TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Verkehrswissenschaften „Friedrich List“ Professur für Verkehrssicherungstechnik

Diplomarbeit

Evaluationsumgebung für IP-basierte Komponenten der Leit- und Sicherungstechnik

eingereicht von Frederic Reiter

Prüfer:

- PD Dr.-Ing. habil. Ulrich Maschek
- Dipl.-Ing. Richard Kahl

Betreuer:

- PD Dr.-Ing. habil. Ulrich Maschek
- Dr. Lukas Iffländer (DZSF)

Dresden, den 30.11.2022 .....

Unterschrift des Studenten

Autorenreferat

Die vorliegende Arbeit befasst sich mit der Evaluation von IP-basierten Komponenten der Leit- und Sicherungstechnik. Die nächste digitale Stellwerksgeneration wird über standardisierte Schnittstellen verfügen und über ein IP-Netz kommunizieren, was neue Chancen und Herausforderungen für Planung, Zulassung und Betrieb mit sich bringt. Hersteller und Behörden benötigen Werkzeuge wie Testumgebungen, um die neuen Komponenten auf effektive Art und Weise zu testen. Das Ziel dieser Arbeit ist es, ein Konzept für ein solches Werkzeug in Form einer Testumgebung zu liefern und, mittels einer prototypischen Implementierung dieser, Optimierungspotentiale der Anwendung von EULYNX Schnittstellen zu untersuchen. Im Rahmen dieser Arbeit werden Anforderungen untersucht, die an eine solche Testumgebung gestellt werden. Als Grundlage dafür erfolgt eine Einführung in digitale Stellwerke, in die EULYNX Spezifikationen sowie in das RaSTA-Protokoll. Darauf aufbauend werden Aufbau und Funktionsweise des entwickelten Testumgebungskonzepts beschrieben. Die Validierung der Testumgebung erfolgt durch den Test einer Open Source Implementierung des RaSTA-Protokolls. Abschließend werden die Ergebnisse des Tests präsentiert und die daraus gewonnenen Erkenntnisse u. a. im Bezug auf Fahrstraßenbildezeiten und Anzahl von steuerbaren Stelleinheiten diskutiert.

Bibliografischer Nachweis

Frederic Reiter

Evaluationsumgebung für IP-basierte Komponenten der Leit- und Sicherungstechnik

Technische Universität Dresden, Fakultät Verkehrswissenschaften „Friedrich List“, Professur für Verkehrssicherungstechnik

Diplomarbeit 2022

Anzahl der Seiten:	100
Anzahl der Abbildungen:	30
Anzahl der Tabellen:	12
Anzahl der Quellenangaben:	95
Anzahl der Anlagen:	3

Abstract

This study deals with the evaluation of IP-based CCS components. The next generation of digital interlockings will support standardized interfaces and will communicate by means of IP networks, introducing new opportunities and challenges for all lifecycle stages of railway infrastructure. Manufacturers and governmental bodies need tools to test and validate the new components. The aim of this study is to provide a concept for such a tool in form of a testing environment and research the potential of the application of the EULYNX interfaces using a prototype implementation. For this purpose, the requirements of such a testing environment are analysed and the basics of digital interlockings, EULYNX specification and RaSTA protocol are introduced. Furthermore, architecture and specific functionality of the testing environment are described. Finally, the results obtained during validation of the testing environment by test runs with the RaSTA protocol are presented and their implications are discussed, specifically in terms of route setting times and number of connectable actuators.

Thesen zur wissenschaftlichen Arbeit

1. Durch die Einführung der nach EULYNX standardisierten Schnittstellen lassen sich Laufzeitverkürzungen innerhalb der Leit- und Sicherungstechnik erreichen.

2. Eine Evaluationsumgebung zur Durchführung von Komponenten- und Integrationstests lässt sich auf Basis von Open Source Software umsetzen.

3. Die bestehende Open Source Implementierung des Rail Safe Transport Application Protokoll muss auf Schwachstellen / mangelnde Umsetzung der Spezifikation (DIN VDE V 0831-200) überprüft werden und eine Fehlerbereinigung erfolgen.

4. Mit einer Evaluationsumgebung zur Projektion von bahnbetrieblichem IP-Netz und EULYNX-Komponenten lassen sich Fahrstraßenbildezeiten messen und die Funktionalität von Netzwerkprotokollen testen.

5. Es existieren kostengünstige COTS-Komponenten, welche konventionelle Elemente der Leit- und Sicherungstechnik ersetzen können.

6. Die Datenrate des bahnbetrieblichen IP-Netzes wird den Anforderungen im Bezug auf die Anzahl der angeschlossenen Object Controller gerecht.

Inhaltsverzeichnis

1 Einleitung ... 8

2 Zielstellung ... 10

3 Grundlagen der digitale Stellwerke ... 11

3.1 Allgemeine Beschreibung und Funktionsweise eines Stellwerks ... 11

3.2 Bedeutung von digitalen Stellwerken ... 12

3.3 Qualitätsmerkmal Fahrstraßenbildezeit ... 13

3.4 Die Weiterentwicklung von ESTW zu DSTW ... 16

3.5 Beschreibung und Funktionsweise von DSTW ... 17

4 Aufbau DSTW ... 19

4.1 Messungen Netzwerkparameter bb-IP-Netz ... 21

4.2 Zulassung neuentwickelter Systeme im Schienenverkehr ... 23

5 Evaluationsumgebung ... 26

5.1 Testverfahren „modulares, hierarchisches Testen“ ... 26

5.2 Anforderungsanalyse für die Evaluationsumgebung ... 27

5.3 Technischer Hintergrund der Evaluationsumgebung ... 28

5.4 OSI-Schichtenmodell der Rechnerkommunikation ... 29

5.5 Testumgebungs-Software von dritten Herstellern ... 32

6 Aufbau der Evaluationsumgebung ... 34

6.1 Uhrensynchronisation ... 37

7 Validierung von DSTW-Sim ... 40

7.1 EULYNX Spezifikation ... 41

7.1.1 Subsystem Point - SCI-P ... 44

7.1.2 Subsystem Light Signal – SCI-LS ... 46

7.2 Rail Safe Transport Application RaSTA ... 47

7.2.1 Konfigurierbare Parameter ... 48

7.2.2 Sicherheits- und Sendewiederholungsschicht ... 49

7.2.3 Prüfung der Sequenznummer ... 50

7.2.4 Prüfung der rechtzeitigen Übertragung ... 51

7.2.5 Redundanzschicht ... 51

7.2.6 Allgemeine Funktionsweise ... 52
7.2.7 Sendewiederholung ... 52
7.2.8 Schnittstelle zur Transportschicht ... 54
7.2.9 Security-Eigenschaften von RaSTA ... 54
7.2.10 Selbstständig entwickelte Komponenten ... 54
7.3 Entwicklung Module SCI-P, SCI-LS und Stellwerk ... 56
7.3.1 Wahl der Programmiersprache ... 56
7.3.2 Architektur der Module ... 57

8 Experimentdurchführung und Messergebnisse ... 59
8.1 Experimentdurchführung ... 59
8.2 Messergebnisse ... 62
8.2.1 Synchronisation der Uhren ... 62
8.2.2 Latenz ... 63
8.2.3 Paketverlustrate ... 68
8.2.4 CPU-Zeit / RAM Auslastung ... 68
8.2.5 Bandbreite ... 70

9 Diskussion ... 71
9.1 Weiterentwicklung von Open Source RaSTA ... 71
9.1.1 Dokumentationsfehler ... 71
9.1.2 Doppeluhrenprinzip ... 71
9.1.3 Logging ... 73
9.1.4 Verbindungsaufbau zu mehreren Teilnehmern ... 73
9.2 Diskussion der Messergebnisse aus den Experimenten ... 74
9.2.1 Uhrensynchronisation ... 74
9.2.2 Latenz ... 74
9.2.3 Systemlaufzeiten ... 75
9.2.4 CPU-Zeit / RAM Auslastung ... 76
9.2.5 Auswirkungen der Laufzeiten auf Fahrstraßenbildezeit ... 77
9.2.6 Paketverlustrate ... 78
9.2.7 Bandbreite ... 79

10 Fazit ... 82
10.1 Ausblick ... 82

Abkürzungsverzeichnis **83**

Abbildungsverzeichnis **85**

Tabellenverzeichnis **87**

Literaturverzeichnis **88**

Erklärung **96**

Anhang A: Übersicht über Logging-Nachrichten **97**

Anhang B: Email-Kommunikation Dr. Lukas Iffländer **98**

Anhang C: Email-Kommunikation „Protokolle von Laufzeiten“ - Petros Matios **100**

1 Einleitung

Der Eisenbahn kommt als Hoffnungsträger für das Erreichen der deutschen Klimaziele im Verkehrssektor eine entscheidende Bedeutung zu. Im Zusammenschluss mit seinen europäischen Partnern hat Deutschland in Bekenntnis zum 2015 unterzeichneten Pariser Klimaabkommen zugesagt, den Ausstoß von Treibhausgasen im Sektor Verkehr bis zum Jahr 2030 um mindestens 40% gegenüber 1990 zu reduzieren. Durch das Klimaschutzgesetz von 2021 sind diese Ziele auf nationaler Ebene verbindlich festgelegt, bei Verfehlung drohen Sanktionen in Form von Sofortprogrammen mit unmittelbarer Wirkung [KLI2]. Der dazugehörige Maßnahmenkatalog, das Klimaschutzprogramm 2030, enthält zwei Maßnahmenbündel für den Schienenpersonen- und -güterverkehr [KLI30]. In diesen wird unter anderem die Beseitigung von Engpässen, Steigerung der Attraktivität für Nutzer und Einführung von digitaler Leit- und Sicherungstechnik angestrebt. Als wesentliches Instrument zur Beseitigung von Engpässen und zur Leistungssteigerung des Schienenverkehrs wird die Digitalisierung von Stellwerken genannt. Während konventionelle Stellwerkstechnik für jedes individuelle Gleisfeldelement einen separaten Anschluss für Stromversorgung und Datenübertragung zentral am Stellwerk vorsieht, werden bei digitalen Stellwerken (DSTW) gemäß des aktuellen Stands der Technik die Gleisfeldelemente über Netzwerke angesteuert. Neben den positiven Effekten auf die Leistungsfähigkeit des Schienennetzes bringen DSTW eine Reihe von weiteren Vorteilen mit sich, beispielsweise die Standardisierung von Komponenten des Gleisfeldes, einfachere Instandhaltung, stabilere Datenleitungen und größere Bedienbereiche, mit jeweils kaskadierenden Gewinnen für wirtschaftlicheren und effizienteren Betrieb [SCH21].

Standardisierte Schnittstellen ermöglichen eine modulare Auftragsvergabe innerhalb von Stellwerksprojekten, was zu einer Öffnung des Marktes für kleine und mittelständische Unternehmen und dadurch zu mehr Wettbewerb und wesentlich günstigeren Kosten für Bau, Betrieb und Instandhaltung von Stellwerken führt. Durch die größeren Bedienbereiche werden beim Umbau von DSTW auf Stelleinheiten bezogen größere Bauvolumina erreicht, was die Kosten durch Skaleneffekte[1] senkt [PAS17].

Hardware und Software von im Schienenverkehr eingesetzten sicherheitsrelevanten Komponenten müssen hohen Sicherheitsanforderungen genügen. Ist die Tauglichkeit durch Gutachten und Sicherheitsprüfungen nachgewiesen, erfolgt die Zulassung durch das Eisenbahnbundesamt. Für die Validierung dieser Komponenten und die Durchführung der Sicherheitsprüfung durch Behörden und Unternehmen sind spezielle Werkzeuge nötig, um Konformität mit allen relevanten Schnittstellen und eventuelle systemübergreifende Auswirkungen zu testen sowie Aussagen über die Leistungsfähigkeit der Komponente machen zu können. Ein solches Werkzeug ist eine Evaluierungsumgebung, welche den unkomplizierten Austausch von zu testenden Modulen ermöglicht, komplexe Gleisfelder darstellen und unterschiedliche

[1] Effizientere Ressourcenauslastung durch Größe

Bedingungen (bspw. negative Beeinträchtigungen des Netzes wie Latenzen) simulieren kann. [SCH22; HUN16]

2 Zielstellung

Ziel der Arbeit soll die Konzeption und prototypische Umsetzung des o.g. Konzeptes einer Evaluierungsumgebung für IP-Netz basierte Komponenten der Leit- und Sicherungstechnik sein. Die Funktionsfähigkeit soll anhand eines Beispiels demonstriert werden. Zu diesem Zweck soll das für die Kommunikation über das bahnbetriebliche-IP-Netz vorgesehene und DIN-genormte Protokoll Rail Safe Application Protocol (RaSTA) einem Leistungstest unterzogen werden.

Die hieraus resultierenden Teilaufgaben sind in den jeweiligen Kapiteln beschrieben. Kapitel 3 führt in die Grundlagen der Stellwerke ein. In Kapitel 4 wird der Aufbau von DSTW beschrieben. In Kapitel 5 werden die Anforderungen an die Evaluierungsumgebung definiert und die technischen Grundlagen der Umsetzung erklärt, bevor Einzelheiten des Aufbaus in Kapitel 6 beschrieben werden. Kapitel 7 führt in die für die Validierung der Umgebung wichtigen Grundlagen von EULYNX und RaSTA ein. Kapitel 8 beginnt mit einer Beschreibung der Experimentdurchführung und enthält die Messergebnisse, die Diskussion dieser schließt sich in Kapitel 9 an. In Kapitel 10 werden die Erkenntnisse der Arbeit zusammengefasst.

3 Grundlagen der digitale Stellwerke

In der vorliegenden Arbeit werden Simulationen von Betriebs- und Informationsabläufen in der Bahninfrastruktur mittels einer digitalen Testumgebung entwickelt. Im Folgenden werden daher zunächst die für das Verständnis der Abläufe erforderlichen Informationen zur Organisation und Technologie digitaler Stellwerke beschrieben:

3.1 Allgemeine Beschreibung und Funktionsweise eines Stellwerks

Aus den charakteristischen Systemeigenschaften der Eisenbahn ergibt sich die Notwendigkeit einer leit- und sicherungstechnischen Ebene für den Schienenverkehr: Lange Bremswege aufgrund des geringen Haftreibungskoeffizienten zwischen Stahlrad und Stahlschiene und die ausschließliche Möglichkeit der Änderung des Fahrwegs an diskreten Punkten als Konsequenz der Spurführung erfordern die Kommunikation der Verkehrsteilnehmer miteinander sowie mit der Infrastruktur. Die zu bewältigenden Aufgaben sind die räumliche Trennung der Schienenfahrzeuge, von welchen aus bei hohen Geschwindigkeiten der Bremsweg nicht vollständig einsehbar ist, die richtige Einstellung der beweglichen Fahrwegelemente (Weichen) und gegebenenfalls die Abstimmung mit systemfremden Verkehrsteilnehmern an Bahnübergängen. Auch wenn es alternative Konzepte gibt, welche die individuelle Kommunikation von Fahrzeugen mit Weichen und untereinander beinhalten, so werden diese Aufgaben im Interesse eines möglichst leistungsfähigen und optimierten Verkehrsgeschehens meist von einer zentralen Stelle übernommen. Historisch betrachtet war diese immer das sogenannte Stellwerk, welches die Steuerung der Feldelemente, deren sicherungstechnische Abhängigkeiten (Sicherungslogik) und Schnittstellen für andere Systeme, wie Gleisfreimeldeanlagen, beinhaltet, räumlich begrenzt auf den Stellbezirk. [MAS15]

Nachdem es in der Anfangszeit des Eisenbahnverkehrs ohne ein übergeordnetes System zur Abstandhaltung oftmals zu Unfällen gekommen war, wurden die Schienenwege in ortsfeste Blöcke eingeteilt, in denen sich nur ein Fahrzeug gleichzeitig aufhalten durfte. Die Belegung eines Blocks wird heutzutage technisch überprüft, in Deutschland fast ausschließlich mittels einer Erfassung der Anzahl der in einen Block ein- und ausfahrenden Achsen (Achszähler) oder der Kontrolle des unterbrechungsfreien Flusses eines Stroms im Gleis (Gleisstromkreis). Ist der zu befahrene Schienenweg frei, kann die Fahrt erlaubt werden, sobald alle beweglichen Fahrwegelemente in der erforderlichen Weise festgelegt und verschlossen sind. Unter der zusätzlichen Voraussetzung, dass diese Einrichtungen während der Dauer der Fahrterlaubnis auch verschlossen bleiben, ist dieser Zusammenhang als Signalabhängigkeit bekannt. Diese Erlaubnis konnte zu Zeiten der ersten Stellwerksbauformen nur durch das Stellwerk und nur per Lichtsignal (oder schriftlichem/fernmündlichem Befehl) an das Fahrzeug übertragen werden.

Der per Festlegung und Verschluss gesicherte Fahrweg wird Fahrstraße genannt. Die Abhängigkeiten zwischen Signalen und Einrichtungen sind für alle möglichen Fahrstraßen im Stellbezirk eines Stellwerks in den Verschlussplänen dargestellt. Durch die Stellwerkslogik ist sichergestellt, dass Verletzungen der Signalabhängigkeit laut Verschlussplan unmöglich sind (vgl. [HEI13; EBU08]).

3.2 Bedeutung von digitalen Stellwerken

Der technische Fortschritt führte seit dem Bau des ersten mechanischen Stellwerks vor über 150 Jahren [TUM] zu verschiedenen Stellwerksbauformen, von denen jede neue Generation Optimierungspotentiale zur Steigerung von Sicherheit, Effizienz und Geschwindigkeit im Betriebsablauf oder in der Wirtschaftlichkeit umsetzen konnte. So ermöglichten Fortschritte in der Elektronik die Entwicklung von elektrischen Weichenantrieben, welche den für das Einstellen des Fahrweges benötigten Kraftaufwand wesentlich reduzierten. Die automatische Gleisfreimeldung führte zu einer Entlastung des Fahrdienstleiters und zu mehr Sicherheit durch Eliminierung der menschlichen Fehlerkomponente bei einer sicherheitskritischen Aufgabe [LAU21].

Aufgrund der Langlebigkeit von Infrastrukturprojekten sind in Deutschland heutzutage alle vier bisherigen Stellwerksbauformen parallel im Einsatz. In diesem Jahr ist es fast 40 Jahre her, seit das erste Stellwerk der vierten Generation, der elektronischen Stellwerke (ESTW), an die Deutsche Bahn übergeben wurde. Hierbei begann im Jahr 1985 die Praxiserprobung eines ESTW auf der Bahnstrecke München–Garmisch-Partenkirchen, als erstes elektronisches Stellwerk der DB AG [ETR85]. Mittlerweile ist die Einführung der fünften Generation, digitale Stellwerke (DSTW), in vollem Gange [LAU21; DB22].

Um die Notwendigkeit der technischen Weiterentwicklung der Bahninfrastruktur zu erkennen und die kleinschrittigen Möglichkeiten zur Implementierung dieser wertschätzen zu können, ist neben dem Blick in die Vergangenheit auch ein Blick in die Zukunft interessant. Die Digitalen Schiene Deutschland, das Kernprogramm der DB AG zur Verfolgung der Digitalisierungsziele, hat ein Zielbild für das Gesamtsystem Bahn formuliert, was unter anderem folgende Stichpunkte enthält [DSD19]:

- KI-basiertes Verkehrsmanagement plant und steuert Züge und Trassen
- Digitale Fahrgastlenkung optimiert Passagierströme in Echtzeit
- Störungen werden automatisiert erkannt und schneller bearbeitet
- Züge fahren automatisiert und erkennen ihr Umfeld
- Züge fahren in optimalen Abständen

Das DSTW auf Basis des bahnbetrieblichen IP-Netzes bildet die grundlegende Schicht für den Informationsaustausch der an der Umsetzung dieser Pläne beteiligten Teilsysteme.

Eine strikte Einteilung des Verantwortungsbereichs des Stellwerks verkompliziert sich jedoch zukünftig. Seit eine Fahrterlaubnis, auch Movement Authority (MA), nicht nur per Lichtsignal, sondern auch per Mobilfunk im Falle des Europäischen Zugbeeinflussungssystems ETCS (oder per Linienleiter im Falle des linienförmigen Zugbeeinflussungssystems LZB) an die Fahrzeuge übertragen werden kann, ist die Erteilung der Fahrterlaubnis nicht mehr zwingend Aufgabe des Stellwerks. Mit dem Blick auf die Entwicklungen in der Leit- und Sicherungstechnik, insbesondere die Ortung der Züge durch das Radio Block Centre (RBC) bei Fahrten unter ETCS Level 3, wird sich die Kernaufgabe des Stellwerks langfristig auf die Ausführung der Befehle zur Steuerung der Weichen beschränken, welche von neuen Elementen wie RBC und Train Management System (TMS) weitergegeben werden [BÜK20].

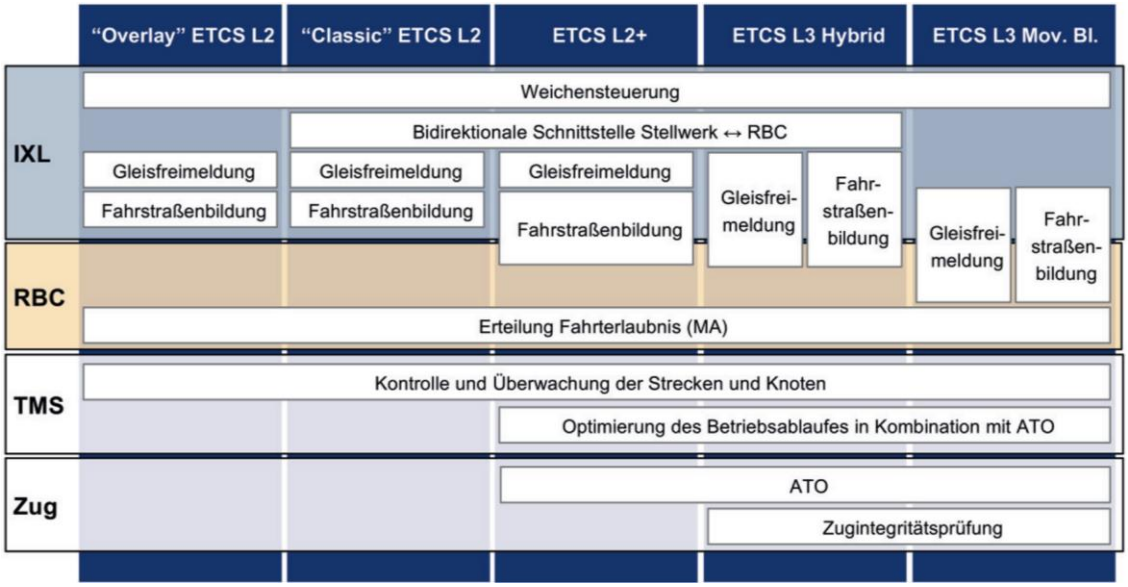

Abb. 3.1: Entwicklung der Verantwortlichkeiten innerhalb der LST mit zunehmender Optimierung des Zugbeeinflussungssystems ETCS [BÜK20]

3.3 Qualitätsmerkmal Fahrstraßenbildezeit

Fahrwegkapazität und Betriebsqualität werden maßgeblich durch die Zeitverbräuche einer Zugfahrt bestimmt. Die zentrale Größe für die Qualität eines Stellwerks im Sinne von dessen Einfluss auf die Dauer von Zugfahrten ist neben der Länge der Fahrwegabschnitte der Anteil der Übertragungs- und Verarbeitungszeiten, auch Laufzeiten genannt (vgl. Abb. 3.2).

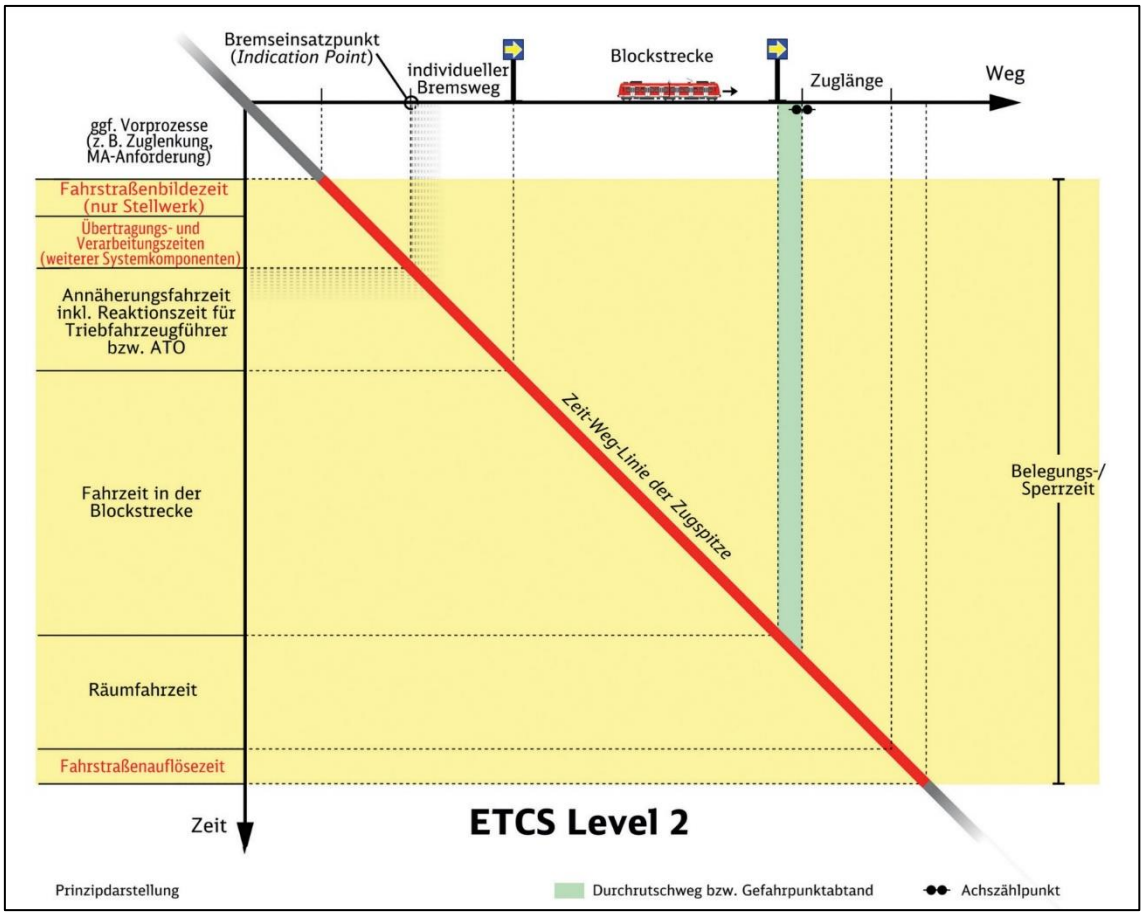

Abb. 3.2: Darstellung der Belegungs- und Sperrzeitkomponenten der LST mit ETCS Level 2. Der Anteil der LST-Laufzeiten ist rot beschriftet, aus [BEH21]

Die als Zugfolgezeit bezeichnete Zeit zwischen zwei Zugfahrten wirkt sich direkt auf die Fahrwegkapazität aus und wird nach Räumung des Fahrwegs durch den Zug nur durch die Geschwindigkeit beeinflusst, mit welcher die alte Fahrstraße aufgelöst und die neue Fahrstraße eingestellt werden kann, inklusive der Durchführung aller dafür nötigen Prüfungen. Die Berücksichtigung der Laufzeiten von Leit- und Sicherungstechnik in der Planung von Stellwerken wurde am Beispiel des Digitalen Knoten Stuttgart untersucht. Unter dem Ziel „maximaler Leistungsfähigkeit“ wurde diesen besondere Aufmerksamkeit gewidmet.

So wurden die Laufzeiten im Vergabeverfahren ein zuschlagsrelevantes Kriterium, welchem 5% der erreichbaren Gesamtpunktzahl beigemessen wurden (neben 85% für Preis und je 5% für Inbetriebnahme- und Redundanzkonzept) [BEH21]. Für sechs aussagekräftige Szenarien anhand der konkreten Infrastruktur mussten entsprechende Angaben gemacht werden. Ein wichtiger Aspekt im Verfahren war die Abkehr vom bloßen Erfassen der Gesamtlaufzeiten von einem bestimmten Start- bis zum Endergebnis zugunsten einer transparenten Aufschlüsselung der elementaren Laufzeiten. Für alle Szenarien existierten Erwartungswerte, welche auf Basis der „Untersuchung zur Einführung von ETCS im Kernnetz der S-Bahn Stuttgart“, durchgeführt von der Ingenieurgemeinschaft „Machbarkeitsstudie ETCS S-Bahn

Stuttgart“, sowie der DB-Richtlinie 819.0732 gebildet wurden [ING19]. Diese Werte konnten innerhalb der Ausschreibung vom Vergabegewinner Thales um 7 bis 10 Sekunden unterboten werden [BEH21].

Ein Beispiel für die Zusammensetzung der Ende-zu-Ende Laufzeiten einer Fahrstraßenbildung, vom Passieren des Fahrstraßenzugschlussstelle durch das vorausfahrende Fahrzeug bis zur Anzeige der Fahrterlaubnis auf dem Fahrzeugcomputer, existiert in Form einer Herstellerbefragung und ist in Tab. 3.1 dargestellt.

Zeitanteil	**Laufzeit ohne Weiche**	**Laufzeit mit Weiche**
Fahrstraßenauflösezeit	3 s	3 s
Fahrstraßenbildezeit	6 s	12 s
Übertragung ESTW-ETCS-Zentrale (ESTW-Z/RBC)	0,5 s	0,5 s
Verarbeitung in der ETCS-Z	0,5 s	0,5 s
Übertragung der Fahrterlaubnis per GSM-R	0,8 s	0,8 s
Laufzeitverzögerungen an Schnittstellen (GSM-R, SCI-RBC)	1 s	1 s
Verarbeitung und Anzeige im Fahrzeuggerät (OBU)	1,5 s	1,5 s
Summe (aufgerundet)	**14s**	**20s**

Tab. 3.1: Zusammensetzung der Ende-zu-Ende Laufzeiten einer Fahrstraßenbildung, nach [ING19]

Im Folgenden sind die einzelnen Laufzeiten, welche für die Zielstellung der Arbeit relevant sind, entsprechend der Definition in der oben genannten S-Bahn-ETCS-Untersuchung beschrieben:

Fahrstraßenauflösezeit
Die Fahrstraßenauflösezeit ist die Zeit, die vom Freifahren des Abschnitts an benötigt wird, bis zu dem Zeitpunkt, an dem alle Bedingungen im Stellwerk für das Einstellen der Fahrstraße für den folgenden Zug erfüllt sind.

Fahrstraßenbildezeit
Die Fahrstraßenbildezeit ist die Zeit, die vom Stellanstoß bis zur Meldung des Signalbegriffs an das RBC, am Ausgang des ESTW gemessen, benötigt wird.

Zeit zur Einstellung der Folgefahrstraße
Da die Fahrstraßenauflösezeit und die Fahrstraßenbildezeit einzeln schwer oder gar nicht zu ermitteln sind, wird angestrebt, die Zeit zur Einstellung der Folgefahrstraße zu ermitteln. Diese Zeit ist die Zeit, die benötigt wird, vom Freifahren des Abschnitts eines vorausfahrenden Zuges (Zug passiert Achszählpunkt), bis zur Meldung des Fahrtbegriffs an das RBC (Ausgang ESTW) für den Folgezug.

Fahrstraßenbildezeit unter ETCS Level 3

Eine Besonderheit bei der Berechnung der Fahrwegkapazität ergibt sich mit einem Blick auf zukünftige Abstandshalteverfahren wie unter ETCS Level 3 Moving Block. Laut Büker et al. (2020) gibt es die Fahrstraßenbildezeit als solche unter ETCS Level 3 Moving Block nicht mehr. So „[„springt“] die [Fahrterlaubnis] eines Zuges […] nicht mehr von Signal zu Signal, sondern wird dynamisch verschoben“ [BÜK20]. Dadurch wird die Bedeutung der Leistungsfähigkeit der Stellwerke bezüglich der Umlaufzeiten noch weiter unterstrichen, da es sonst zu Verzögerungen in der Erteilung der Fahrterlaubnis kommt, wo eigentlich bereits eine Zugfahrt stattfinden könnte [BÜK20].

3.4 Die Weiterentwicklung von ESTW zu DSTW

Das elektronische Stellwerk basiert bereits auf digitalen Rechnersystemen, d.h. in grundlegender Logik und Funktionalität unterscheiden sich ESTW und DSTW nicht. Die entscheidende Weiterentwicklung liegt in der Art und Weise der Vernetzung der am Bahnbetrieb beteiligten Teilsysteme, wie die Steuerung der Weichen und Lichtsignale, und in der Standardisierung der Schnittstellen für die Kommunikation dieser mit dem Stellwerk. „Für die Struktur der einzelnen ESTW gibt es erwartungsgemäß kein einheitliches Design.“, so Maschek in „Analyse zur Gestaltung elektronischer Stellwerke“ (1999) [MAS96]. Das klassische Stellwerk wird als monolithisches System eines einzelnen Herstellers ausgeschrieben, geplant, gebaut und betrieben. Die Abwesenheit eines einheitlichen Designs der ESTW, ganz zu schweigen von stellwerkübergreifend kompatiblen und austauschbaren Komponenten, sorgt für eine komplizierte bzw. unflexible Wartung und eingeschränkte Erweiterungsfähigkeit, für welche von den jeweiligen Herstellern hohe Preise gefordert werden (vgl. Anhang B).

Im Laufe der 40-jährigen Geschichte der ESTW wurden verschiedene Varianten der Datenübertragung zu den Feldelementen genutzt. Im Folgenden ist der Weg der Stellinformation vom Fahrdienstleiter bis zur Weiche in einem der üblichsten ESTW, Bauform EI S von Siemens, nach Maschek (1999) [MAS96] vereinfacht beschrieben (siehe Abb. 5 in [MAS96]).

Die Feldelemente sind über einen sternförmigen Stellwerksbus mit der Buszentrale und diese mit dem Bereichsstellrechner verbunden. Die Stellbefehle werden vom Bedienplatzrechner ausgelöst und im Bedien- und Anzeigerechner auf Zulässigkeit geprüft, dann weiter an den Bereichsstellrechner übermittelt und enden bei sogenannten Stellteilen, einer Kombination aus Elektronik und Signalrelais, welche die eigentlichen Stellelemente stellen und kontrollieren.

