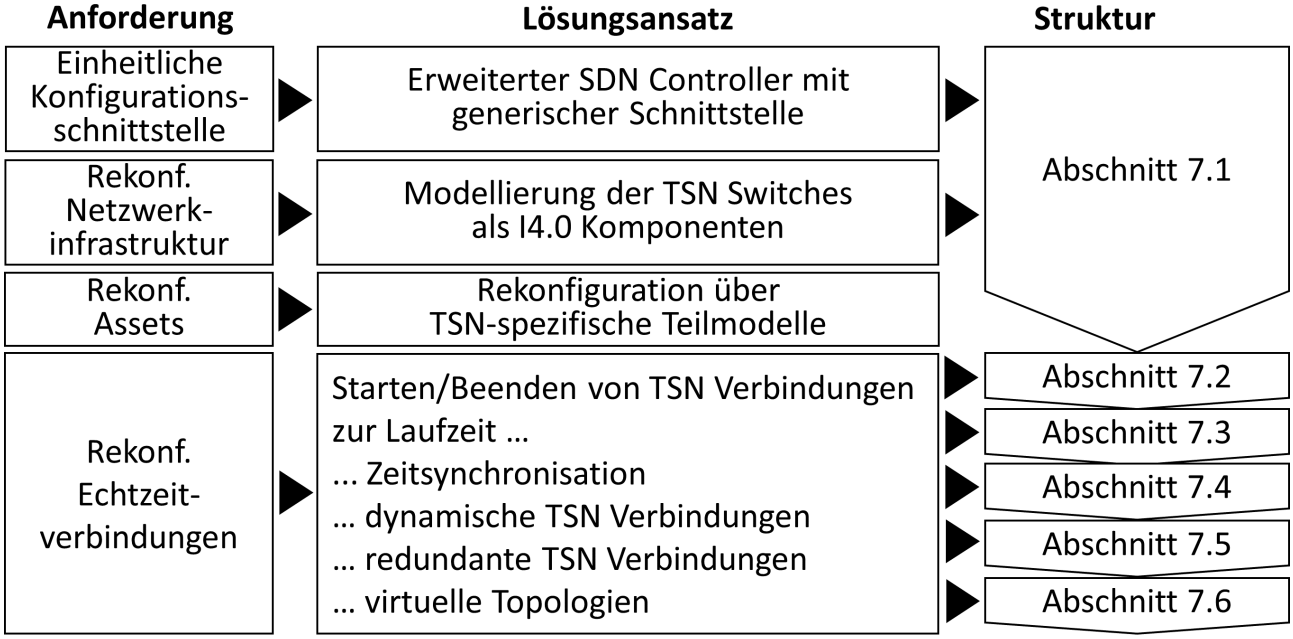
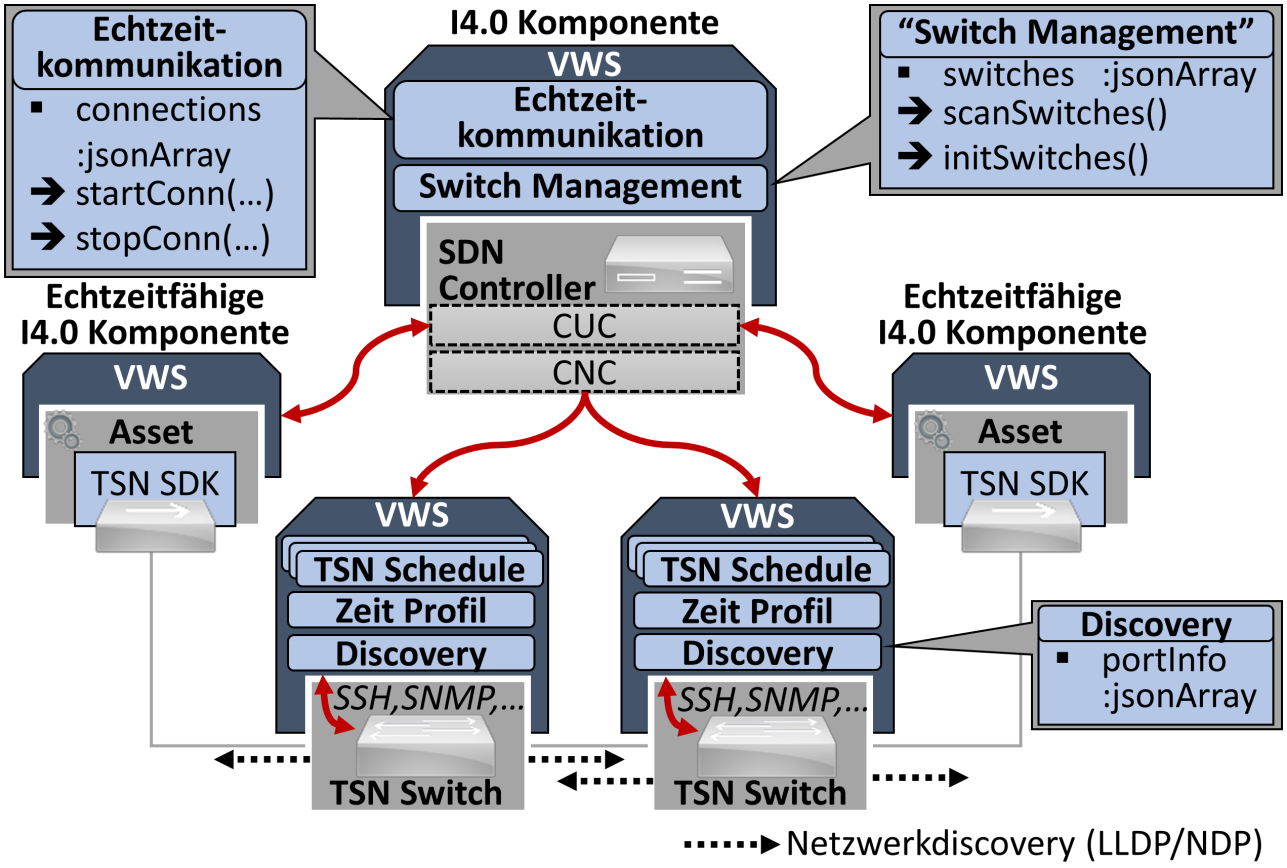


Abbildung 7.1: Der vorgestellte Lösungsansatz basierend auf den Anforderungen an die Netzwerkkonfiguration.

TSN Switches als I4.0 Komponenten Im vorherigen Kapitel wurde das Konzept der echtzeitfähigen I4.0 Komponente zur Parametrierung der echtzeitfähigen Assets vorgestellt. Zur einheitlichen Konfiguration in der Kommunikationsinfrastruktur werden die TSN Switches ebenfalls als I4.0 Komponenten mit einer Verwaltungsschale modelliert. Diese Verwaltungsschale beinhaltet mehrere Teilmodelle und spezifiziert die Kommunikationsmechanismen im TSN Switch unabhängig vom konkreten Switchhersteller (Bild 7.2). Dabei erfolgt die Synchronisation zwischen der Verwaltungsschale und der eigentlichen herstellerspezifischen Firmware des Switches über SSH, SNMP oder NETCONF. Während die interne Kommunikation herstellerspezifisch ist, ermöglicht die einheitliche Beschreibung als I4.0 Komponente eine herstellerunabhängige Konfiguration der gesamten Netzwerkinfrastruktur.

Für die TSN-basierte Echtzeitkommunikation stellt jeder TSN Switch drei Teilmodelle in seiner Verwaltungsschale zur Verfügung. Das erste Teilmodell “Zeit Profil” konfiguriert die Zeitsynchronisation im TSN Switch und bildet damit die Grundlage für das eigentliche TSN Scheduling. Im Gegensatz zu den echtzeitfähigen I4.0 Komponenten mit einer primären Netzwerkschnittstelle erfolgt das Scheduling in den TSN Switches Port-basiert. Folglich wird das vorgestellte Teilmodell “TSN Schedule” für jeden Switchport spezifiziert und mehrfach in der Verwaltungsschale bereitgestellt. Darüber hinaus stellt das dritte Teilmodell “Discovery” lokale Informationen zu den einzelnen Ports des TSN Switches zur Verfügung. Die bereitgestellten



VWS...Verwaltungsschale, SSH...Secure Shell, NDP...Neighbor Discovery Protocol,
SNMP...Simple Network Management Protocol, LLDP...Link Layer Discovery Protocol,
SDK...Software Development Kit

Abbildung 7.2: Zentraler SDN Controller zur dynamischen Konfiguration der echtzeitfähigen I4.0 Komponenten und TSN Switches.

Informationen basieren auf dem Link-Layer Discovery Protocol (LLDP). Die lokalen Informationen der einzelnen TSN Switches ermöglichen die Rekonstruktion der gesamten Netzwerkinfrastruktur und der entsprechenden Topologie durch den zentralen SDN Controller.

SDN Controller als I4.0 Komponente Basierend auf den TSN Switches als I4.0 Komponenten wird der zentrale SDN Controller ebenfalls als I4.0 Komponente mit einer Verwaltungsschale modelliert. Die spezifischen Fähigkeiten des SDN Controllers werden mit Hilfe der Teilmodelle "Switch Management" und "Echtzeitkommunikation" beschrieben (Bild 7.2). Das erste Teilmodell "Switch Management" dient zur Verwaltung der Netzwerkinfrastruktur und beinhaltet ein entsprechendes Array mit allen TSN Switches. Darüber hinaus werden zwei Operationen zur automatischen Discovery und zur Initialisierung der verfügbaren TSN Switches im Netzwerk definiert. Das zweite Teilmodell "Echtzeitkommunikation" des SDN Controllers dient als einheitliche Konfigurationsschnittstelle und ermöglicht das Starten bzw. Beenden von Echtzeitverbindungen (Bild 7.2). Die entsprechenden Operationen können zur Laufzeit von einer echtzeitfähigen I4.0 Komponente, einer Weboberfläche oder einem Engineering-Tool aufgerufen werden. Darüber hinaus beinhaltet das Teilmodell eine Liste der etablierten Echtzeitverbindungen.

Konfiguration der Echtzeitkommunikation Der zentrale SDN Controller bildet die Grundlage für die dynamische Konfiguration der Echtzeitverbindungen in der Kommunikationsinfrastruktur. Dabei konfiguriert der SDN Controller sowohl die Netzwerkinfrastruktur mit den TSN Switches als auch die echtzeitfähigen I4.0 Komponenten. Im Folgenden werden die erforderlichen Konfigurationsmechanismen zum Starten bzw. Beenden einer Echtzeitverbindung im Detail betrachtet und in den SDN Controller integriert. Darüber hinaus werden die zusätzlichen Mechanismen mit der Hilfe von weiteren Teilmodellen in der Verwaltungsschale spezifiziert. Dabei unterteilt sich die Netzwerkkonfiguration allgemein in die fünf identifizierten Teilaspekte der TSN-basierten Echtzeitkommunikation (Bild 7.1).

7.2 Zeitsynchronisation

Die Zeitsynchronisation ist der erste Teilaspekt der Echtzeitkommunikation und bildet die Grundlage für die TSN Verbindungen im Netzwerk. Zur Synchronisation der echtzeitfähigen I4.0 Komponenten und der TSN Switches in der Kommunikationsinfrastruktur ist eine einheitliche

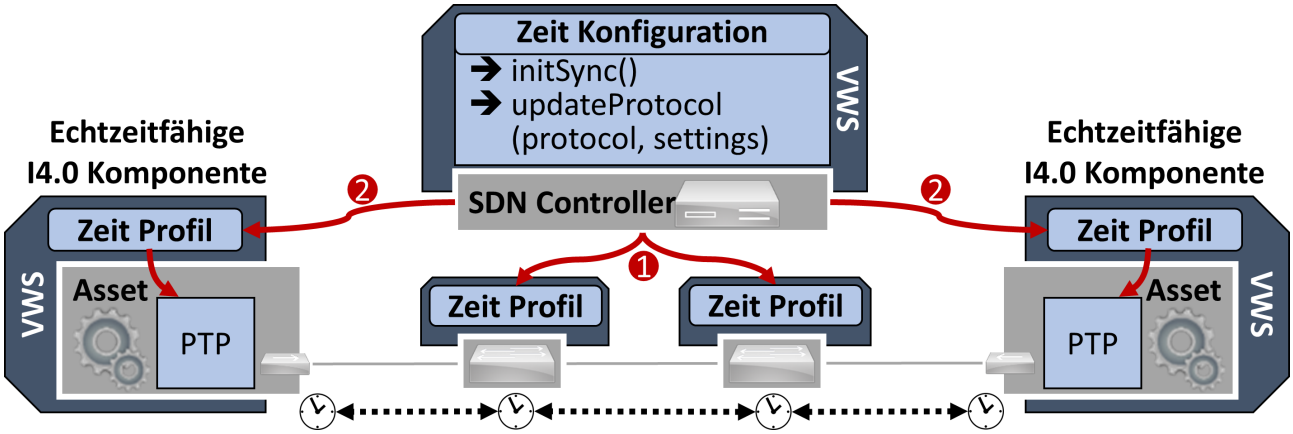


Abbildung 7.3: Die Konfiguration der Zeitsynchronisation im Netzwerk.

Konfiguration erforderlich. Diese Konfiguration erfolgt durch den zentralen SDN Controller, wobei die verfügbaren Operationen in einem Teilmodell beschrieben werden.

Teilmodell zur Konfiguration Das neue Teilmodell "Zeit Konfiguration" definiert zwei spezifische Operationen für die Zeitsynchronisation (Bild 7.3). Die erste Operation "initSync" ermöglicht die automatische Initialisierung aller Netzwerkkomponenten und Netzwerkteilnehmer. Dabei wird zunächst das "Zeit Profil" in der Verwaltungsschale der TSN Switches gesetzt und anschließend das gleiche Teilmodell in den angeschlossenen echtzeitfähigen I4.0 Komponenten parametriert. Basierend auf dem vorgestellten Lösungsansatz wird PTP mit dem spezifischen Profil IEEE 802.1AS spezifiziert (Norm IEEE 802.1AS). Dadurch wird eine einheitliche und für die Echtzeitkommunikation ausreichende Zeitsynchronisation gewährleistet. Darüber hinaus können auch individuelle Anpassungen an der Zeitsynchronisation vorgenommen werden. Dazu wird die zweite Operation "updateProtocol" in dem neuen Teilmodell aufgerufen und das gewünschte Protokoll bzw. die Protokolleinstellungen als Parameter übergeben. Analog zu der ersten Operation wird die gegebene Konfiguration von dem zentralen SDN Controller in der Kommunikationsinfrastruktur aktualisiert. Dabei muss allerdings gewährleistet sein, dass die abweichende Konfiguration von allen TSN Switches und den echtzeitfähigen I4.0 Komponenten unterstützt wird.

7.3 Dynamische TSN Verbindungen

Basierend auf der einheitlich konfigurierten Zeitsynchronisation wird im Folgenden die Konfiguration der dynamischen TSN Verbindungen betrachtet. Dabei werden die erforderlichen

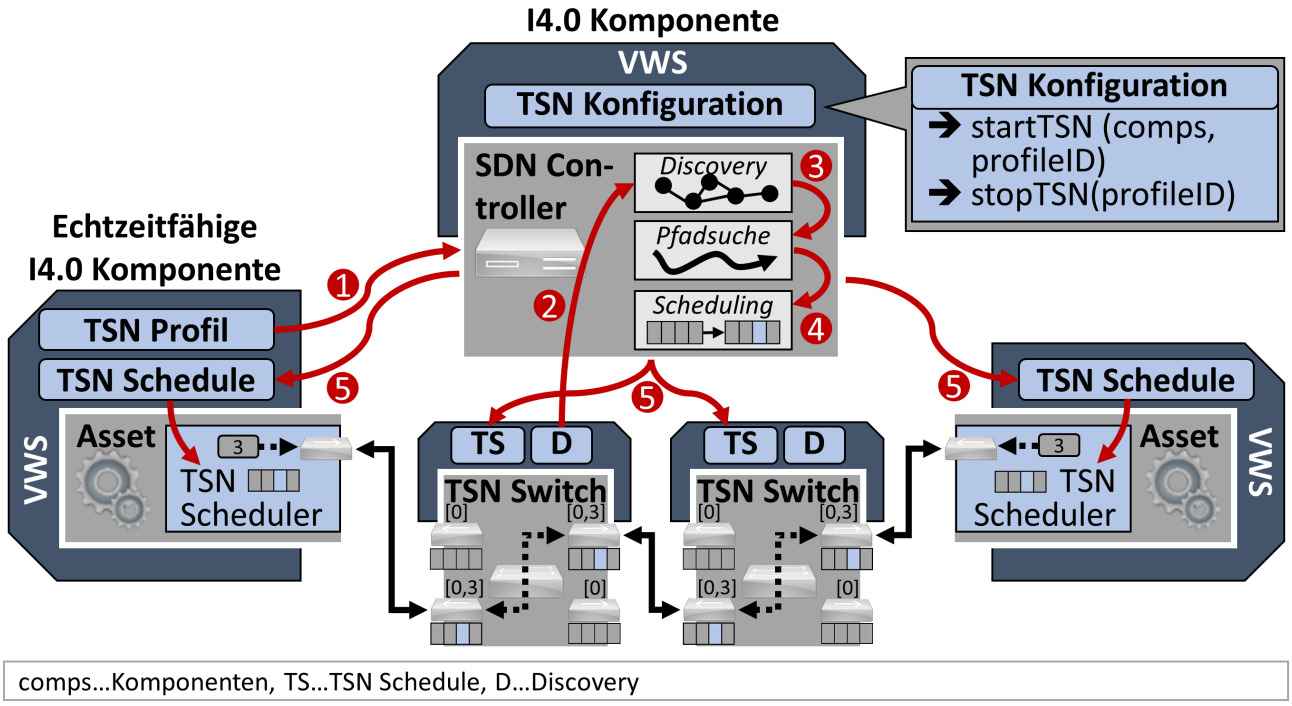


Abbildung 7.4: Die Konfiguration einer dynamischen TSN Verbindung durch den zentralen SDN Controller zur Laufzeit.

Mechanismen im Detail vorgestellt und in den zentralen SDN Controller integriert. Außerdem wird ein weiteres Teilmodell für die Verwaltungsschale des zentralen SDN Controllers definiert. Dieses Teilmodell "TSN Konfiguration" spezifiziert zwei Operationen zum Starten und Beenden von TSN Verbindungen zwischen echtzeitfähigen I4.0 Komponenten (Bild 7.4).

Starten von TSN Verbindungen Eine neue TSN Verbindung zwischen zwei echtzeitfähigen I4.0 Komponenten wird zur Laufzeit über die Operation "startTSN" initiiert. Dazu benötigt der zentrale SDN Controller zum einen die IDs der beiden echtzeitfähigen I4.0 Komponenten. Zum anderen wird die ID eines "TSN Profils" mit den Kommunikationsanforderungen als zweiter Parameter übergeben. Anschließend unterteilt sich die Konfiguration durch den SDN Controller in fünf wesentliche Schritte:

1. Zunächst wird das referenzierte Teilmodell "TSN Profil" aus der Verwaltungsschale der echtzeitfähigen I4.0 Komponente abgerufen. Dieses Profil spezifiziert unter anderem die VLAN-ID und Priorität der übertragenden TSN Frames sowie die maximale Zykluszeit bzw. Framegröße.

2. Im zweiten Schritt wird das Teilmodell “Discovery” aus der Verwaltungsschale der TSN Switches abgerufen und die lokalen Port-Informationen ausgewertet. Darauf aufbauend wird die gesamte physikalische Netzwerkinfrastruktur und insbesondere die aktuelle Netzwerktopologie rekonstruiert. Auf Grund der dynamischen Rahmenbedingungen muss diese Topologie bei jedem Operationsaufruf überprüft werden.

3. Basierend auf der aktuellen Netzwerktopologie wird ein Pfad zwischen den beiden echtzeitfähigen I4.0 Komponenten berechnet. Dabei wird ein möglichst kurzer Netzwerkpfad, d.h. mit möglichst wenigen TSN Switches, angestrebt. Der konkrete Algorithmus zur Pfadsuche wird im Abschnitt 7.3 detailliert vorgestellt.

4. Entlang des berechneten Netzwerkpfades erfolgt das TSN Scheduling in den TSN Switches und den beiden echtzeitfähigen I4.0 Komponenten. Dabei wird basierend auf der erforderlichen Zykluszeit und der spezifizierten Priorität ein Teil der verfügbaren Bandbreite reserviert. Der verwendete Scheduling Algorithmus im zentralen SDN Controller wird im Abschnitt 7.3 detailliert vorgestellt. Sofern kein valider Schedule für den Netzwerkpfad existiert, wird ein alternativer Pfad berechnet bzw. das Starten der TSN Verbindung abgebrochen.

5. Schließlich etabliert der SDN Controller den Netzwerkpfad in der Kommunikationsinfrastruktur. Dazu wird die spezifizierte VLAN-ID entlang dem Pfad gesetzt und die dazugehörigen TSN Schedules aktualisiert. Zur Aktualisierung verwendet der SDN Controller die Operation “updateSchedule” in den TSN Switches und den beiden echtzeitfähigen I4.0 Komponenten. Neben dem eigentlichen TSN Schedule wird dabei ein einheitlicher Startzeitpunkt als Parameter übergeben. Dieser Startzeitpunkt in Kombination mit der Zeitsynchronisation in der Kommunikationsinfrastruktur gewährleistet eine parallele Umstellung.

Nach der erfolgreichen Konfiguration der TSN Verbindung liefert die Operation im Teilmodell den Wert $true$ zurück, andernfalls $false$.

Beenden von TSN Verbindungen Analog zum Starten einer TSN Verbindung spezifiziert das neue Teilmodell eine zweite Operation “stopTSN” zum Beenden einer Verbindung. Dabei wird die “Profile-ID” (im Teilmodell “TSN Profil”) der existierenden TSN Verbindung als Parameter übergeben. Basierend auf der gegebenen ID identifiziert der zentrale SDN Controller die dazugehörige Reservierung im TSN Schedule und entfernt sie. Anschließend aktualisiert der Controller die beiden echtzeitfähigen I4.0 Komponenten und die TSN Switches entlang des

Netzwerkpfades. Dazu wird wieder die Operation “updateSchedule” aus den Teilmodell “TSN Schedule” in der Verwaltungsschale verwendet.

Rekonfiguration einer TSN Verbindung Darüber hinaus ermöglicht die Kombination der beiden Operationen “startTSN” und “stopTSN” die Rekonfiguration von einer existierenden TSN Verbindung zur Laufzeit. Dabei wird zunächst eine parallele TSN Verbindung etabliert (“startTSN”). Diese alternative Verbindung kann sich beispielsweise beim Netzwerkpfad, bei der Zykluszeit der Reservierung oder bei der Priorität von der existierenden TSN Verbindung unterscheiden. Die nahtlose Umstellung auf die neue Verbindung erfolgt durch die synchrone Aktualisierung der TSN Schedules in der Kommunikationsinfrastruktur. Sofern dabei der Zeitslot für die echtzeitkritischen TSN Frames konstant bleibt, ist die Umstellung für die eigentliche Echtzeitanwendung transparent. Andernfalls muss die Echtzeitanwendung in der I4.0 Komponente eine Rekonfiguration, d.h. insbesondere einen Taktwechsel zur Laufzeit, unterstützen. Zum Schluss wird die alte TSN Verbindung abgebaut und die doppelt reservierte Bandbreite wieder freigegeben.

Zusammengefasst ermöglicht das neue Teilmodell mit den beiden Operationen “startTSN” und “stopTSN” eine dynamische Konfiguration von TSN Verbindungen zur Laufzeit. Das allgemeine Vorgehen wurde dabei bereits beschrieben. Darauf aufbauend werden im Folgenden die einzelnen Algorithmen zur Pfadsuche und für das TSN Scheduling im Detail vorgestellt.

Algorithmus zur Pfadsuche

Der erste Algorithmus im SDN Controller berechnet den Netzwerkpfad für die dynamischen TSN Verbindungen basierend auf der aktuellen Kommunikationsinfrastruktur. Dazu wird die gesamte Infrastruktur mit den Netzwerkkomponenten und Netzwerkteilnehmern als ungerichteter, kantengewichteter Graph modelliert. Konkret wird jeder TSN Switch und jede (echtzeitfähige) I4.0 Komponente als Knoten dargestellt, die entsprechend der aktuellen Netzwerktopologie miteinander verbunden sind. Die Kantengewichte im Graphen werden so gewählt, dass der Pfad mit dem geringsten Gesamtgewicht dem bestmöglichen Pfad für die TSN Verbindung entspricht. Die Gewichtung wird nach folgenden drei Regeln bestimmt:

Kürzester Netzwerkpfad Jede Kante verfügt über ein strikt positives Kantengewicht (>0). Dadurch werden unnötige Kanten und insbesondere Schleifen auf dem Netzwerkpfad vermieden.

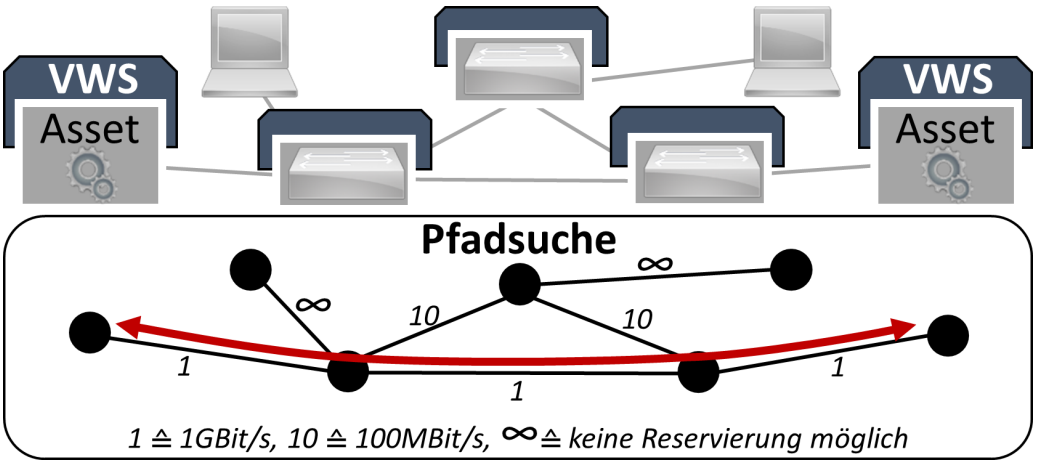


Abbildung 7.5: Pfadsuche für eine neue TSN Verbindung basierend auf einem kantengewichteten Graphen.

