Schlussbericht

ZE: Fraunhofer Gesellschaft e.V. Ausführende Stelle: Fraunhofer IIS/EAS	Förderkennzeichen: 01IS18073A

Vorhabenbezeichnung:
Verbundprojekt: Verteilte Emergente Regelung im IoT mittels Agil kollaborierender Systeme

Laufzeit des Vorhabens: 01.06.2019 bis 30.11.2022

Kontakt: Dr. Andreas Frotzscher, Email: andreas.frotzscher@eas.iis.fraunhofer.de

1 Einleitung

1.1 Aufgabenstellung

Das Fraunhofer IIS/EAS übernahm in VERITAS die Projektkoordination und brachte im Rahmen des Arbeitsplanes wesentliche Kompetenzen aus den Bereichen des Entwurfes, der Überwachung und dem Management industrieller drahtloser Übertragungssysteme sowie dem Entwurf robuster, verteilter Regelungssysteme ein.

Wichtige Arbeitsinhalte des Fraunhofer IIS/EAS waren die Bewertung der vorgestellten Anwendungsszenarien, die Erstellung der Anforderungsanalyse, die Konzeption und Entwicklung eines Architekturmodells und Definition der Schnittstellen sowie der Entwurf und die Implementierung zugeschnittener Metriken zur Bewertung des Laufzeitverhaltens und der Performance jeweils für Co-Design und -Management.

1.2 Stand der Technik

1.) Architektur verteilter Steuerungs- und Regelungssysteme: Zur Anbindung von Ein- und Ausgabebaugruppen, anderen Sensoren oder zur Verbindung mit anderen Steuerungsgeräten werden i.d.R. Feldbussysteme (z.B. PROFINET, EtherCAT) eingesetzt. Verteilte Steuerungsansätze sind in der Industrie noch nicht weit verbreitet. Allerdings existieren in internationalen Normen und der wissenschaftlichen Literatur zahlreiche Vorschläge, welche bereits praktischen Einsatz finden. In IEC 61499 und IEC 61804 werden Steuerungsansätze auf Basis eines verteilten Funktionsblockkonzeptes definiert. Die Kommunikation zwischen den verteilten Komponenten findet ereignisbasiert bzw. zyklisch statt. In den letzten Jahren setzt eine Bewegung zur Nutzung serviceorientierter Architekturen (SOA) in den Ebenen 2-4 nach dem Standard Paradigma ISA'95 ein [1]. Beispiele sind die zunehmende Unterstützung neuer Standards wie OPC-UA, MQTT und AMQP zum Betrieb verteilter Anwendungen. Auch die Referenzarchitektur für Industrie 4.0-konforme Systeme (RAMI 4.0) basiert auf SOA [2].

Die zuvor genannten Technologien erfordern die Konfiguration der Vernetzung und der Geräteparameter zur Entwicklungszeit bzw. Inbetriebnahme. In der Entwicklung komplexer Software sind Entwicklungsmuster, z.B. Strategie, Service Locator, Dependency Injection und Inversion of Control, zum Management der Abhängigkeiten zwischen Komponenten und zur Konfiguration Stand der Technik [3]. Diese strukturieren Software-Komponenten und deren Abhängigkeiten tragen zur Verbesserung der Wartbarkeit und Wiederverwendbarkeit bei. In Kombination erlauben diese Entwicklungsmuster die Software-Komponenten einer Applikation automatisch zu orchestrieren sowie zur Laufzeit zu reorchestrieren. Die für selbstorganisierende Systeme notwendige automatische Konfiguration verteilter industrieller Systeme ist aktuell noch Gegenstand der Forschung. Hierbei wird die standardisierte Beschreibung des Material- und Informationsaustauschs als eine notwendige Grundlage untersucht [4]. Aus einer derartigen Prozessbeschreibung kann die dynamische Konfiguration des Informationsaustauschs bei Änderungen einer Produktionsanlage oder des Prozesses erfolgen [5]. Das im Rahmen der Industrie 4.0 eingeführte Konzept der Verwaltungsschale stellt einen Schritt in Richtung der standardisierten Beschreibung aller für die Produktion genutzten Komponenten dar, mit dem Ziel, deren Vernetzung zu ermöglichen. Mit openAAS [6] gibt es bereits offene Softwarelösungen für Forschungszwecke.

2.) Vernetzung verteilter Steuerungs- und Regelungssysteme: Für die Steuerung oder Regelung verteilter Systeme, bestehend aus mehreren Subsystemen, werden bisher hierarchische Architekturansätze verwendet. Die Subsysteme enthalten lokale Regler und Steuerungen, die lokale Eingabegrößen und Sensorwerte für die Berechnung von Ausgangsgrößen nutzen. Übergeordnete Regler oder Steuerungen koordinieren die Interaktion der Subsysteme im Gesamtsystem [7]. Für verteilte, emergente Systeme ist ein hierarchischer Architekturansatz sowohl für die Struktur als auch für die Regelung selbst nicht möglich. Die Herausforderung für solche Systeme besteht darin, selbstständig gemeinsame Vernetzungs- und Regelungsvorschriften zu identifizieren, um die globalen Regelziele zu erreichen.