Der Datenaustauch wird in Form von normierten Telegrammen durchgeführt, bestehend aus Kopf mit Auftragsart, Empfängeradresse und Absenderadresse, einem Datenteil und einem Sicherungsabschnitt. Wegen Limitierungen in der Adressierung können an ein Bussystem lediglich 127 Rechner angeschlossen werden; die

maximale Anzahl von kontrollierbaren Stelleinheiten pro Bereichsstellrechner beträgt 30 bis 40. Durch Nutzung mehrerer Bedien- und Anzeigerechner pro Stellwerk können trotzdem in ein und demselben ESTW etwa 1000 Stelleinheiten beherrscht werden. Die maximale Stellentfernung über Kupferkabel beträgt lediglich 6,5 km und wurde erst in späteren Bauformen durch die Nutzung von anderen Technologien wie Lichtwellenleitern erweitert.

Der Austausch von Teilsystemen aufgrund von Weiterentwicklung dieser ist bei manchen Stellwerksbauformen für ausgewählte Teilsysteme möglich, bspw. bei Teilsystemen innerhalb der Sicherheitsschale der Bauform EI L. Bei neueren ESTW verschwimmt die Grenze zu DSTW im Bezug auf die Datenübertragung, denn die Nutzung eines IP-Netzes war in Stellwerksvarianten ab 2008 möglich.

3.5 Beschreibung und Funktionsweise von DSTW

Einhergehend mit der Anpassung von Schnittstellen des Stellwerks an die Kommunikation über das bb-IP-Netz sollten diese standardisiert werden. Standardisierte technische Schnittstellen sind ein Bestandteil des Alltags für die meisten Menschen heutzutage, wie beispielsweise die Standards USB oder HDMI im Bereich der persönlichen elektronischen Geräte. Gerade der Universal Serial Bus (USB) Standard ist eine Erfolgsgeschichte für Standardisierung in der Elektrotechnik. So hatte das USB Implementers Forum nur zwei Jahre nachdem 1996 die USB Spezifikation 1.0 veröffentlich wurde bereits mehr als 400 Unternehmen als Mitglieder (mehr als 1000 im Jahr 2022) und mehr als 500 Produkte befanden sich in der Entwicklung [1394NEWS].

Projekte mit dieser Zielsetzung für Schnittstellen der Eisenbahninfrastruktur begannen bereits im Rahmen des Integrated European Signalling System (INESS) Projektes der UIC, welches von 2008 bis 2010 durchgeführt wurde. Das Hauptziel sollte die bedeutende Reduzierung der Lebenszykluskosten für Stellwerke der Zukunft und zugehörige Außenanlagen sein [INE09]. Es wurden Grundlagen und Anforderungen ausgearbeitet, die vom 2012 gestarteten Projekt der Deutschen Bahn NeuPro („Neue Produktionsverfahren“) aufgegriffen wurden. NeuPro ging 2014 in EULYNX auf, einer Organisation von mittlerweile 13 europäischen Eisenbahninfrastrukturbetreibern (Stand: November 2022) mit dem Ziel, die eingangs erwähnte Einführung von Standards europaweit durchzusetzen [EUL17].
In EULYNX wurden folgende Schnittstellen – SCI (Standard Communication Interface) - zu allen wichtigen Außenverbindungen eines Stellwerks definiert:

- SCI-RBC (Radio Block Centre): Schnittstelle zur ETCS-Streckenzentrale
- SCI-ILS (Interlocking System): Schnittstelle zum Nachbarstellwerk
- SCI-LX (Level Crossing): Schnittstelle für Bahnübergänge mit Überwachungsart Fahrstraßenüberwachung (FSÜ)
- SCI-CC (Command & Control): Schnittstelle zum Bedienplatz

- SCI-IO (Input/Output): Schnittstelle für universelle digital Ein- und Ausgabesignale
- SCI-LS (Light Signal): Schnittstelle zu Lichtsignalen
- SCI-TDS (Train Detection System): Schnittstelle zur Gleisfreimeldeeinrichtung
- SCI-P (Point): Schnittstelle zur Steuerbaugruppe für Weichenantriebe

Daneben existieren Schnittstellen für Instandhaltungsdaten, SMI (Standard Maintenance Interface), und Diagnosedaten, SDI (Standard Diagnostic Interface) mit untergeordneter Bedeutung für diese Arbeit.

Von den erhofften Vorteilen wurden einige bereits genannt, an dieser Stelle sollen sie nochmals gebündelt zusammengefasst werden:
Die Nutzung von Standardkomponenten, sog. commercial off-the-shelf (COTS), für die Kommunikation zwischen Teilsystemen wird möglich. Zusätzlich können verschiedene Teilsysteme eines Stellwerkes an unterschiedliche Hersteller vergeben werden oder im Rahmen von Instandhaltungsarbeiten durch Teilsysteme anderer Hersteller ausgetauscht werden [CAS18]. Der Vertrieb von Systemen wird Teilsystemherstellern europaweit möglich, gegebenenfalls auch weltweit, sollte sich EULYNX global ähnlich erfolgreich verbreiten wie ERTMS, bei dem zum Stand 2020 wenig mehr als die Hälfte der ausgerüsteten Streckenkilometer innerhalb Europas liegen [TRI20].

Die DB Netz AG plant die Einführung der EULYNX-konformen DSTW in einem mehrschrittigen Prozess, wie auf der Webseite beschrieben ist [DBNWEB]:

In **Phase 1: Referenzimplementierung** wurden von 2014 bis 2019 DSTW gebaut, die jeweils eine oder mehrere der oben genannten SCI umsetzen, bspw. Stellwerke Neuwiederitzsch (SCI-RBC), Kreiensen/Naensen (SCI-ILS) oder Annaberg-Buchholz (vollständig standardisiert). Diese Implementierungen stellen Proof-of-Concept-Projekte dar, innerhalb derer das erste DSTW, Annaberg-Buchholz, zugelassen wurde.
Phase 2: Vorserie besteht aus vier Projekten, darunter Warnemünde und das in dieser Arbeit als Referenz für die Qualität des bb-IP-Netzes genutzte Meiningen-Mertingen (vgl. Kapitel 4.1).
Phase 3: Industrialisierungsprojekte (Schnellläuferprogramme und Starterpaket). Nach den Erprobungen in Referenzimplementierung und Vorserie ist schließlich der Rollout in ganz Deutschland geplant. Die in der Vorserie erprobten Grundlagen werden in die Regelwerke überführt und die Bauprozesse werden optimiert.
Phase 4: Flächenrollout Deutschlandweite Einführung der DSTW (Detailplanung ist in Vorbereitung).

4 Aufbau DSTW

Im Rahmen des Materials zu Informationsveranstaltungen der DB Netz AG für das Eisenbahnbundesamt (Workshop am 21.10.2021, [DSD21]) konnten für diese Arbeit Einblicke in die geplante, bzw. im Rahmen der Vorserienprojekte bereits umgesetzte Netzwerkarchitektur und den Aufbau der DSTW sowie des bahnbetrieblichen IP-Netzes gewonnen werden.

Ein DSTW besteht aus folgenden Teilsystemen:

- der **Zentraleinheit** (ESTW-ZE), welche die Fahrstraßenlogik realisiert,
- dem **Maintenance & Data Management** (MDM), welche Projektierungsdaten an die übrigen Teilsysteme verteilt und deren Diagnosedaten sammelt und weitergibt,
- den Teilsystemen **Lichtsignal**, welche Lichtsignale, PZB-Magnete, Balisen und Fahrtanzeiger steuern und überwachen,
- den Teilsystemen **Weiche**, welche Weichen, Gleissperren und oder Kreuzungen mit beweglichen Herzstückspitzen steuern und überwachen,
- den **Az-Systemen**, welche Achszählpunkte überwachen,
- und dem **Ein- und Ausgabesystem**, welches Umsysteme mit Kontaktschnittstelle steuert und überwacht, bspw. Schlüsselsperren, Helligkeitssensoren für Tag-/Nacht-Umschaltung und Nahbedienstellen.

Die folgenden mit dem Stellwerk kommunizierenden Umsysteme gehören formal nicht zum DSTW, sollen aber der Vollständigkeit halber erwähnt werden. Darunter sind einige, die in unveränderter Form auch schon in ESTW Bauformen zum Einsatz kamen.

- Integriertes Leit- und Bediensystem (iLBS): Ermöglicht Bedienung DSTW von beliebigem Standort aus
- Zugnummernmeldeanlage (unverändert ggü. ESTW)
- Zuglenkung (unverändert ggü. ESTW)
- Dokumentationssystem (unverändert ggü. ESTW)
- ETCS-Z: ETCS-Zentrale (RBC)

Die Kommunikation zwischen DSTW und den Teil- und Umsystemen erfolgt mittels des bb-IP-Netzes. Kernstück dessen sind zwei WAN-Netze[2] [WGC21]. Das Regionalnetz verbindet die betriebliche Ebene mit den Technikstandorten (TSO), bestehend aus ESTW-ZE und MDM, während das Zugangsnetz diese mit den Gleisfeldkonzentratoren (GFK) verbindet. An die GFK angeschlossen sind die Object Controller (OC), zur Steuerung und Überwachung der Gleisfeldelemente. Durch diese erfolgt ebenfalls die Stromversorgung. Je nach Bauart und Situation befinden sich die OC im Feldelement-Anschlusskasten (FEAK) bzw. Feldelement-Anschlussschrank (FEAS) (vgl. Abb. 4.1).

[2] Kernnetz und Regionalnetz wurden vereinfacht

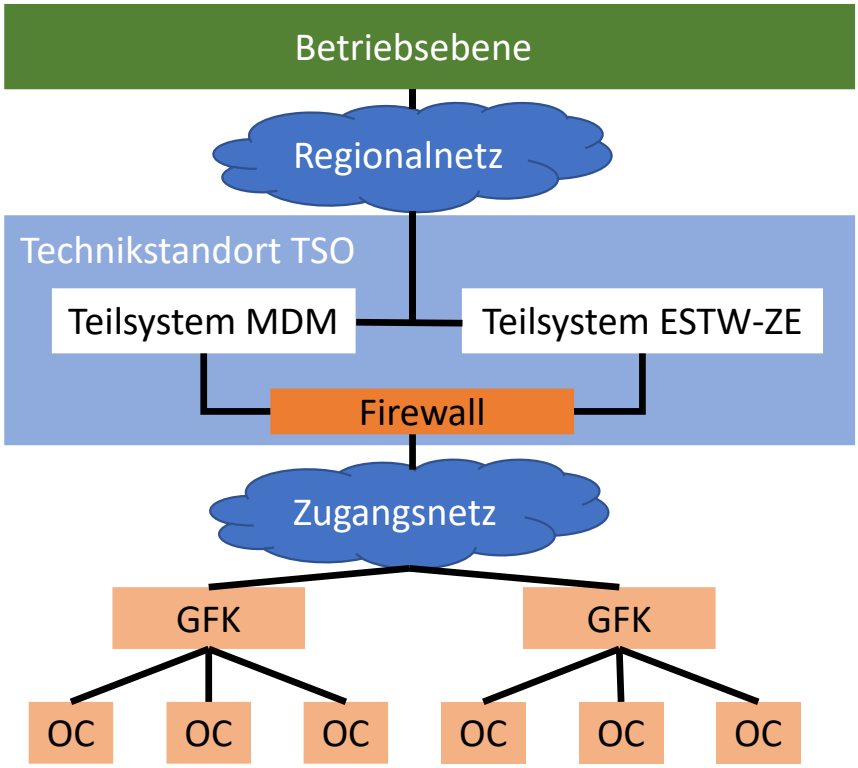

Abb. 4.1: Technische Architektur des DSTW, eigene Abb. nach [DSD21] und [WGC21]

Das bb-IP-Netz selbst besteht aus zwei vollständig redundanten physischen Netzwerkebenen, bezeichnet als Blau und Grau. Jeder TSO sowie jeder einzelne OC verfügt über jeweils einen Netzwerkadapter für die blaue und graue Netzwerkebene. Innerhalb des GFK befinden sich für jeden OC zwei Layer-2-Switches, welche diesen mit der jeweiligen Netzwerkebene verbinden.

Zwischen Switch und OC sind zwei Cyber-Security Komponenten angeordnet, ein VPN-Modul und eine Crypto-Box (vgl. Anhang B). Als Kommunikationsprotokoll für die Verbindung der DSTW-Komponenten untereinander wurde laut EULYNX Spezifikation das Rail Safe Application Protocol (RaSTA) gewählt. Der OC verfügt über eine Funktion zur Interpretation der RaSTA Nachrichten. Eine genaue Beschreibung von RaSTA ist in Kapitel 7.2 zu finden. Je nach gesteuertem und überwachtem Gleisfeldelement verfügt der OC außerdem über eine Verbindung zu diesem, im Fall eines Weichen-Controllers ist dies beispielsweise ein 4-Pol- oder non-4-Pol-Anschluss.

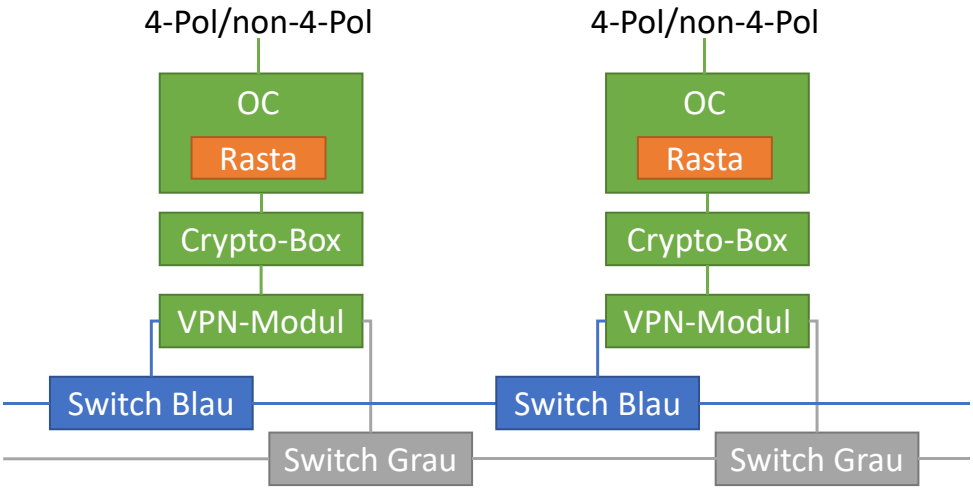

Abb. 4.2: schematische Netzwerkarchitektur Feldelement-Anschlussschrank, Darstellung nach Weber [EBA]

4.1 Messungen Netzwerkparameter bb-IP-Netz

Um die Netzwerkkonditionen im bb-IP-Netz der DB Netz AG abschätzen zu können, wurde per Email um Informationen zu bereits gebauten DSTW-Abschnitten gebeten. Petros Matios, Systemverantwortung Zentralsysteme der DB Netz AG, stellte Messungen zwischen folgenden Betriebsstellen bereit (vgl. Anhang C):

- MDTX: TSO Donauwörth
- MMTG: GFK in Mertingen
- MMEI: GFK in Meitingen

Paketgröße (Bytes)	**Latenz (us)**			
	MMTX - MMEI		**MMTX - MMTG**	
	blau	**grau**	**blau**	**grau**
40	514,01	921,45	340,96	1.094,92
64	526,31	934,45	351,86	1.106,88
128	562,15	967,73	386,86	1.144,25
256	624,77	1.027,69	447,08	1.206,24
512	737,41	1.135,13	555,50	1.322,98
1024	947,45	1.337,25	759,13	1.533,93
1280	1.051,69	1.438,11	858,29	1.635,00
1300	1.061,65	1.447,94	868,40	1.644,10
1500	Fehler	Fehler	Fehler	Fehler

Tab. 4.1: Erweitertes RFC 2544 – Testergebnisse Latenz, Mail von Zentralsysteme DB Netz AG vom 19.09.2022 (vgl. Anhang C)

Paketgröße (Bytes)	Paketfehlerrate (%)			
	MMTX - MMEI		MMTX - MMTG	
	blau	grau	blau	grau
40	0,0	0,0	0,0	0,0
64	0,0	0,0	0,0	0,0
128	0,0	0,0	0,0	0,0
256	0,0	0,0	0,0	0,0
512	0,0	0,0	0,0	0,0
1024	0,0	0,0	0,0	0,0
1280	0,0	0,0	0,0	0,0
1300	0,0	0,0	0,0	0,0
1500	100,0	100,0	100,0	100,0

Tab. 4.2: Erweitertes RFC 2544 – Testergebnisse Paketfehlerrate, Mail von Zentralsysteme DB Netz AG vom 19.09.2022 (vgl. Anhang C)

Latenzen und Paketfehlerraten wurden nach dem in RFC 2544 [INT99] beschriebenen Verfahren gemessen. Genutzt werden folgende Definitionen nach RFC 1242 [BRA91]:

Latenz ist als die Zeit festgelegt, die zwischen dem Verlassen des letzten Bits eines zu versendenden Frames am Port des Sendergerätes und dem Erscheinen des ersten Bits desselben Frames am Port des Empfängergerät liegt.
Paketfehlerrate ist der Prozentteil an Frames, welche von einem Netzwerkgerät weitergeleitet hätten werden müssen, aber aufgrund fehlender Ressourcen nicht weitergeleitet wurden.

Die Spalte der Latenzen ist in den per E-Mail zugesandten Messergebnissen als „Latenz RTD“ betitelt. Nach RFC 2544 wird mit dem Verfahren jedoch nicht die round-trip-time (RTD) gemessen und die Werte wurden entsprechend RFC 1242 interpretiert, so wie oben definiert.
Die Werte wurden als Ausgangspunkt für einige der für die Validierung der im Rahmen der vorliegenden Arbeit entwickelten Evaluationsumgebung durchgeführten Messungen benutzt (vgl. Kapitel 8.2).

In den Messungen waren außerdem Informationen über die Eigenschaften des gemessenen Netzes enthalten. So sind beide Netzwerkebenen als Leitungen mit einer Datenrate von jeweils 1000Mbit im Vollduplex[3]-Modus angegeben.

[3] Netzwerkverkehr in beide Richtungen

4.2 Zulassung neuentwickelter Systeme im Schienenverkehr

Ein in der Literatur immer wieder hervorgehobenes Problem liegt in der Zulassung der neuentwickelten Systeme [LIS20], [DOP18], [FRI21]. Die Inbetriebnahme von Bestandteilen des bundeseigenen Eisenbahnsystems, zu dem etwa die Anlagen der DB Netz AG zählen, erfordert eine Inbetriebnahmegenehmigung des Eisenbahn-Bundesamtes. Die Erteilung dieser Genehmigung erfolgt seit dem 11.08.2018 unter Anwendung der Eisenbahn-Inbetriebnahmegenehmigungsverordnung (EIGV). Maßgaben zur Harmonisierung von Genehmigungsprozessen der EU-Richtlinien des Europäischen Parlaments und Rates (2008/57/EG und 2014/106/EU) werden damit in nationales Recht umgesetzt.

Zur Anwendung der EIGV hat das EBA Verwaltungsvorschriften erlassen, von denen die Verwaltungsvorschrift VV IBG Infrastruktur für die Teilsysteme Infrastruktur, Energie, streckenseitige Zugsteuerung, Zugsicherung, Signalgebung sowie für die übrige Eisenbahninfrastruktur und damit auch für DSTW und EULYNX-konforme Gleisfeldelemente zum Tragen kommt. Weiterhin zu beachten ist die Verwaltungsvorschrift für die Überwachung der Erstellung von Signal-, Telekommunikations- und Elektrotechnischen Anlagen VV BAU-STE in der aktuellen Version 5.1 [TRI20].

Die oben genannten regulatorischen Grundlagen wurden das erste Mal unter anderem bei der Inbetriebnahme des DSTW Warnemünde angewandt, wie von Lisker und Kanis (2020) beschrieben [LIS20]. Dabei lohnt sich die Erwähnung, dass der Zulassungsprozess in dieser Übergangszeit zwischen den Stellwerksgenerationen laut den Autoren ein Übermaß an Kommunikation zwischen EBA, der DB Netz AG und Herstellern erfordere. So nähme die Klärung offener Fragen in Bezug auf die zu verwendenden Dokumente, die Ausgestaltung der neuen Rollen und die Struktur des Genehmigungsgegenstandes den größten Teil der verfügbaren Zeit ein. Demgegenüber gestellt seien jedoch laut den Autoren massive Zeitgewinne bei der behördlichen Tätigkeit.

Die vollständige Konformität mit den grundlegenden Regeln zur Standardisierung der Eisenbahnsysteme in der Europäischen Union, den „Technical Specifications for Interoperability“ (TSI) ist laut §5 VV IBG Infrastruktur Zulassungsvoraussetzung[4] [VVI20]. Obwohl in den TSI ein Abschnitt zur Leit- und Sicherungstechnik (TSI-CCS) existiert, enthält dieser lediglich Spezifikationen bezüglich des Europäischen Zugbeeinflussungssystems „European Rail Traffic Management System“ (ERTMS) mit seinen Bestandteilen European Train Control System (ETCS) und Global System for Mobiles -Railway (GSM-R). Zu einer Standardisierung der Stellwerke bzw. Stellwerksschnittstellen kam es im Rahmen der TSI nicht, auch wenn ein Projekt unter dem Stichwort „Eurointerlocking“ gestartet wurde [SIE18]. Können die TSI nicht angewendet werden, wie im Fall der Stellwerkstechnik, sieht die EIGV einen reduzierten Prüfumfang vor, der unter anderem den Nachweis technischer Kompatibilität und sicherer Integration enthält. In jedem Fall muss die Einhaltung der Vorschriften in Form von Prüfbescheinigungen durch Prüfsachverständige bestätigt werden, nebst eines

[4] §5 EIVG erlaubt Ausnahmen in ungewöhnlichen Fällen (EIGV20).

Sicherheitsbewertungsberichtes von einer unabhängigen Bewertungsstelle im Fall von sicherheitsrelevanten und signifikanten Änderungen, zu denen die Stellwerkstechnik erwiesenermaßen zählt.

Im Zentrum der Zulassung gemäß §8 VV IBG Infrastruktur steht die Beachtung von technischen Vorschriften wie IEC-, DIN-, EN- und VDE-Normen, Planungsrichtlinien der Eisenbahnen, Fachmitteilungen des EBA und vielen weiteren. Das Verfahren, mittels dessen die Eignung von Systemen und Komponenten bewertet und nachgewiesen wird, ist ebenfalls festgelegt und wird von der VV BAU-STE bestimmt. Zur Anwendung kommt die Verwaltungsvorschrift Neue Typzulassung (VV NTZ), welche vorgibt, wie der durch die EN 50126-1 des Europäischen Komitees für elektrotechnische Normung (CENELEC) bestimmte RAMS-Lebenszyklus im Einzelnen durchgeführt werden muss.

Die bei der Zulassung von digitalen Stellwerken und Gleisfeldelementen zu beachtenden Normen sind hauptsächlich folgende CENELEC-Normen:

- EN 50126-1: Bahnanwendungen - Spezifikation und Nachweis von Zuverlässigkeit, Verfügbarkeit, Instandhaltbarkeit und Sicherheit (RAMS) - Teil 1: Generischer RAMS-Prozess
- EN 50128: Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Software für Eisenbahnsteuerungs- und Überwachungssysteme
- EN 50129: Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Sicherheitsbezogene elektronische Systeme für Signaltechnik

Für die geplante Nutzung von und Konformität mit den EULYNX-Spezifikationen im Schienennetz der DB Netz AG gibt es keine rechtliche Verpflichtung [EUL17]. Die diesbezüglichen Bestrebungen finden somit aus freien Stücken statt und können rechtlich im weitesten Sinne durch die in der Einleitung bereits erwähnten Ziele für den Schienenpersonen- und -güterverkehr des Klimaschutzprogramms 2030 begründet werden [KLI30], oder durch §7 der Bundeshaushaltsordnung, „Wirtschaftlichkeit und Sparsamkeit, Kosten- und Leistungsrechnung“, nach dem die Grundsätze der Wirtschaftlichkeit und Sparsamkeit bei der Erfüllung von staatlichen Aufgaben und öffentlichen Zwecken zu beachten sind [BHO22].

Ebenso gibt es keine offizielle Behörde oder Organisation, welche die EULYNX Konformität überprüfen und zertifizieren kann, wie bspw. im Falle von ETCS. Für Labortests von ETCS-Komponenten existiert das akkreditierte Test- und Simulationslabor im Deutschen Zentrum für Luft- und Raumfahrt (DLR) in Braunschweig mit der „Railway Simulation and Testing“ (RailSiTe), welches Prüfeinrichtungen für ETCS nach SUBSET 094 bereitstellt [TRI20].

Der Zeitpunkt für Tests im Rahmen der Zulassung ist wie o. g. nach EN 50126 mittels des V-Modells geregelt, eines Entwicklungsmodells für Gesamtsysteme, und kann

dort eingesehen werden. Nach Han in „ETCS in Deutschland“ (2020) führen Hersteller während der V-Modell Phase SW/HW-Anforderungen Tests im Labor durch, um die Funktionalitäten der Software und Hardware zu verifizieren [TRI20].
Wie im nächsten Kapitel beschrieben, werden durch die EULYNX Spezifikationen jedoch neue Herausforderungen und Chancen für die effiziente Abwicklung des Test- und Zulassungsprozesses hervorgerufen. Durch diese ist nicht immer eine klare Zuordnung zu den Entwicklungsphasen des V-Modells möglich, da sich Phasen überschneiden oder wiederholt durchlaufen werden müssen [BIE22]. Das nächste Kapitel enthält daher die Untersuchung eines alternativen Konzepts aus der Literatur.

5 Evaluationsumgebung

In den nachfolgenden Kapiteln werden alle Entscheidungsschritte zur Entwicklung der Evaluationsumgebung und deren Aufbau hergeleitet und beschrieben.

5.1 Testverfahren „modulares, hierarchisches Testen“

Da die bisherigen Vorgehensweisen der Zulassung von Stellwerken von anderen Voraussetzungen ausgehen, sollten die diesbezüglichen Regelwerke im Zuge der technologischen Erneuerung ebenfalls erneuert werden. In einem Beitrag mit dem Titel „Effizientes Testen modularisierter und standardisierter Stellwerkskomponenten“ in der Zeitschrift SIGNAL+DRAHT werden von Caspar et al. (2018) [CAS18] die veränderten Bedingungen diskutiert, welchen der Sicherheitsnachweis einer modularen Stellwerksarchitektur und damit auch deren Zulassung unterliegt. Zur Verifikation und Validierung (V&V) softwarelastiger Systeme müssen diese getestet werden. Bei den monolithischen ESTWs der Vergangenheit lag die Verantwortung für die Planung und Durchführung der V&V-Aktivitäten bei den Herstellern und wurde in herstellereigenen Testlaboren abgewickelt. Durch die Zusammensetzung des Stellwerks aus generischen Komponenten verschiedener Hersteller müsse sich dieses Verfahren zwangläufig ändern. Der Test des Gesamtsystems sei erst nach vollständiger Zusammenführung aller Teilsysteme und Errichtung der Anlage im Feld möglich, was zu diesem Zeitpunkt zeitaufwendig und teuer sei sowie den Bahnbetrieb behindern würde.

Die Autoren schlagen ein mehrphasiges Testverfahren vor, welches sie „modulares, hierarchisches Testen“ nennen. Dabei werden Testaktivitäten in Module aus vollständig implementierten Komponenten strukturiert, an denen definierte Eigenschaften demonstriert werden können. Im Sinne von EULYNX besteht das simpelste Modul aus genau einem Teilsystem. Auf der nächsthöheren hierarchischen Ebene befinden sich zwei Teilsysteme, welche über eine Standardschnittstelle (SCI) miteinander kommunizieren. Das komplette System inklusive der situationsspezifischen Projektierungsinformationen bildet die höchste Ebene.

In **Phase 1** wird die Konformität der einzelnen Teilsysteme mit funktionalen und nicht-funktionalen Anforderungen der Lasten- und Pflichtenheften getestet. Dies kann weiterhin selbstsuffizient bei den Herstellern erfolgen, unter effizienter Ausnutzung der Testinfrastruktur durch Automatisierung und geringem Bedarf zur Abstimmung.
Phase 2 ist der Interoperabilitätstest der standardisierten Schnittstellen. Das getestete Modul besteht aus dem Stellwerkskern und dem betreffenden Teilsystem. Da dies unter Umständen mehrere Hersteller betreffen kann, ist Kommunikation sowie ggf. die Zusammenarbeit und Kombination von zwei Laboren notwendig. Das Ziel dieser Phase ist der Nachweis der Kompatibilität zwischen zwei kommunizierenden Produkten (Teilsystem A von Hersteller X, Teilsystem B von Hersteller Y). Gegenüber den generischen Tests der Phasen 1 und 2 ist **Phase 3** Teil der spezifischen Tests, die für ein Projekt durchgeführt werden. Die Anordnung der beteiligten

Systeme muss der Zielanlage bereits entsprechen. Während die Autoren einerseits sagen, dass alle Instanzen der Teilsysteme sowie das Übertragungssystem bereits physisch vorhanden sind müssen, lassen sie auf der anderen Seite aber die Möglichkeit der Nutzung von emulierten oder simulierten Komponenten für Teile der Anlage offen, falls dies nicht bewerkstelligt werden kann. Das Ende des Systemtests markiert die abgeschlossene Prüfung der Gesamtsystemanforderungen. Die Konfrontation des Systems mit betrieblichen Bedingungen und Abläufen ist dabei auch ein Test der Korrektheit der Anforderungsspezifikationen.
Phase 4 ist die durch gesetzliche Zulassungsprozesse meist vorgeschriebene Betriebserprobung und wird an dieser Stelle nur der Vollständigkeit halber erwähnt.

Die in dieser Arbeit diskutierte Evaluierungsumgebung stellt ein Werkzeug dar, auf dessen Grundlage Phasen des von Caspar et al. (2018) vorgeschlagenen Testverfahrens „modulares, hierarchisches Testen“ abgewickelt werden können, insbesondere Phase 2. Im nächsten Kapitel sollen die Anforderungen entwickelt werden, die sich daraus ergeben.

5.2 Anforderungsanalyse für die Evaluationsumgebung

Ausgehend von den Zulassungsvoraussetzungen, auf deren Konformität die Komponenten getestet werden sollen, wurden folgende Anforderungen an die zu entwickelnde Evaluationsumgebung abgeleitet:

- Konfigurierbare Simulationsumgebung für alle je nach Experiment beteiligten Komponenten
    - Möglichkeit der Nutzung von Hardware-Komponenten bzw. virtualisierten Komponenten
    - Gegebenenfalls Konfiguration der Leistungsfähigkeit der (simulierten) Hardware für Aussagen über Leistungsanforderungen der Software
- Automatische Provisionierung
- Austauschbare Komponenten
    - Stellwerk
    - Objectcontroller
    - Kommunikationsprotokolle bzw. Implementierungen
- Festlegung der Netzwerkparameter bzw. Beeinträchtigung der Netzwerkleistung
    - Paketverlustrate
    - Paketfehlerrate
    - Latenzen
- Automatische Testdurchführung
    - Formale Definition der Tests
- Mess- und Analyseeinrichtung
    - Protokollierung
    - Protokollauswertung

    - Graphische Darstellung
- Optionale graphische Konfigurationsoberfläche
    - Konfigurierbarkeit aller für das Experiment relevanten Parameter in einer graphischen Oberfläche zwecks Benutzerfreundlichkeit

Die Eingabe in das Evaluationsprogramm zum Starten eines Systemtests entsprechend Phase 3 des modularen hierarchischen Testens bzw. der V-Modell Phase SW/HW-Anforderungen sollte aus einer formellen Beschreibung der am Test beteiligten Teilsysteme (Komponentenplan), einer formellen Beschreibung des zu absolvierenden Testablaufs (Testplan) und der zu testenden Software bzw. Hardware selbst bestehen. Die Ausgabe hängt stark vom Ziel des Testes und dem Testsubjekt ab, aber allgemein gesagt sollte eine Übersicht über den Erfolg der Tests sowie über observierte Leistungsparameter generiert werden, gegebenenfalls inklusive einer graphischen Darstellung.

5.3 Technischer Hintergrund der Evaluationsumgebung

Strukturell ist ein DSTW eine miteinander über das bb-IP-Netz verbundene Ansammlung von Computern mit funktionsspezifischer Hard- und Software und lässt sich durch eine der Zielarchitektur entsprechende Anordnung von Netzwerkkomponenten und physischen Computern bzw. virtualisierten Computern realitätsgetreu simulieren. Voraussetzung dafür ist lediglich die Projektierung von funktionsbeeinflussenden Bedingungen wie der Latenz des Netzwerks und der Leistung der Hardware [BEI221].

Laut EULYNX Spezifikation wird jedes Gleisfeldelement über einen einzelnen Object Controller (OC) angesteuert. Diese OC können an die Evaluationsumgebung entweder physisch angeschlossen oder virtuell generiert werden. Der Vorteil, den eine Möglichkeit der Verwendung von virtuellen OC mit sich bringt, ist offensichtlich: In Abhängigkeit der Anforderungen an das durchzuführende Experiment wäre es unpraktikabel, für die Entsprechung der Komponenten in Hardware zu sorgen, bzw. würde es die Bandbreite der möglichen Experimente einschränken oder Experimente unnötig verkomplizieren (vgl. [BEI221]).

Durch **Virtualisierung** ist es also möglich, alle oder einige der teilnehmenden OC parallel auf der Hardware eines Gastgeber-Systems der Evaluationsumgebung auszuführen. Die als Virtual Machines (VM) bezeichneten virtualisierten Computer teilen sich die Ressourcen des Gastgeber-Systems, ohne dass das Betriebssystem oder die Software der OC davon beeinflusst werden.

Zwei Arten von Virtualisierung sind möglich, Hypervisor-basierte und Container-basierte, mit spezifischen Vor- und Nachteilen. Diese wurden innerhalb der vorliegenden Arbeit im Rahmen der Entscheidung für oder wider der Nutzung eines bereits verfügbaren Testbetts und schließlich in der tatsächlichen Entwicklung der Evaluationsumgebung erwogen. Daher sollen die Eigenschaften beider Methoden kurz in Anlehnung an Eder (2016) diskutiert werden [EDE16]:

Virtualisierung durch einen **Hypervisor** emuliert ein prinzipiell vollständig separates „Guest"-Computersystem neben dem „Host" genannten Gastgebersystem[5]. Die Nutzung der Hardware durch die Software eines Computersystems (meist in Form eines Betriebssystems) wird über Treiber bewerkstelligt, welche Schnittstellen für die Anwendungsprogramme zur Verfügung stellen. So werden beispielsweise die Tastenanschläge auf einer Tastatur durch einen entsprechenden Treiber an das Betriebssystem übertragen, welches diese Informationen an Anwendungsprogramme weiterleitet, die diese zur Funktion benötigen. Der Hypervisor repliziert nun ein Computersystem inklusive dieser Treiber. In dem Guest-Computersystem kann auch ein anderes Betriebssystem ausgeführt werden als auf dem Host. Er erlaubt außerdem eine feingranulare Einstellung der dem virtuellen System zur Verfügung gestellten Leistung.

