Fachbereich Ingenieurwissenschaften und Industriedesign

Hochschule Magdeburg • Stendal
university of applied sciences

B A C H E L O R A R B E I T

Thema:

Konzeption und Untersuchung eines Konzeptes zur hochpräzisen Synchronisierung von verteilten Messsystemen in der Automation

vorgelegt von:

Patrick Biermann

geb. am: 15. Januar 1988

DA-Nr. : 20122431

zur Erlangung des akademischen Grades

Bachelor of Engineering

Erstprüfer: Prof. Dr.-Ing. Olaf Friedewald

Zweitprüfer: Dr.-Ing. Lutz Rauchhaupt

Magdeburg, den 26. Februar 2016

I. Kopie der Aufgabenstellung

Die Diversität von eingesetzten Kommunikationslösungen in der Fabrikautomation aber auch in der Prozessautomation erschwert den Einsatz bekannter monolithischer Diagnosewerkzeuge. Neben der Durchmischung von funk- und kabelgebundener Kommunikationsverfahren nimmt der Einsatz von verteilten Steuerungslösungen zu. Dies erzwingt den Einsatz von dedizierten Messverfahren, die multiple Informationsquellen und vermaschte Kommunikationspfade abdecken. Zum Einsatz kommen demnach auch logisch und örtlich verteilte Messstellen, deren Zeitsynchronizität in der Dimension besser sein muss, als das Zeitverhalten der zu messenden Kommunikationsanwendungen.

Hieraus ergibt sich folgende Aufgabe:

• Recherche existenter Synchronisierungslösungen im industriellen Umfeld mit Bewertung dieser unter anwendungsspezifischen Anforderungen(bzgl. der zu erwartenden Kommunikation)

• Konzepterstellung zum Einsatz ausgewählter Synchronisierungslösungen in eingebetteten Systemen

• Einarbeitung in die technischen Anforderungen zur Inbetriebnahme des Konzepts auf vorgegebenen eingebetteten Plattformen

• Prototypische Inbetriebnahme der Systeme

• Erstellung eines Konzeptes zur Messung der erreichten Synchronizität

• Messung der Synchronizität

• Evaluierung und Auswertung der Messung gemäß der Erwartungen aus der Recherche

• Bewertung der erreichten Synchronizität zum Einsatz in verteilten Messsystemen

• Ausarbeitung von Maßnahmen zur Erhöhung und Aufrechterhaltung der Synchronizität im Last- oder Fehlerfall

Die Ergebnisse dieser Arbeit sind nach den gültigen wissenschaftlichen Standards zu dokumentieren.

II. Kurzreferat mit bibliografischer Beschreibung in deutscher und englischer Sprache

Es werden die Anforderungen verschiedener Kommunikationslösungen in der Automation hinsichtlich der benötigten Synchronizität betrachtet. Im Anschluss wird ein Konzept erstellt und bewertet, welches die bisherigen netzwerkbasierten Synchronisationslösungen PTP und satellitengestütztes NTP anhand existierender Implementierungen kombiniert. Die gewonnenen Erkenntnisse werden mit den im Vorfeld ermittelten Anforderungen verglichen und bewertet. Es wird zu dem Schluss gekommen, dass das vorgestellte Konzept ausreicht, um Ereignisse auf eine Zykluszeit genau zuzuordnen.

The requirements of various existing communications solutions prevalent in industrial automation in respect to their required synchronicity are presented. Then a concept is being presented, in which the network-based synchronization solutions PTP and satellite-based NTP are combined via preexisting implementations. The findings are then compared and evaluated with the determined requirements in mind. It is shown that the proposed concept is indeed capable of correlating events to the precision of one cycle time.

Patrick Biermann
Hochschule Magdeburg - Stendal
Fachbereich Ingenieurwissenschaften und Industriedesign
Institut für Elektrotechnik
Bachelorarbeit-Nr.: 20122431
64 Seiten; 26 Abbildungen.; 11 Tabellen ; 4 Anlagen

III. Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe und noch nirgendwo anders eingereicht habe.

Magdeburg, den 26. Februar 2016 .....

Ort, Datum Unterschrift des Verfassers

IV. Inhaltsverzeichnis

V. Formelzeichenverzeichnis

Tabelle 1: Verwendete Formelzeichen

Formelzeichen	Benennung	Einheit
T	Zeit	[t] = 1 s

VI. Abkürzungsverzeichnis

Tabelle 2: Abkürzungstabelle

Abkürzung	Bedeutung
BC	Boundary Clock
CSMA/CD	Carrier sense multiple access with collision detection
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
IEEE	Institute of Electrical and Electronics Engineers
MESZ	Mitteleuropäische Sommerzeit
MEZ	Mitteleuropäische Zeit
NTP	Network Time Protocol
OC	Ordinary Clock
PPS	Pulse-per-second
PTP	Precision Time Protocol
TAI	Internationale Atomzeit, von frz. "Temps Atomique International"
TC	Transparent Clock
TDMA	Time division multiple access
UDP	User Datagram Protocol
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
UTC	Coordinated Universal Time, Koordinierte Weltzeit
VLAN	Virtual Local Area Network
WAN	Wide Area Network

VII. Abbildungsverzeichnis

VIII. Tabellenverzeichnis

1 Einleitung

Durch die zunehmende Vernetzung von Automatisierungsgeräten wird es immer wichtiger, eine gemeinsame Zeitbasis sicherzustellen, um die zugrundeliegenden Prozesse überwachen, diagnostizieren und steuern zu können. Dies wird durch die Vielzahl existierender Kommunikationslösungen und deren eigene Anforderungen an die Zeitbasis immer schwieriger. Ein zusätzliches Problem stellt die Synchronisierung über Netzwerkgrenzen hinweg dar, wie sie zum Beispiel bei örtlich verteilten oder mobilen Kommunikationspartnern vorkommt, da hier meist auf nichtdeterministische Kommunikationslösungen wie beispielsweise switched Ethernet oder UMTS zurückgegriffen wird. Ebenfalls zu beachten ist die immer mehr zunehmende Durchmischung dieser kabelgebundenen und kabellosen Kommunikationspfade.

Aus diesem Sachverhalt folgen zwei Problemstellungen: Zum einen muss trotz der möglichen räumlichen Trennung der Messstationen zuerst eine geeignete gemeinsame Zeitbasis gefunden werden, zum anderen müssen diese Zeitbasen in allen Messstationen regelmäßig synchronisiert werden. Die regelmäßige Synchronisation sorgt dafür, dass die Lokalzeiten der Messstationen nicht durch leicht unterschiedlich schwingende lokale Oszillatoren oder Phasenrauschen auseinander driften.

Ziel dieser Arbeit ist es nun, unter Betrachtung anwendungsspezifischer Anforderungen, bestehende Synchronisierungslösungen zu evaluieren, deren bestmögliche Parameterkombination zu ermitteln und dann ggf. miteinander zu kombinieren um so deren inhärente Nachteile auszugleichen. Dazu werden erst die Eigenschaften verschiedener Zeitbasen betrachtet, die den betrachteten Synchronisierungslösungen zu Grunde liegen. Aus diesen Informationen wird ein Konzept zur Vernetzung der bestgeeignetsten Einzellösungen erstellt. Daraufhin wird ein Messaufbau entwickelt, in Betrieb genommen und das Konzept anhand dieses Aufbaus unter Berücksichtigung der ermittelten Anforderungen evaluiert.

2 Stand der Technik

In diesem Abschnitt werden derzeitige Lösungen betrachtet und die Kommunikation bezüglich deren Anforderungen beleuchtet.

2.1 Begriffsklärung

2.1.1 Synchronizität

Es herrscht Synchronizität zwischen zwei Punkten, wenn die Differenz der Zeiten beider Punkte Null beträgt. Da dies in der Praxis technisch nicht zu erreichen ist, gilt für das Maß der erreichten Synchronizität: Die Synchronizität zwischen zwei Punkten nimmt zu, je kleiner die Differenz der Zeiten in diesen Punkten ist.

2.1.2 Zeitnormal

Ein Zeitnormal ist die Quelle einer genauen Uhrzeit mit möglichst geringen Abweichungen. Üblicherweise dienen heute Atomuhren als Zeitnormal.

2.1.3 Stratum

In der Network Time Protocol Terminologie ist Stratum (Plural: Strata) ein Ausdruck für die Entfernung eines Zeitservers zu einem Zeitnormal. Stratum 0 bezeichnet ein Zeitnormal wie zum Beispiel ein GPS-Satellitenempfänger oder eine Atomuhr. Ein Zeitserver, der direkten Zugang zu einer Stratum-0-Quelle hat, ist selbst ein Stratum 1 Server. Ein Zeitserver, der seine Zeit von einem Stratum-1-Server bezieht, ist selbst ein Stratum-2 Server, usw. [DMi10]. Niedrige Stratumwerte werden bei der Auswahl eines Zeitservers bevorzugt und gelten als den höherwertigen Stratumwerten als übergeordnet.

2.1.4 Zeitstempel

In dieser Arbeit wird ein Zeitstempel als ein zu einem Ereignis eindeutig zuordenbarer Zeitpunkt verstanden. Es wird zwischen Soft- und Hardwarezeitstempeln unterschieden: Bei Softwarezeitstempeln wird die Zeit des eintretenden Ereignisses durch Programme ermittelt und festgehalten, bei Hardwarezeitstempeln erfolgt die Ermittlung des Zeitpunktes eines Ereignisses auf Hardwareebene. Nähere Erläuterungen folgen in Abschnitt 2.4.4. "Zeitstempeln" bedeutet, den Zeitpunkt eines Ereignisses festzuhalten.

2.2 Zeitnormale

Dieser Abschnitt beschäftigt sich mit einigen Zeitnormalen, die zur Kalibrierung der im System geltenden Zeit zu Grunde gelegt werden können.

2.2.1 Lokalzeit

Unter Lokalzeit wird in dieser Arbeit die Festlegung einer für das System geltenden Zeit von Hand verstanden. Diese hat sowohl in Phase als auch Frequenz nicht zwangsläufig einen Bezug zur Außenwelt.

2.2.2 Atomuhren, internationale Atomzeit und die koordinierte Weltzeit

Die Verwendung von Atomuhren als Zeitnormal hat sich aufgrund ihrer hohen Genauigkeit und minimalen Unsicherheiten schon vor langer Zeit etabliert. Die Dauer einer Sekunde ist seit 1967 an die Energiezustände des Caesium-133-Atoms gebunden [Amt77] und ist mit hoher Zuverlässigkeit bestimmbar.

Prinzipiell funktionieren Cäsiumatomuhren, indem ein gasförmiger Strom aus Cäsium-133-Isotopen anhand derer zwei unterschiedlichen energetischen Grundzustände sortiert wird. Die niederenergetischen Atome werden mit Mikrowellen bestrahlt, erneut sortiert und von einem Detektor aufgelesen. Über die Anzahl der aufgelesenen Atome die ihren Energiezustand gewechselt haben wird die Frequenz des Mikrowellenstrahlers nachgeregelt. Wenn die Frequenz des Mikrowellenstrahlers exakt stimmt, wird die maximale Anzahl an Cs-Atome in ihren höherenergetischen Energiezustand überführt. Dann beträgt die Frequenz der Mikrowellenstrahlung genau 9192631770 Hz [Rie04]. Durch diese Relation sind Zeitintervalle zwar sehr genau messbar.

Oftmals ist aber nicht die Dauer eines Ereignisses, sondern dessen Zeitpunkt gefragt. Um eine weltweit einheitliche Zeitskala zu schaffen werden weltweit die Zeiten von ungefähr 400 Atomuhren verglichen und daraus die internationale Atomzeit TAI gebildet. Aufgrund der sich immer weiter verlangsamenden Erdrotation werden in unregelmäßigen Abständen Schaltsekunden notwendig, die in der internationalen Atomzeit nicht berücksichtigt werden. Aus der internationalen Atomzeit und der Summe aller Schaltsekunden wird schlussendlich die koordinierte Weltzeit UTC gebildet [Pie15].

2.3 Zeitzeichensender

Zeitzeichensender verteilen die Zeit, sodass diese von geeigneten Empfangsgeräten ausgewertet werden kann.

2.3.1 DCF77

Der von der Physikalisch-Technischen Bundesanstalt betriebene Langwellensender DCF77 sendet seit 1959 ein Zeit- und Frequenznormal auf einer Trägerfrequenz von 77,5 kHz. Seit 1973 wird auf dieser Frequenz zusätzlich die für Deutschland rechtlich geltende Uhrzeit MEZ bzw. MESZ durch Amplitudenmodulation aufgeprägt. Die Trägerfrequenz wird von den Atomuhren der Physikalisch-Technischen Bundesanstalt abgeleitet und über eine Messdauer von einem Tag mit einer relativen Genauigkeit von unter 2e-12 angegeben. Der Beginn einer aufmodulierten Sekundenmarke wird im Bereich von 5,5 ±0,3 µs vom tatsächlichen Sekundenbeginn nach UTC gehalten [Dir04].

2.3.2 Satellitennavigation

Es gibt einige satellitengestütze Navigationssysteme (GNSS), wie beispielsweise das amerikanische Global Positioning System, das russische GLONASS oder das europäische Galileo-Projekt. Aufgrund der weiten Verbreitung durch die preiswerte Empfangstechnik beschränkt sich diese Arbeit allerdings vorerst auf das Global Positioning System.

An Bord der derzeit 32 im All befindlichen aktiven Satelliten des Global Positioning Systems befinden sich hochgenaue Atomuhren. Jeder dieser Satelliten sendet kontinuierlich sowohl seine genaue Position als auch die genaue Uhrzeit. Hierbei muss beachtet werden, dass die gesendete Zeit weder die koordinierte Weltzeit noch die internationale Atomzeit ist, sondern die sogenannte GPS-Zeit. Die GPS-Zeit basiert auf der koordinierten Weltzeit zum Zeitpunkt des 6. Januar 1980 um 00:00:00 aber berücksichtigt keine weiteren Schaltsekunden. Zu diesem Zeitpunkt herrschte durch bereits eingefügte Schaltsekunden eine Differenz von 19 s zwischen internationaler Atomzeit und koordinierter Weltzeit [Dan90]. Da wie eben erwähnt die GPS-Zeit keine weiteren Schaltsekunden berücksichtigt, liegt Stand 1. Juli 2015 eine Differenz von17 s zwischen GPS-Zeit und koordinierter Weltzeit, d.h.

zwischen internationaler Atomzeit und koordinierter Weltzeit eine Differenz von 36 s [Gam15].

Nochmal im Überblick:

$$GPS = TAI - 19s = UTC + 17s \qquad \text{(I)}$$

Wenn man von dieser systematischen Abweichung absieht, liegt die Differenz zwischen der GPS-Zeit und koordinierten Weltzeit meist unter 100 ns [Bau16].

2.4 Synchronisationsmechanismen

Es bestehen bereits einige Synchronisationslösungen, von denen hier eine kleine Auswahl näher beleuchtet wird.

2.4.1 Elektrische oder optische Signale

Um die Synchronizität von Geräten sicherzustellen, können getaktete Signalleitungen verwendet werden unter deren Zuhilfenahme eine Synchronisierung vorgenommen werden kann. Dies ist in der Regel bei örtlich verteilten Systemen nicht durchführbar, da neben dem zu betreibenden Mehraufwand des Verlegens der Leitungen auch Signallaufzeitdifferenzen und Dämpfungseffekte berücksichtigt werden müssen.

2.4.2 Network Time Protocol

Das Network Time Protocol bestimmt die zeitlichen Abweichungen von einem Server zu einem Peer und synchronisiert den Peer auf die vom Server verteilte Zeit. Jede Network Time Protocol Instanz kann dabei eine von drei Rollen annehmen: Ein mit einem Zeitnormal wie einer Atomuhr oder einem Satellitenempfänger verbundener Rechner kann als Primärserver dienen, der seine Zeit an ihm untergeordnete Peers weiterreichen kann. Ein Sekundärserver kann sich auf diese Zeit synchronisieren und diese Zeit ebenfalls an ihm untergeordnete Peers weiterreichen. Peers können sich nur mit Servern synchronisieren, die ihnen übergeordnet sind und geben die so erlangte Zeit nicht weiter. In diesem Zusammenhang bedeutet "unter-" bzw. "übergeordnet" eine netzwerktopologische Beziehung, den Strata (siehe 2.1.3).

Die Zeitserver können dabei ihre Zeit auf eine von vier Arten verteilen: Sie können eine Anfrage eines Peers beantworten, sie direkt an einen vorkonfigurierten Peer senden oder per Multi- oder Broadcast im Netz verteilen.

Der Synchronisierungsmechanismus basiert auf einem Request-Response Modell. Der Peer sendet eine Anfrage an den Server und speichert den Sendezeitpunkt. Daraufhin sendet der Zeitserver eine Antwort, die sowohl den Zeitpunkt der Ankunft der Anfrage als auch den serverseitigen Sendezeitpunkt beinhaltet. Aus diesen drei Zeitstempeln und dem Zeitpunkt des Eintreffens der Antwort beim

Peer kann die Abweichung zum Server berechnen. Die Signallaufzeiten auf Hin- und Rückpfad werden als symmetrisch angenommen [DMi10]. Dieser Ablauf ist in Anlehnung an UML-Sequenzdiagramme in Abbildung 1 dargestellt. Die vom Standard abweichende Darstellung wurde gewählt, um explizit auf die auftretenden Verzögerungszeiten hinzuweisen.

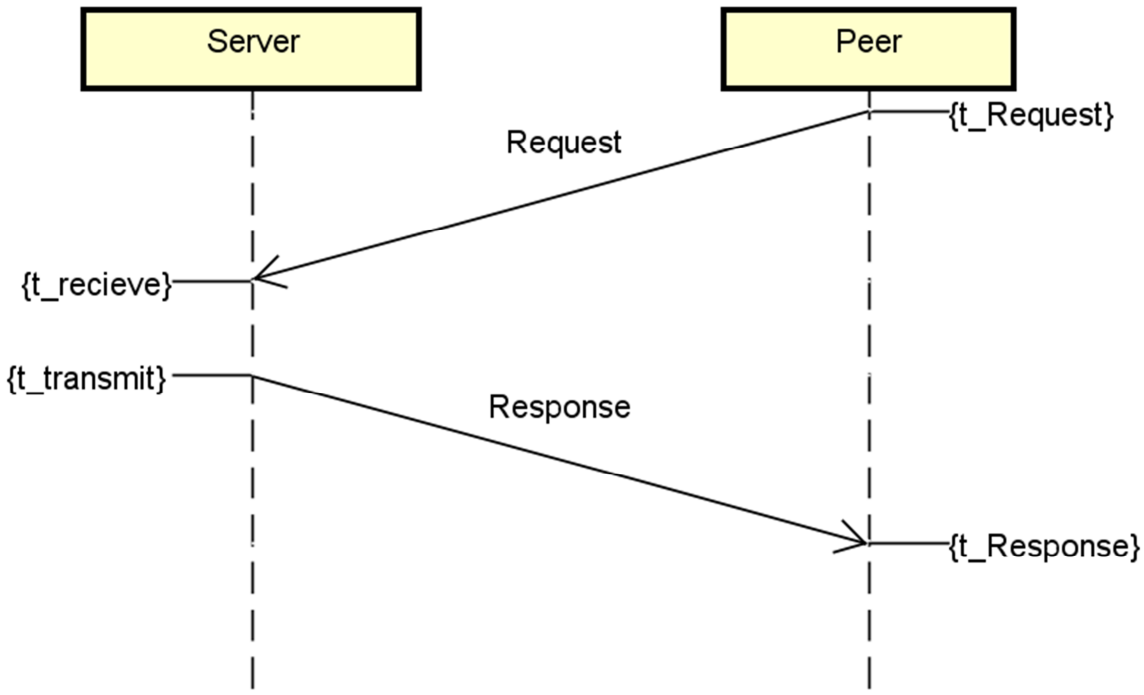

Abbildung 1: Synchronisierungsmechanismus NTP

Die Abweichung von der Uhrzeit des Servers ließe sich demnach mit

$$\Delta t = \frac{(t_{recieve} - t_{Request}) + (t_{transmit} - t_{Response})}{2} \qquad \text{(II)}$$

errechnen.

