OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
FACULTY OF COMPUTER SCIENCE
Communication and Networked Systems

Communication and Networked Systems

Bachelorarbeit

Auswirkungen verschiedener Dienstgüten auf einen "Peg in Hole"-Lösungsalgorithmus in einem Networked Control System

Martin Popp

Betreuer: Prof. Dr. rer. nat. Mesut Güneş
Betreuender Assistent: Frank Engelhardt, M.Sc.

Institut für Intelligente Kooperierende Systeme, Otto-von-Guericke-Universität Magdeburg

3. November 2020, Magdeburg

Kurzzusammenfassung

Diese Bachelorarbeit befasst sich mit den Auswirkungen verschiedener Dienstgüten (QoS) auf die Qualität der Ausführung eines komplexen Regelungsalgorithmus in einem vernetzten Regelungssystem (NCS). Die Auswirkungen sollen anhand eines selbst erstellten NCS experimentell untersucht werden. Alle dafür benötigten Komponenten wurden im Rahmen dieser Arbeit ebenfalls entwickelt und implementiert. Als Referenzalgorithmus ist hierfür ein Lösungsalgorithmus für eine kraftgesteuerte „Peg in Hole“-Aufgabe gewählt worden. Diese in der Industrie häufig vorkommende Montageaufgabe soll von einem Roboterarm in der Robotersimulationsumgebung CoppeliaSim ausgeführt werden. Um reale Netzwerkbedingungen darzustellen und diese für verschiedene Experimente schnell und einfach verändern zu können ist darüber hinaus ein Programm zur Simulation eines Netzwerkes implementiert worden. Mit diesen Komponenten sind anschließend Experimente durchgeführt worden, bei denen insbesondere die zur Lösung der Aufgabe benötigte Gesamtzeit sowie die auftretenden Maximalkräfte im Fokus standen.

Durch den Einfluss der Verzögerungen verlängerte sich die zur Beendigung der Aufgabe benötigte Zeit erwartungsgemäß. Bei hohen Latenzen und Paketverlustraten kam es am Roboterarm zu erheblichen Vibrationen, die für Schwankungen der Kräfte und damit zu Fehlern im Algorithmusablauf führten. Dennoch blieben die Maximalkräfte innerhalb eines akzeptablen Bereichs.

Abstract

This bachelor thesis deals with the effects of different quality of service (QoS) on the quality of execution of a complex control algorithm in a networked control system (NCS). The effects should be studied experimentally using a self-created NCS. All necessary components were also developed and implemented within the scope of this thesis. As reference algorithm a solution algorithm for a force controlled ”peg in hole” task was chosen. This assembly task, which is frequently encountered in industry, is to be executed by a robot arm in the robot simulation environment CoppeliaSim. In order to represent real network conditions and to be able to change them quickly and easily for different experiments, a program for the simulation of a network was also implemented. With these components, experiments were subsequently carried out with a special focus on the total time required to solve the task and the maximum forces that occur.

Due to the influence of the delays, the time needed to complete the task increased as expected. With high latencies and packet loss rates, the robot arm was affected by considerable vibrations, which led to fluctuations of the forces and thus to errors in the algorithm sequence. Nevertheless, the maximum forces remained within an acceptable range.

Genderhinweis

Es wurde in dieser Bachelorarbeit versucht, vorwiegend geschlechtsneutrale Bezeichnungen zu verwenden. Sollte an der einen oder anderen Stelle doch, wie im deutschsprachigen Raum üblich, eine maskuline Bezeichnung auftauchen, so schließt diese selbstredend auch die feminine Form mit ein.

Danksagungen

Der Autor möchte sich hiermit bei Prof. Dr. Mesut Güneş und M.Sc. Frank Engelhardt für dieses Thema und die geleistete Unterstzützung bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

CCD charge-coupled device. 5

DoF Degrees of Freedom. 13, 33

F/T Force/Torque. 6

FIFO First In First Out. 11, 31

MIoT-Lab Magdeburg Internet of Things Lab. 34

NCS Networked Control System. 2, 3, 5, 7, 8, 16, 33, 34

OWD One-Way-Delay. 4, 11, 32

QoS Quality of Service. 4

RTT Round Trip Time. 4, 32

SVN Apache Subversion. 7, 16

TCP Transmission Control Protocol. 11

TSCH Time Synchronized Channel Hopping. 4

UDP User Datagram Protocol. 11

KAPITEL 1

Einleitung

1.1 Motivation

Während meines Studiums nahm ich an einem Softwareprojekt teil, dessen Ziel es war eine Flugdrohne über ein Multi-hop Netzwerk aus mehreren Routern in einem anderen Teil des Gebäudes zu steuern. Bei dieser Arbeit ging es primär um die Entwicklung eines Routingprotokolls, mit dem diese Aufgabe umsetzbar sein sollte. Dabei blieb es nicht aus sich in diesem Zusammenhang mit vernetzten Regelungssystemen, sogenannten *Networked Control Systems (NCS)*, auseinanderzusetzen. Die Drohne, als *gesteuertes Gerät*, besaß neben einigen anderen Sensoren auch eine Kamera, deren Live-Bild, über ein *Netzwerk* aus mehreren Routern, an ein Smartphone (*Controller*) gesandt wurde. Auf dem Smartphone musste eine App ausgeführt werden, mit der das Bild angezeigt und die Steuerung durch einen Nutzer ermöglicht werden konnte. Diese Steuerungsdaten wurden über dasselbe Netzwerk zurück an die Drohne gesandt. Bei der Verwendung vieler Hops im Netzwerk kam es teilweise zu erheblichen Verzögerungen oder gar Aussetzern bei der Datenübertragung, da alle Router kleine Delays beisteuerten, die sich akkumulierten. Besonders bei größeren Strecken und wenn sich (kabelgefüllte) Wände zwischen den Routern befanden oder aufgrund anderer WLAN-Signale im Gebäude, konnten starke Interferenzen auftreten. Dadurch kamen Datenpakte manchmal unvollständig oder gar nicht am Ziel an. Dies wirkte sich dahingehend aus, dass auf der Controllerseite (Smartphone) das Kamerabild oft verzerrt, unvollständig oder ruckelnd war und manchmal sogar ganz ausfiel. Auf der Seite der gesteuerten Drohne kamen keine oder falsche Steuerungssignale an, so dass die Drohne abstürzte oder unkontrolliert im Fluglabor kreiste. Obwohl wir während des Softwareprojektes versuchten diese Auswirkungen, mittels des entwickelten Routingprotokolls und durch das Verringern der Abstände zwischen den Routern, zu minimieren, wurden diese und eventuell von uns noch unbemerkte Auswirkungen nicht weiter untersucht.

1.2 Ziel der Arbeit

Aus diesem Grund soll im Rahmen dieser Bachelorarbeit experimentell ermittelt werden, wie sich Verzögerungen in der Datenübertragung oder der Verlust von Datenpaketen auf

ein Networked Control System (NCS) auswirken.

NCS sind in vielen Bereichen und auch aus modernen Industrieanlagen kaum noch wegzudenken, da sie eine Vielzahl von Vorteilen bieten. Durch die kabellose Kommunikation zwischen den Komponenten lassen sich solche Anlagen wesentlich einfacher warten und erweitern. Darüber hinaus sorgen stetig sinkende Preise der Hard- und Software für eine bessere Kosteneffizienz, so dass herkömmliche kabelgebundene Systeme zunehmend abgelöst werden. [1] Das Netzwerk in solchen Systemen bringt jedoch auch Herausforderungen mit sich. So kann eine verlustfreie und verzögerungslose Datenübertragung nicht durchgehend gewährleistet werden. Die Einflüsse verschiedener Netzwerkqualitäten müssen demnach bereits beim Entwurf vernetzter Regelungssysteme berücksichtigt werden. [2] [3, S. 2]
Da es jedoch keine Studien über das Verhalten von komplexen Regelungsalgorithmen unter schlechten Netzwerkbedingungen gibt, soll dies in dieser Bachelorarbeit experimentell untersucht werden. Mit dem Hintergrund der automatisierten Montage und, um die in den Experimenten durchgeführten Schritte wiederholbar zu halten, wurde sich dazu entschlossen einen Lösungsalgorithmus für eine „Peg in Hole“-Aufgabe zu entwickeln. Anschließend soll die Ausführungsqualität dieses Algorithmus unter verschiedenen Netzwerkbedingungen untersucht werden. Die, im Abschnitt 2.3 erläuterte „Peg in Hole“-Aufgabe, zu deutsch „Zapfen in Loch“ ist in der automatisierten Industrie, insbesondere bei Montagevorgängen, eine häufig vorkommende Tätigkeit. Von Komponenten im Luftfahrtsektor, über Motoren und Windschutzscheiben bis hin zu kleinsten Mikroprodukten sollen Montageroboter diese Aufgabe erfüllen können [4]. Daher ist dieses Problem in der Forschung eine viel untersuchte Thematik.
Obwohl diese Aufgabe auch mit Kameraunterstützung durchgeführt werden kann, wurde sich hier für einen Lösungsansatz mit der Unterstützung eines Kraft-/Momentensensors entschieden. Dies liegt darin begründet, dass im Kontext einer industriellen Anwendung von einer Steuerung unter nicht-optimalen Bedingungen, wie schlechten Lichtverhältnissen in einer Fabrikhalle oder Schmutz in der Anwendungsumgebung, ausgegangen wird. Hier ist der Einsatz eines kameragestützten Systems nur bedingt sinnvoll.[5]
Eine Regelung, die eine Rückmeldung der auftretenden Kräfte verarbeiten kann, ist daher ein hilfreicher und in kritischen Bereichen notwendiger Bestandteil derartiger Montagesysteme. Solch ein Regelungsalgorithmus als *Controller* wird im Rahmen dieser Bachelorarbeit entwickelt. Zur Simulation des *Netzwerkes* wird darüber hinaus ein Routingprogramm mit einstellbaren Parametern für Latenzen oder Paketverluste implementiert. Durch die einfache Anpassbarkeit verschiedener Dienstgüten soll dieses Programm die Experimente erleichtern. Als *gesteuertes System* soll ein Roboterarm in der Robotersimulationsumgebung *CoppeliaSim* verwendet werden, dessen Steuerung über ein ebenfalls entwickeltes Plugin für die Simulationsumgebung umgesetzt wird. Im Anschluß werden mit diesen Komponenten Versuche durchgeführt, die klären sollen, ob und wie die Qualität der Regelung unter den verschiedenen Netzwerkbedingungen variiert und, ob es kritische Umstände gibt, die einen erfolgreichen Abschluss der Aufgabe verhindern. Konkret soll dabei die zur Lösung der Aufgabe benötigte Gesamtzeit sowie die auftretenden Maximalkräfte betrachtet und die Fragen beantwortet werden, ob ein signifikanter Kraftanstieg von mehr als 100% auftritt und inwiefern am Roboter selbst Auswirkungen zu beobachten sind.

KAPITEL 2

Related Work

Im folgenden Kapitel erfolgt eine Einführung in die Thematik der vernetzten Regelungssysteme. Hierfür wird auf grundlegende Begriffe eingegangen, die im Verlauf dieser Bachelorarbeit eine zentrale Rolle spielen. Insbesondere soll das Problem der Zeitverzögerungen und Paketverluste in einem NCS beleuchtet werden. Darüber hinaus werden mögliche Umsetzungen der „Peg in Hole“-Aufgabe dargestellt. Dieses Kapitel soll neben der Literaturdiskussion auch eine Einordnung in den aktuellen Forschungsstand geben.

2.1 Networked Control Systems

Ein NCS ist ein Regelungssystem, bei dem der Regler (im Folgenden Controller genannt) und das von ihm gesteuerte System, nicht mehr in derselben Anlage verbaut und über teure, wartungsintensive Kabel verbunden sein müssen [2] [6]. Die beteiligten Komponenten können stattdessen weit verteilt und über Kontinente hinweg voneinander entfernt sein [7]. Solche Systeme sind flexibler, einfacher zu warten und leichter skalierbar, als ihre kabelgebundenen Pendants [1]. Dank der stetig sinkenden Preise für Netzwerktechnologien werden NCS kosteneffizienter, so dass sie herkömmliche Steuerungs- und Regelungssysteme zunehmend ablösen [2] [6] . Die zu übertragenden Daten eines NCS bestehen üblicherweise aus den Sensordaten des gesteuerten Systems sowie den Steuerungsdaten des Controllers. Übertragen werden diese Daten durch ein Netzwerk, über das auch die Regelschleifen geschlossen sind [3]. Ein solches Netzwerk kann beispielsweise ein lokales Netzwerk in einer Fabrikhalle sein, über das mehrere Fertigungsanlagen oder Roboter von einem zentralen Controller gesteuert werden [8]. `Papacharalampopoulos et al.`(2016) zeigten durch die Modellierung eines solchen Produktionssubsystems Herausforderungen, wie Verzögerungen der Signale, Paketverluste und Messverzerrungen durch Interferenzen eines NCS auf [9]. Dabei stellten sie fest, dass sich insbesondere Verzögerungen und Paketverluste auf die Stabilität des Regelungssytems auswirkten. Ein weiteres Anwendungsgebiet könnte die Telerobotik sein, durch die unter anderem ein medizinischer Operationsroboter in Frankreich über ein Netzwerk von einem spezialisierten Arzt in den USA steuerbar wäre [7]. Bei der Implementierung eines dafür nötigen bidirektionalen Kommunikationsflusses über große Entfernungen bestehen jedoch nach wie vor erhebliche Schwierigkeiten, wie `Avgousti et al.`(2018) in ihrer Studie konstatierten. Insbesondere beim zeitkritischen Datenaustausch über große Entfernungen

sind Kommunikationsverzögerungen und Datenverluste bestehende Herausforderungen.[10]

Daher werden im Rahmen dieser Arbeit Versuche mit einem selbstentwickelten Netzwerk mit einstellbaren Parametern durchgeführt, um unter anderem Netzwerkbedingungen mit großen Verzögerungen oder Datenverlusten und damit weiten Entfernungen zu simulieren.

`Schindler et al.`(2017) führten Versuche mit einem Time Synchronized Channel Hopping (TSCH)-Netzwerk durch und bestimmten unter anderem, wie sich die Netzwerklatenz bei verschiedenen Gegebenheiten des Netzwerkes, wie variablen Timeslots oder einer unterschiedlichen Menge von Hops, verändert. Darüber hinaus überprüften sie anhand eines Algorithmus, ob und wie sich die Regelung eines inversen Pendels unter denselben Bedingungen verhält. Das inverse Pendel ist ein aufrecht auf einem kleinen Cart gelenkig angebrachter Stab, der so balanciert werden muss, dass er nicht umfällt. Dafür kann das Cart entsprechend der Winkelmessungen des Pendels vor und zurück fahren. Dieser Regelungsalgorithmus konnte mit dem vorgestellten TSCH-Netzwerk bei verschiedenen Netzwerklatenzen erfolgreich durchgeführt werden.[11] Hingegen konnten Studien über das Verhalten eines komplexen Algorithmus unter schlechten Netzwerkbedingungen nach bestem Wissen und Gewissen bei den Recherchen nicht gefunden werden. Im Rahmen dieser Bachelorabeit soll ein Lösungsansatz für diese Forschungslücke geboten werden.

2.2 Quality of Service

Unter der Qualität eines Kommunikationsdienstes (Quality of Service (QoS)) versteht man im Allgemeinen, dass dieser die erwartete Funktionalität in angemessener Qualität zur Verfügung stellt. Im Speziellen kann auch die Geschwindigkeit und Zuverlässigkeit einer Kommunikationsverbindung als QoS bezeichnet werden [12]. Einfluß darauf haben unter anderem verschiedene Netzwerkparameter, wie die Bandbreite, die Latenz oder die Paketverlustrate [13].