Darüber hinaus kann das Vernetzungssystem in verteilten, vernetzten Regelungssystemen die Qualität der Regelung (Quality of Control, QoC) durch Verzögerungen und Telegrammverluste signifikant beeinflussen [8]. Die Berücksichtigung der Vernetzung in verteilten Regelungssystemen ist ein grundlegendes, ungelöstes Thema [9]. Erste Arbeiten zur Regelung über Verbindungen mit begrenzter Datenrate bieten bedeutende Erkenntnisse. Jedoch ist sehr wenig darüber erforscht, wie sich die existierende Theorie modifizieren lässt, um die Eigenschaften der Vernetzung in der Regelung zu berücksichtigen [10] [11] [12]. Die in Vernetzungssystemen eingesetzten Fehlerkorrekturverfahren sind nicht unmittelbar im Zusammenhang mit einer Regelung anwendbar, da ihre weichen Zuverlässigkeits- und Resilienzbeschränkungen nicht ausreichen, um selbst eine schwache Momentstabilität für sicherheitskritische Anwendungen sicherzustellen [13].

Die traditionelle Regelungstheorie konzentriert sich auf Systeme, die durch ideale Kanäle verbunden sind, die äquidistante Abtastung, konstante Verzögerungen und keinen Datenverlust annehmen, während die Informationstheorie die Übertragung von Daten über nicht-ideale Kanäle mit variablen Verzögerungen und möglichem Datenverlust untersucht [15]. Die Ausführung von Regelungs- und Kommunikationsaufgaben wird bisher als periodisch mit festen Zeiträumen und bekannter Worst-Case-Ausführungszeit modelliert [16]. Diese Annahmen sind für verteilte, emergente Systeme in dynamischen Umgebungen nicht immer zutreffend.

2 Ablauf des Vorhabens

2.1 Architekturmodell emergent kollaborierender Systeme

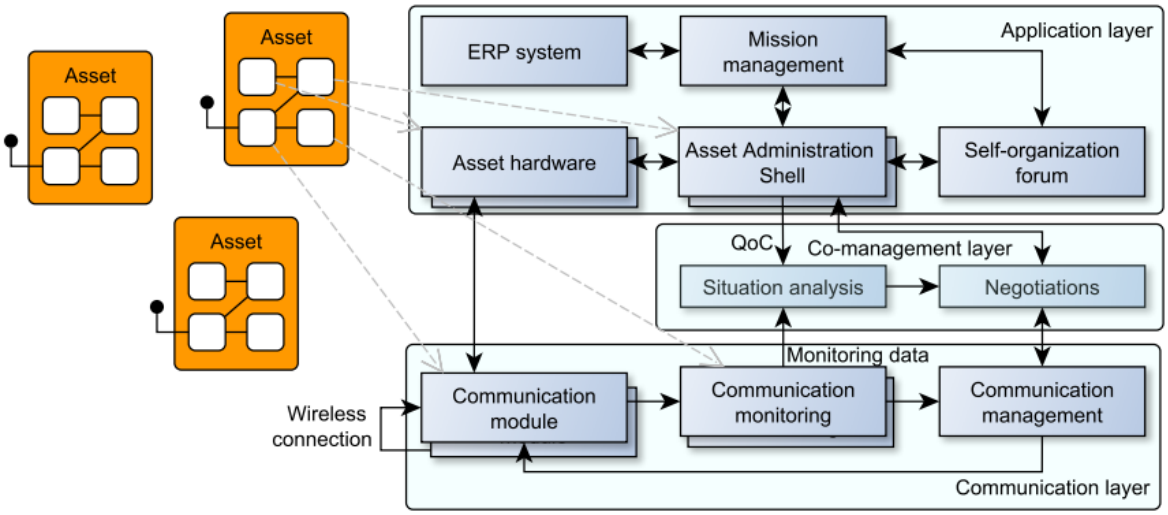


Abbildung 1 – Entwickeltes Architekturmodell

Zur Koordination der verschiedenen Subsysteme wurde eine universelle Systemarchitektur entwickelt, die dezentral auf jeder Komponente lauffähig ist und eine gegenseitige Abstimmung hinsichtlich der auszuführenden Teilaufgaben ermöglicht.

Die logischen und physischen Einheiten der verteilten Architektur sind in drei Schichten gruppiert, deren Aufgaben nachfolgend in aller Kürze skizziert werden.

1. Die Kommunikationsschicht im unteren Teil umfasst die in den IoT-Geräten jeweils installierten Kommunikationsmodule und die Verwaltung des eingesetzten Kommunikationssystems. Die Kommunikationsschicht stellt somit eine Software-Abstraktion des Vernetzungssystems dar. Die in der Realität vorhandene nicht-ideale Datenübertragung zwischen den Subsystemen wird hier im Systemmodell explizit mitbetrachtet, sodass insgesamt ein fehlertolerantes Systemverhalten auch bei gestörter Kommunikation zwischen den Subsystemen erzielt wird.

2. Die Anwendungsschicht konzentriert sich auf die Ausführung der angeforderten Aufgabe, die vom übergeordneten Produktionssystem definiert wird. Sie kümmert sich um die Zerlegung der Anforderung in unabhängig ausführbare Teilaufgaben, organisiert die Zuweisung der Teilaufgaben an einzelne Subsysteme und überwacht deren Abarbeitung. Sie enthält hierfür zentrale Funktionen der Selbstorganisation und die gesamte aufgabenspezifische Funktionalität der verteilten Anlagen.

3. Die Co-Management-Schicht ermöglicht die Selbstorganisation des Gesamtsystems, indem ein automatisierter Interessensausgleich zwischen Anwendungs- und Kommunikationsschicht stattfindet und die Optimierung der Zuweisung von Teilaufgaben an einzelne Komponenten berechnet wird. Sie verleiht dem System die notwendige Adaptivität, indem auf veränderte Situationen im Kommunikationssystem und in der

Anwendung ggf. durch Anpassung der anderen Seite reagiert wird. Im Falle von Unstimmigkeiten wird ein Verhandlungs-Block ausgelöst, um einen tragfähigen Kompromiss zwischen den Anforderungen der Automatisierungsanwendung und den Fähigkeiten des Kommunikationsnetzes zu finden.

