HAW HAMBURG

Bachelorarbeit

Nils-Ole Bickel

Zeitsynchronisation von mobilen Endgeräten mit Bluetooth Low Energy

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Nils-Ole Bickel

Zeitsynchronisation von mobilen Endgeräten mit Bluetooth Low Energy

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Martin Becke
Zweitgutachter: Prof. Dr. Jan Sudeikat

Eingereicht am: 21. Dezember 2021

Nils-Ole Bickel

Thema der Arbeit

Zeitsynchronisation von mobilen Endgeräten mit Bluetooth Low Energy

Stichworte

Zeitsynchronisation, Bluetooth low energy, iOS

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Zeitsynchronisation zwischen mehreren mobilen Endgeräten. Zur Kommunikation zwischen den Geräten wird Bluetooth Low Energy verwendet und die Synchronisation findet im Anwendungsberich der verwendeten Betriebssysteme statt. Nach einer Erläuterung der verwendeten Technologien werden mögliche Algorithmen zur Zeitsynchronisation vorgestellt. Eine Testumgebung wird vorgestellt um diese Algorithmen auf die resultierende Präzision zu überprüfen. Aufgrund dieser Test werden die Algorithmen bewertet mit Beachtung von Anwendungszenarien und möglichen Störfaktoren.

Nils-Ole Bickel

Title of Thesis

Time synchronisation of mobile consumer devices using Bluetooth low energy

Keywords

Time synchronisation, Bluetooth low energy, iOS

Abstract

This thesis looks at time synchronisation between mobile devices. The communication between devices is handled by Bluetooth low-energy. Synchronisation is done by an application within the operating system of the mobile device. Differrent algorithms are presented and run on real hardware, to compare the precision of the time synchronisation. The algorithms are evaluated based on possible use cases.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Mobile Endgeräte, wie Smartphones und Tablets, sind mittlerweile weit verbreitet. Viele Personen tragen stets mindestens eins bei sich oder besitzen mehrere mobile Endgeräte, wodurch es häufig vorkommt, dass sich mehrere dieser mobilen Geräte in unmittelbarer Nähe zueinander befinden. Dementsprechend ist es sinnvoll diese Geräte kooperativ miteinander zu verwenden, um den nutzen und die Anwendungsmöglichkeiten der Geräte zu erweitern.

Kooperation zwischen mehreren Geräten gibt es bereits zum Beispiel mit AirDrop oder Nearby Sharing. Dies sind Möglichkeiten um Dateien leicht zwischen mehreren Geräten zu teilen. Des weiteren können zwischen iOS Geräten Spiele local miteinander über Game Center gespielt werden. Doch eine allgemeine Möglichkeit um auf mehreren Geräten eine Aktion synchronisiert auszuführen wird nicht bereitgestellt. Dies könnte allerdings interessante neue Möglichkeiten eröffnen. Es könnte zum Beispiel genutzt werden um von mehreren Positionen gleichzeitig Fotos zu machen, um Ton und Video automatisch zu synchronisieren, wenn man unterschiedliche Geräte für Tonaufnahme und Videoaufnahme verwendet, um Medien synchron auf mehreren Geräten abzuspielen oder um im Sport Zeiten mit verschiedenen Geräten zu messen.

Um eine solche synchrone Aktion auszuführen, müssen die Geräte über eine synchronisierte Uhrzeit verfügen. Mit Hilfe dieser synchronen Uhrzeit kann dann ein Zeitpunkt festgelegt werden, an dem alle Geräte gleichzeitig eine Aktion durchführen.

Grundlegend gibt es zwei Arten der Zeitsynchronisation: Die interne Synchronisation und die externe Synchronisation. Zwei Uhren sind intern Synchronisiert, wenn der Unterschied zwischen den Zeiten der beiden Uhren einen vorher festgelegten Maximalwert nicht überschreiten. Der Unterschied zwischen den Zeiten der beiden Uhren wird auch Präzision genannt. Bei der externen Synchronisation darf der Unterschied zwischen den Zeiten und einer Referenzzeit ebenfalls nicht einen festgelegten Maximalwert überschreiten. Um eine synchrone Aktion durchzuführen wird dementsprechend nur eine interne Synchronisation benötigt.

Zeitsynchronisation zwischen mehreren Geräten ist bereits seit langem ein bekanntes Problem in verteilten Systemen. Die am weitesten verbreitete Lösung zu Zeitsynchronisation ist das Network Time Protocol (NTP) [6]. Beim NTP sendet ein Gerät eine Anfrage an einen NTP-Server. Der NTP-Server antwortet mit seiner aktuellen Uhrzeit und Informationen über die erwartete Genauigkeit der Uhrzeit. Beim Versenden und Empfangen der Nachrichten wird immer die aktuelle Uhrzeit mit aufgenommen. Anhand dieser Uhrzeiten wird dann die Zeit berechnet, die der Hin- und Rückversand der Nachricht benötigt hat. Mit der Annahme, dass der Hinversand ungefähr die gleiche Zeit benötigt, wie der Rückversand kann dann die gesendete Uhrzeit um die Zeit, die der Rückversand benötigt hat, angepasst werden. Das Ursprungsgerät führt mehrere dieser Messungen, mit teilweise unterschiedlichen NTP-Servern, durch und passt anschließend die eigene Uhrzeit an das gemessene Ergebnis an. NTP ist für Paketbasierten Datentransfer ausgelegt und erfolgt normalerweise über UDP.

Auch im Bereich von IOT Geräten und Sensoren gibt es bereits Lösungen zur Zeitsynchronisation. Diese beruhen häufig auf Bluetooth Low Energie (BLE) um zwischen den Geräten zu kommunizieren. CheepSync [7] ist ein Beispiel für ein Algorithmus, der mit den Broadcastmöglichkeiten von BLE arbeitet. Bei CheepSync gibt es ein spezielles Gerät, dass als Beacon agiert. Der Beacon sendet in regelmäßigen Abständen seine aktuelle Zeit als Broadcast. Zusätzlich sendet er die Zeit, die das Senden des vorangegangene Broadcast benötigt hat. Um diese Zeit zu messen wird spezielle Hardware für den Beacon verwendet. Die Empfänger müssen nur die Broadcast empfangen und können nach zwei empfangenen Nachrichten bereits ihre Uhrzeit anpassen. Eine andere Möglichkeit der Zeitsynchronisation mit BLE basiert auf den Events, wenn eine Verbindung zwischen zwei Geräten aufgebaut wird [4]. Es wird ausgenutzt, dass zum Zeitpunkt der erfolgreichen Verbindung sowohl der Master als auch der Slave fast gleichzeitig ein Event auslösen. Die beiden Geräte können dann zum Zeitpunkt des Events die aktuelle Uhrzeit speichern und sich anschließend über diesen Referenzzeitpunkt synchronisierten.

Insgesamt benötigen die Lösungen zur Zeitsynchronisation häufig Zugriff auf hardwarenahe Schichten oder speziell angepasste Hardware.

1.1 Zielsetzung

In den folgenden Kapiteln wird untersucht welche Möglichkeiten bestehen mehrere mobile Geräte miteinander zu Synchronisiert. Hierbei geht es besonders darum, passende

Algorithmen zu finden und diese so anzupassen, dass sie in dem speziellen Fall mit mobilen Endgeräten verwendet werden können. Diese werden dann für konkrete Hardware realisiert und mit Hilfe von mehreren Test verglichen. Dabei soll besonders auf Fehleranfälligkeit und Präzision der Zeitsynchronisation geachtet werden. Die Zeitsynchronisation soll so durchgeführt werden, dass sie auf dem Betriebssystem der mobilen Endgeräte läuft, ohne die vorgegebenen Einschränkungen zu umgehen. Die Kommunikation soll ausschließlich über Bluetooth low-energy (BLE) realisiert werden.

1.2 Struktur der Arbeit

Zu beginn der Arbeit, in Kapitel 2, werden die Anforderungen und die Entscheidungen für die verwendete Hardware, Technologie und Software beschrieben. Des weiteren wird darauf eingegangen wie diese Funktionieren.

Das Kapitel 3 beschäftigt sich mit den Algorithmen zu Zeitsynchronisation. Anfangs werden die Anforderungen an die Algorithmen dargestellt. Daraufhin werden die drei Algorithmen vorgestellt, die als Teil dieser Arbeit miteinander verglichen werden.

In Kapitel 4 wird die Testumgebung vorgestellt. Es wurde ein Programm für die verwendete Hardware geschrieben und es wird genauer darauf eingegangen, wie die Algorithmen in diesem Programm konkret implementiert wurden.

In Kapitel 5 wird untersuch, welche Faktoren Einfluss auf die Präzision und Fehleranfälligkeit der Zeitsynchronisation haben. Anhand dieser Faktoren werden Tests vorgestellt, die mit der Testumgebung durchgeführt werden sollen und den Einfluss durch die einflussnehmenden Faktoren untersuchen. Die Ergebnisse der Tests werden dargestellt und mit den Annahmen verglichen.

Im letzten Kapitel werden die Ergebnisse der Arbeit anhand der Zielsetzung bewertet und es wird untersucht, welche weiterführenden Tests noch durchgeführt werden können.

2 Technologien

2.1 Hardware

Zum Testen der Algorithmen wurde eine Auswahl aus diversen mobilen Endgeräten der letzten Jahre gewählt. Dies soll zeigen, wie präzise und fehleranfällig die Algorithmen in einem natürlichen Umfeld sind. Eine Liste aller verwendeten Geräte ist in Tabelle 2.1 vorhanden. Bei den Geräten handelt es sich ausschließlich um Geräte mit iOS. Dies soll den Aufwand der Arbeit eingrenzen, da nur die Einschränkungen eines Betriebssystems beachtet werden müssen und das Programm, zum Testen der Algorithmen, speziell auf iOS angepasst werden kann.

Name	Gerätetyp	iOS Version	Bluetooth Version	Prozessor
11	iPhone 11	14.4	5.1	A13
6s	iPhone 6s	14.4	4.2	A9
6s-2	iPhone 6s	14.4	4.2	A9
6	iPhone 6	12.5.1	4.2	A8
iPad	iPad Gen 5	14.4	4.2	A9
Pro	iPad Pro Gen 2	14.4	4.2	A10X

Tabelle 2.1: Auflistung der verwendeten mobilen Endgeräte

2.2 Bluetooth Low Energy

Zur Kommunikation zwischen den mobilen Endgeräten soll Bluetooth Low Energy (BLE) verwendet werden [8]. BLE ist eine Variante von Bluetooth, die besonders durch einen geringen Energieverbrauch überzeugt. In dem Beispiel der Zeitsynchronisation auf mobilen Endgeräten hat BLE einen weiteren Vorteil. Die Einbindung in das Betriebssystem der mobilen Endgeräte sorgt dafür, dass es, im Gegensatz zu anderen Kommunikationsmöglichkeiten, keinen expliziten Input des Benutzers benötigt, um eine Kommunikation

zwischen den mobilen Endgeräten aufzubauen. Dies währe zum Beispiel der Fall bei einer Kommunikation über eine normale Bluetooth-Verbindung.

Das BLE Protokoll ist als Stack aufgebaut und besteht aus drei Hauptbestandteilen:

- **Applikation:** Die Applikation interagiert mit dem Protokoll Stack um die spezifische Anwendung umzusetzen.
- **Host:** Der Host beinhaltet die hören Ebenen des Bluetooth Protokoll Stacks. Er teilt sich auf in:
  - Generic Access Profile (GAP): Das GAP gibt vor, wie Geräte miteinander interagieren. Dies beinhaltet unter anderem, wie sich Geräte entdecken, wie Broadcasts versendet werden oder wie eine sichere Verbindung zwischen zwei Geräten aufgebaut wird.
  - Generic Attribute Profile (GATT): Das GATT stellt das Datenmodell der Daten, die ausgetauscht werden, bereit. Daten werden in Services und Charakteristiken aufgeteilt. Ein Service besteht aus mehreren zusammenhängenden Charakteristiken und eine Charakteristik kann als ein Datenpunkt mit zugehörigen Metadaten angesehen werden. Die gemessene Temperatur eines Thermometers könnte zum Beispiel eine Charakteristik sein. Zum Austausch von Daten verwendet das GATT das ATT.
  - Attribute Protocol (ATT): Dass ATT ist ein simples Protokoll um Attribute von einem Server zu einem Client zu senden. Der Client kann hierzu Schreibe- oder Leseanfragen für ein Attribut senden. Das Protokoll ist so aufgebaut, dass sobald eine Anfrage gesendet wurde erst nach dem Erhalt einer Antwort eine neue Anfrage gesendet werden kann.
  - Security Manager Protokoll (SMP): Der SMP ist dafür zuständig Schlüssel auszutauschen, um eine verschlüsselte Kommunikation über BLE zu ermöglichen.
  - Logical Link Control and Adaptation Protocol (L2CAP): Das L2CAP ist dafür zuständig die Protokolle der höheren Ebenen in einheitliche BLE Pakete umzuwandeln. Außerdem teilt es Pakete, die über die Maximalgröße von 27 Byte hinausgehen in kleinere Teile auf. Beim Empfangen ist das L2CAP dafür zuständig aufgeteilte Pakete wieder zusammenzufügen und an die entsprechende Stelle im Stack weiterzuleiten.

