



hochschule mannheim

**Implementierung einer gestenbasierten
Interaktion zum Datenaustausch zwischen
mobilen Endgeräten**

Benjamin Grab

Bachelor-Thesis
zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.)
Studiengang Informatik

Fakultät für Informatik
Hochschule Mannheim

04.03.2015

Betreuer
Prof. Kirstin Kohler, Hochschule Mannheim
Horst Schneider, B. Sc, Hochschule Mannheim

Grab, Benjamin:

Implementierung einer gestenbasierten Interaktion zum Datenaustausch zwischen mobilen Endgeräten / Benjamin Grab. –

Bachelor-Thesis, Mannheim : Hochschule Mannheim, 2015. 54 Seiten.

Grab, Benjamin:

Implementation of a Gesture Based Interaction to Transfer Data Between Mobile Devices / Benjamin Grab. –

Bachelor-Thesis, Mannheim : University of Applied Sciences Mannheim, 2015. 54 pages.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mannheim, 04.03.2015

Benjamin Grab

Abstract

Implementierung einer gestenbasierten Interaktion zum Datenaustausch zwischen mobilen Endgeräten

Anwender benutzen eine Vielzahl von mobilen Geräten, wie Smartphones und Tablets, um ihren täglichen Aktivitäten nachzugehen. Dabei wechseln Sie täglich mehrmals zwischen ihren Geräten, abhängig von Tätigkeit und Kontext. Bei diesem Wechsel wird jedoch der Fortschritt, der durchgeführten Tätigkeit, vom Ursprungsgerät nicht auf das Folgegerät übertragen. Der Anwender muss manuelle Maßnahmen zum Datentransfer zwischen seinen Endgeräten durchführen, die ihn von seiner eigentlichen Tätigkeit abhalten. Gestensteuerung bietet großes Potenzial, eine einfache Interaktion für den Anwender zu bieten, mit der Daten übertragen werden können. Diese Arbeit untersucht deshalb die gestenbasierte Bump-Interaktion, bei der durch einfaches Anstoßen eines Endgeräts Daten auf ein anderes zur Synchronisierung übertragen werden. Im Rahmen dieser Arbeit wird diese Geste analysiert, verschiedene Konzepte für eine Umsetzung erarbeitet, Vor- und Nachteile der Entwürfe dargestellt und einer der Entwürfe prototypisch umgesetzt.

Implementation of a Gesture Based Interaction to Transfer Data Between Mobile Devices

On a daily basis people use several mobile devices, like Smartphones or Tablets, to fulfil all kinds of tasks. While doing so they often switch from one device to another to leverage all the advantages of their device like screen size. When the user switches his device while performing some task he has to manually transfer data to the new device, to continue the task, he has begun on the previous device. For the user this manual data transfer is tedious and keeps him from doing, what he actually wants to do. He just wants to continue, where he left off, when he switches devices. Gesture based interactions can be used, to solve this problem. With the bump-gesture there is already a known interaction which can be used to transfer data between devices. This thesis discusses design and development of the bump gesture. It analyses the bump-gesture, shows several system designs, discusses advantages and disadvantages of the designs and implements a prototype Application based on one of the designs.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Problemstellung	1
1.2 Ziel der Arbeit	2
1.3 Aufbau der Arbeit	2
2 Stand der Technik	5
2.1 Grundlagen zum gestenbasierten Transfer von Daten	5
2.2 Technische Grundlagen	9
2.2.1 Technologien zur Datenübertragung	9
2.2.2 Datenkommunikation zwischen mobilen Endgeräten	12
2.2.3 Technologien zur Geräteortung	15
2.2.4 Sensoren	19
3 Analyse der Bump-Geste	21
3.1 Ablauf der Interaktion	21
3.1.1 Aktionen des Benutzers	21
3.1.2 Reaktion des Systems	22
3.2 Variationen der Bump-Geste	22
3.3 Geeignete Endgeräte	24
3.4 Umsetzung der Interaktion	25
4 Umsetzung der Interaktion	27
4.1 Grundlagen zu iOS Frameworks	27
4.1.1 Core Motion Framework	27
4.1.2 Core Location Framework	28
4.1.3 Multipeer Connectivity Framework	29
4.2 Konzept zur Bump-Erkennung	32
4.2.1 Versuchsreihe Bump-Mustererkennung	32
4.2.2 Algorithmus zur Mustererkennung	34
4.3 Konzepte zur Identifizierung von Bump-Partnern	36
4.3.1 Identifizierung über iBeacon Geräte-IDs	36
4.3.2 Identifizierung durch charakteristische Bump-Daten	36
4.3.3 Vergleich der Konzepte	37

Inhaltsverzeichnis

4.4 Konzepte zum Datenaustausch	38
4.4.1 Datenaustausch über lokale Netzwerke	38
4.4.2 Datenaustausch über Webserver	45
4.4.3 Vergleich der Konzepte	48
5 Demonstrator Applikation	49
6 Zusammenfassung und Ausblick	53
Abkürzungsverzeichnis	vii
Tabellenverzeichnis	ix
Abbildungsverzeichnis	xi
Literaturverzeichnis	xiii

Kapitel 1

Einleitung

1.1 Problemstellung

Immer mehr Anwender nutzen nicht mehr exklusiv den Desktop PC, sondern eine Vielzahl an Endgeräten um ihren täglichen Aktivitäten nachzugehen. Abhängig vom Kontext, das heißt von Ziel, Aufenthaltsort und Zeitpunkt, entscheiden Anwender, welche Geräte sich für eine Situation am besten eignen und treffen entsprechend eine Auswahl [Goo12]. Dabei kann es vorkommen, dass zur Erledigung einer Aufgabe, mehrere Geräte gleichzeitig oder aufeinanderfolgend genutzt werden. Dadurch lassen sich die Vorteile der verschiedenen Geräte, wie Bildschirmgröße, Mobilität oder Interaktionsmöglichkeiten ausnutzen. Nach einer Studie von Google ist dieses multi-screening bei einem Großteil der Anwender zu beobachten. So beginnen beispielsweise 90% der in der Studie befragten Personen eine Aktivität auf einem Gerät und führen diese zu einem späteren Zeitpunkt auf einem anderen Gerät fort [Goo12]. Für den Anwender ist diese vermischte Nutzung von Endgeräten somit ein alltäglicher Vorgang. Damit ein Anwender seine Aktivität bei einem Gerätewechsel fortsetzen kann, müssen häufig erst Daten zwischen den Geräten ausgetauscht werden. Dieser Datenaustausch geschieht häufig nicht automatisch und erfordert vom Anwender eine Reihe von komplizierten Interaktionen zum Datentransfer zwischen den Geräten. Für den Anwender stellt dies eine Störung seines Arbeitsablaufes dar. Er erwartet fließend zwischen Geräten wechseln zu können da er seine Aktivität über Gerätekanten hinaus als zusammengehörig empfindet.

Mit Gestensteuerung können Interaktionen verwirklicht werden, die über Gerätekanten hinaus wirken. Somit kann der für den Anwender aufwendige Interaktionen zur Datenübertragung beim Gerätewechsel durch eine einzelne intuitive Interaktion

durchgeführt werden. Ein vielversprechender Kandidat für solch eine Interaktion ist die Bump-Geste. Die Bump-Geste ist eine Interaktion, bei der Daten zwischen Endgeräten ausgetauscht werden indem diese sich gegenseitig anstoßen (bumpen). Anwender, die ihr Gerät wechseln, wollen mit dem Gerät, auf dem die Aktivität begonnen wurde, das Folgegerät bumpen und damit alle notwendigen Daten übertragen, um ihre Aktivität sofort auf dem neuen Gerät weiterzuführen zu können. Betrachtet ein Anwender z.B. ein Photoalbum auf seinem Smartphone und möchte diese Aktivität auf seinem Tablet fortsetzen, muss er nur das Smartphone an das Tablet bumpen. Anschließend wird das Photoalbum auf das Tablet übertragen und die Applikation navigiert automatisch zu Photoalbum und Bild, bei dem die Aktivität auf dem Smartphone beendet wurde.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist es, verschiedene Systementwürfe zu identifizieren und zu beschreiben, mit denen die Bump Interaktion realisiert werden kann. Diese Entwürfe berücksichtigen unterschiedliche technische Rahmenbedingungen und werden auf ihre Vor- und Nachteile geprüft. und Aufbauend auf einem der identifizierten Systementwürfe wird eine Demonstrator Applikation entwickelt mit der ein aktives Dokument oder Foto auf ein anderes Gerät übertragen werden kann.

1.3 Aufbau der Arbeit

In Kapitel 2 werden verwandte Arbeiten und Applikationen zum Thema sowie Technische Grundlagen beschrieben. Kapitel 3 analysiert und beschreibt die Bump-Interaktion. Kapitel 4 beschreibt einleitend Grundlagen zu relevanten iOS Frameworks auf deren Basis Konzepte zur Bump Erkennung, Geräte Identifizierung und zum Datenaustausch beschrieben und verglichen werden. In Kapitel 5 wird die entwickelte Demonstrator Applikation vorgestellt. Abschließend bietet Kapitel 6 eine Zusammenfassung über die erarbeiteten Ergebnisse und einen Ausblick auf zukünftige potenzielle Weiterentwicklungen der Arbeit. Die Bedeutungen der verwendeten Abkürzungen befinden sich im Abkürzungsverzeichnis und ein Überblick der dargestellten Abbildungen im Abbildungsverzeichnis sowie der Tabellen im Tabel-

lenverzeichnis. Zudem ist die verwendete Literatur im Literaturverzeichnis angegeben.

Kapitel 2

Stand der Technik

Im folgenden Kapitel werden verwandte Arbeiten und Applikationen zum Thema sowie Technisches Grundlagenwissen, das zur Umsetzung der Interaktion benötigt wird, beschrieben.

2.1 Grundlagen zum gestenbasierten Transfer von Daten

Die Nutzung einer gestenbasierten Interaktion zum Datenaustausch wurde schon von einigen wissenschaftlichen Arbeiten behandelt und in Applikationen umgesetzt. An dieser Stelle wird eine Auswahl an Arbeiten und Applikationen vorgestellt die Ähnlichkeit mit der geplanten Bump-Interaktion aufweisen.

Hinckley [Hin03] beschreibt eine Interaktion, mit der Tablet-Computer über eine Bump-Interaktion miteinander verbunden werden. Das anstoßen der Geräte aneinander (Siehe Abbildung 2.1) baut eine Verbindung zwischen den Geräten auf, indem die generierten Accelerometer Daten zusammen mit einem Timestamp über ein lokales WLAN-Netzwerk ausgetauscht und abgeglichen werden. Die Bump-Interaktion entwickelt Hinckley auf der Metapher des Anstoßens von zwei Trinkgläsern aneinander. Deshalb führt er die Trinkgläser-Metapher auch für den Datenaustausch fort und definiert eine Geste, die dem umschütten von Flüssigkeit aus einem Glas in ein anderes entspricht. Dieser Metapher folgend, schüttet eines der Geräte ausgewählte Daten durch eine drehende Bewegung in das andere Gerät (Siehe Abbildung 2.2).



Abbildung 2.1: Bump Tablet-Computer [Hin03]

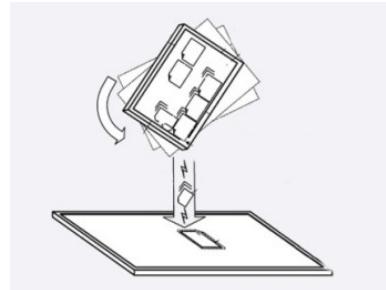


Abbildung 2.2: Daten ausschütten [Chr11]

Ähnlich wie bei Hinckley werden bei der Applikation **Bump** [bum13] die Beschleunigungssensoren von Smartphones und Tablets genutzt um Bump-Events festzustellen (Siehe Abbildung 2.3). Die Applikation sendet, nachdem ein Bump festgestellt wurde, die ausgewählten Nutzdaten, sowie Charakteristiken des Bumps an einen Server. Der Server erkennt eingehende Bumps durch einen Algorithmus und ordnet Bump-Partner basierend auf den Charakteristiken der Bumps einander zu. Der Algorithmus nutzt unter anderem GPS-Standortdaten, um die Anzahl der möglichen Treffer zu reduzieren, sowie eine Anzahl anderer Daten, um die richtigen Bump-Partner zu identifizieren. Nachdem der Server die Geräte einander zugeordnet hat, leitet er die entsprechenden Nutzdaten an die jeweiligen Geräte weiter. Der Datenaustausch zwischen mobilen Endgeräten und Computern ist auch möglich. Dazu muss das mobile Gerät auf die Leertaste der Computertastatur gebumped werden (Siehe Abbildung 2.4).



Abbildung 2.3: Bump-Geste Smartphones [Lan14]



Abbildung 2.4: Bump-Geste PC [bum13]

Yatani et al. [YTH⁺06] nutzen Accelerometer Daten um Wurf- und Schwung-Gesten zu erkennen. Mit einer Wurf-Geste werden Daten, zu einem einzelnen Empfänger geworfen. Durch eine Schwung Geste können mehrere Geräte auf einmal

als Empfänger ausgewählt werden. Der Datenaustausch erfolgt über einen Server in einem lokalen WLAN-Netzwerk. Der Server ist ebenfalls dafür zuständig, die Position und Orientierung der Geräte zu verarbeiten, um so Sender und Empfänger einander zuzuordnen. Für die Positionsbestimmung wird ein speziell entwickeltes System genutzt, mit dem die Position und Orientierung der Endgeräte über eine Kamera erkannt wird.

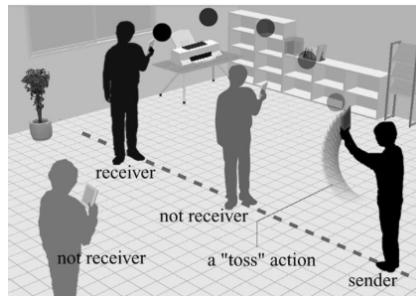


Abbildung 2.5: Wurf- und Fang-Geste [YTH⁺06]

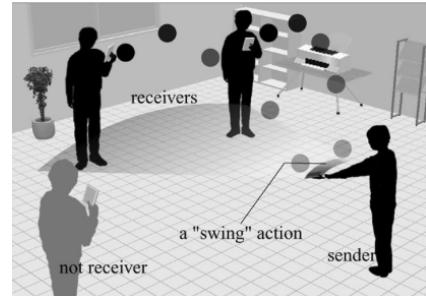


Abbildung 2.6: Schwung-Geste [YTH⁺06]

Hoccer ist eine Applikation, die über zwei Gesten verfügt, mit denen Daten zwischen Endgeräten ausgetauscht werden können. Zum einen gibt es die Swipe Geste (Siehe Abbildung 2.7) und zum anderen eine Wurf und Fang Geste, die vergleichbar mit der Geste von Yatani et al. ist. Mit der Swipe Geste wird über den Touchscreen eine geöffnete Datei von einem Gerät auf das andere geschoben. Bei der Wurf und Fang Geste, führt ein Anwender die Wurf Geste aus und wirft einem zweiten Anwender die Daten, wie mit einem Frisbee, zu. Dieser muss sich als Empfänger identifizieren indem er sein Endgerät nach oben hält, als wollte er das Frisbee fangen. Die Applikation funktioniert technisch ähnlich wie Bump über GPS-Standortdaten mit einem Server, der Geräte zuordnet und den Datenverkehr zwischen den Geräten steuert. [Maa11]

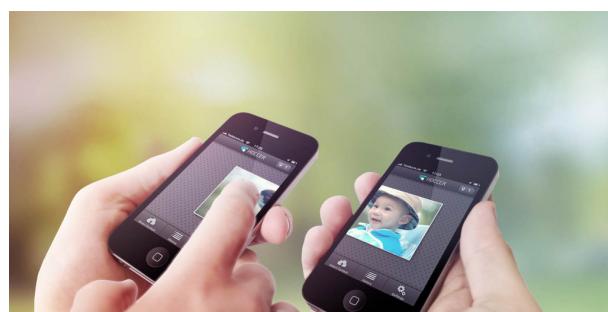


Abbildung 2.7: Hoccer Swipe-Geste [sal]

Lucero et al. [LHJ11] haben ein Gesten-gesteuertes System entwickelt, mit dem eine Gruppe von Personen ihre auf Smartphones gespeicherten Photosammlungen durchblättern und untereinander Photos austauschen können. Dazu müssen die Smartphones aller Anwender zusammen auf einem Tisch liegen, am besten in einem Kreis angeordnet. Mit der Hilfe von Gesten kann jeder Anwender durch seine individuelle Photosammlung blättern. Dazu muss er das Smartphone rechts kippen um ein Photo weiterzublättern (Siehe Abbildung 2.8) oder nach links kippen um zurückzublättern. Durch einen Tap auf den Touchscreen rechts oder links kann auch entsprechend vor oder zurückgeblättert werden. Möchte ein Anwender ein einzelnes Bild teilen, muss er so lange die Tap-Geste auf dem Touchscreen ausführen bis eine Miniaturansicht des ausgewählten Photos erscheint. Dieses Bild kann anschließend durch eine Swipe-Geste in die Richtung des Zielgeräts auf dieses übertragen werden. Das System nutzt Accelerometer Daten, um die Geste zum weiterblättern zu erkennen und tauscht die Daten zwischen den Geräten über ein WLAN Netzwerk aus. Für die Identifizierung des Zielgeräts beim Datenaustausch wurde ein spezielles Multi-Emitter Tracking System verwendet, mit dem die Position der Endgeräte festgestellt werden kann.



