ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Evaluation und Klassifikation der Anforderungen an Datenverbindungen mobiler Endgeräte

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Till Fischer
till.fischer@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Nicolas Mundbrod

2013

Fassung 2. Oktober 2013

Satz: PDF-LaTeX 2ε

1 Kurzfassung

Mobile Endgeräte erfahren seit einigen Jahren ein sehr starkes Wachstum. Über eine ständige Verbindung mit dem Internet bieten sie eine ideale Plattform für eine Vielzahl an Informations-, Kommunikations- und Unterhaltungsdiensten.

Mobile Endgeräte werden mittels verschiedener Mobilfunknetze an das Internet angebunden. Da Funknetze anfälliger für Störungen als Kabelverbindungen sind, müssen diese Störungen kompensiert werden, um einer Applikation eine stabile Verbindung garantieren zu können. Dies ist wichtig bei kritischen Anwendungen, wie in der Medizin oder im Gewerbe, jedoch auch für den alltäglichen Nutzer, um dessen Vertrauen zu stärken und das Endgerät zu einem verlässlichen Begleiter zu machen.

In dieser wird untersucht, welche Qualitätskriterien bei der Datenverbindung für mobile Anwendungen relevant sind. Aus diesen Qualitätskriterien werden Anforderungen an die Anwendung formuliert anhand dieser Beispielanwendungen untersucht. Anschließend werden einige Probleme und Fehler aufgeführt, welche die Anforderungen beeinflussen. Die Probleme und Fehler werden in Kategorien unterteilt und Fehler und ihre Folgefehler beispielhaft in einer Baumstruktur angeordnet. Danach werden recherchierte Lösungen und welche Fehler sie beheben können vorgestellt. Dabei wird die zuvor aufgestellte Baumstruktur genutzt, um die Fehler auf einer hohen Ebene für so viele Anwendungen wie möglich zu lösen. Im Fazit wird die Frage geklärt, wie man mobile Anwendungen gegen instabile Datenverbindungen absichert, in welchem Kontext dies geschehen muss und welche Probleme derzeit noch nicht zu lösen sind.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Smartphones oder genereller mobile Endgeräte im Kontext des *Ubiquitious Computing* (Computer im allgegenwärtigen Gebrauch) erfahren derzeit einen starken Zuwachs. Allein Smartphones mit einem aktiven Breitband-Mobilfunkvertrag besitzen einen globalen Marktanteil von 29,5%, mit einem Wachstum von 40% [mob13]. Zum jetztigen Zeitpunkt (Q3 2013) besitzt fast jeder zweite Einwohner eines Industriestaates ein Smartphone oder ein anderes mobiles Endgerät wie etwa ein Tablet.
Dieses starke Wachstum haben mobile Endgeräte ihrer Vielseitigkeit zu verdanken, die durch eine für mobile Geräte hohe Rechenleistung, eine Vielzahl an Funktechniken und Sensoren erreicht wird. Es gibt für die unterschiedlichen Betriebssysteme dieser mobilen Endgeräte eine Vielzahl von Applikationen, die einfach über Verkaufsportale

auf das Endgerät heruntergeladen werden können. Allein die Verkaufsportale App Store für Apples iOS und Google Play für Google Android bieten jeweils rund eine Million Applikationen an.
Smartphones sind mobile Endgeräte, welche die Definition eines *PDA* (Personal Digital Assistant) erweitern. Das Buzzword *Life Companion* von Samsung aus der Marketing Kampagne des Smartphones Galaxy S4 beschreibt das mobile Endgerät als einen digitalen Begleiter in allen Lebenslagen.
Aufgrund der hohen Verbreitung und Vielseitigkeit dieser Geräte gibt es mittlerweile viele professionelle und auch kritische Applikationen für diese Geräte, z.B. zur Steuerung von Prozessen im Gesundheits- oder Firmenwesen. Aus diesen Gründen benötigt ein solches Endgerät eine robuste Datenversorgung. Ein großer Anteil der benötigten Daten für Applikationen sind Kommunikations- und Informationsdaten. Da viele dieser Informationen weder selbst generiert noch gespeichert werden können, ist hierfür eine Verbindung mit einem Netzwerk erforderlich.
Diese Datenverbindung ist ein Mobilfunknetz oder ein WLAN Hotspot. Mobilfunknetze werden von Mobilfunkanbietern durch flächendeckende Mobilfunkmasten bereitgestellt. WLAN-Hotspots haben weniger Reichweite, können jedoch auch privat, von Firmen oder anderen Institutionen angeboten werden. Durch die Verbreitung von mobilen Endgeräten besteht eine hohe Anforderung an die Bandbreite und Verfügbarkeit der Datenverbindung. In den dichter bewohnten Gebieten der Welt wird eine durchschnittliche Bandbreite von etwa 2,5 Mbit/s und eine Verlässlichkeit von 90% erreicht [ope13c]. Durch den Zuwachs an mobilen Endgeräten wächst diese weiterhin stark an. Gewünscht ist eine Datenverfügbarkeit, die es uns ermöglicht, in jeder Situation mit dem mobilen Endgerät und seinen Applikationen zu interagieren.

1.2 Arten von Anwendungen

Nicht alle Anwendungen benötigen eine konstante Datenverbindung. Viele kommen auch vollständig ohne Daten aus oder benötigen diese nur sporadisch. In Abbildung 1.1 werden Applikationstypen nach der Größe des Zeitfensters, in dem sie ihre Daten benötigen, kategorisiert. Für den Nutzer ist weiterhin eine möglichst geringe Verzögerung der

Datenversorgung ideal, um Wartezeiten bei der Bedienung zu vermeiden.
In die Kategorie *Echtzeit* fallen alle Applikationen, bei denen Daten innerhalb von Millisekunden übertragen werden müssen. Dabei variiert die Verzögerung. Bei Multiplayer-Spielen mit wird z.B. eine geringere Verzögerung benötigt als bei Chats [Jib13]. Zeitnahe Übertragung wird bei Kartenapplikationen, im Instant Messaging oder Social Network Applikationen sowie Internet benötigt. Dabei ist eine kürzere Übertragungszeit der Daten für den Benutzer angenehm, jedoch nicht kritisch. Das Zeitfenster hierbei beträgt 1 bis etwa 20 Sekunden. Zu der Kategorie *Download* zählen alle Applikationen, für die ein Zeitfenster von mehreren Minuten ohne Datenversorgung ausreicht. *Offline* bezeichnet alle Applikationen, die komplett ohne Datenversorgung auskommen, wie z.B. die Wecker-Applikation.