Mithin steht ein Architekturmodell zur Verfügung, dass die verschiedenen Anforderungen der beteiligten Schichten harmonisiert, die beteiligten Komponenten zur Laufzeit koordiniert und damit einen störungsfreien dezentralen Betrieb des Automatisierungssystems sicherstellt.

2.2 Emulation des Vernetzungssystems

Ein wesentlicher Fokus des Vorhabens ist die Nutzung von miteinander vernetzten Robotern, welche drahtlos miteinander kommunizieren. Um vergleichbare Tests der entwickelten Regelungsalgorithmen durchführen zu können, muss sichergestellt werden, dass möglichst wenige Umweltfaktoren die Experimente beeinflussen. Im Falle von funkvernetzten Robotern ist es schwer möglich, die Funkverbindung zwischen den Robotern frei von Umweltfaktoren zu halten.

Um dennoch vergleichbare Experimente durchführen zu können, musste ein Netzwerk-Emulator entwickelt und eingesetzt werden. Die Roboter wurden kabelgebunden untereinander vernetzt, wodurch Umwelteinflüsse, welche die Datenübertragung beeinflussen nahezu ausgeschlossen werden können. Der Netzwerk-Emulator wird entsprechend konfiguriert, sodass ein möglichst realistisches Abbild von Wireless-Netzwerk-Traffic zwischen den Robotern entsteht.

Im Laufe des Projektes wurden verschiedene Varianten entwickelt, welche an unterschiedlichen Stellen die Netzwerkemulation durchführen und somit bei verschiedenen Einsatz-Szenarien genutzt werden können.

Variante 0: State of the Art: ROS-NetSim

Während der Projektlaufzeit wurde die Software ROS-NetSim (https://github.com/alelab-upenn/ros-net-sim) veröffentlicht. Die Software wurde im Hinblick auf ihre Eignung für das Projekt umfassend geprüft. Die Analyse hat ergeben, dass die Software nicht optimal den Anforderungen und Zielen des vorliegenden Szenarios entspricht. Die Fähigkeiten der Software erfüllen nicht die spezifischen Anforderungen und Ansprüche des gegenwärtigen Anwendungsfalls und machen sie somit ungeeignet für die anstehende Aufgabe.

Zudem gab es Versionskonflikte mit der im Projekt Veritas entwickelten Software, was entweder eine komplette Änderung des Veritas-Software-Stacks oder von ROS-NetSim erforderlich gemacht hätte.

Aus den genannten Gründen und aufgrund der Veröffentlichung der Software erst zur Mitte der Projektlaufzeit konnte diese nicht eingesetzt werden.

Variante 1: ROS Topics basierte Netzwerkemulation

Diese Variante basiert auf der direkten Netzwerkemulation in der ROS-Laufzeitumgebung. ROS unterstützt verschiedene Arten der Kommunikation zwischen den Teilnehmern.

- **Topics** werden verwendet, um kontinuierlich Daten auszutauschen (Sensor-Messwerte, Position, etc.). Daten können zu jeder Zeit via publish / subcribe zwischen mehreren Teilnehmern ausgetauscht werden.

- **Services** können genutzt werden, wenn kurze Aufgaben erledigt werden sollen. Jeder Teilnehmer kann Services anbieten, welche von anderen Teilnehmern genutzt werden können. Es wird gewartet, bis das Ergebnis (Aufgabe erledigt / nicht erledigt) zurückgeliefert wird.
- **Actions** werden für Aufgaben genutzt, welche eine längere Zeit zur Ausführung benötigen oder für welche der Status relevant ist, da dieser gespeichert wird. Actions können aufeinander aufbauen.

Die Netzwerkemulation auf ROS Topics wurde im Projekt Veritas wie folgt umgesetzt:

1. Identifikation der Topics, welche zwischen den Robotern verschickt werden sollen
2. Erstellen eines speziellen Launch-Files:
    a. Es gibt keine direkte Roboter-Roboter Kommunikation mehr. Diese wird umgeschrieben
    b. Für jede Roboter-Roboter Verbindung und jede Richtung wird ein extra Topic mit dem Schema „NW_TOPIC_A_B_IN“ und „NW_TOPIC_A_B_OUT“ eingeführt, welches als Netzwerkemulator für diese einzelne Verbindung dient.
3. Der Netzwerkemulator subscribed auf alle Topics mit dem Schema „NW_TOPIC_A_B_IN“.
4. Wenn der Netztwerkemulator eine Nachricht bekommt, dann entscheidet er abhängig von der Konfiguration, ob die Nachricht weiter gesendet wird (dann als „NW_TOPIC_A_B_OUT“) oder nicht. Wird die Nachricht weiter gesendet, wird abhängig von der Konfiguration noch ein Latenz der Verbindung hinzugefügt, also gewartet, bis die Nachricht weiter geleitet wird.
5. Der Roboter, welcher auf eine Nachricht subscribed, tut dies nicht mehr auf das ursprüngliche Topic, sondern auf das neu eingeführte Topic „NW_TOPIC_A_B_OUT“.