Schnellster Netzwerkpfad Jede Kante wird entsprechend der verfügbaren Datenrate skaliert, z.B. 10 für 100 Mbit/s und 1 für 1 Gbit/s. Dadurch wird eine schnellere Übertragung von TSN Frames bevorzugt und die Latenz reduziert.

Möglicher Netzwerkpfad Nicht-TSN-fähige oder ausgelastete Kanten werden mit dem Wert *unendlich* belegt und von der Pfadsuche ausgeschlossen. Dadurch wird ein valider Netzwerkpfad für die neue TSN Verbindung berechnet.

Die Anwendung dieser drei Regeln ist im Bild 7.5 skizziert. Dabei ist eine exemplarische Kommunikationsinfrastruktur mit dem dazugehörigen Graphen und den Kantengewichten dargestellt.

Der Netzwerkpfad für die TSN Verbindung entspricht dem Pfad mit dem geringsten Gesamtgewicht im Graphen und wird mit dem etablierten Dijkstra-Algorithmus berechnet (Dijkstra 1959). Dieser Algorithmus garantiert eine optimale Lösung und benötigt dazu eine polynomielle Laufzeit in Abhängigkeit der Anzahl TSN Switches in der Netzwerkinfrastruktur. Bei einer sehr großen Netzwerkinfrastruktur mit vielen TSN Switches kann auch eine effizientere Heuristik, wie z.B. der A*-Algorithmus (Hart et al. 1968), verwendet werden.

TSN Scheduling Algorithmus

Der zweite Algorithmus im SDN Controller berechnet die TSN Schedules für die echtzeitfähigen I4.0 Komponenten und TSN Switches in der Kommunikationsinfrastruktur. Dabei ist das TSN Scheduling in der Theorie NP-vollständig und skaliert mit der Anzahl TSN Switches und

Echtzeitverbindungen im Netzwerk. Auf Grund der Komplexität des mathematischen Problems wird ein heuristischer Ansatz verfolgt und ein Scheduling Algorithmus vorgestellt. Basierend auf den Anforderungen an die dynamische Netzwerkkonfiguration erfüllt dieser Algorithmus die folgenden drei Eigenschaften:

Umfassendes Scheduling im Produktionssystem Basierend auf den vielfältigen Kommunikationsanforderungen der echtzeitfähigen I4.0 Komponenten im Produktionssystem werden TSN Verbindungen mit unterschiedlichen Zykluszeiten unterstützt. Insbesondere werden harte Echtzeitanforderungen mit Latenzen unter einer Millisekunde (<1 ms) beim Scheduling berücksichtigt.

Schnelles Scheduling zur Laufzeit Im Kontext der dynamischen Netzwerkkonfiguration erfolgt das TSN Scheduling zur Laufzeit und in einer angemessenen Zeit. Auf Basis heutiger Ethernet-Netzwerk ist eine Konfigurationszeit innerhalb von wenigen Sekunden (<3 s) realistisch.

Effizientes Scheduling von Bandbreite Das Reservieren von Netzwerkbandbreite erfolgt dynamisch zur Laufzeit. Folglich werden insbesondere keine statischen Reservierungen für den priorisierten Datenverkehr verwendet, d.h. der gewöhnliche Datenverkehr wird nur bei Bedarf eingeschränkt.

Basierend auf diesen drei wesentlichen Eigenschaften eignet sich der neue TSN Scheduling Algorithmus besonders für eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur. Im Folgenden wird der verwendete Algorithmus detailliert erläutert und an einem Beispiel verdeutlicht (Bild 7.6).

Scheduling Schema Zunächst definiert der zentrale SDN Controller eine globale Zykluszeit. Diese Zeit ist für die gesamte Kommunikationsinfrastruktur, insbesondere alle TSN Switches, einheitlich und vereinfacht die Konfiguration. Darauf aufbauend wird die globale Zykluszeit in gleich große Zeitslots unterteilt. Dabei ist die Anzahl Zeitslots üblicherweise ein Vielfaches von zwei, sodass die Zykluszeit jeweils halbiert wird. Während sich die globale Zykluszeit an den geringsten Echtzeitanforderungen orientiert, richtet sich die Anzahl von Zeitslots nach den höchsten Anforderungen. Das Bild 7.6 zeigt ein Beispiel mit einer globalen Zykluszeit von 500 µs und insgesamt 16 Zeitslots für eine minimale Zykluszeit von 125 µs.

Zu Beginn sind alle Zeitslots der Priorität 0 (Best-Effort) zugeordnet und stehen dem gewöhnlichen Datenverkehr zur Verfügung. Beim Starten einer TSN Verbindung ordnet der SDN Controller einzelne Zeitslots einer höheren Priorität für die echtzeitkritischen TSN Frames zu.

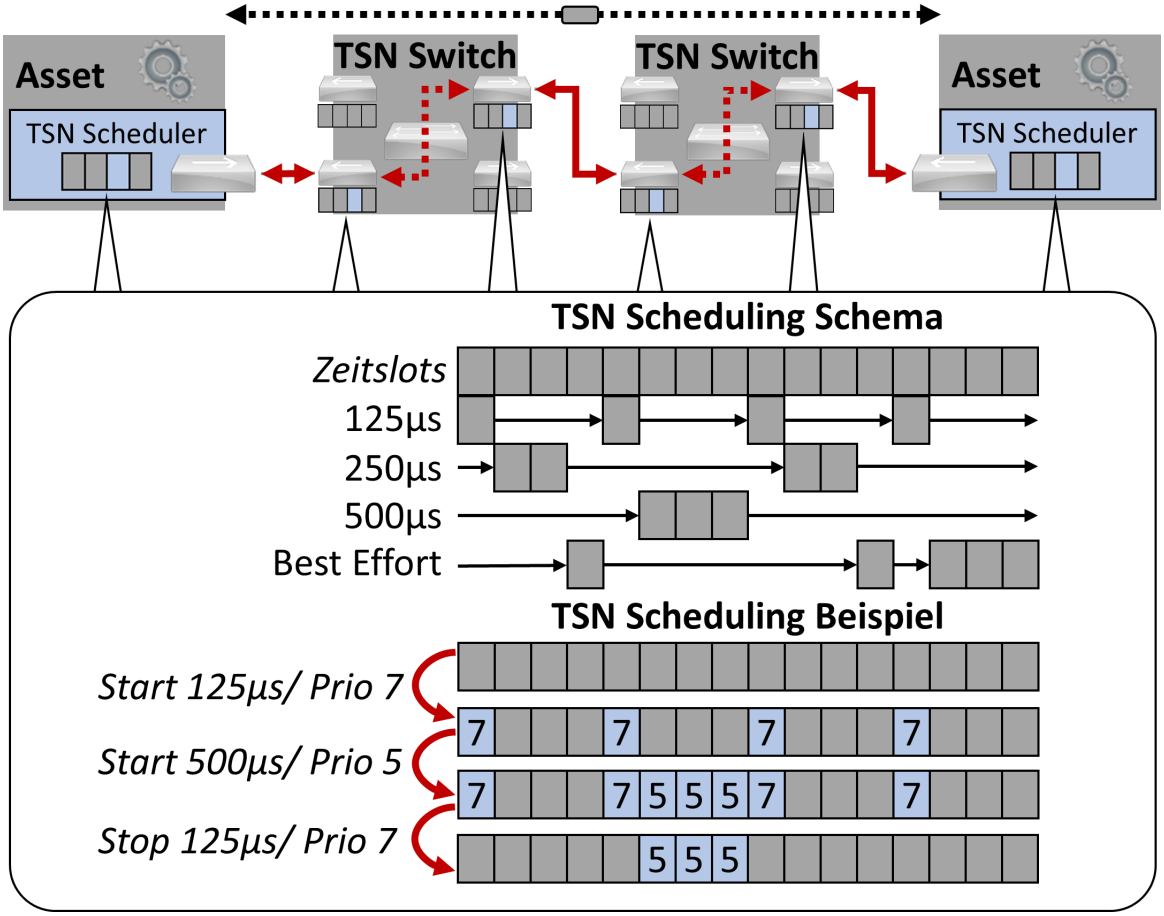


Abbildung 7.6: Das angewendete Scheduling Schema für die TSN Verbindungen und ein Beispiel mit drei dynamischen Reservierungen.

Die konkrete Priorität der reservierten Zeitslots ist im Teilmodell “TSN Profil” festgelegt. Darüber hinaus ist die erforderliche Zykluszeit und die maximale Framegröße ebenfalls im Profil definiert. Diese beiden Anforderungen bestimmen die Anzahl (aufeinanderfolgender) Zeitslots in dem TSN Schedule. Dabei wird die Übertragung von TSN Frames mit maximaler Größe (plus Jitter und Puffer) innerhalb der geforderten Zykluszeit gewährleistet. Beim Beenden einer TSN Verbindung werden die entsprechenden Zeitslots wieder freigeben, d.h. der Priorität 0 (Best-Effort) zugeordnet. Ein Beispiel für das TSN Scheduling mit zwei verschiedenen TSN Verbindungen (Priorität 5/7) ist im Bild 7.6 dargestellt.

Dynamische Konfiguration mit Timeout

Beim Starten und Beenden von TSN Verbindungen aktualisiert der zentrale SDN Controller die TSN Schedules in der Kommunikationsinfrastruktur. Dabei entsteht zusätzlicher Konfigurationsdatenverkehr zwischen dem Controller und den TSN Switches bzw. den echtzeitfähigen I4.0 Komponenten. Um den Konfigurationsdatenverkehr zu beschränken, wird die Operation “stopTSN” um einen Timeout erweitert (d.h. “stopTSN(profileID, timeout)”). Dieser Timeout wird als zusätzlicher Parameter übergeben und verzögert das Beenden der TSN Verbindung. Konkret verfährt der zentrale SDN Controller wie folgt (Bild 7.7):

Keine neue Verbindung Standardmäßig wartet der zentrale SDN Controller die gegebene Zeit ab und beendet anschließend die TSN Verbindung.

Gleiche Verbindung Sofern während dem Timeout die Operation “startTSN” für die gleiche TSN Verbindung aufgerufen wird, bricht der SDN Controller das Beenden ab und setzt die Verbindung unverändert fort. Dabei entsteht insbesondere kein zusätzlicher Konfigurationsdatenverkehr.

Andere Verbindung Sofern die reservierten Zeitslots für eine andere TSN Verbindung benötigt werden, bricht der SDN Controller den Timeout ab und beendet die TSN Verbindung sofort. Dadurch wird der Konfigurationsdatenverkehr der beiden TSN Verbindungen zusammengefasst.

Die zusätzlichen Mechanismen im SDN Controller ermöglichen eine effizientere Netzwerkkonfiguration in der Kommunikationsinfrastruktur. Ein sinnvoller Timeout hängt dabei maßgeblich von der Anzahl TSN Switches in der Netzwerkinfrastruktur und der Anzahl etablierter TSN Verbindungen ab.

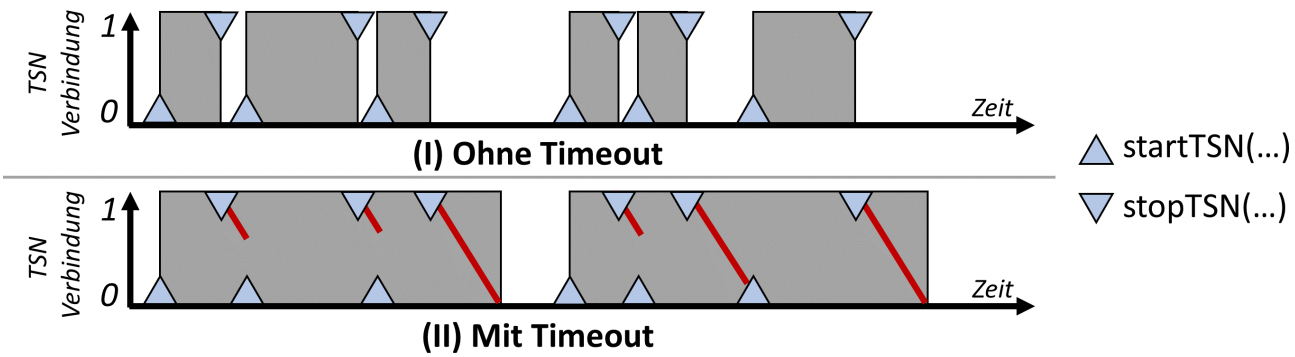


Abbildung 7.7: TSN Verbindungen mit Timeout zur Reduzierung des Konfigurationsaufwandes im Netzwerk.

Zusammengefasst ermöglicht der erweiterte SDN Controller die Konfiguration der dynamischen TSN Verbindungen zur Laufzeit. Dabei wurden die erforderlichen Mechanismen und die verwendeten Algorithmen in dem zentralen SDN Controller im Detail erläutert. Darüber hinaus wurde das zusätzliche Teilmodell "TSN Konfiguration" für die Verwaltungsschale des SDN Controllers definiert.

7.4 Redundante TSN Verbindungen

Im vorherigen Abschnitt wurde die dynamische Konfiguration von TSN Verbindungen im zentralen SDN Controller vorgestellt. Darauf aufbauend wird die Konfiguration von redundanten TSN Verbindungen zwischen zwei echtzeitfähigen I4.0 Komponenten betrachtet. Dazu werden die erforderlichen Mechanismen in den SDN Controller integriert und in einem zusätzlichen Teilmodell spezifiziert.

Teilmodell zur Konfiguration Das Teilmodell "Redundanz Konfiguration" wird zur Verwaltungsschale des SDN Controllers hinzugefügt und beinhaltet insgesamt zwei Operationen (Bild 7.8). Die erste Operation "startRConn" etabliert eine redundante TSN Verbindung. Dazu benötigt der zentrale SDN Controller die "Profile-ID" der existierenden, primären TSN Verbindung als Parameter. Darauf aufbauend wird der Operationsaufruf wie folgt umgesetzt:

1. Basierend auf der gegeben ID wird das "TSN Profil" der primären TSN Verbindung abgerufen. Dieses Teilmodell beinhaltet die Kommunikationsanforderungen der echtzeitfähigen I4.0 Komponenten, wie z.B. die maximale Zykluszeit und die Priorität der echtzeitkriti-

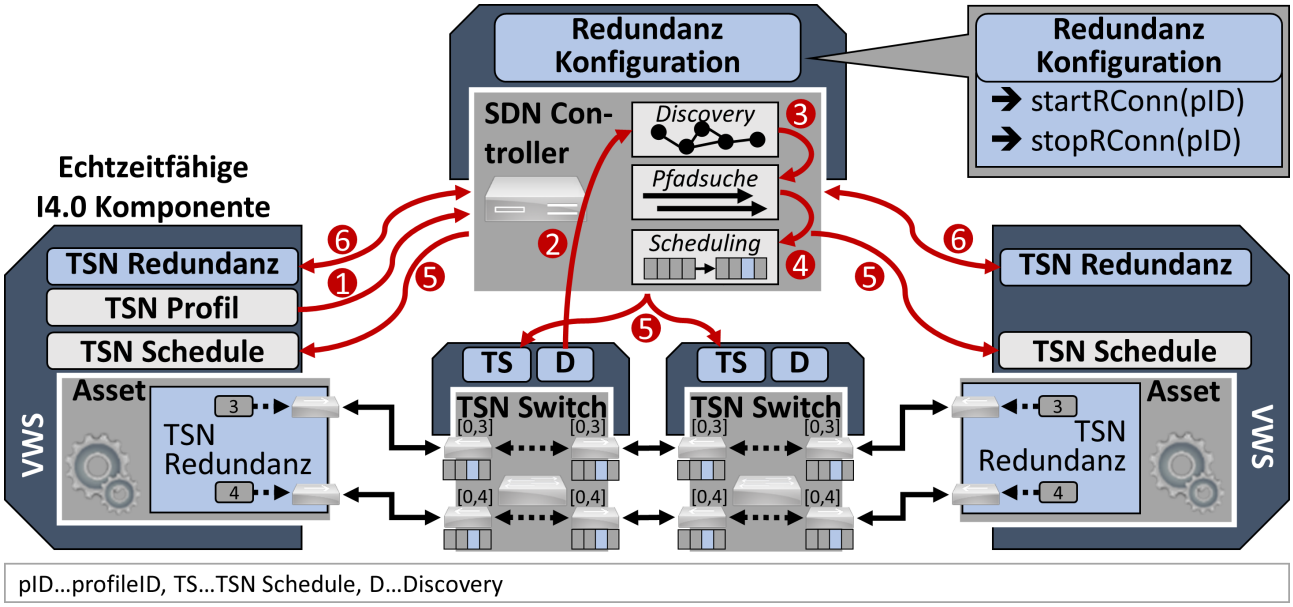


Abbildung 7.8: Die Konfiguration einer redundanten TSN Verbindung durch den zentralen SDN Controller.

schen Frames. Für die redundante TSN Verbindung müssen die gleichen Kommunikationsanforderungen erfüllt werden.

2. Analog zu den dynamischen TSN Verbindungen wird die Netzwerktopologie bei jedem Operationsaufruf verifiziert. Dazu verwendet der SDN Controller die lokalen Informationen der TSN Switches, d.h. das Teilmodell “Discovery”.

3. Basierend auf der aktuellen Netzwerktopologie wird ein redundanter Netzwerkpfad zwischen den beiden echtzeitfähigen I4.0 Komponenten berechnet. Dazu wird die Pfadsuche für die dynamischen TSN Verbindungen modifiziert und die Gewichtung im Graphen angepasst. Konkret werden die Kantengewichte der primären TSN Verbindung auf den Wert 1000 gesetzt (Bild 7.9). Auf Grund der sehr hohen Gewichtung wird eine entsprechende Kante nur dann für den redundanten Pfad ausgewählt, wenn keine andere Kante möglich ist. Dadurch gewährleistet der resultierende Netzwerkpfad eine bestmögliche Ausfallsicherheit.

4. Anschließend wird der TSN Schedule für den redundanten Netzwerkpfad berechnet. Dabei wird der Scheduling Algorithmus für die dynamischen TSN Verbindungen wiederverwendet (Abschnitt 7.3).

5. Sofern ein valider TSN Schedule gefunden wurde, etabliert der SDN Controller die redundante TSN Verbindung in der Kommunikationsinfrastruktur. Dabei wird zunächst eine

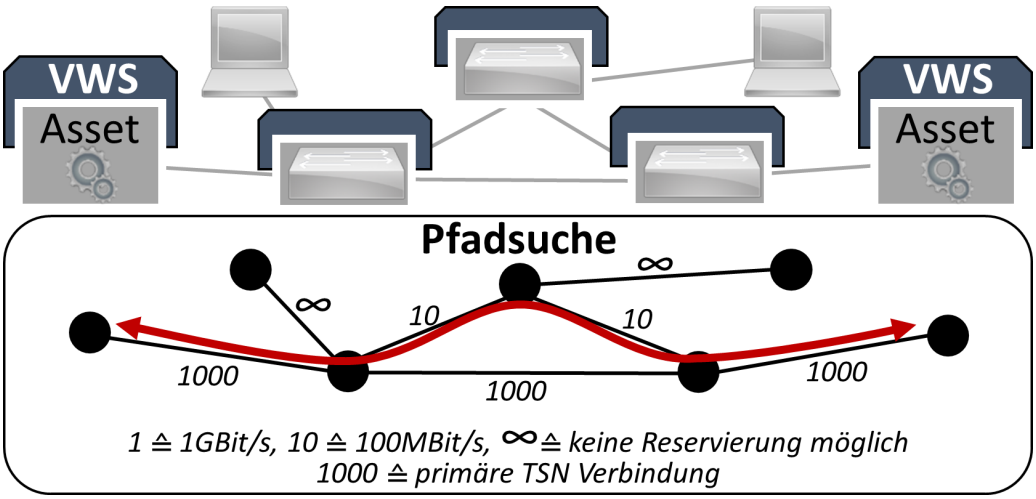


Abbildung 7.9: Pfadsuche für eine redundante TSN Verbindung basierend auf einem kantengewichteten Graphen.

neue VLAN-ID definiert und entlang des Netzwerkpfades gesetzt. Diese VLAN-ID dient zur Unterscheidung der primären und redundanten TSN Verbindung bzw. der dazugehörigen Frames. Anschließend aktualisiert der SDN Controller die TSN Schedules entlang des Netzwerkpfades. Dazu verwendet der SDN Controller die Operation "updateSchedule" in den TSN Switches und den beiden echtzeitfähigen I4.0 Komponenten.

6. Schließlich werden die Redundanzmechanismen in den beiden echtzeitfähigen I4.0 Komponenten über das Teilmodell "TSN Redundanz" parametriert und aktiviert. Basierend auf dem etablierten Netzwerkpfad wird beispielsweise die Netzwerkschnittstelle und die VLAN-ID für die redundanten TSN Frames spezifiziert.

Neben der Operation zum Starten einer redundanten TSN Verbindung, existiert eine weitere Operation "stopRConn" zum Beenden einer Verbindung. Das Vorgehen ist dabei identisch zum Beenden einer dynamischen TSN Verbindung (Abschnitt 7.3).

Rekonfiguration einer redundanten TSN Verbindung Redundante TSN Verbindungen kompensieren Hardware- oder Softwarefehler in der Kommunikationsinfrastruktur. Dabei werden echtzeitkritische TSN Frames dupliziert und parallel über die primäre bzw. redundante TSN Verbindung übertragen. Bei einem Fehler auf einer der beiden Verbindungen existieren zwei Reaktionsmöglichkeiten.

Sofern eine Teilverbindung nur temporär von einem Fehler betroffen ist, wird die redundante Echtzeitkommunikation anschließend unverändert fortgesetzt. Dazu sind keine zusätzlichen

Mechanismen in den echtzeitfähigen I4.0 Komponenten oder dem zentralen SDN Controller erforderlich.

Bei einem permanenten Fehler muss der zentrale SDN Controller die redundante TSN Verbindung zur Laufzeit rekonfigurieren. Dazu etabliert der SDN Controller eine neue Verbindung ("startRConn") und berechnet insbesondere einen alternativen Netzwerkpfad. Anschließend wird das Teilmodell "TSN Redundanz" in den echtzeitfähigen I4.0 Komponenten aktualisiert und die VLAN-ID der neuen Verbindung gesetzt. Schließlich werden die Reservierungen der vorherigen Verbindung in der Kommunikationsinfrastruktur entfernt ("stopRConn").

Zusammengefasst ermöglicht der erweiterte SDN Controllern die dynamische Konfiguration von redundanten TSN Verbindungen zur Laufzeit. Dazu wurden die zusätzlichen Konfigurationsmechanismen in den SDN Controller integriert und ein weiteres Teilmodell für die Verwaltungsschale definiert.

7.5 Virtuelle Netzwerktopologien

Die Vernetzung von mehreren echtzeitfähigen I4.0 Komponenten ist der vierte Teilaspekt der vorgestellten Echtzeitkommunikation. Dabei wurde das Konzept der virtuellen Netzwerktopologien präsentiert und die erforderlichen Kommunikationsmechanismen in die I4.0 Komponenten integriert. Zur Konfiguration dieser Topologien wird der zentrale SDN Controller im Folgenden erweitert und ein zusätzliches Teilmodell für die Verwaltungsschale definiert.

Teilmodell zur Konfiguration Das neue Teilmodell "VT Konfiguration" spezifiziert drei Operationen zum Starten, Beenden und Aktualisieren einer virtuellen Netzwerktopologie. Die erste Operation "startVT" initiiert eine neue Topologie zur Laufzeit. Dabei werden die dazugehörigen echtzeitfähigen I4.0 Komponenten und die angestrebte Netzwerktopologie (d.h. Linien- oder Ring-Topologie) als Parameter übergeben. Darüber hinaus werden die Kommunikationsanforderungen über ein "TSN Profil" referenziert. Der zentrale SDN Controller nimmt den Operationsaufruf entgegen und konfiguriert die virtuelle Netzwerktopologie wie folgt (Bild 7.10):

1. Der SDN Controller ruft das referenzierte Teilmodell "TSN Profil" aus der Verwaltungsschale ab. Dieses Profil spezifiziert die Kommunikationsanforderungen an die virtuelle Netzwerktopologie, wie z.B. die maximale Zykluszeit, Framegröße und Priorität der TSN Frames.