- **Controller:** Der Controller beinhaltet die niedrigen Ebenen des Protokoll Stacks. Er teilt sich auf in:
  - Link Layer (LL): Der LL steuert direkt den PHY und ist für die Rolle des BLE Gerätes in der Kommunikation zuständig. Ein BLE Gerät kann vier unterschiedliche Rollen einnehmen: Advertiser, Scanner, Master und Slave.
  - Physical Layer (PHY): Der PHY ist für das tatsächliche senden der analogen Nachrichten zuständig und wandelt eintreffende analoge Signale in eine digitale Darstellung um.

Die Kommunikation zwischen Host und Controller findet mit Hilfe des Host Controller Interface (HCI) statt. In Abbildung 2.1 ist der Stack noch einmal dargestellt.

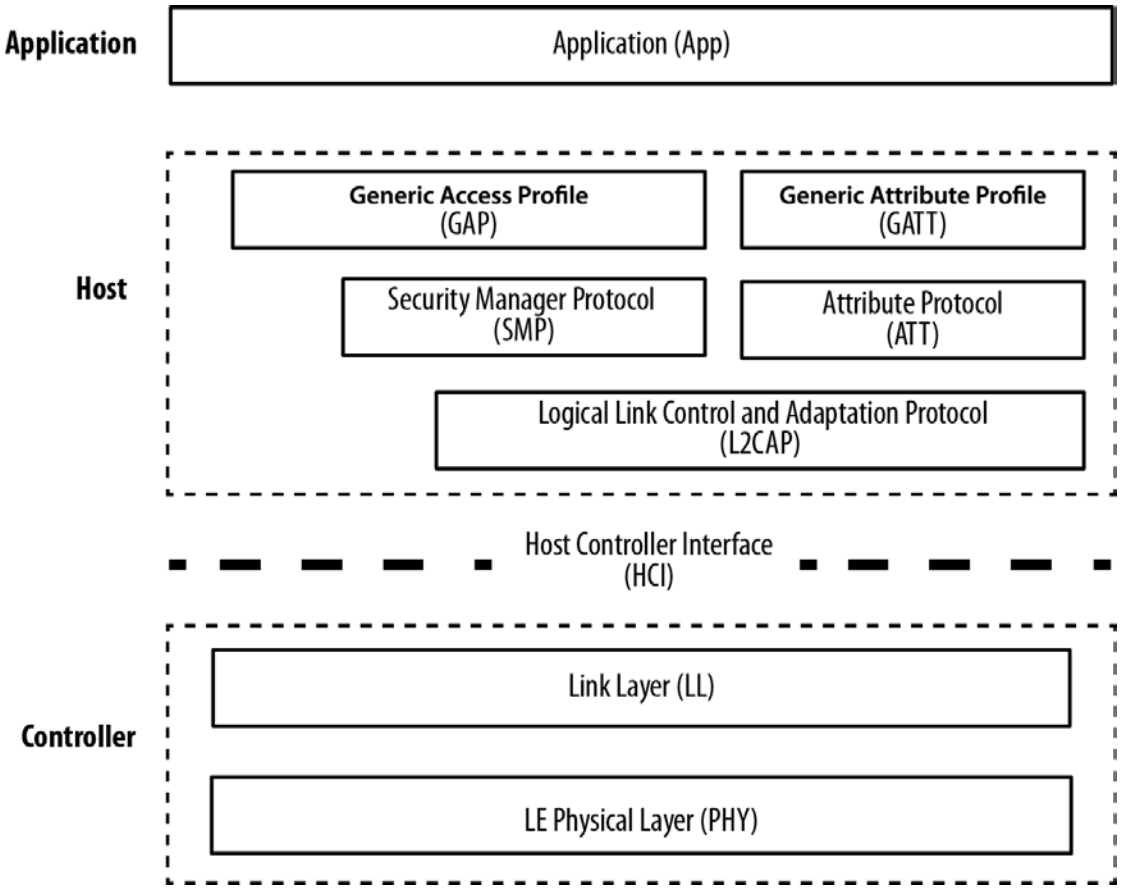


Abbildung 2.1: BLE Protokoll Stack [8]

Es gibt zwei Möglichkeiten, wie Geräte per BLE miteinander kommunizieren können. Sie können entweder einen Broadcast senden oder über eine direkte eins zu eins Verbindung kommunizieren.

2.2.1 Broadcast

Mit einem Broadcast kann eine Nachricht an alle horchenden Geräte in Reichweite gesendet werden. Beim Broadcasten gibt es dementsprechend zwei Rollen, die ein Gerät einnehmen kann:

- Broadcaster: Sendet periodisch advertising packets an alle Geräte in Reichweite.
- Observer: Scannt durchgehend nach advertising packets die den Anforderungen der Suche entsprechen.

Der Broadcast nutzt somit das advertisen aus, das für einen Verbindungsaufbau der eins zu eins Verbindungen benötigt wird, um geringe Datenmengen an mehrere Geräte gleichzeitig zu senden. Ein advertising packet besteht aus 31 byte und beinhaltet Informationen über den Broadcaster und die zu sendende Nachricht.

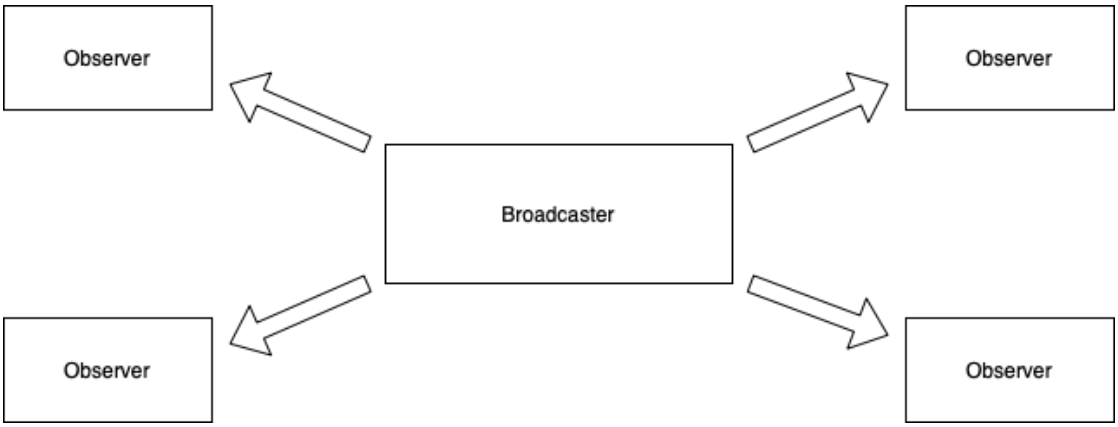


Abbildung 2.2: Broadcast Topology

2.2.2 Kommunikation über direkte Verbindung

Um eine bidirektionale Kommunikation zu führen und um mehr Daten, als im advertising packet enthalten sind, zu senden muss eine Verbindung zwischen zwei Geräten aufgebaut

werden. Mit einer Verbindung können genau zwei Geräte bidirektional miteinander kommunizieren. Die beiden kommunizierenden Geräte nehmen hierbei unterschiedliche Rollen ein:

- Central (master): Scannt durchgehend nach advertising packets. Sobald ein den Vorgaben entsprechendes advertising packet gefunden wird initiiert er eine Verbindung. Nachdem die Verbindung aufgebaut ist, kann der Central Daten mit dem Peripheral austauschen. Der Central initiiert einen periodischen Datenaustausch zwischen den Geräten.
- Peripheral (slave): Sendet periodisch advertising packets mit informationen über sich selbst an alle Geräte in Reichweite und akzeptiert Verbindungsanfragen von einem Central. Nachdem eine Verbindung aufgebaut wurde kann das Peripheral Daten mit dem Central austauschen.

Ein Central kann mit beliebig vielen Peripherls gleichzeitig verbunden sein und ein Central kann mit mehreren Peripherals gleichzeitig verbunden sein. Es ist auch möglich, dass ein Gerät bei einer Kommunikation als Central agiert und gleichzeitig bei einer weiteren als Peripheral.

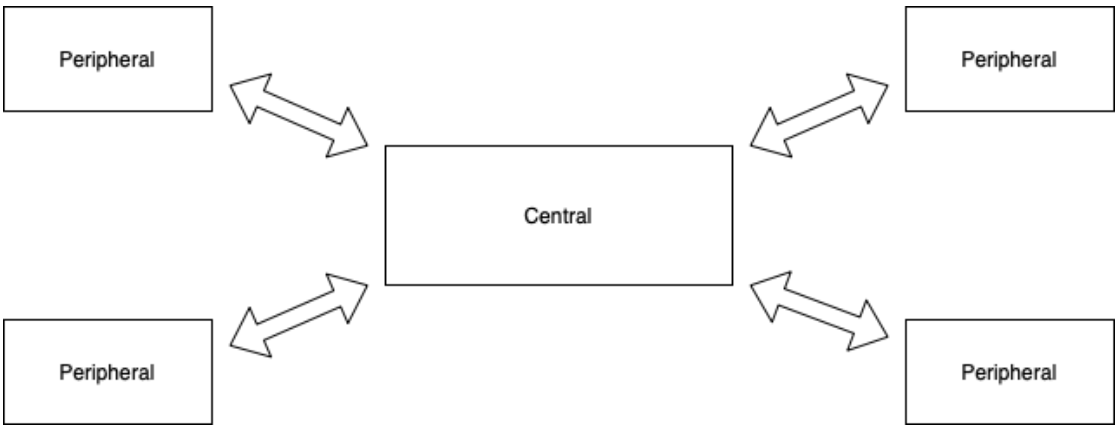


Abbildung 2.3: Verbindungs Topology

Der Datenaustausch zwischen Central und Peripheral findet über Services und Characteristics statt.

- Service: Eine Gruppe von Characteristics die konzeptionell zusammengehören.
- Characteristic: Ein einzelner Datenpunkt. Kann eine maximale Datenmenge von 512 byte enthalten.

Die Services werden vom Peripheral definiert. In den advertising packets des Peripherals kann bereits enthalten sein, dass das Peripheral ein bestimmten Service unterstützt. Dies hilft dem Central die Entscheidung zu treffen sich mit einem bestimmten Peripheral zu verbinden.

Für den Datenaustausch wird nun grundsätzlich vom Central eingeleitet. Der Central einzelne Characteristics lesen, beschreiben oder anfragen Updates zu erhalten, sobald sich der wert einer bestimmten Charakteristik ändert.

2.3 CoreBluetooth

CoreBluetooth [1] ist das Framework, dass einer iOS App die Möglichkeit gibt auf die BLE Funktionen des Gerätes zuzugreifen. CoreBluetooth abstrahiert hierbei die Funktionen des BLE Stacks und stellt einfache Möglichkeiten bereit über BLE mit anderen Geräten zu kommunizieren.

2.3.1 CBCentralManager

Um das Gerät als Central zu konfigurieren wird eine Instanz vom CBCentralManager benötigt. Funktionen, die ein CBCentralManager ausführen kann beinhalten:

- scannen nach Peripherals, die einen vorgegebenen Service zur Verfügung stellen
- Verbindungsaufbau mit einem gefundenen Peripheral
- Abfrage einer Liste mit verbundenen Peripherals

Der CBCentralManager informiert die App über das Auftreten von Ereignissen, wie dem finden eines neuen Peripherals, das erfolgreiche verbinden mit einem Peripheral oder den Verbindungsverlust mit einem Peripheral, über das Delegate-Pattern. Dies wird häufig in iOS-Frameworks verwendet und ist grundsätzlich eine Möglichkeit die Abarbeitung eines Ereignisses an ein anderes Objekt zu deligieren. Wichtige Ereignisse, die vom CBCentralManager an das Delegate weitergeleitet werden beinhalten:

- das Entdecken eines neuen Peripherals, nach dem gescannt wurde
- die Bestätigung einer Verbinden mit einem Peripheral

- das Fehlschlagen einer Verbindung mit einem Peripheral
- den Abbruch einer Verbindung mit einem Peripheral
- die Änderung des Zustands vom CBCentralManager. Der Zustand besagt, ob das Gerät aktuell in der Lage ist Bluetooth zu verwenden.