Abbildung 2.8: Links kippen zum weiterblättern **Abbildung 2.9:** Schwung-Geste Auswahl mehrerer Ziele [LHJ11]



2.2 Technische Grundlagen

Mobile Endgeräte stecken voller Technologien, um ihre Umwelt wahrzunehmen und mit ihr zu kommunizieren. Dieser Abschnitt identifiziert Technologien zu Datenübertragung, Positionsierung und Sensorik, die für die Implementierung der Bump-Interaktion in Frage kommen.

2.2.1 Technologien zur Datenübertragung

WLAN

Ein Wireless Local Area Network (WLAN) ist ein lokales Funknetz, das auf der Normenfamilie IEEE 802.11 basiert und sich über eine Reichweite von 30 - 100 Meter aufspannen kann. Ein solches Netzwerk kann entweder im Infrastruktur-Modus oder im Ad-hoc-Modus betrieben werden. Der Infrastruktur-Modus ist topologisch wie ein Stern (Siehe Abbildung 2.10) aufgebaut. Endgeräte melden sich an einer zentralen Basisstation an und können danach über diese miteinander kommunizieren. Im Ad-hoc-Modus bauen Endgeräte eine direkte Verbindung miteinander auf ohne den Umweg über eine gesonderte Basestation. Jedes Endgerät kann mehrere Verbindungen zu anderen Geräten unterhalten, womit ein vermaschtes Netzwerk (Siehe Abbildung 2.10) gebildet wird [Bau12, 46-55].

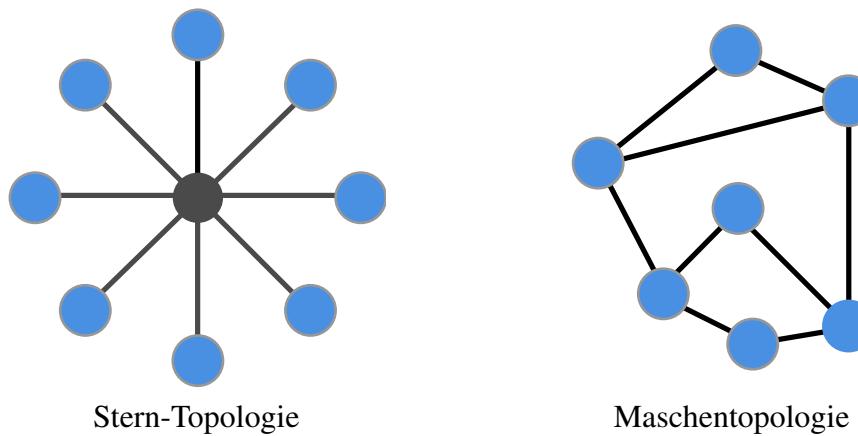


Abbildung 2.10: WLAN-Topologien [Bau12, vgl. Abb. 3.1]

Damit Endgeräte verfügbare WLAN Netze entdecken können, senden Basistationen oder Ad-hoc Geräte in Intervallen Beacons mit Informationen über das Netzwerk an alle Endgeräte im Empfangsbereich. Sobald Geräte in Reichweite eines Netzwerkes sind, können sie sich direkt verbinden. Sollte das Netzwerk jedoch durch ein Pass-

wort geschützt sein muss dieses erst eingegeben und überprüft werden. Es existieren verschiedene WLAN-Standards, die sich vor allem durch ihre Datentransferrate unterscheiden. Der weit verbreitete Standard 802.11n bietet im Infrastruktur Modus Übertragungsraten von Brutto 100-120 Mbit/s [Bau12, 46-55]. Die Geschwindigkeit einer Ad-hoc Verbindungen ist dagegen auf 11 Mbit/s beschränkt [WHBW11].

WIFI-Direct

WIFI-Direct ist eine spezielle Implementierung des WLAN Ad-hoc-Modus. Es liefert signifikant höhere Datenübertragungsraten als der klassische Ad-hoc Modus von bis zu 250 Mbit/s. Für einen einfachen und schnellen Verbindungsaufbau zwischen den Geräten ist es möglich, die Pairing Prozesse von Bluetooth oder Near Field Communication (NFC) zu nutzen. WIFI Direct ist verfügbar auf Geräten mit Android oder Windows Betriebssystemen [Wik14c].

Bluetooth Low Energy

Bluetooth ist ein Funksystem zur Datenübertragung über Distanzen von bis zu 30 Metern [AB12]. Ab der Version 4.0 spricht man auch von Bluetooth Low Energy, da ab dieser Version die Technologie auf sehr niedrigen Stromverbrauch optimiert wurde. Um miteinander zu kommunizieren, bauen Endgeräte eine Peer to Peer Verbindung untereinander auf und organisieren sich in sogenannten Pico Netzen (Siehe Abbildung 2.11). Ein Pico Netz besteht aus maximal 255 Teilnehmern wovon maximal acht aktiv sein dürfen. Einer der aktiven Teilnehmer nimmt dabei die Rolle des Masters ein, während die anderen sieben Slaves sind. Die restlichen 247 Teilnehmer sind passiv, können jedoch jederzeit vom Master aktiviert werden. Der Master steuert die Kommunikation und den Datenverkehr zwischen den Teilnehmern des Netzes [Bau12, 55-57].

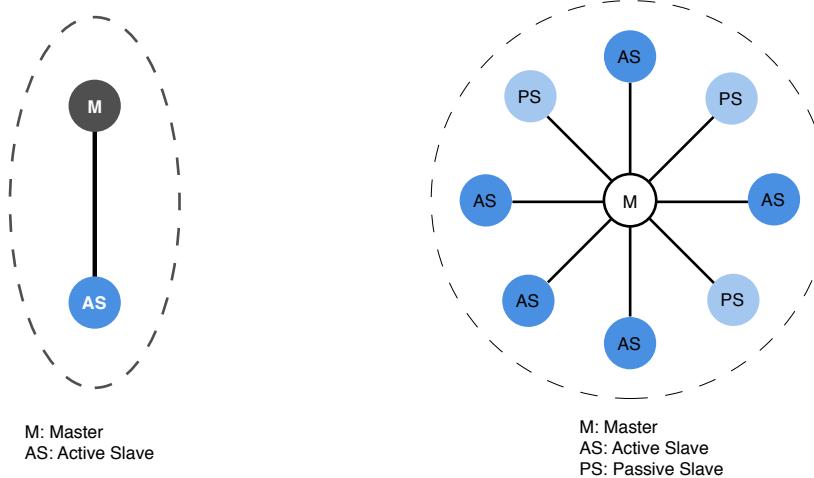


Abbildung 2.11: Bluetooth-Topologie Piconetz [Bau12, vgl. Abb. 5.4]

Bevor Daten übertragen werden können, müssen sich Geräte erst über den Vorgang des Pairings kennen lernen. Bei diesem Vorgang müssen die Benutzer jeweils einen gemeinsamen Code durch Tastendruck auf ihrem Gerät bestätigen. Nachdem Geräte einmal gepaired wurden, ist ein weiteres Pairing nicht mehr nötig, da sie sich bereits kennen. Seit Version 3.0 beherrscht Bluetooth den High Speed Modus, welcher eine Kombination aus Bluetooth und WLAN 802.11g ist. Über die Bluetooth Verbindung mit 3 Mbit/s werden dabei nur noch Steuerdaten sowie Sitzungsschlüssel übertragen. Für die Nutzdaten wird eine Ad-hoc Verbindung über WLAN aufgebaut, die eine Datentransferrate von ca 24 Mbit/s erreicht [Bau12, 55-57].

NFC

Near Field Communication auch als NFC bekannt, ist eine Funktechnologie für sehr kurze Distanzen. In einer Reichweite von optimal 4 cm und maximal 20 cm können Geräte miteinander eine Peer to Peer Verbindung eingehen und Daten mit bis zu 424 kBit/s übertragen. Anders als bei WLAN und Bluetooth ist es mit NFC nicht möglich, Verbindungen zu mehreren Geräten gleichzeitig zu unterhalten. Einen weiteren Unterschied stellt der Verbindungsaufbau zwischen den Geräten dar. Dieser ist bei NFC sehr einfach gestaltet, da das Pairing allein durch die unmittelbare Nähe der Geräte zueinander geschieht und keine weiteren Aktionen des Nutzers benötigt. Aus diesem Grund wird NFC auch in Verbindung mit Bluetooth oder WLAN genutzt um schnell und einfach eine Verbindung aufzubauen ohne Passworteingabe bei WLAN oder Code Bestätigung bei Bluetooth [AB12]. NFC ist verfügbar auf Android- und

Windows-Geräten. Obwohl NFC ab dem iPhone 6 integriert ist, sind keine APIs für Entwickler vorhanden.

Mobilfunk

Moderne Mobilfunknetze bieten neben dem Zugang zu Telefondiensten auch Zugang zu Internetdiensten. Die aktuellsten Mobilfunkstandards sind High Speed Packet Access (HSPA+) und Long Term Evolution (LTE). Bei HSPA+ handelt es sich um eine Erweiterung des Mobilfunkstandards UMTS mit Datenraten von bis zu 42 Mbit/s [3GPa]. LTE dagegen ist eine komplett neu entwickeltes Mobilfunknetz, das in der aktuellen Spezifikation theoretische Datenraten von bis zu 75 Mbit/s im Upload und 300 Mbit/s im Download bietet [3GPb].

2.2.2 Datenkommunikation zwischen mobilen Endgeräten

Über die Technologien, die im vorherigen Abschnitt beschrieben wurden, kann abhängig von der Technologie, entweder direkt zwischen Geräten im Netzwerk oder über einen Server kommuniziert werden. Folgend wird beschrieben, mit welchen Protokollen die Kommunikation in beiden Fällen umgesetzt werden kann.

Datenkommunikation in lokalen Netzwerken

Sollen mobile Applikationen über ein lokales Netzwerk miteinander kommunizieren kann dies durch die Nutzung von Netzwerkdiensten realisiert werden. Ein Netzwerkdienst ist ein Informationsobjekt, das von einer Person, einem Programm, oder von einem anderen Dienst genutzt werden kann. Jeder Dienst stellt eine Funktionalität zur Verfügung, dabei kann es sich z.B. um Anwendungsdienste oder Kommunikationsdienste handeln [Che02]. Eine wichtige Komponente zur Nutzung von Diensten ist *service discovery*, über welches angebotene Dienste in einem Netzwerk gefunden werden können. Für service discovery existieren verschiedene Protokolle. Apple nutzt in seinen Betriebssystemen Bonjour [App14a], Android nutzt Network Service Discovery [Goo15], um nur zwei Vertreter zu nennen. Beide Protokolle ermöglichen das automatische entdecken von Geräten und Diensten in lokalen Netzwerken.

Ein Dienst wird unter Anderem durch seinen Namen beschrieben. Dieser Name ist im Netzwerk sichtbar für alle Geräte, die danach suchen. Angebotene Dienste sollten einen einzigartigen Namen aufweisen, um Konflikte zwischen verschiedenen

Applikationen zu vermeiden. Durch den Einsatz von Diensten können verschiedene Applikationen in einem Netzwerk untereinander kommunizieren. Dabei ist die Topologie des Netzwerkes unerheblich. Netzwerkdienste können in Infrastrukturnetzwerken (Siehe Abbildung 2.12 a) genutzt werden und auch um Ad-Hoc Netzwerke zu bilden (Siehe Abbildung 2.12 b).

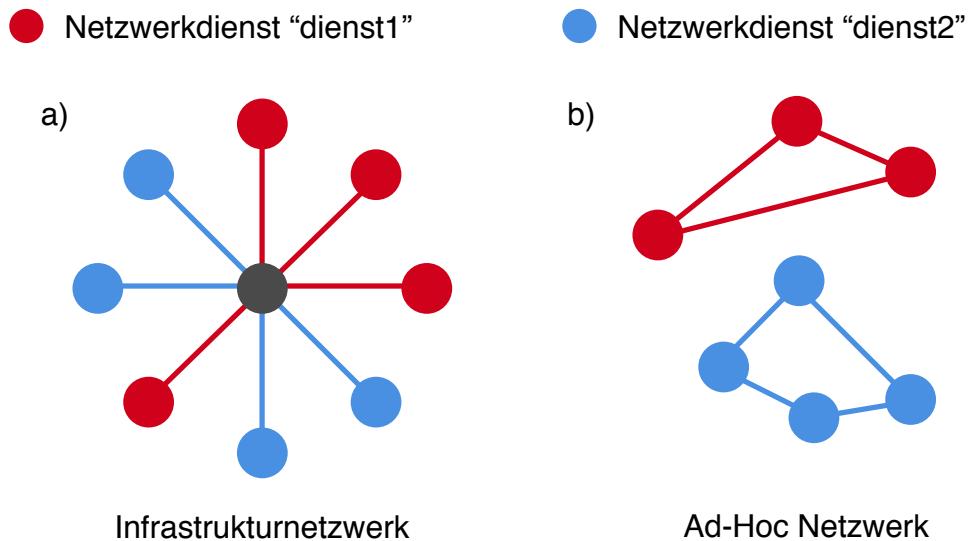


Abbildung 2.12: Netzwerkdienste in Computernetzwerken

Datenkommunikation über einen Server

Anstatt direkt in einem lokalen Netzwerk zu kommunizieren, können Applikationen auch über einen Server Daten austauschen. Das Standardmodell für diese Kommunikation ist das Client-Server-Modell (Siehe Abbildung 2.13). In diesem Modell rufen Clients bestimmte Funktionalitäten bzw. Dienstleistungen, die ein Server zur Verfügung stellt, über ein Netzwerk hinweg auf [SS12, 14].

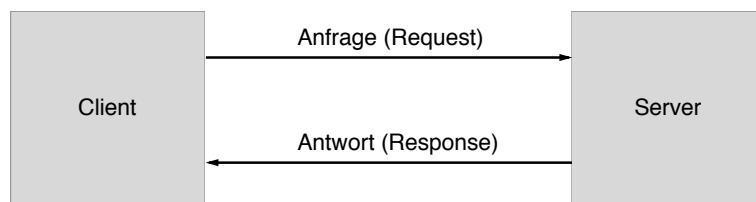


Abbildung 2.13: Client-Server-Modell

HTTP Operationen

Clients stehen HTTP Operationen zur Verfügung, mit denen die Kommunikation mit einem Server erfolgen kann. Die wichtigsten dieser Operationen [Rod08, 14], sind die sogenannten CRUD Operationen: create, read, update and delete. Die entsprechenden HTTP Methoden, mit denen diese Operationen durchgeführt werden können, sind:

- POST um Ressourcen auf dem Server anzulegen.
- GET um Ressourcen aufzurufen.
- PUT um eine Recource zu aktualisieren
- DELETE um eine Recource zu löschen

Endgeräte tauschen über diese Methoden Daten aus, indem eines der Geräte eine Ressource auf dem Server anlegt, während das andere Gerät die entsprechende Ressource aufruft. Der Datenaustausch über diese Methoden hat zur Folge, dass Clients anhaltend überprüfen müssen, ob die Daten bereits auf dem Server verfügbar sind. Für mobile Endgeräte ist dieser Vorgang aber nachteilig, da die ständigen Anfragen an den Server kostbare Batterielaufzeit verschwenden [LH07]. Es existiert jedoch eine alternative Möglichkeit, mit der Client und Server kommunizieren können ohne dieses Problem, in der Form von Web Sockets.

Web Sockets

Web Sockets ermöglichen eine bidirektionale Verbindung zwischen einer Anwendung und einem Webserver. Wurde einmal eine Socket Verbindung durch den Client geöffnet, kann diese Verbindung aus beiden Richtungen verwendet werden. Für einen Datenaustausch bedeutet dies, dass Endgeräte nicht konstant nachfragen müssen, ob eine Ressource vorhanden ist. Der Server sendet die Daten selbstständig an die Clients, sobald Sie verfügbar sind. Socket-Verbindungen bleiben dauerhaft bestehen und werden erst geschlossen, wenn der Client die Verbindung schließt [LG10].