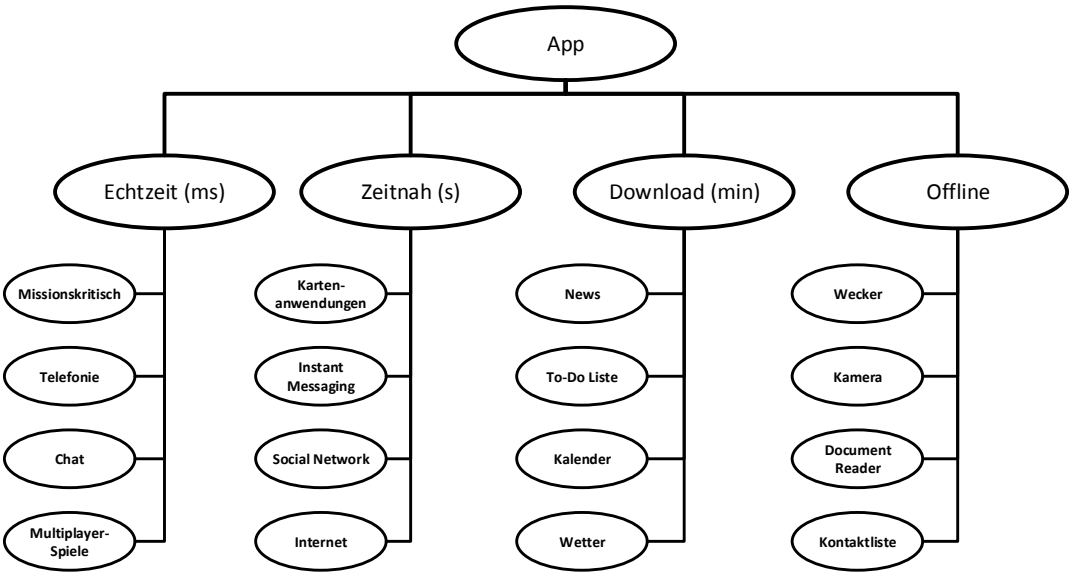

Abbildung 1.1: Einteilung der Applikationen nach Zeitfenster der Datenversorgung

Ein Großteil der betrachteten Applikationen benötigt zumindest eine Datenverbindung, unabhängig davon welche Bandbreite oder Latenz (Verzögerung) diese hat. Daher wird im nachfolgender Kapitel der Weg der Daten im Detail betrachtet.

1.3 Weg der Daten

Daten durchlaufen mehrere logische und physische Schichten. Eine grobe Einteilung ist in Abbildung 1.2 gegeben. Es werden fünf Kategorien betrachtet, die nun beschrieben werden. *Datenquellen*, welche speziell für die Anwendung gehostete Server sein können, Daten von Drittanbietern und spezielle Cloud-Dienste. *Datenquellen* speisen Daten in die *Datenübertragungs*-Schicht ein, welche das Routing durch das Internet beschreibt. Vom Internet aus gehen die Daten in die *Physische Verbindung* über. Die Physische (Funk-)Verbindung ist auf der Seite des Netzzugangs ein WLAN Hotspot oder ein Mobilfunkmast und auf der Seite des Endgeräts die entsprechenden Empfangsgeräte, hier als WLAN und Mobilfunkmodem bezeichnet. Diese Modems werden durch Gerätetreiber an das Betriebssystem angebunden, welches wiederum eine *API* (Programmbibliothek) für die auf dem Betriebssystem laufenden Anwendungen bereitstellt. Nachfolgend wird in Abbildung 1.2 der Weg der Daten durch die Schichten beschrieben.

Beim einfachen Request / Reply Zyklus fordert eine Anwendung Daten über eine Anfrage an die Netzwerk-API des Betriebssystems an. Das Betriebssystem baut daraufhin eine Verbindung über den Treiber des Modems auf. Das Modem verbindet sich mit dem Netzzugangsgerät. Das Netzzugangsgerät stellt wiederum eine Verbindung mit dem Internet her und schickt die Anfrage an die Adresse der Datenquelle. Diese Anfrage wird nun durch das Internet geroutet und an die Datenquelle gesendet. Die Datenquelle antwortet ihrerseits auf denselben Weg entweder mit den angeforderten Daten oder einer Fehlermeldung.
Auf dem Weg von der Datenquelle zur Anwendung können nun in verschiedenen Schichten Probleme auftreten, die Auswirkungen auf die Datenverbindung haben. Diese Probleme werden nun kurz beschrieben.

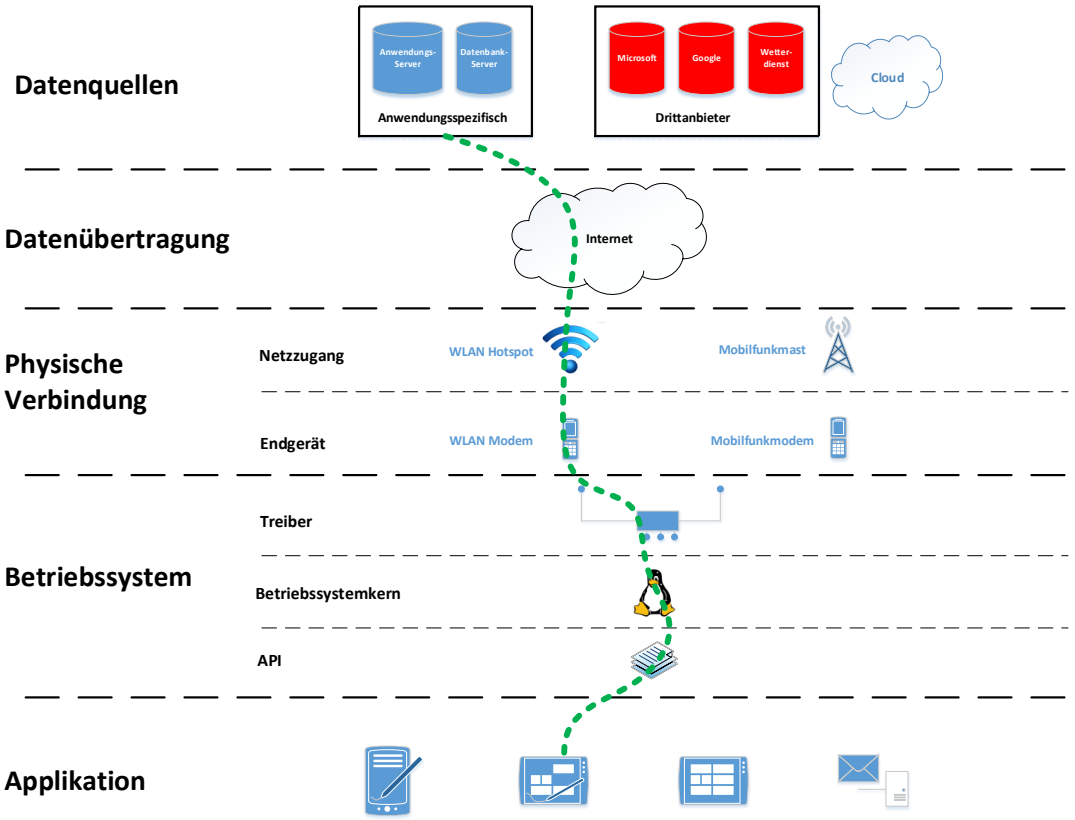

Abbildung 1.2: Weg der Daten zur Applikation durch die verschiedenen Schichten

1.4 Probleme

Die durchschnittliche Verfügbarkeit von Mobilfunknetzen liegt bei 90 % [ope13c]. Auch wenn der Mobilfunknetzausbau weiter voranschreitet, gibt es immer noch Orte, in denen keine oder nur eine unzureichende Datenverbindung besteht. Auch verursachen Smartphones aufgrund der Vielzahl an laufenden Applikationen und deren frequenter Signalübertragung eine große Anzahl an Datenpaketen zu vielen verschiedenen Netzteilnehmern, was die Überlastung der Hardware-Router des Mobilfunknetzes zur Folge hat. Dazu kommen Einschränkungen durch urbane Bebauung, die Funksignale blockieren, z.B. in Tiefgaragen oder Betongebäuden. Ausserdem können Verzögerungen in jeder Schicht entstehen, die Einfluss auf die schnelle Datenübertragung und Latenz haben. Beispiele dafür sind Probleme beim Routing, zu geringe Bandbreite der

Datenverbindung innerhalb und zwischen Schichten (*weakest link*) oder auch unzureichende Berücksichtigung der beteiligten Anwendungen. Um eine Versorgung mit Daten sicherzustellen, muss diese unzureichende Verfügbarkeit kompensiert werden. Dafür gibt es verschiedene Lösungen, die im folgenden Kapitel kurz beschrieben werden.

