Applications

Birgit Vogel-Heuser, Yash Deshpande, Fandi Hartl, Jingyun Zhao, Xuezhou Hou, Dominik Hujo*, Theresa Prinz, Marko Pavlic, Darius Burschka, Klaus Bengler, Wolfgang Kellerer, André Kraft, Bernd Vojanec und Timo Markert

Modellierung und Validierung von Latenzzeiten in industriellen Agentensystemen mit Digitalen Zwillingen der KI.Fabrik

Modeling and validation of latency in industrial agent systems with digital twins in KI.Fabrik

https://doi.org/10.1515/auto-2023-0221
Empfangen Dezember 1, 2023; angenommen August 19, 2024

Zusammenfassung: Durch die zunehmende Automatisierung und Digitalisierung von Produktionsprozessen ist eine effiziente Kommunikation und Koordination in Agentensystemen von entscheidender Bedeutung. Diese Studie analysiert die Verzögerungszeiten der Kommunikations- und Koordinationsprozesse in Agentensystemen anhand von zwei industriellen Anwendungsfällen. Durch die Kategorisierung von Nachrichten werden die lokalen und Netzwerk-Verzögerungen jeder Nachricht modelliert, gemessen und verglichen. Die Ergebnisse zeigen, dass die Modellierung von Verzögerungszeiten vor der Implementierung zu einem effizienteren Ansatz bei der Ausarbeitung der harten und weichen Echtzeitfähigkeiten komplexer Systeme führen kann. Die Generalisierbarkeit der Ergebnisse ermöglicht es, die Ergebnisse auf wiederkehrende Module in Produktionssystemen anzuwenden.

Schlagwörter: Agentensystem; Kommunikations- und Koordinationsprozesse; Messung von Latenzzeiten; Modellierung von Latenzzeiten; Echtzeitfähigkeit; digitaler Zwilling

Abstract: In the context of increasing automation and digitization of production processes, efficient communication and coordination among agents in multi-agent-systems is very important. This study analyzes multi-agent-systems communication and coordination processes in two industrial use cases. By categorizing messages, local and network delays of each communication message are modeled, measured and compared. The results indicate that modeling delay times prior to implementation can lead to a more efficient approach to elaborate on the hard and soft real-time capabilities of a complex system. The generalization of the findings allows for their application to recurring modules within production systems.

Keywords: multi-agent-system; communication and coordination processes; delay measurement; delay modeling; real-time capabilities; digital twin

***Corresponding author: Dominik Hujo**, Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München, Germany,
E-mail: dominik.hujo@tum.de
Birgit Vogel-Heuser, **Fandi Hartl**, **Jingyun Zhao und Xuezhou Hou**, Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München, Germany, E-mail: vogel-heuser@tum.de (B. Vogel-Heuser), fandi.hartl@tum.de (F. Hartl), jingyun.zhao@tum.de (J. Zhao), xuezhou.hou@tum.de (X. Hou)
Yash Deshpande und Wolfgang Kellerer, Lehrstuhl für Kommunikationsnetze, Technische Universität München, Arcisstr. 21, 80333 München, Germany, E-mail: yash.deshpande@tum.de (Y. Deshpande), wolfgang.kellerer@tum.de (W. Kellerer)
Theresa Prinz und Klaus Bengler, Lehrstuhl für Ergonomie, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München, Germany, E-mail: theresa.prinz@tum.de (T. Prinz), bengler@tum.de (K. Bengler)
Marko Pavlic und Darius Burschka, Machine Vision and Perception Group, Technische Universität München, Boltzmannstr. 3, 85748 Garching bei München, Germany, E-mail: marko.pavlic@tum.de (M. Pavlic), burschka@tum.de (D. Burschka)
André Kraft, BMW AG, Lilienthalallee 25, 80788 München, Germany, E-mail: andre.kraft@bmw.de
Bernd Vojanec und Timo Markert, WITTENSTEIN SE, Zentrale Forschung & Entwicklung, Igersheim, Germany,
E-mail: bernd.vojanec@wittenstein.de (B. Vojanec), timo.markert@resense.io (T. Markert)

1 Einleitung

Agentensysteme (AS) ermöglichen eine effektive Umsetzung einer dezentralen Systemarchitektur für adaptive Produktionssysteme. Dies kann für große Systeme von Vorteil sein. Aufgrund der Größe dieser Systeme ist es jedoch schwierig, Bearbeitungszeiten und Übertragungsverzögerungen unmittelbar zu ermitteln. Die Modellierung oder Berechnung von zu übertragenden Daten kann verglichen mit realen Messungen zu Ungenauigkeiten und Verlust der Generalisierbarkeit führen. Die Planung dieser Systeme stellt in der Regel eine Herausforderung dar, da sie vor dem Bau des eigentlichen Systems erfolgen muss. Um die Wirksamkeit und den Nutzen agentenbasierter Systeme zu gewährleisten, sind Einschränkungen und potenzielle Herausforderungen dieser Systeme zu berücksichtigen. Zu den Einschränkungen zählen beispielsweise der Zielkonflikt zwischen dem Echtzeitverhalten und der Menge der auszutauschenden Daten [1]. Das Fähigkeitsmodell definiert die Kompetenzen, über die ein Agent verfügt, um spezifische Aufgaben erfolgreich auszuführen. In der domänenspezifischen Sprache (DSL) existieren Einschränkungen für die Modellierung größerer Systeme und Verzögerungen für roboterähnliche Systeme (DSL4RAS). In diesem Beitrag wird eine erweiterte und angepasste DSL-Version vorgestellt, um diese Einschränkungen zu beheben. Diese DSL-Version wird in zwei industriellen Anwendungen, darunter ihre digitalen Zwillinge (DZ), eingesetzt, und die Modellierungsergebnisse werden mit den gemessenen Bearbeitungs- und Verzögerungszeiten verglichen. Die Methode ermöglicht die Berechnung von Verzögerungszeiten modularer Systeme in einer agentenbasierten Umgebung.

Das Ziel dieser Arbeit ist die Optimierung eines Agentensystems, insbesondere die Untersuchung von Kommunikationsverzögerungen zwischen Agenten. Diese Verzögerungen können anschließend genutzt werden, um die Echtzeitfähigkeit zu optimieren, insbesondere für den DZ, der spezifische Anforderungen an die Zeitverzögerung hat. Um Kommunikationsverzögerungen zwischen AS zu untersuchen, sind einige Herausforderungen zu bewältigen. Die erste Herausforderung (H1) besteht darin, wie die Kommunikations- und Koordinationsprozesse auf Grundlage eines Produktionsprozesses, wie z. B. eines Montageprozesses, zerlegt und analysiert werden können. Zentralisierte Kommunikations- und Koordinationsprozesse bieten zwar zentrale Kontrollvorteile, benötigen jedoch eine leistungsstarke zentrale Recheneinheit, um die enorme Datenmenge zu bewältigen und umfangreiche Planungen durchzuführen [2]. Dezentralisierte Kommunikations- und Koordinationsprozesse hingegen könnten aufgrund des intensiven Nachrichtenaustauschs zwischen den Agenten und deren Mangel an Flexibilität bei der Prozessüberwachung und -steuerung ebenfalls ineffizient sein. Die zweite Herausforderung (H2) ist, wie die Nachrichten in einem AS klassifiziert werden können, um eine gezielte und effiziente Verarbeitung der Nachrichten zu ermöglichen. Nach der Klassifizierung der Nachrichten befasst sich die dritte Herausforderung (H3) damit, wie die Netzwerkverzögerungen jeder Nachricht gemessen und mit den modellierten Verzögerungen verglichen werden können. Das Ziel ist, die Genauigkeit des Modells der Vorhersage von Kommunikationsverzögerungen zu bewerten und potenzielle Unterschiede zwischen modellierten und gemessenen Verzögerungen zu untersuchen. Die vierte Herausforderung (H4) betrifft die Verwendung der geschätzten Verzögerungen in zukünftigen Modellen, um die Vorhersage von Verzögerungszeiten vor der Systemimplementierung zu verbessern. Es sollte geprüft werden, inwieweit die Ergebnisse auf ähnliche Module innerhalb von Produktionssystemen verallgemeinerbar und anwendbar sind.

Der Beitrag ist wie folgt gegliedert. Abschnitt 2 gibt einen Überblick über AS und DZ in der KI.Fabrik und die Anforderungen an die Kommunikationsverzögerung. In Abschnitt 3 wird der aktuelle Forschungsstand zu Netzwerkverzögerungen, AS und DSL4RAS erläutert. In Abschnitt 4 wird das Systemmodell für die Kommunikation zwischen Agenten in einem Netzwerk anhand von zwei industriellen Anwendungsfällen beschrieben. Abschnitt 5 zeigt die Implementierung zur Messung der Kommunikationsverzögerung auf und vergleicht die Ergebnisse mit den modellierten Verzögerungen. Die daraus resultierenden Ergebnisse werden in Abschnitt 6 ausführlich diskutiert, wobei die gemessenen Verzögerungen mit den Anforderungen an die Verzögerung verglichen werden, was auf Verbesserungsmöglichkeiten im AS hinweist. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick in Abschnitt 7.

2 Agentensysteme in der KI.Fabrik

Das Projekt KI.Fabrik strebt an, ein vollständig flexibler und vernetzter Standort für die lokale, krisensichere und wirtschaftliche Produktion von modernsten IT- und mechatronischen High-Tech-Komponenten zu sein. Es basiert auf einer modularen und teilautonomen Infrastruktur, die sich verschiedenen Anwendungsfällen anpassen kann. Aufgabenspezifische Agenten sind modular miteinander verbunden, um den industriellen Anwendungsfall umzusetzen. Eine digitale Nachbildung dieser Agenten wird erstellt, wobei die

Vernetzung zwischen den Agenten, die sich an verschiedenen Standorten befinden, durch das 5G-Netzwerk ermöglicht wird [3].

2.1 Zwei industrielle Anwendungsfälle der KI.Fabrik

In der KI.Fabrik gibt es zwei industrielle Anwendungsfälle: die Montage eines Getriebes von Partner A und die Vormontage des Cockpits von Partner B (siehe Abbildung 1). Der Anwendungsfall von Partner A umfasst die Montage eines zweistufigen Planetengetriebes mit geringem Verdrehspiel. Die Montage wird von autonomen Robotern durchgeführt. Die Schritte der Getriebemontage umfassen die Zusammenarbeit der Roboter beim Entnehmen von Teilen aus dem Lager, den Transport der Teile vom Lager zur Montagestation sowie den Montageprozess durch die Roboter. Dies beinhaltet Aufgaben wie die Objekterkennung, den Umgang mit empfindlichen Teilen, das Aufnehmen und Ablegen von Teilen sowie das Einsetzen von Stiften. Der Partner B-Cockpit-Vormontagefall umfasst die Vorinstallation und Integration verschiedener Komponenten und Systeme im Cockpit eines Fahrzeugs vor der endgültigen Montage. Dabei liegt der Fokus auf der Verkabelung im Cockpit durch Robotik. Ähnlich wie im Anwendungsfall von Partner A werden intelligente Logistikmethoden angewendet, um eine optimale Versorgung mit Komponenten sicherzustellen.

Eine Aufgabenanalyse beider Anwendungsfälle, ausgehend von der Montagesequenz eines menschlichen Experten, lieferte eine erste Aufgabentaxonomie der notwendigen Fähigkeiten. Dazu zählen rudimentäre Fähigkeiten wie Greifen, Bringen, Loslassen, aber auch komplexere Fähigkeiten wie das Einfügen von Gegenständen. Eine weitere bedeutende Fähigkeit ist die Erkennung und Lokalisation von Menschen und Objekten.

Alle diese Fähigkeiten werden von Roboter-Experten so verallgemeinert, dass sie auf unterschiedlicher Hardware eingesetzt werden können und zur Laufzeit an die Umgebungsbedingungen umgesetzt werden.