Die **Bandbreite** gibt im Zusammenhang mit der Signalverarbeitung eine Übertragungsfrequenz in $Hertz$ $[Hz]$ an. Im Kontext von Netzwerken auch Übertragungsrate genannt, bezeichnet sie üblicherweise die maximale Anzahl von Bits, die in einem bestimmten Zeitintervall übertragen werden können [14, S. 29]. Der Begriff Durchsatz, der oft synonym zur Bandbreite genutzt wird, meint hingegen die tatsächlich gemessenen Bits pro Sekunde. Angegeben werden die Bandbreite und der Durchsatz in $\frac{bit}{s}$ oder $\frac{Mbits}{s}$. [15, S. 40f]

Die (Netzwerk-)**Latenz** (auch *Verzögerung*) kann als Round Trip Time (RTT) betrachtet werden, sie bezeichnet dann den Zeitraum, den eine Nachricht bestimmter Größe vom Sender zum Empfänger und wieder zurück zum Sender benötigt [16, S. 42]. Betrachte man die Dienstgüten, ist jedoch meist das One-Way-Delay (OWD) gemeint. Dies ist die Verzögerung, die ein Datenpaket vom Sender mit Hilfe des Netzwerkes zum Empfänger benötigt [13, S. 253]. Im weiteren Verlauf dieser Arbeit bezieht sich die Latenz ausschließlich auf das OWD, da dieses auch als Stellgröße der Experimente implementiert wurde und insbesondere Dienstgüten betrachtet werden. Diese Ein-Weg-Latenz addiert sich aus der Übertragungsverzögerung sowie Wartezeiten im Netzwerk und der Ausbreitungsverzögerung der Signale. Die Übertragungsverzögerung ist dabei das Verhältnis von Größe der Nachricht zur Bandbreite des Netzwerkes [13, S. 253]. Je größer die Nachricht und je geringer die Bandbreite, umso länger wird demnach die Verzögerung [15, S. 42]. Wartezeiten können

darüber hinaus durch Geräte im Netzwerk entstehen, die empfangene Daten vor dem Weiterleiten erst zwischenspeichern [14, S. 29]. Letztlich führt die Ausbreitungsverzögerung, welche von der Länge $l[m]$ des Übertragungsmediums sowie der Lichtgeschwindigkeit $c[\frac{m}{s}]$ und einem Ausbreitungsfaktor γ abhängig ist, zu einer weiteren Verlängerung der Latenz. Hierbei beschreibt der Ausbreitungsfaktor die Art des Mediums. Im Vakkum beträgt dieser beispielsweise 1 und bei der Verwendung von Glasfaserkabeln 0.67. Somit errechnet sich die Ausbreitungsverzögerung nach [14, S. 29]:

$$t_A = \frac{l}{c \cdot \gamma}. \tag{2.1}$$

Die **Paketverlustrate** bezeichnet das Verhältnis von abgesendeten, jedoch nicht beim Empfänger eingetroffenen Datenpaketen und ist ein weiteres Maß für die Qualität eines Kommunikationskanals. Paketverluste in NCS können in zwei Hauptkategorien unterschieden werden. Einerseits können Datenpakete durch Netzwerkinstabilitäten oder nicht korrigierbare Übertragungsfehler tatsächlich nicht beim Empfänger ankommen. Andererseits sind Daten, deren Übertragungslatenz größer ist als die Zeit zwischen zwei Sensorabtastungen, bereits überholt und somit nicht mehr zu gebrauchen. [1]

2.3 „Peg in Hole“

Einen Gegenstand in eine Öffnung einzupassen ist für einen erwachsenen Menschen eine relativ einfache Aufgabe. Mit Hilfe des Sehvermögens lässt sich sowohl die Öffnung leicht identifizieren, als auch der Gegenstand in die grobe Position bringen. Der Tastsinn wird abschließend genutzt, um den Gegenstand passgenau und korrekt ausgerichtet zu platzieren. [17]

In Montageprozessen kommt diese Aufgabe häufig vor und wird heutezutage noch oft von Menschen ausgeführt [18] [19]. Dies bringt jedoch auch einige Probleme mit sich. Unter anderem ist die Unfallgefahr, inbesondere in gefährlichen Umgebungen, für den Menschen höher, als für eine Maschine und mancherorts können Menschen gar nicht eingesetzt werden. Weiter sind Maschinen kosteneffizienter [19]. Daher ist die Erforschung und Verbesserung der Automatisierung der „Peg in Hole“-Aufgabe so relevant und wurde als Aufgabenstellung für diese Bachelorabeit gewählt.

Die Größe und Form des einzupassenden Gegenstandes ist dabei nicht festgelegt. So gibt es Aufgaben im Elektronik- oder Mikroelektronikbereich, bei denen winzige Steckverbindungen ineinandergepasst werden müssen. `Chang et al.`(2011) entwickelten und testeten ein Mikromontagesystem, das mit Hilfe eines visuellen Servosystems und charge-coupled device (CCD)-Sensoren zylindrische Mikrostifte mit Größen von $80\,\mu m$ in Löcher von $100\,\mu m$ Durchmesser einpassen sollte. Die von ihnen entwickelte Regelung konnte diese Aufgabe mit einer Erfolgsquote von ca. 80 % erfüllen [20].
Weitere Anwendungsgebiete sind beispielsweise die Automobil- oder Luftfahrtindustrie, wo größere und unterschiedlich geformte Komponenten ineinandergepasst werden müssen [4]. `Qin et al.`(2016) gelang es Lokalisierungsfehler von nur $0.53\,mm$ Ausrichtung und 0.1° Orientierung bei der Montage eines $710\,mm$ x $625\,mm$ großen Objektes zu erreichen. Dafür verwendeten sie Laserdistanzsensoren und Kameras [21].
Schlechte Sichtverhältnisse, Unregelmäßigkeiten der Montageteile aber auch Positionsungenauigkeiten der Montageapperatur selbst können beim Zusammenbau dazu führen, dass

Bauteile unter großen Krafteinwirkungen aufeinandergepresst und beschädigt werden können [22]. Um dies zu verhindern, müssen andere Strategien angewandt werden. `Zhang et al.`(2017) nutzten ausschließlich einen Kraft-/Momentensensor, um zwei parallel ausgerichtete Zylinder gleichzeitig in zwei parallele Löcher einzupassen, ohne dass die Kontaktkräfte bestimmte Schwellenwerte überschritten. Bei ihrem Experiment waren die Zylinder gut über den Löchern positioniert, so dass primär eine Feinjustierung der Position und die Ausrichtung nötig waren [22]. Von `Tang et al.`(2016) wurde ein Roboter mit Force/Torque (F/T)-Sensor genutzt, um eine eigens entwickelte Methode zu überprüfen, mit der Ausrichtungsungenauigkeiten zwischen einzupassendem Zylinder und Loch von vornherein vermieden werden sollten [23]. Es gelang ihnen einen Zapfen mit $25.37\,mm$ Durchmesser in ein nur $0.03\,mm$ größeres Loch (H7h7 Toleranz) mit einer Erfolgsquote von 96% einzupassen. Anders als in dieser Bachelorabeit fand die Kommunikation zwischen den Sensoren/Aktuatoren und dem Controller jedoch nicht über ein Netzwerk statt, so dass alle Komponenten nahezu verzögerungs- und verlustfrei miteinander kommunizieren konnten.

KAPITEL 3

Thesis-Beitrag

Im folgenden Kapitel wird erläutert, wie die Auswirkungen verschiedener Dienstgüten eines Netzwerkes in einem Networked Control System (NCS) experimentell ermittelt werden sollen. Die dafür entwickelten Komponenten werden hinsichtlich der benötigten Anforderungen und der daraus folgenden Umsetzungen beschrieben und es wird auf einige Details zur Implementierung eingegangen. Der vollständige Quellcode ist in dem, zur Bachelorarbeit gehörenden Apache Subversion (SVN) [1] sowie auf der beiliegenden CD-ROM zu finden.

3.1 Konzept

Diese Bachelorarbeit soll aufzeigen, wie sich Zeitverzögerungen durch eine begrenzte Netzwerkbandbreite oder hohe Latenzen und Paketverluste in der Datenübertragung auf die Qualität der Ausführung eines komplexen Regelungsalgorithmus in einem vernetzen Regelungssystem (NCS) auswirken. Der hierfür entwickelte Algorithmus soll eine „Peg in Hole"-Aufgabe lösen und als Referenz für die Experimente dienen. Um vergleichbare Daten unter verschiedenen Netzwerkbedingungen zu erhalten, muss der Algorithmus, die zur Lösung der Aufgabe nötigen Schritte stets in der gleichen Art und Weise durchführen. Der Algorithmus erhält von einem Roboterarm in der Simulationsumgebung *CoppeliaSim*, bzw. dem dafür entwickelten Plugin als *gesteuertem System*, die Positions- und Orientierungsdaten aller drei Hauptachsen (x,y,z) einer Dummykugel (im Folgenden: *Sphäre* genannt). Diese befindet sich an der Spitze des einzupassenden Zylinders. Der Roboterarm wird durch die Simulationsumgebung so gesteuert, dass er der Sphäre folgt und die Winkel der einzelnen Gelenke automatisch angepasst werden. Darüber hinaus werden auch die Messdaten eines Kraft-/Momentensensors, der im Magnetgreifer an der Spitze des äußersten Armteils befestigt ist, an den Algorithmus übertragen. Auf Grundlage dieser Daten werden daraufhin vom Algorithmus neue Positions- und Orientierungswerte berechnet und über das Netzwerk zurück an das Plugin gesendet. Vom Plugin werden die neuen Steuerungsdaten durch den Aufruf von API-Funktionen an die Simulation übergeben und dort umgesetzt. Das Plugin enthält die Funktionen zur Erfassung der Sensor- und Positionsdaten sowie zum Setzen der eingehenden Steuerungsdaten. Darüber hinaus sind auch die Funktionen zur Kommu-

[1] `https://comsyssvn.ivs.cs.ovgu.de/repos/students/BSc/2019-Popp-Martin/`

nikation mit dem Algorithmus, sowie einige weitere anwendungsspezifische Funktionen, die im Folgenden erläutert werden sollen, implementiert. Um während der Experimente verschiedene Dienstgüten (QoS) des Netzwerkes möglichst unkompliziert und ohne signifikante Abweichungen testen zu könnnen, wird außerdem ein Programm zur Netzwerksimulation implementiert, welches eingehende Nachrichten weiterleiten kann und somit ähnlich wie ein Router funktioniert. Diese Weiterleitung kann entsprechend vorher festgelegter Parameter verzögert oder gar übersprungen werden, um somit die Bandbreite, Latenz und Paketverlustrate eines Netzwerkes zu simulieren. Alle Komponenten wurden in der Programmiersprache C/C++ umgesetzt.

3.2 Implementierung

Einige Details der Implementierungen der vorab erwähnten Komponenten des NCS werden im folgenden Abschnitt genauer erläutert und bestimmte Implementierungsdetails diskutiert.

3.2.1 *Controller:* „Peg in Hole"-Algorithmus

Bei der in Abschnitt 2.3 bereits vorgestellten „Peg in Hole"-Aufgabe soll ein zylindrischer Zapfen, der sich im Greifer eines Roboterarmes befindet, möglichst genau in ein nur wenig größeres Loch eingepasst werden.

Da der hier verwendete Roboter über keinerlei Sensoren zur visuellen Unterstützung verfügt, werden damit beispielsweise Umgebungen imitiert, in denen sehr schlechte Sichtverhältnisse herrschen. Dies können beispielsweise nicht klare Flüssigkeiten wie Farbe, Öl oder ähnliches sein. Die Aufgabe muss daher ausschließlich auf Grundlage der Messdaten des integrierten Kraft-/Momentensensors gelöst werden. Darüber hinaus wird dem Controller, welcher als Regelungsalgorithmus fungiert, nur die aktuelle Position der Sphäre und die Maße der Platte übermittelt. Dies bedeutet, dass auch die grobe Position des Loches auf der Platte unbekannt ist und damit erst ermittelt werden muss. Daher beginnt die vollständige Suche in der ersten Phase in einer der äußersten Ecken der Platte.

Bei einem senkrecht ausgerichteten Zylinder entspricht die Kontaktfläche der Grundfläche des Zylinders, so dass beim Auftreten einer Kraftänderung nicht festgestellt werden kann, an welchem Punkt dies geschieht. Daher wird der Zylinder an der Startposition leicht gekippt, um eine beinahe punktförmige Kontaktfläche zwischen Platte und Zylinder herzustellen und so die Messergebnisse möglichst positionsgenau zuordnen zu können. Daraufhin wird der Zylinder auf die Platte hinabgesenkt, bis eine bestimmte Anpresskraft erreicht ist und dann in einer Bahn parallel zur Plattenkante über die Fläche gezogen. Dabei werden mit einer Frequenz von ca. 60 Hz die Kräfte und Momente des Sensors erfasst und über das Netzwerk an den Algorithmus gesendet. Diese Frequenz ist durch die Aufruffrequenz der *simMessage*-Funktion seitens der Simulationsumgebung vorgegeben. Falls das gegenüberliegende Ende der Bahn erreicht wird ohne, dass die gemessene Kraft einen Grenzwert unterschreitet, hebt der Zylinder ab, wird neu ausgerichtet und auf die nächste parallele Bahn im Abstand von etwa 75% des Zylinderdurchmessers gesetzt. Daraufhin beginnt die Prozedur von Neuem. Sobald die Platte ohne Erfolg komplett abgesucht wurde, wird eine Meldung ausgegeben,

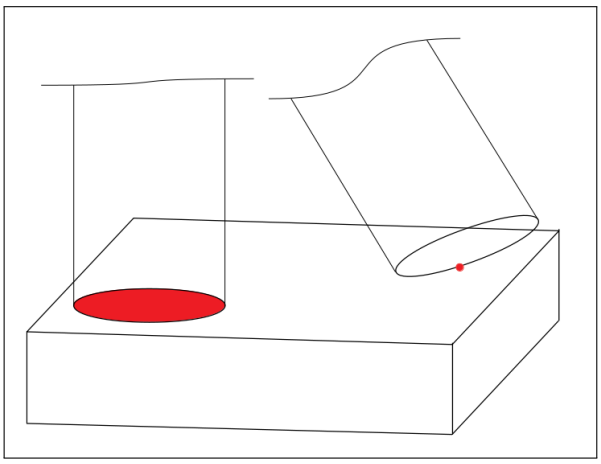

Abbildung 3.1: Die Kontaktfläche des Zylinders in rot. Links senkrechter Zylinder, rechts ist der Zylinder um 20 Grad geneigt (eigene Darstellung)

der Roboterarm wird wieder in die Initialposition bewegt und der Algorithmus ist beendet. Fällt die Kraft während der parallelen Kontaktfahrt unter einen Grenzwert, wird davon ausgegangen die Kante eines Loches überfahren zu haben. Die grobe Lochposition ist nun bekannt und die nächste Phase beginnt.

In Phase zwei muss die exakte Position des Lochmittelpunktes gefunden werden, so dass der Zylinder möglichst genau über dem Loch positioniert und reibungslos eingepasst werden kann.

Diese Teilaufgabe wird hier mittels Dreieck-Mittelpunkt-Berechung gelöst. Der Mittelpunkt eines Dreiecks ist der Schnittpunkt der Mittelsenkrechten der drei Dreiecksseiten. Gleichzeitig ist dieser Punkt auch der Mittelpunkt des Dreiecks-Umkreises, also des Kreises, der durch alle drei Eckpunkte verläuft.

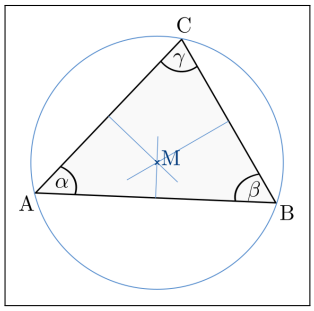

Abbildung 3.2: Mittelpunkt des Dreieck umschließenden Kreises[2]

Hierfür müssen so präzise wie möglich drei Punkte bestimmt werden, die auf der Lochkante liegen. Während der Experimente hat sich herausgestellt, dass die Messwerte des Kraft-/Momentensensors, wie in Kapitel 4 und 5 beschrieben, ungleichmäßig sind. Daher wurde entschieden, insgesamt sechs Punkte zu bestimmen, so dass aus allen sich daraus ergebenden Dreiecken zehn Mittelpunkte berechnet werden können. Aus diesen soll dann ein

[2] https://www.grund-wissen.de/mathematik/elementare-geometrie/planimetrie/dreiecke.html

repräsentativer Mittelwert gebildet werden, der möglichst genau die tatsächliche Position des Lochmittelpunkts widerspiegelt.

Zur Bestimmung dieser sechs Punkte werden in einem Umkreis von ca. zwei Zylinderdurchmessern um den geschätzten Lochmittelpunkt entsprechend viele Punkte in einem sternförmigen Muster (siehe Abb.3.3) angefahren. An jedem dieser Punkte wird der Kontakt zur Platte wiederhergestellt, der Zylinder geneigt und anschließend in Richtung des geschätzten Lochmittelpunktes gezogen. Ebenso wie in Phase 1 wird bei einem plötzlichen Abfall der Kraft, das Überfahren einer Lochkante angenommen und der Punkt wird gespeichert. Endet die Bewegung jedoch genau im geschätzten Lochzentrum und liegt dieses auf einer Kante, wird auch dieser Punkt gespeichert.