Die Uhrzeit auf dem Peer wird allerdings nicht sofort verändert. Erst nach mehreren dieser Abläufe, Filterung und statistischer Auswertung wird der Server als zuverlässig angesehen und die Uhrzeit angepasst. Etwaige Frequenzabweichungen werden durch eine Phasenregelschleife ausgeglichen.

Zu beachten ist auch, dass ein Peer Anfragen an mehrere Server stellen kann und die Genauigkeit der Uhrzeit sowohl mit der Anzahl der zu Verfügung stehenden

Zeitserver als auch über einen längeren Zeitraum hinweg immer besser wird, da hier schlechte Server mit der Zeit aussortiert werden können.

2.4.2.1 SNTP

Bei dem Simple Network Time Protocol handelt es sich um eine vereinfachte Version des Network Time Protocols. Während das Network Time Protocol dafür ausgelegt ist, Sekundärserver mit mehreren über- und untergeordneten Strata zu verwalten, wird das Simple Network Time Protocol durch die Limitierung auf Primärserver mit nur einem Zeitnormal oder Clients mit nur einem übergeordnetem Primärserver vereinfacht. Unter allen anderen Gesichtspunkten sind diese beiden Protokolle kompatibel [DMi10].

2.4.3 Precision Time Protocol

Das im IEEE Standard 1588-2008 definierte Precision Time Protocol (PTPv2) dient zur Synchronisierung von Netzwerkteilnehmern in einem lokalen, multicastfähigen Netzwerk. Ziel des Protokolls ist es, mit nur wenig zusätzlicher Netzauslastung und ohne Zutun des Nutzers eine Synchronizität unterhalb einer Mikrosekunde zu erreichen.

Um diese Synchronizität zu erreichen, wird das Netz als System untereinander kommunizierender Uhren (Ordinary Clocks) betrachtet. Durch den im Standard spezifizierten sogenannten Best Master Clock Algorithmus wird unter allen Ordinary Clocks unter Berücksichtigung ihrer Eigenschaften hinsichtlich der Art ihres Zeitnormals, Stabilität und Genauigkeit eine als "Grandmaster Clock" für das Netz ausgewählt. Diese dient als Zeitbasis für das gesamte Netz, auf die sich alle anderen Uhren, fortan Slaves, synchronisieren.

Netzwerkkomponenten wie Router und Switches werden entweder als Boundary oder Transparent Clocks implementiert. Transparent Clocks leiten die eingehenden PTP-Nachrichten weiter und übermitteln den in ihnen entstehenden Delay im Korrekturfeld der Nachricht, sodass dieses später herausgerechnet werden kann. Boundary Clocks synchronisieren sich als Slave auf die Grandmaster Clock und übernehmen für das ihnen unterstellte Netz dann selbst die Rolle der Grandmaster Clock.

2.4.3.1 Synchronisierungsmechanismen

Die Synchronisation erfolgt nach Auswahl der Grandmaster Clock zyklisch. Die Grandmaster Clock sendet zuerst eine Synchronisierungsnachricht (Sync Message), deren Eintreffzeitpunkt beim Slave gespeichert wird. Unmittelbar danach übermittelt die Grandmaster Clock den genauen Versendezeitpunkt der Sync Message in einer Folgenachricht (Follow Up) an den Slave. Dieser Ablauf ist in Abbildung 2 dargestellt.

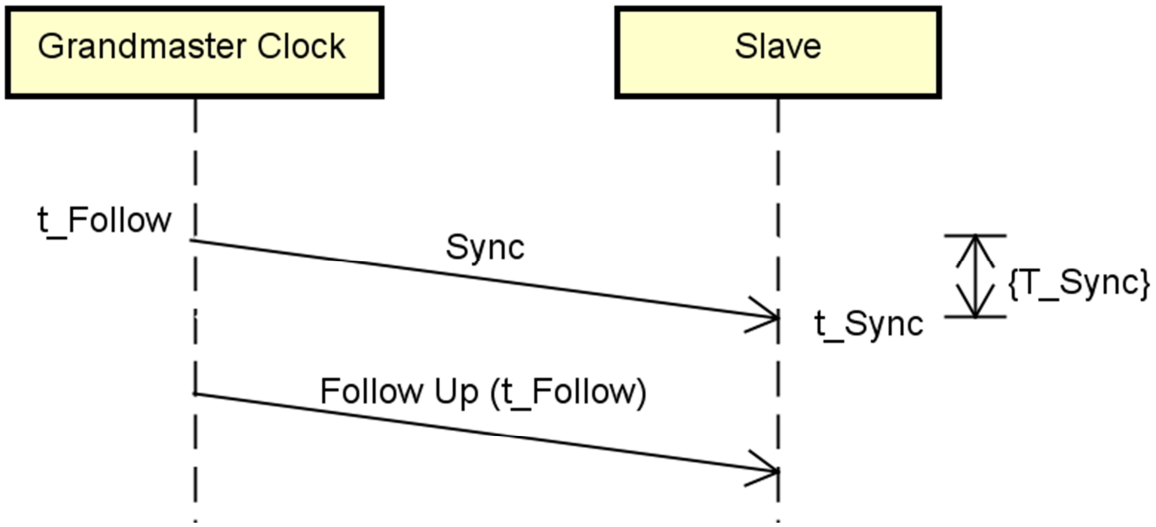

Abbildung 2: PTP Sync Message Ablauf

Nun ist dem Slave bekannt, dass eine zeitliche Abweichung zwischen ihm und dem Grandmaster besteht, aber um den absoluten Offset zu bestimmen und die eigene Uhr nachregeln zu können, müssen die Nachrichtenlaufzeiten bekannt sein. Um diese festzustellen, sind derzeit zwei Mechanismen implementiert.

2.4.3.2 Der end-to-end Mechanismus

Der Slave sendet, wie in Abbildung 3 aufgezeigt, eine Delay-Request-Message an die Grandmaster Clock und speichert deren Sendezeitpunkt. Die Grandmaster Clock antwortet in einer Delay-Response-Message mit dem Zeitpunkt des Eintreffens des Requests.

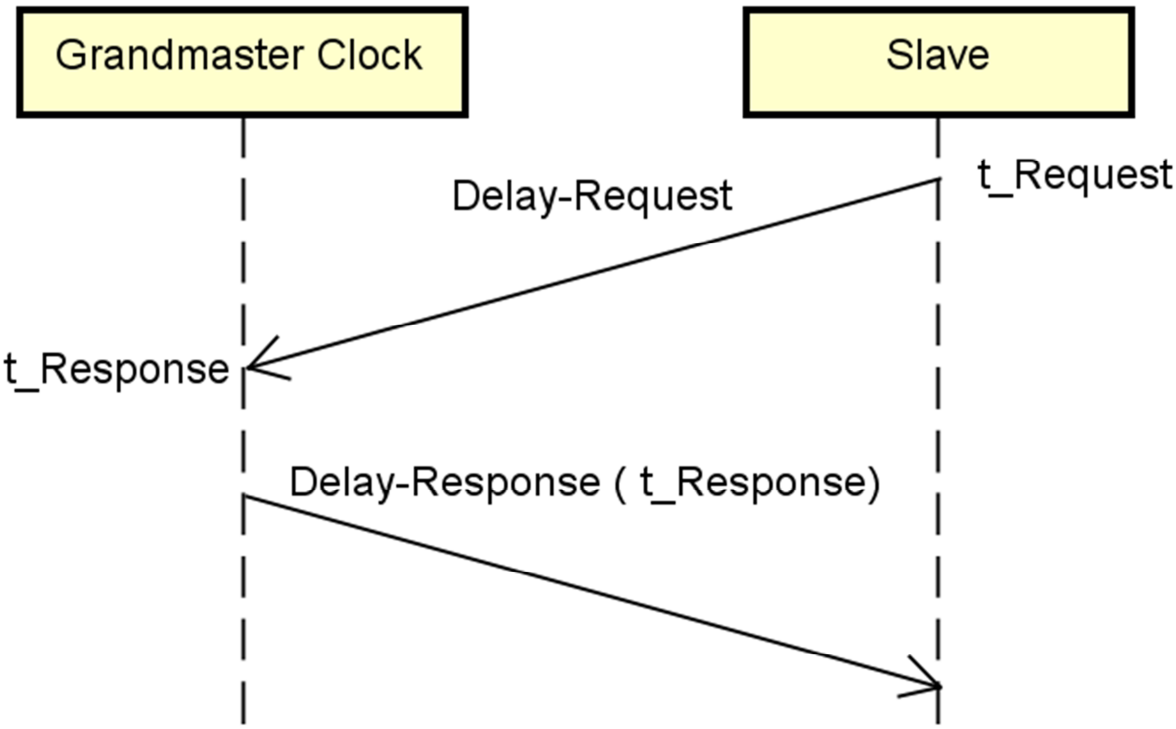

Abbildung 3: Der end-to-end-Mechanismus

Unter der Annahme, dass Hin- und Rückweg symmetrisch sind, lässt sich die Abweichung von Grandmaster Clock und Slave mit

$$\Delta T_{Offset} = \frac{\Delta T_{Sync}}{2} - \Delta t_{Delay} \qquad \text{(III)}$$

berechnen, wobei

$$\Delta T_{Sync} = t_{Sync} - t_{Follow} \qquad \text{(IV)}$$

$$\Delta t_{Delay} = \frac{t_{Request} - t_{Response}}{2} \qquad \text{(V)}$$

2.4.3.3 Der peer-to-peer Mechanismus

In der zweiten Protokollversion von PTP wurde unter anderem der peer-to-peer Mechanismus eingeführt. Im Gegensatz zum end-to-end Mechanismus sendet der Slave die Delay-Request-Message nicht direkt zur Grandmaster Clock, sondern nur zum nächsten PTPv2-fähigen Netzwerkknoten, welcher ebenfalls die

Verzögerung zwischen sich und seinem übergeordneten Knoten ermittelt hat. Der Sendezeitpunkt wird im Slave, der Empfangszeitpunkt im Knoten gespeichert. Dieser Knoten antwortet sofort mit einer Peer-Delay-Response, deren Eingangszeitpunkt vom Slave gespeichert wird. Der Netzwerkknoten sendet zusätzlich ein Peer-Delay-Response Follow-Up, das die Zeitdifferenz zwischen Eintreffen des Requests und senden der Response enthält. Der gesamt Vorgang wird in Abbildung 4 noch einmal verdeutlicht.

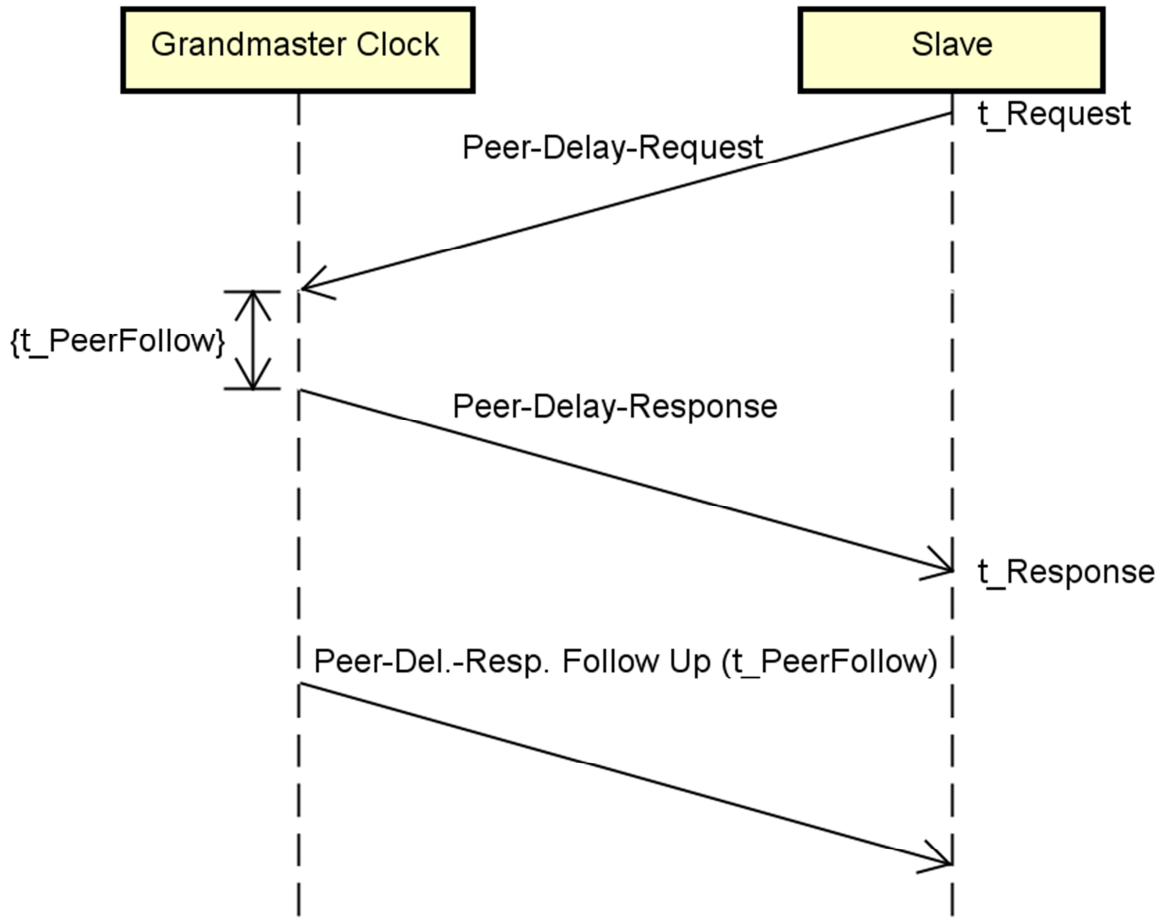

Abbildung 4: Der peer-to-peer-Mechanismus

Damit berechnet sich die Verzögerung durch den Netzwerkabschnitt durch

$$\Delta t_{PeerDelay} = \frac{t_{Response} - t_{Request} - \Delta t_{PeerFollow}}{2} \qquad \text{(VI)}$$

Um nun die Abweichung von der tatsächlichen Zeit in der Grandmaster Clock zu berechnen, werden alle vorhergegangenen $\Delta t_{PeerDelay}$ auf dem Pfad zur

Grandmaster Clock im Korrekturfeld der Sync-Follow-Up-Message aufaddiert, sodass sich effektiv der Zeitpunkt t_{Follow} verschiebt. Somit ergibt sich der komplette Offset als

$$\Delta T_{Offset} = \frac{\Delta T_{Sync}}{2} - \Delta t_{PeerDelay} \qquad \text{(VII)}$$

wobei

$$\Delta T_{Sync} = t_{Sync} - t_{Follow} - \sum_{k=0}^{n-1} \Delta t_{PeerDelay_k} + \Delta t_{stack_k} \qquad \text{(VIII)}$$

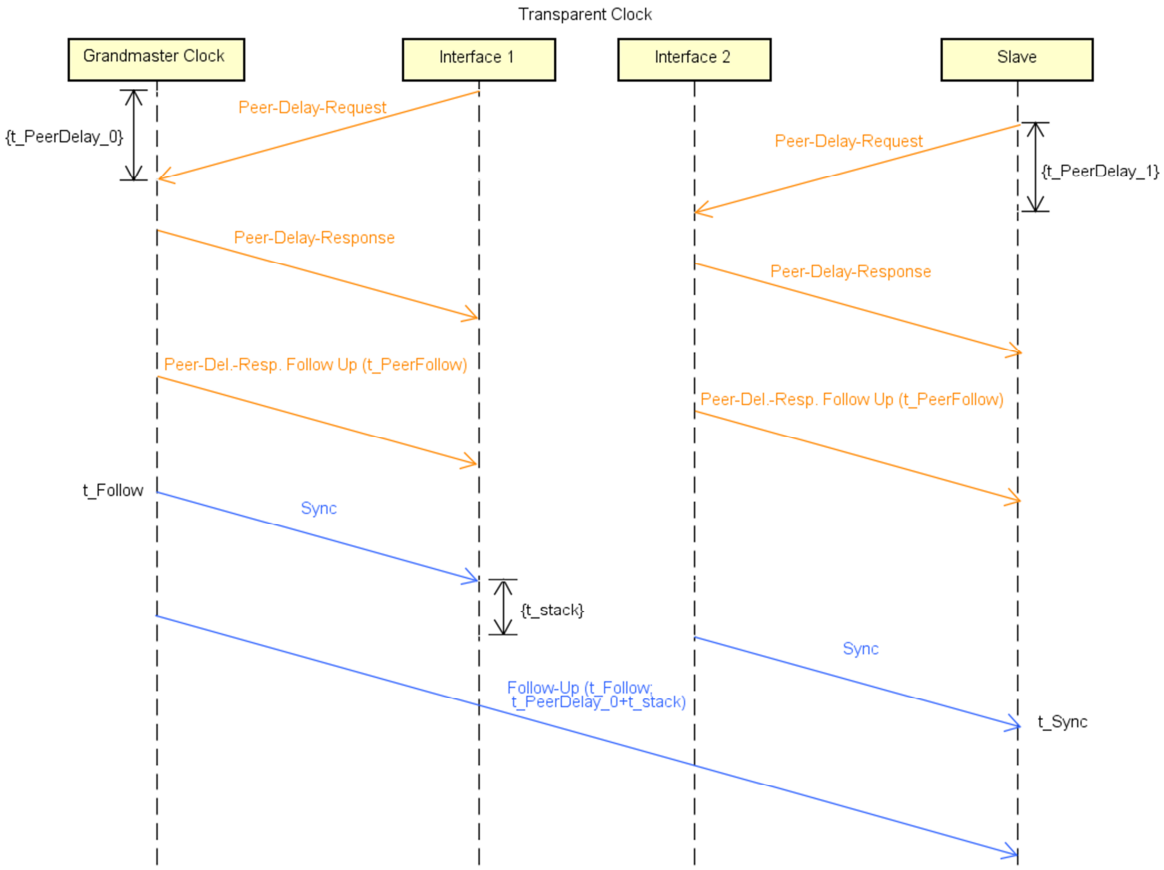

Abbildung 5: Die peer-to-peer Transparent Clock

Der peer-to-peer Mechanismus hat einige Vorteile gegenüber dem end-to-end Mechanismus. Zum einen kann ausgeschlossen werden, dass Sync. und Delay-Messages unterschiedliche Pfade im Netz nehmen, da der Delay immer nur relativ zum Nachbarn gemessen wird, zum anderen wird die Grandmaster Clock entlastet,

da sie nur noch Sync und Follow-Up senden muss und nur die Delay-Requests ihrer direkten Nachbarn beantwortet, anstatt die Requests aller Slaves im Netz bearbeiten zu müssen. Nachteilig ist allerdings, dass alle im Netz verwendeten Switches und Router zwangsweise als Transparent oder Boundary Clock nutzbar sein müssen, da sonst die Delay-Request der Slaves nicht beantwortet werden und die Verzögerung zwischen ihnen und der Grandmaster Clock nicht ermittelt werden kann.

2.4.3.4 Frequenzdrift

Nach Bestimmung des Offsets muss der Frequenzunterschied zwischen dem lokalen Oszillator und der Grandmaster Clock bestimmt und konfiguriert werden, damit die Uhren zwischen den Synchronisationszeitpunkten nicht auseinander driften. Dafür werden slaveseitig die Zeitstempel mehrerer Sync- und Follow-Up-Zyklen überwacht und mittels eines PI-Reglers die lokale Oszillatorfrequenz eingeregelt.

2.4.4 Zeitstempel

Bei Linux-Betriebssystemen gibt es nicht nur eine einzige Uhrzeitquelle, sondern es existieren mindestens zwei Uhren. Eine Systemzeit, die im Userspace zur Verfügung steht und eine Kernelzeit, die Kernelintern verwendet wird. Dabei wird die Systemzeit durch Software von der Kernelzeit abgeleitet. Bei Systemen, die eine hardwaretimestampingfähige Netzwerkschnittstelle besitzen, existiert zusätzlich noch eine Uhr auf dieser Netzwerkschnittstelle. In Abbildung 6 sind die unterschiedlichen Schichten und ihre Übergabepunkte dargestellt. Die unterschiedlichen Uhren sind dabei in den roten Kästen hervorgehoben.