2.2 Softwarearchitektur und Digitaler Zwilling

Verschiedene Agenten kommunizieren über ein 5G-Netzwerk. Dabei werden Informationen durch Nachrichten ausgetauscht. Wie in Abbildung 2, unten rechts, dargestellt, verfügt jeder Agent über vier Module. Das Planungsmodul fasst die lokalen Ziele eines Agenten zusammen und entwickelt situationsabhängig eine Strategie, um diese zu erreichen. Das Kontrollmodul führt den spezifischen technischen Prozess des Agenten durch, während die Wissensbasis ein Modell des lokalen Wissens eines Agents beinhaltet. Das Diagnosemodul übernimmt sicherheitsgerichtete Aufgaben [1]. Die Central.AI fungiert als zentralisierter Agent und dient als Schnittstelle zwischen dem Kunden und der Fabrik. Die „Agenten zum Erkennen von Fähigkeiten“ unterteilen sich in „Agenten zur Prozessbeobachtung und -verständnis“ und „Agenten zur Anpassung von Fähigkeiten“. Die „Agenten

Abbildung 1: Zwei industrielle Anwendungsfälle: Partner A Getriebemontage (links) und Partner B Cockpit-Vormontage (rechts).

zur Prozessbeobachtung und -verständnis“ lernen neue Roboterfähigkeiten wie die Aufnahme von Objekten, indem sie die Bewegungen von Menschen nachahmen. Die „Agenten zur Anpassung von Fähigkeiten“ optimieren diese Roboterfähigkeiten durch die Anpassung von Parametern (wie z. B. dem Druck) an mehr als 50 Robotern. Die Central.AI erhält die neu erworbenen Roboterfähigkeiten von „Agenten zur Prozessbeobachtung und -verständnis“ sowie neue Werkzeuge wie Roboter-Greifer von „Generative.AI-Agenten“. Die Central.AI empfängt Aufträge und Anforderungen für kundenspezifische Produkte, plant die Abläufe anhand vorhandener Roboterfähigkeiten und koordiniert die Produktionsprozesse. Die Lager-, Transport- und andere Montage-Agenten sind dezentrale Agenten, die den Anweisungen von Central.AI folgen und untereinander kommunizieren, um Statusinformationen auszutauschen. Der DZ-Agent sammelt Informationen von anderen Agenten und aktualisiert die Informationen jedes einzelnen Agenten im DZ.

Die DZ verfolgen die Veränderungen jedes Agenten durch die digitale Abbildung der physischen Assets, wodurch die Datenverfolgung dieser Assets und ihrer Prozesse kontinuierlich aktualisiert wird. Für jeden Agenten wird ein DZ erstellt, der sowohl Engineeringdaten als auch Betriebsdaten enthält. Der DZ ermöglicht auch die Echtzeitüberwachung, Analyse und Optimierung der technischen Prozesse. Ein DZ-System, das auf automatisierter DZ-Generierung und flexibler plattformübergreifender Kommunikation basiert, wurde entwickelt, um die nahtlose Integration physischer und digitaler Assets zu gewährleisten (siehe Abbildung 3) [4]. In diesem DZ-System werden Betriebsdaten, wie zum Beispiel Gelenkwerte für einen Roboter, in den DZ in der Cloud (MS Azure) hochgeladen. Die Daten werden als historische Daten im Event Hub gespeichert und können dort direkt abgefragt werden. Anschließend werden die Daten heruntergeladen, um den technischen Prozess in Visualisierungssoftware (Omniverse) darzustellen.

Das KI-Portal in der Avatar-Station fungiert für den menschlichen Operator als Kommandozentrale, durch das einerseits Betriebs- und Prozessdaten der Agenten bereitgestellt werden und andererseits Operator mittels Teleoperation aktiv in Prozesse in der Fabrik eingreifen können. Der zentrale Kontrollagent sendet auf Anfrage Informationen über vergangene, aktuelle und geplante Betriebs- und Prozesszustände an das KI-Portal, wo sie mittels Omniverse in der Avatar-Station visualisiert werden. Weiter ermöglicht es die Avatar-Station dem Operator die Steuerung der Montageagenten in der Fabrik zu übernehmen. Der DZ überwacht in beiden Fällen den Materialfluss sowie die Produktionsprozesse. Die kontinuierliche Überwachung und Analyse der Betriebsdaten im DZ ermöglichen die Überprüfung des Montageprozesses auf reibungslosen Ablauf und eine frühzeitige Fehlererkennung. Die Informationsdarstellung kann bezüglich Informationsbreite, -tiefe und dem Zeitpunkt adaptiert werden. Bei der Teleoperation ermöglicht der DZ eine zeitliche Entkopplung von der Kommandoeingabe durch den Operator und der Kommandoausführung in der Fabrik.

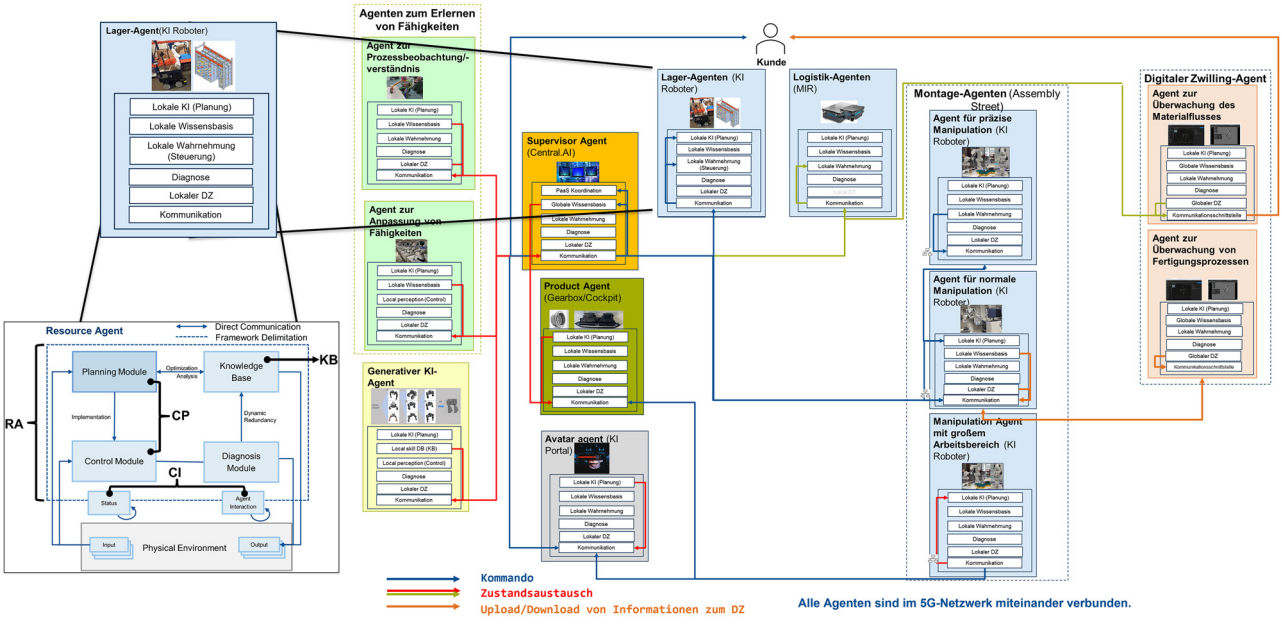

Abbildung 2: Die Softwarearchitektur für Agentensysteme.

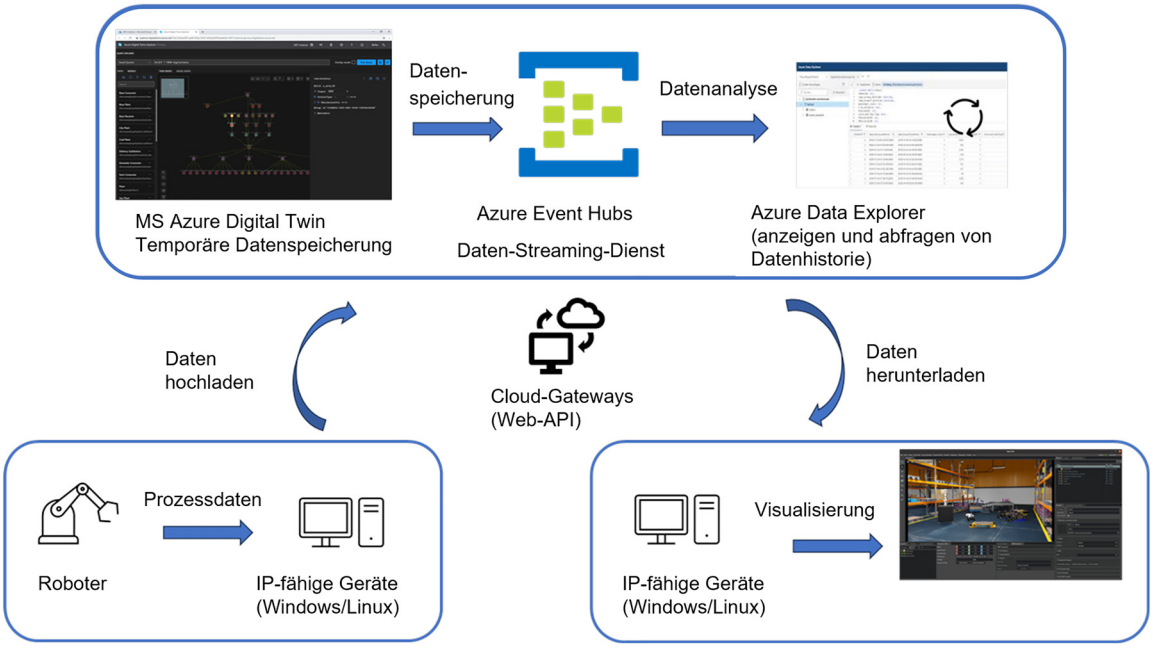

Abbildung 3: Datenfluss des DZ vom physischen Asset zu MS Azure und Omniverse.

2.3 Anforderungen an die Latenzzeiten

In der Roboterkollaboration ist es entscheidend, dass die Roboter den aktuellen Status der anderen kennen, insbesondere während des Kooperations- und Übergabeprozesses. Die Statusinformationen der Agenten werden regelmäßig ausgetauscht und auch im DZ aktualisiert, um die Kommunikation zwischen den Agenten zu überwachen und mögliche Fehler, wie beispielsweise ein Überschreiten der Kommunikationszeit zwischen Robotern, frühzeitig zu erkennen. Dies ermöglicht eine synchronisierte Zusammenarbeit und verhindert potenzielle Ineffizienzen in der Roboterkollaboration [5].

Die Echtzeitanforderung an den DZ in der Avatar-Station ist zweigeteilt. Die Bereitstellung von Betriebs- und Prozessdaten wird entweder vom Operator angefordert oder zustandsbezogen vom System angestoßen. Hier genügt eine Bereitstellung im Millisekundenbereich. Bei der Teleoperation ist die Gesamtlatenz für den Operator entscheidend. Diese besteht hauptsächlich aus der Verzögerung bei der Datenerhebung an Sensoren, der Datenverarbeitung und des Netzwerks zwischen beteiligten Agenten. Die Echtzeit-Anforderung für die Teleoperation ist abhängig von der Art der Steuerung. Für closed-loop (direkte), kollaborative oder überwachende Steuerung gelten unterschiedliche Grenzen, von Millisekunden bis Sekunden. Neben der Latenz ist die Synchronisierung der Signale unterschiedlicher Feedbackmodalitäten (visuell, auditiv, haptisch) für eine effektive Mensch-System Interaktion entscheidend.

Die Anforderungen an die Latenzzeit für nicht echtzeitkritische Anwendungsfälle sind weniger strikt als die Anforderungen an das DZ-Modell. In diesem Kontext genügt es, wenn die Nachrichtenübertragung von einem Agenten zum anderen innerhalb von Millisekunden (<10 ms) erfolgt.

3 Stand der Technik

Im Folgenden werden die handelsüblichen Ethernet-Netzkomponenten und verteilten Protokolle für die Echtzeitkommunikation vorgestellt und ihre Grenzen erläutert. Es werden Steuerungsstrategien für AS, die mittels der DSL hinsichtlich der zeitlichen Merkmale und Anforderungen beschrieben werden, wie Cyber Physical Production Systems (CPPS) und DZ, vorgestellt.

3.1 Netzwerklatenzen

Standardmäßige Commercial off-the-shelf software (COTS)-Ethernet-Netzwerkkomponenten und verteilte Protokolle sind nicht für die Echtzeitkommunikation geeignet. Die durchschnittliche Einweglatenz (One-way-delay, OWD) im Netz kann niedrig sein. Einzelne Ausreißer können jedoch sehr hoch sein. Diese verteilten Internet-Protokoll-Netze (IP-Netze) können keine zuverlässige und rechtzeitige Informationsübermittlung gewährleisten.

Moderne Fabriken mit AS benötigen eine höhere Bandbreite für Anwendungen wie kollektives Lernen und Punktwolken-Updates für Augmented Reality (AR) und Virtual Reality (VR) des DZ. Bestehende Industrienetzwerke, die Echtzeitgarantien bieten, können solch hohe Bandbreiten nicht bereitstellen. Des Weiteren ist es erforderlich, dass Agenten-basierende Endpunkte möglicherweise auf Betriebssystemen (OS) wie Robot OS (ROS) [6]. Solche Betriebssysteme benötigen ein vollständiges IP-Netzwerk und laufen normalerweise auf COTS-Geräten und nicht auf SPS-basierten Echtzeit-Steuerungen.