Der Vorteil dieser Methode ist es, dass sehr individuell und feingranular die Netzwerkemulation konfiguriert und umgesetzt werden kann. Außerdem werden keine tiefgreifende Netzwerkkonfigurations-Kenntnisse benötigt und der Einsatz ist sowohl in der Simulation als auch in der Realität möglich.

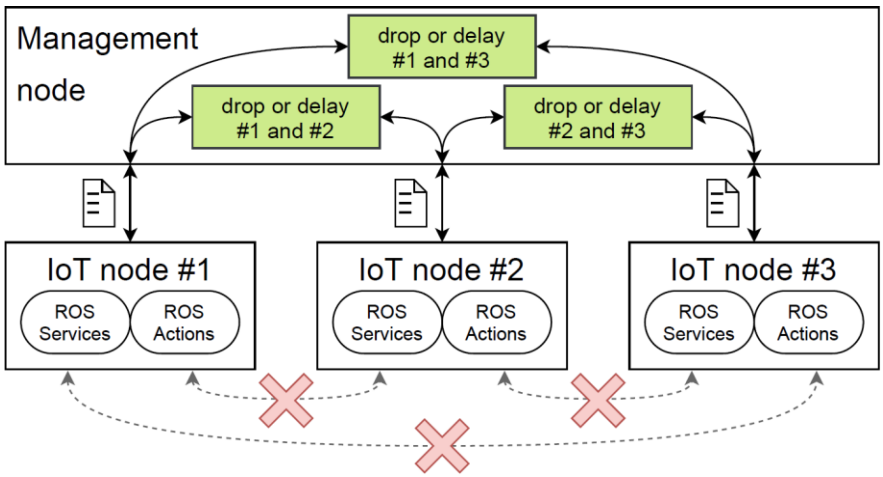
Der Nachteil der Topic basierten Netzwerkemulation liegt darin, dass es nur für ROS Topics genutzt werden kann – nicht jedoch für ROS Services oder Actions. Im Laufe des Projektes sind jedoch verschiedene Funktionalitäten als Services und Actions umgesetzt worden, da dies dem Industrie-Standard entspricht. Als weiterer Nachteil ist die Komplexität zu betrachten. Die Anzahl der Topics, welche für die Netzwerkemulation notwendig sind, steigt quadratisch mit der Anzahl der Roboter. Dies erschwert eine Skalierung für den Einsatz von weiteren Robotern.

Variante 2: Proxy-basierte Netzwerkemulation

Die Software emuliert Unregelmäßigkeiten und Fehler der Kommunikation auf Basis von IP-Paketen (Layer 3). Der Emulator wird auf einem Linux-Host ausgeführt, welcher als Router zwischen zwei oder mehreren Teilnehmern (in unserem Fall Robotern) arbeitet. In Abbildung 2 ist schematisch dargestellt, wie die Kommunikationsemulation zwischen 3 Robotern durchgeführt wird, sobald die Netzwerkemulation aktiv ist. Es findet keine direkte Kommunikation zwischen

den Robotern statt (mit rotem Kreuz markiert), sondern die Pakete werden über den Management-Knoten geleitet, auf dem der Netzwerk-Emulator läuft.

Abbildung 3 - Übersicht zur Kommunikationsemulation



Die Software besteht aus zwei Komponenten:

1. **Routing**: Mittels Routing werden alle ankommenden TCP/IP-Pakete als erstes zum Emulator umgeleitet (also nicht direkt zum Empfänger)

2. **Emulator**: Der Emulator ist dafür verantwortlich, dein ankommenden Traffic zu verändern

Mit den Linux-Paketen `socat` und `iproute2` wird das Routing realisiert. Damit wird eine virtuelle Netzwerkschnittstelle erzeugt. Es werden alle ankommenden IP-Pakete als UDP-Paket verpackt und an die virtuelle Netzwerkschnittstelle (TUN Device) gesendet. Alle Pakete, welche aus dem virtuellen Netzwerk wieder herauskommen, werden zurück auf die realen Netzwerkschnittstellen geschrieben, wodurch sie ihr eigentliches Ziel erreichen können. Durch diese Methode können zwei oder mehr Geräte miteinander verbunden werden, welche alle den gleichen Emulator verwenden.

Der Emulator wurde in `Python` entwickelt. Es werden hierbei nur Python Standard-Funktionen verwendet, um den Traffic zu verändern. Der Emulator wartet auf eingehende UDP Pakete. Anschließend werden diese in Abhängigkeit der Konfiguration entweder verzögert oder komplett verworfen.

Das Skript verarbeitet alle ankommenden Pakete seriell, was dem First In – First Out Prinzip entspricht. Wenn der Emulator wartet, bevor das aktuelle Paket weiter geschickt werden kann, werden in der Zwischenzeit keine weiteren Pakete bearbeitet. Dadurch kann die Anzahl der in der Warteschlange befindlichen Pakete mit der Zeit stark ansteigen. Werden Pakete, welche original als TCP-Paket verschickt wurden, vom Netzwerk-Emulator verworfen oder wird zu lange gewartet, dann werden die verbunden Netzwerkgeräte die Pakete wiederholt senden, bis diese angekommen sind.

Der Netzwerkemulator fungiert insgesamt wie eine Art Proxy für alle einkommenden Netzwerkpakete und kann unabhängig von der Systemsimulation betrieben werden. Die Systemsimulation kann also mit und ohne Netzwerk-Emulator gestartet werden. Wird der Netzwerk-Emulator genutzt, dann kann dieser Informationen mit der Systemsimulation austauschen und im laufenden Betrieb angepasst werden.