Virtualisierung auf **Container**-Basis kreiert eine isolierte Umgebung für Prozesse, in welcher die Hardware über das Gastgeber-Betriebssystems vermittelt wird. Container bieten daher Vorteile bei der Größe des Images und der Startzeit, da keine Treiber und kein Kernel inkludiert werden, beziehungsweise kein vollständiges Betriebssystem hochgefahren werden muss. Eine populäre Anwendung von Containern liegt in der Bereitstellung von Entwicklungsumgebungen (vgl. Kapitel 6). Oft sind für die Kompilierung von Programmen viele verschiedene Bibliotheken und Entwicklungswerkzeuge nötig, welche aufwendig installiert und eingerichtet sowie unter Umständen auch wieder entfernt werden müssen. Mit einem für den geforderten Zweck ausgestatteten Container lässt sich eine abgeschlossene und modulare Umgebung schaffen, in der die Softwareentwicklung erfolgen kann, beziehungsweise eine für eine bestimmte Umgebung kompilierte Software unkompliziert ausgeführt werden kann.

Aufgrund der beschriebenen Vor- und Nachteile entschied sich der Verfasser in der vorliegenden Arbeit, wie von Beilharz et al. (2021) als für Tests realitätsnäher empfohlen, für die Virtualisierung der OCs für Hypervisor-basierte VMs. Die in Kapitel 7.3.2 beschriebenen Module innerhalb der OCs wurden in der Evaluationsumgebung in Form von virtualisierten Containern innerhalb der VMs umgesetzt und sind damit sehr modular und austauschbar. Die Nutzung der Virtualisierung innerhalb der VM ist aber nicht verpflichtend und je nach Layout des Tests und der zu testenden Software kann diese auch unmittelbar auf der VM ausgeführt werden.

5.4 OSI-Schichtenmodell der Rechnerkommunikation

Die für das Verständnis der Arbeitsweise der DSTW nötigen Grundlagen für die Kommunikation von Rechnersystemen über das IP-Protokoll sollen anhand des Open Systems Interconnection Reference Model (OSI-Modell) in Kurzfassung mit Bezug auf Sobe (2012), Wieners (2021) und Balchunas (2012) beschrieben werden [SOB12], [WIE21], [BAL12]:

[5] Die Aufteilung der Hypervisor-Lösungen in Typ-1 und Typ-2 kann für die Ansprüche dieser Arbeit vernachlässigt werden

Damit nicht jede Netzwerkarchitektur, vereinfacht gesagt, „ihr eigenes Süppchen braut“ und individuelle Standards und Verfahrensweisen für die Nutzung voraussetzt, sind Netze wie das IP-Netz als mehrere übereinanderliegende Schichten aufgebaut. Jede Schicht bietet der darüberliegenden einen sogenannten Dienst an, auf den über eine Schnittstelle zugegriffen werden kann. Auf diese Art und Weise bleibt die genaue Implementierung der Schicht verborgen, was die Komplexität reduziert und eine modulare Entwicklung ermöglicht.

Vor dem tatsächlichen Verlassen einer Nachricht beim Sender wird sie von einer Schicht zur anderen weitergereicht. Dies ist vergleichbar mit Worten, die auf ein Blatt Papier geschrieben und als Brief versendet werden sollen. Jede Schicht verpackt den Brief in einen von ihr zu interpretierenden Umschlag, der von der jeweils entsprechenden Schicht auf der Empfängerseite wieder ausgepackt werden und eine Schicht hochgereicht werden kann. Die Regeln, nach denen der Inhalt und der Umschlag, meist Header genannt, formatiert werden, sind in Protokollen vorgegeben. Ebenso wird in ihnen festgelegt, wie sich die Teilnehmer in bestimmten Situationen verhalten müssen, etwa bei Verlust einer Nachricht.

Im OSI-Modell gibt es sieben dieser Schichten. Jedoch können die oberen drei, Anwendungs-, Darstellungs-, und Sitzungsschicht, für die Zwecke dieser Arbeit zur einfachen Anwendungsschicht zusammengefasst werden, da Darstellungs- und Sitzungsschicht keine Aufgaben im DSTW übernehmen (vgl. Abb. 7.4).

Auf der **Anwendungsschicht** befindet sich die tatsächliche Anwendung, die mit einer Anwendung auf einem anderen Computersystem kommuniziert.

Darunter liegt die **Transportschicht**, welche der Nachricht den richtigen Absender- und Empfängerport zuweist, also dem Zielsystem hilft, die richtige Anwendung zu adressieren. Ein Port lässt sich mit einem Briefkasten vergleichen, den eine Anwendung durch die Schnittstelle zur Transportschicht aktivieren kann, um auf Briefe zu warten. Es gibt zwei Protokolle, welche die Ports eines Computersystems im Allgemeinen verwalten, das verbindungs-orientierte Transmission Control Protocol (TCP) und das verbindungslose User Datagram Protocol (UDP).
TCP verfügt über Features wie der Flusskontrolle, kann also mehr oder weniger Pakete anfordern, je nachdem, wie ausgelastet Leitung und Empfangscomputersystem sind, oder der Empfangsbestätigung von Nachrichten. Es ist damit sehr gut für die Übertragung von größeren Dateien geeignet, da die Geschwindigkeit der Übertragung die maximale Übertragungsrate erreichen, aber diese nicht überschreiten wird, und während der Übertragung verloren gegangene Pakete identifiziert und wiederholt werden können. Der in der Literatur als Overhead [MATOJ] bezeichnete zusätzliche Aufwand, der neben der eigentlichen Aufgabe anfällt, ist bei TCP zur Gewährleistung der beschriebenen Verbindungseigenschaften sehr groß.

UDP ist im Gegensatz dazu sehr kompakt und bietet außer der Adressierung von Absender- und Empfängerport lediglich Platz für eine Checksumme[6], um zu überprüfen, ob das Paket und der Header fehlerfrei übertragen wurden. Es ist also mit UDP nicht möglich, auf Ebene der Transportschicht festzustellen, ob die versendeten Nachrichten auch tatsächlich ankommen. Ist diese Information nötig, muss sie auf der Anwendungsschicht ermittelt werden.

Die **Vermittlungsschicht** (oder auch Network Layer) stellt eine logische Adressierung für die Ende-zu-Ende-Übertragung von Paketen bereit und regelt die Weiterleitung dieser zwischen verschiedenen Netzwerken. Die verwendeten Protokolle sind vielfältig, es soll sich aber im Rahmen dieser Arbeit auf die Familie des Internet Protocol (IP) beschränkt werden, genauer auf IPv4. Teilnehmenden Computern innerhalb des Netzes wird eine IP-Adresse zugewiesen. Diese ist vergleichbar mit Straße, Hausnummer und PLZ in der Briefzustellungsanalogie, um das Haus zu finden, an welchem die Transportschicht den richtigen Briefkasten auswählen kann. Computersysteme, die Pakete zwischen verschiedenen Netzen übertragen, werden Router genannt, oder auch Layer-3-Switches (zur Abgrenzung der Weiterleitung von Paketen innerhalb desselben Netzes durch Layer-2-Switches). Ihre Funktion erstreckt sich nur auf die unteren drei Schichten. Bei Empfang eines Paketes leiten sie dieses Anhand von Routingtabellen auf einer „Best-Effort"-Basis an einen dem Zielort näheren Knoten weiter. Das bedeutet, auch von IPv4 werden keine Garantien bezüglich der Zustellung gegeben.

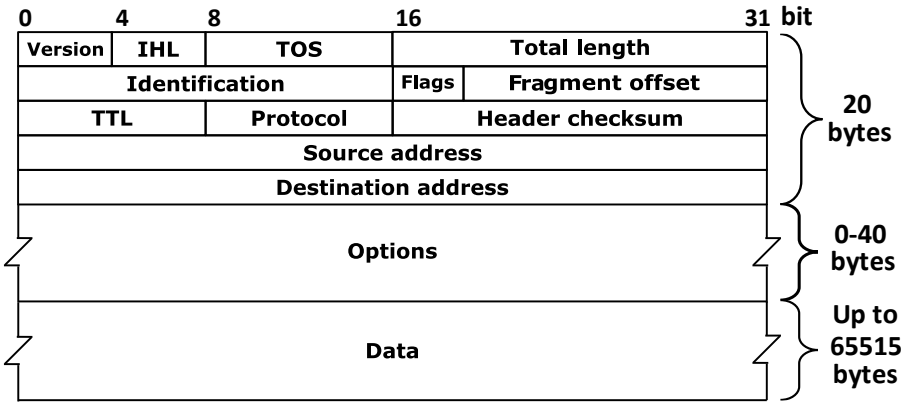

Abb. 5.1 IPv4 Header, Darstellung von [WIK22]

Durch die **Sicherungsschicht** (oder auch Data Link Layer) erfolgt der Zugriff von Geräten auf das physische Medium zur Informationsübertragung. Die Adressierung auf dieser Ebene erfolgt durch Media Access Control (MAC) Adressen, welche innerhalb desselben Layer-2-Netzwerks nur einmal vorkommen dürfen. Switches leiten die nun als sogenannte Frames verpackte und an ein spezifisches Gerät gerichtete Pakete innerhalb des Netzes weiter und merken sich die MAC-Adressen der mit ihnen verbundenen Geräte. Das beinahe ausschließlich genutzte Protocol auf dieser Schicht ist Ethernet.

[6] Fußnote: Die Nutzung dieser ist optional innerhalb der Version vier des Internet-Protokolls (IPv4) und verpflichtend bei Version sechs (IPv6)

Auf der **Bitübertragungsschicht** befinden sich die Methoden und Technologien zur Übertragung der rohen Bits auf dem physischen Medium. Dies können Kupferkabel, Glasfaserkabel oder auch eine Luftbrücke sein. Da der Ethernet-Standard Funktionen für sowohl Sicherungsschicht als auch Bitübertragung enthält, verschwimmen an dieser Stelle die Grenzen zwischen beiden. Auf die Bitübertragungsschicht soll nicht näher eingegangen werden, da diese für die vorliegende Aufgabenstellung nicht weiter relevant ist.

5.5 Testumgebungs-Software von dritten Herstellern

Da das Testen von softwaregestützten Systemen für Aussagen über deren RAMS-Eigenschaften so wichtig ist, werden auch für Internet-of-Things (IoT) Systeme bereits verschiedene Testumgebungen (Testbeds) von prominenten Herstellern angeboten, welche die dem Testen von IoT-Systemen eigenen Herausforderungen angepasst sind. Das Internet der Dinge bezeichnet das Konzept, mit jedem IT-Gerät jederzeit über eine Schnittstelle auf Basis des IP-Netzes kommunizieren zu können, d.h. Benachrichtigungen zu empfangen, Steuerungsbefehle zu senden oder Informationen abzurufen. Das DSTW mit seinen Bestandteilen ist in diesem Sinne eine Anwendung der IoT-Philosophie [WGC21].

Für den Entscheidungsprozess, ob die im Rahmen dieser Arbeit zu entwickelnde Evaluationsumgebung auf einer bestehenden Testumgebungs-Software basieren soll, wurde aus der Literatur beispielsweise das Paper von Beilharz et al. (2021), „Continuously Testing Distributed IoT Systems: An Overview of the State of the Art", zu Rate gezogen. Die Autoren bewerteten die populärsten IoT-Testumgebungen Ende 2021 nach verschiedenen Kriterien, darunter unter anderem die Untersuchung auf relevante Features wie skriptbare Testszenarios, die Unterstützung von sowohl virtualisierten Computern als auch von integrierter Hardware und die Möglichkeit der Fehlerinjektion in simulierte Netzwerke.

Trotz einiger auf dem Markt verfügbarer Testumgebungs-Software, wie beispielsweise Héctor [BEH19], wurde die Entscheidung getroffen, keine Testumgebung von Dritten als Ausgangsmaterial für die Evaluationsumgebung zu verwenden. Héctor wurde insbesondere deshalb ausgeschlossen, weil Bedenken wegen des Debuggings von innerhalb der VMs laufenden Docker-Container bestanden. Das Programm war zum Zeitpunkt der Entscheidungsfindung außerdem seit vier Jahren nicht mehr aktiv weiterentwickelt worden und war nicht auf dem erforderlichen technischen Stand.
Durch die Nutzung von bereits vorgefertigten Strukturen und Funktionen kann zwar zumeist Arbeit eingespart werden, jedoch bringt dies auch einen Verlust an Flexibilität mit sich und ist mit dem Risiko verbunden, dass bei einem komplexen Anforderungsbild wie dem der Evaluationsumgebung ein Verhalten des Systems benötigt wird, was von den Entwicklern nicht vorhergesehen wurde. So kann für die Umsetzung eines essenziellen Details ein aufwendiger Umweg an den eigentlich Komplexitäts-reduzierenden Funktionen des Frameworks vorbei nötig werden, welcher die

ursprünglich gesparte Zeit aufbraucht. Der Nutzen des Frameworks muss darum die eingebüßte Flexibilität offensichtlich aufwiegen.
Der Verfasser traf daher die Entscheidung, unter Berücksichtigung dieser Faktoren und des absehbar beherrschbaren Aufwandes für die neuen bzw. ergänzenden Komponenten, die Evaluationsumgebung im Rahmen der vorliegenden Arbeit selbständig zu entwickeln. Der Aufbau dieser ist in den folgenden Kapiteln beschrieben.

6 Aufbau der Evaluationsumgebung

Für die Erfüllung jeder der im Kapitel 5.2 beschriebenen Anforderungen wurden verschiedene Methoden und Technologien erwogen und schließlich für die Evaluationsumgebung (DSTW-Sim) ausgewählt, welche im Folgenden beschrieben sind.
Die im Kapitel 4 beschriebene Architektur des EULYNX-konformen DSTW wurde in der Evaluationsumgebung möglichst genau abgebildet. Jeder OC sowie das Stellwerk sind durch eine eigene VM dargestellt. Die beiden Switches pro Teilsystem für jeweils blaue und graue Netzwerkebene wurden ebenfalls als VM abgebildet, auf welcher eine Software die Aufgabe eines Switches übernimmt (vgl. Abb. 6.1: Architektur von DSTW-Sim).

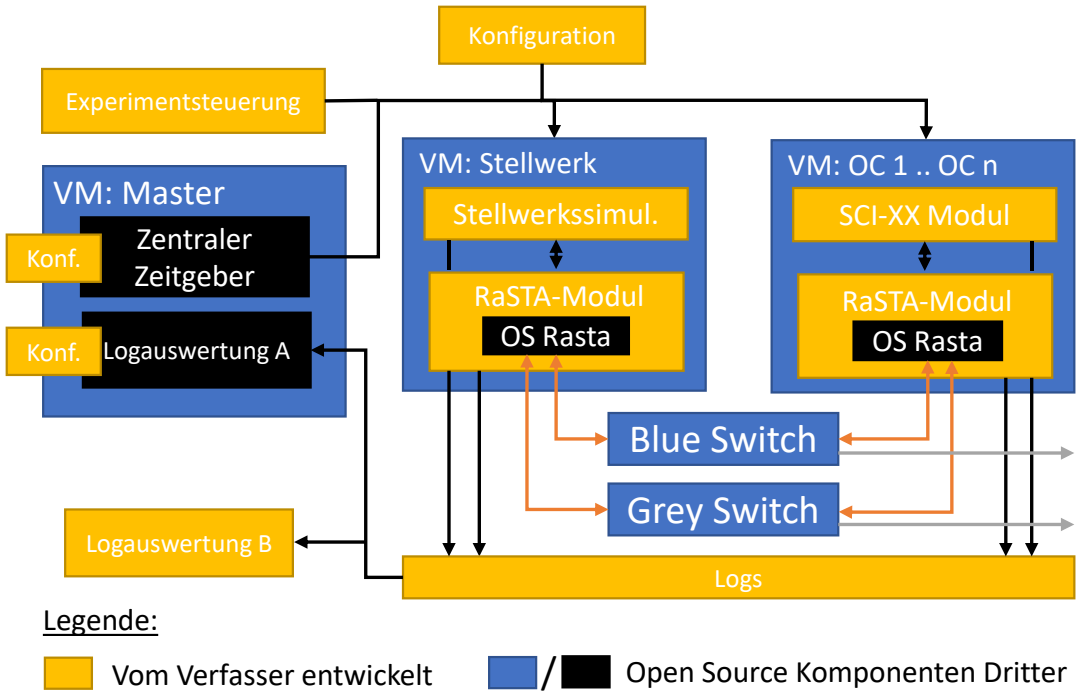

Abb. 6.1: Architektur von DSTW-Sim

Für DSTW-Sim wurde die Open Source Software mit dem Namen OpenVSwitch für diesen Zweck gewählt. OpenVSwitch ist eine schlanke Software zur Emulation von Switches mit einem breiten Spektrum an Features für Enterprise-Kunden, von denen im Rahmen von DSTW-Sim nur die Funktion des Überbrückens oder „bridging“ von zwei oder mehreren Netzwerken genutzt wird.

Für die Diagnostik der über die Netzwerke versandten Pakete kommt der Industriestandard der Netzwerkprotokoll-Analyseprogramme **Wireshark** zum Einsatz. Dessen zur Nutzung per Kommandozeile vorgesehene Version wurde zusammen mit OpenVSwitch auf den VMs installiert.

Als Hypervisor, auf dem die VMs ausgeführt werden, wurde **VirtualBox** gewählt. Andere Enterprise-Hypervisor, deren Nutzung erwogen wurde, sind bspw. Hyper-V von Microsoft und VMware Workstation vom VMware. Jedoch ist VirtualBox der einzige

Enterprise-Hypervisor mit allen benötigten Features, der als Open Source Software frei zugänglich ist [VIR22].
VirtualBox verfügt unter anderem über das Feature der Erstellung sogenannter interner Netze, die unter Angabe des Netznamens nur ausgewählte VMs miteinander verbinden können, was für den Nachbau der DSTW-Architektur nötig war.

Die manuelle Provisionierung aller virtuellen VMs, wie es etwa für die Durchführung eines Systemtests (analog zum Phase 3-Test des modularen, hierarchischen Testens) nötig ist, wäre je nach Größe des zu projektierenden Stellbereiches sehr aufwendig, weshalb die Anforderung an die Automatisierung dieses Prozesses formuliert wurde. Provisionierung bezeichnet dabei die erstmalige Erstellung und Vorbereitung der VM, d.h. die Installation aller nötigen Software, das Kopieren aller Daten und die Konfiguration der virtuellen Netzwerkkarten und Hardware. In DSTW-Sim wurde die Open Source Software zur Provisionierung und Verwaltung von virtuellen Maschinen **Vagrant** benutzt. Vagrant erstellt für alle Guests einen Ordner, über den mit dem Host Dateien ausgetauscht werden können. Als Eingabe wird eine Konfigurationsdatei benötigt, nach Vorgabe welcher die VMs erstellt werden. Die Konfiguration wird in der Syntax der Ruby-Programmiersprache dargestellt. Folgende Informationen werden für die Erstellung einer VM benötigt:

- Name des auszuführenden Betriebssystems
- Virtuelle Netzwerkadapter und angeschlossene Netzwerke mit deren Einstellungen
- Gegebenenfalls Größe von zugewiesenem Arbeitsspeicher und Prozessorleistung

In Abb. 6.2 ist beispielhaft der Ausschnitt aus der Konfigurationsdatei zu sehen, der das Stellwerk betrifft. Es wird das Betriebssystem Ubuntu 18.04 LTS mit dem Codenamen „Bionic Beaver“ verwendet. Darauf folgen die Netzwerke zum Anschluss an die Switches zur Verbindung mit dem simulierten bb-IP-Netz „blue_interlocking“ und „grey_interlocking“ sowie die Verbindung zum Management-Netz. Wichtig ist außerdem das Provisionierungsskript, für das Stellwerk heißt dieses „provisionInterlocking.sh“, welches die zu testende Software nach der erstmaligen Erstellung der VM auf den virtuellen Speicher kopiert, sowie die Software zur Synchronisation der Uhren installiert.
Die Anweisungen für die OC und Switches unterscheiden sich nur unwesentlich.

```
config.vm.define "interlocking" do |interlocking|
    interlocking.vm.box = "hashicorp/bionic64"
    interlocking.vm.network "private_network", ip: "10.152.0.2", netmask: "255.255.255.0", virtualbox__intnet: "blue_interlocking", mac: "00248109d239"
    interlocking.vm.network "private_network", ip: "10.152.1.2", netmask: "255.255.255.0", virtualbox__intnet: "grey_interlocking", mac: "002481024237"
    interlocking.vm.network "private_network", ip: "192.168.202.2", netmask: "255.255.128.0", virtualbox__intnet: "management", mac: "00248104987f"
    interlocking.vm.provider "virtualbox" do |vb|
        vb.customize ["modifyvm", :id, "--nicpromisc2", "allow-all"]
        vb.customize ["modifyvm", :id, "--nicpromisc3", "allow-all"]
    end
    interlocking.vm.provision "shell", path: "provisionInterlocking.sh", args: "'interlocking'"
    interlocking.vm.provision :docker
    interlocking.vm.provision :docker_compose
end
```

Abb. 6.2: Ausschnitt der Konfigurationsdatei für DSTW-Sim

Die Konfigurationsdatei muss vom Nutzer nicht selbst geschrieben werden, sondern wird durch einen anderen Programmteil von DSTW-Sim in Form eines Python-Skripts (Komponentenskript) gemäß Anforderungsanalyse aus dem Komponentenplan erstellt. Für simple Manipulationen von Strings, wie es das Erstellen der Konfigurationsdatei ist, ist die Sprache Python sehr gut geeignet, weil sich mit ihr schnelle Ergebnisse mit wenig Code erzielen lassen.
Der Komponentenplan folgt der yaml-Syntax und sieht für einen Systemtest der OC eines Bahnhofs mit zwei Gleisen unter Abwesenheit der Achszähler (vgl. Abb. 6.3) aus wie in Abb. 6.4 dargestellt.

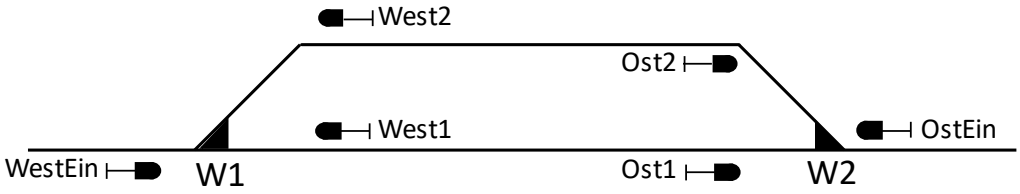

Abb. 6.3: Gleisbild der in DSTW projektieren Gleisfeld-Komponenten

```
---
switches:
  - W1
  - W2
signals:
  - WestEin
  - OstEin
  - West1
  - West2
  - Ost1
  - Ost2
routes:
  - OstEinWest1:
      W2: 1
      OstEin: 4
      WestEin: 1
      Ost2: 1
      Ost1: 1
  - OstEinWest2:
      W2: 2
      OstEin: 4
      WestEin: 1
      Ost2: 1
      Ost1: 1
```

Abb. 6.4: Komponentenplan der projektierten Gleisfeld-Komponenten

Die beteiligten OC werden nach Kategorie aufgeschlüsselt in Listen ihrer Namen angegeben. Weitere Eigenschaften wie IP- und MAC-Adressen werden automatisch generiert, zusätzlich zu den zwei Switches für jeden OC, und allen verbindenden Netzen. Eine einfache Stellwerkslogik für die in Kapitel 7.3 beschriebene Simulation des Stellwerks wurde implementiert, für die an dieser Stelle der Verschlussplan übergeben werden kann. Neben der Konfigurationsdatei für Vagrant werden die Konfigurationsdateien für die innerhalb der OC aktiven Module erstellt, die im Folgenden beschrieben werden.

Wie im Kapitel 4 erwähnt bestehen die OC in DSTW aus drei Modulen (OC mit Rasta-Komponente, Crypto-Box, VPN). In DSTW-Sim wurden die beiden Security-Komponenten vernachlässigt. Stattdessen wurden OC und RaSTA-Komponente

voneinander unabhängig angelegt und innerhalb ihrer eigenen Container[7] umgesetzt, um sie flexibel austauschen zu können, sollte dies im Rahmen eines Testes nötig sein.
Der Industriestandard für die Container-basierte Virtualisierung ist **Docker**. Die Anweisungen zum Erstellen eines Containers werden Docker in Form eines Dockerfiles gegeben. Sie enthalten Informationen über das zu verwendende Betriebssystem, die in den Container zu kopierenden Daten und den Prozess, der beim Start des Containers ausgeführt werden soll. Häufig verwendete Docker-Konfigurationen, auch als Images bezeichnet, werden auf dem Docker Hub veröffentlicht, so dass diese als Ausgangspunkt für die eigenen Images von Entwicklern zur Verfügung stehen. Von den wichtigsten Images, bspw. von populären Linux-Distributionen wie Ubuntu, gibt es als offiziell gekennzeichnete Versionen. Die Docker Engine sowie das Docker Hub sind für nicht-kommerzielle Zwecke und Unternehmen bis zu einer bestimmten Größe von Mitarbeitern bzw. Umsatz kostenfrei verwendbar [DOC22].

6.1 Uhrensynchronisation

Für eine Reihe von vorstellbaren Testszenarien wie Laufzeitenmessungen der Kommunikationsprozesse ist die Synchronisierung der Uhren aller im Netz vorhandenen Computersysteme nötig bzw. hilfreich, da diese deutlich voneinander abweichen können. Der Grund hierfür liegt in ungenauen Quarzen, welche für die Erzeugung der internen Taktfrequenzen verwendet werden [TUBOJ] und bei den Uhren von VMs wird die Zeitgebung weiter verkompliziert. Laut der offiziellen Dokumentation von VirtualBox werden die Uhren der Guest-Systeme mit der Host-Zeit synchronisiert [ORA20], während Docker Container über gar keine vom Host unabhängige Zeitmessung verfügen [STO20].

Durch vom Verfasser durchgeführten Messungen zur Kontrolle der Uhrensynchronisation im Zuge der Entwicklung stellte sich die Notwendigkeit der externen Synchronisation heraus (vgl. Kapitel 8.2.1).
Die Open Source Software **linuxptp** ist eine Implementierung des Precision Time Protocol, spezifiziert in IEEE-Standard 1588-2008, welches eine Server-Client-Architektur nutzt, um in paketbasierten Netzwerken die Uhrzeit der Clients an die des Servers anzupassen. Server und Client tauschen vier Arten von Nachrichten aus, aus welchen der Client die Abweichung von der Server-Uhr berechnen kann. Zur Kompensation kann die Uhr des Clients je nach Anwendung entweder ruckartig auf die Zeit des Servers gestellt oder die Frequenz der eigenen Uhr angepasst werden, um sich der Zeit des Servers langsamer anzunähern. Mit der ersten Variante erfolgt die Synchronisation schneller, dafür werden aber Zeit-Sprünge in Kauf genommen, welche besonders bei Logging-basierten Laufzeitenmessungen für Ungenauigkeiten sorgen [MOB22]. Aus diesem Grund wurde sich dazu entschieden, die Uhr der Clients unmittelbar nach dem Start auf die Serverzeit springen zu lassen, für alle weiteren Synchronisierungsschritte, also während der Durchführung von Tests, mit der Kompensation durch Frequenzanpassung zu arbeiten.

[7] vgl. Kapitel 5.3

Die Genauigkeit der Uhrensynchronisation durch linuxptp wurde in einem Experiment der Fachhochschule Kiel untersucht [AHM18]. Es hat einen signifikanten Einfluss auf die Genauigkeit, ob die Synchronisation rein softwaregestützt oder hardwaregestützt durchgeführt wird. Ohne Hardwareunterstützung wurde eine durchschnittliche Genauigkeit von +- 500ms erreicht, mit Hardwareunterstützung dagegen +- 200ns. Für die mit DSTW-Sim im Rahmen dieser Arbeit durchgeführten Versuche wurde die softwaregestützte Synchronisation verwendet, für Tests mit höchstem Genauigkeitsanspruch lässt sich durch die Nutzung von bestimmten Linux-Versionen die Hardwareunterstützung auch in VMs aktivieren [RED] .
Als Server für die Synchronisation wurde eine zusätzlich erstellte Master-VM verwendet, bei der sich die anderen VMs als Clients über ein internes und von den Netzen zur Simulation des bb-IP-Netzes unabhängiges Management-Netzwerk anmelden.

Die Simulation der bereits genannten Netzwerkparameter Latenz, Paketfehlerrate und Paketverlustrate in den von VirtualBox erstellten internen Netzwerken wird mit der Netzwerk-Emulationssoftware **netem** umgesetzt. Diese ist dazu vorgesehen, die Konditionen eines Wide Area Networks (WAN) in einem Local Area Network (LAN) nachzubilden und ist in Linux Distributionen ab Kernel Version 2.6 integriert. Dazu werden die Netzwerkadapter (Network Interface Controller - NIC) selbst manipuliert, alle ausgehenden Pakete nach den vorgegebenen Regeln zu behandeln. So wird bspw. bei einer eingestellten Latenz von 50 ms der Versand aller Pakete um 50 ms verzögert; bei einer Paketverlustrate von 10% werden zufällig 10% aller Pakete gar nicht versendet. Durch die Konfiguration aller NIC in einem Netz lassen sich so die gewünschten Netzeigenschaften projektieren [NETTLF].

Die Software der OC sowie des simulierten Stellwerks setzen Teile der EULYNX Spezifikationen um und werden im Anschluss an die Einführung in diese im Kapitel 7.3 beschrieben.

Die Teststeuerung in DSTW-Sim erfolgt durch ein Python-Programm (im Folgenden **Experimentskript**). Für die laut Aufgabenstellung prototypische Umsetzung der Evaluationsumgebung wurde auf die Entwicklung einer graphischen Oberfläche zugunsten einer flexibleren Konfiguration und Überwachung per Konfigurationsdatei bzw. Kommandozeile verzichtet, da die Nutzer von DSTW-Sim beinahe ausschließlich Entwickler sein werden und die Handhabung auf diese Weise möglicherweise nicht präferiert, sicherlich aber ohne Hindernisse möglich ist.

Aufgrund der großen Vielfalt an möglichen Testzielen musste bezüglich der Testauswertung eine Auswahl und damit Eingrenzung erfolgen. Die Architektur erlaubt jedoch eine unkomplizierte Erweiterung durch weitere Auswertungsmodule. Für DSTW-Sim wurden zwei verschiedene Auswertungsmethoden implementiert. Die erste basiert auf dem populären Framework **Elasticsearch**. Elasticsearch bietet einen großen Umfang an Enterprise-Funktionen für das Monitoring von vielgliedrigen Computersystemen, zu denen DSTW-Stellbereiche mit vielen Komponenten

gehören. Darunter auch die Sammlung von Logdateien aus vielen verschiedenen Quellen durch die Teilmodule Filebeat und Logstash und die echtzeitgerechte Auswertung und graphische Darstellung dieser durch das Teilmodul Kibana. Es wurde eine vollständige Installation von Elasticsearch, Logstash und Kibana in jeweils eigenen Docker-Containern auf der Master-VM vorgenommen. Die einzelnen VMs führen eine Version von Filebeat aus, welche Logs der OC-Module an Logstash liefern. Logstash verfügt über die Möglichkeit der Verarbeitung von Daten in Form einer konfigurierbaren Daten-Pipeline mit einer eigenen Syntax, welche für die Auswertung von DSTW-Sim Daten vorbereitet wurde.
Jedoch zeigte sich während der Entwicklung eine durch die Logstash zugrundeliegenden Java-Software „Joda“ erzeugte Limitierung in der Präzision von zu verarbeitenden Uhrzeiten, welche nur bis zum Millisekunden-Niveau reichten [BME]. Da die Präzision der Zeitmessungen innerhalb der Entwicklungsumgebung nicht an dieser Stelle beschränkt sein sollte, wurde neben der Auswertung durch Elasticsearch eine Python-Alternative entwickelt.

Diese basiert auf der ebenfalls populären Bibliothek für Hochleistungs-Tabellenkalkulationen, **Pandas.** Die in Schritt fünf der Teststeuerung gesammelten Log-Dateien werden über den bereits erwähnten geteilten Ordner der VMs an den Host-Computer zur Auswertung übertragen. Durch verschiedene in Kapitel 8 näher beschriebene Auswertungsmethoden können so die Testergebnisse ermittelt werden.

7 Validierung von DSTW-Sim

Das vom EULYNX-Konsortium für die Kommunikation zwischen den Teilsystemen des DSTW ausgewählte Protokoll ist Rail Safe Transport Application (**RaSTA**). Der Test einer Open Source Implementierung dieses Protokolls wurde zur Validierung von DSTW-Sim für geeignet und wissenschaftlich interessant gehalten (vgl. Anhang B). Ein solcher Test entspricht exakt dem Szenario der Überprüfung von für den Eisenbahnbetrieb zuzulassenden Softwarekomponenten, für welches DSTW-Sim vorgesehen ist. Laut Recherche des Verfassers existierte zu Beginn des Bearbeitungszeitraumes der Arbeit nur die Open Source Implementierung von Dr. Markus Heinrich, angefertigt im Rahmen einer Dissertation an der Technischen Universität Darmstadt [HEI20]. Die Benutzung einer Closed Source Implementierung vom Team der Digitalen Schiene Deutschland bei der DB Netz AG wurde angefragt, doch aus rechtlichen Gründen konnte keine rechtzeitige Bereitstellung erfolgen und wurde daher ausgeschlossen (Ibtihel Cherif, persönliche Kommunikation vom 22.10.2022).

Die Implementierung von Heinrich wurde in der Programmiersprache C entwickelt und ist auf Github, einer Plattform zur Veröffentlichung von und Kollaboration an Open Source Software, unter dem Namen Railway-CCS/rasta-protocol verfügbar [OSRASTA]. Im Folgenden wird sie als Open Source RaSTA (OS RaSTA) bezeichnet.

Das Ziel des Tests ist die Beantwortung folgender Fragestellungen:

1. Welchen Einfluss hat der Einsatz von RaSTA auf die Fahrstraßenbildezeit?
2. Unter welchen Netzwerkbedingungen erfüllt OS RaSTA die Anforderungen der EULYNX-Spezifikationen bzw. der DB Netz AG, soweit bekannt.
3. Über welche Leistungsfähigkeit muss die Hardware eines OC mindestens verfügen, wenn die Kommunikation über OS RaSTA abgewickelt werden soll.