2.3.2 CBPeripheral

Der CBCentralManager erstellt ein CBPeripheral-Objekt für alle Peripherals, die er findet. Dieses Objekt beinhaltet Informationen zum Peripheral und wird genutzt, um mit dem Peripheral Nachrichten auszutauschen. Funktionen, die ein CBPeripheral ausführen kann beinhalten:

- das Entdecken von angebotenen Services
- das Entdecken von vorhandenen Charakteristiken
- das Lesen der Werte einzelner Charakteristiken
- das Schreiben von Werten auf einzelne Charakteristiken
- die Anfrage updates zu erhalten, sobald sich der Wert einer Charakteristik ändert
- das Messen des RSSI-Wertes zum Periheral

Die Antworten des externen Peripherals auf die Anfragen vom CBPeripheral werden mit dem Delegate-Pattern an die App übertragen. Ereignisse, die übertragen werden beinhalten:

- Erfolgsmitteilungen beim Entdecken von Services und Charakteristiken
- Änderungen von Werten einer Charakteristik
- Erfolgsmitteilung vom Schreiben auf einer Charakteristik
- Gemessene RSSI-Werte

2.3.3 CBPeripheralManager

Um das Gerät als Peripheral zu konfigurieren wird eine Instanz des CBPeripheralManagers benötigt. Funktionen die eine CBPeripheralManager ausführen kann beinhalten:

- das Hinzufügen und Entfernen von Services
- das Starten und Beenden advertising packeges zu senden
- das Senden von einer Wertänderung einer Charakteristik, falls ein Central angefragt hat über diese Änderung informiert zu werden
- das Senden von Antworten bei Lese- oder Schreibanfragen des Centrals

Auch der CBPeripheralManager nutzt das Delegate-Pattern um die App über vorgekommene Ereignisse zu informieren. Diese beinhalten:

- Bestätigungen, dass ein Service hinzugefügt wurde oder angefangen wurde adverising packeges zu senden
- Anfragen von einem Central über Änderungen einer Charakteristik informiert zu werden oder nicht mehr informiert zu werden
- Lese- und Schreibanfragen vom Central für eine Charakteristik
- die Änderung des Zustands vom CBPeripheralManager. Der Zustand besagt, ob das Gerät aktuell in der Lage ist Bluetooth zu verwenden.

2.3.4 Broadcast

Mit CoreBluetooth gibt es keine Möglichkeiten direkt Broadcast-Nachrichten über BLE zu verschicken. Allerdings gibt es zwei Funktionen in iOS, für die BLE Broadcast-Nachrichten versendet werden.

Zum einen funktioniert das Advertisen von BLE Peripherals nach BLE Spezifikation über einen BLE Broadcast. Hierbei handelt es sich allerdings um Nachrichten, mit denen man sich mit dem Peripheral verbinden kann. Des weiteren gibt es keine Möglichkeit diesen BLE Broadcast-Nachrichten eigene Daten hinzuzufügen.

Zum anderen funktionieren iBeacons mit BLE Broadcast-Nachrichten [2].

2.3.5 Senden eine iBeacon-Nachricht

iBeacon ist ein Standart von Apple um sich in geschlossenen Räumen zu lokalisieren. Hierzu sind in einem Raum mehrere iBeacons an festen Positionen verteilt. Diese versenden in regelmäßigen abständen eine BLE Broadcast-Nachricht. Anhand der Signalstärke kann der Abstand zwischen dem empfangenden Gerät und dem iBeacon bestimmt werden. Wenn das empfangende Gerät in Reichweite von mindestens drei iBeacons ist kann die Position des empfangenden Geräts trianguliert werden. Mobile Endgeräte mit iOS können auch als iBeacon eingerichtet werden und somit ist es möglich iBeacon-Nachrichten über einen BLE Broadcast zu versenden. Eine iBeacon Nachricht besteht aus drei Werten: UUID, Major und Minor.

Um eine iBeacon-Nachricht zu versenden wird das CoreLocation Framework benötigt. Aus dem CoreLocation Framework wird ein CLBeaconRegion-Objekt mit Major, Minor und UUID erstellt. Die Major und Minor Werte sind jeweils frei wählbare UInt16-Werte und können zum Versenden von Daten mit dem BLE Broadcast verwendet werden. Um die iBeacon-Nachricht zu versenden wird ein CBPeripheralManager benötigt. Das CLBeaconRegion-Objekt kann dem CBPeripheralManager an stelle eines advertising package übergeben werden, woraufhin die iBeacon-Nachricht in regelmäßigen Abständen als BLE Broadcast versendet wird.

2.3.6 Empfangen eine iBeacon-Nachricht

Eine iBeacon-Nachricht wird mit Hilfe eines CLLocationManagers empfangen. Der CLLocationManager hat eine Funktion, mit der iBeacon-Nachricht empfangen werden. Hierzu muss man der Funktion ein CLBeaconRegion-Objekt übergeben. Sobald eine iBeacon-Nachricht mit den Werten des CLBeaconRegion-Objekts empfangen wird, wird der App, mithilfe des Delegat-Patterns eine Empfangsbestätigung gesendet.

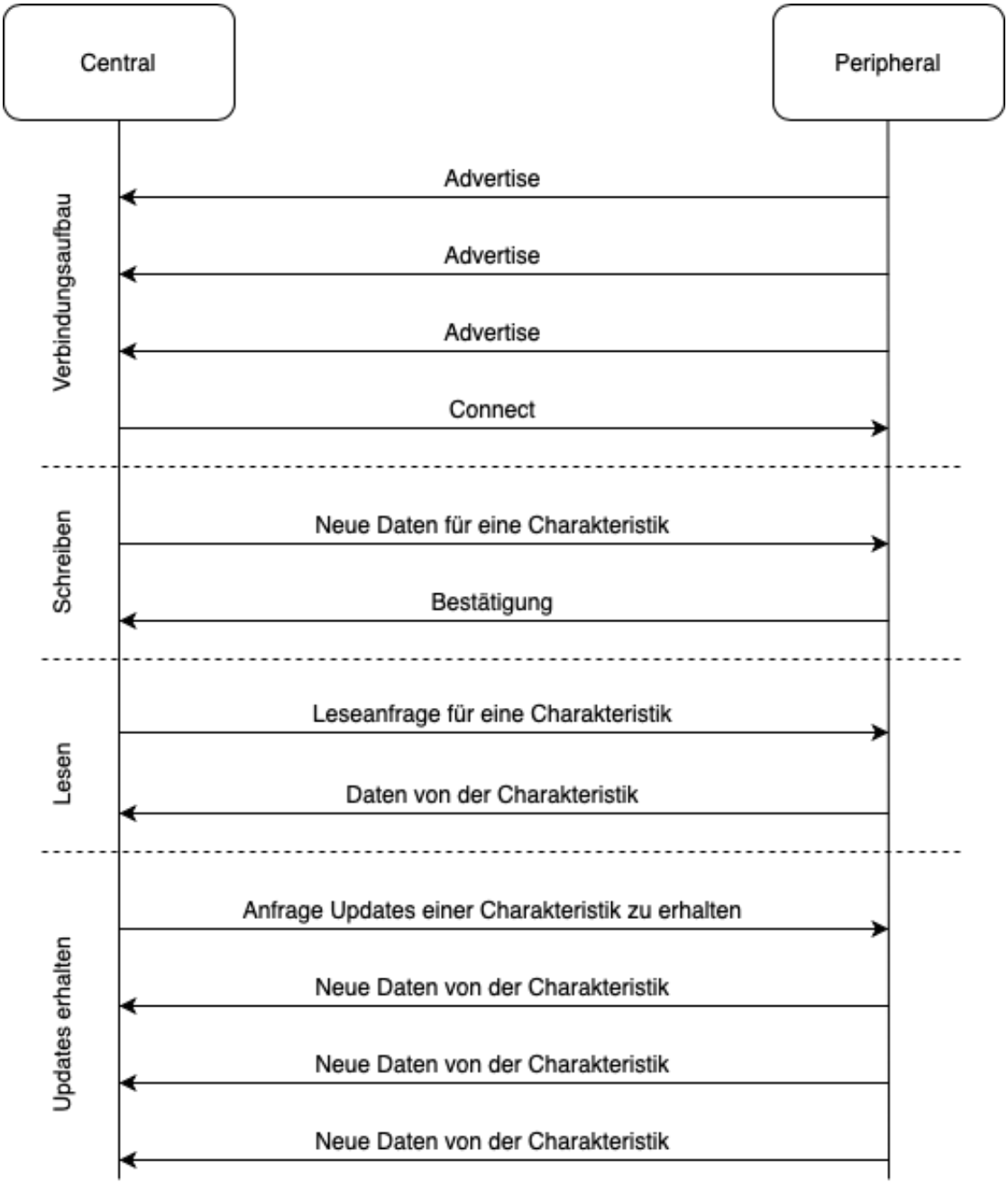


Abbildung 2.4: Datenaustausch zwischen Central und Peripheral

3 Algorithmen

Durch die Kapselung vom BLE Stack in CoreBluetooth sind die Möglichkeiten von BLE in iOS leicht eingeschränkt. Algorithmen, die auf tieferen Ebenen des BLE Stacks ansetzen um eine Zeitsynchronisation zu erzielen, können dementsprechend nicht umgesetzt werden. Speziell ist es nicht möglich Zeitstempel erst beim Erstellen einer Nachricht hinzuzufügen und auch die tatsächlichen Verbindungsnachrichten werden nicht direkt an die Applikation weitergeleitet. Dies bedeutet, dass zum Beispiel der Ansatz von CheepSync [7] für der Zeitsynchronisation zweier mobiler Endgeräte nicht möglich ist. Dementsprechend werden Algorithmen zur Zeitsynchronisation betrachtet, die einen allgemeinen und nicht auf die Technik spezifischen Ansatz verwenden. Hiervon wurden drei Ausgewählt.

3.1 Christion Algorithmus

Der Christian Algorithmus [3] wurde 1989 von Flaviu Cristian veröffentlicht. Er beruht auf der Annahme, dass eine Nachricht, die von einem System A an ein System B gesendet wird nahezu die gleich Zeit benötigt, wie eine Nachricht die vom System B zum System A gesendet wird.

Wenn sich nach dem Algorithmus ein System A mit einem System B synchronisieren möchte sendet System A eine Zeitanfrage an System B und speichert sich den Zeitpunkt(T_0) an dem diese Nachricht versendet wurde. System B sendet als Antwort die aktuelle Zeit des Systems(T_B). Sobald System A die Antwort von System B erhalten hat speichert es sich den Empfangszeitpunkt(T_1). Nun kann System A die Differenz zwischen der eigenen Zeit und der Zeit von System B berechnen. Hierzu wird sich auf die Annahme berufen, dass eine Nachricht, die von einem System A an ein System B gesendet wird nahezu die gleich Zeit benötigt, wie eine Nachricht die vom System B zum System A gesendet wird.

Die Zeit die das Versenden von beiden Nachrichten zusammen benötigt haben ist $RTT = T_1 - T_0$. Diese Zeit wird auch roundtrip time(RTT) genannt. Dementsprechend dauert das Senden einer Nachricht ungefähr $\frac{RTT}{2}$. Somit entspricht der Zeitpunkt $T_B + (\frac{RTT}{2})$ den Zeitpunkt, den System B hatte als System A T_1 gelesen hat. Die Differenz zwischen den Zeiten der Systeme entspricht somit $(T_B + (\frac{RTT}{2})) - T_1$. Anhand dieser Differenz kann man nun die Uhr von System A anpassen um eine Synchronisation zwischen System A und System B zu bewirken.

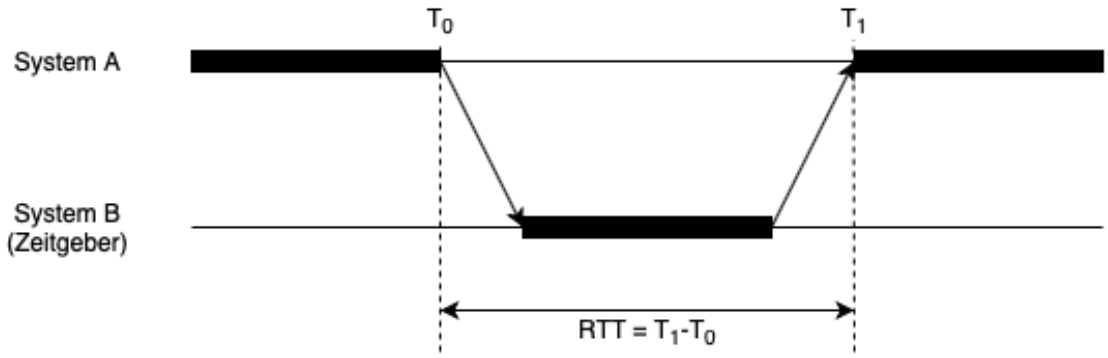


Abbildung 3.1: Christian Algorithmus

3.2 Erweiterter Christion Algorithmus

Der Christion Algorithmus basiert auf der Annahme, dass eine Nachricht, die von einem System A an ein System B gesendet wird nahezu die gleich Zeit benötigt, wie eine Nachricht die vom System B zum System A gesendet wird. Diese Annahme ist bei BLE allerdings nicht immer gegeben, da es durch Störungen zum erneuten Senden von Nachrichten kommen kann oder die Nachricht erst verzögert empfangen werden kann. Der erweiterte Christion Algorithmus versucht die Auswirkungen dieser Fehler zu minimieren.

Beim erweiterten Christion Algorithmus wird die Zeitdifferenz der Uhren von System A und System B nicht nur einmal gemessen sondern 8 mal. Aus diesen 8 Messungen wird nun die Zeitdifferenz gewählt, bei der die RTT am geringsten war, da bei dieser Messung die Wahrscheinlichkeit auf aufgetretene Fehler am geringsten ist. Dementsprechend sollte diese Zeitdifferenz der tatsächlichen Zeitdifferenz am nächsten kommen.