Der Vorteil dieser Variante ist es, dass der Netzwerkemulator von ROS unabhängig ist und somit jegliche Netzwerkkommunikation mit dem System emuliert werden kann. Damit ist es also nicht mehr notwendig Anpassungen in der Veritas-Software vorzunehmen.

Als Nachteil hat sich herausgestellt, dass diese Art der Netzwerkemulation in der Simulation sehr gut funktioniert (jeder IOT-Node und der Management-Node werden als eigene Virtuelle Maschine aufgesetzt) mit realer Hardware jedoch nur schwer umzusetzen ist. Dies hat Hardware-Gründe (Leistungsfähiger PC mit vielen Netzwerkschnittstellen wird benötigt) und Software-Gründe (Die Software für die Roboter-Steuerung benötigt ein Echtzeitbetriebssystem, wodurch es zu Konflikten mit der Netzwerkemulator-Software kommt).

2.3 Co-Design und Co-Management

Die Möglichkeiten zur Datenübertragung in einer Funkverbindung mittels Luftschnittstelle werden durch die zugewiesene Zeit und die zugewiesene Frequenzbandbreite des Übertragungskanals begrenzt. Ein kleiner Teil der Ressourcen wird für die Selbsterhaltung und die interne Verwaltung des Kommunikationsnetzwerks aufgewendet (Netzwerk-Management), der restliche Teil steht für die tatsächliche Datenübertragung zur Verfügung. Modelliert man diesen Zusammenhang auf Ebene der Kommunikationsschicht, kann der Kommunikationsbedarf der Anwendung mit den verfügbaren Kommunikationsressourcen abgeglichen und deren Allokation durch ein intelligentes Management nachjustiert werden.

Im Rahmen der Arbeiten wurde der Fokus auf die Entwicklung eines Netzwerkmodells gelegt, sodass einerseits Anforderungen der Anwendungsschicht zur Ausführung einer Teilaufgabe hinsichtlich Machbarkeit und Ausfallsicherheit bewertet werden können. Andererseits wird dadurch eine Überwachung des Netzwerkzustands ermöglicht, sodass Ausfälle und ein Stillstand der Automatisierungsanlage infolge einer Überlastung des Funknetzwerks durch intelligente Adaption der Kommunikationsressourcen vermieden werden kann. Hierzu wurde ein theoretisches Modell zur dynamischen Berechnung der Netzwerkauslastung entwickelt und in der Kommunikationsschicht des Architekturmodells implementiert.

Die hierbei zugrunde gelegten Annahmen werden im Folgenden kurz skizziert:

1. Das drahtlose Kommunikationssystem wird von einem zentralen Management-Knoten gesteuert. Die drahtlose Kommunikation wird auf IP-Ebene modelliert, d.h. neben der Punkt-zu-Punkt-Kommunikation zwischen dem NW-Manager und bestimmten Knoten sind auch direkte Verbindungen zwischen Knoten für den Datenverkehr möglich. Damit wird das Kommunikationsverhalten der MAC-Schicht auf die IP-Schicht abstrahiert.

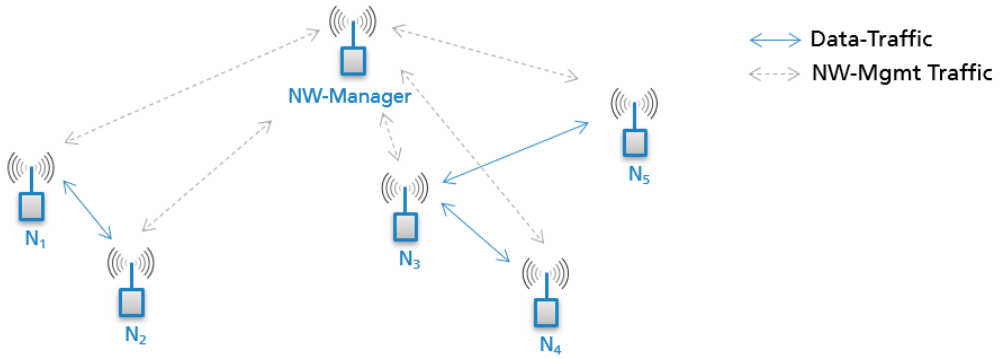


Abbildung 4 - Topologie des Kommunikationsnetzwerks

2. Der Netzwerkmanager ist mit der Co-Management-Schicht verbunden, um den aktuellen Netzwerkzustand auf Link-Ebene weiterzuleiten und die verbleibenden Netzwerkreserven zu analysieren. Die Co-Management-Schicht empfängt Kommunikationsanfragen von der Task-Schicht und wertet aus, ob diese im Hinblick auf den aktuellen Netzwerkzustand bestätigt werden können oder nicht.

3. Der Netzwerkmanager führt die zentrale Ressourcenplanung in einem Zeit-Frequenz-Raster durch, wie in Abbildung 5 beispielhaft dargestellt.