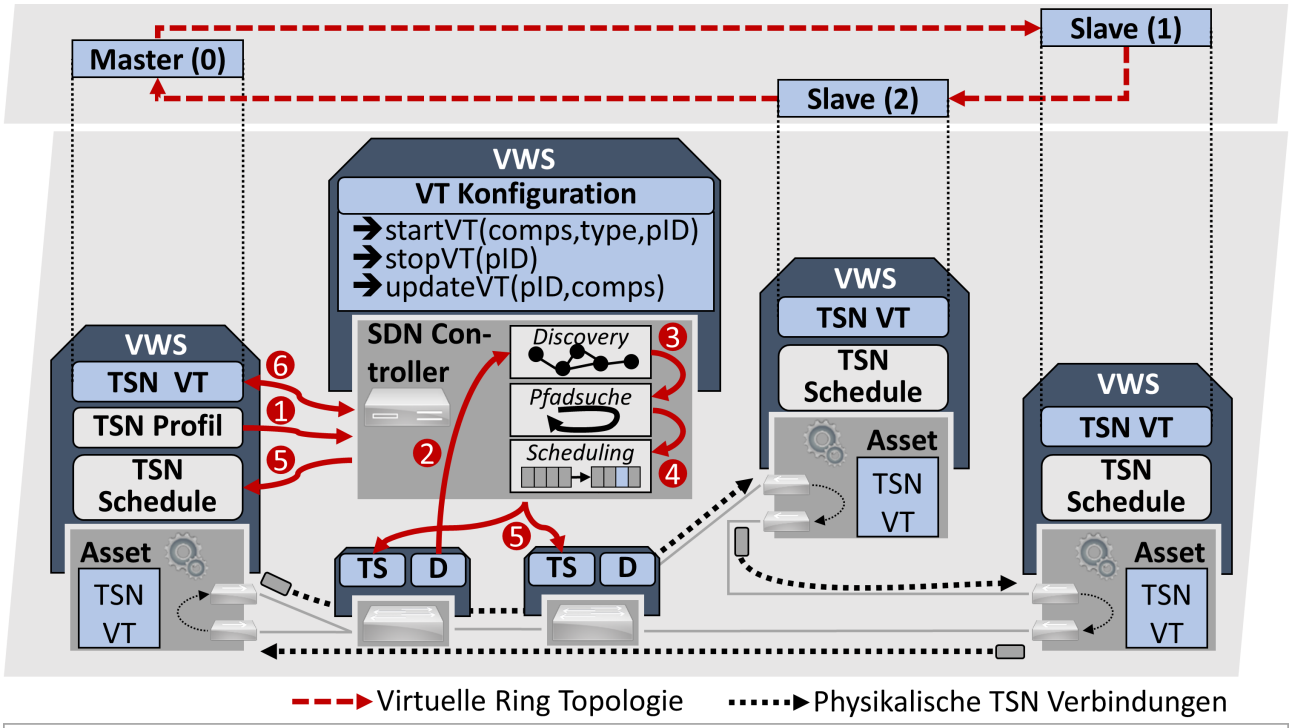


Abbildung 7.10: Die Konfiguration einer virtuellen Netzwerktopologie basierend auf einzelnen TSN Verbindungen durch den zentralen SDN Controller.

2. Der zweite Schritt ist identisch zu den dynamischen TSN Verbindungen (Abschnitt 7.3). Dabei wird das Teilmodell “Discovery” mit den lokalen Informationen der TSN Switches abgerufen und die Netzwerktopologie aktualisiert.

3. Anschließend berechnet der SDN Controller einen Netzwerkpfad entlang der gegebenen echtzeitfähigen I4.0 Komponenten. Dieser Pfad entspricht einer Verkettung von einzelnen TSN Verbindungen zwischen zwei aufeinanderfolgenden I4.0 Komponenten (Bild 7.11). Daher wird zur Pfadsuche der vorgestellte Algorithmus für die dynamischen TSN Verbindungen verwendet (Abschnitt 7.3).

4. Basierend auf dem berechneten Netzwerkpfad konfiguriert der SDN Controller die aufeinanderfolgenden TSN Verbindungen. Um die Gesamtlatenz der virtuellen Netzwerktopologie zu gewährleisten, wird jede einzelne TSN Verbindung mit einer kürzeren Zykluszeit konfiguriert (Bild 7.11). Konkret darf die Summe der Zykluszeiten die maximale Zykluszeit aus dem “TSN Profil” nicht überschreiten. Für das eigentliche Reservieren der Zeitslots verwendet der SDN Controller den vorgestellten TSN Scheduling Algorithmus (Abschnitt 7.3).

5. Sofern ein valider TSN Schedule für die einzelnen TSN Verbindungen in der virtuellen Netzwerktopologie gefunden wurde, aktualisiert der SDN Controller die Kommunikationsinfrastruktur. Dazu wird die Operation “scheduleUpdate” in den TSN Switches und allen beteiligten echtzeitfähigen I4.0 Komponenten aufgerufen.

6. Schließlich werden die beteiligten echtzeitfähigen I4.0 Komponenten für die virtuelle Netzwertopologie konfiguriert. Dazu aktiviert der SDN Controller die entsprechenden Kommunikationsmechanismen im TSN SDK und parametriert das Teilmodell “TSN VT”. Konkret werden die Konfigurationsparameter für die virtuelle Topologie (“active”,“topologyID”, “topologyType”), für die I4.0 Komponente selbst (“index”, “vtInterface”) sowie für die nachfolgende I4.0 Komponente (“successor”) spezifiziert.

Neben dem Starten einer virtuellen Netzwerktopologie definiert das neue Teilmodell “VT Konfiguration” eine weitere Operation “stopVT” zum Beenden. Dabei wird die “Profile-ID” der existierenden Topologie als Parameter übergeben und alle dazugehörigen TSN Verbindungen in der Kommunikationsinfrastruktur beendet. Das Beenden der einzelnen TSN Verbindungen erfolgt analog zu den dynamischen TSN Verbindungen (Abschnitt 7.3).

Rekonfiguration einer virtuellen Netzwerktopologie Darüber hinaus ermöglicht der zentrale SDN Controller die Rekonfiguration einer virtuellen Netzwerktopologie zur Laufzeit.

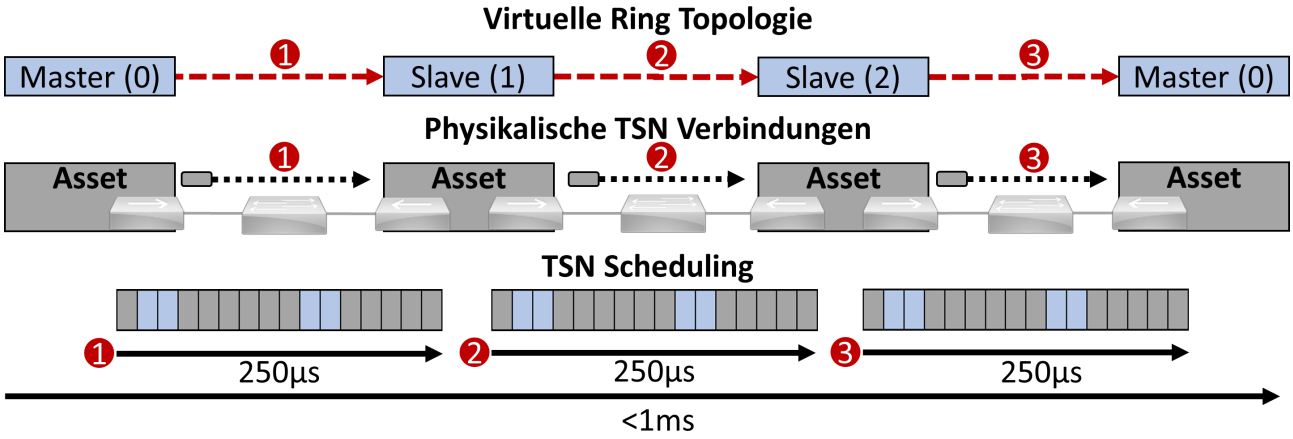


Abbildung 7.11: Das TSN Scheduling für die virtuelle Netzwerktopologie.

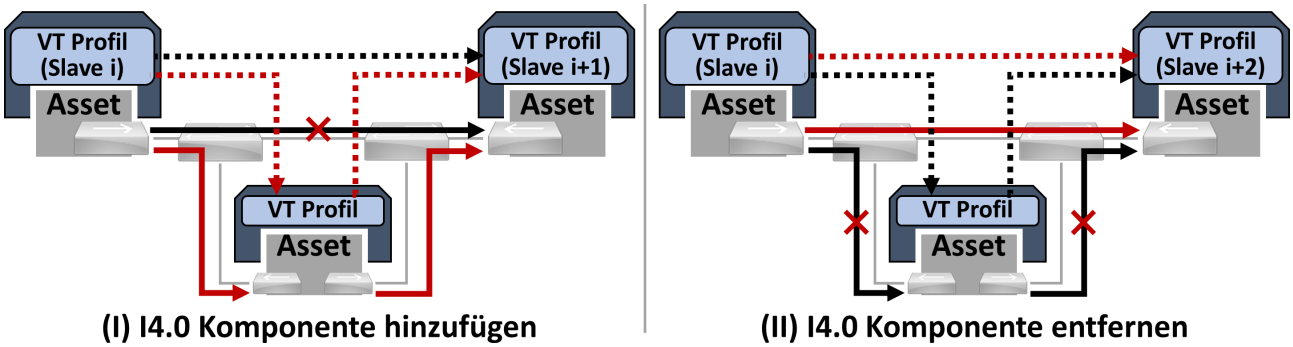


Abbildung 7.12: Rekonfiguration einer virtuellen Netzwerktopologie zur Laufzeit.

Eine Rekonfiguration ist beispielsweise notwendig, wenn eine echtzeitfähige I4.0 Komponente hinzukommt bzw. wegfällt oder wenn sich die physikalische Netzwerkinfrastruktur mit den TSN Switches signifikant ändert. Dabei werden im Gegensatz zum Starten oder Beenden nur einzelne TSN Verbindungen in der virtuellen Netzwerktopologie angepasst.

Zur Rekonfiguration spezifiziert das neue Teilmodell im SDN Controller eine dritte Operation “updateVT”. Diese Operation benötigt zum einen die “Profile-ID” einer existierenden virtuellen Topologie. Zum anderen wird eine aktualisierte Liste der echtzeitfähigen I4.0 Komponenten als Parameter übergeben. Basierend auf diesen Informationen modifiziert der SDN Controller die existierende virtuelle Netzwerktopologie. Dabei ist das konkrete Vorgehen beim Hinzufügen bzw. Entfernen von echtzeitfähigen I4.0 Komponenten im Bild 7.12 skizziert. Zur Rekonfiguration der einzelnen TSN Verbindungen verwendet der SDN Controller jeweils die Operationen “startTSN” und “stopTSN” aus dem Teilmodell “TSN Konfiguration” (Abschnitt 7.3). Darüber hinaus wird das Teilmodell “TSN VT” in den beteiligten echtzeitfähigen I4.0 Komponenten angepasst.

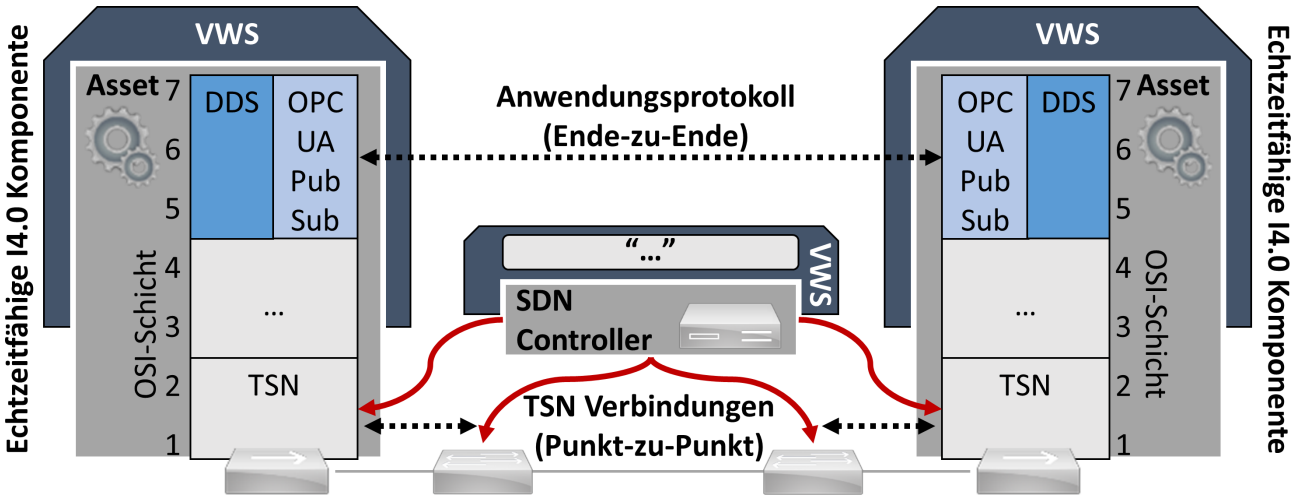


Abbildung 7.13: Im Gegensatz zu den TSN Verbindungen benötigt die Echtzeitkommunikation auf den darüberliegenden OSI-Schichten mit den Anwendungsprotokollen keine Konfiguration der Netzwerkinfrastruktur.

Zusammengefasst spezifiziert das neue Teilmodell drei Operationen zur dynamischen Konfiguration von virtuellen Netzwerktopologien zur Laufzeit. Die dazugehörigen Konfigurationsmechanismen wurden in den zentralen SDN Controller integriert und im Detail erläutert.

7.6 Anwendungsprotokolle

In den vorherigen Abschnitten wurde ein zentraler SDN Controller erweitert und die dynamische Konfiguration von TSN Verbindungen im Netzwerk vorgestellt. Die resultierenden TSN Verbindungen definieren die echtzeitfähige Datenübertragung auf den OSI-Schichten 1-2, wobei es sich um eine sogenannte Punkt-zu-Punkt Kommunikation handelt (Bild 7.13). Darauf aufbauend spezifizieren die Anwendungsprotokolle die Datenübertragung auf den OSI-Schichten 5-7 und ermöglichen eine durchgängige Echtzeitkommunikation. Auf den höheren OSI-Schichten handelt es sich allerdings um eine Ende-zu-Ende Kommunikation.

Ende-zu-Ende Kommunikation Eine Ende-zu-Ende Kommunikation bezeichnet die indirekte Datenübertragung zwischen dem Sender und Empfänger, d.h. den beiden echtzeitfähigen I4.0 Komponenten (Bild 7.13). Dabei ist die Übertragung insbesondere transparent für die gesamte Netzwerkinfrastruktur mit den TSN Switches. Diese TSN Switches verwenden ausschließlich den TSN Header zum Weiterleiten von Frames und beachten die darüberliegenden Protokolle nicht. Erst der Empfänger wertet auch die höheren OSI Schichten inklusive dem

Anwendungsprotokoll und den anwendungsspezifischen Daten aus. Für die dynamische Konfiguration der Echtzeitverbindungen im Netzwerk ergeben sich dadurch zwei Konsequenzen. Zum einen ist keine Konfiguration des Anwendungsprotokolls in den TSN Switches notwendig. Zum anderen erfolgt die Parametrierung der Anwendungsprotokolle ausschließlich in den echtzeitfähigen I4.0 Komponenten und über die entsprechenden Teilmodelle in der Verwaltungsschale. Daher sind keine zusätzlichen Konfigurationsmechanismen im zentralen SDN Controller erforderlich.

7.7 Zusammenfassung

In diesem Kapitel wurde ein zentraler SDN Controller erweitert und die erforderlichen Konfigurationsmechanismen für die TSN-basierte Echtzeitkommunikation hinzugefügt. Der erweiterte SDN Controller etabliert dynamische, redundante und komplexe TSN Verbindungen zwischen verschiedenen echtzeitfähigen I4.0 Komponenten und ermöglicht insbesondere eine dynamische Konfiguration zur Laufzeit. Die dazu notwendigen Konfigurationsmechanismen wurden im Detail vorgestellt und in Form von zusätzlichen Teilmodellen in der Verwaltungsschale beschrieben (Bild 7.14). Der resultierende SDN Controller konfiguriert sowohl die Kommunikationsmechanismen in den echtzeitfähigen I4.0 Komponenten als auch die TSN Switches in der Netzwerkinfrastruktur. Dabei wurden die TSN Switches zur einheitlichen Konfiguration ebenfalls als I4.0 Komponenten mit einer Verwaltungsschale modelliert. Zusammengefasst ermöglicht der vorgestellte SDN Controller eine dynamische Konfiguration der TSN-basierten Echtzeitkommunikation in der neuen Kommunikationsinfrastruktur.

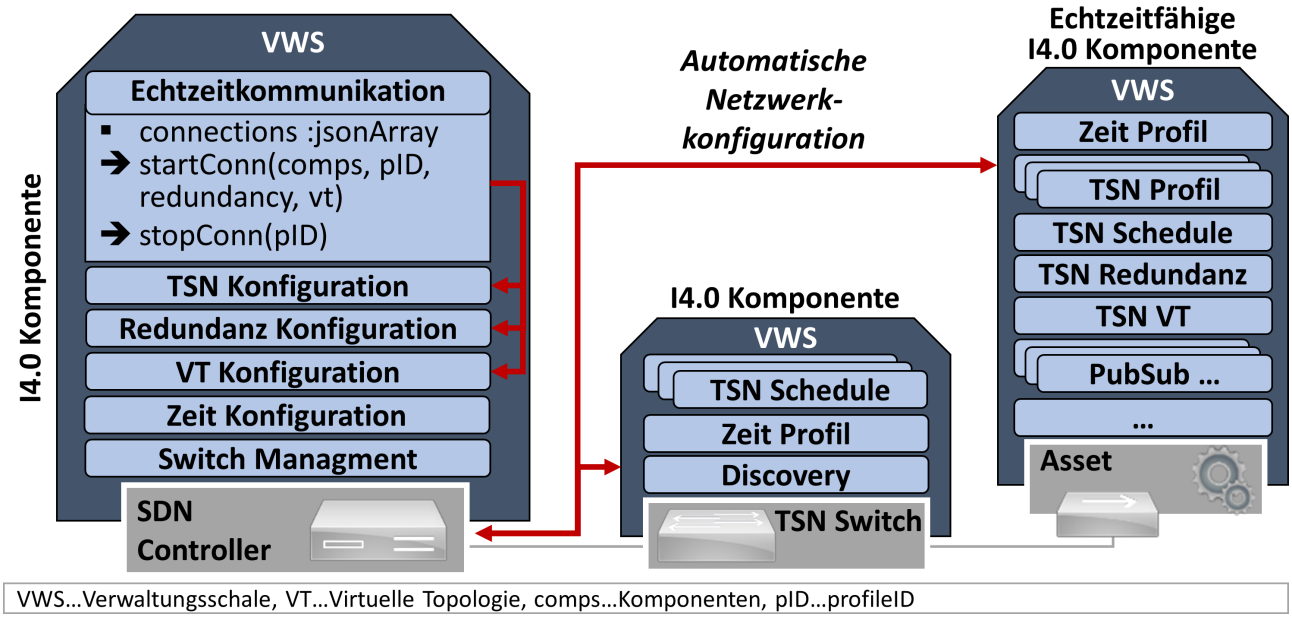


Abbildung 7.14: Zentraler SDN Controller zur dynamischen Konfiguration der TSN Switches und echtzeitfähigen I4.0 Komponenten.

Kapitel 8

Realisierung

In dieser Arbeit wurde das Konzept einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur vorgestellt. Dazu wurde zunächst die TSN-basierte Echtzeitkommunikation erarbeitet (Kapitel 5). Anschließend wurde die notwendige Parametrierung in den echtzeitfähigen Assets durch eine einheitliche Komponentenbeschreibung ermöglicht (Kapitel 6). Schlussendlich wurde ein zentraler SDN Controller zur dynamischen Netzwerkkonfiguration erweitert und die zusätzlichen TSN-spezifischen Konfigurationsmechanismen vorgestellt (Kapitel 7). Im Folgenden wird das neue Konzept basierend auf einer prototypischen Implementierung evaluiert.

8.1 Prototypische Implementierung

Aufbau

Um die Machbarkeit zu zeigen wurde die vorgestellte wandelbare, echtzeitfähige Kommunikationsinfrastruktur prototypisch implementiert. Ziel der prototypischen Implementierung ist eine TSN-basierte Bewegungssteuerung (Motion Control). Dabei werden zwei echtzeitfähige Achsen von einer zentralen Motion Control synchron angesteuert. Im Gegensatz zu einer gewöhnlichen Motion Control sind die einzelnen Assets nicht direkt miteinander verbunden, sondern jeweils an die TSN-basierte Kommunikationsinfrastruktur angeschlossen. Dynamische Echtzeitverbindungen zwischen den Assets ermöglichen die deterministische Datenübertragung mit geringer Latenz und gewährleisten eine synchrone Ansteuerung.

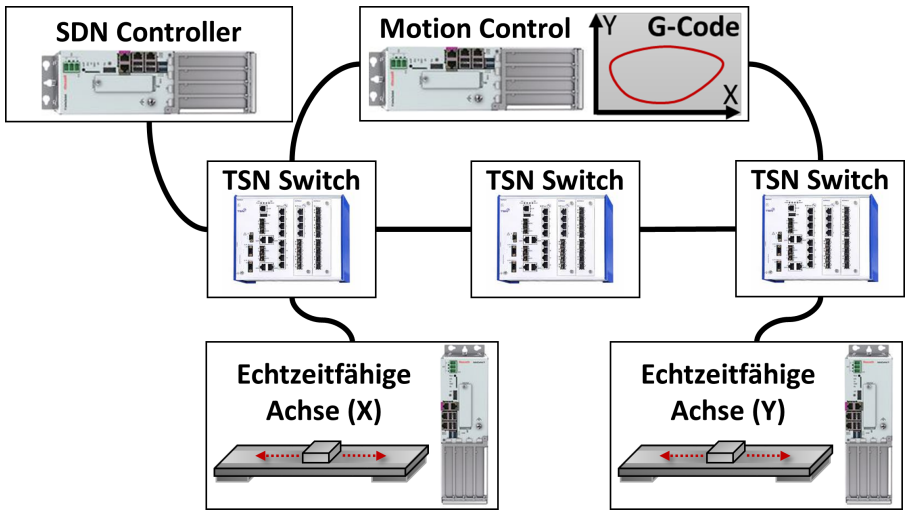


Abbildung 8.1: Prototypischer Aufbau mit TSN Switches, einem erweiterten SDN Controller, einer Motion Control und echtzeitfähigen Achsen (Abbildungen (Rexroth 2020; Belden 2020a)).

Insgesamt umfasst der prototypische Aufbau vier Industrie-PCs (IPCs), die über mehrere TSN Switches miteinander verbunden sind. Die IPCs verfügen über verschiedene (Echtzeit-)Anwendungen und repräsentieren entweder einen SDN Controller, eine zentrale Motion Control oder eine echtzeitfähige Achse (Bild 8.1).

TSN Switches Die Netzwerkinfrastruktur besteht aus mehreren TSN Switches vom Typ "Hirschmann RSPE 35" (Belden 2020b). Diese Switches verfügen über eine prototypische Firmware, die neben den etablierten Ethernet Mechanismen auch ein Port-basiertes Scheduling entsprechend dem TSN Standard IEEE 802.1Qbv unterstützt (Norm IEEE 802.1Qbv). Darüber hinaus ist jeder TSN Switch als I4.0 Komponente modelliert, sodass eine Verwaltungsschale zur herstellerunabhängigen Konfiguration bereitgestellt wird (Bild 8.2). Die Kommunikation zwischen der Software-basierten Verwaltungsschale und der eigentlichen Switch Hardware ist dabei über SSH realisiert. Die Verwaltungsschale beinhaltet die definierten Teilmodelle "Zeit Profil" und "Discovery". Außerdem wird der TSN Schedule für die einzelnen Ports jeweils über ein Teilmodell "TSN Schedule" konfiguriert.

SDN Controller Ein IPC vom Typ "Bosch Rexroth IndraControl VPB 40.3" repräsentiert den zentralen SDN Controller zur dynamischen Netzwerkkonfiguration (Rexroth 2020). Basierend auf dem Betriebssystem Linux 4.16.8 ist eine Anwendung mit der gesamten Konfigurationslogik für die Netzwerkkomponenten (d.h. die TSN Switches) und die Netzwerkteilnehmer (d.h. die echtzeitfähigen I4.0 Komponenten) integriert (Linux 2020). Die

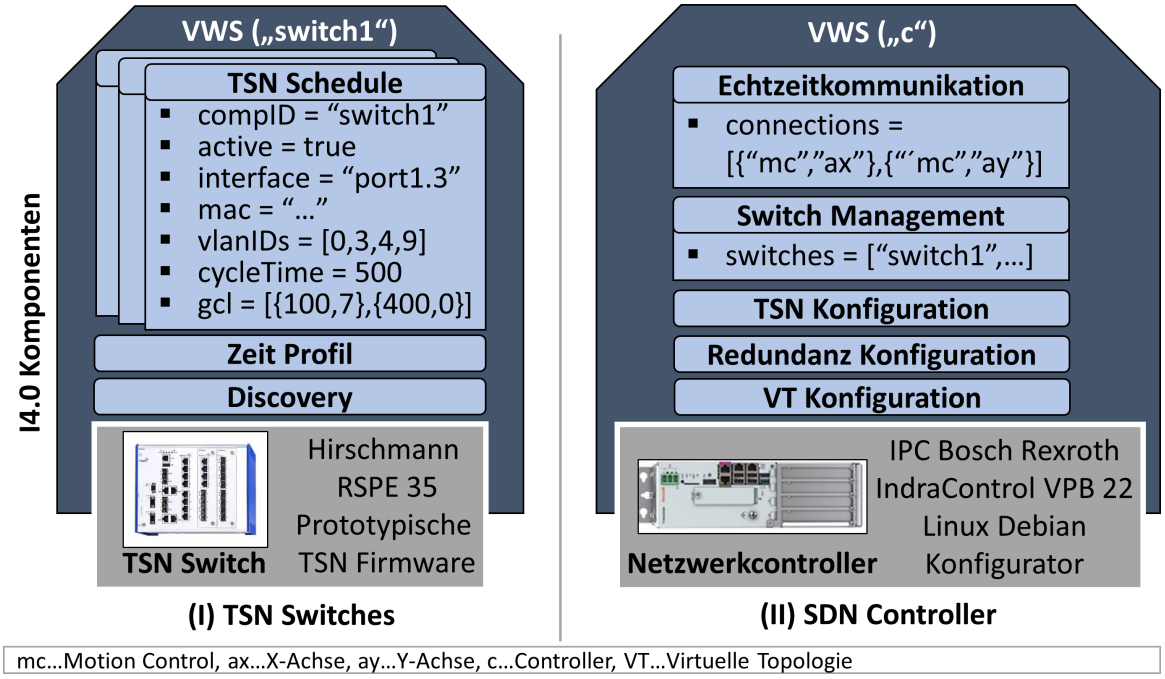


Abbildung 8.2: TSN Switch und zentraler SDN Controller mit den dazugehörigen Verwaltungsschalen.