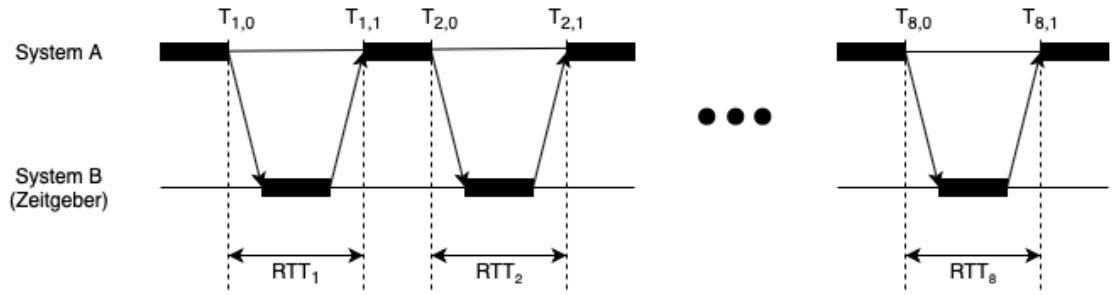


Abbildung 3.2: Erweiterter Christian Algorithmus

3.3 Reference Broadcast Synchronisation

Die grundlegende Idee einer Broadcast Synchronisation ist es, dass ein Sender eine Broadcast Nachricht mit seiner aktuellen Uhrzeit (T_s) sendet. Diese Broadcast Nachricht kann dann von mehreren Empfängern empfangen werden. Die Empfänger messen die Uhrzeit, zu der sie die Nachricht empfangen haben (T_e) und passen die eigene Uhrzeit entsprechend der Differenz zwischen den beiden Zeiten an. Der hierbei entstehende Fehler ist die Zeit, die zwischen dem Messen der beiden Uhrzeiten vergangen ist (t_{ges}). Diese Zeit kann man in sechs Bestandteile aufteilen:

Aufbauzeit: Das ist die Zeit, die für das Zusammenstellen der Broadcast Nachricht und dem Aufrufen vom Bluetooth-Controller benötigt wird. Hier können nicht deterministische Verzögerungen durch das Betriebssystem und andere Operationen auf dem Mobilen Endgerät auftreten.

Zugriffszeit: Das ist die Verzögerung, die der Bluetooth-Controller hat um Zugriff auf den Kanal zum Senden zu bekommen. Diese Zeit ist nicht deterministische und macht mit Ausnahme von Verzögerungen durch das Betriebssystem den Großteil von t_{ges} aus.

Versandzeit: Die Zeit zum Physischen versenden der Nachricht ist nur abhängig von der verwendeten Hardware und der Länge der Nachricht.

Übertragungszeit: Ist bei gleicher Hardware ausschließlich von der Entfernung abhängig und macht nur einen sehr kleinen teil von t_{ges} aus.

Empfangszeit: Das ist die Zeit für den Empfang der Nachricht und dem Weiterleiten an unsere Applikation. Diese Zeit ist nicht deterministisch und wird stark von Verzögerungen durch das Betriebssystem beeinflusst.

Verarbeitungszeit: Das ist die Zeit, die die Applikation zum verarbeiten der Nachricht benötigt.

Die Reference Broadcast Synchronisation (RBS) wurde 2002 im Artikel "Fine-Grained Network Time Synchronization using Reference Broadcasts"[5] vorgestellt. Die Synchronisation beruht auf der Annahme, dass eine Broadcast Nachricht von allen Empfängern nahezu gleichzeitig empfangen wird. Die Idee ist, dass ein Großteil der Zeit, die eine Nachricht zum versenden benötigt in der Aufbauzeit und der Zugriffszeit liegt. Dementsprechend Empfangen die Empfänger die Broadcast-Nachricht fast gleichzeitig nur mit Unterschieden durch Empfangs- und Verarbeitungszeit.

Bei der RBS werden Broadcast-Nachrichten von einem Sender an beliebig viele Empfänger versendet. Die Broadcast-Nachrichten enthalten im Gegensatz zu anderen Synchronisations-Algorithmen keinen Zeitstempel vom Sender. Stattdessen speichern die Empfänger den Zeitpunkt, an dem sie die Broadcast-Nachricht empfangen haben. Da alle Empfänger die Nachricht fast gleichzeitig erhalten haben können sie die Empfangszeitpunkte als Basis verwendet werden, um die Uhrzeiten zwischen den Empfängern anzupassen.

Die RBS Synchronisiert somit nicht die Uhrzeit vom Sender mit den Uhrzeiten der Empfänger sondern die Uhrzeiten der Empfänger untereinander. Dementsprechend wird bei der RBS immer ein Sender benötigt, der nicht an den Tätigkeiten teilnimmt, für die die Zeitsynchronisation benötigt wurde.

4 Testumgebung

Um die Algorithmen miteinander zu vergleichen soll eine Testumgebung aufgebaut werden. Die Algorithmen sollen auf tatsächlicher Hardware ausgeführt werden und die Präzision der Zeitsynchronisation, sowie der Einfluss von bestimmten externen Faktoren soll gemessen werden.

4.1 Received Signal Strength Indicator (RSSI)

Der RSSI-Wert gibt einen relativen Wert an, wie stark das eingehende Bluetooth-Signal ist. Der RSSI-Wert wird in negativen milli dezimal(dBm) ausgedrückt, wobei der Wert bei einer besseren Verbindung höher ist. Somit ist ein RSSI-Wert von 0dBm das Maximum. Bei den, für die Test verwendeten Geräte hat sich gezeigt, dass der RSSI-Wert zwischen -10dbm und -110dbm liegt. Bei Werten geringer als -110dbm ist die Verbindung so schwach, dass sie häufig abbricht oder garnicht zustande kommt.

Da der RSSI-Wert direkt mit der Stabilität einer Bluetooth-Verbindung zusammenhängt kann der RSSI-Wert Aufschluss darüber geben, wie viele Übertragungsfehler bei einer Verbindung zu erwarten sind. Test bei unterschiedlichen RSSI-Werten können somit Aufschluss darüber geben, wie anfällig die Algorithmen auf Verzögerungen von Retransmissions sind.

4.2 Teilnehmer

Die Testumgebung besteht aus drei Teilnehmern:

- Testprogramm: Das Testprogramm läuft lokal auf den mobilen Endgeräten. Es ist für die Durchführung der Zeitsynchronisation zuständig und implementiert dementsprechend auch die Algorithmen zu Zeitsynchronisation. Da das Testprogramm direkten Zugriff auf die Bluetooth-Verbindung hat, werden auch der RSSI-Wert und die Präzision vom Testprogramm gemessen. Das Testprogramm wurde in Swift geschrieben, da dies die native Programmiersprache für iOS ist und somit die meisten Funktionen der Geräte unterstützt und die beste Performance verspricht.
- Instruments: Instruments läuft auch auf den mobilen Endgeräten und bietet die Möglichkeit die CPU-Auslastung und den Energieverbrauch der mobilen Endgeräte aufzuzeichnen.
- Auswertungsprogramm: Das Testprogramm schreibt die gemessenen Daten in eine CSV-Datei. Das Auswertungsprogramm liest diese Dateien ein und bereitet die Ergebnisse graphisch auf.

4.3 Aufbau vom Testprogramm

Das Testprogramm ist in drei Layer aufgeteilt (Abbildung 4.1). Es gibt den Bluetooth-Layer, der für die gesamte Kommunikation über BLE zuständig. Der UI-Layer ist für die Nutzeroberfläche zuständig. Hier werden die Einstellungen für die Tests vorgenommen, der aktuelle Vortschritt der Tests wird angezeigt und die Messungen werden hier vom Nutzer gestartet. Zwischen den beiden Layern liegt der Control-Layer. Dieser enthält alle Informationen und Ergebnisse vom aktuellen Test und steuert die Einstellungen vom Bluetooth-Layer.

4.3.1 UI

Es wird ein einfaches User Interface bereitgestellt, um Test einzurichten, zu starten und auszuführen. Im User Interface werden die verbundenen Geräte angezeigt mit dem zugehörigem RSSI-Wert, um die Geräte so im Raum zu platzieren, dass die Testvorgaben eingehalten werden.

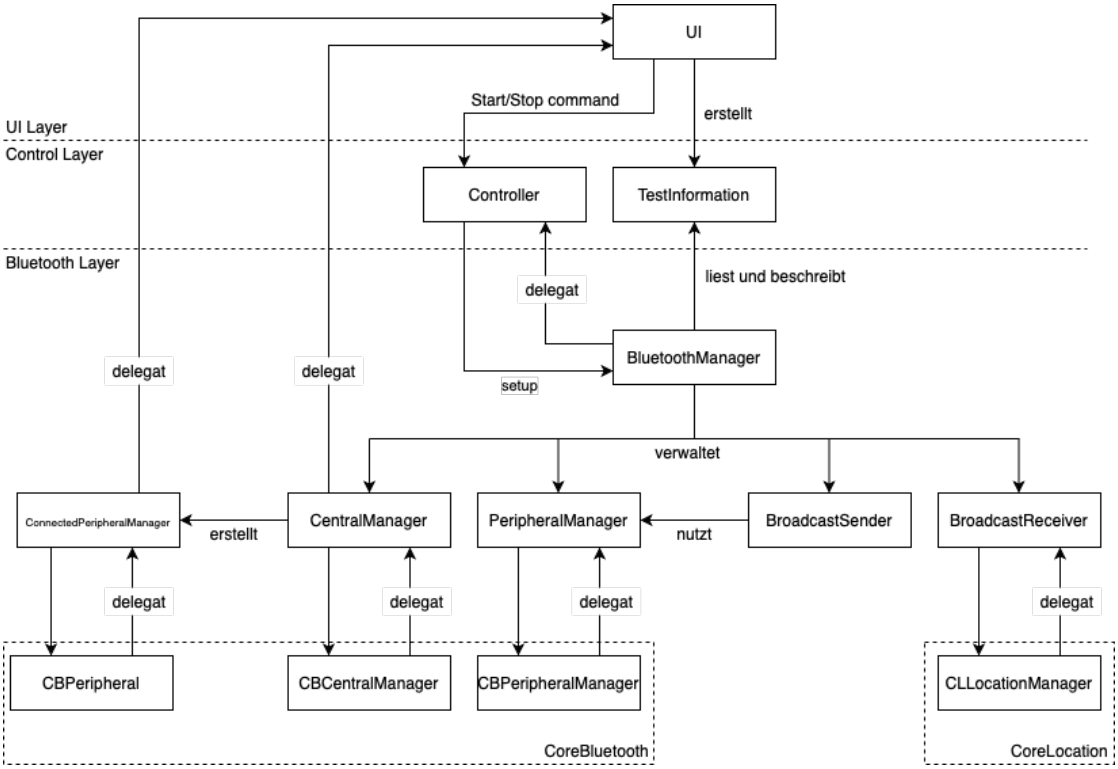


Abbildung 4.1: Aufbau vom Testprogramm

4.3.2 Controller

Der Controller ist dafür zuständig Test und Messungen zu starten. Er erhält vom UI die Eingaben des Nutzers und vom BluetoothManager eine Mitteilung, sobald ein Test oder eine Messung beendet wurden. Des Weiteren ist er dafür zuständig die gemessenen Ergebnisse in eine CSV-Datei zu schreiben.

4.3.3 TestInformation

Die TestInformation beinhaltet alle Informationen von einem Test. Sie beinhaltet:

- die teilnehmenden Geräte
- die Kommunikationesrollen der Geräte
- die Reihenfolge, in der die Geräte die Präzision untereinander messen

- die Anzahl an Wiederholungen, die ein Test durchgeführt werden soll
- die neuen Zeiten vom Zeit-Update
- die gemessenen Präzisionen und RTT-Werte

4.3.4 BluetoothManager

Der BluetoothManager ist dafür zuständig, je nach vorgaben in der TestInformation, entweder den PeripheralManager, CentralManager, BroadcastReceiver oder Broadcast-Sender einzustellen und zu starten.

Er dient als Kommunikationsebene zwischen Control-Layer und Bluetooth-Layer. Er bekommt die Anweisungen vom Control-Layer, welcher Test oder welche Messung durchgeführt werden soll. Vom Bluetooth-Layer bekommt er die Ergebnisse und Bestätigungen, das der Test oder die Messung abgeschlossen wurde. Er schreibt die Ergebnisse in die TestInformation und sorgt dafür das Test, die mehrfach ausgeführt werden sollen, erneut gestartet werden.

Dem Controller teilt der BluetoothManager mit, sobald ein Test oder eine Messung abgeschlossen wurde, sodass der Controller dem BluetoothManager eine neue Anweisung geben kann.

4.3.5 PeripheralManager

Der PeripheralManager ist die Klasse, die direkt mit dem CBPeripherlManager kommuniziert und als Delegate für den CBBluetoothManager agiert. Das Interface vom PeripheralManager beinhaltet drei Funktionen:

- eine um advertising packets für den TimeUpdateService zu senden
- eine um advertising packets mit einer CLBeaconRegion für den Broadcast zu senden
- eine um das Senden von advertising packets zu beenden

Im PeripheralManager wird der TimeUpdateService erstellt und an den CBPeripheralManager übergeben. Es wird die Funktionsweise vom TimeUpdateService sichergestellt und somit auch t_0 und t_1 gemessen. Auch die RTT und die neue Zeit werden vom PeripheralManager berechnet.