Grundsätzlich gibt es zwei Möglichkeiten, die benötigten Zeitstempel zu generieren: entweder in Hardware oder Software. Im Allgemeinen gilt, dass die Zeitstempel umso besser sind, desto näher sie am Übertragungskanal genommen werden. An Abbildung 6 lässt sich erkennen, wieso: Desto weiter wir uns von der Hardware entfernen, umso mehr Übergabepunkte müssen auf dem Weg überwunden werden, in denen eine Verzögerung auftritt und ein Scheduler den zeitlichen Ablauf beeinflusst. Für hochpräzise Synchronisation (< 1 µs) wird eine

dementsprechende Hardwareunterstützung in Form einer geeigneten Netzwerkschnittstelle vorausgesetzt, die nach Erkennung einer Sync- oder Delay-Message direkt an der Eingangsschnittstelle Zeitstempel erzeugen kann. Die meisten Implementierungen erzeugen diese Zeitstempel an der Schnittstelle zwischen Sicherungsschicht und Bitübertragungsschicht [Hor00]. Durch eine Zeitstempelung in Hardware wird das Problem umgangen, dass Pakete in der Sende- und/oder Empfangswarteschleife zusätzlich und im schlimmsten Falle sogar asymmetrisch verzögert werden. Da die vorgestellten Synchronisierungsmechanismen NTP und PTP davon ausgehen, dass die Verzögerung in beide Richtungen identisch ist, könnten so große Abweichungen entstehen.

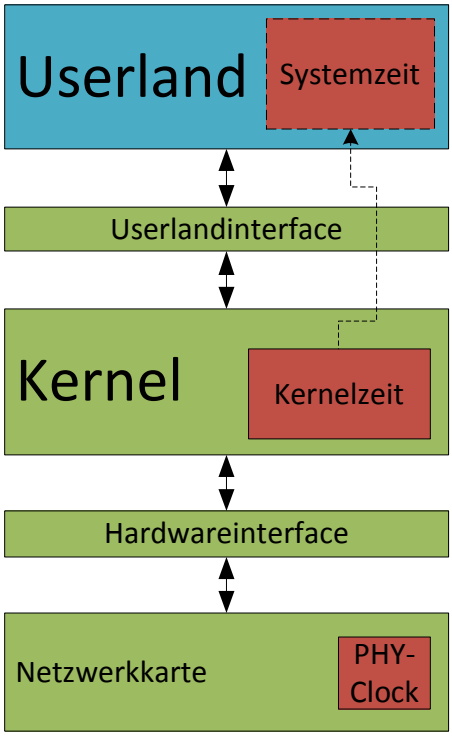

Abbildung 6: Uhrenlayer

2.5 Betrachtung verschiedener Kommunikationsprotokolle in der industriellen Automation

In der Praxis wird in der Industrie auf viele unterschiedliche Kommunikationsprotokolle zurückgegriffen, welche jeder für sich eigene Anforderungen und Ansprüche an die zu erwartende Synchronisationsgenauigkeit fordern. Dieser Abschnitt beschreibt einige Kommunikationsprotokolle, wobei insbesondere auf deren zeitliche Anforderungen eingegangen wird.

2.5.1 PROFIBUS DP

Bei PROFIBUS DP handelt es sich um ein Master/Slave Protokoll, welches ein Tokenweitergabeverfahren nutzt, um mehrere Master pro Bus zu verwalten. Im Gegensatz zu den später aufgeführten Kommunikationsprotokollen basiert PROFIBUS allerdings nicht auf Ethernet, sondern greift meist auf eine Kommunikation über RS485 zurück.

In PROFIBUS Netzwerken gibt es zwei Arten von Geräten: passive Slaves und aktive Master. Master werden zusätzlich noch in die Klassen 1 und 2 unterteilt. Zumeist stellen industrielle Steuerungen einen Master der Klasse 1 dar und die in der Anlage verbauten Sensoren und Aktoren fungieren als Slaves. Der Master sendet dabei zyklisch die Ausgabewerte an die Slaves und erhält die Eingabewerte als Antwort. Dabei versucht der Master die Buszykluszeit möglichst gering zu halten, daraus ergibt sich für jeden Zyklus ein variabler Jitter und eine unbestimmte Zyklusdauer. [Fel09]

PROFIBUS DP wird nochmal unterteilt in drei Unterversionen, die treffenderweise mit V0, V1 und V2 durchnummeriert sind. Zeitsynchronisation wird dabei nur von V2 unterstützt. Innerhalb der PROFIBUS DP V2 Spezifikation existieren zusätzlich Mastergeräte der Klasse 3, die als aktiver Zeitmaster definiert sind. Die Länge der Buszyklen wird nun konstant auf einen vorher definierten Wert gehalten. Zu Beginn jedes Zyklus wird eine Global Control Nachricht an alle Geräte geschickt, welche den Start eines Zyklus markiert. Danach findet der Datenaustausch mit den Slaves statt. Wenn dies geschehen ist, wird der Token weitergereicht und der nächste Master (wenn existent) darf genau einen Datenaustauschzyklus

durchführen. Wenn der Token nun nach einer variablen Zeit zurück am ursprünglichen Master ankommt, belegt dieser nun so lange den Kanal mit Status Request an seine eigene Adresse, bis die konfigurierte Buszykluszeit fast erreicht ist, d.h. die verbleibende Wartezeit nichtmehr für einen vollständigen Request ausreicht. Dies wird aktives Warten genannt. Die restliche Zeit lässt man verstreichen (passives Warten) und sendet zu Beginn des nächsten Zyklus wieder eine Global Control-Nachricht. Auf diesen festen Zyklus können sich alle Slaves synchronisieren, indem sie den Eingangszeitpunkt der Global Control Nachricht zeitstempeln und diesen in eine lokale Phasenregelschleife einspeisen. Über diesen Mechanismus kann dann festgelegt werden, zu welchem Zeitpunkt Eingänge eingelesen und Ausgänge gesetzt werden müssen [Fel09].

2.5.2 EtherCAT

Das EtherCAT Kommunikationsprotokoll basiert auf Ethernet, hat aber die Besonderheit, dass die empfangenen Pakete nicht wie üblich den TCP/IP oder UDP/IP Stack durchlaufen, interpretiert und in die Prozessdaten kopiert werden, sondern die Kommunikationsteilnehmer entnehmen und/oder hinterlegen die Daten noch während der Verarbeitung des Paketes in der Netzwerkschnittstelle. Sämtliche EtherCAT Kommunikation wird von einem Mastergerät initiiert, der in festgelegten Abständen Frames sendet. Angeschlossene Slaves verändern diesen Frame während dieser ihre Netzwerkschnittstelle durchläuft [Doy04]. Die Verarbeitung der Daten wird so noch während des Empfangs angestoßen. Dadurch lässt sich der ansonsten nötige Protokolloverhead reduzieren, so dass der Nutzdatenanteil von 4,7% auf über 90% im Vergleich zu einem normalen Ethernetframe bei 100% Busauslastung steigen kann [IAO04]. Der Frame wird dann umgehend an den nächsten Slave weitergeleitet. Der letzte Slave leitet den Frame zurück an den Master.

Während EtherCAT Mastergeräte durch eine Softwareimplementierung keine besondere Hardware benötigen, müssen die Slaves aufgrund der ungewöhnlichen Verarbeitung des Inhaltes eines Frames im Durchlauf eine hardwareseitige Unterstützung implementieren. EtherCAT ist durch die fehlende Implementierung von IP zwar an das ursprüngliche Subnetz gebunden, unterstützt darin aber

beliebige Netzwerktopologien und kann mit anderen ethernetbasierten Übertragungsverfahren wie der klassischen TCP/IP-Kommunikation oder PROFINET koexistieren [Doy04]. Durch diese Koexistenz können EtherCAT-frames in UDP/IP-Pakete eingekapselt werden und ihre Subnetzgrenze überwinden.

Der Synchronisierungsmechanismus bei EtherCAT basiert auf einem Distributed Clock-Mechanismus. Zu Beginn der Kommunikation sammelt der Master Zeitstempel von den Slaves, berechnet den Offset zu einer Referenzzeitquelle und schreibt diesen Offset zurück an die Slaves, welche daraus die korrekte Zeit ermitteln können. Danach verteilt der Master in periodischen Abständen die Referenzzeit an die Slaves, die damit ihre Abweichungen korrigieren können [Sun13].

2.5.3 PROFINET IO

PROFINET IO funktioniert in der Grundvariante mittels des auf CSMA/CD basierenden Ethernets, welches bekanntlich von Haus aus nichtdeterministisch ist. Nicht zeitkritische Daten wie Diagnose- oder Parametrierungsdaten werden durch die Nutzung des (TCP/UDP)/IP-Protokollstacks übertragen.

Für zeitkritische Daten existieren zwei zusätzliche Varianten des PROFINETs. PROFINET IO RT verwendet einen von der IEEE vergebenen EtherType, der es als PROFINET Real-Time Frames erkenntlich macht. Anhand des EtherTypes und einer den Adressaten kennzeichnenden FrameID kann in den Netzwerkkomponenten schnell gefiltert und priorisiert werden. Zusätzlich wird dem Ethernetframe ein VLAN-Tag hinzugefügt, welches eine bevorzugte Abarbeitung in Switches veranlassen kann [Pop04].

Die taktsynchrone Variante namens PROFINET IO IRT verwendet zusätzlich zu den bei PROFINET IO RT benutzen Eigenschaften noch ein TDMA-Verfahren, welches den Kommunikationskanal nochmals in äquidistante Intervalle zerteilt. Somit besteht ein PROFINET IO RT Buszyklus aus jeweils einem zeitkritischen IRT-Intervall und einem nicht zeitkritischen Intervall. Die anfallende normale (TCP/UDP)/IP-Kommunikation kann weiterhin im nicht zeitkritischen Intervall

stattfinden, während die zeitkritische Kommunikation im IRT-Intervall stattfinden muss. Der Vorteil liegt hier darin, dass eine eindeutige Priorisierung der Daten anhand des verwendeten Zeitschlitzes vorgenommen werden kann.

Um den Beginn der Kommunikationszyklen zwischen allen Geräten synchron zu halten, wird eine gemeinsame Zeitbasis benötigt. Die erforderliche Synchronisierung erfolgt zu Beginn eines Zyklus über das Precision Transparent Clock Protocol, welches in Aufbau und Ablauf identisch mit dem in Abschnitt 2.4.3 geschildertem Precision Time Protocol mit implementiertem Peer-Delay-Mechanismus und Transparent Clocks ist [Pig05]. Es muss also festgehalten werden, dass für PROFINET IO IRT Kommunikation aufgrund dieser Precision Time Protocol Mechanismen hardwaretimestampingfähige Netzwerkkomponenten erforderlich sind

2.6 Anwendungsspezifische Anforderungen

Mit den vorgestellten Kommunikationsprotokollen sind bisher Synchronisierungsgenauigkeiten von 1 µs möglich. PROFINET IRT hat es sich als Ziel gesetzt, eine minimale Zykluszeit von 31,25 µs in 100Mbit/s-Netzen umzusetzen [Jas04]. Bei diesen Kommunikationsprotokollen muss allerdings beachtet werden, dass es sich bei dem Wert von 1 µs um den Jitter handelt, welcher die Differenz zwischen maximaler zu minimaler Startzeit eines Frames beziffert. Für die stattfindende Kommunikation ist der Bezug zur koordinierten Weltzeit also lokal vernachlässigbar. Wenn nun aber zwei Messstationen zum Beispiel durch eine örtliche Trennung nicht über die Kommunikationsprotokolle synchronisierbar sind, die erfassten Messwerte aber trotzdem korreliert werden sollen, muss dieser Bezug sehr wohl hergestellt werden.

Aus diesen beiden Betrachtungen leiten sich für die folgenden Untersuchungen zwei Anforderungsklassen ab:

• Anforderungsklasse I: Für eine bitgenaue Synchronisation der Kommunikation in den Teilnetzen muss gewährleistet sein, dass die Messsysteme derart synchronisiert sind, dass die erreichte Abweichung der Uhren der Messsysteme kleiner als der maximal zulässigen Jitter von 1 µs ist.

• Anforderungsklasse II: Zur Zuordnung von verteilt aufgenommenen Bustelegrammen zueinander muss gewährleistet sein, dass das Messsystem derart synchronisiert ist, dass die erreichte Abweichung der Uhren der Messsysteme kleiner als die Dauer der zu erwartenden Buszyklen ist.

3 Vorbetrachtungen

Bevor ein Konzept zur Synchronisierung von Messstationen erstellt und das Gesamtsystem dann ausgewertet werden kann, müssen zuerst einige Vorbetrachtungen zum Zeitpunkt der Zeitstempelerfassung und der erreichbaren Synchronizität durch die Einzelkomponenten durchgeführt werden.

3.1.1 Verwendete Hardware und deren Einschränkungen

In Tabelle 3 wird die im Messaufbau verwendete Hardware aufgelistet.

Tabelle 3: Verwendete Hardware

Anzahl	Eingesetzte Hardware	Rolle
2	F+S efusA9	Zeitserver
2	Raspberry Pi 2 Model B	Client
2	Siemens SCALANCE X308-2M PoE	Switch
2	Phicomm FIR151B A2	Router
1	Phillips PM 5132	Funktionsgenerator
1	TTi QL335TP POWER SUPPLY	Spannungsversorgung

Zu beachten ist der Flaschenhals der Netzwerkkarte in den Raspberry Pi Plattformen. Es muss berücksichtigt werden, dass es sich bei den auf dem Board verbauten Ethernetports nicht um eine dedizierte Netzwerkschnittstelle handelt, sondern die Schnittstelle über USB implementiert ist [Ben12]. Dies führt zwangsläufig zu einer stärkeren Ausprägung der in Abschnitt 2.4.4 geschilderten Problematik der Zeitstempelerfassung, da hier mehrere Kernelschnittstellen überwindet werden müssen, um einen Zeitstempel zu erhalten.

3.1.2 Zeitstempel aus dem Kernel- oder Userspace

Bei Unterstützung von IEEE1588-2008 konformen Hardwaretimestamping in der Netzwerkkarte gibt es die Möglichkeit, direkt auf die Uhr in der Netzwerkkarte zuzugreifen und die in der Netzwerkkarte gültige, per Precision Time Protocol synchronisierte, Uhrzeit als Systemzeit zu übernehmen. Auf dementsprechende Messungen wird allerdings verzichtet, da die Netzwerkkartenuhr im normalen Anwendungsfall nicht benutzt wird und es sich bei dem später behandelten Versuchsaufbau um einen nicht hardwaretimestampfähigen Client handelt.

Durch die nicht hardwaretimestampfähige Netzwerkschnittstelle der Clients ist eine schlechtere Synchronizität der Systemzeiten zu erwarten, da hier die Delay-Mechanismen des Precision Time Protocols die Zeitstempel nun etwas verzögert erhalten und die Request- und Responsezeiten durch zusätzliche Wartezeiten in den Networkqueues asymmetrisch sein können. Die Implementierung der Synchronizitätsmessung als Kernelmodul soll sicherstellen, dass die schnellstmögliche Erfassung des auszuwertenden Zeitstempels auch bei nicht hardwareseitig PTP-fähigen Geräten vorliegt und zusätzliche Verzögerungen durch das Betriebssystem vermieden werden.

Um dies zu belegen, wurde ein Userspaceprogramm geschrieben, welches die identische Funktion des Kernelmoduls übernimmt, aber einen anderen Mehrzweckpin überwacht. Beide Pins des Gerätes wurden parallel an einen Funktionsgenerator angeschlossen und die so erhaltenen Zeitstempel verglichen. Abbildung 7 zeigt deutlich, dass eine nicht zu vernachlässige Verzögerungen durch das jeweilige Betriebssystem zustande kommt. Gerade beim Graph der efusA9-Plattform sieht man deutlich die periodischen Eingriffe des Betriebssystems in die Ablaufsteuerung. Umso weiter wir uns von der Hardware entfernen, desto mehr Probleme ergeben sich aus dem Einmischen des Betriebssystemschedulers, besonders da im Userspace nicht mehr sichergestellt ist, dass der von der Flanke ausgelöste Interrupt auch wirklich sofort verarbeitet wird. Deswegen ist eine Zeitstempelung im Kernel zu bevorzugen.

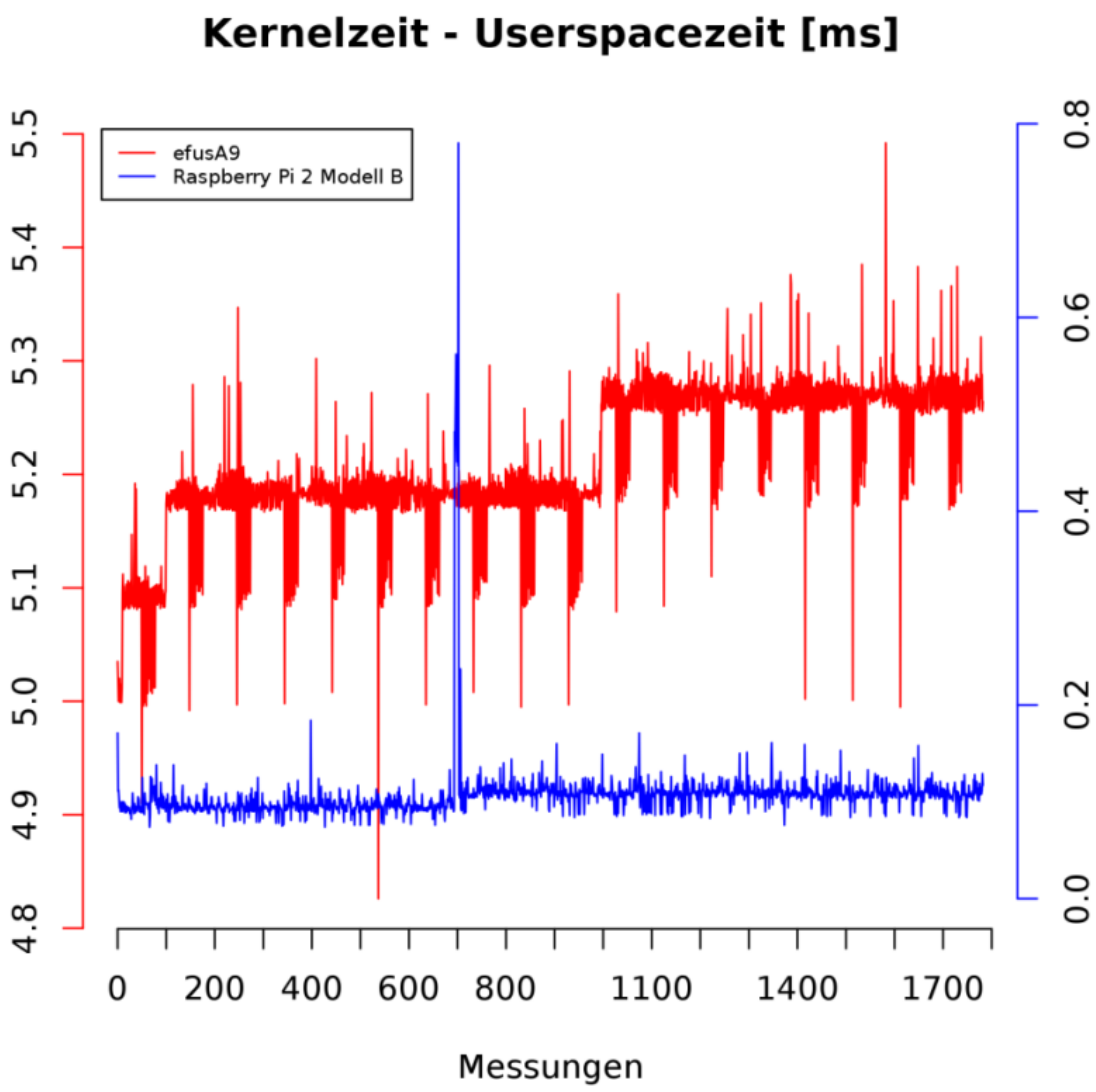

Abbildung 7: Zeitstempelerfassung im Kernel- oder Userspace

3.1.3 Synchronizität von NTP Stratum-1 Servern

Um die Synchronizität zweier Teilsysteme aussagekräftig vergleichen zu können, wurde zuvor die grundsätzlich erreichbare Synchronizität von NTP-Servern untersucht. Dazu wurden zwei Raspberry Pi Plattformen wie in Abbildung 8 dargestellt, mit GPS-Empfängern ausgestattet, sodass diese jeweils als Stratum-1 Zeitserver dienen konnten.

