Universität Rostock

Traditio et Innovatio

Zeitsynchronisation in drahtgebundenen Rechnernetzen

Der Fakultät für Informatik und Elektrotechnik
der Universität Rostock zur Erlangung des
akademischen Grades eines

https://doi.org/10.18453/rosdok_id00003269

Dr.-Ing.

vorgelegte Dissertation von
Henning Puttnies
geb. am 23. April 1990 in Stralsund

eingereicht am 9. März 2021 und verteidigt am 8. September 2021

Gutachter:
Prof. Dr.-Ing. Dirk Timmermann
Universität Rostock
Richard-Wagner-Str. 31
18119 Rostock-Warnemünde

Zweitgutachter:
Prof. Dr.techn. Wolfgang Kastner
Technische Universität Wien
Treitlstraße 3
1040 Wien

Dissertation
Fakultät für Informatik und Elektrotechnik

Henning Puttnies
Institut für Angewandte Mikroelektronik und Datentechnik

Dieses Werk ist lizenziert unter einer
Creative Commons Namensnennung - Weitergabe unter gleichen
Bedingungen 4.0 International Lizenz.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die von mir vorgelegte Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit, einschließlich Tabellen und Abbildungen, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Rostock, den 9. März 2021

_____

Henning Puttnies

Danksagung

Das Verfassen dieser Dissertationsschrift wäre ohne die Hilfe und Unterstützung einiger Personen nicht möglich gewesen. Daher möchte ich an dieser Stelle Dank sagen.

Zunächst ist meine Familie zu nennen, die mir Unterstützung und Motivation auf allen Schritten meines Bildungsweges zuteil hat werden lassen. Insbesondere sei hier meiner lieben Frau Jana gedankt, mit der ich speziell das Hinterfragen des eigenen Standpunktes trainieren durfte, ohne welches die Wissenschaft meines Erachtens nicht denkbar ist.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr.-Ing. Dirk Timmermann. Bedanken möchte ich mich für die Chance überhaupt zu promovieren, aber auch für Motivation, Inspiration sowie Rat bei der Erstellung dieser Forschungsarbeit. Danke, für eine aus meiner Sicht herausragende, sehr angenehme und produktive Atmosphäre des Lernens, Lehrens und Forschens.

Zu guter Letzt gilt mein Dank auch allen Kollegen am Institut für Angewandte Mikroelektronik und Datentechnik. Für die Unterstützung, die gute Zusammenarbeit und insbesondere für wertvolle Diskussionen bzw. Reflexionen möchte ich mich bedanken. Mein Dank gilt auch den Korrekturlesern dieser Dissertationsschrift. Insbesondere bedanken möchte ich mich bei Dr.-Ing. habil. Peter Danielis und Dr.-Ing. Michael Rethfeldt, sowohl für hilfreiche Anmerkungen und Diskussionen als auch für hilfreiche Kritiken.

Lebenslauf

2014 - 2021	Promotionsstudium, Universität Rostock Abschluss: Dr.-Ing.
2013 - 2014	Masterstudium Elektrotechnik, Universität Rostock Abschluss: M.Sc.
2009 - 2013	Bachelorstudium Elektrotechnik, Universität Rostock Abschluss: B.Sc.

Kurzzusammenfassung

Zeitsynchronisation ist essentiell für koordiniertes Agieren in verteilten Systemen. Als Beispiele lassen sich die verteilte Datenerfassung oder die automatisierte Industrieproduktion nennen.

Bestehende Protokolle sind entweder wenig genau, z.B. NTP (Network Time Protocol), oder bieten zwar eine hohe Genauigkeit, benötigen hierfür jedoch spezialisierte Hardware, z.B. PTP (Precision Time Protocol), was zu hohen Kosten führt.

Als Folge dessen wird in dieser Forschungsarbeit zunächst eine umfangreiche Analyse des Standes der Technik auf dem Gebiet der Zeitsynchronisation vorgenommen. Insbesondere wird hierbei evaluiert, inwiefern die Adaption neuer Algorithmen und Verfahren aus dem Bereich der drahtlosen Netzwerke auf drahtgebundene Netzwerke möglich ist.

Ausgehend von dieser Analyse werden neuartige Verfahren für die Zeitsynchronisation in drahtgebundenen Netzwerken vorgestellt. Erstens wird die Synchronisation mittels Broadcast-Nachrichten untersucht, was zu einer deutlichen Verbesserung der Skalierbarkeit führt (PSPI-Sync bzw. Precise, Scalable, and Platform Independent Clock Synchronization). Zweitens wird zum ersten Mal eine Untersuchung vorgenommen, inwiefern sich PTP mittels linearer Optimierung (LP) verbessern lässt (PTP-LP). Dies führt zu einer Verbesserung der Synchronisationsgenauigkeit und dazu, dass unter Umständen auf spezialisierte Hardware im Verbindungsnetzwerk verzichtet werden kann. Drittens wird, zum ersten Mal im Bereich der Zeitsynchronisation, eine Kombination aus LP und Broadcast-Nachrichten vorgestellt (SLMT bzw. Clock Synchronization Using Linear Programming, Multicasts, and Temperature Compensation), was eine Verbindung der jeweiligen Genauigkeits- und Skalierbarkeitsvorteile ermöglicht. Viertens wird, ebenfalls zum ersten Mal im Bereich der Zeitsynchronisation, eine Kombination aus LP und Temperaturkompensation vorgestellt. Dies führt zu einer weiteren Verbesserung der Synchronisationsgenauigkeit. Hierbei wurden zwei grundlegende Temperaturkompensationsansätze untersucht: SLMT-IMM als Verbindung aus SLMT und einem IMM (Interacting-Multiple-Model-Kalman-Filter) und SLMT-TH als Kombination von SLMT und einer Heuristik (Temperature Heuristic).

Zur Bewertung der Ansätze werden umfangreiche Untersuchungen durchgeführt: numerische Simulationen, Netzwerksimulationen mittels OMNeT++ (hierfür wurden die selbst entwickelten Java-Erweiterungen von OMNeT++ eingesetzt), sowie Messungen auf realen Geräten. Die Untersuchungen zeigen sowohl Nachteile der Ansätze als auch deren vielversprechendes Potential. So kann die Synchronisationsgenauigkeit in bestimmten Szenarien gegenüber dem Stand der Technik um mehrere Größenordnungen verbessert werden.

Abstract

Time (or clock) synchronization is essential for coordinated activities in distributed systems. Typical example applications are distributed data acquisition or industrial automation.

Existing synchronization protocols are either not very precise, e.g. NTP (Network Time Protocol), or require specialized hardware to offer high accuracy, e.g. PTP (Precision Time Protocol), which leads to high costs.

Initially, this thesis provides an extensive analysis of the state of the art in the research field of time synchronization. In particular, it evaluates the possibility of adapting new algorithms and methods from the field of wireless networks to the wired domain.

Based on this analysis, several new time synchronization methods for wired networks are proposed. Firstly, broadcast-based synchronization is examined, which leads to significant scalability improvements (PSPI-Sync or Precise, Scalable, and Platform Independent Clock Synchronization). Secondly, for the first time, this thesis examines, if PTP can be extended with linear programming (LP), which improves the accuracy (PTP-LP). Consequently, specialized hardware might not be needed in the network. Thirdly, for the first time in the synchronization domain, a combination of LP and broadcast messages is proposed (SLMT or Clock Synchronization Using Linear Programming, Multicasts, and Temperature Compensation), which allows combining the respective accuracy and scalability advantages. Fourthly, this thesis proposes combining LP and temperature compensation, also for the first time in the field of time synchronization. This further improves the synchronization accuracy. Basically, two temperature compensation approaches are proposed: SLMT-IMM as a combination of SLMT and an IMM (Interacting Multiple Model Kalman Filter) and SLMT-TH as a combination of SLMT and a (temperature) heuristic.

An extensive evaluation examines the approaches. It is based on numerical simulations, network simulations using OMNeT++ (the self-developed Java extensions for OMNeT++ are used for this), as well as measurements on real devices. The evaluation shows both disadvantages of the approaches and their promising potential. It is shown that the synchronization accuracy can be improved by several orders of magnitude in certain scenarios compared to the state of the art.

Thesen

1. Zeitsynchronisation entspricht der Bereitstellung einer gemeinsamen Zeitbasis für Geräte und ist wesentliche Voraussetzung für koordinierte Aktivitäten in einem verteilten System. Als Beispiele sind die Echtzeitkommunikation basierend auf TDMA (Time Division Multiple Access) oder die verteilte Datenerfassung zu nennen.

2. Die größten Herausforderungen der Zeitsynchronisation sind, dass Taktgeber und Kommunikationskanäle in der Realität nicht ideal sind. Dadurch entstehen Ungenauigkeiten, die es zu kompensieren gilt.

3. Folgende Anforderungen an einen Zeitsynchronisationsansatz können definiert werden: Genauigkeit, Bandbreiteneffizienz, Rechenleistungseffizienz und geringe Hardware-Anforderungen.

4. Im besten Fall stellt ein Ansatz keine speziellen Anforderungen an die Switches und kompensiert die im Netzwerk entstandenen Verzögerungen mittels leistungsstarker Schätzer auf den Endknoten.

5. Die Analyse des Standes der Technik ergibt, dass als Schätzer insbesondere die lineare Optimierung (LP) sehr gute Ergebnisse bei der Kompensation der Verzögerungen erzielt. In WSNs (Wireless Sensor Networks) kommt stattdessen insbesondere die lineare Regression (LR) aufgrund ihrer noch geringeren Rechenkomplexität vermehrt zum Einsatz.

6. Aus der Analyse des Standes der Technik ergibt sich weiterhin, dass im drahtlosen Bereich viele Broadcast-basierte Ansätze zur Verbesserung der Skalierbarkeit vorgeschlagen wurden. Broadcasts könnten auch für drahtgebundene Netze interessant sein. Insbesondere für Szenarien im IIoT (Industrial Internet of Things) ist die Skalierbarkeit aufgrund der Vielzahl der zu vernetzenden Geräte essentiell.

7. Der vorgestellte Ansatz PSPI-Sync (Precise, Scalable, and Platform Independent Clock Synchronization) basiert auf zwei grundlegend neuen Konzepten: einem neuen, auf einem linearen Gleichungssystem basierenden Ansatz zur Verzögerungsbestimmung sowie der Trennung von Verzögerungsbestimmung auf Basis von Unicast-Nachrichten und eigentlicher Synchronisation auf Basis von Broadcast-Nachrichten.

8. Der PSPI-Sync-Ansatz ist hochgradig skalierbar, da Synchronisationen und Resynchronisationen Broadcast-Nachrichten verwenden können.

9. Obwohl insbesondere der PTP-Standard (Precision Time Protocol, IEEE 1588) theoretisch eine Genauigkeit im Bereich von Nanosekunden erreichen kann, sinkt die praktische Genauigkeit von PTP bei stark variabler Paketverzögerung. Daher wird in dieser Arbeit der PTP-LP-Ansatz zur Erhöhung der Synchronisationsgenauigkeit von PTP vorgestellt.

10. PTP-LP besteht aus PTP und einem zusätzlichen Verarbeitungsschritt zur Verbesserung der Schätzung. Obwohl bereits vorgeschlagen wurde, PTP um eine Schätzung mittels eines Kalman-Filters zu erweitern (PTP-Kalman), gibt es nach aktuellem Kenntnisstand keinen Ansatz, der LP als Erweiterung vorschlägt oder untersucht.

11. In einer umfangreichen Evaluation wird PTP-LP mit zwei bekannten Ansätzen verglichen: Standard-PTP und PTP-Kalman. PTP-LP zeigt sich robust gegenüber Verzögerungsvariationen, übertrifft beide unter nahezu allen untersuchten Bedingungen und erhöht die Synchronisationsgenauigkeit signifikant.

12. PTP-LP ist vollständig kompatibel zum gängigen PTP-Standard, da weder der PTP-Master noch die Nachrichten zwischen Master und Slave geändert werden.

13. PTP-LP erzielt die besten Ergebnisse für einen stabilen Hardware-Takt (z.B. Hardware-Oszillator) und unbekannte, nicht zu vernachlässigende Variationen der Netzwerkverzögerung. Beides sind sehr realistische Arbeitsbedingungen.

14. Es wird in dieser Arbeit gezeigt, dass Ansätze, die auf LP basieren (bspw. PTP-LP), das Problem der Verzögerungsvariationen stark mildern bzw. lösen können. Änderungen der Taktfrequenz führen jedoch zu nichtlinearen Zeitfunktionen, die durch LP-basierte Ansätze allein nicht gut kompensiert werden können. Als Konsequenz wird der Ansatz SLMT (Clock Synchronization Using Linear Programming, Multicasts, and Temperature Compensation) vorgestellt, der LP, Multicasts und Temperaturkompensation für die Zeitsynchronisation verwendet.

15. In einer umfassenden Evaluation von SLMT und dem Vergleich mit vielen Ansätzen aus dem SOTA (Stand der Technik) wird gezeigt, dass SLMT die SOTA-Ansätze meist übertrifft, insbesondere unter rauen Bedingungen, z.B. Temperaturänderungen und unbekannten, nicht zu vernachlässigenden Netzwerkverzögerungen.

16. Nach bestem Wissen des Autors ist SLMT der erste Synchronisationsansatz, der LP und One-Way-Exchange bzw. Multicasts kombiniert.

17. Die Evaluation von SLMT zeigt, dass die Kombination von Multicasts und LP sowohl möglich als auch vielversprechend ist, da sich der Overhead für die Synchronisation im Vergleich zur herkömmlichen Zwei-Wege-Synchronisation verringert und Multicasts nur zu einer geringfügigen Verringerung der Präzision führen.

18. Darüber hinaus wird nach bestem Wissen des Autors mit SLMT in dieser Arbeit die erste Kombination aus LP und Temperaturkompensation für die Synchronisation vorgeschlagen. Dies ist wichtig, um einem konzeptionellen Nachteil von LP zu begegnen: Nichtlinearitäten der Taktfunktion können nicht kompensiert werden, da mittels LP immer nur lineare Funktionen geschätzt werden können.

19. Die Evaluation von SLMT zeigt, dass auch die Kombination von LP und Temperaturkompensation sowohl möglich als auch vielversprechend ist.

20. Hier lässt sich insbesondere festhalten, dass die einfache Kombination aus SLMT und einer Temperaturheuristik (SLMT-TH), basierend auf dem Offline-Algorithmus TCTS (Temperature Compensated Time Synchronization), auch bei hoher Netzwerklast robust ist. Allerdings wird hier Vorwissen über die Korrelation zwischen Temperatur und Frequenz benötigt.

21. Im Gegensatz dazu funktioniert die Kombination von SLMT mit einem komplexen IMM (Interacting-Multiple-Model-Kalman-Filter), bestehend aus mehreren Kalman-Filtern, zur Temperaturkompensation ohne Vorkenntnisse vor allem bei geringer und mittlerer

Netzwerklast gut. Unter rauen Netzwerkbedingungen nimmt die Genauigkeit jedoch ab, da Kalman-Filter die nicht-gaußschen Unsicherheiten nicht ausgleichen können.

22. Auch die Adaption von drahtlosen Verfahren auf drahtgebundene Netze kann natürlich sinnvoll sein, ist aber explizit nicht Thema dieser Arbeit. Als zukünftige Erweiterung ließen sich die vorgestellten Ansätze allerdings auch für den Bereich drahtloser Netze anpassen und anwenden. An dieser Stelle sei nochmal erwähnt, dass auch im drahtlosen Bereich kein Ansatz mit den Eigenschaften von SLMT existiert. Die Verbindung aus Broadcasts und LP bzw. Temperaturkompensation und LP ist aus Sicht des Autos aber auch für diesen Bereich interessant. Zwar besteht in drahtlosen Echtzeitnetzen nach aktuellem Stand der Forschung weiterhin ein Zuverlässigkeitsproblem, aber dieses gilt nicht für die Synchronisation. Der Verlust einzelner Pakete stellt z.B. für die LP-basierte Synchronisation kein Problem dar und auch das dauerhafte Ausbleiben von Paketen lässt sich kompensieren. Erstens wird das Wegdriften durch die Temperaturkompensation verlangsamt. Zweitens kann das Slave Gerät feststellen, dass keine Pakete mehr ankommen und selbst aktiv angemessen auf diese Situation reagieren (z.B. in einen sichereren Fehlermodus umschalten). In keinem Fall käme es jedoch zu einem Ausfall des Systems.

23. Des Weiteren wäre eine Untersuchung weiterer Schätzverfahren lohnenswert. Diese sollten vor allem robuster gegenüber Burst-Traffic sein als es das Kalman-Filter ist.

Inhaltsverzeichnis

Abbildungsverzeichnis **XVI**

Tabellenverzeichnis **XVII**

Abkürzungsverzeichnis **XIX**

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Problemstellung und Zielsetzungen der Arbeit . . . 2
1.3 Aufbau der Arbeit . . . 3

2 Grundlagen **5**
2.1 Netzwerke: Referenzmodelle und Protokolle . . . 5
2.2 Zeitsynchronisation . . . 7
2.2.1 Beschreibung des Problems der Zeitsynchronisation . . . 7
2.2.2 Oszillatoren und Taktgeber . . . 8
2.2.3 Taktmodelle . . . 9
2.3 Grundlagen für Kalman-Filter: Zustandsraumbeschreibungen und Wahrscheinlichkeitstheorie . . . 10
2.3.1 Zustandsraumbeschreibungen . . . 11
2.3.2 Wahrscheinlichkeitstheorie . . . 12
2.4 Mathematische Verfahren zur Verbesserung der Schätzung von Taktparametern 19
2.4.1 Exponentielle Filterung . . . 19
2.4.2 Lineare Optimierung (LP) . . . 21
2.4.3 Kalman-Filter . . . 24
2.4.4 Adaptive Kalman-Filter - IMM . . . 31

3 Stand der Technik und Forschung **39**
3.1 Grundlegende Beiträge . . . 39
3.1.1 Zeitsynchronisation in Rechnernetzen . . . 39
3.1.2 Synchronisation in komplexen Netzwerken . . . 40
3.1.3 Alternativen zur Synchronisation . . . 40
3.2 Vergleichsstudien (Surveys) . . . 41
3.2.1 Studien ohne Fokus auf WSNs (Wireless Sensor Networks) . . . 41
3.2.2 Studien mit Fokus WSNs (Wireless Sensor Networks) . . . 42
3.3 Methodik zur Bewertung der Ansätze . . . 43
3.4 Protokolle zur Zeitsynchronisation . . . 44
3.5 Forschungsansätze zur Zeitsynchronisation . . . 47
3.5.1 PTP/gPTP Modifikationen . . . 47
3.5.2 Drahtgebundene Ansätze . . . 51
3.5.3 Drahtlose Ansätze . . . 56
3.6 Ansätze zur Taktstabilisierung . . . 61
3.7 Zwischenfazit . . . 64

4 PSPI-Sync: Ein Ansatz für präzise, skalierbare und plattformunabhängige Zeitsynchronisation 66
4.1 Einleitung und Motivation . . . 67
4.1.1 Verzögerungsbestimmung . . . 68
4.1.2 Synchronisation mittels Broadcasts . . . 69
4.2 Vergleich mit dem Stand der Technik . . . 70
4.2.1 Verzögerungsbestimmung und -reduktion . . . 70
4.2.2 Zeitsynchronisation . . . 72
4.3 Der PSPI-Sync-Ansatz . . . 74
4.3.1 Mathematisches Konzept zur Erzeugung eines lösbaren Gleichungssystems . . . 74
4.3.2 Ablauf der Synchronisation . . . 76
4.3.3 Betrachtung von konzeptionellen Fehlern, Ausfallsicherheit und Skalierbarkeit . . . 77
4.4 Implementierung und Versuchsaufbau . . . 78
4.5 Experimente und Auswertung . . . 79
4.5.1 Skalierbarkeit der Zeitsynchronisation . . . 79
4.5.2 Genauigkeit der Zeitsynchronisation . . . 79
4.6 Zwischenfazit . . . 88

5 PTP-LP: Verbesserung der Robustheit von PTP gegenüber Variationen der Paketverzögerung mittels linearer Optimierung 90
5.1 Einleitung und Motivation . . . 90
5.2 Vergleich mit dem Stand der Technik . . . 92
5.3 Notation . . . 94
5.4 IEEE 1588: Precision Time Protocol (PTP) . . . 94
5.5 Der PTP-LP-Ansatz . . . 96
5.5.1 Problemformulierung . . . 96
5.5.2 PTP-H: Eine Heuristik für PTP-LP . . . 99
5.5.3 Konzeptionelle Nachteile von PTP-LP . . . 100
5.6 Verwendetes Taktmodell . . . 100
5.7 Experimente und Auswertung . . . 101
5.7.1 Methodik . . . 102
5.7.2 Verwendete Taktstabilitäten . . . 102
5.7.3 Verwendete Paketverzögerungsverteilungen: Gaußsche Verzögerung und selbstähnliche Verzögerung . . . 103
5.7.4 Auswertung mit gaußscher Paketverzögerung . . . 105
5.7.5 Auswertung mit selbstähnlicher Paketverzögerung . . . 108
5.7.6 Berechnungskomplexität . . . 110
5.8 Zwischenfazit . . . 111

6 SLMT: Zeitsynchronisation mittels linearer Optimierung, Multicasts und Temperaturkompensation 113
6.1 Einleitung und Motivation . . . 113
6.2 Vergleich mit dem Stand der Technik . . . 115

6.3 Notation 116
6.4 Der SLMT-Ansatz 117
6.4.1 Problemformulierung 118
6.4.2 Heuristiken für die LP-Berechnung 120
6.4.3 Temperaturkompensation 121
6.5 Taktmodelle 122
6.5.1 Random-Walk-Taktmodell 122
6.5.2 Temperaturtaktmodell 123
6.5.3 Kompensation von Temperatur bzw. Drift 123
6.6 Experimente und Auswertung 124
6.6.1 Methodik 124
6.6.2 Evaluation mittels Random-Walk-Taktmodell 127
6.6.3 Evaluation mittels Temperaturtaktmodell 129
6.6.4 Berechnungskomplexität 130
6.7 Zwischenfazit 132

7 Zusammenfassung und Ausblick 134
7.1 Zusammenfassung dieser Forschungsarbeit 134
7.2 Ausblick 135

A Literaturverzeichnis I

B Liste der Veröffentlichungen und Fachvorträge auf Tagungen XVII

C Liste der betreuten studentischen Arbeiten XIX

Abbildungsverzeichnis

1.1 Übersicht der Kapitelstruktur dieser Forschungsarbeit. Die wesentlichen eigenen Beiträge sind fett hervorgehoben. 4
2.1 Kommunikation zwischen zwei Netzwerkknoten, die über einen Router verbunden sind. Darstellung der Schichten mit ISO/OSI- Referenzmodell (Netzwerkknoten A, Router) und Beispielprotokollen aus dem Internet-Protokoll-Stack (Netzwerkknoten B) [A 16, 17, 18] 7
2.2 Zustandsraumbeschreibungen eines zeitkontinuierlichen Systems [A 27] . . . 11
2.3 Zustandsraumbeschreibungen eines zeitdiskreten Systems [A 27] 11
2.4 Zustandsraumbeschreibungen eines zeitdiskreten Systems [A 27] 12
2.5 Das Ereignis B ist über verschiedene Zwischenstationen A_i erreichbar [A 29]. 15
2.6 Dichtefunktion der Normalverteilung für verschiedene Erwartungswerte und Standardabweichungen 16
2.7 Menge der gültigen Lösungen und graphische Lösung des Problems 22
2.8 Verlauf messtechnisch erfassbarer Größen der Mondlandefähre: Abstand von der Mondoberfäche $h(t)$ und Beschleunigung $a(t)$, sowie der durch Differenzierung erhaltene Verlauf der Geschwindigkeit $v(t)$ der Mondlandefähre (Abbildungen übernommen aus [A 27]) 26
2.9 Blockdiagramm eines Systems mit dem Eingang u, dem aktuellen Zustand x und dem Ausgang y. 26
2.10 Grundlegende Struktur des Kalman-Filters aus Korrektur und Prädiktion . . . 29
2.11 Verlauf der Größen des korrigierten Zustandsvektors (aus [A 27]) 32
2.12 Grundlegende IMM-Struktur (basierend auf [A 33, 34] 33
2.13 Übergangswahrscheinlichkeiten aus [A 35] 34
2.14 Ergebnisse der eigenen Implementierung von [A 34] 35
4.1 Verzögerung zwischen zwei Geräten in einem drahtgebundenen Netzwerk. . 69
4.2 Die RTTs zwischen mindestens drei Geräten müssen geschätzt werden, um ein lösbares Gleichungssystem zu erhalten. 75
4.3 Untersuchung der Skalierbarkeit auf Basis von Emulation. Die für die Synchronisation benötigte Zeit als Funktion der Anzahl der unterschiedlichen Plattformen im Netz und der Anzahl der Messungen, die pro Schätzung verwendet werden. 80
4.4 SSH-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler. 82
4.5 SSHc-Setup (SSH corrected): Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler. 83
4.6 Serial-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler. 84
4.7 RT-SSH-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler. 86
4.8 RT-Serial-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler. 87

5.1 Zeitfunktionen des Master $T(t)$ und des Slaves $C(t)$ (vereinfacht dargestellt als lineare Funktionen). Hierbei ist $\theta(0)$ der Schnittpunkt von $C(t)$ mit der y-Achse. 94
5.2 Master-Zeit $T(t)$ und Slave-Zeit $C(t)$ mit PTP-Zeitstempeln. Zu beachten ist, dass T_i sich auf die Master-Zeit bezieht und $C(T_i)$ sich auf die Slave-Zeit bezieht ($C(t)$ wird der Einfachheit halber als lineare Funktion dargestellt). Der Punkt $P_1(T_1, C(T_2))$ befindet sich immer oberhalb von $C(t)$ und der Punkt $P_4(T_4, C(T_3))$ immer darunter. 95
5.3 Allan Varianzen für SW- und HW-Takte, die in der Auswertung verwendet wurden (die X-Achse ist logarithmisch dargestellt). 103
5.4 Das obere Diagramm zeigt die lineare Master-Zeit $T(t)$ und die nicht-lineare HW- (bzw. SW-) Zeit $C_{hw}(t)$ (bzw. $C_{sw}(t)$) über einen Zeitraum von 5000 Sekunden. Das untere Diagramm zeigt den Offset zwischen $C_{hw}(t)$ (bzw. $C_{sw}(t)$) und $T(t)$ über den selben Zeitraum 104
5.5 Netzwerkmodell zur Berechnung der selbstähnlichen Verzögerung aus selbstähnlichen Paketankunftszeiten. 105
5.6 Selbstähnlicher Verteilung (Logarithmische Normalverteilung) der Zeit zwischen zwei Paketen aus [A 91] und dieselbe Verteilung skaliert auf 50% Linkauslastung (logarithmische Y-Achse). 106
5.7 Füllungsgrad des Switch-Puffers für selbstähnlichen Datenverkehr mit einer Linkauslastung von 50% 106
5.8 Synchronisationsfehler und Frequenzfehler für **Gaußsche Verzögerung** (die Y-Achse ist logarithmisch dargestellt). 107
5.9 Synchronisationsfehler und Frequenzfehler für **selbstähnliche Verzögerung mit 10%** mittlerer Auslastung (die Y-Achse ist logarithmisch dargestellt). . . . 109
5.10 Synchronisationsfehler und Frequenzfehler für **selbstähnliche Verzögerung mit 90%** mittlerer Auslastung (die Y-Achse ist logarithmisch dargestellt). . . . 110
5.11 Rechenzeit für PTP-LP und PTP-H (die Y-Achse ist logarithmisch dargestellt). 111
6.1 Zeiten am Master $T(t)$ und Slave $C(t)$ mit Zeitstempeln (der Einfachheit halber zeigt diese Abbildung die Zeitstempel von nur einer Synchronisationsperiode) 118
6.2 Master-Zeit $T(t)$, Slave-Zeit $C(t)$ und temperaturkompensierte Slave-Zeit $C'(t)$ sowie Offsets zwischen den Zeitfunktionen. 123
6.3 Die in der Evaluation verwendete Korrelation zwischen Temperatur und Frequenz, entsprechend eines 32KHz Oszillator vom Typ SE2412CT-ND auf dem Mica2 Sensor-Board. 125
6.4 Beispiel für einen in der Evaluation verwendeten Temperaturverlauf (rechts). Aus diesem ergibt sich zusammen mit der Korrelation zwischen Temperatur und Frequenz (vgl. Abb. 6.3) der Skew-Verlauf (links). 126
6.5 Histogramm der Verzögerung für eine Link-Auslastung von 90%. Die Verzögerung ist normiert auf eine Taktschrittweite (engl. Clock Tick) T_{tick} von 0,001s. 127
6.6 Ergebnisse für Random-Walk-Taktmodell. Die oberen Diagramme sind für eine mittlere Link-Auslastung von 10%, die unteren für 90%. 128
6.7 Ergebnisse für Temperaturtaktmodell. Die oberen Diagramme sind für eine mittlere Link-Auslastung von 10%, die unteren für 90%. 131
6.8 Berechnungszeit für verschiedene Ansätze. 132

Tabellenverzeichnis

2.1 Beispiel der Koeffizienten für verschiedene α. 20
3.1 Übersicht über Protokolle zur Zeitsynchronisation. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (MW = Mittelwert, RTT = Round Trip Time, PLL = Phase-Locked Loop, HW = Hardware, ppm = Parts Per Million) 45
3.2 Übersicht über Modifikationen von PTP oder gPTP. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (KF = Kalman-Filter, NA = Keine Angabe, PLL = Phase Locked Loop, SAGE = Space Altering Generalized Expectation-Maximization, RTT = Round Trip Time, PI = Proportional–Integral (Regler), ISR = Interrupt Service Routine, HW = Hardware, SW = Software, CHW = Custom Hardware) 48
3.3 Übersicht über Forschungsansätze, die vorwiegend drahtgebundene Szenarien untersuchen. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (NA = Keine Angabe, KF = Kalman-Filter, EF = exponentielle Filterung, KH = konvexe Hüllen, ADT = Averaged Time Differences, DSR = Direct-Skew Removal, SLW = Sliding-Window, MW = Mittelwert, TS = Zeitstempel, GT = Gaussian Traffic, ST = Self-Similar Traffic, RTT = Round Trip Time, LP = Lineare Optimierung), PI = Proportional–Integral (Regler), SW = Software, HW = Hardware, CHW = Custom Hardware, SVM = Support Vector Machine, TB = Testbed) 52
3.4 Übersicht über Forschungsansätze, die vorwiegend drahtlose Szenarien untersuchen. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen.(NA = Keine Angabe, CA = Consensus Algorithm, MSR = Mean Subsequence Reduced, LCM = Least Common Multiple, EKF = Extended Kalman Filter, CKF = Custom Kalman Filter, ISR = Interrupt Service Routine, RTT = Round Trip Time, LR = Lineare Regression, DS = De-facto-Standard, KF = Kalman-Filter, TB = Testbed, SW = Software, HW = Hardware, TS = Zeitstempel, MW = Mittelwert) 57
3.5 Übersicht über Forschungsansätze, die insbesondere die Taktstabilisierung untersuchen. Gezeigt wird das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (NA = Keine Angabe, LR = lineare Regression, DSC = Direct Skew Compensation, IMM-KF = Interacting-Multiple-Model-Kalman-Filter, RMSE = Root-Mean-Square Error, KF = Kalman-Filter, PI = Proportional–Integral (Regler)) 61
4.1 Übersicht aller Ergebnisse 88
5.1 Konzeptionelle Unterschiede zwischen PTP-LP und PSPI-Sync 92
5.2 In dieser Auswertung verwendete Taktparameter 103
6.1 Konzeptionelle Unterschiede zwischen SLMT, PTP-LP und PSPI-Sync 113
6.2 Übersicht der Ansätze für die Evaluation mittels Random-Walk-Taktmodell . . 127
6.3 Übersicht der Ansätze für die Evaluation mittels Temperaturtaktmodell 130

Abkürzungsverzeichnis

Abb.	Abbildung
AP	(WLAN) Access Point
ATD	Averaged Time Differences
AVB	Audio/Video Bridging
BMCA	Best Master Clock Algorithm
bspw.	beispielsweise
CCS	Consensus Clock Synchronisation
CCT	Coordinated Cluster Time
CERN	European Organization for Nuclear Research
CESP	Coefficient Exchange Synchronization Protocol
CKF	Custom Kalman Filter
CMA	Cumulative Moving Average
CPU	Central Processing Unit
CWA	Confidence Weighted Average
DC	Distributed Clock Synchronization
DS	De-facto-Standard
DSR	Direct Skew Removal
DTP	Datacenter Time Protocol
EACS	Environment-Aware Clock Synchronization
EF	Exponential Filtering
EKF	Extended Kalman Filter
EM	Expectation Maximization (Algorithm)
FPGA	Field Programmable Gate Array
FTM	Fault-Tolerant Midpoint
FTSP	Flooding Time Synchronization Protocol
FWA	Forwards Weighted Average
GD	Gradient Descent (Algorithm)
gPTP	Generalized Precision Time Protocol
GT	Gaussian (Network) Traffic
GTSP	Gradient Time Synchronization Protocol
HTTP	Hypertext Transfer Protocol
HW	Hardware
HW-TS	Hardware-Zeitstempel
i.A.	im Allgemeinen
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things

IMM	Interacting-Multiple-Model-Kalman-Filter
IP	Internet Protocol
IP-Core	Intellectual Property Core
ISO	International Organization for Standardization
ISR	Interrupt Service Routine
ITU	International Telegraph Union
ITU-T	ITU Telecommunication Standardization Sector
JVM	Java Virtual Machine
KF	Kalman-Filter
KH	Konvexe Hüllen
KI	künstlichen Intelligenz
LCM	Least Common Multiple
LGS	lineares Gleichungssystem
LP	Lineare Optimierung
LR	Lineare Regression
MA	Moving Average
MAC	Medium Access Control (Layer)
MII	Media Independent Interface
ML	Machine Learning
MSR	Mean Subsequence Reduced
MW	Mittelwert
NIC	Network Interface Controller
NTP	Network Time Protocol
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PI	Proportional–Integral (Regler)
PLL	Phase-Locked Loop
ppm	Parts Per Million
PSPI-Sync	Precise, Scalable, and Platform Independent Clock Synchronization
PTCP	Precision Transparent Clock Protocol
PTP	Precision Time Protocol
PTP-LP	Precision Time Protocol with Linear Programming
RBIS	Reference Broadcast Infrastructure Synchronization
RBS	Reference-Broadcast Synchronization
RMSE	Root Mean Square Error
RSS	Received Signal Strength
RTSP	Recursive Time Synchronization Protocol
RTT	Round-Trip Time

SAGE	Space Altering Generalized Expectation-Maximization (Algorithm)
SCP	Secure Copy Protocol
SDN	Software-Defined Networking
SLE	System of Linear Equations
SLMT	Synchronization Using Linear Programming, Multicasts, and Temperature Compensation
SLW	Sliding Window
SoC	System on a Chip
SOTA	State of the Art
SPiRT	Synchronization through Piggybacked Reference Timestamps
SSH	Secure Shell
ST	Self-Similar (Network) Traffic
SVM	Support Vector Machine
SW	Software
SyncE	Synchronous Ethernet
TACSC	Temperature-Assisted Clock Self-Calibration
TB	Testbed
TCP	Transmission Control Protocol
TCTS	Temperature Compensated Time Synchronization
TCXO	Temperature Compensated Oscillator
TDMA	Time Division Multiple Access
TKDS	Temperature-compensated Kalman-based Distributed Synchronization
TMAC	Tri-Mode Ethernet Media Access Controller
TPSN	Timing-sync Protocol for Sensor Networks
TSC	Time Stamp Counter
TSCH	Time Slotted Channel Hopping (Networks)
TSN	Time-Sensitive Networking
TTE	Trigger-Time-Event System
UDP	User Datagram Protocol
VANETs	Vehicular Ad-Hoc Networks
VCO	Voltage Controlled Oscillator
W7-X	Wendelstein 7-X
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
z.B.	zum Beispiel

1 Einleitung

Diese Forschungsarbeit befasst sich mit dem Thema der Zeitsynchronisation, welche der Bereitstellung einer gemeinsamen Zeitbasis für mehrere Geräte dient und demzufolge wesentliche Voraussetzung für koordinierte Aktivitäten in einem verteilten System ist.

Bzgl. der Referenznotation sei angemerkt, dass das Literaturverzeichnis in die drei Abschnitte A, B und C unterteilt wurde. Referenzen auf fremde Arbeiten bestehen aus dem Buchstaben A und einer Zahl, während Referenzen auf eigene Veröffentlichungen mit dem Buchstaben B beginnen. Verweise auf betreute studentische Arbeiten beginnen mit dem Buchstaben C.

1.1 Motivation

Im Folgenden seien, zur Motivation dieser Arbeit, einige Anwendungen der Zeitsynchronisation beschrieben.

Im Bereich des IIoT (Industrial Internet of Things) finden sich viele Anwendungen der Zeitsynchronisation. In einer Smart Factory müssen z.B. die an einer Fertigungsstrecke zusammenwirkenden Fertigungsroboter miteinander synchronisiert sein. Exakte Zeitsynchronisation ist auch wichtig, wenn mehrere Motoren gemeinsam eine mechanische Last bewegen [A 1, 2], bzw. allgemein beim Zusammenwirken von Antrieben [A 3]. Im Bereich der Energieversorgung ist Zeitsynchronisation sowohl bei der Analyse von Blackouts als auch für die Steuerung der Stabilität des Stromnetzes von entscheidender Bedeutung [A 4].

Außerdem ist die Zeitsynchronisation essentiell für die Kommunikation mittels TDMA (Time Division Multiple Access). TDMA wird z.B. in Echtzeitnetzen eingesetzt. Jedes Gerät bekommt hier einen exklusiven Sendezeitschlitz. Durch die Einhaltung dieser Zeitschlitze werden Kollisionen verhindert. Je besser die Geräte synchronisiert sind, desto genauer können die Zeitschlitze eingehalten und desto besser kann die vorhandene Bandbreite ausgenutzt werden [A 5]. Sind die Geräte hingegen ungenau synchronisiert, müssen Zeitpuffer eingefügt werden um zu garantieren, dass auch wirklich immer nur ein Gerät sendet, was zu ungenutzter Bandbreite führt. TDMA-Verfahren werden in der Praxis in vielen Echtzeit-Ethernet-Protokollen eingesetzt wie z.B. Ethernet Powerlink und SERCOS III [A 6]. Auch die drahtlose Echtzeitkommunikation basiert meist auf TDMA [A 7]. Neben Echtzeitnetzen wird TMDA aber auch bei WSNs (Wireless Sensor Networks) eingesetzt. Hier kommunizieren die Sensorknoten z.B. nur für eine kurze Zeit und verbleiben ansonsten zum Energiesparen in einem Ruhemodus [A 8, 9]. Als weiteres Beispiel ist die Beobachtung der Stabilität von Brücken, Stadien und Hochhäusern mittels drahtloser Sensoren zu nennen. Hier ist eine Synchronisationsgenauigkeit von mindestens $120\ \mu$ gefordert, um Schwingungen detektieren zu können [A 10].

Weiterhin ist als Anwendung der Zeitsynchronisation die verteilte Datenerfassung zu nennen. Als Beispiel sei auf das Kernfusionsexperiment W7-X (Wendelstein 7-X) verwiesen. Beim W7-X gibt es eine relativ große räumliche Ausdehnung des Experimentes, denn der Durchmesser des plasmaführenden Vakuumgefäßes beträgt $16\ m$. Vor allem aber handelt es sich um ein hochdynamisches System (magnetisch eingeschlossenes Plasma). Sowohl

die Steuerung des Experiments als auch die wissenschaftliche Auswertung der Experimentdurchläufe erfordern ein Zeitstempeln der aufgenommenen Daten mit Genauigkeiten im Bereich von Nanosekunden [A 11]. Gemessene Parameter sind hier z.B. Temperatur, Elektronendichte und Plasmabeschaffenheit [A 12]. Diese Anforderungen gelten auch für ähnliche Großexperimente wie die Teilchenbeschleuniger am CERN (European Organization for Nuclear Research) [A 13]. Ein weiteres Beispiel für die verteilte Datenerfassung als Anwendung der Zeitsynchronisation ist die Zusammenschaltung mehrerer Radioteleskope, um eine deutlich höhere Auflösung zu erreichen. Hierbei nehmen mehrere Teleskope, die z.B. über verschiedene Kontinente verteilt sind, Messdaten auf. Diese Daten müssen mit genauen Zeitstempeln versehen werden, um sie später fusionieren zu können. Mit dem auf diesem Ansatz basierenden Event Horizon Telescope konnte z.B. zum ersten Mal ein Bild der direkten Umgebung eines Schwarzen Loches aufgenommen werden [A 14].

Zeitsynchronisation ermöglicht ganz allgemein die Sicherstellung der Reihenfolge von Ereignissen [A 5]. Daher ist sie auch essentiell für die Finanzwelt und den Onlinehandel. Hier ist insbesondere die Reihenfolge von Transaktionen entscheidend [A 5]. Auch für verteilte Datenbanken ist Zeitsynchronisation wichtig, zur Sicherstellung der Konsistenz sowie Optimierung von Durchsatz und Latenz [A 5].

Aufgrund dieser Vielzahl von Anwendungen ist Zeitsynchronisation ein weitläufiges Forschungsgebiet und Thema zahlreicher Standardisierungsbestrebungen. Die bekanntesten Protokolle sind NTP (Network Time Protocol), PTP (Precision Time Protocol) und gPTP (Generalized Precision Time Protocol).

1.2 Problemstellung und Zielsetzungen der Arbeit

Häufig werden bei der Zeitsynchronisation die Slave-Geräte mit einer Master- bzw. Referenzzeit synchronisiert. Insbesondere für WSNs existieren aber auch komplett verteilte Ansätze, bei denen z.B. alle Geräte eine gemeinsame Zeitbasis finden.

Die größten Herausforderungen der Zeitsynchronisation sind, dass Taktgeber und Kommunikationskanäle in der Realität nicht ideal sind. Hierdurch entstehen Ungenauigkeiten, die es zu kompensieren gilt. Für Taktgeber sind vor allem Quantisierung, Frequenzänderungen (z. B. aufgrund von Temperatureffekten [A 15, 5]), zufällige Variationen bei jedem Tick (engl. Jitter), zufälliger Frequenzgang (engl. Wander) und Alterungseffekte (über lange Zeiträume) zu nennen. Für Kommunikationskanäle sind insbesondere Variationen der Netzwerkverzögerung (z. B. aufgrund von Warteschlangen) problematisch, aber auch variable Verarbeitungszeiten, insbesondere bei der Verarbeitung in Software.

Folgende Anforderungen an einen Zeitsynchronisationsansatz können definiert werden:

- Genauigkeit: Die verbleibende Zeitdifferenz zwischen Master und Slave, welche durch die Synchronisation nicht ausgeglichen werden kann, sollte möglichst klein sein.
- Bandbreiteneffizienz: Die Synchronisationsnachrichten erzeugen immer einen Overhead. Je weniger der verfügbaren Bandbreite für die Synchronisation verwendet wird, desto mehr Bandbreite bleibt für die eigentlichen Aufgaben des verteilten Systems übrig (Steuerung, Datenerfassung, Datenaustausch etc.).

- Rechenleistungseffizienz: Die Zeitsynchronisation benötigt zusätzliche Rechenleistung auf den Endgeräten. Je weniger zusätzliche Rechenleistung für die Synchronisation benötigt wird, desto mehr steht für die eigentlichen Aufgaben der Geräte zur Verfügung.
- Hardware-Anforderungen: Weiterhin sind die Anforderungen an die Hardware entscheidend. Dies gilt sowohl für die Endgeräte (NICs, Networks Interface Controller) als auch für die Netzwerkinfrastruktur (z.B. Switches). Geringere Anforderungen an die Hardware führen zu geringen Installationskosten und höherer Zukunftssicherheit.

Hierbei müssen die einzelnen Anforderungen immer gegeneinander abgewogen werden. Beispielsweise ist die Genauigkeit bei WSN-Ansätzen von geringerer Bedeutung als die Energieeffizienz [A 9]. Im Gegensatz dazu werden bei drahtgebundenen Echtzeitnetzen mitunter ein hoher Energiebedarf und Spezial-Hardware zum Erreichen einer hohen Genauigkeit in Kauf genommen [A 11, 13].

Das Ziel dieser Arbeit ist die Untersuchung algorithmischer Verfahren zur genauen Zeitsynchronisation in drahtgebundenen Netzen. Die Bandbreiteneffizienz soll betrachtet werden, da sie insbesondere in großen Netzen eine Bedeutung hat. Auch die Rechenleistungseffizienz soll untersucht werden. Diese hat aber eine geringere Bedeutung, da in drahtgebundenen Szenarien typischerweise alle Geräte über eine Stromversorgung verfügen und Energieeffizienz somit weniger wichtig ist. Eine besondere Bedeutung sollen die Hardware-Anforderungen haben. Einerseits wurden die Vorteile geringer Hardware-Anforderungen bereits beschrieben. Andererseits ist es in bestimmten Szenarien, z.B. bei der Synchronisation über das Internet, gar nicht möglich, die Netzwerkinfrastruktur zu beeinflussen. Aus diesem Grund soll insbesondere keine spezialisierte Netzwerkinfrastruktur zum Einsatz kommen. Spezialisierte NICs in den Endgeräten hingegen sind vertretbar. Wenn z.B. nur einige Geräte höchste Anforderungen an die Synchronisationsgenauigkeit stellen, dann müssen auch nur diese mit Spezial-Hardware ausgestattet sein.

1.3 Aufbau der Arbeit

Abb. 1.1 bietet eine Übersicht über die Kapitelstruktur dieser Forschungsarbeit, wobei die wesentlichen eigenen Beiträge fett hervorgehoben sind.

Diese Arbeit ist wie folgt strukturiert. In Kapitel 2 werden die zum Verständnis dieser Arbeit benötigten Grundlagen vorgestellt. Kapitel 3 gibt einen Überblick über den Stand der Technik im Bereich der Zeitsynchronisation. In Kapitel 4 wird der Ansatz PSPI-Sync vorgestellt. Hierbei steht PSPI-Sync für **P**recise, **S**calable, and **P**latform **I**ndependent Clock **Sync**hronization. Wichtigstes Element von PSPI-Sync ist die Verwendung von Multicasts für die Zeitsynchronisation. Kapitel 5 stellt den PTP-LP-Ansatz vor, der den PTP-Standard um lineare Optimierung (LP für engl. Linear Programming) erweitert. Durch die Verwendung von LP ist PTP-LP sehr robust gegenüber Schwankungen der Verzögerungszeit. In Kapitel 6 wird der SLMT-Ansatz untersucht. Dieser basiert zwar auf den Überlegungen und Erkenntnissen von PTP-LP, ist aber ein unabhängiger Ansatz. Hierbei steht SLMT für **S**ynchronisation mittels **l**inearer Optimierung, **M**ulticasts und **T**emperaturkompensation. SLMT verbindet also LP mit der Synchronisation auf Basis von Multicasts. Als weiteres Element verwendet SLMT die Temperaturkompensation zur Linearisierung des Taktes, um die

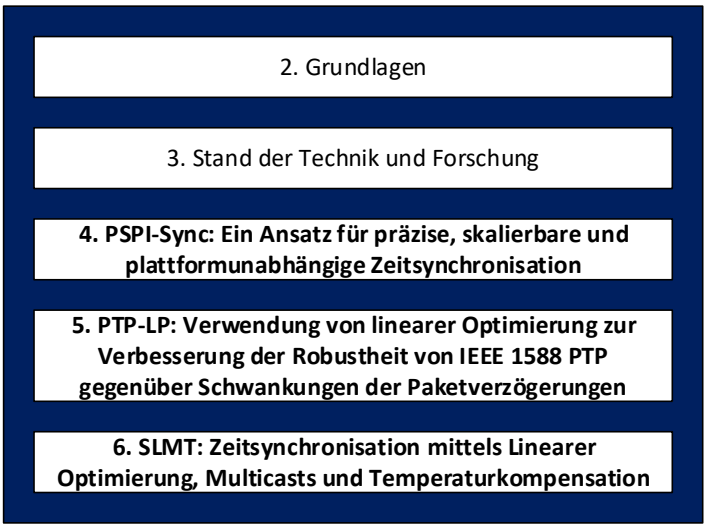


Abbildung 1.1: Übersicht der Kapitelstruktur dieser Forschungsarbeit. Die wesentlichen eigenen Beiträge sind fett hervorgehoben.

Genauigkeit der Synchronisation auf Basis von LP weiter zu verbessern. Den Abschluss dieser Arbeit bildet Kapitel 7 mit einer Zusammenfassung und einem kurzen Ausblick.

2 Grundlagen

2.1 Netzwerke: Referenzmodelle und Protokolle

In diesem Abschnitt erfolgt eine grundlegende Beschreibung der Architektur von Computernetzwerken anhand des ISO/OSI-Referenzmodells (hierbei steht ISO/OSI für: International Organization for Standardization/ Open Systems Interconnection) und Beispielen aus der Familie des Internet-Protokoll-Stacks.

Die historisch mit dem ISO/OSI-Referenzmodell verbundenen Protokolle finden zwar in aktuellen Computernetzen keine Verwendung mehr, dafür definiert es aber abstrakte Schichten und Funktionen. Beide sind immer noch sehr relevant. Jede Schicht beschreibt eine konkrete Funktion und entspricht einem Abstraktionslevel. Ein Ziel des ISO/OSI-Referenzmodells ist es, den Informationsfluss über Schichtgrenzen hinweg möglichst gering zu halten [A 16].

Weiterhin existiert das TCP/IP-Referenzmodell (hierbei steht TCP/IP für: Transmisson Control Protokoll/ Internet Protocol). Dieses ist jünger und orientiert sich eher an bestehenden Netzwerkprotokollen. Das TCP/IP-Modell selbst wird nicht mehr viel genutzt, dennoch finden die zugehörigen Protokolle ihre Anwendung sowohl im Internet als auch in dessen Vorgänger, dem ARPANET (Advanced Research Projects Agency Network).

Bei der Entwicklung des ARPANETs wurden folgende Anforderungen gestellt. Ein Hardwareausfall von Subnetzen soll kompensiert werden können und die Verbindung so lange bestehen bleiben, wie Kommunikationsquelle und -senke funktionieren. Es sollte eine flexible Architektur für verschiedenste Anwendungen geschaffen werden (z.B. Dateiübertragung und Echtzeitsprachübertragung) [A 16].

Im Folgenden werden die Schichten des ISO/OSI-Referenzmodells erläutert und jeweils Beispiele aus dem Internet-Protokoll-Stack genannt (vgl. Abb. 2.1).

Bitübertragungsschicht (Physical Layer): Auf der Bitübertragungsschicht wird spezifiziert, wie ein Bit übertragen werden soll. Dies beinhaltet die Definition des physikalischen Übertragungsmediums sowie der mechanischen, elektrischen und zeitlichen Schnittstellen. Wie in Abb. 2.1 gezeigt, besteht nur auf der Bitübertragungsschicht eine physikalische Verbindung zwischen den Netzwerkknoten und auf den höheren Schichten nur noch eine logische [A 16].

Sicherungsschicht (Data Link Layer): Auf dieser Schicht findet die Korrektur von Fehlern statt, die ggf. auf der Bitübertragungsschicht entstanden sind. In Broadcast-Netzen werden auch Medium-Access-Control-(MAC-)Funktionen auf der Sicherungsschicht umgesetzt. Als Beispielprotokolle sind Ethernet und WLAN zu nennen, wobei beide Protokolle sowohl die Bitübertragungs- als auch die Sicherungsschicht spezifizieren [A 16].

Vermittlungsschicht (Network Layer): Hier findet die Auswahl der Paketrouten von der Quelle zum Ziel statt. Dies kann sowohl statisch als auch dynamisch geschehen, wobei letzteres die dynamische Reaktion auf Lastsituation im Netzwerk ermöglicht. Zusammen mit höheren Schichten ist dann das „Abfedern“ von Überlastsituationen möglich. Zu den weiteren Aufgaben der Vermittlungsschicht gehört die Sicherung von Qualität (z.B. bzgl. der Verzögerung bzw. Übertragungszeit und des Jitters). Außerdem ist in heterogenen Netzen

die Adressübersetzung eine wichtige Aufgabe dieser Schicht. Das typische Beispiel für ein Protokoll auf der Vermittlungsschicht ist IP (Internet Protocol), das sowohl das Paketformat als auch das Protokoll selbst definiert. Darüber hinaus enthält IP eine Überlastkontrolle, die sich in der Praxis jedoch als wenig effizient herausgestellt hat. Zusätzlich wird auch das Internet Control Message Protocol (ICMP) definiert. Bis zur Vermittlungsschicht spricht man auch von verketteten Schichten, da jeder Netzwerkknoten mit seinem Nachbarn kommuniziert (vgl. Abb. 2.1) [A 16].

Transportschicht (Transport Layer): Ab der Transportschicht spricht man von Ende-zu-Ende-Schichten, da eine direkte Kommunikation zwischen Quelle und Senke stattfindet (vgl. Abb. 2.1). Das dazwischenliegende Netzwerk ist für beide vollkommen transparent. Aufgaben dieser Schicht sind Annahme und Übertragung von Daten, sowie die Sicherstellung, dass diese korrekt ankommen. Eine wichtige Anforderung an die Transportschicht ist die Abkopplung von den unteren Schichten, um z.B. gegenüber Änderungen der Hardwaretechnik kompatibel zu sein. Die Transportschicht bietet unterschiedliche Dienste für die höheren Schichten an. Die Auswahl zwischen den Diensten findet beim Verbindungsaufbau statt. Einerseits wird als Dienst ein „fehlerfreier“ Punkt-zu-Punkt-Kanal angeboten, inkl. Sicherstellung der Sendereihenfolge [2.1]. Ein typisches Beispiele ist TCP, das zuverlässig sowie verbindungsorientiert arbeitet und eine Fluss- und Überlastkontrolle bietet. Andererseits ist als Dienst auch der Transport von Nachrichten ohne Gewähr möglich, sowie das Senden an viele Empfänger (Broadcast). Hier ist der typische Vertreter UDP (User Datagram Protocol), das verbindungslos, ohne Reihenfolgegarantie und ohne Absicherung arbeitet. Trotzdem wird UDP von vielen Anwendungen verwendet, denn ggf. werden diese Sicherheiten von den höheren Schichten gar nicht benötigt. Dies kann vorkommen, wenn z.B. Geschwindigkeit wichtiger ist als Genauigkeit, wie bei der Übertragung von Sprache und Video. Außerdem gibt es Protokolle höherer Schichten, welche diese Sicherungsfunktionen lieber selbst umsetzen wie das Protokoll QUIC (Quick UDP Internet Connections) [A 16].

Sitzungsschicht (Session Layer): Aufgabe der Sitzungsschicht ist der Aufbau von Sitzungen und die Dialogsteuerung (Dialog Control). Die Dialogsteuerung definiert, wer wann Daten übertragen darf. Eine weitere Aufgabe ist die Token-Verwaltung (Token Management). Diese stellt sicher, dass eine wichtige Operation nicht von mehreren Teilnehmern gleichzeitig ausgeführt wird. Weiterhin findet eine Synchronisation statt. Bei dieser werden u.A. Fixpunkte bei langen Übertragungen für die Wiederherstellung nach einem Verbindungsabbruch angelegt [A 16].

Darstellungsschicht (Presentation Layer): Die Darstellungsschicht definiert sowohl Syntax als auch Semantik der zu übertragenen Informationen. So werden abstrakte Datenstrukturen für die Übertragung und Standardkodierung definiert [A 16].

Anwendungsschicht (Application Layer): Auf der Anwendungsschicht finden sich all jene Protokolle, die von Nutzern benötigt werden. Dies beinhaltet z.B. HTTP (Hypertext Transfer Protocol) für Websites und Browser aber auch Protokolle für E-Mails und Netznachrichten. Aktuell zeigt sich eher, dass eine noch feinere Unterteilung der Anwendungsschicht stattfindet. Es hat sich auch gezeigt, dass Protokolle der Anwendungsschicht häufig Sitzungs-

[2.1] Ein Kanal kann natürlich niemals komplett frei von Übertragungsfehlern sein aber es handelt sich in diesem Fall um einen sehr robusten Kanal, der als fehlerfrei angenommen werden kann.

und Darstellungsfunktionen bei Bedarf mit einbinden und hierfür keine speziellen Protokolle Verwendung finden. Typische Vertreter für Protokolle auf der Anwendungs-, Darstellungs- und Sitzungsschicht sind neben HTTP auch Telnet (Teletype Network), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol) und RTP (Real-Time Transport Protocol) [A 16].

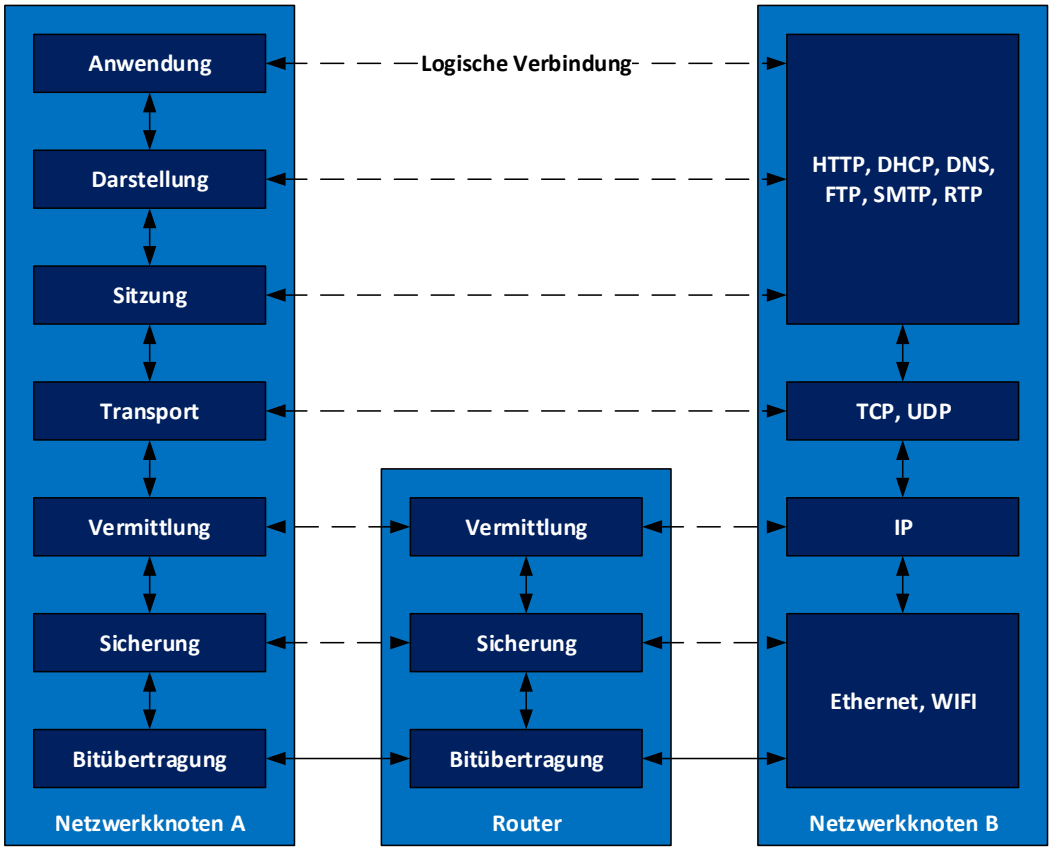


Abbildung 2.1: Kommunikation zwischen zwei Netzwerkknoten, die über einen Router verbunden sind. Darstellung der Schichten mit ISO/OSI- Referenzmodell (Netzwerkknoten A, Router) und Beispielprotokollen aus dem Internet-Protokoll-Stack (Netzwerkknoten B) [A 16, 17, 18]

2.2 Zeitsynchronisation

Im folgenden Abschnitt wird auf die Grundlagen der Zeitsynchronisation in Netzwerken eingegangen.

2.2.1 Beschreibung des Problems der Zeitsynchronisation

Ziel der Zeitsynchronisation ist immer die Schaffung einer einheitlichen Zeitbasis. Diese Zeitbasis wird auch als Master- oder Referenzzeit bezeichnet. Eine solche ist immanent für

koordinierte Aktivitäten in verteilten Systemen. Konkrete Anwendungen wurden ja bereits in Kapitel 1 erläutert.

Zeitsynchronisationsprotokolle basieren fast immer auf dem Austausch von Zeitstempeln zwischen Master und Slave [2.2]. Als Ergebnis der Synchronisation findet entweder eine Anpassung der Slave-Zeit an die Master-Zeit statt oder der Slave merkt sich den Zeitunterschied zum Master, ohne seine eigene Zeit zu korrigieren. Letzteres dient z.B. der Sicherstellung der Monotonie der Slave-Zeit und somit der Vermeidung von Rücksprüngen [A 20]. Rücksprünge können dazu führen, dass Ereignisse am Slave nicht mehr korrekt in eine zeitliche Reihenfolge gebracht werden können und müssen daher in vielen Szenarien vermieden werden.

Als Synchronisationsgenauigkeit wird die verbleibende, nicht durch die Zeitsynchronisation kompensierbare Zeitdifferenz zwischen Referenz und Slave bezeichnet. Es wurde gezeigt, dass bei idealen Bedingungen (konstanten Taktfrequenzen und konstante, symmetrische Verzögerungen zwischen den Netzwerkteilnehmern) perfekte Zeitsynchronisation möglich ist [A 21].

Da reale Computernetze aber niemals ideal sind, hat die Zeitsynchronisation hauptsächlich zwei Herausforderungen: die Kompensation von Ungenauigkeiten durch nichtideales Verhalten der Slave-Zeit selbst sowie im Kommunikationskanal zwischen den Knoten. Die größten Fehlerquellen der Slave-Zeit sind Quantisierung und Frequenzänderungen. Bei Frequenzänderungen lässt sich unterscheiden zwischen Jitter (zufällige Variation bei jedem Tick), Wander (zufälliger Frequenzgang), Temperatureffekten (in größeren Zeiträumen) und Alterungseffekten (in noch längeren Zeiträumen). Die Fehlerquellen des Kommunikationskanals sind einerseits Ausbreitungsverzögerungen, insbesondere aber variable Netzwerkverzögerungen. Letztere sind hochrelevant für die Synchronisationsgenauigkeit, wie z.B. in [A 22] gezeigt, und entstehen durch Verzögerungen in den Queues der Switches sowie durch den etwaigen Verlust von Nachrichten und deren Neuübertragung. Beide Effekte treten insbesondere als Folge von Überlastsituationen auf.

2.2.2 Oszillatoren und Taktgeber

Als Oszillator wird eine Schaltung bezeichnet, die eine ungedämpfte Schwingung bestimmter Frequenz erzeugt. Es gibt unterschiedliche Arten von Oszillatoren, wobei als Taktgeber meist Quarz-Oszillatoren und keine LC-Oszillatoren verwendet werden (LC, da die Schaltung auf einem Schwingkreis mit Induktivität und Kapazität basiert). Daher behandelt dieser Abschnitt ausschließlich Quarz-Oszillatoren. Bei diesen lässt sich zwischen zwei verschiedenen Anforderungen unterscheiden [A 23].

Taktoszillatoren dienen zur Takterzeugung für digitale Schaltungen. Sie sind i.A. freilaufend, d.h. ohne phasenstarre Kopplung an eine Referenzfrequenz. Die Frequenz der ausgegebenen Schwingung ist mehr oder weniger genau, denn für die meisten digitalen Schaltungen ist es irrelevant, ob sie z.B. bei einer Frequenz von 1 GHz oder 1.01 GHz betrieben wer-

[2.2] Es gibt nur wenige Ausnahmen, wie z.B. die in [A 19] vorgestellten Ansätze, die Geräte auf Basis der 50Hz Schwingung im Stromnetz oder anhand der Lichtfrequenzen synchronisieren.

den [2.3]. Daher findet auch i.A. keine schaltungstechnische Temperaturkompensation statt. Außerdem genügen einfache Schaltungen, da geringe Anforderungen an das Rauschen gestellt werden. In der Praxis ist häufig die Oszillatorschaltung bereits in den Baustein integriert und nur noch der gewünschte Quarz muss angeschlossen werden [A 23].

Ganz anders sehen die Anforderungen bei **Referenzoszillatoren** aus. Hier wird eine sehr genaue Frequenz benötigt. Beispiele sind der TCXO (temperatur compensated crystal oscillator) oder der OCXO (oven-controlled crystal oscillator). Während bei TCXOs die Temperaturabhängigkeiten schaltungstechnisch kompensiert werden, enthalten OCXOs ein Heizelement, welches den Oszillator auf einer konstanten Temperatur hält. Referenzoszillatoren können freilaufend sein oder an ein Frequenznormal gekoppelt. In hochwertigen Messgeräten werden z.B. OCXOs verwendet und zusätzlich die Kopplung an eine externe Referenz ermöglicht. Somit können mehrere Messgeräte über die selbe Referenz für phasenstarr gekoppelt werden [A 23].

2.2.3 Taktmodelle

Nachdem im vorherigen Abschnitt kurz auf die schaltungstechnische Umsetzung von Taktgebern eingegangen wurde, wird in diesem Abschnitt auf die Modellierung der Taktgeber und insbesondere der resultierenden Master-Zeitfunktion $T(t)$ und Slave-Zeitfunktion $C(t)$ eingegangen. Zunächst lässt sich die Spannung V(t) am Ausgang einer Oszillatorschaltung wie folgt beschrieben [A 24]:

$$V(t) = V_0 \cdot sin(\Phi(t)) \tag{1}$$

Hierbei sei $\Phi(t)$ die Phase mit $\Phi(t = 0) = 0$. Wenn Schwankungen $\epsilon(t)$ der Amplitude V_0 mit betrachtet werden, dann ergibt sich folgender Zusammenhang [A 25] (nach [A 24] sind Schwankungen von V_0 aber i.A. vernachlässigbar):

$$V(t) = (V_0 + \epsilon(t)) \cdot sin(\Phi(t)) \tag{2}$$

Die Frequenz f(t) sei wie folgt definiert:

$$f(t) = (1/2\pi)d\Phi/dt \tag{3}$$

Zur Messung dieser Frequenz wäre eine unendlich große Bandbreite nötig, daher wird die entsprechende Formel umgeschrieben. Hierbei wird eine konstante nominale Frequenz f_0 angenommen und die restliche Abweichung als $\phi(t)$ modelliert [A 24, 26]:

$$V(t) = V_0 \cdot sin(2\pi f_0 t + \phi(t)). \tag{4}$$

[2.3] Hier gibt es natürlich eine Ausnahme, nämlich wenn Daten zwischen zwei Schaltungen ausgetauscht werden sollen. Zu diesem Zweck werden bei der Datenübertragung zwischen Taktdomains auf einem Chip Dual-Clocked FIFOs verwendet. Bei der Datenübertragung zwischen Geräten über Bussysteme findet i.A. eine doppelte Überabtastung statt, um Frequenzunterschiede auszugleichen.

Hieraus wird die normalisierte Frequenzabweichung abgeleitet:

$$y(t) = (f(t) - f_0)/f_0. \tag{5}$$

Da es sich bei Gl. 4 um eine periodische Funktion handelt, entsteht bei Erfüllung folgender Gleichung eine Sequenz von Ereignissen gleicher Phase [A 26]:

$$f_0 \cdot t + \frac{\phi_0}{2\pi} = n, \qquad n \in \mathbb{N}, \phi(t) = \phi_0. \tag{6}$$

Im Folgenden sei T_n die Zeit für das n-te Ereignis. Auf einem realen Gerät entspricht dieses Ereignis z.B. einem Zähler der pro Taktzyklus inkrementiert wird, wobei ein Oszillator als Taktgeber fungiert. Die Bezeichnungen $T(t)$ bzw. $C(t)$ implizieren Kontinuität aber tatsächlich existiert bei beiden Zeitfunktionen ein endliches Inkrement, wobei das nominale Inkrement bei $1/f_0$ liegt. Das Inkrement darf nicht als konstant angenommen werden, daher wird $\phi(t)$ als Zufallsprozess modelliert mit $E[\phi(t)] = \phi_0 = 0$ ohne Beschränkung der Allgemeinheit [A 26]:

$$f_0 \cdot t + \frac{\phi(t)}{2\pi} = n, \qquad n \in \mathbb{N}. \tag{7}$$

Die Slave-Zeit könnte entsprechend berechnet werden mittels [A 26]:

$$C(t_n) = n/f_0. \tag{8}$$

Hierbei ergibt sich jedoch das Problem, dass sich die durch $\phi(t)$ entstehenden Abweichungen mit der Zeit aufsummieren. Um dem entgegenzuwirken, ist z.B. eine periodische Anpassung der Slave-Zeit nötig. Die Referenzzeit beim n–ten Event sei T_n und die Slave-Zeit $C(T_n)$. Der Offset ergibt sich dann wie folgt:

$$\theta(T_n) = C(T_n) - T_n. \tag{9}$$

Die konkret verwendeten Taktmodelle werden in den jeweiligen Kapiteln im Detail erklärt.

2.3 Grundlagen für Kalman-Filter: Zustandsraumbeschreibungen und Wahrscheinlichkeitstheorie

In diesem Abschnitt wird kurz auf die mathematischen Grundlagen von Kalman-Filtern eingegangen, die essentiell für das Verständnis der späteren Abschnitte über einfache und adaptive Kalman-Filter sind.

2.3.1 Zustandsraumbeschreibungen

Kalman-Filter arbeiten immer mit der Zustandsraum-Darstellung, daher soll zunächst kurz hierauf eingegangen werden. Viele physikalische Systeme lassen sich durch Differentialgleichungen beschreiben. Ein typisches Beispiel sind die Bewegungsgleichungen für den Zusammenhang von Weg, Geschwindigkeit und Beschleunigung. Im Allgemeinen lassen sich solche Differentialgleichungen zur Systembeschreibung in folgende Form bringen [A 27]:

$$x'(t) = A \cdot x(t) + B \cdot u(t) \tag{10}$$
$$y(t) = C \cdot x(t) + D \cdot u(t) \tag{11}$$

Hierbei sei u der Eingang des Systems und y dessen Ausgang. Der innere Zustand des Systems sei mit x betitelt und x' sei dessen Ableitung. Bei u, y und x kann es sich um Skalare oder Vektoren handeln. A, B, C und D sind Matrizen und beschreiben die Zusammenhänge zwischen u, y und x (vgl. Abb. 2.2).