1.5 Lösungen

Die Probleme instabiler Datenverbindungen können durch primitive Lösungen, wie den erweiterten Netzausbau oder das Caching großer Datenmengen, kompensiert werden. Das liegt an der starken Beeinflussbarkeit von äußeren Störungen von Funkverbindungen. Daher ist die Annahme oder auch das Ziel, mit Ausbaumaßnahmen ein verlässliches ideales Netz zu erreichen irreführend. Um eine konstante Versorgung mit Daten sicherzustellen müssen Verfahren angewandt werden, die der Prävention von und der Reaktion auf Störungen der Datenverbindung dienen.
Lösungen zur Kompensation dieser Unzulänglichkeiten reichen von Caching bis hin zu kooperativen Netzen. Umgesetzt werden diese in verschiedenen Smartphone-Schichten, beispielsweise in der Applikation oder dem Betriebssystem. Auf vielen Betriebssystemen werden bereits Lösungen für diese Probleme angeboten, wobei die Güte dieser Techniken vom verwendeten Betriebssystem abhängt. Ein Beispiel hierfür ist der automatische Wechsel von 3G zu WLAN bei einem Verbindungsabbruch. Für die Applikationsprogrammierung gibt es Richtlinien, nach denen vorgegangen werden kann, um eine robuste Applikation zu entwerfen. Ein Vorstoß in diese Richtung wurde unter dem Begriff *network friendliness* gemacht, jedoch sind dies eher Richtlinien, die nicht zwingend eingehalten werden müssen. Ein Beispiel hierfür sind die *Smarter app guidelines* [GSM13a] von der GSMA Association.
Allerdings gibt es keine vollständig schichtenübergreifende Lösung der zu behandelnden Probleme. Dies hat vor allem mit der heterogenen Struktur der an der Übertragung teilnehmenden Techniken und Anbietern zu tun. Daher sind die meisten Lösungen auf Kooperation mit den anderen Schichten angewiesen. Aus diesem Grund sollten Probleme in der Datenverbindung durch Lösungen behandelt werden, auf die eine Einflussnahme möglich ist. Aus der Sicht des Entwicklers sind sind vor allem die Schich-

ten wichtig, auf die Einfluss genommen werden kann. Das sind die Applikation, die Datenquellen, die für die Applikation verwendete Funktechnik und als Maßnahme der Einflussnahme auf die Betriebssystemschicht eine *Middleware* (Zwischenschicht zur Prozesskommunikation), die Aufgaben des Verbindungsmanagements übernimmt. Aus dem Umstand viele verschiedene Probleme und Lösungen auf verschiedenen Schichten zu haben, ist es schwer diese zu erfassen.

1.6 Ziel der Arbeit

Es ist das Ziel dieser Arbeit einen ersten Ansatz zu bieten, eine Übersicht über die Probleme und Lösungen für eine robuste Datenverbindung zu erhalten. Das dient der Problemanalyse und Lösungsfindung in existierenden Applikationen. Auch kann sie genutzt werden um mögliche auftretende Problemfälle in einer beliebigen Granularität zu konstruieren. Danach können ein oder mehrere Lösungsansätze gewählt werden, um eine Applikation mit der bestmöglichen Verlässlichkeit in Bezug auf die Datenverbindung zu entwerfen. Auf diesem Weg wird eine methodische Grundlage geschaffen, Probleme zu erkennen und Lösungen koordiniert anzuwenden.

1.7 Aufbau der Arbeit

Zunächst werden in Kapitel 2 die Qualitätskriterien einer robusten Datenverbindung definiert. Dann werden die Anforderungen an eine Applikation zur Verbesserung dieser Kriterien aufgestellt. In Kapitel 3 werden einige Anwendungen auf die Erfüllung dieser Anwendungen untersucht.Danach werden in Kapitel 4 einige der Probleme und Fehler aufgelistet, die eine Datenverbindung negativ beeinflussen. Außerdem werden einige der Fehler in eine Baumstruktur überführt. Daraufhin werden in Kapitel 5Lösungen für die beschrieben Fehler aufgelistet und für einige Fehler in der zuvor erstellten Baumstruktur dargestellt, welche Fehler damit gelöst werden können. In Kapitel 6 wird die Idealvorstellung einer robusten Datenverbindung dem derzeitigen Stand gegenübergestellt und

ein Ausblick gegeben, welche Möglichkeiten der Problemlösung mit dieser Methodik möglich sind.

2

Qualitätskriterien und Anforderungen

Für die Datenübertragung gibt es verschiedene Kriterien, nach denen eine robuste Datenversorgung beschrieben und bewertet werden kann. Auf Grundlage dieser Kriterien werden Anforderungen beschrieben, die erfüllt werden müssen, um der Applikation eine robuste Versorgung mit Daten zu garantieren.

2.1 Qualitätskriterien der Datenübertragung für eine robuste Applikation

Qualitätskriterien sind Eigenschaften, mit denen eine Applikation oder eine Datenübertragung qualitativ beschreiben werden können. Diese bieten die Grundlage für eine Erarbeitung von notwendigen Anforderungen an eine Applikation und werden im Folgenden vorgestellt.

2.1.1 Network Friendliness

Aufgrund der zentralen Struktur mobiler Netzwerke ist die Übertragungsgeschwindigkeit limitiert. Der Begriff *Network Friendliness* kommt auf der Mobilfunkbranche und beschreibt die generelle Eigenschaft einer Applikation sparsam mit Mobilfunkressourcen umzugehen. Aufgrund der begrenzten Einflussnahme der Mobilfunkbranche auf die Applikationsentwicklung ist dies jedoch nur als freiwillige, kooperative Richtlinie gedacht. Zur Definition des Begriffs Network Friendliness werden die *Smarter apps guidelines* [GSM13b] der GSMA Association zu Hilfe genommen.

2.1.2 Latenz

Latenz beschreibt die Zeit zwischen dem Auftreten eines Ereignisses und dem Auftreten eines erwarteten Folgeereignisses. Dies ist vor allem bei Echtzeit-Anwendungen wichtig, wie Navigation- oder Verkehrsanwendungen. Beeinflusst wird die Latenz von der aktuellen Auslastung im Gesamtnetzwerk, der Antwortzeit des angefragten Dienstes sowie der aktuellen Auslastung des Smartphones selbst.