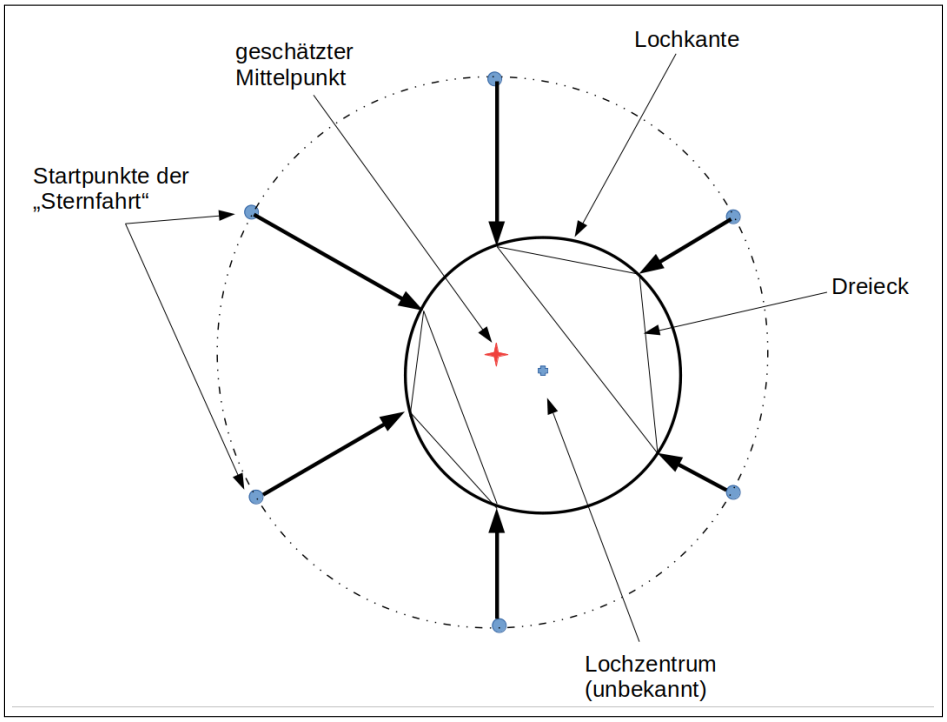

Abbildung 3.3: Die graphische Darstellung der „Sternfahrt„ (eigene Darstellung)

Sind sechs Punkte bestimmt, beginnt die letzte Phase des Algorithmus.
Zunächst werden die zehn Mittelpunkte berechnet und die fünf größten Abweichungen vom Mittelwert entfernt, bevor der Mittelwert der übrigen Punkte als Lochzentrum gesetzt und angefahren wird. Dort wird der Zylinder senkrecht ausgerichtet und langsam in das Loch hinabgesenkt, wo der Magnetgreifer den Zylinder frei gibt. Anschließend begibt sich der Roboterarm zurück auf die Initialposition, zentriert über der Platte.

3.2.2 Netzwerk & Kommunikation

Die Kommunikation muss ohne gewollte Verzögerungen oder Paketverluste seitens des Netzwerkes zuverlässig funktionieren, um während der Experimente immer die gleichen Umstän-

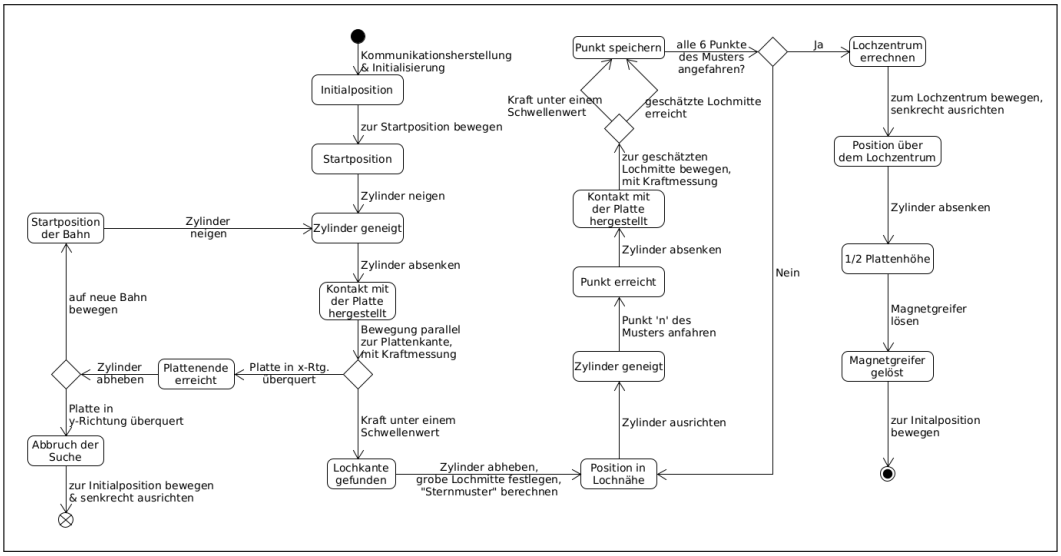

Abbildung 3.4: Der Ablauf des Algorithmus als Zustandsmaschine (groß im Anhang A.1)

de zu bieten. Einzig die Abtastrate mit ihren rund 60 Hz, vorgegeben durch die Aufruffrequenz der *simMessage*-Funktion (etwa alle 16.6 ms) durch die Simulationsumgebung, soll bei der verlustfreien und nicht verzögerten Kommunikation Einfluss auf die Ausführung des Algorithmus haben.

Als verbindungsloses Protokoll, ohne großen Overhead durch informationsbeladene Header, eignet sich das User Datagram Protocol (UDP) besser als das Transmission Control Protocol (TCP). [13, S. 35] Der Umfang der Sensor- und Steuerungsdaten, die bei der Kommunikation übertragen werden müssen, ist so gering, dass die zusätzlichen Headerdaten des TCP eine unnötige Vergrößerung der zu übertragenden Datenpakete bedeuten würden. Auch sind die Checks und der erneute Versand von Nachrichten im Fall von stark verzögerten oder verlorenen Nachrichten nicht erwünscht. Diese würden zu zusätzlichen Netzwerklasten und damit Verzögerungen führen und so die Parameter des Experimentes ändern. [13, S. 179] [3, S. 61] Das entwickelte Programm zur Netzwerksimulation nutzt daher UDP als Transportprotokoll. Jeder Kommunikationskanal, also Roboter $\rightarrow$ Netzwerk $\rightarrow$ Controller und Controller $\rightarrow$ Netzwerk $\rightarrow$ Roboter enthält eine Warteschlange, in der eingehende Nachrichten gespeichert und nach dem First In First Out (FIFO) Prinzip, je nach gewünschten Netzwerkparametern weitergesendet werden. Wenn das Programm ohne Parameter gestartet wird, entspricht dies einer optimalen, verzögerungs- und verlustfreien Weiterleitung von Nachrichten im Netzwerk. In diesem Fall werden der Paketverlust, die Latenz und die durch die Bandbreite entstehende Verzögerung im simulierten Netzwerk auf Null gesetzt, so dass eingehende Nachrichten umgehend weitergeleitet werden. Alternativ können drei Parameter übergeben werden, die für die Bandbreite, die Latenz und die Paketverlustrate stehen. Diese werden in einem Struct gespeichert und von der Sendefunktion genutzt.

Anzumerken ist, dass die Bandbreite hier aus Gründen der Simplizität nicht in $[\frac{bit}{s}]$, sondern in Nachrichten pro Sekunde $[\frac{msg}{s}]$ umgesetzt wurde. Die Implementierung der Latenz erfolgte als One-Way-Delay (OWD), damit diese (hier gewollte) Verzögerung nicht durch die Rechenzeit des Algorithmus oder die Abtastrate des Sensors beeinflusst wird. [13, S. 254]

Nach dem Aufbau und der Bestätigung der Verbindung zum Plugin bzw. dem Algorithmus werden für jeden Kommunikationskanal jeweils ein Sende- und ein Empfangsthread gestartet. Als Übergabeparameter erhalten die Empfangsthreads einen Zeiger auf die Sendewarteschlange des jeweils anderen Kanals, sowie ein Mutex zum Schutz der Warteschlange vor parallelen Zugriffen. Im Thread selbst wird mittels Polling auf dem jeweiligen Port auf eingehende Nachrichten gewartet und diese ohne weitere Bearbeitung in der Sendewarteschlange eingereiht.

In den Sendethreads, muss daraufhin entschieden werden, wie mit den vorliegenden Nachrichten, entsprechend der übergebenen Netzwerkparameter, verfahren werden soll. Dies geschieht in einer Schleife, in der wiederholt der Zeitstempel aus dem Header der Nachricht ausgelesen und von der aktuellen Zeit abgezogen wird, so dass sich daraus die bisherige Laufzeit ergibt. Entspricht diese Laufzeit bereits der für das Experiment gewünschten Netzwerklatenz, so wird die Nachricht weitergeleitet.

Die Differenz der aktuellen Zeit und dem zwischengespeicherten Zeitpunkt des letzten Nachrichtenversands wird mit der für die Bandbreite errechneten Verzögerung verglichen, so dass immer nur maximal so viele Nachrichten pro Sekunde verschickt werden, wie im aktuellen Experimentdurchlauf gewollt sind.

Um Paketverluste zu simulieren wird abschließend anhand einer Zufallszahl entschieden, ob das Datenpaket weitergeleitet oder ganz verworfen werden soll.

Der Ablauf ist im Sequenzdiagramm in Abbildung 3.5 stark vereinfacht dargestellt und soll veranschaulichen, wie die Kommunikation zwischen Simulation/Plugin und dem Algorithmus über das Netzwerkprogramm funktioniert.

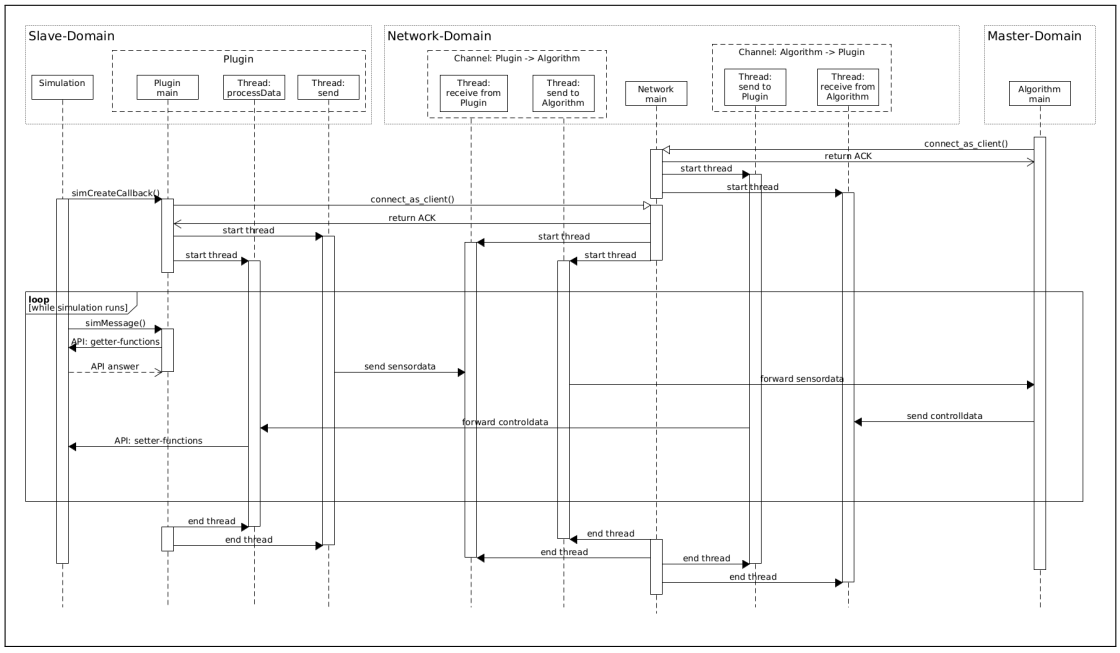

Abbildung 3.5: Ablauf der Kommunikation (groß im Anhang A.2)

3.2.3 *CoppeliaSim* - Szene

Die simulierte Szene in der Robotersimulationsumgebung *CoppeliaSim*, die in Abbildung 3.6 dargestellt ist, besteht aus einem Roboterarm, welcher auf einem Sockel befestigt ist und an dessen Magnetgreifer der einzupassende Zylinder gehalten wird. Innerhalb der Bewegungsreichweite dieses Roboters befindet sich eine Platte mit einem Loch, nur wenig größer als der Durchmesser des Zylinders, in das dieser eingepasst werden soll.

Als Roboterarm wurde in dieser Arbeit das Model eines *KUKA LBR iiwa 14 R820* verwendet, welches bereits in der Simulationsumgebung vorhanden ist. Dieser Roboter ist mit seinen drei Hauptgelenken auf allen 6 Freiheitsgraden (6 Degrees of Freedom (DoF)) beweglich und verfügt über einen Kraft-/Momentensensor, der in seinem Greifer integriert ist.

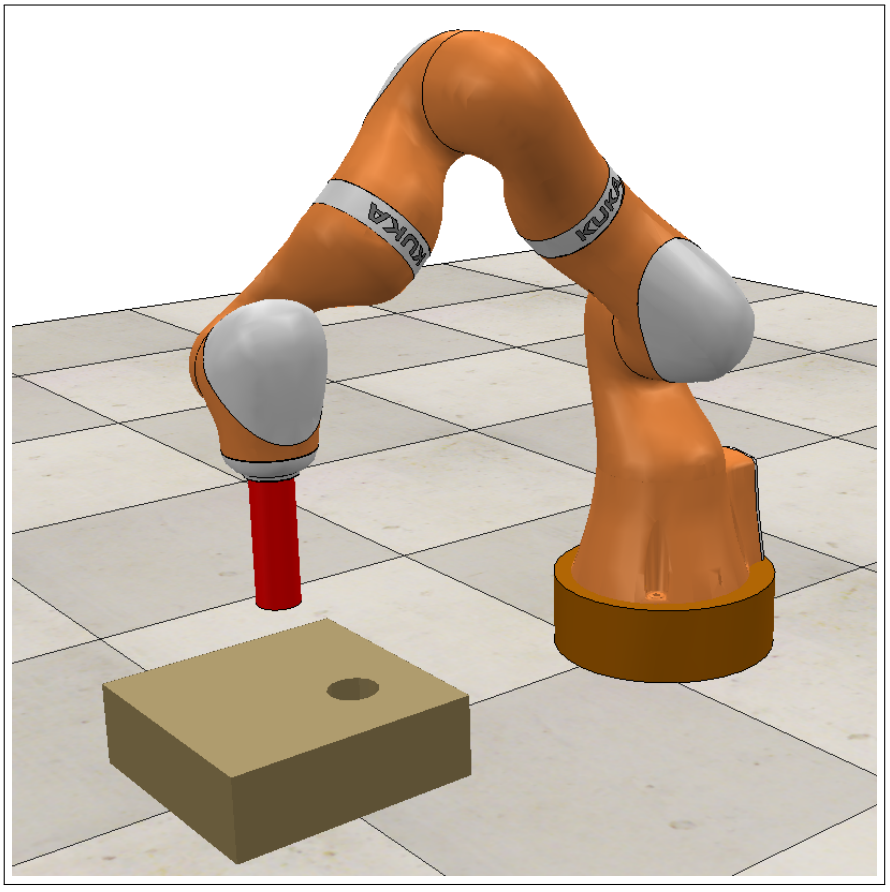

Abbildung 3.6: Die *CoppeliaSim* Szene mit dem Roboterarm und der Platte. Der Zylinder in Rot muss in das Loch eingepasst werden.

Die Bewegungsart des Roboters wurde im Rahmen dieser Arbeit als eine Punkt-zu-Punkt-Steuerung (PTP) umgesetzt. Bei dieser Steuerungsart werden dem Roboter keine Geschwindigkeiten oder Beschleunigungen vorgegeben, sondern die diskreten Positions- und Winkelwerte der Koordinatenachsen des Punktes an dem die Bewegung enden soll. [24, S. 91] Konkret heißt das, dass der Roboter nach jeder Kraft-/Momentenmessung eine Nachricht

mit den Mess- sowie den Positions- und Orientierungsdaten des Messpunktes an den Algorithmus sendet. Von diesem erhält er anschließend neue Positions- und Orientierungswerte eines Punktes zu dem die Steuersphäre bewegt werden soll. Die Schrittweite und Änderungswinkel sind so (klein) gewählt, dass sie eine optisch flüssige Bewegung erzeugen. Diese Werte können in der *initializers.cpp* des Algorithmusprogramms geändert werden.