Der PeripheralManager teilt dem BluetoothManager zwei Ereignisse mit dem Delegate-Pattern mit:

- wenn der externe Central ein neues Zeit Update initiiert hat
- wenn eine neue Zeit vorliegt mit zugehöriger RTT

4.3.6 CentralManager

Der CentralManager ist die Klasse, die direkt mit dem CBCentralManager kommuniziert und als Delegate für den CBCentralManager agiert. Das Interface vom CentralManager beinhaltet drei Funktionen:

- eine um nach einem Peripheral mit dem TimeUpdateService zu scannen
- eine um das scannen zu beenden
- eine um eine Zeit-Update auf allen verbundenen Peripherals zu initiieren

Der CentralManager ist dafür zuständig Verbindungen mit Peripherals aufzubauen und eine Liste mit verbundenen Peripherals zu speichern. Für jedes verbundene Peripheral erstellt er einen ConnectedPeripheralManager, der die Kommunikation mit dem externen Peripheral übernimmt.

Der CentralManager hat ein UI-Delegate, dem immer eine aktuelle List an verbundenen Peripherals übergeben wird, um diese dem Nutzer anzuzeigen.

4.3.7 ConnectedPeripheralManager

Der ConnectedPeripheralManager ist die Klasse, die direkt mit dem CBPeripheral kommuniziert und als Delegate für das CBPeripheral agiert. Das Interface vom ConnectedPeripheralManager hat zwei funktionen:

- eine um ein Zeit-Update mit dem Peripheral zu initiieren

- eine um den RSSI zum Peripheral zu messen

Der ConnectedPeripheralManager ist dafür zuständig die Abläufe vom Christian Algorithmus und vom modifizierten Christian Algorithmus auf Seiten des Centrals durchzuführen. Dementsprechend sendet er eine Nachricht um ein Zeit-Update zu initiieren und sendet die aktuelle Zeit auf Anfrage an das Peripheral.

Über das Delegate-Pattern wird dem BluetoothManager mitgeteilt, sobald ein Zeit-Update beendet wurde. Über ein UI-Delegate wird dem Nutzer der aktuelle RSSI-Wert angezeigt.

4.3.8 BroadcastReceiver

Der BroadcastReceiver kommuniziert mit dem CLLocationManager um einen Broadcast zu empfangen. Das Interface vom BroadcastManager hat zwei Funtionen:

- eine um nach Broadcasts zu scannen
- eine um das scannen zu beenden

Sobald der BroadcastReceiver einen Neuen Broadcast empfängt, den er noch nicht periodisch vorher empfangen hat, setzt er den aktuellen Zeitpunkt als Start der neuen synchronisierten Zeit.

Dem BluetoothManager wird die neue Zeit über das Delegate-Pattern mitgeteilt.

4.3.9 BroadcastSender

Der BroadcastSender ist dafür zuständig eine CLBeaconRegion zu erstellen, die als Broadcast versendet werden kann. Diese gibt er dann zum senden an den PeripheralManager. Das Interface vom BroadcastSender hat nur zwei Funktionen:

- eine um den Broadcast zu starten
- eine um den Broadcast zu beenden

4.4 Umsetzung der Algorithmen im Testprogramm

4.4.1 Christion Algorithmus

Der Sender nimmt beim Christian Algorithmus die Rolle des Bluetooth Central ein, da dies die Verbindung mit mehreren Empfängern vereinfacht. Dementsprechend sind die Empfänger als Bluetooth Peripheral umgesetzt.

Das bedeutet auch, dass sich der Sender wählt, mit welchen Empfängern er sich verbindet und die Empfänger nicht zwischen mehreren Senderen wählen können. Im Gegensatz zum klassischen Christian Algorithmus wird dementsprechend auch die Zeitsynchronisation nicht vom Empfänger initiiert, sondern vom Sender.

Zur Kommunikation zwischen Empfänger und Sender wird ein BLE Service definiert. Der Service hat zwei Charakteristiken. Eine um eine Zeitsynchronisation vom Sender aus zu starten und eine um die aktuelle Zeit vom Sender an den Empfänger zu senden.

In Abbildung 4.2 ist der Ablauf einer Zeitsynchronisation des Christian Algorithmus im Testprogramm dargestellt. Um die Synchronisation zu starten sendet der Sender eine Startnachricht an die Charakteristik zum starten der Zeitsynchronisation. Diese Nachricht beinhaltet die Information, dass es sich um eine Synchronisation mit dem normalen Christian Algorithmus handelt. Der Empfänger speichert sich daraufhin den aktuellen Zeitstempel T_0 und sendet eine Empfangsbestätigung an den Sender. Sobald der Sender diese Empfangsbestätigung erhalten hat sendet er seine aktuelle Zeit T an die Charakteristik für die aktuell Zeit. Wenn der Sender diese Änderung erhält speichert er wieder seine aktuelle Zeit T_1. Daraufhin wird die neue Zeit vom Empfänger als $T_E = T + \frac{T_1 - T_2}{2}$ gesetzt und die Synchronisation wird mit einer Empfangsbestätigung an den Sender beendet.

4.4.2 Erweiterter Christion Algorithmus

Für den erweiterten Christian Algorithmus nehmen der Sender und die Empfänger die gleichen Bluetoothrollen wie beim normalen Christian Algorithmus ein und es wird auch der gleiche Bluetooth Service mit den selben Charakteristiken verwendet.

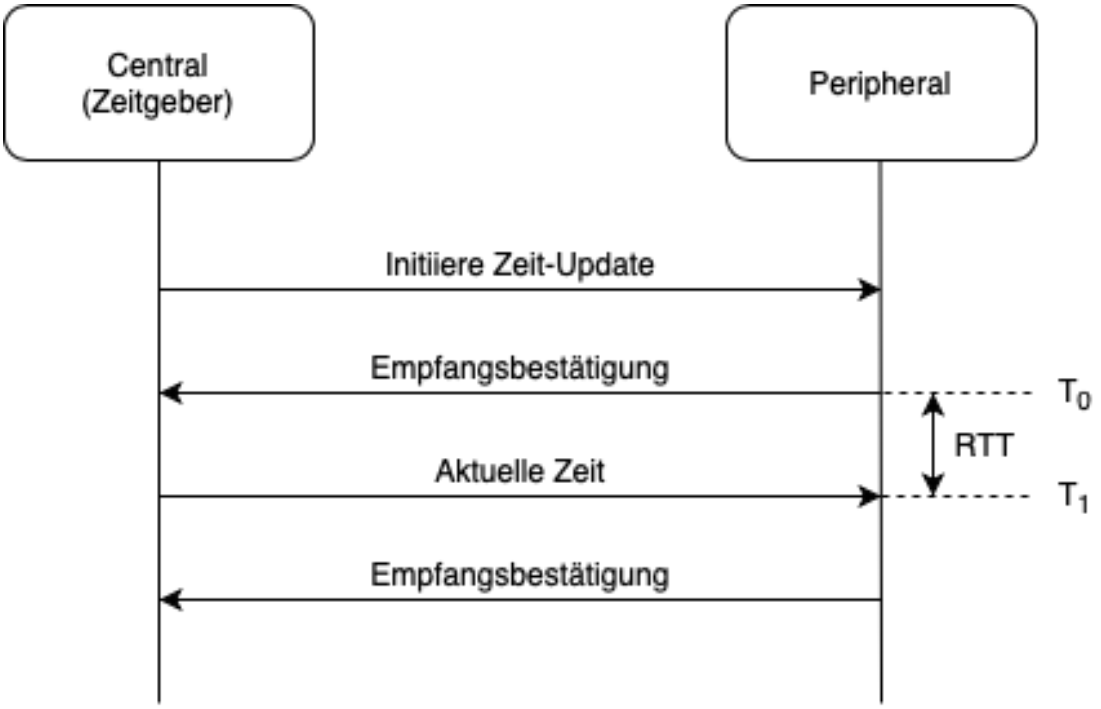


Abbildung 4.2: Umsetzung vom Christian Algorithmus im Testprogramm

Der erweiterte Christian Algorithmus unterscheidet sich somit nur teilweise, im Ablauf und den gesendeten Daten, vom normalen Christian Algorithmus. Diese Änderungen werden in Abbildung 4.3 dargestellt.

Die Startnachricht beinhaltet nun Informationen darüber, wie viele Nachrichten vom Empfänger empfangen werden sollen bis die Zeit des Empfängers geändert wird. Des Weiteren beinhaltet sie die Information um welche Nachricht der aktuellen Synchronisation es sich handelt. Der Empfänger speichert sich nun immer die Zeit bei der $\frac{T_1-T_2}{2}$ am geringsten ist. Der Sender sendet auf die Empfangsbestätigung des Zeitstempels so lange eine neue Startnachricht, bis insgesamt acht Startnachrichten versendet wurden. Nachdem der Empfänger den letzten Zeitstempel erhalten hat, ändert seine aktuelle Zeit, sodass die Zeit verwendet wird, bei der $\frac{T_1-T_2}{2}$ am geringsten war.

4.4.3 Broadcast Synchronisation

Bei der Broadcast Synchronisation nimmt der Sender die Rolle des Bluetooth Peripheral an, da nur ein Bluetooth Peripheral in der Lage ist ein Broadcast zu senden. Die Emp-

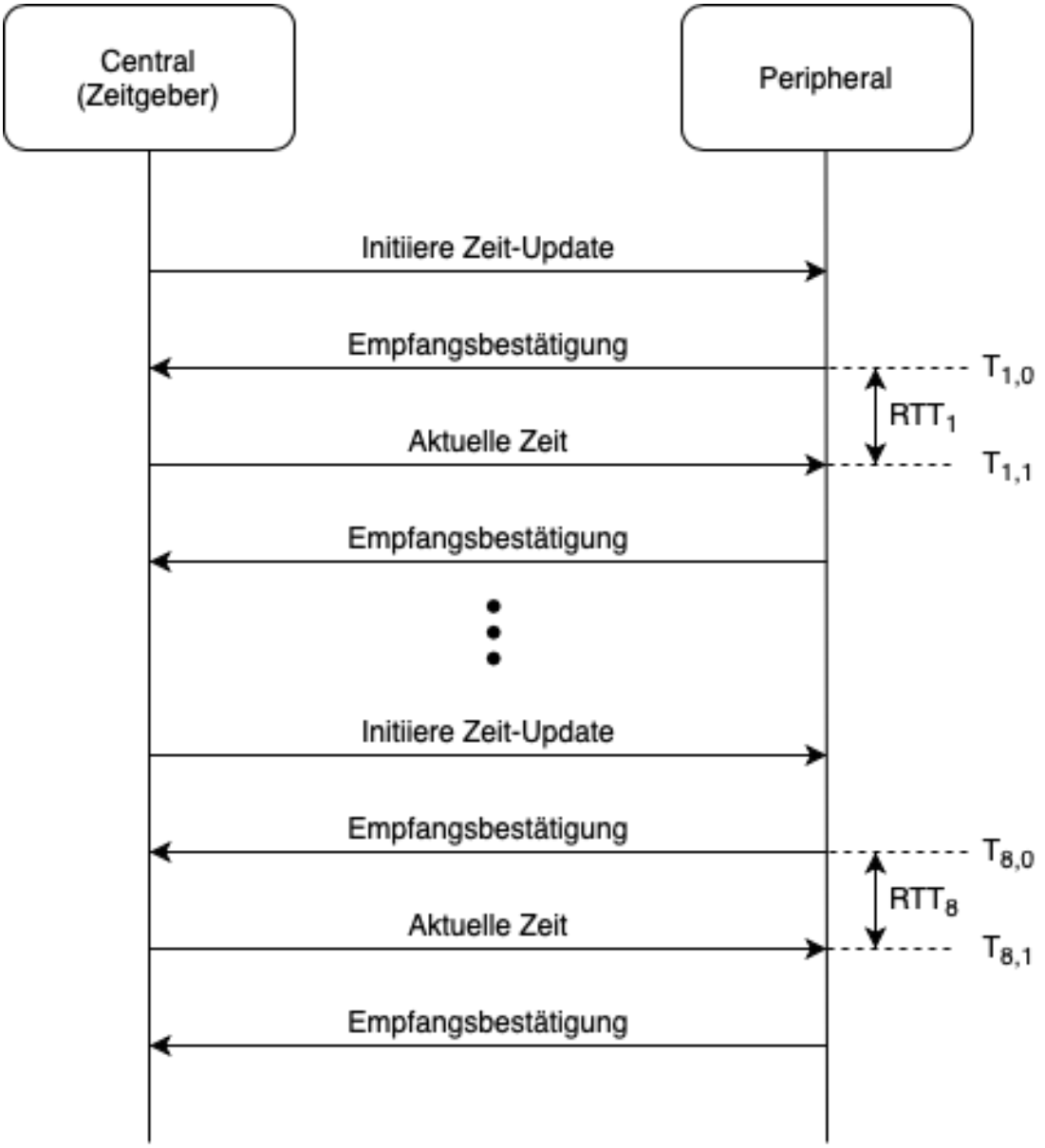


Abbildung 4.3: Umsetzung vom erweiterten Christian Algorithmus im Testprogramm

fänger sind dann dementsprechend Bluetooth Central. Hierbei ist zu beachten, dass nicht ein CBCentralManager verwendet wird sondern CLLocationManager.