In einem einfachen Montagebeispiel ist es notwendig, viele kurze Nachrichten zwischen den Agenten auszutauschen, um den Beginn und den Abschluss einzelner Aufgaben im gesamten Montageprozess zu koordinieren. Die Nachrichten müssen im Sinne der geforderten Quality of Service (QoS) zuverlässig übertragen werden [7] und die Zustellung vom empfangenden Agenten verifiziert werden. Das Transmission Control Protocol (TCP) sorgt dafür, dass die Koordinationsnachrichten der Agenten zuverlässig, geordnet und fehlergeprüft sind. Eine bestimmte TCP-Nachrichtenfolge wird als Fluss bezeichnet. Es wurden verschiedene Methoden zur Modellierung der Nachrichtenlaufzeit eines TCP-Flusses vorgeschlagen [8], [9]. Die in diesem Beitrag beschriebenen Nachrichten sind typischerweise von kleiner Größe (wenigen Bytes), sodass sie in ein einzelnes Paket „passen“. Die Durchlaufzeit solcher Nachrichten kann durch ein geeignetes Modell bestimmt werden, welches die Zeitspanne des Handshakes während der TCP-Initiierung berücksichtigt [10]. Mit einem solchen Modell lässt sich die erwartete Latenz in den unterschiedlichen Phasen des TCP vergleichen. In Abschnitt 4.3 wird ein Modell für die Verarbeitungszeit für eine gegebene TCP-Verbindung in einer Beispielfabrik mit AS vorgeschlagen.

3.2 Agentensysteme und DSL4RAS

AS bestehen aus mehreren selbstverwalteten Agenten, die zusammenarbeiten, um ein gemeinsames Ziel zu erreichen oder ein Problem zu lösen. Jeder Agent innerhalb einer AS ist in der Lage, seine Umgebung wahrzunehmen und mit anderen Agenten zu kommunizieren [11]. In Produktionssystemen wurden AS-Steuerungsstrategien entwickelt, die die Anpassungsfähigkeit komplexer und dynamischer Fertigungssysteme verbessern [12]. Agenten in einem modernen industriellen Netzwerk nutzen die Socket-Schnittstelle, um Daten zu senden. Sockets verwenden das schichtweise Open Systems Interconnection Model (OSI Referenzmodell) [13], um Daten zu senden und zu empfangen. Abbildung 4 zeigt die Schichtensequenz, die ein Paket vom Socket in Agent A bis zum Socket in Agent B durchläuft [14].

Es gibt einen Trend zur Verwendung von DSL und Modellen, um visuelle Notationen zu standardisieren, die den Fachleuten der Industrieautomatisierung vertraut sind [15]. In der frühen Entwurfsphase ermöglichen modellübergreifende Ansätze eine transparentere Untersuchung der Systemleistung. Die genaue Beschreibung der zeitlichen Eigenschaften und Anforderungen modularer und verteilter Systeme ist im Zusammenhang mit CPPS und DZ von entscheidender Bedeutung [16]. Hujo et al. [17] führen eine Erweiterung einer etablierten Notation ein, die die Untersuchung der Timing-Eigenschaften und -Anforderungen von Automatisierungslösungen erleichtert und sie durch die Einbeziehung der Beziehung zwischen physikalischen Geräten und ihrer Steuerungssoftware auf der Grundlage von weichen und harten Echtzeitfähigkeiten erweitert. Der Großteil der Sensor- und Aktorverzögerungen wurde aufgrund der begrenzten Informationen in den entsprechenden Datenblättern durch Messungen ermittelt. Bei Agenten, die den DZ einbeziehen, ist die Anzahl der

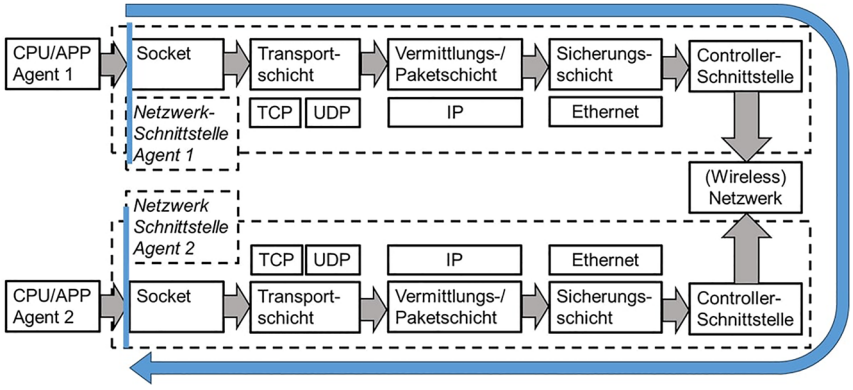

Abbildung 4: Der mehrschichtige Netzwerkstapel auf dem Agent-Host. Das Paket geht von einer Anwendung zur anderen durch jede Schicht. Dies führt zu Verzögerungen. In diesem Beitrag werden einfache Verzögerungsmodelle für die Zeit der Nachrichtenübermittlung zwischen Agenten vorgeschlagen. Das Netzwerk kann entweder ein kabelgebundenes Netzwerk, ein drahtloses Netzwerk oder ein Netzwerk über Internetverbindung sein.

verfügbaren Referenzpunkte noch begrenzt. Durch die Referenzpunkte im DZ gelangen die Daten zur weiteren Verarbeitung in die DZ-Software. Die Referenzpunkte befinden sich auf dem Cloud-Server und im Omniverse und sind in der Regel für Endnutzer nicht sichtbar. Ein sichtbarer Referenzpunkt sind die Programmierschnittstellen (engl. Application Programming Interfaces, APIs) der DZ-Software. Der vorgeschlagene Ansatz besteht darin, die im physischen System gemessenen Verzögerungszeiten mit den von den Modellen vorhergesagten zu vergleichen, um genauere Informationen über die zeitlichen Eigenschaften des Systems zu erhalten.

3.3 Modellierung der Aufgabenverteilung

Im Bereich der Robotik wird die Aufgabenverteilung mithilfe von Fähigkeiten in drei Ebenen unterteilt: Aufgaben-, Fähigkeits-, und Primitivebenen (engl. Tasks, skills and primitives, siehe Abbildung 5, links) [18]. Die Primitivebene enthält die Echtzeit-Regelschleifen des Roboters sowie die erforderlichen Sensormessungen, die für das Funktionieren der Fähigkeiten notwendig sind. Fähigkeiten sind Kombinationen von Primitiven, die dem Benutzer zugänglich sind und ausgeführt werden können. Sie liegen verallgemeinert vor, um den Transfer von Fähigkeiten zwischen verschiedenen Robotern zu erleichtern. Eine Aufgabe ist wiederum eine Abfolge von Fähigkeiten, die direkt mit der Erfüllung spezifischer Ziele in der Fabrik verbunden werden und gemeinsam mit dem Endbenutzer definiert werden.

Im Bereich der Automatisierung werden die Fähigkeiten, Fertigkeiten und Dienstleistungen (engl. Capabilities, Skills and Services Model, CSS-Modell, siehe Abbildung 5, rechts) unter der Produkt-, Prozess- und Ressourcenstruktur (PPR) verwendet [19]. Fähigkeiten spezifizieren die Produktionsfunktionen einer Ressource, während Fertigkeiten die Umsetzung dieser Fähigkeiten darstellen. Die Fertigkeiten beinhalten Details auf der Ebene der Implementierung und des Aufrufs von Automatisierungsfunktionen. Die Fähigkeiten werden den Stakeholdern in einer erweiterten Supply Chain außerhalb einer Produktionsumgebung mit einem Shopfloor in Form von Dienstleistungen angeboten. In diesem Beitrag verwenden wir die Definition aus der Robotik.

4 Systemmodell

Im folgenden Abschnitt werden die Annahmen für das Netzwerk, der industrielle Aufbau und eine detaillierte Beschreibung des Anwendungsfalls vorgestellt.

4.1 Netzwerk

Es wird davon ausgegangen, dass der Netzverkehr über COTS-Hardware abgewickelt wird. Jedes Gerät wird durch eine IP-Adresse identifiziert und Pakete aufgrund von Überlastung im Netzwerk verworfen. Eine zentrale Überwachung und Steuerung des Fabriknetzwerks kann dazu beitragen, ein kontrolliertes Netzwerk zu erreichen. Es wird Chameleon [20] verwendet, um sicherzustellen, dass keine Pakete verworfen werden und die gesamte Kommunikationslatenz im Netzwerk stabil und begrenzt ist. Folgende Vereinfachungen wurden im Netzwerk getroffen, um die Komplexität zu reduzieren: Annahme konstanter Latenzzeiten von Agent A zu B während des gesamten Montageprozesses. Die Ende-zu-Ende-Reaktionszeit nach VDI/VDE 2192 misst die Dauer vom Stimulus bis zur Antwort an den definierten Referenzpunkten innerhalb eines Agenten (Abbildung 6, grüne Linie) [21]. Im Vergleich dazu wird die Kommunikationslatenz als die Zeitdauer der Pakete im Netzwerk von Agent A zu Agent B definiert (Abbildung 6, blaue Linie) [22].

Zukünftige Arbeiten werden sich mit den Verzögerungen auseinandersetzen, die sich aus Fehlern und der Neukonfiguration von Netzwerkpfaden ergeben. Die meisten agentenbasierten Systeme sind auf dem Betriebssystem Linux implementiert, mit Ausnahme des auf Azure Analysis Services (AAS) basierenden DZ. Es wird die Cubic-TCP-Variante [23] verwendet, da sie für den schnellen Betrieb mit kurzen Nachrichten optimiert ist. Für eine vereinfachte

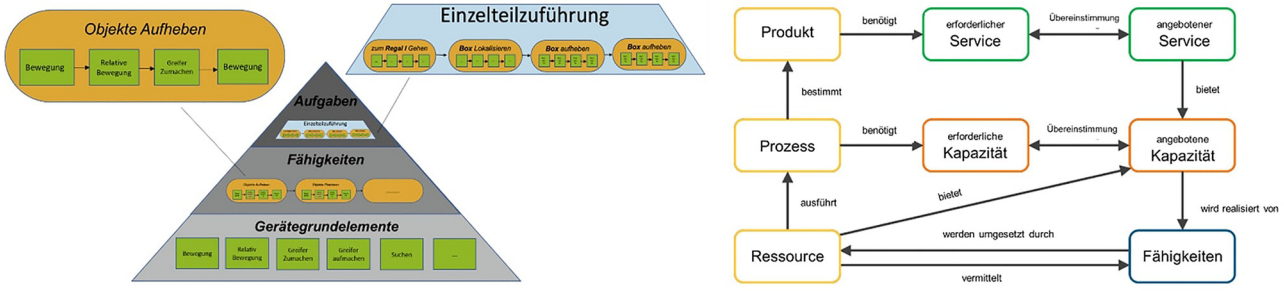

Abbildung 5: Modellierung der Aufgabenverteilung in der Robotik (links, [18]) und in der Automatisierung (rechts, [19]).