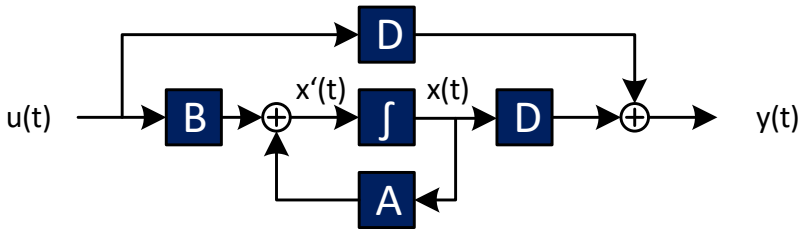


Abbildung 2.2: Zustandsraumbeschreibungen eines zeitkontinuierlichen Systems [A 27]

Dieses zeitkontinuierliche System wiederum lässt sich überführen in ein zeitdiskretes System der Form [A 27] (vgl. Abb. 2.3):

$$x(k+1) = A_d \cdot x(k) + B_d \cdot u(k) \tag{12}$$
$$y(k) = C \cdot x(k) + D \cdot u(k). \tag{13}$$

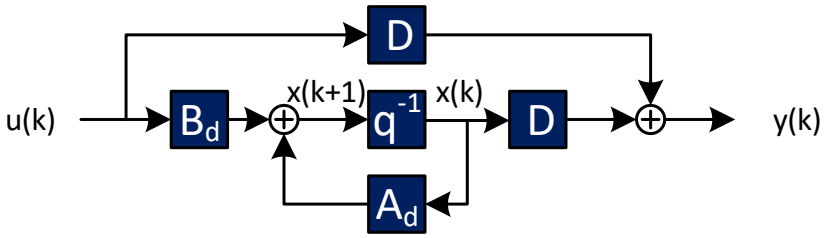


Abbildung 2.3: Zustandsraumbeschreibungen eines zeitdiskreten Systems [A 27]

Die Matrizen A_d und B_d werden hierbei wie folgt mittels Laplace-Transformation aus A und B gebildet, wobei T_s die Abtastperiode bezeichnet (die genaue Herleitung über eine Abtastung mit Dirac-Funktionen findet sich z.B. in [A 27]):

$$A_d = e^{A \cdot T_s}, B_d = \int_0^{T_s} e^{A \cdot v} \cdot B \; dv. \tag{14}$$

In realen Systemen kommen Ungenauigkeiten hinzu. Einerseits kann der Ausgang des Systems niemals ganz exakt gemessen werden, was durch das Messrauschen $v(k)$ modelliert wird. Weiterhin gibt es fast immer eine Ungenauigkeit in der Beschreibung des realen Systems, z.B. da die Bewegungsgleichungen nur für punktförmige Massen gelten und nicht für reale Objekte mit einem Volumen wie Menschen oder Fahrzeuge. Dies wird durch das Systemrauschen $z(k)$ modelliert. Ein solches System lässt sich also wie folgt beschreiben [A 27] (vgl. Abb. 2.4):

$$x(k+1) = A_d \cdot x(k) + B_d \cdot u(k) + G_d \cdot z(k) \tag{15}$$

$$y(k) = C \cdot x(k) + D \cdot u(k) + v(k) \tag{16}$$

Weiterführende Informationen finden sich z.B. in [A 28, 27]

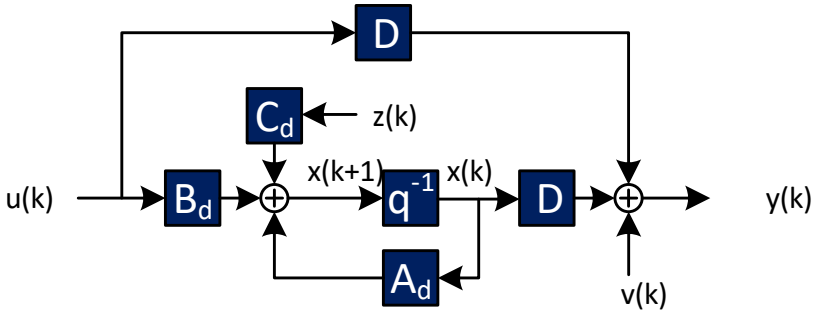


Abbildung 2.4: Zustandsraumbeschreibungen eines zeitdiskreten Systems [A 27]

2.3.2 Wahrscheinlichkeitstheorie

Die zweite wichtige Grundlage von Kalman-Filtern ist die Wahrscheinlichkeitstheorie. Diese ist aber auch relevant, um in Netzwerken stochastische Prozesse wie bspw. die Paketverzögerung zu beschreiben.

Bei einem durchzuführenden Experiment, dessen genaues Ergebnis nicht vorhersehbar ist, z.B. das Werfen eines Würfels, sei Ω die Menge aller möglichen Ergebnisse [A 27]:

$$\Omega = \{\omega_1, \omega_2, \dots, \omega_L\} \tag{17}$$

Hierbei bezeichne ω_i einen möglichen Ausgang des Experimentes. Man spricht von einem elementaren Ereignis. Das Ereignis $A \subseteq \Omega$ wiederum sei eine Menge aus Elementarereignissen [A 27]:

$$A = \{a_1, a_2, ... , a_M\}, \qquad M \leq L \tag{18}$$

Die Zufallsvariable X ordne jedem Ergebnis $\omega \in \Omega$ eine reelle Zahl x zu (z.B. die Anzahl der Augen auf dem Würfel) [A 27]:

$$X : \Omega \rightarrow \mathbb{R} \tag{19}$$

$$\omega \rightarrow X(\omega) \tag{20}$$

Hierbei ist zu beachten, dass X vor der Durchführung des Experimentes eine reelle Zahl mit unbekanntem Wert ist. Nach der Durchführung des Experimentes ergibt sich der Wert x als Realisierung von X [A 27]. Die relative Häufigkeit $h_n(A)$ des Ereignisses A lässt sich wie folgt berechnen, wobei n die Anzahl der Durchführungen des Experimentes bezeichne [A 27]:

$$h_n(A) = \frac{Anzahl\ der\ Experimente\ mit\ Ereignis\ A}{n} \tag{21}$$

Die Wahrscheinlichkeit $P(A)$ des Ereignisses A entspricht der relativen Häufigkeit, wenn das Experiment unendlich oft durchgeführt werden würde [A 27]:

$$P(A) = \lim_{n \rightarrow \infty} h_n(A) \tag{22}$$

Des Weiteren sei die Wahrscheinlichkeit, dass das Ereignis B eintritt unter der Voraussetzung, dass A bereits eingetreten ist, als bedingte Wahrscheinlichkeit $P(B|A)$ wie folgt definiert [A 29]:

Definition 1 (Bedingte Wahrscheinlichkeit) *Die Wahrscheinlichkeit für das Eintreten des Ereignisses B unter der Bedingung/Voraussetzung, dass das Ereignis A bereits eingetreten ist, heißt bedingte Wahrscheinlichkeit von B unter der Bedingung A und wird durch das Symbol $P(B|A)$ gekennzeichnet. Sie wird definiert durch die Gleichung*

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, \qquad P(A) \neq 0. \tag{23}$$

Hiervon ist der Multiplikationssatz abgeleitet. Die Wahrscheinlichkeit für das gleichzeitige Eintreten zweier Ereignisse A und B ist [A 29]:

$$P(A \cap B) = P(A) \cdot P(B|A) \tag{24}$$

In der Praxis kann es vorkommen, dass das Eintreten des Ereignisses A überhaupt nicht von B abhängt und umgekehrt. Es gilt also $P(B|A) = P(B)$ und $P(A|B) = P(A)$. In diesem Fall spricht man von stochastisch unabhängigen Ereignissen. Aus dem Multiplikationssatz folgt dann [A 29]:

Definition 2 (Stochastische Unabhängigkeit) *Zwei Ereignisse A und B heißen stochastisch unabhängig, wenn gilt:*

$$P(A \cap B) = P(A) \cdot P(B). \tag{25}$$

Zum besseren Verständnis der stochastischen Unabhängigkeit sei folgende Rechnung angebracht, aus welcher folgt, dass B nicht wahrscheinlicher durch das Eintreten von A wird:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = P(B) \tag{26}$$

$$\Leftrightarrow \frac{P(A \cap B)}{P(A)} = \frac{P(B)}{1} \tag{27}$$

Weiterhin gilt für die totale Wahrscheinlichkeit eines Ereignisses die Bayessche Formel [A 29]:

Definition 3 (Totale Wahrscheinlichkeit & Bayessche Formel) *Ein Ereignis B trete stets in Verbindung mit genau einem der sich paarweise ausschließenden Ereignisse A_i mit $i = 1, 2, ..., n$ auf, d.h. die Ereignisse A_i sind die möglichen Zwischenstationen auf dem Wege zum Ereignis B. Dann gilt für die totale Wahrscheinlichkeit des Ereignisses B:*

$$P(B) = \sum_{i=1}^{n} P(A_i) \cdot P(B|A_i) \tag{28}$$

Der Summand $P(A_i) \cdot P(B|A_i)$ ist dabei die Wahrscheinlichkeit dafür, das Ereignis B längs des Pfades OA_iB, d.h. über die Zwischenstation A_i zu erreichen. Die totale Wahrscheinlichkeit $P(B)$ für das Eintreten des Ereignisses B erhält man also aus dem Ereignisbaum (vgl. Abb. 2.5), indem man über die Wahrscheinlichkeiten aller nach B führender Pfade aufsummiert.

Unter der Voraussetzung, dass das Ereignis B bereits eingetreten ist, gilt dann für die Wahrscheinlichkeit, dass dieses Ereignis auf dem Pfad OA_jB über die Zwischenstation A_j erreicht wurde, die sog. Bayessche Formel:

$$P(A_j|B) = \frac{P(OA_jB)}{P(B)} = \frac{P(A_j) \cdot P(B|A_j)}{\sum_{i=1}^{n} P(A_i) \cdot P(B|A_i)}. \tag{29}$$

Die Häufigkeitsverteilung einer Zufallsvariablen wird mit einer Dichtefunktion beschrieben. Ein typisches Beispiel ist z.B. die Normalverteilung. Obwohl es eine Vielzahl typischer Verteilungen gibt und die Bestimmung der Verteilung für reale Zufallsprozesse nicht immer einfach bzw. möglich ist, kommt die Normalverteilung sehr häufig in der Natur vor [A 27]:

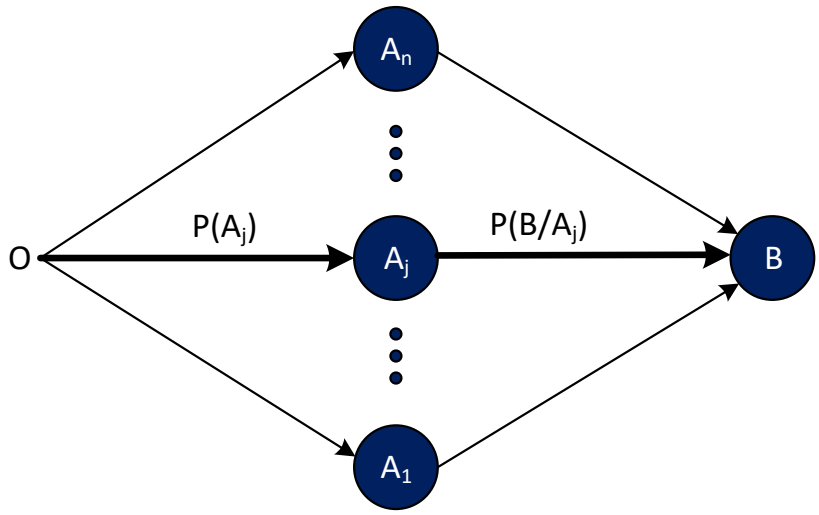


Abbildung 2.5: Das Ereignis B ist über verschiedene Zwischenstationen A_i erreichbar [A 29].

Definition 4 (Dichtefunktion Normalverteilung) *Die Dichtefunktion einer normalverteilten Zufallsvariable ist definiert durch die Funktion:*

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}}, \qquad \sigma > 0. \tag{30}$$

Hierbei bezeichne μ den Erwartungswert (Mittelwert) und σ die Standardabweichung. Die Standardabweichung ist ein Streuungsmaß für die Breite der Verteilung bzw. die Abweichung um den Erwartungswert μ (vgl. Abb. 2.6). Bei der Normalverteilung liegen im Intervall $[\mu - \sigma, \mu + \sigma]$ 68,27% aller Zufallszahlen.

In der Praxis werden für Verteilungen häufig Momente und Zentrale Momente bestimmt. Diese charakterisieren die Verteilungen unvollständig, unterschiedlichste Verteilungen können z.B. denselben Mittelwert besitzen. Das Moment i-ter Ordnung ist definiert als [A 27]:

Definition 5 (i-tes Moment einer Zufallsvariable X) *Für eine Zufallsvariable X, welche die Werte x annehmen kann, mit der Dichtefunktion f(x) gilt für das i-te Moment:*

$$\alpha_i = E(X^i) = \begin{cases} \int_{-\infty}^{+\infty} x^i \cdot f(x)\, dx, & falls\ X\ stetig\ ist \\ \sum_k x(k)^i \cdot f(x(k)), & falls\ X\ diskret\ ist \end{cases} \tag{31}$$

Von besonderer Bedeutung ist das erste Moment, der Erwartungswert [A 27]:

Definition 6 (Erwartungswert einer Zufallsvariable X) *Für eine Zufallsvariable X, welche die Werte x annehmen kann, mit der Dichtefunktion f(x) gilt für den Erwartungswert:*

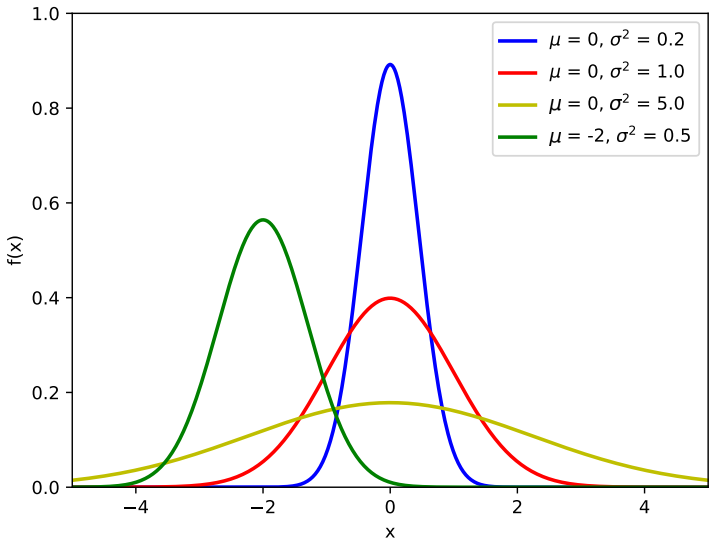


Abbildung 2.6: Dichtefunktion der Normalverteilung für verschiedene Erwartungswerte und Standardabweichungen

$$\alpha_i = E(X) = \begin{cases} \int_{-\infty}^{+\infty} x \cdot f(x)dx, & falls\ X\ stetig\ ist \\ \sum_k\ x(k) \cdot f(x(k)), & falls\ X\ diskret\ ist \end{cases} \tag{32}$$

Für eine Stichprobe bzw. Zahlenfolge ist der Mittelwert eine Näherung für den Erwartungswert.

Definition 7 (Mittelwert) *Der Mittelwert eine Zahlenfolge mit den Elementen x(k) berechnet sich durch:*

$$\alpha_i = E(X) = \left\{ \overline{x} = \frac{1}{n} \cdot \sum_{k=1}^{n} x(k) \right. \tag{33}$$

Der Mittelwert wird zum Erwartungswert für $n \to \infty$ [A 27]:

$$E(X) = \lim_{n\to\infty} \frac{1}{n} \cdot \sum_{k=1}^{n} x(k) \tag{34}$$

Außerdem sind die zentralen Momente i-ter Ordnung wie folgt definiert [A 27]:

Definition 8 (i-tes zentrales Moment einer Zufallsvariable X) *Für eine Zufallsvariable X mit der Dichtefunktion f(x) gilt für das i-te zentrale Moment:*

$$\mu_i = E([X - E(X)]^i) = \begin{cases} \int_{-\infty}^{+\infty} [X - E(X)]^i \cdot f(x)dx, & falls\ X\ stetig\ ist \\ \sum_k [x(k) - E(X)]^i \cdot f(x(k)), & falls\ X\ diskret\ ist \end{cases} \tag{35}$$

Besondere Bedeutung hat das zweite zentrale Moment, die Varianz [A 27]:
Definition 9 (Varianz) *Für eine Zufallsvariable X mit der Dichtefunktion f(x) gilt für die Varianz:*

$$Var(X) = E([X - E(X)]^2) = \begin{cases} \int_{-\infty}^{+\infty} [X - E(X)]^2 \cdot f(x)dx, & falls\ X\ stetig\ ist \\ \sum_k [x(k) - E(X)]^2 \cdot f(x(k)), & falls\ X\ diskret\ ist \end{cases} \tag{36}$$

Bei unbekannter Dichtefunktion wird die Stichprobenvarianz wie folgt berechnet [A 27]:

$$Var(X) = \frac{1}{n-1} \cdot \sum_{k=1}^{n} [x(k) - E(X)]^2 \tag{37}$$

Aus der Varianz berechnet sind auch die Standardabweichung [A 27]:

$$\sigma(X) = \sqrt{Var(X)} \tag{38}$$

Ein weiteres wichtiges Maß ist die Kovarianz mehrerer Zufallsvariablen [A 27]:
Definition 10 (Kovarianz) *Allgemein ist die Kovarianz zweier Zufallsvariablen X und Y definiert als:*

$$Cov(X, Y) = E((X - E(X)) \cdot (Y - E(Y))). \tag{39}$$

Mittels Verschiebungssatz von Steiner ergibt sich:

$$Cov(X, Y) = E((X - E(X)) \cdot (Y - E(Y))) = E(X \cdot Y) - E(X) \cdot E(Y). \tag{40}$$

Für stetige Zufallsvariablen X und Y mit der zwei-dimensionalen Dichtefunktion $F_{XY}(x, y)$ gilt:

$$Cov(X,Y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x - E(X)) \cdot (y - E(Y)))F_{XY}(x,y)\, dxdy \tag{41}$$

Für die Stichprobe diskreter Zufallsvariablen X und Y gilt:

$$Cov(X,Y) = \frac{1}{n-1} \cdot \sum_{k=1}^{n} (x(k) - E(X)) \cdot (y(k) - E(Y))) \tag{42}$$

$$= \frac{1}{n-1} \cdot \sum_{k=1}^{n} (x(k) \cdot y(k)) - \frac{1}{n-1} \cdot E(X) \cdot E(Y) \tag{43}$$

Des Weiteren sei der Korrelationskoeffizient definiert. Dieser gibt an, ob eine lineare Korrelation zwischen zwei Zufallsvariablen existiert. Er ist normiert und nimmt Werte zwischen -1 und +1 an. Hierbei bedeutet der Wert 0, dass beide Zufallsvariablen unkorreliert sind [A 27]:

Definition 11 (Korrelationskoeffizient) *Der Korrelationskoeffizient zweier Zufallsvariablen X und Y ist definiert als:*

$$r_{XY} = \frac{Cov(X,Y)}{\sqrt{Var(X) \cdot Var(Y)}} \tag{44}$$

Weiterhin wird die Kovarianzmatrix wie folgt definiert:

Definition 12 (Kovarianzmatrix) *$X = [X_1, X_2, \ldots, X_n]^T$ sei ein Vektor aus Zufallsvariablen mit $E(X_i) = \mu_i$ und $Var(X_i) = \sigma_i$. Außerdem gelte $\sigma_{ij} = Cov(X_i, X_j),\ \forall i \neq j$. Der zugehörige Erwartungsvektor von X sei folglich:*

$$E(X) = E \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix} = \mu, \tag{45}$$

Die Kovarianzmatrix von X sei dann wie folgt definiert:

$$Cov(X) = E((X-\mu)(X-\mu)^T) \tag{46}$$

$$= E\begin{pmatrix} (X_1-\mu_1)^2 & (X_1-\mu_1)(X_2-\mu_2) & \cdots & (X_1-\mu_1)(X_n-\mu_n) \\ (X_2-\mu_2)(X_1-\mu_1) & (X_2-\mu_2)^2 & \cdots & (X_2-\mu_2)(X_n-\mu_n) \\ \vdots & \vdots & \ddots & \vdots \\ (X_n-\mu_n)(X_1-\mu_1) & (X_n-\mu_n)(X_2-\mu_2) & \cdots & (X_n-\mu_n)^2 \end{pmatrix} \tag{47}$$

$$= \begin{pmatrix} Var(X_1) & Cov(X_1,X_2) & \cdots & Cov(X_1,X_n) \\ Cov(X_2,X_1) & Var(X_2) & \cdots & Cov(X_2,X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n,X_1) & Cov(X_n,X_2) & \cdots & Var(X_n) \end{pmatrix} \tag{48}$$

$$= \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{pmatrix} \tag{49}$$

Die Diagonalelemente der Kovarianzmatrix enthalten also Informationen über die Streuung der einzelnen Zufallsvariablen, alle anderen Elemente jeweils über die Korrelation zwischen ihnen. Weiterführende Informationen finden sich z.B. in [A 28, 29, 27].

2.4 Mathematische Verfahren zur Verbesserung der Schätzung von Taktparametern

Zeitsynchronisation entspricht dem Schätzen von Taktparametern (Offset und Drift). Zur Verbesserung dieser Schätzung lassen sich verschiedene mathematische Verfahren nutzen. Im Folgenden werden einige Verfahren vorgestellt, die für das Verständnis dieser Forschungsarbeit wichtig sind.

2.4.1 Exponentielle Filterung

In diesem Abschnitt wird kurz auf die Exponentielle Filterung bzw. Exponentielle Glättung (exponential smoothing) eingegangen. Das Ziel der Exponentiellen Filterung ist die Mittelung einer Folge von Werte oder die Vorhersage (Schätzung) des kommenden Wertes aus den bisherigen.

Beim arithmetischen Mittelwert werden alle Werte gleich bewertet. Im Gegensatz dazu gewichtet die Exponentielle Filterung neuere Werte exponentiell stärker als ältere [A 30]. Konkret findet hierfür nach [A 31] folgende Formel Anwendung:

Tabelle 2.1: Beispiel der Koeffizienten für verschiedene α.

α	$(1-\alpha)$	$(1-\alpha)^2$	$(1-\alpha)^3$	$(1-\alpha)^4$
0.9	0.1	0.01	0.001	0.0001
0.5	0.5	0.25	0.125	0.0625
0.1	0.9	0.81	0.729	0.6561

$$S_t = \alpha \cdot y_{t-1} + (1-\alpha) \cdot S_{t-1}, \qquad 0 < \alpha \leq 1, t \geq 3. \tag{50}$$

Hierbei bezeichnet y_t den t-ten beobachteten Wert der Folge und S_t den korrespondierenden t-ten gefilterten Wert. Weiterhin bezeichnet α den Glättungsfaktor.

Die Initialisierung kann z.B. mittels $S2 = y1$ erfolgen. Alternativ kann aber auch der arithmetische Mittelwert der ersten 3 bis 5 Werte oder, falls bekannt, der Sollwert verwendet werden [A 30]. Zu beachten ist, dass kein $S1$ existiert. Eine Mittelung über genau einen Wert wäre allerdings auch nicht zweckmäßig.

Nach [A 30] lässt sich der beste Wert für α durch ausprobieren z.B. anhand des MSE (mean of the squared errors) finden. Gesucht wird α, sodass $S_t \approx y_t$ gilt, wobei als Qualitätsmaß der MSE verwendet wird. Automatisiert lässt sich dies mit einem MMSE (minimum mean of the squared errors) Schätzer erreichen, z.B. in Matlab oder Python.

Das exponentielle Verhalten des Filters lässt sich nach [A 30] wie folgt zeigen. Das Einsetzen von S_{t-1} in Gl. 50 ergibt:

$$S_t = \alpha y_{t-1} + (1-\alpha)[\alpha y_{t-2} + (1-\alpha)S_{t-2}] \tag{51}$$

$$= \alpha y_{t-1} + \alpha(1-\alpha)y_{t-2} + (1-\alpha)^2 S_{t-2}. \tag{52}$$

Die Weiterführung bis $S2 = y1$ führt zu:

$$S_t = \alpha \sum_{i=1}^{t-2} (1-\alpha)^{i-1} y_{t-i} + (1-\alpha)^{t-2} S_2, \qquad t \geq 2. \tag{53}$$

Wird hier z.B. S_5 eingesetzt, so entsteht:

$$S_5 = \alpha[(1-\alpha)^0 y_{5-1} + (1-\alpha)^1 y_{5-2} + (1-\alpha)^2 y_{5-3}] + (1-\alpha)^3 S_2. \tag{54}$$

Die einzelnen Werte y_{t-i} werden ergo gewichtet mit $(1-\alpha)^t$. In Tabelle 2.1 findet sich ein Beispiel der Koeffizienten für verschiedene α.

2.4.2 Lineare Optimierung (LP)

Im folgenden Abschnitt wird die lineare Optimierung (LP) als Methode zur Lösung praktischer Optimierungsprobleme vorgestellt. Zunächst wird die Funktionsweise der linearen Optimierung an einem kurzen Beispiel erläutert. Im Anschluss wird allgemein auf die Formulierung linearer Probleme eingegangen.

Herleitung anhand eines Beispiels

Das folgende Beispiel basiert auf verschiedenen Arbeiten [2.4]. In der Literatur findet sich aber eine Vielzahl guter Anwendungsbeispiele aus Wirtschaft, Produktion und Technik. Der interessierte Leser sei z.B. an [A 32] weiterverwiesen.

Ein Landwirt hat eine begrenzte Fläche Land (L = 20 km²) und eine begrenzte Menge Dünger (F = 30 kg) zur Verfügung. Unter diesen Bedingungen versucht er seine Einnahmen zu maximieren. Der Landwirt kann sich entscheiden, ob er Weizen oder Gerste anbauen möchte, oder eine Kombination von beiden. Jedes der Getreide erzielt beim Verkauf am Markt einen bestimmten Preis pro km² und benötigt zur Produktion eine bestimmte Menge an Dünger pro km². Für Weizen gelte:

- Preis pro km² = 750 € = $C1$
- Dünger pro km² = 1 kg = $F1$

Für Gerste gelte:

- Preis pro km² = 1000 € = $C2$
- Dünger pro km² = 2 kg = $F2$

Im Folgenden bezeichne x_1 die Fläche in km² auf der Weizen angebaut wird und x_2 die entsprechende Fläche für Gerste. Das Maximieren der Einnahmen entspricht dem Maximieren folgender Optimierungsfunktion:

$$S(x_1, x_2) = C1 \cdot x_1 + C2 \cdot x_2 \tag{55}$$
$$= 750 \cdot x_1 + 1000 \cdot x_2 \tag{56}$$

Aus der begrenzten Fläche L und begrenzten Menge Dünger F entstehen folgende Constraints (Randbedingungen [2.5]):

$$x_1 + x_2 <= L \tag{57}$$
$$F1 \cdot x_1 + F2 \cdot x_2 <= F \tag{58}$$

Graphisch lässt sich dieses Problem sehr gut im zweidimensionalen Raum darstellen und auch lösen. In Abb. 2.7 ist das Problem graphisch dargestellt. Die Variablen x_1 und x_2 spannen den Lösungsraum auf.

[2.4] Das Beispiel ist rein fiktiv. Die Zahlenwerte sollen nur der besseren Verständlichkeit dienen und erheben keinen Anspruch auf Realitätsnähe. Das Beispiel wurde dabei bewusst in der Landwirtschaft angesiedelt - einer der Schlüsselindustrien in MV.

[2.5] In dieser Arbeit wird für LP-Randbedingungen bewusst der englische Ausdruck Constraints verwendet um Verwechslungen mit Annahmen usw. zu vermeiden.

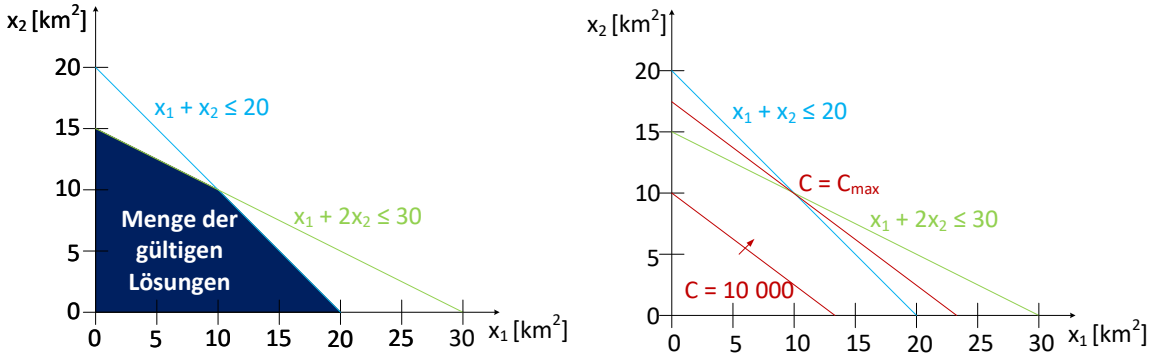


Abbildung 2.7: Menge der gültigen Lösungen und graphische Lösung des Problems

Die Constraints sind Geraden, welche die Menge der gültigen Lösungen einschränken. Eine Lösung ist dann gültig, wenn sie unterhalb aller Constraint-Geraden liegt. Jeder Punkt (x_1, x_2) in diesem Lösungsbereich ist eine gültige Lösung des Problems. Für die Darstellung der Maximierungsfunktion ließe sich jetzt eine dritte Dimension einführen. Durch Umstellung der Optimierungsfunktion mit $S(x_1, x_2) = const$ lassen sich auch Geraden einzeichnen, auf denen alle Lösungen gleich gut sind, denn die Optimierungsfunktion hat für alle Punkte auf dieser Geraden den gleichen Wert. Das Maximieren der Optimierungsfunktion entspricht also einer Parallelverschiebung der Geraden weg vom Koordinatenursprung (vgl. Abb. 2.7). Wie sich in diesem Fall einfach ablesen lässt, liegt die optimale Lösung im Punkt $(10, 10)$. Im Optimalfall sollten also auf genau 10 km² Weizen und auf den anderen 10 km² Gerste angebaut werden.

Bei diesem Problem konnte die Lösung zwar einfach graphisch gefunden werden, der Vollständigkeit halber soll aber auch gezeigt werden, wie sich das Problem rechnerisch lösen lässt. Aus dem „Schieben“ der Lösungsgeraden weg vom Koordinatenursprung folgt, dass die Lösung in einem der Eckpunkte des Lösungsbereichs liegen muss. Die Eckpunkte (x_1, x_2) des Lösungsraums sind $(0, 0)$, $(20, 0)$, $(0, 15)$, sowie der Schnittpunkt der beiden Constraint-Geraden (vgl. Abb. 2.7). Letzterer lässt sich durch Umstellung der Geradengleichungen für die Constraints nach x_2 und Gleichsetzen bestimmen:

$$\begin{aligned} -x_1 + 20 &= -x_1/2 + 15| + x_1/2 \\ -x_1/2 + 20 &= +15| - 20 \\ -x_1/2 &= -5| \cdot (-2) \\ x_1 &= 10 \end{aligned}$$

Durch Einsetzen in $x_2 = -x_1 + 20$ ergibt sich: $x_2 = 10$. Der Schnittpunkt der beiden Geraden ist also $(10, 10)$. Nun gilt es, die vier Kandidaten für die optimale Lösung in die Optimierungsfunktion einzusetzen:

$$\begin{aligned} S &= 750 \cdot x1 + 1000 \cdot x2 \\ &= 0|_{x1=0,x2=0} \\ &= 750 \cdot 20 = 15000|_{x1=20,x2=0} \\ &= 15 \cdot 1000 = 15000|_{x1=0,x2=15} \\ &= 7500 + 10000 = 17000|_{x1=10,x2=10}. \end{aligned}$$

Die optimale Lösung ist also, wie bereits aus dem Diagramm abgelesen, der Punkt $(10, 10)$, da die Optimierungsfunktion mit $17\ 500$ hier den höchsten Wert erreicht. Es sei angemerkt, dass es unendlich viele optimale Lösungen gibt, wenn die Optimierungsfunktion für $S(x_1, x_2) = const$ parallel zu einer Constraint-Geraden verläuft (vgl. Abb. 2.7).

Für lineare Optimierungsprobleme gibt es eine Vielzahl von Solver-Programmen (z.B. in Matlab oder Python SciPy). Diese können auch deutlich komplexere Probleme lösen. Wenn z.B. 20 Pflanzensorten angebaut werden können, hätte der von den Variablen aufgespannte Raum 20 Dimensionen statt nur 2. Weitere Anforderungen könnten eine begrenzte Menge an Wasser und Pestiziden sein, sowie Anforderungen über die Diversität auf den Feldern zum Schutz gegen Missernten. Der Lösungsbereich wäre dann also ein Polyeder im mehrdimensionalen Raum.

Allgemeine Beschreibung

Bei der Modellbildung für lineare Optimierungsprobleme kann es zu Problemen kommen [A 32]. Die Überführung in eine klare mathematische Form ist dann erschwert, wenn es konkurrierende Optimierungsziele gibt. Weiterhin wird eine komplette Übersicht über alle Constraints des Anwenders benötigt. Außerdem gibt es immer unterschiedliche mathematische Beschreibungen für dasselbe praktische Problem, z.B. durch die Einführung zusätzlicher Variablen. Das Problem kann dann für eine Solver-Software deutlich schwerer oder leichter zu lösen sein. Dies kann z.B. zu deutlichen Unterschieden bzgl. der Rechenzeit führen. Weiterhin ist eine Behandlung der Constraints ungünstig, falls diese eine Mischung aus Gleichungen und Ungleichungen ($=$, $\leq$, $\geq$) sind.

Gesucht werden kann ein Maximum oder Minimum der Optimierungsfunktion. Das Maximieren hat sich zwar durchgesetzt, aber beides lässt sich ineinander überführen, denn es gilt für die Optimierungsfunktion $c(x)$:

$$\max_{x \in S} c(x) = -[\min_{x \in S} -c(x)] \tag{59}$$

Für das Simplex-Lösungsverfahren sollte ein Optimierungsproblem immer folgende Form haben [A 32]:

Definition 13 (Lineares Optimierungsproblem in Standardform) *Sei $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $b \geq 0$ und $c \in \mathbb{R}^n$. Ferner sei $m \leq n$ und A habe vollen Rang, also $rang(A) = m$. Dann heißt die folgende Aufgabenstellung lineares Optimierungsproblem in Standardform:*

$$\max(c^T x), \qquad Ax \leq b, x \geq 0. \tag{60}$$

Hierbei gilt Folgendes. Wenn ein x existiert, sodass: $x \in \mathbb{R}^n, x \geq 0$ mit $Ax \leq b$ dann ist das Problem zulässig (es gibt also mindestens eine Lösung). In diesem Fall ist x^* die Optimallösung, falls gilt:

$$x^* \in \mathbb{R}^n \tag{61}$$

$$c^T x \leq c^T x^*, \qquad \forall x \in \mathbb{R}^n, x \leq b. \tag{62}$$

Falls das Problem zulässig ist und die Optimierungsfunktion im Lösungsbereich einen Maximalwert z einnimmt, sodass gilt:

$$c^T x \leq z, \forall x \in \mathbb{R}^n \tag{63}$$

dann ist das Problem beschränkt. Ansonsten ist es unbeschränkt.

Lösungsmethoden für die lineare Optimierung

Für die lineare Optimierung wurden viele automatisierbare und möglichst effiziente Lösungsmethoden entwickelt wie das Innere-Punkte-Verfahren oder das Simplex-Verfahren. Auch in den letzten Jahrzehnten haben Mathematiker auf diesem Gebiet noch neue Erkenntnisse gewinnen können. An dieser Stelle soll nicht mehr detailliert auf die unterschiedlichen Lösungsmethoden eingegangen werden. Weitere Ausführungen finden sich diesbezüglich z.B. in [A 32].

2.4.3 Kalman-Filter

In diesem Abschnitt wird die Funktionsweise des Kalman-Filters vorgestellt. Diesem liegt folgende Idee zu Grunde. Es verwendet verrauschte Messwerte von einigen Größen und ein Systemmodell, das die Zusammenhänge zwischen den Größen beschreibt. Dieses Verfahren kann sowohl zum Herausrechnen des Messrauschens genutzt werden als auch zur Schätzung von Werten, die gar nicht gemessen wurden.

Das Kalman-Filter hat viele technische Anwendungen, wie z.B. die „Fusion“ der Messwerte von GPS-Position und Geschwindigkeit bei der Navigation von Autos, Schiffen und Flugzeugen. Die erste prominente Anwendung fand das Kalman-Filter bei der Mondlandung [A 27].

Herleitung des Kalman-Filters am Beispiel der Mondlandung

Bei der Mondlandung sollten mit Hilfe des Kalman-Filters die Höhe (über dem Mond) $h(t)$, die Geschwindigkeit $v(t)$ und die Beschleunigung $a(t)$ der Mondfähre geschätzt werden. Als Basis hierfür sollten Messwerte für Höhe und Beschleunigung dienen [A 27].

Die gesuchte Geschwindigkeit $v(t)$ ließe sich z.B. durch Differenzierung von $h(t)$ errechnen. Das entstehende Signal wäre dann aber zu verrauscht (vgl. Abb. 2.8). Zwar ließe sich das Signal mittels Tiefpassfilterung glätten, dies würde aber zu einer zeitlichen Verzögerung führen. Wenn nur eine Aussage über die Geschwindigkeit von vor z.B. 3 Sekunden, nicht aber über die aktuelle Geschwindigkeit getroffen werden kann, dann ist dies ein großer Nachteil für viele Echtzeitsysteme mit kurzer Antwortzeit. Auch eine Integration von $a(t)$ wäre möglich um $v(t)$ zu bestimmten. Dann allerdings ergibt sich das Problem der unbekannten Integrationskonstante bzw. Anfangsgeschwindigkeit. Das Kalman-Filter ist zur Lösung der Probleme in diesem Fall sehr gut geeignet, weil es die verschiedenen Messwerte „fusioniert“ und deren Plausibilität gegeneinander abwägt. Außerdem ist es aufgrund seiner kurzen Antwortzeit auch für Echtzeitsysteme geeignet [A 27].

Zunächst muss das System, also das dynamische Verhalten der Mondlandefähre, im Zustandsraum modelliert werden. Der zeitkontinuierliche Zusammenhang von $a(t)$, $v(t)$ und $h(t)$ lässt sich beschreiben durch:

$$a(t) = v'(t) = h''(t). \tag{64}$$

Das System habe den Eingang u, den aktuellen Zustand x und den Ausgang y (vgl. Abb. 2.9). Im Zustandsraum lässt sich dieses System wie folgt beschreiben [A 27]:

$$x'(t) = A \cdot x(t) + B \cdot u(t) + G \cdot z(t) \tag{65}$$
$$y(t) = C \cdot x(t) + D \cdot u(t) \tag{66}$$

Hierbei sei $u(t)$ der Eingangsvektor und $z(t)$ das System- bzw. Prozessrauschen. Weiterhin sei $x(t)$ der frei wählbare Zustandsvektor, z.B.:

$$x(t) = \begin{bmatrix} h(t) \\ v(t) \\ a(t) \end{bmatrix} \tag{67}$$

Dessen Ableitung nach der Zeit sei:

$$x'(t) = \begin{bmatrix} h'(t) \\ v'(t) \\ a'(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ a(t) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot z(t). \tag{68}$$

Hierbei wird angenommen, dass die $a'(t)$, die Ableitung der Beschleunigung 0 sei. Das ist aber i.A. keine Einschränkung, da mögliche Änderungen der Beschleunigung über das Systemrauschen $z(t)$ modelliert werden können. Der Ausgangsvektor $y(t)$ beschreibt die gemessenen Größen:

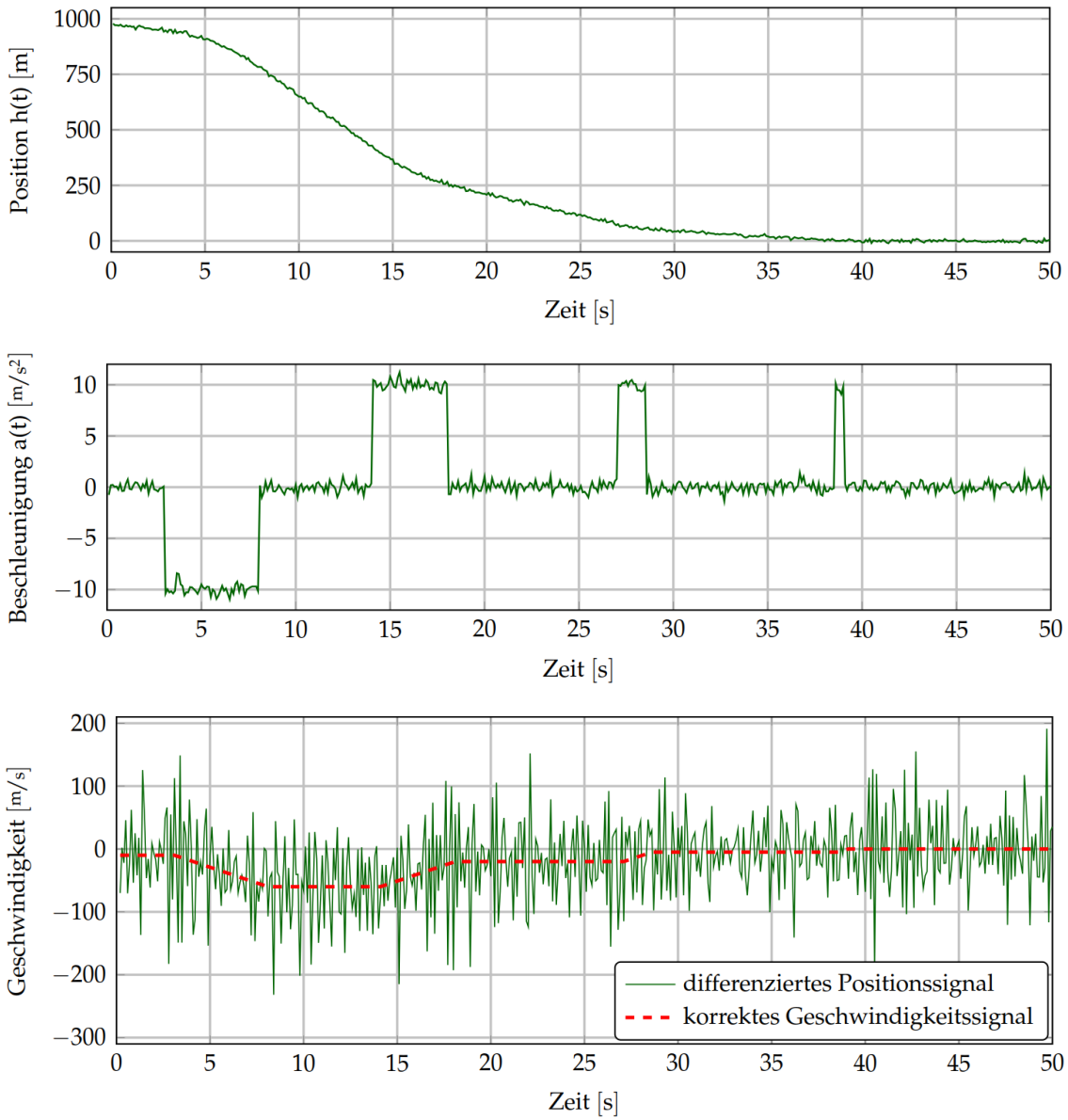


Abbildung 2.8: Verlauf messtechnisch erfassbarer Größen der Mondlandefähre: Abstand von der Mondoberfäche $h(t)$ und Beschleunigung $a(t)$, sowie der durch Differenzierung erhaltene Verlauf der Geschwindigkeit $v(t)$ der Mondlandefähre (Abbildungen übernommen aus [A 27])

Abbildung 2.9: Blockdiagramm eines Systems mit dem Eingang u, dem aktuellen Zustand x und dem Ausgang y.

$$y(t) = \begin{bmatrix} h(t) \\ a(t) \end{bmatrix}. \tag{69}$$

Für die zeitkontinuierliche Systembeschreibung folgt also [A 27]:

$$x'(t) = \begin{bmatrix} h'(t) \\ v'(t) \\ a'(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} h(t) \\ v(t) \\ a(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot u(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot z(t). \tag{70}$$

$$= \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{A}} \cdot x(t) + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{B}} \cdot u(t) + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{\text{G}} \cdot z(t) \tag{71}$$

$$y(t) = \begin{bmatrix} h(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h(t) \\ v(t) \\ a(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdot u(t) \tag{72}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{C}} \cdot x(t) + \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\text{D}} \cdot u(t) \tag{73}$$

Für die Implementierung des Kalman-Filters bspw. in Software ist eine zeitdiskrete Systembeschreibung nötig. Mit dem Abtastzeitraum Ts entsteht ein lineares zeitdiskretes System:

$$x(k+1) = A_d \cdot x(k) + B_d \cdot u(k) + G_d \cdot z(k) \tag{74}$$

$$y(k) = C \cdot x(k) + D \cdot u(k) \tag{75}$$

mit:

$$A_d = e^{A \cdot T_s}, B_d = \int_0^{T_s} e^{A \cdot v} \cdot B \delta v, G_d = A_d \cdot G. \tag{76}$$

Mittels der Laplace-Transformation ergibt sich [A 27]:

$$A_d = \begin{bmatrix} 1 & T_s & T_s^2/2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}, \; B_d = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \; G_d = \begin{bmatrix} T_s^2/2 \\ T_s \\ 1 \end{bmatrix} \tag{77}$$

$$x(k+1) = \begin{bmatrix} h(k+1) \\ v(k+1) \\ a(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & T_s & T_s^2/2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}}_{A_d} \cdot \begin{bmatrix} h(k) \\ v(k) \\ a(k) \end{bmatrix} + \underbrace{\begin{bmatrix} T_s^2/2 \\ T_s \\ 1 \end{bmatrix}}_{G_d} \cdot u(t) \tag{78}$$

An dieser Stelle sei kurz der Begriff der Beobachtbarkeit erwähnt. Das System ist beobachtbar, wenn die Bestimmung des Zustandsvektors aus den Messgrößen in einer endlichen Zeit möglich ist. Das vorliegende Beispielsystem der Mondfähre ist beobachtbar, für den exakten Nachweis sei z.B. auf [A 27] verwiesen.

Bestimmung des Rauschens am Beispiel der Mondlandung

Für die Verwendung eines Kalman-Filters muss das Rauschen korrekt bestimmt werden. Hierbei werden zwei Rauscharten unterschieden. Einerseits bildet das Systemrauschen die Fehler bzw. Ungenauigkeiten in der Systembeschreibung ab. Andererseits bildet das Messrauschen die Fehler bzw. Ungenauigkeiten bei der Messung ab. Das Rauschen wird als mittelwertfrei und normalverteilt angenommen und nur dessen Varianz wird angegeben [A 27].

Zur besseren Verständlichkeit wird wieder das Beispiel der Mondlandung verwendet. Das Systemrauschen kann z.B. praktisch gemessen werden. Im Beispiel der Mondlandung soll angenommen werden, dass es normalverteilt ist und sich die Beschleunigung in 99% aller Fälle weniger als $10m/s^2$ pro Abtastintervall ändert:

$$Q(k) = Var(z(k)) = \sigma_v^2,\ 3 \cdot \sigma_v = 10m/s^2 \tag{79}$$

$$\rightarrow Q(k) = \sigma_v^2 = (10/3\ m/s^2)^2 \approx 11,1m^2/s^4 \tag{80}$$

Auch die Messungen können niemals perfekt sein, z.B. durch Ungenauigkeiten im Messverfahren oder Quantisierung. Dies wird mit dem Messrauschen abgebildet. In den Gleichungen für das Systemmodell findet hierfür eine Überlagerung des Ausgangs $y(k)$ mit $v(k)$ statt:

$$x(k+1) = A_d \cdot x(k) + B_d \cdot u(k) + G_d \cdot z(k) \tag{81}$$

$$y(k) = C \cdot x(k) + D \cdot u(k) + v(k) \tag{82}$$

Auch für das Messrauschen wurden bei der Entwicklung des Kalman-Filters einige Annahmen getroffen. Diese müssen erfüllt sein, damit das Kalman-Filter korrekt arbeitet. So müssen Schätzfehler und Messrauschen unkorreliert sein. Außerdem muss es sich auch hierbei um ein mittelwertfreies und normalverteiltes Rauschen handeln. Die Messvarianzen für das Höhen- und Beschleunigungssignal seien:

$$Var(v_h(k)) = \sigma_h^2 \tag{83}$$

$$Var(v_a(k)) = \sigma_a^2. \tag{84}$$

Diese Werte treffen eine Aussage darüber, wie vertrauenswürdig die Messwerte von $a(k)$ und $h(k)$ sind. Weitere wichtige Annahmen sind, dass sich das Messrauschen über die Zeit nicht ändert und sich beide Messwerte nicht gegenseitig beeinflussen, sie also unkorreliert

sind ($Cov(v_h(k), v_a(k)) = 0$). Die Varianz des Messrauschens lässt sich z.B. aus der Stichprobenvarianz (vgl. Gl. 37) der Messwerte abschätzen:

$$\sigma_h^2 \approx 20m^2 \tag{85}$$
$$\sigma_a^2 \approx 0.2m^2/s^4 \tag{86}$$

und somit:

$$R(k) = Var(v(k)) = \begin{bmatrix} v_h(k) & Cov(v_h(k), v_a(k) \\ Cov(v_h(k), v_a(k) & v_a(k) \end{bmatrix} \tag{87}$$
$$= \begin{bmatrix} \sigma_h^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix} \approx \begin{bmatrix} 20m^2 & 0 \\ 0 & 0.2m^2/s^4 \end{bmatrix} \tag{88}$$

Kalman Gleichungen

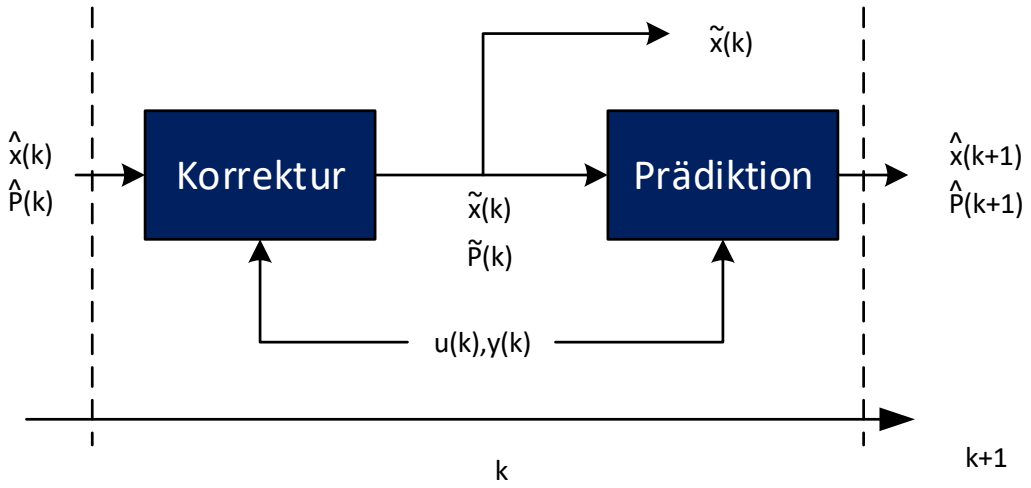


Abbildung 2.10: Grundlegende Struktur des Kalman-Filters aus Korrektur und Prädiktion

Im Folgenden wird näher auf die genaue Funktionsweise des Kalman-Filters eingegangen. Diese wird durch die Kalman-Gleichungen beschrieben. Es erfolgt ein zyklisches Ausführen von zwei Phasen: Korrektur und Prädiktion (vgl. Abb. 2.10) [A 28, 27]. Der Zustandsvektor $x(k)$ und der Ausgangsvektor $y(k)$ seien wie folgt definiert:

$$x(k) = \begin{bmatrix} h(k) \\ v(k) \\ a(k) \end{bmatrix}, \; y(k) = \begin{bmatrix} h(k) \\ a(k) \end{bmatrix}. \tag{89}$$

In der Korrekturphase werden folgende Operationen ausgeführt [A 28, 27]. Zunächst wird aus dem Eingang $u(k)$ und dem geschätzten Zustandsvektor $\hat{x}(k)$ der geschätzte Ausgangsvektor $\hat{y}(k)$ ermittelt:

$$\hat{y}(k) = C \cdot \hat{x}(k) + D \cdot u(k). \tag{90}$$

Dann wird $\Delta y(k)$, die Differenz zwischen Prädiktion und Messung gebildet. Der Wert von Δy wird also klein, wenn prädizierte und gemessene Größe annähernd gleich sind:

$$\Delta y(k) = y(k) - \hat{y}(k). \tag{91}$$

Für die Kalmanverstärkung $K(k)$ sind insbesondere das Messrauschen $R(k)$ und die Kovarianz des Schätzfehlers $P(k)$ entscheidend. Beide Werte haben einen Einfluss darauf, was das Kalman-Filter für plausibler hält: die Messung oder die eigene Schätzung:

$$K(k) = \hat{P}(k) \cdot C^T \cdot (C \cdot \hat{P}(k) \cdot C^T + R(k))^{-1}. \tag{92}$$

Dann wird der korrigierte Zustandsvektor bestimmt aus dem prädizierten Zustandsvektor, der Kalmanverstärkung und Δy. Die Größe der Korrektur hängt also sowohl von Δy ab als auch von K:

$$\tilde{x}(k) = \hat{x}(k) + K(k) \cdot \Delta y. \tag{93}$$

Zum Schluss wird die Kovarianz des Schätzfehlers bestimmt:

$$\tilde{P}(k) = (I - K(k) \cdot C) \cdot \hat{P}(k). \tag{94}$$

In der Prädiktionsphase finden folgende Operationen statt [A 28, 27]. Es wird zunächst das Systemmodell verwendet, um den Zustandsvektor aus dem Eingang vorherzusagen:

$$\hat{x}(k+1) = A_d \cdot \tilde{x}(k) + B_d \cdot u(k). \tag{95}$$

Außerdem wird die Kovarianz des Schätzfehlers extrapoliert. Hierfür wird die Systemungenauigkeit $G_d \cdot Q(k) \cdot G_d^T$ verwendet:

$$\hat{P}(k+1) = A_d \cdot \tilde{P}(k) \cdot A_d^T + G_d \cdot Q(k) \cdot G_d^T. \tag{96}$$

Die genaue Herleitung der Kalman-Filter Gleichungen findet sich u.A. in [A 28, 27].

Bei $P(k)$ handelt es sich um die Kovarianz des Schätzfehlers, also die Differenz zwischen wahrem Wert und geschätztem Wert. Hierbei wird unterschieden zwischen der Differenz zw. wahrem Wert und korrigiertem Wert ($\tilde{\epsilon}(k) = x(k) - \tilde{x}(k)$) und der Differenz zw. wahrem Wert und prädiziertem Wert ($\hat{\epsilon}(k) = x(k) - \hat{x}(k)$). Für den korrigierten Schätzfehler gilt also z.B.:

$$\tilde{\epsilon}(k) = \begin{bmatrix} \tilde{\epsilon}_h(k) \\ \tilde{\epsilon}_v(k) \\ \tilde{\epsilon}_a(k) \end{bmatrix} = \begin{bmatrix} x_h(k) - \tilde{x}_h(k) \\ x_v(k) - \tilde{x}_v(k) \\ x_a(k) - \tilde{x}_a(k) \end{bmatrix} \tag{97}$$

Die Matrix der Kovarianz des Schätzfehlers ($\tilde{P}(k) = Var(\tilde{\epsilon}(k))$) ergibt sich also wie folgt:

$$\tilde{P}(k) = \begin{bmatrix} Var(\tilde{\epsilon}_h(k)) & Cov(\tilde{\epsilon}_h(k), \tilde{\epsilon}_v(k)) & Cov(\tilde{\epsilon}_h(k), \tilde{\epsilon}_a(k)) \\ Cov(\tilde{\epsilon}_h(k), \tilde{\epsilon}_v(k)) & Var(\tilde{\epsilon}_v(k)) & Cov(\tilde{\epsilon}_v(k), \tilde{\epsilon}_a(k)) \\ Cov(\tilde{\epsilon}_h(k), \tilde{\epsilon}_a(k)) & Cov(\tilde{\epsilon}_v(k), \tilde{\epsilon}_a(k)) & Var(\tilde{\epsilon}_a(k)) \end{bmatrix} \tag{98}$$

Hierbei gilt z.B. $Var(\tilde{\epsilon}_h(k)) = \tilde{\sigma}_h^2(k)$. Die Diagonalelemente liefern ein Maß für die Vertrauenswürdigkeit der Schätzung. Generell ist der Schätzwert nach der Korrektur näher an dem realen Wert, sodass gilt: $\tilde{P}(k) \leq \hat{P}(k)$. Je besser das Modell, desto geringer ist die Verbesserung durch die Korrektur, sodass gilt: $\tilde{P}(k) \approx \hat{P}(k)$.

Schätzergebnisse des Kalman-Filters am Beispiel der Mondlandung und Zusammenfassung

Die Ergebnisse für das Beispiel werden in Abb. 2.11 gezeigt. Der Schätzwert ist deutlich glatter als die Messwerte und weist ihnen gegenüber keinen merklichen Versatz auf.

Zusammenfassend lässt sich sagen, dass das Kalman-Filter den Zustand eines linearen Systems mit mittelwertfreien, weißen Störungen schätzt und den Einfluss dieser Störungen verringert [A 28]. Bei der Konzeption des Kalman-Filters wurden jedoch einige Annahmen getroffen, die zu Einschränkungen führen. Das Rauschen muss mittelwertfrei, normalverteilt, von bekannter Varianz und unkorreliert mit dem Schätzfehler sein [A 28, 27].

2.4.4 Adaptive Kalman-Filter - IMM

Bisher waren die Voraussetzungen für die Verwendung eines Kalman-Filters relativ restriktiv. Die Störungen sollten sich modellieren lassen durch bekanntes und mittelwertfreies Gaußrauschen. In der Tat sind diese Voraussetzungen für viele Anwendungen zu restriktiv, denn häufig sind die Störungen unbekannt. Um diesem Umstand gerecht zu werden, wurde das Kalman-Filter erweitert und angepasst. Es sei an dieser Stelle angemerkt, dass die Notation in diesem Kapitel zwar in Teilen nicht sehr intuitiv wirkt, dies aber den Hintergrund hat, dass so insbesondere die Konsistenz zu [A 33, 34] gewahrt bleibt. Es wurden sogenannte adaptive Verfahren entwickelt [A 28]. Die Systembeschreibung verändert sich wie folgt [A 33]:

$$x_{k+1} = F_k(s_{k+1})x_k + G_k(s_{k+1})w_k(s_{k+1}) \tag{99}$$

$$z_k = H_k(s_k)x_k + v_k(s_k). \tag{100}$$

Die Matrizen der Systembeschreibung verändern sich jetzt also mit der Zeit. Hierbei wird insbesondere folgende Annahme getroffen: das System befindet sich zu jedem Zeitpunkt k in einem Modus s^k von mehreren möglichen Modi. Als praktische Ansätze für den Wechsel zwischen den Modi sind in der Literatur IMM (Interacting multiple model) und GPB (Generalized pseudo-Bayesian) beschrieben. Da im Rahmen dieser Arbeit nur IMM zum Einsatz kam, liegt hierauf im Folgenden der Fokus. Die Grundidee von IMM ist wie folgt: für jeden Modus gibt es genau ein Modell, welches das Systemverhalten modelliert. Jedes Modell schätzt den aktuellen Zustand des Systems auf Basis vorheriger Schätzungen. Um ein "Weglaufen"

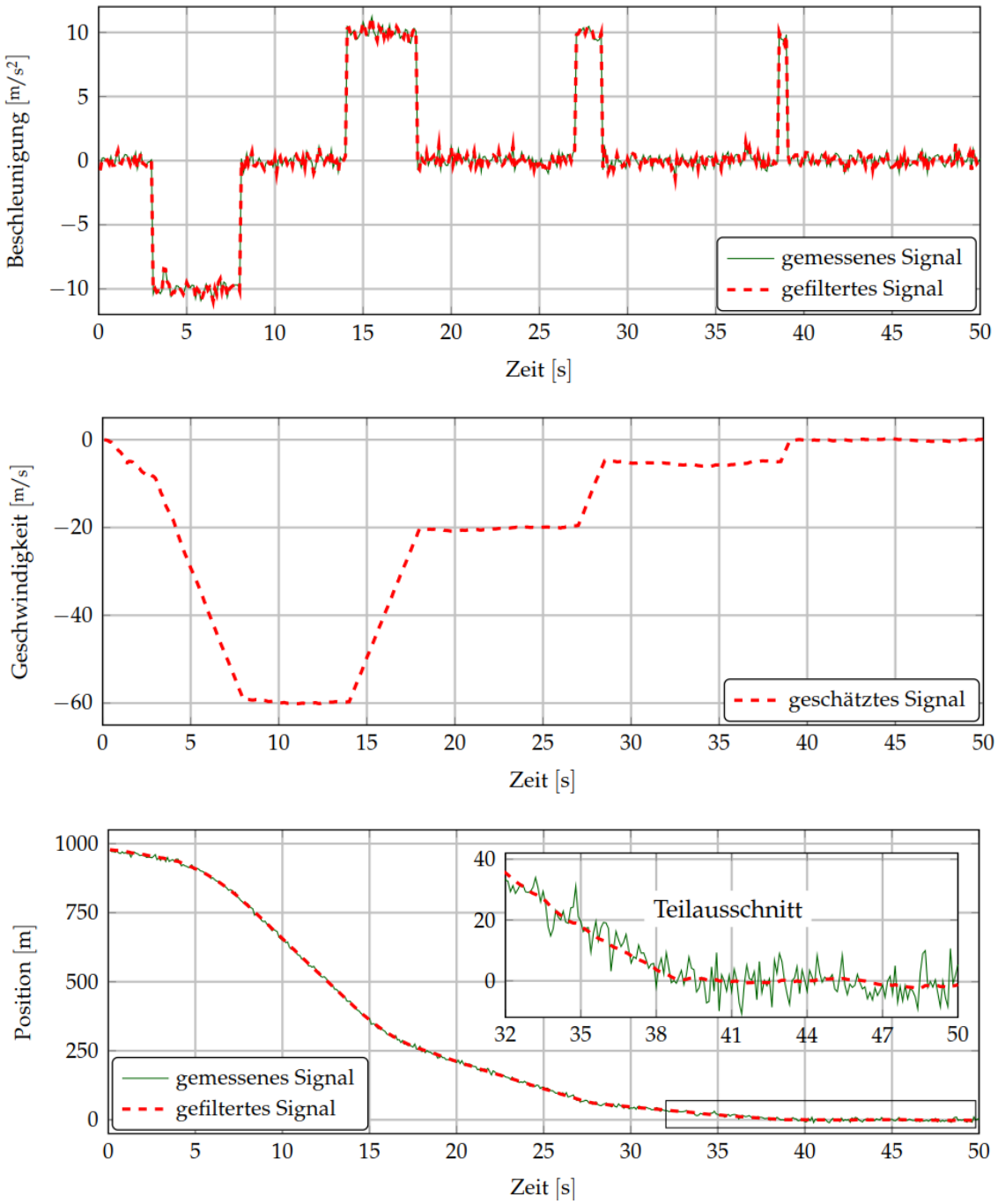


Abbildung 2.11: Verlauf der Größen des korrigierten Zustandsvektors (aus [A 27])

bestimmter Modelle, die nicht zum aktuellen Systemmodus passen, zu verhindern, verwendet jedes Modell eine bestimmte Kombination der vorheriger Schätzungen aller Modelle und nicht nur seiner eigenen Schätzungen (Mixed intial condition) [A 28].

Übersicht und grundlegende Funktionsweise

Zunächst soll eine Übersicht und eine Einführung in die grundlegende Funktionsweise eines IMM-Kalman-Filters gegeben werden. Die grundlegende Struktur ist in Abb. 2.12 dargestellt. In einer Iteration werden folgende Schritte durchlaufen:

- 1. Reinitialisierung: diese ist notwendig, damit keines der Filter "wegläuft", auch wenn dessen Modell aktuell nicht zum Systemmodus passt.
- 2. Filterung nach Modell i: für jeden Systemmodus und dessen Modell gibt es ein Kalman-Filter. Es wird auch von einer Filterbank gesprochen (engl. Filter bank). Jedes dieser Filter nutzt sein Modell und die aktuelle Messung z_k, um den aktuellen Systemzustand x_k zu schätzen.
- 3. Update Modellwahrscheinlichkeiten: In diesem Schritt werden die Modellwahrscheinlichkeiten aktualisiert. Die Modellwahrscheinlichkeit von Filter i ist die Wahrscheinlichkeit, dass das Modell von Filter i aktuell zum Prozess passt.
- 4. Schätzung des Ausgangs: in diesem Schritt findet eine Fusion der Schätzungen aller Filter statt. Hierzu werden insbesondere die Modellwahrscheinlichkeiten genutzt.

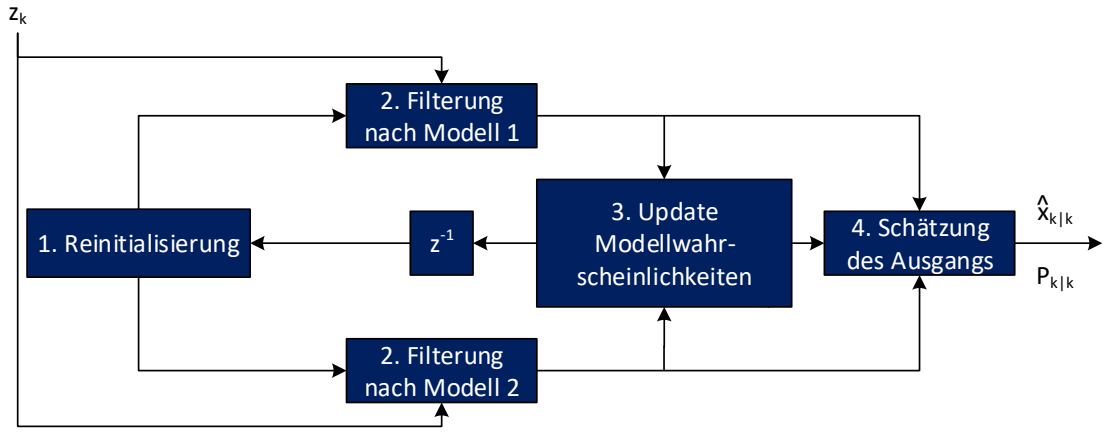


Abbildung 2.12: Grundlegende IMM-Struktur (basierend auf [A 33, 34]

Später wird noch im Detail auf die genauen Berechnungen dieser Schritte eingegangen. Zum besseren Verständnis soll das IMM-Kalman-Filter aber zunächst als Black Box betrachtet werden, um zu beschreiben, wie sich ein IMM-Kalman-Filter verhält und wie es verwendet wird. Zur Umsetzung des Filters könnte z.B. zu einer bestehenden Implementierung aus einer Software-Bibliothek gegriffen werden.

Im Rahmen dieser Forschungsarbeit wurde so z.B. der in [A 34] vorstellte Ansatz implementiert (mittels Python und der IMM-Bibliothek FilterPy). Dieser Ansatz soll im Folgenden als Beispiel dienen. In [A 34] wird ein IMM-basierter Ansatz zur Schätzung des Skews auf Ba-

sis verrauschter Werte vorgestellt. Hierbei kommen zwei Modelle zum Einsatz: ein Modell mit konstantem Skew (Const. Skew) und eines mit konstanter Änderung des Skews (Const. Skew Change). Zur Umsetzung benötigt das IMM Folgendes:

- Die Übergangswahrscheinlichkeiten für den Wechsel von einem Modus in einen anderen.
- Die initiale Wahrscheinlichkeiten für jeden Modus. Diese ist jedoch i.A. nicht so wichtig, da sich das Filter mit der Zeit ”einschwingt”, da es automatisch herausfindet, welches Modell am besten zu den aktuellen Messwerten passt.
- Die unterschiedlichen Systemmodelle für die Kalman-Filter zu den jeweiligen Systemmodi.
- Die Rauschmatrizen für die Kalman-Filter (z.B. experimentell bestimmbar).