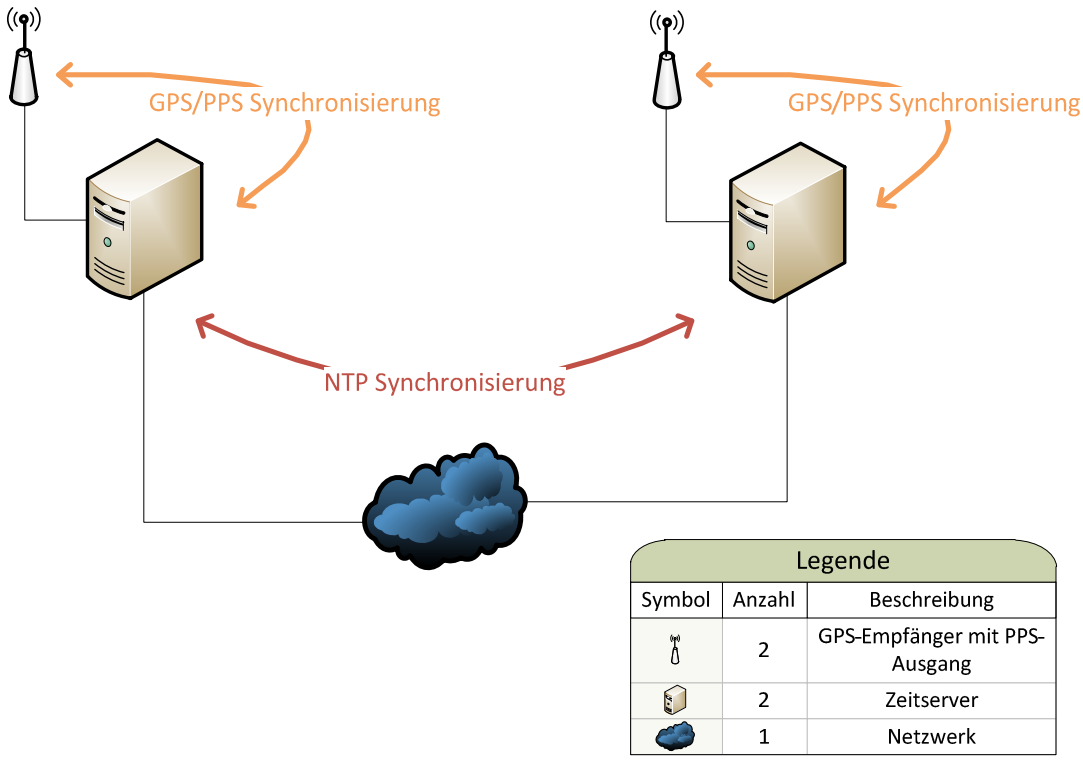

Abbildung 8: Vergleichsmessung NTP

Über die GPS-Empfänger wird sichergestellt, dass beide Raspberry Pi Plattformen sich auf die koordinierten Weltzeit (UTC) synchronisieren, auch wenn keine externen Zeitserver berücksichtigt werden. Es wurden wieder Zeitstempel über eine Interruptroutine erzeugt und verglichen. Im Unterschied zu den restlichen Messungen wurden diese Zeitstempel ausschließlich im Userspace generiert. Die Messungen wurden unter verschiedenen Parametern durchgeführt, u. a. ob eine gegenseitige Synchronisation über das NTP-Protokoll stattfinden kann und/oder öffentliche Zeitserver aus dem Internet mit hinzubezogen werden. Es wurden insgesamt drei Messreihen aufgenommen.

Auf Abbildung 9 ist dargestellt, wie sich die gemittelte Zeitdifferenz zwischen beiden Messstationen unter der Variation von konfigurierbaren Parametern verhält. Die variierten Parameter sind Tabelle 4 zu entnehmen.

Tabelle 4: Variierte Parameter der NTP-Vergleichsmessung

Parameter	Bedeutung
orphan	Verwendung des tos orphan 1 Parameter in der Konfiguration des NTP-Daemon
externe	Miteinbeziehung von NTP-Zeitservern aus dem Internet
sync	Miteinbeziehung der jeweils anderen Messstation als Zeitserver
rtp	Erzwingen einer Echtzeitpriorität für den Zeitstempelerfassungsprozess

Eine Messung wird durch identisch konfigurierte Parameter gekennzeichnet und die drei aufgenommenen Messreihen samt deren Durchschnittswert nebeneinander dargestellt. Messungen mit niedrigen absoluten Differenzen sind zu bevorzugen. Die variierten Parameter sowie die Messergebnisse sind in den Tabelle 5 bis 6 dargestellt. Es ist deutlich zu erkennen, dass die gewählten Konfigurationen einen starken Einfluss auf die erreichbare Synchronizität haben. Was ebenfalls ins Auge fällt sind die teilweise sehr starken Unterschiede zwischen den Messreihen, auch wenn die jeweiligen Messungen mit identischen Parametern konfiguriert wurden. Dies bedeutet, dass die maximal erreichbare Synchronizität über NTP alleine stark schwankt.

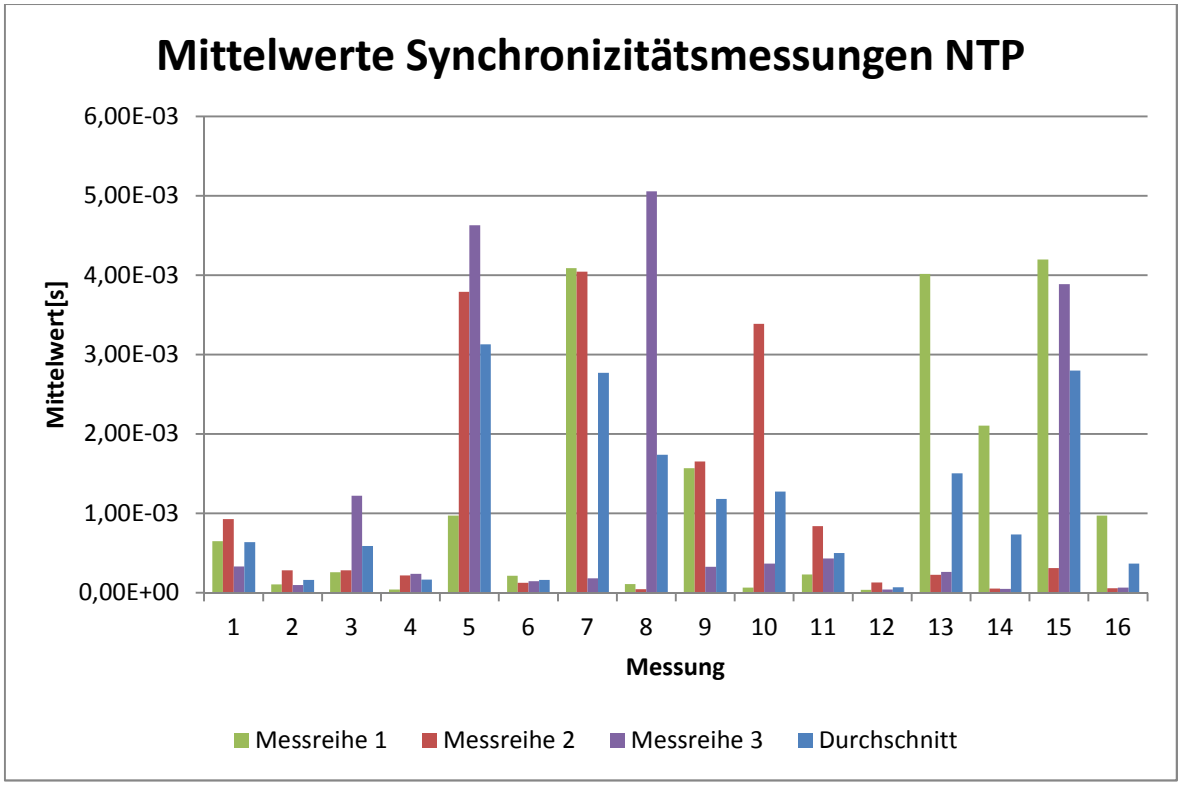

Abbildung 9: Auswertung Synchronizitätsmessung NTP

Die besten Eigenschaften wies Messung 12 auf. Tabelle 5 können wir entnehmen, dass diese Messreihe sich allein auf externe Zeitserver synchronisiert, weshalb diese Einstellung für den unter Abschnitt 5 geschilderten Versuchsaufbau nicht in Frage kommt. Die nächstbesten Ergebnisse lieferte Messung 6. Diese Messung erfolgte unter gegenseitiger Synchronisierung der beidem Raspberry Pi Plattformen (Parameter "sync"). Des Weiteren wurde das Erzwingen von Echtzeitpriorität unterlassen (Parameter "rtp") und keine externen Zeitserver hinzugeholt (Parameter "externe"). Der benutzte Parameter tos orphan 1 bedeutet, dass selbst bei Verlust der Echtzeitquelle, in diesem Fall PPS-diszipliniertes GPS, die Plattform weiter als Zeitserver dienen kann. Dies würde unter der Verwendung des lokalen Oszillators geschehen.

Tabelle 5: Messreihe 1 Synchronizität NTP

	#	Dauer	orphan	externe	sync	rtp	Mittelwert [s]	Standardabweichung
Messreihe 1	1	7:04	Ja	Ja	Ja	Ja	6,51E-04	1,30E-03
	2	2:30	Ja	Ja	Ja	Nein	1,05E-04	5,25E-04
	3	2:28	Ja	Ja	Nein	Ja	2,61E-04	1,22E-03
	4	2:28	Ja	Ja	Nein	Nein	4,14E-05	4,39E-04
	5	7:29	Ja	Nein	Ja	Ja	9,73E-04	1,81E-03
	6	2:20	Ja	Nein	Ja	Nein	2,13E-04	5,60E-04
	7	2:38	Ja	Nein	Nein	Ja	4,09E-03	2,78E-03
	8	2:52	Ja	Nein	Nein	Nein	1,12E-04	4,40E-04
	9	2:33	Nein	Ja	Ja	Ja	1,57E-03	2,66E-03
	10	2:25	Nein	Ja	Ja	Nein	6,50E-05	4,60E-04
	11	2:20	Nein	Ja	Nein	Ja	2,31E-04	1,17E-03
	12	2:23	Nein	Ja	Nein	Nein	3,80E-05	4,24E-04
	13	2:28	Nein	Nein	Ja	Ja	4,02E-03	3,29E-03
	14	2:27	Nein	Nein	Ja	Nein	2,10E-03	1,40E-03
	15	2:30	Nein	Nein	Nein	Ja	4,20E-03	3,10E-03
	16	2:25	Nein	Nein	Nein	Nein	9,72E-04	7,47E-04

Tabelle 6: Messreihe 2 Synchronizität NTP

	#	Dauer	orphan	externe	sync	rtp	Mittelwert [s]	Standardabweichung
Messreihe 2	1	1:57	Ja	Ja	Ja	Ja	9,27E-04	1,29E-03
	2	1:39	Ja	Ja	Ja	Nein	2,85E-04	1,24E-03
	3	1:43	Ja	Ja	Nein	Ja	2,85E-04	6,14E-04
	4	2:41	Ja	Ja	Nein	Nein	2,20E-04	3,33E-03
	5	2:28	Ja	Nein	Ja	Ja	3,79E-03	3,33E-03
	6	6:50	Ja	Nein	Ja	Nein	1,27E-04	4,81E-04
	7	2:40	Ja	Nein	Nein	Ja	4,04E-03	3,23E-03
	8	2:49	Ja	Nein	Nein	Nein	4,61E-05	4,40E-04
	9	1:55	Nein	Ja	Ja	Ja	1,66E-03	2,87E-03
	10	2:25	Nein	Ja	Ja	Nein	3,39E-03	1,77E-03
	11	2:20	Nein	Ja	Nein	Ja	8,41E-04	1,63E-03
	12	2:24	Nein	Ja	Nein	Nein	1,30E-04	4,46E-04
	13	2:29	Nein	Nein	Ja	Ja	2,26E-04	1,16E-03
	14	2:34	Nein	Nein	Ja	Nein	5,38E-05	4,34E-04
	15	3:19	Nein	Nein	Nein	Ja	3,12E-04	1,26E-03
	16	2:25	Nein	Nein	Nein	Nein	5,90E-05	4,40E-04

Tabelle 7: Messreihe 3 Synchronizität NTP

	#	Dauer	orphan	externe	sync	rtp	Mittelwert [s]	Standardabweichung
Messreihe 3	1	7:08	Ja	Ja	Ja	Ja	3,33E-04	1,17E-03
	2	2:52	Ja	Ja	Ja	Nein	9,97E-05	4,34E-04
	3	2:41	Ja	Ja	Nein	Ja	1,22E-03	1,26E-03
	4	2:18	Ja	Ja	Nein	Nein	2,37E-04	1,01E-03
	5	7:29	Ja	Nein	Ja	Ja	4,63E-03	3,15E-03
	6	2:20	Ja	Nein	Ja	Nein	1,46E-04	5,17E-04
	7	2:38	Ja	Nein	Nein	Ja	1,83E-04	1,01E-03
	8	2:52	Ja	Nein	Nein	Nein	5,06E-03	1,05E-03
	9	2:33	Nein	Ja	Ja	Ja	3,29E-04	1,18E-03
	10	2:27	Nein	Ja	Ja	Nein	3,70E-04	8,45E-04
	11	2:21	Nein	Ja	Nein	Ja	4,30E-04	1,22E-03
	12	2:25	Nein	Ja	Nein	Nein	4,20E-05	4,45E-04
	13	2:45	Nein	Nein	Ja	Ja	2,64E-04	1,17E-03
	14	2:34	Nein	Nein	Ja	Nein	4,79E-05	4,46E-04
	15	2:22	Nein	Nein	Nein	Ja	3,89E-03	3,31E-03
	16	3:00	Nein	Nein	Nein	Nein	6,71E-05	4,47E-04

Aus den Messwerten lässt sich entnehmen, dass durch eine Synchronisierung über das Network Time Protocol eine Zeitabweichung im unteren Millisekundenbereich erreichen lässt.

3.1.4 Synchronizität von per Precision Time Protocol synchronisierten Geräten

Um die über PTP erreichte Synchronizität von zwei Geräten in einem lokalen Netz zu bewerten, wurde diese gemessen. Dafür wurden ein als Grandmaster Clock konfiguriertes efusA9 Board über einen PTP-fähigen Switch mit einem Client verbunden. Als Client dienten entweder ein Raspberry Pi 2 Modell B oder ein zweites efusA9 Board. Die variierten Parameter sind Tabelle 8 zu entnehmen.

Tabelle 8: Parameterliste Synchronizitätsmessung PTP

Parameter	Bedeutung	
HW_TS	Verwendung des Hardwaretimestamps der Netzwerkschnittstelle	
PTPswitch	Verwendung und Konfiguration eines PTP-fähigen Switches	
	nein	Kein PTP-fähiger Switch verwendet
	aus	PTP-fähiger Switch, aber nicht als solcher konfiguriert
	e2e	PTP-Switch als Transparent Clock mit End-to-end-Mechanismus konfiguriert
Client	Das als Client verwendete Endgerät. Zu beachten ist das Fehlen der Hardwaretimestampingfähigkeit auf der Raspberry	
Mechanismus	In dem PTP-Geräten konfigurierter Delay-Mechanismus	

Die Ergebnisse sind Tabelle 9 zu entnehmen. Es ist eine deutliche Verbesserung gegenüber einer Synchronisierung über das Network Time Protocol zu erkennen. Ebenfalls fällt auf, das die Qualität der eingesetzten Netzwerkkomponenten einen nicht zu vernachlässigen Einfluss auf die erreichbare Synchronizität hat: Bei Betrachtung der Messungen Nr. 2 und 7 fällt auf, dass sich alleine durch den Einsatz optimierter Hardware eine Verbesserung der Synchronizität um ein Vielfaches erreichen lässt.

Tabelle 9: Messwerte Synchronisierung über PTP

Nr.	Dauer	HW_TS	PTPswitch	Client	Mechanismus	Mittelwert	Standardabweichung
1	00:20	ja	nein	efusA9	Auto	2,59E-06	3,71E-06
2	00:52	nein	nein	efusA9	Auto	-5,24E-03	6,15E-02
3	00:27	ja	nein	efusA9	peer to peer	3,35E-06	1,09E-05
4	00:18	ja	e2e	efusA9	Auto	2,35E-06	2,83E-06
5	01:06	ja	e2e	efusA9	Auto	2,38E-06	1,76E-05
6	00:29	nein	e2e	efusA9	Auto	2,06E-05	1,06E-04
7	00:44	nein	aus	efusA9	Auto	2,95E-06	1,42E-05
8	01:36	ja	aus	efusA9	Auto	-4,66E-07	4,96E-06
9	00:51	nein	e2e	Raspberry	Auto	-1,50E-06	3,21E-06
10	01:45	nein	e2e	Raspberry	Auto	-4,31E-06	5,05E-06
11	01:50	nein	e2e	efusA9	Auto	-2,93E-06	9,20E-06
12	01:23	ja	e2e	efusA9	end to end	2,29E-06	2,96E-06

4 Erstellung eines Konzeptes

Während in lokalen Netzen die volle Kontrolle über alle eingesetzten Netzwerkkomponenten ausgeübt werden kann, muss bei der Betrachtung von verteilten Messsystemen davon ausgegangen werden, dass diese Kontrolle nicht über alle Netzwerkpfade hinweg ausgeübt werden kann. Wenn ein WAN als Kommunikationsmedium über große Distanzen benutzt wird, entfällt diese Kontrolle komplett. In solch großen und vielfach verzweigten Netzen verfällt der Vorteil des Precision Time Protocols, weil nicht mehr sicherstellen werden kann, dass die Signallaufzeitverzögerung auf dem Hin- und Rückpfad identisch ist. Da für die Realisierung des Precision Time Protocols meist auf eine UDP Multicast Implementierung zurückgegriffen wird, kommen zusätzlich die Probleme einer verbindungslosen Kommunikation in verzweigten Netzen wie Paketverlust oder das Eintreffen der Pakete in falscher Reihenfolge hinzu. Dies hat die Folge, dass auf andere Lösungsansätze zurückgegriffen werden muss. Hier greifen die Vorteile des Network Time Protocol, da nicht die Hardwaretimestampingfähigkeit auf dem gesamten Pfad gefordert ist. Zusätzlich kann auf weitere Zeitquellen aus dem internationalen Network Time Protocol Serverpool zurückgegriffen werden, um die Zeitbasis weiter zu stabilisieren.

Um die Synchronizität verteilter Messsysteme dann anhand der in Abschnitt 2.6 untersuchten Anforderungsklassen zu beurteilen, muss zusätzlich ein geeignetes Messkonzept entwickelt werden.

4.1 Grundsätzlicher Aufbau

Es werden zwei Netze aufgebaut. In jedem befindet sich jeweils ein GPS-gestützter NTP-Zeitserver. Der GPS-Empfänger ist zusätzlich in der Lage, ein Pulse-per-second-Signal zu erzeugen, welcher von dem Zeitserver benutzt wird, um die Zeitauflösung und den Jitter zu reduzieren. Jeder Zeitserver dient in seinem Netz als PTP Grandmaster Clock. Neben dem Zeitserver befinden sich Clients und als Transparent Clock konfigurierbare Switches in den jeweiligen Netzen. Diese Netze sind über ihre jeweils eigenen Router miteinander verbunden.

Den NTP-Zeitservern dienen die PPS-gestützten GPS-Empfänger als Stratum-0-Zeitquelle. Die beiden so entstandenen Stratum-1-NTP-Zeitserver synchronisieren sich gegenseitig über ihre Netzgrenzen hinweg mittels des Network Time Protocols. Die Grandmaster Clock nutzt die über das Network Time Protocol synchronisierte Systemzeit als PTP-Quelle und verteilt diese Zeit per Multicast in ihrem Netz an die Clients.