2.1.3 Zeitsynchronität

Zeitsynchron bedeutet, dass Validität und Nutzung der Daten von der Zeit beeinflusst sind. Dies kann bedeuten, dass sie nur eine begrenzte Zeit gültig sind und / oder mit einem fortlaufenden Zeitstempel ausgestattet werden, welche dem Benutzer mit den Daten als Information angeboten werden müssen.
Als Beispiel sei hier die Zusammenarbeit zwischen verschiedenen Verkehrsnetzen genannt. Ein konkretes Beispiel ist hier die Applikation der Deutschen Bahn, die sowohl ihr eigenes als auch öffentliche Verkehrsnetzinformationen anbietet und die Daten dieser Verkehrsnetze zur Verbindungsermittlung zeitsynchron sein müssen, um eine Verkehrsverbindung durch alle Verkehrsnetze ermitteln zu können.

2.1.4 Korrektheit

Korrektheit bezeichnet die Eigenschaft, dass die übertragenen Daten denen entsprechen, die ursprünglich von der Datenquelle gesendet wurden. Dieses Kriterium hat vor allem bei Verbindungstechniken ohne Fehlerkorrektur Bedeutung.

2.1.5 Priorität

Dies beschreibt sowohl die *Priorisierung* bestimmter Daten innerhalb einer Applikation als auch die Priorisierung von zeitkritischen Datenverbindungen im mobilen Betriebssystem selbst. In den meisten Fällen ist es sinnvoll, in der Applikation erst die wichtigsten Daten zu übertragen, ohne welche die Anwendung nicht funktionieren kann, z.B. bei einem Messenger zuerst die Nachrichten zu laden und danach die Profilbilder, welche deutlich größer sind als der Datensatz, der für die grundlegende Funktion der Applikation notwendig ist.

2.2 Notwendige Anforderungen für eine robuste Applikation

Anforderungen für eine robuste Applikation zu definieren ist das Ziel vieler Arbeiten im Themenbereich mobiler Applikationen. Ein Beispiel hierfür bietet z.B. der Artikel *Best practices for mobile development* [Mot12]. Außerdem wurden von der GSMA Association die *Smarter Apps Guidelines* [GSM13b] veröffentlicht, um Entwicklern Richtlinien bereitzustellen, wie sich eine mobile Anwendung verhalten soll. Hierbei gibt es Anforderungen, die zur *Network Friendliness*, den applikationsorientierten Aspekten zur besseren Nutzung der Mobilfunkverbindung zählen sowie welche, die zu einer robusten Verbindung beitragen. Teilweise gibt es Überschneidungen bei Aspekten, die beide oder Teile beider Definitionen erfüllen. Daher können Eigenschaften sowohl zu Network Friendliness oder Connection Robustness zugeordnet werden als auch zu beiden Begriffen. Im Folgenden werden Begriffe und Techniken beschrieben, die von einer Applikation oder ihren unterstützenden Systemen, wie z.B. das Betriebssystem, implementiert werden müssen, um die Qualität der beschriebenen Kriterien zu verbessern.

2.2.1 Vertical handover

Ein *Vertical Handover* bezeichnet den automatischen Wechsel der Funktechnik auf dem Gerät, meistens zwischen 3G und WLAN. Das Betriebssystem des Endgeräts wechselt hierbei den Treiber sowie das für die Verbindung zuständige aktive Modem und leitet den Datenverkehr auf dieses Modem um.
Vertical Handover wird auch als *WLAN Offloading* bezeichnet, falls der Wechsel auf WLAN erfolgt. WLAN Offloading unterscheidet sich von einem Vertical Handover dadurch, dass ein Wechsel auch dann ausgeführt wird, wenn 3G noch zur Verfügung steht. Die Gründe für den Wechsel sind hierbei die möglichen Kosten für die 3G-Verbindung und eine höhere Bandbreite bei WLAN.

2.2.2 Horizontal Handover

Ein *Horizontal Handover* ist der Wechsel zwischen verschiedenen Netzzugangspunkten innerhalb einer Funktechni, welches entweder WLAN Hotspots oder 3G Funkmasten sein können. Dies ist ebenfalls unter dem Begriff *Roaming* bekannt.

2.2.3 Offline-Funktionalität

Eine Applikation muss ein *Offline-Szenario* erkennen können. Ein Offline-Szenario ist der Fall, dass keine Datenverbindung existiert und die Applikation trotzdem mit den vorhandenen Daten arbeiten kann. Ein Beispiel sind auch *Offline outgoing queues* auch, die es möglich machen, eine Textnachricht zu verfassen, zu speichern und zu senden, wenn wieder eine Datenverbindung besteht.

2.2.4 Asynchronität

Asynchronität bedeutet, dass Objekte nicht strikt nacheinander, sondern *out of order* (Ausführung in anderer Reihenfolge) übertragen werden. Dies dient in diesem Kontext der besseren Bandbreitenauslastung und der Reduzierung von Overhead. Die Idee dabei

ist, in einem kurzen Zeitfenster so viele Daten wie möglich zu übertragen. Dies muss vom Betriebssystem intelligent gepuffert und bei einer gewissen Datenmenge oder einem Zeitfenster ausgelöst werden. Es hat auch Vorteile bezüglich des Stromverbrauchs, da das Aufwecken des 3G-Modems oder WLAN-Funkmoduls aus dem Ruhemodus Energie benötigt. Dadurch wird der hohe Energieverbrauch auf einen möglichst kurzen Zeitraum reduziert.

2.2.5 Priorisierung

Die Priorisierung oder dynamische Anpassung der Bandbreite der Applikation findet statt, wenn bestimmte Daten wichtiger als andere klassifiziert werden. Als Beispiel sei hier ein Newsfeed genannt, der zuerst allen Text und danach erst die Bilder lädt.

2.2.6 Caching und Datenspeicherung

Caching bedeutet in diesem Fall die Zwischenspeicherung von Daten im lokalen Speicher des Endgeräts, der erneute Zugriffe auf ein langsameres Medium, hier die Datenverbindung, vermeidet. Dies kann dazu dienen, Inhalte nicht neu laden zu müssen, sollte die Applikation neu gestartet werden. Auch kann ein Cache dazu benutzt werden Daten zu speichern, die in Beziehung mit dem aktuellen Datensatz stehen, um später nicht von der Datenverbindung abhängig zu sein. Die Bezeichnung dafür ist *Predictive Caching*. Hierbei gibt es verschiedene Caching-Strategien sowie Techniken zur Speicherung und zur Organisation des Caches. Caching kann sich mit der Behandlung von Offline-Szenarien (sieheKapitel 2.2.3) überschneiden, da teilweise der Cache dazu genutzt wird, Offline-Funktionalität herzustellen.
Datenspeicherung bezeichnet das Speichern von Daten in einer festen Form außerhalb des Caches. Das ist wichtig, da nach der Speicherhierarchie der Cache ein flüchtigen Speicher ist und von Mechanismen innerhalb des Betriebssystems gelöscht werden kann.