Konfigurationslogik umfasst dabei insbesondere die vorgestellten Mechanismen zur Netzwerkdiscovery, Pfadsuche und das TSN Scheduling. Darüber hinaus ist der zentrale SDN Controller als I4.0 Komponente modelliert und stellt eine eigene Verwaltungsschale zur Verfügung (Bild 8.2). Diese Verwaltungsschale beinhaltet die definierten Teilmodelle zur Netzwerkkonfiguration, wie z.B. “TSN/ Redundanz/ VT Konfiguration”. Das zusätzliche Teilmodell “Echtzeitkommunikation” fasst die spezifischen Konfigurationsmechanismen zusammen und dient als generische Schnittstelle für die echtzeitfähigen I4.0 Komponenten.

Echtzeitfähige Achsen Zwei weitere IPCs vom Typ “Bosch Rexroth IndraControl VPB 40.3” repräsentieren jeweils eine echtzeitfähige Achse in dem prototypischen Aufbau (Rexroth 2020). Auf Grund der Echtzeitanforderungen verfügen beide IPCs über das Betriebssystem Linux 4.16.8 mit RT-Preempt-Patch und TBS-Patch (Linux 2020; Sanchez-Palencia 2018). Darauf aufbauend realisiert das vorgestellte TSN SDK die TSN-konforme Echtzeitkommunikation. Dabei wird das erforderliche Time-Triggered Send von der verwendeten Netzwerkkarte “Intel I210 NIC” unterstützt (Intel 2020). Basierend auf dem TSN SDK ermöglicht ein prototypisches SDK für das Anwendungsprotokoll OPC UA PubSub die durchgängige Echtzeitkommunikation.

In dem prototypischen Aufbau ist jede echtzeitfähige Achse als echtzeitfähige I4.0 Komponente mit Verwaltungsschale modelliert (Bild 8.3). Diese Verwaltungsschale beinhaltet die neuen TSN-spezifischen Teilmodelle zur Parametrierung des TSN SDKs, wie z.B. das Teilmodell "Zeit Profil" und "TSN Redundanz". Darüber hinaus spezifizieren die Teilmodelle "TSN Profil" die Kommunikationsanforderungen der echtzeitfähigen Achse. Neben den TSN-spezifischen Teilmodellen beinhaltet die Verwaltungsschale auch ein Asset-spezifisches Teilmodell "Achse". Dieses Teilmodell ermöglicht die Initialisierung und Ansteuerung der echtzeitfähigen Achse über ihre Verwaltungsschale. Dazu sind die aktuelle Soll- und Ist-Position als Eigenschaften definiert und zwei weitere Operationen spezifiziert.

Motion Control Der vierte IPC im prototypischen Aufbau repräsentiert die Motion Control zur synchronisierten Ansteuerung der echtzeitfähigen Achsen. Dazu verfügt der IPC ("Bosch Rexroth IndraControl VPB 40.3" (Rexroth 2020)) über ein echtzeitfähiges Betriebssystem (Linux 4.16.8 mit RT-Preempt-Patch und TBS-Patch (Linux 2020; Sanchez-Palencia 2018)), das vorgestellte TSN SDK und zwei Netzwerkkarten ("Intel I210 NIC" (Intel 2020)) zur redundanten Übertragung von TSN Frames. Als Echtzeitanwendung ist eine 2D-Bahnsteuerung für die beiden echtzeitfähigen Achsen integriert. Diese Anwendung berechnet zyklisch die Soll-Positionen für die Achsen basierend auf einem gegebenen G-Code Programm und überträgt diese an die Achsen.

Die Motion Control ist ebenfalls als echtzeitfähige I4.0 Komponente mit einem Asset-spezifischen Teilmodell "Motion Control" modelliert (Bild 8.3). Dieses Teilmodell spezifiziert zwei Operationen zum Starten und Beenden der Bahnsteuerung. Als Parameter werden dabei die IDs der beiden echtzeitfähigen Achsen sowie das gewünschte G-Code Programm übergeben. Darüber hinaus beinhaltet die Verwaltungsschale alle spezifischen Teilmodelle zur Parametrierung der Echtzeitkommunikation.

Zusammengefasst sind alle (echtzeitfähigen) Netzwerkkomponenten und Netzwerkteilnehmer in dem prototypischen Aufbau als (echtzeitfähige) I4.0 Komponenten mit Verwaltungsschale modelliert (Bild 8.4). Die Verwaltungsschale spezifiziert dabei die individuellen Fähigkeiten und ermöglicht die einheitliche Konfiguration der TSN-basierten Echtzeitkommunikation.

Physikalische und virtuelle Achsen

Der prototypische Aufbau umfasst zwei IPCs, die eine echtzeitfähige Achse als echtzeitfähige I4.0 Komponente repräsentieren. Dabei existieren zwei verschiedene Varianten von echtzeitfä-

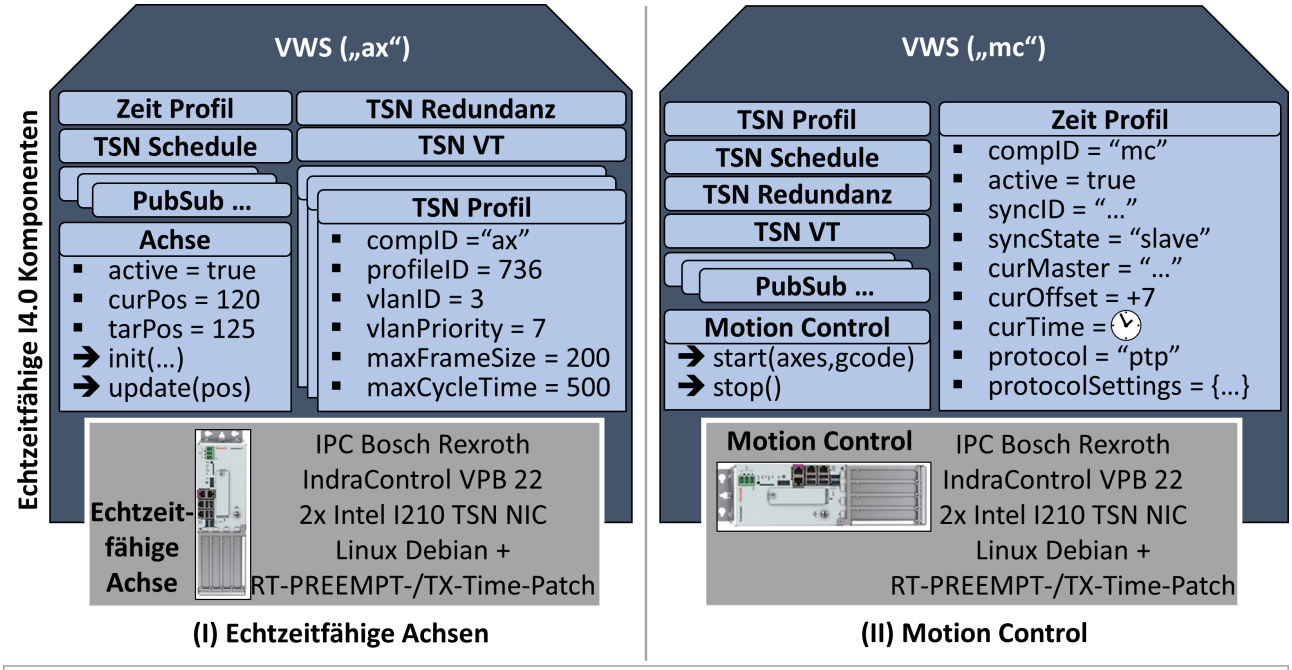


Abbildung 8.3: Echtzeitfähige Achse und zentrale Motion Control mit den Echtzeit-spezifischen und Asset-spezifischen Teilmodellen in der Verwaltungsschale.

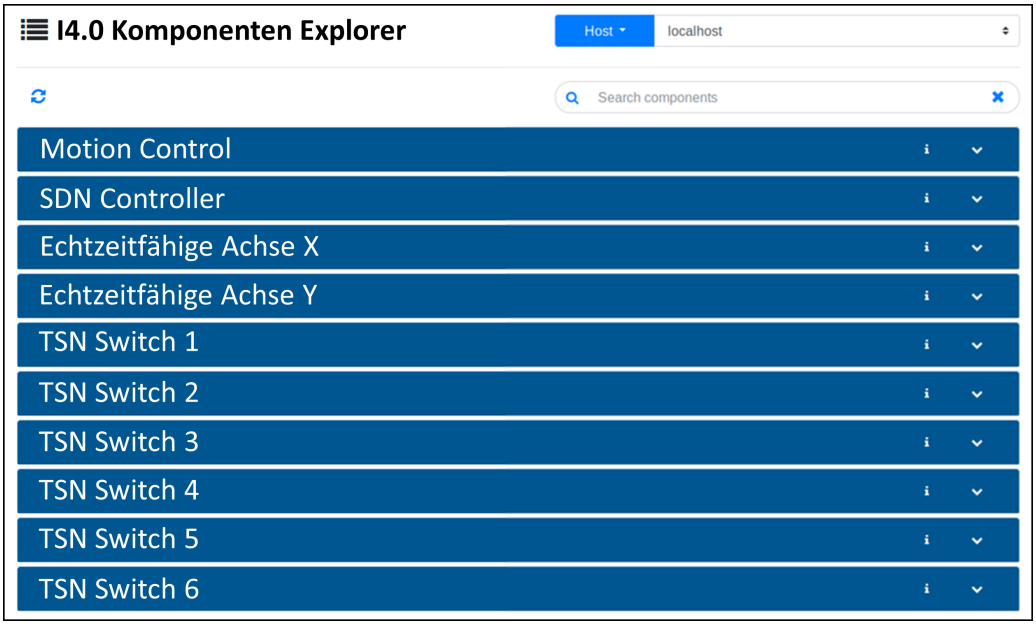


Abbildung 8.4: Grafische Weboberfläche mit einer Übersicht über alle verfügbaren (echtzeitfähigen) I4.0 Komponenten im Demonstrator.

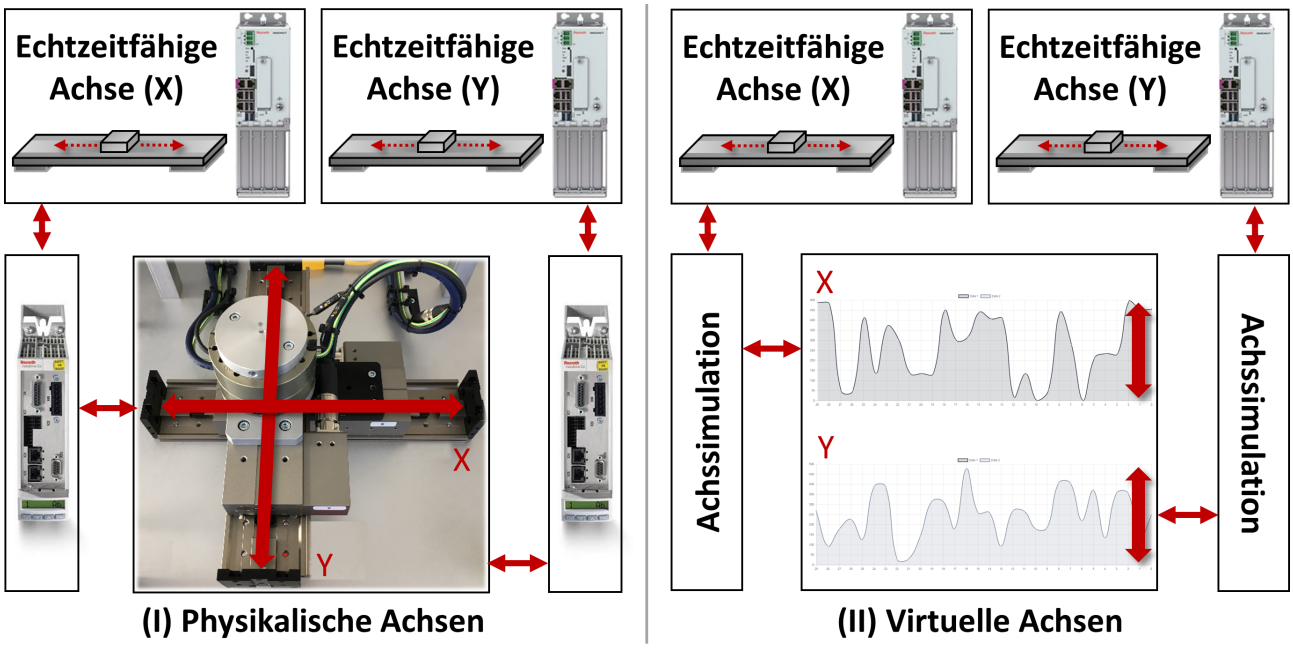


Abbildung 8.5: Prototypischer Aufbau mit konventionellen physikalischen Achsen und virtuellen Achsen zur Simulation.

higen Achsen um die Vorteile der vorgestellten wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur zu demonstrieren (Bild 8.5).

Physikalische Achsen Die erste Variante demonstriert die Machbarkeit einer TSN-basierten Motion Control. Dabei sind die beiden IPCs für die echtzeitfähigen Achsen mit einem zweidimensionalen Achssystem verbunden. Dieses Achssystem besteht aus zwei orthogonal zueinanderstehenden Linearachsen (X-/Y-Achse) und dazugehörigen Antriebsverstärkern. Ein solcher Antriebsverstärker setzt die Soll-Position im IPC unmittelbar in eine physikalische Bewegung der jeweiligen Linearachse um. Gleichzeitig wird die aktuelle Position der physikalischen Achse mit der Ist-Position im IPC in Echtzeit synchronisiert. Dabei sind sowohl die Soll- als auch die Ist-Position im IPC direkt mit den entsprechenden Eigenschaften in der Verwaltungsschale der echtzeitfähigen I4.0 Komponente verknüpft.

Virtuelle Achsen Im Gegensatz zu den physikalischen Achsen werden bei der zweiten Variante ausschließlich virtuelle Achsen verwendet. Dazu verfügen die beiden IPCs für die echtzeitfähigen Achsen über eine Echtzeitanwendung mit einer Achssimulation. Diese Anwendung simuliert die Bewegung einer physikalischen Achse in Echtzeit und synchronisiert die Soll-/Ist-Position mit den entsprechenden Eigenschaften in der Verwaltungsschale. Basierend auf den virtuellen Achsen kann die Wandelbarkeit und Skalierbarkeit der vorgestellten Kommunikationsinfrastruktur gezeigt werden. Dabei ist im Gegensatz zu den physikali-

schen Achsen das Hinzufügen oder Entfernen von beliebig vielen echtzeitfähigen Achsen zur Laufzeit möglich.

Zusammengefasst sind sowohl physikalische als auch virtuelle Achsen in dem prototypischen Aufbau möglich. Unabhängig von der konkreten Varianten sind die Achsen als echtzeitfähige I4.0 Komponenten mit Verwaltungsschale modelliert. Diese Verwaltungsschale beinhaltet insbesondere die gleichen Teilmodelle und ermöglicht somit eine einheitliche Ansteuerung durch die zentrale Motion Control.

TSN-basierte Motion Control

Der vorgestellte prototypische Aufbau ermöglicht eine TSN-basierte Motion Control, bei der die beiden echtzeitfähigen Achsen synchronisiert angesteuert werden. Die echtzeitkritische Kommunikation zwischen der Motion Control und den beiden Achsen erfolgt zyklisch mit einer definierten Zykluszeit zwischen 250 µs und 1 ms. Dabei empfängt die Motion Control die aktuellen Ist-Positionen der Achsen, berechnet neue Soll-Positionen und sendet diese an die Achsen zurück. Dadurch wird ein Regelkreis etabliert und eine synchrone Ansteuerung der echtzeitfähigen Achsen gewährleistet. Die vorgestellte wandelbare, echtzeitfähige Kommunikationsinfrastruktur garantiert die deterministische Datenübertragung zwischen der zentralen Motion Control und den beiden echtzeitfähigen Achsen. Dabei unterteilt sich die etablierte Echtzeitkommunikation in die fünf identifizierten Teilaspekte. Im Folgenden wird die Anwendung der verschiedenen Teilaspekte am Beispiel der TSN-basierten Motion Control erläutert.

Zeitsynchronisation Die Zeitsynchronisation bildet die Grundlage für die TSN-basierte Echtzeitkommunikation (Bild 8.6). Dazu verfügen alle TSN Switches in dem prototypischen Aufbau über eine standardkonforme PTP Implementierung und unterstützen das Profil IEEE 802.1AS (Belden 2020a). Ein Switch fungiert dabei als Master und definiert die globale Netzwerkzeit. Bei den echtzeitfähigen I4.0 Komponenten, d.h. der Motion Control und den beiden echtzeitfähigen Achsen, ist die PTP Implementierung in dem TSN SDK integriert. Die Kompatibilität im Netzwerk wird durch die einheitliche Parametrierung über das Teilmodell “Zeit Profil” in der Verwaltungsschale gewährleistet. Basierend auf der globalen Netzwerkzeit synchronisieren sowohl die TSN Switches als auch die echtzeitfähigen I4.0 Komponenten ihre lokale Systemzeit.

Dynamische TSN Verbindungen Basierend auf der Zeitsynchronisation etabliert der zentrale SDN Controller zwei dynamische TSN Verbindungen zur Laufzeit (Bild 8.6). Dabei wird

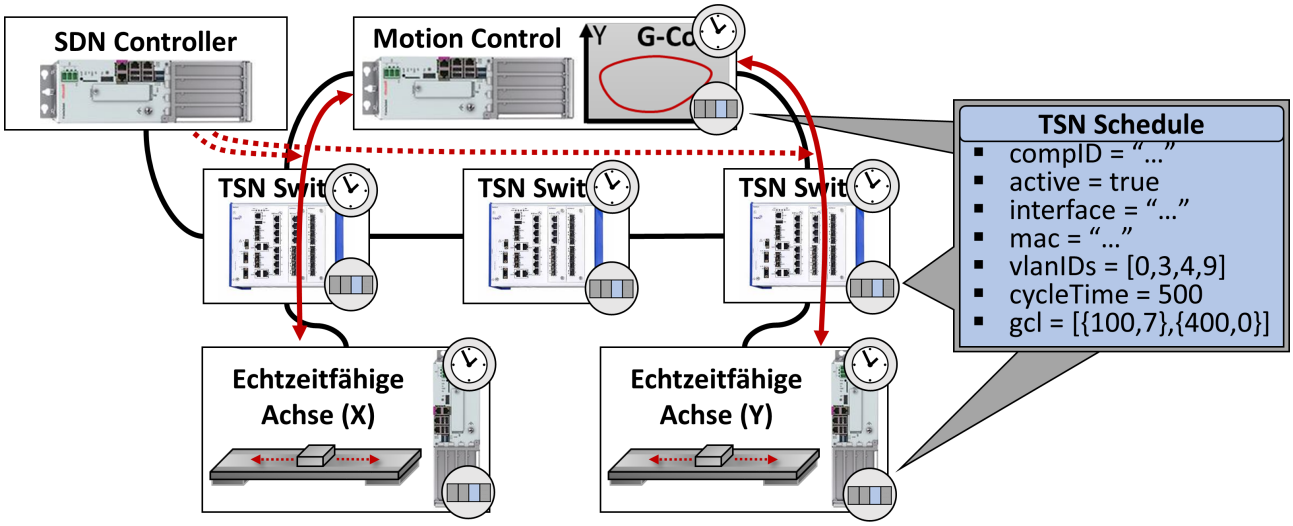


Abbildung 8.6: Basierend auf der Zeitsynchronisation werden dynamische TSN Verbindungen zwischen der Motion Control und den echtzeitfähigen Achsen etabliert.

jeweils eine Verbindung zwischen der Motion Control und einer echtzeitfähigen Achse etabliert. Die Anforderungen an die neuen Echtzeitverbindungen werden vom "TSN Profil" mit der ID 736 beschrieben (Bild 8.3). Dieses Teilmodell definiert insbesondere die VLAN-ID 3 und die Priorität 7 für die übertragenen TSN Frames. Darüber hinaus ist für das TSN Scheduling eine maximale Framegröße von 200 Bytes und eine Zykluszeit von 500 µs spezifiziert. Der resultierende, vom SDN Controller berechnete Schedule hat ebenfalls eine Zykluszeit von 500 µs, wobei die ersten 100 µs für den echtzeitkritischen Datenverkehr mit Priorität 7 reserviert sind. Der neue Schedule wird vom SDN Controller in den echtzeitfähigen I4.0 Komponenten und den TSN Switches auf dem Netzwerkpfad gesetzt. Dazu wird die Operation "updateSchedule" aus dem Teilmodell "TSN Schedule" verwendet und die Zykluszeit bzw. die Gate-Control-List entsprechend aktualisiert.

Redundante TSN Verbindungen Am Beispiel der Motion Control wird eine existierende TSN Verbindung erweitert und eine redundante TSN Verbindung etabliert (Bild 8.7). Dazu ist der IPC mit der Motion Control über zwei Netzwerkadapter mit verschiedenen TSN Switches verbunden. Der zentrale SDN Controller konfiguriert die redundante TSN Verbindung zur Laufzeit, wobei die Anforderungen und das Vorgehen identisch zu der primären Verbindung sind. Für eine bestmögliche Redundanz wird bei der Pfadsuche allerdings ein möglichst disjunkter Netzwerkpfad berechnet. Darüber hinaus parametriert der SDN Controller die Redundanzmechanismen in der Motion Control und den echtzeitfähigen Achsen, d.h. aktualisiert das Teilmodell "TSN Redundanz". Dabei werden insbesondere die Netzwerkschnittstelle sowie die VLAN-IDs für die primären und redundanten TSN

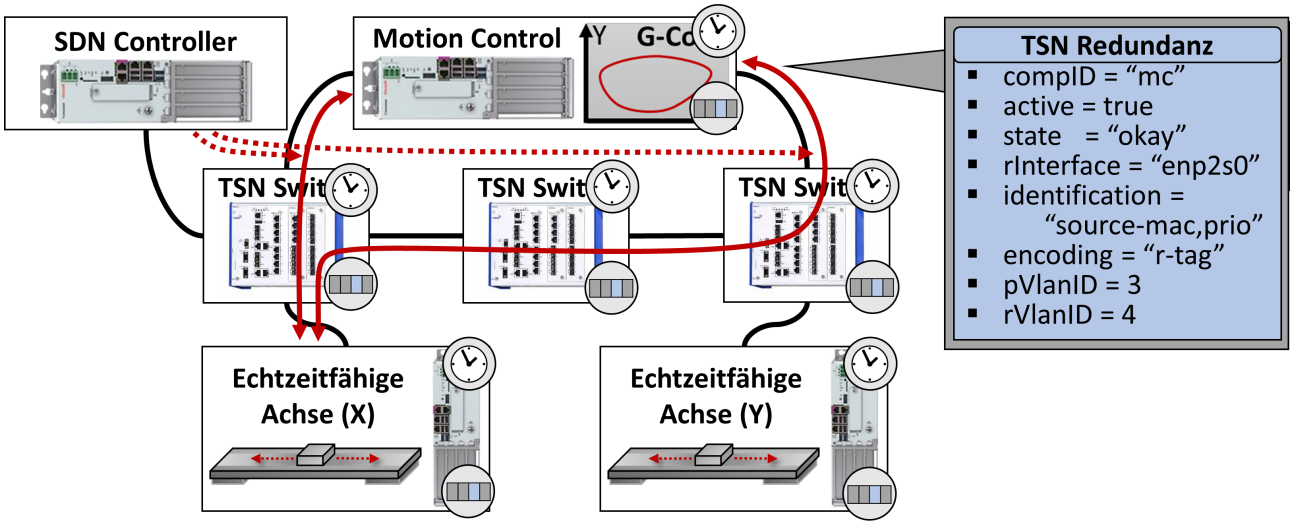


Abbildung 8.7: Redundante TSN Verbindungen zwischen der Motion Control und einer echtzeitfähigen Achsen.

Frames definiert. Außerdem wird das Redundancy-Tag zur Frame Codierung und die Quelladresse bzw. VLAN-Priorität zur Identifikation der Echtzeitverbindung verwendet. Die zusätzlich etablierte TSN Verbindung gewährleistet eine ausfallsichere Kommunikation zwischen der Motion Control und einer echtzeitfähigen Achse. Folglich führt ein simulierter Hardware- oder Softwarefehler auf einer der beiden TSN Verbindungen zu keinem Datenverlust, da alle echtzeitkritischen Frames parallel übertragen werden.

Virtuelle Netzwerktopologien Statt den zwei unabhängigen TSN Verbindungen zwischen der Motion Control und den echtzeitfähigen Achsen kann auch eine virtuelle Netzwerktopologie etabliert werden (Bild 8.8). Dabei werden die drei echtzeitfähigen I4.0 Komponenten in einer Ringtopologie miteinander verbunden und zyklisch ein TSN Frame übertragen. Die Anforderungen an die virtuelle Netzwerktopologie sind analog zu den einzelnen Verbindungen in dem TSN Profil mit der ID 736 spezifiziert (Bild 8.3). Darauf aufbauend erfolgt die Konfiguration durch den zentralen SDN Controller zur Laufzeit. Um die Gesamtlatenz (500 µs) zu gewährleisten werden drei aufeinanderfolgende TSN Verbindungen mit jeweils einer Zykluszeit von 125 µs etabliert. Der resultierende TSN Schedule besteht aus einem ersten Zeitslot für die Priorität 7 (35 µs) und einem weiteren Zeitslot für den restlichen Datenverkehr (90 µs). Darüber hinaus konfiguriert der SDN Controller die drei echtzeitfähigen I4.0 Komponenten. Dabei wird insbesondere das Teilmodell “TSN VT” in der Verwaltungsschale aktualisiert, d.h die Topologie-ID/-Art, die spezifische Netzwerkschnittstelle und die Informationen über die nachfolgende I4.0 Komponente gesetzt.