Durch die Kombination von GPS-gestütztem NTP und PTP soll die größtmögliche Synchronizität über eine unbekannte Netzwerklandschaft hinweg sichergestellt werden. Die Vorteile des Precision Time Protocols werden im lokalen Netz ausgenutzt um Clients und die Grandmaster Clock synchron zu halten. Da eine Synchronisation über den GPS-Empfänger für die maximale Synchronizität nicht ausreicht (vgl. Abschnitt 3.1.3), halten sich die Grandmaster Clocks der Teilsysteme sich mittels des Network Time Protocols über Netzübergänge hinweg möglichst synchron. Dieses Prinzip ist in Abbildung 10 dargestellt.

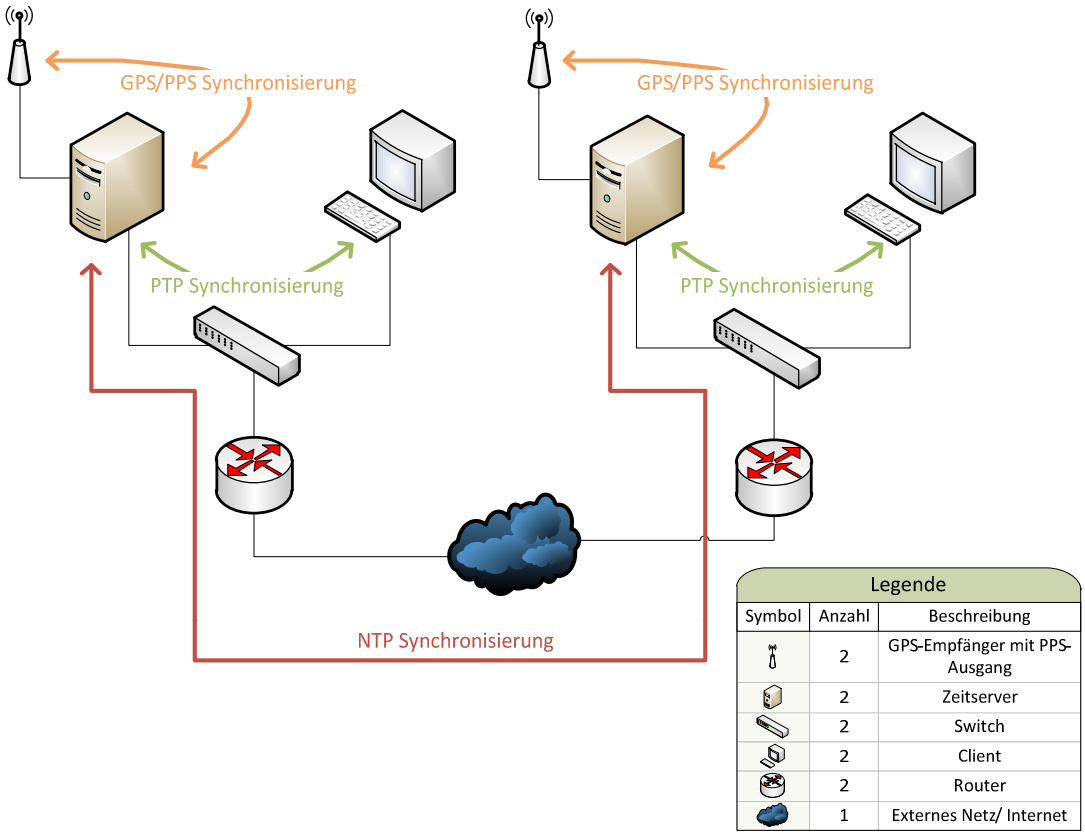

Abbildung 10: Synchronisierung über NTP-PTP-Kombination

4.2 Messprinzip

Um die Synchronizität der beiden Teilsysteme zu vergleichen, werden kernelnah Zeitstempel erzeugt, die am Ende der Messreihe ausgewertet werden können. Zur Erfassung der Zeitstempel wurde ein Kernelmodul geschrieben, welches einen von außen zugänglichen Multifunktionspin der Hardwareplattformen überwacht. Eine positive Flanke an diesem Pin löst eine Interruptroutine aus, welche die aktuelle Kernelzeit erfasst und in das Kernellogfile schreibt. Die Kernelzeit kann entweder als monoton steigender Sekundenzähler seit Systemstart oder als klassische Uhrzeit festgehalten werden. In diesem Messaufbau wurde die Kernelzeit als Uhrzeit aufgenommen. Diese Logfiles können dann über ein weiteres, selbstgeschriebenes Programm ausgewertet und verglichen werden.

Der Vergleich der beiden Logfiles und die Berechnung der Differenz zwischen den beiden Systemzeiten erfolgt, indem beide Logfiles Zeile für Zeile durchgegangen werden und die eingetragenen Werte voneinander subtrahiert werden. Sollte die

ermittelte Differenz größer sein als ein vorher festgelegter Schwellwert, wird der kleinere der Logfilewerte verworfen und die nächste Berechnung mit einem neuen Logeintrag angestoßen. Diese Vorauswahl wurde nötig, weil nicht immer alle Flanken zuverlässig von allen Geräten registriert werden, sodass es zu falschen Korrelationen kommen kann. Als Schwellwert ist hier eine Zeit etwas geringer als die erwartete Wartezeit zwischen zwei positiven Flanken am überwachten Multifunktionspin zu empfehlen. So werden alle Zeitstempelpaare verworfen, bei denen nicht klar feststeht, dass sie von der gleichen Flanke ausgelöst wurden.

5 Inbetriebnahme eines Messaufbaus

Das in Abschnitt 4.1 geschilderte Konzept gilt es nun hinsichtlich der erreichbaren Synchronizität zu untersuchen. Dafür muss ein geeigneter Versuchsaufbauaufbau erstellt werden.

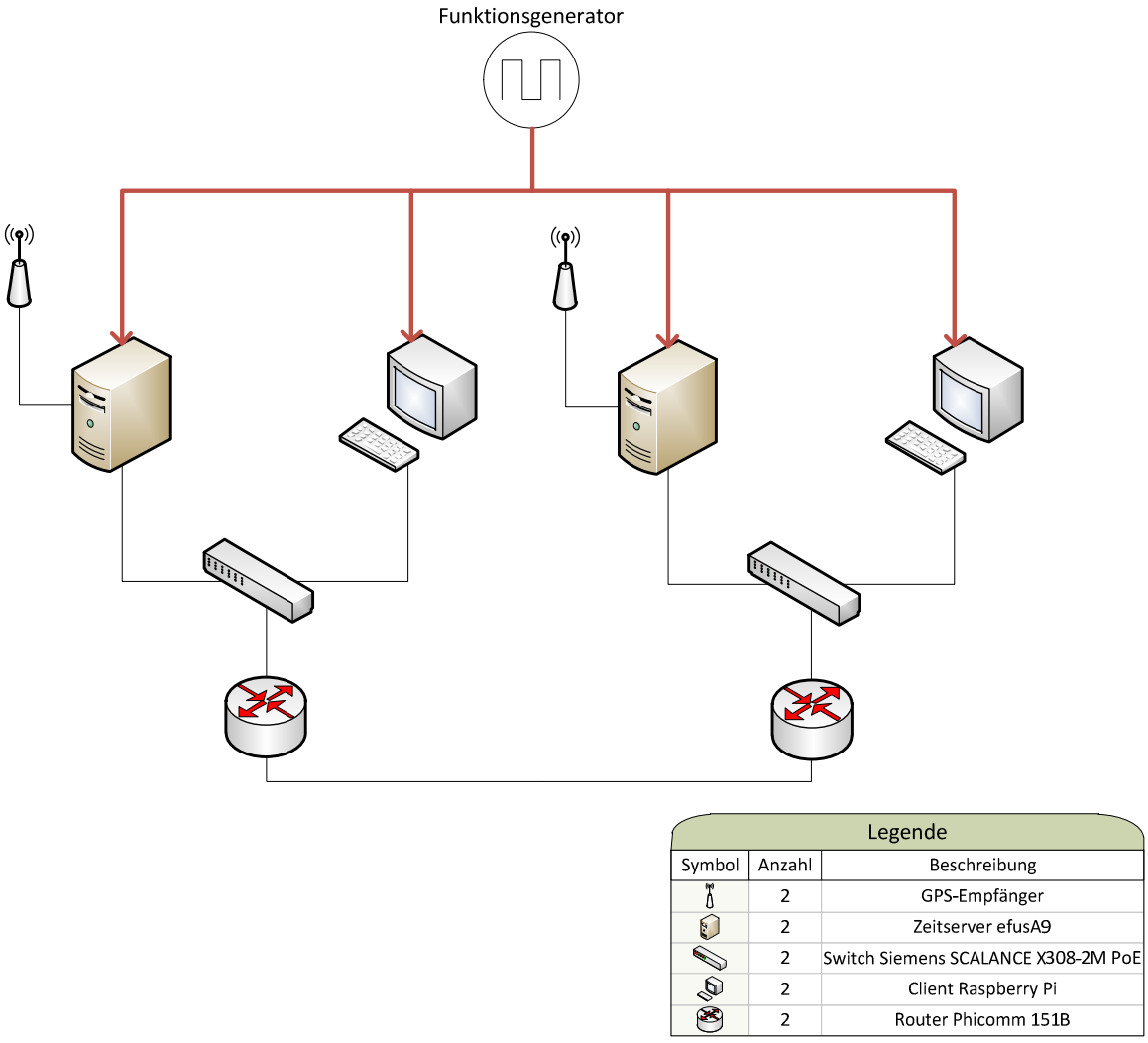

Abbildung 11: Versuchsaufbau zur kombinierten Messung NTP/PTP

Es wurden zwei identische Teilnetze aufgebaut, bestehend aus jeweils einem efusA9 Board mit GPS-Empfänger, einem Raspberry Pi Version 2 Model B und einem Siemens SCALANCE X308-2M PoE Switch aufgebaut. Auf beiden embedded-Plattformen lief ein Linux Betriebssystem ohne Realtime-Erweiterungen. Diese Netze wurden über jeweils einen Phicomm FIR151B A2 Router miteinander verbunden.

Die im Kernelmodul spezifizieren Mehrzweckpins aller Endgeräte wurden parallel an einem Phillips PM 5132 Funktionsgenerator angeschlossen und die in Abschnitt 4.2 erläuterten Kernelmodule zur Zeitstempelung geladen. Der Funktionsgenerator erzeugte positive Signalflanken im Abstand von ca. 1 s.

Die Precision Time Protocol Implementierung ptp4l wurde auf allen Geräten installiert und auf Softwarezeitstempelung und den end-to-end Mechanismus konfiguriert. Diese Einstellungen folgen aus dem Fehlen einer hardwareseitig PTP-fähigen Netzwerkschnittstelle auf den Raspberry Pi Plattformen. Die Switches wurden passend dazu als Transparent Clocks mit end-to-end-Mechanismus konfiguriert.

Auf den efusA9-Plattformen, welche neben ihrer Aufgabe als Grandmaster Clock auch als NTP-Zeitserver dienen sollen, wurde der bereits installierte Dienst ntpd so konfiguriert, dass die GPS-Empfänger als Zeitquelle dienen und eine zusätzliche Frequenzregelung über den PPS-Ausgang des GPS-Empfängers stattfindet.. Ntpd wurde auf den Raspberry Pi-Plattformen abgeschaltet, sodass die Grandmaster Clock als einzige Zeitquelle fungiert.

Der Versuch wurde bei Raumtemperatur durchgeführt.

6 Messergebnisse

Aus den aufgenommenen Messwerten wurden im ersten Schritt Kurven generiert. In diesen Diagrammen ist die Zeitdifferenz zweier Geräte über den aufgenommenen Messpunkten zu erkennen, wobei über die Messpunkte mittels der Frequenz des Funktionsgenerators von ca. 1 Hz auf die seit Versuchsbeginn vergangene Zeit in Sekunden geschlossen werden kann. In Abbildung 12 sind exemplarisch die beiden Kurven für die Zeitdifferenz jeweils zwischen den beiden Raspberry Pi-Plattformen als Clients und den beiden efusA9-Plattformen als Server dargestellt. Auf den Kurvenverläufen sieht man deutlich das übliche Synchronisierungsverhalten des Network Time Protocols nach einem Kaltstart, d.h. das Einregeln der Systemzeit auf die GPS-Zeit, wenn zu Beginn große Abweichungen herrschen. Dies ist zum Beispiel üblicherweise beim ersten Einschalten des Systems zu Tagesbeginn zu beobachten, so wie es hier der Fall war. Die benötigte Einregelungszeit variiert in der Regel etwas, liegt aber normalerweise im Bereich von einer halben bis anderthalb Stunden.

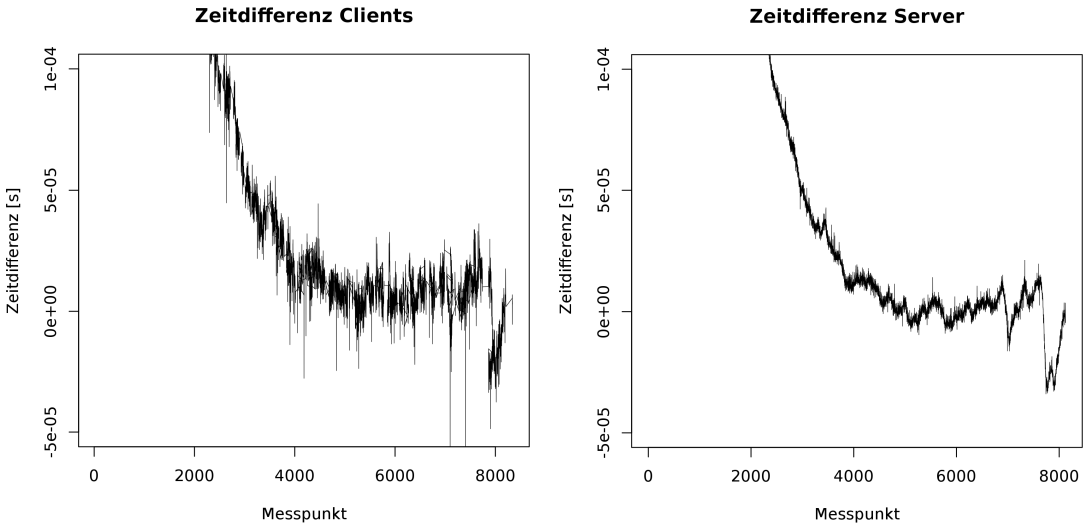

Abbildung 12: Zeitdifferenzen zwischen Clients und Servern nach Einschalten

Im zweiten Schritt wurden die Zeitdifferenzen aller Kurven gemittelt und deren Standartabweichung ermittelt. In Tabelle 10 werden die so ermittelten Werte dargestellt.

Die Messungen mit dem Bezeichner "Netz" stehen für die Synchronizität von Grandmaster Clock und Slave im jeweiligen Subnetz. Sie kann als Anhaltspunkt für die über das Precision Time Protocol erreichbare Synchronizität in den Subnetzen herangezogen werden. Für diese Messung wurde der Zeitstempel des Zeitservers vom Zeitstempel des Clients subtrahiert.

Tabelle 10: Messwerte Kombination NTP/PTP

Messung	1	2	3
Mittelwert Zeitdifferenz der Geräte in Netz 1	-2,46E-06	1,73E-05	-1,90E-06
Standardabweichung Zeitdifferenz der Geräte in Netz 1	3,51E-06	1,07E-04	6,64E-06
Mittelwert Zeitdifferenz der Geräte in Netz 2	-8,12E-06	-1,62E-05	-9,02E-06
Standardabweichung Zeitdifferenz der Geräte in Netz 2	1,27E-05	5,92E-05	7,80E-06
Mittelwert Zeitdifferenz der Clients	1,73E-04	7,89E-05	6,73E-06
Standardabweichung Zeitdifferenz der Clients	3,28E-04	1,77E-04	6,99E-06
Mittelwert Zeitdifferenz der Server	1,72E-04	7,25E-05	6,73E-06
Standardabweichung Zeitdifferenz der Server	3,40E-04	1,77E-04	6,99E-06
Messung unter Serverlast	4	5	6
Mittelwert Zeitdifferenz der Geräte in Netz 1	-9,51E-07	1,29E-06	1,68E-06
Standardabweichung Zeitdifferenz der Geräte in Netz 1	2,99E-06	6,30E-06	7,34E-06
Mittelwert Zeitdifferenz der Geräte in Netz 2	-9,29E-06	-6,84E-06	-7,61E-06
Standardabweichung Zeitdifferenz der Geräte in Netz 2	1,37E-05	7,94E-06	5,42E-06
Mittelwert Zeitdifferenz der Clients	6,27E-04	1,55E-04	2,52E-05
Standardabweichung Zeitdifferenz der Clients	2,22E-04	6,36E-05	1,78E-05
Mittelwert Zeitdifferenz der Server	3,20E-04	1,46E-04	1,59E-05
Standardabweichung Zeitdifferenz der Server	3,09E-06	6,39E-05	1,13E-05
Messung unter Clientlast	7	8	9
Mittelwert Zeitdifferenz der Geräte in Netz 1	-4,06E-06	-1,47E-05	-9,51E-06
Standardabweichung Zeitdifferenz der Geräte in Netz 1	2,23E-05	1,44E-04	1,86E-05
Mittelwert Zeitdifferenz der Geräte in Netz 2	-7,97E-06	-9,37E-06	-6,09E-06
Standardabweichung Zeitdifferenz der Geräte in Netz 2	1,94E-05	2,17E-04	1,88E-04
Mittelwert Zeitdifferenz der Clients	1,58E-05	9,09E-06	-1,06E-05
Standardabweichung Zeitdifferenz der Clients	2,33E-05	5,16E-05	1,88E-04
Mittelwert Zeitdifferenz der Server	6,91E-06	8,38E-06	-4,70E-06
Standardabweichung Zeitdifferenz der Server	2,21E-04	6,32E-06	7,65E-05

Die Messungen mit dem Bezeichner "Clients" stehen für Synchronizität der beiden Raspberry Pi Boards. Hier handelt es sich um die zu betrachtende erreichbare Synchronizität zwischen den beiden Messsystemen.

Die Messungen mit dem Bezeichner "Server" stehen für die Synchronizität der beiden efusA9 Boards. Diese Messungen können als Bezug zur erreichbaren Synchronizität zwischen den Netzen über das GPS-gestützte Network Time Protocol betrachtet werden.

Bei den Messungen 1 handelt es sich um Messungen zu Tagesbeginn, d.h. die alle auf dem System befindlichen Zeitbasen hatten zu Systemstart große Abweichungen zur GPS-Zeit. Der NTP-Daemon der beiden Zeitserver musste also die Systemzeiten erst einregeln. Messung 3 erfolgte direkt im Anschluss an Messung 2.

Um ein leichteren Überblick über die erhaltenen Messwerte zu geben, wurden diese nochmal in Abbildung 13, Abbildung 16 und Abbildung 17 als Diagramm dargestellt. Die Säulenhöhe stellt dabei den gemessenen Mittelwert der Abweichungen dar. Da es sich bei den Messungen 1 und 2 um noch nicht eingeregelte Messungen handelt, ist auf die nicht besonders aussagekräftige Darstellung der Standardabweichung in Abbildung 13 verzichtet worden. Das Säulendiagramm für den eingeschwungenen Zustand in Messung 3 wird in Abbildung 14 zusätzlich mit dieser Information dargestellt.