Als initiale Wahrscheinlichkeiten für die Modi seien z.B. $\mu_0 = [0.7 \;\; 0.3]$ angenommen. Die genauen Werte der Übergangswahrscheinlichkeiten π_{ij} werden in [A 34] zwar nicht genannt, diese können aber z.B. so angenommen werden wie in [A 35]. Hierbei ist die Wahrscheinlichkeit in einem Modell zu bleiben z.B. sehr viel größer als die Wahrscheinlichkeit eines Modellwechsels. An dieser Stelle seien folgende Übergangswahrscheinlichkeiten angenommen (Vgl. Abb. 2.13):

$$M = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} = \begin{bmatrix} 0.97 & 0.3 \\ 0.05 & 0.95 \end{bmatrix} \tag{101}$$

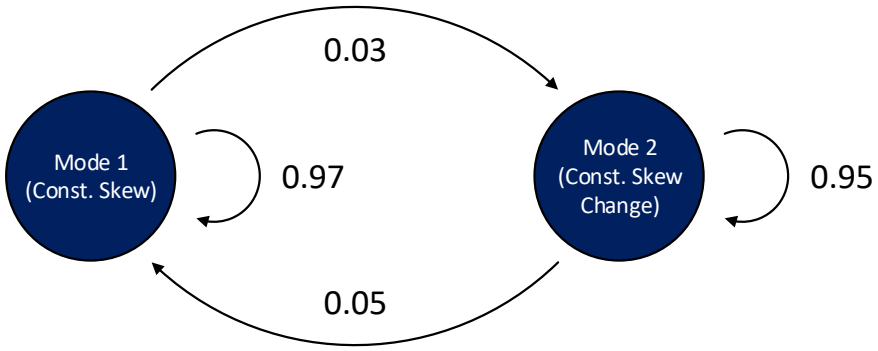
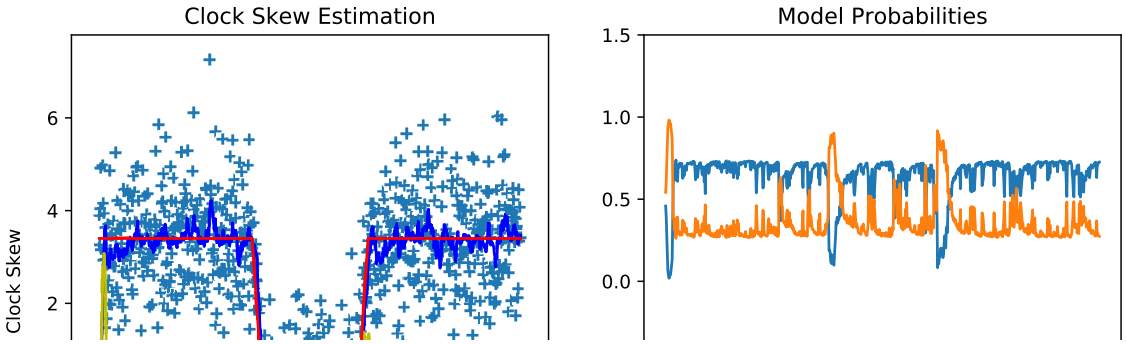


Abbildung 2.13: Übergangswahrscheinlichkeiten aus [A 35]

Die Systemmodelle des i-ten Kalman-Filters haben folgende Form:

$$x_k^{(i)} = A^{(i)} x_{k-1}^{(i)} + v_k^{(i)} \tag{102}$$

$$z_k^{(i)} = H x_{k-1}^{(i)} + \omega_k^{(i)} \tag{103}$$



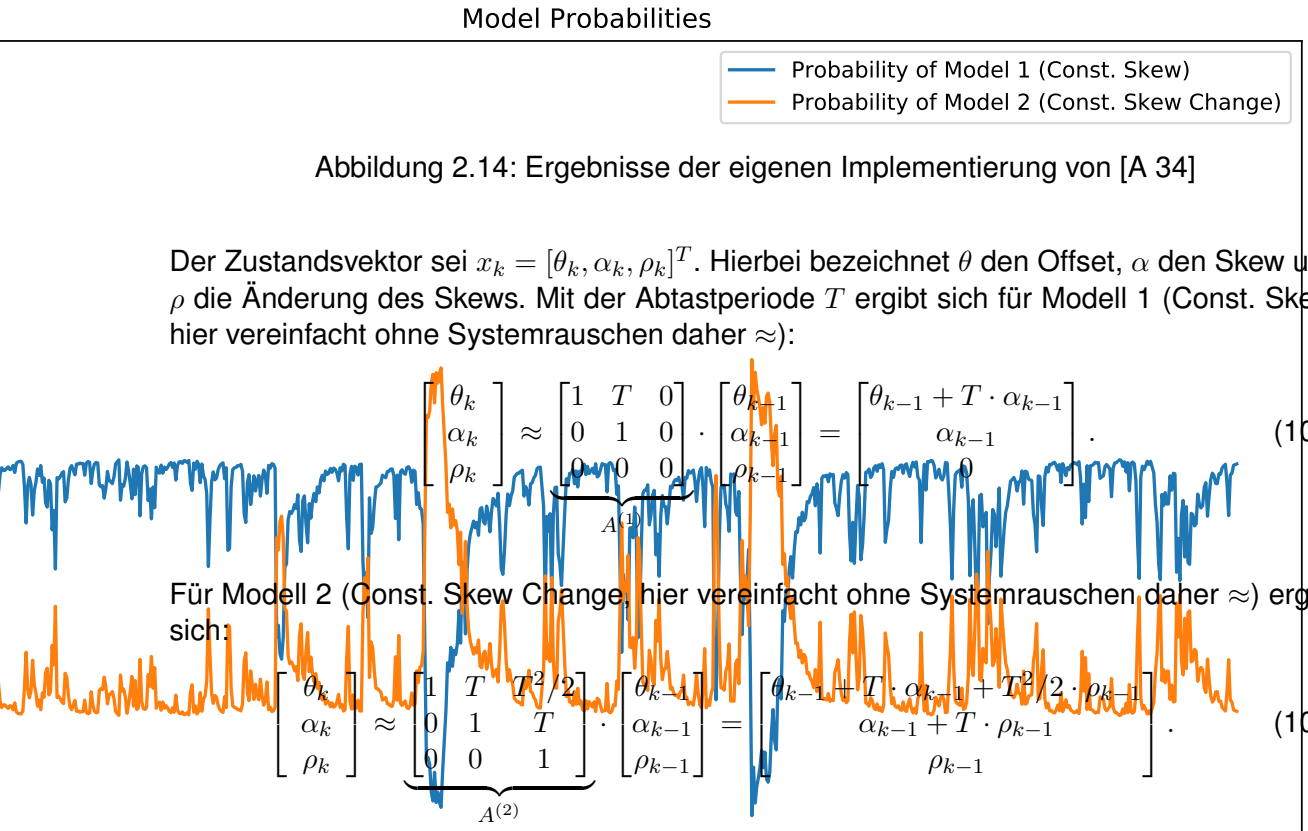


Abbildung 2.14: Ergebnisse der eigenen Implementierung von [A 34]

Der Zustandsvektor sei $x_k = [\theta_k, \alpha_k, \rho_k]^T$. Hierbei bezeichnet θ den Offset, α den Skew und ρ die Änderung des Skews. Mit der Abtastperiode T ergibt sich für Modell 1 (Const. Skew, hier vereinfacht ohne Systemrauschen daher $\approx$):

$$\begin{bmatrix} \theta_k \\ \alpha_k \\ \rho_k \end{bmatrix} \approx \underbrace{\begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A^{(1)}} \cdot \begin{bmatrix} \theta_{k-1} \\ \alpha_{k-1} \\ \rho_{k-1} \end{bmatrix} = \begin{bmatrix} \theta_{k-1} + T \cdot \alpha_{k-1} \\ \alpha_{k-1} \\ 0 \end{bmatrix}. \tag{104}$$

Für Modell 2 (Const. Skew Change, hier vereinfacht ohne Systemrauschen daher $\approx$) ergibt sich:

$$\begin{bmatrix} \theta_k \\ \alpha_k \\ \rho_k \end{bmatrix} \approx \underbrace{\begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}}_{A^{(2)}} \cdot \begin{bmatrix} \theta_{k-1} \\ \alpha_{k-1} \\ \rho_{k-1} \end{bmatrix} = \begin{bmatrix} \theta_{k-1} + T \cdot \alpha_{k-1} + T^2/2 \cdot \rho_{k-1} \\ \alpha_{k-1} + T \cdot \rho_{k-1} \\ \rho_{k-1} \end{bmatrix}. \tag{105}$$

Die Ergebnisse der eigenen Implementierung von [A 34] finden sich in Abb. 2.14. In rot dargestellt ist der tatsächliche Verlauf des Skews (original track). Dieser Verlauf ist dem Filter unbekannt, es erhält lediglich verrauschte Messwerte (blaue Kreuze). In Dunkelblau dargestellt ist die Skew-Schätzung des Filters. In Gelb dargestellt ist die Schätzung des Filters bzgl. der Skew-Änderung. Hier zeigt sich ein grundlegender Vorteil von Kalman-Filtern: der

Wert der Skew-Änderung wurde zwar gar nicht gemessen, kann aber dennoch mit Hilfe des Systemmodells geschätzt werden. Allgemein lässt sich erkennen, wie der tatsächliche Verlauf des Skews relativ gut geschätzt wird, trotz großer Streuung der Messwerte. Außerdem passt sich das Filter sehr schnell an die Änderungen an. Des Weiteren finden sich im rechten Diagramm der Abb. 2.14 die Verläufe der Modellwahrscheinlichkeiten. Anfangs gibt es eine kurze Einschwingphase. Außerdem ist der Wechsel zwischen den Modellen bei ca. 300 s und 500 s sichtbar. In dem Bereich zwischen diesen Zeiten erkennt das IMM-Kalman-Filter, dass das Modell 2 (Const. Skew Change) besser zu den Messwerten passt.

IMM-Gleichungen

In diesem Abschnitt wird im Detail auf die Funktionsweise eines IMM-Kalman-Filters eingegangen. Dabei wird jeder Schritt erklärt. Die Gleichungen stammen sämtlich aus [A 33].

Im ersten Schritt (**1. Reinitialisierung**) findet eine separate Initialisierung des jeweiligen Modells bzw. Filters i, mit $i = 1, 2, \ldots, M$, statt (model-conditioned reinitialization).

Hierbei wird zunächst die Predicted Mode Probability berechnet [A 33]:

$$\mu_{k|k-1}^{(i)} \widehat{=} P\{m_k^{(i)}|z^{k-1}\} = \sum_j \pi_{ji}\mu_{k-1}^{(i)} \tag{106}$$

Hierbei ist $z^k = (z_1, ..., z_k)$ eine Messreihe, die alle Messungen bis zum Zeitpunkt k enthält, wobei z_k genau die Messung zum Zeitpunkt k bezeichnet. Weiterhin ist $\pi_{ji} = P\{m_{k+1}^{(j)}|m_k^{(i)}\}$ die Übergangswahrscheinlichkeit (transition probability) vom Modell $m^{(j)}$ zum Modell $m^{(i)}$. Hierbei bezeichnet $m_k^{(i)} \widehat{=} \{s_k = m^{(i)}\}$ das Ereignis, dass das Modell $m^{(i)}$ zum System Mode s passt (zum Zeitpunkt k). Außerdem bezeichnet μ_{k-1}^i die Wahrscheinlichkeit von Modell $m^{(i)}$ zum Zeitpunkt $k-1$.

Außerdem werden hier die Mixing Weights bestimmt [A 33]:

$$\mu_{k-1}^{(j|i)} \widehat{=} P\{m_{k-1}^{(j)}|m_k^{(i)}, z^{k-1}\} = \pi_{ji}\mu_{k-1}^{(j)}/\mu_{k|k-1}^{(i)} \tag{107}$$

Hierbei ist $\mu_{k|k-1}^{(i)}$ die gerade berechnete Predicted Mode Probability. Weiterhin bezeichnet $\mu_{k-1}^{(j)}$ die Mode Probability des Modells $m^{(j)}$ zum Zeitpunkt $k-1$. Die Übergangswahrscheinlichkeit sei wieder mit π_{ji} bezeichnet.

Als nächstes wird der Mixing Estimate berechnet [A 33]:

$$\overline{x}_{k-1|k-1}^{(i)} \widehat{=} E[x_{k-1}|m_k^{(i)}, z^{k-1}] = \sum_j \hat{x}_{k-1|k-1}^{(j)}\mu_{k-1}^{(j|i)} \tag{108}$$

Hierbei ist $\hat{x}_{k-1|k-1}^{(j)}$ der geschätzte Zustand nach der Update-Phase des Filters j zum Zeitpunkt $k-1$. $\mu_{k-1}^{(j|i)}$ bezeichnet das im vorherigen Schritt bestimmte Mixing Weight.

Weiterhin wird die Mixing Covariance bestimmt [A 33]:

$$\overline{P}_{k-1|k-1}^{(i)} = \sum_j [P_{k-1|k-1}^{(j)} + (\overline{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)})(\overline{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)})^T]\mu_{k-1}^{(j|i)} \tag{109}$$

Hierbei bezeichnet $P^{(j)}_{k-1|k-1}$ die Kovarianz aus der Update-Phase des Filters j zum Zeitpunkt $k-1$. $\overline{x}^{(i)}_{k-1|k-1}$ ist der Mixing Estimate aus der vorherigen Rechnung. Weiterhin ist wieder $\hat{x}^{(j)}_{k-1|k-1}$ der geschätzte Zustand nach der Update-Phase des Filters j zum Zeitpunkt $k-1$. Zum Hintergrund dieser Berechnung sei auf die Berechnung der Kovarianzmatrix verwiesen ($Cov(X) = E[(X-\mu)(X-\mu)^T]$). Außerdem ist allgemein $P_{k|k}$ definiert als $P_{k|k} = MSE(\hat{x}_{k|k}, z^k) = \sum_{i=1}^{M}[P^{(i)}_{k|k} + (\hat{x}^{(i)}_{k|k} - \hat{x}_{k|k})(\hat{x}^{(i)}_{k|k} - \hat{x}_{k|k})^T]\mu^{(i)}_k$ (vgl. [A 33]).

Im zweiten Schritt (**2. Filterung nach Modell** i) findet eine Kalman-Filterung statt. Hierbei verwendet jedes der Kalman-Filter i, mit $i = 1, 2, \ldots, M$, sein eigenes Systemmodell, um den Zustand aus der Messung zu schätzen (model-conditioned filtering). Hierbei werden die normalen Kalman Gleichungen ausgeführt, trotzdem sollen diese der Vollständigkeit halber hier nochmal angegeben werden [A 33]:

$$
\begin{aligned}
Prediced\ state: &\quad \hat{x}^{(i)}_{k|k-1} = F^{(i)}_{k-1}\overline{x}^{(i)}_{k-1|k-1} + G^{(i)}_{k-1}\overline{w}^{(i)}_{k-1} \\
Prediced\ covariance: &\quad P^{(i)}_{k|k-1} = F^{(i)}_{k-1}\overline{P}^{(i)}_{k-1|k-1}(F^{(i)}_{k-1})^T + G^{(i)}_{k-1}Q^{(i)}_{k-1}(G^{(i)}_{k-1})^T \\
Measurement\ residual: &\quad \tilde{z}^{(i)}_k = z_k - H^{(i)}_k\hat{x}^{(i)}_{k|k-1} - \overline{v}^{(i)}_k \\
Residual\ covariance: &\quad S^{(i)}_k = H^{(i)}_k P^{(i)}_{k|k-1}(H^{(i)}_k)^T + R^{(i)}_k \\
Filter\ gain: &\quad K^{(i)}_k = P^{(i)}_{k|k-1}(H^{(i)}_k)^T(S^{(i)}_k)^{-1} \\
Updated\ state: &\quad \hat{x}^{(i)}_{k|k} = \hat{x}^{(i)}_{k|k-1} + K^{(i)}_k\tilde{z}^{(i)}_k \\
Updated\ covariance: &\quad P^{(i)}_{k|k} = P^{(i)}_{k|k-1} - K^{(i)}_k S^{(i)}_k (K^{(i)}_k)^T
\end{aligned}
$$

Im dritten Schritt (**3. Update der Modellwahrscheinlichkeiten**) wird für jedes Modell i, mit $i = 1, 2, \ldots, M$, die Wahrscheinlichkeit bestimmt, dass dieses Modell zum aktuellen System Modus passt (mode probability update).

Hierfür wird zunächst die sogenannte Model Likelihood ermittelt [A 33]:

$$L^{(i)}_k \widehat{=} p[\tilde{z}^{(i)}_k | m^{(i)}_k, z^{k-1}] = \mathcal{N}(\tilde{z}^{(i)}_k; 0, S^{(i)}_k) \tag{110}$$

Hierbei bezeichnet $S^{(i)}_k$ die Residual Covariance und $\tilde{z}^{(i)}_k$ das Measurement Residual (beide aus Schritt 2). Außerdem ist $\mathcal{N}(\tilde{z}^{(i)}_k; 0, S^{(i)}_k)$ eine Gaußsche Wahrscheinlichkeitsdichtefunktion von $\tilde{z}^{(i)}_k$ mit dem Mittelwert 0 und $Cov = S^{(i)}_k$. Zum Hintergrund sei kurz auf die Berechnung der Likelihood verwiesen, wie z.B. in [A 28] beschrieben: $L_z(x) := p(Z|x)$, wobei Z die Messung bezeichnet und x einen Zufallsparameter. Des weiteren sei $L_k(x) := p(Z^k|x)$ und $L(x) = p(z|x) = \mathcal{N}(x; x; \sigma^2)$.

Weiterhin wird die Mode Probability berechnet [A 33]:

$$\mu^{(i)}_k = \frac{\mu^{(i)}_{k|k-1}L^{(i)}_k}{\sum_j \mu^{(j)}_{k|k-1}L^{(j)}_k} \tag{111}$$

Hierbei bezeichne $\mu_{k|k-1}^{(i)}$ die Predicted Mode Probability des Filters i und $L_k^{(i)}$ die Model Likelihood des Filters i.

Im vierten Schritt (**4. Schätzung des Ausgangs**) werden die Schätzungen aller Filter fusioniert, um eine gemeinsame Zustandsschätzung der gesamten Filterbank zu ermitteln (estimate fusion).

Hierzu wird die Overall Estimate bestimmt [A 33]:

$$\hat{x}_{k|k} = \sum_i \hat{x}_{k|k}^{(i)} \mu_k^{(i)} \tag{112}$$

Hierbei bezeichne $\hat{x}_{k|k}^{(i)}$ den Updated State von Filter i und $\mu_k^{(i)}$ dessen Mode Probability.

Zu guter Letzt wird die Overall Covariance bestimmt [A 33]:

$$P_{k|k} = \sum_i [P_{k|k}^{(i)} + (\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)})(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)})^T] \mu_k^{(i)} \tag{113}$$

Hierbei bezeichne $P_{k|k}^{(i)}$ die Updated Covariance und $\hat{x}_{k|k}$ den Overall Estimate. Weiterhin ist $\hat{x}_{k|k}^{(i)}$ der Updated State des Filters i und $\mu_k^{(i)}$ dessen Mode Probability.

Für weitere Hintergründe, Details und Abwandlungen sei auf [A 33, 28] verwiesen.

3 Stand der Technik und Forschung

In diesem Kapitel wird ein Überblick über den Stand der Technik gegeben. Dabei wird sowohl auf etablierte Standards als auch auf Forschungsansätze eingegangen.

Zunächst werden grundlegende Beiträge zur Zeitsynchronisation erläutert. Danach liegt der Fokus auf Ansätzen, die eher auf drahtgebundene Szenarien abzielen. Im Anschluss wird auf drahtlose Ansätze eingegangen. In diesem Bereich gab es in den letzten Jahren besonders intensive Forschungsbestrebungen. Es sei angemerkt, dass die Möglichkeit von drahtlosen Ansätzen für Echtzeit-IIoT-Kommunikation noch eine aktuelle Forschungsfrage [A 7] ist und der Fokus der drahtlosen Arbeiten meist auf Energieeffizienz und nicht auf Präzision liegt, wie es im IIoT typischerweise der Fall ist. Trotzdem ist es aus Sicht des Autors dieser Forschungsarbeit interessant, die verwendeten Algorithmen zu untersuchen, da diese sich ggf. in den drahtgebundenen bzw. IIoT-Bereich übertragen lassen. Schlussendlich wird auf Ansätze eingegangen, die sich vorwiegend mit der Takt- bzw. Frequenzstabilisierung befassen.

Es sei angemerkt, dass in diesem Kapitel kein Vergleich mit den in dieser Arbeit vorgestellten Ansätzen gezogen wird. Dieser Vergleich wird in jedem der folgenden Kapitel in einem separaten SOTA-Abschnitt (State-of-the-Art) vorgenommen. Der Fokus dieses Kapitels liegt hingegen auf der Herausstellung des Forschungsbedarfs.

Basierend auf diesem Kapitel wurde ein umfangreiches Survey Paper im MDPI IoT Journal im November 2020 veröffentlicht [B 1]. Das Paper wurde als Cover Story der 2. Ausgabe im Jahr 2020 ausgewählt und ist als Open Access Artikel frei für die Öffentlichkeit zugänglich.

3.1 Grundlegende Beiträge

Zunächst werden einige grundlegende Beiträge bzgl. der Zeitsynchronisation erläutert. An dieser Stelle wird auch auf die theoretischen und mathematischen Grundlagen verwiesen.

3.1.1 Zeitsynchronisation in Rechnernetzen

In [A 21] untersuchen die Autoren, bis zu welchen theoretischen Grenzen eine Taktsynchronisation überhaupt möglich ist. Dabei betrachten sie Takte mit einer konstanten, aber nicht unbedingt identischen Geschwindigkeit. Jeder Takt ist durch Offset und Drift in Bezug auf eine Referenzzeit gekennzeichnet. Um fundamentale Genauigkeitsgrenzen zu bestimmen, werden die unbekannten Parameter (Offset und Drift) als konstant und zeitinvariant betrachtet. Die wichtigste Erkenntnis ist, dass die Drift korrekt bestimmt werden kann, aber die Bestimmung aller Offsets und Verbindungsverzögerungen nicht ohne weitere Vereinfachung möglich ist.

Wu et al. [A 9] untersuchen umfangreich die Zeitsynchronisation in WSNs (Wireless Sensor Networks) mittels Austausch von Zeitstempeln. Insbesondere betonen die Autoren die Tatsache, dass die Zeit eines Gerätes aufgrund der Ungenauigkeit des Oszillators (z.B. Phaseninstabilität, Drift) von der idealen Zeit abweicht, selbst wenn das Gerät anfangs perfekt synchronisiert war, also Offset und Drift Null sind. Demzufolge ändert sich der Offset mit

der Zeit und es muss periodisch eine (Re-)Synchronisation durchgeführt werden, um die Taktparameter (Offset und Drift) anzupassen. Darüber hinaus wird die netzwerkweite Synchronisation erörtert. Im Gegensatz zur paarweisen Synchronisation zwischen benachbarter Knoten verwendet die netzwerkweite Synchronisation eine hierarchische Baum-Struktur. Die Synchronisation wird dann zwischen benachbarten Ebenen dieser Hierarchie durchgeführt. Hervorhebenswert ist auch die sehr gute Übersicht über allgemeine Ansätze und Schätzverfahren für unterschiedliche Delay-Verteilungen.

In [A 36] untersuchen Zucca et al. den Zusammenhang zwischen mathematischen Taktmodellen, die auf stochastischen Prozessen beruhen, und praktischen Stabilitätsmaßen wie der Allan-Varianz. Es können exakte Berechnungen bspw. für die Allan-Varianz angegeben werden und hierbei werden keine Vereinfachungsannahmen getroffen.

3.1.2 Synchronisation in komplexen Netzwerken

Da sich diese Arbeit mit Zeitsynchronisation in Rechnernetzen beschäftigt, sei an dieser Stelle auch auf das übergeordnete Forschungsgebiet verwiesen: die Synchronisation in (komplexen) Netzen gekoppelter Oszillatoren [A 37, 38, 39, 40, 41, 42, 43]. Häufig kommt hier das Kuramoto-Modell zum Einsatz. Dieses Forschungsgebiet bildet die theoretische Grundlage für verschiedenste Anwendungen und Phänomene wie das Schwingen von Glühwürmchen, Grillen und Zikaden [A 39] und ist stark verwandt mit Consensus-Verfahren bzw. Verhalten (Vögel, Fische, Meinungsbildung in Social Media) [A 39], findet aber auch Anwendung in Stromversorgungsnetzen [A 38, 39], der Regelungstechnik [A 40, 44, 45, 46, 47] oder eben der Zeitsynchronisation in Rechnernetzen [A 39, 48, 49, 40, 50]. Die Arbeiten befassen sich allgemein mit der Synchronisation von Phase und Frequenz, allerdings ist die Zeitsynchronisation genau genommen ein Spezialfall der Phasensynchronisation. Diese eher theoretischen Arbeiten beschäftigen sich allgemein mit abstrakten Fragen: "bei welcher (linearen) Kopplung ist eine Synchronisation noch möglich?"[A 38, 37, 42] oder "wie ist, bei partieller Synchronisation, der Netzwerkgraph optimal zu partitionieren?"[A 39]. Im Gegensatz dazu werden in dieser Arbeit konkrete Verfahren bzgl. ihrer Anwendbarkeit auf reale Computernetzwerke evaluiert.

3.1.3 Alternativen zur Synchronisation

In [A 51, 52] wird ein Vorschlag für eine Asynchrone Echtzeit-Kommunikation vorgestellt. Der Ansatz ist kompatibel zum aktuellen Draft von 802.1 Qcr der IEEE (Institute of Electrical and Electronics Engineers). Untersucht wird der Ansatz mit dem Riverbed Simulator. Die Autoren betonen, dass typische TDMA-Kommunikation bzw. synchrone Planung des Netzwerkverkehrs (z.B. Time-Aware Traffic Shaping bei TSN (Time-Sensitive Networking) auf Basis von IEEE 802.1 Qbv) eine sehr genaue Synchronisation benötigt. Daher untersuchen die Autoren asynchrones Traffic Shaping (nach IEEE 802.1 Qcr). Es findet hierfür keine netzwerkweite Synchronisation statt. Nur die lokale Zeitbasis des Gerätes findet Anwendung. Verschiedene Scheduler-Algorithmen werden untersucht und deren Leistungsfähigkeit anhand der erreichbaren Ende-zu-Ende-Verzögerung, der Buffer-Verwendung und der Paketverluste bewertet. Allerdings muss der Algorithmus konfiguriert werden, da sonst Pakete verloren gehen. Obwohl es sich um einen vielversprechenden Ansatz handelt, fehlt

der wichtige direkte Vergleich zum Time-Aware Traffic Shaping, wie es bspw. bei TSN zum Einsatz kommt.

3.2 Vergleichsstudien (Surveys)

Im folgenden Abschnitt wird, der Vollständigkeit halber, ein Überblick über Survey-Artikel im Bereich der Zeitsynchronisation gegeben. Hierbei sei angemerkt, dass in den letzten Jahren verstärkt auf dem Gebiet der WSNs geforscht wurde, daher wurden die Surveys in WSN-basierte und nicht-WSN-basierte unterteilt. Obwohl diese Arbeit sich mit drahtgebundenen Netzwerken beschäftigt, wird dennoch auf WSN-basierte Arbeiten verwiesen, da sich teilweise deren Konzepte (Kommunikationsmuster, Schätzverfahren) auch auf drahtgebundene Netze übertragen lassen und umgekehrt. Bspw. teilen sowohl WSNs als auch das IIoT die Anforderung der hochgradigen Skalierbarkeit. Des Weiteren sei darauf verwiesen, dass jene Surveys, die ihren Fokus auf Echtzeitkommunikation legen, das Thema der Zeitsynchronisation häufig ausklammern (siehe z.B. [A 6] und [B 8]), da beide Themen orthogonal sind. Zeitsynchronisation ist zwar eine Voraussetzung für jede TDMA-basierte Echzeitkommunikation [A 6], kann aber meist getrennt von dieser betrachtet werden.

3.2.1 Studien ohne Fokus auf WSNs (Wireless Sensor Networks)

[A 53] befasst sich mit der Zeitsynchronisation in vehicular ad-hoc networks (VANETs). Synchronisation ist in VANETs entscheidend für zeit-sensitive Applikationen (Koordination, Kommunikation, Sicherheit). In VANETs gibt es die Anforderung der Lokalisierung und es müssen zeitkritische Nachrichten bzw. Warnungen übertragen werden. Dieses Szenario ist sehr herausfordernd durch die Forderung nach geringer Latenz und hoher Robustheit sowie die dem Szenario immanente hohe Dynamik. Demzufolge unterscheiden sich die Anforderungen in VANETs durchaus von klassischen Synchronisationsanforderungen. Allerdings werden in [A 53] lediglich bestehende Synchronisationsansätze aus anderen Bereichen bzgl. ihrer Anwendbarkeit auf VANETs evaluiert und es werden keine algorithmisch neuen Ansätze vorgestellt.

[A 7] stellt eine Übersicht über Synchronisationsansätze auf Basis von IEEE 802.11 (WLAN) vor und zielt dabei insbesondere auf Echtzeitanwendungen und industrielle Netze. Es wird eine gute Übersicht gegeben über die erreichbare Synchronisationsgenauigkeit - sowohl für verschiedene bestehende Protokolle als auch für einige Forschungsansätze. Es wird allerdings auch betont, dass Sicherheit und Zuverlässigkeit, beispielsweise gegenüber Störsendern, in drahtlosen Netzwerken noch immer als offene Forschungsfrage verbleibt.

Die Autoren in [A 54] geben eine gute Übersicht über Zeitsynchronisationsverfahren in Packet-Switched Networks. Es wird die Relevanz der Synchronisation für eine Vielzahl von Anwendungen betont und Protokolle sowie grundlegende Methoden untersucht. Beachtenswert ist auch die Übersicht über die von verschiedenen Anwendungen benötigten Synchronisationsgenauigkeiten für drahtlose Kommunikation (je nach Anwendung $200\ ns - 12,8\ \mu s$), industrielle Automatisierung (je nach Anwendung $1\ \mu s - 1\ ms$) und Smart Grids (je nach Anwendung $1\ \mu s - 100\ ms$). Allerdings liegt der Fokus der Autoren auf etablierten Proto-

kollen und nicht auf neuen Forschungsansätzen. Speziell fokussiert wird auch auf Security-Aspekte.

3.2.2 Studien mit Fokus WSNs (Wireless Sensor Networks)

Über die bereits genannten Surveys hinaus gibt es eine Vielzahl von Surveys, die sich mit der Synchronisation in WSNs befassen [A 55, 56, 57, 58, 59, 60]

In [A 55] wird ein Überblick über Algorithmen der künstlichen Intelligenz (KI) in WSNs gegeben und es werden auch einige interessante KI-Verfahren zur Synchronisation vorgestellt. Allerdings ist Synchronisation nur ein sehr kleiner Teilaspekt und konkret erreichbare Genauigkeiten werden nicht genannt.

[A 56] führt eine Kategorisierung von WSN-Synchronisationsansätzen bzgl. ihrer Eigenschaften (strukturell, technisch, globale Ziele) durch, welche noch weiter verfeinert wird. Es findet eine kurze Beschreibung grundlegender Protokolle und deren Abwandlungen statt. Weiterhin wird ein allgemeines Modell zur Analyse bestehender Ansätze sowie zur Entwicklung neuer Ansätze vorgeschlagen.

Die Autoren in [A 57] betonen, dass durch die besonderen Anforderungen von WSNs (Beschränkung von Energie, Rechenleistung, Speicher und Bandbreite), typische Ansätze wie NTP und GPS für WSNs ungeeignet sind. Es werden grundlegende Algorithmen und Protokolle für WSNs beschrieben.

[A 58] betont die Relevanz von Zeitsynchronisation in WSNs für Datenerfassung und Energieeffizienz und stellt eine knappe Analyse von Protokollen und Forschungsansätzen vor. Synchronisationsgenauigkeiten werden für die Ansätze RBS (Reference-Broadcast Synchronization), TPSN (Timing-sync Protocol for Sensor Networks), FTSP (Flooding Time Synchronization Protocol), GTSP (Gradient Time Synchronization Protocol) und PulseSync angegeben. Als offene Forschungsfrage wird die Notwendigkeit der Entwicklung einer neuen Klasse von sicheren Protokollen herausgestellt. Darüber hinaus sollten diese auch skalierbar, unabhängig von der Topologie, schnell konvergierend, energieeffizient und weniger anwendungsspezifisch sein. Der Fokus liegt auch hier komplett auf WSNs.

Zu den frühesten Surveys, die sich mit Synchronisation in WSNs befassen, gehören [A 59] und [A 60]. In [A 59] wird darauf verwiesen, dass vor allem die Fortschritte im Bereich der MEMS (Micro-Electro-Mechanical Systems) starken Einfluss auf die Entwicklung des WSN-Bereiches hatten. Als wichtige Anwendung der Synchronisation wird die Datenfusion genannt. Die Ansätze werden in Bezug auf verschiedene Eigenschaften untersucht (Genauigkeit, Kosten, Komplexität), es werden Entwurfsüberlegungen zur Wahl bestehender Protokolle bzw. dem Entwurf neuer Protokolle sowie ein Framework für deren Bewertung vorgestellt. Im Rahmen dessen wird eine detaillierte Übersicht über eine Vielzahl von Ansätzen inkl. Detailerklärungen und Synchronisationsgenauigkeiten vorgestellt. In [A 60] wird auf die Zeitsynchronisation in WSNs bzw., allgemeiner formuliert, in Multi-hop-Ad-hoc-Netzen eingegangen. Neben der Wichtigkeit der Zeitsynchronisation in WSNs werden deren spezielle Eigenschaften betont (begrenzte Energie, Speicher, Rechenleistung, Bandbreite und hohe Dichte). Folgerichtig sind traditionelle Ansätze für WSNs eher ungeeignet und die WSN-Forschung hat eine Vielzahl neuer Ansätze vorgestellt. Es wird das Problem der Synchroni-

sation betrachtet, sowie auf den Synchronisationsbedarf eingegangen. Des Weiteren werden WSN-Ansätze im Detail erläutert und Genauigkeitsangaben für RBS und TPSN gemacht.

3.3 Methodik zur Bewertung der Ansätze

In den folgenden Abschnitten wird auf konkrete Synchronisationsverfahren eingegangen. Um die einzelnen Verfahren vergleichen zu können, werden verschiedene qualitative und quantitative Eigenschaften betrachtet.

Die wichtigste Frage in diesem Kapitel lautet: mit welchem Schätzverfahren können unter welchen Bedingungen welche Genauigkeiten erzielt werden. Die Intention dahinter ist, besonders untersuchenswerte Schätzverfahren zu identifizieren, da diese z.B. bereits im drahtlosen Bereich gute Ergebnisse erzielen, im drahtgebundenen Bereich aber noch nicht untersucht wurden (und umgekehrt).

Die Eigenschaften wurden ausgehend von dieser Fragestellung gewählt.

Quantitativ: Welche Synchronisationsgenauigkeit wurde erreicht?

Um einen quantitativen Vergleich der Verfahren durchführen zu können, ist es wichtig, die erreichbare Genauigkeit zu betrachten. Es sei allerdings angemerkt, dass diese von vielen Faktoren beeinflusst wird, wie den verwendeten Geräten (z.B. Zeitstempelaufnahme in HW (Hardware) oder SW (Software) [A 26], Taktstabilität [A 9]) und den Bedingungen im Netzwerk (z.B. spezialisierte Switches [A 5], Hintergrund-Traffic bzw. die hieraus resultierende Netzwerkverzögerung [A 61]). Die Genauigkeiten aus unterschiedlichen Publikationen sind also nur bedingt miteinander vergleichbar. Um die Informationen der einzelnen Publikationen möglichst original wiederzugeben, wurden zwar die Genauigkeiten so exakt wie möglich angegeben, jedoch sind diese eher als Größenordnungen zu sehen. Sinnvoll ist aus Sicht des Autors auch eine Unterteilung in Bereiche: ms-Bereich ($1\ s - 1\ ms$), us-Bereich ($1\ ms - 1\ \mu s$), ns-Bereich ($1\ \mu s - 1\ ns$) sowie ps-Bereich ($1\ ns - 1\ ps$). Dies wird auch in standardisieren Protokollen so gehandhabt (bspw. in [A 62, 63, 64]).

Qualitativ: Ist der Ansatz kompatibel zu einem Standard? Im besten Falle stellt ein Ansatz gar keine Anforderungen an die HW der Switches und Endgeräte. HW-Zeitstempel führen aber im Allgemeinen zu deutlich besseren Genauigkeiten als SW-Zeitstempel [A 26]. Falls ein Ansatz solche Anforderungen an die HW stellt, dann sollte er bestenfalls standardkompatibel sein.

Qualitativ: Welches Schätzverfahren kam zum Einsatz? Da in diesem Überblick der Fokus insbesondere auf den Schätzverfahren liegt, wurde dieses, sofern möglich, für jeden Ansatz angegeben.

Qualitativ: Welche Bedingungen und Einschränkungen gibt es?

Bei Einschränkungen kann es sich zum Beispiel um HW-Anforderungen oder wichtige Nachteile des Verfahrens handeln. Auch die Frage, wie die Genauigkeitsergebnisse gewonnen wurden (analytisch, simulativ oder experimentell) ist entscheidend.

Analytische Ergebnisse sind meist sehr exakt, für diese müssen aber meist starke Vereinfachungen getroffen werden, was die Aussagekraft der Ergebnisse mindert.

Simulationen sind zwar meist weniger stark vereinfacht als analytische Modelle, jedoch müssen dennoch immer Vereinfachungen und Annahmen getroffen werden. Auch simulative Ergebnisse haben nicht uneingeschränkte Genauigkeit. Der wichtigste Vorteil von Simulationen ist, dass sehr viele Experimente in unterschiedlichen Szenarien (bspw. Topologien, Geräteklassen, Verzögerungswahrscheinlichkeiten) durchgeführt werden können. Demzufolge können nach einer sorgfältigen Simulation häufig differenzierte und umfangreichere Aussagen getroffen werden (bspw. für welche Szenarien der Ansatz besonders geeignet ist).

Experimentell, auf realen Geräten in einem Testbed, gewonnene Ergebnisse haben den Nachteil, dass die Aussagen nur für dieses konkrete Szenario Gültigkeit haben. Eine Verallgemeinerung bzw. Aussage über die Gültigkeit in anderen Szenarien ist nur eingeschränkt möglich. Außerdem müssen alle Parameter wie z.B. die Genauigkeit gemessen werden. Die hierbei entstehenden Messungenauigkeiten erlauben es häufig nicht, ganz genaue Aussagen zu treffen. Vorteil von experimentell gewonnenen Ergebnissen ist, dass gezeigt wird, dass der Ansatz sich auch wirklich implementieren lässt (Proof of Concept, Machbarkeitsnachweis), sowie, dass keine Vereinfachungsannahmen getroffen werden müssen.

Da sowohl analytisch, simulativ als auch experimentell gewonnene Ergebnisse jeweils unterschiedliche Vor- und Nachteile haben, werden in einigen Arbeiten, insb. in qualitativ hochwertigen, sowohl Simulationen als auch Experimente durchgeführt.

Einschränkung Als Einschränkung ist zu sagen, dass nicht in jeder Publikation genaue Angaben zur Standardkompatibilität, Synchronisationsgenauigkeit, dem Schätzverfahren oder den Bedingungen gemacht wurden. In den tabellarischen Übersichten wird hierfür der Eintrag ”NA” verwendet. Dieser ist von dem Zeichen ”-” zu unterscheiden, was für *nicht vorhanden* steht (z.B. weil gar kein Schätzer zum Einsatz kommt).

3.4 Protokolle zur Zeitsynchronisation

In diesem Abschnitt wird auf bestehende Protokolle zur Zeitsynchronisation eingegangen. Diese beziehen sich vorwiegend auf drahtgebundene Netze. Tabelle 3.1 zeigt hierzu eine tabellarische Übersicht.

NTP (Network Time Protocol) [A 62] ist ein Standard (RFC 5905), verwendet die RTT (Round-Trip Time) als Schätzer und hat eine Genauigkeit von ca. 1 ms. Weiterhin ist es das im Internet am häufigsten verwendete Synchronisationsprotokoll. Es handelt sich bei NTP um ein reines SW-Protokoll, so dass keine Änderungen auf den unteren Netzwerkschichten benötigt werden. NTP-Clients senden zu bestimmten Zeitpunkten Pakete an den NTP-Server und warten auf die Antwort. Aus dem Zeitpunkt des Eintreffens der Antwort kann die Paket-RTT (Round Trip Time) berechnet werden. Daraus wiederum lässt sich die Latenz unter der Annahme symmetrischer Paketverzögerungen schätzen. Im Internet kann mit diesem Protokoll eine Genauigkeit von ca. 1 ms erreicht werden. Auf Netzwerkebene gibt es eine hierarchische Struktur teilnehmender Server. Der Vorteil ist, dass sich durch mehrere Server große Netze effizienter synchronisieren lassen und die Auslastung des Servers re-

Tabelle 3.1: Übersicht über Protokolle zur Zeitsynchronisation. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (MW = Mittelwert, RTT = Round Trip Time, PLL = Phase-Locked Loop, HW = Hardware, ppm = Parts Per Million)

Ansatz	SK	Schätzer	Genauigkeit	Bed./Einsch.
NTP [A 62]	Ja	RTT	$1\ ms$	-
PTP [A 65, 63, 66]	Ja	RTT	$< 1\ \mu s$	Custom-HW
gPTP [A 64, 67]	Ja	RTT	$< 1\ \mu s$	Custom-HW
PTCP [A 68, 69]	Ja	RTT	$< 1\ \mu s$	Custom-HW
PTCP-KF [A 70]	Ja	KF	$20 - 200\ ns$	Eval. mit Matlab
SyncE [A 71]	Ja	Offset: MW Freq.: PLL	Freq.: $4\ ppm$	Custom-HW
SyncE + Offset [A 72]	Nein	Offset: MW Freq.: PLL	NA Freq.: $4\ ppm$	Custom-HW

duziert wird. Allerdings verschlechtert sich die Genauigkeit aufgrund der Fehlerausbreitung entlang der Hierarchie. NTP ist jedoch z.B. nicht für Echtzeit-IIoT-Anwendungen geeignet, da es sich um einen rein SW-basierten Ansatz handelt, dessen Genauigkeit abnimmt, wenn Verzögerungsschwankungen auftreten [A 61].

Das in IEEE 1588 spezifizierte Precision Time Protocol (PTP) [A 66, 63, 65] ist ein Standard. Es verwendet die RTT bzw. eine einfache Summe als Schätzer und erreicht eine Genauigkeit unterhalb von $1us$. Allerdings benötigt PTP spezialisierte HW (mindestens auf den Endknoten). Da es mittlerweile drei Versionen von PTP gibt, sollen diese hier kurz eingeordnet werden. Als erste Version wurde PTPv1 1588-2002 vorgestellt [A 65]. Die nächste Version, PTPv2 bzw. 1588-2008 [A 63] hat allerdings keinerlei Kompatibilität zu PTPv1. Die neueste Version, 1588-2019 bzw. PTPv2.1, ist hingegen vollständig abwärtskompatibel zur PTPv2. PTP wurde entwickelt, um eine präzise Synchronisation zwischen Geräten zu ermöglichen. Es liegt sowohl als SW- als auch als HW-Version [3.1] vor. Die PTP-Synchronisation beginnt mit der Bestimmung, welches der Geräte die stabilste und genaueste Zeit hat. Es wird mit Hilfe des BMCA (Best Master Clock Algorithm) der Knoten mit der Referenzzeit (Grandmaster Clock) ausgewählt. Anschließend werden Referenzzeiten ausgehend vom Master an die Slaves gesendet, die diese mit ihrer eigenen Zeit vergleichen. In gleichmäßigen Intervallen antworten Slaves mit ihren eigenen Zeiten auf den Master. Aus den Zeitstempeln in den Nachrichten kann die Verzögerung vom Master zum Slave und die Verzögerung vom Slave zum Master bestimmt werden, um die Slaves synchronisieren zu können. Die HW-Variante erreicht im besten Fall eine Genauigkeit im Bereich von Nanosekunden. Die vorhandene SW-Version nutzt Zeitstempel aus der Anwendungsschicht und erreicht mit einer HW-unterstützten Referenzzeit eine Genauigkeit in der Größenordnung von Mikrosekun-

[3.1] Bspw. integriert in die SoCs (Systems on a Chip) der AM335x Sitara Serie von Texas Instruments oder dem TMAC (Tri-Mode Ethernet Media Access Controller) IP-Core (Intellectual Property Core) der Firma Xilinx für FPGAs (Field Programmable Gate Arrays).

den. Von allen Synchronisationsprotokollen, die HW-Zeitstempel verwenden, ist PTP das am häufigsten verwendete. Folglich nutzte das IEEE-TSN-Gremium PTP, um gPTP (generalized Precision Time Protocol) als TSN-Substandard abzuleiten. Es wird bei PTP jedoch davon ausgegangen, dass die Netzwerkverzögerung konstant und symmetrisch ist. Daher verringern Verzögerungsvariationen die Genauigkeit von PTP, wie bspw. in [A 22] gemessen und vom Autoren dieser Arbeit in [B 5] simulativ untersucht wurde. Solche Variationen können auftreten, wenn Switches PTP nicht auf dem MAC Layer (Media Access Control) unterstützen. Gute Beschreibungen von PTP und Genauigkeitsuntersuchungen finden sich bswp. in [A 73, 1, 74].

Ein weiterer Standard ist gPTP (Generalized Precision Time Protocol) bzw. IEEE 802.1AS [A 64, 67]. Es verwendet die RTT als Schätzer und erreicht eine Genauigkeit unterhalb von $1\ \mu s$. Allerdings benötigt gPTP spezialisierte HW. Im Gegensatz zu PTP fordert es diese nicht nur auf den Endknoten, sondern auch auf den Switches. [A 75, 76, 77, 78] geben einen guten Überblick über den gPTP-Standard. Dieser ist Teil einer Reihe von Standards, die ursprünglich von der Task Group AVB (Audio/Video Bridging) entwickelt wurden. Im Jahr 2012 wurde diese Gruppe in TSN umbenannt. Das Ziel der TSN-Gruppe ist die Entwicklung eines offenen Standards für die Ethernet-basierte Echtzeitkommunikation. Insbesondere liegt der Fokus auf der exakten Zeitsteuerung und Synchronisierung, der Reservierung von Ressourcen und der Gestaltung des Datenverkehrs sowie dem Einreihen in die Queues und der Datenweiterleitung. Grundsätzlich basiert die Synchronisation von gPTP auf PTP. Es wird eine modifizierte Version von PTPs BMCA verwendet und eine Genauigkeit von unter einer $1\ \mu s$ relativ zur Referenzzeit erreicht, diese ist aber auf 7 Hops begrenzt [A 64]. Jeder Port eines sogenannten Time-Sensitive Systems misst die Verzögerung zu seinen Nachbarn auf eine Weise, die ähnlich zu PTP ist. Als Time-Sensitive System wird eine Bridge oder eine End-Station bezeichnet, welche die Anforderungen von IEEE 802.1AS erfüllt. Alle Knoten im Netzwerk (Bridges und End-Stations) müssen solche Time-Sensitive Systems sein. Bei gPTP handelt es sich um ein sogenanntes PTP-Profil. Demzufolge kann eine PTP-Implementierung mit gPTP kompatibel sein, falls sie das gPTP-Profil unterstützt. Gefordert ist dies aber nicht, denn eine PTP-Implementierung gilt als standardkonform, wenn sie ein Default-Profil von PTP unterstützt. Die Unterstützung weiterer PTP-Profile (z.B. gPTP) ist optional. Im Gegensatz zu PTP ist gPTP auch für drahtlose Netze spezifiziert [A 64].

PTCP (Precision Transparent Clock Protocol) [A 68, 69] ist ein weiterer Standard. Es verwendet die RTT als Schätzer und erreicht eine Genauigkeit unterhalb von $1\ \mu s$. Allerdings benötigt PTCP spezialisierte HW auf Switches und Endknoten. PTCP wird verwendet für die Synchronisation im Echtzeit-Ethernet-Protokoll PROFINET und ist, genau wie PROFINET, in IEC 61784-2 [A 68] standardisiert. Auch PTCP basiert auf PTP. Wichtigster Unterschied ist die Verwendung von „Transparent Clocks“. Im Gegensatz zu PTP werden nicht die direkten Nachbarn synchronisiert (entweder Endknoten und Switch oder zwei Switches), sondern nur die Endknoten mit dem Master. Dies wird erreicht, indem die Switches die Zeit zwischen Empfang des Ethernet Frames am Eingangsport bis zum Senden des Frames am Ausgangsport kompensieren (mittels Anpassung des Zeitstempels im Paket). Diese sogenannten „Transparent Clocks“ haben den Vorteil, dass die Synchronisationsgenauigkeit auch über viele Hops hoch bleibt. In [A 70] wird PTPC mit einem Kalman Filter als Schätzer erweitert. Die Standardkompatibilität bleibt hierbei erhalten.

Als weiteres Protokoll ist SyncE (Synchronous Ethernet) bzw. ITU-T G.8262 [A 71] zu nennen. Hierbei steht ITU-T für den Telecommunication Standardization Sector der ITU (International Telegraph Union). Bei SyncE handelt es sich eigentlich um ein Echtzeit-Ethernet-Protokoll. Genau genommen bietet es aber auch eine Synchronisationsfunktionalität für die Frequenz bzw. Drift. Für die Frequenz-Synchronisation wird das Taktsignal auf dem PHY (Physical) Layer mit übertragen und die Geräte synchronisieren sich hierauf mittels einer PLL (Phase-Locked Loop). Die erreichbare Frequenzgenauigkeit liegt bei +/- 4 ppm. Der Offset wird durch SyncE nicht korrigiert. In [A 72] wird jedoch die Erweiterung von SyncE um eine solche Offset-Korrektur vorgeschlagen. Das Verfahren ist dann zwar nicht mehr kompatibel zu SyncE, allerdings wurde es als Erweiterungsvorschlag an die ITU-T eingereicht.

3.5 Forschungsansätze zur Zeitsynchronisation

Im folgenden Abschnitt werden Forschungsansätze untersucht, die bisher nicht in einem Standard niedergeschrieben wurden. Hierbei wird zunächst auf Modifikationen der sehr ähnlichen Protokolle PTP bzw. gPTP eingegangen, da insb. für diese Protokolle einige Verbesserungen untersucht wurden. Danach wird auf (vorwiegend) drahtgebundene Ansätze eingegangen und im Anschluss auf (vorwiegend) drahtlose. Diese Trennung zwischen drahtgebunden und drahtlos ist zwar sinnvoll aufgrund der unterschiedlichen Anforderungen in beiden Bereichen, allerdings ist sie nicht komplett trennscharf, da einige Ansätze auf beide Bereiche abzielen – daher vorwiegend drahtgebunden bzw. drahtlos.

3.5.1 PTP/gPTP Modifikationen

In diesem Abschnitt werden Forschungsansätze untersucht, die Modifikationen von PTP oder gPTP darstellen. Tabelle 3.2 zeigt hierzu eine tabellarische Übersicht.

Der in [A 79] von Exel et al. vorgestellte Ansatz ist standardkonform zu PTP, verwendet keinen Schätzer und erreicht in einem Testbed eine Genauigkeit von 10-100 ns. Es werden verschiedene Maßnahmen analysiert, um asymmetrische Verzögerungen mit IEEE 1588 PTP zu verringern (ohne Protokolländerungen und zusätzliche Nachrichten). Der Autor schlägt vor, die Zeitstempel des Referenzknotens an jedem Ausgangs- und Eingangsport durch den Gerätetreiber mittels ISR (Interrupt Service Routine) zu korrigieren. Der vorgestellte SW-Ansatz kann nicht die Präzision von HW-Zeitstempeln erreichen, aber er kann den Offset in einem WLAN-Netzwerk über einen Hop nahezu eliminieren.

Die Autoren in [A 80] stellen einen Ansatz für die drahtlose Synchronisation auf PTP-Basis vor. Der Ansatz ist allerdings nicht standardkonform zu PTP, da dieser Standard kein WLAN unterstützt. Es wird ein PI-Regler als Schätzer eingesetzt und eine Genauigkeit von 40-350 ns erreicht. Die Untersuchungen wurden mittels Simulation und Testbed durchgeführt. Es wird die Wichtigkeit des Taktreglers (Clock Servo) für die Genauigkeit betont und dessen Verhalten untersucht. Insbesondere analysiert wird der Einfluss von Fehlerquellen (ungenaue bzw. SW-Zeitstempel, Oszillatorinstabilitäten).

In [A 26] stellen Giorgi et. al. einen Ansatz vor, der PTP und Kalman-Filterung kombiniert (PTP-Kalman), um die durch verschiedene Unsicherheiten verursachten Fehler zu kompen-

Tabelle 3.2: Übersicht über Modifikationen von PTP oder gPTP. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (KF = Kalman-Filter, NA = Keine Angabe, PLL = Phase Locked Loop, SAGE = Space Altering Generalized Expectation-Maximization, RTT = Round Trip Time, PI = Proportional–Integral (Regler), ISR = Interrupt Service Routine, HW = Hardware, SW = Software, CHW = Custom Hardware)

Ansatz	**SK**	**Schätzer**	**Genauigkeit**	**Bed./Einsch.**
[A 79]	Ja (PTP)	Keiner (ISR)	$10 - 100\ ns$	Testbed
[A 80]	Nein	PI	$40 - 350\ ns$	Sim + Testbed
PTP-Kalman [A 26]	Ja (PTP)	KF	NA	Sim
[A 4]	Ja (PTP)	RTT	$20\ \mu s$ (SW) $0,01 - 1\ \mu s$ (HW)	Analytisch
[A 81]	NA	KF + PI	$25\ \mu s$	Testbed
[A 82]	Nein	RTT	$< 1\ ns$	Analytisch + Sim
White Rabbit [A 83, 13, 84]	Ja (PTP)	Offset: RTT Freq.: PLL	$200\ ps$	Testbed (CHW)
[A 85]	NA	SAGE	NA	Sim
[A 86]	ja	KF	$0,1 - 1\ \mu s$	Analytisch + Sim
[A 87]	Ja (PTP)	KF	ca. $30\ ns$	Sim
ReversePTP [A 88, 89, 90]	Ja	RTT	ca. $8\ ms$ (SW)	Testbed

sieren. Der Ansatz ist kompatibel zu PTP und verwendet ein Kalman-Filter als Schätzer. Untersucht wurde der Ansatz mittels numerischer Simulation, allerdings wurde keine Genauigkeit angegeben sondern lediglich die Standardabweichung von Offset und Drift. Die Autoren betonen, dass die Genauigkeit der Schätzung von Drift und Offset, die Stabilität der Slave-Zeit sowie die Intervalle des Zeitstempelaustausches die Synchronisationsgenauigkeit beeinflussen. Daher analysieren die Autoren die Auswirkungen und das Zusammenspiel dieser Faktoren bei PTP. Ziel ist es, aufzuzeigen, wie sich diese auf den Entwurf eines PTP-Synchronisationsschemas auswirken können. Die Analyse basiert auf einem Simulationsmodell mit Zustandsvariablen, mit denen bestimmte Aspekte des Taktverhaltens modelliert werden können. Das Kalman-Filter wird für die Verbesserung der Schätzungen von Offset und Drift verwendet. Ein Nachteil von PTP-Kalman ist, dass diese Unsicherheiten (modelliert als Rauschen) für die Einstellung des Kalman-Filters von vornherein bekannt sein müssen, um Präzision und Stabilität des Kalman-Filters sicherzustellen. Darüber hinaus sind Kalman-Filter nur für gaußverteilte Unsicherheiten optimal. Dies sind Hauptprobleme in realistischen Szenarien, da die Verzögerung einem selbstähnlichen Verhalten folgt [A 91] und Taktnichtlinearitäten mit der Temperatur korrelieren [A 92]. Taktnichtlinearität bedeutet in diesem Falle, dass die Taktfrequenz nicht konstant ist und somit die Zeit nicht mehr einer linearen Funktion folgt. Verzögerungsvariationen wurden allerdings auch nicht betrachtet.

In [A 4] schlagen die Autoren einen Ansatz für ein PTP-Profil für Anwendungen in der Energieversorgung vor (IEEE PC37.238). Der Ansatz ist kompatibel zu PTP und verwendet, wie PTP, die RTT bzw. eine einfache Summe als Schätzer. In einer analytischen Untersuchung werden Genauigkeiten im Bereich von $10\ ns$ bis $1\ \mu s$ für HW-Zeitstempel, bzw. ca. $20\ \mu s$ für SW-Zeitstempel festgestellt. Zu betonen ist auch, dass ein Testaufbau zur Bestimmung der Genauigkeit auf realen Geräten vorgestellt wird, der sich speziell an Energieversorgungsanwendungen orientiert.

In [A 81] stellen die Autoren einen drahtlosen Synchronisationsansatz auf PTP-Basis vor. Über dessen Standardkompatibilität wird nichts gesagt. Allerdings ist der Ansatz vermutlich nicht kompatibel zu PTP, da PTP nicht für drahtlose Netze spezifiziert ist. Als Schätzverfahren wird eine Kombination aus Kalman-Filter und PI-Regler vorgeschlagen. Es wird in einem realen Testbed, bestehend aus Boards vom Typ Mica2Dot Mote, eine Genauigkeit von $25\ \mu s$ erreicht. Die Implementierung ist komplett in SW und kommt ohne HW-Zeitstempel aus. Die Autoren betonen jedoch, dass der Ansatz weiterer Verbesserungen bzgl. Fehlertoleranz und Determinismus bedarf.

In [A 82] werden Ansätze vorgestellt, um die Anlaufzeit von gPTP um das bis zu 40-fache zu beschleunigen, wobei die Genauigkeit, den Autoren zufolge, nahezu gleich bleibe. Die Autoren beziehen sich auf Netzwerke in Fahrzeugen, betonen hier die Relevanz von Zeitsynchronisation für sicherheitskritische Funktionen und insbesondere die Bedeutung einer kurzen Anlaufzeit, um das Fahrzeug möglichst schnell in einen betriebsbereiten Zustand zu versetzen. Zur Beschleunigung der Anlaufzeit werden unterschiedlich viele Zwischenschritte von gPTP weggelassen, wodurch die Ansätze nicht mehr kompatibel zu gPTP sind. Als Schätzer wird die RTT verwendet, in einigen Varianten entfällt dieser Schritt aber auch komplett. Untersucht wurden die verschiedenen Varianten mittels mathematischer Analyse sowie Simulation. Die Genauigkeit wird im Bereich unterhalb von $1\ ns$ angegeben. Inwiefern diese Genauigkeit sich aber unter harschen Bedingungen (bspw. starke Verzögerungsschwankun-

gen) erreichen lässt, muss in Frage gestellt werden - insbesondere beim kompletten Wegfall der Verzögerungsmessung mittels RTT.

Einer der bedeutendsten Ansätze auf PTP-Basis ist das White Rabbit Projekt [A 83, 13, 84], das von der Europäischen Organisation für Kernforschung (CERN) zusammen mit verschiedenen Universitäten entwickelt wurde und am CERN auch eingesetzt wird. Das White Rabbit Projekt verbindet verschiedene Standards: PTP [A 63], IEEE 802.1Q (zur Priorisierung) [A 93] und Gigabit Ethernet (allerdings nur auf Basis von Lichtwellenleitern). Als Schätzer wird, wie bei PTP, die RTT bzw. eine Summe verwendet. Die Frequenz-Synchronisation findet auf Basis der Übertragungssymbole auf dem PHY Layer mittels PLL statt (Layer 1 Syntonization). Als erreichbare Genauigkeit werden $200\ ps$ angegeben und dies wurde auch in einem realen Testbed untersucht. Der größte Nachteil von White Rabbit ist allerdings, dass sehr teure Spezial-HW sowohl auf den Endknoten als auch auf den Switches benötigt wird.

Die Autoren in [A 85] stellen einen Ansatz vor, um PTP robuster gegenüber asymmetrischen Verzögerungen zu machen. Es wird ein auf dem Erwartungs-Maximierungs-Algorithmus (EM) basierendes Schätzverfahren vorgeschlagen. Die erreichbare Genauigkeit wird nicht genau angegeben, sondern nur eine normalisierte Form der quadratischen Mittelwertabweichung (RMSE). Dass es sich nicht um den normalen RMSE handelt, macht die Einordnung der Genauigkeit schwierig, diese sollte aber im oder unterhalb des μs-Bereiches liegen. Der Ansatz wurde mittels numerischer Simulation untersucht. Hierfür wurden empirisch ermittelte Verzögerungswahrscheinlichkeiten verwendet. Die Autoren betonen, dass die Synchronisation, also die Kompensation von Drift und Offset, als statistisches Schätzproblem betrachtet werden kann. Zunächst formulieren sie eine untere Schranke für den Schätzfehler, unter der Annahme mehrerer Pfade zwischen Master und Slave sowie bekannter Verteilungsfunktionen der Queuing-Verzögerungen. Da diese Informationen in der Realität nicht vorhanden sind, kann ein Schätzverfahren niemals unterhalb dieser Schranke liegen. Im Anschluss wird ein Schätzschema, mittels einer Zusammensetzung mehrerer Gaußverteilungen unter Nutzung des SAGE-Algorithmus (Space Altering Generalized Expectation-Maximization) vorgestellt. Bei SAGE handelt es sich um eine Form des EM. Die Untersuchungen zeigen, dass das vorgestellte Schätzschema sehr nah an den theoretischen Grenzen liegt.

[A 86] analysiert Einflüsse, die sich negativ auf die Genauigkeit von PTP und gPTP auswirken können. Des Weiteren werden Gegenmaßnahmen und ein eigener Ansatz vorgestellt. Der vorgestellte Ansatz ist standardkompatibel, verwendet ein Kalman-Filter als Schätzer und erreicht eine Genauigkeit im Bereich von $100\ ns$ bis $1\ \mu s$, unter der Verwendung von HW-Zeitstempeln. Die Untersuchungen basieren sowohl auf theoretischen Betrachtungen als auch auf Simulationen. Besonders hervorgehoben wird ein Vorschlag für nahtlose Redundanz und somit höhere Zuverlässigkeit. Hierfür sollen mehrere Zeitquellen (diese können z.B. via GPS synchronisiert sein) über unabhängige Pfade mit dem Slave verbunden sein. Auf dem Slave selbst läuft dann ein Regler, um die unterschiedlichen Zeitsignale miteinander zu verbinden. Der Ausfall einer Zeitquelle oder eines Pfades kann somit nahtlos kompensiert werden.

In [A 87] wird von Fontanelli et al. die Synchronisation in industriellen Netzwerken untersucht. Als Problem wird die Anforderung der genauen Synchronisation auch in Netzen mit großer Ausbreitung und langen Linientopologien herausgestellt. Auch PTPv2 (IEEE1588-

2008) kann dies nicht lösen. Als Lösungsansatz wird PTPv2 um ein Kalman-Filter erweitert, um Offset und Drift zu schätzen. Im Gegensatz zu anderen Arbeiten liegt der Fokus insbesondere auf langen Linientopologien.

Mizrahi et al. stellen ReversePTP vor [A 88, 89, 90], das vom SDN-Paradigma (Software-Defined Networking) inspiriert ist. ReversePTP basiert auf PTP und ist auch als PTP-Profil definiert. Allerdings ist ReversePTP vom Nachrichtenfluss her invertiert. Alle Knoten (Switches) im Netzwerk senden ihre Zeitinformationen periodisch an einen einzelnen Controller. Der Controller kümmert sich quasi als Gehirn um alle Berechnungen. Damit ist ReversePTP flexibel und programmierbar nach dem SDN-Paradigma. In einem Testbed mit 34 Knoten wird ReversePTP mit der PTP-Implementierung PTPd verglichen. Beide erreichen eine vergleichbare Genauigkeit von ca. 8 ms (SW-Implementierung).

3.5.2 Drahtgebundene Ansätze

Im diesen Abschnitt werden Forschungsansätze untersucht, die vorwiegend drahtgebundene Szenarien untersuchen. Tabelle 3.3 zeigt hierzu eine tabellarische Übersicht.

Im Rahmen einer Kooperation zwischen dem Institut für Angewandte Mikroelektronik und Datentechnik der Universität Rostock und dem Kernfusionsexperiment Wendelstein 7-X (W7-X) [A 94], [B 4] ist das Trigger-Time-Event System (TTE) entstanden. TTE hat zwar eine Vielzahl an Funktionen, die wichtigste ist jedoch die Zeitsynchronisation. Seit der Inbetriebnahme W7-X im Jahr 2015 ist TTE im Einsatz und liefert die Zeitbasis für die Untersuchungen und Experimente. TTE ist nicht standardkompatibel, verwendet als Schätzverfahren die RTT bzw. eine Summe und erreicht Genauigkeiten von ca. 10-100 ns. Für die Frequenz-Synchronisation (Syntonization) kommt eine PLL in Verbindung mit einer Manchester-Kodierung auf dem PHY Layer zum Einsatz (Layer 1 Syntonization). Die Genauigkeit von TTE wurde in Laborversuchen untersucht (Testbed). Allerdings benötigt auch TTE sehr teure Spezial-HW sowohl auf den Endknoten als auch auf den Switches.

In [A 95] wird ein Ansatz vorgestellt, der auf dem bei EtherCat eingesetzten Synchronisationsansatz DC (Distributed Clock) basiert. DC wiederum basiert auf PTP. Der vorgestellte Ansatz selbst ist aber nicht standardkompatibel. Es wird kein neues Schätzverfahren verwendet, vielmehr werden HW-Verbesserungen für das Aufnehmen von Zeitstempeln und das Verfahren zur Verzögerungsmessung (innerhalb des Gerätes und auf der Leitung) vorgestellt. Der Ansatz wird in einem Testbed untersucht und die erreichte Genauigkeit kann von 36 ns (ohne HW-Verbesserungen) auf 11 ns verbessert werden. Der Ansatz benötigt HW-Zeitstempel und Anpassungen auf dem PHY bzw. MAC Layer. Die Autoren betonen, dass ungenaue Zeitstempel und statistische Schwankungen bzw. Asymmetrien der Verzögerungen die Synchronisationsgenauigkeit beeinflussen. Bei Verfahren, die komplett in HW umgesetzt werden, haben folgerichtig HW-Asymmetrien den größten Einfluss. Daher schlagen die Autoren Anpassungen bei der Aufnahme der Zeitstempel und der Verzögerungsmessung vor. Hierfür werden Signale des MII (Media Independent Interface) verwendet, das sich auf dem PHY Layer befindet. Die MII-Signale werden zum Zählen der Taktzyklen verwendet, für die das Paket auf dem Gerät verbleibt. Die Zeitstempel werden um die entsprechenden Zeiten korrigiert.

Tabelle 3.3: Übersicht über Forschungsansätze, die vorwiegend drahtgebundene Szenarien untersuchen. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (NA = Keine Angabe, KF = Kalman-Filter, EF = exponentielle Filterung, KH = konvexe Hüllen, ADT = Averaged Time Differences, DSR = Direct-Skew Removal, SLW = Sliding-Window, MW = Mittelwert, TS = Zeitstempel, GT = Gaussian Traffic, ST = Self-Similar Traffic, RTT = Round Trip Time, LP = Lineare Optimierung), PI = Proportional–Integral (Regler), SW = Software, HW = Hardware, CHW = Custom Hardware, SVM = Support Vector Machine, TB = Testbed)

Ansatz	**SK**	**Schätzer**	**Genauigkeit**	**Bed./Einsch.**
[A 51, 52]	Ja (802.1Qcr)	-	-	Sim
TTE [A 94], [B 4]	Nein	RTT	$10\ ns - 100\ ns$	TB (CHW)
[A 95]	Nein	-	$11\ ns$	TB (CHW)
[A 96]	Nein	RTT	$< 7\ \mu s$	Sim (CHW)
DTP [A 22]	Nein	RTT	ca. $25\ ns$	(CHW)
[A 20]	Nein	EF	$5 - 20\ \mu s$	Analytisch + TB
HYGENS [A 5]	Fast (komplett SW)	SVM	ca. $100\ ns$	TB (HW-TS)
[A 97]	Nein	KH	$1 - 1,6\ ms$	Sim
RADclock [A 98, 99]	Fast (PTP)	EF	ca. $10\ \mu s$	TB
[A 100]	Nein	PI + KF	$< 18\ ns$	TB
[A 101]	Nein	KF	ca. $100\ \mu s$	TB
[A 61]	Ja (NTP)	KF/LP/ADT	$< 30\ ms$ (GT) $< 0,5\ s$ (ST,LP) $< 0,75\ s$ (ST,KF) $< 1,2\ s$ (ST,ADT)	Sim + TB
[A 102]	NA	MW/DSR/ SLW/KH	NA	Sim
[A 103]	NA	LP	NA	Sim
[A 104]	Nein	KF	$< 0,5\ \mu s$	Sim
[A 105]	Nein	KF	$< 1\ \mu s$	Sim
[A 106]	NA	KF	ca. $0,1\ \mu s$	Sim
[A 107]	Nein	LP	NA	Analytisch
KaDisSy [A 108, 109]	nein	RTT	$< 4.5\ ms$	Sim + TB

In [A 96] wird ein Synchronisationsansatz für Rechenzentren vorgestellt, wobei der Fokus auf optischen Netzwerken liegt. Der Ansatz ist nicht standardkompatibel, es wird als Schätzverfahren eine Summe bzw. RTT verwendet und eine Genauigkeit von unter $7\ \mu s$ erreicht (mittels Simulation). Synchronisation ist in Rechenzentren für eine ganze Reihe von Anwendungen relevant (z.B. Big-Data-Analyse in Echtzeit, Hochleistungsrechnen und Finanzhandel). Es wird zunächst ein sehr grober Überblick über für Rechenzentren relevante Protokolle gegeben. Im Anschluss wird ein eigener Ansatz vorgestellt. Grundidee ist es, dass keine speziellen Pakete für die Synchronisation verschickt werden, sondern die Zeitstempel on-the-fly in normale Daten-Frames eingefügt werden. Zur Evaluation des Ansatzes im Rahmen einer Simulation kamen verschiedene Traffic-Verteilungen zum Einsatz: Pareto-Verteilung, Gleichverteilung und logarithmische Normalverteilung. Als klare Einschränkung zu nennen ist, dass für das Einfügen on-the-fly immer spezialisierte HW (Anpassungen auf dem MAC Layer) auf allen Switches und Endknoten benötigt wird.

In [A 22] stellen die Autoren einen Ansatz namens Datacenter Time Protocol (DTP) für die Synchronisation in Rechenzentren vor. Der Ansatz ist nicht standardkompatibel, verwendet die RTT bzw. eine Summe als Schätzverfahren und erreicht in einem Testbed eine Genauigkeit von ca. 25 ns für benachbarte Knoten bzw. ca. 150 ns für Rechenzentren mit 6 Hops. Allerdings benötigt DTP Spezial-HW auf Switches und Endknoten. DTP ist ein dezentralisierter Ansatz. Da alle Nachrichten auf dem PHY Layer verschickt werden, findet quasi keine Störung der Kommunikation auf den höheren Schichten statt. Interessanterweise kann für DTP eine obere Genauigkeitsgrenze angegeben werden. Diese berechnet sich aus dem längsten Abstand zwischen zwei Knoten und dem präzisesten Takt im Netzwerk.

Mallada et al. [A 20] schlagen einen Ansatz ohne explizite Schätzung der Drift vor und demonstrieren dessen Überlegenheit gegenüber NTP und IBM CCT [A 110]. Der Ansatz ist nicht standardkompatibel, verwendet eine exponentielle Filterung als Schätzer und erreicht in einem Testbed eine Genauigkeit von ca. $5 - 20\ \mu s$. Des Weiteren wird die Konvergenz des Ansatzes analytisch untersucht. Die verwendeten Zeitstempel basieren auf dem TSC (Time Stamp Counter), der die CPU-Zyklen seit dem letzten Neustart zählt. Die Zeitmessungen basieren auf einem verbesserten Ping-Pong-Mechanismus. Diese werden von jedem Knoten zu jedem seiner Nachbarn durchgeführt. Im Gegensatz zu PTP oder NTP werden Schleifen nicht unterbunden (z.B. mittels Spanning Tree), sondern verbessern die Synchronisationsgenauigkeit sogar. Der vorgeschlagene Algorithmus verwendet den aktuellen Offset sowie eine exponentielle Filterung der vergangenen Offsets. Dies vermeidet das Speichern einer langen Offset-Historie sowie teure Berechnungen. Abgesehen von ihren Vorteilen (effiziente Berechnung ohne lange Historie) ist die exponentielle Filterung anfälliger für Verzögerungsschwankungen als LP-Ansätze (Lineare Optimierung) [A 61].

Als Problem bestehender Ansätze stellen die Autoren in [A 5] heraus, dass NTP für viele Anwendungen nicht genau genug ist und genauere Ansätze wie PTP und DTP spezielle HW benötigen. Daher wird der HUYGENS-Ansatz vorgestellt, der versucht, eine präzise Synchronisation ohne spezielle HW zu erreichen. Hierfür werden verrauschte Zeitstempeldaten zunächst gefiltert und mittel Support Vector Machines (SVMs) verarbeitet. Allerdings wird bei HYGENS ein statischer Referenzwert verwendet, anhand dessen Daten als zu ungenau bewertet und herausgefiltert werden. Es handelt sich um einen reinen SW-Ansatz und damit ist er kompatibel zu Standard-HW. Allerdings benötigt HUYGENS Zugriff auf HW-Zeitstempel.

Der Ansatz verwendet SVMs als Schätzer und erreicht in einem Testbed eine Genauigkeit von ca. 100 ns. Ein Problem ist, dass die Taktfunktion als schrittweise lineare Funktion approximiert wird. Bei starken Temperatur- bzw. Frequenzänderungen (vgl. [A 111]) könnte das zu Problemen führen.

Der RADclock-Ansatz [A 98] ist als Open Source verfügbar, komplett in SW implementiert und robust gegenüber Verzögerungsschwankungen. Während der Ansatz ursprünglich nur NTP-Server als Zeitquelle nutzen konnte und keine Vorteile durch PTP-fähige Geräte hatte (HW-Zeitstempel), stellen die RADclock-Autoren in [A 99] eine (fast) PTP-kompatible Version des Ansatzes vor. Diese kann sowohl SW-Zeitstempel als auch HW-Zeitstempel verarbeiten. Evaluiert wird der Ansatz unter verschiedenen Bedingungen und mit den PTP-Implementierungen PTPd und TimeKeeper verglichen. Der Ansatz ist nicht komplett kompatibel zu PTP. Zwar können PTP Master als Zeitquelle und PTP-HW-Zeitstempel genutzt werden, jedoch wurde z.B. die Multicast SYNC-Nachricht nicht umgesetzt. Als Schätzer kommt eine exponentielle Filterung zum Einsatz, auf Basis derer Drift und Offset angepasst werden. In einem Testbed wurde eine Genauigkeit von bis zu ca. $10\ \mu s$ bestimmt.

In [A 61] wird untersucht, inwiefern NTP um verschiedene zusätzliche Verarbeitungsschritte erweitert werden kann. Alle vorgestellten Ansätze sind komplett kompatibel zu NTP. Als Schätzer werden das Kalman-Filter, LP und ATD (Averaged Time Differences) miteinander verglichen. Die Untersuchungen werden mittels Simulation und Testbed durchgeführt. Als wichtigste Erkenntnis wird festgehalten, dass das Kalman-Filter zwar optimal ist für gaußverteilte Verzögerungen aber nicht für Burst-Traffic. Im Burst-Fall ist LP besser geeignet. ATD ist zwar weniger genau als die beiden anderen Ansätze, dafür aber sehr einfach.