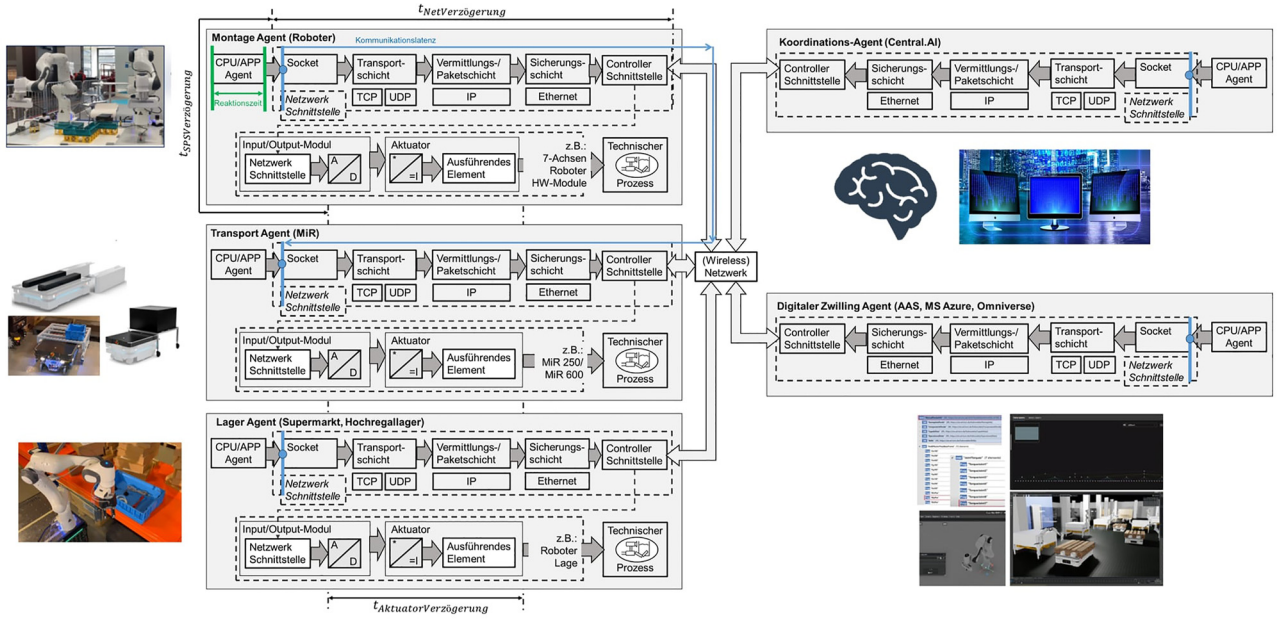

Abbildung 6: DSL-Modellierung [15], [16] mit den Laborimplementierungen der Agenten für den Anwendungsfall. Die Referenzpunkte sind mit blauen Strichen markiert, an diesen Punkten werden die Nachrichten über Sockets gesendet und empfangen.

Evaluierung der Modelle ist ein einziger Netzwerk-Switch erforderlich, der alle Agenten in diesem Netzwerk mit Ausnahme des DZ verbindet.

4.2 Anwendungsfall

Als Beispielanwendung für CPPS wurde ein Anwendungsfall ausgearbeitet. Der Systemaufbau umfasst einen Lageragenten (Supermarktzelle), einen Koordinationsagenten („Central.AI“), einen Logistikagenten (MiR-Transportroboter), einen Montageagenten (KI Roboter) und den digitalen Zwillingsagenten (implementiert durch Verwaltungsschale, MS Azure Digital Twin und Nvidia Omniverse), (vgl. Abbildung 6). In diesem Kontext werden die Echtzeitfähigkeiten des Systems sowie die mittleren Latenzsfaktoren, die mit den verschiedenen Kommunikationskanälen zusammenhängen, analysiert.

4.3 Beschreibung des Anwendungsfalls mit DSL

In dieser Untersuchung wird der industrielle Einsatz von zwei Getriebekomponenten (Partner A) und die Montage der Kabel in Panels (Partner B) für Kundenaufträge untersucht. Eine DSL für die Modellierung von Produktionssystemen (DSL4Production) wird auf den in Abbildung 6 dargestellten Montage-Anwendungsfall angewendet. Der Workflow (vgl. Abbildung 3) beinhaltet die Zerlegung des Kundenauftrags in Aufgaben, die wiederum in Fähigkeiten und Primitives zerlegt werden. Das AS überprüft die Fähigkeiten und Verfügbarkeit der benötigten Agenten und aktualisiert gleichzeitig ihren individuellen DZ und den zentralen Planungsagenten nach Erledigung jeder Aufgabe. Dadurch wird der aktuelle Status jedes Agenten aktualisiert, sodass jeder Agent im DZ die Statusinformationen anderer Agenten abrufen kann, um Informationen auszutauschen. Die Verwendung der Aufgabenzerlegung, der Agentenzuweisung und der Verfolgung des DZ ermöglicht die Bewertung der automatisierten Auftragsabwicklung, der Latenzzeiten und der Ressourcenauslastung. Im Rahmen der DSL4Production werden die meisten Latenzzeiten durch Messungen ermittelt. Es ist jedoch mit einem erheblichen Aufwand verbunden, jede Kommunikation einzeln zu messen. Die Kommunikationslatenz zwischen zwei Agenten, gemessen an deren Referenzpunkten (Abbildung 6), wird als Verzögerungszeit gemessen. Gesamtsystemverzögerungen resultieren aus der Addition aller Einzelkommunikationslatenzen zwischen den Agenten in einem Anwendungsfall [22]. Nachdem die Modellierung der Systemverzögerung überprüft wurde und die gemessenen Verzögerungszeiten mit den modellierten Verzögerungen korrelieren, können die gewonnenen Erkenntnisse in mathematische Gleichungen integriert werden. Dies erleichtert den Prozess für weitere Verzögerungsschätzungen und kann als Grundlage für die Optimierung des Designs von AS dienen.

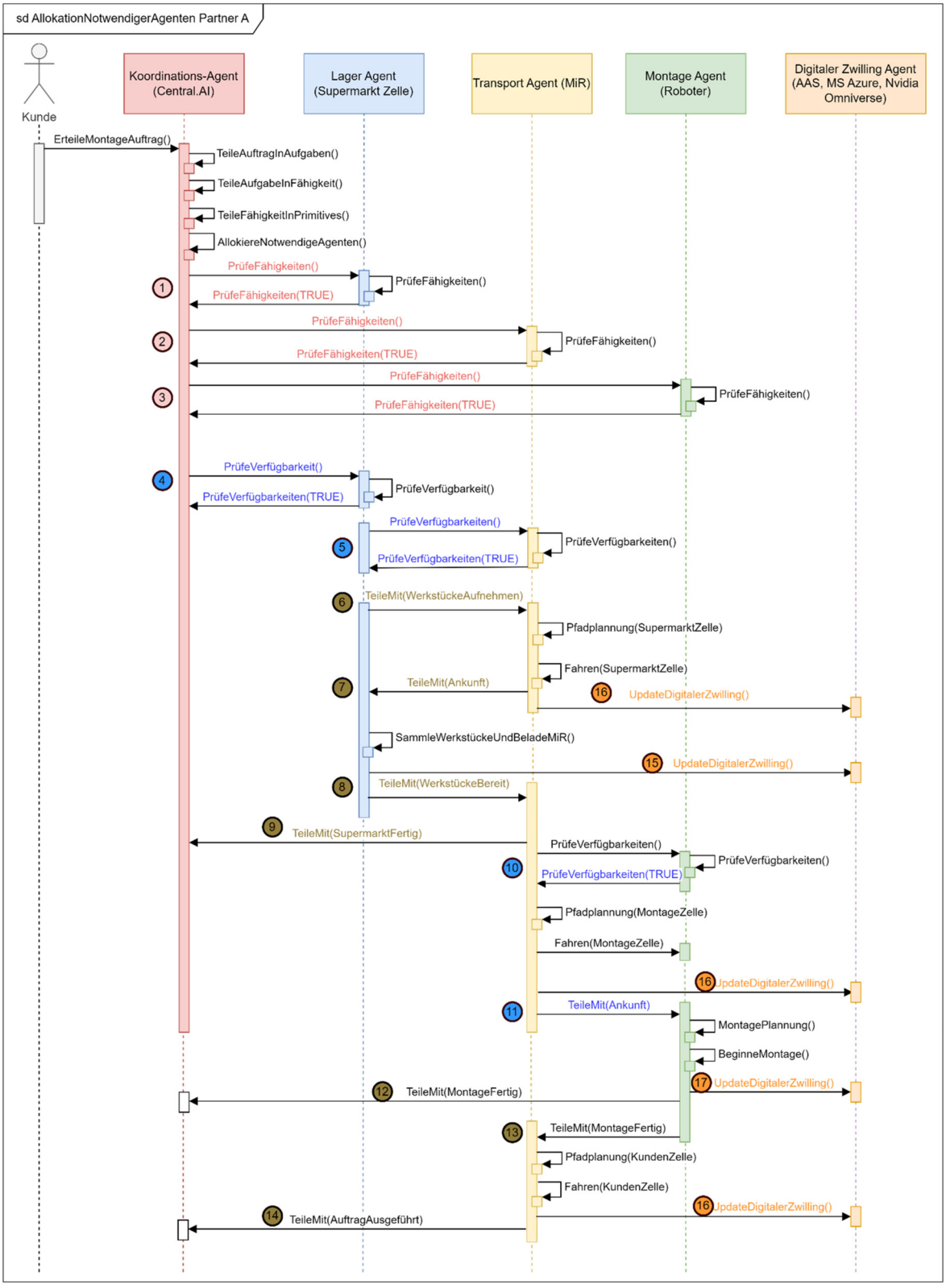

Abbildung 7: Sequenzdiagramm des Montageprozesses von zwei Getriebeteilen in der AS-Architektur (Partner A).

5 Analyse

Im Folgenden werden die experimentellen Ergebnisse dargestellt und diskutiert.

5.1 Zerlegung der Agentensystemkommunikation

Die Analyse konzentriert sich ausschließlich auf die Netzwerklatenz beim Nachrichtenaustausch zwischen Agenten und nicht auf Bearbeitungszeiten des Montageprozesses. Die Wartezeiten zwischen Montageschritten können sich durch längere Kommunikationslatenzen verlängern und somit die Montageprozesse verzögern. Eine Nichteinhaltung der Echtzeitgrenzen kann zu Kollisionen von zwei Robotern und Maschinenschäden führen. Die Dekomposition des Systems basiert auf dem Montageprozess (vgl. Abbildung 7 und 8). Die Messung der Kommunikationslatenz erfolgt entlang der in der Abbildung dargestellten Verbindungspfeile, vom Senden der Nachricht durch einen Agenten bis zum Empfang oder zur Antwort eines anderen Agenten. Die Kommunikation wurde nach der Sequenzierung der Sender- und Empfängeragenten in unterschiedliche Kommunikationstypen, Nachrichtenkategorien und spezifische Abfrageinhalte unterteilt (siehe Tabelle 1 und Tabelle 2). Die Kommunikation kann nach ihrer Art klassifiziert werden, zum Beispiel in Bezug auf Verfügbarkeitsanfragen, detaillierte Anfragen zu Fähigkeiten und primitiven Daten, Arbeitsaufgaben oder Statusmeldungen, wie zum Beispiel die des Montage- oder Logistikagenten.

Das Sequenzdiagramm für die Koordination und Kommunikation des AS-Systems ist in Abbildung 7 (Partner A) und 8 (Partner B) dargestellt. Die Nummern kennzeichnen die Kommunikationsschritte zwischen den Agenten. Der Koordinations-Agent (Central.AI) erhält Aufträge von Kunden und erstellt den Prozessplan. Dazu sendet er Nachrichten an andere Agenten und überprüft, ob sie die erforderlichen Fähigkeiten haben (Nachrichten 1–3), um das kundenspezifische Produkt herzustellen. Für den Anwendungsfall von Partner B müssen auch Anzahl und Position der verwendeten Halterungen in Central.AI berechnet werden. Anschließend überprüft der Supervisor-Agent die Verfügbarkeit der Supermarkt-Roboter und Transportroboter (Nachrichten 4–5) und gibt dem Supermarkt-Agenten die Anweisung, die benötigten Teile abzuholen (Nachricht 6). Daraufhin beauftragt der Supermarkt-Agent