Die Platte, mit dem zu findenden Loch, wurde mit der 3D-CAD Software *Solid Edge Student Edition* erstellt. Dies war nötig, da die Robotersimulationsumgebung *CoppeliaSim* Volumina, wie Würfel, Quader oder Zylinder, mittels Dreiecksnetzen beschreibt, aber über keine einfache Möglichkeit verfügt, Löcher in diese Volumen einzufügen. Mit *Solid Edge* können erstellte Körper ebenfalls mit Dreiecksnetzen beschrieben werden, so dass eine hier erstellte Platte mit einem nicht-zentrierten Loch ohne Schwierigkeiten in *CoppeliaSim* importiert werden konnte.

3.2.4 Plugin

Das Plugin enthält Funktionen, die seitens der Simulationsumgebung aufgerufen werden. Es ist zwingend notwendig, dass jedes *CoppeliaSim* Plugin mindestens die drei Funktionen:

- simStart()
- simEnd()
- simMessage()

enthält, um korrekt arbeiten zu können. Darüber hinaus ist es möglich eigene Funktionen zu implementieren, die von allen Simulation(en), die das entsprechende Plugin nutzen, mittels sogenannter *Childscripts* aufgerufen werden können. [25]

Die simStart()-Funktion wird einmalig beim Starten der Simulationsumgebung aufgerufen und dient zur Registrierung der vorab erwähnten eigenen Funktionen, sowie der Initialisierung der CoppeliaSim-Bibliothek. Diese enthält unter anderem API-Funktionen, mit denen Objekte (z.B. Kraftsensor) in der Simulation manipuliert werden können. Außerdem sollte hier die Kompatibilitätsüberprüfung der verwendeteten CoppeliaSim-Version stattfinden.

Die simEnd()-Funktion wird beim Beenden der Simulationsumgebung einmal aufgerufen. Spätestens hier sollte jeglicher allozierter Speicher freigegeben und laufende Threads beendet werden.

Um Parameter und Funktionalitäten bei jeder neuen Simulation (z.B. mit neuen Netzwerkeinstellungen) zu reinitialisieren, ist als eine spezialisierte Funktion die *createCallbackFunction()* implementiert worden. Diese Funktion wird bei jedem Start der Simulation durch das für den Roboterarm implementierte *Childscript* aufgerufen. Es werden darin wichtige Variablen und Objekte zurückgesetzt und Referenzen (Handles) von Objekten aus der CoppeliaSim-Szene erfasst. Da das Netzwerkprogramm nach jedem Durchlauf mit neuen Parametern gestartet wird, ist diese Funktion außerdem dafür vorgesehen, die Kommunikation erneut aufzubauen und die Initialisierungsnachricht mit den Basisdaten der Szene (Platten- und Zylindermaße) an den Algorithmus zu senden. Kurz vor dem Verlassen der Funktion werden der Sende- sowie der Nachrichtenverarbeitungsthread gestartet und der Zeitpunkt des Simulationsbeginns gespeichert, da direkt im Anschluss die erste Sensorabfrage stattfindet und der Algorithmus beginnt.

Die *simMessage*-Funktion wird während der Laufzeit der Simulationsumgebung wiederholt aufgerufen. Um die Aufruffrequenz dieser Funktion zu maximieren und so stabil wie möglich zu halten, werden hier ausschließlich die Daten des Kraft-/Momentensensors sowie die Positions- und Orientierungsdaten der Steuersphäre mittels API-Funktionen abgefragt und in die Sende-Warteschlange eingefügt.

Der Sendevorgang selbst wurde in einen extra Thread ausgelagert, da eine Nachricht vor dem Versand um einen Header erweitert wird, der einen Zeitstempel, die Nachrichten-Id sowie die Länge der Nachricht enthält. Obwohl dieser Vorgang insgesamt wenig Zeit kostet, beeinflusst er dennoch die Aufruffrequenz der *simMessage*-Funktion und damit die Abtastrate des Sensors.

Auch die Verarbeitung von Nachrichten wurde in einem extra Thread implementiert. Durch diesen Programmteil wird die Empfangsfunktion aufgerufen, die auf dem eingestellten Port auf eingehende Nachrichten wartet. Nach dem Empfang einer Nachricht wird ihr Header ausgelesen und anhand der enthaltenen Nachrichtenlänge überprüft, ob es sich um eine Nachricht mit neuen Steuerungsdaten handelt. Zu den Steuerungsdaten zählen sowohl neue Positions- und Orientierungswerte für die Sphäre, als auch ein Flag, welches den Bruch des Kraft-/Momentensensors initiiert und damit das Lösen des Magnetgreifers simuliert.

Stimmt die im Header enthaltene Nachrichtenlänge mit der einer Steuerungsdatennachricht überein, werden die enthaltenen Positions- und Orientierungsdaten mit den zwischengespeicherten Werten des vorhergehenden Empfangsdurchlaufs verglichen. Sind diese identisch, werden sie ohne weitere Verarbeitung ignoriert, wodurch unnötige Funktionsaufrufe vermieden werden können. Sind die Positions- und/oder Orientierungsdaten neu, werden die API-Funktionen zum Setzen der Werte aufgerufen und die Sphäre in der Simulationsumgebung bewegt sich zur entsprechenden Postion. Darüber hinaus wird anhand eines Flags überprüft, ob der Magnetgreifer gelöst werden soll und dies gegebenenfalls initiiert. Dadurch ist das Ende des Algorithmus markiert und der Roboterarm bewegt sich zurück in die Initialposition.

In Kapitel 4 wird erläutert, wie die Experimente mit diesen drei Komponenten durchgeführt werden.

KAPITEL 4

Experimente

Die in Kapitel 3 vorgestellten Komponenten werden verwendet, um Experimente durchzuführen mit denen der Einfluss verschiedener Netzwerkqualitäten auf den Regelungsalgorithmus eines NCS ermittelt werden soll. In diesem Kapitel werden der Aufbau und die Durchführung der einzelnen Experimente und gegebenenfalls vorgenommene Anpassungen erklärt. Anschließend werden die Ergebnisse präsentiert und Auszüge der aufgenommenen Messwerte dargestellt. Die vollständigen Messwerte befinden sich in dem, zur Bachelorabeit gehörenden SVN [1] sowie auf der beiliegenden CD-ROM.

4.1 Allgemeiner Aufbau

Der Experimentaufbau besteht aus drei Komponenten. Zur Steuerung, also als Controller im NCS, dient das Programm, in dem der Algorithmus zur Lösung der „Peg in Hole“-Aufgabe abläuft. Der Controller ist eventgesteuert, das heißt, der Empfang von Sensordaten des Kraft-/Momentensensors sowie der Positions- und Orientierungsdaten der Sphäre löst den Berechnungsvorgang neuer Steuerungsdaten aus. Das Netzwerk wird mit dem Routingprogramm simuliert und kann mit variablen Einstellungen für Netzwerkbandbreite, Paketverlustrate und Netzwerklatenz gestartet werden. Als gesteuertes System wird der Roboter in der Szene der Robotersimulationsumgebung *CoppeliaSim* genutzt. Dieser wird durch das Plugin gesteuert. Die Sensordaten- und Positionserfassung erfolgt dabei zeitgesteuert innerhalb der *simMessage*-Funktion des Plugins. Hier werden API-Aufrufe getätigt, die das Auslesen der entsprechenden Objekte innerhalb der Simulationsumgebung ermöglichen. Das Umsetzen der vom Controller erhaltenen Steuerungsdaten geschieht eventgesteuert ebenfalls im Plugin, sobald neue Daten eintreffen. Alle Komponenten laufen auf demselben in Tabelle 4.1 dargestellten System.

[1] `https://comsyssvn.ivs.cs.ovgu.de/repos/students/BSc/2019-Popp-Martin/`

Hardware	Intel® Core™ i5-2520M
Betriebssystem	Ubuntu 18.04.5 LTS, 4.15.0-122-generic (64-bit)
Compilerflags	-std=c++11 -pthread -O3

Tabelle 4.1: Systeminformationen

4.2 Ergebnisse

In diesem Teil werden die Ergebnisse der Experimente deskriptiv vorgestellt. Eine Auswertung und Diskussion findet in Kapitel 5 statt.

4.2.1 Abtastrate und Berechnungszeit

Um die Ergebnisse der Experimente richtig bewerten zu können soll zunächst die Frequenz der zeitgesteuerten Komponente sowie die Rechendauer des Algorithmus ermittelt werden. Die Berechnungszeiten des Algorithmus sollen erwartungsgemäß geringer sein als die zeitgesteuerte Abtastrate des Sensors, so dass er keine Verzögerungen in die Regelschleife einbringt.

Abtastrate des F/T-Sensors

Die Aufruffrequenz der *simMessage*-Funktion ist vorgegeben durch Interna der Simulationsumgebung *CoppeliaSim* und wurde nicht geändert. Da die vorgegebene Frequenz unbekannt ist, muss diese zunächst ermittelt werden. Hierfür werden in zehn Durchläufen des Experimentes die Intervalle zwischen zwei Aufrufen der *simMessage*-Funktion aufgenommen. Die Frequenz f berechnet sich aus einem Intervall T mit der Formel aus [26, S. 1]:

$$f = \frac{1}{T}. \tag{4.1}$$

Ohne Beeinträchtigungen des Netzwerkes, also ohne Verzögerungen und Paketverluste, wird die Funktion in einem Durchgang etwa 1280 Mal aufgerufen. Bei den zehn Durchläufen ergeben sich so $n = 12780$ Intervalle.

Daraus ergibt sich der Mittelwert $\overline{x}$, wobei x_i einen Wert der Frequenzmesswerte darstellt nach [27, S. 33]:

$$\overline{x} = \frac{1}{n} \sum_{i=0}^{n} x_i = 60,274\,Hz \tag{4.2}$$

mit einer Standardabweichung s nach [27, S. 43]

$$s = \sqrt{\frac{1}{n-1} \sum_{i=0}^{n} (x_i - \overline{x})^2} = 3,507\,Hz. \tag{4.3}$$

Das Konfidenzintervall errechnet sich mit einem Vertrauensniveau von 95% und einem STUDENT-Faktor $t_s = 1,96$ aus der Tabelle [27, S. 156] zu

$$\Delta\overline{x} = t \cdot \frac{s}{\sqrt{n}} = 0,060804\,Hz \tag{4.4}$$

nach der Formel aus [27, S. 126].

Das Histogramm in Abbildung 4.1 zeigt die relative Häufigkeit der gemessenen Abtastraten. Hier ist zu erkennen, dass nur wenige Werte den Mittelwert treffen, da die Frequenz schwankt. Die meisten gemessenen Intervalle liegen in einem Frequenzbereich zwischen $59 - 63\,Hz$, was bei den späteren Auswertungen berücksichtigt werden muss.

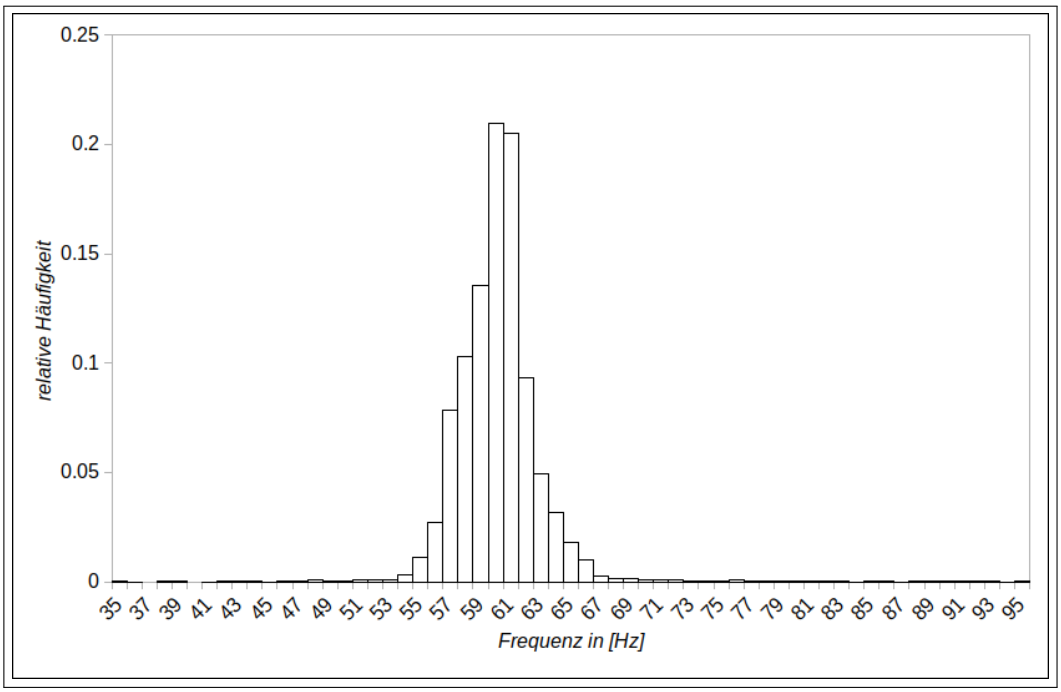

Abbildung 4.1: Das Histogramm zeigt die relative Häufigkeit der Abtastraten des F/T-Sensors.

Rechendauer des Controllers

In einem weiteren Experiment wird der Einfluss des Algorithmus bestimmt. Das Controllerprogramm arbeitet eventgesteuert, das heißt Berechnungen und die Rückgabe von Steuerungsdaten an den Roboter finden erst statt, sobald Sensor- und Koordinatendaten vom Roboter bzw. vom Plugin empfangen werden. Daher ist die Ausführungshäufigkeit sowohl abhängig von der Abtastrate des Sensors, als auch von der Netzwerkqualität. Die zurückgesendeten Steuerungsdaten lösen im Plugin das Setzen der neuen Positions- und Orientierungswerte aus. Eine Verzögerung, durch die Berechnungsdauer des Controllers, kann demnach auch den Gesamtablauf beeinflussen und muss ermittelt werden.

Dafür werden die Zeitabstände zwischen dem Eingang und dem Versenden einer Nachricht im Controllerprogramm während eines vollständigen Durchlaufs gemessen. Das Ergebnis ist in Abbildung 4.2 zu sehen und zeigt anschaulich, dass die Rechendauer durchschnittlich kleiner als $4\,\mu s$ ist. Die drei Spitzen der Kurve konnten durch Beobachtung der Bewegungsabläufe des Roboters und im Programmcode identifiziert werden.
Nach dem Empfang der ersten Sensordaten wird die Startposition der Suche, anhand der übermittelten Daten aus der Initialisierungsnachricht festgelegt und berechnet. Dafür wird

eine Rechenzeit von $\approx 25\,\mu s$ ermittelt. Das zweite Maximum, mit etwa $50\,\mu s$, stellt den Zeitpunkt der Berechnung der sechs Punkte des „Sternmusters“ dar. Dieses wird für die lokale Suche zur präzisen Bestimmung des Lochmittelpunktes benötigt. Das letzte und größte der drei Maxima zeigt sich nach der Beendigung der lokalen Suche. Nach Speicherung der sechs Punkte auf der Lochkante, werden mehrere Dreiecke sowie die Mittelpunkte ihrer Umkreise errechnet. Darüber hinaus werden im selben Schritt die größten Ausreißer der berechneten Mittelpunkte entfernt und der Mittelwert aus den restlichen Werten gebildet. Diese, im Vergleich zu den sonstigen Berechnungen, zahlreichen und aufwändigen Kalkulationen dauern ungefähr $115\,\mu s$. In Zusammenschau dieser Ergebnisse sind die längsten Berechnungsintervalle dennoch vernachlässigbar klein, da sie alle in deutlich unter $1\,ms$ beendet werden.

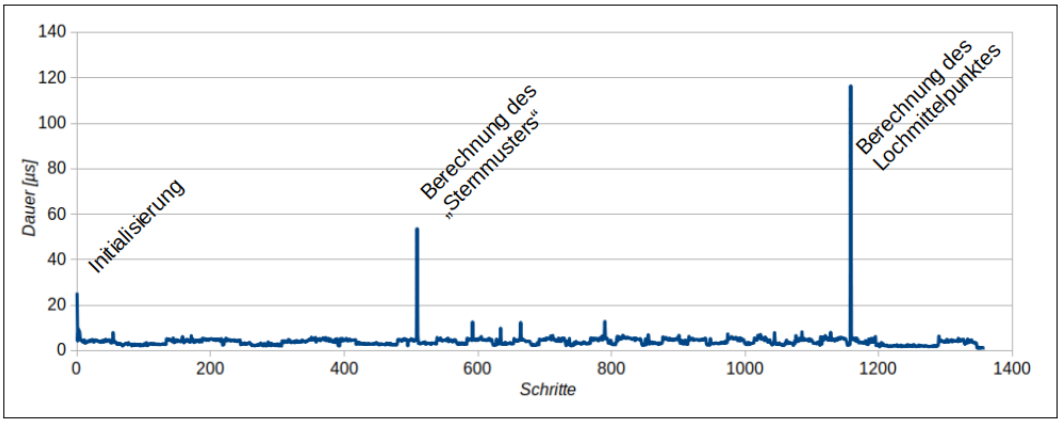

Abbildung 4.2: Das Diagramm zeigt die Berechnungsdauer der einzelnen Schritte des Algorithmus.