Der Zeitpunkt, wann eine Synchronisation stattfinden soll geht vom Sender aus. Dieser sendet für eine kurze Zeit eine Broadcast Nachricht in Form einer CLLocationRegion. Über UUID, Major und Minor wird hierbei definiert, von wem die Nachricht versen-

det wurde. Alle Empfänger die auf diese Kombination aus UUID, Major und Minor hören setzen den Zeitpunkt, an dem sie die Nachricht erhalten haben, als Startpunkt für die synchronisierte Zeit. Eine Synchronisation weitere Kommunikation zwischen den Empfängern ist nicht notwendig, da die Synchronisierte Zeit nicht der tatsächlichen Zeit entsprechen muss sondern nur als Referenz zum Ausführen synchroner Tätigkeiten verwendet wird.

4.5 Messung der Präzision

Um die Genauigkeit und Nutzbarkeit der Algorithmen zu bestimmen soll die Präzision (p), also der Unterschied zwischen den Zeiten auf den verschiedenen Geräten, gemessen werden. Hierzu gibt es die Möglichkeit alle Geräte gleichzeitig eine Aktion ausführen zu lassen und zu messen mit welchem zeitlichen Verzug die Aktion ausgeführt wurde. Die genaue Messung, wann eine Aktion ausgeführt wurde stellt sich allerdings als schwierig heraus. Des weiteren entstehen Ungenauigkeiten durch die Durchführung der Aktionen, die auf unterschiedlichen Geräten unterschiedliche Zeit beanspruchen kann. Auf Grund dessen wird die Präzision mit einer reinen Softwarelösung gemessen. Hierzu wird das gleiche Verfahren verwendet wie auch für die Zeitsynchronisation. Das bedeutet, dass nachdem eine Zeitsynchronisation zwischen zwei Geräten durchgeführt wurde zwei weitere Zeit-Updates zwischen den Geräten durchgeführt werden. Hierbei agiert jedes Gerät einmal als Zeitgeber. Mit diesem Zeit-Update wird allerdings nicht die synchronisierte Zeit (T_S) verändert sondern die Zeit (T_M), die über das Zeit-Update ermittelt wurde, wird mit der synchronisierten Zeit verglichen und es wird die Differenz bestimmt. Diese Differenz stellt nun die Präzision der Zeitsynchronisation dar. $p = |T_S - T_M|$

Der Fehler (e) der Messung kann durch die Differenz der beiden gemessenen Präzisionen bestimmt werden. $e = |p_{A->B} - p_{B->A}|$

Die Präzision zwischen zwei Geräten wird mit dem Mittelwert der beiden Messungen geschätzt: $P = p_{A->B} + \frac{p_{B->A}}{2}$

Um die Präzisions-Messungen zwischen den unterschiedlichen Algorithmen vergleichbar zu machen wird für alle Test der gleiche Algorithmus verwendet um die Zeit-Updates der Messung durchzuführen. Der Broadcast Algorithmus kommt hierbei nicht in Frage, da er nicht in der Lage ist ein Zeit-Update mit vorgegebener Zeit durchzuführen. Da erwartet wird, dass der erweiterte Christian Algorithmus eine bessere Präzision hat, als

der normale Christian Algorithmus werden die Präzisionsmessungen mit dem erweiterten Christian Algorithmus durchgeführt.

4.6 Ablauf eines Tests

Um einen Test durchzuführen muss das Testprogramm auf allen teilnehmenden Geräten installiert werden. Bei allen Geräten müssen nun die Testparameter eingestellt werden. Bei den Parametern handelt es sich um die Rolle, die das Gerät während des Tests einnehmen wird und den Algorithmus, der verwendet wird. Außerdem werden noch Angaben darüber gemacht, um welches Gerät es sich handelt und welche weiteren Geräte an dem Test teilnehmen. Diese Informationen werden nicht für die Zeitsynchronisation benötigt, sondern für die Reihenfolge in der die Messungen durchgeführt werden und als Informationen in der CSV-Datei mit den Ergebnissen.

Daraufhin kann der Test gestartet werden. Der Sender baut eine BLE Verbindung zu den Empfängern auf und misst periodisch die RSSI-Werte. Nun müssen die Geräte so verteilt werden, dass die RSSI-Werte den Vorgaben des Tests entsprechen. Der RSSI-Wert steigt mit größerer Distanz zwischen den Geräten oder Hindernissen, wie Mauern.

Auf allen Geräten wird die Aufzeichnung von Energieverbrauch und CPU Auslastung gestartet. Anschließend kann beim Sender die Synchronisation initiiert werden. Der Sender führt nun 20 Zeitsynchronisationen mit den Empfängern durch. Dies wird gemacht um einzelne Abweichungen der Ergebnisse zu erkennen. Für jede Zeitsynchronisation speichern sich die teilnehmenden Geräte ihre Zeiten.

Sobald die Zeitsynchronisationen abgeschlossen sind müssen die Geräte wieder zusammengebracht werden, um einen möglichst geringen RSSI-Wert bei den Messungen zu haben. Anschließend kann vom Sender die Messung gestartet werden. Der Sender startet nun Messung der Präzision für alle 20 Zeiten. Die Empfänger speichern sich die Ergebnisse. Darauf hin startet nacheinander jeder Empfänger eine Messung, sodass am Ende die Präzision von jedem Gerät zu jedem anderen Gerät für alle 20 Zeiten durchgeführt wurde.

Die Ergebnisse werden in eine CSV-Datei geschrieben und können anschließend in dem Auswertungsprogramm zusammen mit anderen Testergebnissen ausgewertet werden.

5 Evaluation

5.1 Einflussnehmende Faktoren

Es gibt mehrere Faktoren, die das Verhalten der Algorithmen beeinflussen können. Grundsätzlich lassen sich die Faktoren in zwei Kategorien aufteilen.

Zum einen gibt es Faktoren bei der BLE Übertragung. Bei der BLE Übertragung kann es durch starke Auslastung der Kanäle zu Verzögerungen oder Übertragungsfehlern kommen. In den Tests wurde das beachtet, indem die Auslastung vor den Tests gemessen wurde und alle Tests bei einer ähnlichen Auslastung durchgeführt wurden. Insgesamt war die Auslastung während der Test relativ gering und hatte somit nur einen geringen Einfluss auf die Ergebnisse. Die BLE Übertragung wird außerdem durch die Signalstärke beeinflusst. Die Signalstärke ist von der verwendete Hardware, der Übertragungsentfernung und Störobjekten, wie Mauern, abhängig. In den Tests wird die Signalstärke durch den RSSI-Wert gemessen. Der Einfluss der Signalstärke soll außerdem in einem der Tests untersucht werden (siehe 5.2.5).

Als zweites gibt es die Faktoren bei der Verarbeitung auf den Geräten. Hierbei kann es zu Verzögerungen kommen, wenn das Betriebssystem mit anderen Aufgaben beschäftigt ist und somit das Versenden oder Verarbeiten einer Nachricht länger dauert. Um diesen Faktor möglichst gering zu halten werden auf allen Geräten Hintergrundaktivitäten, so weit wie möglich, abgeschaltet. Außerdem wird die CPU Auslastung während der Tests gemessen, um mögliche Unregelmäßigkeiten zu erkennen. Die Geschwindigkeit des Prozessors und der verwendeten Hardware kann auch einen Einfluss haben. Die Tests werden mit unterschiedlichen Geräten durchgeführt. Dementsprechend soll der Einfluss durch die Hardware in einem Test untersucht werden (siehe 5.2.6).

5.2 Tests

5.2.1 Messgenauigkeit

Es soll untersucht werden, wie genau das Verfahren zur Messung der Präzision ist.

Hierzu wird im Testprogramm eine Präzisionsmessung immer einmal von Gerät A zu Gerät B durchgeführt und einmal von Gerät B zu Gerät A. Da es sich jeweils um die gleiche synchronisierte Zeit handelt müssten die Ergebnisse von beiden Messungen im Zusammenhang $P_{A->B} = -P_{B->A}$ stehen. Die Abweichung von diesem Zusammenhang gibt dementsprechend Aufschluss über die Messgenauigkeit. Der Messfehler(E) wird somit mit $|E = P_{A->B} + P_{B->A}|$ berechnet.

Da die Messung mit dem erweiterten Christian Algorithmus durchgeführt wurde wird erwartet, dass der Messfehler ebenfalls von Faktoren, wie der Anzahl an Geräten oder der Diversität an Geräten beeinflusst wird. Der RSSI-Wert sollte keinen Einfluss auf den Messfehler haben, da die Messungen bei allen Tests mit gleichem RSSI-Wert durchgeführt werden.

5.2.2 Präzision bei optimalen Umständen

Um die Algorithmen zu vergleichen sollen sie im einfachsten Fall der Synchronisation verglichen werden. Hierbei werden zwei gleichwertige Geräte, bei einem RSSI-Wert größer als -40dBm, miteinander Synchronisiert. Da die Broadcast Synchronisation erst ab drei Geräten möglich ist, stellen, bei der Broadcast Synchronisation, die beiden gleichwertigen Geräte die Empfänger dar und ein drittes Gerät wird als Sender verwendet.

Bei dem Test wird erwartet, dass der Christian Algorithmus und der erweiterte Christian Algorithmus sehr ähnlich abschneiden. Durch den hohen RSSI-Wertes sollte es zu wenig Übertragungsfehlern kommen. Da es sich zusätzlich um gleichwertige Geräte handelt sollte die RTT keine starken Schwankungen haben, wodurch der Vorteil vom erweiterten Christian Algorithmus entfällt. Der hohe RSSI-Wert und die gleichwertigen Geräte sorgen außerdem dafür, dass die Übertragungszeiten gleichmäßig sind. Dementsprechend sollten die beiden Algorithmen relativ gut abschneiden.

Auch der Broadcast Algorithmus sollte von den gleichwertigen Geräten profitieren, da die Verarbeitung der Nachricht bei den Empfängern so eine ähnliche Zeit betragen sollt.

5.2.3 Unterschied zwischen Sender-Empfänger und Empfänger-Empfänger

Es soll untersucht werden, ob es einen Unterschied gibt zwischen der Präzision bei Sender und Empfänger und bei zwei Empfängern. Da sich beim Broadcast Algorithmus nur die Empfänger miteinander Synchronisieren entfällt dieser bei dem Test.

Bei der Durchführung werden drei Geräte miteinander Synchronisiert. Da keine drei gleichwertigen Geräte zur Verfügung stehen sind nur der Sender und ein Empfänger gleichwertig. Verglichen wird die Präzision des zweiten Empfängers mit jeweils dem anderen Empfänger und dem Sender.

Es wird erwartet, dass die Präzision zwischen Sender und Empfänger etwas genauer ist, da sie sich direkt miteinander Synchronisiert haben. Zwischen den beiden Algorithmus sollte es keine merklichen unterschiede geben.

5.2.4 Einfluss durch Anzahl der teilnehmenden Geräte

Es wird untersucht, ob die Anzahl an Geräten, die an der Zeitsynchronisation teilnehmen, einen Einfluss bei der resultierenden Präzision haben. Hierzu wurden zwei Paarungen an Geräten untersucht. Einmal zwei gleichwertigen Geräten (6s und 6s2) und einmal zwei relative unterschiedliche Geräte (11 und 6). Nun werden drei Test durchgeführt. Beim Ersten werden nur jeweils die beiden Geräte miteinander Synchronisiert. Beim Zweiten nimmt ein weiteres Gerät an der Synchronisation teil und beim dritten werden vier weitere Geräte mit Synchronisiert. Die Tests werden mit allen drei Algorithmen durchgeführt, wobei beim Broadcast Algorithmus der Test mit nur zwei Geräten entfällt.

Um zu schauen, ob der RSSI-Wert einen verstärkenden Einfluss auf die Ergebnisse hat werden alle Test einmal mit einem RSSI-Wert größer als -40dBm durchgeführt und einmal mit einem RSSI-Wert kleiner als -90dBm

Für den Christian Algorithmus und den erweiterten Christian Algorithmus wird erwartet, dass die Präzision mit zunehmender Teilnehmeranzahl schlechter wird, da die Anzahl an Nachrichten, die der Sender versenden muss, mit der Anzahl an Teilnehmern wächst. Dementsprechend kann es zu größeren Verzögerungen beim Senden der Nachrichten kommen, wodurch die Annahme, dass die halbe RTT die Zeit zwischen Senden der Uhrzeit und Empfangen der Uhrzeit darstellt, schlechter wird.

Der Broadcast Algorithmus sollte von einer hören Anzahl an Geräten nicht beeinflusst werden, da weiterhin nur eine Broadcast-Nachricht versendet wird.

5.2.5 Einfluss vom RSSI-Wert

Es soll untersucht werden, welchen Einfluss der RSSI-Wert, zwischen Sender und Empfänger, auf die Präzision der Zeitsynchronisation hat. Hierzu werden jeweils 20 Zeitsynchronisationen zwischen den gleichen Geräten durchgeführt. Einmal mit einem RSSI-Wert größer als -40 dBm und einmal einem RSSI-Wert kleiner als -90 dBm. Bei den Zeitsynchronisationen, bei denen mehr als zwei Geräte teilnehmen, wird ein weiterer Test durchgeführt. Hier hat ein Gerät einen RSSI-Wert größer als -40 dBm hat und das andere Gerät einen RSSI-Wert kleiner als -90 dBm.