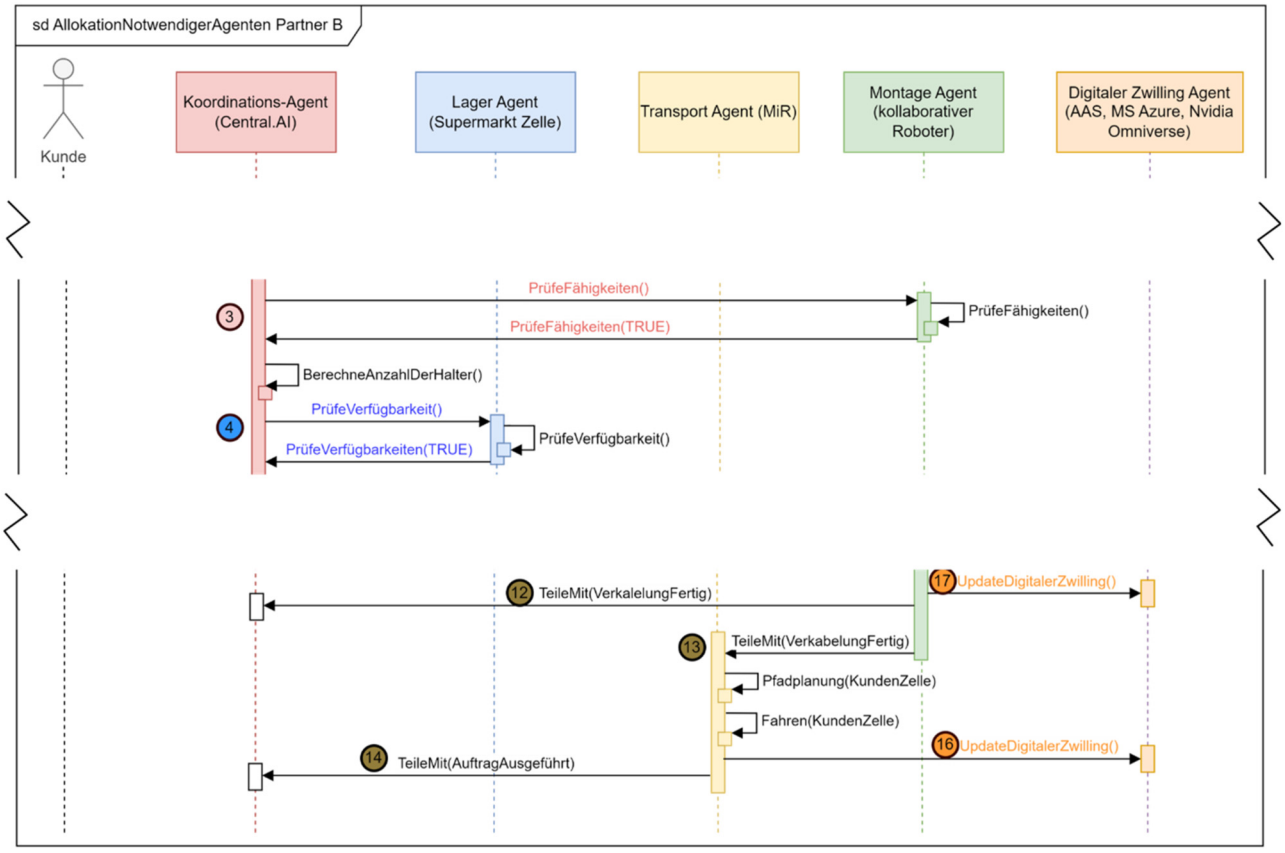

Abbildung 8: Sequenzdiagramm des Montageprozesses der Verkabelung in der AS-Architektur (Partner B).

Tabelle 1: Überblick über die Nachrichten zwischen Agenten (Partner A).

Ind.	Name und Kontext	Sender	Empfänger	Kommu. Protokoll	Nach. Typ	Abfrageinhalt	Gemessene gemittelte Latenzzeit (ms)
				Central.AI plant Aufgaben für Supermarktzelle, MiR und Roboter für Montage.			
1	**Capability_SupCell**: Central.AI prüft die Fähigkeiten des Supermarktroboters.	CenAI	SupCell	TCP	str	{“Task”: “Assembly”, “Product”: “Gearbox”, “Required resources”: [“1,000 –122079b01000_Zahnrad_LP070.stp”, “1,000 –122036B01000 Bolzen LP090.stp”, “1000 –122323b01000_Ritzel_LP070_i10.stp”], “Required time”: “60s”}	0,495
				TCP	byte	“SupCell can do task assembly.”	
2	**Capability_MiR**: Central.AI prüft die Fähigkeiten des Transroboters MiR.	CenAI	MiR	TCP	str	{“Task”: “Assembly”, “Product”: “Gearbox”, “Required time”: “120s”}	0,495
				TCP	byte	“MiR can do task assembly.”	
3	**Capability_RobAssem**: Central.AI prüft die Fähigkeiten des Montageroboters.	CenAI	RobAssem	TCP	str	{“Task”: “Assembly”, “Product”: “Gearbox”, “Required procedures”: [“pick up item”, “assembly”, “place item”], “Required time”: “300s”}	0,530
				TCP	byte	“RobotAssem can do task assembly.”	
				MiR entnimmt Teile aus Supermarktzelle.			
4	**Avalability_SupCell**: Central.AI prüft die Verfügbarkeit des Supermarktroboters.	CenAI	SupCell	TCP	byte	“Is SupCell free?”	0,555
				TCP	byte	“SupCell is free.”	
5	**Avalability_MiR**: Supermarktzelle prüft die Verfügbarkeit des Transroboters MiR.	SupCell	MiR	TCP	byte	“Is MiR free?”	0,474
				TCP	byte	“MiR is free.”	
6	**SupCell_MiR_PickUpParts**: Supermarktroboter meldet MiR für Teileaufnahme.	SupCell	MiR	TCP	byte	“MiR picks up parts.”	0,269
7	**MiR_SupCell_AtSupCell**: MiR meldet Ankunft bei Supermarktzelle.	MiR	SupCell	TCP	byte	“MiR is at SupCell.”	0,265
8	**SupCell_MiR_PartsReady**: SupCell meldet MiR Teilebereitschaft.	SupCell	MiR	TCP	byte	“SupCell finishes task collect parts.”	0,291
9	**SupCell_CenAI_Done**: Supermarktzelle informiert Central.AI über abgeschlossene Teileentnahme.	SupCell	CenAI	TCP	byte	“SupCell finishes task collect parts.”	0,264

Tabelle 1: (continued)

Ind.	Name und Kontext	Sender	Empfänger	Kommu. Protokoll	Nach. Typ	Abfrageinhalt	Gemessene gemittelte Latenzzeit (ms)
			MiR bringt Teile zum Roboter für Montage und lässt Roboter Montageaufgabe erledigen.				
10	**Avalability_RobAssem**: Central.AI prüft die Verfügbarkeit des Montageroboters.	CenAI	RobAssem	TCP	byte	"Is RobAssem free?"	0,605
				TCP	byte	"RobAssem is free."	
11	**MiR_RobAssem_AtRobotAssem**: MiR meldet Ankunft beim Montageroboter.	MiR	RobAssem	TCP	byte	"MiR is at robot assembly"	0,303
12	**RobAssem_CenAI_Done**: Montageroboter meldet Central.AI Montageende.	RobAssem	CenAI	TCP	byte	"RobAssem finishes task assembly."	0,325
13	**RobAssem_MiR_Done**: Montageroboter meldet Montageende an MiR.	RobAssem	MiR	TCP	byte	"MiR picks up assembled parts."	0,304
14	**MiR_CenAI_Delivery**: MiR meldet Central.AI Lieferbereitschaft.	MiR	CenAI	TCP	byte	"MiR finishes task delivery."	0,309
			Supermarktzelle, MiR und Roboter aktualisieren ihre digitalen Zwillinge.				
15[a]	**SupCell_UpdateDT**: Supermarktzelle aktualisiert Betriebsdaten in DZ.	SupCell	DT	HTTP	json	"[{'op': 'replace', 'joint0': 'radius', 'value': 0.7899125}, …, 'position theta': 'radius', 'value': 0.2145814}]"	917,352
16	**MiR_UpdateDT**: MiR aktualisiert Betriebsdaten in DZ.	MiR	DT	HTTP	json	"[{'op': 'replace', 'x': 'm', 'value': 10.688876152038574}, 'op': 'replace', 'y': 'm', 'value': 31.457216262817383}, 'op': 'replace', 'theta': 'degree', 'value': 101.11448669433594}]"	815,126
17[a]	**RobAssem_UpdateDT**: Montageroboter aktualisiert Betriebsdaten in DZ.	RobAssem	DT	HTTP	json	"[{'op': 'replace', 'joint0': 'radius', 'value': 0.1254187349436}, …, 'position theta': 'radius', 'value': 0.3654874215457}]"	933,470

[a] Der Nachrichteninhalt enthält Informationen über Gelenke, Greifer und Position des Roboters, jeweils mit 12 Freiheitsgraden. DT: digital twin.

Tabelle 2: Überblick über die Nachrichten zwischen Agenten (Partner B).

Ind.	Name und Kontext	Sender	Empfänger	Kommu. Protokoll	Nach. Typ	Abfrageinhalt	Gemessene gemittelte Latenzzeit (ms)
				Central.AI plant Aufgaben für Supermarktzelle, MiR und Roboter für Verkabelung.			
1	**Capability_SupCell:** Central.AI prüft die Fähigkeiten des Supermarktroboters.	CenAI	SupCell	TCP	str	{“Task”: “Cable routing”, “Product”: “Cockpit panel with cable”, “Required resources”: [“Cable_red”, “10 fixtures”], “Required time”: “120s”}	0,468
				TCP	byte	“SupCell can do task cable routing.”	
2	**Capability_MiR**: Central.AI prüft die Fähigkeiten des Transroboters MiR.	CenAI	MiR	TCP	str	{“Task”: “Cable routing”, “Product”: “Cockpit panel with cable”, “Required time”: “120s”}	0,546
				TCP	byte	“MiR can do task cable routing.”	
3	**Capability RobCable:** Central.AI prüft die Fähigkeiten des Verkabelungsroboters.	CenAI	RobCable	TCP	str	{“Task”: “Cable routing”, “Product”: “Cockpit panel with cable”, “Required procedures”:[“pick up cable”, “move cable”, “fix cable in fixture”, “place cable”], “Required time”: “00s”}	0,597
				TCP	byte	“RobCable can do task cable routing.”	
				MiR entnimmt Teile aus Supermarktzelle.			
4	**Avalability_SupCell**: Central.AI prüft die Verfügbarkeit des Supermarktroboters.	CenAI	SupCell	TCP	byte	“Is SupCell free?”	0,556
				TCP	byte	“SupCell is free.”	
5	**Avalability_MiR**: Supermarktzelle prüft die Verfügbarkeit des Transroboters MiR.	SupCell	MiR	TCP	byte	“Is MiR free?”	0,449
				TCP	byte	“MiR is free.”	
6	**SupCell_MiR_PickUpParts**: Supermarktroboter meldet MiR für Teileaufnahme.	SupCell	MiR	TCP	byte	“MiR picks up parts.”	0,264
7	**MiR_SupCell_AtSupCell**: MiR meldet Ankunft bei Supermarktzelle.	MiR	SupCell	TCP	byte	“MiR is at SupCell.”	0,241
8	**SupCell_MiR_PartsReady**: SupCell meldet MiR Teilebereitschaft.	SupCell	MiR	TCP	byte	“SupCell finishes task collect parts.”	0,285
9	**SupCell_CenAI_Done**: Supermarktzelle informiert Central.AI über abgeschlossene Teileentnahme.	SupCell	CenAI	TCP	byte	“SupCell finishes task collect parts.”	0,287

Tabelle 2: (continued)

Ind.	Name und Kontext	Sender	Empfänger	Kommu. Protokoll	Nach. Typ	Abfrageinhalt	Gemessene gemittelte Latenzzeit (ms)
				MiR bringt Teile zum Roboter für Verkabelung und lässt Roboter Verkabelungsaufgabe erledigen.			
10	**Avalability_RobCable**: Central.AI prüft die Verfügbarkeit des Verkabelungsroboters.	CenAI	RobCable	TCP	byte	“Is RobCable free?”	0,593
				TCP	byte	“RobCable is free.”	
11	**MiR_RobCable_AtRobCable**: MiR meldet Ankunft beim Verkabelungsroboter.	MiR	RobCable	TCP	byte	“MiR is at robot cable routing.”	0,276
12	**RobCable_CenAI_Done**: Verkabelungsroboter meldet Central.AI Verkabelungsende.	RobCable	CenAI	TCP	byte	“RobCable finishes task cable routing.”	0,305
13	**RobCable_MiR_Done** Verkabelungsroboter meldet Verkabelungsende an MiR.	RobCable	MiR	TCP	byte	“MiR picks up cable routing parts.”	0,294
14	**MiR_CenAI_Delivery**: MiR meldet Central.AI Lieferbereitschaft.	MiR	CenAI	TCP	byte	“MiR finishes task delivery.”	0,289
				Supermarktzelle, MiR und Roboter aktualisieren ihre digitalen Zwillinge.			
15[a]	**SupCell_UpdateDT**: Supermarktzelle aktualisiert Betriebsdaten in DZ.	SupCell	DT	HTTP	json	“[{‘op’: ‘replace’, ‘joint0’: ‘radius’, ‘value’: 0.1625733}, …, ‘position theta’: ‘radius’, ‘value’: 0.3657283}]”	659,524
16	**MiR_UpdateDT**: MiR aktualisiert Betriebsdaten in DZ.	MiR	DT	HTTP	json	“[{‘op’: ‘replace’, ‘x’: ‘m’, ‘value’: 9.345246164257473}, ‘op’: ‘replace’, ‘y’: ‘m’, ‘value’: 17.145748362453819}, ‘op’: ‘replace’, ‘theta’: ‘degree’, ‘value’: 90.98263548109376}]”	653,278
17[a]	**RobCable_UpdateDT:** Verkabelungsroboter aktualisiert Betriebsdaten in DZ.	RobCable	DT	HTTP	json	“[{‘op’: ‘replace’, ‘robleftjoint0’: ‘radius’, ‘value’: 0.1827364029351}, …, ‘robrightposition theta’: ‘radius’, ‘value’: 0.1938201938543}]”	856,615

[a] Der Nachrichteninhalt enthält Informationen über Gelenke, Greifer und Position des Roboters, jeweils mit 12 Freiheitsgraden. DT: digital twin.

den Transport-Agenten, die Teile zum Montage-Agent zu bringen (Nachrichten 7–8). Der Montage-Agent informiert die Transport-Agenten, sobald diese fertig sind und die Transport-Agenten liefern dann das Produkt an die Kundenzellen (Nachricht 9). Wichtige Statusupdates, wie z. B. das Beenden ihrer Aufgaben, werden an den Supervisor-Agent, Transport-Agenten oder Montage-Agenten gemeldet (Nachrichten 12–14). Zudem wird der aktuelle Status jedes Agenten, einschließlich seiner Position, dem DZ-Agenten übermittelt (Nachrichten 15–17).