2.2.7 Kompression

Kompression ist wichtig, um in Anbetracht der begrenzten Übertragungsressourcen das Volumen der Daten zu reduzieren. Je nach Anwendungsfall muss abgewogen werden, ob eine Kompression Sinn macht, da auf einem Smartphone CPU-Ressourcen und in diesem Zusammenhang vor allem Energie begrenzt ist.

2.2.8 Reconnection Support

Bei einer Unterbrechung der Verbindung - dies kann z.B. beim Wechsel der genutzten Funktechnik von WLAN nach 3G sein oder einer kurzen Verbindungsunterbrechung - sollte die Verbindung ohne Einwirken des Nutzers wiederaufgenommen und die bereits gesendeten Authentifizierungsdaten sowie Nutzdaten weiterverwendet werden können. Dies wird auch als *Session Recovery* bezeichnet.

2.2.9 Interpolation

Interpolation beschreibt die Errechnung von neuen Daten anhand der bereits übertragenen Daten. Ein Beispiel hierfür ist die Schätzung der aktuellen GPS-Position anhand des GPS-Koordinaten-Verlaufs.

2.2.10 Push-Techniken

Push-Techniken beziehen sich auf das Senden von Daten vom Server an den Client. Eine andere Technik ist Polling, bei welcher erst eine Anfrage vom Client an den Server gesendet werden muss, bevor dieser Daten schicken kann. Push-Techniken verringern daher die Netzlast, da keine Anfrage vom Client an den Server geschickt werden muss, damit eine Datenübertragung möglich ist.

2.2.11 Verbindungsstatus

Ein wichtiger Aspekt ist die Benachrichtigung des Benutzers über den Verbindungsstatus. Dieser muss sich auf die Angaben der Applikation verlassen können, insbesondere auf die Genauigkeit der Angabe. Außerdem sind anwendungsspezifische Einstellungen wichtig, z.B. selbst gesetzte Timeouts und Funkpräferenzen. Diese können auch systemweit eingestellt werden.

2.3 Zwischenfazit

Eine Applikation, welche sich auf einem modernen mobilen Endgerät befindet, muss mehr bieten als die Applikationen der ersten Stunde, die sehr naiv implementiert und noch sehr sparsam mit dem Datenverbrauch waren. Eine Applikation sollte verlässlich und für den Benutzer nachvollziehbar mit der Funkverbindung und den darüber übertragenen Daten umgehen können. Außerdem sollte sie sparsam mit dem Datenvolumen umgehen, welches dem Nutzer zur Verfügung steht. Wenn in der jeweiligen Anwendung möglich, sollte sie außerdem Rücksicht auf die aktuelle Mobilfunknetz-Situation nehmen und zumindest Möglichkeiten zum WLAN-Offloading nutzen. Im nächsten Kapitel werden einige Anwendungen auf die beschriebenen Anforderungen untersucht.

3

Anwendungsfälle

In diesem Kapitel werden eingie Applikationen aus verschiedenen Anwendungsszenarien beschrieben. Dabei wird eine Übersicht über deren grundlegende Funktionen gegeben und danach die Architekturen der Anwendungen betrachtet. Dann wird anhand der in Kapitel 2 beschriebenen Anforderungen untersucht, ob die Anwendungen diese erfüllen.
Dabei werden Applikationen aus dem Bereich des Instituts für Datenbanken und Informationssysteme vorgestellt. Außerdem werden weitere Anwendungen vorgestellt, um mehr Anwendungsbereiche und mögliche Lösungen abzudecken. Dies schließt ein welche Aspekte sie erfüllen und welche Techniken bzw. Lösungen dafür angewandt werden.

3.1 SmartTravel

SmartTravel ist eine Applikation für Android sowie iOS, die Echtzeit-Nahverkehrsdaten des aktuellen Standorts bereitstellt. Dies sind Informationen über Haltestellen in der Nähe sowie die Abfahrtszeiten der entsprechenden Busse. Weiter bietet die Anwendung Informationen über nahegelegene car2gos (ein CarSharing Service) auf einer Karte. Die Daten müssen daher auch Lokalitätsinformationen bereitstellen.
Die Informationen müssen außerdem schnell zur Verfügung stehen, da sich stetig ändern und anhand dieser schnell eine Entscheidung getroffen werden muss. Ein Beispiel dafür ist ein Anwender, der in einem Bus sitzt und Informationen über die Abfahrtszeiten der nächsten Haltestellen benötigt, um zu entscheiden, welche Buslinie er als nächstes nimmt. Es gibt jedoch einige Probleme, die noch ungelöst sind. Das betrifft ist vor allem die Datenverfügbarkeit, da die Datenquelle der SWU-Busdaten fehleranfällig ist. Das liegt daran, dass die Daten von einer Webseite abgerufen werden und mittels eines HTML-Parsers extrahiert werden müssen.

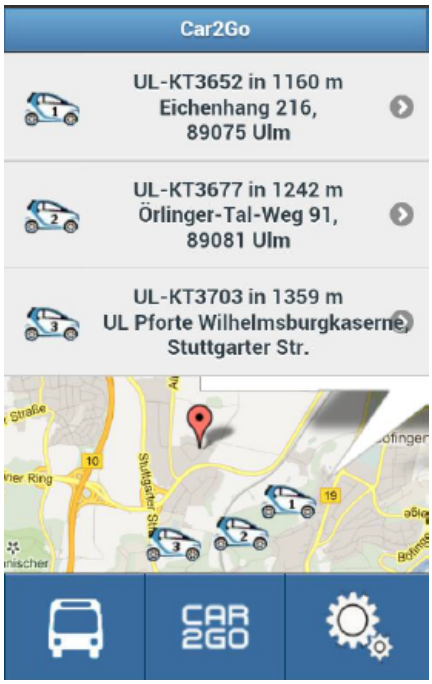

Abbildung 3.1: SmartTravel App mit car2go Ansicht

3.1.1 Architektur

SmartTravel wird mit PhoneGap, einem Cross-Platform Development Framework umgesetzt. Dieses erlaubt die einfache Portierung der Applikation auf mehrere Betriebssysteme für mobile Endgeräte. An Datenquellen verwendet PhoneGap die auf dem Endgerät ausgewählte Map-Applikation, die car2go API sowie einen Parser für SWU-Busfahrplandaten. Hierbei werden die SWU-Busfahrplandaten alle 20 Sekunden mittels Polling aktualisiert.
SmartTravel lässt sich nur starten, wenn eine Internetverbindung verfügbar ist. Bei fehlender Funkverbindung während der Laufzeit der Applikation wird eine Fehlermeldung angezeigt und die Applikation beendet. Die Applikation ist abhängig von einer festen, verlässlichen Datenverbindung. Ungelöste Probleme sind außerdem die Datenverfügbarkeit der verwendeten SWU-Fahrplandaten sowie dem fehlenden Caching und Zeitsynchronisierung derselben. In Tabelle 3.1 wird dargestellt, ob die in Kapitel 2 vorgestellten Anforderungen erfüllt werden.