In [A 100] befassen sich die Autoren mit Synchronisation in Echtzeit-Netzen. Es wird betont, dass die Synchronisationsgenauigkeit bei langen Pfaden im Netzwerk sinkt, was ein Problem für Skalierbarkeit von Echtzeitnetzen darstellt. Um dem zu begegnen, wird als Schätzer eine Kombination aus Kalman Filter und PI-Regler vorgeschlagen.

In [A 101] wird betont, dass bestehende Protokolle wie NTP und GPS zwar für viele Szenarien geeignet sind, aber dennoch nicht alle abdecken. So ist GPS bspw. sehr kostenintensiv und funktioniert nicht innerhalb von Gebäuden. Als Lösung wird ein Ansatz vorgeschlagen, der hochauflösende Takte und statistische Methoden verwendet, aber ohne zusätzliche HW-Kosten auskommt. Als Schätzer wird ein Kalman-Filter verwendet. Es wird eine Genauigkeit von ca. $100\ \mu s$ erreicht. Als Einschränkung ist zu sagen, dass der vorgeschlagene Ansatz stark abhängig von der Windows Systemzeit ist.

In [A 102] untersuchen die Autoren die Driftkorrektur bei Ende-zu-Ende-Messungen, was äquivalent zur Synchronisation ist. Sie betonen, dass Knoten normalerweise nicht synchronisiert sind und die Drift daher detektiert, kompensiert und im Anschluss herausgerechnet werden muss. Es werden zwei Offline-Ansätze untersucht: der Mittelwert und die neu vorgestellte Direct-Skew Removal-Technik (DSR). Bei letzterer werden iterativ verschiedene mögliche Werte für die Drift berechnet, bis der bester Wert erreicht wurde. Dieses Vorgehen ist den Autoren zufolge sehr genau. Weiterhin wird betont, dass der Mittelwert schneller berechnet werden kann als bspw. LP und konvexe Hüllen (KH). Allerdings ist der Mittelwert im Allgemeinen weniger genau. Weiterhin werden zwei Online-Ansätze untersucht Sliding-Window-Ansatz und ein kombinierter Ansatz aus Sliding Window und konvexen Hüllen.

In [A 103] wird ein Ansatz zur Korrektur von Drift und Offset mit dem Ziel der Verbesserung von Verzögerungsmessungen vorgestellt, was äquivalent zur Synchronisation ist. Als Schätzer wird LP verwendet. Der Fokus der Autoren liegt auf dem Vergleich zwischen LP und anderen Algorithmen. Es wird festgestellt, dass LP eine Komplexität von O(n) hat und die Fehlerspanne für die Drift unabhängig von deren Betrag ist. Des Weiteren wird mittels Simulation gezeigt, dass LP besser geeignet ist als andere Algorithmen (bspw. lineare Regression).

In [A 104] wird von Giorgi et al. speziell die Relevanz des Takt- bzw. Zeitreglers (Clock Servo) betont. Dieser sei kritisch für die Synchronisationsgenauigkeit, glätte diverse Fehlereinflüsse und solle möglichst energieeffizient sein. Als Ansatz wird ein spezielles Kalman Filter vorgestellt: das Event-based-Kalman-Filter. Dieses ist energieeffizienter als das normale Kalman-Filter und weist auch eine geringere Rechenkomplexität auf.

In [A 105] untersuchen Giorgi et al. die Synchronisation in Multi-Pfad-Netzen. Sie betonen, dass die Synchronisationsgenauigkeit abhängig von den Variationen der Verzögerung und der Pfadsymmetrie ist, sowie dass Multi-Pfad-Synchronisation eine Verbesserung der Robustheit und Genauigkeit verspricht. Der vorgeschlagene Ansatz basiert auf einem Kalman-Filter, das Informationen von verschieden Pfaden verarbeiten kann. Es wird gezeigt, dass der Ansatz Informationen adaptiv verarbeiten und hierbei unterschiedliche Messgenauigkeiten der Zeitinformationen mit einbeziehen kann. Des Weiteren wird festgehalten, dass sich die Kosten der Redundanz (mehrere Pfade) aufgrund der verbesserten Robustheit lohnen.

Die Autoren Giorgi et al. in [A 106] betonen, dass die Synchronisationsgenauigkeit einerseits abhängig ist von der Genauigkeit der Zeitstempel und dies auch schon umfangreich untersucht wurde. Allerdings ist auch die Vertrauenswürdigkeit und Zuverlässigkeit der Referenzzeitquelle wichtig. Die Autoren schlagen einen kombinierten Algorithmus bestehend aus zwei Kalman-Filtern vor. Das erste ist ein spezielles Kalman-Filter, das eine zusätzliche Funktion zur Detektion von Ausreißern besitzt. Das zweite Kalman-Filter dient als Absicherung bspw. falls die Referenzzeitquelle ausfällt. Es wird gezeigt, dass der Ansatz die Robustheit verbessert und eine gute Genauigkeit erreicht.

In [A 97] befassen sich die Autoren mit Zeitsynchronisation bzw. der Korrektur von Zeitfehlern bei Verzögerungsmessungen. Hierbei wird auf das Problem eingegangen, dass die Takte bzw. lokalen Zeiten unterschiedlich schnell sind und daher synchronisiert werden müssen. Als Lösung für dieses Problem werden verschiedene Algorithmen vorgestellt, die alle auf konvexen Hüllen basieren. Des Weiteren wird auf die Vorteile von konvexen Hüllen gegenüber LP und linearer Regression eingegangen. Der Ansatz ist nicht standardkompatibel, den Autoren zufolge aber als Erweiterung oder Ersatz für NTP geeignet. Als Schätzer werden Algorithmen verwendet, die auf konvexen Hüllen basieren. Mittels numerischen Simlatuionen wird eine Genauigkeit von 1-1,6 ms erreicht.

In einer sehr frühen Arbeit befassen sich Leommon et al. mit LP-basierter Synchronisation [A 107]. Der vorgeschlagene Ansatz basiert auf der Idee, dass jeder Knoten die Zeit seiner Nachbarn schätzt. Die Synchronisation wird erreicht, indem lokale Zeitstempel zu den Zeiten der Nachbarn transformiert werden können (transformed times).

In [A 108, 109] stellen Skodzik et al. den KaDisSy-Ansatz für Synchronisation auf Basis des P2P-Protokolls (Peer-to-Peer-Protokolls) Kad vor. Hierbei werden lediglich kleine Änderun-

gen an Kad vorgenommen. Durch Parallelität in der Kommunikation wird die zur Synchronisation erforderliche Zeit signifikant verringert, bei Verringerung der Synchronisationsgenauigkeit. Es wird jedoch betont, dass eine kurze Synchronisationszeit die Genauigkeit wiederum verbessere. Hervorhebenswert ist der komplett dezentrale und P2P-basierte Ansatz zur Orchestration der Synchronisation mit konfigurierbarer Parallelität.

3.5.3 Drahtlose Ansätze

In diesem Abschnitt werden Forschungsansätze untersucht, die vorwiegend drahtlose Szenarien untersuchen. Tabelle 3.4 zeigt hierzu eine tabellarische Übersicht.

Der Ansatz FLIGHT [A 19] basiert auf der Idee, die Frequenz von Leuchtstoffröhren zur Drift-Kompensation zu verwenden, denn das Licht schwingt dabei mit der Hälfte der Netzfrequenz. Mittels Lichtsensoren können sich die Geräte auf diese Frequenz synchronisieren. Die Autoren betonen die Vorteile der hohen Genauigkeit und des niedrigen Energieverbrauchs. In einem Testbed mit Boards vom Typ TelosB Mote wird eine Genauigkeit im us-Bereich erreicht. Obwohl es sich um einen interessanten Ansatz handelt, ist die Drift-Synchronisation relativ einfach im Gegensatz zur Offset-Synchronisation, denn die Drift kann i.A. komplett kompensiert werden [A 21]. Als weitere Einschränkung ist zu nennen, dass der Ansatz Lichtsensoren benötigt und vor allem, dass das Licht mittels Leuchtstofflampen erzeugt werden muss. Daher ist der Ansatz bspw. ungeeignet für Outdoor-Szenarien.

In [A 112] stellen die Autoren einen neuartigen Ansatz für WSNs vor, der ohne explizite Synchronisation auskommt. Sie betonen, dass die Synchronisation von Messdaten entscheidend ist, WSNs aber auch spezielle Anforderungen haben (begrenzte Energie und Ressourcen, Forderung nach hoher Robustheit gegenüber extremen Umgebungsbedingungen). Als leichtgewichtigen Ansatz schlagen sie vor, die Daten zu synchronisieren und nicht die Zeiten der Geräte. Dies führt zu weniger Overhead, da keine Synchronisationsnachrichten verschickt werden müssen. Hierfür wird der Zeitstempel in jedem Messdatenpaket korrigiert. Diese Korrektur wird einfach anhand der Differenz zwischen der Zeit am Sender und der Zeit am Empfänger durchgeführt. Die Verzögerung zwischen beiden wird somit jedoch nicht kompensiert. Der Ansatz wird analytisch und simulativ untersucht und erreicht eine Genauigkeit unterhalb von 1 ms, wobei diese abhängig ist von der Anzahl der Hops, der Drift und der Verzögerung. Es wird festgehalten, dass der Ansatz sinnvoll sein könnte für WSN- bzw. IoT-Szenarien.

Die Autoren in [A 113] stellen einen verteilten, Consensus-basierten [3.2] Ansatz vor. Sie betrachten vor allem zwei Probleme, die in WSNs auftreten können. Einerseits können Sensorknoten fehlerhaft oder bösartig sein und so die Kommunikation stören. Andererseits ist die Kommunikation in WSNs i.A. unzuverlässig, sodass häufig Paketverluste auftreten können. Daher wird ein Consensus-Ansatz vorgestellt und zusätzlich die MSR-Technik (Mean Subsequence Reduce), die zum Herausfiltern von Ausreißern dient. Bei MSR findet eine Sortierung der von den Nachbarknoten empfangenen Zeitinformationen statt. Die j größten Zeiten werden verworfen (falls weniger als j größer sind als die lokale Zeit des Knotens,

[3.2] Bei einem Consensus-Algorithmus versuchen sich verschiedene Teilnehmer auf einen Wert, in diesem Falle eine gemeinsame Zeitbasis, zu einigen (enlg. consensus = Konsens).

Tabelle 3.4: Übersicht über Forschungsansätze, die vorwiegend drahtlose Szenarien untersuchen. Gezeigt wird deren Standardkompatibilität (SK), das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen.(NA = Keine Angabe, CA = Consensus Algorithm, MSR = Mean Subsequence Reduced, LCM = Least Common Multiple, EKF = Extended Kalman Filter, CKF = Custom Kalman Filter, ISR = Interrupt Service Routine, RTT = Round Trip Time, LR = Lineare Regression, DS = De-facto-Standard, KF = Kalman-Filter, TB = Testbed, SW = Software, HW = Hardware, TS = Zeitstempel, MW = Mittelwert)

Ansatz	**SK**	**Schätzer**	**Genauigkeit**	**Bed./Einsch.**
FLIGHT [A 19]	Nein	-	ca. $1\ \mu s$	Testbed
[A 112]	Nein	Differenz	< $1\ ms$	Analytisch + Sim
[A 113]	Nein	CA+MSR	NA	$< 100\ ms$
[A 114]	NA	CA	NA	Analytisch
Pulsar [A 115]	NA	RTT	$< 5\ ns$	Testbed
SPiRT [A 116]	NA	LCM	NA	Sim + Testbed
TKDS [A 117]	NA	KF	NA	Sim
[A 118]	NA	EKF	$0,5 - 34\ ns$	Sim
RBIS [A 119]	Ja (802.11)	Differenz	$0,2 - 3\ \mu s$	Testbed
PulseSync [A 120, 121]	NA	LR	ca. $2\ \mu s$	Sim + Testbed
RBS [A 122]	Ja (DS, 802.11)	LR	$6,29\ \mu s$	Sim + Testbed
FTSP [A 123]	Ja (DS)	LR	$1,48\ \mu s$	Testbed
TPSN [A 124]	NA	RTT	$16,9\ \mu s$	Sim + TB
GTSP [A 125]	Fast (komplett SW)	MW	$4 - 14\ \mu s$	Sim + Testbed (HW-TS + ISR)
WizSync [A 126]	NA	Faltung	ca. $0,12\ ms$	Testbed
[A 127]	NA	Summe	$76\ \mu s$	Sim + Testbed
[A 128]	NA	CKF	NA	Testbed
R-Sync [A 129]	Nein	RTT	ca. $50\ \mu s$	Sim + Testbed

werden alle verworfen). Das gleiche Vorgehen wird für die kleinsten Zeiten durchgeführt. Somit kann eine Synchronisation auch dann noch erreicht werden, wenn bis zu j Knoten im Netzwerk fehlerhaft oder bösartig sind. Allerdings kann der Ansatz nicht auf dünn besetzte Netzwerke angewendet werden. Der Ansatz wird mittels numerischer Simulation untersucht.

Auch in [A 114] stellen die Autoren einen verteilten, Consensus-basierten Ansatz vor. Es wird vor allem dessen schnelle Konvergenz betont, so benötigt der Ansatz d Iterationen für die Drift-Synchronisation und $2 \cdot d$ Iterationen für die Offset-Synchronisation, wobei d der Durchmesser des Netzwerks in Hops ist. Des Weiteren ist der Ansatz effizient bzgl. Rechenkomplexität und Kommunikationsaufwand sowie robust. Der vorgestellte Ansatz ist skalierbarer als andere Consensus-Verfahren, da jeder Knoten nur eigene Zeitinformationen und die der Nachbarn verwendet, es keinen Master-Knoten gibt und die Komplexität des vorgeschlagenen Algorithmus unabhängig von der Netzwerkgröße ist (insbesondere nicht mit dieser ansteigt).

Pulsar [A 115] ist ein Ansatz für eine Plattform, die versucht, Synchronisation im ns-Bereich für drahtlose Echtzeit-Szenarien zu erreichen. Der Bedarf an derart genauer Synchronisation ist gegeben durch Anwendungen wie räumliches Multiplexing und Lokalisierung. Die Pulsar-Plattform nutzt hierfür einen stabilen CSAS-Takt (Chip-Scale Atomic Clock) und einen UWB-Sender (Ultra-WideBand). Der Fokus der Autoren liegt auf der HW-Plattform. Das verwendete Synchronisationschema basiert auf PTP und somit kommt die RTT als Schätzer zum Einsatz. In einem Testbed wird gezeigt, dass der Ansatz eine Genauigkeit unterhalb von 5 ns pro Hop erreichen kann.

Der Ansatz SPiRT (Synchronization through Piggybacked Reference Timestamps) [A 116] greift das Problem auf, dass durch die Synchronisation häufig ein hoher Kommunikations-Overhead entsteht, was nicht optimal ist für WSNs. SPiRT nutzt eine 2-Tier Netzwerkarchitektur, indem das Netzwerk in Cluster eingeteilt wird. Die Cluster Heads synchronisieren sich mit dem Referenzknoten mittels Nachrichtenaustausch. Die Zeitstempel der Referenzzeit werden von den Cluster Heads in deren Nachrichten mit eingefügt, sodass sich auch die Cluster Member synchronisieren können. In einer Evaluation mittels Matlab Simulation und Testbed (Micaz Boards) werden SPiRT und dessen Erweiterung E-SPiRT mit anderen Ansätzen verglichen.

In [A 117] wird der TKDS-Ansatz (Temperature-compensated Kalman-based Distributed Synchronization) vorgeschlagen. TKDS verwendet grundsätzlich die normale Zwei-Wege-Synchronisation. Allerdings wird die Drift anhand ihrer physikalischen Charakteristik modelliert. Durch die Kombination der Schätzungen von Nachbarknoten ist der Ansatz genauer als Ansätze, die auf dem Spanning Tree basieren.

Die Autoren von [A 118] schlagen einen Ansatz zur Positionierung in 5G-Netzen vor, der die Zeitsynchronisation als Nebenprodukt ermöglicht. Sie schlagen eine kombinierte Schätzung von Ankunftszeit (Time of Arrival) und Ankunftsrichtung (Direction of Arrival) vor, die auf einer kaskadierten Struktur von erweiterten Kalman-Filtern (EKF) basiert. In diesem Ansatz werden die Schätzungen auf der Netzwerkinfrastruktur durchgeführt.

In [A 119] wird der Ansatz RBIS (Reference Broadcast Infrastructure Synchronization) vorgestellt. Der Ansatz zielt insbesondere auf die Industrie- und Heimautomatisierung ab. RBIS verwendet herkömmliches IEEE 802.11 Equipment und kann komplett in SW imple-

mentiert werden. Es handelt sich um einen Master/Slave-Ansatz, der Receiver/Receiver-Synchronisation nutzt. Hierbei werden zwei Empfänger derselben Broadcast-Nachricht miteinander synchronisiert. In einem Testbed konnte eine Genauigkeit von unter $3\ \mu s$ erreicht werden.

Elson et al. stellen RBS (Reference-Broadcast Synchronization) [A 122] vor, dass als ein De-facto-Standard für WSN-Synchronisation gilt. Bei RBS senden die Knoten Referenz-Beacons als Broadcasts zu allen ihren Nachbarn. Ein solcher Referenz-Broadcast enthält keinen Zeitstempel, sondern der Empfänger nutzt die Ankunftszeit als Referenzzeitpunkt. Als Schätzer wird die lineare Regression verwendet. Der Ansatz ist mit Standard-802.11-HW einsetzbar. Mittels Simulation und Testbed wird eine Genauigkeit von ca. $6\ \mu s$ erreicht.

Marti et. stellen das FTSP (Flooding Time Synchronization Protocol) [A 123] vor, dass als ein weiterer De-facto-Standard für WSN Synchronisation gilt. FTSP ist speziell zugeschnitten auf Anwendungen, die hohe Genauigkeit in ressourcenbeschränkten drahtlosen Netzwerken erfordern. FTSP benötigt nur wenig Bandbreite und ist robust gegebüber dem Ausfall von Knoten und Verbindungen. Diese Robustheit wird erreicht durch regelmäßiges Fluten des Netzwerks mit Synchronisationsnachrichten und die daraus folgende implizite Anpassung an Veränderungen der Topologie. In einem Testbed, bestehend aus 60 Mica2 Knoten, wird eine Genauigkeit von $1,48\ \mu s$ erreicht. Die Präzision erreicht FTSP durch Mac-Layer-Zeitstempel und Schätzung mittels linearer Regression.

Die Autoren Lenzen et. al. schlagen in [A 120, 121] PulseSync vor, das insb. für große Netzwerke geeignet ist. PulseSync flutet das Netzwerk mit schnellen und kurzen (Nachrichten-)Impulsen. Dies führt zu einer kurzen Initialisierungsphase und einer schnellen Anpassung der Synchronisation an Änderungen der Topologie und der Drift. Als Schätzer wird die Lineare Regression verwendet. Die Autoren geben an, dass sie FTSP, einen De-facto-Standard für WSNs, übertreffen. In einer Evaluation mittels Testbed und Simulation wird eine Genauigkeit von ca. $2\ \mu s$ für 1 Hop und maximal $19\ \mu s$ für 31 Hops erzielt. Darüber hinaus ist PulseSync hinsichtlich der Robustheit suboptimal, da es einen Single Point of Failure (den Referenzknoten) aufweist.

Ganeriwal et al. stellen in [A 124] das TPSN (Timing-sync Protocol for Sensor Networks) für netzwerkweite Zeitsynchronisation in WSNs vor. TPSN besteht aus zwei Schritten. Zunächst wird im Netzwerk eine hierarchische Struktur gebildet. Im Anschluss wird an den Kanten dieser Struktur eine paarweise Synchronisation durchgeführt. Infolgedessen sind alle Knoten im Netzwerk mit einem Referenzknoten synchronisiert. Als Schätzer kommt die RTT bzw. eine Summe zum Einsatz. In einem Testbed (Berkley Motes) wird gezeigt, dass zwischen benachbarten Knoten eine Genauigkeit von unter $20\ \mu s$ erreicht werden kann. Zum Vergleich wird auch RBS im selben Testbed untersucht und im Vergleich erreicht TPSN eine zweifach bessere Genauigkeit. Mittels Simulation wird die gute Skalierbarbeit von TPSN nachgewiesen.

In [A 125] stellen Sommer et al. den GTSP-Ansatz (Gradient Time Synchronization Protocol) für WSNs vor. Den Autoren zufolge erreichen viele Ansätze zwar eine genaue globale Synchronisation, nahe Knoten sind aber häufig nur ungenau synchronisiert. Der vorgeschlagene, komplett dezentrale GTSP-Ansatz versucht daher vor allem eine genaue lokale Synchronisation zu erreichen, während die globale (netzwerkweite) Synchronisation aus-

reichend genau ist. Hierfür sendet jeder Knoten seine Zeitinformationen via Broadcast aus und nutzt die Zeitinformationen seiner Nachbarn für die Anpassung der eigenen Zeit. Im Endeffekt konvergieren benachbarte Knoten zu einer gemeinsamer Zeitbasis. Der Ansatz verwendet keine Baumstruktur und keinen Referenzknoten. Demzufolge ist er sehr robust gegenüber dem Ausfall von Knoten oder Links. Die Anpassung der Zeit nehmen die Knoten durch Mittelwertbildung vor, wobei vorwiegend die Drift angepasst wird. Dies führt zwar zu Problemen, falls zwei Knoten eine stark unterschiedliche Drift aufweisen, dieser Fall wird jedoch abgefangen, indem dann einfach die Drift des Nachbarknotens übernommen wird. Der Ansatz verwendet MAC-Layer-Zeitstempel, die mittels ISR aufgenommen werden. Mittels Simulation und Testbed (Mica2 Boards und TinyOS) wird eine Genauigkeit von ca. $4 - 14\ \mu s$ erreicht.

Hao et al. stellen WizSync [A 126] für die Synchronisation in WSNs vor. Sie betonen, dass bestehende Ansätze zwar eine hohe Genauigkeit bieten, häufig aber eine schlechte Skalierbarkeit und insbesondere hohen Overhead erzeugen. Daher wird ein Ansatz vorgeschlagen, der die bestehende WLAN-Infrastruktur verwendet, indem sich die Sensorgeräte auf die Beacons der WLAN-APs (Access Points) synchronisieren. Aufgrund der hohen Reichweite der APs können auf diese Weise viele Sensoren synchronisiert werden. Der Ansatz wurde auf Basis einer experimentellen Untersuchung der Zeitcharakteristik der Beacons entworfen und mit FTSP verglichen. Die Synchronisation findet mittels Aufnahme und anschließender Faltung des RSS-Signals (Received Signal Strength) statt. In einem Testbed (TelosB Boards, TinyOS) konnte eine Genauigkeit von ca. 0,12 ms erzielt werden.

In [A 127] wird ein Ansatz für die Synchronisation in TSCH-Netzwerken (Time Slotted Channel Hopping) vorgestellt. Den Autoren zufolge ist TSCH wichtig für zuverlässige, Ultra-Low-Power-Drahtloskommunikation und Teil von vielen Standards. In Mesh-Netzwerken sei mit TSCH auch bereits 99,999% Zuverlässigkeit erreicht worden. Demzufolge sei TSCH ein Schlüssel für IIoT. Gleichzeitig ist Zeitsynchronisation essentiell für TSCH. Es wird ein Ansatz zur adaptiven Synchronisation vorgeschlagen: jeder Knoten lernt sein eigenes Wegdriften gegenüber seinen Nachbarn und passt dementsprechend auch seine Synchronisationsperiode an. Des Weiteren wird eine Koordination der Synchronisation in Multi-Hop-Netzwerken vorgeschlagen. Hierbei wird der Kindknoten direkt nach dem Elternknoten synchronisiert, da der Elternknoten zu diesem Zeitpunkt noch nicht weggedriftet und demzufolge sehr genau synchronisiert ist. Grundsätzlich basiert der Ansatz auf der Aufnahme einer Offset-History, aus welcher die Drift bestimmt wird (mittels einer Summe). Der grundlegende Ansatz wurde von den Autoren bereits in [A 130] vorgestellt. Der Ansatz führt zu einer Verbesserung der Genauigkeit sowie Verringerung des Energieverbrauchs. Mittels Simulation wurde eine Genauigkeit von $76\ \mu s$ für ein 3-Hop-Szenario ermittelt. Gleichzeitig konnte die Anzahl der Pakete um 83% reduziert werden. Mit Experimenten wird gezeigt, dass die dem Ansatz eigene Adaption hochgradig geeignet ist um Interoperabilität zu ermöglichen.

In [A 128] schlagen Nilsson et al. einen Ansatz vor, der statistisch robust und für die passive (Einwegkommunikations-) Taktsynchronisation in WSNs geeignet ist. Um Verzögerungen zu messen, werden Nachrichten am Messknoten mit einem Zeitstempel versehen, an einen zentralen Knoten übertragen und dort erneut mit einem Zeitstempel in Bezug auf die Referenzzeit versehen. Für die Schätzung der genauen Zeit wird dann ein spezielles Filter vorgeschlagen. Dieses ist ähnlich zu einem Kalman-Filter, nutzt allerdings eine Heavy-Tailed-

Tabelle 3.5: Übersicht über Forschungsansätze, die insbesondere die Taktstabilisierung untersuchen. Gezeigt wird das verwendete Schätzverfahren, die Genauigkeit sowie Bedingungen und Einschränkungen. (NA = Keine Angabe, LR = lineare Regression, DSC = Direct Skew Compensation, IMM-KF = Interacting-Multiple-Model-Kalman-Filter, RMSE = Root-Mean-Square Error, KF = Kalman-Filter, PI = Proportional–Integral (Regler))

Ansatz	Schätzer	Genauigkeit	Bed./Einsch.
[A 132]	PI	$6,4\ ns$ (RMSE)	Sim
[A 133]	KF	$< 100\ \mu s$ (RMSE)	Sim
TCTS [A 111]	MW	$100 - 200\ \mu s$	Sim
DualSync [A 134]	LR	$< 100\ \mu s$	Sim + Testbed
Yang10 [A 34]	IMM	$< 1\ \mu s$ (RMSE)	Sim
EACS [A 15]	IMM	$1\ \mu s - 1\ ms$	Sim + Testbed
TACSC [A 135]	DSC	External: $15\ \mu s$ Internal: $160\ \mu s$	Analytisch + Sim + Testbed
[A 136]	NA	$0,2\ ms$	Testbed

Likelihood-Funktion für das Update. Demzufolge werden Ausreißer weniger stark gewichtet, was den Ansatz, im Vergleich zu einem normalen Kalman-Filter, robuster gegenüber Ausreißern und somit genauer machen. Der Ansatz wurde auf der Android-Plattform implementiert.

Qiu et al. stellen in [A 129] den robusten und energieeffizienten R-Sync-Ansatz für WSNs und IIoT vor. Als Problem wird herausgestellt, dass viele bestehende Ansätze nicht energieeffizient genug seien. In einer eigenen Vorarbeit wurde zwar der energieeffiziente STETS-Ansatz vorgestellt [A 131] , bei diesem kann es aber dazu kommen, dass nicht alle Knoten synchronisiert werden (isolierte Knoten). Daher wird der R-Sync vorgestellt, der zwei Timer verwendet: einen normalen Timer zur Synchronisation (Sync Timer) und einen weiteren, der dafür sorgt, dass isolierte Knoten eine Synchronisation von sich aus anfragen (Pulling Timer). Der Ansatz wird verglichen mit TPSN, GPA und STETS.

3.6 Ansätze zur Taktstabilisierung

Im Folgenden wird auf Ansätze eingegangen, die insbesondere die Taktstabilisierung untersuchen. Tabelle 3.5 zeigt hierzu eine tabellarische Übersicht.

In [A 132] analysieren Exel et al. die Parametrisierung eines PI-Reglers für die Synchronisation. Sie betonen, dass ein Taktregler (Clock Servo) essentiell für die Synchronisation

ist. Dessen Struktur und Parametrisierung sollten sich dabei an folgenden Anforderungen orientieren: kurze Einschwingzeit, Minimierung des Jitters und das Halten des Offsets unterhalb einer vordefinierten Grenze. Typischerweise werden Addierer-basierte Takte oder VCOs (Voltage Controlled Oscillators) verwendet in Kombination mit PI-Reglern. Die Autoren untersuchen die Aufnahme und Berechnung von Größen, die die Taktregelung beeinflussen (bspw. die Drift). Basierend hierauf wird gezeigt, dass die korrektere Parametrisierung des PI-Reglers essentiell für die Minimierung des Offsets ist.

In [A 133] untersuchen Fontanelli et al. die Synchronisation in industriellen Netzen. Als Problem stellen die Autoren heraus, dass Oszillatoren bei Temperaturschwankungen und mechanischen Schocks bzw. Vibrationen instabil werden und hierdurch die Synchronisationsgenauigkeit verringert wird. Von besonderer Bedeutung ist dies für lange Pfade in industriellen Netzen, da hier die Einflüsse entlang des Pfades akkumuliert werden. Als Lösungsansatz schlagen die Autoren ein modifiziertes Kalman Filter (mit Korrekturfaktor) zur Schätzung des Taktzustands vor. Bzgl. der Zeitstempelgenauigkeit werden die Einflüsse des Quantisierungs-, Taktphasen- und Übertragungsrauschens betrachtet. Das Ergebnis der simulativen Untersuchung sind Design-Richtlinien, mit welchen die Genauigkeit auch in industriellen Netzen unterhalb der geforderten Schranken gehalten werden kann.

In [A 111] untersuchen Schmid et al. die Zeitsynchronisation in WSNs und stellen den TCTS-Ansatz (Temperature Compensated Time Synchronization) vor. Die Autoren postulieren, dass in vielen bestehenden Ansätzen der Einfluss der Temperatur nicht betrachtet wird und daher eine häufige Resynchronisation notwendig ist. Den Autoren zufolge, ist die Synchronisationsgenauigkeit im Allgemeinen begrenzt durch die Quantisierung sowie durch temperaturbedingte Frequenzänderungen. Der vorgeschlagene Lösungsansatz besteht aus zwei Teilen. Während der Kalibrierung werden Drift und Temperatur gemessen und gespeichert. Während der Kompensation wird nur noch die Temperatur gemessen. Daraufhin können zwei Fälle eintreten. Entweder ist der zur aktuellen Temperatur gehörende Driftwert bekannt, dann wird dieser Wert zur Kompensation der lokalen Zeit verwendet, oder der aktuelle Driftwert ist unbekannt. Nur in diesem Fall wird die Drift neu gemessen. Da die Kalibrierung aufgrund von Messfehlern nicht perfekt ist, muss dennoch resynchronisiert werden (jedoch weniger häufig). Hierfür wird ein adaptives Verfahren vorgeschlagen, dass sich am adaptiven Fenster der TCP-Flusskontrolle orientiert. Im Rahmen einer Simulation wird TSTC mit FTSP [A 123] verglichen.

Jin et al. schlagen DualSync [A 134] für die Synchronisation in WSNs vor. Als Problem identifizieren die Autoren, dass die Umgebungsbedingungen die Drift beeinflussen. Dies führt zu häufigen Resynchronisationen und demzufolge hohem Energieverbrauch. Der vorgeschlagene DualSync-Ansatz nutzt sowohl Zeitstempel als auch Spannungs- und Temperaturinformationen, um ein genaues Taktmodell zu erzeugen und infolgedessen die Synchronisationsintervalle exakt abschätzen zu können. Hiefür wechselt DualSync zwischen zwei Phasen. In der Inter-Sync-Phase werden Nachrichten ausgetauscht und Zeitstempel aufgenommen, um die Korrelation zwischen Drift und Umwelteinflüssen (Temperatur und Spannung) zu bestimmen. Dies wird als lineare Regression vereinfacht, um Ressourcen zu sparen. In der Self-Sync-Phase werden nur interne Informationen verarbeitet. Die Untersuchungen zeigen die Vorteile von DualSync bzgl. Genauigkeit und Energieeffizienz.

In [A 34] schlagen Yang et al. einen Ansatz vor, der auf einem Interacting-Multiple-Model-Kalman-Filter (IMM) basiert, um die Drift abzuschätzen. Der Ansatz wird in dieser Arbeit Yang10 genannt. Die Autoren betonen, dass die Drift von Umwelteinflüssen abhängt und es folgerichtig entscheidend ist, diese Dynamik zu betrachten. Ein IMM besteht aus mehreren Kalman-Filtern. Jedes Filter verwendet ein anderes Systemmodell, um die Drift zu schätzen. Das IMM wählt dann adaptiv das Filter aus, welches am besten zu den Driftmessungen passt. Mittels Simulation wird gezeigt, dass der Ansatz eine gute Genauigkeit bei moderater Rechenkomplexität erreicht.

Als Erweiterung zu Yang10 schlägt der selbe Hauptautor [A 15] EACS vor. Hierbei werden zusätzlich Temperaturinformationen zur Driftschätzung verwendet. Es wird betont, dass die Drift mit der Temperatur korreliert und demzufolge nicht stationär ist. Allerdings sind TCXOs (Temperaturkompensierte Oszillatoren) relativ teuer und auch diese weisen eine, wenn auch geringe, Drift auf. Die Korrelation zwischen Drift und Temperatur wird experimentell über einen Zeitraum von 6 Monaten untersucht. Als Ergebnis wird festgehalten, dass beide stark korrelieren. Beim EACS-Ansatz verwendet eines der Kalman-Filter ein Modell mit konstanter Temperatur ($Temp = const$) und das andere ein Modell mit konstanter Temperaturänderung ($\delta Temp/\delta t = const$). In gleicher Weise schätzen zwei Kalman-Filter die Drift, wobei eines ein Modell mit konstanter Drift und das andere ein Modell mit konstanter Driftänderung verwendet. Das IMM berechnet, welches Modell am besten zur Temperaturmessung passt. Das korrespondierende Modell wird dann auch zum Schätzen der Drift verwendet. Dieser Ansatz funktioniert, da Temperatur und Drift stark korrelieren [A 15, 92]. Es wird weiterhin gezeigt, dass die Verwendung von diesen zwei Modellen (konstante Temperatur/Drift und konstante Änderung von Temperatur/Drift) ausreichend ist. Die Ansätze Yang10 und EACS ordnen sich selbst insbesondere in WSN-Szenarien ein, da durch die Driftschätzung die Synchronisationshäufigkeit und somit die Anzahl der Nachrichten bzw. der Energieverbrauch minimiert werden soll.

Auch in [A 135] untersuchen Yang et al. die Synchronisation in WSNs. Als Problem wird sowohl die dynamische Umgebung als auch die Ressourcenbeschränkung von WSNs herausgestellt. Als Lösungsansatz wird ein zweiphasiges System vorgestellt. In der ersten Phase wird der Slave mit einem externen Master synchronisiert und es findet ein normaler Zeitstempelaustausch statt. Neu vorgeschlagen wird eine direkte Drift-Kompensation der Zeitstempel. Den Autoren zufolge ist dies sowohl weniger rechenintensiv als auch genauer. In der zweiten Phase findet eine Selbstkalibrierung ausschließlich auf Basis der Temperaturinformation statt. Dieses Verfahren wird TACSC genannt (Temperature-Assisted Clock Self-Calibration).

In einer sehr kurzen Publikation betonen die Autoren in [A 136], dass Synchronisation zwar sehr wichtig ist für WSNs, der Austausch der Zeitinformationen jedoch auch viel Energie benötigt. Als Lösungsansatz schlagen sie vor, die Temperatur für die Synchronisation mit zu berücksichtigen. Hierfür werden Drift und Offset korrigiert. Nach der Drift-Korrektur kann das Gerät seinen Transceiver ausschalten und somit Energie sparen. Trotz einer Sleep-Zeit von 10 Minuten wird eine Genauigkeit von ca. 0,2 ms erreicht bei deutlich verbesserter Energieeffizienz.

3.7 Zwischenfazit

In diesem Kapitel wird ein Überblick über den Stand der Technik gegeben und die untersuchten Ansätze werden anhand folgender Eigenschaften analysiert: Standardkompatibilität, verwendetes Schätzverfahren, erreichbare Synchronisationsgenauigkeit sowie Bedingungen und Einschränkungen. Zusammenfassend ist Folgendes festzuhalten:

- Das Erreichen hoher Synchronisationsgenauigkeiten ($< 1\ \mu s$) ohne den Einsatz spezialisierter HW verbleibt als offene Forschungsfrage.
- Im drahtlosen Bereich wurden viele Broadcast-basierte Ansätze zur Verbesserung der Skalierbarkeit vorgeschlagen. Broadcasts könnten auch für drahtgebundene Netze interessant sein. Insbesondere für IIoT-Szenarien ist die Skalierbarkeit entscheidend, aufgrund der Vielzahl der zu vernetzenden Geräte.
- Um möglichst wenige Anforderungen an die Plattform zu stellen, sollte ein Ansatz möglichst in SW umgesetzt sein. Dies reduziert die Kosten und verbessert die Zukunftssicherheit. In einigen Ansätzen können dennoch, sofern verfügbar, genaue HW-Zeitstempel verwendet werden. Hervorhebenswert ist, dass dies lediglich eine Anforderung an die Endknoten stellt, nicht jedoch an die Switches.
- Im besten Falle stellt ein Ansatz keine speziellen Anforderungen an die Switches und kompensiert die auf diesen entstandenen Verzögerungen mittels leistungsstarker Schätzer auf den Endknoten. Folglich konzentriert sich diese Forschungsarbeit und insbesondere dieses Kapitel auf Schätzer und deren erreichbare Synchronisationsgenauigkeit.
- Als Schätzer erzielt insbesondere LP sehr gute Ergebnisse. In WSNs kommt stattdessen vor allem LR häufig zum Einsatz. Im Vergleich zu LP weist LR eine noch geringere Rechenkomplexität auf, während die Genauigkeit leicht beeinträchtigt wird. LR wurde bereits ausführlich im drahtlosen Bereich untersucht, könnte jedoch für weitere Untersuchungen im drahtgebundenen Bereich interessant sein.
- Der Fokus dieser Forschungsarbeit wird im Weiteren insbesondere auf LP und LR in drahtgebundenen Szenarien liegen. Allerdings ist bzgl. anderer vielversprechender Schätzverfahren Folgendes festzuhalten:
  - Ein EM-basierter Ansatz wurde als PTP-Erweiterung vorgestellt. EM könnte auch allgemein für die drahtgebundene und drahtlose Synchronisation interessant sein.
  - Einige Arbeiten befassen sich mit KH in drahtgebundenen Netzwerken. Aus Sicht des Autors sind KH ein interessanter Ansatz und könnten weiter untersucht werden.
  - Eine Arbeit schlägt SVMs für die drahtgebundene Synchronisation vor. Aus Sicht des Autors könnten SVMs sowohl für drahtlose Szenarien als auch als Erweiterung bestehender Protokolle interessant sein.

- Bzgl. der Taktstabilisierung ist festzuhalten, dass vor allem die Temperatur großen Einfluss auf die Taktfrequenz hat. Dieser Einfluss lässt sich jedoch durchaus kompensieren, wie in einigen Arbeiten eindrucksvoll gezeigt werden konnte.
- Obwohl es erste Ansätze für Sicherheit (Security) im Zeitsynchronisationsbereich gibt, bleiben sichere Zeitsynchronisationsprotokolle im Allgemeinen eine offene Forschungsfrage [7].

4 PSPI-Sync: Ein Ansatz für präzise, skalierbare und plattformunabhängige Zeitsynchronisation

In diesem Kapitel wird der PSPI-Sync-Ansatz vorgestellt. PSPI-Sync steht hierbei für: **P**recise, **S**calable, and **P**latform **I**ndependent Clock **Sync**hronization. Wie bereits erläutert, ist die Zeitsynchronisation ein wichtiges Thema in drahtgebundenen und drahtlosen Netzwerken, da eine gemeinsame Zeitbasis für koordinierte Aktivitäten von Knoten in verteilten Systemen unerlässlich ist.

PSPI-Sync basiert insbesondere auf einem neuartigen Verfahren zur Verzögerungsbestimmung. Aktuell ergeben sich immer mehr verzögerungssensitive IoT-Anwendungen, z.B. auf den Gebieten der Digitalisierung von Industrieanlagen (IIoT, Smart Factory), der Telemedizin und des autonomen Fahrens. Solche Anwendungen stellen hohe Anforderungen an die Verzögerung. Die gesamte Ende-zu-Ende-Verzögerung zwischen zwei Geräten setzt sich aus mehreren einzelnen Verzögerungen zusammen. Grob lässt sich unterscheiden zwischen der Software-Verzögerung auf der Anwendungsschicht, der Software-Verzögerung im Netzwerk-Stack [4.1], der Hardware-Verzögerung auf dem physikalischen Kommunikationsmedium (z.B. Kupfer, Glasfaser, Luft) und der Hardware-Verzögerung in den Switches. Die Hardware-Verzögerungen sind in drahtgebundenen Netzwerken relativ unveränderlich (konstant), solange es nicht zu hohen Lastsituationen kommt. Im Gegensatz dazu ist die Software-Verzögerung sehr variabel und daher schwer zu bestimmen. Die Software-Verzögerung ist also für verzögerungssensitive Anwendungen von entscheidender Bedeutung, da sie starken Einfluss auf die Ende-zu-Ende-Verzögerung haben kann.

In der Literatur werden zur Bestimmung der Software-Verzögerung verschiedene Untersuchungen anhand von Simulationen sowie praktische Prüfstände mit Spezialhardware vorgeschlagen. Im Gegensatz dazu wird in dieser Forschungsarbeit eine Methode ohne spezielle Hardware vorgestellt. Der vorgestellte Ansatz basiert auf der Messung der Umlaufzeiten zwischen n Gerätepaaren für ein Netzwerk mit n Knoten und der Lösung des resultierenden Gleichungssystems, um die Verzögerung jedes einzelnen Geräts abzuschätzen. Gegenüber dem Stand der Technik weist das vorgeschlagene Verfahren auch einige Vorteile hinsichtlich Skalierbarkeit und Ausfallsicherheit auf.

Basierend auf diesem neuen Ansatz zur Verzögerungsbestimmung wird der PSPI-Sync-Ansatz zur Zeitsynchronisation vorgestellt. Die Grundidee des PSPI-Sync-Ansatzes besteht darin, zunächst alle Verzögerungen in einem Netzwerk abzuschätzen. Anschließend berechnet PSPI-Sync basierend auf diesen Schätzungen die Verzögerung zwischen einem Referenzknoten und allen anderen Knoten im Netzwerk. PSPI-Sync verwendet Broadcast-Nachrichten zur Synchronisation. Unter Verwendung einer FPGA-basierten Messmethode und einer Prototyp-Implementierung in Java wird gezeigt, dass PSPI-Sync eine hohe Synchronisationsgenauigkeit erreicht (ca. $123\ \mu s$). Darüber hinaus ist PSPI-Sync hochgradig skalierbar und plattformunabhängig. Da PSPI-Sync auf der Anwendungsschicht ausgeführt wird, eignet sich der Ansatz theoretisch sowohl für drahtgebundene als auch für drahtlose

[4.1] Genauer gesagt sind insbesondere die unteren Schichten des Netzwerk-Stacks oft auch in Hardware implementiert, z.B. der MAC-Layer bei WLAN. Die hier entstehenden Hardware-Verzögerungen sind aber deutlich konstanter als die Software-Verzögerungen auf den oberen Schichten.

Netzwerke. Die Evaluation fand jedoch lediglich in drahtgebundenen Netzen statt und das nachfolgende Konzept diskutiert auch nur diese, da die Entwicklung drahtloser Echtzeittechnologien noch Gegenstand aktueller Forschungen ist [A 7].

Veröffentlicht wurde das Verfahren zur Verzögerungsbestimmung auf der IEEE IEMCON 2016 [B 12] und der darauf basierende PSPI-Sync-Ansatz auf der IEEE CCNC 2017 [B 11]. Das folgende Kapitel basiert auf diesen Veröffentlichungen, enthält allerdings auch einige bisher unveröffentlichte Inhalte.

4.1 Einleitung und Motivation

Das Network Time Protocol (NTP) und das Precision Time Protocol (PTP) sind etablierte Standards für die Zeitsynchronisation. NTP ist in Software implementiert, aber nicht sehr genau (ca. $1\ ms$) [A 8]. PTP kann eine höhere Genauigkeit erreichen ($< 1\ \mu s$), benötigt hierfür jedoch spezielle Hardware. Da es auf Hardware-Zeitstempeln basiert, ist es in hohem Maße plattformabhängig, um diese Präzision zu erreichen. Software-Implementierungen von PTP weisen viel geringere, mit NTP vergleichbare Genauigkeiten auf [A 74].

Dem PSPI-Sync-Ansatz liegen zwei Überlegungen zugrunde. Die erste Überlegung ist, dass die Schätzung der Verzögerung zwischen zwei Knoten N_0 und N_1 in einem Netzwerk und die Synchronisation dieser Knoten äquivalente Probleme sind. Wenn die Verzögerung t_{delay} zwischen N_0 und N_1 bekannt ist, kann ein Zeitstempel t_0 am Knoten N_0 genommen werden. Dieser kann an den Knoten N_1 gesendet und die lokale Zeit des Knotens N_1 auf den Wert gestellt werden, der $t_0 + t_{delay}$ entspricht. Andererseits kann, wenn zwei Knoten perfekt synchronisiert sind, ein Zeitstempel t_0 vom Knoten N_0 zu Knoten N_1 gesendet und dort ein zweiter Zeitstempel t_1 genommen werden. Die Verzögerung t_{delay} entspricht dann exakt $t_1 - t_0$. Daraus folgt, dass eine präzise Bestimmung der Verzögerung zu einer präzisen Synchronisation führt. Die zweite Überlegung ist, dass verschiedene unbekannte Parameter zur Verzögerung beitragen. Es kann unterschieden werden zwischen Software-Verzögerungen und Hardware-Verzögerungen. Während u.a. [A 137] postuliert, dass die Software-Verzögerungen sehr variabel und demzufolge schwer abzuschätzen sind kann angenommen werden, dass die Hardware-Verzögerungen weniger variabel sind [4.2].

Basierend auf diesen Überlegungen wird in diesem Kapitel der PSPI-Sync-Ansatz vorgestellt, der eine hohe Präzision erreicht, obwohl er in Software implementiert ist, da eine geeignete Schätzung der variablen Verzögerungsanteile vorgenommen wird. Darüber hinaus erreicht PSPI-Sync eine sehr gute Skalierbarkeit, da, unter der Annahme einer zeitinvarianten Verteilung der Verzögerung, die Verzögerung im Netzwerk nur einmal geschätzt werden muss und die Synchronisation sowie jede Resynchronisation Broadcast-Nachrichten verwenden können. Dies ist ein entscheidender Vorteil um den durch die Synchronisationsnachrichten entstehenden Overhead zu minimieren, da jedes Netzwerk nach einer bestimmten Zeitspanne neu synchronisiert werden muss. Diese Zeitspanne ist, aufgrund der unvollkommenen Frequenzstabilität realer Oszillatoren [A 9], abhängig von der Taktgenauigkeit der Geräte [A 49].

[4.2] Insbesondere die Queueing-Verzögerung auf den Switches kann sich jedoch infolge der variablen Netzwerkbelastung im Laufe der Zeit ändern. Diese statischen Variationen lassen sich aber relativ gut kompensieren, wie in den folgenden Kapiteln gezeigt wird.

4.1.1 Verzögerungsbestimmung

Das in diesem Kapitel vorgestellte Verfahren zur Verzögerungsbestimmung ist in seiner Anwendung nicht auf die Zeitsynchronisation beschränkt.

Viele Anwendungen stellen Anforderungen an die Verzögerung, deren Nichteinhaltung jedoch in gewissem Umfang toleriert werden kann (weiche Echtzeitanforderungen). Typische Beispiele für weiche Echtzeitanwendungen sind Musik- bzw. Video-Streaming, Gaming und Voice over IP-Kommunikation. Im Gegensatz dazu muss für einige Anwendungen, wie in automatisierten Industrieanlagen und beim autonomen Fahren eine deterministische Verzögerung (harte Echtzeit) garantiert werden [A 138]. Hier muss also eine Verzögerungsschranke unbedingt eingehalten werden [4.3].

Für harte Echtzeit ist daher wichtig, die Ende-zu-Ende-Verzögerung zwischen den Geräten im Worst Case zu bestimmen. Die Verzögerung zwischen zwei Geräten hat dabei variable und invariante (konstante) Komponenten. Abb. 4.1 zeigt eine schematische Übersicht über die Kommunikation zwischen zwei Geräten und die eingeführten Verzögerungen. Die konstanten Komponenten ergeben sich z.B. aus der Hardware-Verzögerung auf dem Kabel. Die Hardware-Verzögerung kann leicht anhand der Spezifikation der Switches, der Kabellänge und der maximalen Übertragungsdatenraten bestimmt werden.

Im Gegensatz dazu ist die durch die Verarbeitung eines Pakets im Software-Stack verursachte Software-Verzögerung schwerer zu bestimmen. Viele Parameter wie das verwendete Betriebssystem, die Systemauslastung und die ausgeführte Anwendung beeinflussen die Software-Verzögerung. Die Software-Verzögerung ist jedoch für Echtzeitanwendungen von entscheidender Bedeutung, da sie bekannt sein muss, um die Worst-Case-Verzögerung zu bestimmen.

In der Literatur werden verschiedene Simulationsauswertungen sowie praktische Prüfstände mit spezieller Hardware vorgeschlagen. Um die Verzögerung zwischen zwei Geräten zu messen, verwenden die meisten Ansätze die Round-Trip-Time (RTT). Unter Verwendung einer RTT ist es jedoch nur möglich, die Summe $d_0 + d_1$ zu bestimmen, wobei d_0 die Software-Verzögerung von Knoten N_0 und d_1 die Software-Verzögerung von Knoten N_1 ist. Dies gilt selbst unter der wenig realistischen Annahme, dass die Leitungsverzögerung bekannt ist.

Stattdessen wird in dieser Forschungsarbeit ein konzeptionell anderer Ansatz vorgeschlagen, der die Software-Verzögerung aus der Messung der RTTs von n Gerätepaaren für ein Netzwerk mit n Knoten schätzt. Als Folge dessen kann die mittlere Verzögerung geschätzt werden. Darüber hinaus kann die statistische Genauigkeit der Schätzung angegeben werden. Außerdem kann die Software-Verzögerung jedes einzelnen Knotens (also d_0 und d_1) und nicht nur die Summe der Verzögerungen zweier Knoten ($d_0 + d_1$) bestimmt werden.

Die Vorteile des vorgeschlagenen Ansatzes zur Verzögerungsbestimmung sind wie folgt:

[4.3] Es sei angemerkt, dass auch bei weichen Echtzeitanwendungen eine Verletzung der Verzögerungsanforderungen für den Nutzer unangenehm sein kann und sich sehr negativ aus die Dienstgüte auswirken kann. Dennoch sind die Konsequenzen weitaus weniger dramatisch im Vergleich zu harten Echtzeitanwendungen wie einem Air-Bag oder dem Nothalt einer Fabrik bzw. eines Kraftwerks.

- Da es sich um einen algorithmischen und Software-basierten Ansatz handelt, ist keine spezielle Hardware erforderlich. Als Folge können die Verzögerungen mehrerer Geräte in einem Netzwerk vor Ort (im tatsächlichen Anwendungsbereich) jederzeit ohne Vorbereitung abgeschätzt werden.
- Das Verfahren ist robust, da es keinen Single Point of Failure gibt und daher prinzipiell auch für industrielle Szenarien geeignet.
- Das Verfahren ist skalierbar, da es n Schätzungen für n verschiedene Geräte mit unbekannten Verzögerungen benötigt. Außerdem muss die Verzögerung nur einmal geschätzt werden, wenn deren Wahrscheinlichkeitsverteilung zeitinvariant ist.
- Das Verfahren ist relativ genau, da die Verzögerung jedes Geräts separat geschätzt wird. Dies konvergiert schneller als die Schätzung der Summe zweier Verzögerungsverteilungen. Die mittlere Verzögerung wird aus mehreren Werten geschätzt, es besteht also die Möglichkeit ein Konfidenzintervall (vgl. [A 29] S. 514ff) anzugeben oder ggf. Rückschlüsse auf die Wahrscheinlichkeitsverteilung der Verzögerung bzw. die Worst-Case-Verzögerung zu ziehen.

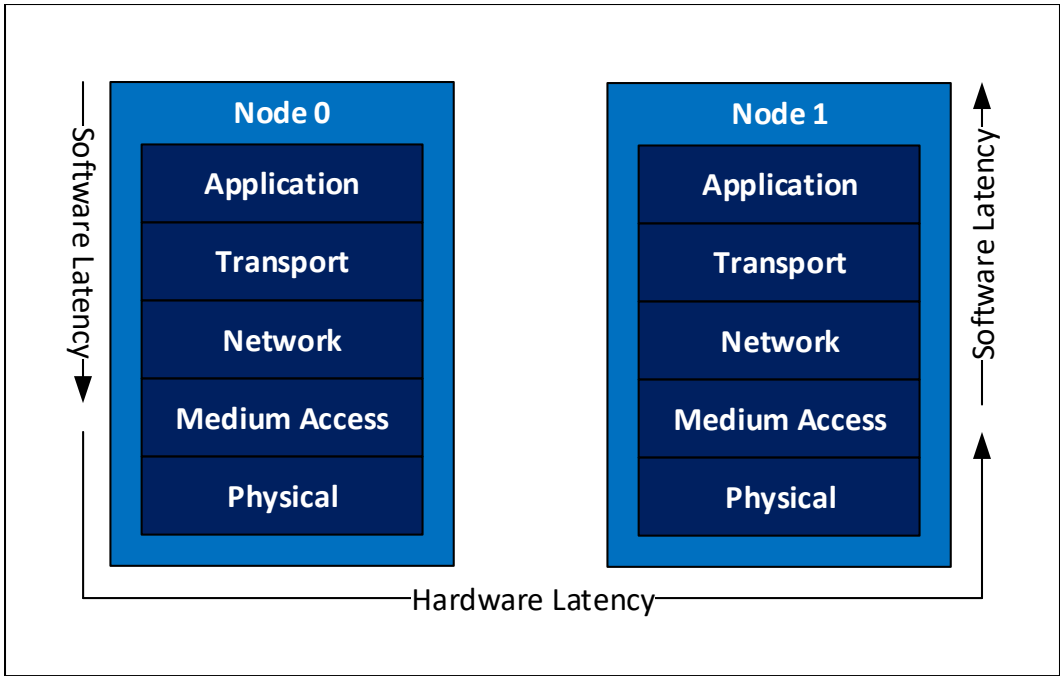


Abbildung 4.1: Verzögerung zwischen zwei Geräten in einem drahtgebundenen Netzwerk.

4.1.2 Synchronisation mittels Broadcasts

Basierend auf der Verzögerungsbestimmung wird der PSPI-Sync-Ansatz zur Broadcast-basierten Synchronisation mit folgendem Ablauf vorgestellt:

1. Ermittlung der Hardware-Verzögerungen im Netzwerk. Diese Verzögerungen können z.B. durch Messungen ermittelt werden. Alternativ können die Hardware-

Verzögerungen aus der Spezifikation der Switches und der Netzwerktopologie ermittelt werden. Letztere lässt sich z.B. automatisiert mittels SNMP (Simple Network Management Protocol) ermitteln [A 139].

2. PSPI-Sync schätzt die Software-Verzögerung jedes Knotens im Netzwerk, indem es Round-Trip-Zeiten (RTTs) zwischen n Knotenpaaren misst für ein Netzwerk aus n Knoten, um ein lösbares Gleichungssystem zu erhalten. In der Folge kann die mittlere Software-Verzögerung geschätzt werden.
3. PSPI-Sync verwendet die geschätzte Verzögerung für eine Synchronisation basierend auf Broadcast-Nachrichten.

Gegenüber vergleichbaren (SW-basierten) Forschungsansätzen und etablierten Standards wie NTP bietet PSPI-Sync folgende Vorteile:

Skalierbarkeit: Wenn mehrere Knoten im Netzwerk dieselbe Software-Verzögerung haben (z.B. viele Sensoren mit identischer Hard- und Software in einem industriellen Netzwerk), kann die Anzahl der Pakete auf ein theoretisches Minimum reduziert werden. Es sind keine weiteren Schätzungen der Verzögerung für die Synchronisation oder die Resynchronisation erforderlich. Stattdessen kann der Referenzknoten, der dynamisch gewählt werden kann, Broadcast-Nachrichten verwenden, um alle anderen Knoten zu synchronisieren. Außerdem muss jeder Knoten nur eine Verzögerung lokal speichern: die Verzögerung zwischen sich selbst und dem Referenzknoten. Der Referenzknoten muss keine Verzögerungen speichern.

Hohe Genauigkeit: Im Vergleich zu anderen Software-basierten Ansätzen erreicht PSPI-Sync eine sehr hohe Genauigkeit der Synchronisation. Im Gegensatz zum Stand der Technik kann PSPI-Sync die Software-Verzögerung jedes einzelnen Knotens bestimmen.

Plattformunabhängigkeit: Der vorgestellte PSPI-Sync-Ansatz ist unabhängig von der physikalischen Schicht (z.B. Ethernet, WLAN), da die Schätzung der Verzögerungen auf der Anwendungsschicht stattfindet. Somit betrachtet PSPI-Sync auch explizit die Software-Verzögerung der höheren Schichten, die für verzögerungssensitive Anwendungen (z.B. Telemedizin) wichtig ist. Außerdem ist keine teure Spezial-Hardware erforderlich.

4.2 Vergleich mit dem Stand der Technik

In diesem Abschnitt wird der PSPI-Sync-Ansatz mit den Stand der Technik verglichen. Da dieser als neuartige Komponenten die vorgestellte Methode zur Verzögerungsbestimmung und die Idee der Broadcast-basierten Synchronisation enthält, wird zunächst der Stand der Technik für die Verzögerungsbestimmung und -reduktion untersucht. Danach wird der PSPI-Sync-Ansatz als Ganzes mit anderen Synchronisationsprotokollen verglichen.

4.2.1 Verzögerungsbestimmung und -reduktion

NTP und PTP sind die etablierten Standard-Synchronisationsprotokolle. Beide verwenden die RTT zwischen zwei Geräten, um die Verzögerung zu bestimmen. Trotzdem kann auf diese Weise selbst im besten Falle [4.4] nur die Summe $d_0 + d_1$ bestimmt werden, wobei d_0

[4.4] Also bei konstanter Gesamtverzögerung und bekannter Hardware-Verzögerung.

die Software-Verzögerung von Knoten N_0 und d_1 die Software-Verzögerung von Knoten N_1 ist.

Der Standard für die Durchführung von Verzögerungsmessungen ist in RFC 2544 [A 140] definiert. Die Autoren geben an, dass die Messungen mit einem Messobjekt und einem Tester durchgeführt werden sollten. Die Verzögerungszeit l ist definiert als $l = t_t - t_r$, wobei der Tester den Zeitstempel t_t nach vollständiger Übertragung der Daten und den Zeitstempel t_r nach vollständigem erneutem Empfang der Daten verwendet. Grundsätzlich entspricht diese Verzögerungsdefinition der RTT und es ist daher auch hier unmöglich, zwischen Hardware- und Software-Verzögerung sowie zwischen der vom Tester eingeführten Verzögerung d_0 und der vom Messobjekt eingeführten Verzögerung d_1 zu unterscheiden. Darüber hinaus muss ein spezielles Testgerät verwendetet werden.

In [A 137] präsentieren die Autoren eine NS3-basierte Simulation sowie praktische Messungen der Software-Verzögerung. Insbesondere werden der Network Interface Controller (NIC) und die New API (NAPI) untersucht. Die Messungen werden mit einem speziellen Lastgenerator und dem Messobjekt durchgeführt.

Emmerich et al. untersuchen die Netzwerkverzögerung von Gaming-Servern in [A 141]. Sie geben an, dass der Hauptteil der Verzögerung durch die Kommunikation über das Internet verursacht wird und dass das Erreichen hoher Puffer-Füllstände (engl. Buffer Bloat) in diesem Szenario zu hohen Verzögerungen führen kann. Sie berücksichtigen die Verzögerungen des Netzwerk-Stacks und schlagen Stacks im User Space vor, um die Verzögerung und den Durchsatz zu optimieren. Die Autoren führen die Messungen mit einem 10 GBit/s Lastgenerator und einem Gaming-Server durch. Darüber hinaus verwenden sie PTP-Zeitstempel eingehender und ausgehender Pakete.

Rotsos et al. präsentieren ein Framework für die Evaluierung von Switching-Hardware in [A 142]. Sie konzentrieren sich hierbei auf Verzögerungsmessungen von OpenFlow-fähigen Switches für Software-Defined Networks (SDNs). Sie stellen ein generisches Software-Framework vor, verwenden jedoch eine dedizierte FPGA-basierte Hardware-Implementierung für die Messungen und keine Standardhardware.

In [A 143] untersuchen Inoue et al. Software-Verzögerungen und stellen eine Hardware-/Software-Implementierung eines Netzwerk-Stacks mit geringer Verzögerung vor. Sie geben an, dass Kontextwechsel und das Kopieren zwischen verschiedenen Puffern eine Verzögerung erzeugen und der Einsatz von Co-Prozessoren diese Verzögerung verringern kann. Darüber hinaus geben sie an, dass Multi-Core-Verarbeitung und -Parallelisierung sich positiv auf den Durchsatz, aber negativ auf die Verzögerung auswirken. Dies bestätigt, dass viele Parameter zur Software-Verzögerung beitragen und diese nicht als konstant angenommen werden kann. Das in [A 143] vorgestellte System erzielt geringe Verzögerungen durch direkten Zugriff auf die Hardware vom User Space aus sowie durch spezialisierte Software. Daher ist ihr Ansatz sehr plattformabhängig.

In [A 144] wird die Ende-zu-Ende-Verzögerung in TCP/IP-Netzwerken analysiert. Die Autoren geben Interrupt-Management, Head-of-Queue-Effekte, Konflikte um Systemressourcen, Kontextwechsel, hohe Prozessorlast und hohe Verkehrslast als Ursachen für Verzögerungsschwankungen an. Für verzögerungssensitive Szenarien schlagen die Autoren vor, einzelne Prozesse auf dedizierten Cores auszuführen. Außerdem wird ein adaptiver Wechsel zwi-

schen Interrupts von der Netzwerkkarte und eine aktive Abfrage durch den Betriebssystem-Kernel empfohlen. Darüber hinaus schlagen sie adaptive Interrupts vor: für Anforderungen mit geringer Verzögerung soll der Interrupt sofort ausgelöst werden. Im Gegensatz dazu kann es für mehrere Pakete einen Interrupt geben, um einen hohen Durchsatz zu erzielen. Dies bestätigt, dass die Software-Verzögerung stark von Anwendung und Implementierung abhängt.

Die Autoren in [A 145] präsentieren eine Analyse des Linux-Netzwerkstacks. Insbesondere konzentrieren sie sich auf die Untersuchung von Verzögerung und Durchsatz. Sie geben an, dass das NAPI einen Kompromiss zwischen Verzögerung und Durchsatz bietet. Eine gemeinsame Optimierung auf niedrige Verzögerung und hohen Durchsatz wird jedoch nicht unterstützt. Es kann allerdings Low-Latency Polling eingesetzt werden, z.B. durch Verwendung des Intel ixgbe-Treibers, was laut Intel zu einer Verzögerungsreduktion von 30% führt.

Gegenüber dem Stand der Technik weist die in dieser Forschungsarbeit vorgeschlagene Methode zur Verzögerungsbestimmung mehrere Verbesserungen auf:

- Sie bietet die Möglichkeit, die Software-Verzögerung eines einzelnen Gerätes zu schätzen, was unmöglich ist, wenn nur eine RTT gemessen wird.
- Die Verzögerung wird nicht als konstant angenommen. Stattdessen können statistische Parameter wie Mittelwert, Standardabweichung und statistische Genauigkeit angegeben werden.
- Der vorgestellte, algorithmische Ansatz führt eine Software-basierte Schätzung ein. Somit ist er plattformunabhängig und es wird keine spezielle Hardware benötigt. Darüber hinaus sind Messungen im realen Anwendungsbereich im Feld möglich.
- Der vorgeschlagene Ansatz bietet eine gute Skalierbarkeit: Wenn eine zeitliche Invarianz der Verzögerungsverteilung angenommen werden kann [4.5], muss die Schätzung für jedes Gerät nur einmal durchgeführt werden .

4.2.2 Zeitsynchronisation

Zunächst soll zur Einordnung von PSPI-Sync auf etablierte Standardprotokolle eingegangen werden. Im Anschluss wird PSPI-Sync mit anderen Forschungsansätzen verglichen. An dieser Stelle sei angemerkt, dass in diesem Abschnitt vorwiegend auf die Unterschiede zwischen den Ansätzen und PSPI-Sync eingegangen wird. Für eine detailliertere Beschreibung und Einordnung der Ansätze sei auf Kapitel 3 verwiesen.

Der Standard PTP [A 63] benötigt spezielle Hardware, um eine hohe Präzision zu erzielen. Als reine Software-Version entspricht dessen Genauigkeit in etwa der von NTP. Die Hauptnachteile der Hardware-Variante sind jedoch, dass diese spezielle PTP-Hardware verfügbar sein muss und Zeitstempel in niedrigeren Schichten verwendet werden, wodurch die durch höhere Schichten verursachte Verzögerung ignoriert wird. Im Gegensatz zum PSPI-Sync-Ansatz unterliegt PTP einigen Hardware- und Software-Einschränkungen und kann daher

[4.5] Bspw. in [A 91] werden umfangreich die statistischen Eigenschaften von Netzwerk-Traffic untersucht. Dieser lässt sich z.B. häufig mit einer zeitinvarianten logarithmischen Normalverteilung beschreiben.

nicht als plattformunabhängig angesehen werden. Darüber hinaus kann nur die Summe der Verzögerungen von zwei Geräten bestimmt werden und es ist unmöglich, zwischen den einzelnen Verzögerungen zu unterscheiden, die zur RTT beitragen.

Das NTP (Network Time Protocol) [A 62] ist das im Internet am häufigsten verwendete Synchronisationsprotokoll. Es handelt sich bei NTP um ein reines Softwareprotokoll, sodass keine Änderungen auf den unteren Netzwerkschichten benötigt werden. Die letztgenannten Nachteile bzgl. der RTT gelten auch für NTP. Weiterhin verschlechtert sich dessen Genauigkeit aufgrund der Fehlerausbreitung entlang der Baumhierarchie.

Grundsätzlich basiert die Synchronisation von gPTP (IEEE 802.1AS) [A 64] auf PTP. Allerdings müssen bei gPTP alle Knoten im Netzwerk (Switches und Endknoten) den gPTP-Standard in Form von spezieller Hardware unterstützen [A 75]. Demzufolge stellt gPTP noch strengere Anforderungen an die Plattform als PTP.

Wie bereits dargestellt, kombiniert der viel beachtete Ansatz PTP-Kalman den Standard PTP und Kalman-Filterung, um die durch verschiedene Unsicherheiten verursachten Fehler zu minimieren [A 26]. Im Vergleich zu PSPI-Sync verbessert die Verwendung von Kalman-Filtern die Synchronisationsfähigkeiten, da sie bei Vorhandensein von bekanntem, gaußverteiltem Messrauschen optimale Schätzungen liefert. Allerdings ist die Verwendung eines Kalman Filters aufgrund möglicher Einschränkungen (z.B. Verfügbarkeit der Gleitkomma-Verarbeitung) möglicherweise nicht immer realisierbar.

Mallada et al. [A 20] schlagen einen Ansatz ohne explizite Schätzung der Drift vor und demonstrieren dessen Überlegenheit gegenüber NTP. Der vorgeschlagene Algorithmus verwendet die aktuellen Driftinformationen sowie eine exponentielle Filterung der vergangenen Offsets. Die Autoren schlagen jedoch einen Algorithmus vor, der Schätzungen zwischen jedem Knoten und den Nachbarn dieses Knotens umfasst. Für ein Netzwerk von n Knoten werden für jede Resynchronisation demzufolge mindestens n Schätzungen eingeführt (vorausgesetzt, jeder Knoten hat nur einen Nachbarn). Im Gegensatz dazu ist die Zeit für Resynchronisationen beim PSPI-Ansatz unabhängig von der Anzahl der Knoten im Netzwerk.

In [A 128] schlagen Nilsson et al. einen Ansatz vor, der statistisch robust, für die passive (Einwegkommunikations-) Taktsynchronisation in drahtlosen Sensornetzwerken (Wireless Sensor Networks, WSNs) geeignet ist und die Heavy-Tailed-Likelihood-Funktion nutzt. Dieser Ansatz eignet sich gut für Messreihen, bei denen Ausreißer ausgeschlossen werden sollen. Es werden jedoch keine Maßnahmen untersucht, um einen zentralen Knoten als Single Point of Failure (SPoF) zu vermeiden.

Weiterhin, schlagen Lenzen et. al. in [A 120] PulseSync vor, das insb. für große Netzwerke geeignet ist. PulseSync flutet das Netzwerk mit schnellen und kurzen Impulsen. Dies führt zu einer kurzen Initialisierungsphase und einer schnellen Anpassung der Synchronisation an Änderungen der Topologie und der Drift. Die Autoren von PulseSync erwähnen allerdings, dass sie die Robustheit ihres Protokolls verbessern müssen, da der Wurzelknoten einen Single Point of Failure darstellt. Dies steht im Gegensatz zu PSPI-Sync. Weiterhin werden Zeitstempel auf der MAC-Schicht verwendet. Daher ist keine Berücksichtigung der durch höhere Schichten verursachten Verzögerung möglich.

Eine wesentliche Verbesserung gegenüber dem Stand der Technik ist die Skalierbarkeit des PSPI-Ansatzes. In den vorhandenen Ansätzen gilt für die Ausführungszeit T die folgende Proportionalität: $T \sim n$. Hierbei bezeichnet n die Anzahl der Knoten im Netzwerk, wenn alle Knoten mit einem Referenzknoten synchronisiert werden sollen. Alternativ gilt $T \sim log(n)$, wenn eine Baumstruktur mit mehreren Referenzknoten existiert. In einer Baumstruktur ist der Fehler jedoch aufgrund der Fehlerausbreitung auch proportional zu $log(n)$. Im Gegensatz dazu skaliert der PSPI-Sync-Ansatz mit m, wobei m die Anzahl der unterschiedlichen Plattformen bezeichnet ($m \leq n$). Daher benötigt PSPI-Sync die Zeit $T \sim m$, um die Software-Verzögerungen aller Plattformen im Netzwerk einmal abzuschätzen. Im Folgenden können Broadcast-Nachrichten für die Synchronisierung und Resynchronisierung verwendet werden. PSPI-Sync braucht also, unter der Annahme, das sich die Netzwerktopologie nicht ändert, für jede (Re-) Synchronisation eine Zeit $T \sim 1$, die unabhängig von der Anzahl der Knoten ist [4.6].

4.3 Der PSPI-Sync-Ansatz

In diesem Abschnitt liegt der Fokus auf dem mathematischen Konzept des PSPI-Sync-Ansatzes. Der vorgeschlagene Synchronisationsalgorithmus basiert auf der Schätzung aller Verzögerungen im Netzwerk. Anschließend kann die Verzögerung zwischen dem Referenzknoten und allen anderen Knoten im Netzwerk ermittelt werden. Der Referenzknoten stellt hierbei keinen Single Point of Failure dar. Bei einem Ausfall des Referenzknotens könnte jeder andere Netzwerkknoten dessen Funktion übernehmen, ähnlich wie beim BMCA (Best Master Clock Algorithm) von PTP.