2.2.3 Technologien zur Geräteortung

GPS

Das Global Positioning System Global Positioning System (GPS) ist ein globales mit Satelliten betriebenes Navigationssystem zur Positionsbestimmung. Ist ein Endgerät mit GPS ausgestattet, ist es möglich, den Standort dieses Geräts zu bestimmen. Der Standort wird in geographische Koordinaten angegeben, mit denen sich die Lage eines Punktes auf der Erde beschreiben lässt [Wik14a]. Die Funktionsfähigkeit von GPS ist maßgeblich von der Empfangsqualität abhängig, so ist eine Ortung in Gebäuden häufig nicht möglich und auch die Genauigkeit der Positionsbestimmung kann negativ beeinflusst werden. In der Regel wird GPS in Kombination mit anderen Technologien wie Mobilfunk-, WLAN- oder Bluetooth genutzt. Dadurch kann zum einen der Verbindungsaufbau zu den Satelliten beschleunigt und zum anderen die Genauigkeit der Standortbestimmung verbessert werden. Trotz dieser Technik unterliegen die mit GPS ermittelten Standortdaten häufig einer Ungenauigkeit die mehrere Meter hoch ausfallen kann [App14c].

iBeacon

Mit iBeacon können standortbezogene Dienste und Navigationslösungen in geschlossenen Räumen realisiert werden. iBeacon ist Apple's Markenname für eine Technologie, die Teil des Bluetooth Low Energy Standards ist. Dadurch ist iBeacon auf allen Endgeräten verfügbar die über Bluetooth LE verfügen und deren Betriebssystem kompatibel ist. Momentan sind das iOS7 und Android 4.3. Neben Software Komponenten umfasst die Beacon Technologie auch Hardware Komponenten. Dies sind speziell entwickelte Signalgeber, ebenfalls als iBeacons oder Beacons bezeichnet, die mit Bluetooth LE Technik ausgestattet sind. Diese Hardware Beacons werden jedoch nicht in allen Anwendungsszenarien benötigt, da jedes kompatible Smartphone ebenfalls die Funktionen eines Beacons übernehmen kann [Wik14b].

iBeacon unterscheidet drei Grundlegende Funktionen: *Advertising*, *Ranging* und *Region Monitoring*. Im folgenden werden diese Funktionen beschrieben und die Funktionsweise der Beacon Technologie dargestellt.

Advertising ist der Vorgang mit dem Beacons andere Geräte über ihre Anwesenheit informieren. Jedes Beacon spannt einen Bereich um sich, der als *region* bezeichnet wird. Innerhalb dieses Aktionsradius senden Beacons kontinuierlich Signale mit Informationen über ihre Identität aus.

Die Identität eines Beacons setzt sich zusammen aus einer 16 Byte *UUID* (Universally Unique Identifier) sowie einem *major* und einem *minor* Wert die jeweils 2 Byte groß sind. Die *UUID* ist ein Identifikator der je nach Anwendungsfall von einem oder mehreren Beacons genutzt werden kann. Mit *major* und *minor* Wert kann für jedes Beacon eine einzigartige ID gebildet werden mit der dieses eindeutig aus einer Gruppe von Beacons mit der selben *UUID* identifiziert werden kann (Siehe Abbildung 2.14).

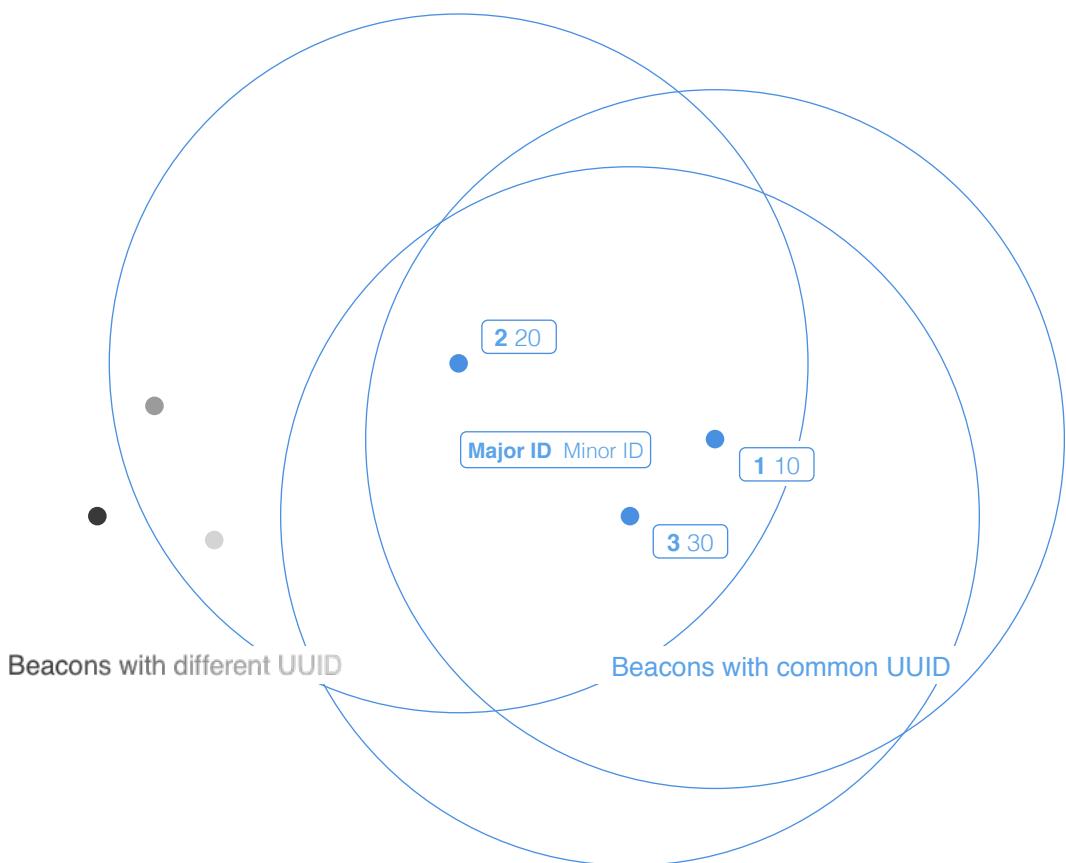


Abbildung 2.14: iBeacon regions

Die folgende Tabelle zeigt ein Beispiel wie die Werte genutzt werden können um iBeacons in einer Einzelhandelskette einzusetzen.

Store Location		San Francisco	Paris	London
UUID		D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C		
Major		1	2	3
Minor	Clothing	10	10	10
	Housewares	20	20	20
	Automotive	30	30	30

Tabelle 2.1: Einsatz von Beacons in einer Einzelhandelskette [Com14]

In diesem Beispiel wird die *UUID* von den Beacons an allen Standorten gemeinsam genutzt und identifiziert die Handelskette als ganzes. Jede Filiale wird durch einen *major* Wert identifiziert. Innerhalb einer jeden Filiale werden die einzelnen Abteilungen durch einen *minor* Wert repräsentiert. Mit Hilfe dieser Informationen ist eine Applikation in der Lage festzustellen, ob der Benutzer eine der Filialen betritt, verlässt und in welcher Abteilung er sich gerade befindet [Com14].

Ranging ist die Funktion mit der eine Applikation nach Signalen sucht die von Beacons ausgesendet werden. Es können dabei nur Signale empfangen werden von Beacons deren *UUID* der Applikation bekannt sind. Werden zum ersten mal Signale eines solchen Beacons empfangen, wurde die *region* dieses Beacons betreten und ein Event wird in der Applikation ausgelöst. Umgekehrt bedeutet der langfristige Verlust des Beacon-Signals, dass die *region* verlassen wurde. Wie beim Betreten, wird auch beim Verlassen einer *region* ein entsprechendes Event in der Applikation ausgelöst [Com14].

Nachdem eine *region* betreten wurde, kann der Abstand zum Beacon anhand der Signalstärke *RSSI* (Received Signal Strength Indication) näherungsweise bestimmt werden. Die Ermittlung des Abstands ist anfällig, ungenaue Ergebnisse zu produzieren da das Signal durch externe Faktoren beeinflusst werden kann [Est14]. Bei diesen Faktoren kann es sich sowohl um Räumliche Begebenheiten handeln als auch um Interferenzen die durch andere Funksignale z.B. durch WLAN ausgelöst werden können. Der iBeacon Hersteller Estimote ermittelte in internen Tests Abweichungen von 5-6cm bei einem Abstand von 20cm sowie 2-3m bei Abständen von über 10m [Est14]. Die Genauigkeit der Messung scheint also fundamental vom Abstand zu Beacons bzw. der Signalstärke abzuhängen. Um Entwicklern eine Möglichkeit zu geben, auf Ungenaue Messergebnisse zu reagieren, gibt iOS eine Einschätzung

zur Genauigkeit der Messung mit dem *accuracy* Parameter aus. Ist der *accuracy* Wert niedrig bedeutet dies, dass sich iOS sehr sicher ist über die Genauigkeit der Messung. Umgekehrt bedeutet ein hoher *accuracy* Wert, dass der ermittelte Abstand sehr wahrscheinlich ungenau ist [Com14].

Ist für ein Anwendungsszenario eine ungefähre Abschätzung über den Abstand zu einem Beacon ausreichend, kann man diese über die vier vordefinierten *proximity states* erhalten. Diese lauten: *unknown*, *far*, *near* und *immediate*. *Unknown* beschreibt dabei die Entfernung, die ermittelt wird, sollte sich das Smartphone außerhalb der Reichweite des Beacons befinden oder sollte die Qualität der empfangenen Signale nicht ausreichen, um den Abstand festzustellen. Wird *far* als Abstand ausgegeben, werden Beacon Signale empfangen, jedoch sind Sie schwach oder iOS ist sich unsicher über die Genauigkeit der Messung. Der Abstand zum Beacon kann zwischen 70m und 3m liegen. *near* beschreibt eine Zone in der die Signalstärke gut ist und die Genauigkeit ebenfalls als gut eingeschätzt wird. Der Abstand zum Beacon bewegt sich zwischen 0.5m und 3m. Der letzte mögliche *proximity state* ist *immediate*. Wird dieser Status angezeigt ist die Signalstärke sehr hoch und iOS ist sich sehr sicher über die Genauigkeit der Messung. Der Empfänger befindet sich in einem Abstand von unter 0.5m zum Sender (Siehe Abbildung 2.14). In einer Applikation wird für jeden der *proximity state* ein entsprechendes Event ausgelöst sobald dieser festgestellt wird [Est14].

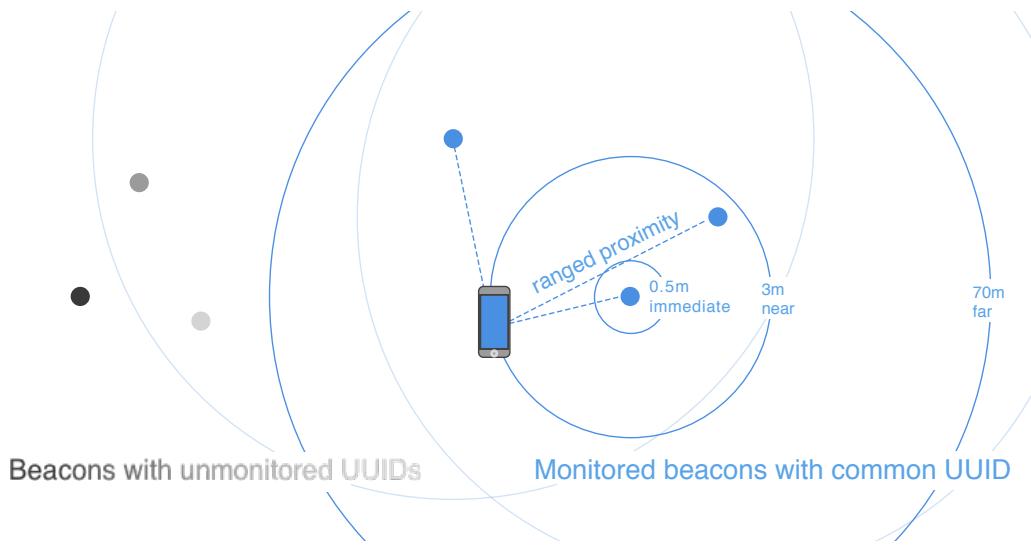


Abbildung 2.15: iBeacon ranging

Region Monitoring bietet eine weitere Funktionalität mit der Applikationen Events auslösen können sobald das Smartphone eine definierte Beacon *region* betritt oder

verlässt. Dafür muss die Bump-Applikation im Gegensatz zur selben Funktionalität beim *Ranging* nicht aktiv sein. Geschlossene Applikationen können durch das *Region Monitoring* im Hintergrund gestartet werden und z.B. *Push Notifications* auslösen. Andere Aktionen die ausgeführt werden können, während eine Applikation im Hintergrund ist, sind auch möglich.

2.2.4 Sensoren

Mobile Endgeräte sind mit einer Reihe von Sensoren ausgestattet, mit denen es möglich ist, unentwegt Daten zu sammeln. Applikationen können diese Daten auswerten, um zu sehen, hören und fühlen, was mit dem Gerät und in seiner Umgebung geschieht. Bewegungen des Anwenders bzw. des Endgeräts können durch die Auswertung der Daten des Beschleunigungssensors erkannt werden. Ein Beschleunigungssensor, auch Accelerometer genannt, ist ein Sensor, mit dem Beschleunigung oder g-Kraft in Meter pro Quadratsekunde (m/m^2) gemessen werden kann. In mobilen Endgeräten sind in der Regel drei Sensoren eingebaut, mit denen die Beschleunigung auf x, y und z-Achse bestimmt werden kann (Siehe Abbildung 2.16). Durch die Auswertung der Sensordaten können Bewegungen des Benutzers z.B. laufen, rennen oder Treppensteigen erkannt werden. Die Daten können aber auch genutzt werden, um zu erkennen, ob sich ein Endgerät im freien Fall befindet, oder um Erschütterungen festzustellen, wenn das Gerät gegen etwas stößt [RDML05].

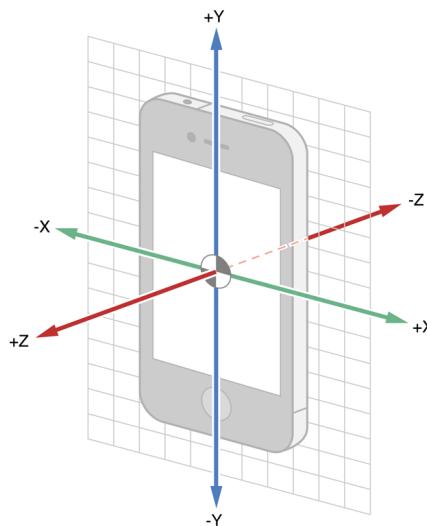


Abbildung 2.16: 3-Achsen Beschleunigungssensor [App15]

Kapitel 3

Analyse der Bump-Geste

Bei der Bump-Geste handelt es sich um eine Interaktion bei der von einem Quellgerät, auf dem ein Anwender eine Aktivität begonnen hat, Daten durch einen Bump auf ein Zielgerät übertragen werden, damit der Anwender auf diesem seine Aktivität fortsetzen kann. Um genau zu klären, wie dieser Vorgang funktioniert, wird in diesem Kapitel untersucht, wie die Interaktion abläuft, in welchen Varianten die Geste durchgeführt werden kann, welche Endgeräte sich für die Geste eignen und wie die Geste in einer Applikation umgesetzt werden kann.

3.1 Ablauf der Interaktion

Die Interaktion kann aufgeteilt werden in die Aktionen des Benutzers und die Reaktion der Systeme auf die Interaktion.

3.1.1 Aktionen des Benutzers

Wird die Interaktion zwischen zwei Anwendern durchgeführt, halten beide Benutzer jeweils ein Gerät fest in der Hand und lassen die Geräte zusammenstoßen/-bumpen. Erfolgt die Interaktion zwischen einem Anwender und einem stationären Endgerät hält entsprechend ein Anwender sein Gerät fest in der Hand und lässt es auf das stationäre Gerät bumpen.

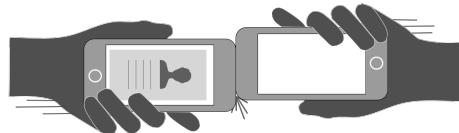


Abbildung 3.1: Aktion des Benutzers

3.1.2 Reaktion des Systems

Die hauptsächliche Reaktion des Systems ist es, die Daten des Quellgeräts auf das Zielgerät zu übertragen und dort anzuzeigen (Siehe Abbildungen 3.2, 3.3). Die Geräte sollten aber auch eine visuelle oder akustische Rückmeldung an die Benutzer geben, um zu signalisieren, dass ein Bump von den Endgeräten festgestellt wurde, z.B. über Vibration und mit welchem Gerät/Person Daten ausgetauscht werden.