Folgendes Testszenario wurde für die Validierung formuliert: Die Testprojektierung soll aus einem zweigleisigen Kreuzungsbahnhof mit zwei Weichen bestehen (vgl. Abb. 6.3). Gleisfreimeldeanlagen werden vernachlässigt. Während des Tests sollen die zur Einstellung einer Fahrstraße inklusive Freigabe der Fahrt nötigen Nachrichten durch einen TSO und alle beteiligten OC ausgetauscht werden, analog zu den Vorgängen, die zur Durchführung einer Zugfahrt von einem DSTW absolviert werden müssen. Ausgangslage soll ein vollständig initialisiertes, im Betrieb befindliches Stellwerk sein, d. h. die Schritte zum Hochfahren des Stellwerks und der OC sind für die hier gegebene Aufgabenstellung nicht von Interesse. Nachdem die zur Freigabe der Zugfahrt nötigen Signale auf Fahrt gestellt sind, wird angenommen, dass die Zugfahrt stattgefunden hat, die Signale wieder auf Halt gefallen sind und das Stellwerk über den angezeigten Signalbegriff der Signale informiert wurde.

Zum besseren Verständnis wird im Folgenden zunächst auf die in der EULYNX-Spezifikation beschriebenen Schnittstellen und die über diese auszutauschenden Daten eingegangen, da diese die Gegenstände der Übertragung durch RaSTA darstellen. Ebenso werden dort auch die variablen Parameter definiert, die in der RaSTA-Norm zur Konfiguration durch den jeweiligen Anwender offengelassen sind.

Im Anschluss erfolgt eine kurze Einführung in die wesentlichen Aspekte des RaSTA-Protokolls.

7.1 EULYNX Spezifikation

Die EULYNX Spezifikation verfolgt einen modellbasierten Ansatz und ist in Notationen der Systems Modeling Language (SysML) verfasst, welche den europäischen Standards für funktionale Sicherheit genügen, wie sie in relevanten CENELEC-Normen gefordert sind (EN 50126, EN 50128, EN 50129, EN 50159). Ein geringer Teil der Spezifikation ist öffentlich über die offizielle Webseite zugänglich (https://eulynx.eu), der Zugang zu den technischen Dokumenten muss beantragt werden. Auf entsprechenden Antrag des Verfassers wurde die Nutzung der Dokumente im Rahmen der vorliegenden Arbeit gewährt.

Die Spezifikation besteht aus mehr als einhundert Dokumenten. Diese sind identifiziert durch eine Angabe des Versionsdatums, des Namens und einer Ordnungsnummer nach dem Schema Eu.Doc.XX, unter deren Nutzung sie auch innerhalb der Texte referenziert werden; für die vorliegende Arbeit wurde sich an diesem Schema orientiert. Es existieren unter anderem Dokumente, welche das verwandte Model-based Systems Engineering Specification Framework (MBSE SF) zur Erstellung der Spezifikation beschreiben [Eu.Doc.30], oder eine Anleitung zur Interpretation der Inhalte für mit SysML nicht vertraute Leser enthalten [Eu.Doc.29]. Eine vollständige Liste mit allen der aktuellen Version (Baseline 4.0) der Spezifikation angehörigen Dokumenten findet sich im Documentation Plan [Eu.Doc.11].

Aufgrund des Umfangs der in den EULYNX Dokumenten präsentierten Informationen sollen im Folgenden lediglich die im Rahmen der Entwicklung von DSTW-Sim relevanten Dokumente bzw. die relevanten Ausschnitte aus diesen beschrieben werden. So sind für die Implementierung jeder SCI Schnittstelle Informationen aus zwei für diese Schnittstelle spezifischen Dokumenten nötig, der Requirement Specification for XX und der Interface Specification SCI-IXX. Darüber hinaus werden allgemein gültige Informationen für SCI Schnittstellen in den folgenden Dokumenten gegeben. Auf die im Rahmen von DSTW-Sim verwandten und umgesetzten Schnittstellen, SCI-P und SCI-LS, wird im Anschluss an den allgemein gültigen Abschnitt eingegangen.

Interface definition SCI [Eu.Doc.92] legt die bereits genannten RaSTA-Parameter fest (siehe Kapitel 7.2.1).

Interface specification SCI Generic [Eu.Doc.93] definiert die allgemeine Telegrammstruktur sowie die Struktur der zum Aufrüsten und Abrüsten der OC auszutauschenden Telegramme. Ein EULYNX-Telegram folgt dem internen Process Data Interface (PDI) Protokoll und ist wie folgt aufgebaut:

PDI Telegram

Ein PDI-Telegram besteht aus Header und Payload (vgl. Tab. 7.1). Insgesamt besteht es aus maximal 1024 Bytes, von denen 43 immer auf den Header entfallen. Die Daten im Payload-Feld besteht aus einer schnittstellenspezifischen Sequenz von Parametern.

Byte	**Feldwert**
00	Specific Protokoll Type
01..02	Message Type
03..22	Sender Identifier
23..42	Receiver Identifier
43..1023	Payload

Tab. 7.1: PDI Telegram Struktur, aus [Eu.Doc.93]

Der *Specific Protokoll Type* gibt die SCI Schnittstelle an. Die möglichen Werte können einer Tabelle entnommen werden (für SCI-P ist es bspw. 0x40).
Der *Message Type* identifiziert das Paket als eines von einer Menge an vorbestimmten Telegrammen. Anhand dieser Information kann die im Payload befindliche Information interpretiert werden.
Sender- und Receiver Identifier kennzeichnen jeweils das Sender und Empfänger EULYNX Subsystem. Jedes Subsystem muss einen einmaligen Namen haben [Eu.Doc.16].

Generic interface and subsystem requirements for SCI [Eu.Doc.119] regelt Abläufe die alle EULYNX Subsysteme betreffen: Den Ablauf beim Aufrüsten, Abrüsten und bei Kommunikationsproblemen wie Protokoll-Fehlern, formalen Telegram-Fehlern und Inhaltsfehlern. Eine bestehende Verbindung auf Basis des Process Data Interface wird PDI-Verbindung genannt. Die Aufrüstung läuft nach dem in Abb. 7.1 gezeigten Schema ab.

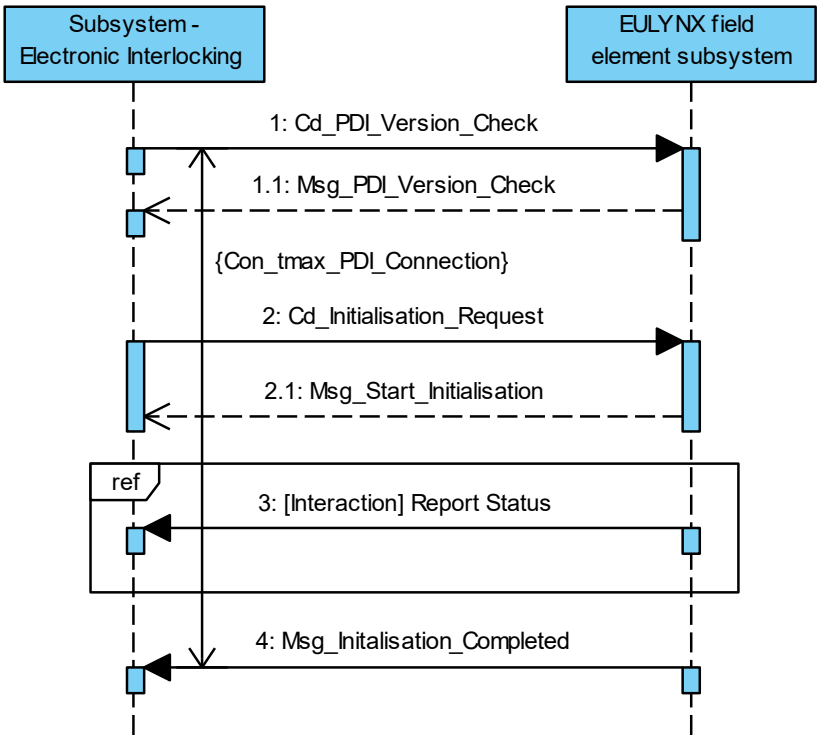

Abb. 7.1: Allgemeine Aufrüstung von EULINX-Subsystemen nach [Eu.Doc.119]

EULYNX System architecture specification [Eu.Doc.16] präzisiert die Systemarchitektur des Kommunikationsnetzes. Unter anderem werden die Kommunikationspartner in einen primären und einen sekundären eingeteilt. Die Aufgabe des primären Partners besteht darin, die PDI-Verbindung mittels RaSTA herzustellen, zu überwachen und gegebenenfalls wiederherzustellen. Der sekundäre Partner soll die Verbindung lediglich auf Verbindungsabbrüche überwachen (mittels RaSTA).

Specification of Point of Service-Signalling [Eu.Doc.100] definiert die Anforderungen an das Netzwerk, über welches die EULYNX Kommunikation abgewickelt wird. Unter anderem wird hier die vollständige Redundanz des physischen Layers gefordert, wie sie von der DB Netz AG mittels blauer und grauer Netzwerkebene umgesetzt wird. Weiterhin werden Anforderungen an sogenannte Quality of Life Kriterien gestellt, getrennt nach den Übertragungsprofilen Standard Wired Profile, High-Performance Wireless/Radio Profile und Low-Performance Wireless/Radio Profile. Für manche Vorgaben wird unterschieden zwischen der Schnittstelle SCI-XX und den übrigen Schnittstellen SDI-XX, SMI-XX und SSI-XX.

Standard Wired

- Die maximale Latenz für SCI-XX Verbindungen beträgt 50 ms, für die übrigen sind es 250ms
- Die maximale Paketfehlerrate zwischen zwei EULYNX Endgeräten beträgt für SCI-XX Verbindungen < 1% über einen Zeitraum von 5 Minuten, für die übrigen sind es 5% über einen Zeitraum von 5 Minuten

High-Performance Wireless/Radio Profile

- Die maximale Latenz für SCI-XX, SDI-XX, SMI-XX und SSI-XX Verbindungen beträgt 250 ms
- Die maximale Paketfehlerrate zwischen zwei EULYNX Endgeräten beträgt für SCI-XX, SDI-XX, SMI-XX und SSI-XX Verbindungen < 5% über einen Zeitraum von 5 Minuten

Low-Performance Wireless/Radio Profile

- Die maximale Latenz für SCI-XX, SDI-XX, SMI-XX und SSI-XX Verbindungen beträgt 1300 ms
- Die maximale Paketfehlerrate zwischen zwei EULYNX Endgeräten beträgt für SCI-XX, SDI-XX, SMI-XX und SSI-XX Verbindungen < 1% über einen Zeitraum von 5 Minuten

Das Standard Wired Profile ist für die normale Nutzung vorgesehen. Das High-Performance Wireless/Radio Profile ist für Situationen vorgesehen, in denen kabellose Hochleistungstechnologien innerhalb der DSTW-Architektur eingesetzt werden und die Spezifikation weist lediglich darauf hin, die etwas niedrigere garantierte Leistungsfähigkeit bei der Sicherheitsbewertung zu beachten. Das Low-Performance Wireless/Radio Profile ist nur bei regionalen Eisenbahnlinien mit sehr niedriger Verkehrsintensität einzusetzen, falls die Abdeckung per Kabel oder drahtlosen Hochleistungstechnologien nicht machbar ist.

High Performance Wireless ist aufgrund der höheren Paketfehlerrate hauptsächlich mit TLS over TCP empfohlen, da TCP über eine schnellere Übertragungswiederholung verfügt.

7.1.1 Subsystem Point - SCI-P

Die „Requirements specification for subsystem Point“ [Eu.Doc.36] definiert das Verhalten des OC einer Weiche, während die „Interface specification SCI-P“ [Eu.Doc.38] den Aufbau der jeweiligen Telegramme definiert. Der OC ist mit der Point-Machine über einen 4-Pol- oder non-4-Pol-Anschluss verbunden. Point-Machine bezeichnet eine Einrichtung zur Bewegung von Weichenzungen und zu deren Positionsüberwachung. Ein OC kann mit mehreren Point-Machines verbunden sein, was allerdings keine wesentlichen Auswirkungen auf die Funktionsweise hat und deshalb vernachlässigt werden soll. SCI-P definiert vier verschiedene Typen von Telegrammen, die OC und Stellwerk nach der in [Eu.Doc.119] beschriebenen generischen Initialisierung austauschen können. Der in [Eu.Doc.93] beschriebene Header mit 43 Byte Länge wird jedes Mal vorangestellt, die in [Eu.Doc.119] beschriebenen Inhalte definieren also nur neue Informationen für den Telegramminhalt ab dem 44. Byte.

Befehl der Positionsänderung (Cd_Move_Point)
Messagetype: 0x0001
Payload: Commanded Point Position – 1 Byte

Positionsstatus (Msg_Point_Position)
Messagetype: 0x000B
Payload: Reported Point Position – 1 Byte
Degraded Point Position – 1 Byte

Ablauf des Timers (Msg_Timeout)
Messagetype: 0x000C
Payload: keine Daten

Bereitschaft zur Positionsänderung (Msg_Ability_To_Move_Point)
Messagetype: 0x000D
Payload: Reported Ability To Move Point Status – 1 Byte

Nachdem das Stellwerk am Ende des Aufrüstvorgangs einen Status Report in Form des Positionsstatus empfangen hat, sendet es im Normalbetrieb lediglich Befehle zur Positionsänderung. Der Regelfall für die Umstellung einer Weiche (Main Success Scenario: Moving of the Point with a single point machine) läuft folgendermaßen ab: Der OC hat die Ausgangsposition „End-Position X“ oder „Trailed“ (Aufgefahren). Das Stellwerk sendet einen Befehl der Positionsänderung zur „End-Position Y“. Nach Empfang des Befehls weist der OC den verbundenen Weichenantrieb zur Bewegung an. Sobald der OC bei der Überwachung der Point-Machine eine Änderung in der Lage detektiert, sendet es einen Positionsstatus mit der Information „No end Position“ an das Stellwerk. Ist der Stellvorgang abgeschlossen und es wird die Ankunft in der neuen Endlage registriert, sendet der OC wieder einen Positionsstatus mit der neuen Endlage und fährt mit der Überwachung dieser fort.

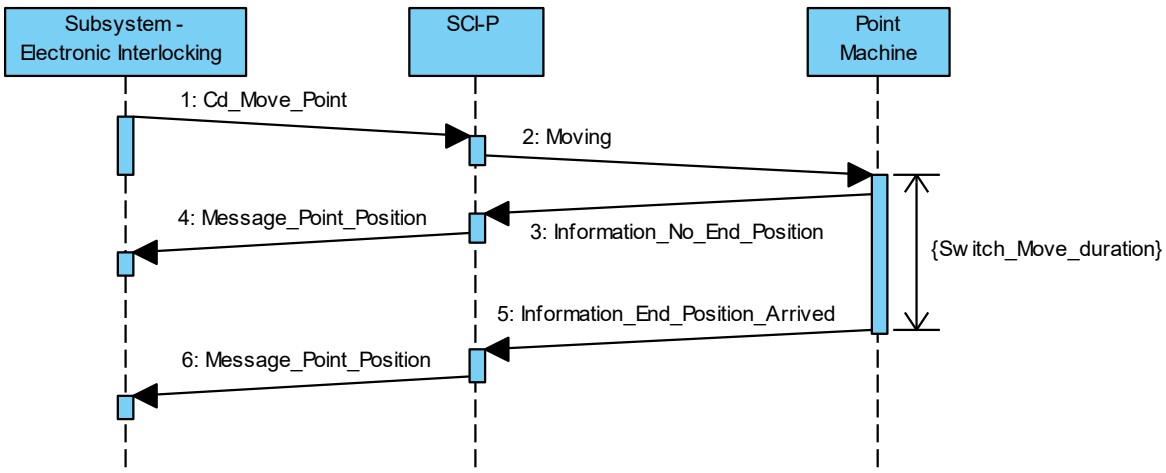

Abb. 7.2: EULYNX Umstellung einer Weiche - Main Success Scenario: Moving of the Point with a single point machine, Darstellung in Anlehnung an [Eu.Doc.36]

Die Kommunikation zwischen Stellwerk und OC funktioniert also ereignisbasiert. Prinzipiell gibt es keine direkte Antwort auf den Befehl zur Positionsänderung,

sondern nur eine neue Nachricht mit der Information über den aktuellen Status des OC. Diese wird bei jeder neuen Meldung über die Lage der Point-Machine ausgelöst. Die Spezifikation macht Vorgaben für die maximale Zeitdauer der Vorgänge im OC. Wenn der OC eine Nachricht über einen Zustandswechsel in der Point-Machine empfängt, muss er innerhalb von 250 ms das Stellwerk benachrichtigen. Eine Positionsänderung muss innerhalb von 500 ms nach Erhalt des Befehls dazu in der Point-Machine gestartet werden. Für das Erreichen in einer Endlage vom Zeitpunkt des Starts der Positionsänderung an ist ein Zeitfenster von 100 ms bis 30 s zulässig, welches vom EIB gewählt werden kann. Wird dieses überschritten, feststellbar durch das Auslaufen des Timers „Con_tmax_Point_Operation“, sendet der OC eine Timeout Nachricht an das Stellwerk.

7.1.2 Subsystem Light Signal – SCI-LS

Analog zu SCI-P wird das Verhalten des Subsystem Light Signal (LS) durch die „Requirements specification for subsystem Light Signal“ [Eu.Doc.32] beschrieben, der Aufbau der entsprechenden Telegramme ist in der „Interface specification SCI-LS“ [Eu.Doc.33] definiert. Im Folgenden sind die möglichen Telegramme der Schnittstelle beschrieben, auf die vollständige Auflistung der möglichen Payload-Felder bei den Telegrammen zur Änderung bzw. Statusmeldung eines Signalbegriffes wurde jedoch verzichtet. Es handelt sich um neun weitere Felder, die in [EU.Doc.33] nachgeschlagen werden können.

Befehl zum Anzeigen eines Signalbegriffs (Cd_Indicate_Signal_Aspect)
Messagetype: 0x0001
Payload: code for basic aspect types – 1 Byte
code for extension of basic aspect types – 1 Byte
weitere…

Statusmeldung angezeigter Signalbegriff ()
Messagetype: 0x0003
Payload: code for basic aspect types – 1 Byte
code for extension of basic aspect types – 1 Byte
weitere…

Befehl zum Einstellen der Helligkeit (Cd_Set_Luminosity)
Messagetype: 0x0002
Payload: Luminosity – 1 Byte

Statusmeldung eingestellte Helligkeit (Msg_Set_Luminosity)
Messagetype: 0x0004
Payload: Luminosity – 1 Byte

Die Helligkeit (Luminosity) kann entweder auf Tag oder Nacht gestellt sein. Der Regelfall für die Umstellung eines Signalbegriffs (Main Success Scenario: Indicate signal aspect) ist trivial: Das Stellwerk sendet einen Befehl zum Anzeigen eines

Signalbegriffs. Dieser wird mit allen verfügbaren Lampen in der aktuell eingestellten Helligkeit angezeigt. Das LS antwortet mit einer Statusmeldung über den angezeigten Signalbegriff.

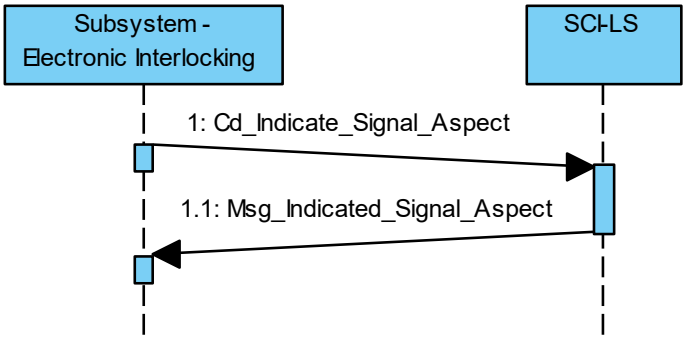

Abb. 7.3: EULYNX Änderung eines Signalbegriffs - Main Success Scenario: Indicate signal aspect, Darstellung in Anlehnung an [Eu.Doc.32]

Die maximal zulässigen Verzögerungen lauten wie folgt: Das LS muss den befohlenen Signalbegriff oder die einzustellende Helligkeit innerhalb von 1000 ms nach Empfang des Befehls anzeigen bzw. eingestellt haben. Dies gilt nicht für „Stop"-Signalbegriffe, für diese beträgt die zulässige Verzögerung maximal 500 ms. Eine Information über den aktuell angezeigten Signalbegriff muss spätestens 800 ms nach Änderung desselben an das Stellwerk übermittelt werden.

7.2 Rail Safe Transport Application RaSTA

In der EULYNX System architecture specification [Eu.Doc.16] verlangt die EULYNX Spezifikation nach einem sogenannten Safe Communication Protocol (SCP) als Grundlage für das Process Data Interface Protocol (PDI) auf der Anwendungsebene. Dieses SCP soll folgende Anforderungen erfüllen:

- Garantierte Zustellung der Nachrichten in der richtigen Reihenfolge
- Garantierte Benachrichtigung bei Misserfolgen in der Zustellung von Nachrichten
- Garantierter Verbindungsabbruch, falls die oben genannten Garantien nicht gegeben werden können
- Garantie, dass Verbindungsauf- und Abbau von beiden Kommunikationspartnern gleichzeitig erkannt werden[8]

Zur Erfüllung dieser Anforderungen soll das in der Vornorm DIN VDE V 0831-200 spezifizierte Rail Safe Transport Application (RaSTA) Protokoll verwendet werden. Zusätzlich zu den oben genannten Eigenschaften verfügt RaSTA über weitere, wie die kontinuierliche Überwachung der Kanalgüte durch den Austausch von Heartbeat-Nachrichten, die Fähigkeit der Nutzung von mehreren Transportkanälen zur Steigerung der Ausfallsicherheit und ein Diagnose-Modul.

[8] Dies bedeutet, dass beide Kommunikationspartner den Status der Verbindung ändern, bevor weitere Nachrichten übertragen werden können

RaSTA besteht aus zwei voneinander unabhängigen Schichten, der Sicherheits- und Sendewiederholschicht und der Redundanzschicht. Wie in Abb. 7.4 dargestellt, ordnen diese sich zwischen Anwendungs- und Transportschicht des TCP/IP-Modells ein. Ebenfalls dargestellt sind die zwei Möglichkeiten für die Transportschicht unter RaSTA: Sowohl UDP als auch TLS over TCP können genutzt werden, mit jeweils entsprechenden Implikationen für Security- bzw. funktionale Eigenschaften, welche später noch diskutiert werden. In OS RaSTA wurde UDP für die Transportschicht verwandt.

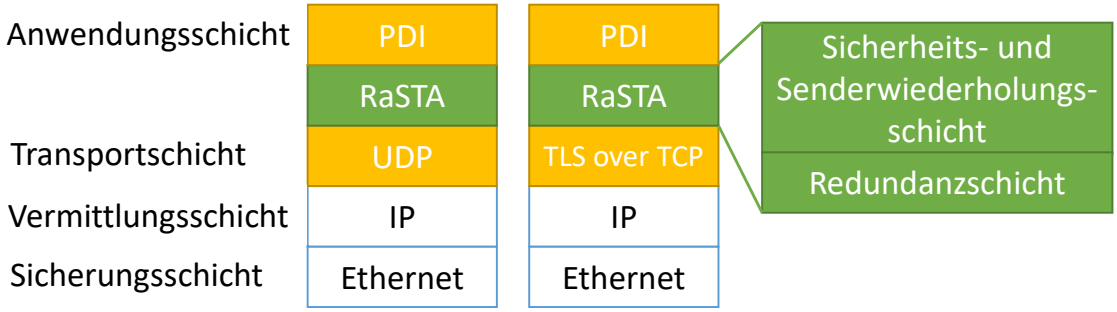

Abb. 7.4: Schematische Einordnung von Rasta in das OSI-Modell, Darstellung in Anlehnung an [Eu.Doc.92]

Da Kenntnis über einige Aspekte der Funktionalität von RaSTA für das Verständnis der im Rahmen der Arbeit mit OS RaSTA durchgeführten Tests nötig ist, werden die entsprechenden Details in Anlehnung an die Vornorm DIN VDE V 0831-200 kurz beschrieben. Das Ziel der Tests liegt unter anderem in der Ermittlung der Grenzen der Beeinträchtigungen des Netzes, unter denen OS RaSTA ordnungsgemäß funktioniert. Daher kommt besonders den Funktionen Bedeutung zu, durch die es zum Versagen einer Verbindung kommen kann.

7.2.1 Konfigurierbare Parameter

RaSTA verfügt über eine Reihe von konfigurierbaren Parametern. Die meisten davon betreffen die Generierung der Kontrollsumme und die Leistungsfähigkeit der Hardware in Form der Anzahl von maximal hintereinander zu sendenden bzw. zu buffernden Nachrichten. Für die Aufrechterhaltung der Verbindung relevant sind folgende drei Zeiten für RaSTA über UDP in [Eu.Doc.92] festgelegt:

1. T_{max}: das maximal akzeptierte Alter einer Nachricht ist mit 1800 ms festgelegt.
2. T_h: die Zeitspanne für das Senden von Heartbeat-Nachrichten ist auf 300 ms festgelegt.
3. T_{seq}: die Zeitspanne, die von der Redundanzschicht bei Empfang einer außerhalb der Sequenz empfangenen Nachricht gewartet wird, ist auf 100 ms festgelegt.

7.2.2 Sicherheits- und Sendewiederholungsschicht

Eine Protokolldateneinheit der Sicherheits- und Sendewiederholungsschicht hat folgenden Aufbau:

Byte	Feldwert
0..1	Nachrichtenlänge
2..3	Nachrichtentyp
4..7	Empfängerkennung
8..11	Absenderkennung
12..15	Sequenznummer
16..19	Bestätigte Sequenznummer
20..23	Zeitstempel
24..27	Bestätigter Zeitstempel
28..n	Nutzdaten
28+n..44+n	Sicherheitscode

Tab. 7.2: Aufbau der Protokolldateneinheit der Sicherheits- und Sendewiederholungsschicht von RaSTA nach DIN VDE V 0831-200 [RAS15]

Nachrichtenlänge (2 Bytes)
Angabe der Anzahl Bytes der gesamten PDU.
Nachrichtentyp (2 Bytes)
Identifikationsnummer mit Zuordnung zum entsprechenden Nachrichtentyp. Mögliche Nachrichtentypen sind Verbindungsanfrage, Verbindungsantwort, Sendewiederholungsanfrage, Sendewiederholungsantwort, Trennaufforderung, Lebenszeichen, Daten und erneut übertragene Daten.
Empfängerkennung (4 Bytes)
Eindeutige Kennnummer des innerhalb eines RaSTA-Netzwerks empfangenen Clients.
Absenderkennung (4 Bytes)
Eindeutige Kennnummer des innerhalb eines RaSTA-Netzwerks gesendeten Clients.
Sequenznummer (4 Bytes)
Nummer einer Nachricht, die mit jeder weiteren Nachricht inkrementiert wird.
Bestätigte Sequenznummer (4 Bytes)
Nummer einer Nachricht, deren Erhalt mit der aktuellen Nachricht bestätigt wird.
Zeitstempel (4 Bytes)
Zeitstempel des Nachrichten-Absenders.
Bestätigter Zeitstempel (4 Bytes)
Bestätigter Zeitstempel der Nachricht.
Nutzdaten (Variable Anzahl Bytes)
Nutzdaten, die sowohl Kontrollnachrichten des RaSTA-Protokolls selbst, oder übertragene Daten der Anwendungsschicht sein können.

Sicherheitscode (0, 8 oder 16 Bytes)
RaSTA stellt die Integrität der PDU mit einem Sicherheitscode sicher. Dieser basiert auf der MD4 Hashfunktion, deren 128-Bit Ausgabe auf 64 Bit gekürzt wird, um in das Feld für den Safety Code zu passen. Der Initialisierungsvektor dieser Funktion, oder auch Schlüssel, kann durch einen dem RaSTA-Netzwerk zugeordneten Code ersetzt werden. Wird dieser Schlüssel geheim gehalten, können nur legitime Teilnehmer des Netzwerkes die ausgetauschten Nachrichten entschlüsseln (vgl. Kap 7.2.9 bezüglich der Sicherheitsqualität dieses Verfahrens) [HEI20]. Es kann der vollständige Hashwert verwendet werden (16 Bytes), nur die untere Hälfte des Hashwertes (8 Bytes), oder es kann auf einen Sicherheitscode verzichtet werden (0 Bytes).

Bei Versenden einer Nachricht werden diese Felder mit den entsprechenden Werten befüllt, bei Empfang ausgewertet. Es wird eine Prüfung jedes Wertes durchgeführt, jedoch ist nur der Erfolg der Prüfungen von Sequenznummer, rechtzeitiger Übertragung und Flusskontrolle abhängig von Netzwerkparametern.

7.2.3 Prüfung der Sequenznummer

Der Prüfung der Sequenznummer geht die Prüfung des Sequenznummernbereichs voraus, bei der Nummern als ungültig erkannt werden, die kleiner sind als die erwartete Sequenznummer oder unglaubwürdig weit von ihr entfernt liegen. Anschließend wird die Sequenznummer mit der erwarteten verglichen, bei Ungleichheit wird die PDU verworfen und mit einer Sendewiederholungsanfrage geantwortet.

7.2.4 Prüfung der rechtzeitigen Übertragung

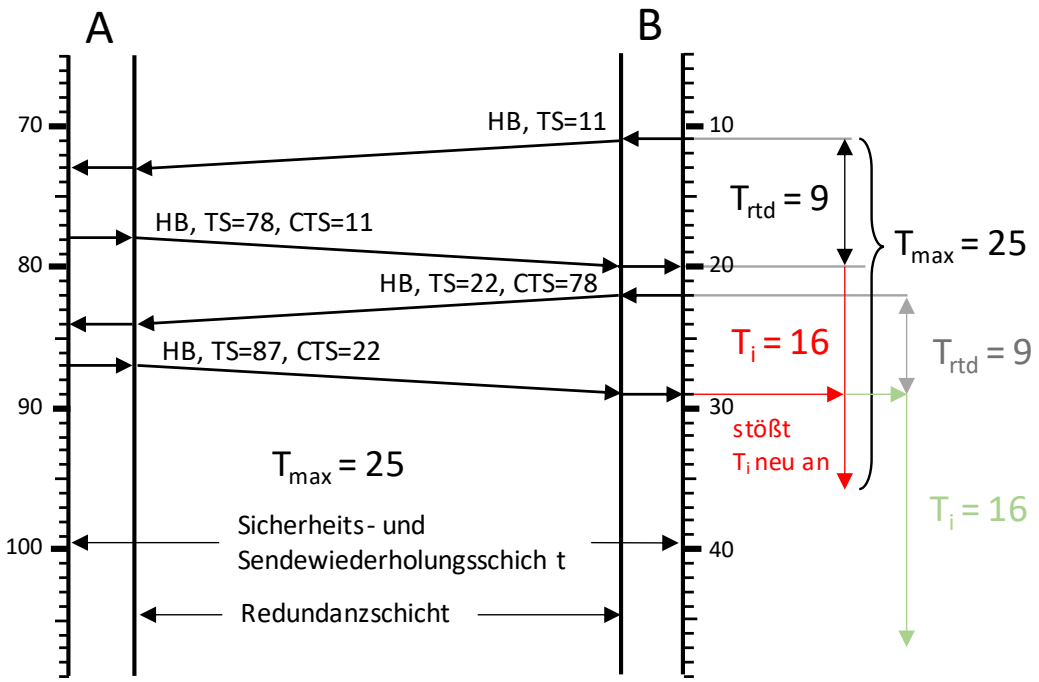

Abb. 7.5: Adaptive Kanalüberwachung, modifiziert nach DIN VDE V 0831-200 [RAS15]

Zum Feststellen des Alters einer Nachricht benutzt RaSTA das Prinzip des Doppel-Zeitstempels. Dadurch wird vermieden, dass die Uhren der Kommunikationspartner miteinander synchronisiert sein müssen. Bei Empfang einer Nachricht wird der enthaltene Zeitstempel darin in der nächsten Nachricht ohne jegliche Interpretation oder Änderung im Feld „Bestätigter Zeitstempel“ (Confirmed Timestamp - CTS) zurückgeschickt. Die Gesamtumlaufzeit einer Nachricht kann von jedem Kommunikationspartner nun aus der lokalen Uhrzeit bei Empfang der Nachricht und dem enthaltenen bestätigten Zeitstempel, der dem Partner bei der letzten Nachricht als Zeitstempel mitgeteilt wurde, errechnet werden.

Liegt der Empfangszeitpunkt der Nachricht innerhalb von T_{max} seit dem bestätigten Zeitstempel, der in der Nachricht mitgeschickt wird, ist diese rechtzeitig.

$$T_{rtd} = T_{local} + \Delta t - CTS_{PDU}$$

7.2.5 Redundanzschicht

Die Aufgabe der Redundanzschicht ist es, an sie weitergeleitete Nachrichten auf allen verfügbaren Kommunikationskanälen zu versenden. Sie versieht die von der Sicherheits- und Sendewiederholschicht heruntergereichten Nachrichten mit einem eigenen Header, der lediglich die Felder Länge, Reserve, Sequenznummer (Redundanzschicht-eigene Nummer), Nutzdaten und Prüfcode enthält.

Bei Ankunft einer Nachricht wird die Sequenznummer der PDU mit der zuletzt empfangenen Sequenznummer verglichen. Ist sie niedriger als erwartet, handelt es sich um ein Duplikat und sie wird verworfen, entspricht sie der erwarteten Nummer genau, wird sie an die nächste Schicht weitergeleitet. Ansonsten wird eine Plausibilitätskontrolle durchgeführt, d.h. liegt die Sequenznummer unglaubwürdig weit weg von der erwarteten, wird sie verworfen, ansonsten ebenfalls weitergeleitet, nachdem eine Zeitspanne T_{seq} abgelaufen ist. T_{seq} gibt an, wie lange außerhalb der Sequenz empfangene Nachrichten zwischengespeichert werden.

7.2.6 Allgemeine Funktionsweise

Der Verbindungsaufbau erfolgt durch eine Nachricht von Kommunikationspartner A vom Typ Verbindungsanfrage, die durch eine Nachricht von Kommunikationspartner B vom Typ Verbindungsantwort bestätigt wird. Durch diese Nachrichten werden unter anderem Verbindungsparameter abgestimmt, wie die Anzahl der Nachrichten, die ein Partner senden darf, ohne eine Bestätigungsnachricht empfangen zu haben. Im Abstand von T_h tauschen die Partner nun Heartbeat-Nachrichten aus. Durch das Versenden einer Daten-Nachricht wird der Heartbeat-Timer ebenfalls zurückgesetzt.