4.2.2 Kräfte

Im Zuge der Untersuchungen sollen auch starke Kraftanstiege, die durch den Einfluss verschiedener Dienstgüten auftreten könnten, identifiziert werden.

Maximalkräfte

Zur Erfassung der Maximalkräfte können innerhalb jedes Experimentdurchlaufs, bei allen Sensorabtastungen die auftretenden Kräfte aufgenommen und der größte dieser Werte aus den verschiedenen Durchgängen ermittelt werden.
In den Messreihen ohne Einfluss von Verzögerungen oder Paketverlusten wird regelmäßig eine Basiskraft von rund $38\,N$ ermittelt. Abbildung 4.3 zeigt die Verteilungen aller Kräfte bei den unterschiedlichen Experimenten anhand von Boxplots. Es wird deutlich, dass die meisten Kraftmesswerte bei verschiedenen Bandbreiten sehr nah beieinander in der Nähe des Minimalwertes von $F_{min} = 37,596\,N$ liegen. Dennoch weist die Standardabweichung von $3,378\,N$ darauf hin, dass es starke Abweichungen gibt, was sich durch die im Boxplot der Abbildung 4.3 erkennbaren Ausreißer, zu denen auch der Maximalwert gehört, widerspiegelt. Die Steigerung der Basiskraft ($38\,N$) auf die Maximalkraft von $52,087\,N$ beträgt ca. $37\,\%$.

	n	min[N]	q1[N]	median[N]	q3[N]	max[N]	mean[N]	sd[N]
Bandbreite	100	37,596	37,796	37,821	38,134	52,087	39,171	3,378
Latenz	194	37,616	43,004	45,280	48,150	52,700	45,328	3,630
Paketverlust	80	36,368	38,515	40,559	42,240	46,136	40,656	2,378

Tabelle 4.2: Zusammenfassung der Daten der Boxplots.

Wie in Tabelle 4.2 zu erkennen, ist die Standardabweichung bei den Latenz-Experimenten mit $3,630\,N$ ebenfalls hoch. Auffällig ist jedoch, dass der Minimalwert ($F_{min} = 37,616\,N$), der Median ($F_{med} = 45,28\,N$) und der Maximalwert ($F_{max} = 52,7\,N$) erheblich weiter gestreut sind, als die Messwerte der Bandbreitenmessreihe. Grafisch dargestellt ist dies im Boxplot Abbildung 4.3. Der Maximalwert bei den Latenz-Experimenten ist gleichzeitig der höchste aller Kraftmesswerte und stellt eine Steigerung vom Basiswert ($38\,N$) um $39\,\%$ dar.

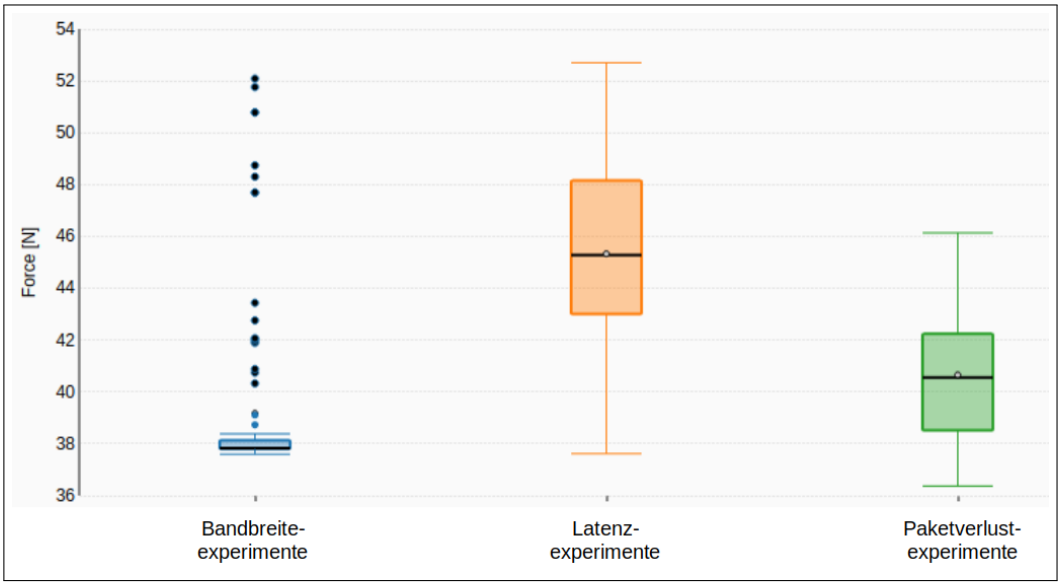

Abbildung 4.3: Der Boxplot zeigt die Verteilung aller gemessenen Maximalkräfte in den drei Messreihen.

Die Werte bei den Paketverlust-Experimenten sind ähnlich verteilt, wie die Werte der Latenzmessungen, jedoch ist die Standardabweichung mit $2.378\,N$ (vgl. Tabelle 4.2), ebenso wie der Median und der Maximalwert mit $F_{med} = 40,559\,N$ und $F_{max} = 46,136\,N$ geringer. Gegenüber der Basiskraft ($38\,N$) stellt der Maximalwert eine Steigerung von $21\,\%$ dar.

Kraftverläufe

Schon während der ersten Testdurchläufe fiel auf, dass die Kraftkurven nicht gleichmäßig verlaufen. Daher wird, abgesehen von der Maximalkraft, auch jeweils der Kraftverlauf eines vollständigen Experiments ohne und mit verzögernden Netzwerkparametern durchgeführt.

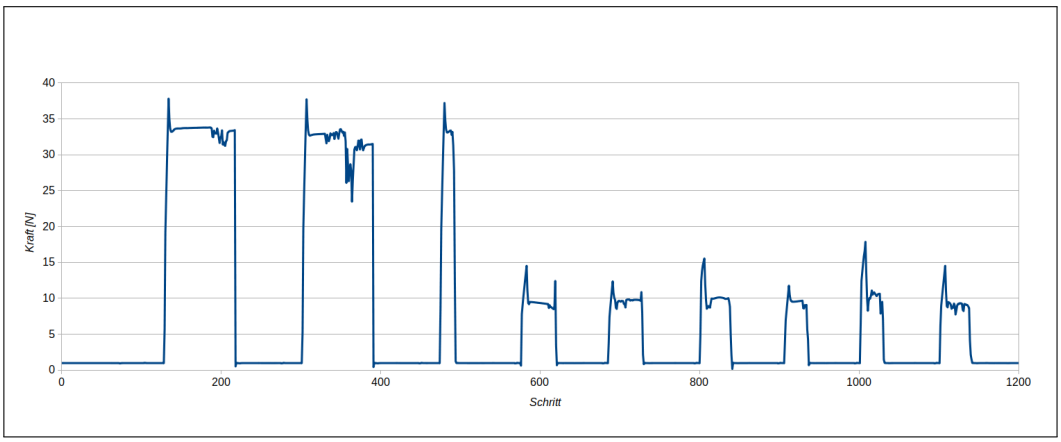

Abbildung 4.4: Das Diagramm zeigt die Kraftkurve eines unbeeinflussten Durchlaufs.

Die erste Kurve in Abbildung 4.4 zeigt den Verlauf der resultierenden Kräfte aus einem vollständigen Durchlauf. Während dieses Versuches ist das Netzwerk vollständig verlust- und verzögerungsfrei. Deutlich zu erkennen sind die einzelnen Algorithmenphasen. Bis zu Schritt ≈ 120 ist die Kraft bei $\approx 1\,N$, da nur das Gewicht des Zylinders vom Kraftsensor registriert wird. Der Zylinder wiegt $100\,g$, woduch sich mit der Erdbeschleunigung $g = 9,81\,\frac{kg \cdot m}{s^2}$ nach

$$F = m \cdot g$$

eine Kraft von $0,981\,N$ ergibt. Anschließend, im Moment des Aufsetzens, steigt die Kraft kurzeitig auf $\approx 38\,N$ an und wird dann auf $\approx 34\,N$ heruntergeregelt. Daraufhin wird der Zylinder über die Platte gezogen. Am Ende dieser ersten Bewegung, mit aufgesetztem Zylinder, sind mehrere leichte Schwankungen der Kraftkurve zu erkennen. Anschließend wird der Zylinder für den Bahnwechsel abgehoben, so dass für etwa 80 Schritte erneut die alleinige Zylinderkraft gemessen wird. Während der zweiten Kontaktfahrt treten derartige Schwankungen an mehreren Stellen auf. Jedoch sei erwähnt, dass der Zylinder auf dieser Bahn das Loch bereits randständig überfährt, so dass hier ein etwas größerer Abfall der Kraft ersichtlich ist. Auf der dritten Bahn, dargestellt durch den dritten Ausschlag der Kraftkurve, wird das Loch erfasst und der Zylinder abgehoben, was sich in einem erneuten Kraftabfall präsentiert. Daraufhin wird die nächste Phase eingeleitet. Anhand der folgenden sechs kleineren Erhöhungen mit einer Kraft von durchschnittlich $10\,N$ ist die lokale Suche mit der „Sternfahrt“ zu erkennen. (vgl. Abschnitt 3.2.1)

In der Kurve der Abbildung 4.5 ist der Verlauf der Kraft bei einer Latenz von $16\,ms$ dargestellt. Es ist klar zu konstatieren, dass der grobe Verlauf der Kraftkurve, derjenigen des unbeeinträchtigten Versuches ähnelt. Im Vergleich zu diesem schwanken die Kräfte jedoch deutlich sobald der Zylinder aufgesetzt wird. Auch treten hier höhere Maximalkräfte bis $\approx 46\,N$ bei Schritt 420

Die Kurve in Abbildung 4.6 zeigt den Verlauf der Kraft bei einer Paketverlustrate von $50\,\%$. Zu erkennen ist, dass die Maximalkraft nicht die $40\,N$ Marke übersteigt und die Schwankungen der Kräfte bei aufgesetzem Zylinder niederfrequenter sind, als bei den Latenz-Experimenten (Abb. 4.8).

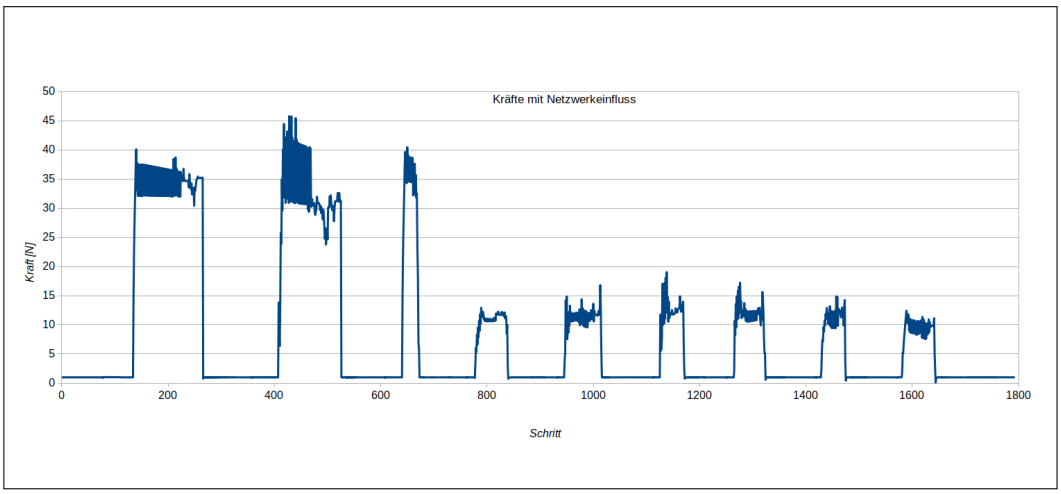

Abbildung 4.5: Das Diagramm zeigt den Kraftverlauf während eines Durchlaufs mit einer Latenz von 16ms.

4.2.3 Bandbreite, Latenz und Paketverlustrate

Die nachstehenden Versuche erfolgen alle mit den gleichen Komponenten sowie Zeit-, beziehungsweise Kraft-Messpunkten.

Die Gesamtzeit, die der Roboter benötigt, um den Zylinder in das Loch auf der Platte einzupassen, soll eine der abhängigen Messgrößen sein. Es ist naheliegend, dass Verzögerungen bei der Übertragung und nicht-eintreffende Sensor- oder Steuerungsdaten diese Zeit beeinflussen. Daher ist es sinnvoll den Einfluss der Netzwerkqualitäten auf die Gesamtzeit zu evaluieren.
Das Intervall zwischen dem ersten *simMessage*-Funktionsaufruf und dem Lösen des Magnetgreifers am Ende des Algorithmus wird als zu messende Zeitspanne gewählt. Dies eignet sich insbesondere deshalb zur Messung der Gesamtzeit, da die genannten Aktionen beide im Roboter bzw. im Plugin umgesetzt werden und der Algorithmusablauf hier eingeleitet und beendet wird.
Bis zehn verwertbare Messwerte erfasst werden können, werden alle Fehlversuche aufsummiert. Als Fehlversuche gelten Durchläufe bei denen das Loch an der falschen Position oder gar nicht identifiziert wird.

Bandbreite

Die Netzwerkbandbreite, ist hier nicht wie allgemein üblich in *Bit pro Sekunde*, sondern, wie in Kapitel 3 erwähnt, in *Nachrichten pro Sekunde* $[\frac{msg}{s}]$ angegeben. Implementiert wurde sie als eine Verzögerung zwischen dem Versand zweier Nachrichten, so dass höchstens m Nachrichten pro Sekunde verschickt werden können, wobei m die gewählte Bandbreite repräsentiert. Die Ausgangsmessungen erfolgen ohne Einschränkungen der Bandbreite. Es wird also jede im Netzwerkprogramm eintreffende Nachricht umgehend an den Empfänger weitergeleitet. Anschließend wird die Bandbreite schrittweise verringert und dabei die Messwerte aufgenommen. Da sich diese erwartungsgemäß bis zu Bandbreiten in der Nähe

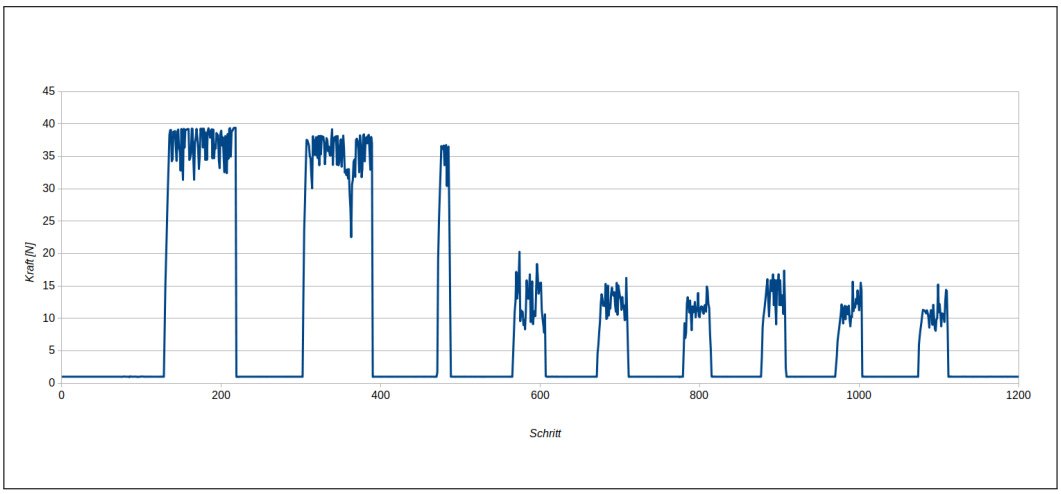

Abbildung 4.6: Das Diagramm zeigt den Kraftverlauf während eines Durchlaufs mit einer Paketverlustrate von 50 %.

der Sensor-Abtastrate nicht signifikant ändern, können die Schritte zunächst in größeren Abständen und später kleiner gewählt werden.