Alle Nachrichten sind vom Typ TCP. Die Nachrichten 15–17 werden jedoch über das Gateway zur AAS Cloud Hosted DZ versendet. Die restlichen Nachrichten werden über das lokale Netzwerk übertragen. Die Klassifizierung der Nachrichten wird aus dem Anwendungsfall des Montageprozesses abgeleitet, erfolgt für alle Agenten, die keine DZ-Agenten sind, auf Ebene 4 (Transportebene) und für DZ-Agenten auf Ebene 7 (Anwendungsebene) gemäß dem OSI-Referenzmodell. In diesem Anwendungsfall werden die lokalen Nachrichten (Tabelle 1 und Tabelle 2) in drei Kategorien eingeteilt:

(1) Nachrichten 1–3 beginnen umgehend mit der Verbindung und schließen diese sofort (bestätigte Dienste, verbindungsorientierte Verbindungen).
(2) Nachrichten 4, 5, 10 und 15–17 initiieren die Verbindung, stellen die Verbindung her, schließen sie aber nicht (unbestätigte Dienste, verbindungsorientierte Verbindungen).
(3) Nachrichten 6–9 und 11–14 senden die Nachricht über eine bereits bestehende TCP-Verbindung und beenden diese (unbestätigte Dienste, verbindungsorientierte Verbindungen).

5.2 Messung der Verzögerungen

Zur Messung des Netzwerks und seiner Verzögerung werden Zeitstempel der Anwendungsschicht am Anfang und Ende einer Nachricht verwendet. Die Anwendungen des Sender- und Empfängeragenten werden auf derselben Computerhardware ausgeführt. Dadurch können Fehler in den Verzögerungsmessungen ausgeschlossen werden, die durch unsynchronisierte Uhren zwischen den Agenten entstehen. Die CLOCK REALTIME übernimmt die Zeitstempel auf einem Linux-Betriebssystem. Diese Zeit wird über einen Network Time Protocol (NTP)-Daemon, der auf dem Host-Computer läuft, mit dem DZ synchronisiert.

Die Nachrichten, die die Betriebsdaten des Montageroboters übermitteln und darstellen, werden im realen System mithilfe eines Roboters und eines Datenzentrums in MS Azure erfasst. Für alle anderen Nachrichten, die keine Betriebsdaten, sondern nur Kommandos umfassen, wird nur im Testaufbau simuliert. Es wird ein Computer mit 16-Core Intel i9 Prozessoren und 32 Gigabyte RAM verwendet, um die CPUs nicht übermäßig auszulasten. Der Sender- und Empfänger-Agent verfügt über eigene Netzwerkschnittstellen-Controller, die an separaten Peripheral Component Interconnect Express (PCIe)-Steckplätzen auf der Hauptplatine des Computers angeschlossen sind. Die Ethernet-Ports und IP-Adressen aus verschiedenen Netzwerk-Namensräumen werden getrennt, da der Linux-Kernel sonst die Pakete intern weiterleitet. Das Beispielnetzwerk eines 1-Gbps-Ethernet-Switches ist zwischen dem Sender- und dem Empfängerport angeschlossen.

Dieser Ansatz garantiert zuverlässige Verzögerungsmessungen, da für alle Messungen die gleiche CPU-Leistung und Netzwerkhardware verwendet wird. Zur Bestimmung der Verzögerungszeit wurden für jede Nachricht 100 Messungen durchgeführt. Die Verteilung der Messwerte für jeden Nachrichtentyp ist in Abbildung 9 (Partner A) and 10 (Partner B) und Abbildung 10 (Partner B) dargestellt. Die mittleren Verzögerungszeiten der einzelnen Nachrichten sind in Tabelle 1 und Tabelle 2 aufgeführt. Alle lokalen Nachrichten wurden mit einer Verzögerung von weniger als 1 ms gemessen, was auf die Verwendbarkeit des Systems in der Echtzeitkommunikation hinweist, wo solche Verzögerungen tolerierbar sind. Die Verzögerungen bei den Nachrichten mit dem DZ waren um drei Größenordnungen höher als die zwischen den anderen drei Agenten. Diese sind in Abbildung 11 (Partner A) und Abbildung 12 (Partner B) dargestellt. Die Verzögerungsmessungen weisen auch auf die Tatsache hin, dass sich die Latenz als Engpass für die Echtzeitkommunikation erweisen könnte, wenn Cloud-basierte

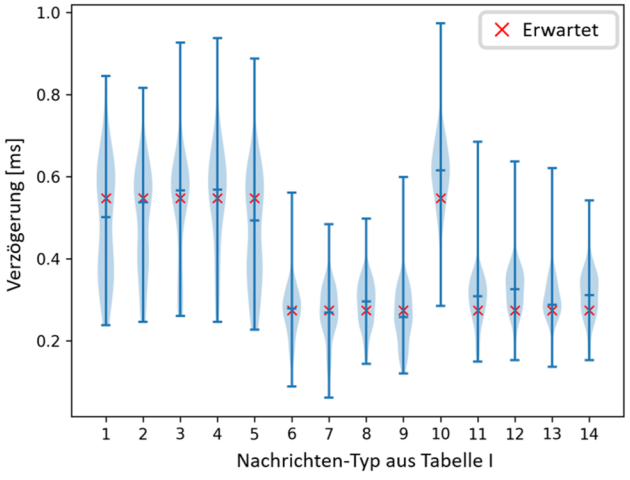

Abbildung 9: Gemessene Kommunikationsverzögerung für alle lokalen Nachrichten. Der gemessene Mittelwert korreliert in den meisten Fällen mit dem in Tabelle 2 angegebenen geschätzten Mittelwert (Partner A).

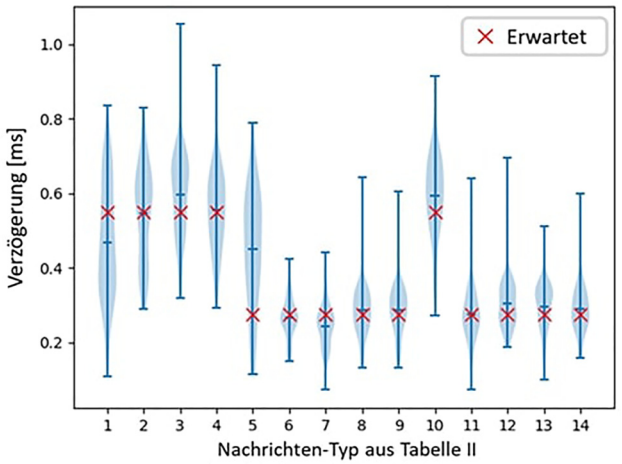

Abbildung 10: Gemessene Kommunikationsverzögerung für Nachrichten im Zusammenhang mit DZ (Partner A). Diese Verzögerungen sind viel höher, da sie im lokalen Netz nicht vorhanden sind und die Antwortzeit der Anwendungsschicht auf die API-Aufrufe an den DZ unbekannt ist.

DZ-Lösungen synchron mit der Produktionsumgebung eingesetzt werden sollen (Abbildung 10).

5.3 Modellierung der Verzögerungen

Das Modell aus [11, Gleichung 25] vereinfacht sich durch die Verzögerung durch Paketverluste und den langsamen Start des TCP-Algorithmus. Die gesamte erwartete Zeit für die Fertigstellung einer Nachricht für einen TCP-Fluss m ist gegeben durch

$$E[C_m] = E[D_{\text{init}}] + nE[D_{\text{paket}}],$$

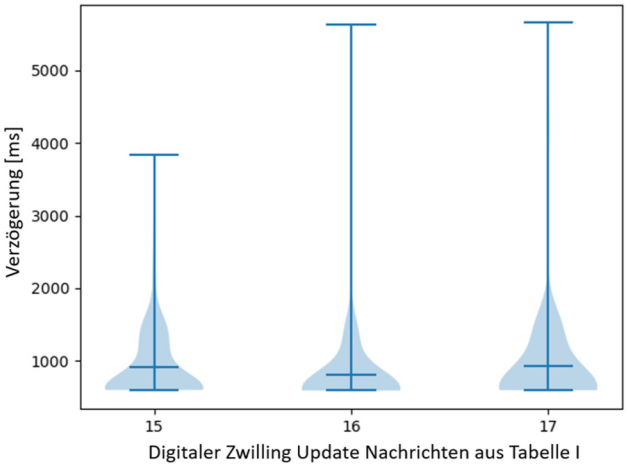

Abbildung 11: Gemessene Kommunikationsverzögerung für alle lokalen Nachrichten. Der gemessene Mittelwert korreliert in den meisten Fällen mit dem in Tabelle 2 angegebenen geschätzten Mittelwert (Partner B).

wobei $E[D_{\text{init}}]$ die Zeit ist, die benötigt wird, um den 3-Wege-Handshake abzuschließen und $E[D_{\text{paket}}]$ die Zeit ist, die benötigt wird, um ein Paket zu senden und seine Bestätigung zu erhalten. Da der Zeitablauf der Nachrichten äußerst knapp ist, wird die Nachricht mit dem Inhalt „FIN“ für den Fluss mit dem letzten Paket als Unterbrechung des Flusses verschickt und kein verzögerter Bestätigungscode (ACK)-Mechanismus eingesetzt. Die Paketgrößen für die Nachrichten in Tabelle 1 und Tabelle 2 betragen maximal 3 Byte. Der Zeitunterschied zwischen der Übertragung eines Pakets von 1 Byte und eines Pakets von 3 Byte über eine 1-Gbps-Verbindung beträgt 16 Nanosekunden. Aus diesem Grund wird die Dauer der Zustellung sämtlicher Pakete als dieselbe betrachtet. Die Round Trip Time (RTT) ist das Maß (in Millisekunden) für die Netzwerklatenz, welche die Zeit zwischen der Initiierung einer Netzwerkanfrage und dem Erhalt einer Antwort misst [10]. Da es nur einen Switch zwischen allen lokalen Agenten im Netz gibt, der nicht überlastet ist, ist diese RTT gleich lang. Die RTT wird mit einer einfachen Ping-Nachricht bewertet, die auf allen Linux-Betriebssystemen verfügbar ist. Die mittlere RTT wurde hier durch 100 Experimente mit 0,274 ms gefunden. Dieser Wert wird zusammen mit den in Abschnitt 4.1 erläuterten Nachrichtentypen aus Tabelle 1 verwendet. Die geschätzten mittleren Zeiten für jede Nachricht sind in Tabelle 3 aufgeführt.

5.4 Vergleich der gemessenen und der modellierten Verzögerungen

In der Mehrzahl der Fälle korreliert das Modell mit den Messungen. Alle theoretischen mittleren Verzögerungszeiten liegen innerhalb einer Standardabweichung des gemessenen Wertes. Obwohl die gemessene Kommunikationslatenz zwischen den Agenten im Millisekundenbereich liegt, bestimmt die Einhaltung der erforderlichen Latenzgrenzen, ob die nachfolgenden Montageschritte wie geplant initiiert werden können. Auffällig ist die Differenz zwischen Modellvorhersage und Messungen für Nachrichten mit dem DZ aus Abbildung 11 (Partner A) und Abbildung 12 (Partner B), die mehrere Größenordnungen beträgt. Die mittlere RTT von jedem Agenten zum DZ wurde mit 10 ms gemessen. Dies deutet darauf hin, dass die Antworten auf die API-Aufrufe

Tabelle 3: Geschätzte Verzögerungszeiten für Nachrichtentypen aus Tabelle 1 und Tabelle 2.