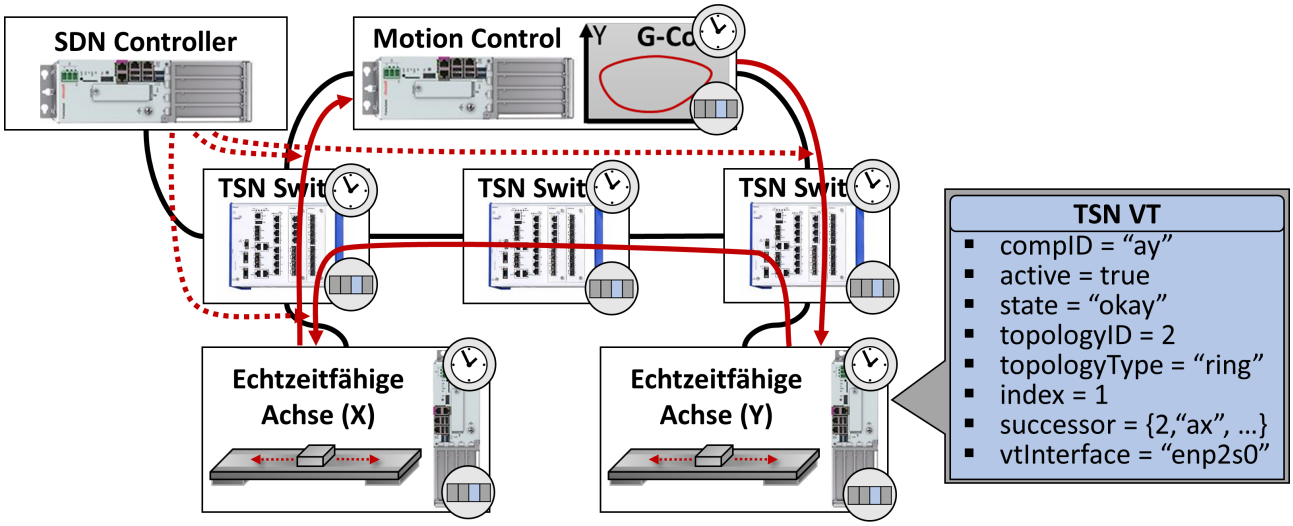


Abbildung 8.8: Virtuelle Ring-Topologie zwischen der Motion Control und den echtzeitfähigen Achsen.

Während die Motion Control als Master den Index 0 erhält, haben die darauffolgenden echtzeitfähigen Achsen den Index 1 bzw. 2.

Anwendungsprotokolle Basierend auf den TSN Verbindungen zwischen der Motion Control und den echtzeitfähigen Achsen wird exemplarisch OPC UA PubSub als Anwendungsprotokoll verwendet. Dabei sind alle echtzeitfähigen I4.0 Komponenten sowohl als OPC UA Publisher als auch Subscriber parametriert. Die zyklischen Nachrichten bestehen aus dem Protokoll-spezifischen Header und zwei Datasets, die jeweils für eine echtzeitfähige Achse bestimmt sind. In einem solchen Dataset sendet die Motion Control zyklisch die Soll-Position an die jeweilige Achse. Gleichzeitig wird die aktuelle Ist-Position von der echtzeitfähigen Achse zurückgemeldet und somit ein Regelkreis ermöglicht. Zur Codierung der gesamten Nachrichten wird UADP (UDP based UA) eingesetzt und TSN als echtzeitfähiges Transportprotokoll verwendet.

Zusammengefasst wurde die neue wandelbare, echtzeitfähige Kommunikationsinfrastruktur prototypisch implementiert und die einzelnen Teilaspekte demonstriert. Der resultierende Aufbau zeigt eine TSN-basierte Motion Control mit zwei synchron angesteuerten, echtzeitfähigen Achsen.

8.2 Evaluation TSN SDK

Die prototypische Implementierung der wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur umfasst echtzeitfähige I4.0 Komponenten mit dem vorgestellten TSN SDK (Kapitel 5). Das neue TSN SDK fügt die erforderlichen Kommunikationsmechanismen zu den echtzeitfähigen I4.0 Komponenten hinzu und ermöglicht eine durchgängige, TSN-konforme Echtzeitkommunikation. Die zusätzlichen Kommunikationsmechanismen implizieren allerdings eine zeitliche Verzögerung beim Senden und Empfangen von TSN Frames. Davon unabhängig muss die Echtzeitfähigkeit der Datenübertragung gewährleistetet sein. Daher werden im Folgenden die einzelnen Teilaspekte des TSN SDKs und die Kombination mit einem App SDK im Detail evaluiert.

Zeitsynchronisation

Die Zeitsynchronisation ist der erste Teilaspekt des TSN SDKs und bildet die Grundlage für die TSN-basierte Echtzeitkommunikation. Die prototypische Implementierung in dem TSN SDK basiert dabei auf dem etablierten Precision Time Protocol (PTP) mit dem Profil IEEE 802.1AS (Norm IEEE 1588; Stanton 2018). Konkret wird das “Linux PTP Project” in den echtzeitfähigen I4.0 Komponenten verwendet und ermöglicht eine standardkonforme Zeitsynchronisation mit den TSN Switches (Cochran 2020). Die Performance der Implementierung wurde bereits evaluiert und gewährleistet eine Genauigkeit der Zeitsynchronisation im Nanosekundenbereich (<1 µs) (Gutiérrez et al. 2017b; Cochran et al. 2011; Leong et al. 2018). Damit werden die Anforderungen der echtzeitfähigen I4.0 Komponenten erfüllt und die Grundlage für das TSN Scheduling geschaffen.

Dynamische TSN Verbindungen

Neben den TSN Switches in der Netzwerkinfrastruktur müssen auch die echtzeitfähigen I4.0 Komponenten die TSN-spezifischen Kommunikationsmechanismen unterstützen. Dabei ist insbesondere das TSN Scheduling nach IEEE 802.1Qbv für die dynamischen TSN Verbindungen erforderlich (Norm IEEE 802.1Qbv). Dieses Scheduling ist als Softwarebaustein in dem TSN SDK realisiert. Basierend auf dem Software-basierten Scheduling wird das zeitgesteuerte Senden (TTS) von der Netzwerkkarte “Intel I210 NIC” unterstützt (Intel 2020). Die entsprechende Konfiguration im Linux Betriebsystem erfolgt über die TBS-Qdiscs und die Socket-Option

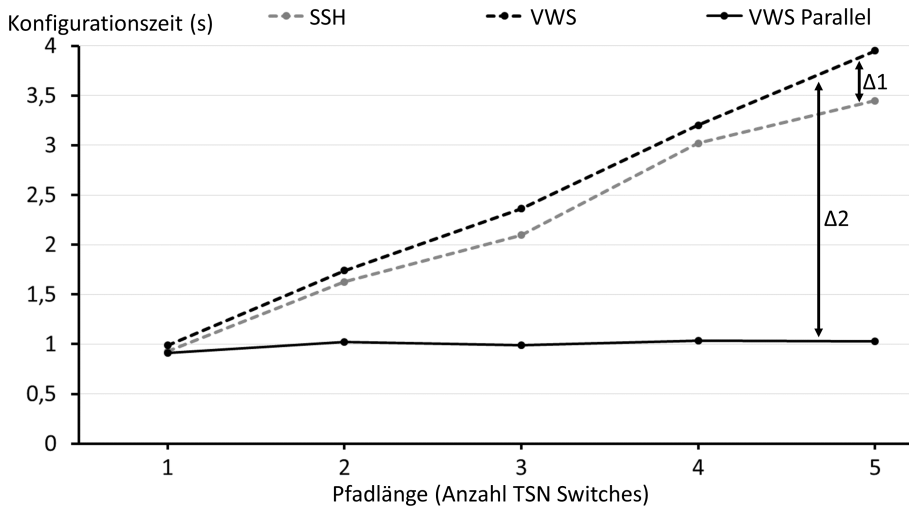


Abbildung 8.9: Die benötigte Zeit zur Konfiguration einer dynamischen TSN Verbindung im Netzwerk.

"SO_TXTIME" aus dem TX-Time-Patch (Sanchez-Palencia 2018). Die resultierende Genauigkeit beim Senden von TSN Frames wurde bereits evaluiert und die Echtzeitfähigkeit gewährleistet (Sanchez-Palencia 2018; Pfrommer et al. 2018; Alhady et al. 2018).

Parametrierung zur Laufzeit Der TSN Scheduler im vorgestellten TSN SDK wird über die dazugehörigen Teilmodelle in der Verwaltungsschale parametriert. Diese Parametrierung erfolgt durch den zentralen SDN Controller dynamisch zur Laufzeit. Das Bild 8.9 zeigt die benötigte Zeit zur Parametrierung und zur Konfiguration einer durchgängigen TSN Verbindung in der neuen Kommunikationsinfrastruktur. Basierend auf der Netzwerkdiscovery ist die Gesamtzeit für die Pfadsuche, das Scheduling und die Aktualisierung der TSN Schedules in den relevanten TSN Switches bzw. echtzeitfähigen I4.0 Komponenten dargestellt. Dabei wurde eine Netzwerkinfrastruktur mit insgesamt 100 TSN Switches angenommen. Die resultierende Konfigurationszeit wird maßgeblich von der Pfadlänge der TSN Verbindung beeinflusst, d.h. von der Anzahl TSN Switches zwischen den beiden echtzeitfähigen I4.0 Komponenten. Darüber hinaus benötigt eine Konfiguration über die vorgestellte Verwaltungsschale der TSN Switches durchschnittlich 70 ms mehr Zeit im Vergleich zu einer konventionellen Konfiguration über SSH ($\Delta 1$). Die Konfiguration eines einzelnen TSN Switches ist dabei nahezu konstant und dauert durchschnittlich 760 ms. Folglich steigt die Gesamtzeit bei einer sequentiellen Konfiguration der TSN Switches linear an. Als Optimierung wurde die parallele Konfiguration der TSN Schedules in der Netzwerkinfrastruktur realisiert. Dadurch bleibt die benötigte Konfigurationszeit durch den SDN Controller konstant bei ungefähr einer Sekunde ($\Delta 2$).

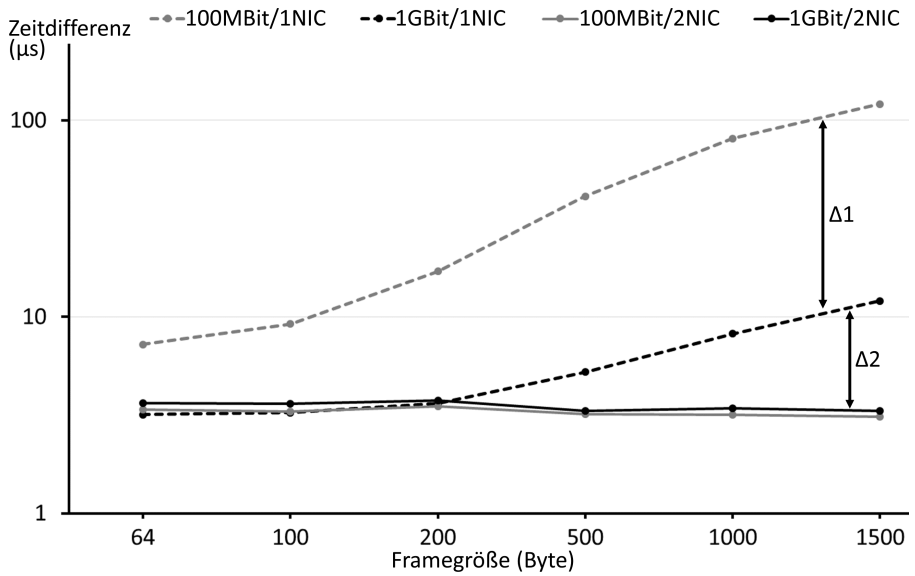


Abbildung 8.10: Die durchschnittliche Zeitdifferenz beim Senden eines primären und redundanten TSN Frames in Abhängigkeit der Framegröße, der Datenrate und der Anzahl Netzwerkadapter.

Redundante TSN Verbindungen

Basierend auf den dynamischen TSN Verbindungen ermöglichen redundante TSN Verbindungen eine ausfallsichere Echtzeitkommunikation im Netzwerk. Dazu wurden die vier erforderlichen Redundanzmechanismen in das TSN SDK der echtzeitfähigen I4.0 Komponenten integriert. Dabei muss das Senden des redundanten TSN Frames möglichst zeitsynchron zum primären Frame erfolgen. Ansonsten überschreitet der redundante Frame bei einem Fehler die echtzeitkritischen Zeitgrenzen und verletzt damit die harten Echtzeitanforderungen der Assets. Daher wird die Zeitdifferenz beim Senden des primären und redundanten TSN Frames in einer echtzeitfähigen I4.0 Komponente im Folgenden detailliert analysiert.

Zeitdifferenz beim Senden Basierend auf der prototypischen Implementierung zeigt das Bild 8.10 die durchschnittliche Zeitdifferenz beim Senden eines primären und redundanten TSN Frames. Dabei wurde die Zeitdifferenz zwischen beiden Frames mit Hilfe eines Ethernet-Taps an den Netzwerkadaptern der I4.0 Komponente gemessen. Die resultierenden Werte sind in Abhängigkeit der Framegröße, der Datenrate und der Anzahl Netzwerkschnittstellen in der echtzeitfähigen I4.0 Komponente dargestellt. Dabei verfügen alle Netzwerkschnittstellen entsprechend der vorgestellten Implementierung über den gleichen TSN Schedule, sodass der primäre und redundante TSN Frame im gleichen Zeitslot übertragen wird. Außerdem handelt es

sich um einen exklusiven Zeitslot ohne Verzögerungen durch anderen Datenverkehr mit der gleichen Priorität.

Zunächst werden echtzeitfähige I4.0 Komponenten mit nur einer Netzwerkschnittstelle betrachtet. Dabei wird der redundante TSN Frame direkt nach dem primären TSN Frame übertragen. Folglich beeinflusst die Sendedauer in Abhängigkeit der Framegröße signifikant die Zeitdifferenz des redundanten Frames. Während dieser Einfluss bei einer Datenrate von 100 MBit/s am größten ist, unterstützt die neue Kommunikationstechnologie TSN auch eine Übertragung mit 1 Gbit/s. Die höhere Datenrate ermöglicht eine schnellere Übertragung pro Byte und reduziert dadurch den Effekt der Framegröße auf die Zeitdifferenz beim Senden ($\Delta 1$).

Heutzutage sind echtzeitfähige I4.0 Komponenten mit mehreren Netzwerkschnittstellen weit verbreitet. Dadurch wird die Verzögerung beim Senden des redundanten TSN Frames minimiert, da dieser parallel zum primären Frame übertragen wird. Konkret wird die durchschnittliche Zeitdifferenz zwischen beiden TSN Frames auf unter 10 µs unabhängig von der Datenrate (100 MBit/s bzw. 1 GBit/s) reduziert. Darüber hinaus verbessern zwei Netzwerkschnittstellen auch die Ausfallsicherheit der echtzeitfähigen I4.0 Komponente. Durch die redundante Netzwerkanbindung können auch Hardwarefehler in einer der beiden Netzwerkschnittstellen oder den daran angeschlossenen Netzwerkkabeln kompensiert werden.

Zusammengefasst liegt die durchschnittliche Verzögerung beim Senden des redundanten TSN Frames über zwei verschiedene Netzwerkschnittstellen konstant im einstelligen Mikrosekundenbereich (<10 µs). Damit genügt die Software-basierte Redundanzimplementierung den harten Echtzeitanforderungen der echtzeitfähigen I4.0 Komponenten im Produktionssystem. Um die Verzögerung weiter zu reduzieren ist eine Hardwareunterstützung im Netzwerkadapter der echtzeitfähigen I4.0 Komponente in Zukunft denkbar. Dazu werden die notwendigen Redundanzmechanismen in spezielle Netzwerkarten integriert und die Software-basierte Lösung ersetzt.

Virtuelle Netzwerktopologien

Basierend auf den einzelnen echtzeitfähigen TSN Verbindungen ermöglichen virtuelle Netzwerktopologien die deterministische Kommunikation zwischen mehreren echtzeitfähigen I4.0 Komponenten. Dazu wurden die erforderlichen Kommunikationsmechanismen in das vorgestellte TSN SDK integriert. Konkret verarbeitet jede beteiligte echtzeitfähige I4.0 Komponente den eingehenden TSN Frame, modifiziert den Payload mit ihren aktuellen Werten und sendet den Frame zum Nachfolger innerhalb der Topologie. Aus der Kombination dieser Schritte ergibt sich die Latenz pro echtzeitfähiger I4.0 Komponente. Die Latenz der gesamten virtuellen

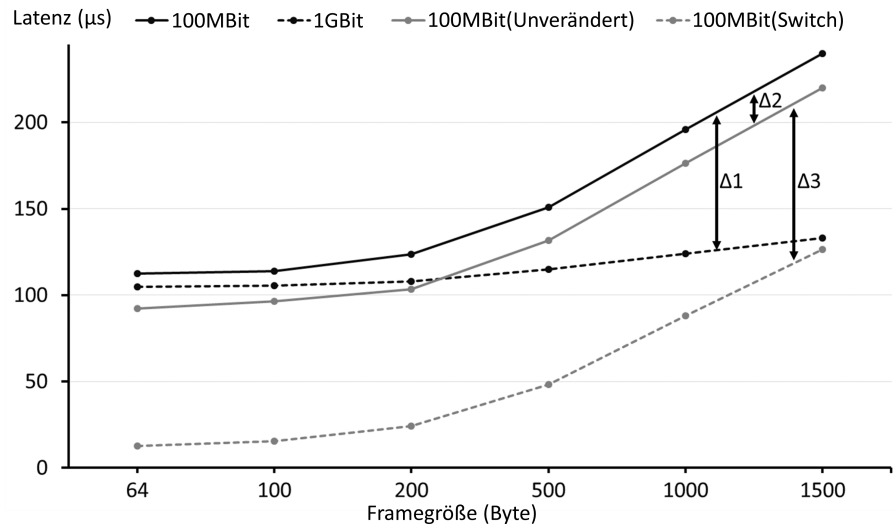


Abbildung 8.11: Die durchschnittliche Latenz beim Weiterleiten eines (modifizierten) TSN Frames in einer echtzeitfähigen I4.0 Komponente.

Netzwerktopologie wird maßgeblich von der Latenz einer einzelnen echtzeitfähigen I4.0 Komponente sowie der Anzahl von I4.0 Komponenten in der Topologie bestimmt. Da die resultierende Gesamtlatenz weiterhin den Echtzeitanforderungen der I4.0 Komponenten genügen muss, wird die Latenz pro echtzeitfähiger I4.0 Komponente im Folgenden detailliert analysiert.

Latenz pro echtzeitfähiger I4.0 Komponente Basierend auf der Implementierung im TSN SDK zeigt das Bild 8.11 die durchschnittliche Latenz innerhalb einer echtzeitfähigen I4.0 Komponente. Dabei wurde die Zeitdifferenz eines eingehenden und wieder ausgehenden TSN Frames mit Hilfe eines Ethernet-Taps an dem Netzwerkadapter der I4.0 Komponente gemessen. Die resultierenden Werte variieren in Abhängigkeit der zulässigen Framegröße von 64 bis 1522 Bytes. Bei einer Datenrate von 100 MBit/s hat die Framegröße einen signifikanten Einfluss auf die durchschnittliche Latenz. Konkret führt die Übertragung eines maximalen TSN Frames zu einer Verdopplung der Latenz im Vergleich zu einem minimalen Frame. Im Gegensatz zu existierenden Industrial Ethernet Protokollen unterstützt TSN auch eine Datenrate von 1 Gbit/s. Die höhere Datenrate ermöglicht eine schnellere Übertragung pro Byte und reduziert somit den Effekt der Framegröße auf die Latenz pro echtzeitfähiger I4.0 Komponente ($\Delta 1$).

Innerhalb einer virtuellen Netzwerktopologie wird der zyklische TSN Frame von jeder echtzeitfähigen I4.0 Komponente modifiziert. Dabei wird der Payload aktualisiert und der TSN Header entsprechend der nachfolgenden I4.0 Komponente hinzugefügt. Die resultierende zeitliche Verzögerung zur Modifikation des TSN Frames ist nahezu konstant und beträgt durchschnittlich

19 µs ($\Delta 2$). Als Vergleichswert zu der Latenz in den echtzeitfähigen I4.0 Komponenten werden TSN Switches ("Hirschmann RSPE 35" (Belden 2020b)) betrachtet. Die spezifische Firmware und Hardwareunterstützung in einem solchen TSN Switch ermöglicht eine geringere Latenz bei der Weiterleitung von unmodifizierten TSN Frames ($\Delta 3$). Daher sind weitere Verbesserungen zur Reduzierung der Latenz in den echtzeitfähigen I4.0 Komponenten zukünftig denkbar.

Cut-Through Switching Konkret ist der Einsatz von Cut-Through Switching in den echtzeitfähigen I4.0 Komponenten in der Zukunft möglich. Cut-Through Switching beschreibt die sofortige Verarbeitung und Weiterleitung eines eingehenden TSN Frames, nachdem die Zieladresse im Frame Header empfangen wurde (Astarloa et al. 2014). Im Gegensatz dazu basiert die bisherige Implementierung im TSN SDK auf dem standardkonformen Store-and-Forward Switching. Dabei wird der Empfang des gesamten TSN Frames abgewartet und die Korrektheit des Frames auf Basis der CRC Checksumme im Frame Trailer verifiziert. Obwohl nicht explizit von der IEEE standardisiert, wird Cut-Through Switching bereits in verschiedenen Industrial Ethernet Protokollen angewendet und könnte auch zur Reduzierung der Latenz in den TSN-fähigen I4.0 Komponenten beitragen. Die daraus resultierende verringerte Latenz verbessert die Gesamtperformance der virtuellen Netzwerktopologie und ermöglicht eine größere Anzahl echtzeitfähiger I4.0 Komponenten in einer Topologie.

Anwendungsprotokolle

Das vorgestellte TSN SDK stellt eine API für die Protokolle auf den darüberliegenden OSI-Schichten zur Verfügung. Diese API ermöglicht insbesondere die Verwendung von beliebigen Anwendungsprotokollen in einer echtzeitfähigen I4.0 Komponente. Folglich wird die Performance des Anwendungsprotokolls maßgeblich von dem konkreten Protokoll und der dazugehörigen Implementierung in einem App SDK bestimmt. Dabei muss auch die Echtzeitfähigkeit von dem verwendeten App SDK gewährleistet werden. Die zusätzlichen Anpassungen der Operationsaufrufe an die neue API des TSN SDKs sind vor dem Hintergrund der Performance zu vernachlässigen, da sich die Parameter nur minimal von der bisherigen Socket API des Betriebssystems unterscheiden.

Zusammengefasst genügen die einzelnen Teilaspekte des neuen TSN SDKs den Kommunikationsanforderungen der echtzeitfähigen I4.0 Komponenten. Insbesondere führen die zusätzlichen Kommunikationsmechanismen beim Senden und Empfangen von TSN Frames zu keiner signifikanten zeitlichen Verzögerung, die zu einer Überschreitung der echtzeitkritischen Zeitgren-

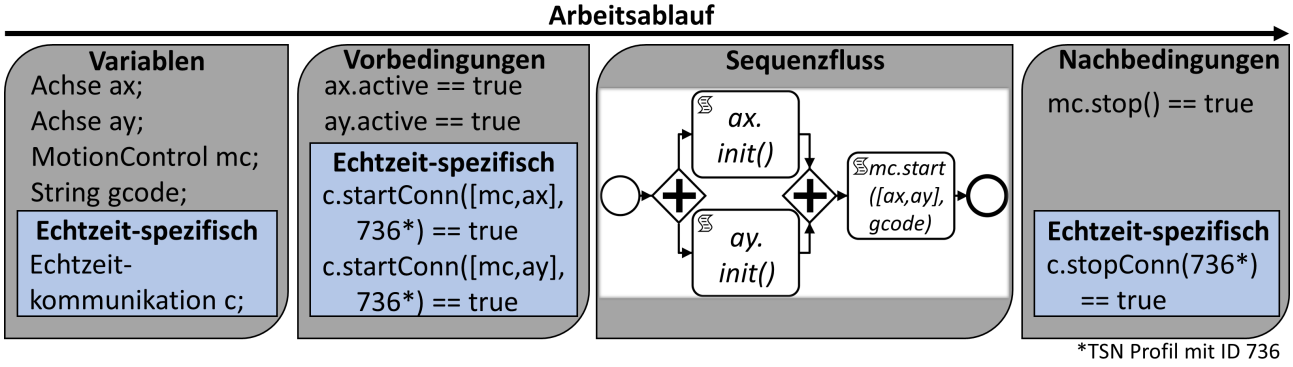


Abbildung 8.12: Ein exemplarischer Arbeitsablauf mit echtzeitfähigen I4.0 Komponenten und den Echtzeit-spezifischen Erweiterungen.

zen führen würde. In Kombination mit einem App SDK wird damit eine durchgängige, TSN-konforme Echtzeitkommunikation zwischen verschiedenen echtzeitfähigen I4.0 Komponenten gewährleistet.

8.3 Integration ins Engineering

Die vorgestellte wandelbare, echtzeitfähige Kommunikationsinfrastruktur ermöglicht die dynamische Konfiguration von Echtzeitverbindungen zur Laufzeit. Dazu verfügt der zentrale SDN Controller über das Teilmodell “Echtzeitkommunikation” als generische Schnittstelle. Durch einen entsprechenden Operationsaufruf kann eine Echtzeitverbindung von einer I4.0 Komponente initiiert oder beendet werden. Ein weiterer Vorteil dieser generischen Schnittstelle ist die Integrierbarkeit in eine beliebige Engineering-Methode.