Da insbesondere die Software-Verzögerung nach [A 137] variabel und schwer abzuschätzen ist, liegt der Fokus auf der Schätzung der Software-Verzögerung. Es wird die Annahme getroffen, dass die Hardware-Verzögerungen im Netzwerk (Switches, Kabel etc.) bereits im Vorfeld ermittelt wurden.

4.3.1 Mathematisches Konzept zur Erzeugung eines lösbaren Gleichungssystems

Wie in Abb. 4.2 gezeigt, kann bei Schätzung der RTT zwischen nur zwei Knoten nicht zwischen der Verzögerung d_0 des Knotens N_0 und der Verzögerung d_1 des Knotens N_1 unterschieden werden, da beide zur RTT beitragen. Durch Aufstellen eines Gleichungssystems ist es möglich, jede Verzögerung separat zu schätzen. Dies führt zu einer genaueren Schätzung der Verzögerungen d_i. Es wird die mittlere Software-Verzögerung eines bestimmten Knotens geschätzt, was immer weniger komplex ist als die Schätzung der Summe zweier Verzögerungen, da diese sich aus der Überlagerung von zwei Verzögerungswahrscheinlichkeitsverteilungen ergibt [4.7].

[4.6] Die Zeit zwischen dem Senden des Broadcast-Paketes am Referenzknoten bis zu dessen Empfang am letzten Endknoten ist die für die Synchronisation benötigte Zeit T.

[4.7] Bspw. ergibt sich die Varianz der Summe $X = X_1 + X_2$ der Zufallsvariablen X_1 und X_2 als $Var(X) = Var(X_1) + Var(X_2) + 2 \cdot Cov(X_1, X_2)$ (vgl. [A 146] S. 233). Die Varianz der Summe der Zufallsvariablen ist also mindestens so groß wie die Summe der Einzelvarianzen.

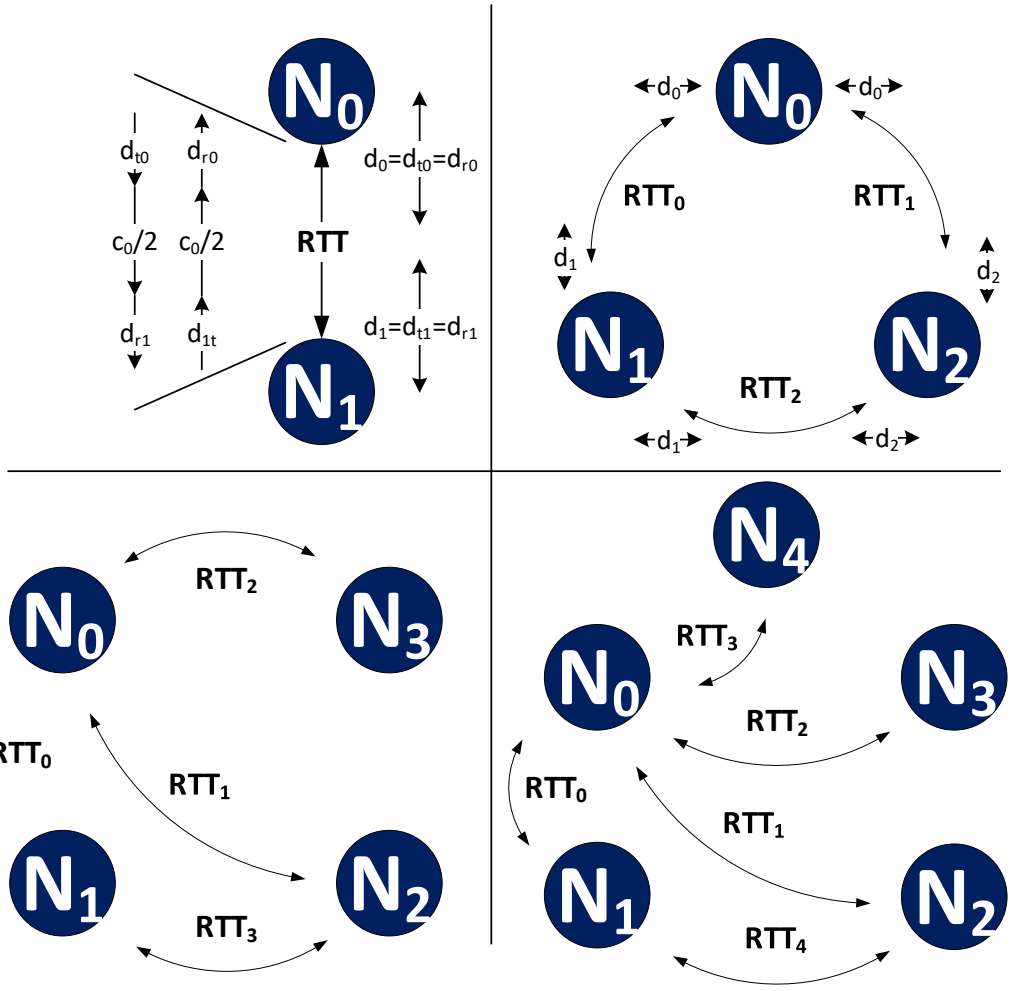


Abbildung 4.2: Die RTTs zwischen mindestens drei Geräten müssen geschätzt werden, um ein lösbares Gleichungssystem zu erhalten.

Jeder Knoten im Netzwerk hat eine bestimmte Übertragungsverzögerung d_t sowie eine bestimmte Empfangsverzögerung d_r. Um ein lösbares Gleichungssystem zu erzeugen, muss die Annahme getroffen werden, dass die Übertragungsverzögerung d_t eines Knotens gleich der Empfangsverzögerung d_r dieses Knotens ist. Diese Vereinfachung ist wesentlich, um ein lösbares Gleichungssystem zu erhalten. Anderenfalls können nicht alle unbekannten Parameter bestimmt werden, wie bspw. in [A 21] postuliert.

Wenn sich n Knoten im Netzwerk befinden, werden n unabhängige Gleichungen für ein Gleichungssystem von Rang n benötigt. Daher werden zuerst die $n-1$ RTTs zwischen dem Referenzknoten und den $n-1$ anderen Knoten im Netzwerk gemessen. Im Anschluss wird eine zusätzliche RTT zwischen zwei Knoten gemessen, wobei keiner dieser Knoten der Referenzknoten ist. Da die Verzögerung jedes Knotens zweimal zu jeder RTT beiträgt, ergibt sich das folgende Gleichungssystem für drei Knoten (entsprechend Abb. 4.2):

$$\begin{bmatrix} RTT_0 \\ RTT_1 \\ RTT_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 0 & 2 \\ 0 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} + \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix}. \tag{114}$$

Hierbei sei RTT_x die x-te gemessene RTT, d_x ist die Verzögerung des Knotens N_x und c_x ist die als konstant angenommene Hardware-Verzögerung zwischen den beiden Knoten. Für ein Netzwerk von n Knoten ergibt sich ein Gleichungssystem in der Form (vgl. Abb. 4.2):

$$\begin{bmatrix} RTT_0 \\ RTT_1 \\ \vdots \\ RTT_{n-2} \\ RTT_{n-1} \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & \dots & 0 \\ 2 & 0 & 2 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 2 & 0 & \dots & 0 & 2 \\ 0 & 2 & 2 & 0 & \dots \end{bmatrix} \cdot \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \end{bmatrix} + \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}. \tag{115}$$

Wenn die Hardware-Verzögerung c_x einmal gemessen wurde, kann die Software-Verzögerung eines bestimmten Geräts durch Lösen dieses Gleichungssystems geschätzt werden.

4.3.2 Ablauf der Synchronisation

Im Folgenden wird die Prozedur von PSPI-Sync beschrieben:

1. Es wird ein Referenzknoten N_0 gewählt. Der einfachste Weg, diesen Knoten auszuwählen, besteht darin, eine zufällige anfängliche Wartezeit einzuführen und den Knoten, der zuerst aufwacht, als Referenzknoten auszuwählen. Eine alternative Methode zur Auswahl des Referenzknotens ist die Verwendung eines Ansatzes ähnlich zum BMCA von PTP [A 74]. Der BMCA wählt automatisch einen der Knoten mit der höchsten Taktgenauigkeit als Master aus.
2. Der Referenzknoten N_0 schätzt die RTTs zwischen sich selbst und allen anderen Knoten (N_1, N_2, ..., N_{n-1}). Da angenommen wird, dass die Software-Verzögerung variabel ist, wird die RTT mehrfach gemessen und dann geschätzt (z.B. durch Berechnen des Mittelwertes).
3. Der Referenzknoten N_0 sendet ein Token an einen der anderen $n-1$ Knoten (z.B. N_1).
4. Der Empfänger des Tokens (z.B. N_1) schätzt (z.B: durch Mittelung über mehrere Messungen) eine weitere RTT zu einem beliebigen Knoten (z.B. N_2) und sendet das Ergebnis zurück an den Referenzknoten (N_0).
5. Der Referenzknoten löst das eingeführte Gleichungssystem unter Verwendung der im Vorfeld ermittelten Hardware-Verzögerung sowie der geschätzten RTTs.
6. Der Referenzknoten sendet die Hardware- und die Software-Verzögerungen aller Knoten an die anderen $n-1$ -Knoten. Folglich kann jeder Knoten die Verzögerung zwischen sich und dem Referenzknoten berechnen.

7. Der Referenzknoten synchronisiert das gesamte Netzwerk, indem er Pakete an alle anderen Knoten sendet. Diese Pakete enthalten einen Zeitstempel t_0, der am Referenzknoten direkt vor dem Senden des Pakets auf der Anwendungsschicht genommen wurde, und jeder Knoten N_x nimmt direkt nach dem Empfang des Pakets auf der Anwendungsschicht einen Zeitstempel t_x. Der Offset zwischen dem Referenzknoten N_0 und dem Knoten N_x beträgt $t_0 + t_{delay} - t_x$, wobei t_{delay} die zuvor berechnete Verzögerung zwischen diesem Knoten und dem Referenzknoten ist. Wiederum wird, aufgrund der variablen Software-Verzögerung, der Offset geschätzt, indem der Mittelwert mehrerer Messungen berechnet wird.

8. Für die Resynchronisationen muss die Gesamtverzögerung nicht erneut geschätzt werden (unter der Annahme einer zeitinvarianten Wahrscheinlichkeitsverteilung der Verzögerung). Somit kehren alle Knoten zu Schritt 6 zurück.

4.3.3 Betrachtung von konzeptionellen Fehlern, Ausfallsicherheit und Skalierbarkeit

Konzeptionelle Fehler: Für die Entwicklung des PSPI-Sync-Konzeptes werden einige Annahmen getroffen, durch welche konzeptionelle bzw. systematische Fehler entstehen. Es ist bspw. möglich, den maximalen Fehler anzugeben, der sich aus der angenommenen Gleichheit der Sendeverzögerung d_s eines Knotens x und dessen Empfangsverzögerung d_r ergibt, wenn beide Verzögerungen konstant sind. Obwohl die Einzelwerte d_s und d_r unbekannt sind, kann die Summe von beiden angeben werden: $d_s + d_r = 2 \cdot d_x$ (vgl. Abb. 4.2). Dabei ist d_x die geschätzte Software-Verzögerung des Knotens x. Demzufolge kann eine obere Grenze $Delay_{max}$ für d_s und d_r angegeben werden: $Delay_{max} = 2 \cdot d_x$. Sowohl d_s als auch d_r müssen kleiner als $Delay_{max}$ sein, denn im schlimmsten Fall gilt z.B. $d_s \approx 2 \cdot d_x$ und $d_r \approx 0$, da beide Werte positiv sein müssen.

Ausfallsicherheit: Es gibt keinen Single Point of Failure, da eine dynamische Auswahl des Referenzknotens möglich ist und dieser somit bei einem Ausfall durch einen anderen Knoten ersetzt werden kann. Die Synchronisation ist so lange möglich, wie mindestens drei Knoten im Netzwerk vorhanden sind.

Skalierbarkeit: die Skalierbarkeit ist ein wichtiger Aspekt in IoT-Szenarien, da Tausende von Geräten vernetzt werden sollen. Die vorgeschlagene Methode weist eine gute Skalierbarkeit auf: Bezüglich der Zeit T, die benötigt wird um die Software-Verzögerungen von n Knoten in einem Netzwerk abzuschätzen, gilt $T \sim n$. Es sind also immer n Messungen nötigt, um ein Gleichungssystem von Rang n zu erhalten, das aus n unabhängigen Gleichungen besteht. Der PSPI-Sync-Ansatz ermöglicht es jedoch weiterhin, die Anzahl der Unbekannten zu reduzieren. Die Anzahl der Unbekannten entspricht nicht unbedingt der Anzahl der Knoten im Netzwerk. Stattdessen kann davon ausgegangen werden, dass verschiedene Knoten mit identischen Plattformen und identischer Lastsituation auch identische Verzögerungsverteilungen aufweisen. Daher entspricht die Anzahl der Unbekannten der Anzahl der unterschiedlichen Plattformen im Netzwerk. Eine Plattform P_x sei definiert als ein eindeutiges Paar aus einer Hardware H_y und einer Software S_z: $P_x = (H_y, S_z)$.

Weiterhin sei die Hardware definiert als Kombination aller Hardware-Komponenten (Prozessor, Speicher, NIC, ...) eines Knotens und die Software als Kombination aller Software-

Komponenten (Betriebssystem, Treiber, Anwendungen, Netzwerk-Stack, ...) dieses Knotens.

$$H_y = \{Processor, NIC, Memory, ...\} \tag{116}$$

$$S_z = \{OS, Application, Drivers, ...\} \tag{117}$$

Wenn zwei Knoten das gleiche Software- und Hardware-Setup haben, kann angenommen werden, dass sie die gleiche Verzögerungsverteilung aufweisen. Selbst für Tausende von Knoten in einem IoT-Szenario kann davon ausgegangen werden, dass die Anzahl der unterschiedlichen Plattformen viel geringer ist als die Anzahl der Knoten, z.B. wenn in einer Smart-Factory viele Sensoren desselben Herstellers in der selben Konfiguration an unterschiedlichen Stellen in der Fabrikhalle zum Einsatz kommen. Wenn eine zeitliche Invarianz der Verzögerungsverteilung angenommen werden kann, wird nur eine Schätzung für jede Plattform benötigt. Folglich können für die Synchronisation und jede Resynchronisation Broadcast-Nachrichten verwendet werden und somit ist die Ausführungszeit für die (Re-)Synchronisation unabhängig von der Anzahl der Knoten $T \sim 1$ [4.8]. Darüber hinaus muss, wenn neue Plattformen dem Netzwerk beitreten, nur deren Software-Verzögerung neu abgeschätzt werden.

4.4 Implementierung und Versuchsaufbau

Um den PSPI-Sync-Ansatz zu evaluieren, wurde ein Software-Prototyp entwickelt. Eine Simulation (z.B. mit OMNeT++) wäre als erster Schritt nicht sinnvoll, da viele Parameter auf komplexe Weise zur Verzögerung beitragen. Da diese Parameter nicht bekannt sind, müssen sie in einem realen Versuchsaufbau ermittelt werden. Der PSPI-Sync-Ansatz wurde in Java implementiert, um eine hohe Plattformunabhängigkeit zu erreichen. Java wird in verzögerungssensitiven und Echtzeitszenarien nicht häufig verwendet. Dennoch zeigen die Autoren von [A 147], dass Java-Implementierungen auch harte Echtzeitanforderungen erfüllen können. Aus diesem Grund kam unter anderem die Jamaica VM [A 148] der Version 1.6.0 zum Einsatz, eine echtzeitfähige Java Virtual Machine (VM). Die Prototyp-Implementierung ist flexibel und generisch. Konfigurierbar ist z.B. die Anzahl der Geräte, die Größe der Pakete sowie die Anzahl der Pakete, um jede RTT zu schätzen. Insbesondere die Größe und die Anzahl der Pakete sind wichtige Parameter, da angenommen werden kann, dass die Software-Verzögerung als Verarbeitungsverzögerung von der Paketgröße abhängig ist. Darüber hinaus ist die Anzahl der Pakete wichtig, die zur Schätzung der RTT verwendet werden, um eine ausreichende statistische Genauigkeit sicherzustellen.

Als Versuchsaufbau kamen drei Intel Galileo-Boards [A 149] und ein 1 GBit/s-Switch zur Verbindung der Geräte zum Einsatz.

[4.8] Die Dauer, die ein Broadcast-Paket braucht bis es bei jedem Endknoten ankommt, ist genau genommen auch topologieabhängig, denn mit der Knotenanzahl steigt logarithmisch auch die Anzahl der Switches, die benötigt werden um alle Knoten miteinander zu verbinden. Dieser Zusammenhang wird hier aber vernachlässigt, denn bspw. mit einer dreistufigen Baumstruktur aus Switches mit jeweils 48 Ports lassen sich bereits $48^3 = 110\ 592$ Geräte verbinden. Dieser Einfluss ist also relativ gering. Es handelt sich weiterhin genau genommen um eine Abhängigkeit von der Topologie und nicht um eine Abhängigkeit von der Geräteanzahl.

4.5 Experimente und Auswertung

In diesen Kapitel werden die durchgeführten Experimente beschrieben und ausgewertet.

4.5.1 Skalierbarkeit der Zeitsynchronisation

Um die Skalierbarkeit zu bewerten, wurden bis zu 100 Knoten auf einem physischen Gerät unter Verwendung der Loopback-IP-Adresse emuliert. Genauer gesagt wurde die für die Synchronisation benötigte Zeit in Abhängigkeit von der Anzahl der Knoten im Netzwerk untersucht. Hierbei wird davon ausgegangen, dass jeder Knoten eine eigene Plattform darstellt. Wenn mehrere Knoten dieselbe Plattform besitzen, würde dies zu einer Verringerung der Anzahl von Unbekannten und somit zu einer Verringerung der Synchronisationszeit führen. Für jeden Knoten wurde ein einzelner Thread erzeugt. Alle Knoten haben zwar dieselbe IP-Adresse (Loopback), aber eine andere UDP-Portnummer.

Abb. 4.3 bestätigt die Annahme, dass die Ausführungszeit proportional zur Anzahl der Plattformen im Netzwerk ist. Um alle n unbekannten Parameter, also die Software-Verzögerung jeder einzelnen Plattform, abzuschätzen, wird ein Gleichungssystem vom Rang n benötigt. Daher müssen n RTTs geschätzt werden. Die Anzahl der Plattformen wiederum ist kleiner oder gleich der Anzahl der Knoten. Im Gegensatz zum Stand der Technik, nutzt der PSPI-Sync-Ansatz die Tatsache, dass die Verzögerung für jede einzelne Plattform nur einmal geschätzt werden muss, wenn deren Verteilung zeitinvariant ist. Jede Resynchronisation kann Broadcast-Nachrichten verwenden und ist daher unabhängig von der Anzahl der Knoten oder der unterschiedlichen Plattformen. Darüber hinaus wird die Synchronisationszeit in Abhängigkeit von der Anzahl der RTTs untersucht, mit denen alle unbekannten Parameter geschätzt werden.

Abb. 4.3 bestätigt auch die Annahme, dass die Ausführungszeit auch proportional zur Anzahl der Messungen pro RTT ist. Wenn mehr Messungen für die Schätzung verwendet werden, dann führt dies zu einer genaueren Abschätzung der Software-Verzögerungen und damit zu einer genaueren Synchronisation. Die Anzahl der Messungen, die für ein konkretes Konfidenzintervall und ein konkretes Konfidenzniveau der Schätzung der Software-Verzögerung benötigt werden, hängt von der Wahrscheinlichkeitsverteilung der Verzögerung dieser Plattform ab. Der PSPI-Sync-Ansatz unterstützt somit die Anpassung von Präzision und Ausführungszeit an die Anforderungen der Anwendung.

4.5.2 Genauigkeit der Zeitsynchronisation

Im Folgenden wird auf die Bestimmung der Synchronisationsgenauigkeit eingegangen.

Methodik: Um die Synchronisationsgenauigkeit zu bestimmen, wurde mit einer zusätzlichen Vergleichsmessung der tatsächliche Offset zwischen Referenzknoten und einen Endknoten bestimmt. Hierfür wurde mit einem FPGA ein Rechtecksignal erzeugt, das eine Periode von ca. $0,67\ s$ hat (exakt sind es $2^{26} \cdot 10\ ns$). Die ansteigenden Flanken dieses Signals werden von Referenzknoten und Endknoten (den Galileo-Boards) mit Zeitstempeln versehen. Der Offset ergibt sich jeweils direkt aus der Differenz der Zeitstempel, die zur selben steigenden Flanke gehören. Um den Einfluss von Fehlern durch die Software-Verzögerung zu verrin-

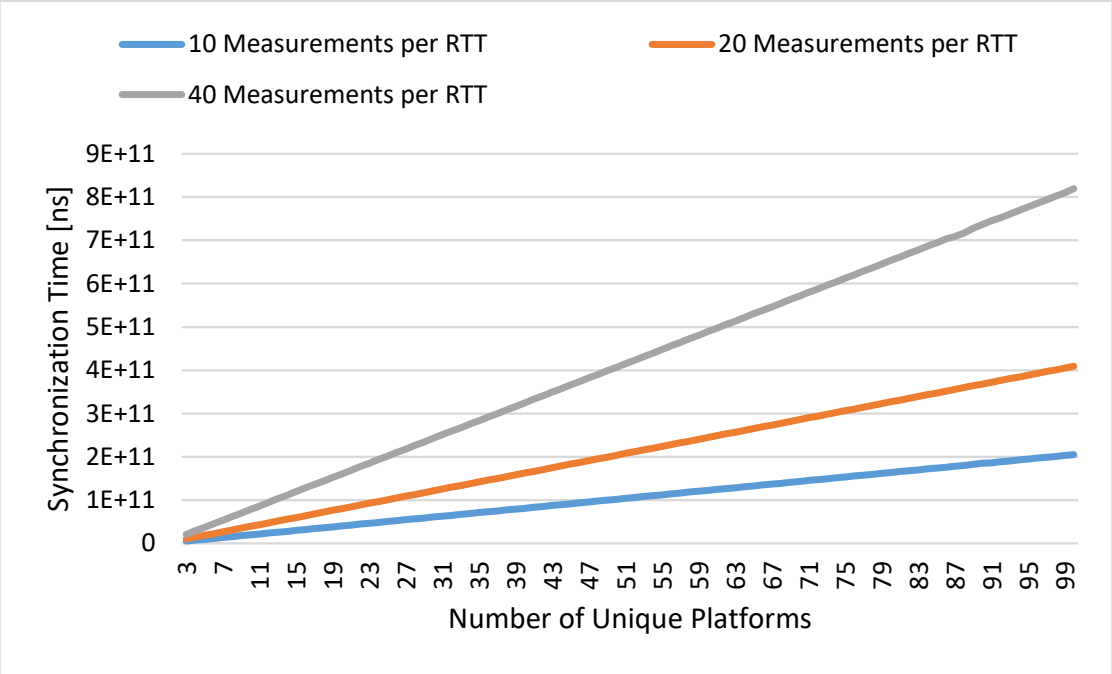


Abbildung 4.3: Untersuchung der Skalierbarkeit auf Basis von Emulation. Die für die Synchronisation benötigte Zeit als Funktion der Anzahl der unterschiedlichen Plattformen im Netz und der Anzahl der Messungen, die pro Schätzung verwendet werden.

gern, wird der Offset bei den Experimenten immer mindestens 2000-mal gemessen. Hierbei kann angenommen werden, dass der Jitter der Zeitstempel durch die Software-Verzögerung bei der Verarbeitung des GPIO-Signals und nicht durch den FPGA verursacht wird.

Ein Problem bei der Bestimmung der Synchronisationsgenauigkeit ist, dass ein Skew zwischen Referenzknoten und Endknoten existiert und sich somit der Offset mit der Zeit ändert. Demzufolge wurde der Skew mit folgender Methode nachträglich kompensiert. Aus den Zeitstempeln von Referenzknoten und Endknoten lässt sich zunächst der Offset bestimmen und man erhält ein Wertepaar (Zeit am Endknoten, Offset-Messung). Hierfür lässt sich eine lineare Regression berechnen, unter der Annahme, dass der Skew konstant ist. PSPI-Sync liefert das Wertepaar (Zeit am Endknoten, Offset-Schätzung). Der Fehler der Offset-Schätzung und somit die Synchronisationsgenauigkeit von PSPI-Sync lässt sich berechnen aus der Differenz der Offset-Schätzung und dem Ergebnis der linearen Regressionsfunktion für diese Zeit am Endknoten. In den Experimenten wurden immer Mittelwert und Standardabweichung des absoluten Fehlers bestimmt, bei einer Stichprobengröße von $N = 100$. In allen Experimenten verwendet PSPI-Sync 10 Messungen zur Bestimmung der RTT.

Für die Messung kamen zwei verschiedene Setups zum Einsatz. Im SSH-Setup bestanden offene SSH-Verbindungen zu allen Boards sowie eine offene SCP-Verbindung zum Refe-

renzknoten [4.9]. Beim Serial-Setup bestanden keine störenden SSH/SCP-Verbindungen im Hintergrund und die einzige Verbindung zum Starten des Experimentes war eine serielle Schnittstelle zum Referenzknoten.

Experimente mit Oracle JVM: Im Folgenden werden die Experimente unter Verwendung einer normalen Oracle JVM der Version 1.8.0.11 beschrieben.

Für das SSH-Setup (siehe Abb. 4.4) wurde für den absoluten Fehler ein Mittelwert von $66,5\ \mu s$ und eine Standardabweichung von $44\ \mu s$ bestimmt. Es zeigt sich hierbei aber ein Problem, denn die Offset-Messung und die lineare Regression sollten den Skew kompensieren. Somit müsste der Fehler (siehe Abb. 4.4, unteres Diagramm) eig. durch eine Gerade mit dem Anstieg Null approximierbar sein. Dies ist aber nicht der Fall. Der Grund hierfür ist die starke Streuung der Offset-Messwerte (insb. in Form von zwei Ausreißern, siehe Abb. 4.4, oberes Diagramm). Die Ausreißer entstehen vermutlich durch die ungenaue Abtastung der GPIO-Pins mittels Software. Das vom FPGA ausgegebene Signal zeichnet sich als reines Hardware-Signal durch eine hohe Genauigkeit aus. Da der Offset nicht angepasst wird, kann dieser folglich auch keine Sprünge aufweisen. Der von PSPI-Sync geschätzte Offset (grau in oberer Abb. 4.4) spiegelt einen solchen Verlauf auch wieder. Im Gegensatz dazu ist es sehr wahrscheinlich, dass es sich bei den wenigen deutlich abweichenden Werten des via GPIOs gemessenen Offsets (blau in oberer Abb. 4.4) um Messfehler handelt. Diese Ausreißer führen zu Ungenauigkeiten in der linearen Regression. Die lineare Regression bedient sich der Methode der kleinsten Quadrate. Somit geht der Abstand der Messwerte zur Regressionsgeraden quadratisch ein und Ausreißer haben sehr großen Einfluss auf die Regressionsgerade. Dieses Problem lässt sich lösen, indem die zwei deutlichen Messausreißer mit über $100\ \%$ Abweichung nicht betrachtet werden.

Wenn diese Messausreißer nicht betrachtet werden, ergibt sich im SSH-Setup (siehe Abb. 4.5) für den absoluten Fehler ein Mittelwert von $61,2\ \mu s$ und eine Standardabweichung von $45,5\ \mu s$. Tatsächlich ergibt sich kein großer Unterschied der Fehlerwerte aber diese sind jetzt deutlich vertrauenswürdiger, da der Skew wirklich kompensiert wird und der Fehler sich durch eine Gerade mit dem Anstieg Null approximieren lässt.

Für das Serial-Setup (siehe Abb. 4.6) wurde für den absoluten Fehler ein Mittelwert von $118\ \mu s$ und eine Standardabweichung von $43,1\ \mu s$ bestimmt. Die Standardabweichung ist also vergleichbar mit dem SSH-Setup. Entgegen den Erwartungen ist der mittlere Fehler aber größer, d.h. die Synchronisation in diesem Setup weniger genau. Eine Erklärung hierfür ist allerdings die relativ ungenaue Referenzmessung, denn die Messwerte haben eine relativ große Streuung (siehe Abb. 4.6, oberes Diagramm). Vermutlich ist also der Einfluss der Ungenauigkeiten der Referenzmessung größer als die vermutlich leicht verbesserte Schätzgenauigkeit von PSPI-Sync, die durch die Verringerung des Netzwerk-Traffics entsteht. Als Folge der Messungenauigkeiten lässt sich auch hier der Fehler nicht mit einer Geraden approximieren, deren Anstieg Null ist. In diesem Setup lässt sich die Ungenauigkeit der Offset-Messung aber nur schwer kompensieren, da es viele Ausreißer gibt.

[4.9] Der durch SSH und SCP erzeugte Overhead wurde nicht genau gemessen. Durch Konsolenausgaben auf dem Referenzknoten und durch Nutzung des Unix-Tools TOP auf den anderen Knoten wurde aber definitiv zumindest ein geringer Hintergrund-Traffic erzeugt.

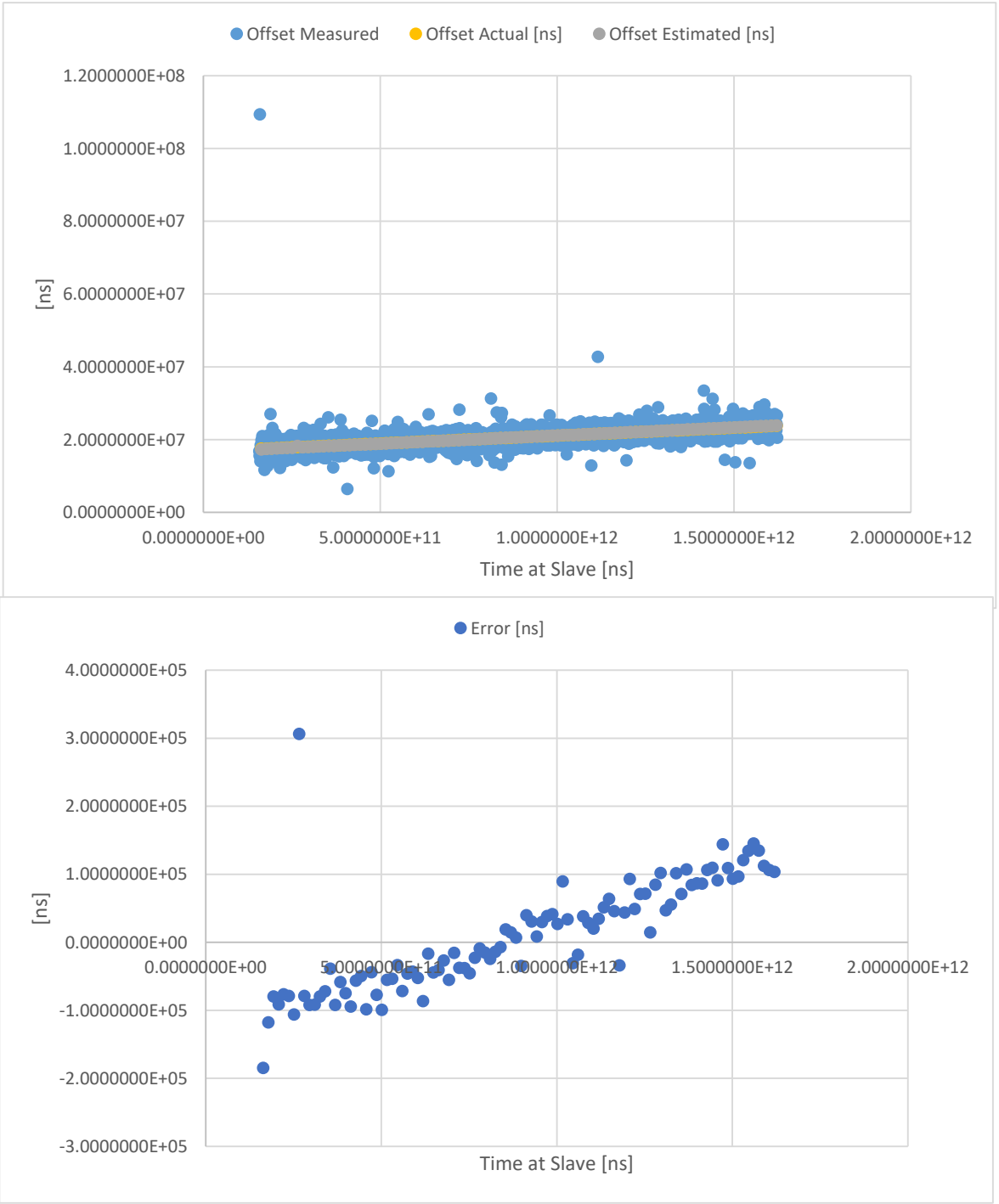


Abbildung 4.4: SSH-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler.

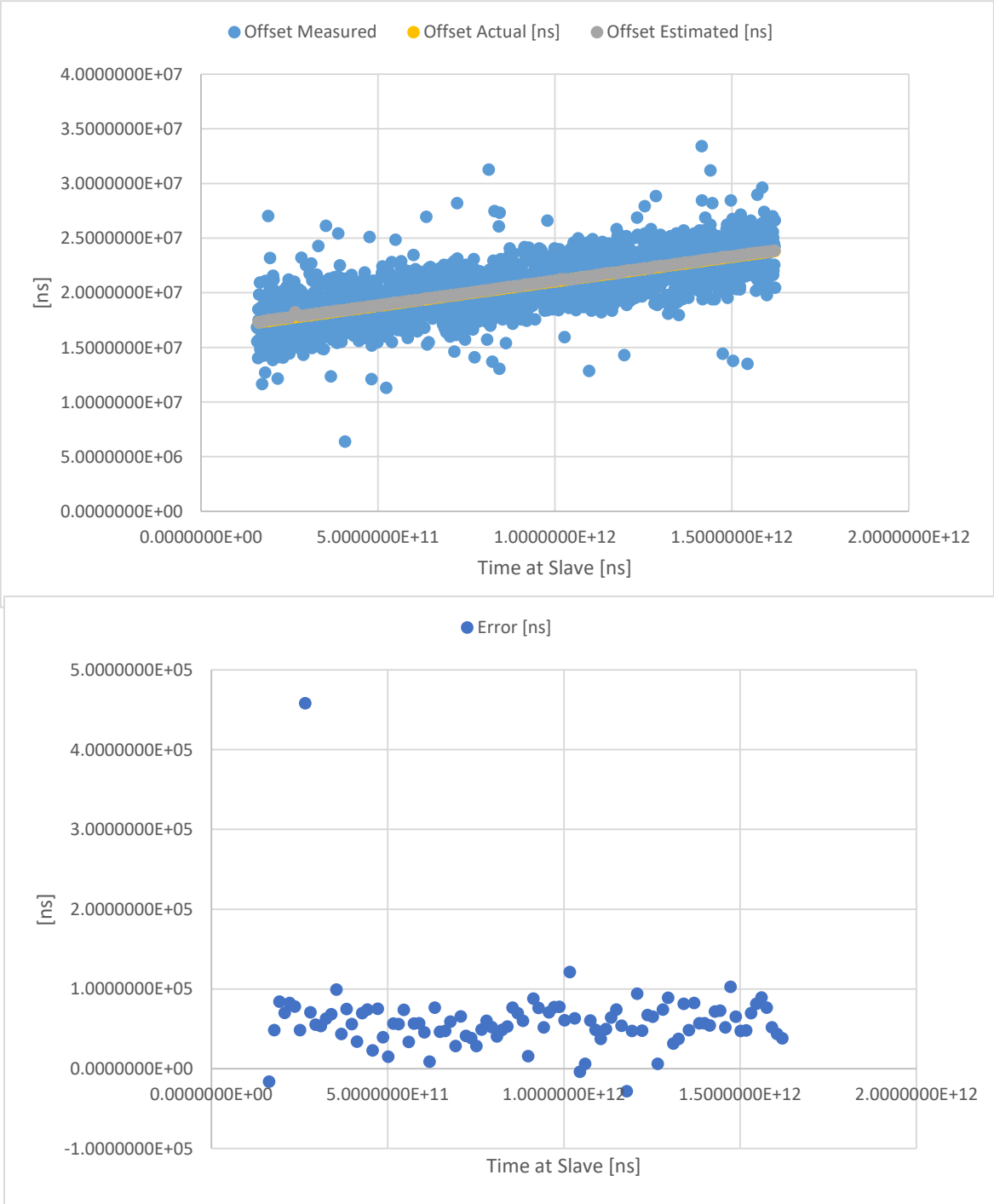


Abbildung 4.5: SSHc-Setup (SSH corrected): Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler.

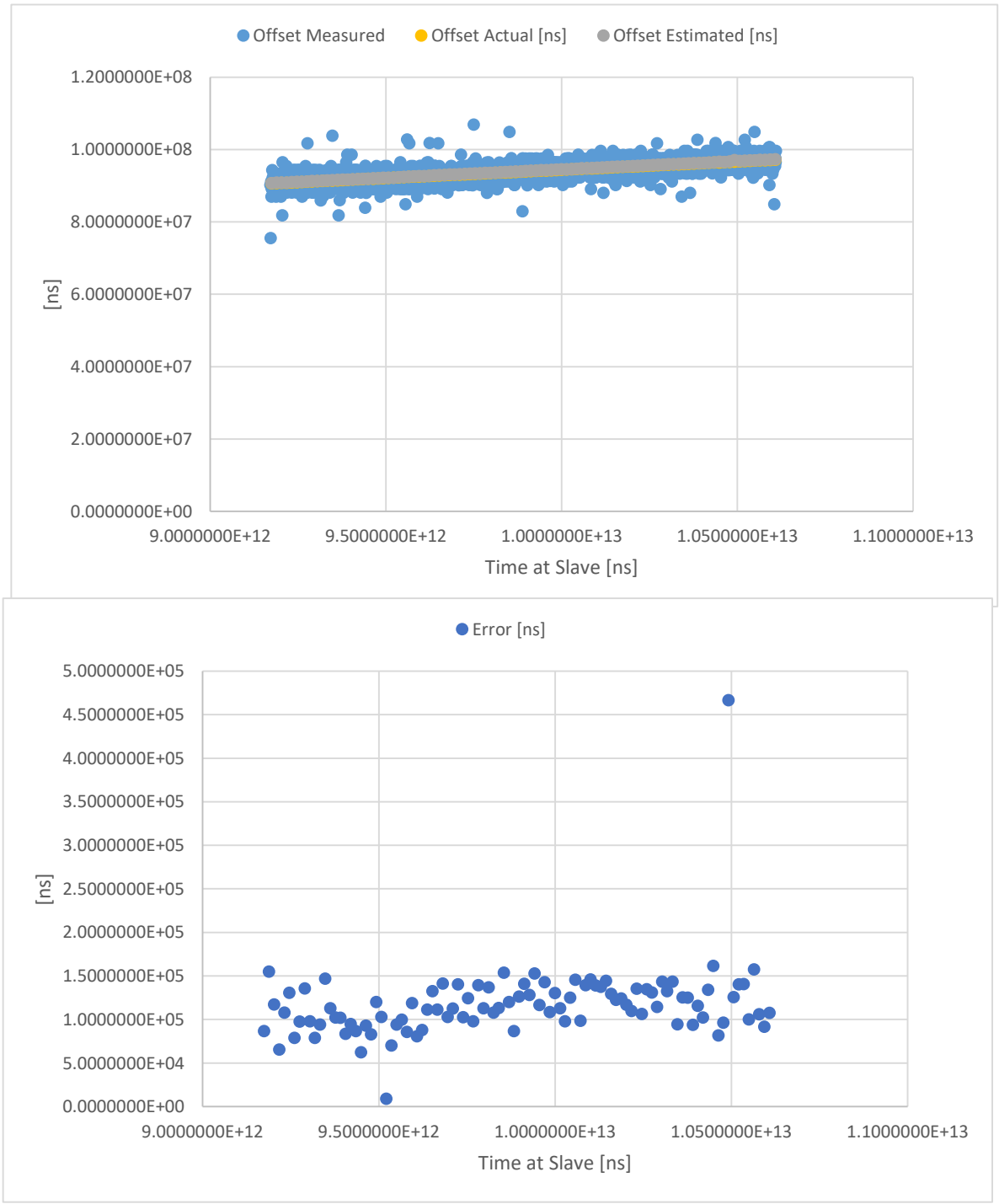


Abbildung 4.6: Serial-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler.

Experimente mit Jamaica Echtzeit-JVM: Im Folgenden werden die Experimente unter Verwendung einer Echtzeit-JVM von Jamaica der Version 1.6.0 beschrieben. Hierbei hat nur PSPI-Sync Echtzeit-Priorität (mittels CHRT-Tool auf 99 gesetzt), die Offset-Messung jedoch hat keine Echtzeit-Priorität, damit PSPI-Sync nicht von dieser gestört wird.

Für das SSH-Setup (siehe Abb. 4.7) wurde für den absoluten Fehler ein Mittelwert von $86\ \mu s$ und eine Standardabweichung von $133\ \mu s$ bestimmt. Auch hier kann der Skew nicht komplett kompensiert werden, denn der Fehler lässt sich nicht durch eine Gerade mit dem Anstieg Null approximieren. Außerdem gibt es durch die Ungenauigkeit der GPIOs erneut zu viele Ausreißer in der Offset-Messung, sodass sich dieses Problem nicht kompensieren lässt. Der Fehler ist vergleichbar mit dem der Messung unter Verwendung einer Nicht-Echtzeit-JVM, was nicht den Erwartungen entspricht. Eine plausible Erklärung ist jedoch, dass zur Bestimmung der RTT von PSPI-Sync immer 10 Messungen vorgenommen werden. Verzögerungen bei einer Messung fallen also weniger ins Gewicht und die Verwendung einer Echtzeit-JVM hat nur einen geringen Einfluss auf den gemittelten Wert der RTT. Allerdings finden sich periodische Ausreißer in den Fehlerwerten. Diese entstehen vermutlich durch die zusätzliche Belastung durch die SSH/SCP-Verbindungen im Hintergrund.

Für das Serial-Setup (siehe Abb. 4.8) wurde für den absoluten Fehler ein Mittelwert von $109\ \mu s$ und eine Standardabweichung von $67\ \mu s$ bestimmt. Erneut lässt sich der Fehler nicht durch eine Gerade mit dem Anstieg Null approximieren und wieder gibt es zu viele Ausreißer in der Offset-Messung, um diese Abweichungen zu kompensieren. Der Fehler ist auch hier vergleichbar mit der Verwendung einer Nicht-Echtzeit-JVM. Im Vergleich zum SSH-Setup ist wieder die Standardabweichung des Fehlers gering, da es keine SSH/SCP-Verbindungen im Hintergrund gibt.

Zusammenfassung der Genauigkeitsmessung: Es sei angemerkt, dass die Synchronisationsgenauigkeit in hohem Maße von vielen Betriebsparametern abhängt, wie dem Taktgenerator, der Arbeitslast des Geräts, der Verkehrslast im Netzwerk, dem Betriebssystem und der Implementierung des Ansatzes, um nur einige zu nennen. Dies spiegelt sich auch in den Messergebnissen in diesem Kapitel wider. Folglich ist es schwer, eine allgemeingültige Aussage über die erreichbare Genauigkeit von PSPI-Sync zu treffen. Als weitere Einschränkung sei die Ungenauigkeit der Vergleichsmessung zu nennen. Diese hat einen so großen Einfluss auf die Ergebnisse, dass z.B. für die Messungen mit Echtzeit-JVM eine geringere Synchronisationsgenauigkeit gemessen wurde als mit Nicht-Echtzeit-JVM. Zusammenfassend lassen sich dennoch die folgenden Erkenntnisse formulieren:

- Die gemessene Synchronisationsgenauigkeit liegt im μs-Bereich und übertrifft die Genauigkeit von NTP von ca. $1\ ms$ um ein bis zwei Größenordnungen.
- Da PSPI-Sync bei der RTT-Messung immer über mehrere Werte mittelt, hat die Verwendung einer Echtzeit-JVM keinen großen Einfluss auf die Synchronisationsgenauigkeit. Es sei allerdings angemerkt, dass sich durchaus Vorteile der Echtzeit-JVM ergeben können, wenn die Geräte eine höhere CPU-Auslastung haben bzw. wenn es mehr konkurrierende Tasks im System gibt, gegenüber denen PSPI-Sync priorisiert werden muss.
- Wenn weitere Netzwerkanwendungen wie SSH oder SCP für Hintergrund-Traffic sorgen, sorgt dies für eine größere Standardabweichung des Fehlers.

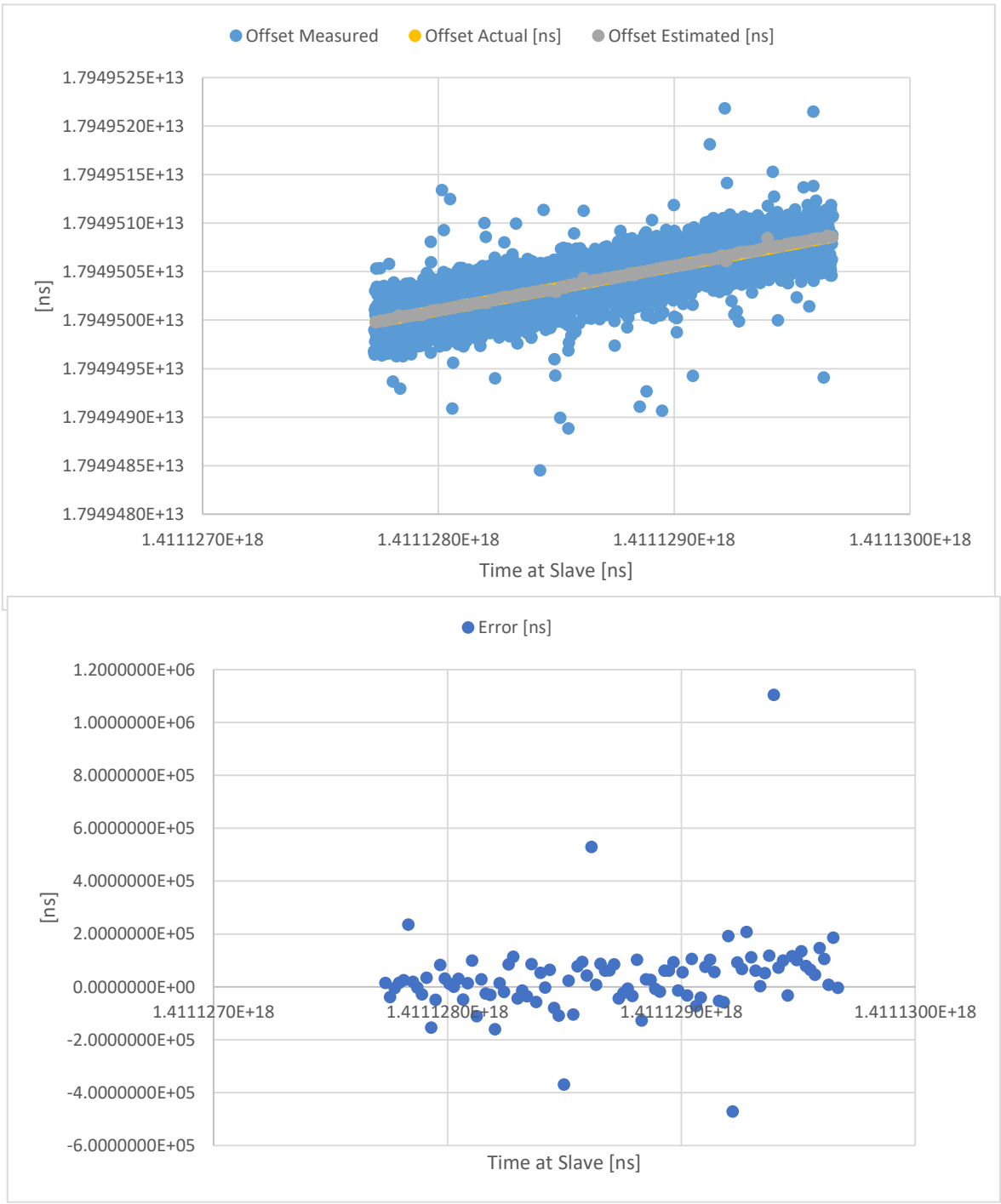


Abbildung 4.7: RT-SSH-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler.

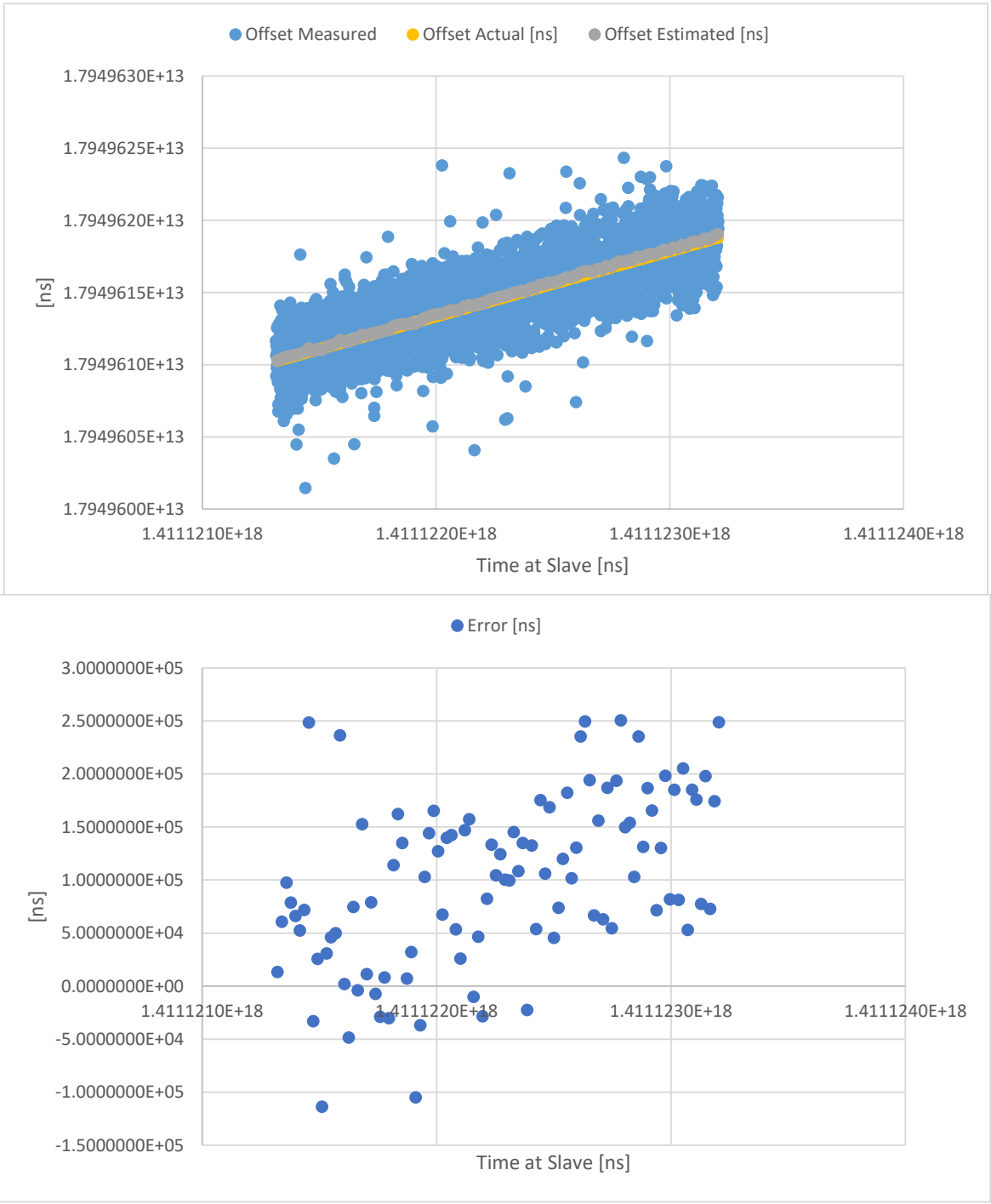


Abbildung 4.8: RT-Serial-Setup: Offset-Messungen, deren lineare Regression und die Offset-Schätzungen von PSPI-Sync, sowie der (vorzeichenbehaftete) Fehler.

Tabelle 4.1: Übersicht aller Ergebnisse

Experiment	Std-Dev [s]	Mittelwert [s]
SSH	$44\ \mu s$	$66,5\ \mu s$
SSHc	$45,7\ \mu s$	$61,2\ \mu s$
Serial	$43,1\ \mu s$	$118\ \mu s$
RT-SSH	$133\ \mu s$	$86\ \mu s$
RT-Serial	$67\ \mu s$	$109\ \mu s$

- Für noch genauere Messungen empfiehlt es sich, die Offset-Messung zu verbessern. Hierfür können z.B. noch mehr Messwerte aufgenommen werden. Allgemein scheint eine Messung via GPIO-Pin aber zu ungenau. Ein weiterer Ansatz wäre es, der Offset-Messung auch Echtzeit-Priorität zu geben. Dann könnte aber PSPI-Sync selbst durch die Offset-Messung behindert werden, was dessen Genauigkeit verschlechtern würde. Eine letzte Möglichkeit wäre die Anwendung weiterer Verarbeitungsschritte und Filterungen um den Einfluss der Messausreißer zu verringern.

Tabelle 4.1 zeigt eine Übersicht aller Ergebnisse. Hierbei stehen Serial und SSH für die Ergebnisse mit Oracle JVM im Serial- bzw. SSH-Setup. SSHc (SSH corrected) steht für die Ergebnisse mit Oracle JVM im SSH-Setup, wobei die zwei deutlichen Ausreißer bei den Offset-Messungen nicht zur Berechnung der Regression verwendet wurden, um eine bessere Skew-Kompensation zu erreichen. RT-Serial und RT-SSH stehen für die Ergebnisse mit Jamaica Echtzeit-JVM im Serial- bzw. SSH-Setup.

4.6 Zwischenfazit

In diesem Kapitel wird der präzise, skalierbare und plattformunabhängige PSPI-Sync-Ansatz für die Zeitsynchronisation in Netzwerken vorgestellt, der auf der Schätzung aller unbekannten Verzögerungen im Netzwerk basiert. PSPI-Sync basiert auf zwei grundlegend neuen Konzepten: einem neuen Ansatz zur Verzögerungsbestimmung sowie der Trennung von Verzögerungsbestimmung auf Basis von Unicast-Nachrichten und eigentlicher Synchronisation auf Basis von Broadcast-Nachrichten.

Das vorgeschlagene Verfahren zur Verzögerungsbestimmung ist plattformunabhängig und erfordert daher keine spezielle Hardware. Daher kann es verwendet werden, um die Software-Verzögerung für einen Zeitsynchronisationsalgorithmus zu bestimmen.

Nach bestem Wissen des Autors ist PSPI-Sync ein völlig neues Schätzverfahren. Es verbindet RTT-Messungen mit einem Gleichungssystem um durch Informationen von mehr als zwei Geräten eine höhere Genauigkeit zu erziehen. Basieren auf der umfangreichen Literaturübersicht in Kapitel 3 lässt sich PSPI-Sync von den Anforderungen her am ehestens mit reinen Software-Verfahren wie NTP [A 62] bzw. dessen Erweiterungsvorschlägen [A 61] vergleichen. Da es sich bei PSPI-Sync um einen verteilten Ansatz handelt, ist eine algorithmische Vergleichbarkeit mit anderen verteilten Ansätzen wie [A 20], KaDisSy [A 108, 109] oder mit Consensus-Verfahren [A 113, 114] gegeben.

Es wird eine Genauigkeitsmessung vorgestellt, bei welcher der PSPI-Sync-Ansatz die vorhandenen Software-basierten Synchronisationsansätze (z.B. NTP) übertrifft (bezogen auf deren in der Literatur genannten Ergebnisse). Darüber hinaus kann festgestellt werden, dass PSPI-Sync alle vorhandenen Methoden, die nicht Broadcast-basiert sind, hinsichtlich der Skalierbarkeit übertrifft, da die Software-Verzögerung nur einmal abgeschätzt werden muss und alle weiteren Synchronisationen und Resynchronisationen Broadcast-Nachrichten verwenden können, die von der Anzahl der Knoten im Netzwerk unabhängig sind.

Eine klarer Nachteil von PSPI-Sync ist zwar, dass Broadcasts nicht über Netzwerkgrenzen hinweg möglich sind, dies stellt in Industrie-Szenarien aber i.A. keine Einschränkung dar. Eine echte Einschränkungen von PSPI-Sync ist allerdings, dass die Hardware-Verzögerung im Netzwerk bei hohen Lastsituationen nicht mehr konstant ist. In den restlichen Kapiteln dieser Arbeit werden daher weitere Methoden und Ansätze vorgestellt, um dies zu kompensieren.

Erweitert werden könnte PSPI-Sync durch eine Drift-Kompensation sowie eine konkrete Berücksichtigung der Hardware-Verzögerung. Darüber hinaus können verschiedene Methoden untersucht werden (z.B. Kalman-Filter), um noch genauere Schätzungen der Verzögerungen zu erhalten. Des Weiteren wurde bereits ein Java-Simulationsmodell für PSPI-Sync erstellt. Hierbei kam die im Rahmen dieses Promotionsvorhabens entstandene neueste Version der Java-Erweiterungen für den Netzwerksimulator OMNeT++ zum Einsatz [B 10, 7]. Vorerst wurden auf Basis dieses Simulationsmodells noch keine genaueren Untersuchungen durchgeführt, dies wäre in Zukunft aber möglich.

5 PTP-LP: Verbesserung der Robustheit von PTP gegenüber Variationen der Paketverzögerung mittels linearer Optimierung

Zeitsynchronisationsprotokolle wie NTP und PTP werden häufig zum Synchronisieren von Komponenten verteilter Systeme verwendet, um zeitgesteuerte und koordinierte Aktivitäten zu ermöglichen. Obwohl insbesondere PTP theoretisch eine Genauigkeit im Bereich von Nanosekunden erreichen kann, sinkt die praktische Genauigkeit von PTP bei stark variabler Paketverzögerung. In diesem Kapitel wird daher der PTP-LP-Ansatz vorgestellt zur Erhöhung der Synchronisationsgenauigkeit von IEEE 1588 PTP. PTP-LP ist voll kompatibel mit bestehenden Standards (PTP und gPTP) und es wird gezeigt, dass PTP-LP gegenüber variierenden Paketverzögerungen sehr robust ist. PTP-LP basiert auf PTP, um präzise Hardware-Zeitstempel zu erhalten und übergibt diese als Constraints für ein lineares Optimierungsproblem (engl. linear program) an einen LP-Solver (engl. linear programming solver). Dieser Solver schätzt Offset und Drift zwischen den Geräten. PTP-LP wird evaluiert im Vergleich zu Standard-PTP und einem weiteren Ansatz aus dem Stand der Technik (PTP-Kalman) unter verschiedenen Bedingungen hinsichtlich Taktstabilität und unterschiedlichen Verteilungen für die Paketverzögerung. Darüber hinaus wird untersucht, welchen Einfluss die Anzahl der Pakete auf die Synchronisationsgenauigkeit hat. Der PTP-LP-Ansatz erreicht unter nahezu allen untersuchten Bedingungen eine sehr gute Synchronisationsgenauigkeit. Die besten Ergebnisse werden erzielt bei Verwendung eines stabilen Hardware-Takts (z.B. eines Hardware-Zeitzählers) und einer unbekannten, nicht zu vernachlässigenden Paketverzögerung im Netzwerk. Beides sind realistische Arbeitsbedingungen. Unter diesen Bedingungen übertrifft PTP-LP die beiden verglichenen Ansätze (PTP und PTP-Kalman) und erhöht die Synchronisationsgenauigkeit um einen Faktor von bis zu 10^4.

Veröffentlicht wurde der PTP-LP-Ansatz auf der IEEE GLOBECOM Konferenz im Dezember 2018 [B 5]. Das folgende Kapitel basiert auf dieser Veröffentlichung.

5.1 Einleitung und Motivation

Die Zeitsynchronisation wird, wie bereits erwähnt, verwendet, um Komponenten eines verteilten Systems eine gemeinsame Zeitbasis bereitzustellen. Dies ist für zeitlich abgestimmte und koordinierte Aktivitäten wie z.B. verteilte Messungen in IIoT-Szenarien (z.B. Smart Factory) wesentlich, da die erfassten Daten genaue Zeitstempel für die weitere Verarbeitung benötigen.

Im Allgemeinen verwenden Synchronisationsprotokolle Nachrichten, um Zeitstempel zwischen Master und Slaves auszutauschen, die Verzögerungen zwischen beiden zu schätzen und die Slave-Zeit an die Master-Zeit anzupassen (über die Bestimmung von Offset und Drift).

Einer der wichtigsten Parameter eines Taktsynchronisationsansatzes ist die Synchronisationsgenauigkeit. Obwohl insbesondere PTP eine theoretische Genauigkeit in der Größenordnung von Nanosekunden liefern kann, leidet die praktische Genauigkeit von PTP, wenn

die gemessenen Verzögerungen variieren. Dies wurde z.B. in [A 22, 73] praktisch untersucht und nachgewiesen.

In diesem Kapitel wird der PTP-LP-Ansatz vorgestellt zur Verbesserung der Präzision von IEEE 1588 PTP bei variabler Paketverzögerung. Nachfolgend wird PTP-LP zusammengefasst:

- Zuerst nutzt PTP-LP die PTP-Nachrichten. Es werden Synchronisationspakete gesendet und diese mit Zeitstempeln versehen.
- Zweitens wird, um die Robustheit gegenüber Variationen der Paketverzögerung zu erhöhen, das CFD-P (Clock Function Determination Problem) formuliert, um die Slave-Zeitfunktion $C(t)$ unter Bezugnahme auf die Master- (bzw. Referenz-) Zeit $T(t)$ zu schätzen. Die PTP-Zeitstempel, die über mehrere Synchronisationsperioden aufgenommen wurden, finden als Constraints für das CFD-P Verwendung.
- Hierzu werden zwei lineare Teilprobleme formuliert: je eines zur Bestimmung der oberen und der unteren Grenze von $C(t)$. Die PTP-Zeitstempel werden als Constraints für diese Teilprobleme verwendet.

Die Vorteile von PTP-LP lassen sich wie folgt zusammenfassen:

- PTP-LP ist voll kompatibel zu PTP bzw. gPTP, da die Kommunikation zum PTP-Master nicht geändert wird. Stattdessen werden lediglich die PTP-Zeitstempel (bzw. gPTP-Zeitstempel) auf dem Slave-Gerät genutzt.
- PTP-LP besteht daher aus PTP und einem zusätzlichen Verarbeitungsschritt zur Verbesserung der Schätzung. Obwohl in [A 26] bereits vorgeschlagen wurde, eine weitere Schätzung mittels eines Kalman-Filters vorzunehmen (PTP-Kalman), gibt es nach aktuellem Kenntnisstand keinen Ansatz, der LP als Erweiterung anwendet.
- Es wird eine umfassende Evaluation von PTP-LP durchgeführt unter Berücksichtigung unterschiedlicher Taktstabilitäten (Hardware-/Software-Takt) sowie verschiedener Wahrscheinlichkeitsverteilungen für die Paketverzögerung (Gauß und logarithmische Normalverteilung). Insbesondere wird hierfür eine realistische selbstähnliche Verteilung für die Paketverzögerung verwendet. Diese wurde abgeleitet aus einer Verteilung für die Paketankunftszeit, welche in mehreren realen Netzwerken in [A 91] gemessen wurde.
- Im Evaluierungsabschnitt wird PTP-LP verglichen mit zwei Ansätzen aus dem Stand der Technik: Standard-PTP und PTP-Kalman [A 26]. PTP-LP zeigt sich robust gegenüber Verzögerungsvariationen, übertrifft beide unter nahezu allen untersuchten Bedingungen und erhöht die Synchronisationsgenauigkeit um einen Faktor von bis zu 10^4.

In Tabelle 5.1 finden sich die konzeptionellen Unterschiede zwischen PTP-LP und PSPI-Sync. Im Vergleich zu PSPI-Sync ist PTP-LP kompatibel mit dem PTP-Standard, dafür ermöglicht PTP-LP nicht mehr die Verwendung von Broadcasts für die Synchronisation. Während PSPI-Sync die RTT (Round-Trip Time) und ein LGS (Lineares Gleichungssystem) zur Schätzung des Offsets verwendet, kommt bei PTP-LP ein LP-basiertes Schätzverfahren für Offset und Drift zum Einsatz.

Ansatz	Kompatibel mit Standards	Sync mit Broadcasts	Schätzer für Offset	Schätzer für Drift
PSPI-Sync	nein	ja	RTT+LGS	-
PTP-LP	ja	nein	LP	LP

Tabelle 5.1: Konzeptionelle Unterschiede zwischen PTP-LP und PSPI-Sync

5.2 Vergleich mit dem Stand der Technik

In diesem Abschnitt werden verschiedene Synchronisationsansätze vorgestellt und mit PTP-LP verglichen, die entweder etablierte Standards oder vielversprechende Forschungsarbeiten sind. Viele der Forschungsansätze sind nicht mit Standardprotokollen kompatibel. Bei allen in diesem Abschnitt betrachteten Ansätzen verringert eine variierende Paketverzögerung die Synchronisationsgenauigkeit erheblich. Im Gegensatz dazu ist PTP-LP mit vorhandenen Standards (PTP) kompatibel und zeigt sich robust gegenüber Schwankungen der Paketverzögerung. An dieser Stelle sei angemerkt, dass in diesem Abschnitt vorwiegend auf die Unterschiede zwischen den Ansätzen und PTP-LP eingegangen wird. Für eine detailliertere Beschreibung und Einordnung der Ansätze sei auf Kapitel 3 verwiesen.

Wie bereits erwähnt, ist NTP (Network Time Protocol) [A 62] das im Internet am häufigsten verwendete Synchronisationsprotokoll. Es handelt sich bei NTP um ein reines Softwareprotokoll, sodass keine Änderungen auf den unteren Netzwerkschichten benötigt werden. Wie in [A 61] gezeigt, nimmt die Synchronisationsgenauigkeit von NTP ab, wenn nicht vernachlässigbare Verzögerungsschwankungen auftreten. Darüber hinaus ist die Genauigkeit der NTP-Software-Zeitstempel geringer als die Genauigkeit von Hardware-Zeitstempeln (z.B. PTP-Zeitstempeln), was zu einer weiteren Verringerung der Synchronisationsgenauigkeit führt.

Der Standard PTP [A 63] benötigt, wie bereits ausgeführt, spezielle Hardware um eine hohe Präzision zu erzielen. Als reine Software-Version entspricht die Genauigkeit in etwa der von NTP. PTP-LP basiert auf PTP, da PTP-Zeitstempel verwendet werden. Obwohl PTP eine hohe Synchronisationsgenauigkeit erreichen kann, wird angenommen, dass die Paketverzögerung zwischen Master und Slave konstant und symmetrisch ist. Daher können Variationen der Paketverzögerung die Synchronisationsgenauigkeit erheblich verringern. Variationen können auftreten, wenn nicht alle Switches PTP-Support bieten. Wie in der Auswertung gezeigt, kann PTP-LP mit diesen Ungenauigkeiten umgehen, indem ein zusätzlicher Schätzschritt unter Verwendung von LP angewendet wird. Es ist zu beachten, dass ein zusätzlicher Verarbeitungsschritt (z.B. exponentielle Filterung) die Robustheit gegenüber Verzögerungsvariationen erhöhen würde. Die exponentielle Filterung z.B. ist jedoch nicht so robust wie LP, da bei Letzteren implizites Wissen über die Taktfunktion genutzt wird.

Grundsätzlich basiert die Synchronisation von gPTP (IEEE 802.1AS) [A 64] auf PTP, wie bereits erläuert. Allerdings müssen bei gPTP alle Knoten im Netzwerk (Switches und Endknoten) den gPTP-Standard in Form von spezieller Hardware unterstützen [A 75]. Der IEEE TSN-Substandard gPTP fordert also, dass alle Switches die gPTP-Synchronisation in ihren MAC-Schichten unterstützen. Somit kann gPTP auch eine garantierte Synchronisati-

onsgenauigkeit bieten. Im Gegensatz dazu benötigt PTP-LP keine spezielle Hardware auf den Switches, da die Verzögerungsschwankungen auf den Netzwerkknoten ausgeglichen werden können. Daher könnte PTP-LP auch eine interessante Erweiterung des gPTP-Standards sein, insbesondere für Anwendungen, die keine garantierte Synchronisationsgenauigkeit benötigen.

Wie bereits dargestellt, kombiniert der viel beachtete Ansatz PTP-Kalman den Standard PTP und Kalman-Filterung, um die durch verschiedene Unsicherheiten verursachten Fehler zu minimieren [A 26]. Die Autoren bewerten jedoch nicht die Anzahl der für die Synchronisation und die Kalman-Filterung verwendeten Synchronisationsperioden (diese Untersuchungen wären für die Bewertung der Konvergenz und Stabilität des Kalman-Filters wichtig). Darüber hinaus müssen die gaußschen Variationen zur Design-Zeit bekannt sein, um Präzision und Stabilität des Kalman-Filters sicherzustellen. Außerdem geben die Autoren in [A 26] nur Standardabweichungen des Offsets und der Drift an. Die Synchronisationsgenauigkeit (z.B. der mittlere Fehler) wäre jedoch viel interessanter, um die Genauigkeit des Ansatzes zu bewerten. Darüber hinaus findet nur eine konstante Paketverzögerung Berücksichtigung, was weit von einem realistischen Szenario entfernt ist. In diesem Kapitel wird gezeigt, dass PTP-LP den PTP-Kalman-Ansatz übertrifft, insbesondere wenn eine hohe Netzwerklast vorliegt.

Wie bereits ausgeführt, schlagen Mallada et al. [A 20] einen Ansatz ohne explizite Schätzung der Drift vor und demonstrieren dessen Überlegenheit gegenüber NTP. Der vorgeschlagene Algorithmus verwendet die aktuellen Driftinformationen sowie eine exponentielle Filterung der vergangenen Offsets. Abgesehen von ihren Vorteilen (effiziente Berechnung ohne lange Historie) ist die exponentielle Filterung anfällig für Verzögerungsschwankungen, im Gegensatz zum PTP-LP-Ansatz.

In [A 128] schlagen, wie bereits beschrieben, Nilsson et al. einen Ansatz vor, der statistisch robust, für die passive (Einwegkommunikations-) Taktsynchronisation in drahtlosen Sensornetzwerken (Wireless Sensor Networks, WSNs) geeignet ist und die Heavy-Tailed-Likelihood-Funktion nutzt. Da Ausreißer die Kalman-Filterschätzung grundsätzlich verfälschen, erzielen die Autoren im Vergleich zu einem Kalman-Ansatz eine höhere Genauigkeit.

Wie bereits erläutert, schlagen Lenzen et. al. in [A 120] PulseSync vor, das insb. für große Netzwerke geeignet ist. PulseSync flutet das Netzwerk mit schnellen und kurzen Impulsen. Dies führt zu einer kurzen Initialisierungsphase und einer schnellen Anpassung der Synchronisation an Änderungen der Topologie und der Drift. Obwohl die Autoren die Generalisierbarkeit ihres Ansatzes betonen, stützt er sich stark auf die WSN-MAC-Schicht (z.B. in Bezug auf die Zeitstempel). Darüber hinaus ist PulseSync hinsichtlich der Robustheit suboptimal, da es einen Single Point of Failure (den Referenzknoten) aufweist. Im Gegensatz dazu erbt PTP-LP von PTP den besten Best-Master-Clock-Algorithmus (BMCA), der die Wahl eines neuen Masters im Falle eines ausfallenden Referenzknotens sicherstellt.