Im Folgenden sind die Ergebnisse der Messungen, der Aufgabenlösungszeiten unter dem Einfluss verschiedener Netzwerkbandbreiten aufgeführt.

bandwidth $[msg/s]$	mean runtime $[s]$	SD $[s]$	missed -
∞	21,285 ± 0,005	0,009	0
75	21,294 ± 0,016	0,027	1
65	21,359 ± 0,029	0,050	0
63	21,495 ± 0,029	0,051	0
61	21,866 ± 0,093	0,163	0
60	58,245 ± 4,240	7,399	3

Tabelle 4.3: Zusammenfassung der Messwerte der durchschnittlich benötigten Zeit zur Beendigung der Aufgabe unter schrittweiser Verringerung der Bandbreite.

Die Tabelle 4.3 zeigt den nach (4.2) berechneten arithmetischen Mittelwert der gemessenen Zeiten von jeweils zehn erfolgreichen Durchläufen je Bandbreite sowie deren Standardabweichung nach (4.3) und das 95%ige Konfidenzintervall nach (4.4). Die Standardabweichung der Messungen ist bei einer Bandbreite von $\approx 60\,\frac{msg}{s}$ mit $7,3\,s$ am größten. Bei allen anderen Bandbreiten kann eine Standardabweichung von unter $1\,s$ erreicht werden. Die Aufgabe wird, bis auf einen Fehlversuch bei $75\,\frac{msg}{s}$, im gesamten Messbereich erfolgreich abgeschlossen. Erst bei der Grenzbandbreite von $60\,\frac{msg}{s}$ kommt es zu drei Fehlversuchen bis zehn

erfolgreiche Durchläufe aufgenommen werden können.

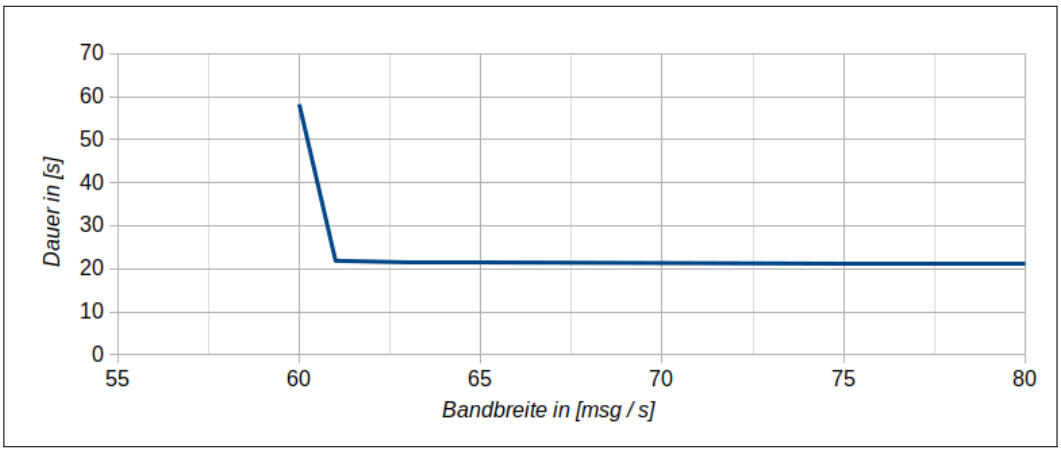

Abbildung 4.7: Die durchschnittliche benötigte Gesamtdauer der Aufgabe in Abhängigkeit von der Netzwerkbandbreite.

Entsprechend dem Versuchsablauf ist das Diagramm 4.7 von rechts nach links zu lesen. Zunächst erfolgen die Versuche mit großen Bandbreiten, welche anschließend nach und nach verringert werden. Wie im Diagramm zu erkennen ist, können bei Netzwerkbandbreiten von über 60 Nachrichten pro Sekunde durchschnittlich etwa 21 s für die Beendigung der Aufgabe gemessen werden. Bei einer Bandbreite von $\approx 60\,\frac{msg}{s}$ beträgt die durchschnittliche Ausführungsdauer etwa 58 s. Bezogen auf das Ausführungsintervall mit einer nicht begrenzten Bandbreite ist das ein Anstieg von über 270 %. Weitere Ergebnisse mit einer noch geringeren Bandbreite können nicht ermittelt werden, da sich die Bewegungen des Roboterarmes direkt nach dem Start der Simulation stark verlangsamen und dann ganz stoppen.

Latenz

Die Netzwerklatenz, als die Zeit zwischen dem Senden und Empfangen eines Datenpaketes, wird im Folgenden in ms angegeben. Implementiert wird sie als eine Verzögerung, die anhand des Zeitstempels einer jeden Nachricht bestimmt wird. Für jede Latenz von 0 bis 40 ms werden je zehn Messwerte aufgenommen. Die Messwerte über 44 ms sind Einzelmessungen. Die Versuchsreihe wird ohne eine Einschränkung durch Netzwerklatenzen begonnen. Jede im Netzwerk eintreffende Nachricht wird also direkt weitergeleitet. Im weiteren Verlauf wird die Latenz langsam erhöht. Erwartungsgemäß zeigen sich dabei einige interessante Messbereiche, die daraufhin engmaschiger untersucht werden.

In Tabelle 4.4 sind die nach (4.2) errechneten arithmetischen Mittelwerte einschließlich ihrer 95%igen Konfidenzintervalle nach (4.4) der benötigten Gesamtzeit zur Beendigung der „Peg in Hole“-Aufgabe angegeben. Eine Stichprobe besteht dabei aus $N = 10$ Messwerten, je eingestellter Netzwerklatenz. Die Standardabweichung, berechnet nach (4.3), ist bei den Messungen ohne den Einfluss der Netzwerklatenz am kleinsten mit $0,03\,s$. Bei einer Erhöhung der Latenz zeigen sich unregelmäßige Schwankungen der Standardabweichung von

lateny [ms]	mean runtime [s]	SD [s]	missed -
0	21,372 ± 0,017	0,030	0
8	28,150 ± 0,188	0,231	0
9	41,228 ± 0,105	0,183	0
16	43,364 ± 0,185	0,323	0
17	60,241 ± 0,279	0,486	1
24	64,012 ± 0,306	0,535	1
25	76,903 ± 0,191	0,333	0
32	84,397 ± 0,248	0,433	0
33	88,394 ± 0,298	0,519	1
40	104,753 ± 0,220	0,384	3

Tabelle 4.4: Zusammenfassung der Messwerte der durchschnittlich benötigten Zeit zur Beendigung der Aufgabe unter schrittweiser Erhöhung der Netzwerklatenz.

$0,183s$ bei $9\,ms$ bis zu $0,535\,s$ bei $24\,ms$ Verzögerung. Dennoch bleiben die Standardabweichungen unter $1\,s$, was zeigt, dass die Messwerte eines Messbereiches nah beieinander liegen. Zu einzelnen Fehlversuchen kommt es bei Latenzen von 17, 24 und $33\,ms$. Bei größeren Verzögerungen kann die Aufgabe häufiger nicht abgeschlossen werden, wobei die häufigsten Fehler bei einer Latenz von $42\,ms$ auftreten.

In Diagramm 4.8 erkennt man den stufenartigen Verlauf der Kurve. Zu Beginn ist der durchschnittliche Zeitraum bis zur Beendigung der Aufgabe $\approx 21s$ und steigt bis zu einer Latenz von $8\,ms$ auf $\approx 28\,s$ an. Bei $9\,ms$ kommt es zu einem größeren Sprung auf $\approx 41\,s$. Der Wert erhöht sich dann bis $16\,ms$ nur leicht auf $\approx 43\,s$ bis es bei $17\,ms$ wieder zu einem größeren Sprung auf $\approx 60\,s$ kommt. Regelmäßig, etwa alle $8\,ms$ Steigerung der Latenz, kommt es zu einem sprunghaften Anstieg der benötigten Zeit. Bis zu einer Latenz von $40\,ms$ werden jeweils zehn Messwerte aufgenommen. Danach werden fünf weitere Einzelmessungen bei 44, 48, 49, 52 und $56\,ms$ vorgenommen und so als längstes Intervall $\approx 146\,s$ bei $56\,ms$ Latenz ermittelt, was bezogen auf die Eingangsmessungen ohne Verzögerungen einem Anstieg um ca. 700 % entspricht.
Die Gleichung der Regressionsgerade errechnet sich nach den Formeln in [28, S. 308] zu

$$y = 2,382399112 \cdot x + 14,18078054 \tag{4.5}$$

mit einem Korrelationskoeffizienten von $r = 0,992048399$.

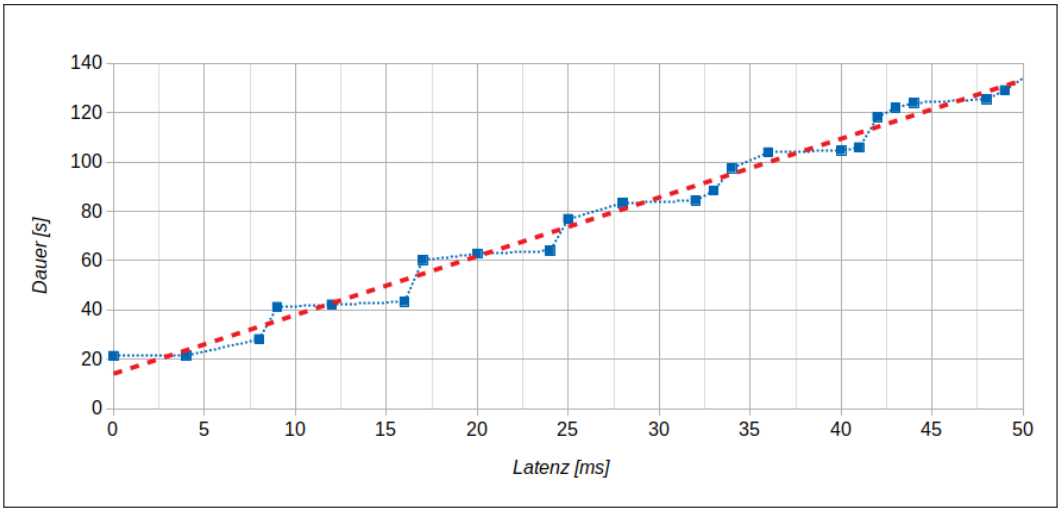

Abbildung 4.8: Die durchschnittlich benötigte Gesamtdauer der Aufgabe in Abhängigkeit von der Netzwerklatenz. In Rot die Regressionsgerade.

Paketverlust

Die Paketverlustrate ist das Verhältnis der beim Sender abgeschickten und der nicht beim Empfänger angekommen Datenpakete. Sie wird angegeben in %. Das Netzwerkprogramm entscheidet anhand einer Zufallszahl, ob eine Nachricht weitergeleitet werden soll oder als Verlust verworfen wird. Begonnen wird die Versuchsreihe mit einer Paketverlustrate von 0 %. Dies bedeutet, dass jede eintreffende Nachricht weitergeleitet wird. Anschließend kann die Paketverlustrate in 10 % Schritten erhöht und dabei jeweils zehn Messungen vorgenommen werden.

packet loss [%]	mean runtime [s]	SD [s]	missed -
0	21,294 ± 0,011	0,018	0
10	26,063 ± 0,187	0,327	0
20	32,913 ± 2,283	0,493	0
30	42,238 ± 0,515	0,899	0
40	57,418 ± 0,769	1,342	1
50	81,450 ± 1,275	2,225	1
60	128,446 ± 2,238	3,906	2
70	222,301 ± 2,695	4,703	2

Tabelle 4.5: Zusammenfassung der Messwerte der durchschnittlich benötigten Zeit zur Beendigung der Aufgabe unter Erhöhung der Paketverlustrate.

Die Tabelle 4.5 zeigt den arithmetischen Mittelwert nach (4.2) der Gesamtdauer der 10 Messungen je Stufe sowie das 95 %ige Konfidenzintervall nach (4.4). Darüber hinaus ist die Standardabweichung nach (4.3) zu erkennen.

Die Standardabweichung ist ohne Paketverluste mit $0,018\,s$ am geringsten und steigt anschließend mit jeder höheren Paketverlustrate an. Bei einer Verlustrate von 70 % aller gesendeten Datenpakete ist die Standardabweichung mit $4,703\,s$ am höchsten. Im abgebildeten Diagramm 4.9 ist dies grafisch dargestellt und zeigt, dass die zum Abschluss der Aufgabe benötigte Zeit nahezu exponentiell, nach der hergeleiteten Formel

$$t \approx t_0 \cdot (1 - \frac{p}{100})^2 \tag{4.6}$$

mit der Gesamtdauer t in $[s]$, der Gesamtdauer ohne Paketverluste $t_0 = 21,3\,s$ und der Paketverlustrate p in [%], ansteigt. Der erste Fehlversuch tritt bei einer 40 % Paketverlusten auf. Bei 60 % und 70 % kommt es zu jeweils zwei Fehlern.

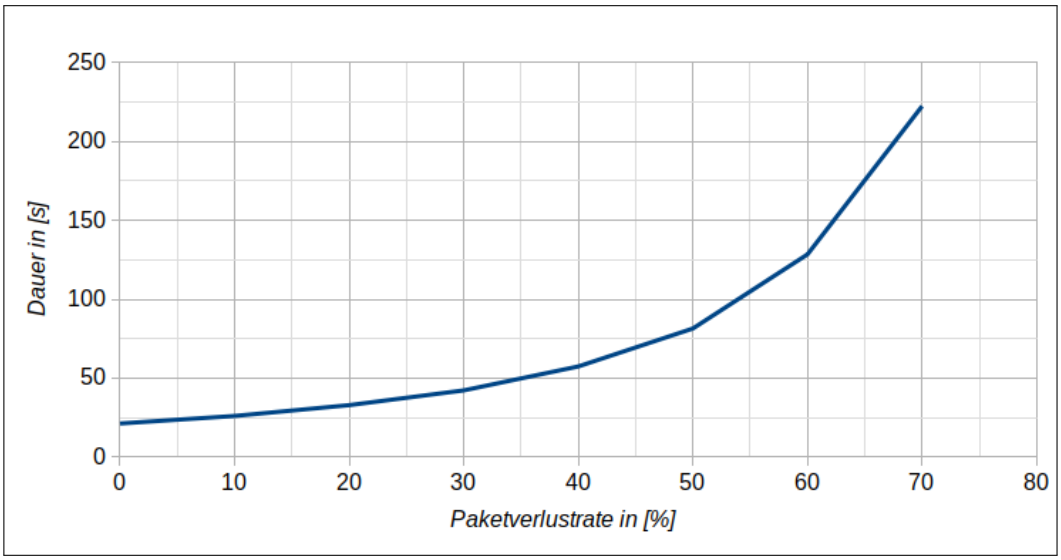

Abbildung 4.9: Die durchschnittliche benötigte Gesamtdauer der Aufgabe in Abhängigkeit von der Paketverlustrate.

KAPITEL 5

Evaluation

Die in Kapitel 4 durchgeführten Experimente werden hier hinsichtlich ihrer Ergebnisse diskutiert und eingeordnet.

5.1 Bewertung der Abtastrate und Berechnungszeit

Abtastrate

In Abschnitt 4.2.1 kann eine, von der Simulationsumgebung *CoppeliaSim* vorgegebene Abtastfrequenz f des Sensors, von durchschnittlich $60.274 \pm 0.061\ Hz$ ermittelt werden.

Dies entspricht nach

$$T = \frac{1}{f} \tag{5.1}$$

aus [26, S. 1] einer Verzögerung T zwischen zwei Abtastungen von $16.59 \pm 0.061\ ms$. Wie im Histogramm 4.1 ersichtlich und an der großen Standardabweichung zu erkennen, schwankt die tatsächliche Frequenz der Abtastrate zwischen 57 Hz und 64 Hz. Diese Frequenz kann nicht beeinflusst werden. Die vergleichsweise hohen Abweichungen beeinflussen, besonders bei Messungen in Grenzbereichen, die Ergebnisse einiger Experimente.

Berechnungszeit

Die Rechenzeit des Algorithmus wird mit einer durchschnittlichen Dauer von unter 4 μs und dem größten Maximum von 115 μs als vernachlässigbar eingestuft. Da die Gesamtzeit der Aufgabenbeendigung ein mehr-sekündiger Vorgang ist, sind Verzögerungen im μs-Bereich für die Messauswertung unerhebllich. Auch auf den Regelungsvorgang selbst haben diese Verzögerungen keinen Einfluss. Die Abtastfrequenzen bzw. die Verzögerungen zwischen zwei Sensormessungen, geben die maximal mögliche Anzahl von Nachrichten je Sekunde, die versendet und anschließend vom Controller verarbeitet werden müssen, vor. Selbst mit der, in diesen Experimenten aufgenommenen, maximalen Bandbreite von ca. 130 $\frac{msg}{s}$ liegt die Verzögerung zwischen zwei Abtastungen bei 8 ms (vgl. (5.1)) und damit deutlich

über der maximalen Rechenzeit des Algorithmus von 115 μs. Daher haben die wenigen μs-Verzögerungen durch den Algorithmus keinen Einfluss auf das rechtzeitige Eintreffen von Steuerungsdaten innerhalb eines Abtastintervalls und damit auch nicht auf die Qualität der Regelung.