In (Prinz et al. 2019c) wurde eine Engineering-Methode für gewöhnliche I4.0 Komponenten ohne Echtzeitanforderungen vorgestellt. Diese Methode basiert auf der Verwaltungsschale und der Fähigkeitenbeschreibung in Form von Teilmodellen. Dabei unterteilt sich das Engineering und der resultierende Arbeitsablauf in die vier Teilaspekte: Variablen, Vorbedingungen, Sequenzfluss und Nachbedingungen. Für echtzeitfähige I4.0 Komponenten werden die Arbeitsabläufe erweitert und dynamische Echtzeitverbindungen während der Ausführung etabliert. Am Beispiel der TSN-basierten Motion Control wird im Folgenden ein Arbeitsablauf mit echtzeitfähigen I4.0 Komponenten skizziert und die Echtzeit-spezifischen Erweiterungen erläutert (Bild 8.12):

Variablen Variablen referenzieren die erforderlichen (echtzeitfähigen) I4.0 Komponenten und zusätzliche Eingabedaten. Für die eigentliche Anwendung werden in diesem Beispiel zwei

echtzeitfähige Achsen sowie eine zentrale Motion Control benötigt. Außerdem wird das auszuführende G-Code Programm als zusätzlicher Parameter übergeben. Darüber hinaus wird für die dynamischen Echtzeitverbindungen der zentrale SDN Controller benötigt. Dieser Controller ist ebenfalls als I4.0 Komponente modelliert und wird über die generische Schnittstelle, das Teilmodell "Echtzeitkommunikation", referenziert.

Vorbedingungen Die Vorbedingungen werden zu Beginn des Arbeitsablaufs überprüft und müssen erfüllt sein. Beispielsweise wird die Verfügbarkeit der Achsen über ihre Statuseigenschaft "active" geprüft. Zusätzliche Echtzeit-spezifische Vorbedingungen ermöglichen das Starten der erforderlichen Echtzeitverbindungen zur Laufzeit. Dazu wird die Operation "startConn(...)" des SDN Controllers für jede Echtzeitverbindung aufgerufen und ein erfolgreicher Operationsaufruf gewährleistet.

Sequenzfluss Der Sequenzfluss modelliert den zeitlichen Ablauf und die Interaktionen während des Arbeitsablaufs. In diesem Beispiel werden zunächst die beiden echtzeitfähigen Achsen initialisiert und anschließend das übergebene G-Code Programm mit Hilfe der zentralen Motion Control ausgeführt. Währenddessen garantieren die etablierten Echtzeitverbindungen eine deterministische Kommunikation zwischen den beteiligten I4.0 Komponenten. Darüber hinaus sind für den Sequenzfluss keine Echtzeit-spezifischen Erweiterungen notwendig. Zur Modellierung des Sequenzflusses wird in der verwendeten Engineering-Methode die grafische Business Process Model and Notation (BPMN (Chinosi et al. 2012)) eingesetzt. Davon unabhängig können auch auch andere etablierte Engineering-Methoden, wie z.B. IEC 61131 oder IEC 61499 (Tiegelkamp et al. 2010; Zoitl et al. 2014), verwendet werden.

Nachbedingungen Zum Schluss werden die Nachbedingungen überprüft und müssen ebenfalls erfüllt sein. Zusätzliche Echtzeit-spezifische Nachbedingungen beenden die etablierten Echtzeitverbindungen, sodass die für diesen Arbeitsablauf reservierte Bandbreite wieder freigeben wird. Analog zu den Vorbedingungen wird dazu die Operation "stopConn(...)" des SDN Controllers aufgerufen und die erfolgreiche Ausführung überprüft.

Die Ausführung des Arbeitsablaufs mit den einzelnen Schritten in zeitlicher Reihenfolge ist im Bild 8.13 dargestellt. Nach der Initialisierung der Variablen werden die TSN-basierten Echtzeitverbindungen etabliert (1,2,3,4). Diese Echtzeitverbindungen gewährleisten eine deterministische Kommunikation mit geringer Latenz zwischen den beteiligten echtzeitfähigen I4.0 Komponenten (5,6). Nach der Ausführung werden die Echtzeitverbindungen wieder beendet und die reservierte Bandbreite wieder freigegeben (7,8). Zusammengefasst ermöglicht die generische

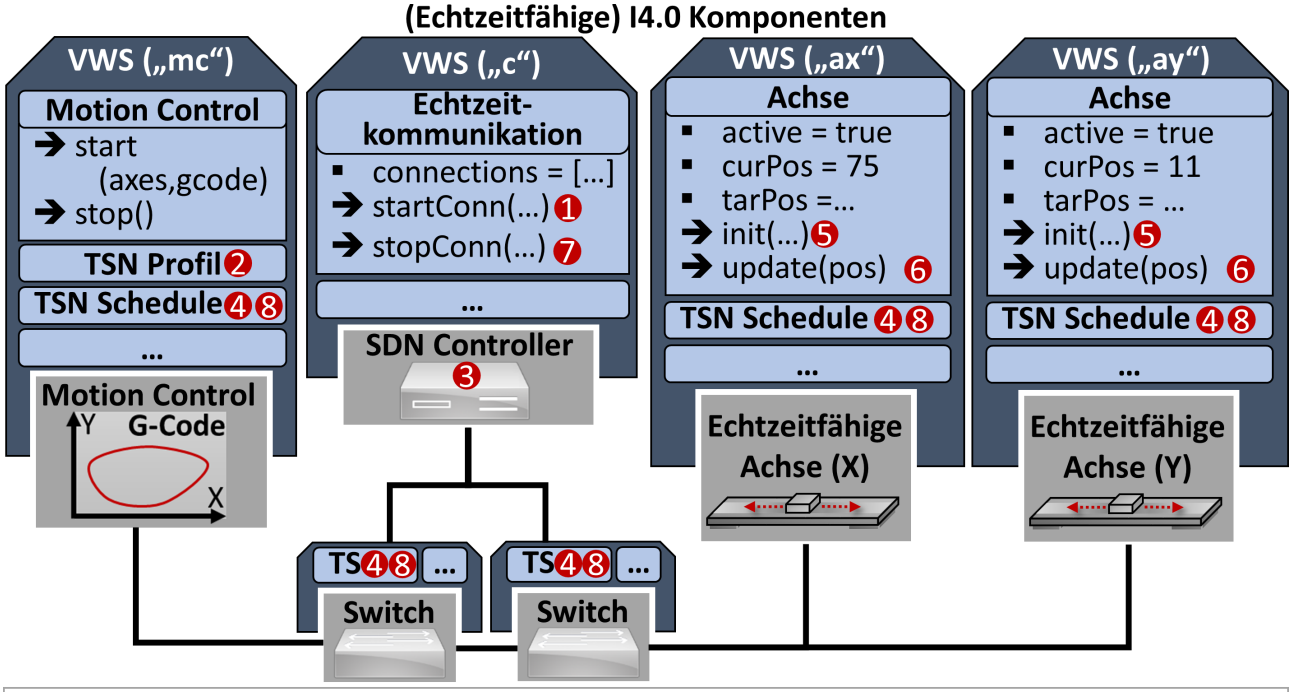


Abbildung 8.13: Die Ausführung des exemplarischen Arbeitsablaufs mit dynamischen Echtzeitverbindungen zur Laufzeit.

Schnittstelle des SDN Controller eine nahtlose Integration ins Engineering und Arbeitsabläufe mit echtzeitfähigen I4.0 Komponenten.

8.4 Bewertung der Lösung

Im Folgenden wird der vorgestellte Lösungsansatz bezüglich den Anforderungen an eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur bewertet. Als Bewertungsgrundlage dienen die identifizierten Anforderungen aus Kapitel 3, die auch schon für die Bewertung existierender Lösungen verwendet wurden. Die Bewertung gliedert sich dabei in die drei wesentlichen Aspekte einer wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur (Bild 8.14).

Echtzeitkommunikation Die vorgestellte Echtzeitkommunikation basiert auf der neuen IEEE Technologie Time-Sensitive Networking (TSN), die eine **einheitliche Kommunikationsbasis** für zukünftige Produktionssysteme ermöglicht. Dazu wurden die notwendigen TSN-Mechanismen für eine **echtzeitfähige und ausfallsichere Kommunikation** in die echtzeitfähigen Assets integriert und in einem TSN SDK zusammengefasst. Zusätzliche

Abbildung 8.14: Die Ergebnisse der Arbeit basierend auf den identifizierten Anforderungen.

Kommunikationsmechanismen im TSN SDK ermöglichen die Vernetzung von mehreren echtzeitfähigen Assets in einer **beliebigen (virtuellen) Netzwerktopologie**. Darauf aufbauend ermöglicht die definierte TSN API die Integration von **beliebigen Anwendungsprotokollen** und entsprechenden SDKs zur durchgängigen Echtzeitkommunikation.

Parametrierung Das eingeführte Konzept der echtzeitfähigen I4.0 Komponente mit einer standardisierten Verwaltungsschale ermöglicht eine **einheitliche Beschreibung** von echtzeitfähigen Assets. Dabei beinhaltet die Verwaltungsschale neben den Asset-spezifischen Teilmodellen zusätzliche Teilmodelle zur Parametrierung der **durchgängigen Echtzeitkommunikation**. Darauf aufbauend wurde die Parametrierung von **beliebigen Anwendungsprotokollen** mit Hilfe von weiteren Teilmodellen in der Verwaltungsschale erläutert. Gleichzeitig bleibt durch die Definition von zusätzlichen Echtzeit-spezifischen Teilmodellen die **Kompatibilität** zu gewöhnlichen I4.0 Komponenten ohne Echtzeitanforderungen im Produktionssystem erhalten.

Netzwerkkonfiguration Der erweiterte SDN Controller ermöglicht die dynamische Konfiguration der Echtzeitkommunikation zur Laufzeit und stellt eine **einheitliche Konfigurationsschnittstelle** zur Verfügung. Dabei bildet die Modellierung der TSN Switches als I4.0 Komponente mit Verwaltungsschale die Grundlage für eine einheitliche, herstellerunabhängige

unabhängige Konfiguration der **Netzwerkinfrastruktur**. Darauf aufbauend konfiguriert der zentrale SDN Controller auch die erforderlichen Kommunikationsmechanismen in den **echtzeitfähigen I4.0 Komponenten** und verwendet dazu die definierten TSN-spezifischen Teilmodelle. Basierend auf der Rekonfigurierbarkeit von Netzwerkkomponenten und Netzwerkteilnehmern erfolgt die dynamische Konfiguration der eigentlichen **Echtzeitverbindungen** durch den zentralen SDN Controller.

Zusammengefasst erfüllt die vorgestellte wandelbare, echtzeitfähige Kommunikationsinfrastruktur die identifizierten Anforderungen an die Echtzeitkommunikation, die Parametrierung und die Netzwerkkonfiguration. Damit wird im Gegensatz zu existierenden Lösungen eine ganzheitliche Lösung ermöglicht und ein wesentlicher Beitrag zu einer CPS-basierten Automation geleistet. Darüber hinaus bildet die vorgestellte Kommunikationsinfrastruktur die Grundlage für die Wandelbarkeit von echtzeitfähigen Assets und zukünftigen Produktionssystemen allgemein.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Heutzutage agieren Unternehmen in einem immer volatileren Umfeld mit verschiedenen Wandlungstreibern, die einen unmittelbaren Einfluss auf die Produktionssysteme haben. Um die Wandelbarkeit von zukünftigen Produktionssystemen zu verbessern wird daher eine CPS-basierte Automation mit einer einheitlichen Kommunikationsinfrastruktur angestrebt. Neben der Wandelbarkeit ist dabei die durchgehende Echtzeitfähigkeit ein wesentliches Merkmal einer zukünftigen Kommunikationsinfrastruktur.

Stand heute existieren verschiedene Lösungsansätze mit unterschiedlichen Technologien zur Etablierung einer echtzeitfähigen Kommunikationsinfrastruktur. Diese Lösungen erfüllen allerdings die Anforderungen an eine zukünftige Kommunikationsinfrastruktur nur teilweise und stellen daher keine ganzheitliche Lösung dar. Insbesondere der Aspekt der Wandelbarkeit mit einer dynamischen Konfiguration von Echtzeitverbindungen zur Laufzeit wird heutzutage nur bedingt in einer Kommunikationsinfrastruktur unterstützt.

Als Grundlage für eine CPS-basierte Automation wurde in dieser Arbeit eine **wandelbare, echtzeitfähige Kommunikationsinfrastruktur** für zukünftige Produktionssysteme präsentiert. Diese einheitliche Kommunikationsinfrastruktur erfüllt die Anforderungen von echtzeitfähigen Assets an die Kommunikation und ermöglicht gleichzeitig die dynamische Konfiguration der Echtzeitverbindungen zur Laufzeit. Mit dem Fokus auf einer ganzheitlichen Lösung basiert die vorgestellte Kommunikationsinfrastruktur auf den drei wesentlichen Aspekten: Echtzeitkommunikation, Parametrierung und Netzwerkkonfiguration.

Echtzeitkommunikation Die vorgestellte Echtzeitkommunikation basiert auf der neuen IEEE Technologie Time-Sensitive Networking (TSN), die eine einheitliche Kommunikationsbasis für zukünftige Produktionssysteme ermöglicht. Die erforderlichen Kommunikationsmechanismen wurden im Detail erarbeitet und in die echtzeitfähigen Assets integriert. Insbesondere wurden die TSN-spezifischen Mechanismen für eine echtzeitfähige und ausfallsichere Kommunikation standardkonform umgesetzt und in einem TSN SDK zusammengefasst. Darauf aufbauend ermöglicht die Kombination des TSN SDKs mit einem beliebigen App SDK die durchgängige Echtzeitkommunikation zwischen verschiedenen echtzeitfähigen Assets.

Parametrierung Zur einheitlichen Parametrierung der echtzeitfähigen Assets und der spezifischen Kommunikationsmechanismen wurde das Konzept der echtzeitfähigen I4.0 Komponente eingeführt. Das neue Konzept beschreibt ein echtzeitfähiges Asset mit einer herstellerunabhängigen Verwaltungsschale und zusätzlichen Echtzeit-spezifischen Teilmodellen. Die zusätzlichen Teilmodelle für die TSN-basierte Echtzeitkommunikation wurden im Detail vorgestellt und die erforderlichen Konfigurationsparameter und Statusinformationen spezifiziert. Darauf aufbauend wurde die Parametrierung von beliebigen Anwendungsprotokollen mit Hilfe von weiteren Teilmodellen in der Verwaltungsschale erläutert.

Netzwerkkonfiguration Zur automatischen und dynamischen Konfiguration der Echtzeitverbindungen wurde ein zentraler SDN Controller erweitert. Der erweiterte SDN Controller etabliert dynamische, redundante und komplexe TSN Verbindungen zwischen verschiedenen echtzeitfähigen I4.0 Komponenten zur Laufzeit. Dabei werden sowohl die TSN Switches in der Netzwerkinfrastruktur als auch die Kommunikationsmechanismen in den echtzeitfähigen I4.0 Komponenten konfiguriert. Die notwendigen Konfigurationsmechanismen wurden im Detail erläutert und entsprechende Teilmodelle für die Verwaltungsschale des SDN Controllers definiert.

Schließlich wurde die Machbarkeit der neuen wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur durch eine prototypische Implementierung gezeigt. Der prototypische Aufbau zeigt, dass eine TSN-basierte Motion Control echtzeitfähige Achsen mit Hilfe dynamisch konfigurierter TSN Verbindungen ohne Einschränkungen synchronisieren und ansteuern kann. Darauf aufbauend wurde die Performance des vorgestellten TSN SDKs evaluiert und die Integration ins Engineering beschrieben.

9.2 Ausblick

Migration zu TSN Die neue IEEE Technologie TSN bildet die Grundlage für die Echtzeitkommunikation in der neuen wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur. Durch die Echtzeitfähigkeit und gleichzeitige Ethernet-Kompatibilität wird TSN als eine vielversprechende Schlüsseltechnologie für eine einheitliche Kommunikationsinfrastruktur in zukünftigen Produktionssystemen angesehen. Auf Grund der langen Lebenszyklen von Maschinen im industriellen Umfeld werden sich allerdings ausschließlich TSN-basierte Lösungen erst mit der Zeit etablieren. Deswegen gibt es einen längeren Migrationspfad mit einer Koexistenz von neuen TSN-fähigen Assets und existierenden Assets mit etablierten Kommunikationslösungen. Daher ist die Migration von existierenden Produktionssystemen mit ihren Kommunikationslösungen hin zu TSN ein sehr relevanter Forschungsbereich. Dies betrifft insbesondere die Migration von den heutzutage weitverbreiteten Industrial Ethernet Protokollen (ohne TSN) hin zu einer TSN-basierten Echtzeitkommunikation.

Kombination mit 5G Stand heute erfordert eine deterministische Kommunikation mit geringer Latenz eine drahtgebundene Kommunikationstechnologie. Auch die vorgestellte Echtzeitkommunikation in der neuen wandelbaren, echtzeitfähigen Kommunikationsinfrastruktur basiert auf der drahtgebundenen IEEE Technologie TSN. Alternative drahtlose Technologien, wie z.B. WLAN oder 4G (IEEE 2020a), ermöglichen Stand heute auf Grund fehlender Mechanismen keine Garantien für eine Echtzeitkommunikation. Abhilfe könnte in Zukunft der Mobilfunkstandard 5G schaffen (Nasrallah et al. 2018), der die fünfte Evolution der mobilen, drahtlosen Kommunikationsnetzwerke beschreibt. Dieser Standard verspricht neben einer performanten und skalierbaren Datenübertragung auch eine echtzeitfähige Übertragung mit geringer Latenz. Damit ermöglicht 5G eine drahtlose Anbindung und Vernetzung von echtzeitfähigen Assets in zukünftigen Produktionssystemen. Neben der Entwicklung der 5G Standards ist insbesondere die Kombination von 5G und TSN mit einer übergreifenden Netzwerkkonfiguration ein sehr relevanter Forschungsbereich. Mit Fokus auf der Wandelbarkeit von zukünftigen Produktionssystemen wird dabei ein nahtloser Wechsel zwischen TSN und 5G sowohl in der Netzwerkinfrastruktur als auch in den echtzeitfähigen Assets angestrebt.

Literaturverzeichnis

Abele et al. 2006
Abele, Eberhard; Liebeck, Tobias; Wörn, Arno, 2006.
Measuring flexibility in investment decisions for manufacturing systems.
CIRP Annals, **55**, S. 433–436.
DOI: 10.1016/S0007-8506(07)60452-1

Ademaj 2019
Ademaj, Astrit, 2019.
Traffic Type Introduction.
Verfügbar unter: http://www.ieee802.org/1/files/public/docs2019/60802-ademaj-traffic-type-introduction-0319-v03.pdf
Zugriff am: 01. 03. 2020

Agarwal et al. 2019
Agarwal, Tanushree; Niknejad, Payam; Barzegaran, Mohammadreza; Vanfretti, Luigi, 2019.
Multi-Level Time-Sensitive Networking (TSN) Using the Data Distribution Services (DDS) for Synchronized Three-Phase Measurement Data Transfer.
IEEE Access, **7**, S. 131407–131417.
DOI: 10.1109/ACCESS.2019.2939497

Alhady et al. 2018
Alhady, Syed Sahal Nazli; Othman, Amir Fuad Wajdi, 2018.
Time-aware Traffic Shaper using Time-based Packet Scheduling on Intel I210.
International Journal of Research and Engineering, **5** (9), S. 494–499.
DOI: 10.21276/ijre.2018.5.9.1

Alisch et al. 2013
Alisch, Katrin; Winter, Eggert; Arentzen, Ute, 2013.
Gabler Wirtschafts Lexikon.
Wiesbaden: Springer.
ISBN 978-3-658-19570-0

Álvarez et al. 2018a Álvarez, Inés; Barranco, Manuel; Proenza, Julián, 2018a.
Towards a Fault-Tolerant Architecture Based on Time Sensitive Networking.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1113–1116.
DOI: 10.1109/ETFA.2018.8502614

Álvarez et al. 2019 Álvarez, Inés; Čavka, Drago; Proenza, Julián; Barranco, Manuel, 2019.
Simulation of the Proactive Transmission of Replicated Frames Mechanism over TSN.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1375–1378.
DOI: 10.1109/ETFA.2019.8868997

Álvarez et al. 2018b Álvarez, Inés; Proenza, Julián; Barranco, Manuel, 2018b.
Mixing Time and Spatial Redundancy Over Time Sensitive Networking.
In: Esteves-Verissimo, Paulo (Hrsg.): *DSN Workshops*,
S. 63–64.
DOI: 10.1109/DSN-W.2018.00031

Álvarez et al. 2017 Álvarez, Inés; Proenza, Julián; Barranco, Manuel; Knezic, Mladen, 2017.
Towards a time redundancy mechanism for critical frames in Time-Sensitive Networking.
In: Charalambous, Charalambos D.; Mahulea, Cristian (Hrsg.): *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1–4.
DOI: 10.1109/ETFA.2017.8247721

Ansah et al. 2019a Ansah, Frimpong; Abid, Mohamed Amine; de Meer, Hermann, 2019a.
Schedulability Analysis and GCL Computation for Time-Sensitive Networks.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2019 17th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 926–932.
DOI: 10.1109/INDIN41052.2019.8971965

Ansah et al. 2019b
Ansah, Frimpong; Majumder, Mainak; de Meer, Hermann; Jasperneite, Jürgen, 2019b.
Network Slicing: An Industry Perspective.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1367–1370.
DOI: 10.1109/ETFA.2019.8869073

Astarloa et al. 2014
Astarloa, Armando; Lázaro, Jesús; Bidarte, Unai; Araujo, Jose Angel; Moreira, Naiara, 2014.
FPGA implemented cut-through vs store-and-forward switches for reliable Ethernet networks.
In: Berenguer, Roc; Portilla, Jorge; de la Torre, Eduardo (Hrsg.): *2014 Conference on Design of Circuits and Integrated Circuits (DCIS)*,
S. 1–6.
DOI: 10.1109/DCIS.2014.7035561

Barrett et al. 2018
Barrett, Clark; Tinelli, Cesare, 2018.
Satisfiability modulo theories.
In: Clarke, Edmund M.; Henzinger, Thomas A.; Veith, Helmut; Bloem, Roderick (Hrsg.): *Handbook of Model Checking*,
S. 305–343.
DOI: 10.1007/978-3-319-10575-8_11

Belden 2020a
Belden Electronics GmbH, 2020a.
Hirschmann Managed RSP Switches.
Verfügbar unter: https://www.hirschmann.de/de/Hirschmann/Industrial_Ethernet/managed_rsp_switches/index.phtml
Zugriff am: 01. 03. 2020

Belden 2020b
Belden Electronics GmbH, 2020b.
Managed Industrial DIN Rail Fast/Gigabit Ethernet Switches – RSPE-Expandable.
Verfügbar unter: http://www.hirschmann.com/en/Hirschmann_Produkte/Industrial_Ethernet/managed_rsp_switches/rspe_expandable/index.phtml
Zugriff am: 01. 03. 2020

Bello et al. 2019
Bello, Lucia Lo; Steiner, Wilfried, 2019.
A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems.
Proceedings of the IEEE, **107** (6), S. 1094–1120.
DOI: 10.1109/JPROC.2019.2905334

Bjorklund 2010
Bjorklund, Martin, 2010.
YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF).
Verfügbar unter: `https://tools.ietf.org/html/rfc6020`
Zugriff am: 01. 03. 2020

Böhm et al. 2019a
Böhm, Martin; Ohms, Jannis; Kumar, Manish; Gebauer, Olaf; Wermser, Diederich, 2019a.
Time-Sensitive Software-Defined Networking: A Unified Control-Plane for TSN and SDN.
In: Informationstechnische Gesellschaft im VDE (Hrsg.): *Mobile Communication-Technologies and Applications; 24. ITG-Symposium*,
S. 1–6.
ISBN 978-3-8007-4961-4

Böhm et al. 2019b
Böhm, Martin; Ohms, Jannis; Wermser, Diederich, 2019b.
Multi-Domain Time-Sensitive Networks-An East-Westbound Protocol for Dynamic TSN-Stream Configuration Across Domains.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1363–1366.
DOI: `10.1109/ETFA.2019.8869280`

Rexroth 2020
Bosch Rexroth AG, 2020.
Box-PC VPB.
Verfügbar unter: `https://www.boschrexroth.com/de/de/produkte/produktgruppen/elektrische-antriebe-und-steuerungen/industrie-pc-und-bedien-panels/box-pc/vpb`
Zugriff am: 01. 03. 2020

Bruckner et al. 2018
Bruckner, Dietmar; Blair, Richard; Stănică, Marius-Petru; Schriegel, Sebastian; Leurs, Ludwig et al., 2018.
OPC UA TSN - A new Solution for Industrial Communication.
Verfügbar unter: `https://cdn.weka-fachmedien.de/whitepaper/files/OPC_UA_TSN_-_A_new_Solution_for_Industrial_Communication.pdf`
Zugriff am: 01. 03. 2020

Bruckner et al. 2019
Bruckner, Dietmar; Stănică, Marius-Petru; Blair, Richard; Schriegel, Sebastian; Kehrer, Stephan; Seewald, Maik; Sauter, Thilo, 2019.
An introduction to OPC UA TSN for industrial communication systems.
Proceedings of the IEEE, **107** (6), S. 1121–1131.
DOI: `10.1109/JPROC.2018.2888703`

BMWi 2020 Bundesministerium für Wirtschaft und Energie (BMWi), 2020. *Plattform Industrie I4.0*.
Verfügbar unter: http://www.plattform-i40.de
Zugriff am: 01. 03. 2020

BMWi 2019 Bundesministerium für Wirtschaft und Energie (BMWi), 2019. *Verwaltungsschale in der Praxis*.
Verfügbar unter: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/2019-verwaltungsschale-in-der-praxis.html
Zugriff am: 01. 03. 2020

BMWi 2018a Bundesministerium für Wirtschaft und Energie (BMWi), 2018a. *Details of the Asset Administration Shell*.
Verfügbar unter: https://www.zvei.org/presse-medien/publikationen/details-of-the-asset-administration-shell
Zugriff am: 01. 03. 2020