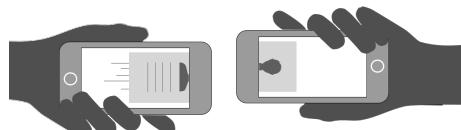


Abbildung 3.2: Daten übertragen

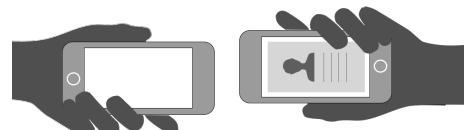


Abbildung 3.3: Daten anzeigen

3.2 Variationen der Bump-Geste

Die Bump-Geste lässt sich auf verschiedene Art und Weisen durchführen. An dieser Stelle sollen einige dieser Variationen als Beispiel dienen, um zu illustrieren wie eine Bump-Geste aussehen kann.



Abbildung 3.4: Stirnseite an Stirnseite

Stirnseite an Stirnseite

Die Endgeräte werden waagerecht in den Händen der Anwender gehalten und werden an den Stirnseiten zusammengestoßen.

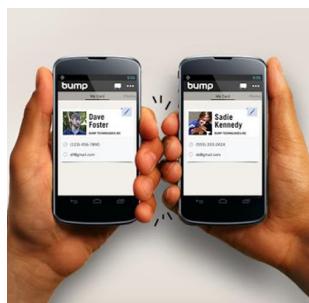


Abbildung 3.5: Längsseite an Längsseite

Längsseite an Längsseite

Die Endgeräte werden aufrecht in den Händen der Anwender gehalten und werden an den Längsseiten zusammengestoßen. Dabei können die Geräte auch von den Händen umschlossen werden wodurch die Interaktion wie ein Fist-bump aussieht.

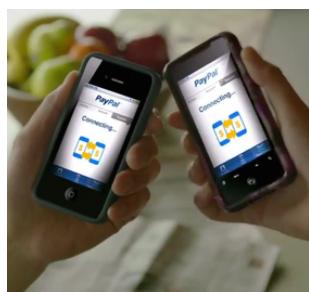


Abbildung 3.6: Ecke an Ecke

Ecke an Ecke

Die Endgeräte werden aufrecht in den Händen der Anwender gehalten und werden an den oberen Ecken zusammengestoßen.



Abbildung 3.7: Stirnseite aufstoßen

Stirnseite aufstoßen

Ein Endgerät wird vom Anwender aufrecht in der Hand gehalten und mit der oberen oder unteren Kante auf ein stationäres Gerät (Surface-Table, Notebook) aufgestoßen.

3.3 Geeignete Endgeräte

Aus den dargestellten Bump-Variationen ist zu erkennen, dass die Bump-Interaktion zwischen verschiedenen Typen von Geräten erfolgen kann. Es gilt an dieser Stelle allgemein zu klären, zwischen welchen Gerätetklassen die Bump-Geste eingesetzt werden kann. Die Geräte werden nach Terrenghi et al. [TQD09] basierend auf ihrer Bildschirmgröße kategorisiert. Die folgende Tabelle liefert einen Überblick über diese Kategorien.

Kategorie	Bildschirmgröße	Beispiel
Inch size	$\geq 2,54$ cm	Smartphone
Foot size	$\geq 30,48$ cm	Tablet
Yard size	$\geq 91,44$ cm	Tabletop
Perch size	≥ 05 m	TV-Bildschirm
Chain size	≥ 20 m	Mehrere Displays

Tabelle 3.1: Kategorisierung von Endgeräten

Um für die Interaktion geeignete Geräte zu identifizieren gilt es erst zu definieren welche verschiedenen Rollen ein Gerät in der Interaktion einnehmen kann. Die Endgeräte können aufgeteilt werden in Quell- und Zielgeräte. Ein Quellgerät ist der Informationsträger, von welchem Daten durch den Bump auf das Zielgerät übertragen werden. Bei einem Quellgerät muss es sich um ein mobiles Endgerät handeln. Es muss vom Anwender einfach in einer oder zwei Händen gehalten, bewegt und auf das Zielgerät gestoßen werden können. Bei Zielgeräten hingegen kann es sich sowohl um ein mobiles als auch um ein stationäres Endgerät handeln. Stationäre Endgeräte zeichnen sich dadurch aus, dass Sie vom Anwender bei der Nutzung nicht in der Hand gehalten werden.

Die definierten Gerätetypen können jetzt auf die jeweiligen Rollen aufgeteilt werden. Quellgeräte können nur Geräte aus den Kategorien Inch size und Foot size sein. Zielgeräte können zusätzlich auch aus den Kategorien Yard-, Perch- und Chain size kommen.

Nachdem die Gerätetypen zugeordnet sind gilt, es noch zu klären, ob die Geräte aus diesen Kategorien auch für die Interaktion geeignet sind. Inch und Foot size Geräte eignen sich für die Interaktion. Sie können vom Anwender einfach geführt

werden und sowohl als Quell- als auch als Zielgerät dienen. Yard size Geräte eignen sich als Zielgerät. Der Anwender kann sein mobiles Endgerät auf das Gerät aufstoßen und kann sehen, welche Auswirkungen die Interaktion auf das Zielgerät hat. Für Perch- und Chain size Geräte gilt dies nicht, da ihre Bildschirme so groß sind und der Anwender für die Interaktion so nah an den Geräten stehen muss, dass er nicht sehen kann, was auf den Zielgeräten geschieht. Geräte aus diesen Kategorien eignen sich daher nicht gut für die Bump-Geste.

3.4 Umsetzung der Interaktion

Um die Bump-Interaktion umzusetzen, müssen drei Teilsysteme entwickelt werden: Eines dieser Systeme muss erkennen, wenn Endgeräte angestoßen werden. Ein weiteres System muss Endgeräte identifizieren können, damit die Daten zwischen den richtigen Geräten ausgetauscht werden. Außerdem wird ein System benötigt, über das ein Kommunikationskanal zwischen den Geräten hergestellt wird, über den Daten ausgetauscht werden können.

Kapitel 4

Umsetzung der Interaktion

Dieses Kapitel beschreibt Konzepte für die im vorangegangen Kapitel identifizierten Teilsysteme. Diese Konzepte beziehen sich zu einem großen Teil auf iOS Frameworks, mit denen die Demonstrator Applikation entwickelt wurde. Aus diesem Grund werden diese Frameworks einleitend in Abschnitt 4.1 beschrieben. Darauf folgt in Abschnitt 4.2 ein Konzept zur Erkennung von Bumps, in Abschnitt 4.3 werden Konzepte zur Identifizierung der Endgeräte beschrieben und in Abschnitt 4.4 Konzepte zum Datenaustausch zwischen den Endgeräten.

4.1 Grundlagen zu iOS Frameworks

iOS bietet durch Frameworks Zugang zu verschiedenen Technologien. Im folgenden wird die grundlegende Funktionsweise von drei Frameworks beschrieben, die zur Umsetzung der Bump-Interaktion genutzt wurden.

4.1.1 Core Motion Framework

Das Core Motion Framework bietet Applikationen Zugang zu Sensordaten die durch die Gerätehardware erfasst werden. Die Klasse *CMMotionManager* ist die zentrale Anlaufstelle für den Zugriff auf alle Sensoren die Daten durch die Bewegung des Endgeräts erzeugen. Somit bietet sie auch Zugang zu den Daten des Beschleunigungssensors. Wird dieser konfiguriert ist die Variable *accelerometerUpdateInterval* sehr wichtig.

```
var accelerometerUpdateInterval: NSTimeInterval
```

Die Variable legt das Intervall in Sekunden fest, in denen die Accelerometer-Daten aktualisiert werden und sollte dem Anwendungskontext entsprechend gewählt werden. Um Entwicklern Unterstützung zu bieten, die richtige Wahl zu treffen, bietet Apple die folgenden Empfehlungen [App15].

Frequenz (Hz)	Nutzung
10–20	Geeignet um die Ausrichtung des Geräts festzustellen.
30–60	Geeignet für Spiele und andere Applikationen die Eingaben des Benutzers in Echtzeit verarbeiten müssen.
70–100	Geeignet für Applikationen die Hochfrequente Bewegungen (Schütteln, Stöße) feststellen müssen.

Tabelle 4.1: Beschleunigungssensor Aktualisierungsintervall [App15]

Um die Daten des Sensors abzufragen, wird die Funktion *accelerationUpdated* genutzt. Es handelt sich dabei um eine Callback-Funktion, die im festgelegten Intervall aufgerufen wird [App15].

```
func accelerationUpdated(accelerometerData: CMAccelerometerData!, error: NSError!) -> Void
```

4.1.2 Core Location Framework

Das Core Location Framework bietet Zugriff auf die Technologien GPS und iBeacon, mit denen standortbasierte Dienste realisiert werden können. Die Funktionsweise dieser Technologien wurde bereits in den Abschnitten 2.2.3 und 2.2.3 beschrieben. Dieser Abschnitt beschreibt für die Implementierung relevante Klassen und Protokolle.

Die zentrale Anlaufstelle für Ortungsdienste ist die Klasse *CLLocationManager* und das dazugehörige Protokoll *CLLocationManagerDelegate*. Werden GPS Standortdaten gesucht, sind diese, sobald Sie verfügbar sind, über die Delegate-Methode *didUpdateLocations* verfügbar.

```
func locationManager(manager: CLLocationManager!, didUpdateLocations: [AnyObject]!)
```

Wird nach iBeacons gesucht, wird die Delegate-Methode `didRangeBeacons` aufgerufen. Diese liefert ein Array mit allen gefundenen Beacons in Empfangsreichweite. Die Beacons in diesem Array sind sortiert nach ihrer gemessenen Entfernung, beginnend mit dem Beacon das die kürzeste Entfernung aufweist.

```
func locationManager(manager: CLLocationManager!, didRangeBeacons: [CLBeacon]!, inRegion: CLBeaconRegion!)
```

4.1.3 Multipeer Connectivity Framework

Mit dem Multipeer-Connectivity-Framework können iOS und MacOS Geräte über WLAN oder Bluetooth kommunizieren. Dabei nutzen die Endgeräte jene Kommunikationskanäle, die verfügbar sind. Sind sie Teil eines Infrastruktornetzwerkes, können Sie darüber Daten austauschen. Ist keine Verbindung zu solch einem Netzwerk vorhanden, wird je nachdem, welche Technologie bei den Kommunikationspartnern aktiviert, ist eine Ad-Hoc Verbindung über Bluetooth oder WLAN aufgebaut [App13a].

Das Framework operiert in zwei Phasen, der *Discovery Phase* und der *Session Phase*. Diese werden folgend beschrieben.

Discovery Phase

In der Discovery Phase können sich Endgeräte, die in Empfangsreichweite zueinander sind, gegenseitig entdecken und eine Verbindung miteinander aufbauen. Mit den Klassen `MCNearbyServiceAdvertiser` und `MCNearbyServiceBrowser` können die Geräte Dienste anbieten und suchen. Um einen *Advertiser* oder einen *Browser* zu initialisieren müssen eine `MCPeerID` und ein `serviceType` als Parameter übergeben werden. Während die `peerID` jedes Peer identifiziert, handelt es sich beim `serviceType` um einen bis zu 15 Zeichen langen Bezeichner, der bei allen Peers, die sie sich gegenseitig entdecken möchten, übereinstimmen muss [App13a].

```
class MCNearbyServiceAdvertiser(peer: MCPeerID!, discoveryInfo: [NSObject : AnyObject!], serviceType: String!)
```

```
class MCNearbyServiceBrowser(peer: MCPeerID!, serviceType: String!)
```

Zusätzlich zu `peerId` und `serviceType` verfügt der `MCNearbyServiceAdvertiser` noch den optionalen Paramter `discoveryInfo`. Dabei handelt es sich um ein *dictionary*, in dem zusätzliche Informationen vom *Advertiser* an den *Browser* übermittelt werden

können. Ein dictionary ist ein Datentyp bei dem Daten (*values*) eines bestimmten Typs ungeordnet gespeichert werden. Diese können durch einen einzigartigen Bezeichner, auch *key* genannt, referenziert und gesucht werden [App14b]. Key und value des dictionarys müssen String Objekte sein und dürfen zusammen eine Größe von 255 Bytes nicht überschreiten. Für optimale Performance sollte das gesamte dictionary nicht größer als 400 Bytes sein, damit es in einem einzelnen Bluetooth Datenpaket untergebracht werden kann [App13b].

Advertising und Browsing

Startet ein Peer das Advertisement, wird dieses Peer für alle Browser in Empfangsreichweite sichtbar und teilt diesen Peers die Bereitschaft zum Verbindungsauftbau mit. Advertiser in Empfangsreichweite werden dem Browser über die Delegate Methode *foundPeer* des *MCNearbyServiceBrowserDelegate* Protokolls gemeldet.

```
func browser(browser: MCNearbyServiceBrowser!, foundPeer: MCPeerID!, withDiscoveryInfo: [NSObject : AnyObject]!)
```

Gefundene Peers können vom Browser zu einer *MCSession* über die Funktion *invitePeer* eingeladen werden. Eine Session verwaltet die Verbindungen und die Kommunikation zwischen den Peers. An einer einzelnen Session können bis zu 8 Peers teilnehmen. Bei der Einladung zu einer Session kann der Browser über den optionalen Parameter *context* zusätzliche Daten an den Advertiser übermitteln. Bei *context* handelt es sich um ein NSData Objekt. NSData Objekte sind Bytepuffer die typischerweise zur Datenspeicherung aber auch zum Datenaustausch zwischen Applikationen genutzt werden. Mit dem Parameter *timeout* kann zusätzlich noch festgelegt werden, wie lange der Browser auf die Antwort des Advertisers wartet [App14d].

```
func invitePeer(peer: MCPeerID!, toSession: MCSession!, context: NSData!, timeout: NSTimeInterval)
```

Ein Advertiser erhält wiederum eine Einladung von einem Browser über die Delegate Methode *didReceiveInvitationFromPeer* des *MCNearbyServiceAdvertiserDelegate* Protokolls.

```
func advertiser(advertiser: MCNearbyServiceAdvertiser!, didReceiveInvitationFromPeer: MCPeerID!, withContext: NSData!, invitationHandler: ((Bool, MCSession!) -> textcolorparameterVoid)!)
```

Dabei ist dem Advertiser die peerID des Browsers bekannt und er kann auf das NSData Objekt zugreifen, das übermittelt wurde. Mit dem invitationHandler entscheidet der Advertiser, ob er die Einladung des Browsers annimmt oder ablehnt. Wird die Einladung akzeptiert, endet die Discovery Phase und die Session Phase beginnt.

Session Phase

In der Session Phase können alle Teilnehmer der Session miteinander kommunizieren (Siehe Abbildung 4.1). Ob für die Kommunikation Bluetooth oder WLAN genutzt wird, entscheidet das Framework automatisch. Das Framework bietet auch die Möglichkeit, dass Endgeräte über ein drittes Peer kommunizieren können, sollten Sie jeweils nur Bluetooth oder WLAN aktiviert haben. Voraussetzung ist, dass dieses Peer beide Technologien aktiviert hat.

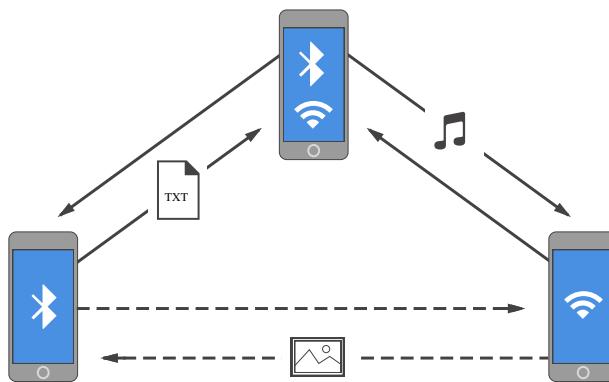


Abbildung 4.1: Session-Phase

Das Framework unterscheidet Messages, Streams und Recourcen, die zwischen Teilnehmern der Session ausgetauscht werden können. Messages sind kurze Textnachrichten oder kleine serialisierte Objekte. Ein Stream ist ein kontinuierlich übertragener Datentransfer z.B. ein Audio- oder Videostream. Bei Recourcen handelt es sich um Dateien, wie Bilder, Filme oder Dokumente [App13a].

4.2 Konzept zur Bump-Erkennung

Ein wesentliches Merkmal der Bump-Interaktion ist es, dass sich Geräte gegenseitig anstoßen, um den Verbindungsaufbau und den Datenaustausch zwischen den Geräten auszulösen. Der dabei auftretende Zusammenprall der Geräte aneinander erzeugt Kräfte, die durch Beschleunigungssensoren gemessen werden können. Auf dieser Grundlage wurde ein Algorithmus entwickelt, der die durch den Aufprall erzeugten Daten des Accelerometers auswertet, um zu erkennen, wann ein Bump stattgefunden hat.