Nachrichten	$E[C_m]$	Verzögerungszeit (ms)
1–3	$2 \cdot E[RTT]$	0,548
4, 5, 10	$2 \cdot E[RTT]$	0,548
6–9, 11–14	$1 \cdot E[RTT]$	0,274

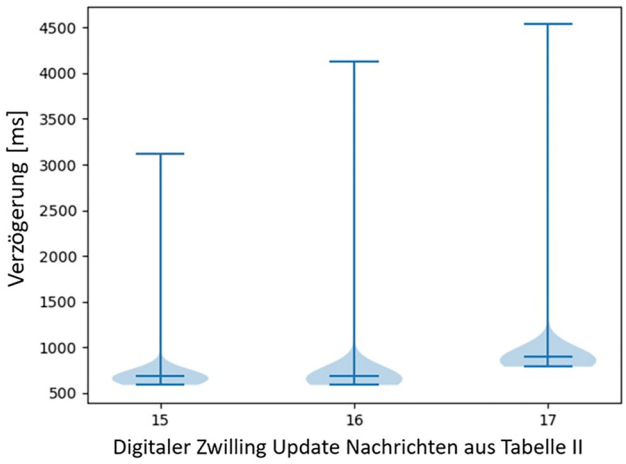

Abbildung 12: Gemessene Kommunikationsverzögerung für Nachrichten im Zusammenhang mit DZ (Partner B). Diese Verzögerungen sind viel höher, da sie im lokalen Netz nicht vorhanden sind und die Antwortzeit der Anwendungsschicht auf die API-Aufrufe an den DZ unbekannt ist.

an den von AAS gehosteten DZ intern wesentlich länger dauern. Die Modellierung der API-Reaktionszeiten des DZ würde den Rahmen dieser Arbeit erheblich sprengen. Die Messungen könnten eine grobe Schätzung der Zeit liefern, die für die Fertigstellung der Nachrichten 15, 16 und 17 erforderlich ist.

6 Diskussion

Gemäß der Erkenntnisse über das zugrundeliegende Netzwerk und den industriellen Aufbau wird davon ausgegangen, dass der Netzwerkverkehr gemäß der Annahme in Abschnitt 4.1 mittels COTS-Hardware abgewickelt wird, jede Komponente des Netzwerkes eine IP-Adresse besitzt und die Netzwerkbedingungen über die Zeit konstant bleiben. Das AS umfasst einen Lageragenten, einen Koordinationsagenten, einen Logistikagenten, einen Montageagenten und einen digitalen Zwillingsagenten. Das System ist nicht durch harte oder weiche Echtzeitgrenzen eingeschränkt, wobei der Schwerpunkt auf der Untersuchung der Echtzeitfähigkeiten des Systems liegt. Mittlere Verzögerungen bei der Zustellung von Nachrichten, die mit jedem der Kommunikationskanäle des Systems verbunden sind, sind hier eingeschlossen. Die Studie untersucht zwei industrielle Anwendungsfälle.

6.1 Echtzeitfähigkeit des Netzwerks

Die Anforderungen an die Kommunikationslatenzzeit im AS sind nur teilweise erfüllt, wie der Vergleich zwischen dem gemessenen Mittelwert der Latenzzeiten (Tabelle 1 und 2) und den geforderten Latenzzeiten (Tabelle 4) zeigt. Bei DZ und Central.AI ist die Echtzeitanforderung weich, während bei den Agenten mit Robotersteuerung die Echtzeitanforderung hart ist, weil bei Nichteinhaltung hohe Kosten durch Verletzungen, Maschinenschäden und -ausfälle entstehen können. Für Nicht-DZ-Agenten wird die Echtzeitfähigkeit des Netzwerks, einschließlich der Einhaltung der Echtzeitgrenze von 1 ms gemäß der QoS Klasse III (Synchronisierte Bewegungsabläufe) [24], mit gemessenen Verzögerungen von höchstens 0,530 ms gewährleistet. Die geforderte Echtzeitgrenze des DZ gemäß QoS Klasse I (Steuerung-zu-Steuerung) wird jedoch nicht eingehalten. Um die Wartezeiten zwischen den Schritten des Montageprozesses zu verkürzen, wird die Optimierung der Latenzzeit bei der Datenübertragung vom DZ in die Cloud empfohlen. Dies betrifft vornehmlich die Nutzung der Applikationsebene des MS Azure Servers für die Kommunikation, da hier die signifikantesten Verzögerungen festgestellt wurden.

6.2 Grenzen des Ansatzes und des Anwendungsfalls

Der Testaufbau ist auf ein einfaches Netzwerk mit einem Ethernet-Switch zwischen zwei lokalen Agenten beschränkt. Es ist noch zu prüfen, ob die Modelle auch für größere lokale Netzwerke Gültigkeit besitzen. Die Netzbedingungen können durch Methoden wie Network Calculus (NC) [25] modelliert werden, um die Netzbedingungen wie Überlastung und Paketverluste zu analysieren. Das Verzögerungsmodell würde sich dadurch deutlich komplexer gestalten, aber es wäre auch in einem realen industriellen Einsatz robuster. Die Dauer der

Tabelle 4: Vergleich der Anforderungen und Messungen der Kommunikationslatenzzeit.

Typ der Nachrichten	Anforderung an Latentzeit (ms)	Gemessene Latenzzeit (Mittelwert in ms, Messgenauigkeit beträgt 0,001 ms.)
Lokale Nachrichten mit Rückantwort	<1 ms	0,530 ms
Lokale Nachrichten ohne Rückantwort	<1 ms	0,286 ms
DZ	<100 ms	805,894 ms

Verzögerungen wird durch die gewählten Hardwarespezifikationen erheblich beeinflusst, z. B. in diesem Fall durch die langen Verarbeitungszeiten des MiR-Roboters und seines entsprechenden Flottenmanagers für Status- und Lageinformationen. Zusätzlich weist die Kommunikation mit dem cloudbasierten DZ eine längere Latenzzeit von 805,894 ms (Tabelle 4) auf. Die Datenverarbeitungszeit im DZ auf der MS Azure Serverwebseite beträgt ca. 200 ms [4]. Um eine umfassende Modellierung und Vorberechnung für DSL4Production vor der Durchführung von Messungen oder der Implementierung des Systems zu ermöglichen, ist es notwendig, über transparente und umfassende Informationen hinsichtlich Hardware-Verzögerungen (z. B. aus industriellen Datenblättern, einschließlich Sensoren und Aktoren) zu verfügen.

7 Fazit und Ausblick

Dieser Beitrag untersucht Annahmen über das Netzwerk, stellt grundlegende Voraussetzungen für einen industriellen Aufbau dar und liefert eine detaillierte Beschreibung von zwei Anwendungsfällen in Form von industriellen Montageszenarien. Die Studie wendet DSL4Production auf zwei industrielle Anwendungsfälle an, um die automatisierte Verarbeitung von Aufträgen, Verzögerungen und die Ressourcenauslastung zu bewerten. Die AS nutzen eine kombinierte zentrale und dezentrale Struktur, um eine effiziente Kommunikation zwischen den Agenten durchzuführen (H1). Durch Messungen wird die Dauer der Verzögerung bestimmt, während mathematische Gleichungen weitere Annahmen über Verzögerungen ermitteln (H3). Die Schwerpunkte liegen auf der Überprüfung der Echtzeitfähigkeiten des Systems sowie der Analyse der mittleren Verzögerungsfaktoren, die mit jedem der Kommunikationskanäle des Systems in Verbindung stehen. Die Systemarchitektur wurde auf der Grundlage des Montageprozesses entwickelt und die Nachrichten anhand von Sender- und Empfängeragenten, Kommunikationstypen, Nachrichtentypen und konkreten Abfrageinhalten klassifiziert (H2). Es wurden Messungen der Netzwerklatenz für jede Nachricht durchgeführt und die mittleren, maximalen und minimalen Latenzzeiten ermittelt. Die Gruppierung der Geigerdiagramme erfolgte anhand der Nachrichten und deren Inhalt für die Analyse. Die modellierten Latenzen wurden mit den gemessenen Latenzen verglichen, wobei sich herausstellte, dass die mittleren Latenzzeiten des Modells in den meisten Szenarien den gemessenen Mittelwerten nahekommen. Durch die Analyse der Daten konnten die geschätzten Latenzzeiten ermittelt werden, die für zukünftige DSL4Production-Modelle verwendet werden könnten (H4).

Weitere Forschungen umfassen die Anwendung von maschinellem Lernen zur Verbesserung und Optimierung der AS in der Robotik und Automatisierung. Die AS-Architektur könnte durch die Einführung von Local.AI, welche die Central.AI unterstützt, optimiert werden, sodass eine effizientere Kommunikation durch die optimale Nachrichtenverteilung an jeden Agenten erreicht wird. Die Verbesserung der AS-Architekturen erfolgt auf der Grundlage der ermittelten Ergebnisse, zum Beispiel durch die Identifizierung von Bereichen mit den höchsten Verzögerungen und die Untersuchung der Agenten, die die meisten Nachrichten versenden und empfangen. Besondere Bedeutung kommt der Reduzierung der Latenz in Echtzeit-Anwendungen wie DZ zu, in denen Echtzeit-Anforderungen erfüllt werden müssen. In den vergangenen Jahren wurde bereits intensiv zur Auswirkung von Latenzen bei teleoperierten Tätigkeiten geforscht. Zukünftige Arbeiten sollten untersuchen, wie stark sich bestimmte Konstellationen von heterogenem Verzögerungsverhalten unterschiedlicher Feedbackmodalitäten auf die Performanz, die Belastung und Beanspruchung der Nutzer:innen auswirken. Auf Basis dieser Erkenntnisse können dann Echtzeit-Anforderungen weiter spezifiziert und priorisiert werden. Eine große Herausforderung ist der DZ des Menschen und dessen Integration in übergeordnete Prozess-DZ. Eine unmittelbare Datenerhebung am Menschen würde eine effektive Einbindung gemäß individueller Fähigkeiten ermöglichen. Hierfür müssen jedoch zunächst datenschutzrechtliche Hürden überwunden werden. Die synchronisierten Nachrichten an die Agenten sollten genauer untersucht werden, wobei besonders die Priorität der Nachrichten berücksichtigt werden muss. Da eine große Menge an Nachrichten an die Agenten gesendet wird, ist es wichtig, die Nachrichten mit höchster Priorität zuerst zu bearbeiten. Die Priorität der Nachrichten sollte als zusätzliches Kriterium für die Klassifizierung im Beitrag zu existierenden Protokollen herangezogen werden. Außerdem sollte die Netzwerksicherheit in der AS beachtet werden, um eine sichere Kommunikation zu gewährleisten, auch wenn dies zu einer erhöhten Verzögerung führen kann. Dieser Beitrag bietet zudem einen Ausgangspunkt für die Untersuchung von Kommunikationsverzögerungen in 6G-Netzwerken.

Die Einführung der 6G-Technologie verspricht eine erhebliche Verbesserung der industriellen Kommunikation, da sie wesentlich höhere Datenübertragungsraten, eine hohe Abdeckung, geringe Latenzzeiten und Zuverlässigkeit bietet. Anwendungen in zukünftigen Fabriken, wie z. B. die individuelle Fertigung von Produkten durch hochvernetze lernender und rekonfigurierbarer Systeme können enorm von 6G profitieren. Die erhöhten Konnektivitätsfähigkeiten

von 6G erleichtern die Echtzeitüberwachung und -steuerung industrieller Prozesse, wodurch die betriebliche Effizienz optimiert und Systemausfallzeiten minimiert werden. Die Integration der fortschrittlichen Funktionen von 6G, einschließlich umfassender Gerätekonnektivität und extrem zuverlässiger Kommunikation mit geringer Latenz, wird eine nahtlose Koordination zwischen intelligenten Fabriken und industriellen Automatisierungssystemen ermöglichen und damit einen Paradigmenwechsel hin zu vernetzten und hocheffizienten industriellen Abläufen einleiten.

Research ethics: Not applicable.
Author contributions: All authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Competing interests: All other authors state no conflict of interest.
Research funding: Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) Lighthouse Initiative KI.FABRIK, (Phase 1: Infrastructure and R&D program, grant no. DIK0249).
Data availability: Not applicable.