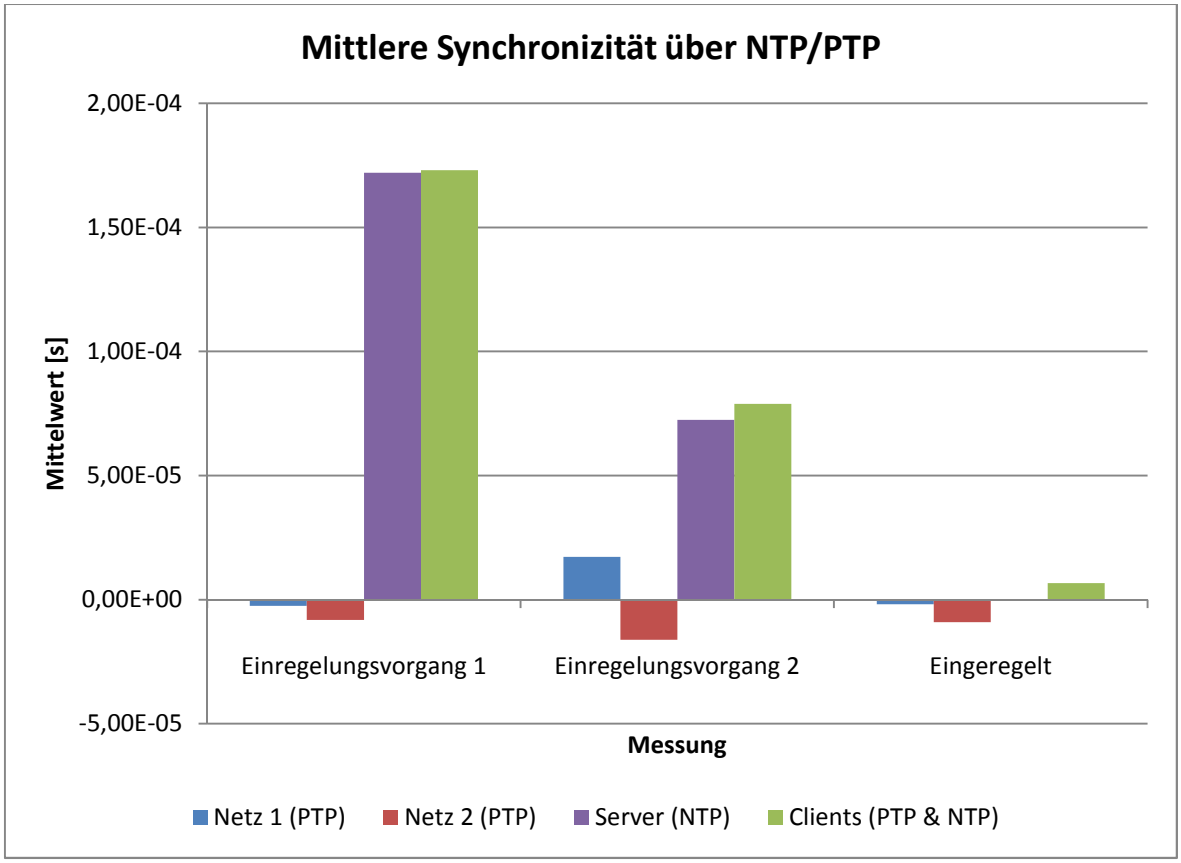

Abbildung 13: Messwerte mittlere Synchronizität mittels der Kombination von NTP und PTP

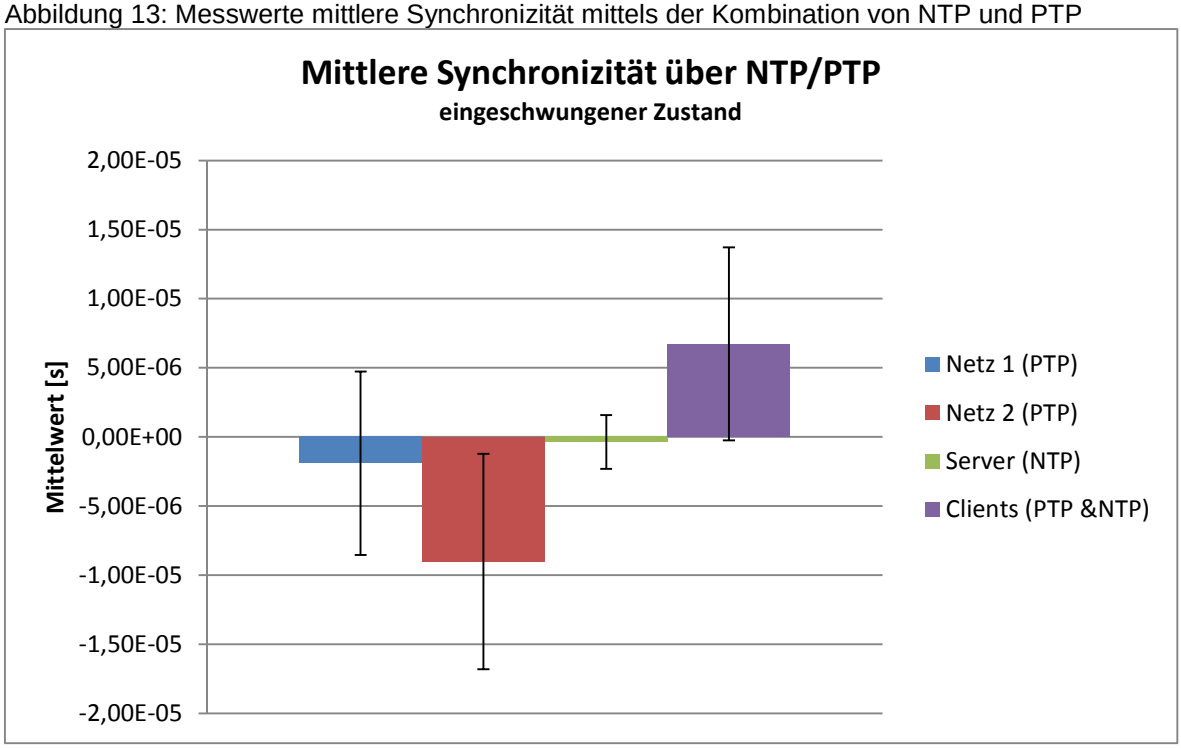

Abbildung 14: Messwerte mittlere Synchronizität NTP/PTP, eingeschwungener Zustand

Um für eine spätere Betrachtung Netzwerklast zu simulieren, wurde für die restlichen Messungen ein weiterer Raspberry Pi in Netz 1 eingefügt, welcher ununterbrochen Pakete an die vom Precision Time Protocol benutzen UDP Ports sendet. Dabei handelt es sich um die Ports 319/UDP für Sync-Messages und Delay-Request sowie Port 320/UDP für die Announce- und Follow-Up-Nachrichten sowie die Delay Responses.

Für die Messungen 4, 5 und 6 wurde der Zeitserver mit UDP Paketen belastet, wobei bei Messung 4 sowohl Port 319/UDP als auch 320/UDP angesprochen wurden, bei Messung 5 jeweils nur Port 320/UDP und bei Messung 6 nur Port 319/UDP.

Analog dazu wurde bei den restlichen Messungen der andere Client unter Last gestellt. Wieder wurde in aufsteigender Reihenfolge beide Ports, nur Port 320/UDP und dann nur Port 319/UDP angesprochen.

Der modifizierte Versuchsaufbau ist in Abbildung 15 dargestellt.

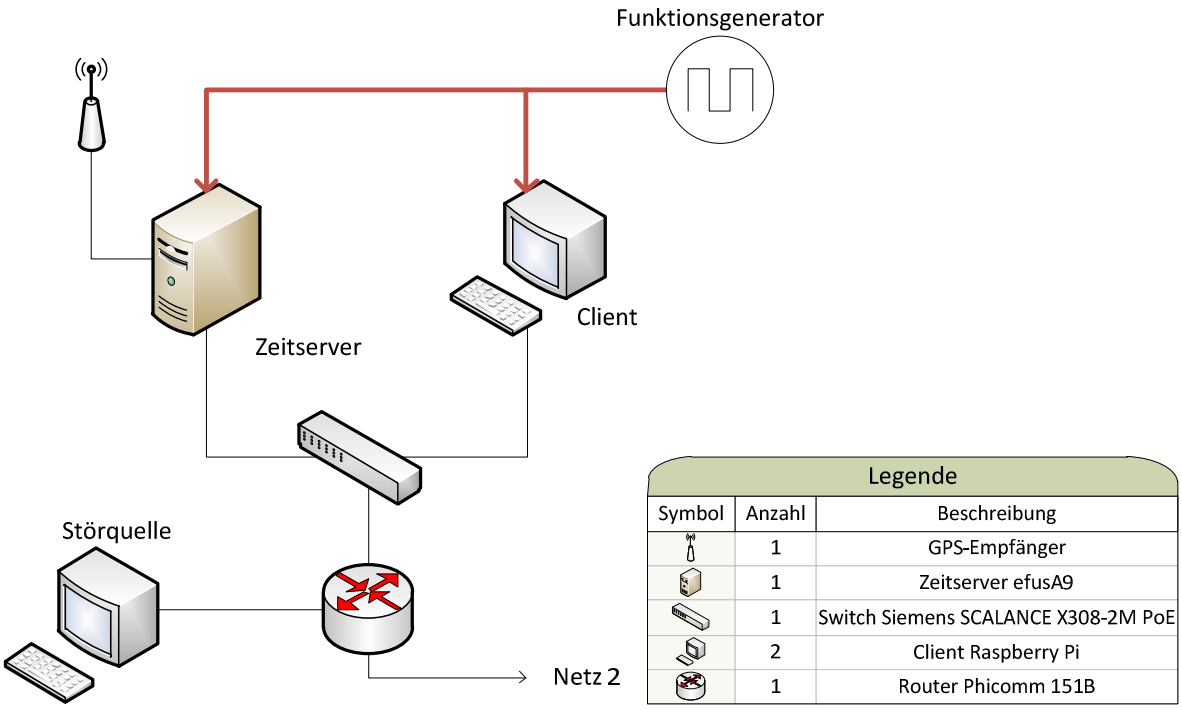

Abbildung 15: Modifikation des Versuchsaufbaus zur Lastsimulation

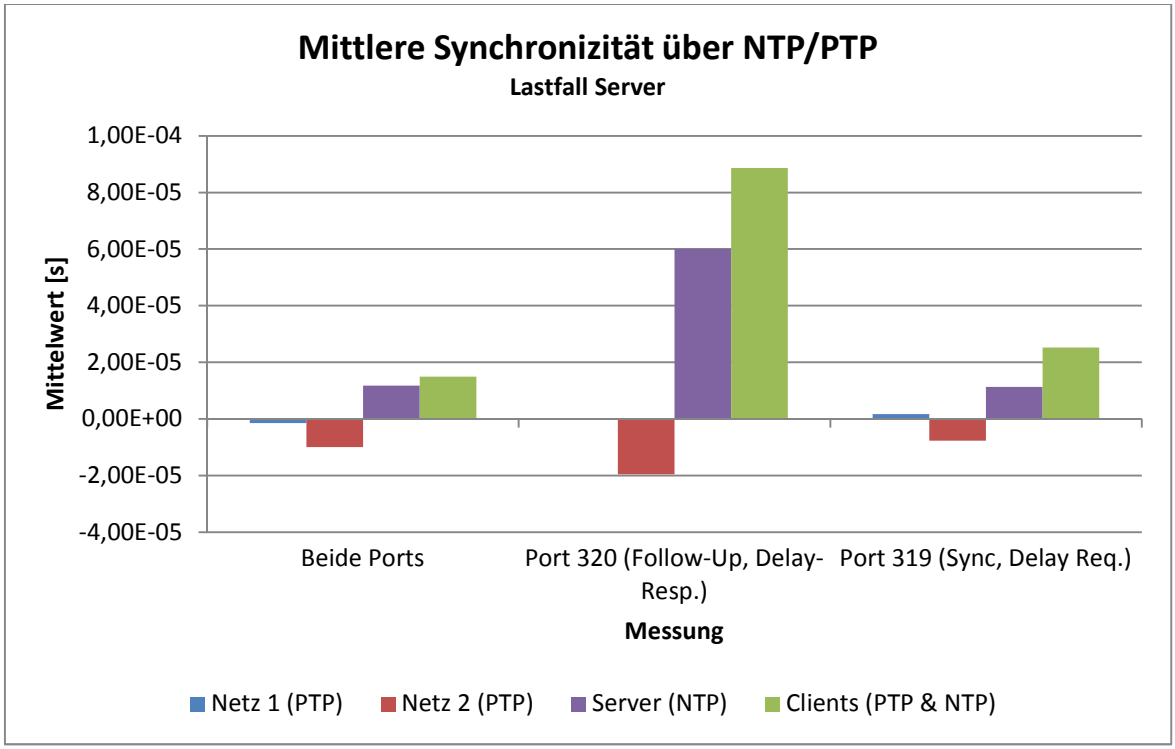

Abbildung 16: Messwerte mittlere Synchronizität mittels der Kombination von NTP und PTP, UDP-Flooding auf Server

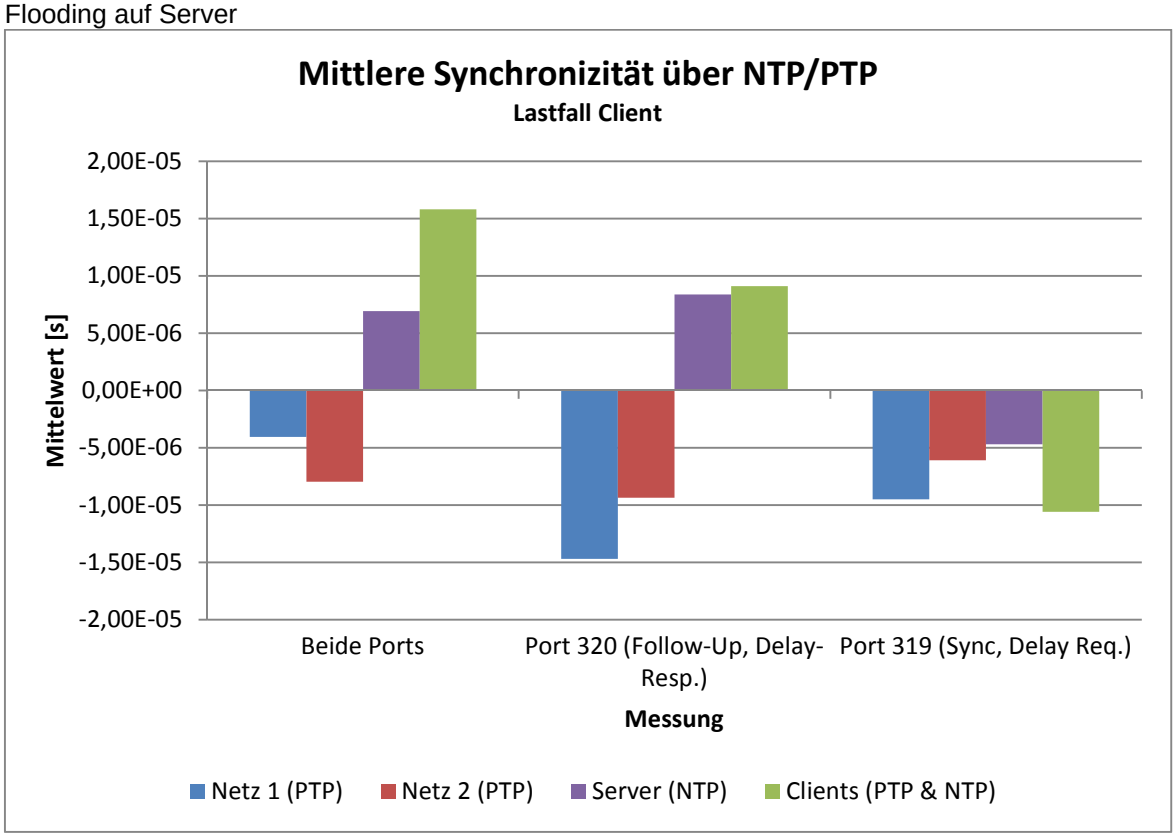

Abbildung 17: Messwerte mittlere Synchronizität mittels der Kombination von NTP und PTP, UDP-Flooding auf Client

7 Auswertungen

Wenn man nun die Messungen der Clients untereinander sowie die Messung der Server untereinander vergleicht, fällt ein makroskopisch ähnlicher Kurvenverlauf auf, der jedoch deutlich verrauscht ist. Zur Wiederholung sind in Abbildung 18 nochmal die Kurvenverläufe der Zeitdifferenzen der Clients und Server der Messung 1dargestellt.

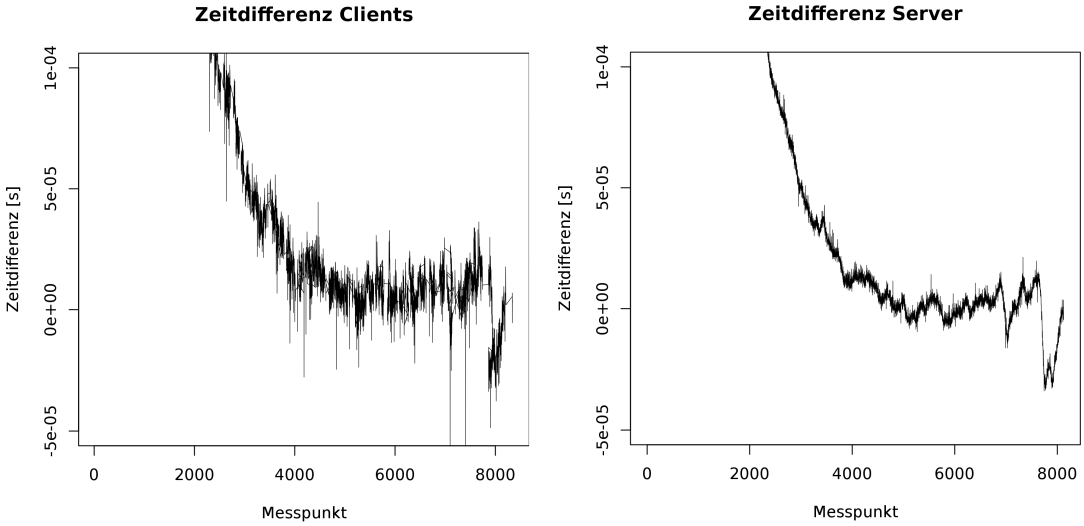

Abbildung 18: Zeitdifferenzen zwischen Clients und Servern nach Einschalten, Messung 1

Die Vermutung liegt nahe, dass der Grund für dieses Rauschen die durch das Precision Time Protocol verursachten Abweichungen zwischen Client und Zeitserver in den Teilnetzen sein könnten. Zur Überprüfung wurden die Zeitdifferenzen zwischen den Zeitservern und den Clients beider Teilnetze aufaddiert. Dann wurden die in Abbildung 18 dargestellten Kurven voneinander subtrahiert, um so den Kommunikationsanteil der Server herauszufiltern. Bei einer Gegenüberstellung des durch die Subtraktion der Kurven isolierten Rauschanteils mit der Summe der Abweichungen von der jeweiligen Grandmaster Clock, dargestellt in Abbildung 19, fällt ein Offset besonders im Bereich des Einregelungsvorganges auf. Wenn wir uns die beiden Häufigkeitsverteilungen in Abbildung 20 ansehen, erkennen wir diesen Offset nun genauer. Aus den Messwerten lässt sich ein Offset der Mediane von 6,4 µs entnehmen.