7.2.7 Sendewiederholung

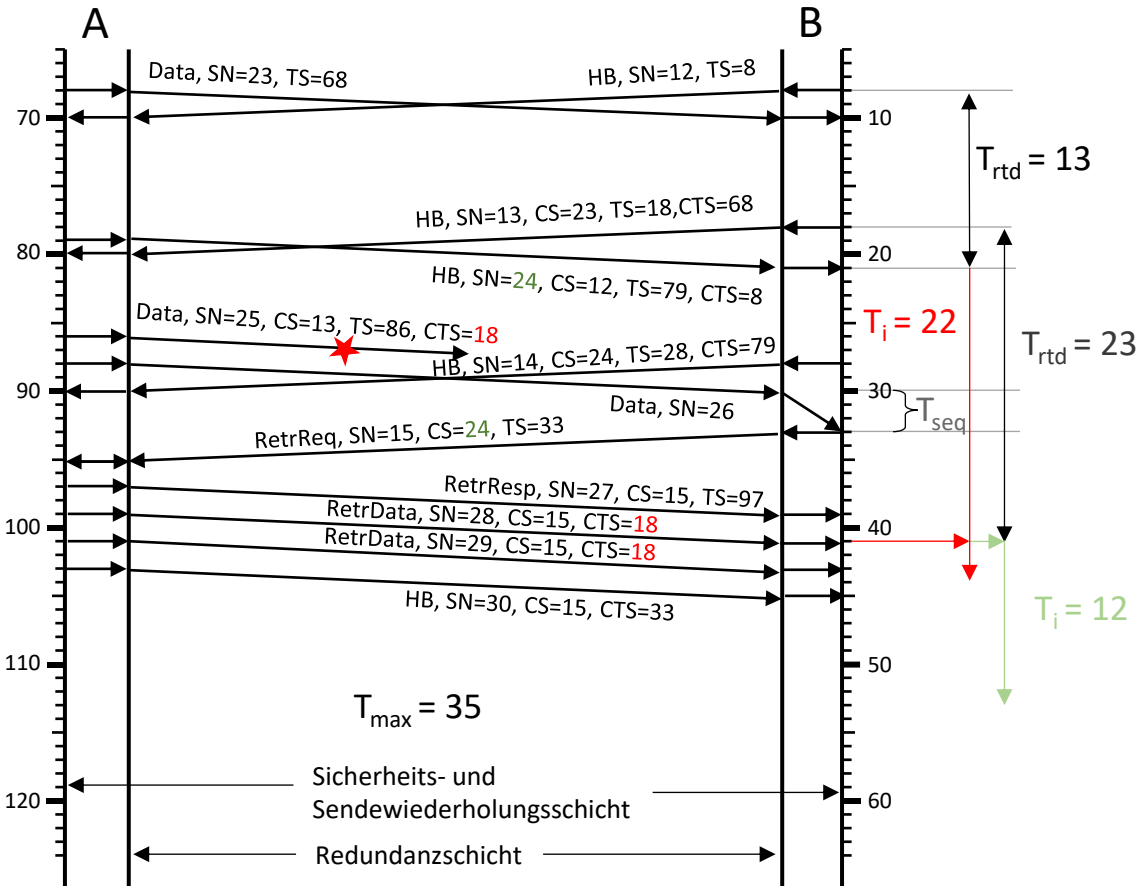

Abb. 7.6: Sendewiederholungsszenario, modifiziert nach DIN VDE V 0831-200 [RAS15]

Die Ursache für eine Sendewiederholung ist immer der Empfang einer Nachricht außerhalb der Sequenz. Eine erfolgreiche Sendewiederholung läuft wie dargestellt in Abb. 7.6[9] nach folgendem Ablauf ab:

1. Die Nachricht mit Sequenznummer 25 geht auf allen Transportkanälen verloren.
2. Die Nachricht mit Sequenznummer 26 kommt bei Kommunikationspartner B an. Bereits die Redundanzschicht erwartet eine andere Nummer und verzögert die Weitergabe der Nachricht um T_{seq} . Im Anschluss wird auf der Sicherheits- und Sendewiederholschicht ein Sequenznummernfehler festgestellt und die Nachricht mit Sequenznummer 26 verworfen. Das Sendewiederholungsszenario beginnt.
3. B sendet eine Sendewiederholungsanfrage. In dieser wird die Sequenznummer 24 bestätigt.
4. A antwortet mit der Sendewiederholungsantwort, in welcher die Sequenznummer normal weitergezählt wird.
5. Nun werden von A alle noch nicht bestätigten Nachrichten wiederholt. Die ursprüngliche Nachricht mit Sequenznummer 25 wird nun als 28 gesendet, in der weiterhin der letzte vor dem Sendewiederholungsszenario empfangene Zeitstempel bestätigt wird (18). Am Nachrichtentyp „Erneut übertragene Daten“ kann die Nachricht von B richtig zugeordnet werden.
6. Zum Abschluss der Sendewiederholung sendet A eine Heartbeat-Nachricht, welche das Ende der Sendewiederholung markiert.

Die adaptive Kanalüberwachung, d.h. die Prüfung der Rechtzeitigkeit von Nachrichten, läuft während des Aushandelns der Sendewiederholung weiter. Erst mit der Ankunft der Heartbeat-Nachricht, welche einen neuen bestätigten Zeitstempel enthält, nämlich den der Sendewiederholungsantwort, gilt ein neuer Ausgangszeitpunkt für die Berechnung von T_{max}. Das bedeutet, die gesamte Sendewiederholung muss innerhalb von T_{max} seit der letzten ausgesendeten Nachricht erledigt sein. Daraus ergibt sich folgender Zusammenhang für die empfohlene Größe von T_{max} und T_h:

$$T_{max} > 3 * T_h + 2 * (T_a + T_b) + T_{seq}$$

T_a, T_b : Übertragungszeit Teilnehmer A, B
T_{max} : maximales Alter einer Nachricht
T_h : Zeitspanne zwischen Heartbeat-Nachrichten
T_{seq} : Wartezeit von außerhalb der Sequenz empfangenen Nachrichten

Diese Zeit ergibt sich so: Die Gesamtumlaufzeit einer Nachricht wird maximal $T_a + T_h + T_b$ dauern. Der Verlust einer Nachricht offenbart sich etwa nach $2 * T_h$, zusätzlich wird die Offenbarungsnachricht durch T_{seq} verzögert. Die Vereinbarung zur Sendewiederholung dauert ebenfalls $T_a + T_b$. In Summe entsteht daraus o.g. Formel.

[9] Die Darstellung enthält lediglich die für das Verständnis nötigen Felder. Das vollständige Sendewiederholungsszenario kann in der Norm nachgeschlagen werden.

Selbstverständlich kann es während des Sendewiederholungsszenarios zu einem neuen Paketverlust kommen. Auch dieser stößt wieder ein neues Sendewiederholungsszenario an, bis das Auslaufen des Timers T_i zur Überwachung von T_{max} schließlich zu einem Abbruch der Verbindung führt.

7.2.8 Schnittstelle zur Transportschicht

RaSTA stellt zwei Anforderungen an die Schnittstelle zur Transportschicht: Sie soll eine Protokolldateneinheit der Redundanzschicht entgegennehmen und zusammen mit der angegebenen IP-Adresse und Port-Nummer an die Transportschicht übergeben, außerdem soll sie alle an dem für RaSTA definierten Port des Systems ankommenden Nachrichten an die RaSTA-Redundanzschicht übergeben.

Die Transportschicht selbst kann frei gewählt werden, explizit genannt sind die COTS-Protokolle UDP und TCP sowie Dienste mit ähnlichen Eigenschaften (RFC 1006, RFC 3550). Sie muss nicht verbindungsorientiert sein, da RaSTA die Verbindung selbst überwacht. Sie muss jedoch die hinreichend schnelle Bearbeitung eines Sendeauftrags gewährleisten können und hier unterscheiden sich UDP und TCP. Wie bereits erwähnt erzeugen die zusätzlichen Garantien von TCP zusätzliche Verzögerungen zwischen Ankunft einer Nachricht und der Übergabe an die nächste Schicht. Da RaSTA über Eigenschaften von TCP, wie die garantierte Zustellung, selbst bereits verfügt, entstehen diese Verzögerungen teilweise ohne Qualitätsgewinn.

7.2.9 Security-Eigenschaften von RaSTA

Rasta verfügt über eine für den aktuellen Stand der Technik sehr primitiven Form der Nachrichtenverschlüsselung durch den MD4-Algorithmus. Die dadurch generierte Sicherheit wurde ausführlich von Heinrich et al. in „Security Analysis of the RaSTA Safety Protocol“ (2018) [HEI18] untersucht. Da RaSTA als nicht sicher gilt, darf es laut EN 50129 ohne weitere Vorkehrungen nicht als sicherheitskritischer Bestandteil in einem Netz verwendet werden, bei dem der externe Zugriff nicht ausgeschlossen werden kann [DIN26]. Andere Möglichkeiten zur Absicherung der Kommunikation sind die Nutzung von VPN-Technologie wie IPSec und die in EULYNX-vorgeschlagene Methode TLS over TCP. Zur Vermeidung der zusätzlichen Latenzen, welche durch diese beiden Lösungen entstehen, schlagen Heinrich et al. die Nutzung eines anderen kryptographischen Algorithmus‘ vor, wie SipHash-2–4 oder BLAKE2.

7.2.10 Selbstständig entwickelte Komponenten

Für die Modellierung der Testprojektierung wurden nun die zusätzlich im Rahmen der vorliegenden Arbeit zu entwickelnden Komponenten identifiziert. Die RaSTA-Implementierung von Heinrich musste um diese Komponenten erweitert werden. Dazu gehörten die OC für Weichen und Lichtsignale unter Nutzung der Schnittstellen SCI-P und SCI-LS, welche durch OS RaSTA kommunizieren können sollen. Da der Test von anderen RaSTA Implementierungen ebenfalls möglich sein sollte, muss das RaSTA-Modul unabhängig von der restlichen Logik des OC angeordnet sein. Weiterhin war eine Komponente zur Simulation des Stellwerks nötig.

OS RaSTA verfügt über eine Dokumentation, die in knapper Form über die Installation und die für die Benutzung durch Anwender der Bibliothek vorgesehenen Funktionen informiert. Darüber hinaus existiert eine Reihe von Beispielen, aus denen die zur Benutzung nötigen Schritte ersichtlich werden. Neben den reinen RaSTA-Modulen sind in OS RaSTA ebenfalls Implementierungen von SCI-P und SCI-LS Schnittstellen enthalten. Diese wurden als ungeeignet für die Nutzung in DSTW-Sim eingeschätzt, da sie keine Trennung von SCI Schnittstelle und RaSTA-Modul aufweisen und im Falle von SCI-P nicht den Vorgaben der EULYNX-Spezifikation entsprechen, da keine Simulation der Point-Machine integriert ist und die Meldung über die erfolgte Umstellung nicht verzögert gesendet werden kann.
Die in [Eu.Doc.92] definierte Konfiguration für RaSTA ist in OS RaSTA in einer separaten Konfigurationsdatei abgelegt, welche beim Start des Programmes eingelesen wird. Die OS Rasta Instanz jedes OC benötigt eine individuelle Konfigurationsdatei, da diese auch die ID enthält, welche zum Nachrichtenversand benötigt wird. Diese Dateien werden in DSTW-Sim automatisch erstellt.

Entsprechend der Anforderungen wurde OS RaSTA um ein Modul zur OC-internen Kommunikation erweitert, um ein austauschbares System zu erhalten. Dieses sollte SCI Nachrichten von der SCI-Schnittstelle annehmen und versenden bzw. empfangen und an diese weitergeben. Als Protokoll für die interne Kommunikation wurde UDP gewählt. Die im Rahmen der Arbeit erweiterte Version von OS RaSTA soll im Folgenden als RaSTA-Modul bezeichnet werden.

Weiterhin wurde für das RaSTA-Modul des Stellwerks, welches laut [Eu.Doc.16] die Funktion des primären Kommunikationspartners übernimmt, eine Funktion zum Herstellen der Verbindung mit allen projektierten OC entwickelt.

Dazu wird in der Konfigurationsdatei des Stellwerks eine Liste der zu verbindenden OC abgelegt, welche IP-Adressen, Ports und IDs enthält. Nach der Initialisierung wird die Verbindung zu jedem der OC hergestellt. Eine Funktion zur automatischen Wiederherstellung einer abgebrochenen Verbindung wurde ebenfalls implementiert, diese wurde jedoch im Rahmen des Testes nicht benutzt, da ein Test laut dem gewählten Testszenario mit Abbruch der Verbindung bereits als gescheitert gilt.

Die RaSTA Module der OC informieren das zentrale Programm zur Steuerung der Testdurchführung innerhalb von DSTW-Sim (Experimentskript) nach erfolgreichem Hochfahren und Herstellen der Verbindung mit dem RaSTA-Modul des Stellwerks über ihre Einsatzbereitschaft.

Im Folgenden wird die Entwicklung der beiden SCI-Module (SCI-P und SCI-LS) sowie der Simulation des Stellwerks beschrieben.

7.3 Entwicklung Module SCI-P, SCI-LS und Stellwerk

7.3.1 Wahl der Programmiersprache

In der europäischen Norm EN 50128: Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Software für Eisenbahnsteuerungs- und Überwachungssysteme ist keine Programmiersprache namentlich empfohlen oder erwähnt. Stattdessen werden qualitative Erwartungen definiert. So soll die Sprache „mit einem Minimum an Aufwand zu leicht verifizierbarem Code führen [und] eher anwender- und problemorientiert [statt] maschinenorientiert sein“ [DIN28]. Zu den Erwartungen gehört außerdem der Wunsch, dass die Sprache kompiliert ist und von angemessenen Bibliotheken bereits entwickelter Software-Komponenten begleitet wird. Auf den Unterschied von interpretierten und kompilierten Programmiersprachen soll nicht genauer eingegangen werden, außer dass kompilierte Sprachen üblicherweise mit einem deutlich niedrigeren Ressourcenverbrauch auskommen als interpretierte Sprachen [AMP17]. Die Nutzung von interpretierten Sprachen wird von der Norm nicht ausgeschlossen.

Weiterhin ist in der Aufgabenstellung der vorliegenden Arbeit die direkte Anweisung vermerkt, zu entwickelnde Komponenten in einer zur Compile-Zeit[10] speichersicheren Programmiersprache zu schreiben.

Mit Speichersicherheit in Programmiersprachen wird der Ausschluss von Zugriffen auf ungewollte Stellen im Arbeitsspeicher bezeichnet. Dies kann beispielsweise durch Lesen eines Pointers auf die elfte Position in einem Array mit nur zehn Elementen passieren (out-of-bounds read). Anstatt in einem solchen Falle die zufälligen Werte auszugeben, welche sich aus vergangenen Operationen an dieser Stelle befinden und so ungeplantes Verhalten auszulösen, stürzen mit speichersicheren Sprachen geschriebene Programme ab. Sprachen ohne diese Eigenschaft übertragen die Verantwortung der sicheren Speicherverwaltung auf den Programmierer, ein Umstand, der für Programmierer viel Flexibilität bringt, aber zu Problemen mit industrieweiten Ausmaßen führt. Laut eigenen Schätzungen der Unternehmen sind 60-70% der Sicherheitslücken bei Apple, 70% der Sicherheitslücken bei Microsoft und 90% der Sicherheitslücken des von Google entwickelten Smartphone-Betriebssystems Android auf Ausnutzung von Speicherfehlern zurückzuführen [LAN19; GOO19; MIC19] [GOO19]. Zu den speichersicheren Sprachen gehören unter anderem Python und Rust, aber auch Go, C#, Java, Swift und JavaScript. Das für Hochleistungssysteme sehr populäre und auch für OS RaSTA verwendete C gehört nicht dazu [PRO; AZE18].

Ebenfalls berücksichtigt wurden Aspekte der Leistungsfähigkeit. Mit dem Ziel der Demonstration der Funktionstüchtigkeit von SCI-konformer Software auf sehr leistungsarmer Hardware sollte der Ressourcenverbrauch des entstehenden

[10] Kompilieren bezeichnet den Vorgang des Übersetzens der menschenlesbaren Programmiersprache in Maschinensprache zum Ermöglichen der Ausführung durch den Computer

Programms so klein wie möglich sein. In einem Benchmark-Test von Bugden und Alahmar (2022) [BUG22] wurden sechs Programmiersprachen auf ihre Geschwindigkeit und Arbeitsspeichernutzung bei der Bewältigung von drei arbeitsintensiven Algorithmen getestet: C, C++, Go, Java, Python und Rust. Die Sprache Rust tat sich hier als außerordentlich leistungsfähig hervor. Im Bezug auf Geschwindigkeit musste sie den ersten Platz nur bei einem der Algorithmen knapp an C abtreten, bei der Arbeitsspeichernutzung belegt sie bei allen drei Tests den zweiten Platz hinter C. Besonders groß ist der Abstand zu Java und Python. Die Autoren bestätigen damit Ergebnisse, die schon in anderen vergleichbaren Benchmarks erzielt wurden, wie von Pereira et al. (2021) [PER21]. Da C nicht zu den speichersicheren Sprachen gehört, lieferten die betrachteten Arbeiten ausreichende Gründe für die Entscheidung zur Verwendung der Programmiersprache Rust. Zusätzlich lag eine Empfehlung zur Nutzung von Rust von Dr. Iffländer vor, die in Bezug auf Benutzerfreundlichkeit der Programmiersprache getroffen wurde (vgl. Anhang B).
An die Implementierung der Simulation des Stellwerks wurden nicht der gleiche Anspruch bezüglich der Leistung gestellt, da dieses nicht auf ähnlich beschränkter Hardware ausgeführt werden soll wie die OC. Daher wurde im Interesse der Anwender- und Problemorientiertheit in Abstimmung mit Iffländer die Sprache Python gewählt.

7.3.2 Architektur der Module

Das **SCI-LS Modul** muss nach Erhalt eines Befehls vom Stellwerk lediglich den entsprechenden Signalbegriff anzeigen und das Stellwerk über die Änderung des angezeigten Signalbegriffs informieren. Solange der OC mit keinem tatsächlichen Lichtsignal verbunden ist, ließe sich das Umschalten des Signalbegriffs mit einer kurzen Wartezeit simulieren. Unter der Annahme, dass diese Umlaufzeit in modernen Lichtsignalanlagen unerheblich ist, wurde diese im Rahmen der Versuche mit DSTW-Sim jedoch auf 0 gesetzt. Das SCI-LS Modul wurde also lediglich mit einem Thread[11] umgesetzt, der wie in Abb. 7.2 dargestellt auf ankommende Nachrichten auf der internen UDP-Schnittstelle wartet, welche vom RaSTA Modul weitergeleitet werden und dann die entsprechende Antwort sendet.

Das **SCI-P Modul** muss etwas komplexer ausfallen, denn alle zu sendenden Nachrichten sind in Abhängigkeit des Status' der Point-Machine. Da im Falle der Umstellung der Weiche die volle Weichenumlaufzeit ablaufen muss, bevor die nächste Nachricht zu senden ist und in dieser Zeit unter Umständen eine neue Nachricht zu empfangen ist, darf das Programm nicht mit einer Wartezeit blockiert werden, um eine Weichenumlaufzeit zu simulieren. Die Point-Machine wurde darum in einem zweiten Thread simuliert, in dem sie unabhängig Befehle zum Umstellen empfangen und Mitteilungen über Änderungen ihres Status versenden kann, welche im SCI-P Modul PDI-Nachrichten an das Stellwerk auslösen (vgl. Abb. 7.2).

[11] Ein Thread ist ein Ausführungsstrang bei der Abarbeitung eines Programms. Per Multithreading können mehrere Threads gleichzeitig bearbeitet werden

Beide Module verfügen über Methoden zum Interpretieren und Erstellen der Telegramme, wie sie laut Spezifikation ausgetauscht werden sollen.

Die **Simulation des Stellwerks** enthält eine Liste mit Abhängigkeiten der projektierten Gleisfeldelemente für die Einstellung von Fahrstraßen, welche aus der vom Komponentenskript erstellten Konfigurationsdatei gelesen wird. Das Stellwerk kommuniziert entsprechend der SCI-Spezifikation mit den OC zum Einstellen einer Fahrstraße. Über die Funktionen zur PDI-Kommunikation hinaus verfügt es über eine Komponente zur Kommunikation mit der Experimentsteuerung. Die Nachrichten der Experimentsteuerung sind an den Test des RaSTA-Protokolls angepasst. So besteht eine Nachricht zum Start des Experimentes aus zwei Feldern: Der Anzahl an zu absolvierenden Wiederholungen und dem Experimentnamen. Nach jeder im Sinne des Testszenarios erfolgreichen Einstellung einer Fahrstraße und Freigabe der Zugfahrt erfolgt ein Vergleich der Anzahl der bereits durchgeführten Vorgänge mit der Vorgabe. Nach Durchführung der letzten Wiederholung wird eine Erfolgsnachricht an die Experimentsteuerung gesendet.

Folgende Annahmen bzw. Einschränkungen existieren im Design der Stellwerkssimulation:

- Es kann nur eine Fahrstraße zur gleichen Zeit eingestellt sein.
- Als Ausgangssituation steht nur die bereits initialisierte PDI-Verbindung zur Verfügung, d. h. die Funktionen zum Verbindungsaufbau fehlen (dies ist unabhängig vom Aufbau der RaSTA-Verbindung, das RaSTA-Modul enthält diese).
- Es wird keine technische Gleisfreimeldung berücksichtigt.
- Die Stellwerkssimulation durchläuft lediglich die Schritte, welche zum Austausch der EULYNX-konformen PDI-Nachrichten nötig sind. Mechanismen wie die Festlegung einer Fahrstraße wurden nicht umgesetzt.
- Das Teilsystem zur Experimentsteuerung kann lediglich abwechselnd Fahrstraßen für eine bestimmte Anzahl an Wiederholungen einstellen

8 Experimentdurchführung und Messergebnisse

In diesem Kapitel werden die Prozesse der Experimentdurchführung sowie die Messungen bzw. Messversuche detailliert beschrieben und die Ergebnisse präsentiert, bevor diese im nächsten Kapitel ausgewertet und diskutiert werden.

8.1 Experimentdurchführung

Das **Experimentskript** zur Steuerung der Testdurchführung auf Python-Basis steuert die Vorgänge auf den VMs über das Secure Shell Protocol (SSH), einer Standard-Schnittstelle zur sicheren Fernkontrolle von Computern. Die Testdurchführung folgt einem generischen Ablauf:

1. Spezifikation der Tests durch den Nutzer
2. Hochfahren bzw. Zurücksetzen aller VMs
3. Experiment-Schleife:
    a. Herstellen des Ausgangszustands (Gegebenenfalls Herunterfahren der Docker Container in den VMs)
    b. Start der OC-Module und des Stellwerk-Moduls (Hochfahren der Docker Container in den VMs)
    c. Konfigurieren der Netzwerkkonditionen aller Netzwerkadapter im Netz
    d. Durchführen der Tests
4. Einsammeln der Testergebnisse in Form von Logdateien
5. Herunterfahren der VMs bzw. Standby dieser für weitere Tests

Die Messung der Systemlaufzeiten erfolgte nach dem Vorbild des Vergabeverfahrens des DKS Stuttgart möglichst weit aufgeschlüsselt (vgl. Kapitel 3.3). Dazu wurde die jeweils im Teilsystem verstrichene Zeit durch Log-Einträge davor und danach erfasst.
Die Fahrstraßenbildezeit besteht im Testszenario aus den in Abb. 8.1 dargestellten Schritten.

Wie im vorigen Kapitel beschrieben, verfügt die Simulation des Stellwerks über einen Verschlussplan. Auf die Anweisung zum Stellen einer Fahrstraße werden die aktuellen Stellungen der beweglichen Fahrwegelemente mit den erforderlichen verglichen und falls nötig Befehle zum Umstellen ausgesendet. Die entsprechenden OC antworten zunächst mit einer „Keine Endlage“-Nachricht, welche keine Auswirkungen nach sich zieht, und senden nach Ablauf der Weichenumlaufzeit eine Bestätigung der angeforderten Endlage. Sind alle Elemente in der benötigten Endlage angekommen, wird die Fahrstraße als festgelegt angenommen. Es werden nun an alle OC der zur Fahrstraße gehörenden Signale, welche einen Fahrtbegriff zeigen sollen, Nachrichten zum Befehl der Fahrtstellung gesendet. Nachdem die Bestätigung über die Fahrtstellung in der Stellwerkssimulation angekommen ist, ist die Fahrstraßenbildung beendet.

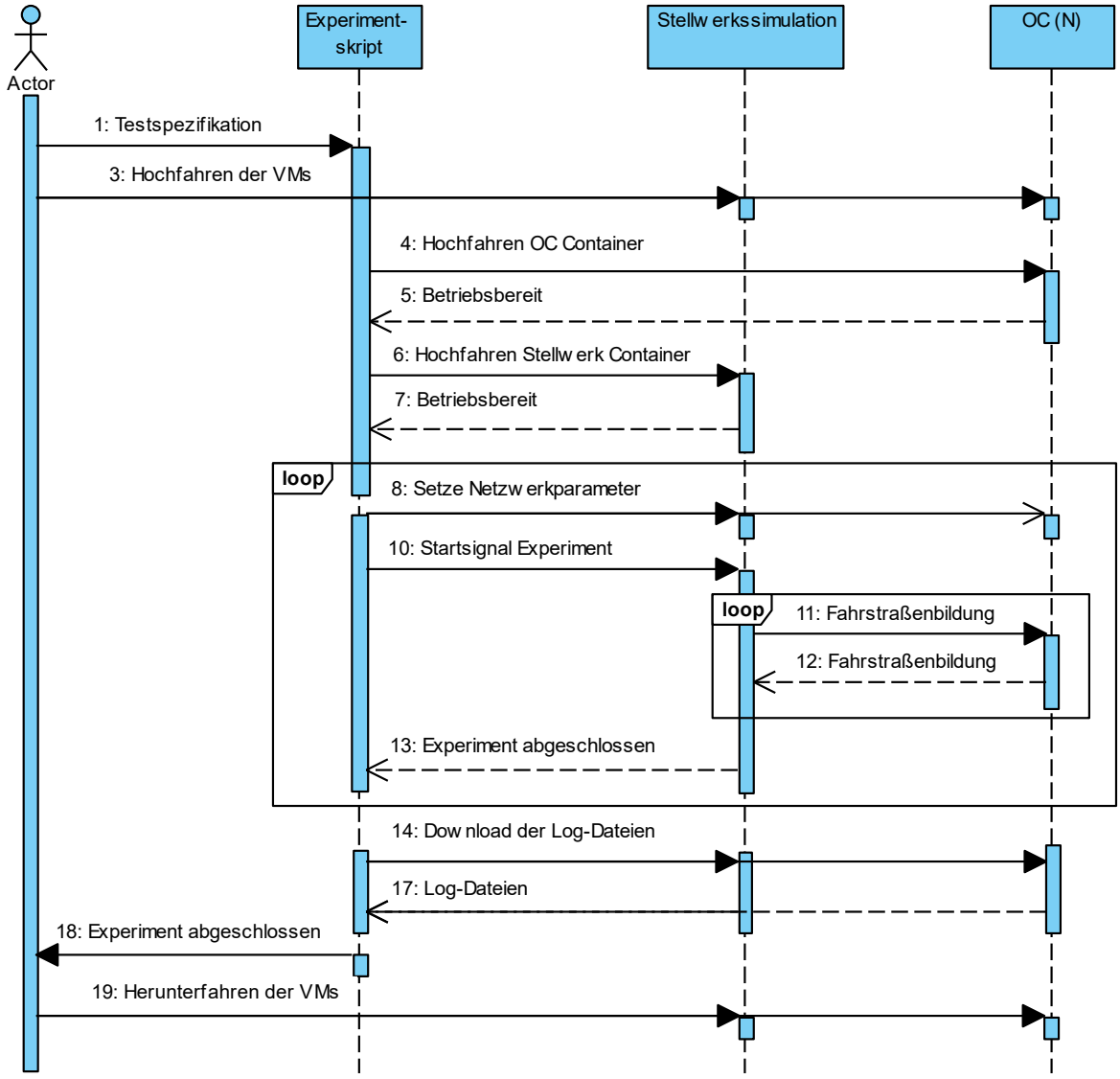

Abb. 8.1: Experimentdurchführung in DSTW-Sim

Zu Beginn der Einstellung einer Fahrstraße wird ein zufälliger Code generiert, der an alle mit dieser Fahrstraße verbundenen Telegramme angehangen und in den Logeinträgen genannt wird, um die Zuordnung dieser zum selben Vorgang in der Auswertung zu ermöglichen. Die Zusammensetzung der Logeinträge lässt die eindeutige Einteilung eines Eintrages als einer der Schritte zu, diese ist in Anhang A aufgeführt. Die Berechnung der Dauer eines Schrittes ergibt sich aus der Bildung der Differenz von aufeinanderfolgenden Schritten.

Die Messungen wurden auf dem Cyber-Security Testsystem des Deutschen Zentrum für Schienenverkehrswesen durchgeführt. Das Computersystem verfügt über einen AMD EPYC 7313 16-Core Prozessor und 64GB Arbeitsspeicher.

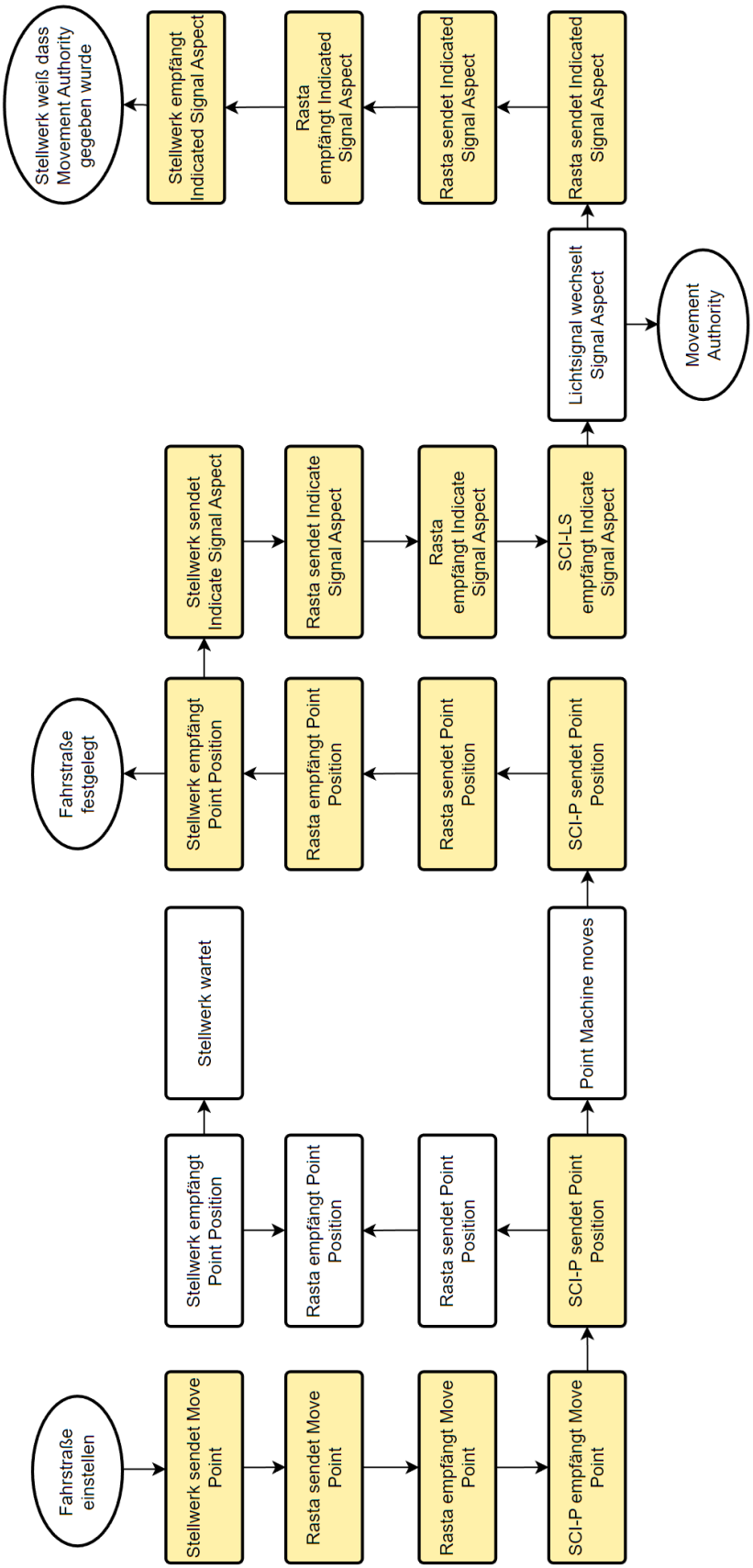

Abb. 8.2: Schritte der Fahrstraßenbildung mittels der Stellwerksimulation in DSTW-Sim

8.2 Messergebnisse

8.2.1 Synchronisation der Uhren

Um den Einfluss der Uhrensynchronisation festzustellen, wurde ein Messaufbau ohne Beeinflussung der Netzwerkparameter gewählt, bei dem der Ablauf der Fahrstraßenbildung mit 100 Wiederholungen durchgeführt wurde. In Versuch 1 war die Uhrensynchronisation deaktiviert, in Versuch 2 ist sie aktiviert.