Es wird erwartet, dass es bei kleinen RSSI-Werten zu mehr Fehlern bei der Übertragung kommt und somit die Präzision sich verschlechtert. Der erweiterte Christian Algorithmus wird vermutlich bei niedrigen RSSI-Werten deutliche besser abschneiden als der normale Christian Algorithmus, da das Ergebnis mit der geringsten RTT vermutlich auch die wenigsten Fehler bei der Übertragung hatte.

5.2.6 Einfluss durch unterschiedliche Geräte

Es wird untersuchen, ob es einen Unterschied in der Präzision gibt, je nachdem wie wie unterschiedlich die sich synchronisierenden Geräte sind. Hierzu werden zwei Paare von Geräten betrachtet. Einmal zwei gleichwertige Geräte (6s und 6s2) und einmal zwei sehr unterschiedliche Geräte (11 und 6). Die Geräte werden bei unterschiedlichen RSSI-Werten und unterschiedlicher Anzahl teilnehmender Geräte miteinander Synchronisiert und es wird untersucht, ob es unterschiede zwischen den beiden Paaren gibt.

Voraussichtlich werden die gleichwertigen Geräte insgesamt auf eine bessere Präzision kommen, da das Verarbeiten von BLE Nachrichten eine ähnlichere Zeit beanspruchen sollet, als bei unterschiedlichen Geräten. Dies sorgt dafür, das die halbe RTT ehr der Zeit einer Übertragung beträgt. Auch eine Broadcast Nachricht sollte in einer ähnlicheren Zeit verarbeitet werden. Somit profitiert auch die Broadcast Synchronisation von gleichwertigen Geräten.

5.3 Ergebnisse

5.3.1 Messgenauigkeit

Im Durchschnitt über allen durchgeführten Test hat der Messfehler 2,2ms betragen. Die Anzahl an teilnehmenden Geräten hatte keinen erkennbaren Einfluss auf die Messgenauigkeit. Genauer wird dies nochmal in Kapitel 5.3.4 betrachtet.

Welche Geräte die Messung miteinander durchführen hat auch einen Einfluss auf die Messgenauigkeit. Der Messfehler reicht von durchschnittlich 1,2ms zwischen 6s und 6s2 zu durchschnittlich 3,5ms zwischen 6 und Pro. In Tabelle 5.1 ist die Präzision für alle Paarungen aufgelistet. Der Einfluss von den unterschiedlichen Geräten wird in Kapitel 5.3.6 genauer betrachtet.

Paarung	Messfehler
11-6s	2,5ms
11-6s2	2,7ms
11-6	2,2ms
11-iPad	3,3ms
11-Pro	2,5ms
6s-6s2	1,2ms
6s-6	1,8ms
6s-iPad	2,8ms
6s-Pro	3,2ms
6s2-6	2,0ms
6s2-iPad	1,9ms
6s2-Pro	2,5ms
6-iPad	2,0ms
6-Pro	3,5ms
iPad-Pro	2,7ms

Tabelle 5.1: Auflistung der durchschnittlichen Messfehler für alle Gerätekombinationen

5.3.2 Präzision der Algorithmen

In Abbildung 5.1 werden die Algorithmen im einfachsten Fall der Synchronisation verglichen.

Die angegebenen Präzisionswerte stellen immer den Mittelwert zwischen den beiden Präzisionsmessungen, die zwischen zwei Geräten durchgeführt wurden, da. Über die jeweils

20 Messungen, die mit gleichen Voraussetzungen durchgeführt wurden wird ein Mittelwert gezogen und das 95% Konfidezintervall wird berechnet um die Variabilität in den Ergebnissen darzustellen.

Beim vergleich der Ergebnisse fällt auf, dass der Christian Algorithmus und der erweiterte Christian Algorithmus relativ ähnliche Ergebnisse aufweisen, mit Präzisionen im geringen einstelligen Millisekundenbereich. Das 95%-Konfidenzintervall und somit die Variabilität der Ergebnisse ist beim erweiterten Christian Algorithmus allerdings nochmal circa 50% kleiner als beim Christian Algorithmus.

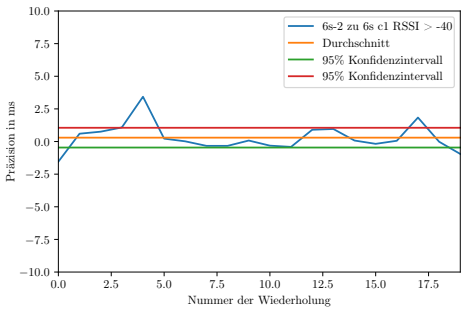
Die Broadcast Synchronisation hingegen zeigt deutlich schlechtere Ergebnisse mit Präzisionen, die bis zu zwei Größenordnungen schlechter sind. Der deutliche Unterschied zwischen Broadcast Synchronisation und den anderen Algorithmen ist in allen Test festzustellen und lässt darauf schließen, dass die Annahme der Broadcast Synchronisation fehlerhaft ist oder, dass durch die Umsetzung mit den iBeacons nicht deterministische Verzögerungen entstehen. Die Schwankungen zwischen einzelnen Messungen sind so groß, dass die einflussnehmenden Faktoren keinen merklichen Einfluss auf das gemessene Ergebnis haben. Dementsprechend wird die Broadcast Synchronisation in der Analyse der einflussnehmenden Faktoren weitgehend vernachlässigt.

5.3.3 Unterschied zwischen Sender-Empfänger und Empfänger-Empfänger

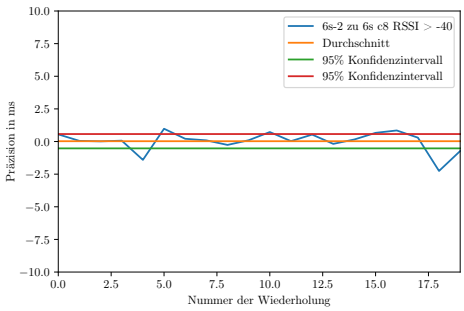
In den Ergebnissen ist kein Unterschied festzustellen bei der Präzision zwischen dem Sender und einem Empfänger und zwischen zwei Empfängern. Falls es einen Unterschied gibt ist dieser im Bereich der Messungenauigkeit und kann dementsprechend vernachlässigt werden.

5.3.4 Einfluss durch Anzahl der teilnehmenden Geräte

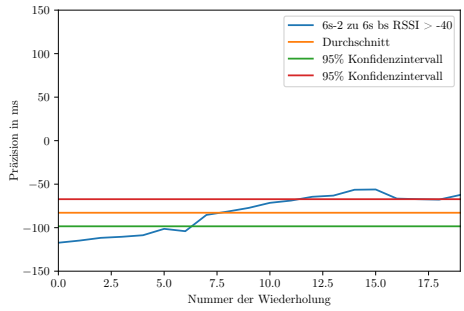
In Abbildung 5.2 sieht man, dass beim normalen Christian Algorithmus mit zunehmender Teilnehmeranzahl eine klare Verschlechterung der Präzision stattfindet. Bei der Paarung mit gleichwertigen Geräten (6s und 6s2) ist die Präzision, ob zwei oder drei Geräte an der Synchronisation teilgenommen haben relativ identisch. Bei sechs teilnehmenden Geräten verschlechtert sich die mittlere Präzision allerdings deutlich, von ca. 2,5 ms auf über 10 ms. Das 95% Konfidenzintervall erweitert sich um den Faktor drei. Bei



(a) Sender: 6s, Empfänger: 6s2
Christian Algorithmus



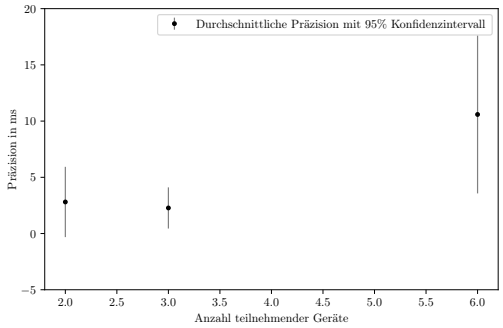
(b) Sender: 6s, Empfänger: 6s2
Erweiterter Christian Algorithmus



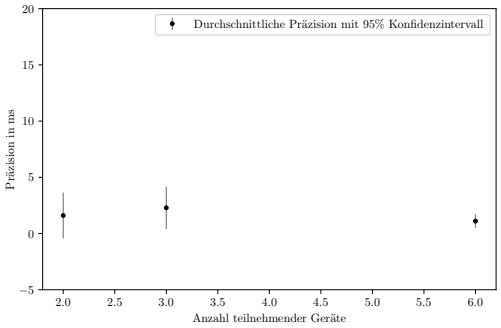
(c) Sender:11, Empfänger: 6s, 6s2
Broadcast Synchronisation

Abbildung 5.1: Verglichen wird die Präzision der unteschiedlichen Algorithmen. Die Messungen wurden bei einem hohen RSSI, größer als -40 dBm, durchgeführt. Es wird die Präzision von dem Geräte 6s2 zum Gerät 6s dargestellt.

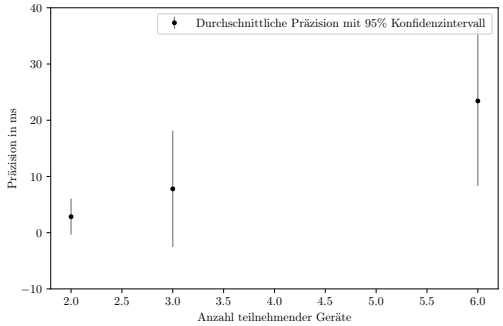
der Synchronisation von zwei unterschiedlichen Geräten (11 und 6) gibt es bereits eine starke Verschlechterung zwischen der Teilnahme von zwei und der Teilnahme von drei Geräten. Die mittlere Präzision bei zwei Geräten liegt noch bei 2 ms mit einem relativ geringen 95% Konfidenzintervall. Bei drei Geräten liegt die mittlere Präzision bereits bei über 7 ms mit einem mehr als doppelt so großen 95% Konfidenzintervall und bei sechs Geräten verschlechtert sich die Präzision weiter auf über 20 ms mit einem 95% Konfidenzintervall dreimal höher als bei zwei Geräten. Des weiteren fällt auf, dass bei drei und sechs Teilnehmenden Geräten die Präzision zwischen den unterschiedlichen Geräten (11 und 6) deutlich schlechter ist als die Präzision zwischen den gleichwertigen Geräten (6s und 6s2).



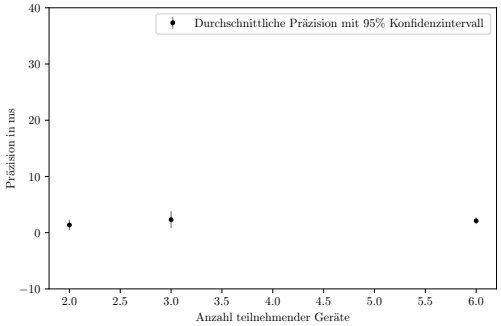
(a) Präzision zwischen 6s und 6s2 beim Christian Algorithmus



(b) Präzision zwischen 6s und 6s2 beim erweiterten Christian Algorithmus



(c) Präzision zwischen 11 und 6 beim Christian Algorithmus

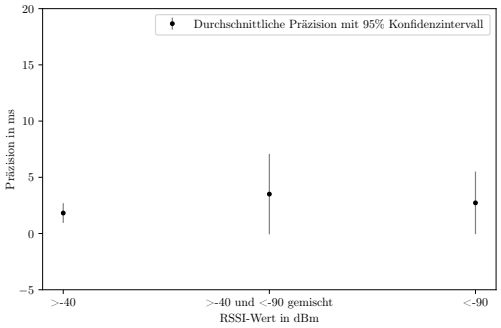


(d) Präzision zwischen 11 und 6 beim erweiterten Christian Algorithmus

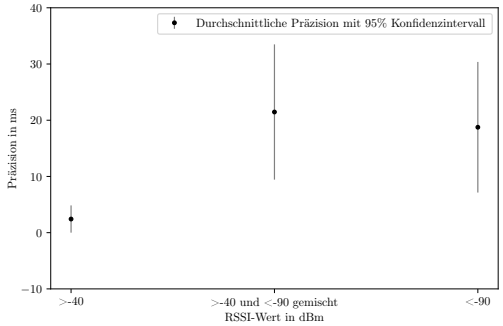
Abbildung 5.2: Dargestellt ist jeweils die durchschnittliche Präzision mit zugehörigem 95% Konfidenzintervall. Der Durchschnitt setzt sich zusammen aus 20 Messungen mit einem RSSI-Wert kleiner als 40 und 20 Messungen mit einem RSSI-Wert größer als 90. Es wird immer die gleiche Paarung aus zwei Geräten verglichen mit unterschiedlich vielen weiteren Testteilnehmern.

Beim erweiterten Christian Algorithmus ist keine Verschlechterung der Präzision mit zunehmender Teilnehmeranzahl festzustellen. Die mittlere Präzision ist immer bei ca. 2 ms mit einem kleinen 95% Intervall. Ein unterschied zwischen den gleichwertigen Geräten und den unterschiedlichen Geräten ist beim erweiterten Christian Algorithmus auch nicht festzustellen.