4.2.1 Versuchsreihe Bump-Mustererkennung

Der Algorithmus wertet die vom Beschleunigungssensor erfassten Daten in einem Intervall von 10ms aus und sucht nach Mustern, die typisch für einen Bump sind. Um zu ermitteln, nach welchen Mustern der Algorithmus suchen muss, wurde für jede in Kapitel 3 identifizierte Bump-Variante eine Versuchsmessung durchgeführt, bei der die durch den Aufprall erzeugten Daten aufgenommen wurden.

Stirnseite an Stirnseite

Die Endgeräte werden waagerecht in den Händen der Anwender gehalten und werden an den Stirnseiten zusammengestoßen.

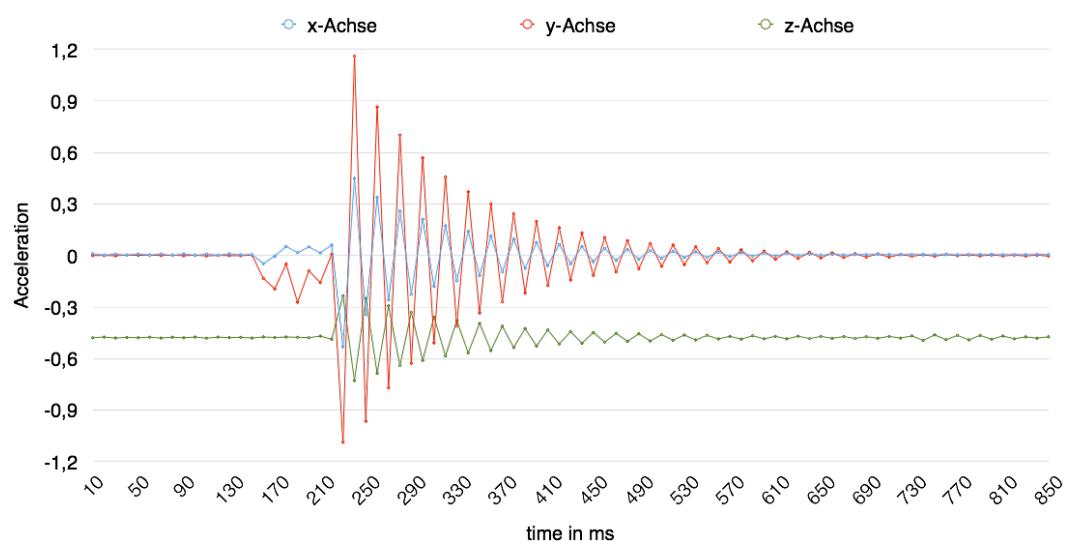


Abbildung 4.2: Bump-Muster Stirnseite an Stirnseite

Längsseite an Längsseite

Die Endgeräte werden aufrecht in den Händen der Anwender gehalten und werden an den Längsseiten zusammengestoßen.

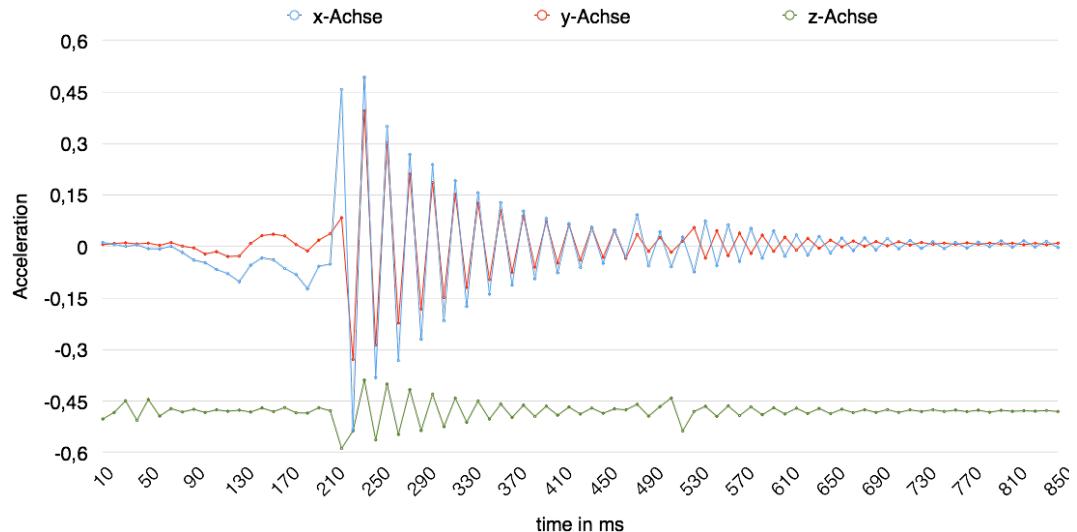


Abbildung 4.3: Bump-Muster Längsseite an Längsseite

Ecke an Ecke

Die Endgeräte werden aufrecht in den Händen der Anwender gehalten und werden an den oberen Ecken zusammengestoßen.

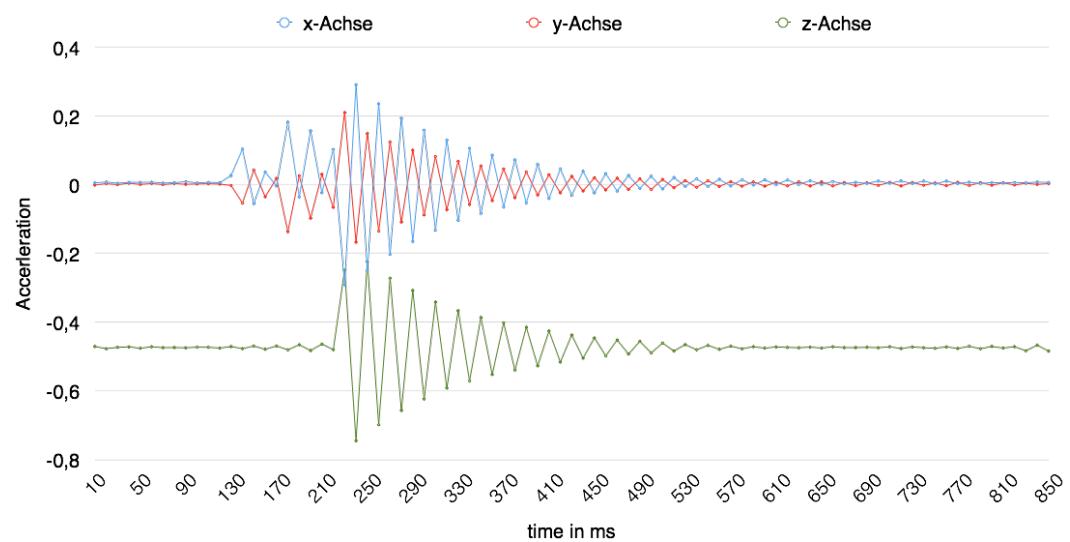


Abbildung 4.4: Bump-Muster Ecke an Ecke

Stirnseite aufstoßen

Ein Endgerät wird vom Anwender aufrecht in der Hand gehalten und mit der oberen oder unteren Kante auf ein stationäres Gerät aufgestoßen.

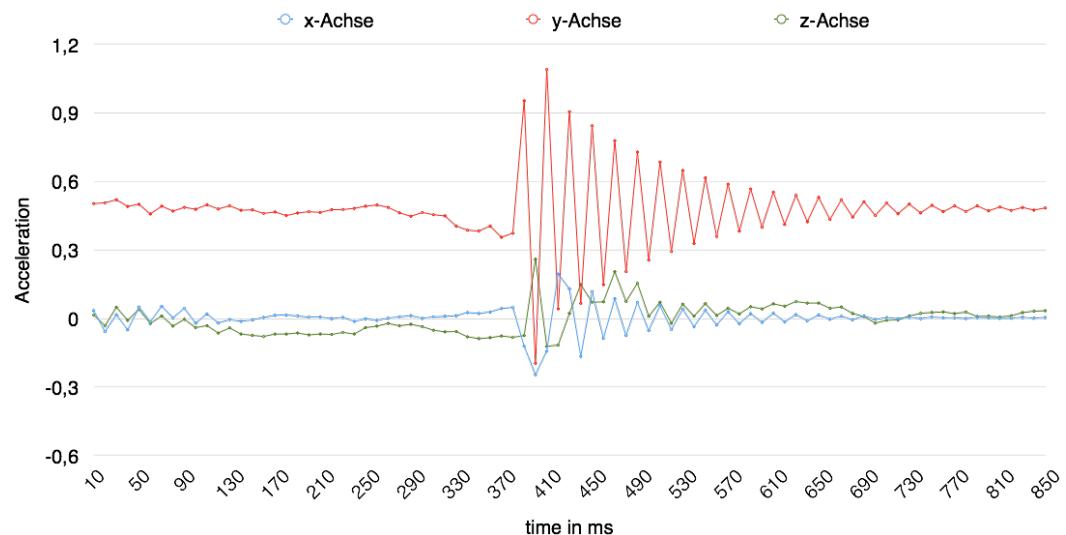


Abbildung 4.5: Bump-Muster Stirnseite aufstoßen

Erkenntnisse aus den Versuchen

Jede Bump-Variante generiert ein etwas anderes Muster. Es gibt jedoch Gemeinsamkeiten, die einen Bump typisieren und von anderen Bewegungen eines Benutzers unterscheiden. Die folgenden Charakteristiken können aus den Daten abgeleitet werden:

1. Der Aufprall generiert initial eine hoch ausschlagende Flanke.
2. Die nachfolgenden Daten sind alternierende Spitzen.

4.2.2 Algorithmus zur Mustererkennung

Basierend auf den Beobachtungen aus den Versuchen, überprüft der Algorithmus die Daten des Beschleunigungssensors in zwei Schritten. Im ersten Schritt werden die Daten aller Achsen auf hoch ausschlagenden Flanken überprüft, welche einen Indikator für einen Bump darstellen. Dafür wird jeder Wert zwischengespeichert und mit dem Folgewert der Abstand (Siehe Abbildung 4.6) zwischen den Werten berechnet.

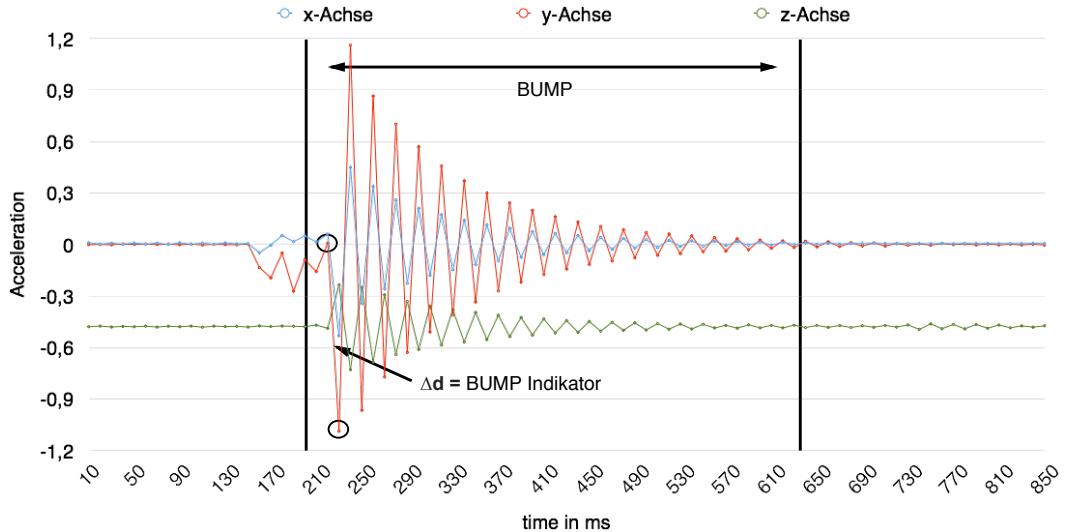


Abbildung 4.6: Algorithmische Erkennung eines Bumps

Hat der Algorithmus initiale Flanken erkannt, wird im nächsten Schritt die durch die Kraft des Bumps beeinflussten Daten in einem Vektor für jede Achse aufgezeichnet. Um zu verifizieren, dass ein Bump stattgefunden hat, läuft anschließend eine Schleife über die Vektoren und zählt die Spitzen. Sind nur Spitzen im Vektor vorhanden, handelt es sich um einen Bump.

4.3 Konzepte zur Identifizierung von Bump-Partnern

Um Daten zwischen Bump-Partnergeräten, in einem lokalen Netzwerk oder über einen Server, auszutauschen müssen die Endgeräte, aus allen Teilnehmern eines Netzwerkes bzw. allen Clients die mit einem Server verbunden sind, ihren jeweiligen Partner identifizieren. Folgend werden zwei Verfahren vorgestellt mit denen diese Identifizierung erfolgen kann.

4.3.1 Identifizierung über iBeacon Gerät-IDs

Über die Major- und Minor-Werte kann mit iBeacon jedem Endgerät eine einzigartige ID zugeordnet werden mit der Endgeräte identifiziert werden können. Um diese ID zwischen Bump-Partnern auszutauschen, wird auf beiden Geräten iBeacon zum Zeitpunkt des Bumps für kurze Zeit aktiviert. Gerade lange genug, damit die Endgeräte alle aktiven Beacons in ihrem Umfeld sehen können. Dadurch besitzt jedes Gerät eine Liste an Beacons die zu einen bestimmten Zeitpunkt an einem Bump, in ihrer Empfangsreichweite, beteiligt waren. Können die Endgeräte jeweils nur ein anderes Beacon sehen, haben Sie ihren Bump-Partner identifiziert. Ist mehr als ein Beacon sichtbar, fanden mehrere Bumps zeitgleich statt. In diesen Fällen können die Partner, über die Entfernung der Geräte zueinander, ermittelt werden. Bei den Geräten mit dem geringsten Abstand handelt es sich um die Bump-Partner. Die Erkennung der Partnergeräte über die Entfernung macht es erforderlich, dass zeitgleiche Bumps mindestens einige Zentimeter voneinander entfernt stattfinden. Dies stellt sicher, dass eine falsche Zuordnung durch ungenaue Abstandsmessungen vermieden wird.

4.3.2 Identifizierung durch charakteristische Bump-Daten

Ein Bump ist ein Ereignis, bei dem beteiligte Geräte verschiedene Daten erfassen können, die jeden Bump einzigartig charakterisieren. Eine Kombination aus Zeitpunkt und Standort bietet in den meisten Fällen eine ausreichende Datenbasis, um Bumps voneinander unterscheiden zu können. Der Zeitpunkt des Bumps kann mit einem Timestamp festgehalten werden. Bei der Nutzung von Timestamps muss jedoch beachtet werden, dass die Uhren der Geräte nicht synchron laufen. Um möglichst übereinstimmende Timestamps zu erhalten, sollte die Applikation deshalb die Systemuhr in bestimmten Zeitintervallen synchronisieren. Die Position der Endge-

räte zum Zeitpunkt des Bumps kann über GPS Koordinaten erfasst werden. Da sich Geräte bei einem Bump berühren, sollten im besten Falle die Koordinaten, die auf beiden Geräten ermittelt werden, übereinstimmen. Bei der Nutzung von GPS muss aber beachtet werden, dass nicht immer eine genaue Positionsbestimmung garantiert ist. Trotz der etwaigen Abweichungen von sowohl Timestamp als auch GPS, sollte eine Kombination dieser Daten in den meisten Fällen ausreichen, um Endgeräte zuverlässig einem Bump zuordnen zu können.

4.3.3 Vergleich der Konzepte

Der Vergleich von Timestamps und GPS-Positionsdaten sollte in den meisten Fällen zuverlässige Zuordnung von Bump-Partnern ermöglichen. Jedoch erreicht das System seine Grenzen, wenn mehrere Bumps zum selben Zeitpunkt und am selben Standort stattfinden. Die potenzielle Ungenauigkeit von GPS kann dazu führen, dass die jeweiligen Partnergeräte nicht zugeordnet werden können. Des Weiteren kann gerade im Inneren von Gebäuden dieses System unter Umständen durch den verminderten Satellitenempfang gar nicht oder nur mit Einschränkungen genutzt werden. Zum Vergleich ist das System mit iBeacon und Gerätetypen in Innenräumen besser nutzbar als GPS und ermöglicht, den Abstand zwischen Endgeräten genauer zu bestimmen. Die Identifizierung durch die Entfernungsmessung mit iBeacon besitzt aber auch ihre Grenzen. Finden zeitgleich Bumps in nur wenigen Zentimetern Entfernung zueinander statt, ist auch hier keine fehlerfreie Zuordnung garantiert. Im Vergleich ist iBeacon genauer als GPS und sollte in den Fällen in denen der Abstand der Geräte entscheidend ist die besseren Ergebnisse erzielen.