5.2 Bewertung der Kraftmessungen

Bei den Experimenten und in einer gesonderten Messreihe werden sowohl die größten aufgetretenen Kräfte, als auch die Kraftverläufe unter verschiedenen Netzwerkbedingungen bestimmt. Dabei zeigt sich, dass die unterschiedlichen Dienstgüten anderen Einfluss auf das Verhalten der Kräfte haben.

Bei der Punkt-zu-Punkt Steuerung (vgl. 3.2.3) des Roboterarmes entsteht eine flüssige Bewegung durch das regelmäßige Setzen neuer Positions- und Orientierungswerte. Bleiben, bedingt durch Verzögerungen oder Datenverluste, diese Werte aus, stoppt die Bewegung abrupt bis zum Eintreffen neuer Daten. Wiederholt sich diese *stop-and-go*-Bewegung, kann es, je nach Frequenz der Verzögerungen, zu einem ruckartigen Bewegungsablauf kommen. Bei aufgesetztem Zylinder spiegeln sich diese Vibrationen in den Messwerten wider (vgl. Kraftkurven Latenz-Experiment 4.5 und Paketverlust-Experiment 4.6). Diese durch nicht eingetroffene Steuerungsdaten hervorgerufene Problematik, ist insbesondere bei den Messwerten des Paketverlust-Experimentes erkennbar.

In der Kurve des Kraftverlaufes, unter dem Einfluss verschiedener Latenzen, zeigen sich die Ausschläge wesentlich intensiver und hochfrequenter. Ein Erklärungsansatz für die Schwankungen in der Kraftkurve des Latenz-Experimentes ist, dass die Datenpakete nicht verloren gehen, wie es durch Paketverluste der Fall ist, sondern verzögert eintreffen. Durch das Ausbleiben einer Antwort vom Controller innerhalb eines Abtastintervalls werden bei der nächsten Abtastung dieselben Positons- und Orientierungswerte erneut gesandt. Dies kann bis zum Eintreffen einer Antwort vom Controller mehrmals vorkommen, was von diesem durchgehend mit identischen Steuerungsdaten beantwortet wird. Das implementierte Plugin identifiziert die doppelten Nachrichten (vgl. 3.2.4) und ignoriert diese.
Bewegt sich der Roboter jedoch durch den Einfluss von Beschleunigungen vorheriger Bewegungen oder der Erdbeschleunigung, werden bei den Sensorabtastungen minimal abweichende Werte aufgenommen (vgl. Abb. 5.1). Diese werden auch vom Controller mit leicht verschiedenen Steuerungsdaten beantwortet, so dass diese *Doppelnachrichten* nicht mehr vom Plugin aussortiert werden können. Liegen die neuen Koordinaten beispielsweise entgegen der vorherigen Beschleunigungsrichtung, wird der Roboter, wenn auch minimal, neu ausgerichtet. Dadurch bekommt er gleichzeitig eine neue Beschleunigungsrichtung, die Schwingungen und stärkere Vibrationen induziert, als das Ausbleiben von Positionsdaten durch Paketverluste. Diese Vibrationen werden auch auf den Zylinder übertragen, so dass sich die Anpress- und Maximalkräfte ändern. Die Regelung ist ebenfalls betroffen, da die Möglichkeit besteht, dass bestimmte Grenzwerte während der Kraftmessung, die beispielsweise das Überfahren einer Lochkante markieren, falsch oder gar nicht registriert werden.

Die Auswirkungen dieser Erklärungsansätze sind darüber hinaus anschaulich in den divergierenden Boxplots in Abbildung 4.3 dargestellt. Hier kommt die Verteilung der Kraftmesswerte der Bandbreite-Experimente mit einer geringen Verteilung zum Ausdruck. Erst

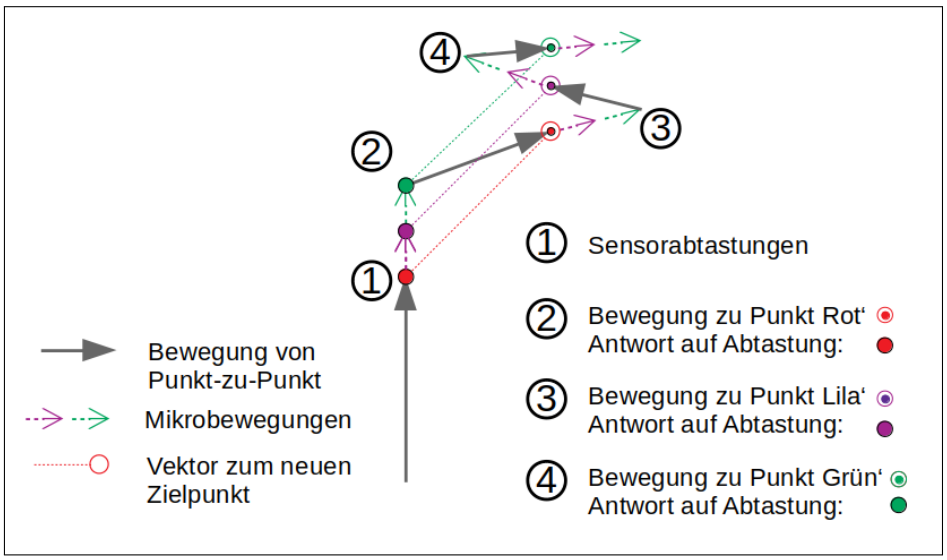

Abbildung 5.1: Darstellung der Bewegungsabläufe durch Mikrobewegungen.
Die Bewegung wird am Zielpunkt ① schwunginduziert weitergeführt (dünne gestrichtelte Pfeile grün/lila). Kommen keine neuen Zielkoordinaten vom Controller, werden weitere Positionsdaten auf den Weg gebracht. Diese lösen nach der Ausführung ② des eigentlichen neuen Bewegungsziels zuätzliche Bewegungen aus ③ u. ④. (eigene Darstellung)

bei sehr geringen Bandbreiten treten die oben beschriebenen Probleme auf, da es beim Ausbleiben von Steuerungsdaten oder dem Eintreffen unerkannter *Doppelnachrichten* zu Vibrationen kommt. Hierüber erklären sich die Ausreißer, wie auch die Maximalkraft von $\approx 52\ N$.
Die Paketverlust-Experimente sind allein von erstgenannter Erklärungstheorie betroffen, da hier verzögerte Nachrichten, also *Doppelnachrichten*, nicht vorkommen. Hingegen sind ausbleibende Steuerungssignale durch Datenverluste für die *stop-and-go*-Bewegung und damit für Vibrationen verantwortlich. Verglichen mit den Bandbreite-Experimenten sind die gemessenen Kräfte daher während aller Durchläufe der Paketverlust-Experimente weiter gestreut.
Bei den Latenz-Experimenten treffen beide Erklärungsansätze zu, da hier einerseits durch das Ausbleiben von korrekten neuen Steuerungsdaten zum richtigen Zeitpunkt die *stop-and-go*-Bewegung induziert wird, zusätzlich aber auch *Doppelnachrichten* gehäuft vorkommen können. Aufgrund der hochfrequenteren Vibrationen kommt es zu noch stärkeren Schwingungen, als bei den Paketverlust-Experimenten und damit zu einer breiteren Spannweite der Kraftmesswerte.
Die höchste gemessene Kraft (Maximalkraft) betrug 52.7 N, was bezogen auf die Basiskräfte aller Versuche von etwa 38 N einem Anstieg von rund 39% entspricht. Zu einem signifikanten Kraftanstieg von über 100% ist es demnach nicht gekommen.

Eine höhere Dämpfung könnte das Schwingungsverhalten minimieren, wodurch sich die Stabilität des Systems erhöhen würde. Dies kann jedoch zu einem Qualitätsverlust der Regelung führen.[29]

5.3 Bewertung der Experimente mit unterschiedlichen Dienstgüten

In diesen Experimenten soll ermittelt werden, wie sich die zur Beendigung der Aufgabe benötigte Gesamtzeit abhängig von der gewählten Netzwerkqualität ändert.

5.3.1 Bewertung der Bandbreite-Experimente

Während der Versuchsreihen mit variablen Bandbreiten können erwartungsgemäß keine großen Abweichungen der Gesamtdauer beobachtet werden. (vgl. Abb. 4.7) Der Roboter ist, bei verschiedenen Bandbreiten, regelmäßig in der Lage die „Peg in Hole“-Aufgabe innerhalb von ca. 21 s zu beenden. Im Netzwerk wird bei einer Bandbreite, die größer ist als die Abtastrate des Sensors, jede eingetroffene Nachricht umgehend weitergeleitet. Auch die Schwankungen der Abtastrate können kompensiert werden, so dass die Ausführungsqualität des Algorithmus nicht beeinträchtigt wird. Dies ist auch an der geringen Standardabweichung dieses Messbereichs zu erkennen. (vgl. Tab. 4.3)

Bei einer Empfangsrate (vorgegeben durch die Abtastrate) von $\lambda \simeq 60\ \frac{msg}{s}$ und einer Netzwerkbandbreite von $\mu = 60\ \frac{msg}{s}$ verlängert sich erwartungsgemäß die Gesamtdauer der Aufgabe. Bei $\mu < 60\ \frac{msg}{s}$ ist eine erfolgreiche Beendigung nicht mehr möglich, da das Netzwerk nicht alle eintreffenden Nachrichten weiterleiten kann und diese sich in der Netzwerkwarteschlange stauen. Dies erklärt sich auch durch den Auslastungsgrad ρ des Netzwerkes, welcher bedingt durch die Warteschlangenarchitektur (vgl. Abschnitt 3.2.2), nach

$$\rho = \frac{\lambda}{\mu} \tag{5.2}$$

aus [14, S. 29] errechnet wird.
Mit $\mu = 60\ \frac{msg}{s}$ und $\lambda \simeq 60\ \frac{msg}{s}$ ergibt sich somit ein Auslastungsgrad von etwa 1. Wie in Abschnitt 5.1 hervorgehoben unterliegen die Abtastraten, jedoch erheblichen Schwankungen. So kann es im Grenzfall $\lambda \approx \mu$ vorkommen, dass der Auslastungsgrad nach (5.2) während der Ausführung teilweise oberhalb und unterhalb von 1 liegt. Nachrichten, die sich im Fall $\lambda > \mu \rightarrow \rho > 1$ in der Warteschlange des Netzwerkprogrammes stauen, werden im Fall von $\lambda < \mu \rightarrow \rho < 1$ wieder abgearbeitet. Bei Bandbreiten unter $60\ \frac{msg}{s}$ tritt dieser zweite Fall erheblich seltener auf, da nur vereinzelte Werte der Abtastrate bis unter 60 Hz schwanken. Dies führt dazu, dass sich die Datenpakete in der Warteschlange exponentiell stauen. Da diese Schwankungen jedoch zufällig und unregelmäßig auftreten, ist der Einfluss auf die gesamte Ausführungsdauer nicht konstant, so dass diese teilweise voneinander abweichen und sich die hohe Standardabweichung ergibt. Die Nachrichten in der Warteschlange werden nach dem FIFO-Prinzip abgearbeitet und weitergeleitet, so dass der eventgesteuerte Controller ebenfalls im Takt der Bandbreite antwortet. Bis zum Eintreffen einer Antwort vom Controller sind daher mehrfach Nachrichten mit denselben Daten vom Roboter verschickt worden, welche sich fortlaufend im Netzwerk stauen und nach Weiterleitung identisch vom Controller beantwortet werden. Somit verlängert sich nicht nur die Größe der Warteschlange im Netzwerk exponentiell, sondern auch die Anzahl der Nachrichten mit gleichem Inhalt. Obwohl am Roboter Nachrichten ankommen, sind diese identisch und werden verworfen. Jedoch ist so keine neue Positionierung möglich und der Roboter bleibt stehen. Unter diesem kritischen Umstand kann die Aufgabe nicht abgeschlossen wer-

den.

5.3.2 Bewertung der Latenz-Experimente

Es zeigt sich, dass die Latenz die größten Auswirkungen auf den Roboter und damit auf die Ausführungsqualität des Referenzalgorithmus hat. Wie in Diagramm 4.8 dargestellt ist, steigt die zur Aufgabenlösung benötigte Gesamtzeit ungefähr alle 8 ms Latenzerhöhung um etwa 20 s. Summiert man die OWD-Latenzen für die Strecken *Sensor* $\rightarrow$ *Controller* und *Controller* $\rightarrow$ *Aktuator* auf, ergibt sich, bei vernachlässigbarer Algorithmus-Rechendauer, eine RTT von ca. 16 ms. Diese Verzögerung entspricht etwa dem Intervall von zwei Sensorabtastungen. Daher ist davon auszugehen, dass innerhalb dieser Zeit eine weitere Nachricht mit denselben Daten und derselben Verzögerung auf den Weg gebracht wird. Die Antwort dieser zweiten Nachricht wird verworfen, dies bedeutet, dass bei einer Latenz von 16 ms nur noch jede zweite Nachricht neue Positionsdaten enthält. Die Gesamtdauer wird demnach nahezu verdoppelt. Bei 24 ms Latenz werden bereits zwei zusätzliche Nachrichten mit denselben Daten versandt, bevor die erste Antwort vom Controller eintrifft, so dass es hier zum nächsten Sprung um 20 s Ausführungsdauer kommt. Als Konsequenz daraus wird der Roboter zunehmend langsamer und es treten vermehrt die in Abschnitt 5.2 diskutierten Schwingungen auf.

Die höchste in diesem Experiment verwendete Latenz beträgt $56ms$, was nach (2.1) bei einer Glasfaserverbindung mit einem Ausbreitungsfaktor von 0.67 einer Ein-Weg-Distanz von etwa 11.000 km entspricht. Das bedeutet, dass der Regelungsvorgang auch unter den Netzwerkbedingungen großer Entfernungen erfolgreich durchgeführt werden könnte.

5.3.3 Bewertung der Paketverlust-Experimente

Die Ausführungsqualität des Algorithmus unter dem Einfluss verschiedener Paketverlustraten (vgl. Abb. 4.9) entspricht den Erwartungen. Da Nachrichten gemäß den Netzwerkeinstellungen anhand von Wahrscheinlichkeiten (vgl. Abschnitt 4.2.3) als Verluste behandelt werden, müssen diese wiederholt versandt werden. Die Paketverluste treten sowohl auf der Strecke zwischen Sensor und Controller, als auch zwischen Controller und Aktuator auf. Ein vollständiger Regelungsvorgang kann daher als mehrstufiges Zufallsexperiment angesehen werden. [27, S. 98f] Die benötigte Gedamtdauer je Verlustrate errechnet sich somit nach der hergeleiteten Formel 4.6.
Ohne dass eine Antwort mit neuen Steuerungsdaten vom Controller eintrifft, verlangsamt sich der Bewegungsablauf des Roboters. Darüber hinaus treten die in Abschnitt 5.2 erläuterten Vibrationen auf.
Erreicht die Paketverlustrate den kritischen Wert von 100% ist keine Lösung der Aufgabe mehr möglich.