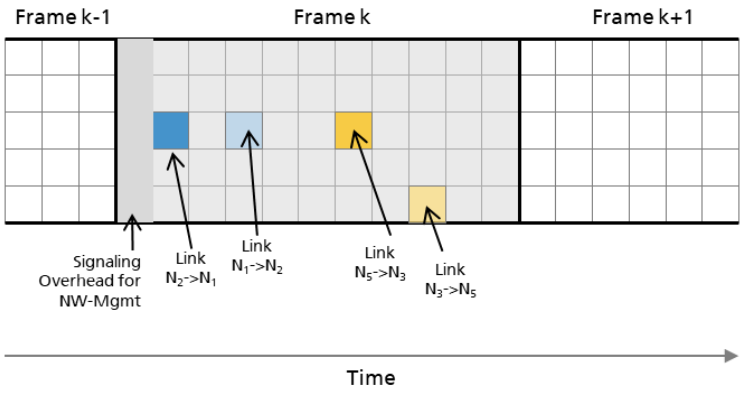


Abbildung 6 - Prinzip der Ressourcenallokation in einem Zeit-Frequenz-Raster

4. Der Netzwerkmanager wird durch eine verteilte Link-Analyse unterstützt, d. h. alle Knoten verfolgen den Paketversand und -empfang im Hinblick auf Kommunikationsstörungen und führen periodisch eine statistische Analyse dieser Parameter durch.

Zur Umsetzung des entwickelten Konzepts wurde eine eigenständige Softwarekomponente in ROS mittels `Python` entwickelt, die ausgehend von den physikalischen Netzwerkparametern und den Ausbreitungsbedingungen im Raum das Signal-Rausch-Verhältnis der einzelnen Teilverbindungen berechnet und daraus abgeleitet jeweils die spektrale Effizienz bestimmt. Durch Multiplikation der spektralen Effizienz mit der Kanalbandbreite ergibt sich die theoretisch maximal erzielbare Datenrate pro Teilverbindung (bei einer Ressourcenallokation von 100%). So kann durch eine anteilige Zuweisung von Ressourcen (0 … 100%) zu einzelnen Links die tatsächlich für die Anwendung benötigte Datenrate eingestellt werden. Treten auf einzelnen Teilverbindungen Störungen des Übertragungskanals auf, kann durch dies durch eine temporäre zusätzliche Allokation von Netzwerkressourcen ggf. ausgeglichen werden.

In die Berechnung der Übertragungsbedingungen gehen u.a. folgende Parameter ein:

- Geometrische Position der IoT-Devices und der Basisstation (NW-Manager) im Raum
- Bestehende Teilverbindungen (Links) zwischen den IoT-Devices
- Maximal verfügbare Sendeleistung pro Sender
- Verfügbare Kanalbandbreite pro Kanal
- Rauschleistung des Empfängers
- Parameter zur Berechnung des Pfadverlustes und des Signal-Rausch-Verhältnisses pro bestehender Teilverbindung
- Verwendeter Funkstandard (WLAN/5G)

Daraus werden sender- und empfängerseitig für jede der Teilverbindungen folgende Qualitätsmaße als Metrik für die jeweilige Übertragungsqualität bestimmt:

- Auf Senderseite:
  - Auslastung der Netzwerkressourcen pro Link
  - Paketwiederholrate
  - Durchschnittliche Netto-Datendurchsatz (hinsichtlich der tatsächlich transferierten Nutzdaten, abzüglich der fehlerhaften Pakete)

- Auf Empfängerseite:

- Mittlere Paketfehlerrate (Anzahl fehlerhafter Datenpakete pro Zeiteinheit)

In einem weiteren Schritt wird durch Aufsummieren bzw. Mittelwertbildung der aufgeführten Metriken (je nach Art der Metrik) über alle Teilverbindungen der Gesamtzustand des Netzwerks berechnet. Dies ermöglicht ein dynamisches Task Assessment, d.h. die Bewertung einer Kommunikationsanfrage für eine zusätzliche Teilaufgabe des Automationssystems zur Laufzeit des Systems.

Dazu werden folgende Parameter von der Task-Schicht benötigt:

- Statistische Parameter des Datendurchsatzes auf den betroffenen Teilverbindungen
- Statistische Parameter der zu übertragenden Nutzdatenmenge pro Zeiteinheit

Auf dieser Basis wird ein oberer Grenzwert für den benötigten Datendurchsatz bestimmt, der mithilfe des Netzwerk-Ressourcenmodells in die Menge der hierfür zusätzlich benötigten NW-Ressourcen umgerechnet werden kann (0… 100%). Beträgt die Summe der aktuell und zusätzlich, auf Basis der neuen Anforderung benötigten Ressourcen mehr als 100%, wird die Anfrage abgelehnt, da der Kommunikationsbedarf die Ressourcen des Netzwerks übersteigt. Andernfalls wird die Anfrage bestätigt und die notwendigen Ressourcen werden auf den angefragten Teilverbindungen reserviert.

Bei der Bestimmung der belegten Netzwerkressourcen wird neben den einzelnen Teilverbindungen auch ein Anteil für NW-Management-Kommunikation und ein zusätzlicher Sicherheitspuffer berücksichtigt, sodass eine vollständige Auslastung (100%) vermieden wird. Der entwickelte ROS-Node auf der Kommunikationsschicht und die Schnittstellen (ROS-Messages, ROS-Actions) zu den anderen Schichten sind in Abbildung 7 dargestellt.