BMWi 2018b Bundesministerium für Wirtschaft und Energie (BMWi), 2018b. *Structure of the Administration Shell*.
Verfügbar unter: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/hm-2018-trilaterale-coop.html
Zugriff am: 01. 03. 2020

Case et al. 2002 Case, Jeffrey; Mundy, Russ; Partain, David; Stewart, Bob, 2002. *Introduction and Applicability Statements for Internet Standard Management Framework*.
Verfügbar unter: https://tools.ietf.org/html/rfc3410
Zugriff am: 01. 03. 2020

Chinosi et al. 2012 Chinosi, Michele; Trombetta, Alberto, 2012.
BPMN: An introduction to the standard.
Computer Standards & Interfaces, **34**, S. 124–134.
DOI: 10.1016/j.csi.2011.06.002

Cochran 2020 Cochran, Richard, 2020.
The Linux PTP Project.
Verfügbar unter: http://linuxptp.sourceforge.net/
Zugriff am: 01. 03. 2020

Cochran et al. 2011 Cochran, Richard; Marinescu, Cristian; Riesch, Christian, 2011.
Synchronizing the Linux system time to a PTP hardware clock.
In: Loschmidt, Patrick; Bartos, Radim (Hrsg.): *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*,
S. 87–92.
DOI: 10.1109/ISPCS.2011.6070158

Collins 2020
Collins Dictionary, 2020.
Engineering.
Verfügbar unter: `https://www.collinsdictionary.com/de/worterbuch/englisch/engineering`
Zugriff am: 01. 03. 2020

Corsten et al. 2012
Corsten, Hans; Gössinger, Ralf, 2012.
Produktionswirtschaft: Einführung in das industrielle Produktionsmanagement.
München: Oldenbourg.
ISBN 978-3-486-58298-7

Craciunas et al. 2016
Craciunas, Silviu S.; Oliver, Ramon Serna; Chmelík, Martin; Steiner, Wilfried, 2016.
Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks.
In: Plantec, Alain; Singhoff, Frank; Faucou, Sébastian; Pinho, Luís Miguel (Hrsg.): *Proceedings of the 24th International Conference on Real-Time Networks and Systems*,
S. 183–192.
DOI: `10.1145/2997465.2997470`

Danielis et al. 2018
Danielis, Peter; Puttnies, Henning; Schweissguth, Eike; Timmermann, Dirk, 2018.
Real-Time Capable Internet Technologies for Wired Communication in the Industrial IoT-a Survey.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 266–273.
DOI: `10.1109/ETFA.2018.8502528`

Danielis et al. 2014
Danielis, Peter; Skodzik, Jan; Altmann, Vlado; Schweissguth, Eike Bjoern; Golatowski, Frank; Timmermann, Dirk; Schacht, Joerg, 2014.
Survey on real-time communication via Ethernet in industrial automation environments.
In: Grau, Antoni; Martinez, Herminio (Hrsg.): *2014 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1–8.
DOI: `10.1109/ETFA.2014.7005074`

Deering et al. 2017
Deering, Steve; Hinden, Robert, 2017.
Internet Protocol, Version 6 (IPv6) Specification.
Verfügbar unter: `https://tools.ietf.org/html/rfc8200`
Zugriff am: 01. 03. 2020

Deppe et al. 2019
Deppe, Torben; Elfaham, Haitham; Epple, Ulrich, 2019.
Discovery Service for Industry 4.0 based on Property Value Statements.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2019 17th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 727–732.
DOI: 10.1109/INDIN41052.2019.8972076

Dijkstra 1959
Dijkstra, Edsger W., 1959.
A note on two problems in connexion with graphs.
Numerische Mathematik, S. 269–271.
DOI: 10.1007/BF01386390

Drath et al. 2014
Drath, Rainer; Horch, Alexander, 2014.
Industrie 4.0: Hit or hype?
IEEE Industrial Electronics Magazine, **8** (2), S. 56–58.
DOI: 10.1109/MIE.2014.2312079

Dumitrescu et al. 2015
Dumitrescu, Roman; Gausemeier, Jürgen; Kühn, Arno; Luckey, Markus; Plass, Christoph; Schneider, Marcel; Westermann, Thorsten, 2015.
Auf dem Weg zu Industrie 4.0: Erfolgsfaktor Referenzarchitektur.
Verfügbar unter: https://www.its-owl.de/fileadmin/PDF/Informationsmaterialien/2015-Auf_dem_Weg_zu_Industrie_4.0_Erfolgsfaktor_Referenzarchitektur.pdf
Zugriff am: 01. 03. 2020

Dürkop et al. 2013
Dürkop, Lars; Imtiaz, Jahanzaib; Trsek, Henning; Wisniewski, Lukasz; Jasperneite, Jürgen, 2013.
Using OPC UA for the autoconfiguration of real-time Ethernet systems.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 248–253.
DOI: 10.1109/INDIN.2013.6622890

Dürkop et al. 2015
Dürkop, Lars; Jasperneite, Jürgen; Fay, Alexander, 2015.
An analysis of real-time Ethernets with regard to their automatic configuration.
In: Pedreiras, Paulo; Vitturi, Stefano (Hrsg.): *2015 IEEE World Conference on Factory Communication Systems (WFCS)*,
S. 1–8.
DOI: 10.1109/WFCS.2015.7160548

Dürr et al. 2016
Dürr, Frank; Nayak, Naresh Ganesh, 2016.
No-wait packet scheduling for IEEE Time-Sensitive Networks (TSN).
In: Plantec, Alain; Singhoff, Frank; Faucou, Sébastian; Pinho, Luís Miguel (Hrsg.): *Proceedings of the 24th International Conference on Real-Time Networks and Systems*,
S. 203–212.
DOI: 10.1145/2997465.2997494

Eckhardt et al. 2019
Eckhardt, Andreas; Müller, Sebastian, 2019.
Analysis of the Round Trip Time of OPC UA and TSN based Peer-to-Peer Communication.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 161–167.
DOI: 10.1109/ETFA.2019.8869060

Eckhardt et al. 2018
Eckhardt, Andreas; Müller, Sebastian; Leurs, Ludwig, 2018.
An Evaluation of the Applicability of OPC UA Publish Subscribe on Factory Automation Use Cases.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1071–1074.
DOI: 10.1109/ETFA.2018.8502445

Ehrlich et al. 2018
Ehrlich, Marco et al., 2018.
Software-Defined Networking as an Enabler for Future Industrial Network Management.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1109–1112.
DOI: 10.1109/ETFA.2018.8502561

ElMaraghy et al. 2009
ElMaraghy, Hoda A.; Wiendahl, Hans-Peter, 2009.
Changeability–an introduction.
In: ElMaraghy, Hoda A. (Hrsg.): *Changeable and reconfigurable manufacturing systems*,
S. 3–24.
DOI: 10.1007/978-1-84882-067-8_1

Enns et al. 2011
Enns, Rob; Bjorklund, Martin; Schoenwaelder, Juergen; Bierman, Andy, 2011.
Network Configuration Protocol (NETCONF).
Verfügbar unter: https://tools.ietf.org/html/rfc6241
Zugriff am: 01. 03. 2020

Falk et al. 2019
Falk, Jonathan; Hellmanns, David; Carabelli, Ben; Nayak, Naresh; Dürr, Frank; Kehrer, Stephan; Rothermel, Kurt, 2019.
NeSTiNg: Simulating IEEE Time-Sensitive Networking (TSN) in OMNeT++.
In: Carle, Georg; Hossfeld, Tobias; Kellerer, Wolfgang; Ott, Jörg (Hrsg.): *2019 International Conference on Networked Systems (NetSys)*,
S. 1–8.
DOI: 10.1109/NetSys.2019.8854500

Farkas 2018
Farkas, Janos, 2018.
IEEE 802.1 Time-Sensitive Networking (TSN) Task Group (TG) Overview.
Verfügbar unter: http://www.ieee802.org/1/files/public/docs2018/detnet-tsn-farkas-tsn-overview-1118-v01.pdf
Zugriff am: 01. 03. 2020

Felser 2005
Felser, Max, 2005.
Real-time Ethernet - Industry Prospective.
Proceedings of the IEEE, **93** (6), S. 1118–1129.
DOI: 10.1109/JPROC.2005.849720

Finn 2018
Finn, Norman, 2018.
Introduction to Time-Sensitive Networking.
IEEE Communications Standards Magazine, **2** (2), S. 22–28.
DOI: 10.1109/MCOMSTD.2018.1700076

Finzi et al. 2019
Finzi, Anaïs; Craciunas, Silviu S., 2019.
Integration of SMT-based Scheduling with RC Network Calculus Analysis in TTEthernet Networks.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 192–199.
DOI: 10.1109/ETFA.2019.8869365

Fuchs et al. 2019
Fuchs, Jonathan; Schmidt, Jan; Franke, Jörg; Rehman, Kasim; Sauer, Manuel; Karnouskos, Stamatis, 2019.
I4.0-compliant integration of assets utilizing the Asset Administration Shell.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1243–1247.
DOI: 10.1109/ETFA.2019.8869255

Geisberger et al. 2012
Geisberger, Eva; Broy, Manfred, 2012.
agendaCPS: Integrierte Forschungsagenda Cyber-Physical Systems.
Berlin Heidelberg: Springer.
ISBN 978-3-642-29099-2

Gerhard et al. 2019 Gerhard, Tim; Kobzan, Thomas; Blöcher, Immanuel; Hendel, Maximilian, 2019.
Software-defined Flow Reservation: Configuring IEEE 802.1Q Time-Sensitive Networks by the Use of Software-Defined Networking.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 216–223.
DOI: 10.1109/ETFA.2019.8869040

Givehchi et al. 2013 Givehchi, Omid; Trsek, Henning; Jasperneite, Jürgen, 2013.
Cloud computing for industrial automation systems—A comprehensive overview.
In: Seatzu, Carla (Hrsg.): *2013 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1–4.
DOI: 10.1109/ETFA.2013.6648080

Gogolev et al. 2019 Gogolev, Alexander; Braun, Roland; Bauer, Philipp, 2019.
TSN Traffic Shaping for OPC UA Field Devices.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2019 17th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 951–956.
DOI: 10.1109/INDIN41052.2019.8972252

Gogolev et al. 2018 Gogolev, Alexander; Mendoza, Francisco; Braun, Rol, 2018.
TSN-enabled OPC UA in Field Devices.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 297–303.
DOI: 10.1109/ETFA.2018.8502597

Gutiérrez et al. 2017a Gutiérrez, Marina; Ademaj, Astrit; Steiner, Wilfried; Dobrin, Radu; Punnekkat, Sasikumar, 2017a.
Self-configuration of IEEE 802.1 TSN networks.
In: Charalambous, Charalambos D.; Mahulea, Cristian (Hrsg.): *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1–8.
DOI: 10.1109/ETFA.2017.8247597

Gutiérrez et al. 2017b Gutiérrez, Marina; Steiner, Wilfried; Dobrin, Radu; Punnekkat, Sasikumar, 2017b.
Synchronization quality of IEEE 802.1AS in large-scale industrial automation networks.
In: Parmer, Gabriel (Hrsg.): *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, S. 273–282.
DOI: 10.1109/RTAS.2017.10

Hart et al. 1968 Hart, Peter E.; Nilsson, Nils J.; Raphael, Bertram, 1968.
A formal basis for the heuristic determination of minimum cost paths.
IEEE Transactions on Systems Science and Cybernetics, **4** (2), S. 100–107.
DOI: 10.1109/TSSC.1968.300136

Heinen et al. 2010 Heinen, Tobias; Peter, Kathrin; Erlach, Klaus; Nyhuis, Peter; Lanza, Gisela; Westkämper, Engelbert, 2010.
Zukunftsthemen der Fabrikplanung.
ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb, **105** (5), S. 405–409

Heinrich et al. 2015 Heinrich, Berthold; Linke, Petra; Glöckler, Michael, 2015.
Grundlagen Automatisierung.
Wiesbaden: Springer.
ISBN 978-3-658-17582-5

Hermann et al. 2016 Hermann, Mario; Pentek, Tobias; Otto, Boris, 2016.
Design principles for industrie 4.0 scenarios.
In: Bui, Tung X.; Sprague, Ralph H. (Hrsg.): *2016 49th Hawaii International Conference on System Sciences (HICSS)*, S. 3928–3937.
DOI: 10.1109/HICSS.2016.488

Hoffmeister 2015 Hoffmeister, Michael, 2015.
The Industrie 4.0 Component.
ZVEI - German Electrical and Electronic Manufacturers' Association

IEEE 2020a IEEE 802 LAN/MAN Standards Committee, 2020a.
IEEE 802.11 Wireless LAN Working Group.
Verfügbar unter: http://www.ieee802.org/11/
Zugriff am: 01. 03. 2020

IEEE 2020b IEEE 802 LAN/MAN Standards Committee, 2020b.
IEEE 802.3 Ethernet Working Group.
Verfügbar unter: http://www.ieee802.org/3
Zugriff am: 01. 03. 2020

IEEE 2020
IEEE 802.1 Working Group, 2020.
Time-Sensitive Networking Task Group.
Verfügbar unter: `http://www.ieee802.org/1/pages/tsn.html`
Zugriff am: 01. 03. 2020

Imtiaz et al. 2008
Imtiaz, Jahanzaib; Jasperneite, Jürgen; Weber, Karl; Goetz, Franz-Josef; Lessmann, Gunnar, 2008.
A novel method for auto configuration of realtime Ethernet networks.
In: Bello, Lucia Lo; Vasques, Francisco (Hrsg.): *2008 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 861–868.
DOI: `10.1109/ETFA.2008.4638498`

IndustryWeek 2016
IndustryWeek Custom Research and Kronos Incorporated, 2016.
The Future of Manufacturing: 2020 and Beyond.
Verfügbar unter: `https://www.nist.gov/sites/default/files/documents/2016/11/16/iw_kronos_research_report_2016.pdf`
Zugriff am: 01. 03. 2020

Intel 2020
Intel Corporation, 2020.
Intel Ethernet-Controller I210-AT.
Verfügbar unter: `https://ark.intel.com/content/www/de/de/ark/products/64400/intel-ethernet-controller-i210-at.html`
Zugriff am: 01. 03. 2020

IETF 2020
Internet Engineering Task Force (IETF), 2020.
Deterministic Networking (DetNet) Working Group.
Verfügbar unter: `https://datatracker.ietf.org/wg/detnet/about/`
Zugriff am: 01. 03. 2020

Jahn 2017
Jahn, Myriam, 2017.
Industrie 4.0–Betriebswirtschaftliche Theorie und Praxis.
In: Jahn, Myriam (Hrsg.): *Industrie 4.0 konkret*,
S. 3–10

Jasperneite 2005
Jasperneite, Jürgen, 2005.
Echtzeit-Ethernet im Überblick.
Automatisierungstechnische Praxis (atp), **47** (3), S. 29–34.
ISBN 3-486-63052-0

Jasperneite et al. 2007 Jasperneite, Jürgen; Schumacher, Markus; Weber, Karl, 2007.
Limits of increasing the performance of Industrial Ethernet protocols.
In: Serpanos, Dimitrios; Hanisch, Hans-Michael; Bello, Lucia Lo (Hrsg.): *2007 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 17–24.
DOI: 10.1109/EFTA.2007.4416748

Jeschke et al. 2017 Jeschke, Sabine; Brecher, Christian; Song, Houbing; Rawat, Danda B., 2017.
Industrial Internet of Things: Cybermanufacturing Systems.
Cham: Springer.
ISBN 978-3-319-42559-7

Jiang et al. 2019 Jiang, Junhui; Li, Yuting; Hong, Seung Ho; Yu, Mengmeng; Xu, Aidong; Wei, Min, 2019.
A Simulation Model for Time-Sensitive Networking (TSN) with Experimental Validation.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 153–160.
DOI: 10.1109/ETFA.2019.8869206

Kagermann 2017 Kagermann, Henning, 2017.
Chancen von Industrie 4.0 nutzen.
In: Bauernhansl, Thomas; ten Hompel, Michael; Vogel-Heuser, Birgit (Hrsg.): *Handbuch Industrie 4.0 Bd. 4*,
S. 237–248.
DOI: 10.1007/978-3-658-04682-8_31

Kalør et al. 2018 Kalør, Anders Ellersgaard; Guillaume, Rene; Nielsen, Jimmy Jessen; Mueller, Andreas; Popovski, Petar, 2018.
Network slicing in Industry 4.0 applications: Abstraction methods and end-to-end analysis.
IEEE Transactions on Industrial Informatics, **14** (12), S. 5419–5427.
DOI: 10.1109/TII.2018.2839721

Karakus et al. 2017 Karakus, Murat; Durresi, Arjan, 2017.
Quality of Service (QoS) in Software Defined Networking (SDN): A survey.
Journal of Network and Computer Applications, **80**, S. 200–218.
DOI: 10.1016/j.jnca.2016.12.019

Kehrer et al. 2014 Kehrer, Stephan; Kleineberg, Oliver; Heffernan, Donal, 2014.
A comparison of fault-tolerance concepts for IEEE 802.1 Time Sensitive Networks (TSN).
In: Grau, Antoni; Martinez, Herminio (Hrsg.): *2014 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1–8.
DOI: 10.1109/ETFA.2014.7005200

Kirrmann et al. 2007 Kirrmann, Hubert; Hansson, Mats; Muri, Peter, 2007.
IEC 62439 PRP: Bumpless recovery for highly available, hard real-time industrial networks.
In: Serpanos, Dimitrios; Hanisch, Hans-Michael; Bello, Lucia Lo (Hrsg.): *2007 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1396–1399.
DOI: 10.1109/EFTA.2007.4416946

Kirrmann et al. 2011 Kirrmann, Hubert; Weber, Karl; Kleineberg, Oliver; Weibel, Hans, 2011.
Seamless and low-cost redundancy for substation automation systems (High-availability Seamless Redundancy, HSR).
In: Taylor, Kevin (Hrsg.): *2011 IEEE Power and Energy Society General Meeting*,
S. 1–7.
DOI: 10.1109/PES.2011.6038906

Koch 2016 Koch, Eckart, 2016.
Globalisierung: Wirtschaft und Politik.
Wiesbaden: Springer.
ISBN 978-3-658-08707-4

Koziolek 2017 Koziolek, Heiko, 2017.
Industrie 4.0 plug-and-produce for adaptable factories: Example use case definition models and implementation.
Bundesministerium für Wirtschaft und Energie (BMWi)

Lang et al. 2019 Lang, Dorota; Grunau, Sergej; Wisniewski, Lukasz; Jasperneite, Jürgen, 2019.
Utilization of the Asset Administration Shell to Support Humans During the Maintenance Process.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2019 17th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 768–773.
DOI: 10.1109/INDIN41052.2019.8972236

Lechler 2011
Lechler, Armin, 2011.
Konzeption einer funktional einheitlichen Applikationsschnittstelle für Ethernet-basierte Bussysteme.
Stuttgart: Jost-Jetter Verlag.
ISW/IPA-Forschung und Praxis, Bd. 184.
ISBN 978-3-939890-78-2.
Stuttgart, Univ., Diss., 2011

Lee 2008
Lee, Edward A., 2008.
Cyber physical systems: Design challenges.
In: Son, Sang Hyuk; Ezhilchelvan, Paul; Kim, Jung Guk; Shokri, Eltefaat (Hrsg.): *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*,
S. 363–369.
DOI: 10.1109/ISORC.2008.25

Lennvall et al. 2017
Lennvall, Tomas; Gidlund, Mikael; Åkerberg, Johan, 2017.
Challenges when bringing IoT into industrial automation.
In: Cornish, Darryn R. (Hrsg.): *2017 IEEE AFRICON*,
S. 905–910.
DOI: 10.1109/AFRCON.2017.8095602

Leong et al. 2018
Leong, Ong B.; Kumar, Anil; Sarwar, Usman, 2018.
A comparative analysis of Precision Time Protocol in native, virtual machines and container-based environments for consolidating automotive workloads.
Verfügbar unter: https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/other/d2-05_ong_comparative_analysis_of_precision_time_protocol.pdf
Zugriff am: 01. 03. 2020

Li et al. 2017
Li, Rixin; Wang, Renlei; Halachmi, Nir; Zhong, Qiwen; Cheng, WeiQiang; Wang, Lei; Wang, Jiao, 2017.
X-Ethernet: Enabling integrated fronthaul/backhaul architecture in 5G networks.
In: Latva-aho, Matti; Kantola, Raimo (Hrsg.): *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*,
S. 121–125.
DOI: 10.1109/CSCN.2017.8088609

Linux 2020
Linux Kernel Organization, 2020.
The Linux Kernel Archives.
Verfügbar unter: https://www.kernel.org/
Zugriff am: 01. 03. 2020

Lipnicki et al. 2018 Lipnicki, Piotr; Lewandowski, Daniel; Pareschi, Diego; Pakos, Waldemar; Ragaini, Enrico, 2018.
Future of IoTSP–IT and OT integration.
In: Younas, Muhammad; Disso, Jules Pagna (Hrsg.): *2018 IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*,
S. 203–207.
DOI: 10.1109/FiCloud.2018.00037

Löffler 2011 Löffler, Carina, 2011.
Systematik der strategischen Strukturplanung für eine wandlungsfähige und vernetzte Produktion der variantenreichen Serienfertigung.
Stuttgart: Jost-Jetter Verlag.
ISW/IPA-Forschung und Praxis, Bd. 519.
ISBN 978-3-939890-90-4.
Stuttgart, Univ., Diss., 2011

Marcon et al. 2017 Marcon, Petr; Zezulka, Frantisek; Vesely, Ivo; Szabo, Zita; Roubal, Zdenek; Sajdl, Ondrej; Gescheidtova, Eva; Dohnal, Premysl, 2017.
Communication technology for Industry 4.0.
In: Andronov, Ivan V. (Hrsg.): *2017 Progress In Electromagnetics Research Symposium-Spring (PIERS)*,
S. 1694–1697.
DOI: 10.1109/PIERS.2017.8262021

Maruyama et al. 2016 Maruyama, Yuya; Kato, Shinpei; Azumi, Takuya, 2016.
Exploring the performance of ROS2.
In: Eles, Petru; Mangharam, Rahul (Hrsg.): *Proceedings of the 13th International Conference on Embedded Software*,
S. 5.
DOI: 10.1145/2968478.2968502

McKeown 2009 McKeown, Nick, 2009.
Software-defined Networking.
INFOCOM keynote talk, **17** (2), S. 30–32

Meudt et al. 2017 Meudt, Tobias; Pohl, Malte; Metternich, Joachim, 2017.
Die Automatisierungspyramide-Ein Literaturüberblick.
Verfügbar unter: https://tuprints.ulb.tu-darmstadt.de/id/eprint/6298
Zugriff am: 01. 03. 2020

Mijumbi et al. 2016 Mijumbi, Rashid; Serrat, Joan; Gorricho, Juan-Luis; Bouten, Niels; de Turck, Filip; Boutaba, Raouf, 2016.
Network function virtualization: State-of-the-art and research challenges.
IEEE Communications Surveys & Tutorials, **18**, S. 236–262.
DOI: 10.1109/COMST.2015.2477041

Mizrahi et al. 2016
Mizrahi, Tal; Mayer, Danny, 2016.
Network Time Protocol Version 4 (NTPv4) Extension Fields.
Verfügbar unter: `https://tools.ietf.org/html/rfc7822`
Zugriff am: 01. 03. 2020

Monostori 2014
Monostori, László, 2014.
Cyber-physical production systems: Roots, expectations and R&D challenges.
Procedia CIRP, **17**, S. 9–13.
DOI: `10.1016/j.procir.2014.03.115`

Nasrallah et al. 2019
Nasrallah, Ahmed; Thyagaturu, Akhilesh S.; Alharbi, Ziyad; Wang, Cuixiang; Shao, Xing; Reisslein, Martin; Elbakoury, Hesham, 2019.
Performance Comparison of IEEE 802.1 TSN Time Aware Shaper (TAS) and Asynchronous Traffic Shaper (ATS).
IEEE Access, **7**, S. 44165–44181.
DOI: `10.1109/ACCESS.2019.2908613`

Nasrallah et al. 2018
Nasrallah, Ahmed; Thyagaturu, Akhilesh S.; Alharbi, Ziyad; Wang, Cuixiang; Shao, Xing; Reisslein, Martin; ElBakoury, Hesham, 2018.
Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research.
IEEE Communications Surveys & Tutorials, **21** (1), S. 88–145.
DOI: `10.1109/COMST.2018.2869350`

Nayak et al. 2017
Nayak, Naresh Ganesh; Dürr, Frank; Rothermel, Kurt, 2017.
Incremental flow scheduling and routing in time-sensitive software-defined networks.
IEEE Transactions on Industrial Informatics, **14** (5), S. 2066–2075.
DOI: `10.1109/TII.2017.2782235`

Nayak et al. 2016
Nayak, Naresh Ganesh; Dürr, Frank; Rothermel, Kurt, 2016.
Time-sensitive software-defined network (TSSDN) for real-time applications.
In: Plantec, Alain; Singhoff, Frank; Faucou, Sébastian; Pinho, Luís Miguel (Hrsg.): *Proceedings of the 24th International Conference on Real-Time Networks and Systems*,
S. 193–202.
DOI: `10.1145/2997465.2997487`

Norm DIN 44300
DIN 44300-1:1988-11.
Informationsverarbeitung - Begriffe - Allgemeine Begriffe

Norm DIN EN ISO 6385
DIN EN ISO 6385:2004-05.
Grundsätze der Ergonomie für die Gestaltung von Arbeitssystemen

Norm IEC 61784-2 IEC 61784-2:2019.
Industrial communication networks - Profiles - Part 2 - Additional fieldbus profiles for real-time networks based on ISO/IEC/IEEE 8802-3

Norm IEC 61800-7-1 IEC 61800-7-1:2015.
Adjustable speed electrical power drive systems - Part 7-1 - Generic interface and use of profiles for power drive systems - Interface definition