Der in Kapitel 4 vorgestellte Ansatz PSPI-Sync rechnet mit mittleren Verzögerungen, was wiederum zu Fehlern führt, wenn die Verzögerung selbstähnlich ist (z.B. einer logarithmischen Normalverteilung folgt), da in diesem Fall der Mittelwert nicht der Wert mit der höchsten Häufigkeit ist. Im Gegensatz zum PTP-LP-Ansatz ist PSPI-Sync nicht mit vorhandenen Standards kompatibel.

5.3 Notation

Zunächst sei angenommen, dass die Master-Zeit und die Slave-Zeit den linearen Funktionen $T(t)$ und $C(t)$ folgen (siehe Abbildung 5.1 sowie Gl. 118 und 119). Wir betrachten $T(t)$ als Referenzzeit und $C(t)$ als lineare Funktion von t mit Steigung γ. γ leitet sich ab aus der Drift Γ zwischen Master und Slave. Hierbei entspricht γ der Taktfrequenz f_i des Slave-Geräts geteilt durch die Taktfrequenz f_o des Master-Geräts (siehe Gl. 120). Außerdem wird der Offset θ definiert als Differenz zwischen $C(t)$ und $T(t)$. Folglich ist $\theta(0)$ der Schnittpunkt von $C(t)$ mit der y-Achse (siehe Gl. 121).

$$T(t) = t \tag{118}$$

$$C(t) = \gamma \cdot t + \theta(0) \tag{119}$$

$$\gamma = \frac{f_i}{f_0} \Rightarrow \Gamma = \gamma - 1 \tag{120}$$

$$\theta(0) = C(0) \tag{121}$$

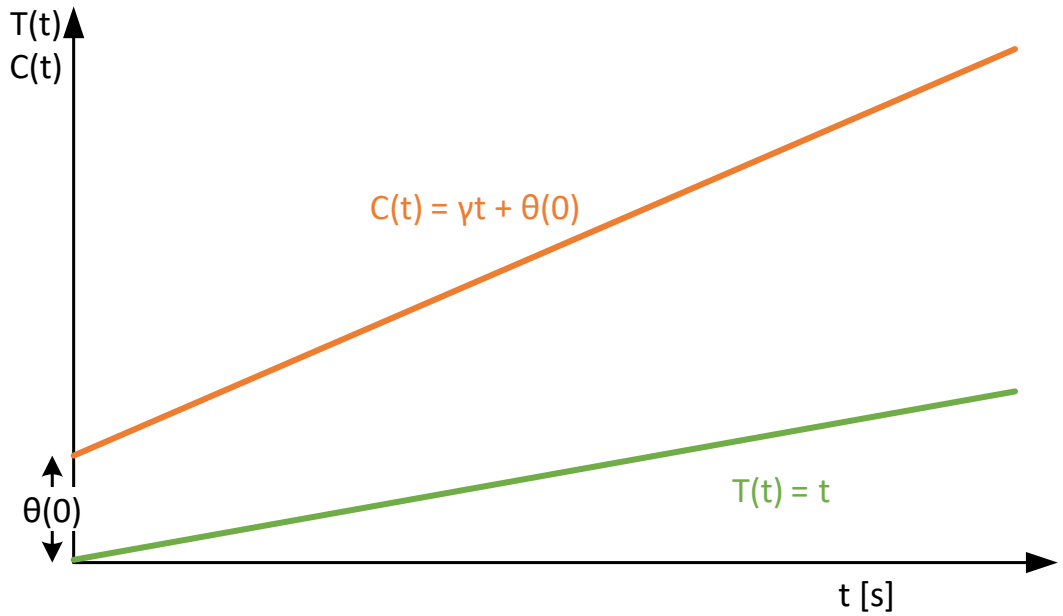


Abbildung 5.1: Zeitfunktionen des Master $T(t)$ und des Slaves $C(t)$ (vereinfacht dargestellt als lineare Funktionen). Hierbei ist $\theta(0)$ der Schnittpunkt von $C(t)$ mit der y-Achse.

5.4 IEEE 1588: Precision Time Protocol (PTP)

Da PTP-LP auf PTP basiert, wird hier kurz auf den Ablauf einer PTP-Synchronisation eingegangen.

PTP schätzt sowohl den Offset als auch die Drift zwischen Master-Zeit $T(t)$ und Slave-Zeit $C(t)$ unter Verwendung mehrerer Zeitstempel. Abbildung 5.2 zeigt beide Taktfunktionen sowie alle PTP-Zeitstempel für eine Synchronisationsperiode (alle Variablen in dieser Abbildung werden im Folgenden eingeführt). Da die Genauigkeit der Zeitstempel entscheidend

ist, werden präzise Hardware-Zeitstempel (z.B. auf der MAC-Schicht genommen) gegenüber Software-Zeitstempeln (z.B. auf der Anwendungsschicht genommen) bevorzugt.

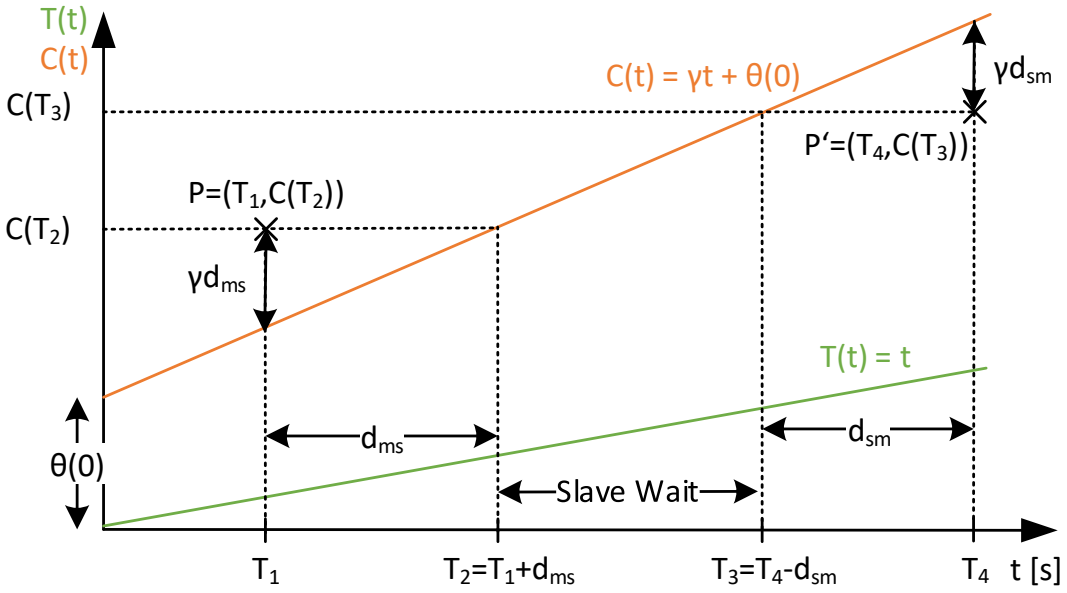


Abbildung 5.2: Master-Zeit $T(t)$ und Slave-Zeit $C(t)$ mit PTP-Zeitstempeln. Zu beachten ist, dass T_i sich auf die Master-Zeit bezieht und $C(T_i)$ sich auf die Slave-Zeit bezieht ($C(t)$ wird der Einfachheit halber als lineare Funktion dargestellt). Der Punkt $P_1(T_1, C(T_2))$ befindet sich immer oberhalb von $C(t)$ und der Punkt $P_4(T_4, C(T_3))$ immer darunter.

PTP nimmt an, dass die Verzögerung zwischen Master und Slave auf dem Vorwärtspfad d_{ms} symmetrisch ist zu jener auf dem Rückwärtspfad d_{sm} ($d_{ms} = d_{sm} = d$). Unter Verwendung der Zeitstempel schätzt PTP den Offset $\theta_M(n)$ für die n-te Synchronisationsperiode wie in Gl. 122 angegeben.

$$\theta_M(n) = \frac{[C(T_2^n) - T_1^n] - [T_4^n - C(T_3^n)]}{2} \tag{122}$$

Hierbei stehen T_x^n bzw. $C(T_x^n)$ für die Zeitstempel T_x bzw. $C(T_x)$ in der n-ten Synchronisationsperiode. Darüber hinaus kann PTP die Drift $\gamma_M(n)$ wie aus Gl. 123 ersichtlich schätzen.

$$\gamma_M(n) = \frac{\theta_M(n) - \theta_M(n-1)}{T_1^n - T_1^{n-1}} \tag{123}$$

Bei einer Offset-Korrektur der Slave-Zeit $C(t)$ um u_θ ändert sich die Schätzung der Drift $\gamma_M(n)$ zu der in Gl. 124 angegebenen Rechenvorschrift:

$$\gamma_M(n) = \frac{\theta_M(n) - [\theta_M(n-1) - u_\theta(n-1)]}{T_1^n - T_1^{n-1}} \tag{124}$$

Unter der Annahme von Gaußschen Unsicherheiten der beim Slave genommen Zeitstempel $\sigma^2_{C(t)}$, der beim Master aufgenommen Zeitstempel σ^2_T, und der Ausbreitungsverzögerung σ^2_d, beeinflussen diese Unsicherheiten die Varianz der PTP-Schätzung für den Offset $\sigma^2_{\theta_M}$, wie aus Gl. 125 ersichtlich (vgl. [A 26]).

$$\sigma^2_{\theta_M} = \frac{1}{2}(\sigma^2_{C(t)} + \sigma^2_T + \sigma^2_d) \tag{125}$$

Weiterhin kann der Einfluss auf die Varianz der Driftschätzung $\sigma^2_{\gamma_M}$ wie in Gl. 126 angegeben werden. Dabei ist $T_1^n - T_1^{n-1} = \Delta T$ die Synchronisationsperiode (vgl. [A 26]).

$$\sigma^2_{\gamma_M} = \frac{2 \cdot \sigma^2_{\theta_M}}{(T_1^n - T_1^{n-1})^2} = 2\left(\frac{\sigma_{\theta_M}}{\Delta T}\right)^2 \tag{126}$$

5.5 Der PTP-LP-Ansatz

Der in dieser Forschungsarbeit vorgestellte PTP-LP-Ansatz basiert auf PTP, um die genauen PTP-Hardware-Zeitstempel zu erhalten, und wendet eine zusätzliche Schätzung auf diese an. PTP-LP schätzt die lineare Funktion, der die Slave-Zeit $C(t)$ folgt, unter Verwendung der Master-Zeit $T(t)$ als Referenz. Genauer gesagt schätzt PTP-LP die Steigung γ und den Schnittpunkt mit der y-Achse $\theta(0)$ von $C(t)$ (vgl. Gleichungen 120 und 121). Hierzu verwendet PTP-LP verschiedene Zeitstempel T_i^n vom Master und $C(T_i^n)$ vom Slave aus mehreren Synchronisationsperioden $n \in [1, .., N]$ und findet mittels LP zwei lineare Funktionen: eine über und eine unter der originalen Slave-Zeit $C(t)$. PTP-LP schätzt $C(t)$ als Mittelwert beider linearer Funktionen. Hierbei wird implizites Wissen verwendet, das aus Abb. 5.2 hervorgeht: der Punkt $P_1(T_1, C(T_2))$ muss immer über $C(t)$ und dem Punkt $P_4(T_4, C(T_3))$ immer unter $C(t)$. Dies gilt auch dann, wenn $C(t)$ eine nichtlineare Funktion ist.

5.5.1 Problemformulierung

Um γ und $\theta(0)$ von $C(t)$ zu finden, wird das Clock Function Determination Problem (CFD-P) definiert, welches die Zeitstempel T_1^n, $C(T_2^n)$, $C(T_3^n)$ und T_4^n mehrerer Synchronisationsperioden $n \in [1, .., N]$ als Constraints verwendet [5.1].

Vorwärtspfad: Verwendung von $(T_1^n, C(T_2^n))$, um f_{ub} als lineare obere Grenze für $C(t)$ zu finden: PTP-LP verwendet die PTP-Zeitstempel T_1 und $C(T_2)$, die auf dem Vorwärtspfad eines Synchronisationspakets vom Master zum Slave genommen werden, um f_{ub} als obere Grenze von $C(t)$ zu ermitteln. Wie aus Abbildung 5.2 hervorgeht, liegt der Punkt $P_1(T_1, C(T_2))$ immer oberhalb $C(t)$. Folglich muss der Punkt (T_1^n, C_2^n) für jedes Synchronisationsintervall n oberhalb von $C(t)$ liegen. Gesucht ist also die Funktion f_{ub} mit der Steigung α_1 und dem Schnittpunkt mit der y-Achse β_1. Im Idealfall gelten für f_{ub} Gl. 127 und Gl. 128.

[5.1] Genauer gesagt werden die Zeitstempel als Input bzw. Eingabeparameter verwendet um die Constraints des LP bilden.

$$\alpha_1 = \gamma \tag{127}$$
$$\beta_1 = \theta(0) + \gamma \cdot d_{ms} \tag{128}$$

Da die Punkte (T_1^n, C_2^n) auch eine obere Grenze für f_{ub} sind, wird das LP-Problem für den Vorwärtspfad formuliert unter Verwendung der in Gl. 129 angegebenen Constraints.

$$\begin{aligned} \alpha_1 T_1^n + \beta_1 \leq C(T_2^n), \forall n \in [1, ..., N] \\ \Leftrightarrow \alpha_1 T_1^n + \beta_1 - C(T_2^n) \leq 0 \\ \Leftrightarrow C(T_2^n) - \alpha_1 T_1^n - \beta_1 \geq 0 \end{aligned} \tag{129}$$

Dies bedeutet im Grunde, dass jeder Punkt von $f_{ub}(t)$ unter oder auf der Menge von Punkten $(T_1^n, C(T_2^n))$ liegen muss, die als Constraints dienen, da $C(t)$ immer unterhalb von $(T_1^n, C(T_2^n))$ liegen muss. Darüber hinaus kann die Minimierungsfunktion für dieses LP-Problem wie in Gl. 130 angegeben werden.

$$\begin{aligned} f(\alpha_1, \beta_1) &= \sum_{n=1}^{N} (C(T_2^n) - \alpha_1 T_1^n - \beta_1) \\ &= \sum_{n=1}^{N} C(T_2^n) - \alpha_1 \sum_{n=1}^{N} T_1^n - \beta_1 \cdot N \\ &= -\alpha_1 \sum_{n=1}^{N} T_1^n - \beta_1 \cdot N \end{aligned} \tag{130}$$

Dies bedeutet im Grunde, dass $f_{ub}(t)$ so nah wie möglich an den Punkten $(T_1^n, C(T_2^n))$ liegen sollte. Somit konvergiert $f_{ub}(t)$ gegen die Constraints. Der Term $\sum_{n=1}^{N} C(T_2^n)$ kann weggelassen werden, da es sich um eine Konstante handelt, die keinerlei Einfluss auf die Minimierungsfunktion hat.

Rückwärtspfad: Verwendung von $(T_4^n, C(T_3^n))$, um f_{lb} als lineare untere Grenze für $C(t)$ zu finden: Außerdem verwendet PTP-LP die PTP-Zeitstempel T_4 und $C(T_3)$, die auf dem Rückwärtspfad vom Slave zum Master genommen werden, um die untere Grenze von $C(t)$ zu ermitteln. Das Vorgehen ist grundsätzlich äquivalent zum Vorwärtspfad. Wie in Abbildung 5.2 zu sehen ist, liegt der Punkt $P_4((T_4, C(T_3))$ immer unterhalb $C(t)$. Folglich muss der Punkt (T_4^n, C_3^n) für jedes Synchronisationsintervall n unterhalb von $C(t)$ liegen. Gesucht ist also die Funktion f_{lb} mit der Steigung α_2 und dem Schnittpunkt mit der y-Achse β_2. Im Idealfall gelten für f_{lb} Gl. 131 und Gl. 132.

$$\alpha_2 = \gamma \tag{131}$$
$$\beta_2 = \theta(0) - \gamma \cdot d_{sm} \tag{132}$$

Da die Punkte $(T_4^n, C(T_3^n))$ eine untere Grenze für $C(t)$ sind, kann das LP-Problem für den Rückwärtspfad wie in Gl. 133 formuliert werden.

$$\begin{aligned} \alpha_2 T_4^n + \beta_2 \geq C(T_3^n), \forall n \in [1, ..., N] \\ \Leftrightarrow \alpha_2 T_4^n + \beta_2 - C(T_3^n) \geq 0 \end{aligned} \tag{133}$$

Dies bedeutet im Grunde, dass jeder Punkt von $f_{lb}(t)$ über oder auf den Constraints $(T_4^n, C(T_3^n))$ liegen muss, da $C(t)$ oberhalb von $(T_4^n, C(T_3^n))$ verlaufen muss. Darüber hinaus kann die Minimierungsfunktion wie in Gl. 134 angeben werden:

$$\begin{aligned} f(\alpha_2, \beta_2) &= \sum_{n=1}^{N} (\alpha_2 T_4^n + \beta_2 - C(T_3^n)) \\ &= \alpha_2 \sum_{n=1}^{N} T_4^n + \beta_2 \cdot N - \sum_{n=1}^{N} C(T_3^n) \\ &= \alpha_2 \sum_{n=1}^{N} T_4^n + \beta_2 \cdot N \end{aligned} \tag{134}$$

Dies bedeutet im Grunde, dass $f_{lb}(t)$ so nahe wie möglich an den Constraints $(T_4^n, C(T_3^n))$ liegen sollte. Somit konvergiert $f_{lb}(t)$ gegen die Constraints. Der Term $\sum_{n=1}^{N} C(T_3^n)$ kann weggelassen werden, da es sich um eine Konstante handelt, die keinerlei Einfluss auf die Minimierungsfunktion hat.

Bestimmung von $C(t)$ mittels f_{ub} und f_{lb}: Schließlich wird $C(t)$ als Mittelwert der oberen und unteren Schranken f_{ub} und f_{lb} geschätzt, Gl. 135 und 136.

$$\hat{\gamma} = \frac{\alpha_1 + \alpha_2}{2} \tag{135}$$

$$\hat{\theta}(0) = \frac{\beta_1 + \beta_2}{2} \tag{136}$$

Form für den LP-Solver: Aus Gründen der Reproduzierbarkeit wird kurz die Problemformulierung für den LP-Solver angegeben, in der die obigen Gleichungen Berücksichtigung finden. Es wurde das Python-Interface des Solvers *scipy.optimize.linprog()* aus der SciPy-Bibliothek verwendet. Der Solver benötigt das Problem in der Form, die in den Gleichungen 137 und 138 angegeben ist.

$$A_{ub} \cdot x \leq b_{ub} \tag{137}$$

$$f(x) = c^T \cdot x \tag{138}$$

Dabei ist A_{ub} die Koeffizientenmatrix für das lineare Gleichungssystem der Constraints. $x = [\alpha, \beta]^T$ ist der Vektor der Variablen und b_{ub} ist der Vektor der Constraints als obere Schranke. $f(x)$ ist die Minimierungsfunktion, wobei c der Vektor der Koeffizienten ist.

$$129 \Rightarrow \begin{bmatrix} T_1^1 & 1 \\ \vdots & \vdots \\ T_1^N & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \leq \begin{bmatrix} C(T_2^1) \\ \vdots \\ C(T_2^N) \end{bmatrix} \tag{139}$$

$$130 \Rightarrow \begin{bmatrix} -\sum_{n=1}^{N} T_1^n & -N \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = f(x) \tag{140}$$

$$133 \Rightarrow \begin{bmatrix} -T_4^1 & -1 \\ \vdots & \vdots \\ -T_4^N & -1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \leq \begin{bmatrix} -C(T_3^1) \\ \vdots \\ -C(T_3^N) \end{bmatrix} \tag{141}$$

$$134 \Rightarrow \begin{bmatrix} \sum_{n=1}^{N} T_4^n & N \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = f(x) \tag{142}$$

5.5.2 PTP-H: Eine Heuristik für PTP-LP

Zusätzlich zu PTP-LP wird in dieser Forschungsarbeit eine als PTP-H bezeichnete Heuristik vorgeschlagen, um $\hat{f}_{ub}(t)$ und $\hat{f}_{lb}(t)$ als Näherungswerte für $f_{ub}(t)$ und $f_{lb}(t)$ rechnerisch effizienter zu bestimmen. Es sei $P_1 = \{((T_1^1, C(T_2^1)), ..., (T_1^n, C(T_2^n))\}$ die Menge von Constraint-Punkten vom Forward Path und $f_{lr1}(t)$ die lineare Regression für alle Punkte in P_1, dann kann $\hat{f}_{ub}(t)$ wie folgt bestimmt werden:

$$\hat{f}_{ub}(t) = f_{lr1}(t) - m_1 \tag{143}$$

$$m_1 = \max_{(T_1^i, C(T_2^i)) \in P_1} (f_{lr1}(T_1^i) - C(T_2^i)). \tag{144}$$

Weiterhin sei $P_2 = \{(T_4^1, C(T_3^1)), ..., (T_4^n, C(T_3^n))\}$ die Menge der Constraint-Punkte vom Reverse Path und $f_{lr2}(t)$ die lineare Regression für alle Punkte in P_2, dann ergibt sich $\hat{f}_{lb}(t)$ wie folgt:

$$\hat{f}_{lb}(t) = f_{lr2}(t) + m_2 \tag{145}$$

$$m_2 = \max_{(T_4^i, C(T_3^i)) \in P_2} (C(T_3^i) - f_{lr2}(T_4^i)). \tag{146}$$

Grundsätzlich wird die Regressionslinie in Richtung des Constraint-Punktes verschoben, der $C(t)$ am nächsten liegt (siehe Abb. 5.2). PTP-H ersetzt den LP-Solver vollständig.

5.5.3 Konzeptionelle Nachteile von PTP-LP

Es ist zu beachten, dass für die Evaluation von PTP-LP ein komplexes, realistisches und nichtlineares Taktmodell verwendet wird. Der PTP-LP-Ansatz geht jedoch von einer linearen Funktion für $C(t)$ aus. Aufgrund dieser Tatsache entsteht ein systematischer Fehler. Dieser Fehler kann jedoch vernachlässigt werden, wenn davon ausgegangen werden kann, dass $C(t)$ für einen ausreichend kurzen Zeitraum linear ist. Wie in [A 92] gezeigt, ändert sich die Taktfrequenz schneller, wenn hohe Temperaturgradienten auftreten. Darüber hinaus führen viele Synchronisationspakete zu einer langen Synchronisationszeit. Nach langer Zeit wird die Nichtlinearität der Slave-Zeit $C(t)$ relevant, $C(t)$ lässt sich nur noch schlecht durch eine lineare Funktion approximieren und die Synchronisationsgenauigkeit sinkt. Es kann auch vorkommen, dass der LP-Solver länger braucht um eine gültige Lösung zu finden. Allerdings wurde das LP so formuliert, dass immer eine lineare Taktfunktion existiert, welche die Constraints erfüllt. Das LP ist folglich immer lösbar.

Es besteht also das Problem, dass die optimale Anzahl von Paketen, die für die Synchronisation verwendet werden sollten, von der Stabilität des Takts abhängt (z.B. ist ein Hardware-Takt meist für einen längeren Zeitraum ausreichend linear als ein Software-Takt [5.2]).

Trotzdem kann eine Lösung für dieses Problem gefunden werden. Zunächst sollte versucht werden, so viele Zeitstempel wie möglich zu erhalten. Ab einem bestimmten Punkt läuft der LP-Solver in den Timeout (wenn der Einfluss der Nichtlinearität des Taktes steigt und das LP schwieriger zu lösen wird). Es sei N_{Max} die niedrigste Anzahl von Zeitstempeln, für die der LP-Solver in den Timeout läuft, dann ist die Größe der zu lösenden Probleminstanz auf einige Zeitstempel weniger (z.B. $N_{Max} - 5$) zu begrenzen. Dies ist immer noch eine ausreichende Anzahl von Synchronisationspaketen, damit der LP-Solver Offset und Drift mit einer hohen Genauigkeit abschätzen kann. Die Idee ist, dass sich der Takt über $N_{Max} - 5$ Perioden dann immer noch linear genug verhält.

5.6 Verwendetes Taktmodell

In diesem Abschnitt wird kurz das verwendete Taktmodell beschrieben. Dieses diskrete Random-Walk-Taktmodell wurde in [A 26] vorgeschlagen und basiert auf dem kontinuierlichen Taktmodell aus [A 36]. Es ist ein komplexes, nichtlineares Taktmodell, das Unsicherheiten des Inkrementes und der Frequenz berücksichtigt.

Wie bereits erwähnt, folgen Takte in der Realität nichtlinearen Funktionen. Daher findet diese Nichtlinearität im Folgenden Betrachtung. Insbesondere wird eine perfekte Master-Zeit angenommen (da sie als Referenz dient) und nichtlineare Slave-Zeiten. Alle Nichtlinearitäten der Master-Zeit werden als Teil der Slave-Zeit $C(t)$ modelliert, wobei $C(t)$ als schrittweise lineare Funktion mit stochastischem Rauschen approximiert wird. Dem Taktmodell liegt die Annahme zugrunde, dass der Offset und die Drift (zwischen $T(t)$ und $C(t)$) zufällige gaußsche Unsicherheiten haben. Die Gleichungen 147 und 148 zeigen w_θ bzw. w_γ, welche die

[5.2] Hardware-Takt meint hier z.B. einen Hardware-Zähler, der von einem Hardware-Oszillator getaktet wird. Software-Takt meint z.B. einen Software-Zähler, der unter Verwendung von Software-Interrupts inkrementiert wird.

zufälligen Schwankungen von Offset bzw. Drift bezeichnen. Dabei ist $N(m, sd)$ eine Gaußverteilung mit dem Mittelwert m und der Standardabweichung sd. Weiterhin bezeichnet σ_γ (bzw. σ_θ) die Standardabweichung der Offset- (oder Frequenz-) Schwankung. Außerdem bezeichnet n den aktuellen Taktschritt von $C(t)$ und δT die Zeit zwischen zwei Schritten (Ticks).

$$w_\gamma(n) \sim N\left(0, \sqrt{\sigma_\gamma^2 \cdot \delta T}\right) \tag{147}$$

$$w_\theta(n) \sim N\left(0, \sqrt{\sigma_\theta^2 \cdot \delta T}\right) \tag{148}$$

Unter Berücksichtigung dieser Unsicherheiten kann der Offset $\theta(n+1)$ modelliert werden (Gl. 149). Weiterhin lässt sich die Drift $\gamma(n+1)$ modellieren (Gl. 150) als Funktion von n unter der Annahme, dass sich γ auf den Offset θ auswirkt (da sich θ mit der Zeit in Abhängigkeit von γ ändert)).

$$\theta(n+1) = \theta(n) + \gamma(n) \cdot \delta T + w_\theta(n) \tag{149}$$

$$\gamma(n+1) = \gamma(n) + w_\gamma(n) \tag{150}$$

Wenn die Slave-Zeit korrigiert wird (z.B. wird $C(t)$ nach jeder Synchronisationsperiode (ΔT) aktualisiert und um θ_M angepasst), ändert sich das Taktmodell wie in den Gleichungen 151 und 152 gezeigt, mit den Korrekturausdrücken u_θ für den Offset θ und u_γ für die Drift γ.

$$\theta(n+1) = \theta(n) - u_\theta(n) + [\gamma(n) - u_\gamma(n)] \cdot \Delta T + w_\theta(n) \tag{151}$$

$$\gamma(n+1) = \gamma(n) - u_\gamma(n) + w_\gamma(n) \tag{152}$$

5.7 Experimente und Auswertung

In diesem Abschnitt werden die Ergebnisse erläutert, die durch numerische Simulationen mit Python 3.6.2 und SciPy 1.0.0 auf einem Desktop-PC (i7-3770, 32 GB RAM) erzielt wurden.

PTP-LP und PTP-H werden verglichen mit Standard-PTP [A 63] und PTP-Kalman [A 26]. PTP wurde zum Vergleich in dieser Bewertung herangezogen, da es das am weitesten verbreitete Synchronisationsprotokoll ist, das präzise Hardware-Zeitstempel verwendet und daher als Benchmark dient. Weiterhin wird ein Vergleich mit PTP-Kalman [A 26] durchgeführt, da dies der vielversprechendste und bekannteste Synchronisationsansatz ist, der PTP erweitert.

Im Allgemeinen hängt die Synchronisationsgenauigkeit von der Taktstabilität [A 26] und der Netzwerkverzögerung ab. Daher hängt die Genauigkeit von den verwendeten Geräten und Verkehrsmustern im jeweiligen Anwendungsszenario ab. Aus diesem Grund werden in diesem Auswertungskapitel eine Vielzahl unterschiedlicher Taktstabilitäten und Verzögerungsverteilungen untersucht.

5.7.1 Methodik

Die Dauer einer Synchronisationsperiode wurde auf eine Sekunde festgelegt. Dies ist der Standardwert für PTP [A 63]. Jedes Experiment wurde $M = 100$ mal durchgeführt.

Die verschiedenen Ansätze wurden anhand von zwei Metriken miteinander verglichen: dem Synchronisationsfehler und dem Frequenzfehler. Der Synchronisationsfehler ist die verbleibende Differenz zwischen Master- und Slave-Zeit, die durch den Synchronisationsansatz nicht kompensiert werden kann. Der Frequenzfehler $F_{err} = |F_{est}/F_{ref}|$ ist der absolute Wert des Quotienten aus der geschätzten Slave-Frequenz F_{est} und der tatsächlichen Slave-Frequenz F_{ref}.

5.7.2 Verwendete Taktstabilitäten

Für die Auswertung findet das Taktmodell aus Abschnitt 5.6 Verwendung. Zur Modellierung verschiedener Taktstabilitäten werden Taktparameter verwendet:

- die Varianz des Offset-Rauschens am Slave-Gerät σ_θ^2,
- die Varianz des Frequenzrauschens am Slave-Gerät σ_γ^2 ,
- und die Varianz des Zeitstempelrauschens am Slave-Gerät $\sigma_{C(t)}^2$. Diese bildet die Ungenauigkeit beim Aufnehmen von Zeitstempeln am Slave ab. Die Zeitstempel werden berechnet aus der tatsächliche Slave-Zeit und Zufallswerten, die einer Gaußverteilung der Form $N(0, \sigma_{C(t)})$ folgen.

Um diese Auswertung übersichtlich zu halten, werden zwei Klassen von Taktstabilitäten verwendet: ein HW-Takt (z.B. ein Hardware-Zähler, der von einem Hardware-Oszillator getaktet wird) und ein SW-Takt (z.B. ein Software-Zähler, der unter Verwendung von Software-Interrupts inkrementiert wird). Die Tabelle 5.2 zeigt die Werte der in der Auswertung verwendeten Taktparameter. Sie entsprechen realen Taktstabilitäten, die in [A 74] und [A 81] gemessen wurden. Außerdem wird die Allan-Varianz $\sigma_y^2(\tau)$ beider Takte in Abb. 5.3 dargestellt. Hierfür wurde das Taktmodell numerisch simuliert und die Allan-Varianz berechnet. Die selben Verläufe lassen sich auch auch mit folgender Formel bestimmen (vgl. [A 26]):

$$\sigma_y^2(\tau) = \frac{\sigma_\theta^2}{\tau} + \frac{\sigma_\gamma^2 \cdot \tau}{3} \tag{153}$$

Für die Varianz des Zeitstempelrauschens am Slave-Gerät $\sigma_{C(t)}^2$ mussten eigene Annahmen getroffen werden. Für die HW-Clock wird eine Genauigkeit in der Größenordnung von Nanosekunden für die Standardabweichung angenommen und für die Zeitstempel der SW-Clock eine Genauigkeit der Standardabweichung in der Größenordnung von Millisekunden [5.3].

Zu beachten ist, dass ein komplexes, realistisches und nichtlineares Taktmodell verwendet wurde. PTP-LP geht jedoch von einer linearen Funktion für $C(t)$ aus. Aufgrund dieser

[5.3] Zu beachten ist, dass in Tabelle 5.2 Varianzen angeben sind. Die Varianz ist als Quadrat der Standardabweichung definiert. Daher gilt: $\sigma_{Chw(t)}^2 = (1^{-9})^2 = 1^{-18}$ und $\sigma_{Csw(t)}^2 = (1^{-3})^2 = (1^{-6})$.

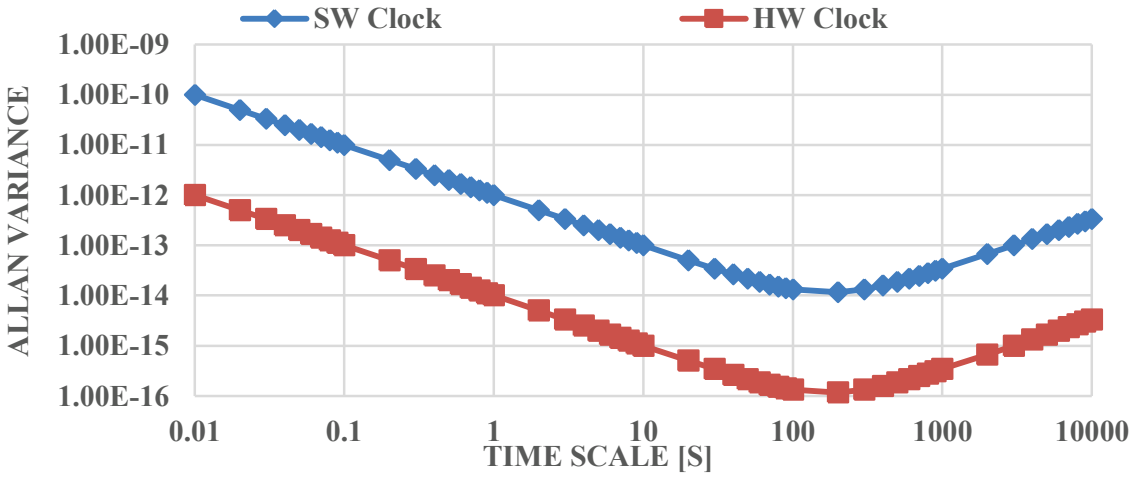


Abbildung 5.3: Allan Varianzen für SW- und HW-Takte, die in der Auswertung verwendet wurden (die X-Achse ist logarithmisch dargestellt).

Tabelle 5.2: In dieser Auswertung verwendete Taktparameter

	σ^2_θ	σ^2_γ	$\sigma^2_{C(t)}$
HW-Takt	1^{-14}	1^{-18}	1^{-18}
SW-Takt	1^{-12}	1^{-16}	1^{-16}

Diskrepanz macht PTP-LP einen systematischen Fehler. Allerdings kann dieser Fehler vernachlässigt werden, falls $C(t)$ über einen begrenzten Zeitraum ausreichend linear ist. Abb. 5.4 zeigt die lineare Master-Zeit $T(t)$, die genaue HW-Zeit $C_{hw}(t)$ und die (weniger genaue) SW-Zeit $C_{sw}(t)$ über eine (relativ lange) Zeit von 5000 Sekunden. Obwohl $C_{hw}(t)$ und $C_{sw}(t)$ nicht linear sind, sehen sie über diesen Zeitraum immer noch linear aus. Nur der Offset zwischen $C_{hw}(t)$ bzw. $C_{sw}(t)$ und $T(t)$ (unterer Teil von Abb. 5.4) zeigt Nichtlinearitäten, die immer noch kleiner sind als der anfängliche Frequenzversatz zwischen $C_{hw}(t)$ (oder $C_{sw}(t)$) und $T(t)$, welcher hier auf 1 ppm gesetzt wurde.

5.7.3 Verwendete Paketverzögerungsverteilungen: Gaußsche Verzögerung und selbstähnliche Verzögerung

In diesem Abschnitt werden die untersuchten Paketverzögerungsverteilungen (gaußsch und selbstähnlich) eingeführt.

Es wird davon ausgegangen, dass Master und Slave über einen Switch verbunden sind und es ein zusätzliches Gerät gibt, das Hintergrundverkehr erzeugt (vgl. Abb. 5.5). Dieses Gerät modelliert den gesamten Hintergrundverkehr im Netzwerk und kann Verkehr erzeugen, der sogar größer ist als es die Übertragungsrate einer einzelnen Leitung eigentlich erlaubt.

Die gaußsche Verzögerungsverteilung geht von einer einfachen gaußschen Verzögerungswahrscheinlichkeit mit einer mittleren Verzögerung von 5 ms und einer Standardabweichung

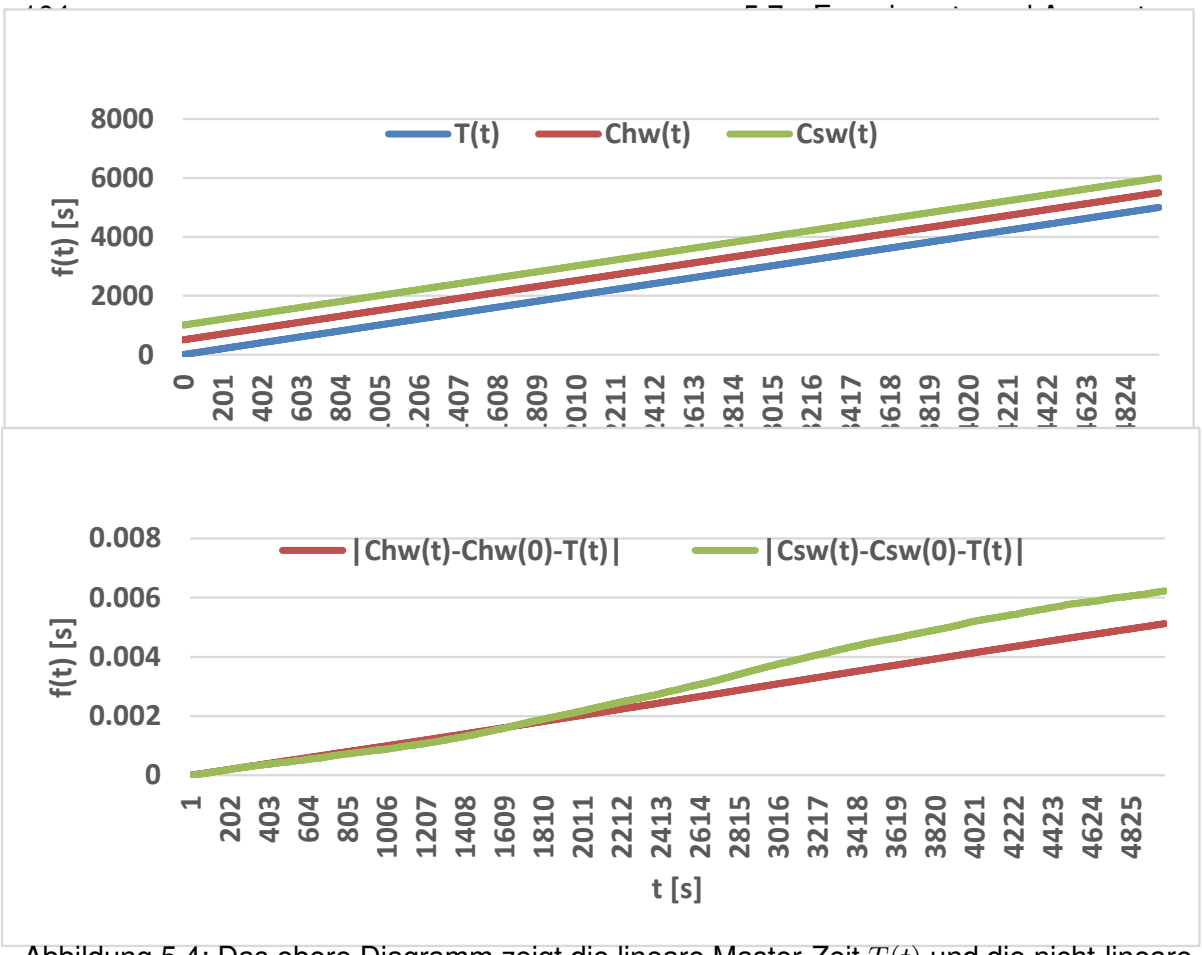


Abbildung 5.4: Das obere Diagramm zeigt die lineare Master-Zeit $T(t)$ und die nicht-lineare HW- (bzw. SW-) Zeit $C_{hw}(t)$ (bzw. $C_{sw}(t)$) über einen Zeitraum von 5000 Sekunden. Das untere Diagramm zeigt den Offset zwischen $C_{hw}(t)$ (bzw. $C_{sw}(t)$) und $T(t)$ über den selben Zeitraum

von 2 ms aus. Die gaußsche Verzögerungsverteilung ist einfach zu berechnen (unter Verwendung der mittleren Verzögerung und der Standardabweichung).

Selbstähnliche Verteilungen sind für Netzwerkverkehr laut [A 91] jedoch realistischer. Dabei bedeutet der Begriff selbstähnlich, dass die Wahrscheinlichkeitsverteilung für verschiedene Zeitskalen ähnlich aussieht [A 150, 151, 91]. In der Simulation wird eine komplexe Berechnung für die selbstähnlichen Verzögerung durchgeführt. Dabei wird angenommen, dass die Paketverzögerung der Summe aus Ausbreitungs- und Warteschlangenverzögerung entspricht. Ferner wird angenommen, dass die Ausbreitungsverzögerung konstant ist und dass die Warteschlangenverzögerung ein ungleichmäßig verteilter statistischer Prozess ist. Es wurden die selbstähnlichen Verteilungen (logarithmische Normalverteilungen) der Paketankunftszeit aus [A 91] verwendet, um eine realistische Wartezeit zu berechnen (vgl. Abb. 5.6). Diese wurden auf eine bestimmte Verbindungsauslastung skaliert und der Warteschlangenfüllungsgrad $fl_{queue}(t)$ an einem Switch-Port bestimmt, unter Verwendung einer Paketgröße $size_{Paket}$ von 800 Bytes (vgl. Abb. 5.7). Dies entspricht der mittleren Paketgröße in [A 91], daher wird angenommen, dass dies ein realistischer Wert für den allgemeinen Hintergrundverkehr ist. Es ist allerdings zu beachten, dass die Paketgröße nur einen geringen Einfluss

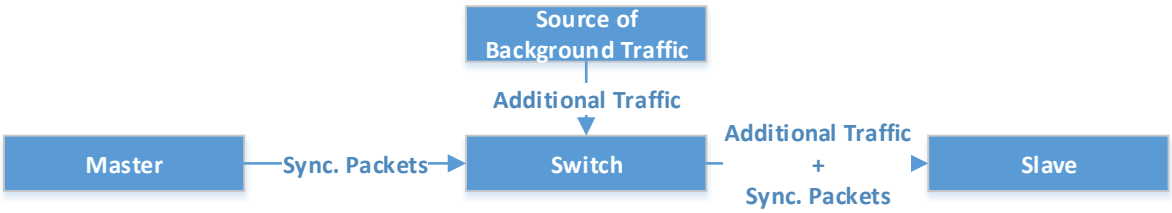


Abbildung 5.5: Netzwerkmodell zur Berechnung der selbstähnlichen Verzögerung aus selbstähnlichen Paketankunftszeiten.

auf die Simulationsergebnisse hat. Die Warteschlangenverzögerung d_{packet} eines Synchronisationspakets berechnet sich aus $fl_{queue}(t)$ zur Ankunftszeit des Synchronisationspakets $t_{arrival}$ und der Übertragungsgeschwindigkeit s_{trans} (z.B. 1 GBit/s für GBit-Ethernet):

$$d_{package} = fl_{queue}(t_{arrival})/s_{trans}. \tag{154}$$

Folglich ist $d_{package}$ die Zeit, die benötigt wird, um die Warteschlange nach $t_{arrival}$ zu leeren. Dies entspricht der Warteschlangenverzögerung eines Synchronisationspaketes. Die Verzögerungszeit folgt auch einer selbstähnlichen Verteilung [5.4].

5.7.4 Auswertung mit gaußscher Paketverzögerung

Da in dieser Auswertung eine nicht zu vernachlässigende Netzwerkverzögerung verwendet wird, erreicht PTP keine Genauigkeit in der Größenordnung von Nanosekunden. PTP-LP erreicht unter allen in diesem Abschnitt untersuchten Bedingungen eine höhere Genauigkeit als PTP (siehe Abb. 5.8).

Es gibt keine wesentlichen Unterschiede zwischen HW- und SW-Clock, da in diesem Szenario die Verzögerungsunsicherheiten dominanter sind als die Taktnichtlinearitäten. Tatsächlich führen die Ausreißer der gaußschen Verzögerung bei PTP-LP und PTP-H oft dazu, dass eine der Schranken (obere oder untere) viel näher an $C(t)$ liegt als die andere. Dies führt zu Fehlern, da $C(t)$ als Mittelwert beider Grenzen geschätzt wird.

Bei der Untersuchung der SW-Clock erreicht das Kalman-Filter eine höhere Genauigkeit als PTP-LP hinsichtlich der Schätzung des Frequenzfehlers (siehe Abb. 5.8). Dafür gibt es zwei Gründe. Erstens ergibt sich dies aus den gaußschen Unsicherheiten (Rauschen) des Offsets, der Frequenz und der Zeitstempel, die am Slave-Gerät unter Verwendung einer SW-Clock aufgenommen wurden. Diese gaußschen Unsicherheiten können durch das Kalman-Filter ausgeglichen werden, das per definitionem optimal für Gaußrauschen ist. Darüber hinaus kann das Kalman-Filter auch die Paketverzögerung in diesem Experiment kompensieren, da auch diese einer gaußschen Verteilung folgt. Es ist jedoch zu beachten, dass die gaußschen Variationen für die Einstellung des Kalman-Filters zur Design-Zeit bekannt sein müssen, um dessen Präzision und Stabilität sicherzustellen. In diesem Szenario wird die Verzögerung also nur kompensiert, weil sie zufälligerweise auch gaußverteilt ist. Zweitens

[5.4] Ein Beispiel für die sich ergebene Verteilung der Verzögerung findet sich z.B. in Kapitel 6 in Abb. 6.5.

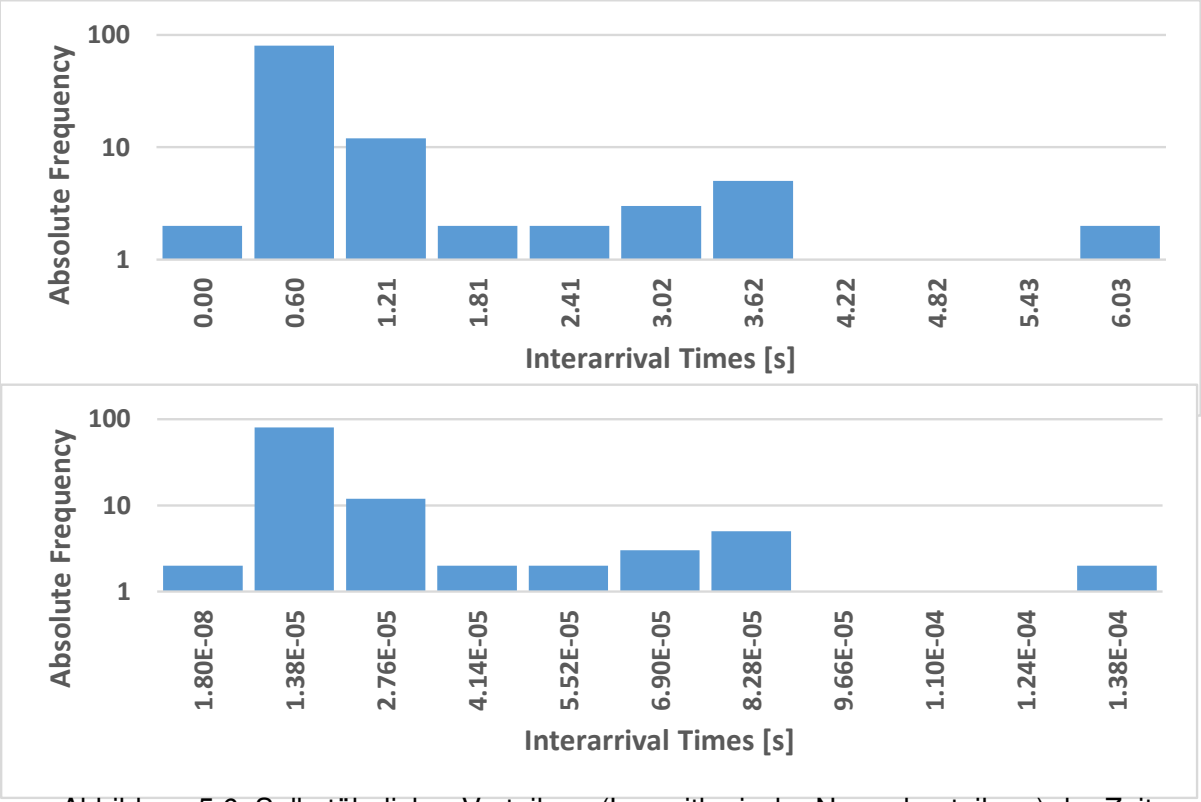


Abbildung 5.6: Selbstähnlicher Verteilung (Logarithmische Normalverteilung) der Zeit zwischen zwei Paketen aus [A 91] und dieselbe Verteilung skaliert auf 50% Linkauslastung (logarithmische Y-Achse).

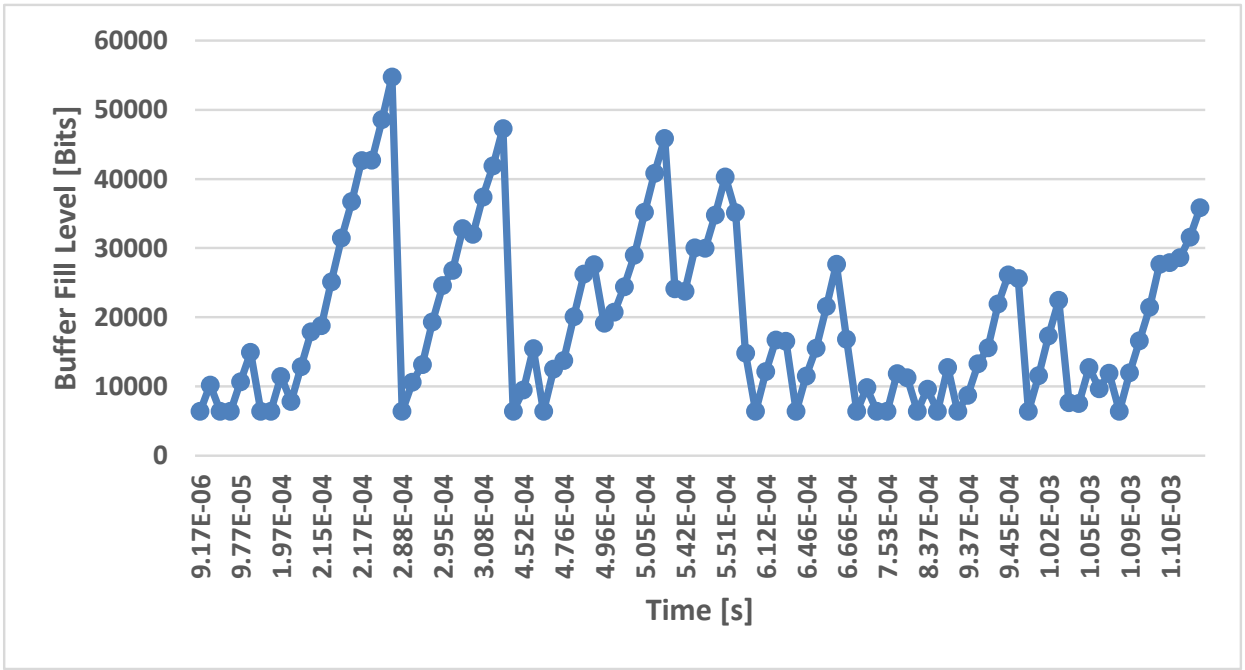


Abbildung 5.7: Füllungsgrad des Switch-Puffers für selbstähnlichen Datenverkehr mit einer Linkauslastung von 50%

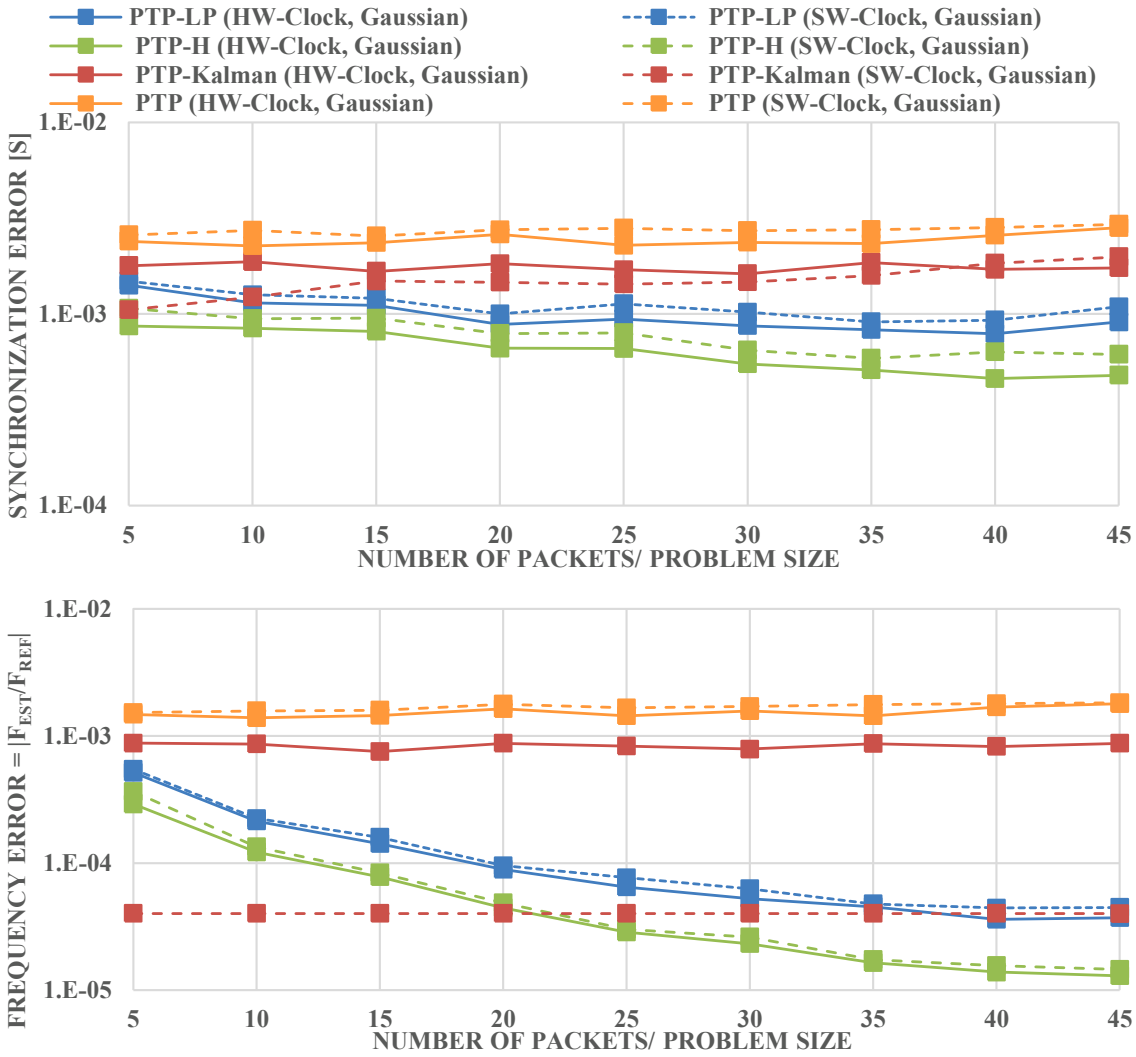


Abbildung 5.8: Synchronisationsfehler und Frequenzfehler für **Gaußsche Verzögerung** (die Y-Achse ist logarithmisch dargestellt).

führen die Unsicherheiten der SW-Clock am Slave-Gerät zu einer nichtlinearen Zeitfunktion am Slave $C(t)$, die von PTP-LP und PTP-H nicht gut verarbeitet werden kann.

Bei der Untersuchung der HW-Clock kann der Schluss gezogen werden, dass PTP-LP den auf Kalman basierenden Ansatz übertrifft (siehe Abb. 5.8). Der Grund hierfür ist, dass die sehr geringe Unsicherheit der HW-Clock zu einer hinreichend linearen Zeitfunktion $C(t)$ führt. Somit ist LP sehr gut geeignet, um $C(t)$ zu schätzen.

Die Genauigkeiten von PTP-LP und PTP-H erhöhen sich, wenn mehr Pakete für die Synchronisation verwendet werden. Der Grund dafür ist, dass mehr PTP-Synchronisationspakete mehr Zeitstempel und damit mehr Informationen für den LP-Solver liefern. Infolgedessen ist es für den LP-Solver einfacher, den Offset und die Drift zu schätzen, da somit der Einfluss der Verzögerungsschwankungen verringert wird. PTP-H erreicht eine Präzision, die sogar etwas besser ist als jene von PTP-LP, da der verwendete LP-Solver nicht immer die optimale Lösung findet und auch die lineare Regression von mehr Informationen profitiert.

5.7.5 Auswertung mit selbstähnlicher Paketverzögerung

Darüber hinaus wurde eine Auswertung mit einer selbstähnlichen Wahrscheinlichkeitsverteilung für die Verzögerung der Synchronisationspakete durchgeführt.

Hierfür wurden feste Werte von ca. 10% und 90% für die mittlere Auslastung gewählt um einen möglichst großen Bereich abdecken zu können. Die Bewertung vieler unterschiedlicher Werte wäre zwar möglich, würde allerdings den Umfang sprengen. Darüber hinaus hängt die tatsächliche Auslastung stark vom Szenario ab, was allgemeine Aussagen erschwert. Durch die Verwendung einer hohen Netzwerkauslastung von 90% kann jedoch untersucht werden, ob PTP-LP unter diesen schwierigen Bedingungen immer noch eine hohe Synchronisationsgenauigkeit erzielt. Der HW-Takt führt zu einer höheren Genauigkeit als der SW-Takt, da die Taktunsicherheiten in diesem Szenario dominanter sind als die Verzögerungsunsicherheiten. Darüber hinaus ist der HW-Takt ausreichend linear und daher sind PTP-LP und PTP-H sehr gut geeignet, ihn abzuschätzen.

Abb. 5.9 zeigt die Ergebnisse für 10% mittlere Auslastung. Es gibt hier keine wesentlichen Unterschiede, aber PTP-LP und PTP-H schneiden etwas besser ab als PTP und PTP-Kalman. Der Grund für die geringen Unterschiede zwischen den Ansätzen ist, dass nur wenige Pakete merklich verzögert werden. Dies führt dazu, dass die Verzögerung fast konstant und somit einfach zu bestimmen bzw. kompensieren ist.

Im Gegensatz dazu führt eine Verbindungsauslastung von 90% zu großen Fehlern, die durch Standard-PTP oder PTP-Kalman nicht mehr kompensiert werden können. PTP-LP erzielt eine höhere Genauigkeit (siehe Abb. 5.10), da es Verzögerungsunsicherheiten handhaben kann, unabhängig von deren Wahrscheinlichkeitsverteilung. Durch die Formulierung als LP kann jede Art von Verzögerungsverteilung kompensiert werden. PTP-H arbeitet nicht so gut wie PTP-LP, da PTP-H nur den größten Abstand zu einem Constraint-Punkt auswertet, der häufig am Anfang oder am Ende liegt [5.5]. Im Gegensatz dazu wertet PTP-LP alle

[5.5] Dies ist eine Folge der Tatsache, dass zuvor die lineare Regression der nichtlinearen Funktion $C(t)$ gebildet wird.

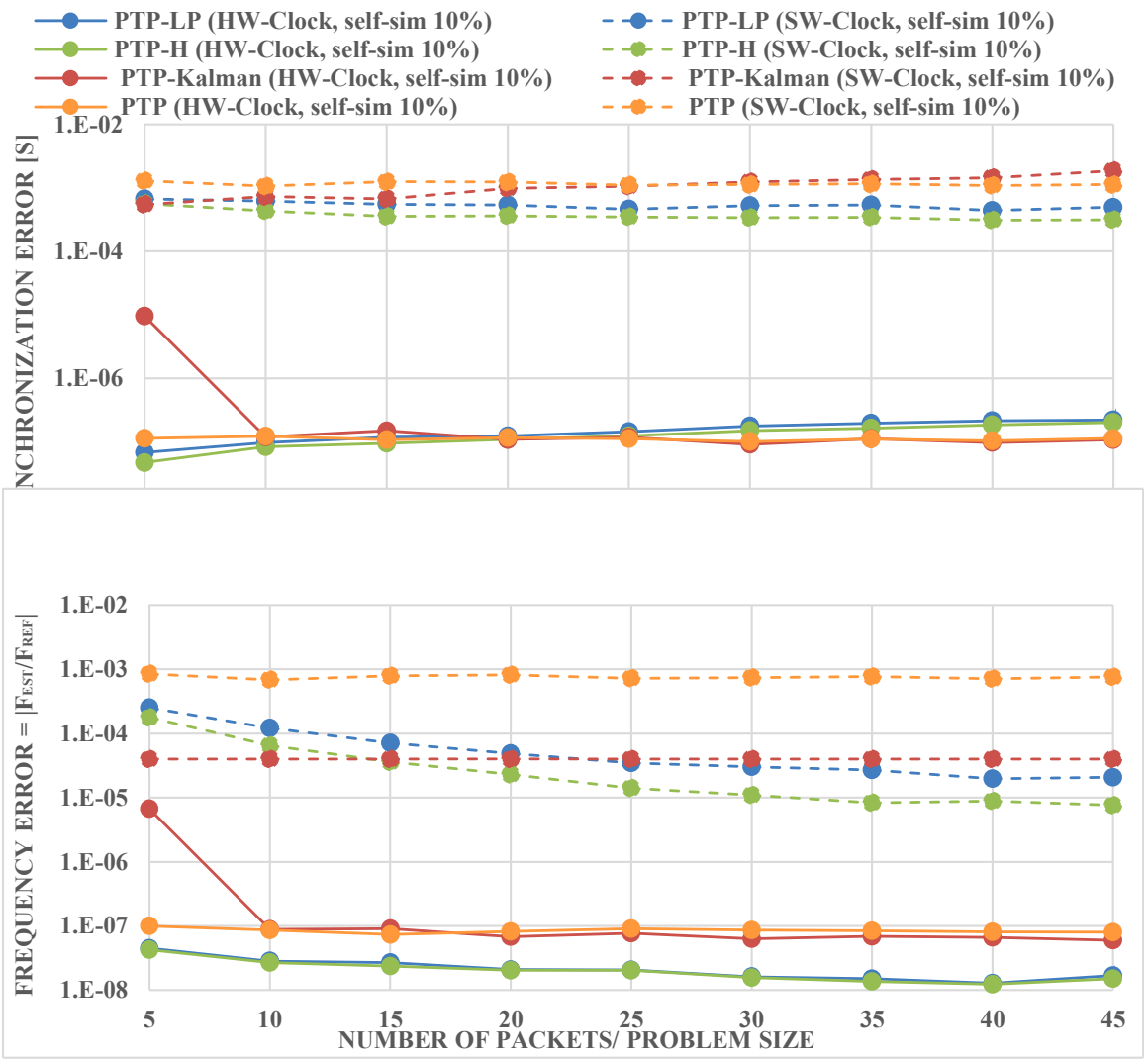


Abbildung 5.9: Synchronisationsfehler und Frequenzfehler für **selbstähnliche Verzögerung mit 10%** mittlerer Auslastung (die Y-Achse ist logarithmisch dargestellt).

Punkte gleichermaßen aus, um eine optimale Lösung zu finden. Wiederum erhöht sich die Genauigkeit von PTP-LP und PTP-H, wenn mehr Pakete für die Synchronisation verwendet werden, da der LP-Solver somit auch mehr Informationen verwenden und folglich genauer schätzen kann. Infolgedessen nimmt der Einfluss statistischer Unsicherheiten ab.

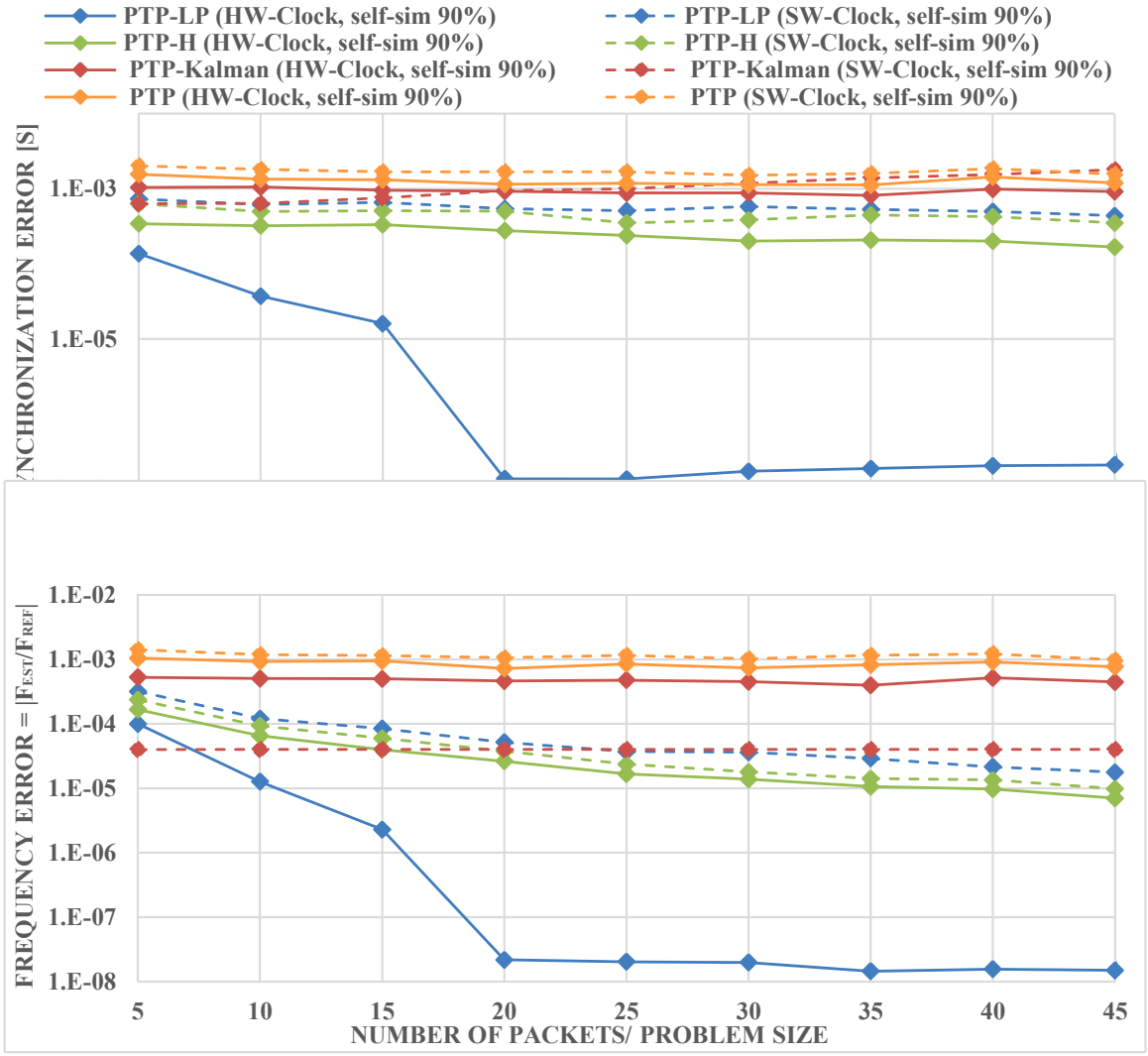


Abbildung 5.10: Synchronisationsfehler und Frequenzfehler für **selbstähnliche Verzögerung mit 90%** mittlerer Auslastung (die Y-Achse ist logarithmisch dargestellt).

5.7.6 Berechnungskomplexität

Fig. 5.11 zeigt die Rechenzeit von PTP-LP und PTP-H über der Anzahl von Synchronisationspaketen. Diese entsprechen der Schwere des zu lösenden Problems. LP hat bekanntermaßen eine lineare Komplexität. Dementsprechend scheint die Rechenzeit in Abhängigkeit von der Problemgröße linear zu sein. PTP-H hat eine viel geringere Rechenzeit als PTP-LP. Die Lösungszeit der LP lag jedoch immer unter $10\ ms$. Obwohl das Lösen des LP-Problems

ein zusätzlicher Rechenschritt ist, kann die Synchronisationsgenauigkeit verbessert werden. Folglich bietet PTP-LP einen Kompromiss zwischen Rechenzeit und Genauigkeit. Weiterhin scheint PTP-H auch einen guten Kompromiss zwischen Rechenzeit und Präzision zu bieten und ist möglicherweise für Geräte mit eingeschränkten Ressourcen geeignet.

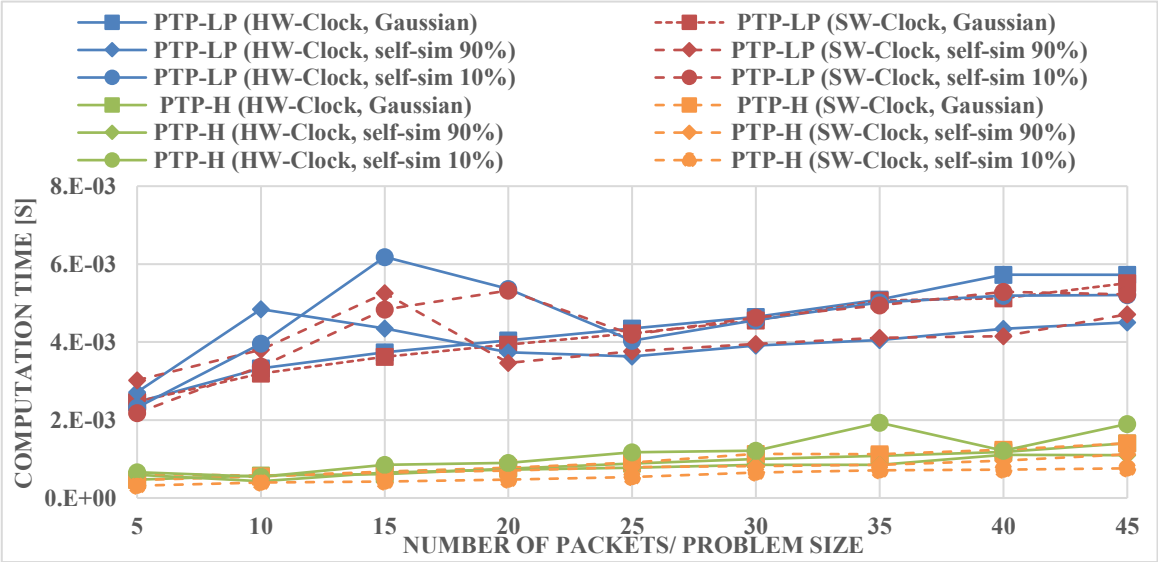


Abbildung 5.11: Rechenzeit für PTP-LP und PTP-H (die Y-Achse ist logarithmisch dargestellt).