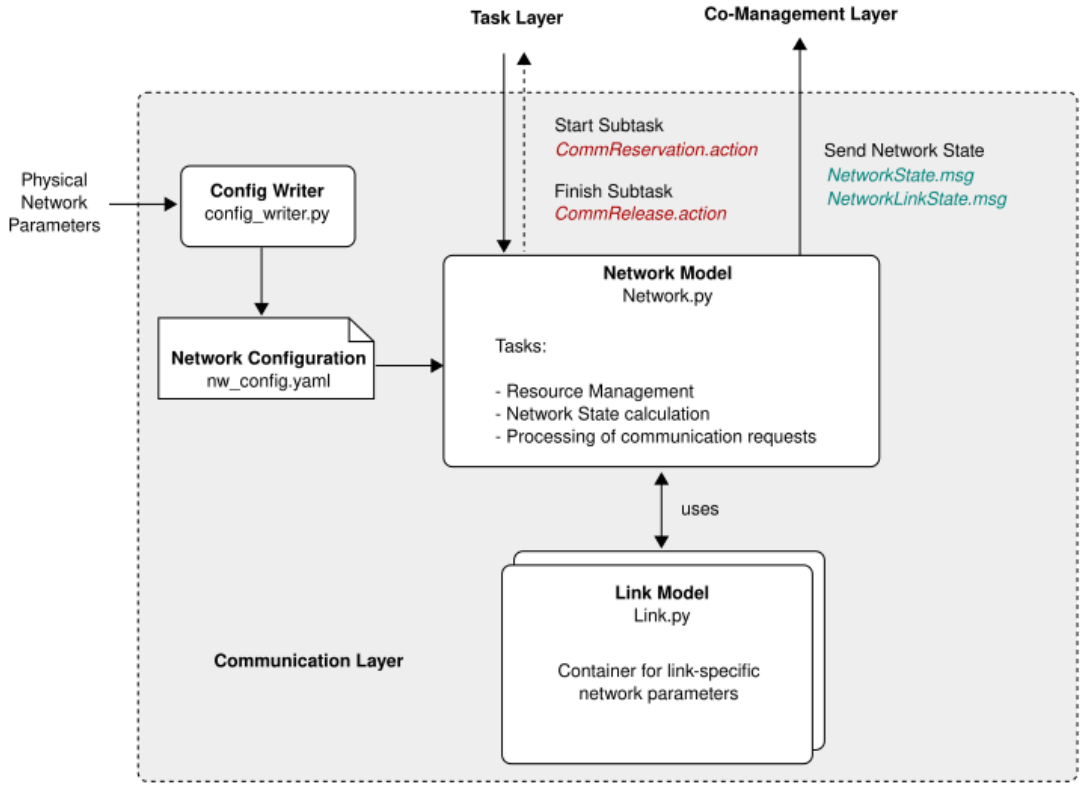


Abbildung 8 - ROS Node Communication Management und seine Schnittstellen

2.4 Demonstrator und Testumgebung für Kollaboration von Robotern

Analog zum Demonstrator bei der TUM/TQ hat das Fraunhofer IIS/EAS den Aufbau in leicht abgewandelter Form repliziert, um die Projektergebnisse zukünftig aufwandsarm nachnutzen, verwerten sowie öffentlichkeitswirksam demonstrieren zu können. Hierzu wurde eine flexible, modulare Demonstrator Plattform entwickelt, die in Abbildung 9 veranschaulicht ist.

Abbildung 10: Veritas Demonstrator am Fraunhofer IIS/EAS

Diese dient dem Fraunhofer IIS/EAS auch über das Projekt hinaus zum Kompetenzaufbau in der stationären und kollaborativen Robotik. Im Unterschied zum Projektdemonstrator der TUM wurde hier bewusst auf ein flexibles und mobiles Hardware-Konzept gesetzt, um in Zukunft verschiedenste Kollaborationsszenarien mit geringem Änderungsaufwand umsetzen zu können. So sind beispielsweise Mensch-Roboter-Interaktion, eine vernetzte Echtzeit-Kollaboration über Funksysteme und auch die Kombination der Franka Emika Roboter mobilen Robotern denkbar.

Als Basis für die drei Franka Emika Roboter wurde jeweils ein mobiler Robotertisch mit Heberollen und einem Unterschrank für Schaltschrank und zusätzliches Equipment verwendet.

Für den Zweck des Veritas-Demonstrators wurden die Plattformen temporär fest mittels einer Montageplatte in der Mitte verbunden, um eine stabile Positionierung der Roboter zu erhalten. Dies ist jedoch je nach zukünftigem Anwendungszweck frei änderbar.

Als ursprüngliches Anwendungsszenario war geplant, dass eine Mauer mit einem vorgegebenen Muster von den 3 Robotern selbständig aufgebaut wird (siehe Abb. 11). Dieses wurde zur Vereinfachung im Rahmen auf einen einzelnen, auswählbaren Buchstaben reduziert.

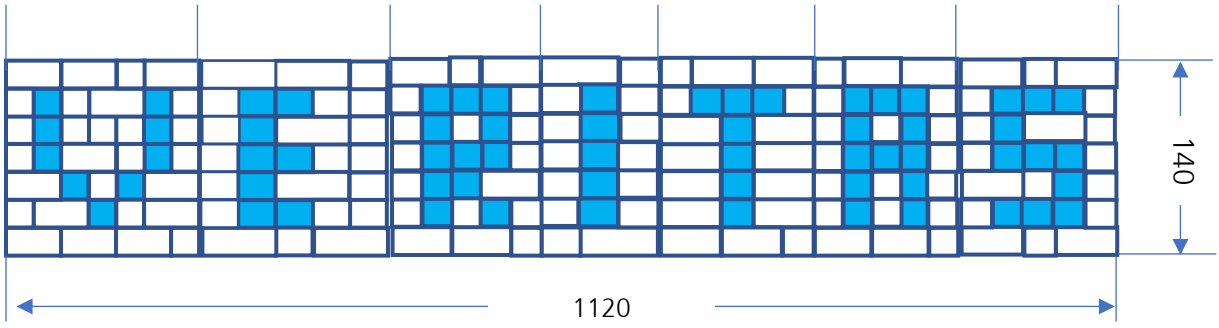


Abbildung 12: Ursprüngliches Montageszenario mit Verteilung langer/kurzer Steine

Die entsprechenden Bausteine sind in Behältern, erreichbar von allen Roboterarmen, vorrätig. Zur Hervorhebung werden zwei Farben sowie lange und kurze Steine miteinander kombiniert. Im Regelfall sind die Bausteine einem einzelnen Roboter zugeordnet und der Buchstabe wird entsprechend der Vorgaben aus Abb. 13 gemeinsam errichtet. Bei Nichtverfügbarkeit eines Roboterarms übernimmt selbständig ein anderer Roboterarm dessen Tätigkeit.