KAPITEL 6

Zusammenfassung und Ausblick

Ziel dieser Bachelorarbeit ist es die Auswirkungen von Bandbreitenbegrenzungen, verschiedenen Latenzen und Paketverlustraten auf einen komplexen Regelungsalgoritmus in einem NCS experimentell zu untersuchen. In diesem Zusammenhang sollte die Frage beanwortet werden, welche weiteren Effekte, neben einer verzögerungsbedingten verlängerten Ausführungsdauer auftreten. Auf der Controllerseite im NCS wurde ein Lösungsalgorithmus für eine kraftgesteuerte „Peg in Hole“-Aufgabe in der Programmiersprache C/C++ implementiert. Ausgeführt werden sollte diese Aufgabe von einem *KUKA LBR iiwa 14 R820* 6-DoF-Roboterarm in der Simulationsumgebung *CoppeliaSim*. Als Verbindung zwischen dem Roboter in der Simulationsumgebung und der Außenwelt fungiert ein C/C++-Plugin, über welches unter anderem die Verbindung zum Controller, bzw. zum Netzwerk hergestellt wird. Um während der Experimente die verschiedenen Dienstgüten schnell und einfach anpassen zu können, wurde darüber hinaus ein Routingprogramm entwickelt. Dieses Programm dient einerseits zur Weiterleitung der Datenpakete zwischen Roboter und Controller und andererseits zur Umsetzung von gewollten Störungen des Netzwerkes, wie Verzögerungen und Paketverlusten. Nach einer Überprüfung der Komponenten auf die erwartete Funktionalität wurden mehrere Experimente durchgeführt. Messwerte waren dabei die zur Aufgabenlösung benötigte Gesamtzeit sowie die aufgetretenen Kräfte und die Anzahl der Fehlversuche. Es konnte gezeigt werden, dass sich die Aufgabenlösungszeiten unter dem Einfluss der verschiedenen Dienstgüten erwartungsgemäß verhielten. Durch Vibrationen des Roboterarmes und die damit einhergehenden Schwankungen der Kraft kam es in allen Experimentierreihen nur zu sporadischen Fehlversuchen, die sich bei größeren Verzögerungen nicht wesentlich vermehrten. Die Bandbreite, die im Rahmen dieser Arbeit in *Nachrichten pro Sekunde* implementiert wurde, hat bis zu einer Übertragungsrate, die etwa der Abtastrate des Sensors entspricht, keine Auswirkungen. Bei niedrigeren Bandbreiten kommt es zunächst durch Schwankungen der Abtastfrequenz zu einem Anstieg der Aufgabenlösungszeit und schließlich zum Stillstand der Ausführung. Die aufgetretenen Maximalkräfte nehmen im Verlauf des Experimentes leicht zu und steigen bei Erreichen der Abtastrate des Sensors trotz der Vibrationen am Roboterarm nicht wesentlich an. Im Rahmen der Latenz-Experimente zeigen sich regelmäßig, bei Erhöhungen der Netzwerklatenz um die Hälfte der Abtastrate des Sensors, ein sprunghafter Anstieg der Aufgabenlösungszeit. Bis zu einer Latenz von $56ms$ kann die „Peg in Hole“-Aufgabe erfolgreich abgeschlossen werden. Sowohl am Roboterarm,

als auch in den aufgenommenen Kraftkurven kann bei steigenden Latenzen eine Zunahme von Vibrationen registriert werden. Unter diesen Bedingungen steigen die gemessenen Maximalkräfte jedoch nicht wesentlich an. Bei den Experimenten mit zunehmender Paketverlustrate verlängert sich die Aufgabenlösungszeit erwartungsgemäß exponentiell. Darüber hinaus kommt es bei steigender Paketverlustrate zu ähnlichen Instabilitäten wie bei den Latenz-Experimenten. Die aufgetretenen Vibrationen sorgen auch hier für keinen relevanten Anstieg der Maximalkraft.
Zusammenfassend läßt sich sagen, dass der entwickelte Regelungsalgoritmus zur Lösung einer „Peg in Hole“-Aufgabe in einem NCS bis auf wenige Ausnahmen stabil funktioniert. Gemessen an der benötigten Gesamtdauer zur Beendigung der Aufgabe treten die erwarteten Qualitätseinbußen im Sinne einer Verlängerung der Gesamtzeit auf.

In zukünftigen Arbeiten, die sich mit dieser Thematik beschäftigen, sollten größere Messreihen aufgenommen werden, um zuverlässigere Aussagen zu netzwerkinduzierten Effekten zu generieren. Weitere Regelungsalgorithmen, wie beispielsweise das inverse Pendel oder ”Peg in Hole” mit nicht-zylindrischen Formen zu testen, wären interessante Herausforderungen. Auch Versuche mit reellen Robotern sowie echten (Test-)Netzwerken, wie dem Magdeburg Internet of Things Lab (MIoT-Lab) sind anzustreben, um Effekte zu identifizieren, die in virtuellen Simulationen nicht vorkommen.

Literatur

[1] Steffen Siegl. *Networked Control Systems: Ein Überblick*. Techn. Ber. Technical report, Universität der Bundeswehr München Institut für Steuer–und …, 2017.

[2] Wei Zhang, M. S. Branicky und S. M. Phillips. „Stability of networked control systems“. In: *IEEE Control Systems Magazine* 21.1 (2001), S. 84–99.

[3] Alberto Bemporad, Maurice Heemels, Mikael Johansson u. a. *Networked control systems*. Bd. 406. Springer, 2010.

[4] Jing Xu u. a. „Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies“. In: *arXiv preprint arXiv:1904.05240* (2019).

[5] Yin Xu, Yue Hu und Lei Hu. „Precision peg-in-hole assembly strategy using force-guided robot“. In: *2015 3rd International Conference on Machinery, Materials and Information Technology Applications*. Atlantis Press. 2015.

[6] E. C. Martins und F. G. Jota. „Design of Networked Control Systems With Explicit Compensation for Time-Delay Variations“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40.3 (2010), S. 308–318.

[7] Qi Zhang, Jianhui Liu und Guodong Zhao. „Towards 5G enabled tactile robotic telesurgery“. In: *arXiv preprint arXiv:1803.03586* (2018).

[8] Yanfeng Wang u. a. „Stabilization for networked control system with time-delay and packet loss in both SC side and CA side“. In: *IEEE access* 8 (2019), S. 2513–2523.

[9] Alexios Papacharalampopoulos u. a. „The effect of communications on networked monitoring and control of manufacturing processes“. In: *Procedia CIRP* 41 (2016), S. 723–728.

[10] S. 20 u. a. „Medical Telerobotics and the Remote Ultrasonography Paradigm Over 4G Wireless Networks“. In: *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*. 2018, S. 1–6.

[11] C. B. Schindler u. a. „Implementation and characterization of a multi-hop 6TiSCH network for experimental feedback control of an inverted pendulum“. In: *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. 2017, S. 1–8.

[12] Winarno Sugeng u. a. „The impact of QoS changes towards network performance“. In: *International Journal of Computer Networks and Communications Security* 3.2 (2015), S. 48–53.

[13] Patrick-Benjamin Bök u. a. *Computernetze und Internet of Things*. Springer, 2020.

[14] Christian Baun. *Computernetze kompakt*. Springer-Verlag, 2018.

[15] Larry L Peterson und Bruce S Davie. *Computer networks: a systems approach*. Elsevier, 2007.

[16] Jürgen Scherff. *Grundkurs Computernetze*. Springer-Verlag, 2007.

[17] Mustafa W Abdullah u. a. „An approach for peg-in-hole assembling using intuitive search algorithm based on human behavior and carried by sensors guided industrial robot". In: *IFAC-PapersOnLine* 48.3 (2015), S. 1476–1481.

[18] T. Yamashita u. a. „Peg-and-hole task by robot with force sensor: Simulation and experiment". In: *Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation* (1991), 980–985 vol.2.

[19] Kuangen Zhang u. a. „Jamming analysis and force control for flexible dual peg-in-hole assembly". In: *IEEE Transactions on Industrial Electronics* 66.3 (2018), S. 1930–1939.

[20] R. J. Chang, C. Lin und P. Lin. „Visual-Based Automation of Peg-in-Hole Microassembly Process". In: *Journal of Manufacturing Science and Engineering* 133 (Aug. 2011), S. 041015.

[21] Z. Qin u. a. „Precise Robotic Assembly for Large-Scale Objects Based on Automatic Guidance and Alignment". In: *IEEE Transactions on Instrumentation and Measurement* 65.6 (2016), S. 1398–1411.

[22] Kuangen Zhang u. a. „Force control for a rigid dual peg-in-hole assembly". In: *Assembly Automation* (2017).

[23] Te Tang u. a. „Autonomous alignment of peg and hole by force/torque measurement for robotic assembly". In: *2016 IEEE international conference on automation science and engineering (CASE)*. IEEE. 2016, S. 162–167.

[24] Peter Hehenberger. *Computerunterstützte Fertigung: Eine kompakte Einführung*. Springer-Verlag, 2011.

[25] CoppeliaRobotics. *http://www.coppeliarobotics.com/helpFiles/en/plugins.htm*. URL: `http://www.coppeliarobotics.com/helpFiles/en/plugins.htm` (besucht am 24. 08. 2020).

[26] Kurt Magnus, Karl Popp und Walter Sextro. *Schwingungen: physikalische Grundlagen und mathematische Behandlung von Schwingungen*. Springer-Verlag, 2013.

[27] Claus Brell, Juliana Brell und Siegfried Kirsch. *Statistik von Null auf Hundert*. Springer, 2017.

[28] L Papula. „Mathematische Formelsammlung: für Ingenieure und Naturwissenschaftler (10., überarbeitete und erweiterte Auflage)". In: *Online-Ressource: v.: digital. Wiesbaden: Vieweg+ Teubner.(Siehe S. 111)* (2009).

[29] Takashi Imaida und Kei Senda. „Performance Improvement of PD-Based Bilateral Teleoperator with Time Delay by Introducing High-Pass Filter". In: *Robotica* 38.7 (2020), S. 1318–1342.

Anhang

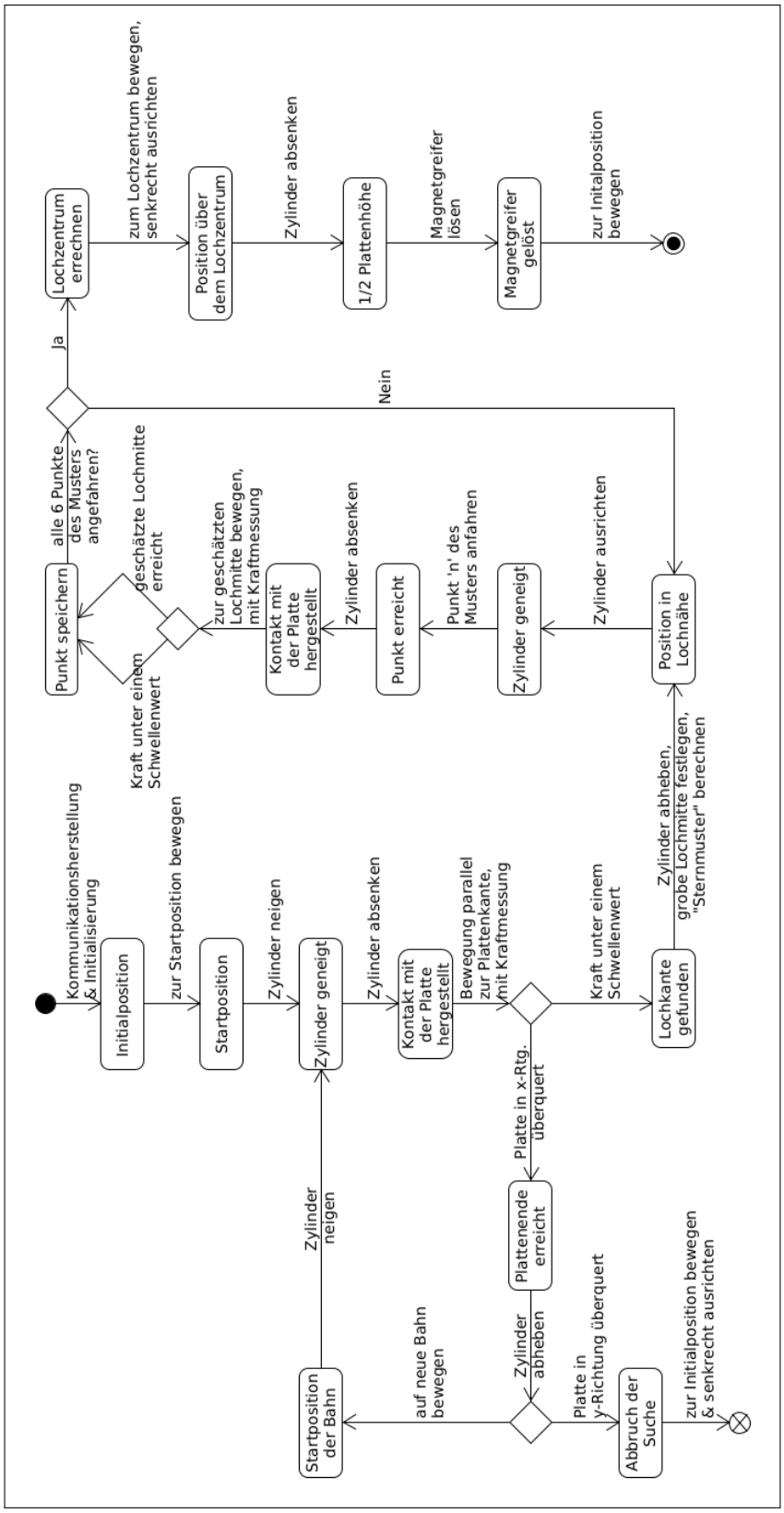

Abbildung A.1: Der Ablauf des Algorithmus als Zustandsmaschine

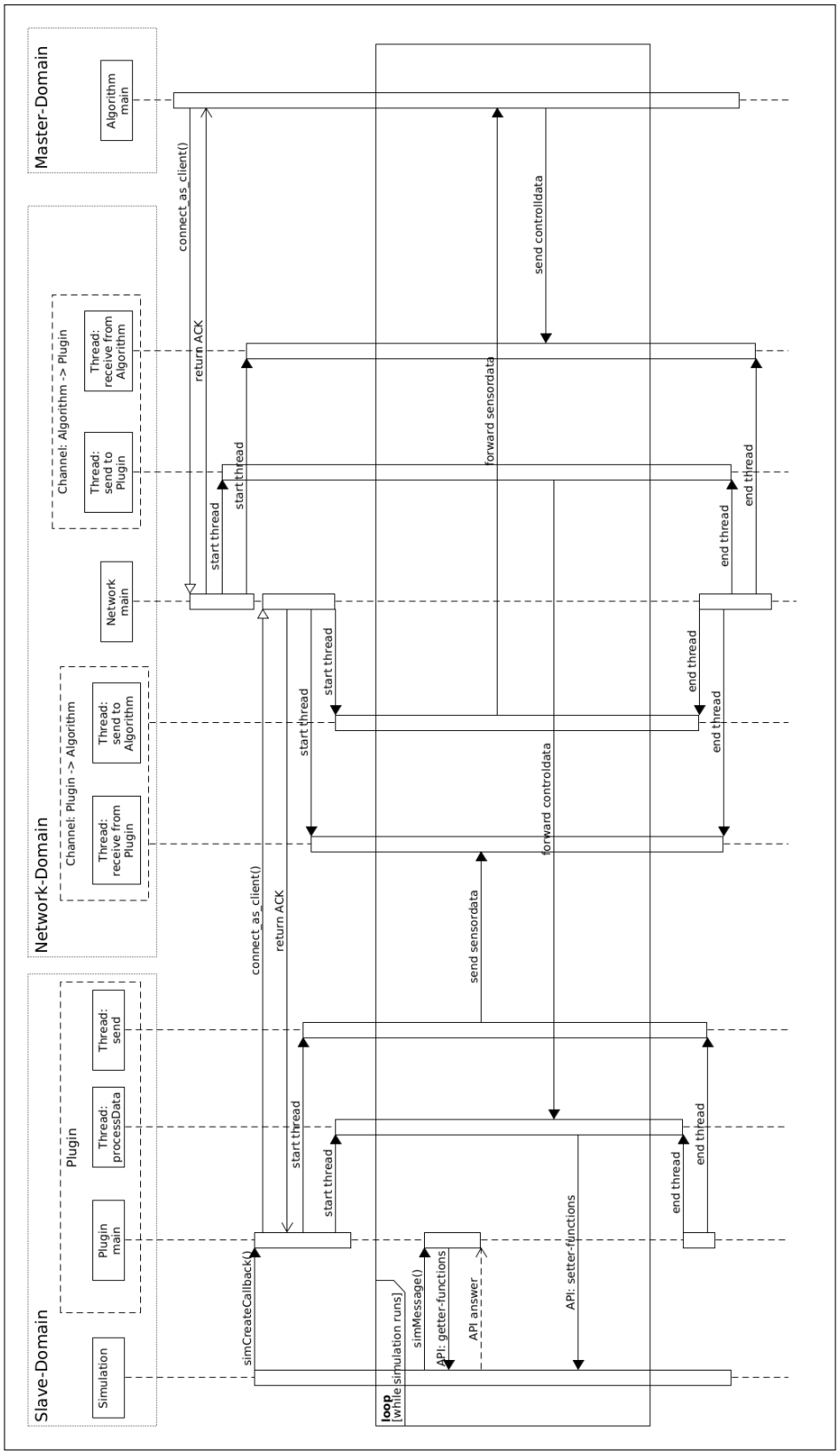

Abbildung A.2: Ablauf der Kommunikation

Hiermit versichere ich, dass die vorliegende Arbeit mit dem Titel *Auswirkungen verschiedener Dienstgüten auf einen "Peg in Hole"-Lösungsalgorithmus in einem Networked Control System* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommenen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

Magdeburg, 30. November 2020 _____

(Martin Popp)