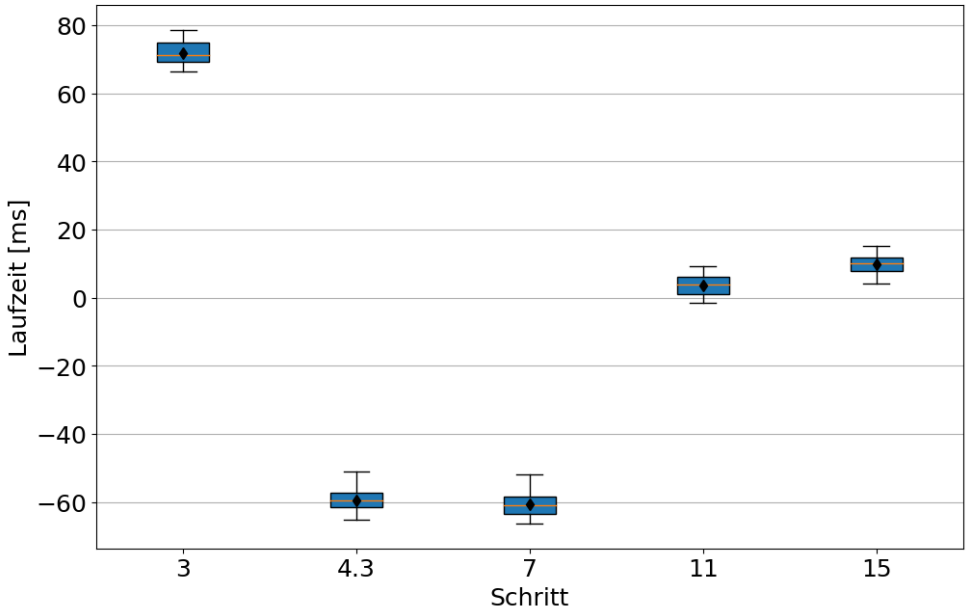

Abb. 8.3: **Messergebnisse der RaSTA-Laufzeiten bei deaktivierter Uhrensynchronisation**

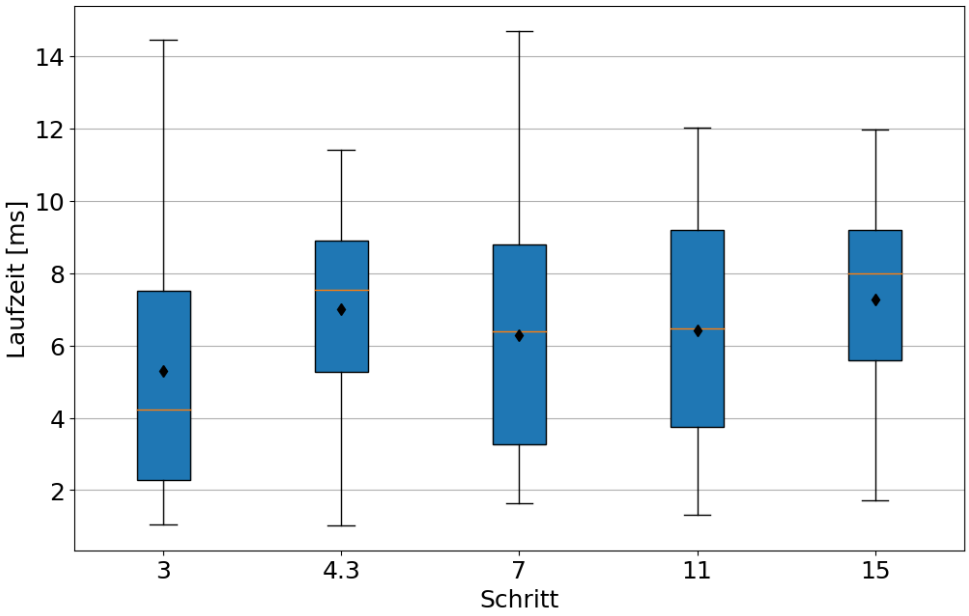

Abb. 8.4: **Messergebnisse der RaSTA-Laufzeiten bei aktivierter Uhrensynchronisation**

8.2.2 Latenz

Zum Testen des Einflusses der Latenz des Netzwerkes wurden Messungen mit steigenden Latenzen durchgeführt, beginnend bei 0. Die Latenz wurde in Schritten von 10 ms erhöht, bis die Verbindung abbrach und auch nach fünf Testversuchen keine vollständige Testdurchführung mehr absolviert werden konnte.
Ebenfalls getestet wurden Störungen von nur entweder der grauen oder der blauen Netzwerkebene.

Die RaSTA-Kommunikation erfolgt jeweils in Schritt 3, 4.3, 7, 11 und 15 des Ablaufs der Fahrstraßenbildung in der Simulation. Diese markieren folgende Laufzeiten:

- 3: Übertragung Nachricht MOVE_POINT zum OC SCI-P
- 4.3: Übertragung Nachricht POINT_POSITION zur Stellwerkssimulation
- 7: Übertragung Nachricht POINT_POSITION zur Stellwerkssimulation
- 11: Übertragung Nachricht INDICATE_SIGNAL_ASPECT zum OC SCI-LS
- 15: Übertragung Nachricht INDICATED_SIGNAL_ASPECT zur Stellwerkssimulation

Die Ergebnisse sind aufgeteilt nach Laufzeiten über das Netzwerk mittels RaSTA und den Systemlaufzeiten innerhalb der OC bzw. innerhalb der Stellwerkssimulation. Als Boxplot-Diagramme abgebildet sind die RaSTA-Laufzeiten unter der Störung beider Netzwerkebenen durch Latenzen von 0ms, 30ms und 60ms (siehe. Abb. 8.5, Abb. 8.6, Abb. 8.7), sowie die Systemlaufzeiten bei 0ms und 60ms (siehe Abb. 8.9, Abb. 8.10). Die über alle Schritte gemittelten RaSTA-Laufzeiten im Verhältnis zu den Latenzen sind in Abb. 8.8 als Punktdiagramm für alle erfolgreich durchgeführten Versuche dargestellt.
Die RaSTA-Laufzeiten unter Beeinträchtigung von nur einer der beiden Netzwerkebenen sind in Tab. 8.3 abgebildet. Die durchschnittlichen Laufzeiten aller durchgeführten Ergebnisse werden in Tab. 8.1, die summierten Laufzeiten in Tab. 8.2 präsentiert.

RaSTA Laufzeiten

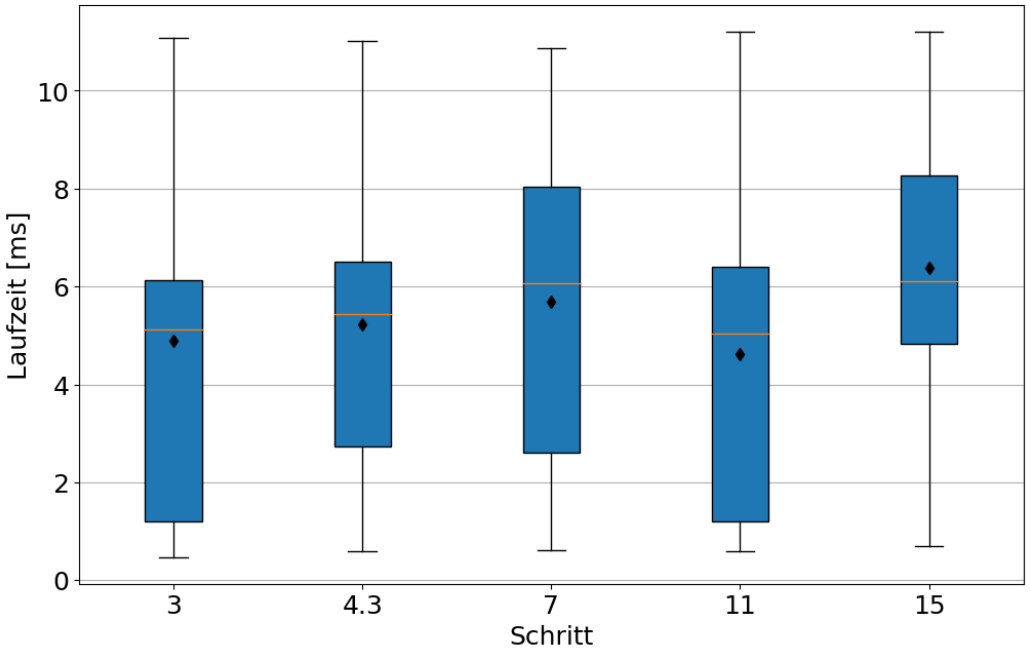

Abb. 8.5: **Messergebnisse der RaSTA-Laufzeiten bei einer Netzwerklatenz von 0 ms**

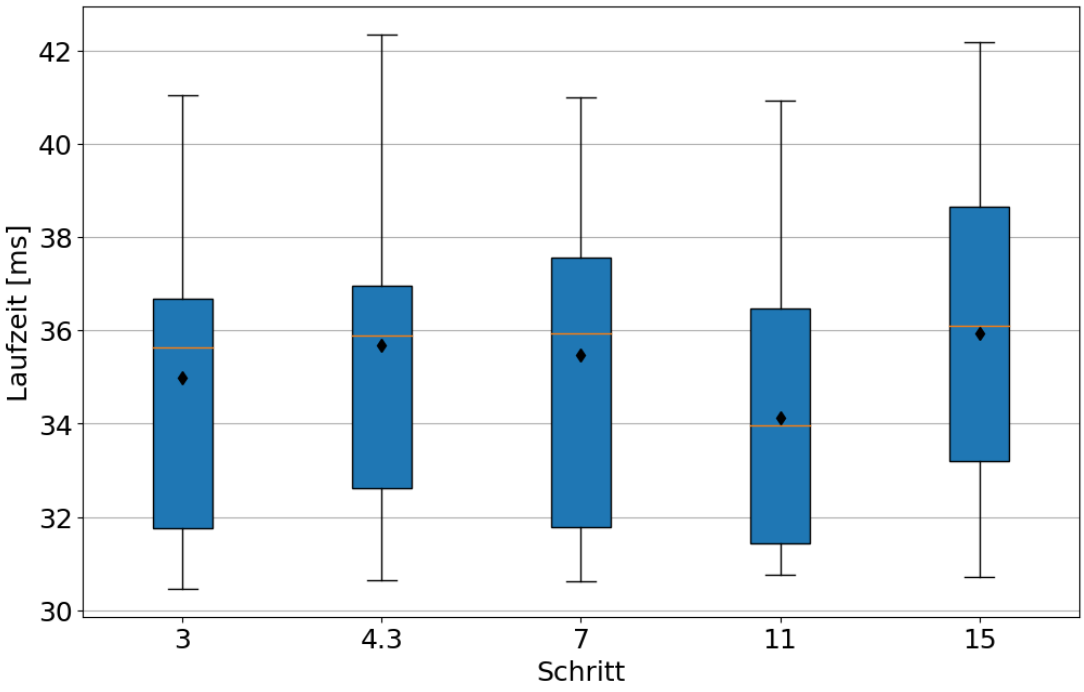

Abb. 8.6: **Messergebnisse der RaSTA-Laufzeiten bei einer Netzwerklatenz von 30 ms**

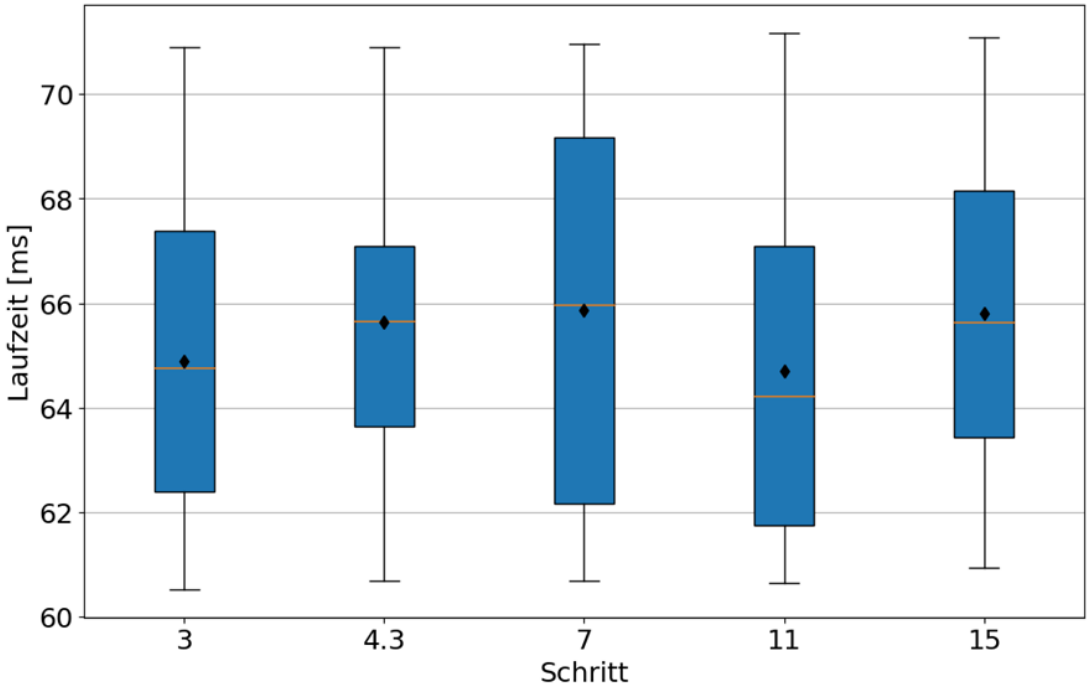

Abb. 8.7: Messergebnisse der RaSTA-Laufzeiten bei einer Netzwerklatenz von 60 ms

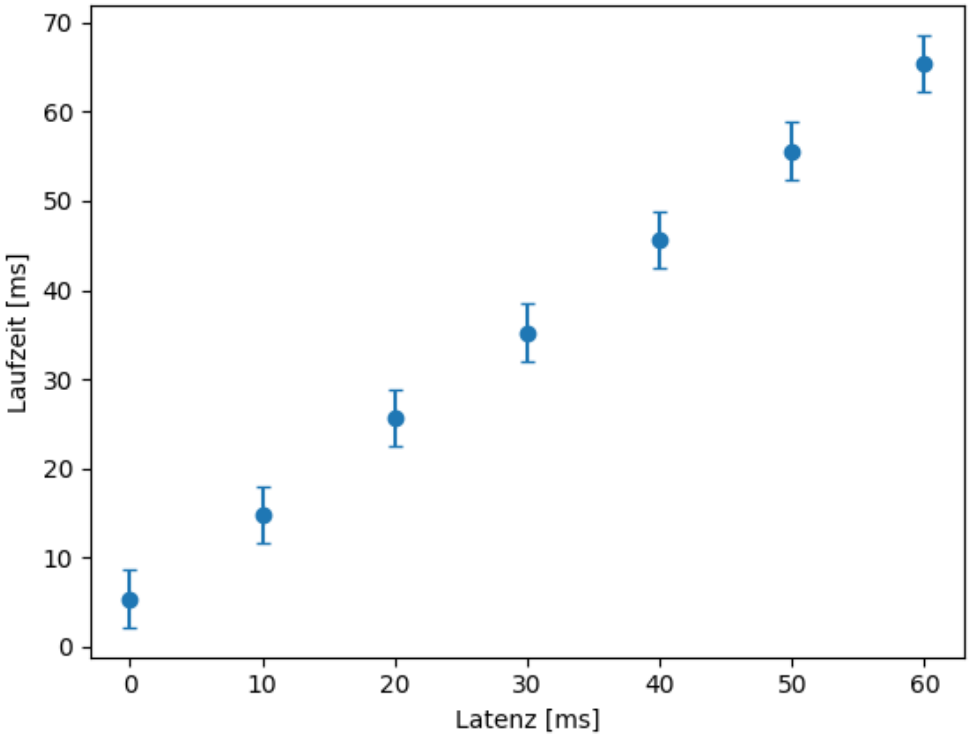

Abb. 8.8: Durchschnittliche RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz

	0	0,562	10	20	30	40	50	60	70
3	4,897	5,953	14,226	25,627	34,979	45,903	55,366	64,893	Fehler
4.3	5,230	5,78	14,831	26,037	35,689	46,300	56,237	65,638	Fehler
7	5,697	6,883	15,301	25,551	35,473	45,593	55,235	65,862	Fehler
11	4,615	6,09	13,647	25,14	34,129	44,156	54,420	64,711	Fehler
15	6,380	7,129	15,773	25,887	35,933	46,263	56,656	65,811	Fehler

Tab. 8.1: Durchschnittliche RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz

Latenz	Laufzeit [ms]
0	26,818
0,562	31,834
10	73,778
20	128,242
30	176,203
40	228,214
50	277,914
60	326,915

Tab. 8.2: Summierte RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz

	Blau beeinträchtigt		Grau beeinträchtigt	
	0	**100**	**0**	**100**
3	4,641	5,092	4,533	4,935
4.3	5,043	5,689	4,872	5,456
7	5,253	5,442	5,891	5,563
11	4,990	4,541	3,659	4,520
15	6,367	6,665	6,475	6,035

Tab. 8.3: Durchschnittliche RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz, Beeinträchtigung einer einzelnen Netzwerkebene

Systemlaufzeiten

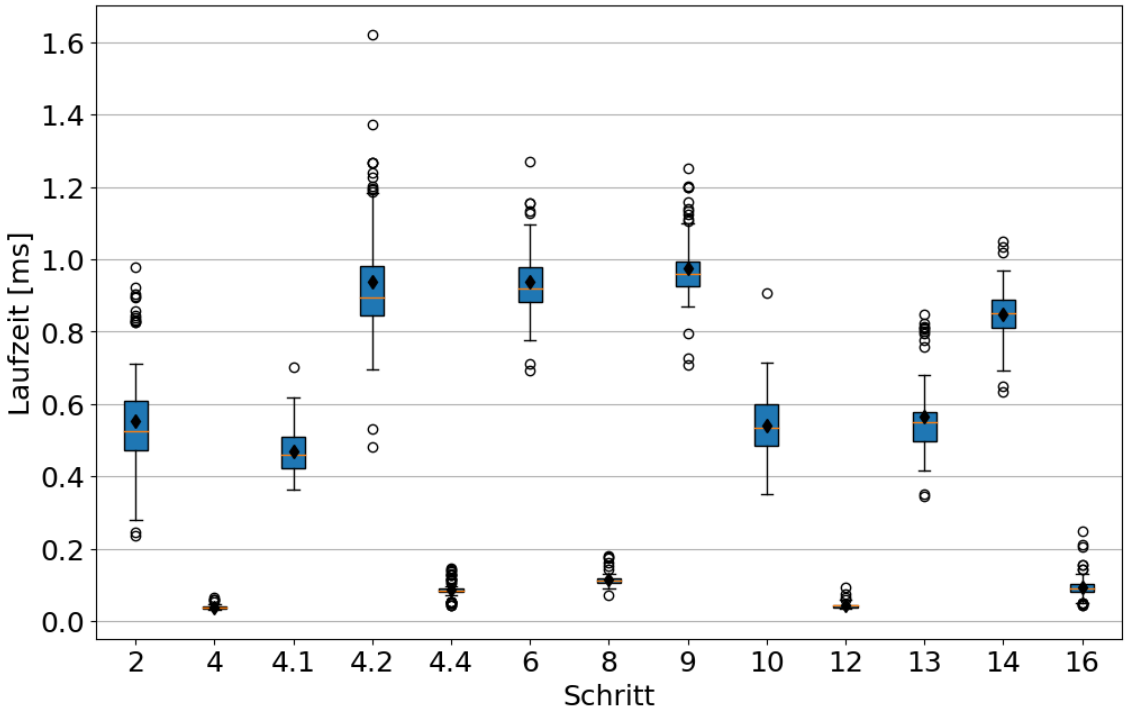

Abb. 8.9: Messergebnisse von Systemlaufzeiten bei einer Netzwerklatenz von 0 ms

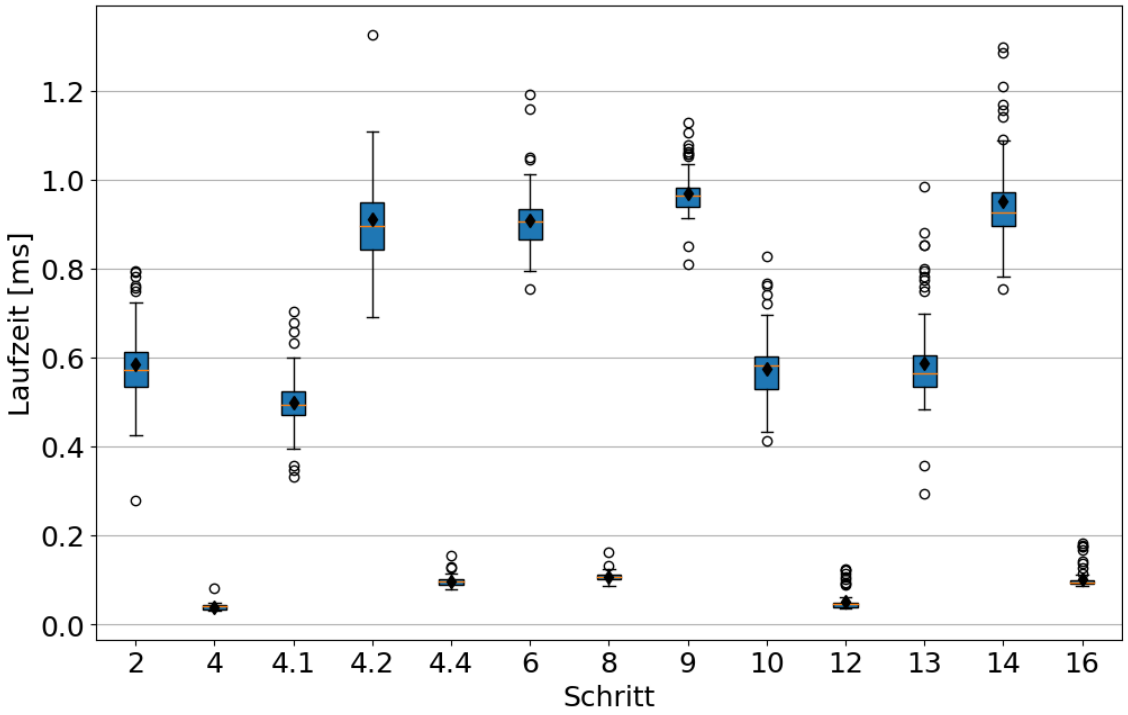

Abb. 8.10: Messergebnisse von systeminternen Laufzeiten bei einer Netzwerklatenz von 60 ms

Latenz [ms]	Laufzeit [ms]
0	6,349
0,562	6,606
10	6,341
20	6,675
30	6,617
40	6,367
50	6,408
60	6,447

Tab. 8.4: Summierte Systemlaufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz

8.2.3 Paketverlustrate

Analog zum Test des Einflusses von Latenzen im Netzwerk wurden Messungen mit steigender Paketverlustrate durchgeführt. Die Paketverlustrate wurde in 0,1% Schritten bis 1% und dann in 0,5% Schritten weiter erhöht, bis die Testdurchführung auch nach fünf Testversuchen nicht mehr erfolgreich abgeschlossen werden konnte. Aufgrund der Abhängigkeit dieses Versuches von Wahrscheinlichkeiten wurde mit 200 eine doppelt so große Anzahl an Wiederholungen gewählt.

Um die Vergleichbarkeit mit dem bb-IP-Netz zu erhöhen, wurden auf Basis der vorliegenden Informationen über die Qualitätseigenschaften dessen die Netzwerklatenz für die Paketverlustraten-Versuche auf 0,562 ms festgelegt. Dieser Wert wurde aufgrund der typischen RaSTA-Paketgrößen genutzt.

Paketverlustrate [%]	Laufzeit [ms]
0,0	31,834
0,1	32,335
0,2	32,336
0,3	Fehler
0,4	32,614
0,5	31,548
0,8	32,472
1,0	Fehler
1,5	Fehler
2,0	Fehler

Tab. 8.5: Summierte RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Paketverlustrate

8.2.4 CPU-Zeit / RAM Auslastung

Zur Erhebung der Hardwareanforderungen eines OC wurde der Ressourcenverbrauch der VMs auf dem Gastgebersystem gemessen. Der Hypervisor VirtualBox stellt zu diesem Zweck ein Kommandozeilenwerkzeug, **VBoxManage**, zur Verfügung, in dem eine Funktion zur Sammlung von Metriken der virtualisierten Systeme

enthalten ist. Diese Metriken wurden während der Messung im Zeitabstand von einer Sekunde abgefragt und in eine Textdatei gespeichert. DSTW-Sim erstellt aus dieser automatisch ein Liniendiagramm zur graphischen Interpretation. In Abbildung Abb. 8.11 ist der Ressourcenverbrauch der SCI-LS VM „Signal1“ über den Zeitraum einer Messung ohne Netzwerkbeeinträchtigung dargestellt.
Aufgezeichnet wurde der Anteil der CPU-Zeit des Gastgebersystems, der für die VM aufgewendet wurde, und die Größe des von der VM belegten Bereichs des Arbeitsspeichers des Gastgebersystems.

	Messwert
CPU-Auslastung durchschnittlich	1,15%
CPU-Auslastung maximal	1,22%
Arbeitsspeicher durchschnittlich	92,32 MB
Arbeitsspeicher maximal	92,53 MB

Tab. 8.6: CPU-Auslastung / Arbeitsspeicherverbrauch in „Signal1“ während Simulation der Fahrstraßenbildung

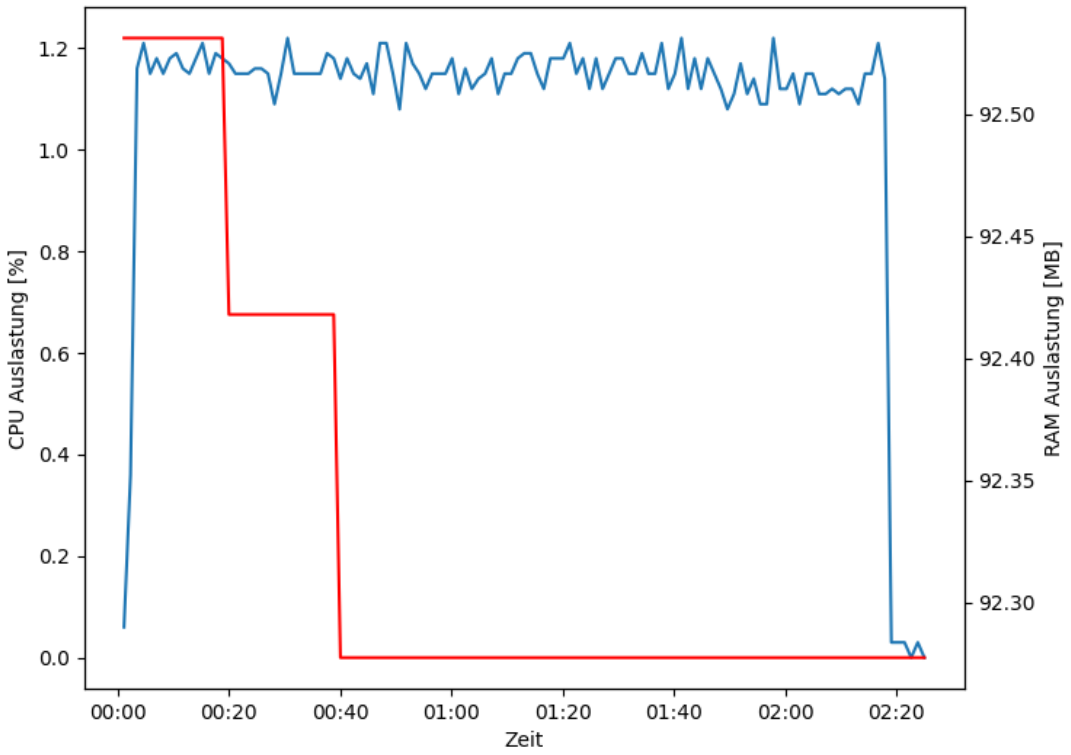

Abb. 8.11: CPU-Auslastung / Arbeitsspeicherverbrauch in „Signal1“ während Simulation der Fahrstraßenbildung

8.2.5 Bandbreite

Die Datenrate wurde mittels der Durchführung von 100 Wiederholungen der Fahrstraßenbildung an beiden Netzwerkadapter der Simulation des Stellwerks gemessen. Die Stellwerkssimulation war dabei über die blaue und graue Netzwerkebene über eine RaSTA Verbindung mit einer SCI-P Schnittstelle und einer SCI-LS Schnittstelle verbunden.
Für die Messung wurde das Kommandozeilenwerkzeug **iftop** genutzt, welches Funktionen zur Überwachung des Netzwerkverkehrs bereitstellt [IFTOP]. Zu Beginn jedes Experiments wird pro Netzwerkadapter eine Instanz dieses Programms gestartet und zum Ende wieder geschlossen. Die tabellarische Ausgabe am Ende des Experiments ist in Abb. 8.12 dargestellt. Die pro Sekunde übertragenen Daten in der Einheit Bytes werden aufgeschlüsselt nach dem Kommunikationspartner in drei unterschiedlichen Zeiträumen (letzte zwei Sekunden, letzte zehn Sekunden, letzte vierzig Sekunden) präsentiert. Zusätzlich sind die über die vollständige Dauer des Versuches übertragene Datenmenge und die aufsummierten Werte für die ein- und ausgehende Datenrate für den Netzwerkadapter insgesamt angegeben.
Die SCI-P Schnittstelle trägt die IP-Adresse 10.152.1.3, der SCI-LS Schnittstelle wurde die IP-Adresse 10.152.1.4 zugewiesen.

```
   # Host name (port/service if enabled)            last 2s   last 10s   last 40s cumulative
--------------------------------------------------------------------------------------------
   1 vagrant                                 =>        467B       460B       455B     22.1KB
     10.152.1.3                              <=        790B       783B       770B     36.9KB
   2 vagrant                                 =>        467B       460B       455B     21.9KB
     10.152.1.4                              <=        467B       460B       455B     21.9KB
--------------------------------------------------------------------------------------------
Total send rate:                                      934B       920B       911B
Total receive rate:                                 1.23KB     1.21KB     1.20KB
Total send and receive rate:                        2.14KB     2.11KB     2.09KB
--------------------------------------------------------------------------------------------
Peak rate (sent/received/total):                      934B     1.23KB     2.14KB
Cumulative (sent/received/total):                   44.0KB     58.8KB      103KB
============================================================================================
```

Abb. 8.12: Datenrate und kumulierte Datenmenge für den blauen Netzwerkadapter der VM zur Stellwerkssimulation über die Dauer eines Experimentes

9 Diskussion

9.1 Weiterentwicklung von Open Source RaSTA

Beim ersten Zugriff auf OS RaSTA durch den Verfasser über GitHub.com stand die Bibliothek in Form des aktuellsten Commit[12] vom 18.07.2022 zur Verfügung [OSRASTA]. Auf Basis der darin inkludierten Beispiele begann die Entwicklung von RaSTA-Modul. Während der Entwicklungszeit wurde am 28.07.2022 ein neuer Commit unter dem Namen „Switch to single thread event system“ veröffentlicht, mit Einführung dessen sich die mitgelieferten Beispiele nicht mehr ausführen ließen. Diese wurden im Rahmen der neuen Version nicht aktualisiert und die Dokumentation ebenfalls nicht angepasst. Dieser Commit wurde deswegen bei der Entwicklung von RaSTA-Modul ausgelassen. Weitere Commits wurden während der Bearbeitungszeit nicht hinzugefügt.
Dieses Ereignis markierte das erste von mehreren Problemen welche im Verlauf der Arbeit mit der Bibliothek auftraten. Auch wenn die Bibliothek keinen Anspruch auf Richtigkeit oder Vollständigkeit erhebt, soll im Folgenden trotzdem kurz auf einige dieser Problemlagen (Dokumentationsfehler, Doppeluhrenprinzip, Logging, Verbindungsaufbau mit mehreren Teilnehmern) eingegangen werden.

9.1.1 Dokumentationsfehler

Das Build, d.h. das Überführen des Quellcodes in eine ausführbare Datei durch ein Übersetzungsprogramm, auch Compiler genannt, ist immer der erste Schritt bei der Nutzung bzw. Entwicklung von kompilierter Software. In der Dokumentation wurde die Anfertigung eines Builds mit dem Software-Werkzeug „Gradle“ beschrieben. Es existierten jedoch keine Konfigurationsdateien für dieses Werkzeug in der Bibliothek. Diese waren bereits in einem mehr als ein Jahr zurückliegenden Commit entfernt worden.

9.1.2 Doppeluhrenprinzip

Zu Beginn der ersten Verbindungstests zwischen zwei RaSTA-Teilnehmern brach die Verbindung verlässlich während des Austauschs der ersten zehn Heartbeat-Nachrichten ab, manchmal bereits bei der zweiten Nachricht. Eine Untersuchung des Problems durch im Quellcode platzierte Log-Nachrichten förderte eine genauere Beschreibung zutage. Die korrekte Implementierung des Algorithmus vorausgesetzt, resultiert dies in einem Zusammenhang wie in Abb. 9.1 graphisch dargestellt. So konnte es vorkommen, dass der Wert für den bestätigten Zeitstempel einer ankommenden Nachricht (im Beispiel 14) älter war als der zuvor empfangene Wert, der bereits einmal bestätigt worden war (im Beispiel 21).

[12] Ein „Commit“ bezeichnet eine einzelne Änderung/Version von Software im Entwicklungsstadium, im Gegensatz zu einem „Release“, was eine offiziell veröffentlichte und (vermeintlich) stabile Version bezeichnet.

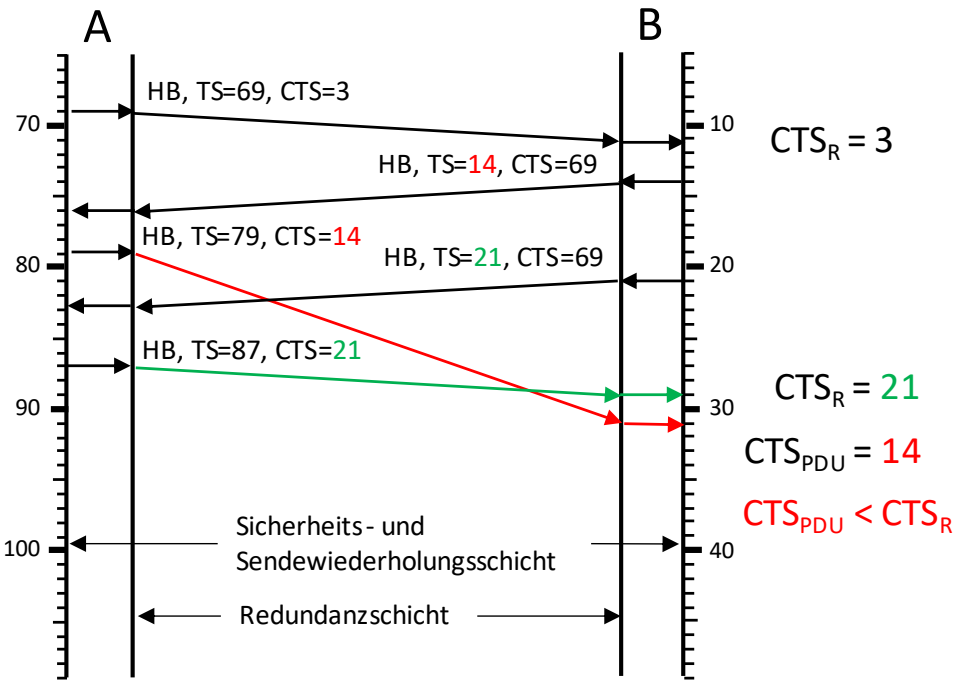

Abb. 9.1: Graphische Beschreibung des OS RaSTA Problems „Doppeluhrenprinzip“

Die Entstehung eines solchen Falles ist bei einem korrekt eingerichteten und nicht beeinträchtigten Netzwerk, korrekt implementierten Protokollen und fehlerfrei funktionierenden Netzwerkgeräten beinahe unmöglich. Nachdem sämtliche mögliche Fehlerquellen ohne berechtigten Zweifel beseitigt waren, sowohl externer Natur als auch durch die vom Verfasser bis dahin vorgenommenen Änderungen, wurde der Quellcode von OS RaSTA mit der RaSTA-Norm verglichen, bis eine wichtige Abweichung festgestellt wurde. Laut Norm ist vorgesehen, während des Initialisierungsprozesses einer Verbindung den Wert des lokal gespeicherten letzten bestätigten Zeitstempels auf die lokale Zeit zu setzen („CTS_R wird mit T_{local} initialisiert“ [RAS15]). Dies geschieht nur ein einziges Mal, wurde in OS RaSTA jedoch für jede neue ankommende Heartbeat-Nachricht durchgeführt (vgl. `rasta_new.c` Commit:„Fix CTS_R bug“ [GIT22]). Die lokale Uhrzeit wurde anstelle des bestätigten Zeitstempels nun an den Kommunikationspartner zurückgeschickt.
Auf diese Art und Weise wurde das maximale Zeitfenster für das rechtzeitige Eintreffen einer neuen Nachricht auf einen Bruchteil reduziert, aber zusätzlich wurde dadurch auch das Doppeluhrenprinzip verletzt, da nun die lokale Zeit mit der Uhr eines anderen Teilnehmers verglichen wurde. Durch die Abhängigkeit von der Differenz der Uhrzeiten lässt sich das unvorhersehbare Auftreten der Verbindungsabbrüche erklären.
Dass der Fehler bisher noch nicht entdeckt worden war, könnte daran gelegen haben, dass OS RaSTA bisher nur mittels Instanzen getestet wurde, die auf ein und demselben Computersystem ausgeführt wurden, oder sich in Containern auf demselben Computersystem befanden und darum dieselbe lokale Uhrzeit hatten, und nicht wie in DSTW-Sim mittels Instanzen auf unterschiedlichen VMs.