4.4 Konzepte zum Datenaustausch

Der Datenaustausch zwischen Endgeräten lässt sich über zwei verschiedene Ansätze realisieren. Zum einen können Daten direkt zwischen den Geräten über ein lokales Netzwerk getauscht werden und zum anderen über einen zentralen Server. Für beide Varianten werden folgend Konzepte vorgestellt, in denen sich Endgeräte identifizieren und Daten austauschen können.

4.4.1 Datenaustausch über lokale Netzwerke

Mit Hilfe von drahtlosen Netzwerktechnologien kann Datenaustausch zwischen Endgeräten entweder über ein WLAN-Infrastrukturnetzwerk oder durch die Bildung eines Ad-Hoc Netzwerkes realisiert werden. Unabhängig von der Topologie des Netzwerkes läuft die Kommunikation zwischen den Netzwerkteilnehmern über Netzwerkdienste. Entscheidend ist dabei, dass sich die an einem Bump-Event beteiligten Geräte im Computernetzwerk gegenseitig identifizieren können. Folgend wird einleitend beschrieben wie die beiden in 4.3 beschriebenen Konzepte in lokalen Netzwerken angewendet werden. Anschließend werden für beide Möglichkeiten der genaue Ablauf des Verbindungsaufbaus zwischen den Geräten beschrieben.

Geräteidentifizierung in lokalen Netzwerken

Es werden zwei Maßnahmen ergriffen, mit denen sichergestellt werden soll, dass sich die an einem Bump beteiligten Geräte zuverlässig identifizieren. Im ersten Schritt wird dazu die Anzahl an Netzwerkteilnehmern, die den selben Netzwerkdienst zeitgleich verwenden, verringert. Dazu wird der Netzwerkdienst, der von der Bump-Applikation angeboten wird, erst gestartet, nachdem das Bump-Event stattgefunden hat und wieder gestoppt, sobald zwischen den Bump-Partnern eine Verbindung aufgebaut wurde. Dadurch ist der Netzwerkdienst nur für kurze Zeit bei jenen Endgeräten aktiv, die zum gleichen Zeitpunkt einen Bump durchgeführt haben. Der Zeitraum, in dem der Netzwerkdienst von den Bump-Partner angeboten und gesucht wird, ist sehr kurz, in den meisten Fällen werden deswegen immer nur Bump-Partner gleichzeitig den selben Netzwerkdienst nutzen, wodurch eine weitere Identifizierung gar nicht nötig wäre (Siehe Abbildung 4.7 a).

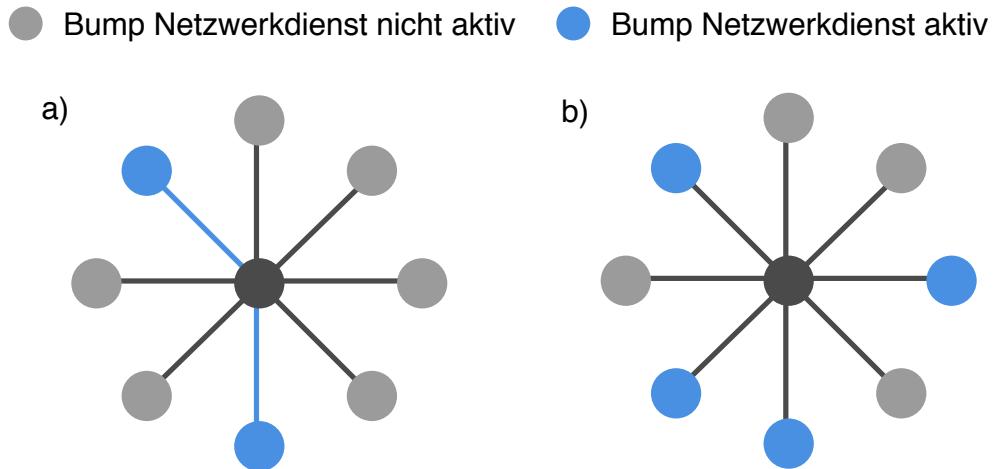


Abbildung 4.7: Netzwerkdienste in einem Infrastrukturnetzwerk

Da jedoch nicht sichergestellt werden kann, dass in der Zeitspanne in der Bump-Partner den Netzwerkdienst nutzen, kein weiterer Bump auftritt und dadurch mehr als zwei Engeräte den Netzwerkdienst nutzen (Siehe Abbildung 4.7 b), wird zusätzlich noch eine der in Abschnitt 4.3 beschriebenen Methode benötigt, mit der sich Bump-Partner im Netzwerk identifizieren können (Siehe Abbildung 4.8).

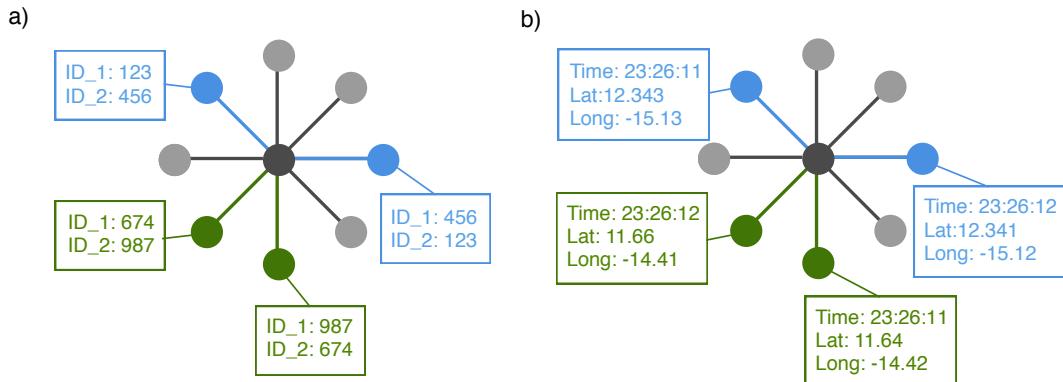


Abbildung 4.8: Identifizierung von Netzwerkteilnehmern

Ablauf mit iBeacon Geräte-IDs

Wird auf einem Gerät ein Bump registriert, ist der erste Schritt die Generierung von Zufallszahlen für den Major- und Minor-Wert von iBeacon. Diese Zahlen bilden eine eindeutige ID, mit der sich jedes Gerät im Netzwerk identifizieren kann. Anschließend wird iBeacon aktiviert, die Geräte können sich gegenseitig sehen, GeräteIDs lesen und die Distanz zu allen sichtbaren Beacons erfassen. Ist mehr als ein Beacon sichtbar, wird jenes ermittelt, welches die geringste Distanz zum suchenden Gerät aufweist. Die GeräteID dieses Geräts wird lokal gespeichert und iBeacon wird deaktiviert. Anschließend startet die DiscoveryPhase des Multipeer-Connectivity-Frameworks. In dieser Phase agiert das Gerät, dessen Minor Wert den kleineren Zahlenwert aufweist, als Advertiser und bietet einen Netzwerkdienst an. Die discoveryInfo dieses Dienstes enthält die durch Major und Minor gebildete GeräteID des Advertisers. Das Gerät mit dem höheren Zahlenwert ist der Browser und sucht den Dienst des Advertisers. Diese Aufgabenverteilung verhindert, dass Probleme beim Verbindungsauftreten auftreten, sollten sich die Geräte gleichzeitig zu einer Session einladen. Findet der Browser diesen Dienst, kann er die GeräteID des Advertisers mit der GeräteID vergleichen, die das zu Beginn ermittelte Beacon aufweist. Stimmen die IDs überein, handelt es sich bei dem Advertiser um den Bump-Partner und der Browser kann eine Session erstellen und eine Einladung zu dieser an den Advertiser schicken. Mit dieser Einladung wird in der contextData wiederum die GeräteID des Browsers mitgeschickt. Damit kann der Advertiser überprüfen, ob er auch wirklich von seinem Partner eingeladen wird und nicht von jemand anderem. Haben die Geräte eine Verbindung aufgebaut, können Sie über die Methoden des Frameworks die gewünschten Daten austauschen, und anschließend die Session verlassen.

Der beschriebene Ablauf kann auch noch einmal im folgenden UML Diagramm nachvollzogen werden.

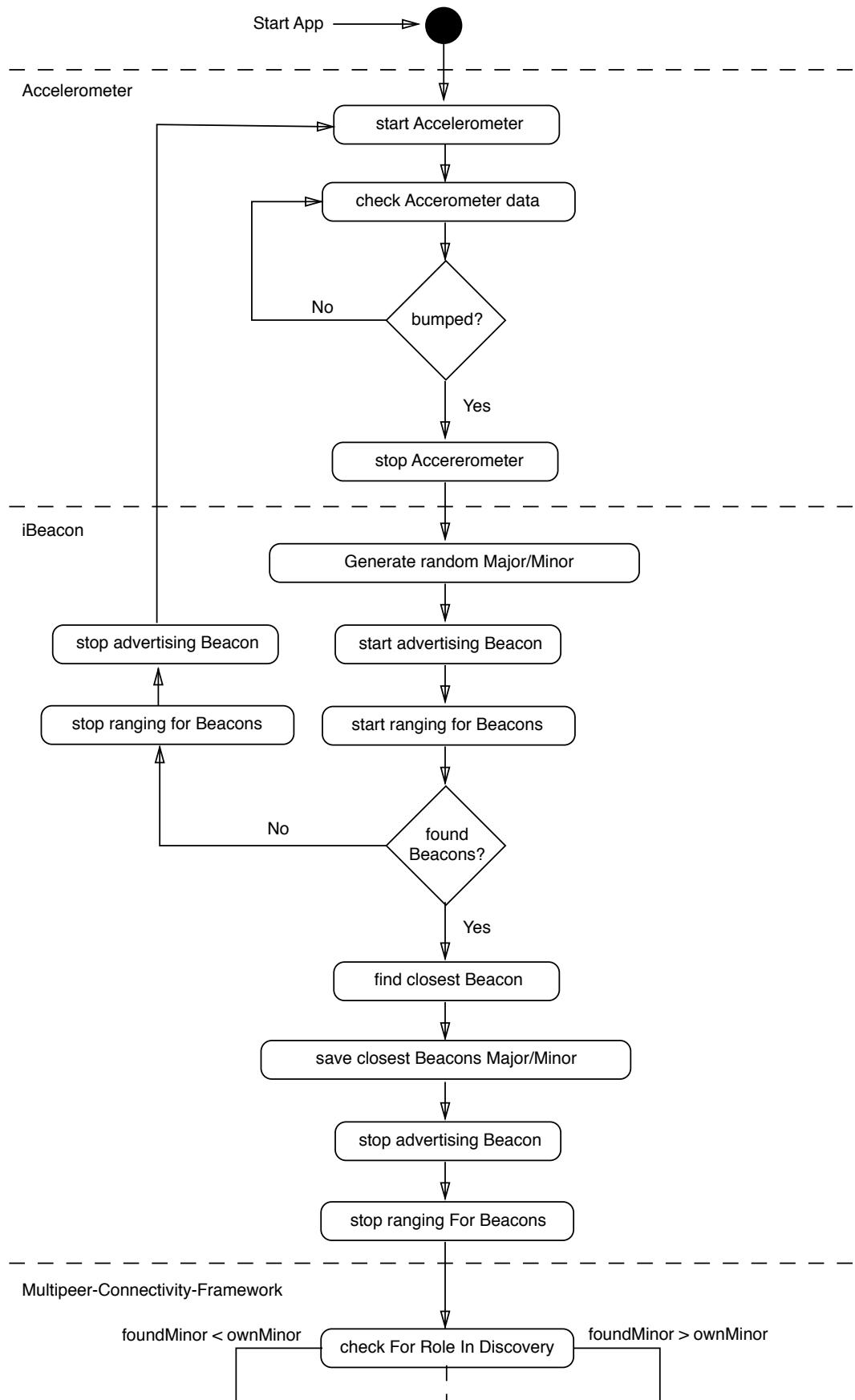
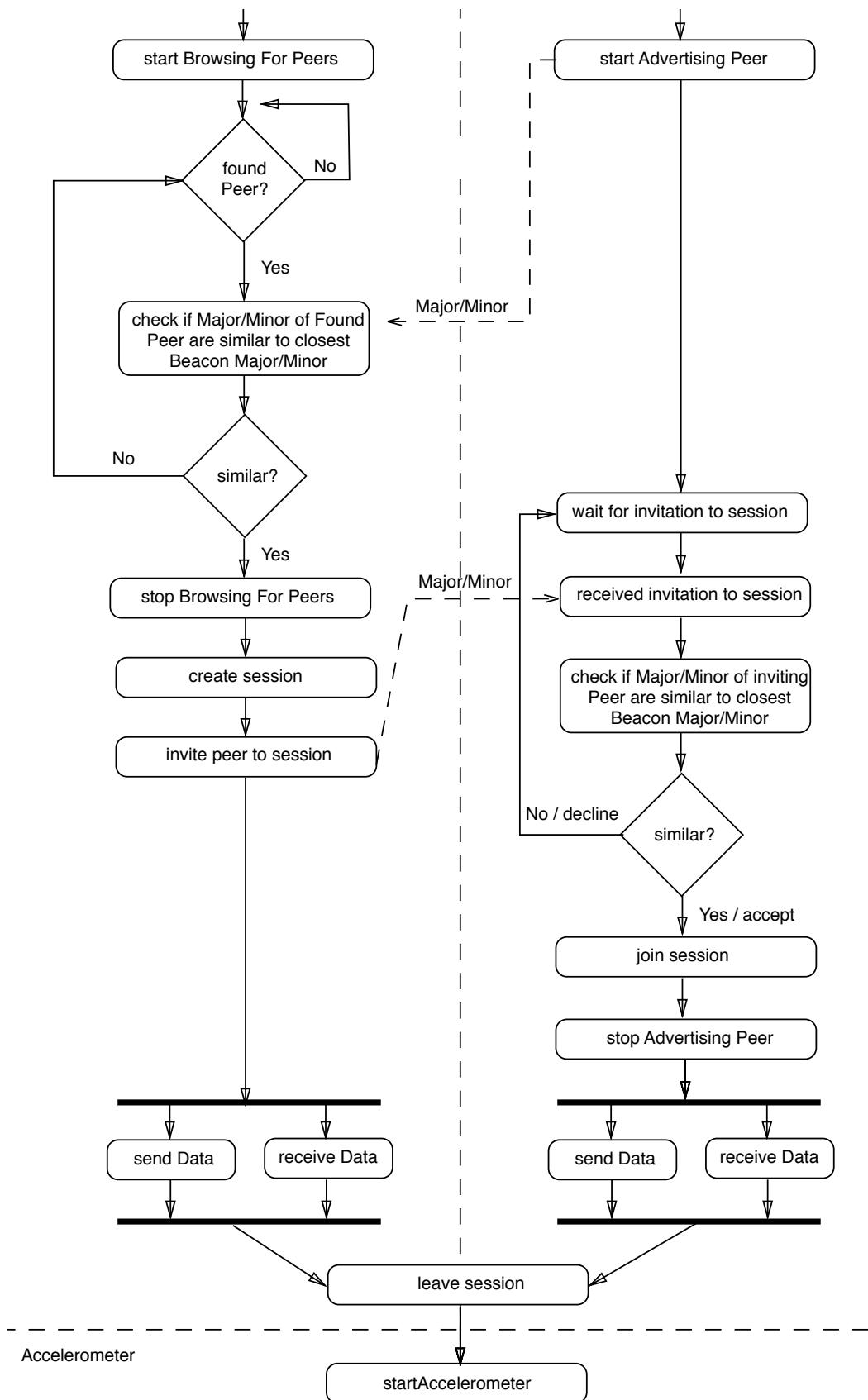


Abbildung 4.9: System mit iBeacon UML Diagramm

4 Umsetzung der Interaktion



Ablauf mit Timestamp und GPS

Mit Timestamp und GPS läuft der Verbindungsauflauf etwas anders ab, als mit iBeacon. Zuerst werden die GPS Daten der Endgeräte erfasst und lokal gespeichert. Im nächsten Schritt eröffnen beide Endgeräte einen Browser- und Advertiser-Dienst mit einem gemeinsamen Bezeichner. Die zuvor gespeicherten GPS Daten werden bei beiden Geräten in der discoveryInfo zusammen mit einer generierten Zufallszahl untergebracht. Beide Geräte können nun auf Timestamp und GPS Daten der gefundenen Geräte zugreifen und Timestamp und GPS Daten mit den eigenen abgleichen. Stimmen die gefundenen Daten mit den eigenen überein, wurde identifiziert welches Gerät zu einer Session eingeladen werden muss. Wer diese Session erstellt und den Partner zu dieser einlädt, wird über die generierte Zufallszahl die ebenfalls in der discoveryInfo gespeichert wird, ausgehandelt. Nachdem die Session zustande gekommen ist, werden die Netzwerkdienste deaktiviert und der Datenaustausch kann erfolgen.

Der beschriebene Ablauf ist im folgenden UML Diagramm noch einmal dargestellt.