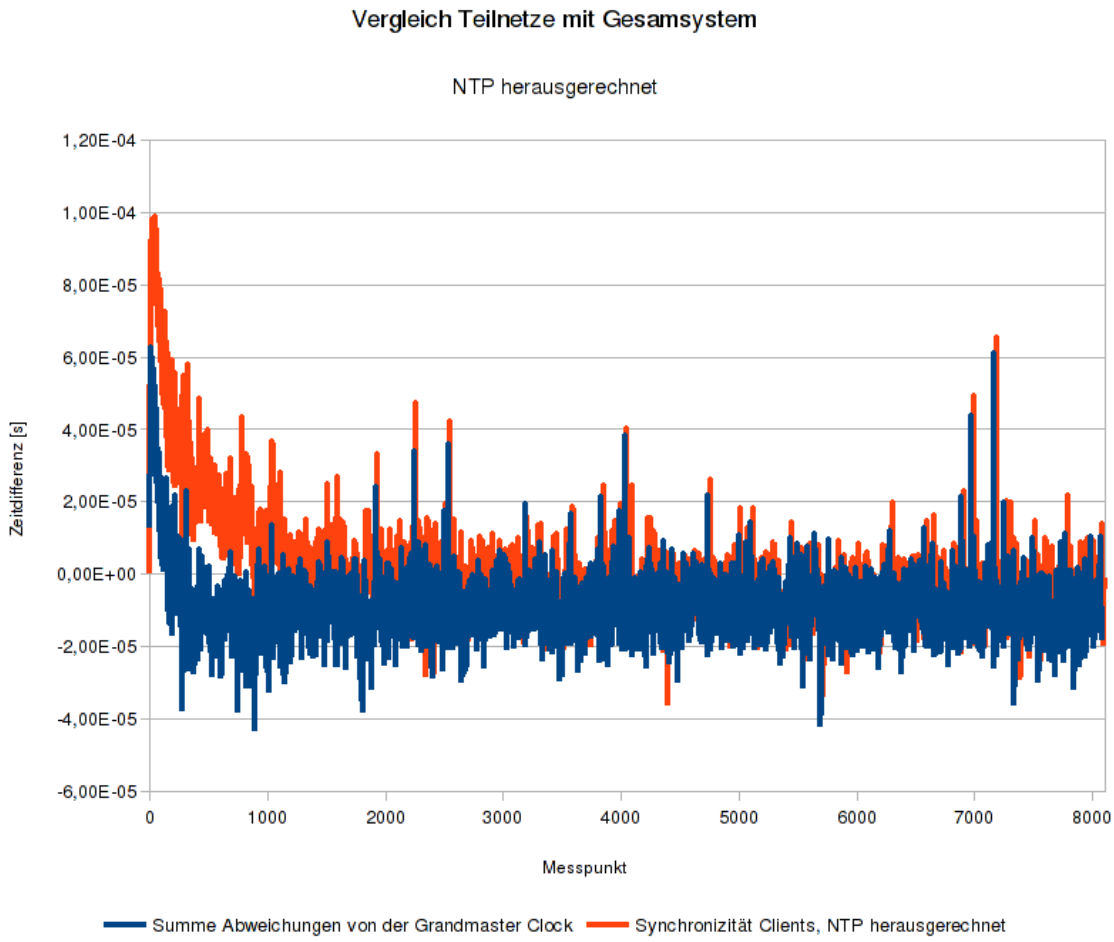

Abbildung 19: Gegenüberstellung Abweichungen in Teilsystemen und Gesamtsystem ohne NTP

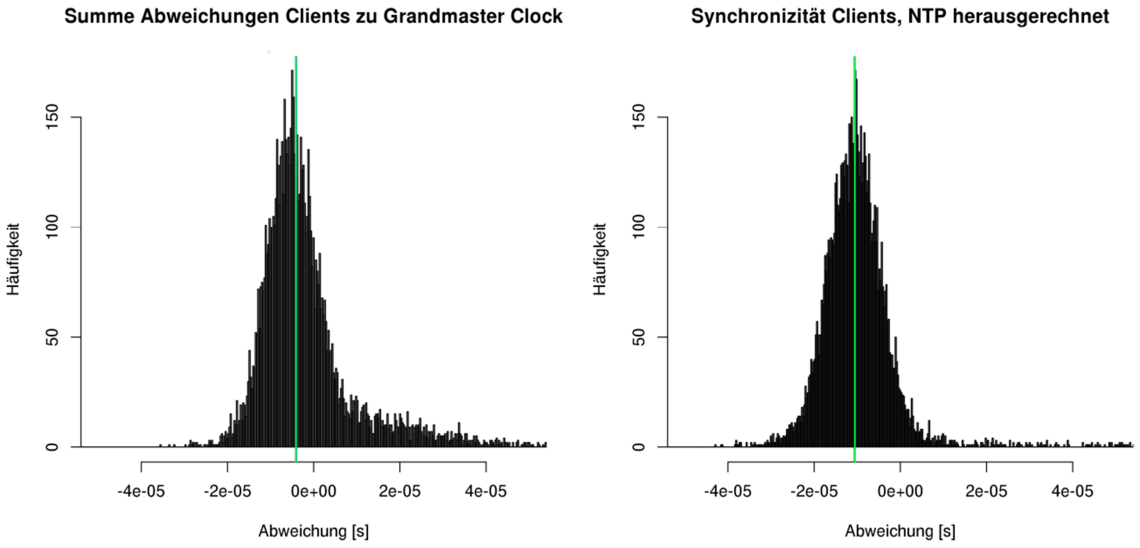

Abbildung 20: Vergleich der Häufigkeitsverteilungen in den Teilsystemen mit dem Gesamtsystem, ohne NTP (Median grün eingezeichnet)

Der Grund dieses Offsets lässt sich in diesem Messaufbau schlecht herleiten. Es ist anzunehmen, dass es sich um in den Netzwerkqueues sowohl der Router als auch der Netzwerkschnittstellen der Server und Clients entstandene Verzögerungszeit handelt.

Im weiteren ist allerdings eine deutliche Verbesserung der Synchronizität gegenüber den in Abschnitt 3.1.3 untersuchten, alleine über das Network Time Protocol synchronisierten, Stratum-1-Servern zu erkennen. Dazu dient Abbildung 21 als Verdeutlichung: hier wurden die vorher durchgeführten Messreihen in Abschnitt 3.1.3 gemittelt und als Vergleichswert hinzugezogen. Dieses Ergebnis ist allerdings kritisch zu betrachten, denn in dem hier durchgeführten Versuchsaufbau waren keine zusätzlichen Teilnehmer oder weitere Netze vorhanden, während sich in der Vergleichsmessung zwischen den Synchronisierungspartnern ein weiteres Netz befand.

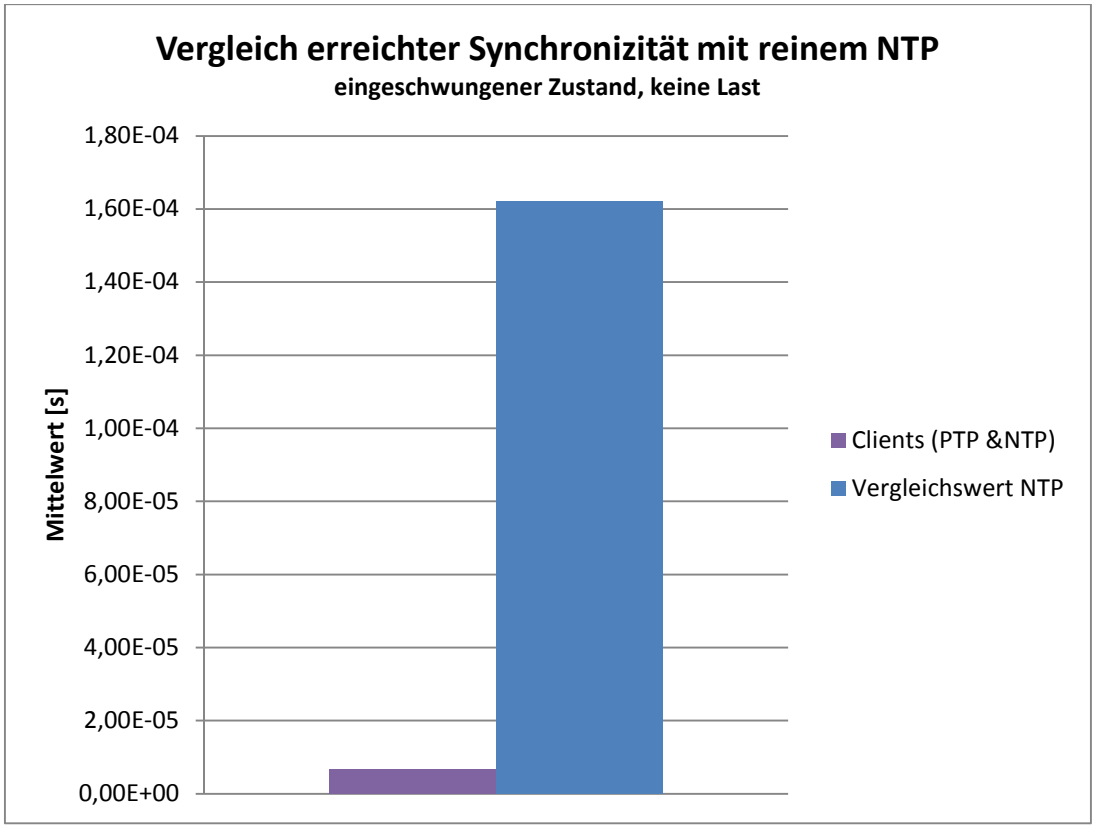

Abbildung 21: Vergleich erreichter Synchronizität mit dem in Abschnitt 3.1.3 gemessenen reinen NTP

In den vorgenommenen Messungen wurde eine Synchronizitäts von wenigen µs erreicht, es konnte aber keine zuverlässige Synchronizität im ns-Bereich sichergestellt werden. Bemerkenswert ist ebenfalls die erreichte Synchronizität zwischen den Zeitservern. Bei einer Gegenüberstellung der der Synchronizität der Server untereinander und der Clients untereinander, wie sie in Abbildung 22 vorgenommen wurde, wird deutlich, dass der Anteil des Precision Time Protocols das Gesamtergebnis eher negativ beeinflusst. Das kann zum einen durch nicht optimale Konfiguration, z.B. des PI-Reglers zur Kompensation des Frequenzdriftes, zum anderen durch die nicht für das Precision Time Protocol optimierte Anbindung der Netzwerkschnittstelle der Raspberry Pi Boards zustande gekommen sein. Die Vorbetrachtungen aus Abschnitt 3.1.4 lassen allerdings darauf schließen, dass eine deutliche Verbesserung eintritt, sobald hardwaretimestampingfähige Geräte als Clients verwendet werden.

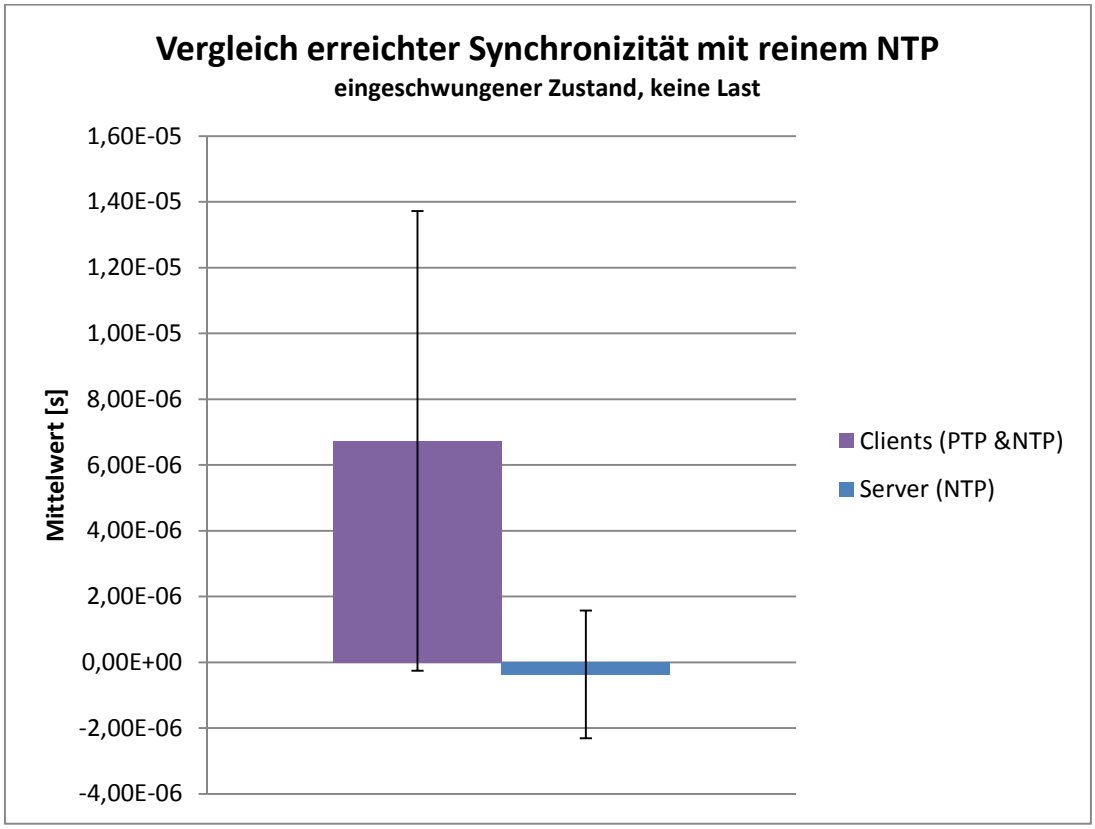

Abbildung 22: Vergleich der aufgenommenen Messwerte zwischen Server und Client, eingeschwungener Zustand, ohne Last

Es wurde somit anhand der aufgenommenen Messwerte gezeigt, dass dieses Konzept die in Abschnitt 2.6 geschilderte Anforderungsklasse II erfüllt und damit Messwerte auf einen Buszyklus genau korrelierbar bleiben.

Dennoch ist die erreichte Synchronizität über die Kombination des Network Time Protocols und des Precision Time Protocols für eine Erfüllung der Anforderungsklasse I nicht ausreichend. In den Vorbetrachtungen des Abschnitts 2.6 haben wir ermittelt, dass für die vorgegebenen Kommunikationsprotokolle eine Synchronizität von unter 1 µs gefordert ist. Dies konnte in diesem Versuchsaufbau nicht gezeigt werden.

7.1 Betrachtungen für den Lastfall

Wie bereits in Abschnitt 6 geschildert, wurden Messungen durchgeführt, in denen abwechselnd Zeitserver und Client unter Last gestellt wurden indem kontinuierlich eine große Anzahl von UDP-Paketen an sie gesendet wurden. Wenn wir diese Messungen mit den lastlosen Messungen in Tabelle 11 und Abbildung 23 vergleichen, fällt auf, dass die Zeitserver unter der zusätzlichen Last kaum zu beeindrucken waren. Lediglich das gezielte Senden an den für die Follow-Up-Nachrichten zuständigen Ports hatte deutliche Qualitätseinbußen zur Folge.

Tabelle 11: Auszug Messergebnisse für den Lastfall

		Messung 1	Messung 2	Messung 3
ohne Last	Mittelwert Clients	1,73E-04	7,89E-05	6,73E-06
	Standardabweichung Clients	3,28E-04	1,77E-04	6,99E-06
		beide Ports	Port 320	Port 319
Serverlast	Mittelwert Clients	1,49E-05	8,86E-05	2,52E-05
	Standardabweichung Clients	1,66E-05	7,72E-05	1,78E-05
Clientlast	Mittelwert Clients	1,58E-05	9,09E-06	-1,06E-05
	Standardabweichung Clients	2,33E-05	5,16E-05	1,88E-04

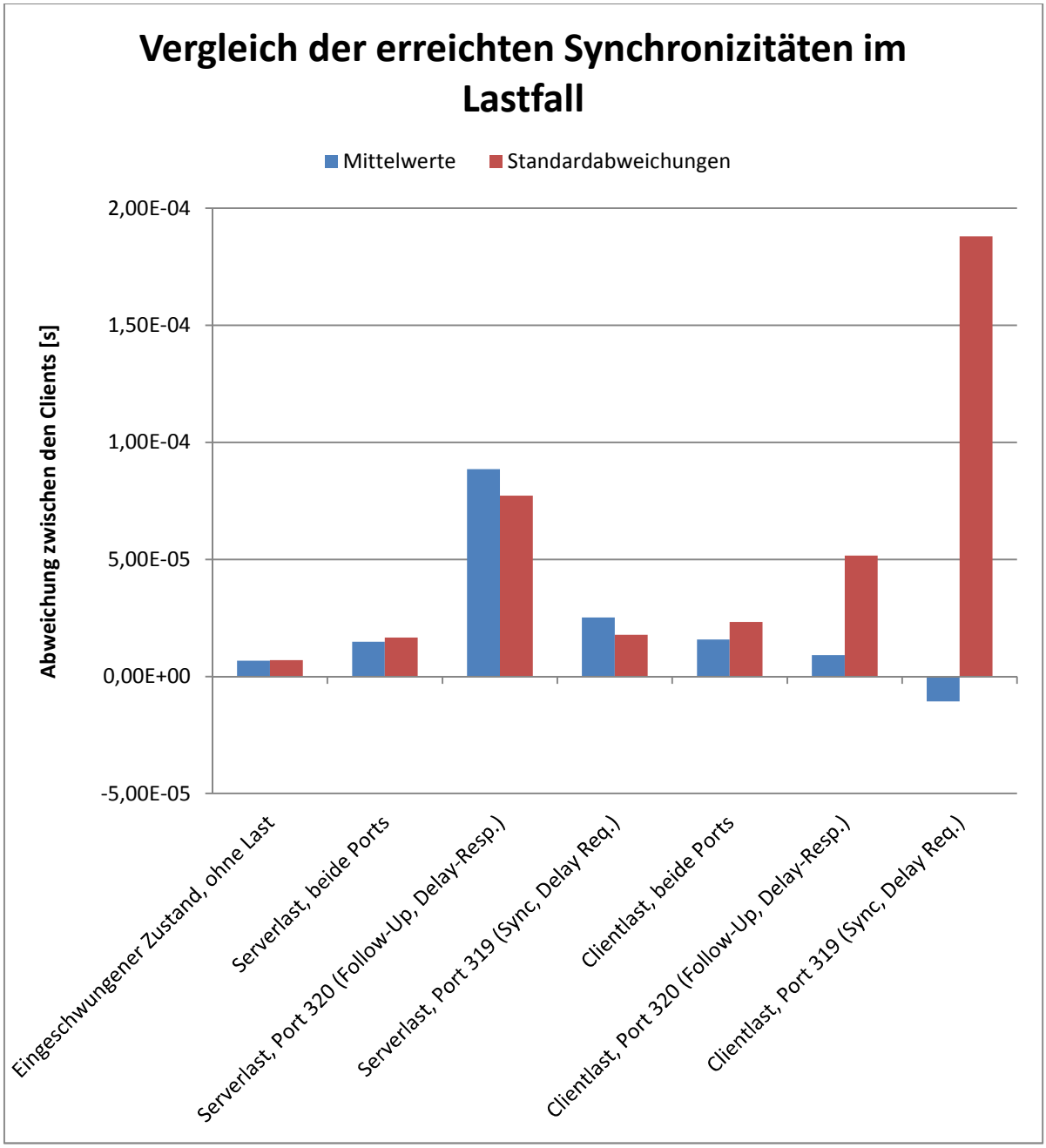

Abbildung 23: Vergleich der erreichten Synchronizität im Lastfall

Die Störung des Clients hingegen hatte zwar wenige Auswirkungen auf die gemittelte Synchronizität, hat allerdings deutliche Auswirkungen auf die Stabilität des Zeitsignals. Besonders deutlich wird das im Vergleich der Histogramme

(dargestellt in Abbildung 24, Abbildung 25 und Abbildung 26).