Anforderung	Unterstützung
Vertical Handover	
Horizontal Handover	(•)
Offline-Funktionalität	
Asynchronität	
Priorisierung	
Caching	(•)
Kompression	
Reconnection Support	
Interpolation	
Push-Techniken	
Verbindungsstatus	•

Tabelle 3.1: Erfüllung der Anforderungen bei SmartTravel

3.2 DBIS Scholar

DBIS Scholar ist eine Zitationsanwendung für iOS zur Berechnung von Zitationsindizes. Die benötigten Daten erhält die Anwendung von Google Scholar. Diese werden über

die Websuche angefragt und per Parser in ein für die Anwendung nutzbares Format umgewandelt. Es wäre vorteilhaft, das Parsing auf einen Server auszulagern, um Datenvolumen sowie Batterielaufzeit zu sparen. Diese Funktionalität müsste über eine selbst programmierte Serverkomponenten geschaffen werden. Google Scholar blockiert jedoch eine IP bei zu vielen Anfragen. Die Applikation umgeht die Blockierung mit Proxies und dem Caching der Anfragen. Beim Caching werden Anfrageergebnisse im Cache abgelegt und bei erneuter Anfrage ausgegeben, wenn sie nicht älter als 12 Stunden sind. Damit wird die Anzahl der Anfragen reduziert und die Blockierung hinausgezögert.

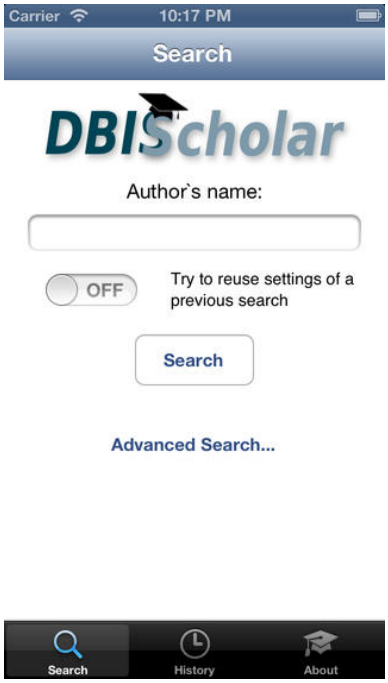

Abbildung 3.2: DBIS Scholar

3.2.1 Architektur

Die Applikation wurde mit XCode, dem SDK von Apple für iOS, implementiert. Sie parst Daten von Google Scholar und ist damit abhängig von der Verfügbarkeit dieses Dienstes, welcher jedoch eine hohe Robustheit aufweist. Abfragen werden von der Applikation in den Cache gespeichert. Dadurch ist auch das Offline-Arbeiten mit diesen Daten möglich. Da als Programmierplattform die iOS-SDK XCode verwendet wurde, musste

sich der Entwickler hier selbst um die Offline-Verfügbarkeit kümmern. In Tabelle 3.2 wird dargestellt, ob die in Kapitel 2 vorgestellten Anforderungen erfüllt werden.

Anforderung	Unterstützung
Vertical Handover	•
Horizontal Handover	(•)
Offline-Funktionalität	•
Asynchronität	
Priorisierung	
Caching	•
Kompression	
Reconnection Support	•
Interpolation	
Push-Techniken	
Verbindungsstatus	•

Tabelle 3.2: Erfüllung der Anforderungen bei DBIS Scholar

3.3 Öffi

Öffi ist eine ähnliche Applikation wie SmartTravel, jedoch konzentriert sie sich auf den öffentlichen Nahverkehr und unterstützt viele Städte in Europa. Sie ist derzeit nur für Android verfügbar. Sie kann im Gegensatz zu SmartTravel auch Wege von A nach B anzeigen und hat einige Komfort-Funktionen, wie z.B. ein Widget.

3.3.1 Architektur

Öffi wurde mit Hilfe der Android SDK entwickelt. Sie bietet sowohl Caching von angeforderten Daten als auch Zeitsynchronität an, da bei einem Offline-Szenario sowohl die Live-Abfahrtszeiten als auch die Fahrplandaten der in der Nähe befindlichen Haltestellen angezeigt werden. Außerdem wird die Live-Anzeige aktualisiert, sodass die Daten trotz fehlender Verbindung zeitsynchron sind. Des Weiteren wird über eine Färbung der Abfahrtszeiten angezeigt, dass diese nicht mehr aktuell sind. Über das Caching der verwendeten Kartenanwendung funktioniert auch die Standortanzeige trotz fehlender

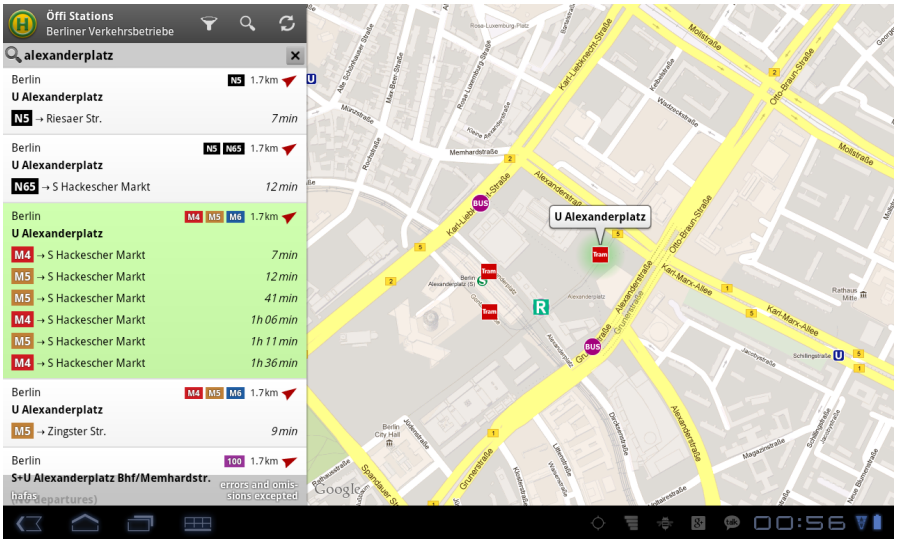

Abbildung 3.3: Öffi mit Kartenansicht

Datenverbindung. Öffi bietet auch *Reconnection Support*. Selbstverständlich kann ohne Datenverbindung keine Live-Funktionalität angeboten werden. Jedoch wäre es möglich über den Download der aktuellen Fahrplaninformationen eine Offline-Funktionalität anzubieten. In Tabelle 3.3 wird dargestellt, ob die in Kapitel 2 vorgestellten Anforderungen erfüllt werden.

Anforderung	Unterstützung
Vertical Handover	•
Horizontal Handover	•
Offline-Funktionalität	•
Asynchronität	
Priorisierung	
Caching	•
Kompression	
Reconnection Support	•
Interpolation	
Push-Techniken	
Verbindungsstatus	•

Tabelle 3.3: Erfüllung der Anforderungen bei Öffi

3.4 Umfragesystem DBIS

Mit Hilfe des Umfragesystems sollen bisher papier-basierte Studien mit einem computergestützten Fragebogensystem ergänzt werden. Dieses soll in Ländern der dritten Welt eingesetzt werden. Wegen des schlechten Mobilfunknetzausbaus in diesen Ländern benötigt dieses System die Eigenschaft auch komplett offline verfügbar zu sein.