(a) Präzision zwischen 6s und 6s2 mit zwei teilnehmenden Geräten



(b) Präzision zwischen 6s und 6s2 mit drei teilnehmenden Geräten



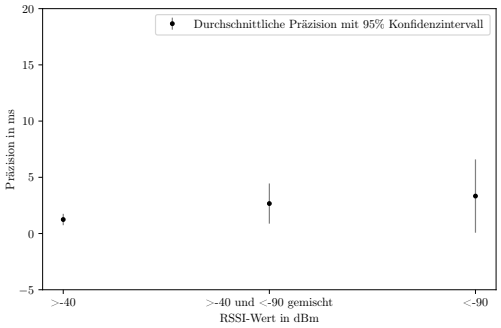
(c) Präzision zwischen 6s und 6s2 mit sechs teilnehmenden Geräten

Abbildung 5.3: Die Präzision bei unterschiedlich hohem RSSI wird beim Christian Algorithmus verglichen. Es wird immer die Präzision vom Gerät 6s2 zum Gerät 6s dargestellt. Zusätzlich wird die Anzahl an teilnehmenden Geräten variiert. Der RSSI ist in drei Bereiche aufgeteilt: größer als -40 dBm, kleiner als -90 dBm und wenn ein Gerät im ersten RSSI-Bereich ist und das Zweite im zweiten RSSI-Bereich.

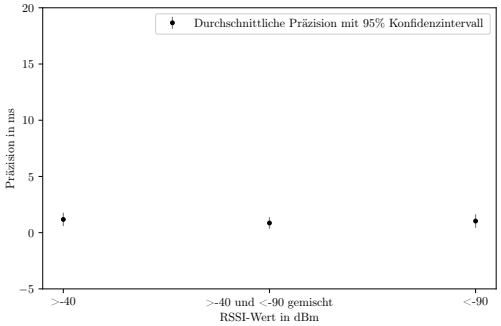
5.3.5 Einfluss vom RSSI-Wert

In Abbildung 5.3 und 5.4 werden die Ergebnisse der Versuch dargestellt. Beim normalen Christian Algorithmus ist ein klarer unterschied zwischen der Präzision bei einem RSSI-Wert größer als -40 dBm und der Präzision bei einem RSSI-Wert kleiner als -90 dBm festzustellen. Bei den Tests mit zwei und drei, an der Synchronisation teilnehmenden Geräten, hat sich die mittlere Präzision um bis zu 4 ms verschlechtert und das 95% Konfidenzintervall hat sich von weniger als 2 ms auf 5-10 ms vergrößert. Bei sechs Teilnehmenden Geräten wurde eine deutlich stärkere Verschlechterung festgestellt. Hier ist die mittlere Präzision um das zehnfache auf ca. 20 ms gestiegen und das 95% Konfi-

(a) Präzision zwischen 6s und 6s2 mit zwei teilnehmenden Geräten



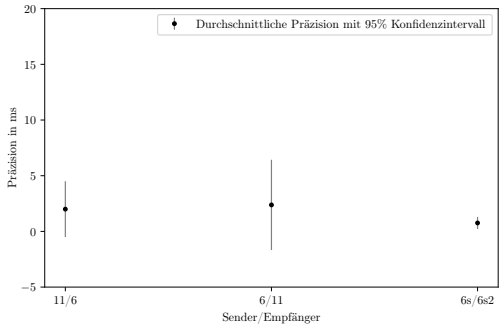
(b) Präzision zwischen 6s und 6s2 mit drei teilnehmenden Geräten



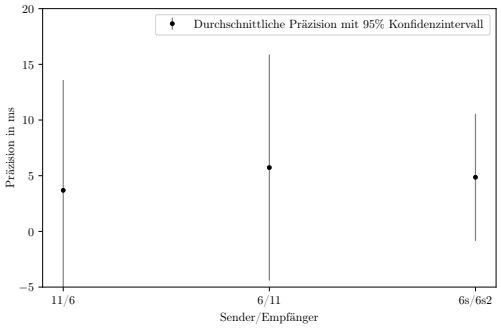
(c) Präzision zwischen 6s und 6s2 mit sechs teilnehmenden Geräten

Abbildung 5.4: Die Präzision bei unterschiedlich hohem RSSI wird beim erweiterten Christian Algorithmus verglichen. Es wird immer die Präzision vom Gerät 6s2 zum Gerät 6s dargestellt. Zusätzlich wird die Anzahl an teilnehmenden Geräten variiert. Der RSSI ist in drei Bereiche aufgeteilt: größer als -40 dBm, kleiner als -90 dBm und wenn ein Gerät im ersten RSSI-Bereich ist und das Zweite im zweiten RSSI-Bereich.

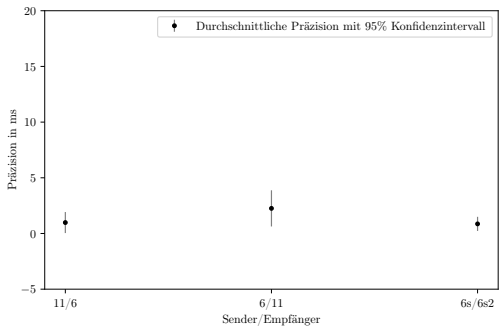
denzintervall ist um einen ähnlichen Faktor gewachsen. Des weiteren ist auffällig, dass die Verschlechterungen ungefähr gleich sind, egal ob eins der beiden Geräte oder beide Geräte einen RSSI-Wert kleiner als -90 dBm hatten. Beim erweiterten Christian Algorithmus ist auch eine Verschlechterung bei erhöhten RSSI-Werten festzustellen. Diese ist aber etwas geringer als beim normalen Christian Algorithmus. Auffällig ist, das bei sechs teilnehmenden Geräten keine Verschlechterung festgestellt werden konnte, obwohl dort beim normalen Christian Algorithmus die stärkste Verschlechterung festgestellt wurde.



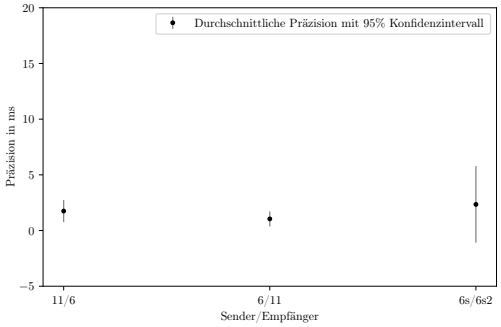
(a) Präzision beim Christian Algorithmus und einem RSSI größer als -40 dBm



(b) Präzision beim Christian Algorithmus und einem RSSI kleiner als -90 dBm



(c) Präzision beim erweiterten Christian Algorithmus und einem RSSI größer als -40 dBm



(d) Präzision beim erweiterten Christian Algorithmus und einem RSSI kleiner als -90 dBm

Abbildung 5.5: Es werden drei Gerätepaare miteinander verglichen. Das zuerst aufgeführte Gerät ist jeweils der Sender. Es wurde jeweils eine Messung bei nur zwei teilnehmenden Geräten durchgeführt.

5.3.6 Einfluss durch unterschiedliche Geräte

In Abbildung 5.5 wird der Einfluss von unterschiedlichen Geräten auf die Präzision dargestellt. Beim einfachen Christian Algorithmus ist festzustellen, dass zwei gleichwertige Geräte eine bessere Präzision haben, als zwei sehr unterschiedliche Geräte. Besonders beim 95% Konfidenzintervall ist ein deutlicher Unterschied von mehreren Millisekunden festzustellen. Außerdem macht es einen Unterschied welches der Geräte als Sender agiert. Wenn das schwächere Gerät als Sender agiert ist auch die Präzision schlechter. Der RSSI-Wert hat hingegen keinen verstärkenden Einfluss auf die unterschiedlichen Geräte. Der RSSI-Wert sorgt bei allen Gerätepaaren für eine ungefähr gleiche Veränderung.

Beim erweiterten Christian Algorithmus ist keine Veränderung der Präzision durch die unterschiedlichen Geräte festzustellen. Die Unterschiede sind so gering, dass sie innerhalb der Messungenauigkeit sind.

5.3.7 Bewertung der Algorithmen

In den Tests hat sich gezeigt, dass die Präzision beim erweiterten Christian Algorithmus am genausten ist. Die Präzision war in den allen Tests unter 5ms. Beim normalen Christian Algorithmus ist die Präzision teilweise auf über 20ms gestiegen und bei der Broadcast Synchronisation war sie bei mehreren hundert Millisekunden. Der erweiterte Christian Algorithmus benötigt allerdings acht mal so viele Nachrichten wie der normale Christian Algorithmus und dementsprechend ist auch der Versatz zwischen der Anfrage, dass die Zeit synchronisiert werden soll, und der fertiggestellten Synchronisation acht mal so lang. Bei einer Synchronisation zwischen zwei Geräten beträgt diese Zeit 12 ms beim normalen Christian Algorithmus und 75 ms beim erweiterten Christian Algorithmus. Die Broadcast Synchronisation ist in diesem Punkt sogar noch besser als der normale Christian Algorithmus, da bei der Broadcast Synchronisation nur eine Nachricht versendet wird. Die genaue Zeit zwischen Synchronisationsanfrage und fertiger Synchronisation kann bei der Broadcast Synchronisation nicht gemessen werden, da es kein Feedback von den Empfänger Geräten zum Sender Gerät gibt. Es ist allerdings davon auszugehen, dass die Zeit geringer ist als die vom Christian Algorithmus. Eine Einschränkung der Broadcast Synchronisation ist allerdings, dass eine Synchronisation nur circa alle 13 Sekunden durchgeführt werden kann, da CoreBluetooth erst nach circa 10 Sekunden erkennt, dass ein Beacon nicht mehr erreichbar ist.

Die CPU-Auslastung und der Energieverbrauch skalieren auch mit der Anzahl an gesendeten Nachrichten. Sie sind allerdings so gering, dass sie in den Tests und den meisten Anwendungen zu vernachlässigen sind.

6 Fazit

Die durchgeführten Tests haben gezeigt, dass eine Zeitsynchronisation zwischen mobilen Endgeräten über BLE grundsätzlich möglich ist. Unter den meisten Umständen ist eine Präzision von weniger als 5ms konstant zu erreichen.

Der normale Christian Algorithmus ist bereits in der Lage eine gute Zeitsynchronisation mit sehr geringem Aufwand herzustellen. Er ist allerdings nicht sehr robust gegenüber Übertragungsfehlern. Sobald die Geräte etwas weiter voneinander entfernt liegen kommt es zu einer starken Verschlechterung der Präzision. Außerdem verschlechtert sich die Präzision mit der Anzahl an teilnehmenden Geräten.

Der erweiterte Christian Algorithmus benötigt achtmal mehr Nachrichten als der normale Christian Algorithmus ist aber deutlich toleranter gegenüber Fehlern. Die Präzision verschlechtert sich zwar auch bei einem hören Abstand zwischen Geräten allerdings deutlich weniger als beim normalen Christian Algorithmus. Es ist davon auszugehen, dass eine höre Anzahl an teilnehmenden Geräten die Präzision verschlechtert. Bei den Tests mit bis zu sechs Geräten ist dies allerdings nicht aufgefallen.

Die Broadcast Synchronisation hat in den Test deutliche schlechter abgeschnitten. Sie hat den Vorteil, dass die Anzahl an teilnehmenden Geräten keinen Einfluss auf die Performance hat, da immer nur eine Nachricht versendet werden muss. Die Tests haben allerdings Gezeigt, dass die Präzision im Gegensatz zu den anderen Algorithmen um bis zu zwei Größenordnungen schlechter ist.

Dementsprechend ist in den meisten Fallen der erweiterte Christian Algorithmus zu empfehlen.

Weiterführend sollte der erweiterte Christian Algorithmus noch einmal in größeren Netzwerken untersucht werden, da bei sechs teilnehmenden Geräten noch keine merkliche Verschlechterung festzustellen war. Außerdem sollte untersucht werden wie gut das Zusammenspiel mit anderen Betriebssystemen funktioniert. Die versendeten Nachrichten können bereits von anderen Betriebssystemen empfangen werden.

Literaturverzeichnis

[1] CoreBluetooth. In: *Apple Developer Documentation*

[2] Turning an iOS Device into an iBeacon Device. In: *Apple Developer Documentation*

[3] Christian, F.: Turning an iOS Device into an iBeacon Device. In: *Apple Developer Documentation* 3 (1989), 09, S. 146–158

[4] Dian, F. J. ; Yousefi, A. ; Somaratne, K.: A study in accuracy of time synchronization of BLE devices using connection-based event. In: *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2017, S. 595–601

[5] Elson, J. ; Girod, L. ; Estrin, D.: Fine-Grained Network Time Synchronization using Reference Broadcasts. In: *ACM SIGOPS Operating Systems Review* 36 (2002), 06

[6] Mills, D. L.: Network Time Protocol (NTP). (1985), 09

[7] Sridhar, S. ; Misra, P. ; Gill, G. S. ; Warrior, J.: Cheepsync: a time synchronization service for resource constrained bluetooth le advertisers. In: *IEEE Communications Magazine* 54 (2016), Nr. 1, S. 136–143

[8] Townsend, K. ; Davidson, R. ; Cufí, C.: *Getting Started with Bluetooth Low Energy*. O'Reilly, 2014. – ISBN 9781491949511

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Zeitsynchronisation von mobilen Endgeräten mit Bluetooth Low Energy

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	[redacted]
Ort	Datum	Unterschrift im Original