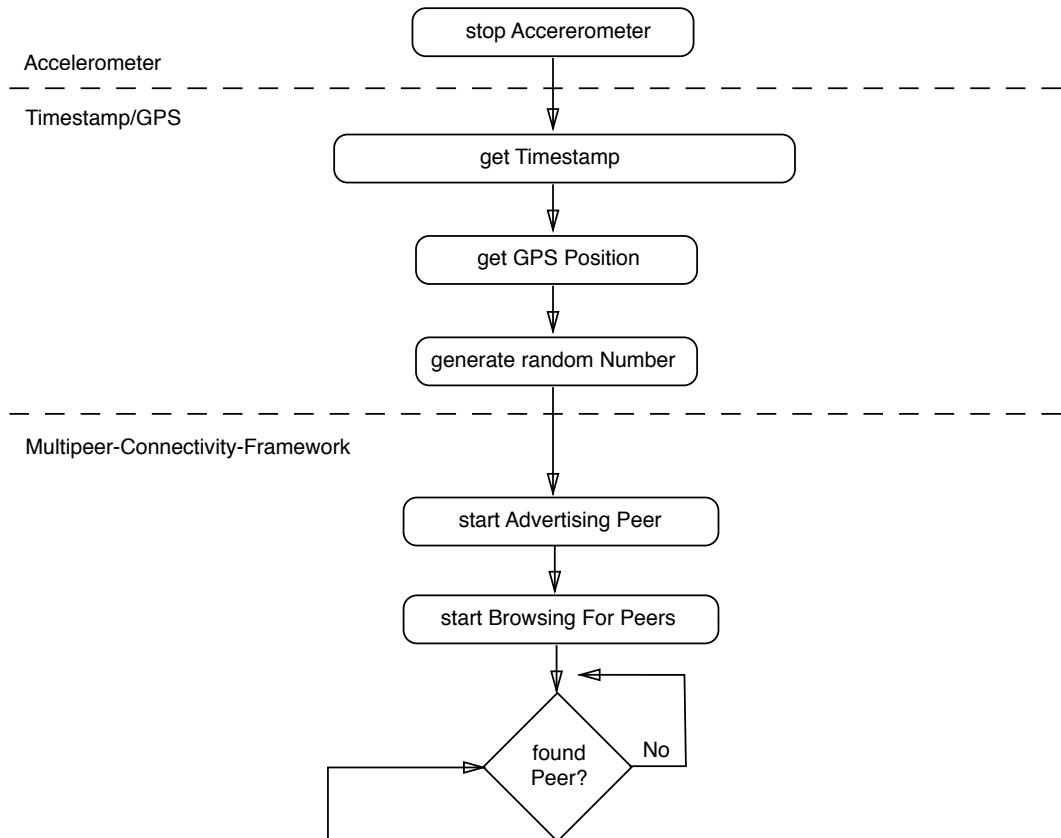


Abbildung 4.11: System mit Timestamp und GPS UML Diagramm

4 Umsetzung der Interaktion

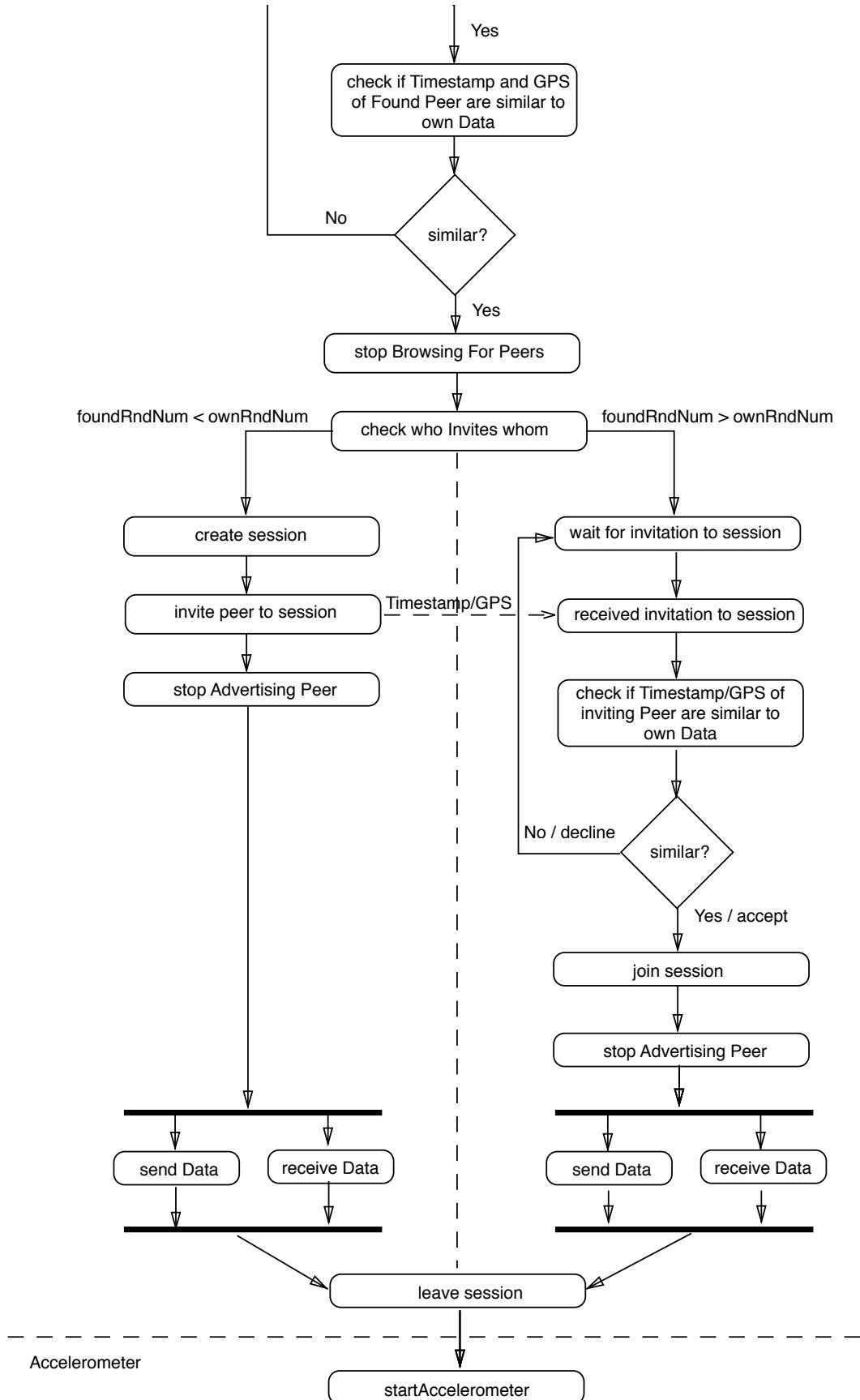


Abbildung 4.12: System mit Timestamp und GPS UML Diagramm

4.4.2 Datenaustausch über Webserver

Die Kommunikation der Endgeräte über Webserver ist eine alternative Möglichkeit den Datenaustausch zwischen den Endgeräten zu implementieren. Im folgend beschriebenen System werden zwei Server genutzt, die nach verschiedenen Aufgaben aufgeteilt sind. Einer der Server, im folgenden als Fileserver bezeichnet, dient als Austausch Plattform für die von den Anwendern ausgewählten Daten. Diese werden von den Endgeräten auf den Server hochgeladen und sind dort anschließend unter einer bestimmten URL erreichbar. Für die Übermittlung der URL von Sender zu Empfänger ist der zweite Server, im folgenden als Messaging-Server bezeichnet, zuständig. Mit diesem Server verbinden sich die Bump-Partner über Web-Sockets und sind damit Teil eines Nachrichtensystems, mit dem der Server Nachrichten von allen verbundenen Endgeräten empfangen, aber auch senden kann. Damit der Matchmaking-Server die Nachrichten an die richtigen Geräte weiterleiten kann, müssen diese wie auch in lokalen Netzwerken identifiziert werden. Wie die Kommunikation mit den einzelnen Servern genau abläuft, wird folgend erklärt.

Ablauf der Kommunikation

Verbinden sich Clients über WebSockets mit einem Server, wird jeder Socket mit einer eigenen SocketId erstellt. Über diese SocketID kann der Server einem bestimmten Client gezielt Nachrichten zuschicken. Da die Bump-Partnergeräte die SocketID des anderen nicht kennen, muss die SocketID auf dem Server mit zusätzlichen Informationen verknüpft werden, durch die der Server Bump-Partner einander zuordnen kann. Aus diesem Grund senden die Clients beim verbinden zum Server ihre GerätetIDs oder Timestamp und GPS an den Server. Dieser erstellt anschließend ein User Objekt, speichert in diesem die SocketID zusammen mit den Identifikationsinformationen und fügt es einer Liste mit allen verbundenen Clients hinzu. Möchten Endgeräte ihren Partnern eine Nachricht schicken, müssen sie z.B. nur die GerätetID ihrer Partners mitsenden und der Server kann anhand dieser ID die SocketID über das User Objekt ermitteln und die Nachricht weiterleiten. Sobald sich beide Partnergeräte verbunden haben, informiert der Server die jeweiligen Clients darüber. Daraufhin können die Endgeräte die vom Anwender ausgewählten Daten auf den Fileserver hochladen, der ihnen, sobald der upload abgeschlossen ist, eine URL zurückliefert, an der die Daten gespeichert sind. Diese URL kann anschließend über den Messaging-Server an das Partnergerät übermittelt werden, das darauf den Download vom Fileserver starten kann. Sobald die Endgeräte jeweils ihre Downloads

abgeschlossen haben, können Sie sich vom Messaging-Server abmelden und der Datenaustausch ist abgeschlossen. Im folgenden ist ein kleiner Codeausschnitt zu finden, in dem exemplarisch einige Funktionen zur WebSocket Kommunikation für den beschriebenen Ablauf dargestellt sind. Außerdem kann der beschriebene Ablauf noch einmal im Sequenzdiagramm (4.13) nachvollzogen werden.

```
// on the server
var users = {} // all connected users

// Identify device and add to users at connection
io.sockets.on('connection', function (socket) {
    socket.emit('who are you');
    socket.on('check in', function (incoming) {
        users[incoming.majorMinorID] = socket.id;
    });
});
// Get socketID from user array and send URL to device
socket.on('message to partner', function(msg) {
    io.sockets.socket(users[msg.deviceID]).emit(msg.URL);
});
});

// on the client
// Tell Server my deviceID
socket.on('who are you', function (incoming) {
    socket.emit('check in', {devideID: beaconMajor + beaconMinor});
});
// send URL to Partner
socket.emit('message to partner', {minorMajorID : "1233552345", url: "URL"});
```

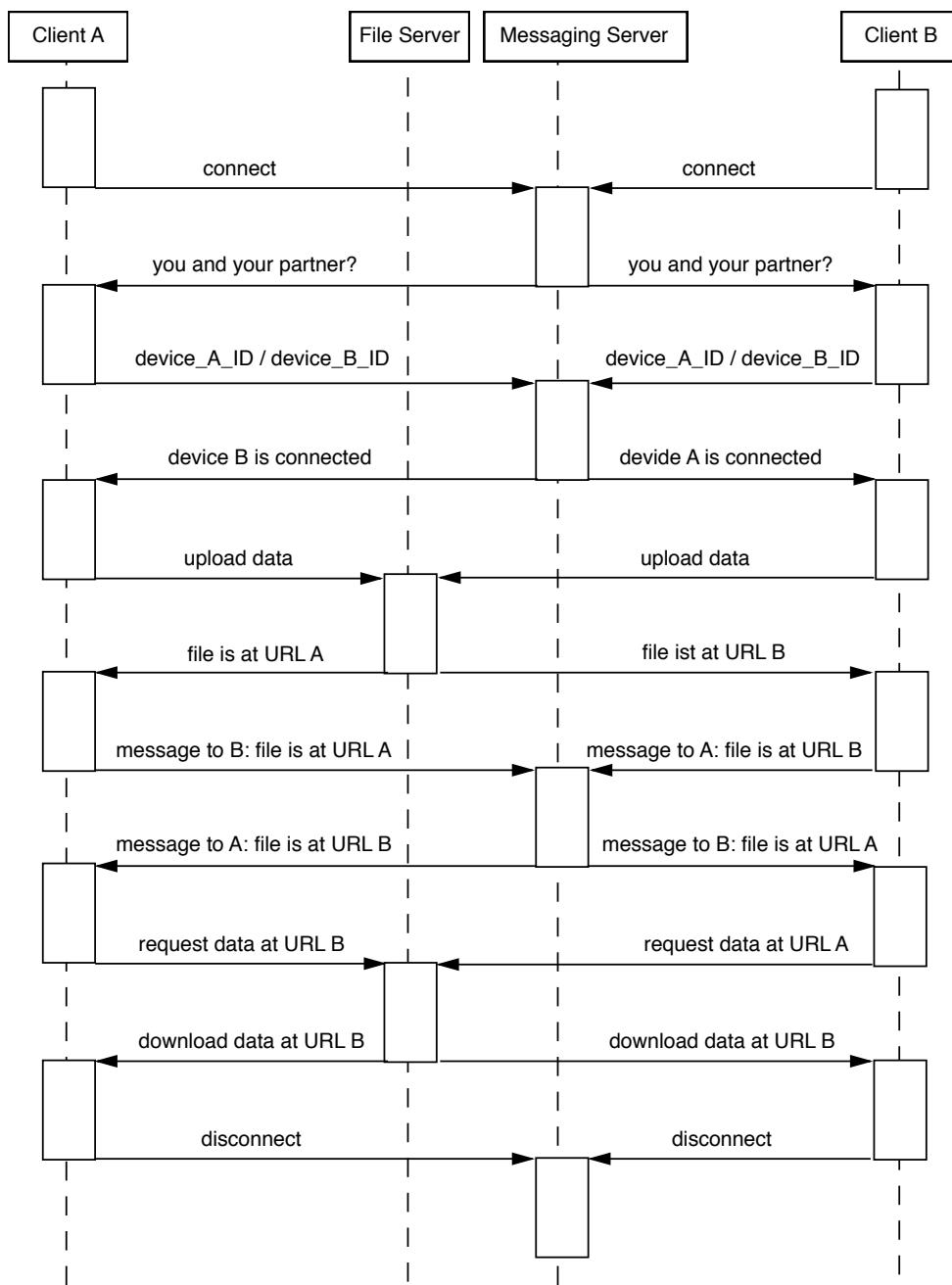


Abbildung 4.13: Sequenzdiagramm Datenaustausch über Webserver

4.4.3 Vergleich der Konzepte

Direkte Kommunikation bietet den Vorteil, ohne zusätzliche Server Infrastruktur zu operieren und es wird, falls der Server nur über das Internet erreichbar ist, auch keine Internetverbindung benötigt. Nachteilig ist jedoch der höhere Implementierungsaufwand, um für jede Plattform ein System anzubieten. Die von den Plattformen genutzten Technologien und Frameworks sind teilweise nicht kompatibel, was es erforderlich macht, für jede Plattform Einzellösungen zu implementieren. Wird dagegen eine Serverarchitektur gewählt, ist der Implementierungsaufwand wesentlich geringer, da die meiste Implementierung nur einmal für den Server erfolgen muss. Nachteilig ist der erforderliche Zugang zu entweder einem lokalen WLAN oder zu mobilem Internet. Weiterhin handelt es sich bei einem Server um einen Single-Point-Of-Failure. Fällt der Server aus, funktioniert das komplette System nicht mehr. Hier bieten lokale Lösungen einen klaren Vorteil, da diese ausfallsicher sind. Weiterhin ist der Betrieb eines Servers auch mit Kosten verbunden die in einem lokalen System nicht anfallen.

Kapitel 5

Demonstrator Applikation

Als praktischer Teil dieser Arbeit wurde das Netzwerk basierte System implementiert. Mit der Applikation können zwei Anwender auf Endgeräten gespeicherte Fotos auswählen und durch einen Bump übertragen.