3.4.1 Architektur

Das System besteht aus drei Teilen, einen Konfigurator, einer Middleware und einem Client. Der Konfigurator ist eine Weboberfläche zur Erstellung von Fragebögen. Diese Fragebögen werden über die Middleware in der Datenbank abgelegt. Der Client lädt diese Fragebögen auf ein mobiles Endgerät solange noch eine Netzverbindung besteht. Danach können diese ausgefüllt und lokal abgelegt werden. Diese Daten können bei einer bestehenden Netzverbindung in die Datenbank hochgeladen werden. Das Umfragesystem wurde mit dem Fokus auf Offline-Funkionalität entworfen. Die Verwendung der Datenverbindung ist daher ein einfaches Herunter- und Hochladen. Aus diesem Grund benötigt die Anwendung weder Reconnection-Unterstützung noch Caching. In Tabelle 3.4 wird dargestellt, ob die in Kapitel 2 vorgestellten Anforderungen erfüllt werden.

Anforderung	**Unterstützung**
Vertical Handover	•
Horizontal Handover	(•)
Offline-Funktionalität	•
Asynchronität	
Priorisierung	
Caching	•
Kompression	
Reconnection Support	•
Interpolation	
Push-Techniken	
Verbindungsstatus	•

Tabelle 3.4: Erfüllung der Anforderungen beim Umfragesystem

3.5 PowerSource

PowerSource ist ein Kundeninformations- und Verkaufssystem für Verkäufer der Eaton Corporation, entworfen um den Verkaufsprozess trotz großer Datenmengen und schlechter Datenverbindung zu bewerkstelligen. Die Eaton Corporation ist ein US-amerikanisches Industrieunternehmen. Es bieten seinen Kunden weitreichende Elektronik, Hydraulik, Energieverteilung, Komponenten für industrielle Anlagen sowie den Fahr- und Flugzeugbau an. Aufgrund dieses großen Portfolios mussten die Verkäufer des Unternehmen zuvor große Mengen an ausgedruckten Handbüchern für Produkte mit sich führen. Außerdem musste eine große Menge an (papier-basierten) Notizen in die Server von Eaton eingegeben werden, um den Verkaufsprozess einzuleiten. Dieser komplizierte Vorgang machte den Verkaufsprozess sehr langsam und musste daher verkürzt werden. Dies wurde mit der Anwendung PowerSource bewerkstelligt. PowerSource nutzt die MEAP AMPChroma von Antenna, welche eine Plattform bietet, die das komplette Datenmanagement übernehmen kann [Sof13].

3.5.1 Architektur

AMPChroma verwendet eine modulare Architektur und besteht im Falle von PowerSource aus einem Client und einem Framework, welches als mobile Platform-as-a-Service realisiert ist. Zur Datenanbindung gibt es eine *Services Engine*, die dem Client ein Daten-Interface bietet. Über diese Services Engine können Server und Dienste beliebig hinzugefügt werden und über eine Administrationsschnittstelle verwaltet werden. Die Feature-Unterstützung schließt viele Qualitätsaspekte mit ein, z.B. einen verlässlichen Messaging-Dienst mit Queueing oder eine Synchronisierungs-Architektur. Probleme liegen hier hauptsächlich bei der verwendeten Client-Applikation. Wenn diese eine sehr hohe Abstraktion im Sinne einer Cross-Platform Applikation aufweist hat sie wenig Kontrolle über die Verbindung des mobilen Endgeräts. Damit ist sie bezüglich einer verlässliche Verbindung mehr vom Betriebssystem des Geräts abhängig. In Tabelle 3.5 wird dargestellt, ob die in Kapitel 2 vorgestellten Anforderungen erfüllt werden.

Abbildung 3.4: Eaton PowerSource mit 3D Model Ansicht

Anforderung	Unterstützung
Vertical Handover	•
Horizontal Handover	•
Offline-Funktionalität	•
Asynchronität	
Priorisierung	•
Caching	•
Kompression	
Reconnection Support	•
Interpolation	
Push-Techniken	•
Verbindungsstatus	•

Tabelle 3.5: Erfüllung der Anforderungen bei Eaton PowerSource

3.6 Zwischenfazit

Die Tabelle 3.6 fasst die Fähigkeiten der untersuchten Applikationen zusammen. Auch wenn einige der aufgelisteten Applikationen Caching und Reconnection-Unterstützung bieten, sind diese Funktionen weniger als konkretes Ziel des reibungsfreien Ablaufs implementiert worden, sondern eher als sekundäre Effekte durch das Design und der beteiligten Frameworks der Applikation. Wenige Applikationen nehmen auf kurzzeitige Verbindungsunterbrechungen, wie sie im urbanen Raum auftreten, Rücksicht. Die Reaktion auf eine Verbindungsunterbrechung ist meist eine singuläre Fehlermeldung, auf die mit einem Refresh- oder Beenden-Button vom Nutzer manuell reagiert werden kann. Auch wenn es möglich wäre und unter anderem von Mobilfunkanbietern gewünscht ist [GSM13b], gibt es geringe Unterstützung für Offline-Szenarien, es sei denn diese sind explizit per Applikations-Design gewünscht. In Tabelle 3.6 wird der Vergleich der zuvor vorgestellten Applikationen bezüglich der Erfüllung der in Kapitel 2 vorgestellten Anforderungen dargestellt.

Architektur	Simple App (Caching)			Middleware	MEAP
Applikation	SmartTravel	Scholar	Öffi	Umfrage	PowerSource
Vertical Handover		•	(•)	•	•
Horizontal Handover	(•)	•	•	•	•
Offline-Funktionalität		(•)	•	•	•
Asynchronität					
Priorisierung					•
Caching	(•)	•	•		•
Kompression					
Reconnection Support		•			•
Interpolation					
Push Techniken					•
Verbindungsstatus	•	•	•	•	•

Tabelle 3.6: Zusammenfassung der Erfüllung der Anforderungen

4

Probleme und Fehler

Bei vielen Anwendungen können Probleme auftreten, die den Datenfluss von der Datenquelle zum mobilen Endgerät unterbrechen. Hier werden diese Probleme in Kategorien unterteilt sowie aus den Problemen entstehende weitere Fehler aufgeführt. Dies dient dazu eine klare Strukturierung vom auftretenden Problem zum auftretenden Fehler zu haben. Um dies besser beschreiben zu können werden nochmals die Schichten des Weges von der mobilen Anwendung zum Anwendungsserver oder dem verwendeten Dienst dargestellt. Danach werden die Schichten weiter erklärt und differenziert. Schließlich werden Probleme und Fehler in einen geordneten Baum überführt, um die Abfolge von Problem zu Fehlern beschreiben zu können. Hierbei wird die ursprüngliche Ordnung der Schichten beibehalten. Außerdem werden Kombinationsfehler beschreiben, die nur in Kombination mit anderen Fehlern auftreten. Diese können Zyklen im Baum sein, z.B. bei *flattern*, dem ständigen Vertical Handover bei zwei schlechten Verbindungen.