Die Software des Projekts Veritas wurde auf das vorliegende, leicht veränderte Hardware-Setup angepasst und erfolgreich in Betrieb genommen.

3 Ergebnisse

3.1 Wissenschaftliche Verwertung

Folgende Publikationen wurden während der Projektlaufzeit erstellt:

- Konferenzbeitrag auf der Automation 2021: "Selbstorganisation und Co-Management verteilter Automatisierungssysteme mit dem Kommunikationssystem"[1], Die Veröffentlichungsrechte liegen bei VDE Verlag, daher ist das Paper kein Teil dieses Berichts

- Journalbeitrag in IEEE Proceedings on System and Control (eess.SyS), 2020: "Wireless Control for Smart Manufacturing: Recent Approaches and Open Challenges"[2], Die Veröffentlichungsrechte liegen bei VDE Verlag, daher ist das Paper kein Teil dieses Berichts

- Konferenzbeitrag auf der IEEE CASE 2023: „Self-Organization and Collaboration in Robotic Manufacturing Systems"[3], eingereicht

Neben einer Vielzahl studentischer Arbeiten im Rahmen von Praktika wurden 1 Studienarbeit betreut und werden aktuell 2 Diplomabschlussarbeiten im Kontext der Roboter-Kollaboration betreut.

3.2 Voraussichtliche Nutzen und Verwendbarkeit der Ergebnisse

In dem Projekt VERITAS hat das Fraunhofer IIS/EAS mehrere unterschiedliche Ergebnisse erarbeitet. Als erstes ist hier das entworfene Architekturmodell für verteilte, kollaborierende Produktionssystem zu benennen, das neben Elementen der Selbstorganisation und Kollaboration auch das Co-Design der verteilten agentenbasierten Produktionssysteme und des eingesetzten Kommunikationssystems mit einbezieht. Das Architekturmodell bildet die Basis für die weitere Vernetzung auf nationaler und internationaler Ebene mit Forschungspartnern und Industrieverbünden des Bereiches Industrie 4.0 bzw. Industrial-IoT. Durch diesen Austausch werden die Anpassung und Weiterentwicklung des Architekturmodells sichergestellt. Es bietet gleichermaßen die Basis für die Initiierung von Folgeprojekten zur Weiterentwicklung und Optimierung. Gleichzeitig wird damit die Grundlage dafür geschaffen, darauf aufbauende Consulting- und Beratungsdienstleistungen zu neuen Systemkonzepten im Umfeld der Industrieautomation und autonomer verteilter, agentenbasierter Fertigungssysteme anbieten zu können. Darüber hinaus werden weitere Anwendungsgebiete (z.B. Smart City, Smart Energy und

[1] T. Zeun, A. Frotzscher, A. Nicklas, E. Schmitt, D. Schulze-Zipper, H. Zipper, „Selbstorganisation und Co-Management verteilter Automatisierungssysteme mit dem Kommunikationssystem", AUTOMATION 2021, 2021, ISBN: 978-3-18-092392-4

[2] D. Baumann, F. Mager, U. Wetzker, L. Thiele, M. Zimmerling, S. Trimpe, „Wireless Control for Smart Manufacturing: Recent Approaches and Open Challenges", IEEE Proceedings on System and Cotnrol (eess.SyS), 2020

[3] A. Frotzscher, X. Chen, D. Tran, H. Sadeghian, M. Dehmel, T. Wohlfahrt, M. Riedl, S. Haddadin, „Self-Organization and Collaboration in Robotic Manufacturing Systems", Proceeding of IEEE conference on Automation Science and Engineering (CASE), 2023, Eingereicht

vernetzte Mobilitätskonzepte) analysiert und hinsichtlich einer Übertragbarkeit des Architekturmodells insgesamt oder einzelnen Teilaspekten dessen bewertet.

Ein weiteres Ergebnis ist das entwickelte Framework zur Modellierung des Übertragungsverhaltens von Kommunikationssystemen und dessen Auswirkung auf verteilte Applikationen. Im Projekt Veritas wurde das Framework in eine ROS-Umgebung integriert und kann so für die Systemverifikation von ROS-basierten verteilten Automatisierungsanwendungen unter Einbeziehung des Kommunikationssystems genutzt werden. Dieses Projektergebnis wird zur Initiierung von Nachfolge-Projekten genutzt und für Beratungsdienstleistungen im Kontext verteilter Produktionssysteme eingesetzt werden.

4 Zahlenmäßiger Nachweis

Die wichtigsten Positionen des zahlenmäßigen Verwendungsnachweises sind die Personalkosten für Wissenschaftler und Hilfswissenschaftler bzw. Ingenieure. In wesentlich geringerem Maße sind Kosten für Reisen (Projekttreffen, Konferenzen) sowie Materialkosten entstanden.

Der detaillierte zahlenmäßige Nachweis erfolgt separat durch die zentrale Verwaltung der Fraunhofer-Gesellschaft.