Die Implementierung wurde auf der iOS Plattform in der Programmiersprache Swift durchgeführt und ist als Git Repository erreichbar unter:

<https://github.com/informatik-mannheim/thesis-bump-to-transfer/tree/master/sources/Bumper>

Für das User-Interface der Applikation wurde ein Wireframe erstellt, das als PDF aufzufinden ist unter:

<https://github.com/informatik-mannheim/thesis-bump-to-transfer/blob/master/sources/Wireframe.pdf>

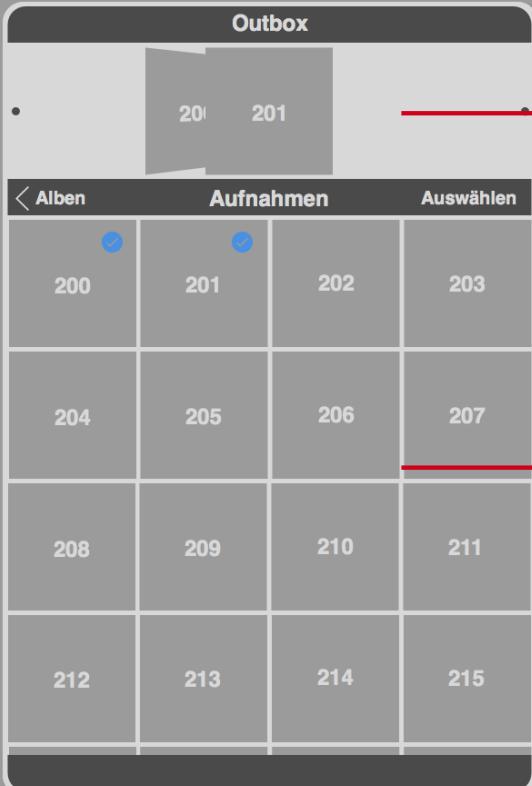
Folgend wird das Wireframe in Auszügen dargestellt um einen Einstieg in die Nutzung der Applikation zu bieten. Auf den Screenshots ist sichtbar welche Interaktionsmöglichkeiten dem Anwender, vor und nach der Durchführung der Interaktion, geboten werden. Von besonderem Interesse ist dabei zum einen wie der Anwender gespeicherten Inhalte durchsuchen und für den Versand auswählen kann und zum anderen wie empfangene Daten angezeigt und gespeichert werden können.

Profilbild Bump-Partner
Anwender können anhand des Profilbilds überprüfen, ob Sie mit dem Endgerät der gewünschten Person verbunden sind.



Connection
Besteht eine Verbindung zwischen zwei Endgeräten, sind die Anwender symbolisch durch ein Kabel mit Stecker verbunden.

Auswahl Übersicht
Alle aufgewählten Fotos werden in einer Cover-Flow Ansicht angezeigt, damit der Anwender jederzeit Überblick über ausgewählte Inhalte hat.



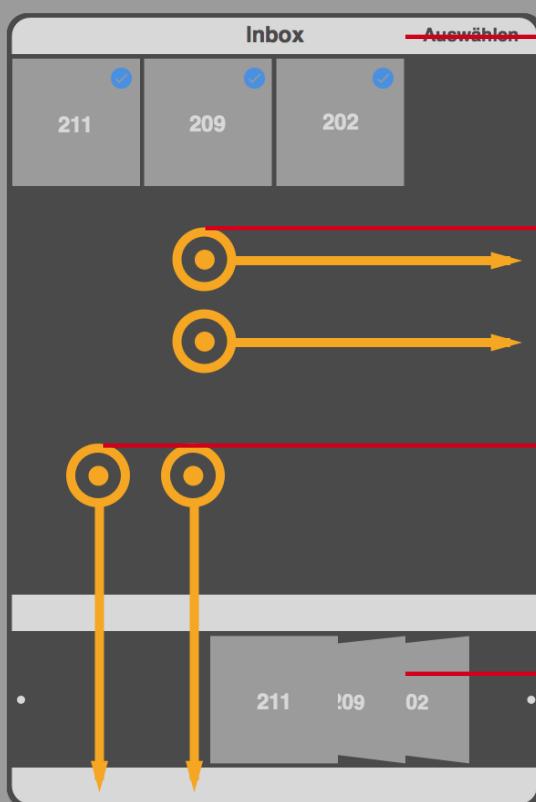
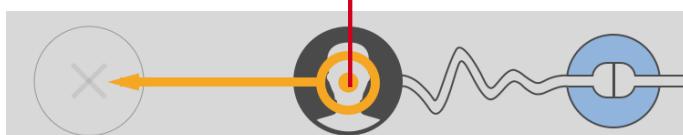
Content Browser
Alle auf dem Endgerät verfügbaren Fotos können durchsucht und ausgewählt werden.

Profilbild Gerätbesitzer
Jeder Anwender besitzt ein Profilbild, über das er vom Gegenüber identifiziert werden kann.



Disconnect

Verbindungen können getrennt werden, indem das Profilbild lang links gezogen und damit der Stecker gezogen wird.

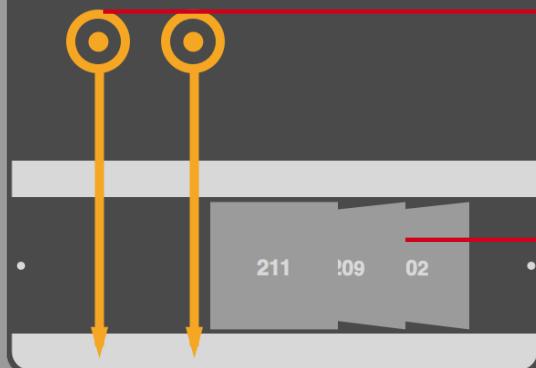


Inbox

Posteingang, in dem empfangene Fotos angezeigt werden.

Geste zum wechseln

Zwei Finger Swipe nach rechts oder links dreht den Content Browser und wechselt auf Postein- oder ausgang.

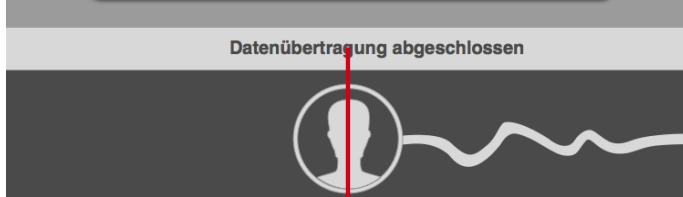


Geste zum speichern

Zwei Finger Swipe nach unten führt einen virtuellen bump des eigenen Profilbilds aus, durch das die Auswahl gespeichert wird.

Auswahl zum speichern

Aus den empfangenen Fotos ausgewählt, um gespeichert zu werden.



Nachrichtenzentrum

Systemnachrichten an den Anwender werden hier angezeigt.

Kapitel 6

Zusammenfassung und Ausblick

Das zu Beginn dieser Arbeit formulierte Ziel wurde durch die folgenden Teilschritte erreicht: Zu Beginn wurde beschrieben, in welchen Varianten die Bump-Geste durchgeführt werden kann und welche Kategorien von Endgeräten für die Geste geeignet sind. Es wurden anschließend Technologien identifiziert, die eine Umsetzung der Geste auf mobilen Endgeräten ermöglicht. Für die Erkennung von Bumps wurde untersucht welche Daten von einem Beschleunigungssensor durch einen Bump erzeugt werden. Basierend auf diesen Erkenntnissen wurde ein Algorithmus entwickelt der Bump charakteristische Muster in Sensordaten erkennt. Für die Datenübertragung wurde ein Konzept entwickelt, mit dem Daten, entweder über vorhandene WLAN-Netzwerke oder durch die Bildung von AD-Hoc Netzen, ausgetauscht werden. Des weiteren wurde ein Konzept entwickelt, in dem Daten zwischen den Endgeräten über einen Server ausgetauscht werden können. Damit der Datenaustausch zwischen den richtigen Geräten erfolgt, wurden zwei Konzepte entwickelt mit denen Endgeräte in lokalen Netzwerken und auf Servern identifiziert werden können. Diese funktionieren zum einen über die Erfassung und den Vergleich von Timestamp und GPS und zum anderen über die Identifizierung über iBeacon GeräteIDs. Als praktischer Teil der Arbeit wurde eine Applikation auf der iOS Plattform implementiert mit der sich Endgeräte über iBeacon identifizieren und über lokale WLAN Netzwerke oder AD-Hoc Netzwerke Bilddateien austauschen können.

Aufbauend auf den Ergebnissen dieser Arbeit sind einige weiterführende Arbeiten denkbar. Es wäre interessant, auch eine Applikation auf der Android Plattform zu implementieren und dort zu prüfen, ob auch NFC anstatt iBeacon zur Identifizierung der Partnergeräte genutzt werden kann. Ebenfalls von Interesse wäre eine zusätzli-

6 Zusammenfassung und Ausblick

che Implementierung eines Server basierten Systems um die Leistungsfähigkeit der Systeme vergleichen zu können.

Weitere Arbeiten zur Usability der Interaktion sind ebenfalls möglich. Es muss noch erforscht werden wie stark ein Bump sein muss, damit die Interaktion für den Anwender ein angenehmes Nutzergefühl besitzt. In diesem Zusammenhang sollte der Algorithmus zur Bump-Erkennung auch auf seine Zuverlässigkeit überprüft und optimiert werden. Gerade bei Bewegungen die Bump ähnliche Muster erzeugen muss die Erkennung noch optimiert werden, da dort häufig fälschlicherweise Bumps erkannt werden. Es stellt sich auch die Frage ob die Bump-Interaktion für Anwender im Kontext der Datenübertragung überhaupt intuitiv ist und welche Anwendungskontexte generell vom Anwender mit der Interaktion assoziiert werden. So könnte die Interaktion aus Anwendersicht eventuell besser geeignet sein um die Displays mobiler Endgeräte durch das Zusammenstoßen aneinander zu klonen oder zu erweitern.

Abkürzungsverzeichnis

API Application Programming Interface

CRUD Create, Read, Update, Delete

GPS Global Positioning System

HDSPA+ High Speed Packet Access

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

iOS iPhone Operating System

LTE Long Term Evolution

MacOS Macintosh Operating System

NFC Near Field Communication

RSSI Received Signal Strength Indication

UMTS Universal Mobile Telecommunications System

URL Uniform Resource Identifier

UUID Universally Unique Identifier

WLAN Wireless Local Area Network

Tabellenverzeichnis

2.1	Einsatz von Beacons in einer Einzelhandelskette [Com14]	17
3.1	Kategorisierung von Endgeräten	24
4.1	Beschleunigungssensor Aktualisierungsintervall [App15]	28

x

Abbildungsverzeichnis

2.1	Bump Tablet-Computer [Hin03]	6
2.2	Daten ausschütten [Chr11]	6
2.3	Bump-Geste Smartphones [Lan14]	6
2.4	Bump-Geste PC [bum13]	6
2.5	Wurf- und Fang-Geste [YTH ⁺ 06]	7
2.6	Schwung-Geste [YTH ⁺ 06]	7
2.7	Hoccer Swipe-Geste [sal]	7
2.8	Links kippen zum weiterblättern [LHJ11]	8
2.9	Schwung-Geste Auswahl mehrerer Ziele [LHJ11]	8
2.10	WLAN-Topologien [Bau12, vgl. Abb. 3.1]	9
2.11	Bluetooth-Topologie Piconetz [Bau12, vgl. Abb. 5.4]	11
2.12	Netzwerkdienste in Computernetzwerken	13
2.13	Client-Server-Modell	13
2.14	iBeacon regions	16
2.15	iBeacon ranging	18
2.16	3-Achsen Beschleunigungssensor [App15]	19
3.1	Aktion des Benutzers	22
3.2	Daten übertragen	22
3.3	Daten anzeigen	22
3.4	Stirnseite an Stirnseite	23
3.5	Längsseite an Längsseite	23
3.6	Ecke an Ecke	23
3.7	Stirnseite aufstoßen	23
4.1	Session-Phase	31
4.2	Bump-Muster Stirnseite an Stirnseite	32
4.3	Bump-Muster Längsseite an Längsseite	33
4.4	Bump-Muster Ecke an Ecke	33
4.5	Bump-Muster Stirnseite aufstoßen	34
4.6	Algorithmische Erkennung eines Bumps	35
4.7	Netzwerkdienste in einem Infrastrukturnetzwerk	39
4.8	Identifizierung von Netzwerkteilnehmern	39
4.9	System mit iBeacon UML Diagramm	41
4.10	System mit iBeacon UML Diagramm	42

Abbildungsverzeichnis

4.11 System mit Timestamp und GPS UML Diagramm	43
4.12 System mit Timestamp und GPS UML Diagramm	44
4.13 Sequenzdiagramm Datenaustausch über Webserver	47

Literaturverzeichnis

- [3GPa] 3GPP. Hdspa+. @ONLINE. Zuletzt besucht am 15.12.2014.
- [3GPb] 3GPP. Lte. @ONLINE. Zuletzt besucht am 15.12.2014.
- [AB12] Pankaj Agrawal and Sharad Bhuraria. Near field communication. *IT Matters*, 67, 2012.
- [App13a] Apple Computer. About multipeer connectivity. @ONLINE, September 2013. Zuletzt besucht am 04.01.2015.
- [App13b] Apple Computer. Mcnearbyserviceadvertiser. @ONLINE, 2013. Zuletzt besucht am 11.02.2015.
- [App14a] Apple Computer. About bonjour. @ONLINE, April 2014. Zuletzt besucht am 19.02.2015.
- [App14b] Apple Computer. Collection types. @ONLINE, 2014. Zuletzt besucht am 11.02.2015.
- [App14c] Apple Computer. ios 7: Grundlagen zu den ortungsdiensten. @ONLINE, June 2014. Zuletzt besucht am 15.12.2014.
- [App14d] Apple Computer. Nsdata. @ONLINE, September 2014. Zuletzt besucht am 11.02.2015.
- [App15] Apple Computer. Motion events. @ONLINE, February 2015. Zuletzt besucht am 15.12.2014.
- [Bau12] C. Baun. *Computernetze kompakt*. IT kompakt. Springer Berlin Heidelberg, 2012.
- [bum13] bump technologies. Bump your computer. @ONLINE, February 2013. Zuletzt besucht am 14.02.2015.
- [Che02] Liang Cheng. Service advertisement and discovery in mobile ad hoc networks. In *Proc. of CSCW*. Citeseer, 2002.
- [Chr11] Christian Remse. Diese neuen technologien plant apple. @ONLINE, August 2011. Zuletzt besucht am 15.02.2015.

- [Com14] Apple Computer. *Getting Startet with iBeacon*. Apple Computer, 1 edition, 6 2014.
- [Est14] Estimote, Inc. Beacons' signal characteristics. @ONLINE, 2014. Zuletzt besucht am 09.01.2015.
- [Goo12] Google Inc. The new multi-screen world: Understanding cross-plattform consumer behavior. @ONLINE, August 2012. Zuletzt besucht am 13.10.2014.
- [Goo15] Google. Network service discovery. @ONLINE, 2015. Zuletzt besucht am 21.02.2015.
- [Hin03] Ken Hinckley. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 149–158, New York, NY, USA, 2003. ACM.
- [Lan14] Landon Robinson. Bump and flock apps being discontinued this month. @ONLINE, January 2014. Zuletzt besucht am 14.02.2015.
- [LG10] Peter Lubbers and Frank Greco. Html5 web sockets: A quantum leap in scalability for the web. *SOA World Magazine*, 2010.
- [LH07] Noam Lando and Amit Haller. Efficient server polling system and method, September 28 2007. US Patent App. 11/864,271.
- [LHJ11] Andrés Lucero, Jussi Holopainen, and Tero Jokela. Pass-them-around: Collaborative use of mobile phones for photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1787–1796, New York, NY, USA, 2011. ACM.
- [Maa11] Maarten Jansen. App review: Hoccer. @ONLINE, 2011. Zuletzt besucht am 14.02.2015.
- [RDML05] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *AAAI*, volume 5, pages 1541–1546, 2005.
- [Rod08] Alex Rodriguez. Restful web services: The basics. *IBM developerWorks*, 2008.
- [sal] salesworker.com GmbH. Hoccer. @ONLINE. Zuletzt besucht am 15.02.2015.
- [SS12] Alexander Schill and Thomas Springer. *Verteilte Systeme: Grundlagen und Basistechnologien*. Springer-Verlag, 2012.

- [TQD09] Lucia Terrenghi, Aaron Quigley, and Alan Dix. A taxonomy for and analysis of multi-person-display ecosystems. *Personal Ubiquitous Comput.*, 13(8):583–598, November 2009.
- [WHBW11] Hanno Wirtz, Tobias Heer, Robert Backhaus, and Klaus Wehrle. Establishing mobile ad-hoc networks in 802.11 infrastructure mode. In *Proceedings of the 6th ACM workshop on Challenged networks*, pages 49–52. ACM, 2011.
- [Wik14a] Wikipedia. Global positioning system — wikipedia, die freie enzyklopädie, 2014. [Online; Stand 15. Dezember 2014].
- [Wik14b] Wikipedia. Ibeacon — wikipedia, the free encyclopedia, 2014. [Online; Stand 15-December-2014].
- [Wik14c] Wikipedia. Wi-fi direct — wikipedia, the free encyclopedia, 2014. [Online; Stand 28-October-2014].
- [YTH⁺06] Koji Yatani, Koiti Tamura, Keiichi Hiroki, Masanori Sugimoto, and Hiromichi Hashizume. Toss-it: Intuitive information transfer techniques for mobile devices using toss and swing actions. *IEICE - Trans. Inf. Syst.*, E89-D(1):150–157, January 2006.