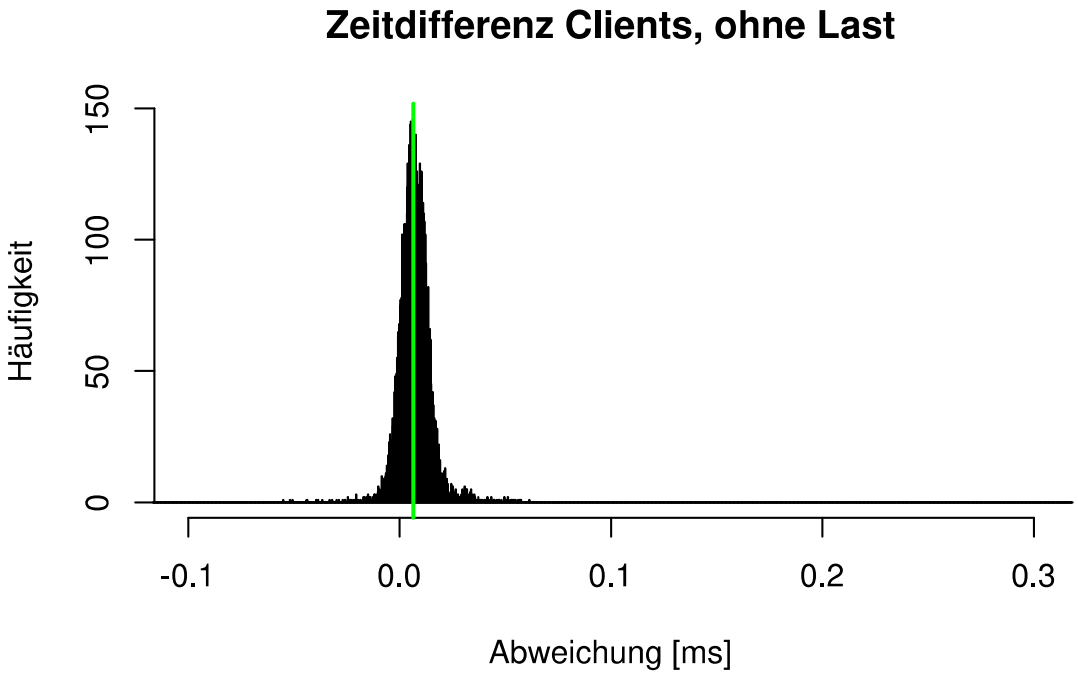

Abbildung 24: Histogramm Synchronizität Clients, ohne Last
Median grün eingezeichnet

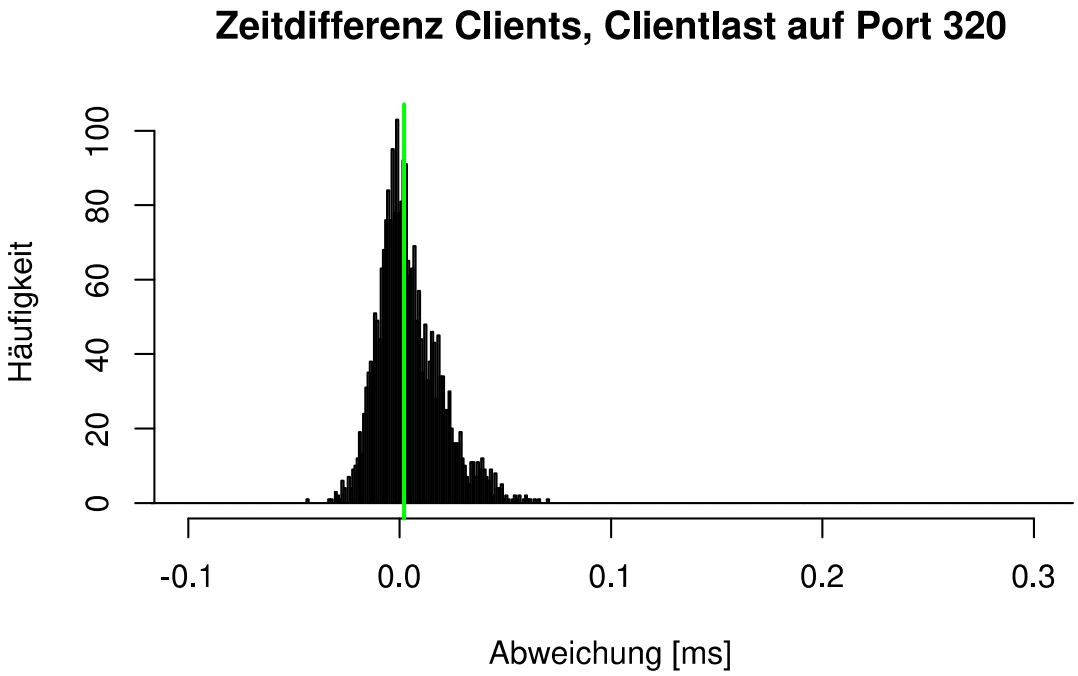

Abbildung 25: Histogramm Synchronizität Clients, Clientlast auf Port 320 (Follow-Up & Del.-Resp.)
Median grün eingezeichnet

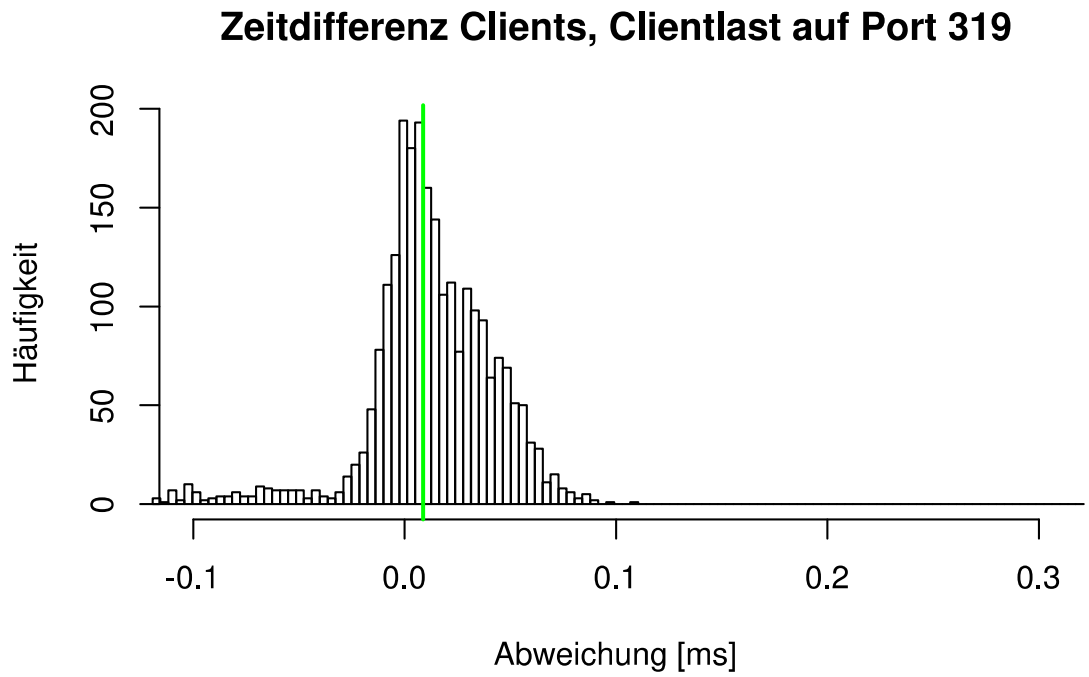

Abbildung 26: Histogramm Synchronizität Clients, Clientlast auf Port 319 (Sync, Del.-Req.) Median grün eingezeichnet

Man erkennt deutlich die breitere Streuung der Messwerte in den Lastfällen. Diese Beobachtung ist ebenfalls auf die Softwarezeitstempelung im Raspberry Pi zurückzuführen, da hier die Netzwerkqueues belastet werden und so die Verzögerungszeiten für die dem Precision Time Protocol zu Grunde liegenden Algorithmen zum einen stark verschoben und zum anderen asymmetrisch werden.

Trotz dieser Effekte bleibt es uns möglich, eine Synchronizität von unter 1 ms sicherzustellen, was immer noch für Motion-Control-Anwendungen ausreicht [Hei]. Jedoch lässt sich durch die gezielte Belastung der vom Precision Time Protocol genutzten UDP-Ports das Erfüllen der Anforderungsklasse II verhindern.

8 Zusammenfassung

In dieser Arbeit konnte gezeigt werden, dass das in Abschnitt 4.1 vorgeschlagene Konzept eine deutliche Verbesserung gegenüber einer einfachen Synchronisierung über das Network Time Protocol darstellt. Mit Hilfe des vorgestellten Konzeptes können Messdaten über eine große Distanz hinweg über ihre Zeitbasen mit geringer Abweichung zugeordnet werden, sodass die in Abschnitt 2.6 formulierte Anforderungsklasse II erfüllt ist. Bei gezielter Belastung der benötigten UDP-Ports der Teilnetze lässt sich eine in der Anwendungsklasse II geforderte Synchronizität von unter 31,25 µs nicht mehr garantieren, allerdings gibt es bisher auch keine PROFINET IO IRT Kommunikationsgeräte auf dem Markt, welche diese minimale Zykluszeit im Anwendungsfall ermöglichen.

Um Messsysteme derart zu synchronisieren, dass Anforderungsklasse I erfüllt ist, bietet das vorgestellte Konzept in der jetzigen Form einen guten Anhalt, sollte aber durch weitere Hardwareunterstützung erweitert werden, um die Abweichung dauerhaft unter 1 µs zu halten.

In Folge dieser Arbeit bieten sich weitere Untersuchungen zum Einfluss hardwaretimestampingfähiger Clients, Auswirkungen der Anzahl der Clients in einem Teilnetz und die Einbindung eines WAN in den Pfad zwischen den Teilnetzen an.

IX. Quellenverzeichnis

[Amt77] Amt für Standardisierung, Meßwesen und Warenprüfung. (1977). *SI Das Internationale Einheitensystem.* Vieweg+Teubner Verlag.

[Bau16] Bauch, A. (kein Datum). *GPS-Zeitvergleiche*. (Physikalisch-Technische Bundesanstalt) Abgerufen am 16. Februar 2016 von https://www.ptb.de/cms/ptb/fachabteilungen/abt4/fb-44/ag-442/internationale-zeitvergleiche/gps-zeitvergleiche.html

[Dan90] Dana, P. H., & Penrod, B. M. (August 1990). The Role of GPS in Precise Time and Frequency Dissemination. *GPSWorld* .

[Doy04] Doyle, P. (Juli 2004). *Introduction to Real-Time Ethernet II.* Abgerufen am 21. Februar 2016 von http://www.ccontrols.com.cn/: http://www.ccontrols.com.cn/pdf/volume5n4.pdf

[Fel09] Felser, M. (2009). Fieldbus based isochronous automation application. *14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009).* Palma.

[Fel15] Felser, M. (2015). *PROFIBUS Handbuch.* Berlin: epubli.

[Fer06] Ferrari, P., Flammini, A., & Taroni, A. (2006). An experimental approach to estimate real-time characteristic of PROFINET IO versus PROFIBUS DP V2. *5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems.* Hangzhou.

[Gam15] Gambis, D. (5. Januar 2015). *Bulletin C 49.* Abgerufen am 16. Februar 2016 von https://hpiers.obspm.fr/eoppc/bul/bulc/bulletinc.49

[Hei] Heitzer, B., & Mottok, J. (2012). *Real-time behaviour of Ethernet on the example of PROFINET.* Regensburg.

[Hor00] Horauer, M., Kerö, N., & Schmid, U. (2000). *A network interface for highly accurate clock synchronization.* Wien.

[IAO04] IAONA e.V. (2004). *IAONA Handbuch Industrial Ethernet.* Magdeburg: Industrial Automation Open Networking Alliance e.V.

[Jas04] Jasperneite, J., & Elsayed, E. (2004). Investigations on a Distributed Time-triggered Ethernet Realtime Protocol Used by PROFINET. *PROCEEDINGS OF THE 2004 INTERNATIONAL WORKSHOP ON REAL-TIME NETWORKS RTN 2004* (S. 69-72). Aveiro: Universidade de Aveiro.

[DMi10] Mills, D., Martin, J., Ed, Burbank, J., & Kasch, W. (2010). *RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification.*

[Ben12] Nuttall, B. (18. Oktober 2012). *Schaltplan Raspberry Pi Rev 2.0.* Abgerufen am 21. Februar 2016 von www.raspberrypi.org:

https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/Raspberry-Pi-Rev-2.0-Model-AB-Schematics.pdf

[Pie15] Piester, D., & Bauch, A. (2 2015). Atomuhren, Weltzeit und das Drumherum. *Astronomie + Raumfahrt* , S. 24-26.

[Dir04]Piester, D., Hetzel, P., & Bauch, A. (2004). PTB Themenschwerpunkt: Zeit- und Normalfrequenzverbreitung mit DCF77. *PTB-Mitteilungen 114* (4).

[Pig05] Pigan, R., & Metter, M. (2005). *Automating with PROFINET: Industrial communication based on industrial Ethernet.* Weinheim: Wiley-VCH.

[Pop04] Popp, M., & Weber, K. (2004). *Der Schnelleinstieg in PROFINET.* Karlsruhe: PROFIBUS Nutzerorganisation e. V.

[Rie04] Riehle, F. (2004). *Frequency Standards: Basics and Applications.* Weinheim: Wiley-VCH.

[Sun13] Sung, M., Kim, I., & Kim, T. (August 2013). Toward a Holistic Delay Analysis of EtherCAT Synchronized Control Processes. *INT J COMPUT COMMUN* , S. 608-621.

X. Anlagenverzeichnis

Anlage 1: Quelltext Kernelmodul zur Zeitstempelerfassung

Anlage 2: Quelltext Auswertungssprogramm

Anlage 3: Header Auswertungsprogramm

Weitere Anlagen: CD

Ordner	Inhalt
Hauptverzeichnis	Dieses Dokument als PDF
Kernelmodul	Quelltext des selbstgeschriebenen Linuxkernelmoduls zur Messwerterfassung
Auswertungssoftware	Quelltexte der selbstgeschriebenen Auswertungssoftware

Anlage 1: Quelltext Kernelmodul zur Zeitstempelerfassung

```
1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include <linux/kernel.h>
4 #include <linux/interrupt.h>
5 #include <linux/gpio.h>
6 #include <linux/time.h>
7
8 MODULE_LICENSE("GPL");
9 MODULE_AUTHOR("Patrick Biermann");
10 MODULE_DESCRIPTION("GPIO Interruptmodul zum kernelnahen Zeitstempeln");
11 MODULE_VERSION("0.5.3");
12
13 //GPIO defines
14 #define GPIO_INTERRUPT      17
15 #define GPIO_DESC           "GPIO interrupt module"
16 #define GPIO_DEVICE_DESC    "GPIO interrupt module"
17 #define TRIGGER_MODE        IRQF_TRIGGER_RISING // IRQF_TRIGGER_RISING oder IRQF_TRIGGER_FALLING
18
19 short int   irq_gpio    = 0;
20 static int  identifier  = 0;
21
22 //ISR für den GPIO Pin, gibt einen KERNEL_NOTICE aus
23 static irq_handler_t irq_handler(int irq,void *dev_id, struct pt_regs *regs) {
24    unsigned long flags;
25    struct timespec timestamp;
26    getnstimeofday(&timestamp);
27    local_irq_save(flags); //abschalten von hard interrupts
28
29    printk(KERN_NOTICE "gpio_timing_interrupt: Interrupt;%d;%ld;%ld\n",identifier,timestamp.tv_sec,timestamp.tv_nsec);
30    identifier++;
31
32    local_irq_restore(flags);//(wieder-)einschalten von hard interrupts
33    return (irq_handler_t) IRQ_HANDLED;
34 }
35
36 //Modul entry-point
37 static int __init gpio_timing_init(void){
38    printk(KERN_INFO "gpio_timing_interrupt: Modul startet\n");
39
40    if (gpio_request(GPIO_INTERRUPT, GPIO_DESC)) {
41      printk("gpio_timing_interrupt: GPIO request faiure: %s\n", GPIO_DESC);
42      return -1;
43    }
44
45    gpio_direction_input(GPIO_INTERRUPT);
46    gpio_export(GPIO_INTERRUPT, false);
47
48   if ( (irq_gpio = gpio_to_irq(GPIO_INTERRUPT)) < 0 ) {
49      printk("gpio_timing_interrupt: GPIO to IRQ mapping faiure %s\n", GPIO_DESC);
50      return -1;
51   }
52
53   printk(KERN_NOTICE "gpio_timing_interrupt: Mapped int %d\n", irq_gpio);
54
55   if (request_irq(irq_gpio,(irq_handler_t ) irq_handler, TRIGGER_MODE, GPIO_DESC,GPIO_DEVICE_DESC)){
56      printk("gpio_timing_interrupt: Irq Request failure\n");
57      return -1;
58   }
59    return 0;
60 }
61
62 //Modul exit-point
63 static void __exit gpio_timing_exit(void){
64    printk(KERN_INFO "gpio_timing_interrupt: Modul wird entfernt\n");
```

```
65     free_irq(irq_gpio, GPIO_DEVICE_DESC);
66     gpio_unexport(GPIO_INTERRUPT);
67     gpio_free(GPIO_INTERRUPT);
68
69     return;
70 }
71
72 module_init(gpio_timing_init);
73 module_exit(gpio_timing_exit);
```

Anlage 2: Auswertungsprogramm

```c
/*
 * File:   main.c
 * Author: Patrick Biermann
 *
 * Created on 19. Januar 2016, 14:02
 */

#include    "synchro.h"

int main(int argc, char** argv) {
    FILE *fp_efus, *fp_raspi, *fp_result;

    if (argc != 4) {
        printf("Benötige genau zwei Messwertdateien und die max. Zeitdifferenz.\nBeende.\n");
        exit(-1);
    }

    if ((fp_result = fopen(RESULT, "w+")) < 0) {
        printf("Error: Öffnen der Ergebnisdatei gescheitert: %s.\n", strerror(errno));
        exit(-1);
    } else {
        fprintf(fp_result, "seq;diff;\n");
        fclose(fp_result);
    }

    if ((fp_raspi = fopen(argv[1], "r")) < 0) {
        printf("Error: Öffnen der Raspi-Datei gescheitert: %s.\n", strerror(errno));
        exit(-1);
    }

    if ((fp_efus = fopen(argv[2], "r")) < 0) {
        printf("Error: Öffnen der Efus-Datei gescheitert: %s.\n", strerror(errno));
        exit(-1);
    }

    long int seq = 0, treffer = 0;
    struct oldline oldline = {FALSE, FALSE};
    double maxdiff = strtod(argv[3], NULL) / (double) 1000.0;
    int read_raspi = 0, read_efus = 0, dump = 0;

    while ((read_raspi != -1) && (read_efus != -1)) {
        //double diff = 0.0;
        struct long_time ts_efus, ts_raspi;
        size_t len = 0;
        char *line = NULL;

        if (dump = getline(&line, &len, fp_raspi) == -1) break;
        else fseek(fp_raspi, -strlen(line), SEEK_CUR);
        if (dump = getline(&line, &len, fp_efus) == -1) break;
        else fseek(fp_efus, -strlen(line), SEEK_CUR);
        free(line);

        if (!oldline.raspi) read_raspi = read_line(fp_raspi, &ts_raspi);

        if (!oldline.efus) read_efus = read_line(fp_efus, &ts_efus);


        long double diff = (ts_raspi.tv_sec - ts_efus.tv_sec) + (1e-9) * (ts_raspi.tv_nsec - ts_efus.tv_nsec);
```

```
59
60          //printf("%i,%i,%i,%i,%f,%i\n", ts_raspi.tv_sec, ts_efus.tv_sec, ts_raspi.tv_nsec, ts_efus.tv_nsec, diff, seq);
61          if (fabs(diff) < maxdiff) {
62              if ((fp_result = fopen(RESULT, "a")) < 0) {
63                  printf("Error: Öffnen der Ergebnisdatei gescheitert: %s.\n", strerror(errno));
64                  exit(-1);
65              } else {
66                  treffer++;
67                  fprintf(fp_result, "%ld;%Le;\n", seq, diff);
68                  fclose(fp_result);
69                  seq++;
70              }
71              oldline.efus = FALSE;
72              oldline.raspi = FALSE;
73          } else if (diff > maxdiff) {
74              oldline.efus = FALSE;
75              oldline.raspi = TRUE;
76              seq = ts_raspi.seq;
77          } else {
78              oldline.efus = TRUE;
79              oldline.raspi = FALSE;
80              seq = ts_efus.seq;
81          }
82      }
83
84      printf("%ld Ergebnisse. Done.\n", treffer);
85      fclose(fp_raspi);
86      fclose(fp_efus);
87      return (EXIT_SUCCESS);
88  }
89
90  ssize_t read_line(FILE* file, struct long_time* retval) {
91      size_t len = 0;
92      char *line = NULL;
93      ssize_t read = 0;
94      if (read = getline(&line, &len, file) != 1) {
95          char *field[3] = {0}, *ptr = NULL;
96          int i = 0;
97          //Divide input-string into token and store these
98          ptr = strtok(line, ";");
99          while (ptr != NULL) {
100             if (ptr != NULL) field[i] = ptr;
101             ptr = strtok(NULL, ";");
102             i++;
103         }
104
105         retval->seq = strtol(field[0], NULL, 10);
106         retval->tv_sec = strtold(field[1], NULL);
107         retval->tv_nsec = strtold(field[2], NULL);
108
109         free(line);
110     }
111     return read;
112 }
113
```

Anlage 3: Header Auswertungsprogramm

```
1 /*
2  * File:   synchro.h
3  * Author: Patrick Biermann
4  *
5  * Created on 19. Januar 2016, 14:29
6  */
7
8 #pragma once
9
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13 #include <errno.h>
14 #include <time.h>
15 #include <math.h>
16
17 #define RESULT  "result.csv"
18 #define MAXDIFF maxdiff
19 #define __MAXDIFF   (fabs(diff.current - diff.old) < MAXDIFF)
20 #define MAXTRIES 20
21 #define TRUE    1
22 #define FALSE   0
23
24 struct long_time {
25     long double tv_sec;
26     long double tv_nsec;
27     long int seq;
28 };
29
30 struct oldline {
31     int efus;
32     int raspi;
33 };
34
35 ssize_t read_line(FILE*, struct long_time*);
```