5.8 Zwischenfazit

In diesem Kapitel wurde der LP-basierte Ansatz PTP-LP vorgestellt, um die Präzision von PTP zu erhöhen. PTP-LP ist besonders robust gegenüber Schwankungen der Paketverzögerung und voll kompatibel zum gängigen Standard PTP, da weder der PTP-Master noch die Nachrichten zwischen Master und Slave geändert werden. PTP-LP verwendet lediglich die PTP-Zeitstempel auf dem Slave-Gerät. Darüber wird die Heuristik PTP-H vorgestellt, die eine vergleichbare Genauigkeit erzielt, aber weniger rechenintensiv ist.

Es wird eine umfassende Auswertung durchgeführt unter Berücksichtigung unterschiedlicher Taktstabilitäten sowie unterschiedlicher Paketverzögerungsverteilungen. PTP-LP und PTP-H werden mit zwei Ansätzen aus dem Stand der Technik (Standard-PTP [A 63] und PTP-Kalman [A 26]) verglichen und übertreffen diese. Insbesondere zeigt sich PTP-LP bei einer hohen Auslastung robust gegenüber Verzögerungsschwankungen. In der Auswertung konnten folgende Erkenntnisse gewonnen werden:

- Die Anzahl der verwendeten Synchronisationsperioden (bzw. Pakete) ist sehr relevant für die Genauigkeit des PTP-LP-Ansatzes, von geringer Relevanz für PTP-Kalman und von keiner Relevanz für PTP.
- PTP-LP erzielt eine sehr hohe Synchronisationsgenauigkeit und übertrifft die beiden verglichenen Ansätze unter fast allen Bedingungen. Die Genauigkeit wird nur durch das vom LP-Solver festgelegte Timeout begrenzt, das für den verwendeten Solver

nicht geändert werden kann (*scipy.optimize.linprog()* [A 152]). Es wird jedoch eine allgemeine Methode im Abschnitt 5.5.3 beschrieben, um das Timeout zu vermeiden und die Anzahl von Synchronisationsperioden unter allen Bedingungen zu ermitteln. Da sich die Slave-Taktfunktion dann über eine kurze Zeit ausreichend linear verhält, erreicht PTP-LP eine hohe Synchronisationsgenauigkeit.

- PTP-LP erzielt die besten Ergebnisse für einen stabilen HW-Takt (z.B. Hardware-Oszillator) und unbekannte[5.6], nicht zu vernachlässigende Variationen der Netzwerkverzögerung. Beides sind sehr realistische Arbeitsbedingungen.

Es wurde auch für PTP-LP in C++ ein Simulationsmodell für den Netzwerksimulator OMNeT++ erstellt. Auf diese Simulationsergebnisse wird in der Evaluation aber nicht eingegangen, da sich keine signifikanten Unterschieden zur numerischen Python-Simulation zeigten. Eine komplette Implementierung von PTP-LP auf echten Geräten wurde nicht durchgeführt. Dies gilt es in Zukunft noch zu untersuchen. Hierfür wurde bereits der Zugriff auf die PTP-Hardware-Zeitstempel auf BeagleBone Boards erfolgreich getestet.

[5.6] Mit unbekannten Variationen ist gemeint, dass keine Annahme über deren Wahrscheinlichkeitsverteilung getroffen wird. Dies ist ein klarer Unterschied zu Kalman-Filtern, bei deinen Gaußverteilungen angenommen werden. Zusätzlich wird auch noch die Varianz dieser Verteilungen als bekannt vorausgesetzt. Wenn eine dieser Annahmen nicht zutrifft, sind die Schätzungen des Kalman-Filters alles andere als optimal.

6 SLMT: Zeitsynchronisation mittels linearer Optimierung, Multicasts und Temperaturkompensation

Wie bereits erwähnt, benötigen die präzisesten Zeitsynchronisationsprotokolle wie PTP und gPTP spezielle Hardware, um ihre maximale Präzision zu erreichen. Ohne diese spezielle Hardware können sie massive Paketverzögerungen, z.B. als Folge einer hohen Netzwerkauslastung, nicht ausgleichen. Es wurde in dieser Arbeit bereits gezeigt, dass Ansätze, die auf LP (lineare Optimierung, engl. linear programming) basieren, dieses Problem mildern können. Änderungen der Taktfrequenz führen jedoch zu nichtlinearen Taktfunktionen, die durch LP-basierte Ansätze allein nicht gut kompensiert werden können. Als Konsequenz wird in diesem Kapitel der SLMT-Ansatz vorgestellt, der LP, Multicasts und Temperaturkompensation für die Zeitsynchronisation verwendet. Nach bestem Wissen des Autors ist SLMT der erste Synchronisationsansatz, der LP und One-Way-Exchange bzw. Multicasts kombiniert. Folglich ist SLMT hinsichtlich der Anzahl von Nachrichten effizient. Darüber hinaus ist SLMT nach bestem Wissen des Autors der erste Synchronisationsansatz, der LP mit einer Temperaturkompensation kombiniert, um den konzeptionellen Nachteil von LP bei nichtlinearen Taktfunktionen zu verringern. In einer umfassenden Evaluation und im Vergleich mit vielen Ansätzen aus dem Stand der Technik wird gezeigt, dass SLMT diese Ansätze übertrifft, insbesondere unter rauen Bedingungen wie schnellen Temperaturänderungen und unbekannten, nicht zu vernachlässigenden Netzwerkverzögerungen bzw. hoher Netzwerklast.

In Tabelle 6.1 finden sich die konzeptionellen Unterschiede zwischen SLMT, PTP-LP und PSPI-Sync. Im Vergleich zu PTP-LP ist SLMT nicht mehr kompatibel mit dem PTP-Standard, dafür ermöglicht SLMT die Verwendung von Broadcasts für die Synchronisation. Zusätzlich kommt die IMM-Schätzung zur Temperaturkompensation hinzu.

Eine erste Version des SLMT-Ansatzes wurde auf der IEEE GLOBECOM Konferenz im Dezember 2019 veröffentlicht [B 3]. Das folgende Kapitel basiert auf dieser Veröffentlichung, enthält allerdings auch viele bisher unveröffentlichte Inhalte.

Ansatz	**Kompatibel mit Standards**	**Sync mit Broadcasts**	**Schätzer für Offset**	**Schätzer für Drift**	**Schätzer für Temperatur**
PSPI-Sync	nein	ja	RTT+LGS	-	-
PTP-LP	ja	nein	LP	LP	-
SLMT	nein	ja	LP	LP	IMM

Tabelle 6.1: Konzeptionelle Unterschiede zwischen SLMT, PTP-LP und PSPI-Sync

6.1 Einleitung und Motivation

Wie bereits erläutert, wird bei der Zeitsynchronisation die Zeit eines Slave-Geräts mit einer Master- oder Referenzzeit synchronisiert. Die beiden Hauptherausforderungen bestehen darin, die Ungenauigkeiten des Taktes selbst und die des Kommunikationskanals zu kompensieren, da beide in der Realität nicht ideal sind. Ursachen für Ungenauigkeiten und

Nichtlinearitäten der Slave-Zeit sind: Quantisierung, Frequenzänderungen (z. B. aufgrund von Temperatureffekten [A 15, 5]), zufällige Variationen bei jedem Tick (Jitter), zufälliger Frequenzgang (Wander) und Alterungseffekte in langen Zeiträumen. Ursachen für Ungenauigkeiten und Abweichungen im Kommunikationskanal sind: variable Netzwerkverzögerungen (z. B. aufgrund von Queues) und variable Verarbeitungsverzögerungen in den Netzwerk-Stacks der Geräte. Die Genauigkeit eines Synchronisationsansatzes ist die verbleibende Zeitdifferenz, die durch die Synchronisation nicht ausgeglichen werden kann.

In diesem Kapitel wird ein neuartiger Ansatz vorgestellt, der SLMT abgekürzt wird (Zeitsynchronisation mittels linearer Optimierung, Multicasts und Temperaturkompensation, engl. time synchronization with linear programming, multicasts and temperature compensation). Im Vergleich zu Ansätzen wie PTP und gPTP kann SLMT eine sehr hohe Präzision erreichen, ohne spezielle Netzwerk-Hardware (z.B. Switches) zu verwenden. SLMT benötigt zwar keine Hardware-Zeitstempel auf den Endgeräten, allerdings können diese verwendet werden, falls sie verfügbar sind. Darüber hinaus gibt der gPTP-Standard an, dass die Präzision von gPTP auf 7 Hops beschränkt ist. Im Gegensatz dazu eignet sich der SLMT-Ansatz sehr gut für große Topologien mit unbekannter Infrastruktur (z. B. Virtual Private Networks (VPNs) über das Internet). Es gilt zu beachten, dass SLMT ein neuartiger und ganzheitlicher Ansatz ist. Die Idee, LP mit Temperaturkompensation zu kombinieren, ist jedoch auch eine gültige und vielversprechende Erweiterung von PTP (bzw. PTP-LP aus Kapitel 5). Eine Prototyp-Implementierung könnte das Nachrichtenformat von PTP, die Hardware-Zeitstempel von PTP und eine PTP-Software-Implementierung (z. B. LinuxPTP oder PTPd) als Ausgangspunkt verwenden.

Mögliche Anwendungen von SLMT sind zum einen Experimente mit großer räumlicher Ausbreitung und vielen verteilten Messpunkten wie das Fusionsexperiment W7-X [A 12]. Hier sollten die Messungen so genau wie möglich mit einem Zeitstempel versehen werden. Darüber hinaus soll nach Möglichkeit einfache und standardisierte Netzwerk-Hardware verwendet werden, um die Kosten zu senken und die Zukunftssicherheit zu verbessern. Meist wird Netzwerk-Hardware möglichst über einen längeren Zeitraum von ca. 10-20 Jahren betrieben und etablierte Standards überleben zumeist proprietäre Lösungen [A 153]. Andererseits sind VPNs über das Internet eine mögliche Anwendung für SLMT, wenn verschiedene Messstellen so genau wie möglich über das Internet synchronisiert werden sollen.

Der Ansatz lässt sich wie folgt zusammenfassen. SLMT besteht aus zwei Phasen. In der *Delay*-Phase werden LP und bidirektionaler Nachrichtenaustausch verwendet, um die minimale Verzögerung zwischen Master und Slave zu schätzen. Die *Delay*-Phase muss nur einmal ausgeführt werden, solange sich das Netzwerk nicht ändert. In der *Sync*-Phase werden LP- und unidirektionaler, Multicast-basierter Nachrichtenaustausch verwendet, um Master und Slave zu synchronisieren. Dies führt zu einer hohen Effizienz bzgl. der Anzahl der Nachrichten, daher eignet sich SLMT auch für sehr große Netzwerke. LP nutzt implizite Randbedingungen, die durch die Zeitstempel gegeben sind, um die Schätzung der Taktfunktion zu verbessern. Somit ist die LP-basierte Synchronisation sehr robust gegenüber Schwankungen der Verzögerung, wie in [A 61] und [B 5] gezeigt wird. Darüber hinaus verwendet SLMT eine Temperaturkompensation, um die Präzision der LP weiter zu verbessern. Hierbei ist zu beachten, dass es vorteilhaft ist, Temperaturkompensation und LP zu entkop-

peln, da die Nichtlinearitäten von Temperatur und Takt (bzw. Frequenz) stark korrelieren und Temperaturinformationen für nahezu jedes Gerät verfügbar sind.

Die Beiträge des Kapitels lassen sich wie folgt zusammenfassen:

- Nach bestem Wissen des Autos wird die erste Kombination aus LP und Multicasts (oder unidirektionalem Nachrichtenaustausch) in einem Zeitsynchronisationsansatz vorgeschlagen.
- Darüber hinaus wird nach bestem Wissen des Autors die erste Kombination aus LP und Temperaturkompensation für die Synchronisation vorgeschlagen. Dies ist wichtig, um einem konzeptionellen Nachteil von LP zu begegnen: Nichtlinearitäten der Taktfunktion können nicht kompensiert werden, da LP lineare Funktionen schätzt. Infolgedessen ist es ausreichend, nur einen hoch genauen temperaturstabilisierten Oszillator (der in der Regel mehr als 100 US-Dollar kostet) für den Master bereitzustellen und für alle Slave-Geräte eine algorithmische Temperaturkompensation zu verwenden.
- Es wird eine umfassende Evaluation durchgeführt, bei der unterschiedliche Taktstabilitäten und verschiedene Wahrscheinlichkeitsverteilungen für die Netzwerkverzögerung verwendet werden, um SLMT mit konkurrierenden Ansätzen zu vergleichen.

6.2 Vergleich mit dem Stand der Technik

Wie schon in den vorherigen Kapiteln, wird in diesem Abschnitt erneut vorwiegend auf die Unterschiede zwischen dem Stand der Technik und SLMT eingegangen wird. Für eine detailliertere Beschreibung und Einordnung der einzelnen Ansätze sei erneut auf Kapitel 3 verwiesen.

Wie bereits erwähnt, ist NTP (Network Time Protocol) [A 62] das im Internet am häufigsten verwendete Synchronisationsprotokoll. Es ist jedoch bspw. nicht für Echtzeit-IIoT-Anwendungen geeignet, da es sich um einen rein software-basierten Ansatz handelt, dessen Genauigkeit abnimmt, wenn Verzögerungsschwankungen auftreten [A 61].

Der PTP-Standard [A 63] benötigt, wie bereits ausgeführt, spezielle Hardware, um eine hohe Präzision zu erzielen. Verzögerungsvariationen verringern die Genauigkeit von PTP, wie bspw. in [A 22] praktisch untersucht wurde. Solche Variationen können auftreten, wenn Switches PTP nicht auf der MAC-Ebene unterstützen. Von PTP hat die Time-Sensitive Networking (TSN) Gruppe der IEEE das Protokoll gPTP als TSN-Substandard abgeleitet. Im Gegensatz zu PTP müssen bei gPTP alle Switches gPTP auf der MAC-Ebene unterstützen. Die in dieser Arbeit vorgestellten LP-basierten Ansätze können jedoch die Schwankungen der Netzwerkverzögerung bei den Slave-Geräten ausgleichen und benötigen daher im Gegensatz zu PTP und gPTP keine speziellen Switches zum Erreichen hoher Genauigkeiten.

Wie bereits dargestellt, kombiniert der viel beachtete Ansatz PTP-Kalman PTP und Kalman-Filterung, um verschiedene Fehlerquellen auszugleichen [A 26]. Ein Nachteil von PTP-Kalman ist, dass diese Unsicherheiten (modelliert als Rauschen) für die Einstellung des Kalman-Filters von vornherein bekannt sein müssen, um Präzision und Stabilität des Filters sicherzustellen. Darüber hinaus sind Kalman-Filter nur für gaußsche Unsicherheiten optimal. Dies sind Hauptprobleme in realistischen Szenarien, da die Verzögerung einem

selbstähnlichen Verhalten folgt [A 91] und Taktnichtlinearitäten mit der Temperatur korrelieren [A 92].

In [A 34] schlagen Yang et al. einen Ansatz vor, der auf einem Interacting-Multiple-Model-Kalman-Filter (IMM) basiert, um die Drift abzuschätzen (Yang10) und erweitern diesen in [A 15] um Temperaturinformationen (EACS). In dieser Forschungsarbeit werden Yang10 und EACS zu einer IMM-basierten Temperaturkompensation kombiniert und in einer Version von SLMT verwendet. Wie in der Evaluation gezeigt, übertrifft SLMT die Ansätze Yang10 und EACS.

Der TCTS-Ansatz [A 92, 111] misst (offline oder online) die Drift für eine bestimmte Temperatur und verwendet diese Information zur Driftkompensation. In einer zweiten Version von SLMT wird TCTS zur Temperaturkompensation verwendet und mit der IMM-Version verglichen.

Mit dem HUYGENS-Ansatz [A 5] wird versucht eine präzise Synchronisation ohne spezielle Hardware zu erreichen. Hierbei werden verrauschte Zeitstempel zunächst gefiltert und dann mittels Support Vector Machines (SVMs) verarbeitet. Ein wesentlicher Unterschied besteht also darin, dass HUYGENS SVMs zur Schätzung von Drift und Offset verwendet, während SLMT hierfür LP nutzt. Im Gegensatz zu SLMT verwendet HUYGENS jedoch keine Temperaturkompensation und trifft die Annahme, dass die Zeitfunktion stückweise linear sei. Dies führt zu Fehlern, wenn hohe Temperaturgradienten auftreten, wie sie z.B. in [A 92] gemessen wurden.

Die Autoren in [A 118] schlagen einen Ansatz zur Positionierung und Zeitsynchronisation in 5G-Netzen vor, der auf einer kaskadierten Struktur von erweiterten Kalman-Filtern (EKFs) basiert. Im Gegensatz zu [A 118] werden die Schätzungen bei SLMT auf den Slave-Geräten durchgeführt, und solch ein dezentrale Ansatz ist immer skalierbarer. Darüber hinaus würde die Genauigkeit von EKFs unter den typischerweise nicht-gaußschen Verzögerungen in drahtgebundenen Netzen leiden.

PSPI-Sync wurde in Kapitel 4 vorgestellt und ist ein skalierbarer, präziser und plattformunabhängiger Synchronisationsansatz. PSPI-Sync teilt die Synchronisation ebenfalls in zwei Phasen auf: eine Delay Phase und die eigentliche Broadcast-basierte Sync Phase. Da PSPI-Sync mittlere Verzögerungen verwendet, ist es nicht sehr robust gegenüber selbstähnlichen Verzögerungen. Darüber hinaus wurde die Robustheit gegenüber Taktinstabilität bisher nicht bewertet.

PTP-LP wurde in Kapitel 5 vorgestellt und ist ein Ansatz, der PTP und LP kombiniert. Im Gegensatz zu PTP-LP unterstützt SLMT Multicasts für eine effizientere Synchronisation. Darüber hinaus verbessert die Temperaturkompensation von SLMT die Präzision des LP weiter.

6.3 Notation

Die in diesem Kapitel verwendete Notation ist zwar äquivalent zu Kapitel 5, soll der Vollständigkeit halber an dieser Stelle erneut beschrieben werden. Die Master- (oder Referenz-) Zeit $T(t)$ sei eine lineare Funktion mit der Steigung 1 (vgl. Gl. 155). Alle Nichtlinearitäten werden

als Teil der Slave-Zeit $C(t)$ modelliert, da die Master-Zeit per definitionem ideal ist. $C(t)$ kann zunächst auch als lineare Funktion mit der Steigung γ und dem y-Achsenabschnitt $\theta(0)$ modelliert werden (vgl. Gl. 155). Hierbei wird zunächst angenommen, dass die Taktfrequenz des Slaves stabil ist [6.1]. Später werden auch komplexere Modelle der Slave-Zeit vorgestellt, die realistischer sind, zunächst genügt aber die einfache, lineare Slave-Zeit. Der Offset $\theta(t)$ ist definiert als Differenz zwischen $T(t)$ und $C(t)$ (vgl. Gl. 156). Des Weiteren entspricht γ dem Quotienten aus der Taktfrequenz des Slaves f_{slave} und der Taktfrequenz des Masters f_{master}. Außerdem sei Γ definiert als die Frequenzdifferenz zwischen Master und Slave, normiert durch f_{master}. Hierbei wird Γ auch als Skew oder Drift (vgl. Gl. 157) bezeichnet. Wenn also $f_{master} = f_{slave}$ gilt, dann folgt $\Gamma = 0$. Die Drift kann mittels zweier Offset-Messungen bestimmt werden, wobei ΔT die Zeit zwischen den Messungen bezeichnet (vgl. Gl. 157).

$$T(t) = t, C(t) = \gamma \cdot t + \theta(0) \tag{155}$$

$$\theta(t) = C(t) - T(t) \tag{156}$$

$$\gamma = \frac{f_{slave}}{f_{master}} \Rightarrow \Gamma = \gamma - 1 = \frac{\theta(t + \Delta T) - \theta(t)}{\Delta T} \tag{157}$$

6.4 Der SLMT-Ansatz

In [A 92, 34, 15] wurde gezeigt, dass es möglich ist, Temperatur und Drift auszugleichen. In [A 61] und [B 5] wurde gezeigt, dass LP-basierte Ansätze die Verzögerung kompensieren können und somit den Offset sehr genau zu bestimmen. Die Hauptidee von SLMT ist es, beides in einem Ansatz zu kombinieren.

SLMT besteht aus zwei Phasen (vgl. Abb. 6.1). Da die Netzwerkverzögerung aus einem konstanten Teil (z.B. die Ausbreitungsverzögerung) und einem probabilistischen Teil (z. B. Queueing-Verzögerung) besteht, kann der konstante Teil einmal geschätzt werden (*Delay*-Phase) und dann können effiziente Multicasts für die tatsächliche Synchronisation Verwendung finden (*Sync*-Phase). Während der *Delay*-Phase sendet der Slave eine erste Nachricht an den Master (Forward Path). Der Sendezeitpunkt wird mit $C(T_1)$ und der Empfangszeitpunkt mit T_2 zeitgestempelt. Auf dem umgekehrten Weg vom Master zum Slave wird der Sendezeitpunkt mit T_3 und der Empfangszeitpunkt mit $C(T_4)$ zeitgestempelt (Reverse Path). Zu beachten sei, dass $C(T_1)$ und $C(T_4)$ in Bezug auf die Slave-Zeit genommen werden und die tatsächlichen Zeiten am Master T_1 und T_4 unbekannt sind. SLMT nutzt das Wissen, dass $P = (T_2, C(T_1))$ immer unter $C(t)$ und $P' = (T_3, C(T_4))$ immer über $C(t)$ sein muss, um $C(t)$ zu schätzen. Wie in Abb. 6.1 dargestellt, liegen P oder P' sehr nahe an $C(t)$, wenn die Verzögerung sehr gering ist. Wenn ein Paket nur minimal verzögert wird, verwendet der LP-Solver diese Informationen, um die Genauigkeit der Schätzung zu erhöhen. Während der *Sync*-Phase sendet der Master Multicast-Nachrichten an alle Slaves (Synchronization Path). Der Sendezeitpunkt wird mit T_5 und der Empfangszeitpunkt mit $C(T_6)$ versehen. Hier wird das Wissen verwendet, dass $P'' = (T_5, C(T_6))$ über $C(t)$ liegen muss.

[6.1] Genauer gesagt wird angenommen, dass die Slave-Zeit bzgl. der Master-Zeit stabil ist, d.h. die Gleichung $T(t)' - C(t)' = 1 - \gamma = const$ muss erfüllt sein.

Darüber hinaus wendet SLMT eine Temperaturkompensation auf die Slave-Zeit an. Hieraus ergibt sich folgender Vorteil. Mittels LP kann zwar die probabilistische Verzögerung vollständig kompensiert werden, wenn ein Punkt $((T_i, C(T_i))$ so nahe wie $y \cdot d_{min}$ an $C(t)$ liegt. Wobei d_{min} der minimale, konstante Teil der Paketverzögerung ist (z.B. die Ausbreitungsverzögerung). Die Genauigkeit der LP-Schätzung nimmt jedoch ab, sobald die Slave-Zeit nichtlinear wird. Daher linearisiert SLMT die Slave-Zeit mithilfe einer Temperaturkompensation, um die Ergebnisse des LP-Solvers zu verbessern. Die Temperaturkompensation wird parallel während der *Delay*- und *Sync*- Phasen ausgeführt. Wenn das Slave-Gerät seine Taktfrequenz nicht steuern kann, um die Temperatur so zu kompensieren, schätzt er eine temperaturkompensierte virtuelle Zeit $C'(t)$ und die Zeitstempel am Slave werden in Bezug auf $C'(t)$ genommen. Der Speicheraufwand ist sehr gering, da der Slave immer nur den Offset zwischen seiner physischen Zeit $C(t)$ und seiner virtuellen Zeit $C'(t)$ speichert.

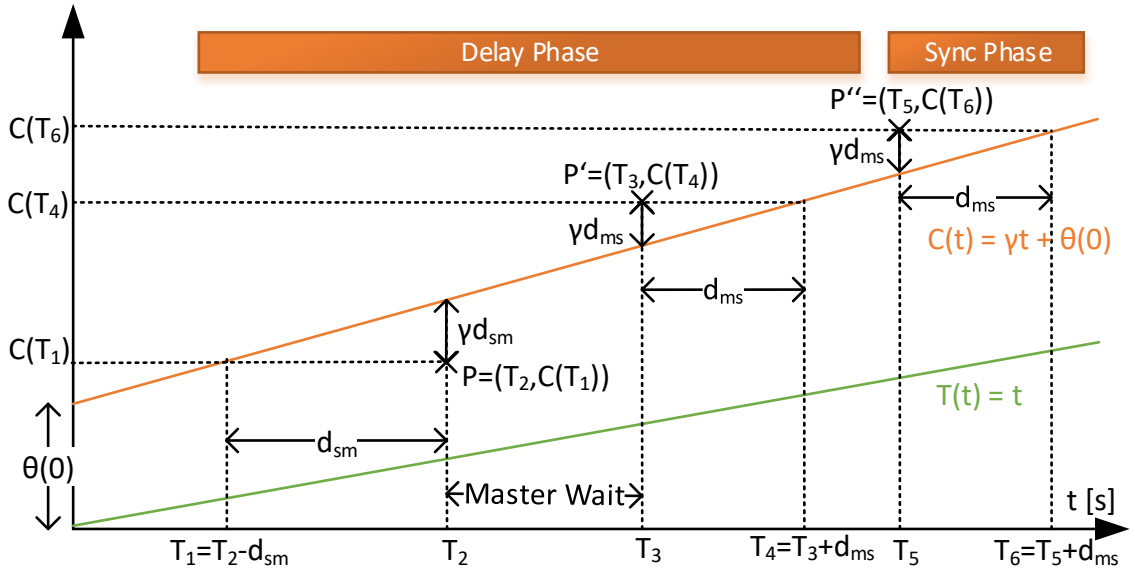


Abbildung 6.1: Zeiten am Master $T(t)$ und Slave $C(t)$ mit Zeitstempeln (der Einfachheit halber zeigt diese Abbildung die Zeitstempel von nur einer Synchronisationsperiode)

6.4.1 Problemformulierung

In diesem Abschnitt wird auf die Problemformulierung eingegangen, d.h. die Umwandlung der Überlegungen des vorherigen Abschnitts in ein Problem der linearen Optimierung (LP).

SLMT schätzt die Slave-Zeit $C(t)$ anhand von $T(t)$ als Referenz. Genauer gesagt werden wieder der Anstieg γ und der Schnittpunkt mit der y-Achse $\theta(0)$ geschätzt. Daher wird das Multicast-based Clock Function Determination Problem (MCFD-P) formuliert. Dazu finden die Zeitstempel $C(T_1^n)$, T_2^n, T_3^n, $C(T_4^n)$, T_5^n und $C(T_6^n)$ aus N Synchronisationsperioden ($n \in [1, ..., N]$) als Bedingungen für das LP Verwendung.

Delay Phase - Forward Path und untere Grenze, Reverse Path und obere Grenze: Da der erste Schritt der *Delay*-Phase ähnlich zu [A 61] und Kapitel 5 ist, wird eine detaillierte Beschreibung der hier erfolgenden Schätzung von $f_{lb}(t)$ und $f_{ub}(t)$ übersprungen, wobei es

sich um die untere und obere Schranke von $C(t)$ handelt. Trotzdem werden nochmal kurz die wichtigsten Berechnungen angegeben.

Auf dem Forward Path wird die untere Grenze $f_{lb}(t)$ wie folgt angenommen:

$$f_{lb}(t) = \alpha_1 \cdot t + \beta_1 \tag{158}$$
$$\alpha_1 = \gamma, \beta_1 = \theta(0) - \gamma \cdot d_{ms} \tag{159}$$

Um diese zu finden werden dem LP-Solver folgende Constraints und die Minimierungsfunktion $f(\alpha_1, \beta_1)$ übergeben:

$$\alpha_1 T_2^n + \beta_1 \geq C(T_1^n), \forall n \in [1, ..., N] \tag{160}$$
$$f(\alpha_1, \beta_1) = \sum_{n=1}^{N} (\alpha_1 T_2^n + \beta_1 - C(T_1^n)) \tag{161}$$

Auf dem Reverse Path wird die obere Grenze $f_{ub}(t)$ wie folgt angenommen:

$$f_{ub}(t) = \alpha_2 \cdot t + \beta_2 \tag{162}$$
$$\alpha_2 = \gamma, \beta_2 = \theta(0) + \gamma \cdot d_{sm} \tag{163}$$

Um diese zu finden werden dem LP-Solver folgende Constraints und die Minimierungsfunktion $f(\alpha_2, \beta_2)$ übergeben:

$$\alpha_2 T_3^n + \beta_2 \leq C(T_4^n), \forall n \in [1, ..., N] \tag{164}$$
$$f(\alpha_2, \beta_2) = \sum_{n=1}^{N} (C(T_4^n) - \alpha_2 T_3^n - \beta_2) \tag{165}$$

Delay Phase - Schätzung von C (t) und Ausbreitungsverzögerung: Ähnlich wie bei [A 61] und [B 5] wird $C(t)$ geschätzt als $\hat{C}_{delay}(t)$ unter Verwendung von $f_{lb}(t)$ und $f_{ub}(t)$ (vgl. Gl. 166 und Gl. 167).

$$\hat{\gamma}_{delay} = \frac{\alpha_1 + \alpha_2}{2} \tag{166}$$
$$\hat{\theta}_{delay}(0) = \frac{\beta_1 + \beta_2}{2} \tag{167}$$

Die Ansätze aus [A 61] und [B 5] enden hier. Im Gegensatz dazu wird bei SLMT $\hat{C}_{delay}(t)$ verwendet, um die minimale Verzögerung $dmin_{ms}$ zwischen Master und Slave zu schätzen. Genauer gesagt wird das Produkt $\gamma \cdot dmin_{ms}$ geschätzt, welches der Projektion von $dmin_{ms}$ auf die y-Achse entspricht (vgl. Gleichung 169 und 170).

$$\hat{\gamma}_{delay} = \frac{\alpha_1 + \alpha_2}{2}, \hat{\theta}_{delay}(0) = \frac{\beta_1 + \beta_2}{2} \tag{168}$$

$$P_{rev} = \{(T_3^1, C(T_4^1)), ..., (T_3^N, C(T_4^N))\} \tag{169}$$

$$\gamma \cdot dmin_{ms} = \min_{(T_3^n, C(T_4^n)) \in P_{rev}} (C(T_4^n) - \hat{C}_{delay}(T_3^n)) \tag{170}$$

Es wird folglich die Menge P_{rev} aller Zeitstempelpunkte vom Reverse Path genutzt, um die minimale Differenz zwischen einem Punkt in P_{rev} und $\hat{C}_{delay}(t)$ zu ermitteln (vgl. Abb. 6.1).

Sync Phase - obere Grenze: In der *Sync*-Phase können Multicasts bzw. unidirektionaler Nachrichtenaustausch verwendet werden, um $C(t)$ erneut zu schätzen. Genauer gesagt wird eine neue Obergrenze für $C(t)$ geschätzt und um $\gamma \cdot dmin_{ms}$ nach unten verschoben. Das Ergebnis der LP-Schätzung für $C(t)$ ist $\hat{C}_{sync}(t) = \hat{\gamma}_{sync} \cdot t + \hat{\theta}_{sync}(0)$ (vgl. Gl. 173). Zu diesem Zweck minimiert der LP-Solver die Funktion $f(\alpha, \beta)$ (vgl. Gl. 172) unter den Bedingungen von Gl. 171. Dies bedeutet im Grunde, dass $\hat{C}_{sync}(t) + \gamma \cdot dmin_{ms}$ so nah wie möglich an den Constraint-Punkten $(T_5^n, C(T_6^n))$ liegen sollte.

$$\alpha T_5^n + \beta \leq C(T_6^n) - \gamma \cdot dmin_{ms}, \forall n \in [1, ..., N] \tag{171}$$

$$f(\alpha, \beta) = \sum_{n=1}^{N} (C(T_6^n) - \gamma \cdot dmin_{ms}) - (\alpha T_5^n + \beta) \tag{172}$$

$$\hat{\gamma}_{sync} = \alpha, \hat{\theta}_{sync}(0) = \beta \tag{173}$$

6.4.2 Heuristiken für die LP-Berechnung

Darüber hinaus werden in dieser Forschungsarbeit Heuristiken für alle LP-Berechnungen vorgeschlagen, um den Rechenaufwand zu verringern. Hierfür wird zunächst die lineare Regression aller Constraint-Punkte berechnet. Im Anschluss wird diese lineare Regressionsfunktion in Richtung desjenigen Constraint-Punktes verschoben, welcher $C(t)$ am nächsten liegt. Auch hier findet das bereits beschriebene implizite Wissen über die Positionen der Punkte P, P' und P'' Verwendung (vgl. Abb. 6.1).

Delay Phase - Forward Path und untere Grenze: Auf dem Forward Path wird die untere Grenze $f_{lb}(t)$ wie folgt geschätzt:

$$P_{df} = \{(T_2^1, C(T_1^1)), ..., (T_2^N, C(T_1^N))\} \tag{174}$$

$$\hat{f}_{lb}(t) = f_{lrdf}(t) + m_{df} \tag{175}$$

$$m_{df} = \max_{(T_2^n, C(T_1^n)) \in P_{df}} (C(T_1^n) - f_{lrdf}(T_2^n)). \tag{176}$$

Hierbei ist P_{df} die Menge der Constraint-Punkte des Forward Paths und $f_{lrdf}(t)$ deren lineare Regression. Außerdem bezeichnet $\hat{f}_{lb}(t)$ die Schätzung der unteren Grenze und m_{df} die maximale y-Distanz zwischen einem Punkt in P_{df} und $f_{lrdf}(t)$ (vgl. Abb. 6.1).

Delay Phase - Reverse Path und obere Schranke: Auf dem Reverse Path wird die obere Schranke $f_{ub}(t)$ wie folgt abgeschätzt:

$$P_{dr} = \{(T_3^1, C(T_4^1)), ..., (T_3^N, C(T_4^N))\} \tag{177}$$

$$\hat{f}_{ub}(t) = f_{lrdr}(t) - m_{dr} \tag{178}$$

$$m_{dr} = \max_{(T_3^n, C(T_4^n)) \in P_{dr}} (f_{lrdr}(T_3^n) - C(T_4^n)). \tag{179}$$

Hier ist P_{dr} die Menge der Constraint-Punktes vom Reverse Path und $f_{lrdr}(t)$ ist deren lineare Regression. Außerdem ist $\hat{f}_{ub}(t)$ die Schätzung der oberen Schranke und m_{dr} die maximale Distanz in y-Richtung zwischen einem Punkt in P_{dr} und $f_{lrdr}(t)$ (vgl. Abb. 6.1).

Sync Phase - Sync Path und obere Schranke: Auf dem Sync Path wird die obere Schranke $f_s(t)$ wie folgt geschätzt:

$$P_s = \{(T_5^1, C(T_6^1)), ..., (T_5^N, C(T_6^N))\} \tag{180}$$

$$\hat{f}_s(t) = f_{lrs}(t) - m_s \tag{181}$$

$$m_s = \max_{(T_5^n, C(T_6^n)) \in P_s} (f_{lrs}(T_5^n) - C(T_6^n)) \tag{182}$$

$$\hat{C}_{sync}(t) = \hat{f}_s(t) - \gamma \cdot dmin_{ms} \tag{183}$$

Hierbei bezeichne P_s die Menge der Constraint-Punkte vom Sync Paths und $f_{lrs}(t)$ deren lineare Regression. Des Weiteren bezeichne $\hat{f}_s(t)$ die Schätzung der oberen Schranke und m_s den maximalen Abstand in y-Richtung zwischen einem Punkte in P_s und $f_{lrs}(t)$ (vgl. Abb. 6.1). Final wird $\hat{C}_{sync}(t)$ durch das Verschieben von $\hat{f}_s(t)$ um $\gamma \cdot dmin_{ms}$ geschätzt. Es sei angemerkt, dass es sich bei $\gamma \cdot dmin_{ms}$ um die auf die y-Achse projizierte minimale Verzögerung zwischen Master und Slave $dmin_{ms}$ handelt, welche mittels Gl. 170 bestimmt wurde.

6.4.3 Temperaturkompensation

Für die Temperaturkompensation finden zwei Ansätze Verwendung: Ein komplexer Ansatz, der auf einem IMM basiert und der relativ einfache TCTS-Ansatz [A 111]. Grundsätzlich sind beides bereits vorhandene Ansätze zur Temperaturkompensation. Die neuartige Idee dieser Forschungsarbeit ist es jedoch, mit Hilfe dieser Ansätze die Funktion $C'(t)$ zu berechnen. Hierbei ist $C'(t)$ eine temperaturkompensierte und somit linearisierte Version von $C(t)$ (vgl. Abb. 6.2). Der Slave kompensiert permanent die Temperatur und nimmt Zeitstempel auf, die sich auf $C'(t)$ anstelle von $C(t)$ beziehen. Die in den vorherigen Abschnitten beschriebenen Verarbeitungsschritte werden dann mit diesem, auf $C'(t)$ statt auf $C(t)$ bezogenen, Zeitstempeln durchgeführt.

Falls das Slave-Gerät die Möglichkeit besitzt, die eigene Taktfrequenz zu steuern, kann dies natürlich alternativ geschehen. Hierzu muss lediglich die im Rahmen der Temperaturkom-

pensation erfolgende Driftschätzung in die entsprechende Steuerspannung für den Taktoszillator umgerechnet werden. Da diese Möglichkeit der Taktsteuerung jedoch eine erhebliche Anforderung an die Hardware-Plattform darstellt, wird im Folgenden davon ausgegangen, dass der Slave lediglich in Software die virtuelle Zeit $C'(t)$ berechnet. Der Speicheraufwand ist sehr gering, da der Slave immer nur den Offset $\Theta_{temperature}(t)$ zwischen seiner physischen Zeit $C(t)$ und seiner virtuellen Zeit $C'(t)$ speichert (vgl. Abb. 6.2) [6.2]. Mittels $\Theta_{temperature}(t)$ kann der Slave zu jeder Zeit $C'(t)$ aus $C(t)$ berechnen.

IMM-Mixed: In dieser Arbeit wird eine Kombination aus Yang10 [A 34] und EACS [A 15] zur IMM-basierten Temperaturkompensation mit dem Namen IMM-Mixed vorgeschlagen. IMM-Mixed verwendet das Temperatur-IMM und die Struktur von EACS. Es werden jedoch die Driftmodelle von Yang10 verwendet, da sie den Offset und die Drift berücksichtigen. Darüber hinaus wird vorgeschlagen, die Messrauschmatrizen aus [A 26] zu verwenden, da sie genauer sind als jene aus den Arbeiten [A 34] und [A 15]. IMM-Mixed alleine ist ein neuartiger Synchronisationsansatz.

Der TSTC-Ansatz als Heuristik: Der von [A 111] vorgeschlagene einfache TCTS-Ansatz besteht darin, die Drift für eine bestimmte Temperatur (entweder offline oder online) zu messen und diesen Messwert zu verwenden, um $C(t)$ zu kompensieren.

6.5 Taktmodelle

In diesem Abschnitt werden die realistischen Taktmodelle beschrieben, die im Rahmen der Evaluation verwendet wurden. Da numerische Simulationen durchgeführt wurden, ist der Slave-Takt standardmäßig ideal. Um realistische Ergebnisse zu erzielen, wurden Taktmodelle implementiert, welche die Taktungenauigkeiten realer Geräte reproduzieren.

6.5.1 Random-Walk-Taktmodell

Giorgi et al. schlagen in [A 26] das bereits in Kapitel 5 beschriebene Random-Walk-Modell vor. Hier wird der Offset des nachfolgenden Taktschrittes $\theta(s+1)$ unter Verwendung des Offsets des aktuellen Taktschritt $\theta(s)$, der Drift $\Gamma(s)$ und der Taktschrittweite T_{tick} berechnet. Außerdem tritt bei jeder Iteration eine gaußverteilte Unsicherheit w_θ auf (z. B. ein Jitter der Periode des Taktgebers). In ähnlicher Weise wird die Drift $\Gamma(s+1)$ unter Verwendung der Drift des aktuellen Taktschrittes $\Gamma(s)$ und der gaußverteilten Unsicherheit w_Γ berechnet, die z. B. eine langsame Frequenzänderung bewirkt:

[6.2] Eine Implementierung auf echten Geräten wäre z.B. vereinfacht möglich mittels $\Theta_{temperature}(s) = \Theta_{temperature}(s-1) - \hat{\Gamma}(T) \cdot \Delta t$. Hierbei ist s der aktuelle Iterationsschritt, $\Theta_{temperature}(s-1) = C'(t) - C(s)$ der Offset zwischen realer und temperaturkompensierter Zeit am Slave und Δt das Intervall in welchem $\Theta_{temperature}$ abgepasst wird (bspw. immer wenn eine neue Schätzung $\hat{\Gamma}(T)$ für die Drift vorliegt). Hierbei wird angenommen, dass die Temperatur vollständig kompensiert wird und somit $C'(t)$ parallel zu $T(t)$ verläuft ($\Gamma'(s) = 0$). Ansonsten würde $\Theta_{temperature}(s) = \Theta_{temperature}(s-1) + (\Gamma'(s) - \hat{\Gamma}(T)) \cdot \Delta t$ gelten. Allerdings ist es im Allgemeinen auf realen Geräten nicht möglich $\Gamma'(s)$, die verbleibende Drift nach der Temperaturkompensation, zu bestimmen.

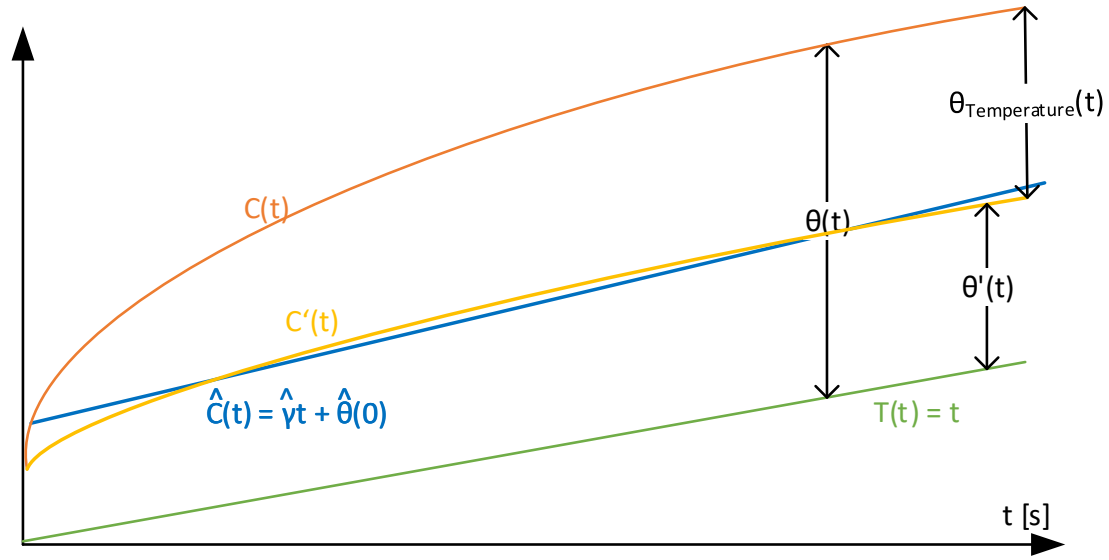


Abbildung 6.2: Master-Zeit $T(t)$, Slave-Zeit $C(t)$ und temperaturkompensierte Slave-Zeit $C'(t)$ sowie Offsets zwischen den Zeitfunktionen.

$$\theta(s+1) = \theta(s) + \Gamma(s) \cdot T_{tick} + w_\theta(s) \tag{184}$$
$$\Gamma(s+1) = \Gamma(s) + w_\Gamma(s). \tag{185}$$

6.5.2 Temperaturtaktmodell

Weiterhin wird hier ein neues Temperaturmodel vorgeschlagen. Unter Bezugnahme auf [A 15] ist die Temperatur der wichtigste Grund für Änderungen der Frequenz und somit der Drift. Die Berechnung des Offsets ist äquivalent zum Random-Walk-Modell. Die Drift $\Gamma(T)$ wird jedoch als Funktion der Temperatur T modelliert und auch hier wird eine gaußverteilte Unsicherheit w_Γ addiert. Zu beachten ist jedoch, dass sich diese nicht über die Zeit aufsummiert. Zwar ändert sich auch in diesem Modell die Frequenz mit der Zeit, diese Änderung entsteht aber maßgeblich durch Temperaturänderungen.

$$\Gamma(s+1) = \Gamma(T) + w_\Gamma(s) \tag{186}$$

6.5.3 Kompensation von Temperatur bzw. Drift

Für die in Kap. 6.4.3 vorgestellte Kompensation von Temperatur bzw. Drift kann entweder Offset oder Drift korrigiert werden, um $C'(t)$ als linearisierte Version von $C(t)$ zu berechnen. Das gleichzeitige Korrigieren von Drift und Offset würde zu einer Überkorrektur führen, da für

die Linearisierung lediglich eine der beiden Größen korrigiert werden muss. Für die Offset-Korrektur ändert Gl. 184 sich wie folgt:

$$\theta(s+1) = \theta(s) + (\Gamma(s) - \hat{\Gamma}(T)) \cdot T_{tick} + w_\theta(s). \tag{187}$$

Hierbei ist $\Gamma(T)$ die Drift als Funktion der Temperatur T und $\hat{\Gamma}(T)$ deren Schätzung (z. B. mittels der Verfahren TSTC oder IMM-Mixed). Für die Driftkompensation ändert sich Gl. 186 wie folgt:

$$\Gamma(s+1) = \Gamma(T) - \hat{\Gamma}(T) + w_\Gamma(s). \tag{188}$$

6.6 Experimente und Auswertung

In diesem Abschnitt wird eine Evaluation des SLMT-Ansatzes und seiner Variationen durchgeführt. Hierzu wird zunächst auf die Methodik eingegangen. Im Anschluss findet die Evaluation sowohl mittels Random-Walk-Taktmodell als auch mittels Temperaturtaktmodell statt. Zuletzt wird eine Untersuchung der Berechnungskomplexität vorgenommen.

6.6.1 Methodik

Es wird erneut davon ausgegangen, dass Master und Slave über einen Switch verbunden sind und es ein zusätzliches Gerät gibt, das Hintergrundverkehr erzeugt (vgl. Abb. 5.5). Dieses Gerät modelliert den gesamten Hintergrundverkehr im Netzwerk, der ggf. durch eine Vielzahl von Geräten erzeugt werden würde. Daher kann es Verkehr erzeugen, der sogar höher als die eigentliche Übertragungsrate einer einzelnen Leitung ist. Somit kann es zu einem Volllaufen des Switch-Puffers kommen. Es wurde eine numerische Simulation mit der Python SciPy-Bibliothek durchgeführt. Jedes Experiment wurde 100 Mal ausgeführt und es wurden der mittlere Synchronisationsfehler (Offset-Fehler) und der mittlere Frequenzfehler (Drift-Fehler) bestimmt. Als Synchronisationsperiode wurde eine Sekunde verwendet, was z.B. der Standardwert für PTP ist. Als Taktschrittweite T_{tick} wurde eine Millisekunde verwendet. Die Anzahl der Perioden (bzw. Pakete), die für die *Delay*- und *Sync*-Phase verwendet wurden, wurde variiert.

Taktstabilität: Es wurden die in Kap. 6.5 vorgestellten Taktmodelle verwendet. Für das Random-Walk-Modell fanden die gleichen Werte für w_θ und w_Γ Anwendung wie auch bereits in [A 26] und Kapitel 5. Wie in Kapitel 5 gezeigt wurde, führen diese Werte zu Allan-Varianzen, die auf realen Geräten gemessen wurden.

Für das Temperaturmodell wurde die in [A 135] gemessene Korrelation zwischen Temperatur und Drift verwendet (vgl. Abb. 6.3). Die Werte entsprechen einem 32KHz Oszillator SE2412CT-ND auf dem Mica2 Sensor-Board. Für die Experimente wurde ein Temperaturverlauf von 20° C auf 50° C angenommen (vgl. Abb. 6.4). Solche Temperaturänderungen

wurden z.B. in [A 92] gemessen, insbesondere bei schnellen Übergängen zwischen Sonnenlicht und Schatten außerhalb von Gebäuden. Da eine unterschiedliche Anzahl von Synchronisationsperioden ausgewertet wird, variiert die Simulationszeit t_{sim} und es findet eine Skalierung des Temperaturverlaufs statt. Der Startpunkt des Temperaturübergangs von 20 °C auf 50 °C liegt bei $0.3 \cdot t_{sim}$ und der Endpunkt bei $0.7 \cdot t_{sim}$. Folglich führt ein höherer Wert von t_{sim} zu einer langsameren Temperaturänderung. Aus dem Temperaturverlauf (linke Seite von Abb. 6.4) ergibt sich zusammen mit der Korrelation zwischen Temperatur und Frequer
ter und
halber v
nominal

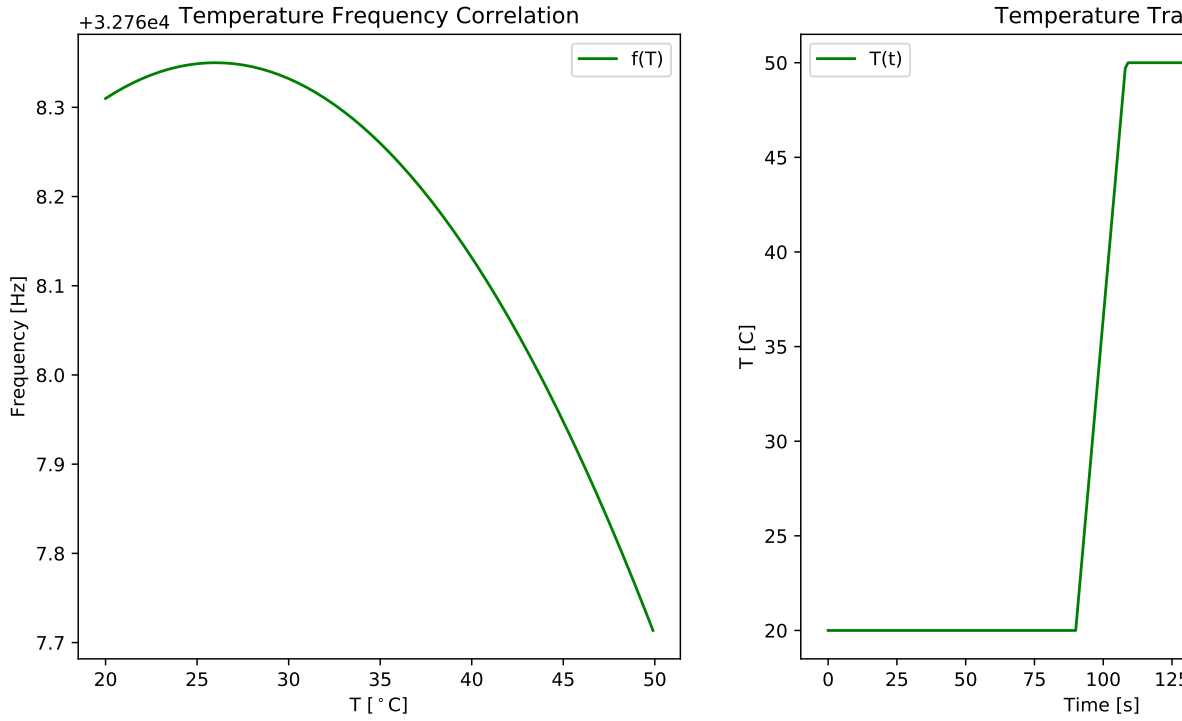
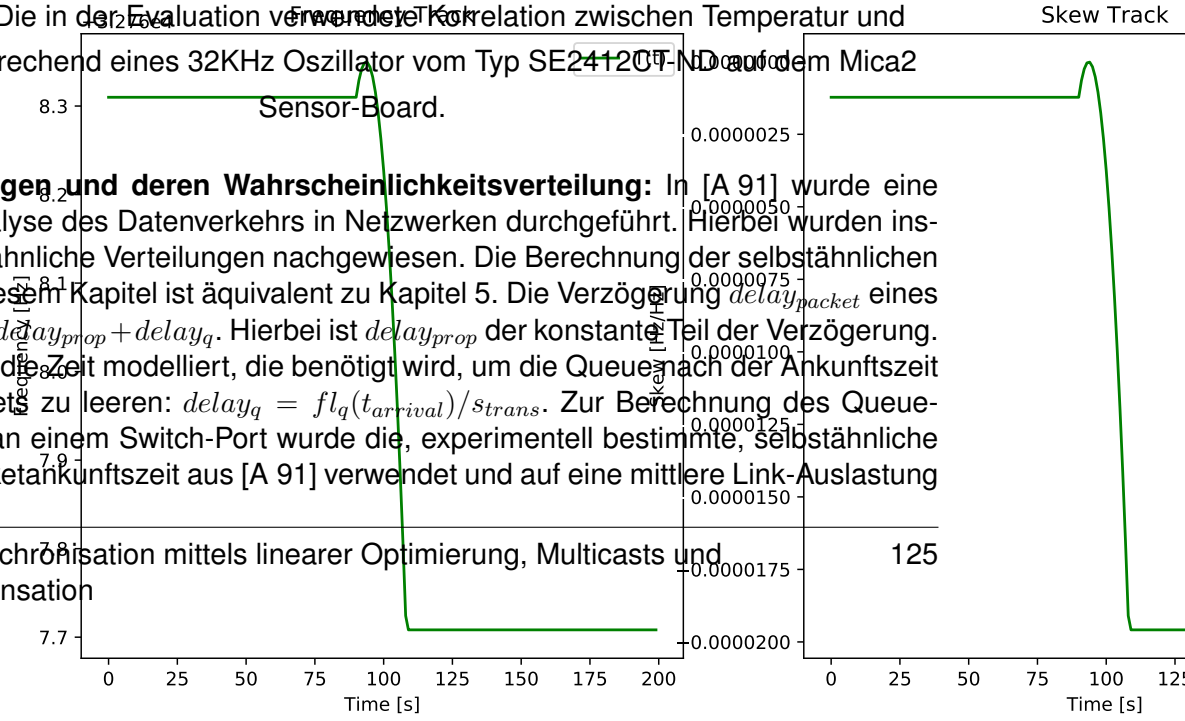


Abbildung 6.3: Die in der Evaluation verwendete Korrelation zwischen Temperatur und Frequenz, entsprechend eines 32KHz Oszillator vom Typ SE2412CT-ND auf dem Mica2 Sensor-Board.



Paketverzögerungen und deren Wahrscheinlichkeitsverteilung: In [A 91] wurde eine umfangreiche Analyse des Datenverkehrs in Netzwerken durchgeführt. Hierbei wurden insbesondere selbstähnliche Verteilungen nachgewiesen. Die Berechnung der selbstähnlichen Verzögerung in diesem Kapitel ist äquivalent zu Kapitel 5. Die Verzögerung $delay_{packet}$ eines Pakets entspricht $delay_{prop} + delay_q$. Hierbei ist $delay_{prop}$ der konstante Teil der Verzögerung. Es wird $delay_q$ als die Zeit modelliert, die benötigt wird, um die Queue nach der Ankunftszeit $t_{arrival}$ eines Pakets zu leeren: $delay_q = fl_q(t_{arrival})/s_{trans}$. Zur Berechnung des Queue-Füllstands $fl_q(t)$ an einem Switch-Port wurde die, experimentell bestimmte, selbstähnliche Verteilung der Paketankunftszeit aus [A 91] verwendet und auf eine mittlere Link-Auslastung

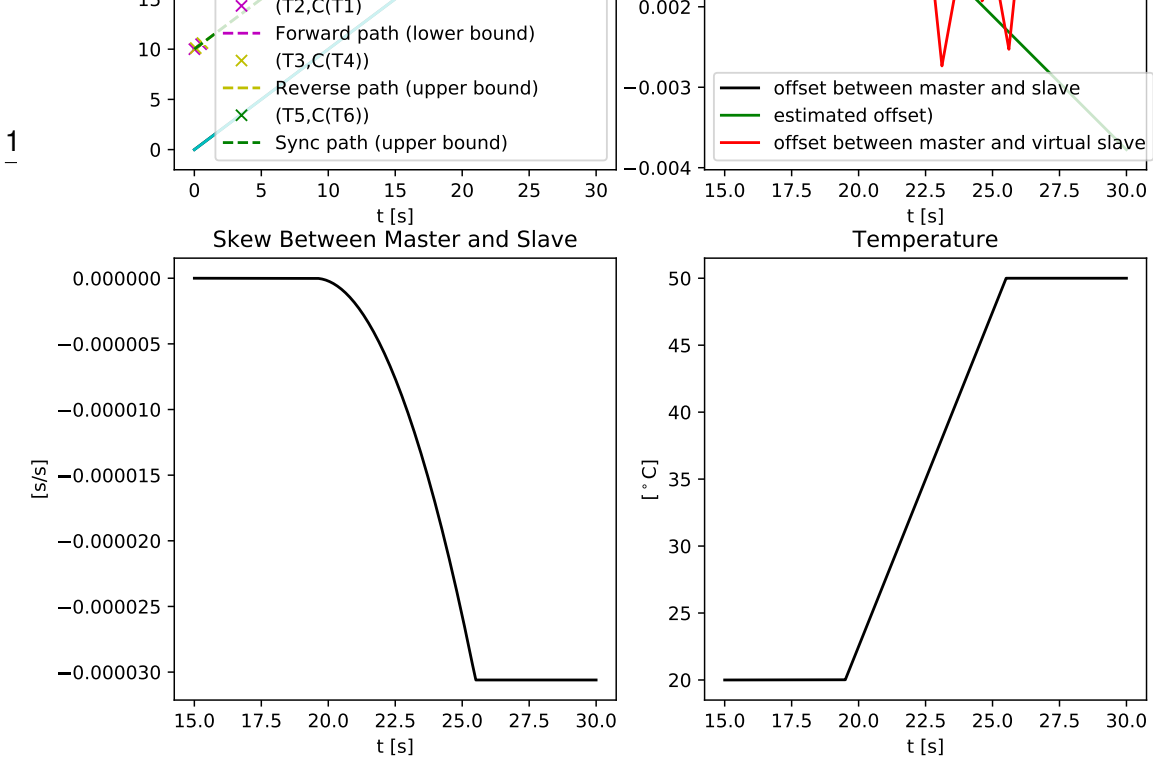


Abbildung 6.4: Beispiel für einen in der Evaluation verwendeten Temperaturverlauf (rechts). Aus diesem ergibt sich zusammen mit der Korrelation zwischen Temperatur und Frequenz (vgl. Abb. 6.3) der Skew-Verlauf (links).

skaliert (z.B. 10% oder 90%). Hierfür kam eine Paketgröße von 800 Byte (Mittelwert in [A 91]) und eine Übertragungsgeschwindigkeit s_{trans} von 1 GBit / s (für GBit Ethernet) zum Einsatz. Ein Beispiel für die sich ergebene Verteilung der Verzögerung findet sich in Abb. 6.5.

Berechnung des Synchronisations- und Frequenzfehlers: Für den SLMT-Ansatz werden Synchronisations- und Frequenzfehler wie folgt berechnet:

$$Error_{\theta} = |\theta - \hat{\theta}| = |\theta - [C(t) - \hat{T}(t)]| = |\theta - [C(t) - \frac{C(t) - \hat{\theta}(0)}{\hat{\gamma}}]| \tag{189}$$

$$Error_{\Gamma} = |\Gamma - \hat{\Gamma}| \tag{190}$$

Hierbei ist $\hat{\theta}$ die Schätzung des Offset θ, $\hat{C}(t) = \hat{\gamma} \cdot t + \hat{\theta}(0)$ die Schätzung der Taktfunktion $C(t)$ und $\hat{\Gamma} = \hat{\gamma} - 1$ deren Drift. Die Grundüberlegung hierbei ist, dass der Slave seine Taktfunktion $C(t)$ zwar nicht anpasst, aber als $\hat{C}(t)$ schätzt. Mittels $\hat{\gamma}$ und $\hat{\theta}(0)$ kann dann jederzeit die Master-Zeit als $\hat{T}(t)$ bzw. der aktuelle Offset als $\hat{\theta}(t)$ geschätzt werden. Der Synchronisationsfehler ergibt sich dann aus der Abweichung zwischen geschätzten und tatsächlichem Offset.

Für den virtuellen Takt ändert sich die Berechnung wie folgt:

$$Error_{\theta'} = |\theta' - \hat{\theta}| = |\theta' - [C'(t) - \hat{T}(t)]| = |\theta' - [C'(t) - \frac{C'(t) - \hat{\theta}(0)}{\hat{\gamma}}]| \tag{191}$$

$$Error_{\Gamma'} = |\Gamma' - \hat{\Gamma}| \tag{192}$$

Hierbei ist $C'(t) = \gamma' \cdot t + \theta'(0)$ die temperaturkompensierte (virtuelle) Version von $C(t)$, $\theta' = C'(t) - T(t)$ der Offset von $C'(t)$ und $\Gamma' = \gamma' - 1$ die Drift von $C'(t)$.

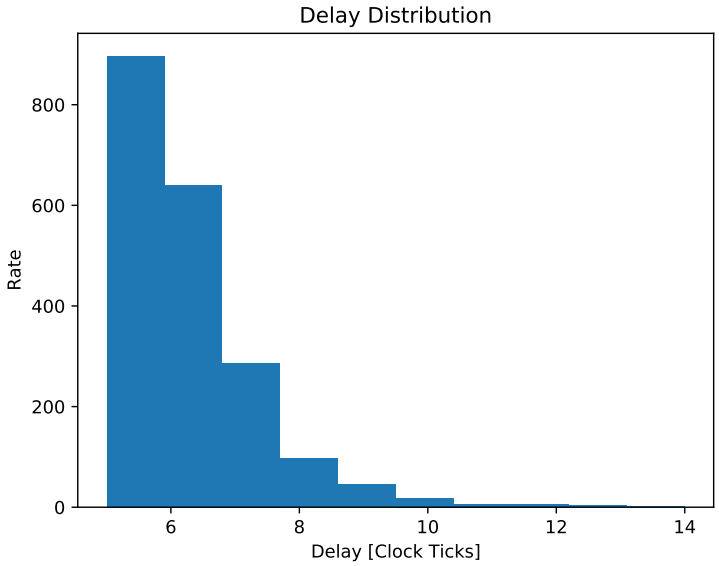


Abbildung 6.5: Histogramm der Verzögerung für eine Link-Auslastung von 90%. Die Verzögerung ist normiert auf eine Taktschrittweite (engl. Clock Tick) T_{tick} von 0,001s.

Tabelle 6.2: Übersicht der Ansätze für die Evaluation mittels Random-Walk-Taktmodell

Abkürzung	**Beschreibung**
PTP	Precision Time Protocol
PTP-EF	PTP mit zusätzlicher Exponentieller Filterung
PTP-H [B 5]	PTP mit LP-Heuristik
PTP-Kalman [A 26]	PTP mit zusätzlicher Kalman Filterung
PTP-LP [B 5]	PTP mit LP
SLMT	Nutzt LP und Multicasts
SLMT-H	Nutzt LP-Heuristik und Multicasts
Yang10 [A 34]	IMM zur Zeitsynchronisation

6.6.2 Evaluation mittels Random-Walk-Taktmodell

In diesem Abschnitt findet die Evaluation mit Hilfe des Random-Walk-Taktmodells statt. Aus Gründen der Übersichtlichkeit werden in diesem Abschnitt nur Diagramme für einen relativ stabilen Takt (HW-Clock, z. B. einen Hardware-Zeitzähler) gezeigt. Bei Verwendung eines weniger stabilen Taktes übertreffen die LP-basierten Ansätze weiterhin die SOTA-Ansätze. Allerdings sind die Ergebnisse von SLMT und PTP-LP hier weniger gut als bei Verwendung der HW-Clock, da die Ansätze die stärkeren Nichtlinearitäten schlechter ausgleichen können. Tabelle 6.2 zeigt eine Übersicht über alle Ansätze, die in diesem Abschnitt untersucht werden.

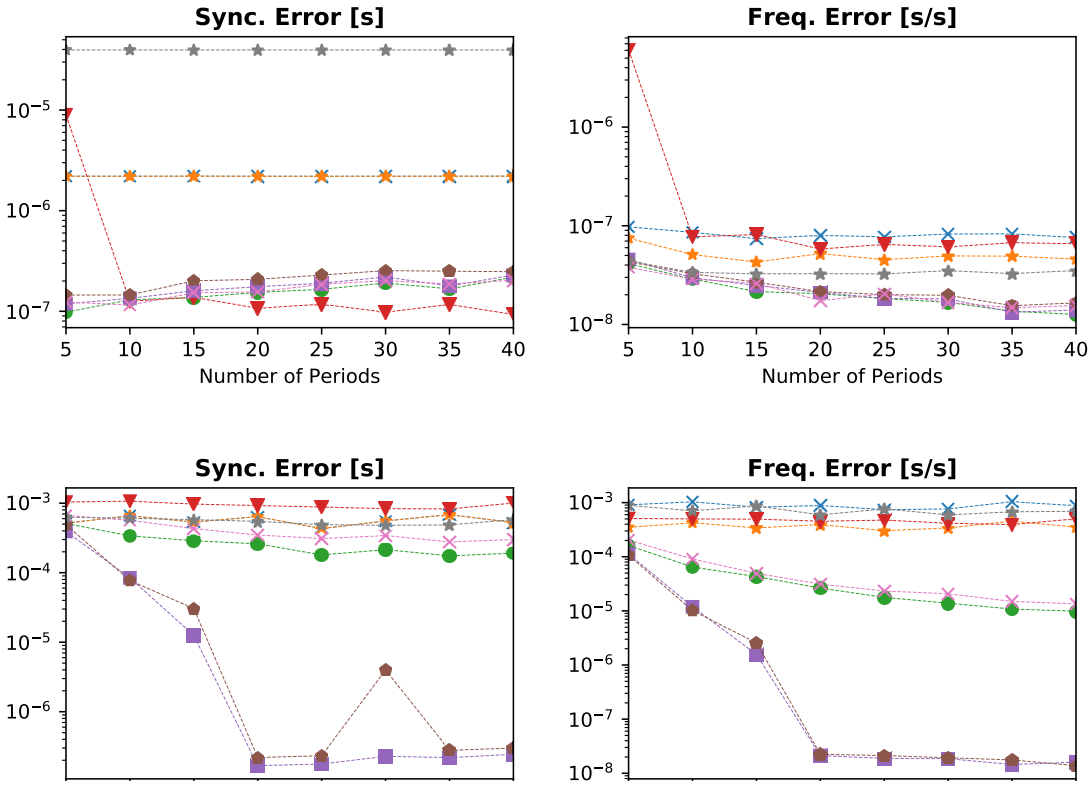


Abbildung 6.6: Ergebnisse für Random-Walk-Taktmodell. Die oberen Diagramme sind für eine mittlere Link-Auslastung von 10%, die unteren für 90%.

Bei einer mittleren Link-Auslastung von 10% gibt es keine wesentlichen Unterschiede zwischen den Ansätzen hinsichtlich des Frequenzfehlers (vgl. Abb. 6.6). In Bezug auf den Synchronisationsfehler schneiden die Ansätze PTP-Kalman, PTP-LP und SLMT viel besser ab als PTP und PTP-EF (PTP mit exponentieller Filterung) (vgl. Abb. 6.6). PTP und PTP-EF sind hier etwas schlechter, weil für die Simulation eine initiale Drift von 40 ppm angenommen wird. Dies führt zu leichten Ungenauigkeiten in den Offset-Schätzungen, da sich der Offset während der Übertragungs- und Verarbeitungszeit der Pakete bereits geringfügig ändert. Dies lässt sich ausgleichen, indem die Drift in Gl. 122 zusätzlich berücksichtigt wird. Wenn alternativ eine initiale Drift von 0 angenommen wird, so liegt die Genauigkeit von PTP und PTP-EF auch im Bereich von 10^{-7} s. Der Grund für die relativ geringen Unterschiede zwischen den Ansätzen ist, dass in diesem Setup relativ geringe Verzögerungsschwankungen auftreten und die Synchronisation damit vergleichsweise einfach ist.

Für eine mittlere Link-Auslastung von 90% erzielen die LP-basierten Ansätze SLMT und PTP-LP deutlich bessere Genauigkeiten als die übrigen Ansätze (vgl. Abb. 6.6)[6.3]. Der Grund hierfür ist, dass LP sehr robust gegenüber Verzögerungsschwankungen ist. PTP-LP erreicht eine etwas bessere Genauigkeit als SLMT, da es bidirektionale Unicasts und SLMT unidirektionale Multicasts für die Synchronisation verwendet. Folglich kann PTP-LP mehr Informationen verwenden und eine genauere Schätzung vornehmen. Die Multicasts von SLMT verursachen jedoch deutlich weniger Overhead. Es gibt einen Ausreißer für SLMT bei 30 Synchronisationspaketen. Hier tritt bei einem Durchlauf der seltene Fall auf, dass $C(t)$ zu nichtlinear ist, um mittels LP geschätzt zu werden. Dies ist jedoch kein wirklicher Nachteil von SLMT, da es in diesem Fall einfach weniger Pakete für die LP-Schätzung verwenden kann, vor allem, da die Präzision für mehr als 20 Pakete ansonsten sehr stabil ist (vgl. Diagramm unten links in Abb. 6.6).

6.6.3 Evaluation mittels Temperaturtaktmodell

Da LP-basierte Ansätze konzeptionelle Probleme mit nichtlinearen Takten haben, wird in dieser Forschungsarbeit eine Temperaturkompensation vorgeschlagen, welche in diesem Abschnitt mittels Temperaturtaktmodell ausgewertet wird (vgl. Abb. 6.7). Tabelle 6.3 zeigt eine Übersicht über alle Ansätze, die in diesem Abschnitt untersucht werden.

Bei einer mittleren Link-Auslastung von 10% ist die Genauigkeit von PTP-LP mit der von SLMT-NTC (SLMT ohne Temperaturkompensation) vergleichbar. Der Grund hierfür ist, dass es sich bei beiden um vergleichbare LP-basierte Ansätze ohne Temperaturkompensation handelt. SLMT-IMM (Temperaturkompensation mittels IMM) und SLMT-TH (Temperaturkompensation mittels TCTS) arbeiten jedoch viel besser, da hier der Temperatureinfluss durch Linearisieren des Taktes kompensiert wird. SLMT-IMM und SLMT-TH weisen eine in etwa vergleichbare Genauigkeit auf, SLMT-TH schneidet jedoch noch etwas besser ab. Der Grund hierfür ist die unterschiedliche Prozedur zur Driftmessung und die entsprechend unterschiedlichen Reaktionszeiten der Ansätze. SLMT-TH misst offline die Korrelation zwischen Drift und Temperatur. SLMT-TH kann daher die Drift direkt ausgleichen, wenn sich die Temperatur ändert. Im Gegensatz dazu misst das IMM die Drift online. Da für eine Driftmessung zwei Offsetmessungen erforderlich sind (vgl. Gl. 157), benötigt jeder Online-Ansatz eine Synchronisationsperiode, um die Drift zu messen und zu kompensieren. Um diesen Effekt zu verringern, kann die Synchronisationsperiode verkürzt werden. Zu beachten ist, dass hier nicht davon ausgegangen wird, dass die Driftmessungen von SLMT-TH perfekt sind. Stattdessen wurde die Messunsicherheit experimentell unter Verwendung einer selbstähnlichen Verzögerung und einer mittleren Link-Auslastung von 50% bestimmt. Der Grund für den leicht zunehmenden Synchronisationsfehler bei mehr als ca. 25 Perioden ist wie folgt. Wie in Abschnitt 6.6.1 beschrieben, wird der Temperaturverlauf skaliert mit steigender Anzahl von Synchronisationsperioden. Ein langsamerer und längerer Temperaturverlauf

[6.3] Im Diagramm oben links in Abb. 6.6 werden die Ergebnisse von PTP mit denen von PTP-EF überdeckt. Dies liegt aber nur daran, dass beide Verfahren die gleiche Offset-Berechnung verwenden. Lediglich die Drift wird bei PTP-EF gefiltert. Diese Design-Entscheidung wurde so getroffen, da der Offset aus Gründen der Vergleichbarkeit bei allen Ansätzen nicht kompensiert wird und sich über der Zeit ändert (vgl. Abb. 6.1). Dadurch ist es aus Sicht des Autos nicht zweckmäßig alte Offset-Informationen zu verwenden, da diese in der Tat "veraltet" sind.