Norm IEC 62264-3 IEC 62264-3:2016.
Integration von Unternehmensführungs- und Leitsystemen - Aktivitätsmodelle für das Betriebsmanagement

Norm IEC 62439-3 IEC 62439-3:2016.
Industrial communication networks - High availability automation networks - Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)

Norm IEC 62541 IEC 62541:2016.
OPC Unified Architecture – Part 1-10

Norm IEC 62541-14 IEC 62541-14:2019.
OPC Unified Architecture - Part 14: PubSub

Norm IEC/IEEE 60802 IEC/IEEE 60802.
TSN Profile for Industrial Automation

Norm IEEE 1588 IEEE 1588.
Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

Norm IEEE 802.1AB IEEE 802.1AB.
IEEE Standard for Local and Metropolitan Area Networks-Station and Media Access Control Connectivity Discovery

Norm IEEE 802.1AS IEEE 802.1AS.
Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks

Norm IEEE 802.1AS-Rev IEEE 802.1AS-Rev.
Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications

Norm IEEE 802.1BA IEEE 802.1BA.
Standard for Local and Metropolitan Area Networks-Audio Video Bridging (AVB) Systems

Norm IEEE 802.1CB IEEE 802.1CB.
Standard for Local and Metropolitan Area Networks-Frame Replication and Elimination for Reliability

Norm IEEE 802.1CBcv IEEE 802.1CBcv.
FRER YANG Data Model and Management Information Base Module

Norm IEEE 802.1D IEEE 802.1D.
Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges

Norm IEEE 802.1DG IEEE 802.1DG.
TSN Profile for Automotive In-Vehicle Ethernet Communications

Norm IEEE 802.1Q IEEE 802.1Q.
Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks

Norm IEEE 802.1Qat IEEE 802.1Qat.
Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks - Amendment 9 - Stream Reservation Protocol (SRP)

Norm IEEE 802.1Qbu IEEE 802.1Qbu.
Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks - Amendment - Frame Preemption

Norm IEEE 802.1Qbv IEEE 802.1Qbv.
Standard for Local and Metropolitan Area Networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks - Amendment - Enhancements for Scheduled Traffic

Norm IEEE 802.1Qcc IEEE 802.1Qcc.
Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks - Amendment 9 - Stream Reservation Protocol (SRP) Enhancements and Performance Improvements

Norm IEEE 802.1Qcp IEEE 802.1Qcp.
Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 30 - YANG Data Model

Norm IEEE 802.1Qcw IEEE 802.1Qcw.
YANG Data Models for Scheduled Traffic, Frame Preemption, and Per-Stream Filtering and Policing

Norm IEEE 802.1Qdd IEEE 802.1Qdd.
Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment - Resource Allocation Protocol (RAP)

Norm IEEE 802.1w IEEE 802.1w.
Rapid Reconfiguration of Spanning Tree

Norm IEEE 802.3.2 IEEE 802.3.2.
IEEE Standard for Ethernet - YANG Data Model Definitions

Norm ISO/IEC 2382 ISO/IEC 2382:2015.
Information Technology - Vocabulary

Nsaibi et al. 2018
Nsaibi, Seifeddine; Leurs, Ludwig, 2018.
Abbildung von TDMA-basierten Industrial Ethernet Protokollen auf TSN am Beispiel von Sercos III.
In: Jasperneite, Jürgen; Lohweg, Volker (Hrsg.): *Kommunikation und Bildverarbeitung in der Automation*,
S. 122–135.
DOI: 10.1007/978-3-662-55232-2_10

Nsaibi et al. 2017a
Nsaibi, Seifeddine; Leurs, Ludwig; Schotten, Hans D., 2017a.
Formal and simulation-based timing analysis of Industrial Ethernet SERCOS III over TSN.
In: D'Ambrogio, Andrea; Grande, Robson E. De; Garro, Alfredo; Tundis, Andrea (Hrsg.): *Proceedings of the 21st International Symposium on Distributed Simulation and Real Time Applications*,
S. 83–90.
DOI: 10.1109/DISTRA.2017.8167670

Nsaibi et al. 2017b
Nsaibi, Seifeddine; Leurs, Ludwig; Schotten, Hans D., 2017b.
2017 Kommunikation in der Automation (KommA). Worst-case timing analysis of integrating TSN in the field-level.
Verfügbar unter: https://www.researchgate.net/publication/320709529_Formal_and_Simulation-based_Timing_Analysis_of_Industrial-Ethernet_Sercos_III_over_TSN
Zugriff am: 01. 03. 2020

Nyhuis et al. 2010
Nyhuis, Peter; Reinhart, Gunther; Abele, Eberhard, 2010.
Wandlungsfähige Produktionssysteme.
Garbsen: TEWISS.
ISBN 978-3-939-02696-9

Oktian et al. 2017
Oktian, Yustus Eko; Lee, SangGon; Lee, HoonJae; Lam, Jun-Huy, 2017.
Distributed SDN controller system: A survey on design choice.
Computer Networks, **121**, S. 100–111.
DOI: 10.1016/j.comnet.2017.04.038

Oliver et al. 2018
Oliver, Ramon Serna; Craciunas, Silviu S.; Steiner, Wilfried, 2018.
IEEE 802.1Qbv gate control list synthesis using array theory encoding.
In: Pellizzoni, Rodolfo (Hrsg.): *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*,
S. 13–24.
DOI: 10.1109/RTAS.2018.00008

OIF 2018
Optical Internetworking Forum (OIF), 2018.
Flex Ethernet 2.0 Implementation Agreement.
Verfügbar unter: https://www.oiforum.com/wp-content/uploads/2019/01/OIF-FLEXE-02.0-1.pdf
Zugriff am: 01. 03. 2020

Pahlevan et al. 2018a
Pahlevan, Maryam; Obermaisser, Roman, 2018a.
Evaluation of time-triggered traffic in time-sensitive networks using the OPNET simulation framework.
In: Merelli, Ivan; Liò, Pietro; Kotenko, Igor V. (Hrsg.): *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*,
S. 283–287.
DOI: 10.1109/PDP2018.2018.00048

Pahlevan et al. 2018b
Pahlevan, Maryam; Obermaisser, Roman, 2018b.
Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 337–344.
DOI: 10.1109/ETFA.2018.8502515

Pardo-Castellote 2003
Pardo-Castellote, Gerardo, 2003.
OMG Data-Distribution Service: Architectural overview.
In: McKinley, Philip K.; Shatz, Sol (Hrsg.): *2003 IEEE 23rd International Conference on Distributed Computing Systems Workshops (ICDCS)*,
S. 200–206.
DOI: 10.1109/MILCOM.2003.1290110

Pfrommer et al. 2018
Pfrommer, Julius; Ebner, Andreas; Ravikumar, Siddharth; Karunakaran, Bhagath, 2018.
Open Source OPC UA PubSub over TSN for Realtime Industrial Communication.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1087–1090.
DOI: 10.1109/ETFA.2018.8502479

Phemius et al. 2014
Phemius, Kevin; Bouet, Mathieu; Leguay, Jérémie, 2014.
Disco: Distributed multi-domain SDN controllers.
In: Pach, Andrzej R.; Zuckerman, Douglas N. (Hrsg.): *2014 IEEE Network Operations and Management Symposium (NOMS)*,
S. 1–4.
DOI: 10.1109/NOMS.2014.6838330

Pigan et al. 2008
Pigan, Raimond; Metter, Mark, 2008.
Automating with PROFINET: Industrial communication based on Industrial Ethernet.
Erlangen: Publicis Publishing.
ISBN 978-3-89578-294-7

Pop et al. 2018
Pop, Paul; Raagaard, Michael Lander; Gutierrez, Marina; Steiner, Wilfried, 2018.
Enabling fog computing for industrial automation through Time-Sensitive Networking (TSN).
IEEE Communications Standards Magazine, **2** (2), S. 55–61.
DOI: 10.1109/MCOMSTD.2018.1700057

PwC 2020
PricewaterhouseCoopers (PwC), 2020.
22nd Annual Global CEO Survey.
Verfügbar unter: https://www.pwc.com/gx/en/ceo-survey/2019/report/pwc-22nd-annual-global-ceo-survey.pdf
Zugriff am: 01. 03. 2020

Prinz et al. 2019a
Prinz, Frederick; Schoeffler, Michael; Eckhardt, Andreas; Lechler, Armin; Verl, Alexander, 2019a.
Configuration of Application Layer Protocols within Real-time I4.0 Components.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2019 17th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 971–976.
DOI: 10.1109/INDIN41052.2019.8972255

Prinz et al. 2019b
Prinz, Frederick; Schoeffler, Michael; Lechler, Armin; Verl, Alexander, 2019b.
Virtual network topologies for real-time I4.0 components based on Time-Sensitive Networking.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1388–1391.
DOI: 10.1109/ETFA.2019.8869074

Prinz et al. 2019c
Prinz, Frederick; Schoeffler, Michael; Lechler, Armin; Verl, Alexander, 2019c.
A novel I4.0-enabled engineering method and its evaluation.
The International Journal of Advanced Manufacturing Technology, **102** (5-8), S. 2245–2263.
DOI: 10.1007/s00170-019-03382-1

Prinz et al. 2018a Prinz, Frederick; Schoeffler, Michael; Lechler, Armin; Verl, Alexander, 2018a.
End-to-end redundancy between real-time I4.0 components based on Time-Sensitive Networking.
In: Seatzu, Carla; Mahulea, Cristian (Hrsg.): *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1083–1086.
DOI: 10.1109/ETFA.2018.8502553

Prinz et al. 2018b Prinz, Frederick; Schoeffler, Michael; Lechler, Armin; Verl, Alexander, 2018b.
Dynamic real-time orchestration of I4.0 components based on Time-Sensitive Networking.
Procedia CIRP, **72**, S. 910–915.
DOI: 10.1016/j.procir.2018.03.174

PNO 2017 PROFIBUS Nutzerorganisation e.V. (PNO), 2017.
The goals of integration of TSN into PROFINET.
Verfügbar unter: https://www.profibus.com/newsroom/news/the-goals-of-integration-of-tsn-into-profinet/
Zugriff am: 01. 03. 2020

Prytz 2006 Prytz, Gunnar, 2006.
Redundancy in Industrial Ethernet networks.
In: Cena, Gianluca; Vasques, Francisco (Hrsg.): *2006 IEEE International Workshop on Factory Communication Systems (WFCS)*,
S. 380–385.
DOI: 10.1109/WFCS.2006.1704183

Raagaard et al. 2017 Raagaard, Michael Lander; Pop, Paul; Gutiérrez, Marina; Steiner, Wilfried, 2017.
Runtime reconfiguration of Time-Sensitive Networking (TSN) schedules for fog computing.
In: Zhang, Tao (Hrsg.): *2017 IEEE Fog World Congress (FWC)*,
S. 1–6.
DOI: 10.1109/FWC.2017.8368523

Reinhart et al. 2010 Reinhart, Gunther; Krug, Sacha; Hüttner, Susanne; Mari, Zoltan et al., 2010.
Automatic configuration (plug & produce) of Industrial Ethernet networks.
In: Cardoso, Jose Roberto; Furukawa, Celso Massatoshi (Hrsg.): *2010 9th IEEE/IAS International Conference on Industry Applications (INDUSCON)*,
S. 1–6.
DOI: 10.1109/INDUSCON.2010.5739892

Rostan et al. 2010 Rostan, Martin; Stubbs, Joseph E.; Dzilno, Dmitry, 2010. EtherCAT enabled advanced control architecture. In: Schoenleber, Walter; Williams, Brett (Hrsg.): *2010 IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, S. 39–44. DOI: 10.1109/ASMC.2010.5551414

Roth 2016 Roth, Armin, 2016. *Einführung und Umsetzung von Industrie 4.0: Grundlagen, Vorgehensmodell und Use Cases aus der Praxis.* Berlin Heidelberg: Springer. ISBN 978-3-662-48505-7

Said et al. 2019 Said, Siwar Ben Hadj; Truong, Quang Huy; Boc, Michael, 2019. SDN-based configuration solution for IEEE 802.1 Time Sensitive Networking (TSN). *ACM SIGBED Review*, **16** (1), S. 27–32. DOI: 10.1145/3314206.3314210

Sanchez-Palencia 2018 Sanchez-Palencia, Jesus, 2018. *Time-based packet transmission.* Verfügbar unter: https://lwn.net/Articles/748744/ Zugriff am: 01. 03. 2020

Sauter et al. 2011 Sauter, Thilo; Soucek, Stefan; Kastner, Wolfgang; Dietrich, Dietmar, 2011. The evolution of factory and building automation. *IEEE Industrial Electronics Magazine*, **5** (3), S. 35–48. DOI: 10.1109/MIE.2011.942175

Schemm 2004 Schemm, Eberhard, 2004. SERCOS to link with Ethernet for its third generation. *Computing and Control Engineering*, **15** (2), S. 30–33. DOI: 10.1049/cce:20040205

Schönwälder et al. 2010 Schönwälder, Jurgen; Bjorklund, Martin; Shafer, Phil, 2010. Network configuration management using NETCONF and YANG. *IEEE Communications Magazine*, **48** (9), S. 166–173. DOI: 10.1109/MCOM.2010.5560601

Schriegel et al. 2018 Schriegel, Sebastian; Kobzan, Thomas; Jasperneite, Jürgen, 2018. Investigation on a distributed SDN control plane architecture for heterogeneous time sensitive networks. In: Cucu, Liliana; Seno, Lucia (Hrsg.): *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, S. 1–10. DOI: 10.1109/WFCS.2018.8402356

Schriegel et al. 2017 Schriegel, Sebastian; Pieper, Carsten; Gamper, Sergej; Biendarra, Alexander; Jasperneite, Jürgen, 2017.
2017 Kommunikation in der Automation (KommA). Vereinfachtes Ethernet TSN-Implementierungsmodell für Feldgeräte mit zwei Ports.
Verfügbar unter: `https://www.researchgate.net/publication/321244667_Vereinfachtes_Ethernet_TSN-Implementierungsmodell_fur_Feldgerate_mit_zwei_Ports`
Zugriff am: 01. 03. 2020

Schweissguth et al. 2016 Schweissguth, Eike; Danielis, Peter; Niemann, Christoph; Timmermann, Dirk, 2016.
Application-aware Industrial Ethernet based on an SDN-supported TDMA approach.
In: Bello, Lucia Lo; Behnam, Moris (Hrsg.): *2016 IEEE World Conference on Factory Communication Systems (WFCS)*,
S. 1–8.
DOI: 10.1109/WFCS.2016.7496496

Sercos 2020 Sercos International e.V., 2020.
Sercos over TSN.
Verfügbar unter: `https://www.sercos.de/technologie/sercos-in-verbindung-mit-tsn-opc-ua/sercos-in-verbindung-mit-tsn/`
Zugriff am: 01. 03. 2020

Siepmann et al. 2016 Siepmann, David; Graef, Norbert, 2016.
Industrie 4.0–Grundlagen und Gesamtzusammenhang.
In: Roth, Armin (Hrsg.): *Einführung und Umsetzung von Industrie 4.0*,
S. 17–82.
DOI: 10.1007/978-3-662-48505-7

Silva et al. 2019 Silva, Luis; Pedreiras, Paulo; Fonseca, Pedro; Almeida, Luis, 2019.
On the adequacy of SDN and TSN for Industry 4.0.
In: Gokhale, Aniruddha; Liu, Weichen; Pacher, Mathias (Hrsg.): *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*,
S. 43–51.
DOI: 10.1109/ISORC.2019.00017

Spath et al. 2013 Spath, Dieter; Ganschar, Oliver; Gerlach, Stefan; Hämmerle, Moritz; Krause, Tobias; Schlund, Sebastian, 2013.
Produktionsarbeit der Zukunft-Industrie 4.0.
Stuttgart: Fraunhofer Verlag.
ISBN 978-3-8396-0570-7

Stanton 2018
Stanton, Kevin B., 2018.
Distributing deterministic, accurate time for tightly coordinated network and software applications: IEEE 802.1AS, the TSN profile of PTP.
IEEE Communications Standards Magazine, **2** (2), S. 34–40.
DOI: 10.1109/MCOMSTD.2018.1700086

Steegmüller et al. 2014
Steegmüller, Dieter; Zürn, Michael, 2014.
Wandlungsfähige Produktionssysteme für den Automobilbau der Zukunft.
In: Bauernhansl, Thomas; ten Hompel, Michael; Vogel-Heuser, Birgit (Hrsg.): *Industrie 4.0 in Produktion, Automatisierung und Logistik*,
S. 103–119.
DOI: 10.1007/978-3-662-45279-0_23

Szancer 2017
Szancer, Sebastian, 2017.
Ein OMNeT++ Simulationsmodell für SERCOS III mit TSN Interface.
Hamburg: Fakultät Technik und Informatik.
Hamburg, Univ., Bachelorarbeit

Terzimehić et al. 2019
Terzimehić, Tarik; Bayha, Andreas; Dorofeev, Kirill, 2019.
Function Blocks for the Interaction with the Asset Administration Shell.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1235–1238.
DOI: 10.1109/ETFA.2019.8868995

Tiegelkamp et al. 2010
Tiegelkamp, Michael; John, Karl-Heinz, 2010.
IEC 61131-3: Programming industrial automation systems.
Berlin Heidelberg: Springer.
ISBN 978-3-642-12015-2

Tseng et al. 2014
Tseng, Mitchell M.; Hu, S. Jack, 2014.
Mass customization.
In: Laperrière, Luc; Reinhart, Gunther (Hrsg.): *CIRP Encyclopedia of Production Engineering*,
S. 836–843.
DOI: 10.1007/978-3-642-35950-7_16701-3

VDI/VDE 2013
VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, 2013.
Thesen und Handlungsfelder Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation

Veichtlbauer et al. 2017 Veichtlbauer, Armin; Ortmayer, Martin; Heistracher, Thomas, 2017.
OPC UA integration for field devices.
In: Bello, Lucia Lo; Sauter, Thilo; Luo, Ren (Hrsg.): *2017 15th IEEE International Conference on Industrial Informatics (INDIN)*,
S. 419–424.
DOI: 10.1109/INDIN.2017.8104808

Ventre et al. 2018 Ventre, Pier Luigi; Tajiki, Mohammad Mahdi; Salsano, Stefano; Filsfils, Clarence, 2018.
SDN architecture and southbound APIs for IPv6 segment routing enabled wide area networks.
IEEE Transactions on Network and Service Management, **15** (4), S. 1378–1392.
DOI: 10.1109/TNSM.2018.2876251

Vogel-Heuser 2017 Vogel-Heuser, Birgit, 2017.
Herausforderungen und Anforderungen aus Sicht der IT und der Automatisierungstechnik.
In: Bauernhansl, Thomas; ten Hompel, Michael; Vogel-Heuser, Birgit (Hrsg.): *Handbuch Industrie 4.0 Band 4*,
S. 33–44.
DOI: 10.1007/978-3-658-04682-8_2

Wagner et al. 2017 Wagner, Constantin; Grothoff, Julian; Epple, Ulrich; Drath, Rainer; Malakuti, Somayeh; Grüner, Sten; Hoffmeister, Michael; Zimermann, Patrick, 2017.
The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant.
In: Charalambous, Charalambos D.; Mahulea, Cristian (Hrsg.): *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 1–8.
DOI: 10.1109/ETFA.2017.8247583

Wan et al. 2016 Wan, Jiafu; Tang, Shenglong; Shu, Zhaogang; Li, Di; Wang, Shiyong; Imran, Muhammad; Vasilakos, Athanasios V., 2016.
Software-defined industrial internet of things in the context of Industry 4.0.
IEEE Sensors Journal, **16** (20), S. 7373–7380.
DOI: 10.1109/JSEN.2016.2565621

Wang et al. 2017 Wang, Yi; Ma, Hai-Shu; Yang, Jing-Hui; Wang, Ke-Sheng, 2017.
Industry 4.0: a way from mass customization to mass personalization production.
Advances in Manufacturing, **5** (4), S. 311–320.
DOI: 10.17261/Pressacademia.2017.486

Weber 2019
Weber, Karl, 2019.
EtherCAT and TSN – Best Practices for Industrial Ethernet System Architectures.
Verfügbar unter: `https://www.ethercat.org/download/documents/Whitepaper_EtherCAT_and_TSN.pdf`
Zugriff am: 01. 03. 2020

Westkämper et al. 2008
Westkämper, Engelbert; Zahn, Erich, 2008.
Wandlungsfähige Produktionsunternehmen: Das Stuttgarter Unternehmensmodell.
Berlin Heidelberg: Springer.
ISBN 978-3-540-68890-7

Westkämper et al. 2000
Westkämper, Engelbert; Zahn, Erich; Balve, Patrick; Tilebein, Meike et al., 2000.
Ansätze zur Wandlungsfähigkeit von Produktionsunternehmen. Ein Bezugsrahmen für die Unternehmensentwicklung im turbulenten Umfeld.
wt Werkstattstechnik Online, **90**, S. 22–26

Wiendahl et al. 2007
Wiendahl, Hans-Peter; ElMaraghy, Hoda A.; Nyhuis, Peter; Zäh, Michael F.; Wiendahl, Hans-Hermann; Duffie, Neil; Brieke, Michael, 2007.
Changeable manufacturing – classification, design and operation.
CIRP Annals, **56**, S. 783–809.
DOI: `10.1016/j.cirp.2007.10.003`

Wiendahl et al. 2014
Wiendahl, Hans-Peter; Reichardt, Jürgen; Nyhuis, Peter, 2014.
Handbuch Fabrikplanung: Konzept, Gestaltung und Umsetzung wandlungsfähiger Produktionsstätten.
M{ü}nchen: Carl Hanser Verlag.
ISBN 978-3-446-43892-7

Wisniewski 2017
Wisniewski, Lukasz, 2017.
New methods to engineer and seamlessly reconfigure time triggered Ethernet based systems during runtime based on the PROFINET IRT example.
Magdeburg: Springer.
ISBN 978-3-662-54650-5.
Magdeburg, Univ., Diss., 2017

Wollschlaeger et al. 2017
Wollschlaeger, Martin; Sauter, Thilo; Jasperneite, Jürgen, 2017.
The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0.
IEEE Industrial Electronics Magazine, **11** (1), S. 17–27.
DOI: `10.1109/MIE.2017.2649104`

Wörn 2006
Wörn, Heinz, 2006.
Echtzeitsysteme: Grundlagen, Funktionsweisen, Anwendungen.
Berlin Heidelberg: Springer.
ISBN 978-3-540-27416-2

Ye et al. 2019
Ye, Xun; Hong, Seung Ho, 2019.
Toward Industry 4.0 Components: Insights Into and Implementation of Asset Administration Shells.
IEEE Industrial Electronics Magazine, **13** (1), S. 13–25.
DOI: 10.1109/MIE.2019.2893397

Yin et al. 2012
Yin, Hongtao; Xie, Haiyong; Tsou, Tina; Lopez, Diego; Aranda, Pedro; Sidi, Ron, 2012.
SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains.
Verfügbar unter: https://datatracker.ietf.org/doc/draft-yin-sdn-sdni/
Zugriff am: 01. 03. 2020

Zäh et al. 2005
Zäh, Michael F.; Möller, Niklas; Vogl, Wolfgang, 2005.
Symbiosis of changeable and virtual production – the emperor's new clothes or key factor for future success.
Verfügbar unter: https://www.utzverlag.de/assets/pdf/40541ein.pdf
Zugriff am: 01. 03. 2020

Zezulka et al. 2018
Zezulka, František; Marcon, Petr; Bradac, Zdenek; Arm, Jakub; Benesl, Tomas; Vesely, Ivo, 2018.
Communication Systems for Industry 4.0 and the IIoT.
IFAC-PapersOnLine, **51** (6), S. 150–155.
DOI: 10.1016/j.ifacol.2018.07.145

Zhang et al. 2019
Zhang, Jiayi; Chen, Lihao; Wang, Tongtong; Wang, Xinyuan, 2019.
Analysis of TSN for Industrial Automation based on Network Calculus.
In: Indri, Marina; Vilanova, Ramon (Hrsg.): *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*,
S. 240–247.
DOI: 10.1109/ETFA.2019.8869053

Zhao et al. 2018
Zhao, Luxi; Pop, Paul; Zheng, Zhong; Li, Qiao, 2018.
Timing analysis of AVB traffic in TSN networks using network calculus.
In: Pellizzoni, Rodolfo (Hrsg.): *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*,
S. 25–36.
DOI: 10.1109/RTAS.2018.00009

Zimmermann 1980
Zimmermann, Hubert, 1980.
OSI reference model - the ISO model of architecture for open systems interconnection.
IEEE Transactions on Communications, **28** (4), S. 425–432.
DOI: 10.1109/9780470546543.ch49

Zoitl et al. 2014 Zoitl, Alois; Lewis, Robert, 2014.
Modelling control systems using IEC 61499.
Stevenage: IET.
ISBN 978-1-84919-760-1

Heutzutage agieren Unternehmen in einem immer volatileren Umfeld mit einem unmittelbaren Einfluss auf die Produktionssysteme. Um die Wandelbarkeit von zukünftigen Produktionssystemen zu verbessern, wird daher eine Automation basierend auf Cyber-Physischen Systemen (CPS) angestrebt.

Als Grundlage für eine CPS-basierte Automation wird in dieser Arbeit eine wandelbare, echtzeitfähige Kommunikationsinfrastruktur für zukünftige Produktionssysteme vorgestellt. Diese Kommunikationsinfrastruktur ermöglicht eine deterministische Kommunikation mit geringer Latenz und erfüllt die harten Echtzeitanforderungen in Produktionssystemen. Gleichzeitig wird die dynamische Konfiguration von Echtzeitverbindungen zur Laufzeit ermöglicht. Mit dem Fokus auf einer ganzheitlichen Lösung basiert die vorgestellte Kommunikationsinfrastruktur auf den drei wesentlichen Aspekten: Echtzeitkommunikation, Parametrierung und Netzwerkkonfiguration.

ISBN 978-3-8396-1743-4

9 783839 617434

FRAUNHOFER VERLAG