Dieser Fehler war in der letzten Version von OS RaSTA zum Ende der Bearbeitungszeit noch enthalten. Vom Verfasser wurde ein Pull-Request[13] mit dem ausgebesserten Quellcode zur Mitarbeit an der Open Source Entwicklung des RaSTA-Protokolls eingereicht.

9.1.3 Logging

Das Logging in OS RaSTA funktioniert mittels einer FIFO-Warteschlange[14]. Log-Nachrichten von allen Teilsystemen des Programms werden in diese Warteschlange aufgenommen und bei Kapazitäten im selbstständigen Logging-Thread entweder in die Standardausgabe (die Konsole) ausgegeben, in die Log-Datei gespeichert oder beides. Hier gab es ein weniger bedeutendes Problem in der Konfiguration. Die Unterscheidung funktionierte nicht, bei der Einstellung für beide Ausgaben fiel das Logging vollständig aus.

Wesentlich wichtiger war jedoch die Größe der Warteschlange. Diese war mit zehn Elementen definierten. Falls eine neue Nachricht geloggt werden sollte, wurde die Belegung der Warteschlange überprüft und die zu loggende Nachricht bei vollständiger Belegung ohne Benachrichtigung oder andere Indikation verworfen. Aufgrund der Anzahl an gleichzeitig zu loggenden Einträgen wurde diese Beschränkung bei den Testläufen des Verfassers immer wieder erreicht und so fehlten entscheidende Log-Nachrichten scheinbar ohne Zusammenhang. Da die zu gewinnenden Messergebnisse auf dem Logging basierten, musste das Log-System sehr verlässlich sein. Zur Lösung dieses Problems musste jedes Teilsystem, welches etwas mit dem Logging zu tun hatte, untersucht werden, da es sich sehr schwer reproduzieren ließ, weil es in den für die Fehlersuche kreierten, isolierten Fällen mit wenig Log-Belastung nicht auftrat. Außer einer Benachrichtigung über einen Warteschlangen-Überlauf wurde die Warteschlange außerdem auf 100 Elemente erweitert (vgl. Commit: „increase logging buffer" [GIT22b]).

9.1.4 Verbindungsaufbau zu mehreren Teilnehmern

Beim Versuch des Verbindungsaufbaus mit mehreren RaSTA-Teilnehmern wurde die RaSTA-ID des ersten Teilnehmers immer mit der des zuletzt hinzugefügten Teilnehmers überschrieben. Zur Lösung des Problems musste lediglich eine Zeile des Quellcodes entfernt werden. Dass dieser Fehler unbemerkt geblieben war, lag an der Reihenfolge der Teilnehmer beim Verbindungsaufbau in den Beispielen. Wie bereits genannt liegt es laut EULYNX Spezifikation in der Verantwortung des primären Kommunikationspartners und damit des Stellwerks, die Verbindung zu anderen aufzubauen, in den o.g. Beispielen war es umgekehrt eingerichtet.

[13] Bereitstellen von selbstgeschriebenem Quellcode zu einer Software-Bibliothek, welcher angenommen oder abgelehnt werden kann

[14] First-in-first-out: Elemente, die am längsten warten, werden als erstes bearbeitet

9.2 Diskussion der Messergebnisse aus den Experimenten

9.2.1 Uhrensynchronisation

Auf Basis der in Kapitel 6.1 eingeführten Informationen über die Handhabung der internen Uhren von VMs im Hypervisor VirtualBox wurde anfangs angenommen, dass bei rein virtuellen Tests keine externe Synchronisation nötig sei. In Experimenten, welche im Rahmen der Validierung von DSTW-Sim durchgeführt wurden, konnte allerdings beobachtet werden, dass es durchaus zu erheblichen Verwerfungen zwischen den verschiedenen Uhren kommt (vgl. Abb. 8.3).
Ein Grund dafür könnte in einer zu groben Granularität der Synchronisation durch VirtualBox liegen, welche weder angegeben ist noch in Erfahrung gebracht werden konnte. Die Implementierung einer externen Synchronisation wäre bei der Verwendung von physischen Computern ohnehin nötig, stellte sich so aber als unvermeidbar heraus. Mit der Synchronisierung durch linuxptp ließ sich eine größere Genauigkeit erreichen.

9.2.2 Latenz

Anhand der in der RaSTA-Norm beschriebenen Eigenschaften des RaSTA-Protokolls wurden folgende Erwartungen an die Laufzeitmessungen unter steigender Netzwerklatenz formuliert: Die Laufzeit der Nachrichten sollte analog zu der dem Netzwerk unterworfenen Latenz linear ansteigen, bis die Laufzeit einer einzelnen Nachricht den Parameter T_{max}, das maximal zulässige Alter einer Nachricht, übersteigt. Dann sollte ein Verbindungsabbruch erfolgen, unter der Angabe des Fehlers „T_i - Timer expired - send DisconnectionRequest“. Wie in Kapitel 7.2.1 beschrieben ist T_{max} laut EULYNX Spezifikation auf 1800 ms zu setzen und diese Zeit sollte mit der Latenz auch annährungsweise zu erreichen sein.

Dank der vollständigen Redundanz des Netzwerkes in Form von blauer und grauer Netzwerkebene sollte die Beeinträchtigung nur einer der beiden Ebenen keinen Effekt auf die Laufzeiten haben, da die Pakete der jeweils unbeeinträchtigten Ebene die Netzwerkteilnehmer mit derselben Geschwindigkeit wie vorher erreichen.
Der Effekt der Redundanz konnte im Testlauf unter Beeinträchtigung von nur einer der beiden Netzwerkebenen gezeigt werden (vgl. Tab. 8.3). Diese Beobachtung wurde als Bestätigung von sowohl der Funktionsfähigkeit der Redundanzschicht als auch des Netzwerkdesigns innerhalb von DSTW-Sim interpretiert. Im Anschluss wurde von weiteren Experimenten mit einseitiger Beeinträchtigung des Netzwerks abgesehen.

Die übrigen Ergebnisse fielen nicht vollständig erwartungsgemäß aus. Zwar zeigten die Laufzeiten bis zu einer Latenz von 60 Sekunden einen linearen Anstieg, doch bereits ab einer Latenz von 70 Sekunden kam es zum Verbindungsabbruch. Die damit einhergehende Fehlermeldung entsprach ebenfalls nicht der Erwartung. So wurden mehrere Pakete mit der Fehlermeldung „Received packet cs invalid“ verworfen, bis zum Erreichen des Ablaufzeitpunktes von T_{max} und damit einhergehendem

Verbindungsabbruch. Die Gründe für dieses Programmverhalten von OS RaSTA[15] sind unklar. Durch die Nutzung von DSTW-Sim ist es möglich, dieses Problem zu isolieren und zu beheben, ähnlich dem Verfahren bei der Fehlersuche bezüglich des Doppeluhrenprinzips (vgl. Kapitel 9.1.2).

```
[(1669246802330539)][MEASURE][RASTA_SENT] 64-11-1-70*0*0*0+2+nCbG4YtAUXbp8htBa3nTkr+links2
[(1669246802733310)][ERROR][RaSTA RECEIVE] Received packet cs invalid
[(1669246803035665)][ERROR][RaSTA RECEIVE] Received packet cs invalid
[(1669246803338056)][ERROR][RaSTA RECEIVE] Received packet cs invalid
[(1669246803640597)][ERROR][RaSTA RECEIVE] Received packet cs invalid
```

Abb. 9.2: Abbildung der Log-Einträge eines OC zum Zeitpunkt des Verbindungsabbruchs

Die gemessene Leistung von OS RaSTA ist ausreichend für eine sichere Kommunikation von SCI-XX Schnittstellen gemäß den Anforderungen der EULYNX Spezifikation für das Profil Standard Wired mit einer niedrigeren Latenz als 50 ms. Latenzen von 70 ms und höher führen zum unerwarteten Verbindungsabbruch. Vor dem Einsatz in einem Produktionsumfeld sollte das unerwartete Verhalten in jedem Fall genauer untersucht und etwaige weitere Fehler im Quellcode gegebenenfalls behoben werden.

9.2.3 Systemlaufzeiten

Die internen Systemlaufzeiten wurden aufgeschlüsselt nach den in Anlage A genannten Schritten erfasst. Um auszuschließen, dass eine Abhängigkeit dieser von den Netzwerkbedingungen vorliegt, wurde ein T-Test durchgeführt. Der Freiheitsgrade von Zweistichproben-t-Test für unabhängige Stichproben sind nach $n_1 + n_2 - 2 = 100 + 100 - 2 = 198$. Die für ein Signifikanzniveau von 97,5% nötigen P-Werte für den Freiheitsgrad 198 von 1,97 wurden für keinen der Schritte erreicht. Die Latenz hat also keinen Einfluss auf die Systemlaufzeiten.

Schritt	0-30	30-60	0-60
2	0,326	0,001	0,116
4	0,000	0,037	0,062
4.1	0,096	0,492	0,006
4.2	0,179	0,732	0,139
4.4	0,004	0,508	0,000
6	0,509	0,116	0,061
8	0,525	0,160	0,049
9	0,271	0,213	0,279
10	0,353	0,005	0,404
12	0,010	0,772	0,006
13	0,120	0,277	0,119
14	0,001	0,494	0,000
16	0,475	0,157	0,987

[15] spezifisch in der Version vom 18.07.2022

Tab. 9.1: Ergebnisse des Zweistichproben-t-Tests der Systemlaufzeiten bei Latenzen von 0 und 60 ms

9.2.4 CPU-Zeit / RAM Auslastung

Die im Experiment gemessenen Werte der Prozessorauslastung sind abhängig von der Leistungsfähigkeit des Gastgebersystems. Um unterschiedliche Prozessoren miteinander vergleichen zu können, werden normierte Tests mit diesen durchgeführt, bekannt als Benchmarks. Zur Normalisierung der gemessenen Anteile der CPU-Zeit können diese als Anteile der von diesem Prozessor erreichten Punktzahl in einem solchen Benchmark ausgedrückt werden (vgl. Anhang B).

Eine Bestrebung der Digitalisierung der Eisenbahninfrastruktur ist es, die Kosten von Infrastrukturprojekten zu senken. Dies soll unter anderem durch die Nutzung von kostengünstiger COTS-Hardware umgesetzt werden. Besonders Einplatinenrechner, welche nur die Größe einer Kreditkarte annehmen, haben sich im Bezug auf ihr Preis-Leistungsverhältnis und Größe-Leistungsverhältnis einen Namen gemacht, allen voran der „Raspberry Pi“. Jede Version dieses Computers erscheint in verschiedenen Ausführungen und die günstigste Variante der neuesten Version trägt den Namen „Zero 2 W“ mit folgenden relevanten Hardwarespezifikationen:

- Prozessor: 1GHz quad-core 64-bit Arm Cortex-A53 CPU
- Arbeitsspeicher: 512MB SDRAM

Um die Eignung dieses Einplatinenrechners für den Einsatz als OC abzuschätzen, wurden Ergebnisse des Benchmarks CoreMark verglichen, welches sowohl mit dem Prozessor des Gastgebersystems im Experiment als auch mit dem „Raspberry Pi Zero 2 W“ durchgeführt wurde. Der AMD EPYC 7313 des Testsystems erreicht darin 645919 Punkte, der Raspberry Pi 13917 Punkte, was 2,2% des EPYC 7313 ausmacht [CORE20].

Die durchschnittliche CPU-Auslastung während des Experiments, d.h. während des Annehmens und Versendens der für die Umstellung der Weiche nötigen Pakete etwa zwei Mal in der Sekunde, wurde mit 1,15% gemessen, der höchste erreichte Wert betrug 1,22%. Die während der Messung erfasste Belegung des Arbeitsspeichers folgte nicht den Ereignissen des Experimentes, d.h. der Start des Experimentes geht nicht mit einem Anstieg der Belegung einher. Dies kann an von den Programmen zur Darstellung des OC unabhängigen Vorgängen auf der VM liegen. Unbenommen davon bleibt die Tatsache, dass sich alle für die Simulation nötigen Vorgänge, inklusive Docker Engine und Betriebssystem mit der maximal gemessenen Belegung von 92,53 MB ausführen lassen (vgl. Tab. 8.6).

Anhand dieser Erkenntnisse kann ausgeschlossen werden, dass die Ergebnisse der Systemlaufzeiten von der Leistungsfähigkeit des Gastgebersystems eingeschränkt wurden. Es lässt sich ebenfalls annehmen, dass die verwendeten Software-

Komponenten in dieser Art und Weise problemlos auf einem Einplatinenrechner wie dem Raspberry Pi Zero 2 W ausführbar sind. Dies entspricht vollständig der Erwartung, da zwei der schnellsten Programmiersprachen für die Simulation der Software-Komponenten OC verwandt wurden und keine rechenintensiven Prozesse, sondern hauptsächlich die Abhandlung von Netzwerkkommunikation und das simple Erstellen von Byte-Arrays der Telegramme nötig sind.

Obwohl die Anforderungen an die Leistung für den Einsatz, wie in der Arbeit ermittelt, ausreichen würden, entspricht COTS-Hardware wie der Raspberry Pi Zero 2 W ohne Änderungen nicht den strengen RAMS-Anforderungen, welche an Bestandteile von sicherheitskritischen Infrastruktursystemen gestellt werden. Die Nutzung von Einplatinencomputern in der Entwicklung sicherheitskritischer Systeme ist aber möglich, wie vom Unternehmen „Revolution Pi“ demonstriert [REV22]. Als KUNBUS GmbH härtet dieses Unternehmen den Raspberry Pi für das nötige Sicherheitsintegritätslevel. Die Realisierung der in dieser Arbeit mittels DSTW-Sim aufgezeigten Potentiale des Einsatzes der kostengünstigsten Hardware durch EIU ist daher sicherlich nur eine Frage der Zeit.

9.2.5 Auswirkungen der Laufzeiten auf Fahrstraßenbildezeit

Die innerhalb von DSTW-Sim gemessenen Fahrstraßenbildezeiten lassen sich nun mit denen vergleichen, welche von Herstellern im Rahmen der Planung von ETCS im Kernnetz der S-Bahn Stuttgart angeboten wurden. Unter Vernachlässigung der Weichenumlaufzeit betrug die vollständige Fahrstraßenbildezeit laut Definition aus Kapitel 3.3 unter Projektierung der im bb-IP-Netz gemessenen Konditionen (vgl. Tab. 4.1 und Tab. 4.2) in DSTW-Sim 38,4 ms. Die Vorgabe für eine Fahrstraßenbildezeit ohne Weichenumlauf beträgt laut NeuPro 3 Sekunden, die im Rahmen der S-Bahn-ETCS-Untersuchung bei Herstellern erfragten Zeiten betragen zwischen 7 und 8 Sekunden [ING19].
Die Diskrepanz zwischen den in der Simulation gemessenen und den tatsächlich angebotenen Zeiten lässt sich aufgrund ihrer Größe nicht mit den Vereinfachungen der Simulation erklären. Stattdessen wurden folgende Gründe gefunden: Die Zeiten der Hersteller beziehen sich auf Garantien, die in bestimmten Konfidenzintervallen gegeben werden können. Weiterhin wurde in der Stellwerksplanung historisch keine besondere Aufmerksamkeit auf die Erzielung der kürzesten Laufzeiten innerhalb des Stellwerks gelegt. Laut Lastenheften müssen lediglich die im BTSF großzügig festgelegten Werte erreicht werden, was nicht zur Minimierung der Laufzeiten beigetragen hat. Nicht zuletzt wurde im Rahmen der S-Bahn-Untersuchung herausgefunden, dass diese Werte in der Realität vielfach nicht eingehalten werden [ING19].

Die mit DSTW-Sim durchgeführten Experimente halfen, um die Größenordnungen der Potentiale zu Laufzeitenverkürzung aufzuzeigen. Es liegt nun an den Eisenbahninfrastrukturunternehmen, die Hersteller zur Nutzung dieser anzuhalten und für Wettbewerb zu sorgen. Die Autoren der S-Bahn-ETCS-Untersuchung vermerken, dass jede einzelne Sekunde wertvoll sei, da sie direkt zum Verspätungsausgleich genutzt werden könne [ING19].

9.2.6 Paketverlustrate

Der Umgang mit der Beeinträchtigung des Netzwerks durch den Verlust von Paketen ist in der paketorientierten Kommunikation sehr ernst zu nehmen. Bei der Übertragung einer Datei kann der Verlust eines einzigen Paketes zur vollständigen Korruption dieser führen. Gründe für den Paketverlust können in Engstellen liegen, in denen Paket-weiterleitende Netzwerkgeräte vollständig ausgelastet sind und Pakete verwerfen müssen, Ausfall von Netzwerkgeräten, Fehler in Routing-Tabellen, aufgrund derer Pakete zu oft umgeleitet und schließlich verworfen werden[16] oder externe Einflüsse physischer Natur.
Wie bereits im Kapitel 7.2.8 erwähnt, verfügt die RaSTA über UDP Lösung über keine Mechanismen zu Erkennung von Paketverlusten auf der Transportschicht. In diesem Experiment hätten sich also die bereits eingeführten Sendewiederholungseigenschaften von OS RaSTA zeigen sollen.

Erwartet wurde folgendes Verhalten: Nach Verlust eines Pakets auf allen Redundanzebenen wird von der Sicherheits- und Sendewiederholungsschicht ein Fehler in den fortlaufenden Sequenznummern erkannt und ein Sendewiederholungsszenario angestoßen. Die zur Behebung des Paketverlusts nötige Zeit lässt sich unter der Annahme, dass die Übertragung eines Paketes durchschnittlich 6,5 ms in Anspruch nimmt (vgl. Tab. 8.1), durch die Summierung der Laufzeiten der dazu nötigen Schritte abschätzen:

- Erkennung Paketverlust spätestens mit nächster Heartbeat-Nachricht: 300 ms
- Laufzeit der Heartbeat-Nachricht: 6,5 ms
- Wartezeit von außerhalb der Sequenz empfangenen Daten: 100 ms
- Laufzeit der Sendewiederholungsanfrage: 6,5 ms
- Laufzeit der Sendewiederholungsbestätigung: 6,5 ms
- Übertragung der verlorengegangenen Nachricht: 6,5 ms
- Abschluss des Sendewiederholungsszenarios mit Heartbeat-Nachricht: 6,5 ms

Ein einzelner Paketverlust sollte sich also mit etwa 432,5 ms auf die Laufzeit der RaSTA-Übertragung auswirken. Dieses Verhalten konnte jedoch bei den Testversuchen nicht beobachtet werden. Bei niedrigen Paketverlustraten war gar keine Änderung der Laufzeiten verzeichnen. Dies liegt am probabilistischen Auftreten des Paketverlustes und der vollständigen Redundanz des Netzwerkes in Form von blauer und grauer Netzwerkebene. Ein Paketverlust von 0,1% führt dazu, dass jedes 1000. Paket nicht versendet wird. Das Zusammentreffen eines verworfenen Paketes auf beiden Netzwerkebenen ist daher sehr unwahrscheinlich und so führte die Redundanz zur Fehlerkompensation der niedrigen Paketverlustraten.

Nach dem ersten, vollständigen Verlust eines Paketes und dem Eintreffen des

[16] Der IP-Header besitzt ein Feld, um die Anzahl der maximalen Weiterleitungen eines Pakets zu zählen. Wenn diese Zahl auf 0 fällt, wird das Paket verworfen

darauffolgenden Paketes außerhalb der Sequenz kam es nicht nur zum Abbruch der RaSTA-Verbindung, sondern auch zum Absturz des RaSTA-Moduls. Der Absturz konnte auf einen unter der Meldung „Segmentation fault" bekannten Fehler zurückgeführt werden. Dieser trat nach jedem erlittenen Paketverlust auf, so dass die Funktionalität der Sendewiederholung von OS RaSTA nicht bestätigt werden oder für Experimente benutzt werden konnte. Die erfolgreich absolvierten Versuche sind somit allein auf die Redundanz des Netzwerkes zurückzuführen.

Für die Validierung von DSTW-Sim ist allerdings auch dieses Ergebnis als Erfolg zu werten, da demonstriert wurde, dass erfolgreich Konditionen simuliert werden können, unter denen die Eigenschaften zum Fehlerausgleich von Netzwerkprotokollen gefordert werden, wie es auch im bb-IP-Netz der Fall ist.

9.2.7 Bandbreite

Für die Bildung einer Fahrstraße, mit welcher die Umstellung einer Weiche und die Änderung des Signalbegriffs eines Lichtsignals verbunden ist, ergibt sich laut dem in den EULYNX Spezifikationen definierten PDI Protokoll die Notwendigkeit für den Austausch von 5 PID-Einheiten. Dazu gehören folgende Nachrichten:

1. MOVE_POINT: X – 44 Bytes
2. POINT_POSITION: Keine Endlage – 45 Bytes
3. POINT_POSITION: X – 45 Bytes
4. INDICATE_SIGNAL_ASPECT: A – 61 Bytes
5. INDICATED_SIGNAL_ASPECT: A – 61 Bytes

Dies macht zusammen 256 Bytes aus. Dazu kommen die zum Prüfen der Verbindungsqualität gehörenden RaSTA-Heartbeat-Nachrichten mit einer Größe von 36 Bytes, welche von jedem Teilnehmer einer RaSTA-Verbindung alle 300 ms versandt werden müssen, falls in dieser Zeitspanne keine Daten-Nachricht versandt worden ist. Der für den Versand dieser Anwendungsdaten nötige Overhead der niedriger gelegenen Schichten von 50 Bytes setzt sich wie folgt zusammen:

- UDP-Header, Größe: 8 Bytes [POS80]
- IP-Header, Größe: 20 bis 24 Bytes [TEC22]
- Ethernet-Header, Größe: 18 Bytes [IEE18]

In Form der IP-Hilfsprotokolle werden auf dem bb-IP-Netz und auch in der Simulation weitere Overhead-Daten ausgetauscht. Zu den Hilfsprotokollen gehören das Internet Control Message Protocol (ICMP), über welches beispielsweise Fehler bei der Verarbeitung oder Zustellung von Paketen gemeldet werden [INT81], oder das Address Resolution Protocol (ARP), durch welches die zugehörige Layer-2 Adresse (typischerweise MAC-Adresse) eines IP-Gerätes angefragt werden kann, zu dem nur die IP-Adresse bekannt ist [INT82].

```
33 3.891635467 HewlettP_34:f1:16 → HewlettP_1f:9e:9d ARP 60 Who has 10.152.0.4? Tell 10.152.0.2
34 3.893567552 HewlettP_1f:9e:9d → HewlettP_34:f1:16 ARP 60 10.152.0.4 is at 00:24:81:1f:9e:9d
```

Abb. 9.3: Beispiel für die Anfrage einer Layer-2 Adresse über ARP, Ausgabe von Wireshark

Die Messung der genutzten Bandbreite pro Netzwerkadapter ergab für die Kommunikation während des Experimentes mit der SCI-P Schnittstelle 1,25 KB/s (9,6 Kbit/s), für SCI-P waren es 0,93 KB/s (7,47 Kbit/s). Der Unterschied zwischen beiden kommt u.a. durch das zusätzliche Paket über die Information des Verlassens der Weichenendlage des OC der Weiche an das Stellwerk zustande.

Wie in Kapitel 4.1 erwähnt werden beide Netzwerkebenen des bb-IP-Netzes mit einer Vollduplex-Datenrate von 1000 Mbit ausgelegt. Die in DSTW-Sim verwendeten internen Netze bilden dies qualitativ nach und bieten dieselbe Leistung. Ein Einfluss der Bandbreite auf die übrigen Messergebnisse kann also ausgeschlossen werden. Eine Abschätzung der Leistungsfähigkeit des bb-IP-Netzes bezüglich der Bandbreite lässt sich auf Basis der im Test beobachteten Datenrate ebenfalls treffen. Innerhalb eines Zugangsnetzes wäre es unter diesen Umständen von der Datenrate her möglich, 104.166 OC mit einem DSTW zu steuern.

Dabei zu beachten sind die Vereinfachungen, welche in DSTW-Sim bezüglich der Cyber-Security gemacht wurden. Die von der DB Netz AG vorgesehenen Komponenten Crypto-Box und VPN-Modul wurden nicht simuliert. Der Einfluss dieser auf die Datenrate kann durch eine Betrachtung der sich zusätzlich ergebenden Daten abgeschätzt werden, wenn der Einsatz der Enterprise-VPN-Lösung IPSec und der in EULYNX definierten kryptographisch sicheren Alternative zu UDP, TLS over TCP, angenommen wird.

Durch den IPSec-Header können im ungünstigsten Fall, wie bei dem Senden von eines einzelnen Bytes, 123 Byte hinzugefügt werden [PAP22]. Der TLS Header kann je nach Art des verwendeten Kryptographischen Profils 21 bis 40 Bytes betragen [MATOJ]. Der Overhead von TCP selbst, welches anstelle von UDP eingesetzt werden würde, lässt sich schwieriger beziffern, da zur Kommunikation über TCP neben einem Header von 20 Bytes für die Nutzdatenpakete auch zusätzliche Nachrichten zum Verbindungsaufbau und je nach Größe der Pakete in unterschiedlichen Intervallen Nachrichten zur Bestätigung der bisher empfangenen Nutzdaten versendet werden [TCP92]. Dieses sogenannte Acknowledgement hat die Mindestgröße für ein über Ethernet zu versendendes Paket, 64 Bytes.

Unter der Annahme, dass jedes einzelne Paket ein Acknowledgement hervorruft und sich dieses negativ auf die Bandbreite auswirkt, ergibt sich ein Security-Overhead von 256 Bytes:

- IPSec-Header, Größe: 123 Bytes
- TLS-Header, Größe: 21 bis 40 Bytes
- TCP-Header, Größe: 16 bis 20 Bytes
- Acknowledgement, Größe: 64 Bytes

Zu diesem müssen IP- und Ethernet-Header addiert werden, für eine Overhead-Gesamtgröße von 298 Bytes, was einer Steigerung um 596% gegenüber dem Szenario in DSTW-Sim entspricht. Unter diesen Umständen wären nur noch 17.477 OC möglich.

Es ist jedoch fraglich, ob die Nutzung von sowohl IPSec als auch TLS over TCP nötig ist bzw. für zusätzliche Sicherheit sorgt. So wird die Sicherung mit TLS over TCP in der EULYNX Spezifikation bereits als ausreichend bewertet [Eu.Doc.16].

In den zwei angenommenen Szenarios ist entweder die Steuerung von 104.166 Stelleinheiten im optimistischen Szenario, oder von 17.477 Stelleinheiten im pessimistischen Szenario durch ein DSTW möglich. Üblich sind im Netz der Deutschen Bahn 14 Stelleinheiten für mechanische Stellwerke, 31 Stelleinheiten für elektromechanische Stellwerke, 77 an Relaisstellwerken und 236 an ESTW [JÖR07].
Die im Jahr 2003 etwa 250.000 installierten Stelleinheiten in ganz Deutschland ließen sich je nach Szenario von 3 bis 15 TSO Standorten aus über dieselbe Anzahl an Zugangsnetzen steuern.

10 Fazit

Als Produkt der Zusammenarbeit der größten Infrastrukturbetreiber in Europa kommt EULYNX als Lösung für die Standardisierung der Schnittstellen zwischen den Bestandteilen der Leit- und Sicherungstechnik in der Zukunft eine große Bedeutung zu. Im Gegensatz zur Testinfrastruktur rund um das Leit- und Sicherungssystem ERTMS ist sie für EULYNX noch unzureichend. Das Entstehen einer Einrichtung wie das „Railway Simulation and Testing“ (RailSiTe) für EULYNX wäre durchaus wünschenswert. Solche Labore sowie interne Testeinrichtungen bei Herstellern müssen Umgebungen wie DSTW-Sim einsetzen, um ihre Komponenten zu verifizieren.

Im Rahmen der vorliegenden Arbeit wurde ein Konzept für eine für Komponenten- oder Integrationstests benötigte Evaluationsumgebung entwickelt. In Form von DSTW-Sim konnte dieses Konzept auf Basis von COTS Open Source Software prototypisch umgesetzt werden. Durch den Test der Open Source RaSTA-Implementierung von Heinrich konnte die Funktionalität der Evaluationsumgebung demonstriert und die im Vorfeld definierten Anforderungen validiert werden.

Es konnten außerdem Aussagen über die Potentiale der DSTW-Technik im Bezug auf Geschwindigkeit gemacht werden. Der eruierte Stand der Wissenschaft und Technik stimmt optimistisch für eine deutlich kostengünstigere und schnellere Entwicklung der Eisenbahninfrastruktur. Die Vereinfachungen, die durch die Einführung von EULYNX im Zulassungsprozess möglich werden, lassen erwarten, dass die Planung von Eisenbahninfrastruktur zukünftig schneller abläuft und die im Einsatz befindliche Technologie dem Stand der Technik in kürzerem Abstand folgen kann. Mit der DSTW-Technologie und Standardisierung durch EULYNX kann ein Beitrag für das Erreichen der deutschen Klimaziele geleistet werden.

10.1 Ausblick

Bei der Simulation des Stellwerks und insbesondere bei der Fahrstraßenbildung wurden einige Annahmen getroffen, welche die Nutzung von DSTW-Sim nur für Tests mit einer den Zielen dieser Arbeit ähnlichen Zielstellung qualifizieren. Dazu gehört, dass immer nur eine Fahrstraße gleichzeitig eingestellt werden kann und keine EULYNX-konforme Initialisierung der PDI-Verbindung mit den OC durchgeführt wird.

Für Tests von allgemeineren Fällen ist daher eine Erweiterung der Stellwerkssimulation nötig, welche sich an der bisher gewählten Architektur orientieren kann. Die Erweiterung um eine bspw. Browser-basierte graphische Oberfläche mit einem Python-Backend wie „FastAPI“ wäre auf Basis der vorhandenen Schnittstellen mit geringem Aufwand möglich.