Tabelle 6.3: Übersicht der Ansätze für die Evaluation mittels Temperaturtaktmodell

Abkürzung	**Beschreibung**
IMM-Mixed	IMM zur Temperaturkompensation
PTP	Precision Time Protocol
PTP-EF	PTP mit zusätzlicher Exponentieller Filterung
PTP-H [B 5]	PTP mit LP-Heuristik
PTP-Kalman [A 26]	PTP mit zusätzlicher Kalman Filterung
PTP-LP [B 5]	PTP mit LP
SLMT-IMM	IMM zur Temperaturkompensation, nutzt LP und Multicasts
SLMT-IMM-H	IMM zur Temperaturkompensation, nutzt LP-Heuristik und Multicasts
SLMT-NTC	keine Temperaturkompensation, nutzt LP und Multicasts
SLMT-NTC-H	keine Temperaturkompensation, nutzt LP-Heuristik und Multicasts
SLMT-TH	TCTS zur Temperaturkompensationn, nutzt LP und Multicasts
SLMT-TH-H	TCTS zur Temperaturkompensationn, nutzt LP-Heuristik und Multicasts
Yang10 [A 34]	IMM zur Zeitsynchronisation
EACS [A 15]	IMM zur Zeitsynchronisation, nutzt Temperaturinformationen

führt demzufolge zu einer längeren Zeit der Drift-Änderung und somit zu einer stärkeren Nichtlinearität der Slave-Zeit, welche wiederum schwerer zu kompensieren ist.

Bei Betrachtung des Frequenzfehlers sind SLMT-IMM und SLMT-TH weniger genau als andere Ansätze. Der Grund hierfür ist, dass die tatsächliche Drift aufgrund der Temperaturkompensation fast Null ist. Die LP-Ansätze versuchen jedoch die tatsächlich immer noch leicht nichtlineare Slave-Zeitfunktion zu schätzen, was per definitionem zu Fehlern führt. Je mehr Pakete verwendet werden, desto mehr folgt $C(t)$ einer linearen Funktion und die Genauigkeit steigt. Um diesen Fehler zu reduzieren, kann angenommen werden, dass die Drift aufgrund der Temperaturkompensation immer Null ist. Demzufolge muss mittels SLMT-IMM oder SLMT-TH nur der Offset bestimmt werden.

Für eine mittlere Link-Auslastung von 90% zeigt sich, dass SLMT-TH eine viel höhere Genauigkeit als alle anderen Ansätze erzielt. Der Grund dafür ist, dass SLMT-TH den Takt basierend auf den Offline-Messungen weiterhin linearisieren kann. Im Gegensatz dazu hat das IMM von SLMT-IMM große Probleme, da die Driftschätzungen aufgrund der hohen Verzögerungsschwankungen sehr ungenau sind. Dadurch wird der Takt eher verzerrt und nicht linearisiert.

6.6.4 Berechnungskomplexität

Abb. 6.8 zeigt die mittleren Berechnungszeit für verschiedene Ansätze. Für alle Ansätze gilt, dass die Berechnungszeit linear mit der Anzahl der verwendeten Zeitstempel bzw. Synchronisationsperioden steigt. Dies war auch so zu erwarten, da sowohl die lineare Regression

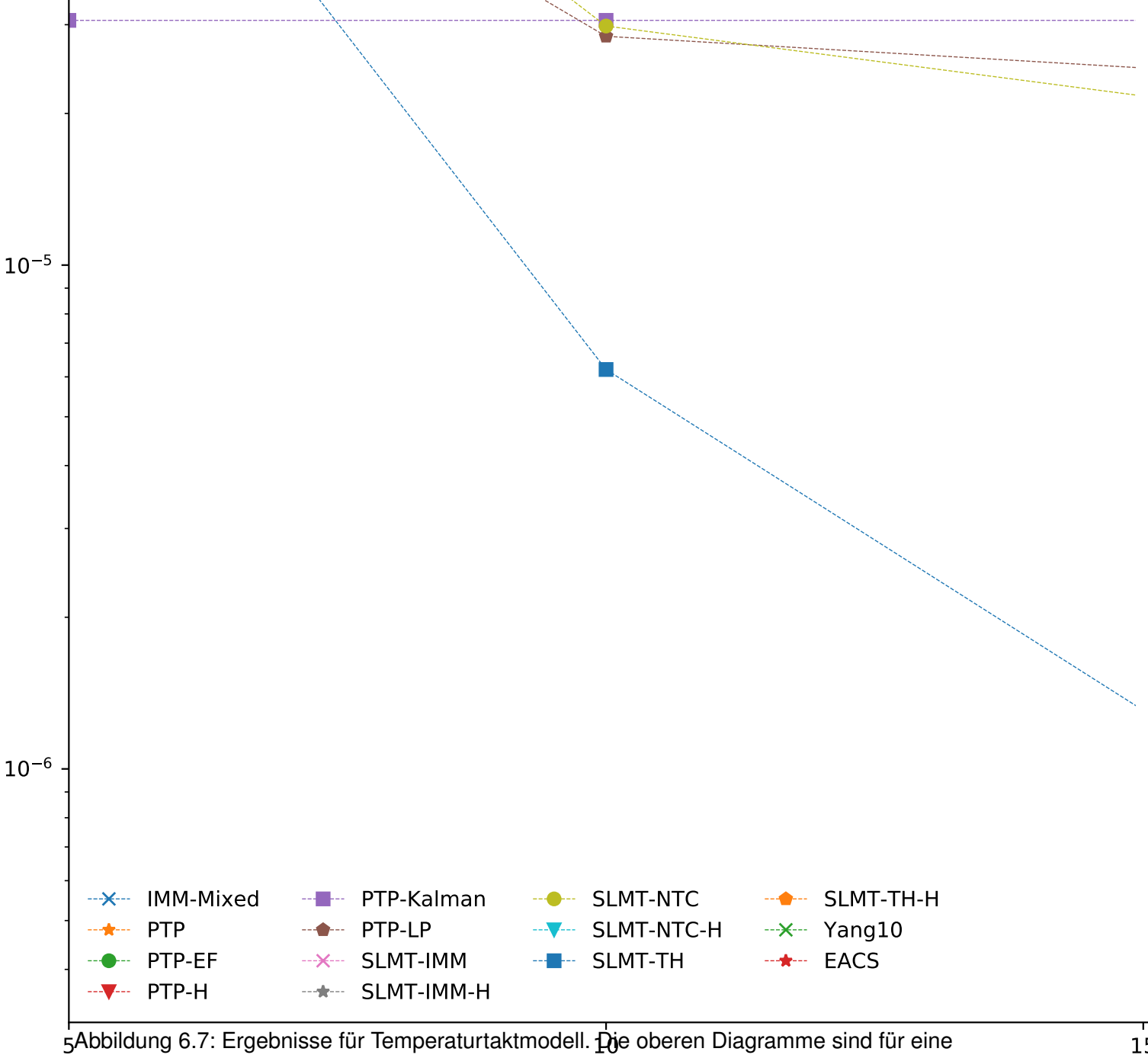


Abbildung 6.7: Ergebnisse für Temperaturtaktmodell. Die oberen Diagramme sind für eine mittlere Link-Auslastung von 10%, die unteren für 90%.

als auch LP eine polynomielle Komplexität haben. Eine wichtige Erkenntnis ist jedoch, dass die absolute Berechnungszeit für die LP-Heuristik deutlich geringer ist als für das LP. Es ist zu beachten, dass alle Berechnungszeiten sehr unkritisch sind, da es beispielsweise 45 Sekunden dauert, um 45 Pakete zu übertragen und die darin enthaltenen Zeitstempel zu sammeln. Diese Zeit ist also um mehrere Größenordnungen größer ist als die Berechnungszeit. Interessanterweise sind die temperaturkompensierten Ansätze schneller als jene ohne Temperaturkompensation, da ein linearisierter Takt zur einem einfacheren LP-Problem führt.

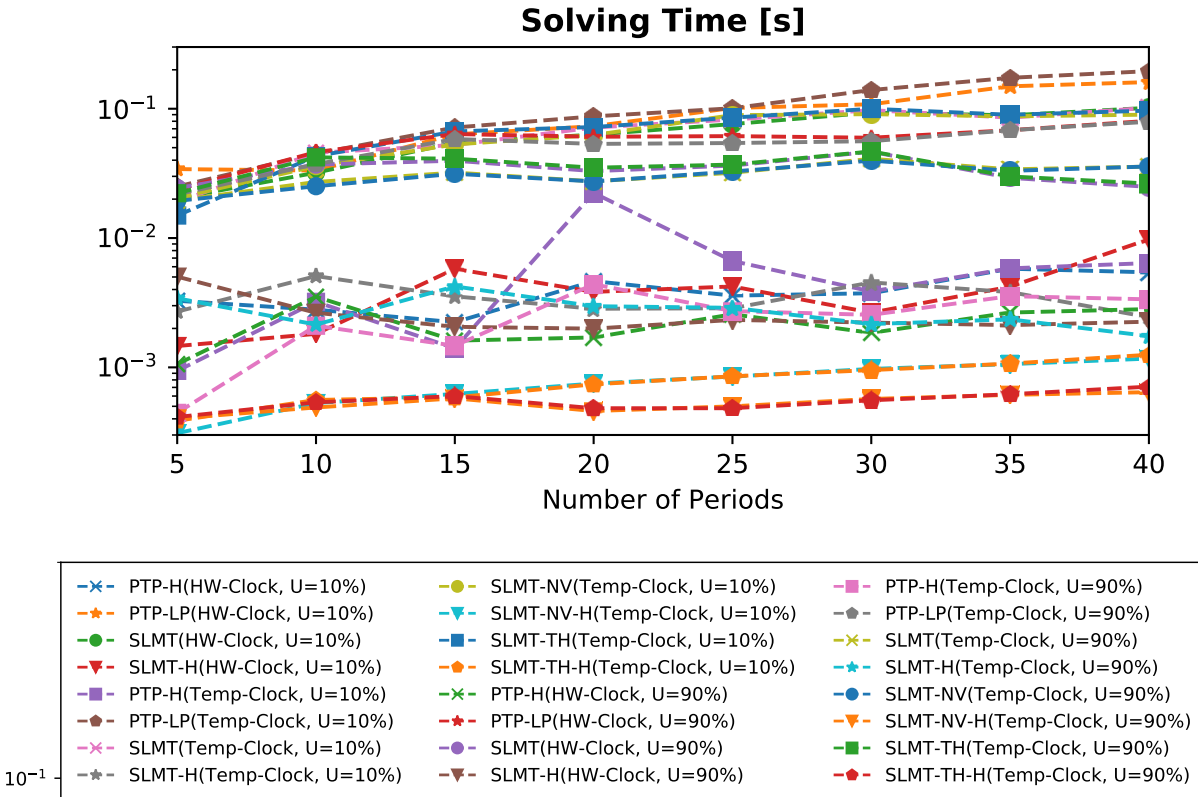


Abbildung 6.8: Berechnungszeit für verschiedene Ansätze.

6.7 Zwischenfazit

In diesem Kapitel wird der SLMT-Ansatz vorgestellt, der lineare Programmierung, Multicasts und Temperaturkompensation für die Zeitsynchronisation verwendet. Nach besten Wissen des Autors ist SLMT der erste Synchronisationsansatz, der lineare Programmierung (LP) mit Einwegvermittlung (bzw. Multicasts) und/oder Temperaturkompensation kombiniert.

In einer umfassenden Evaluation und einem Vergleich mit vielen SOTA-Ansätzen wird gezeigt, dass SLMT die SOTA-Ansätze meist übertrifft, insbesondere unter rauen Bedingungen, z.B. Temperaturänderungen und unbekannten nicht zu vernachlässigenden Netzwerkverzögerungen. Folgende Erkenntnisse können anhand der Evaluation gewonnen werden:

- Erstens ist die Kombination von Multicasts und LP sowohl möglich als auch vielversprechend, da sich der Overhead für die Synchronisation im Vergleich zur herkömmlichen Zwei-Wege-Synchronisation verringert und Multicasts nur zu einer geringfügigen Verringerung der Präzision führen.

- Zweitens ist auch die Kombination von LP und Temperaturkompensation sowohl möglich als auch vielversprechend.

- Hier lässt sich insbesondere festhalten, dass die einfache Temperaturheuristik SLMT-TH auch bei hoher Netzwerklast robust ist. Allerdings wird hier Vorwissen über die Korrelation zwischen Temperatur und Drift benötigt.
- Im Gegensatz dazu funktioniert die Verwendung eines komplexen IMM zur Temperaturkompensation ohne Vorkenntnisse gut bei geringer und mittlerer Netzwerklast. Unter rauen Netzwerkbedingungen nimmt die Genauigkeit jedoch ab, da Kalman-Filter die nicht-gaußschen Unsicherheiten nicht ausgleichen können, die zu Fehlern in der Driftmessung führen.

Auch von SLMT wurde ein Java-Simulationsmodell erstellt. Hierbei kam die, im Rahmen dieses Promotionsvorhabens entstandene, neueste Version der Java-Erweiterungen für den Netzwerksimulator OMNeT++ zum Einsatz [B 10, 7]. Auf diese Simulationsergebnisse wird aber nicht in der Evaluation eingegangen, da sich keine signifikanten Unterschiede zur numerischen Simulation ergaben, was in ähnlicher Weise bereits bei PTP-LP der Fall war.

SLMT wurde auch auf realen Geräten (BeagleBone Boards) untersucht. Die HW-Zeitstempel wurden in C++ ausgelesen und in Java weiterverarbeitet. Grundsätzlich ist die Implementierung auf realen Geräten also möglich. Eine Genauigkeit unterhalb von $100\ \mu s$ kann definitiv erreicht werden. Allerdings kann aktuell noch keine konkretere Aussage über die Synchronisationsgenauigkeit getroffen werden, da auf realen Geräten für die Bestimmung der Synchronisationsgenauigkeit immer eine exakte Referenzmessung nötig ist, um den tatsächlichen Offset bzw. Drift zu bestimmen. Eine Referenzmessung im Nanosekundenbereich fehlt aktuell noch auf den BeagleBone Boards. Dennoch wurde gezeigt, dass sich der Ansatz auch real implementieren lässt und eine vielversprechende Genauigkeit aufweist.

SLMT ist ein algorithmischer Ansatz. Eine Implementierung ist in Hard- oder Software möglich. Darüber hinaus könnte eine Hybridversion unter Verwendung der Hardware-Zeitstempel von PTP und der Lösung der LP-Probleme sowie der Temperaturkompensation in Software interessant sein. Weiterhin wird die Kombination von Temperaturkompensation und LP auch als wertvolle Erweiterung zu PTP bzw. PTP-LP [B 5] betrachtet.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung dieser Forschungsarbeit

In dieser Arbeit wurde zunächst eine umfangreiche Analyse des Standes der Technik vorgenommen. Hieraus konnten folgende Erkenntnisse gewonnen werden:

- Das Erreichen hoher Synchronisationsgenauigkeiten ($< 1\ \mu s$) ohne den Einsatz spezialisierter Hardware verbleibt als offene Forschungsfrage.
- Zur Reduktion der Kosten und Verbesserung der Zukunftssicherheit sollte ein Ansatz möglichst in Software umgesetzt werden. Die im Netzwerk entstandenen Verzögerungen könnten dann mittels leistungsstarker Schätzer auf den Endknoten kompensiert werden.
- Als Schätzer erzielt insbesondere die lineare Optimierung sehr gute Ergebnisse. Im Vergleich zu dieser weist die lineare Regression eine noch geringere Rechenkomplexität auf, während die Genauigkeit leicht beeinträchtigt wird.
- Jeweils ein auf EM (Expectation Maximization) und ein auf SVMs (Support Vector Machines) basierender Ansatz wurden bereits in der Literatur vorgestellt. Beide Schätzverfahren können auch allgemein für die drahtgebundene und drahtlose Synchronisation interessant sein.
- Einige Arbeiten befassen sich mit KH (konvexen Hüllen) in drahtgebundenen Netzwerken. Auch dies ist ein interessanter Ansatz, der weiter untersucht werden könnte.
- Obwohl es erste Ansätze für Sicherheit (Security) im Zeitsynchronisationsbereich gibt, bleiben sichere Zeitsynchronisationsprotokolle im Allgemeinen eine offene Forschungsfrage [7].

Ausgehend von diesen Erkenntnissen liegt der Fokus der vorliegenden Forschungsarbeit insbesondere auf der linearen Optimierung und der linearen Regression in drahtgebundenen Netzen. Es werden verschiedene Ansätze für die Zeitsynchronisation vorgestellt.

Der vorgestellte PSPI-Sync-Ansatz (Precise, Scalable, and Platform Independent Clock Synchronization) verwendet ein neues Schätzverfahren. Dieses verbindet Messungen der RTT (Round-Trip Time) mit einem Gleichungssystem, um durch Informationen von mehr als zwei Geräten eine höhere Genauigkeit zu erziehen. Des Weiteren wird der PTP-LP-Ansatz vorgestellt. PTP-LP bildet die erste Kombination von PTP und linearer Optimierung (engl. Linear Programming) sowie bis zur Veröffentlichung des SLMT-Ansatzes die umfangreichste Untersuchung der linearen Optimierung zur Zeitsynchronisation. Der SLMT-Ansatz (Synchronization Using Linear Programming, Multicasts, and Temperature Compensation) verbindet lineare Optimierung mit Multicasts und/oder Temperaturkompensation. Gleichzeitig werden zusammen mit SLMT und PTP-LP auch neue Verfahren in drahtgebundenen Netzwerken vorgestellt und untersucht, die auf linearer Regression basieren.

Es wird eine umfangreiche Evaluation aller vorgestellter Ansätze durchgeführt. Hierzu kommen einerseits numerische Simulationen und der Netzwerksimulator OMNeT++ zur Anwendung. Andererseits werden die Ansätze teilweise in realen Testumgebungen evaluiert.

Zusammenfassend konnten folgende Erkenntnisse gewonnen werden:

- Die lineare Optimierung eignet sich sehr gut für die Zeitsynchronisation und erweist sich als robust gegenüber starken Schwankungen der Verzögerung. Gleichzeitig ist die lineare Optimierung bzgl. der Berechnungskomplexität absolut unkritisch.
- Die in dieser Arbeit vorgestellten auf linearer Regression basierenden Ansätze sind weniger rechenintensiv und erzielen häufig eine vergleichbare Genauigkeit. Daher sind sie auch für drahtgebundene Szenarien geeignet, obwohl die lineare Regression in der Literatur bisher vor allem in drahtlosen Netzen untersucht wurde.
- Die Verbindung von Multicasts und linearer Optimierung ist möglich und vielversprechend. Hierbei entsteht weniger Overhead für die Synchronisation als bei der Zwei-Wege-Kommunikation. Gleichzeitig führen die Multicasts lediglich zu einer geringfügigen Verringerung der Synchronisationsgenauigkeit.
- Die Verbindung von linearer Optimierung und Temperaturkompensation ist möglich und lohnend. Einerseits wurden einfache Offline-Verfahren mit Vorwissen untersucht. Diese erwiesen sich als robust auch bei hoher Netzwerklast. Ein weiterer Vorteil ist, dass durch das Vorwissen eine schnellere Reaktion auf Temperaturänderungen ermöglicht wird. Andererseits wurden komplexere Online-Verfahren auf Basis von IMMs (Interacting-Multiple-Model-Kalman-Filter) untersucht. Diese erzielten gute Ergebnisse bei geringer und mittlerer Netzwerkauslastung. Bei hoher Netzwerklast kommt es jedoch zu großen Schätzfehlern. Da die Messfehler nicht gaußverteilt sind, kann das Kalman-Filter sie nicht ausgleichen. Als Folge nimmt die Genauigkeit drastisch ab.

Die Einzelmaßnahmen lineare Optimierung und lineare Regression liefern schon sehr gute und aufwandsarme Verbesserungen der Zeitsynchronisation, die untersuchte Kombination mit Multicasts oder Temperaturkompensation steigert die Genauigkeit weiter. Die Grenzen des Ansatzes werden herausgearbeitet.

7.2 Ausblick

In dieser Forschungsarbeit wurden verschiedene algorithmisch neue Ansätze untersucht. deren Implementierung ist also auf verschiedenen Plattformen in Hardware oder Software möglich. Auch die Entwicklung von Hybridversionen wäre interessant. Beispielsweise unter Verwendung von Hardware-Zeitstempeln und Weiterverarbeitung (lineare Optimierung, Temperaturkompensation) der Zeitstempel in Software.

Konzeptionell kommt die Kombination von Temperaturkompensation und linearer Optimierung auch als Erweiterung zu PTP bzw. PTP-LP in Betracht. Außerdem ließen sich die vorgestellten Ansätze auch für den Bereich drahtloser Netze anpassen und anwenden. Einerseits für den Nichtechtzeitbereich bzw. für weiche Echtzeit (Gaming, Video-Streams). Hier könnte sich die lineare Optimierung zur Kompensation von Verzögerungen in den Queues bzw. bei Weiterleitungsverzögerungen anbieten. So könnte eine hohe Genauigkeit bei vertretbaren Kosten erreicht werden. Andererseits könnten die vorgestellten Ansätze auch für die drahtlose Synchronisation im Echtzeitbereich (Smart Factory, Smart Power Plant) interessant sein, als Entlastung der drahtgebundenen Verbindungen vom Overhead der Syn-

chronisation. Zwar besteht, wie bereits erwähnt, in drahtlosen Echtzeitnetzen nach aktuellem Stand der Forschung weiterhin ein Zuverlässigkeitsproblem [A 7], dieses gilt aber nicht für die Synchronisation.

Der Verlust einzelner Pakete stellt für auf linearer Optimierung basierende Synchronisation kein Problem dar. Selbst, wenn über eine längere Zeit gar keine Pakete mehr ankommen, kann das System weiter funktionieren. Erstens wird das Wegdriften durch die Temperaturkompensation verlangsamt. Zweitens kann das Slave Gerät feststellen, dass keine Pakete mehr ankommen und selbst aktiv angemessen auf diese Situation reagieren. So kann das Gerät entweder in einen sichereren Fehlermodus umschalten oder weiterhin Zeitstempel aufnehmen und diese als von geringer Qualität kennzeichnen. In keinem Fall käme es jedoch zu einem Ausfall des Systems.

Des Weiteren wäre eine Untersuchung weiterer Schätzverfahren lohnenswert. Diese sollten vor allem robuster gegenüber Burst-Traffic sein, als es das Kalman-Filter ist. Solche Verfahren wurden z.B. bereits in [A 128] vorgeschlagen aber bisher noch nicht umfangreich genug evaluiert. Auch KI-basierte Ansätze (künstliche Intelligenz) wurden bisher für die Synchronisation wenig untersucht. Wie z.B. in [A 154] gezeigt, können diese durchaus in Netzwerkprotokollen zum Einsatz kommen. Als weitere konkrete Schätzverfahren konnten aus dem Stand der Technik EM, SVM und KH identifiziert werden.

A Literaturverzeichnis

[1] R. L. Scheiterer, C. Na, D. Obradovic und G. Steindl, „Synchronization Performance of the Precision Time Protocol in Industrial Automation Networks," *IEEE Transactions on Instrumentation and Measurement*, Jg. 58, Nr. 6, S. 1849–1857, 2009, ISSN: 0018-9456. DOI: 10.1109/TIM.2009.2013655.

[2] B. Chen, Y.-P. Chen, J.-M. Xie, Z.-D. Zhou und J.-M. Sa, „Control methodologies in networked motion control systems," in *Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005*, Guangzhou, China: IEEE Operations Center, 2005, 1088–1093 Vol. 2, ISBN: 0-7803-9091-1. DOI: 10.1109/ICMLC.2005.1527105.

[3] M. Felser, „Real-Time Ethernet - Industry Prospective," *Proceedings of the IEEE*, Jg. 93, Nr. 6, S. 1118–1129, 2005, ISSN: 0018-9219. DOI: 10.1109/JPROC.2005.849720.

[4] F. Steinhauser, C. Riesch und M. Rudigier, „IEEE 1588 for time synchronization of devices in the electric power industry," in *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS), 2010*, Portsmouth, NH, USA: IEEE, 2010, S. 1–6, ISBN: 978-1-4244-5978-0. DOI: 10.1109/ISPCS.2010.5609787.

[5] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum und A. Vahdat, „Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *USENIX NSDI 2018*.

[6] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowski, D. Timmermann und J. Schacht, „Survey on real-time communication via ethernet in industrial automation environments," in *2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Barcelona, Spain, S. 1–8. DOI: 10.1109/ETFA.2014.7005074.

[7] A. Mahmood, R. Exel, H. Trsek und T. Sauter, „Clock synchronization over IEEE 802.11—A survey of methodologies and protocols," *IEEE Trans. on Industrial Informatics*, Jg. 13, Nr. 2, 2017.

[8] M. Akhlaq und T. R. Sheltami, „RTSP: An Accurate and Energy-Efficient Protocol for Clock Synchronization in WSNs," *IEEE Transactions on Instrumentation and Measurement*, Jg. 62, Nr. 3, S. 578–589, 2013, ISSN: 0018-9456. DOI: 10.1109/TIM.2012.2232472.

[9] Y.-C. Wu, Q. Chaudhari und E. Serpedin, „Clock Synchronization of Wireless Sensor Networks," *IEEE Signal Processing Magazine*, Jg. 28, Nr. 1, S. 124–138, 2011, ISSN: 1053-5888. DOI: 10.1109/MSP.2010.938757.

[10] A. B. Noel, A. Abdaoui, T. Elfouly, M. H. Ahmed, A. Badawy und M. S. Shehata, „Structural Health Monitoring Using Wireless Sensor Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, Jg. 19, Nr. 3, S. 1403–1423, 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2691551.

[11] J. Schacht, H. Laqua und H. Niedermeyer, „Synchronization of Processes in a Distributed Real Time System Exemplified by the Control System of the Fusion Experiment WENDELSTEIN 7-X," *IEEE Transactions on Nuclear Science*, Jg. 53, Nr. 4, S. 2187–2194, 2006, ISSN: 0018-9499. DOI: 10.1109/TNS.2006.877930.

[12] R. C. Wolf, A. Ali, A. Alonso, J. Baldzuhn, C. Beidler, M. Beurskens, C. Biedermann, H.-S. Bosch, S. Bozhenkov, R. Brakel u. a., „Major results from the first plasma campaign of the Wendelstein 7-X stellarator," *Nuclear Fusion*, Jg. 57, Nr. 10, 2017.

[13] M. Lipinski, T. Wlostowski, J. Serrano und P. Alvarez, „White rabbit: a PTP application for robust sub-nanosecond synchronization," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Munich, Germany: IEEE, 12.09.2011 - 16.09.2011, S. 25–30, ISBN: 978-1-61284-893-8. DOI: 10.1109/ISPCS.2011.6070148.

[14] K. Akiyama, A. Alberdi, W. Alef, K. Asada, R. Azulay, A.-K. Baczko, D. Ball, M. Baloković, J. Barrett, D. Bintley u. a., „First M87 event horizon telescope results. IV. Imaging the central supermassive black hole," *The Astrophysical Journal Letters*, Jg. 875, Nr. 1, S. L4, 2019.

[15] Z. Yang, L. Cai, Y. Liu und J. Pan, „Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *IEEE INFOCOM 2012*. DOI: 10.1109/INFCOM.2012.6195457.

[16] A. S. Tanenbaum, D. J. Wetherall und K. Pieper, *Computernetzwerke*, 5., aktualisierte Aufl., Ser. it - Informatik. München: Pearson Higher Education, 2012, Bd. 4137, ISBN: 978-3-86894-137-1.

[17] F. Halsall, *Computer networking and the Internet*. Pearson Education India, 2006.

[18] L. L. Peterson und B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[19] Z. Li, W. Chen, C. Li, M. Li, X.-Y. Li und Y. Liu, „FLIGHT: clock calibration using fluorescent lighting," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, O. B. Akan, Hrsg., Istanbul, Turkey: ACM, 2013, S. 329, ISBN: 9781450311595. DOI: 10.1145/2348543.2348584.

[20] E. Mallada, X. Meng, M. Hack, L. Zhang und A. Tang, „Skewless Network Clock Synchronization Without Discontinuity: Convergence and Performance," *IEEE/ACM Transactions on Networking*, Jg. 23, Nr. 5, S. 1619–1633, 2015, ISSN: 1063-6692. DOI: 10.1109/TNET.2014.2345692.

[21] N. M. Freris, S. R. Graham und P. R. Kumar, „Fundamental Limits on Synchronizing Clocks Over Networks," *IEEE Transactions on Automatic Control*, Jg. 56, Nr. 6, S. 1352–1364, 2011, ISSN: 0018-9286. DOI: 10.1109/TAC.2010.2089210.

[22] K. S. Lee, H. Wang, V. Shrivastav und H. Weatherspoon, „Globally Synchronized Time via Datacenter Networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*, M. Barcellos, Hrsg., Florianopolis, Brazil: Association for Computing Machinery, Aug. 2016, S. 454–467, ISBN: 9781450341936. DOI: 10.1145/2934872.2934885.

[23] U. Tietze, C. Schenk und E. Gamm, *Halbleiter-Schaltungstechnik*, 16., erweiterte und aktualisierte Auflage. 2019, ISBN: 3662485532.

[24] D. W. Allan, „Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Jg. 34, Nr. 6, S. 647–654, 1987, ISSN: 0885-3010. DOI: 10.1109/t-uffc.1987.26997.

[25] W. J. Riley, *Handbook of frequency stability analysis*. US Department of Commerce, National Institute of Standards and Technology, 2008.

[26] G. Giorgi und C. Narduzzi, „Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks," *IEEE Trans. on Instrumentation and Measurement*, Jg. 60, Nr. 8, 2011. DOI: 10.1109/TIM.2011.2113120.

[27] R. Marchthaler und S. Dingler, *Kalman-Filter*. Springer, 2017.

[28] Y. Bar-Shalom, X. R. Li und T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[29] L. Papula, *Mathematik für Ingenieure und Naturwissenschaftler Band 3: Vektoranalysis, Wahrscheinlichkeitsrechnung, Mathematische Statistik, Fehler-und Ausgleichsrechnung*. Springer-Verlag, 2016.

[30] M. Natrella u. a., „NIST/SEMATECH e-handbook of statistical methods," Jg. 49, 2010.

[31] J. S. Hunter, „The exponentially weighted moving average," *Journal of quality technology*, Jg. 18, Nr. 4, S. 203–210, 1986.

[32] W. Hochstättler, *Lineare Optimierung*, Ser. Lehrbuch. Berlin: Springer Spektrum, 2017, ISBN: 978-3-662-54425-9. Adresse: http://www.springer.com/.

[33] X. R. Li und V. P. Jilkov, „Survey of maneuvering target tracking. part v: multiple-model methods," *IEEE Transactions on Aerospace and Electronic Systems*, Jg. 41, Nr. 4, S. 1255–1321, 2005, ISSN: 0018-9251. DOI: 10.1109/TAES.2005.1561886.

[34] Z. Yang, J. Pan und L. Cai, „Adaptive Clock Skew Estimation with Interactive Multi-Model Kalman Filters for Sensor Networks," in *IEEE ICC 2010*. DOI: 10.1109/ICC.2010.5502549.

[35] R. Labbe, *Kalman and bayesian filters in python*, 1.10.2020. Adresse: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.

[36] C. Zucca und P. Tavella, „The clock model and its relationship with the Allan and related variances," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Jg. 52, Nr. 2, S. 289–296, 2005, ISSN: 0885-3010. DOI: 10.1109/TUFFC.2005.1406554.

[37] S. Jafarpour und F. Bullo, „Synchronization of Kuramoto Oscillators via Cutset Projections," *IEEE Transactions on Automatic Control*, Jg. 64, Nr. 7, S. 2830–2844, 2019, ISSN: 0018-9286. DOI: 10.1109/TAC.2018.2876786.

[38] S.-Y. Ha, D. Ko, J. Park und X. Zhang, „Collective synchronization of classical and quantum oscillators," *EMS Surveys in Mathematical Sciences*, Jg. 3, Nr. 2, S. 209–267, 2016, ISSN: 2308-2151. DOI: 10.4171/EMSS/17.

[39] M. T. Schaub, N. O'Clery, Y. N. Billeh, J.-C. Delvenne, R. Lambiotte und M. Barahona, „Graph partitions and cluster synchronization in networks of oscillators," *Chaos (Woodbury, N.Y.)*, Jg. 26, Nr. 9, S. 094 821, 2016. DOI: 10.1063/1.4961065.

[40] F. Dörfler und F. Bullo, „Synchronization in complex networks of phase oscillators: A survey," *Automatica*, Jg. 50, Nr. 6, S. 1539–1564, 2014, ISSN: 00051098. DOI: 10.1016/j.automatica.2014.04.012.

[41] Y. Tang, F. Qian, H. Gao und J. Kurths, „Synchronization in complex networks and its application – A survey of recent advances and challenges," *Annual Reviews in Control*, Jg. 38, Nr. 2, S. 184–198, 2014, ISSN: 13675788. DOI: 10.1016/j.arcontrol.2014.09.003.

[42] F. Dörfler, M. Chertkov und F. Bullo, „Synchronization in complex oscillator networks and smart grids," *Proceedings of the National Academy of Sciences of the United States of America*, Jg. 110, Nr. 6, S. 2005–2010, 2013. DOI: 10.1073/pnas.1212134110.

[43] J. Gómez-Gardeñes, S. Gómez, A. Arenas und Y. Moreno, „Explosive synchronization transitions in scale-free networks," *Physical review letters*, Jg. 106, Nr. 12, S. 128 701, 2011. DOI: 10.1103/PhysRevLett.106.128701.

[44] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen und V. Kumar, „A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, Jg. 34, Nr. 4, S. 837–855, 2018, ISSN: 1552-3098. DOI: 10.1109/TRO.2018.2857475.

[45] J. Cao und R. Li, „Fixed-time synchronization of delayed memristor-based recurrent neural networks," *Science China Information Sciences*, Jg. 60, Nr. 3, 2017, ISSN: 1674-733X. DOI: 10.1007/s11432-016-0555-2.

[46] W. Zhang, X. Yang, C. Xu, J. Feng und C. Li, „Finite-Time Synchronization of Discontinuous Neural Networks With Delays and Mismatched Parameters," *IEEE transactions on neural networks and learning systems*, Jg. 29, Nr. 8, S. 3761–3771, 2018. DOI: 10.1109/TNNLS.2017.2740431.

[47] X. Liu, D. W. C. Ho, Q. Song und W. Xu, „Finite/Fixed-Time Pinning Synchronization of Complex Networks With Stochastic Disturbances," *IEEE transactions on cybernetics*, Jg. 49, Nr. 6, S. 2398–2403, 2019. DOI: 10.1109/TCYB.2018.2821119.

[48] B. Wei, F. Xiao und Y. Shi, „Synchronization in Kuramoto Oscillator Networks With Sampled-Data Updating Law," *IEEE transactions on cybernetics*, Jg. 50, Nr. 6, S. 2380–2388, 2020. DOI: 10.1109/TCYB.2019.2940987.

[49] I. Bojic und K. Nymoen, „Survey on synchronization mechanisms in machine-to-machine systems," *Engineering Applications of Artificial Intelligence*, Jg. 45, S. 361–375, 2015, ISSN: 09521976. DOI: 10.1016/j.engappai.2015.07.007.

[50] Y. Wang und F. J. Doyle, „Exponential synchronization rate of Kuramoto oscillators in the presence of a pacemaker," *IEEE Transactions on Automatic Control*, Jg. 58, Nr. 4, 2012, ISSN: 0018-9286. DOI: 10.1109/TAC.2012.2215772.

[51] Z. Zhou, M. S. Berger, S. R. Ruepp und Y. Yan, „Insight into the IEEE 802.1 Qcr asynchronous traffic shaping in time sensitive network," *Advances in Science, Technology and Engineering Systems Journal*, Jg. 4, Nr. 1, S. 292–301, 2019.

[52] Z. Zhou, Y. Yan, M. Berger und S. Ruepp, „Analysis and modeling of asynchronous traffic shaping in time sensitive networks," in *WFCS 2018*, Imperia, Italy: IEEE, 2018, S. 1–4, ISBN: 978-1-5386-1066-4. DOI: 10.1109/WFCS.2018.8402376.

[53] K. F. Hasan, C. Wang, Y. Feng und Y.-C. Tian, „Time synchronization in vehicular ad-hoc networks: A survey on theory and practice," *Vehicular Communications*, Jg. 14, S. 39–51, 2018, ISSN: 22142096. DOI: 10.1016/j.vehcom.2018.09.001.

[54] M. Levesque und D. Tipper, „A Survey of Clock Synchronization Over Packet-Switched Networks," *IEEE Communications Surveys & Tutorials*, Jg. 18, Nr. 4, S. 2926–2947, 2016, ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2590438.

[55] D. Praveen Kumar, T. Amgoth und C. S. R. Annavarapu, „Machine learning algorithms for wireless sensor networks: A survey," *Information Fusion*, Jg. 49, S. 1–25, 2019, ISSN: 15662535. DOI: 10.1016/j.inffus.2018.09.013.

[56] A. R. Swain und R. C. Hansdah, „A model for the classification and survey of clock synchronization protocols in WSNs," *Ad Hoc Networks*, Jg. 27, S. 219–241, 2015, ISSN: 15708705. DOI: 10.1016/j.adhoc.2014.11.021.

[57] S. M. Lasassmeh und J. M. Conrad, „Time synchronization in wireless sensor networks: A survey," in *Proceedings of the IEEE SoutheastCon 2010*, Concord, NC, USA: IEEE, 2010, S. 242–245, ISBN: 978-1-4244-5854-7. DOI: 10.1109/SECON.2010.5453878.

[58] P. Ranganathan und K. Nygard, „Time synchronization in wireless sensor networks: a survey," *International journal of ubicomp*, Jg. 1, Nr. 2, S. 92–102, 2010.

[59] B. Sundararaman, U. Buy und A. D. Kshemkalyani, „Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, Jg. 3, Nr. 3, S. 281–323, 2005, ISSN: 15708705. DOI: 10.1016/j.adhoc.2005.01.002.

[60] F. Sivrikaya und B. Yener, „Time synchronization in sensor networks: a survey," *IEEE Network*, Jg. 18, Nr. 4, S. 45–50, 2004, ISSN: 0890-8044. DOI: 10.1109/MNET.2004.1316761.

[61] A. Bletsas, „Evaluation of Kalman filtering for network time keeping," *IEEE Trans. on Ultrasonics, Ferroelectrics and Frequency Control*, Jg. 52, Nr. 9, 2005. DOI: 10.1109/TUFFC.2005.1516016.

[62] D. Mills, J. Martin, J. Burbank und W. Kasch, „Network time protocol version 4: Protocol and algorithms specification," 2010.

[63] *IEEE 1588-2008 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Piscataway, NJ, USA, 2008. DOI: 10.1109/IEEESTD.2008.4579760.

[64] *IEEE 802.1AS-2011 Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, Piscataway, NJ, USA, 2011. DOI: 10.1109/IEEESTD.2011.5741898.

[65] *IEEE 1588-2002 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Piscataway, NJ, USA, 2002. DOI: 10.1109/IEEESTD.2002.94144.

[66] *IEEE 1588-2019 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Piscataway, NJ, USA, 2019. DOI: 10.1109/IEEESTD.2020.9120376.

[67] *IEEE 802.1AS-2020 Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications*, Piscataway, NJ, USA, 2020. DOI: 10.1109/IEEESTD.2020.9121845.

[68] International Electrotechnical Commission, Hrsg., *IEC 61784-2, Digital data communications for measurement and control-Part 2: Additional profiles for ISO/IEC 8802-3 based communication networks in real-time applications*, 2019.

[69] R. Pigan und M. Metter, *Automating with PROFINET: Industrial Communication Based on Industrial Ethernet*. Somerset: Publicis MCD Werbeagentur GmbH, 2015, ISBN: 9783895782947. Adresse: https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4691836.

[70] D. Fontanelli, D. Macii, S. Rinaldi, P. Ferrari und A. Flammini, „Performance analysis of a clock state estimator for PROFINET IO IRT synchronization," in *2013 IEEE International Instrumentation and Measurement Technology Conference*, I. Staff, Hrsg.,

Minneapolis, MN, USA: IEEE, 2013, S. 1828–1833, ISBN: 978-1-4673-4623-8. DOI: 10.1109/I2MTC.2013.6555730.

[71] International Telecommunication Union, *ITU-I G.8262: Timing characteristics of a synchronous equipment slave clock*, 2018.

[72] K. Hann, S. Jobert und S. Rodrigues, „Synchronous ethernet to transport frequency and phase/time," *IEEE Communications Magazine*, Jg. 50, Nr. 8, S. 152–160, 2012, ISSN: 0163-6804. DOI: 10.1109/MCOM.2012.6257542.

[73] S. T. Watt, S. Achanta, H. Abubakari, E. Sagen, Z. Korkmaz und H. Ahmed, „Understanding and applying precision time protocol," in *2015 Saudi Arabia smart grid (SASG)*, Jeddah, Saudi Arabia: IEEE, 2015, S. 1–7, ISBN: 978-1-4673-9454-3. DOI: 10.1109/SASG.2015.7449285.

[74] K. Correll, N. Barendt und M. Branicky, „Design considerations for software only implementations of the IEEE 1588 precision time protocol," *Conference on IEEE 1588*, Jg. 2005, S. 11–15,

[75] M. D. Johas Teener und G. M. Garner, „Overview and timing performance of IEEE 802.1AS," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, Ann Arbor, MI, S. 49–53. DOI: 10.1109/ISPCS.2008.4659212.

[76] G. Garner und H. Ryu, „Synchronization of audio/video bridging networks using IEEE 802.1AS," *IEEE Communications Magazine*, Jg. 49, Nr. 2, S. 140–147, 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5706322.

[77] M. Gutierrez, W. Steiner, R. Dobrin und S. Punnekkat, „Synchronization Quality of IEEE 802.1AS in Large-Scale Industrial Automation Networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Pittsburg, PA, USA, S. 273–282. DOI: 10.1109/RTAS.2017.10.

[78] T. Steinbach, H.-T. Lim, F. Korf, T. C. Schmidt, D. Herrscher und A. Wolisz, „Tomorrow's In-Car Interconnect? A Competitive Evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802)," in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, Quebec City, QC, Canada, S. 1–5. DOI: 10.1109/VTCFall.2012.6398932.

[79] R. Exel, „Mitigation of Asymmetric Link Delays in IEEE 1588 Clock Synchronization Systems," *IEEE Communications Letters*, Jg. 18, Nr. 3, S. 507–510, 2014, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2014.012214.132540.

[80] A. Mahmood und R. Exel, „Servo design for improved performance in software timestamping-assisted WLAN synchronization using IEEE 1588," in *IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), 2013*, Cagliari, Italy: IEEE, 2013, S. 1–8, ISBN: 978-1-4799-0864-6. DOI: 10.1109/ETFA.2013.6647964.

[81] H. Abubakari und S. Sastry, „IEEE 1588 style synchronization over wireless link," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2008, S. 127–130. DOI: 10.1109/ISPCS.2008.4659226.

[82] A. Diarra, T. Hogenmueller, A. Zimmermann, A. Grzemba und U. A. Khan, „Improved clock synchronization start-up time for Ethernet AVB-based in-vehicle networks," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, Luxembourg, Luxembourg, S. 1–8. DOI: 10.1109/ETFA.2015.7301412.

[83] J. Serrano, M. Lipinski, T. Wlostowski, E. Gousiou, E. van der Bij, M. Cattin und G. Daniluk, „The white rabbit project," *2nd International Beam Instrumentation Conference*, 2013.

[84] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt und G. Gaderer, „White rabbit: Sub-nanosecond timing distribution over ethernet," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS 2009)*, Brescia, Italy, S. 1–5. DOI: 10.1109/ISPCS.2009.5340196.

[85] A. K. Karthik und R. S. Blum, „Robust Clock Skew and Offset Estimation for IEEE 1588 in the Presence of Unexpected Deterministic Path Delay Asymmetries," *IEEE Transactions on Communications*, S. 1, 2020, ISSN: 0090-6778. DOI: 10.1109/TCOMM.2020.2991212.

[86] N. Kero, A. Puhm, T. Kernen und A. Mroczkowski, „Performance and Reliability Aspects of Clock Synchronization Techniques for Industrial Automation," *Proceedings of the IEEE*, Jg. 107, Nr. 6, S. 1011–1026, 2019, ISSN: 0018-9219. DOI: 10.1109/JPROC.2019.2915972.

[87] D. Fontanelli und D. Macii, „Accurate time synchronization in PTP-based industrial networks with long linear paths," in *2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, Portsmouth, NH, USA, S. 97–102. DOI: 10.1109/ISPCS.2010.5609785.

[88] T. Mizrahi und Y. Moses, „ReversePTP: A clock synchronization scheme for software-defined networks," *International Journal of Network Management*, Jg. 26, Nr. 5, S. 355–372, 2016, ISSN: 10557148. DOI: 10.1002/nem.1942.

[89] T. Mizrahi und Y. Moses, „Using ReversePTP to distribute time in Software Defined Networks," in *2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, Austin, TX, USA: IEEE, 2014, S. 112–117, ISBN: 978-1-4799-2699-2. DOI: 10.1109/ISPCS.2014.6948702.

[90] T. Mizrahi und Y. Moses, „ReversePTP: a software defined networking approach to clock synchronization," in *HotSDN '14: Proceedings of the third workshop on Hot*

topics in software defined networking, A. Akella und A. Greenberg, Hrsg., Chicago, Illinois, USA, 2014, S. 203–204. DOI: 10.1145/2620728.2620764.

[91] T. Benson, A. Akella und D. A. Maltz, „Network Traffic Characteristics of Data Centers in the Wild," in *ACM SIGCOMM 2010*. DOI: 10.1145/1879141.1879175.

[92] T. Schmid, P. Dutta und M. B. Srivastava, „High-resolution, low-power time synchronization an oxymoron no more," in *ACM/IEEE International Conference on Information Processing in Sensor Networks 2010*.

[93] *IEEE 802.1Q-2011 Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*, Piscataway, NJ, USA, 2011. DOI: 10.1109/IEEESTD.2011.6009146.

[94] J. Schacht und J. Skodzik, „Multifunction-Timing Card ITTEV2 for CoDaC Systems of Wendelstein 7-X," *IEEE Transactions on Nuclear Science*, Jg. 62, Nr. 3, S. 1187–1194, 2015, ISSN: 0018-9499. DOI: 10.1109/TNS.2015.2425895.

[95] Q. Wu, L. Yang und J. Chen, „Enhancement for Real-Time Ethernet Clock Synchronization by Internal Processing Delay Measurement," *IEEE Communications Letters*, Jg. 23, Nr. 11, S. 2063–2067, 2019, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2019.2937520.

[96] T. Ahmed, S. Rahman, M. Tornatore, K. Kim und B. Mukherjee, „A survey on high-precision time synchronization techniques for optical datacenter networks and a zero-overhead microsecond-accuracy solution," *Photonic Network Communications*, Jg. 36, Nr. 1, S. 56–67, 2018, ISSN: 1387-974X. DOI: 10.1007/s11107-018-0773-9.

[97] L. Zhang, Z. Liu und C. Honghui Xia, „Clock synchronization algorithms for network measurements," in *IEEE INFOCOM 2002*, P. Kermani, Hrsg., New York, NY, USA: IEEE, 2002, S. 160–169, ISBN: 0-7803-7476-2. DOI: 10.1109/INFCOM.2002.1019257.

[98] D. Veitch, J. Ridoux und S. B. Korada, „Robust Synchronization of Absolute and Difference Clocks Over Networks," *IEEE/ACM Transactions on Networking*, Jg. 17, Nr. 2, S. 417–430, 2009, ISSN: 1063-6692. DOI: 10.1109/TNET.2008.926505.

[99] M. Davis, B. Villain, J. Ridoux, A.-C. Orgerie und D. Veitch, „An IEEE-1588 compatible RADclock," in *2012 International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, San Francisco, CA, USA: IEEE, 2012, S. 1–6, ISBN: 978-1-4577-1716-1. DOI: 10.1109/ISPCS.2012.6336624.

[100] X. Xu, Z. Xiong, X. Sheng, J. Wu und X. Zhu, „A New Time Synchronization Method for Reducing Quantization Error Accumulation Over Real-Time Networks: Theory and Experiments," *IEEE Transactions on Industrial Informatics*, Jg. 9, Nr. 3, S. 1659–1669, 2013, ISSN: 1551-3203. DOI: 10.1109/TII.2013.2238547.

[101] L. F. Auler und R. d'Amore, „Adaptive Kalman Filter for Time Synchronization over Packet-Switched Networks: An Heuristic Approach," in *2nd International Conference on Communication Systems Software and Middleware, 2007*, Bangalore, India: IEEE Service Center, 2007, S. 1–7, ISBN: 1-4244-0613-7. DOI: 10.1109/COMSWA.2007.382439.

[102] H. Khlifi und J.-C. Gregoire, „Estimation and removal of clock skew from delay measures," in *LCN 2004*, Tampa, FL, USA: IEEE Computer Society, 2004, S. 144–151, ISBN: 0-7695-2260-2. DOI: 10.1109/LCN.2004.54.

[103] S. B. Moon, P. Skelly und D. Towsley, „Estimation and removal of clock skew from network delay measurements," *IEEE INFOCOM 1999*, 227–234 vol.1, DOI: 10.1109/INFCOM.1999.749287.

[104] G. Giorgi, „An Event-Based Kalman Filter for Clock Synchronization," *IEEE Transactions on Instrumentation and Measurement*, Jg. 64, Nr. 2, S. 449–457, 2015, ISSN: 0018-9456. DOI: 10.1109/TIM.2014.2340631.

[105] G. Giorgi und C. Narduzzi, „Kalman filtering for multi-path network synchronization," in *2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, Austin, TX, USA: IEEE, 2014, S. 65–70, ISBN: 978-1-4799-2699-2. DOI: 10.1109/ISPCS.2014.6948693.

[106] G. Giorgi und C. Narduzzi, „A resilient Kalman filter based servo clock," in *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings*, Lemgo, Germany: IEEE, 22.09.2013 - 27.09.2013, S. 59–64, ISBN: 978-1-4799-0242-2. DOI: 10.1109/ISPCS.2013.6644764.

[107] M. D. Lemmon, J. Ganguly und L. Xia, „Model-based clock synchronization in networks with drifting clocks," in *Dependable Computing, 2000. Proceedings. 2000 Pacific Rim International Symposium on*, 2000, S. 177–184.

[108] J. Skodzik, P. Danielis, V. Altmann und D. Timmermann, „Time synchronization in the DHT-based P2P network Kad for real-time automation scenarios," in *2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Madrid, S. 1–6. DOI: 10.1109/WoWMoM.2013.6583490.

[109] J. Skodzik, V. Altmann, P. Danielis, A. Wall und D. Timmermann, „A kad prototype for time synchronization in real-time automation scenarios," in *WTC 2014; World Telecommunications Congress 2014*, 2014, S. 1–6.

[110] S. Froehlich, M. Hack, X. Meng und L. Zhang, „Achieving precise coordinated cluster time in a cluster environment," in *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2008*, Ann Arbor,

MI: IEEE, 2008, S. 54–58, ISBN: 978-1-4244-2274-6. DOI: 10.1109/ISPCS.2008.4659213.

[111] T. Schmid, Z. Charbiwala, R. Shea und M. B. Srivastava, „Temperature Compensated Time Synchronization," *IEEE Embedded Systems Letters*, Jg. 1, Nr. 2, S. 37–41, 2009, ISSN: 1943-0663. DOI: 10.1109/LES.2009.2028103.

[112] K. Skiadopoulos, A. Tsipis, K. Giannakis, G. Koufoudakis, E. Christopoulou, K. Oikonomou, G. Kormentzas und I. Stavrakakis, „Synchronization of data measurements in wireless sensor networks for IoT applications," *Ad Hoc Networks*, Jg. 89, S. 47–57, 2019, ISSN: 15708705. DOI: 10.1016/j.adhoc.2019.03.002.

[113] Y. Kikuya, S. M. Dibaji und H. Ishii, „Fault-Tolerant Clock Synchronization Over Unreliable Channels in Wireless Sensor Networks," *IEEE Transactions on Control of Network Systems*, Jg. 5, Nr. 4, S. 1551–1562, 2018. DOI: 10.1109/TCNS.2017.2732169.

[114] K. Xie, Q. Cai und M. Fu, „A fast clock synchronization algorithm for wireless sensor networks," *Automatica*, Jg. 92, S. 133–142, 2018, ISSN: 00051098. DOI: 10.1016/j.automatica.2018.03.004.

[115] A. Dongare, P. Lazik, N. Rajagopal und A. Rowe, „Pulsar: A Wireless Propagation-Aware Clock Synchronization Platform," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Pittsburg, PA, USA: IEEE, 18.04.2017 - 21.04.2017, S. 283–292, ISBN: 978-1-5090-5269-1. DOI: 10.1109/RTAS.2017.9.

[116] C. Benzaïd, M. Bagaa und M. Younis, „Efficient clock synchronization for clustered wireless sensor networks," *Ad Hoc Networks*, Jg. 56, S. 13–27, 2017, ISSN: 15708705. DOI: 10.1016/j.adhoc.2016.11.003.

[117] F. Gong und M. L. Sichitiu, „Temperature compensated Kalman distributed clock synchronization," *Ad Hoc Networks*, Jg. 62, S. 88–100, 2017, ISSN: 15708705. DOI: 10.1016/j.adhoc.2017.04.009.

[118] M. Koivisto, M. Costa, J. Werner, K. Heiska, J. Talvitie, K. Leppänen, V. Koivunen und M. Valkama, „Joint device positioning and clock synchronization in 5G ultra-dense networks," *IEEE Trans. on Wireless Communications*, Jg. 16, Nr. 5, 2017.

[119] G. Cena, S. Scanzio, A. Valenzano und C. Zunino, „Implementation and Evaluation of the Reference Broadcast Infrastructure Synchronization Protocol," *IEEE Transactions on Industrial Informatics*, Jg. 11, Nr. 3, S. 801–811, 2015, ISSN: 1551-3203. DOI: 10.1109/TII.2015.2396003.

[120] C. Lenzen, P. Sommer und R. Wattenhofer, „PulseSync: An Efficient and Scalable Clock Synchronization Protocol," *IEEE/ACM Transactions on Networking*, Jg. 23, Nr. 3, S. 717–727, 2015, ISSN: 1063-6692. DOI: 10.1109/TNET.2014.2309805.

[121] C. Lenzen, P. Sommer und R. Wattenhofer, „Optimal clock synchronization in networks," *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, S. 225, 2009. DOI: 10.1145/1644038.1644061.

[122] J. Elson, L. Girod und D. Estrin, „Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, Jg. 36, Nr. SI, S. 147–163, 2002, ISSN: 0163-5980. DOI: 10.1145/844128.844143.

[123] M. Marti, B. Kusy, G. Simon und k. Ldeczi, „The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, J. A. Stankovic, Hrsg., Baltimore, MD, USA: ACM, 2004, S. 39, ISBN: 1581138792. DOI: 10.1145/1031495.1031501.

[124] S. Ganeriwal, R. Kumar und M. B. Srivastava, „Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, I. Akyildiz, Hrsg., Los Angeles, California, USA: ACM, 2003, S. 138, ISBN: 1581137079. DOI: 10.1145/958491.958508.

[125] P. Sommer und R. Wattenhofer, „Gradient clock synchronization in wireless sensor networks," in *2009 International Conference on Information Processing in Sensor Networks*, 2009, S. 37–48.

[126] T. Hao, R. Zhou, G. Xing, M. W. Mutka und J. Chen, „WizSync: Exploiting Wi-Fi Infrastructure for Clock Synchronization in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, Jg. 13, Nr. 6, S. 1379–1392, 2014, ISSN: 1536-1233. DOI: 10.1109/TMC.2013.43.

[127] T. Chang, T. Watteyne, K. Pister und Q. Wang, „Adaptive synchronization in multi-hop TSCH networks," *Computer Networks*, Jg. 76, S. 165–176, 2015, ISSN: 13891286. DOI: 10.1016/j.comnet.2014.11.003.

[128] J. O. Nilsson und P. Händel, „Robust recursive network clock synchronization," in *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2014, S. 1–5. DOI: 10.1109/CONECCT.2014.6740335.

[129] T. Qiu, Y. Zhang, D. Qiao, X. Zhang, M. L. Wymore und A. K. Sangaiah, „A Robust Time Synchronization Scheme for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, S. 1, 2017, ISSN: 1551-3203. DOI: 10.1109/TII.2017.2738842.

[130] T. Chang und Q. Wang, „Adaptive Compensation for Time-Slotted Synchronization in Wireless Sensor Network," *International Journal of Distributed Sensor Networks*, Jg. 10, Nr. 4, S. 540 397, 2014, ISSN: 1550-1477. DOI: 10.1155/2014/540397.

[131] T. Qiu, L. Chi, W. Guo und Y. Zhang, „STETS: A novel energy-efficient time synchronization scheme based on embedded networking devices," *Microprocessors and Mi-*

crosystems, Jg. 39, Nr. 8, S. 1285–1295, 2015, ISSN: 01419331. DOI: 10.1016/j.micpro.2015.07.006.

[132] R. Exel und F. Ring, „Improved clock synchronization accuracy through optimized servo parametrization," in *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings*, Lemgo, Germany: IEEE, 22.09.2013 - 27.09.2013, S. 65–70, ISBN: 978-1-4799-0242-2. DOI: 10.1109/ISPCS.2013.6644765.

[133] D. Fontanelli, D. Macii, P. Wolfrum, D. Obradovic und G. Steindl, „A clock state estimator for PTP time synchronization in harsh environmental conditions," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Munich, Germany: IEEE, 12.09.2011 - 16.09.2011, S. 99–104, ISBN: 978-1-61284-893-8. DOI: 10.1109/ISPCS.2011.6070142.

[134] M. Jin, T. Xing, X. Chen, X. Meng, D. Fang und Y. He, „DualSync: Taming clock skew variation for synchronization in low-power wireless networks," in *IEEE INFOCOM 2016*. DOI: 10.1109/INFOCOM.2016.7524335.

[135] Z. Yang, L. He, L. Cai und J. Pan, „Temperature-Assisted Clock Synchronization and Self-Calibration for Sensor Networks," *IEEE Trans. on Wireless Communications*, Jg. 13, Nr. 6, 2014. DOI: 10.1109/TWC.2014.051414.130270.

[136] W. Lasoi und S. Pornpromlikit, „Temperature-aware Time Synchronization with an Accuracy-efficiency Trade-off in Wireless Sensor Networks," *Procedia Engineering*, Jg. 168, S. 1706–1709, 2016, ISSN: 18777058. DOI: 10.1016/j.proeng.2016.11.495.

[137] A. Beifus, D. Raumer, P. Emmerich, T. M. Runge, F. Wohlfart, B. E. Wolfinger und G. Carle, „A study of networking software induced latency," in *2015 IEEE International Conference and Workshops on Networked Systems (NetSys)*, Cottbus, Germany, S. 1–8. DOI: 10.1109/NetSys.2015.7089065.

[138] M. Felser, „Real Time Ethernet: Standardization and implementations," in *2010 IEEE International Symposium on Industrial Electronics (ISIE 2010)*, Bari, Italy, S. 3766–3771. DOI: 10.1109/ISIE.2010.5637988.

[139] Y. Bejerano, Y. Breitbart, M. Garofalakis und R. Rastogi, „Physical topology discovery for large multisubnet networks," in *IEEE INFOCOM 2003*, F. Bauer, Hrsg., San Francisco, CA, USA: IEEE, 2003, S. 342–352, ISBN: 0-7803-7752-4. DOI: 10.1109/INFCOM.2003.1208686.

[140] S. Bradner und J. McQuaid, „Benchmarking methodology for network interconnect devices," *RFC2544*, 1999.

[141] P. Emmerich, D. Raumer, F. Wohlfart und G. Carle, „A study of network stack latency for game servers," in *2014 13th Annual Workshop on Network and Systems Support*

for Games (NetGames), Nagoya, Japan, S. 1–6. DOI: 10.1109/NetGames.2014.7008960.

[142] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood und A. W. Moore, „OFLOPS: An Open Framework for OpenFlow Switch Evaluation," *International Conference on Passive and Active Network Measurement*, S. 85–95, 2012.

[143] K. Inoue, D. Pasetto, K. Lynch, M. Meneghin, K. Muller und J. Sheehan, „Low-latency and high bandwidth TCP/IP protocol processing through an integrated HW/SW approach," in *IEEE INFOCOM 2013 - IEEE Conference on Computer Communications*, Turin, Italy, S. 2967–2975. DOI: 10.1109/INFCOM.2013.6567108.

[144] S. Larsen, P. Sarangam, R. Huggahalli und S. Kulkarni, „Architectural Breakdown of End-to-End Latency in a TCP/IP Network," *International Journal of Parallel Programming*, Jg. 37, Nr. 6, S. 556–571, 2009, ISSN: 0885-7458. DOI: 10.1007/s10766-009-0109-6.

[145] L. M. Märdian, „What's New in the Linux Network Stack?" *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, Jg. 2, S. 6, 2014.

[146] L. Fahrmeir, C. Heumann, R. Künstler, I. Pigeot und G. Tutz, *Statistik: Der Weg zur Datenanalyse*, 8., überarbeitete und ergänzte Auflage, Ser. Springer-Lehrbuch. Berlin und Heidelberg: Springer Spektrum, 2016, ISBN: 978-3-662-50371-3. DOI: 10.1007/978-3-662-50372-0.

[147] B. Konieczek, M. Rethfeldt, F. Golatowski und D. Timmermann, „Real-Time Communication for the Internet of Things Using jCoAP," in *2015 IEEE 18th International Symposium on Real-Time Distributed Computing (ISORC)*, Auckland, New Zealand, S. 134–141. DOI: 10.1109/ISORC.2015.35.

[148] *Jamaica Virtual Machine*, 4.03.2021. Adresse: https://www.aicas.com/wp/products-services/jamaicavm/.

[149] *Intel Galileo Board*, 4.03.2021. Adresse: https://www.arduino.cc/en/ArduinoCertified/IntelGalileo.

[150] B. B. Mandelbrot, *The fractal geometry of nature*. New York, NY: Freeman, 1983, ISBN: 978-0716711865.

[151] W. E. Leland, M. S. Taqqu, W. Willinger und D. V. Wilson, „On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, Jg. 2, Nr. 1, S. 1–15, 1994, ISSN: 1063-6692. DOI: 10.1109/90.282603.

[152] Eric Jones, Travis Oliphant, Pearu Peterson, *SciPy: Open source scientific tools for Python*, 4.03.2021. Adresse: https://www.scipy.org/.

[153] B. Moayeri, *RZ-Zukunft: Standards oder Proprietäres?* 4.03.2021. Adresse: https://www.comconsult-research.de/rz-zukunft-standards-2/.

[154] T. O'Shea und J. Hoydis, „An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, Jg. 3, Nr. 4, S. 563–575, 2017. DOI: 10.1109/TCCN.2017.2758370.

B Liste der Veröffentlichungen und Fachvorträge auf Tagungen

[1] H. Puttnies, P. Danielis, A. R. Sharif und D. Timmermann, „Estimators for Time Synchronization—Survey, Analysis, and Outlook (Cover Story),“ *MDPI IoT*, Jg. 1, Nr. 2, S. 398–435, 2020. DOI: 10.3390/iot1020023. Adresse: https://www.mdpi.com/2624-831X/1/2/23.

[2] C. Niemann, C. Ewert, H. Puttnies, M. Rethfeldt, D. Timmermann und P. Danielis, „Modeling Energy Consumption for Task-Offloading Decisions on Mobile and Embedded Devices,“ in *2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech)*, Kyoto, Japan: IEEE, 3/10/2020 - 3/12/2020, S. 400–404, ISBN: 978-1-7281-7063-3. DOI: 10.1109/LifeTech48969.2020.1570618809.

[3] H. Puttnies, E. Schweissguth, D. Timmermann und J. Schacht, „Clock Synchronization Using Linear Programming, Multicasts, and Temperature Compensation,“ in *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA: IEEE, 2019, S. 1–6, ISBN: 978-1-7281-0962-6. DOI: 10.1109/GLOBECOM38437.2019.9013260.

[4] J. Schacht, H. Laqua, I. Muller, H. Puttnies und J. Skodzik, „The Trigger-Time-Event System for Wendelstein 7-X: Overview and First Operational Experiences,“ *IEEE Transactions on Nuclear Science*, Jg. 66, Nr. 6, S. 969–973, 2019, ISSN: 0018-9499. DOI: 10.1109/TNS.2019.2913802.

[5] H. Puttnies, P. Danielis und D. Timmermann, „PTP-LP: Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP,“ in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates: IEEE, 2018, S. 1–7, ISBN: 978-1-5386-4727-1. DOI: 10.1109/GLOCOM.2018.8647777.

[6] H. Puttnies, P. Danielis, E. Janchivnyambuu und D. Timmermann, „A Simulation Model of IEEE 802.1 AS gPTP for Clock Synchronization in OMNeT++,“ in *OMNeT++ Summit 2018*, Pisa, Italy, 2018, S. 63–72.

[7] H. Puttnies, P. Danielis, L. Thiele und D. Timmermann, „jUDPWrapper: A Lightweight Approach to Access the OMNeT++/INET UDP Functionality from Java,“ in *OMNeT++ Summit 2018*, Pisa, Italy, 2018, S. 1–10.

[8] P. Danielis, H. Puttnies, E. Schweissguth und D. Timmermann, „Real- Time Capable Internet Technologies for Wired Communication in the Industrial IoT-a Survey,“ in *Proceedings 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Turin: IEEE, 2018, S. 266–273, ISBN: 978-1-5386-7108-5. DOI: 10.1109/ETFA.2018.8502528.

[9] H. Puttnies, C. Niemann, S. Rohde, D. Timmermann und J. Schacht, „Towards software performance estimation based on register-transfer level descriptions,“ in *2017*

IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP & International Symposium on System-on-Chip (SoC), J. Nurmi, Hrsg., Linkoping, Sweden: IEEE, 2017, S. 1–6, ISBN: 978-1-5386-2844-7. DOI: 10.1109/NORCHIP.2017.8124967.

[10] H. Puttnies, P. Danielis, C. Koch und D. Timmermann, „Java Extensions for OMNeT++," in *OMNeT++ Summit 2017*, Bremen, Germany, 2017.

[11] H. Puttnies, D. Timmermann und P. Danielis, „An approach for precise, scalable, and platform independent clock synchronization," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, S. 461–466. DOI: 10.1109/CCNC.2017.7983152.

[12] H. Puttnies, B. Konieczek, J. Heller, D. Timmermann und P. Danielis, „Algorithmic approach to estimate variant software latencies for latency-sensitive networking," in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016, S. 1–7. DOI: 10.1109/IEMCON.2016.7746281.

[13] J. Schacht, H. Laqua, I. Muller, J. Skodzik und H. Puttnies, „The trigger-time-event-system for Wendelstein 7-X: Overview and first operational experiences," in *2016 IEEE-NPSS Real Time Conference (RT)*, Padova, Italy: IEEE, 2016, S. 1–5, ISBN: 978-1-5090-2014-0. DOI: 10.1109/RTC.2016.7543115.

[14] H. Puttnies, V. Altmann, F. Golatowski und D. Timmermann, „Cost-efficient universal approach for remote meter reading using web services and computer vision (workshop paper)," in *2015 IEEE 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, L. Bononi, Hrsg., Boston, MA, USA: IEEE, 2015, S. 1–6, ISBN: 978-1-4799-8461-9. DOI: 10.1109/WoWMoM.2015.7158205.

C Liste der betreuten studentischen Arbeiten

[1] C. Koch, *Masterarbeit: Konzeption und Implementierung eines FPGA-basierten Konverters vom Zeitformat Trigger-Time-Event (TTE) zum Precision Time Protocol (PTP)*, 2020.

[2] F. Ahmad, *Spezialisierungsmodul: Evaluation of an HTTP-based Interface for Configuration and Status Data of the Third Generation of Trigger-Time-Event (TTE) Devices*, 2020.

[3] C. Koch, *Projektarbeit: Implementierung und Evaluation eines FPGA-basieren Konverters zwischen Trigger-Time-Event-System und dem Precision Time Protocol für das W7-X Experiment*, 2020.

[4] A. R. Sharif, *Masterarbeit: FPGA-Based Implementation and Evaluation of the Time Synchronization Approach SLMT*, 2020.

[5] A. R. Sharif, *Spezialisierungsmodul: Using a Testbed to Evaluate PTP-LP: An Approach Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP*, 2020.

[6] L. S. Thiele, *Bachelorarbeit: Verwendung eines Testbeds zur Evaluation von SLMT: Einem Ansatz zur Zeitsynchronisation mittels linearer Programmierung, Multicasts und Temperaturkompensation*, 2020.

[7] H. Asghar, *Masterarbeit: Implementation and Evaluation of PTP-LP: An Approach Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP*, 2019.

[8] C. Ewert, *Bachelorarbeit: Modellierung des Energieverbrauchs für Mobile Cloud Computing*, 2019.

[9] C. Koch, *Bachelorarbeit: Weiterentwicklung einer Steuerungseinheit für die Piezo-Gasventile der Wendelstein 7-X Experimentieranlage*, 2019.

[10] A. R. Sharif, *Projektseminar: Implementing the Approach: Exploiting a Natural Network Effect for Scalable, Fine-grained Clock Synchronization (HYGENS)*, 2019.

[11] E. Janchivnyambuu, *Projektarbeit: Development of a Simulation Model of IEEE 802.1AS (gPTP) in OMNeT++*, 2018.

[12] E. Mahmoud, *Projektarbeit: Optimization of MQTT-SN for the Industrial Internet of Things*, 2018.

[13] J. Heller, *Masterarbeit: Entwicklung und Bewertung einer verteilten Content-Caching-Lösung für IEEE-802.11s-WLAN-Mesh-Netzwerke*, 2018.

[14] H. Krünägel, *Projektseminar: Das Kalman-Filter: Eine Einführung*, 2018.

[15] J. Heller, *Projektarbeit: Berücksichtigung von physikalischer Netzwerk-Nähe im DHT-Protokoll Kademlia*, 2017.

[16] H. Asghar, *Spezialisierungsmodul: Development of an MQTT-SN Simulation Model in OMNeT++ and its Evaluation Regarding the Industrial IoT*, 2017.

[17] S. Behl, *Bachelorarbeit: Implementierung und Performanceanalyse des Autorisierungs-verfahrens OAuth 2.0 für Smart-Home-Szenarien*, 2016.

[18] J. Heller, *Bachelorarbeit: Implementierung eines softwarebasierten, plattformunabhängigen Ansatzes zur Gerätesynchronisation*, 2016.

[19] S. Rohde, *Masterarbeit: Betrachtung der Hardware-Software-Partitionierung im Rahmen der Entwicklung eines Hardware-Software-Co-Designs*, 2016.