Literatur

[1] A. Wannagat, *Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen*, Dissertation, Technische Universität München, hg. von Sierke Verlag, Göttingen, 2010.
[2] A. Sharma, D. Srinivasan, and D. S. Kumar, "A comparative analysis of centralized and decentralized multi-agent architecture for service restoration," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 311—318.
[3] TUM — MIRMI, "KI.Fabrik," [Online]. Verfügbar unter: https://kifabrik.mirmi.tum.de/team/ [Zugriff am: Okt. 28, 2023].
[4] J. Höfgen, *et al*., "Architecture of a versatile digital twin with socket-based communication and azure DT," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1—8.
[5] B. Vogel-Heuser, "Industrie 4.0 and robots — roboter integrated agent network (RIAN)," [Online]. Verfügbar unter: https://www.mec.ed.tum.de/en/ais/research/equipment/automatica/ [Zugriff am: Nov. 03, 2023].
[6] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, p. eabm6074, 2022.
[7] S. Willmann, A. Gnad, E. Hintze, M. Krätzig, and L. Rauchhaupt, *Aspekte der Zuverlässigkeitsbewertung, Fachgruppe 1 „Anwendungen, Anforderungen und Validierung"*, Bremen, BMBF-Förderprogramm „IKT 2020 — Zuverlässige drahtlose Kommunikation in der Industrie" (BZKI), 2017.
[8] G. Luan, "Estimating TCP flow completion time distributions," *J. Commun. Netw.*, vol. 21, no. 1, pp. 61—68, 2019.
[9] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67—82, 1997.
[10] N. Cardwell, S. Savage, and T. E. Anderson, "Modeling TCP latency," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 3, 2000, pp. 1742—1751.
[11] B. Vogel-Heuser, M. Seitz, L. A. C. Salazar, F. Gehlhoff, and A. D. A. Fay, "Multi-agent systems to enable Industry 4.0," *at — Automatisierungstechnik*, vol. 68, no. 6, pp. 445—458, 2020.
[12] I. Kovalenko, D. Tilbury, and K. Barton, "The model-based product agent: a control oriented architecture for intelligent products in multi-agent manufacturing systems," *Control Eng. Pract.*, vol. 86, no. 1, pp. 105—117, 2019.
[13] M. M. Madden, "Challenges using the Linux network stack for real-time communication," in *AIAA Scitech 2019 Forum*, 2019, p. 503.
[14] H. Zimmermann, "OSI reference model-the ISO model of architecture for open systems interconnection," *IEEE Trans. Commun.*, vol. 28, no. 4, pp. 425—432, 1980.
[15] B. Vogel-Heuser, *et al*., "Current challenges in the design of drives for robot-like systems," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 1923—1928.
[16] L. Ribeiro and M. Hochwallner, "Time-related constraints in administration shell design within cyber-physical production systems," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 1564—1569.
[17] D. Hujo, B. Vogel-Heuser, and L. Ribeiro, "Toward a graphical modeling tool for response-time requirements based on soft and hard real-time capabilities in industrial cyber-physical systems," *IEEE J. Emerg. Sel. Top. Ind. Electron.*, vol. 3, no. 1, pp. 13—22, 2022.
[18] M. R. Pedersen, *et al*, "Robot skills for manufacturing: from concept to industrial deployment," *Robot. Comput.-Integr. Manuf.*, vol. 37, no. 1, pp. 282—291, 2016.
[19] C. Diedrich, *et al*, *Information Model for Capabilities, Skills & Services: Definition of Terminology and Proposal for a Technology-independent Information Model for Capabilities and Skills in Flexible Manufacturing*, Berlin, Plattform Industrie 4.0, 2022.
[20] A. van Bemten, *et al*., "Chameleon: predictable latency and high utilization with queue-aware and adaptive source routing," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 451—465.
[21] VDI/VDE, *VDI/VDE 2192 Interoperabilität in Industrie-4.0-Systemen — Qualität von Diensten — Kenngrößen und Einflussgrößen*, Düsseldorf, Engl. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, 2021.
[22] B. Vogel-Heuser, S. Feldmann, T. Werner, and C. Diedrich, "Modeling network architecture and time behavior of Distributed Control Systems in industrial plant automation," in *IECON*, 2011, pp. 2232—2237.
[23] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64—74, 2008.
[24] J. Jasperneite and P. Neumann, "How to guarantee realtime behavior using Ethernet," *IFAC Proc. Vol.*, vol. 37, no. 4, pp. 91—96, 2004.
[25] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, Berlin, Springer, 2001.

Bionotes

Birgit Vogel-Heuser
Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München,
Germany
vogel-heuser@tum.de

Prof. Dr.-Ing. Birgit Vogel-Heuser erwarb ein Diplom in Elektrotechnik und einen Doktortitel in Maschinenbau an der RWTH Aachen. Seit 2009 ist sie Professorin und Ordinaria des Lehrstuhls für Automatisierung und Informationssysteme an der TU München. Ihre aktuellen Forschungsschwerpunkte liegen im Bereich Systems- und Software Engineering. Sie ist Mitglied der acatech, IEEE Fellow, Senior Editor der IEEE T-ASE und für das Thema Digitaler Zwilling in der KI.Fabrik verantwortlich.

Yash Deshpande
Lehrstuhl für Kommunikationsnetze, Technische Universität München, Arcisstr. 21, 80333 München, Germany
yash.deshpande@tum.de

Yash Deshpande erwarb 2016 und 2018 den B.Sc. und M.Sc. in Elektronikwissenschaften an der University of Pune und 2021 den M.Sc. in Kommunikationstechnik an der TU München. Seit 2021 ist er als Research Fellow am Lehrstuhl für Kommunikationsnetze der TU München tätig. Seine Forschungsinteressen umfassen Zufallszugriffsalgorithmen und industrielle drahtlose Kommunikation.

Fandi Hartl
Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München,
Germany
fandi.hartl@tum.de

Fandi Hartl erhielt 2019 einen M.Sc. in Maschinenbau von der TU München und promoviert derzeit am Lehrstuhl für Automatisierung und Informationssysteme. Ihr Hauptforschungsfokus liegt im Management von technischer Schulden in mechatronischen Systemen und in der Weiterentwicklung von Automatisierungskonzepten im Rahmen der Digitalisierung und Vernetzung.

Jingyun Zhao
Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München,
Germany
jingyun.zhao@tum.de

Jingyun Zhao hat 2022 einen M.Sc. in Maschinenbau an der TU München erworben und promoviert derzeit am Lehrstuhl für Automatisierung und Informationssysteme an der TU München. Ihre Forschung konzentriert sich auf Informationsmodellierung, maschinelles Lernen und die Entwicklung von digitalen Zwillingen in automatisierten Produktionssystemen.

Xuezhou Hou
Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München,
Germany
yash.deshpande@tum.de

Xuezhou Hou hat 2021 einen B.Sc. in Maschinenbau an der TU Dortmund erworben. Sie absolviert derzeit ihr Masterstudium im Fach Mechatronik und Robotik an der TU München. Ihre Forschung konzentriert sich auf modellbasiertes Systems Engineering, Spannungsanalyse mit FEM und Prozessoptimierung.

Dominik Hujo
Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München,
Germany
dominik.hujo@tum.de

Dominik Hujo erhielt 2020 seinen M.Sc. in Mechatronik an der TU München. Derzeit promoviert er am Lehrstuhl für Automatisierung und Informationssysteme derselben Universität. Seine Forschungsschwerpunkte liegen auf heterogenen, verteilten Steuerungssystemen sowie modellbasierten Engineering-Methoden. Besonders fokussiert er sich dabei auf die Analyse und Optimierung von Latenzen innerhalb und zwischen verschiedenen Automatisierungsebenen.

Theresa Prinz
Lehrstuhl für Ergonomie, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München, Germany
theresa.prinz@tum.de

Theresa Prinz erhielt 2020 ihren M.Sc. in Maschinenwesen an der TU München mit den Schwerpunkten Logistik, Prozesssimulation und Medizintechnik. Seit 2021 ist sie als wissenschaftliche Mitarbeiterin am Lehrstuhl für Ergonomie an der TU München tätig. Ihre Forschungsschwerpunkte liegen im Bereich der Mensch-Maschine-Interaktionsformen und der Mensch-Maschine-Kooperation im Kontext von Robotik.

Marko Pavlic
Machine Vision and Perception Group, Technische Universität München, Boltzmannstr. 3, 85748 Garching bei München, Germany
marko.pavlic@tum.de

Marko Pavlic erhielt 2019 sein Diplom in Elektrotechnik an der TU Graz, österreich. Anschließend arbeitete er als Systems Engineer und entwickelte Software zur Lokalisierung von Schienenfahrzeugen. Seit 2021 ist er wissenschaftlicher Mitarbeiter in der Forschungsgruppe „Machine Vision and Perception“ unter der Leitung von Prof. Dr.-Ing. Darius Burschka. Seine Forschungsinteressen umfassen Visuelle und taktile Perception für die Robotik und Lernen aus Demonstrationen.

Darius Burschka
Machine Vision and Perception Group, Technische Universität München, Boltzmannstr. 3, 85748 Garching bei München, Germany
burschka@tum.de

Prof. Dr.-Ing. Darius Burschka erhielt 1998 seinen Doktortitel in Elektrotechnik und Informationstechnik an der TU München. Seit 2005 ist er Professor für Informatik an der TU München und leitet die Forschungsgruppe „Machine Vision and Perception“. Er war Bereichskoordinator im DFG-Exzellenzcluster „Kognition in technischen Systemen“ und ist derzeit Co-Vorsitzender des IEEE RAS Technical Committee on Computer and Robot Vision sowie Mitglied des Wissenschaftsrates des MIRMI. Sein Forschungsschwerpunkt liegt auf der bildbasierten Navigation und der dreidimensionalen Rekonstruktion aus Sensordaten.

Klaus Bengler
Lehrstuhl für Ergonomie, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München, Germany
bengler@tum.de

Prof. Dr. phil. Klaus Bengler erhielt 1991 sein Diplom und 1995 seinen Doktortitel an der Universität Regensburg, in Zusammenarbeit mit BMW, beide in Psychologie. Seit Mai 2009 ist er Leiter des Lehrstuhls für Ergonomie an der TU München.

Wolfgang Kellerer
Lehrstuhl für Kommunikationsnetze, Technische Universität München, Arcisstr. 21, 80333 München, Germany
wolfgang.kellerer@tum.de

Prof. Dr.-Ing Wolfgang Kellerer erwarb 2002 den Doktortitel in Elektrotechnik an der TU München. Im Jahr 2001 war er Gastforscher im Information Systems Laboratory, Stanford University. Vor seinem Start an der TU München, war er mehr als zehn Jahre in den europäischen Forschungslabors von NTT DOCOMO tätig. Derzeit ist er ordentlicher Professor an der TU München und leitet den Lehrstuhl für Kommunikationsnetze. Im Jahr 2015 wurde er mit dem ERC Consolidator Grant der Europäischen Kommission für seine Forschung zur Flexibilität in Kommunikationsnetzen ausgezeichnet.

André Kraft
BMW AG, Lilienthalallee 25, 80788 München, Germany
andre.kraft@bmw.de

André Kraft absolvierte 2013 sein Diplom im Maschinenbau an der TU Dresden und ist gegenwärtig in der Forschung und Entwicklung intelligenter Robotik-Plattformen bei der BMW AG tätig. Sein Forschungsschwerpunkt umfasst die innovative Integration von KI in die Fabrikumgebung durch Roboter und ähnliche Systeme, mit einem besonderen Fokus auf kundenspezifische Anpassungsfähigkeit in flexiblen Produktionsprozessen.

Bernd Vojanec
WITTENSTEIN SE, Zentrale Forschung & Entwicklung, Igersheim, Germany
bernd.vojanec@wittenstein.de

Bernd Vojanec absolvierte 2018 seinen Master in Management und IT an der Donau-Universität Krems, Niederösterreich. Aktuell ist er in der WITTENSTEIN Gruppe tätig, wo er sich auf die Entwicklung und Optimierung von I4.0-Systemen spezialisiert. Seine Forschung konzentriert sich hauptsächlich auf den Einsatz von industriellen digitalen Zwillingen und datengetriebenen Diensten im Kontext von Industrie 4.0, insbesondere in Bezug auf intelligente Produkte & Services, Standardisierung und der Verwaltungsschale.

Timo Markert
WITTENSTEIN SE, Zentrale Forschung & Entwicklung, Igersheim, Germany
timo.markert@resense.io

Timo Markert erwarb einen M.Sc. in Wirtschaftsinformatik an der Hochschule Aalen und promoviert derzeit an der Universität Osnabrück im Bereich der Robotik und des maschinellen Lernens. Seine Forschung konzentriert sich auf die Analyse taktiler Sensordaten in robotischen Manipulationsaufgaben. Gleichzeitig ist er Geschäftsführer der Resense GmbH, einem Joint Venture der Firmen WIKA und WITTENSTEIN, das innovative Sensorlösungen entwickelt.