Abkürzungsverzeichnis

ARP	Address Resolution Protocol
ATO	Automatic Train Protection (Automatischer Zugfahrbetrieb)
Az	Achszähler
bb-IP-Netz	bahnbetriebliches IP-Netz
BTSF	Betrieblich-technische Systemfunktionen
CENELEC	Comite europeen de normalisation electrotechnique (europäische Komitee für elektrotechnische Normung)
COTS	Commercial off-the-shelf
CPU	Central Prozessing Unit (Prozessor)
DB AG	Deutsche Bahn Aktiengesellschaft
DIN	Deutsches Institut für Normung
DLR	Deutschen Zentrum für Luft- und Raumfahrt
DSTW	digitales Stellwerk
DSTW	DSTW-Simulation
EBA	Eisenbahn-Bundesamt
EIGV	Eisenbahn-Inbetriebnahmegenehmigungsverordnung
EN	Europäische Normen
ERTMS	European Rail Traffic Management System
ESTW	elektronisches Stellwerk
ETCS	European Train Control System
EULYNX	Organisation europäischer Eisenbahninfrastrukturbetreiber
EVC	European Vital Computer (ETCS-Bordrechner)
FEAK	Feldelement-Anschlusskasten
FEAS	Feldelement-Anschlussschrank
FIFO	First-in-first-out (Elemente, die am längsten warten, werden als erstes bearbeitet)
GFK	Gleisfeldkonzentrator
GSM	Global System for Mobiles
HDMI	High Definition Multimedia Interface
ICMP	Internet Control Message Protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
iLBS	Integriertes Leit- und Bediensystem
INESS	Integrated European Signalling System
IoT	Internet-of-Things (Internet der Dinge)
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IT	Information Technology (Informationstechnik)
IXL	Stellwerk
KI	Künstliche Intelligenz
LAN	Local Area Network
LS	Light Signal
LZB	Linienförmiges Zugbeeinflussungssystem

MA	Movement Authority (Fahrterlaubnis)
MAC	Media Access Control (Medienzugriffssteuerung)
MBSE SF	Model-based Systems Engineering Specification Framework
MD4	Message Digest Algorithm 4
MDM	Maintenance & Data Management
MPLS	Multiprotocol Label Switching
Msg	Message
NIC	Network Interface Controller (Netzwerkadapter, Netzwerkkarte)
NTP	Network Time Protocol
OBU	On-Bord Unit (Fahrzeuggerät)
OC	Object Controller
OSI	Open System Interconnection (Offenes System für Kommunikationsverbindungen)
OS RaSTA	Open Source RaSTA
PDI	Process Data Interface
PDU	Protocol Data Unit (Protokolldateneinheit)
PTP	Precision Time Protocol
PZB	Punktförmige Zugbeeinflussung
RailSiTe	Railway Simulation and Testing
RAM	Random Access Memory (Arbeitsspeicher)
RAMS	Reliability, Availability, Maintainability, Safety
RaSTA	Rail Safe Application Protocol
RBC	Radio Block Centre
RFC	Request for Comments (begutachtete technische Spezifikation)
SCI	Standard Communication Interface
SCP	Safe Communication Protocol
SDI	Standard Diagnostic Interface (Schnittstelle für Diagnosedaten)
SMI	Standard Maintenance Interface (Schnittstelle für Instandhaltungsdaten)
SysML	Systems Modeling Language
TCP	Transmission Control Protocol (Übertragungssteuerungsprotokoll)
TLS	Transport Layer Security
TMS	Traffic Management System
TSI	Technical Specifications for Interoperability
TSO	Technikstandort
UDP	User Datagram Protocol (verbindungsloses paketorientiertes Netzwerkprotokoll)
UIC	Union internationale des chemins de fer (Internationaler Eisenbahnverband)
USB	Universal Serial Bus
VDE	Verband deutscher Elektrotechniker
VM	Virtual Machine (virtuelle Maschine)
VPN	Virtual Private Network (virtuelles privates Kommunikationsnetz)
V&V	Verifikation und Validierung
WAN	Wide Area Network

Abbildungsverzeichnis

Abb. 3.1: Entwicklung der Verantwortlichkeiten mit zunehmender Optimierung des Zugbeeinflussungssystems ETCS 13
Abb. 3.2: Darstellung der Belegungs- / Sperrzeitkomponenten der LST mit ETCS Level 2. Der Anteil der LST-Laufzeiten ist rot beschriftet, aus [BEH21] 14
Abb. 4.1: Technische Architektur des DSTW, eigene Abb. nach [DSD21] und [WGC21] 20
Abb. 4.2: schematische Netzwerkarchitektur Feldelement-Anschlussschrank, Darstellung nach Weber [EBA] 21
Abb. 5.1 IPv4 Header, Darstellung von [WIK22] 31
Abb. 6.1: Architektur von DSTW-Sim 34
Abb. 6.2: Ausschnitt der Konfigurationsdatei für DSTW-Sim 35
Abb. 6.3: Abbildung: Gleisbild der in DSTW projektieren Gleisfeld-Komponenten 36
Abb. 6.4: Komponentenplan der projektierten Gleisfeld-Komponenten 36
Abb. 7.1: Allgemeine Aufrüstung von EULINX-Subsystemen nach [Eu.Doc.119] 43
Abb. 7.2: EULYNX Umstellung einer Weiche - Main Success Scenario: Moving of the Point with a single point machine, Darstellung in Anlehnung an [Eu.Doc.36] 45
Abb. 7.3: EULYNX Änderung eines Signalbegriffs - Main Success Scenario: Indicate signal aspect, Darstellung in Anlehnung an [Eu.Doc.32] 47
Abb. 7.4: Schematische Einordnung von Rasta in das OSI-Modell, Darstellung in Anlehnung an [Eu.Doc.92] 48
Abb. 7.5: Adaptive Kanalüberwachung, modifiziert nach DIN VDE V 0831-200 [RAS15] 51
Abb. 7.6: Sendewiederholungsszenario, modifiziert nach DIN VDE V 0831-200 [RAS15] 52
Abb. 8.1: Experimentdurchführung in DSTW-Sim 60
Abb. 8.2: Schritte der Fahrstraßenbildung mittels der Stellwerksimulation in DSTW-Sim 61
Abb. 8.3: Messergebnisse der RaSTA-Laufzeiten bei deaktivierter Uhrensynchronisation 62
Abb. 8.4: Messergebnisse der RaSTA-Laufzeiten bei aktivierter Uhrensynchronisation 62
Abb. 8.5: Messergebnisse der RaSTA-Laufzeiten bei einer Netzwerklatenz von 0 ms 64
Abb. 8.6: Messergebnisse der RaSTA-Laufzeiten bei einer Netzwerklatenz von 30 ms 64
Abb. 8.7: Messergebnisse der RaSTA-Laufzeiten bei einer Netzwerklatenz von 60 ms 65
Abb. 8.8: Durchschnittliche RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz 65
Abb. 8.9: Messergebnisse von Systemlaufzeiten bei einer Netzwerklatenz von 0 ms 67

Abb. 8.10: Messergebnisse von systeminternen Laufzeiten bei einer Netzwerklatenz von 60 ms 67
Abb. 8.11: CPU-Auslastung / Arbeitsspeicherverbrauch in „Signal1“ während Simulation der Fahrstraßenbildung 69
Abb. 8.12: Datenrate und kumulierte Datenmenge für den blauen Netzwerkadapter der VM zur Stellwerkssimulation über die Dauer eines Experimentes 70
Abb. 9.1: Graphische Beschreibung des OS RaSTA Problems „Doppeluhrenprinzip“ 72
Abb. 9.2: Abbildung der Log-Einträge eines OC zum Zeitpunkt des Verbindungsabbruchs 75
Abb. 9.3: Beispiel für die Anfrage einer Layer-2 Adresse über ARP, Ausgabe von Wireshark 80

Tabellenverzeichnis

Tab. 3.1: Zusammensetzung der Ende-zu-Ende Laufzeiten einer Fahrstraßenbildung, nach [ING19] 15
Tab. 4.1: Erweitertes RFC 2544 – Testergebnisse Latenz, Mail von Zentralsysteme DB Netz AG vom 19.09.2022 (vgl. Anhang C) 21
Tab. 4.2: Erweitertes RFC 2544 – Testergebnisse Paketfehlerrate, Mail von Zentralsysteme DB Netz AG vom 19.09.2022 (vgl. Anhang C) 22
Tab. 7.1: PDI Telegram Struktur, aus [Eu.Doc.93] 42
Tab. 7.2: Aufbau der Protokolldateneinheit der Sicherheits- und Sendewiederholungsschicht von RaSTA nach DIN VDE V 0831-200 [RAS15] 49
Tab. 8.1: Durchschnittliche RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz 66
Tab. 8.2: Summierte RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz 66
Tab. 8.3: Durchschnittliche RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz, Beeinträchtigung einer einzelnen Netzwerkebene 66
Tab. 8.4: Summierte Systemlaufzeiten in Abhängigkeit der Netzwerkbedingungen – Latenz 68
Tab. 8.5: Summierte RaSTA-Laufzeiten in Abhängigkeit der Netzwerkbedingungen – Paketverlustrate 68
Tab. 8.6: CPU-Auslastung / Arbeitsspeicherverbrauch in „Signal1“ während Simulation der Fahrstraßenbildung 69
Tab. 9.1: Ergebnisse des Zweistichproben-t-Tests der Systemlaufzeiten bei Latenzen von 0 und 60 ms 76

Literaturverzeichnis

[1394NEWS] 1394 Monthly Newsletter 4 (1998), Nr. 2, https://books.google.at/books?id=fRvbxgH4wmsC&pg=PA7.

[AHM18] Ahmed, Mudassar: Implementation and Performance Analysis of Precision Time Protocol on Linux based System-On-Chip Platform. Fachhochschule Kiel, https://www.creative-technologies.de/wp-content/uploads/2018/05/Master_Project__PTP_1588.pdf (aufgerufen am 29.11.2022), 2018.

[AMP17] Ampomah, Ernest; Mensah, Ezekiel; Gilbert, Abilimi: Qualitative Assessment of Compiled, Interpreted and Hybrid Programming Languages. In: *Communications on Applied Electronics* 7 (2017), S. 8–13.

[AZE18] Azevedo de Amorim, Arthur; Hriţcu, Cătălin; Pierce, Benjamin: The Meaning of Memory Safety. In: Lujo Bauer und Ralf Küsters (Hrsg.): Principles of security and trust. 7th international conference, POST 2018, held as part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018: proceedings. Berlin: Springer (Lecture notes in computer science Advanced research in computing and software science, 10804), 2018, - ISBN978-3-319-89721-9, S. 79–105.

[BAL12] Aaron Balchunas: OSI Reference Model, 2012, https://www.routeralley.com/guides/osi.pdf (aufgerufen am 002.11.2022).

[BAU] Bauer, Lujo; Küsters, Ralf (Hrsg.): Principles of security and trust. 7th international conference, POST 2018, held as part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018: Proceedings. ETAPS, Berlin: Springer (Lecture notes in computer science Advanced research in computing and software science, 10804), 2018.

[BEH19] Behnke, Ilja; Thamsen, Lauritz; Kao, Odej: Héctor: A Framework for Testing IoT Applications Across Heterogeneous Edge and Cloud Testbeds. In: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion - UCC '19 Companion. Auckland, New Zealand: ACM Press (2019), S. 15–20, http://dl.acm.org/citation.cfm?doid=3368235.3368832 (aufgerufen am 11.11.2022).

[BEH21] Behrens, Marc; Caspar, Mirko; Distler, Andreas; Fries, Nikolaus; Hardel, Sascha; Kreßner, Jan et al.: Schnelle Leit- und Sicherungstechnik für mehr Fahrwegkapazität. In: *EI - Der Eisenbahningenieur* (2021), S. 50–55, https://elib.dlr.de/147827/.

[BEI221] Beilharz, Jossekin; Wiesner, Philipp; Boockmeyer, Arne; Pirl, Lukas; Friedenberger, Dirk; Brokhausen, Florian et al. (2021): Continuously

Testing Distributed IoT Systems: An Overview of the State of the Art, http://arxiv.org/pdf/2112.09580v1.

[BHO22] Bundesfinanzministerium: Bundeshaushaltsordnung. BHO, vom 01.07.2022.

[BIE22] Biehl, Anika: Einbindung moderner Überwachungsarten für Bahnübergänge auf ETCS-Strecken. TU Dresden, https://tud.qucosa.de/api/qucosa%3A80730/attachment/ATT-0/ (aufgerufen am 29.11.2022), 2022.

[BME] bmerry: Feature request: support for nanosecond timestamps · Issue #10822 · elastic/logstash, 2019, https://github.com/elastic/logstash/issues/10822 (aufgerufen am 14.11.2022).

[BRA91] Bradner, Scott O.: Benchmarking Terminology for Network Interconnection Devices. Request for Comments. Internet Engineering Task Force (RFC 1242), 1991, https://datatracker.ietf.org/doc/rfc1242 (aufgerufen am 09.11.2022).

[BUG22] Bugden, William; Alahmar, Ayman (2022): Rust: The Programming Language for Safety and Performance: arXiv (aufgerufen am 21.11.2022).

[BÜK20] Büker, Thorsten; Schotten, Simon; Hennig, Eike: Kapazitätsberechnung im Moving Block – die Tücke im Detail. In: *Eisenbahntechnische Rundschau* (2020), Nr. 7+8, S. 32–37.

[CAS18] Mirko Caspar; Hardi Hungar; Daniel Schwencke: Effizientes Testen modularisierter und standardisierter Stellwerkskomponenten. In: *Signal+Draht* 110, Nr. 9/2018.

[CORE20] Coremark Benchmark - OpenBenchmarking.org (2020), https://openbenchmarking.org/test/pts/coremark (aufgerufen am 01.11.2022).

[DB22] Deutsche Bahn AG (Hrsg.): Die digitale Revolution im Schienennetz Deutsche Bahn AG, https://fahrweg.dbnetze.com/fahrweg-de/kunden/nutzungsbedingungen/digitale_lst/allgemein-3084902 (aufgerufen am 09.10.2022).

[DBNWEB] DB Netz AG (Hrsg.): Rolloutstrategie, https://fahrweg.dbnetze.com/fahrweg-de/kunden/nutzungsbedingungen/digitale_lst/rolloutstrategie-3084914 (aufgerufen am 03.11.2022).

[DIN26] DIN EN 50126: Bahnanwendungen - Spezifikation und Nachweis von Zuverlässigkeit, Verfügbarkeit, Instandhaltbarkeit und Sicherheit (RAMS) - Teil 1: Generischer RAMS-Prozess. 2018-10.

[DIN28] DIN EN 50128: Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Software für Eisenbahnsteuerungs- und Überwachungssysteme. 03.2012.

[DOC22] Docker: Accelerated, Containerized Application Development (2022), https://www.docker.com/, zuletzt aktualisiert am 10.05.2022 (aufgerufen am 30.11.2022).

[DOP18] Doppelbauer, Josef: Neue Züge „auf Schiene“ bringen. In: *Deine Bahn* SPEZIAL (2018), Nr. 9, S. 7.

[DSD19] Deutsche Bahn AG (2019): Broschüre: Digitale Schiene Deutschland - Die Zukunft der Eisenbahn: Deutsche Bahn AG: Vorstand Infrastruktur.

[DSD21] Elsweiler, Bernd; Fries, Nikolaus; Summen, Christian: EBA-Infotage: Einordnung und Übersicht über das Programm Digitale Schiene Deutschland (DSD). EBA. Digitale Schiene Deutschland. 2021.

[EBU08] Stellwerke + Fahrstraßenbildung. Schulung Grundlagen des Eisenbahnbetriebs. Eisenbahn-Betriebs- und Experimentierfeld Berlin. 10.10.2008.

[EDE16] Eder, Michael: Hypervisor- vs. Container-based Virtualization (2016), http://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_01.pdf (aufgerufen am 09.11.2022).

[EIGV20] Bundesministerium für Verkehr und digitale Infrastruktur (17.06.2020): Eisenbahn-Inbetriebnahmegenehmigungsverordnung. EIGV.

[ETR85] Erstes elektronisches Stellwerk der DB – ein Beitrag zur Zukunftssicherung der Deutschen Bundesbahn. In: *Eisenbahntechnische Rundschau* (1985), Nr. 12.

[Eu.Doc.100] EULYNX Dokument: Specification of Point of Service-Signalling.

[Eu.Doc.11] EULYNX Dokument: Documentation plan.

[Eu.Doc.119] EULYNX Dokument: Generic interface and subsystem requirements for SCI.

[Eu.Doc.16] EULYNX Dokument: EULYNX System architecture specification.

[Eu.Doc.29] EULYNX Dokument: Interpretation rules for model-based requirements.

[Eu.Doc.30] EULYNX Dokument: Modelling Standard.

[Eu.Doc.32] EULYNX Dokument: Requirements specification for subsystem Light Signal.

[Eu.Doc.33] EULYNX Dokument: Interface specification SCI-LS.

[EU.Doc.33] EULYNX Dokument: Interface specification SCI-LS.

[Eu.Doc.36] EULYNX Dokument: Requirement specification for subsystem Point.

[Eu.Doc.38] EULYNX Dokument: Interface specification SCI-P.

[Eu.Doc.92] EULYNX Dokument: Interface definition SCI.

[Eu.Doc.93] EULYNX Dokument: Interface specification SCI Generic.

[EUL17] Report on Industry Workshop held 25th January 2017. EULYNX. 25.01.2017, https://eulynx.eu/index.php/documents/presentations-given/39-20170125-workshop-industry-report/file.

[FRI21] Fries, Nikolaus: Betrieblich-Technisches Zielbild für die „Digitale Schiene“ 3 (2021), S. 5.

[GIT22] Fix CTS - SgtChrome/rasta-protocol@24ce71d, https://github.com/SgtChrome/rasta-protocol/commit/24ce71d5a24a32bb906a04b7a8101b04b5fa236c (aufgerufen am 24.11.2022).

[GIT22b] increase logging buffer · SgtChrome/rasta-protocol@1be9916, https://github.com/SgtChrome/rasta-protocol/commit/1be9916125208bab93d47b30ba8cb0bf931e2cfc (aufgerufen am 24.11.2022).

[GOO19] Google (Hrsg.): Queue the Hardening Enhancements, 2019, https://security.googleblog.com/2019/05/queue-hardening-enhancements.html (aufgerufen am 21.11.2022).

[HEI13] Hein, Ferdinand: Sicherer Fahrweg - sichere Zugfahrt (Teil 1), Berlin: Eisenbahn-Fachverlag Heidelberg, 2013.

[HEI18] Heinrich, Markus; Vieten, Jannik; Arul, Tolga; Katzenbeisser, Stefan: Security Analysis of the RaSTA Safety Protocol. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI): IEEE (2018), S. 199–204.

[HEI20] Heinrich, Markus: Security Engineering in Safety-critical Railway Signalling. TU Darmstadt, Darmstadt, 2020.

[HUN16] Hungar, Hardi; Asbach, Lennart; Meyer zu Hörste, Michael (2016): Laborarchitekturen für den Test von Stellwerks-Schnittstellen. 01.2016.

[IEE18] IEEE Standard for Ethernet. In: *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)* (2018), S. 1–5600.

[IFTOP] iftop(8): bandwidth usage on interface by host - Linux man page, https://linux.die.net/man/8/iftop (aufgerufen am 25.11.2022).

[INE09] INESS Project (INtegrated European Signalling System) - UIC Communications (2009), https://uic.org/com/enews/nr/142/article/iness-project-integrated-european?page=thickbox_enews (aufgerufen am 19.10.2022).

[ING19] Ingenieurgemeinschaft „Machbarkeitsstudie ETCS S-Bahn Stuttgart“ (2019): Untersuchung "ETCS im Kernnetz der S-Bahn Stuttgart, https://vm.baden-wuerttemberg.de/fileadmin/redaktion/m-

mvi/intern/Dateien/PDF/Abschlussbericht_Untersuchung_ETCS_Stuttgart.pdf (aufgerufen am 30.08.2022).

[INT81] Internet Engineering Task Force: Internet Control Message Protocol. Request for Comments (RFC 792), 1981, https://datatracker.ietf.org/doc/rfc792 (aufgerufen am 25.11.2022).

[INT82] Internet Engineering Task Force: An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. Request for Comments (RFC 826), 1982, https://datatracker.ietf.org/doc/rfc826 (aufgerufen am 25.11.2022).

[INT99] Internet Engineering Task Force: Benchmarking Methodology for Network Interconnect Devices. Request for Comments (RFC 2544), 1999, https://datatracker.ietf.org/doc/rfc2544 (aufgerufen am 09.11.2022).

[JÖR07] Jörg Bormet: Anforderungen des Betreibers an den Life-cycle in der Fahrwegsicherungstechnik. In: *Signal+Draht* 99 (2007), Nr. 1+2, S. 22.

[KLI2] Bundesministerium für Umwelt, Naturschutz, nukleare: Klimaschutzgesetz Cluster BMUVBMUV, https://www.bmuv.de/faqs/klimaschutzgesetz (aufgerufen am 05.10.2022).

[KLI30] Klimaschutzprogramm 2030 der Bundesregierung zur Umsetzung des Klimaschutzplans 2050, S. 173.

[LAN19] Langui (Hrsg.): Memory Unsafety in Apple's Operating Systems, 2019, https://langui.sh/2019/07/23/apple-memory-safety/ (aufgerufen am 21.11.2022).

[LAU21] Laumen, Peter; Finken, Klaus (Hrsg.) (2021): Digitale Stellwerke - Stellwerksarchitekturen im Wandel der Zeit. Proceedings of the 3rd International Railway Symposium Aachen 2021.

[LIS20] Lisker, André; Kanis, Kersten: Inbetriebnahme des Digitalen Stellwerks Warnemünde. In: *SIGNALLING+DATACOMMUNICATION* (2020), Nr. 112, S. 11, https://assets.new.siemens.com/siemens/assets/api/uuid:9581fd71-61a2-4910-a523-5b7abbae3778/kanis-lisker.pdf.

[MAS15] Maschek, Ulrich: Sicherung des Schienenverkehrs. Grundlagen und Planung der Leit- und Sicherungstechnik (Lehrbuch). 3., überarb. u. erw. Aufl., Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2015, http://www.springer.com/.

[MAS96] Maschek: Analyse zur Gestaltung elektronischer Stellwerke, http://rcswww.urz.tu-dresden.de/~umaschek/estw/ESTW.pdf (aufgerufen am 28.11.2022), 1996.

[MATOJ] John Mattsson: Overview and Analysis of Overhead Caused by TLS - IETF, https://www.ietf.org/archive/id/draft-mattsson-uta-tls-overhead-00.xml.

[MIC19] Team, MSRC: We need a safer systems programming language – Microsoft Security Response Center. Hg. v. Microsoft, 2019, https://msrc-blog.microsoft.com/2019/07/18/we-need-a-safer-systems-programming-language/ (aufgerufen am 21.11.2022).

[MOB22] MOBATIME (2022): Zusammenfassung der NTP- und PTP-Informationen – MOBATIME, https://www.mobatime.com/de/Artikel/ntp-und-ptp-Informationen/ (aufgerufen am 14.11.2022).

[NETTLF] The Linux Foundation: networking:netem [Wiki], https://wiki.linuxfoundation.org/networking/netem (aufgerufen am 08.11.2022).

[ORA20] Oracle VM VirtualBox Admin Guide (2020), https://docs.oracle.com/en/virtualization/virtualbox/6.0/admin/fine-tune-timers.html (aufgerufen am 13.11.2022).

[OSRASTA] RaSTA. Railway CCS (2022), https://github.com/Railway-CCS/rasta-protocol (aufgerufen am 24.11.2022).

[PAP22] IPSec Bandwidth Overhead Using AES - Packet Pushers, https://packetpushers.net/ipsec-bandwidth-overhead-using-aes/ (aufgerufen am 26.11.2022).

[PAS17] Pascual, Silvia; Makkinga, Frits: Reduzierung der Betriebskosten für ERTMS Level 2 durch Implementierung einer standardisierten Stellwerksschnittstelle. In: *SIGNAL + DRAHT* (2017), Nr. 109, S. 2–9.

[PER21] Pereira, Rui; Couto, Marco; Ribeiro, Francisco; Rua, Rui; Cunha, Jácome; Fernandes, João Paulo; Saraiva, João: Ranking programming languages by energy efficiency. In: *Science of Computer Programming* 205 (2021), S. 102609, https://linkinghub.elsevier.com/retrieve/pii/S0167642321000022 (aufgerufen am 21.11.2022).

[POS80] Postel, Jon (1980): User Datagram Protocol - RFC 768, https://www.ietf.org/rfc/rfc768.txt (aufgerufen am 25.11.2022).

[PRO] Prossimo (Hrsg.): What is memory safety and why does it matter?, https://www.memorysafety.org/docs/memory-safety/ (aufgerufen am 21.11.2022).

[RAS15] DIN VDE V 0831-200: Elektrische Bahn-Signalanlagen Teil 200: Sicheres Übertragungsprotokoll RaSTA nach DIN EN 50159.

[RED] Red Hat Customer Portal (Hrsg.): Chapter 8. KVM Guest Timing Management Red Hat Enterprise Linux 7, https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/chap-kvm_guest_timing_management (aufgerufen am 14.11.2022).

[REV22] Revolution Pi (Hrsg.): Smart Rail - Industrial Raspberry Pi, 2022, https://revolutionpi.com/success-stories/smart-rail/, zuletzt aktualisiert am 29.11.2022 (aufgerufen am 29.11.2022).

[SCH21] Schmid, Robert; Boockmeyer, Arne; Pirl, Lukas; Berglehner, Randolf; Cherif, Ibtihel; Korff, Andreas et al.: EULYNX-Live: Eine Methodik zum Validieren von Systemspezifikationen in hybriden Feldtests. In: *Signal+Draht* 6 (2021), Nr. 113, S. 24–31.

[SCH22] Schmid, Robert (2022): EULYNX-Live Lab: Testautomatisierung auf Basis von digitalen Planungsdaten Digital Rail Summer School, Jöhstadt. 15.06.2022.

[SIE18] „TSI CCS: Gut gemeint – aber auch gut gemacht?“. Siemes Mobility. 2018, https://www.oevg.at/fileadmin/user_upload/Editor/Dokumente/Veranstaltungen/2018/regelungswut2/Biester.pdf.

[SOB12] Peter Sobe (2012): Internet-Grundlagen und Überblickswissen, HTW Dresden, https://www2.htw-dresden.de/ sobe/Vorjahre/Vo_Internet_2012/1_Internet_Grundlagen_bis37.pdf.

[STO20] BMitch (2020): Time in Docker container out of sync with host machine - Stackoverflow. 01.04.2020, https://stackoverflow.com/a/60982142/10106315.

[TCP92] Information Sciences Institute University of Southern California (Hrsg.): TRANSMISSION CONTROL PROTOCOL, 1992, https://www.ietf.org/rfc/rfc793.txt, zuletzt aktualisiert am 16.10.1992 (aufgerufen am 29.11.2022).

[TEC22] Techtarget (Hrsg.): Routing First-Step: IP header format, https://www.techtarget.com/searchnetworking/tutorial/Routing-First-Step-IP-header-format (aufgerufen am 25.11.2022).

[TRI20] Trinckauf, Jochen; Kahl, Richard; Maschek, Ulrich; Krahl, Claudia (Hrsg.): ETCS in Deutschland. Unter Mitarbeit von Buder, Jens; Han, Yadi; Kretschmer, Elisabeth; Kunze, Michael Dieter und Sommer, Martin Erik. 1. Auflage, Leverkusen, Hamburg: PMC (Edition Eurail press), 2020.

[TUBOJ] TU Braunschweig: Zeitsynchronisation: Institut für Betriebssysteme und Rechnerverbund - TU Braunschweig, https://www.ibr.cs.tu-bs.de/users/thuerman/time/sync.html (aufgerufen am 13.11.2022).

[TUM] TU München: Stellwerksgeschichte, https://www5.in.tum.de/lehre/seminare/semsoft/unterlagen_02/eisenbahn/website/stellwerksgeschichte.html.

[VIR22] Oracle VM VirtualBox (2022), https://www.virtualbox.org/, zuletzt aktualisiert am 29.11.2022 (aufgerufen am 29.11.2022).

[VVI20] Eisenbahn-Bundesamt (01.07.2020): VV IBG Infrastruktur, vom 1.1.

[WGC21] WG CYSIS & UG IT Security IoRT (2021): IT/ OT-Security for Internet of Railway Things (IoRT): Working Group CYSIS: cisco and Fraunhofer SIT and INCYDE and OBB. 18.01.2021, https://www.sit.fraunhofer.de/fileadmin/dokumente/artikel/210117_Whitepaper_IoRT-Security_v13_EN_final.pdf.

[WIE21] Wieners, Jan G.: Rechnerkommunikation I Schichtenmodelle, Universität Köln. Sommersemester 2021.

[WIK22] Wikimedia (Hrsg.): Data packet of IPv4, 2022, https://en.wikipedia.org/w/index.php?title=IPv4&oldid=1123185459 (aufgerufen am 27.11.2022).

Erklärung

Hierdurch erkläre ich, dass ich die von mir am heutigen Tage eingereichte Diplomarbeit selbstständig verfasst und andere als die angegebenen Hilfsmittel nicht benutzt habe.

Der Verfasser erteilt der Technischen Universität an den Ergebnissen seiner Diplomarbeit ein nichtausschließliches, nichtübertragbares, zeitlich unbegrenztes, kostenloses und unwiderrufliches Nutzungsrecht.

Dresden, den 30.11.2022 .....

Unterschrift des Studenten

Anhang A: Übersicht über Logging-Nachrichten

	Actor	Protocol	Daten	OrderID	Messagetype
1	INTERLOCKING_SENT	SCI-P	1/2	x	MOVE_POINT
2	RASTA_SENT	SCI-P	1/2	x	MOVE_POINT
3	RASTA_RECEIVED	SCI-P	1/2	x	MOVE_POINT
4	OC_RECEIVED	SCI-P	1/2	x	MOVE_POINT
4.1	OC_SENT	SCI-P	3/4	x	POINT_POSITION
4.2	RASTA_SENT	SCI-P	3/4	x	POINT_POSITION
4.3	RASTA_RECEIVED	SCI-P	3/4	x	POINT_POSITION
4.4	INTERLOCKING_RECEIVED	SCI-P	3/4	x	POINT_POSITION
5	OC_SENT	SCI-P	1/2	x	POINT_POSITION
6	RASTA_SENT	SCI-P	1/2	x	POINT_POSITION
7	RASTA_RECEIVED	SCI-P	1/2	x	POINT_POSITION
8	INTERLOCKING_RECEIVED	SCI-P	1/2	x	POINT_POSITION
9	INTERLOCKING_SENT	SCI-LS	4	x	INDICATE_SIGNAL_ASPECT
10	RASTA_SENT	SCI-LS	4	x	INDICATE_SIGNAL_ASPECT
11	RASTA_RECEIVED	SCI-LS	4	x	INDICATE_SIGNAL_ASPECT
12	OC_RECEIVED	SCI-LS	4	x	INDICATE_SIGNAL_ASPECT
13	OC_SENT	SCI-LS	4	x	INDICATED_SIGNAL_ASPECT
14	RASTA_SENT	SCI-LS	4	x	INDICATED_SIGNAL_ASPECT
15	RASTA_RECEIVED	SCI-LS	4	x	INDICATED_SIGNAL_ASPECT
16	INTERLOCKING_RECEIVED	SCI-LS	4	x	INDICATED_SIGNAL_ASPECT

	Kommentar
4	Keine Endlage
8	Fahrstraße festgelegt
12	Signalbegriff Fahrt am Signal
16	Bestätigung Fahrtbefehl im Stellwerk

Anhang B: Email-Kommunikation Dr. Lukas Iffländer

Von: Frederic Reiter <reiter.frederic@gmail.com>
Gesendet: 14.10.2022, 14:24
An: Dr. Lukas Iffländer <IfflaenderL@dzsf.bund.de>
Betreff: Fachaussage zu relevanten Themen der Diplomarbeit - Frederic Reiter

Sehr geehrter Herr Dr. Iffländer,

im Laufe der Bearbeitung meiner Diplomarbeit sprachen wir über verschiedene Themenbereiche mit Bedeutung für die Inhalte der Arbeit und Relevanz für Aufbau und Detailgestaltung der Evaluationsumgebung. Ich möchte Sie diesbezüglich noch einmal schriftlich um eine fachliche Aussage bitten.

Es geht insbesondere um folgende Themenkomplexe:

1. Auswirkungen des monolithischen Designs von Stellwerken auf Konditionen, die den EIU geboten werden.

2. Auswahl des RaSTA-Protokolls als Testobjekt für die Validierung der Evaluationsumgebung.

3. Auswahl der zu nutzenden Programmiersprachen, darunter Rust für die Object Controller und Python für die Simulation des Stellwerks und die Experimentsteuerung.

4. Anordnung von Komponenten in den Object Controllern.

5. Möglichkeit der Normalisierung und Vergleichbarkeit von Prozessoren mittels der erreichten Punktzahlen in Benchmarks.

Mit freundlichen Grüßen

Frederic Reiter

Von: Dr. Lukas Iffländer <IfflaenderL@dzsf.bund.de>
Gesendet: 14.10.2022, 16:10
An: Frederic Reiter <reiter.frederic@gmail.com>
Betreff: RE: Fachaussage zu relevanten Themen der Diplomarbeit - Frederic Reiter

Sehr geehrter Herr Reiter,

1. **Auswirkungen des monolithischen Designs von Stellwerken auf Konditionen, die den EIU geboten werden.**

Der Austausch von einzelnen Komponenten gegen Komponenten anderer Hersteller ist schwer bis unmöglich. Dies schafft für Erweiterungen und Umbauten künstliche Monopole und diese werden dadurch teils massiv verteuert. Bsp.: Digitale S-Bahn Hamburg Aufrüstung von SIMIS-C Stellwerk mit ETCS Level 2 im mittleren zweistelligen Millionenbereich.

2. **Auswahl des RaSTA-Protokolls als Testobjekt für die Validierung der Evaluationsumgebung.**

Ich würd hier die Implementierung von Markus Heinrich (Open Source) als Startpunkt empfehlen. Das ermöglicht uns auch eventuell festgestellte Probleme und Fehler selbst zu korrigieren. https://github.com/Railway-CCS/rasta-protocol

3. **Auswahl der zu nutzenden Programmiersprachen, darunter Rust für die Object Controller und Python für die Simulation des Stellwerks und die Experimentsteuerung.**

Rust ist wünschenswert, da es sich um eine speichersichere Programmiersprache handelt. Dies erfüllt dadurch die Anforderungen aus EN 50128 und dürfte für Zulassung und Nachweise hilfreich sein. Zudem ist sie effizient und in kompiliert, was ein Deployment auf einem kleinen System ermöglicht. Für die Object Controller wäre es im Sinne der Realitatsnähe daher wünschenswert, in dieser Sprache zu arbeiten.

Python ist für das Stellwerk hier ausreichend, da wir das rein als Simulationstreiber nutzen und keine große zu validierende Fahrstraßenlogik o. Ä. integrieren.

4. **Anordnung von Komponenten in den Object Controllern.**

DB Netz AG hat bei den bisher geplanten und gebauten DSTW die Object Controller mittels IPSec-VPN und einer Crypto-Box geschützt. Die genauen Implementierungen und Konfigurationen dieser Geräte werden leider nicht transparent veröffentlicht, so dass darüber keine hinreichend präzise Aussage möglich ist.

5. **Möglichkeit der Normalisierung und Vergleichbarkeit von Prozessoren mittels der erreichten Punktzahlen in Benchmarks.**

Standardbenchmarks wie z.B. CineBench oder SecureMark sollten hier für den Vergleich ausreichend sein. Es geht hier ja primär um die Größenordnung und nicht um die präzise Bestimmung der notwendigen Leistung auf das letzte MHz.

Viele Grüße

Dr. Lukas Iffländer

Wissenschaftlicher Referent

Deutsches Zentrum für Schienenverkehrsforschung beim Eisenbahn-Bundesamt

August-Bebel-Straße 10

01219 Dresden

Tel: +49 (0)351 47931-144

E-Mail: IfflaenderL@dzsf.bund.de

Forschung@dzsf.bund.de

Internet: www.dzsf.bund.de

Nähere Informationen zum Datenschutz finden Sie auf der Homepage des DZSF.

Anhang C: Email-Kommunikation „Protokolle von Laufzeiten“ - Petros Matios

Von: Dr. Lukas Iffländer <IfflaenderL@dzsf.bund.de>
Gesendet: 19.09.2022, 15:37
An: Frederic Reiter <reiter.frederic@gmail.com>
Betreff: WG: Protokolle von Laufzeiten!

[…]

Latenzen der DB.

Blau ist primäres Netzwerk, Grau ist Rückfallebene.

MDTX ist der TSO (Technikstandort aka Stellwerk) MMTG ist der GFK in Mertingen und MMEI der in Meitingen.

[…]

-----Ursprüngliche Nachricht-----
Von: Petros Matios <petros.matios@deutschebahn.com>
Gesendet: Montag, 19. September 2022 15:29
An: Iffländer, Lukas <IfflaenderL@dzsf.bund.de>
Cc: Frank F Witte -Extern <Frank.F.Witte-extern@deutschebahn.com>
Betreff: Protokolle von Laufzeiten!

Mit freundlichen Grüßen

Petros Matios
Systemverantwortung Zentralsysteme, I.NAI 513

DB Netz AG
Adam-Riese-Str. 11-13, Zentrale DB Netz, 60327 Frankfurt a. Main Tel. +49 69 265 52537, intern: 95552537
Mobil: 0151 19524997

Pflichtangaben anzeigen <http://www.deutschebahn.com/pflichtangaben/20220916>

Nähere Informationen zur Datenverarbeitung im DB-Konzern finden Sie hier: http://www.deutschebahn.com/de/konzern/datenschutz <http://www.deutsche-bahn.com/de/konzern/datenschutz>

4 Anhänge — Alle Anhänge herunterladen