4.1 Schichtenmodell

Die Verbindung einer Applikation auf einem mobilen Endgerät zu den verwendeten Daten durchläuft viele Schichten nach welchen die Fehler klassifiziert werden. Diese Schichten sind die Hardware des Endgeräts, das Betriebssystem inklusive der darin enthaltenen Treiber, optional eine Middleware und die Anwendung selbst. Illustriert werden diese in Abbildung 1.2. Diese Schichten können noch weiter unterteilt werden, um Probleme spezieller adressieren zu können. Jedoch genügen diese für eine erste Klassifizierung der Probleme.

4.2 Methodik und Fehlerbaum

Probleme und Fehler werden aus der Sicht des mobilen Endgeräts und des Benutzers klassifiziert. Fehler werden hierbei in Kategorien anhand des Schichtenmodells aus Abbildung 1.2 unterteilt. Auch ist es möglich dass zwei Fehler in Wechselwirkung zueinander auftreten, dies wird Kombinationsfehler genannt. Folgefehler können auch innerhalb einer einzigen Schicht auftreten. Probleme werden auch als Fehler angesehen, jedoch sind Probleme immer der Ursprung eines auftretenden Fehlers.
Um Folgefehler einordnen zu können werden diese in eine Baumstruktur überführt. Begonnen wird mit dem Problem, das die Ursache aller aus ihr folgenden Fehler ist. Aus Fehlern können wiederum Folgefehler entstehen, die im Baum als Blätter an den Fehler gehängt werden. Hierbei wird das Schichtenmodell aus von oben nach unten abgearbeitet. Ein Beispielhafter Baum wird in Kapitel 4.2 dargestellt. Die Farbe Schwarz bedeutet hier Fehlverhalten, die Farbe blau korrektes Verhalten. Im Folgenden werden die Fehler kategorisiert nach Schicht aufgelistet.

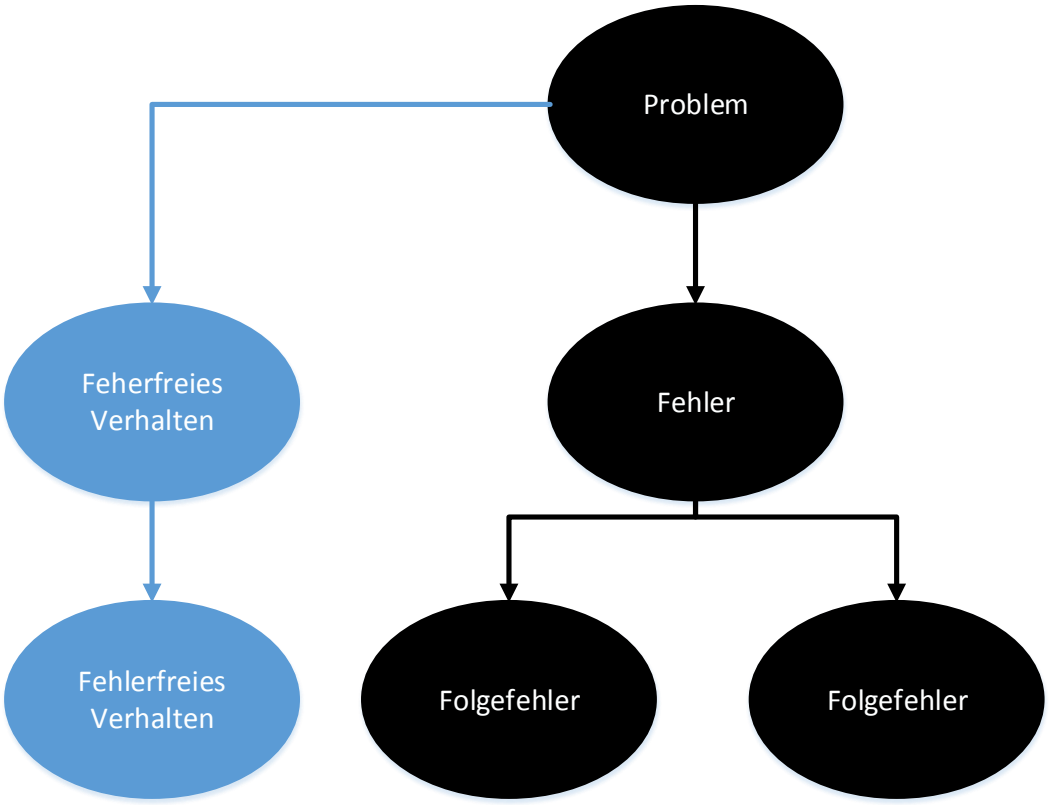

Abbildung 4.1: Darstellung der Fehlerarchitektur

4.2.1 Datenquellen

Mit *Datenquellen* sind alle Quellen gemeint, die Daten über das Internet auf für mobile Endgeräte anbieten. Im Schichtenmodell sind Datenquellen hierarchisch oben angeordnet, auch wenn diese durch die Übertragung durch mehrere Hardware- und Softwareebenen laufen. Dies schließt P2P-Clients und Drittquellen ein. Drittquellen können z.B. DNS, NTP oder andere öffentliche Server sein. Je nach Implementierung können dies einfache Download-Server sein, jedoch auch Techniken zur Vermeidung von Unterbrechungen, zum Beispiel redundante Server in der Nähe des Standorts des mobilen Endgeräts oder die dynamische Anpassung der Transferprotokolle.

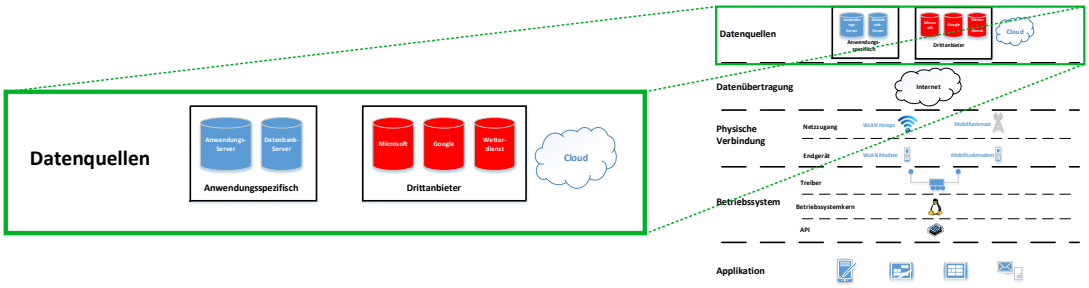

Abbildung 4.2: Datenquellen

4.2.1.1 Drittserver-Limitierungen

Bei der Verwendung von Servern von Drittanbietern, wie beispielsweise Google Scholar oder Wetterdiensten, ist zu beachten, dass die Kontrolle über das Verhalten dieser Server trotz einer angebotenen API nicht immer beim Nutzer liegt. Zum Beispiel limitiert Google Scholar die Anzahl der Anfragen pro IP [Rob11]. Außerdem können Probleme bei den übertragenen Daten auftreten, z.B. wenn sich das Datenschema ändert.

In Abbildung 4.3 wird ein sehr einfacher Fehlerfall beschrieben. Das grundlegende Problem ist, dass die Anwendung vom Drittserver in der Wurzel des Baums blockiert wird. In einem Fall informiert er die Anwendung nicht von der Sperre. Diese fordert die Daten neu an, was zur Folge hat dass die Sperrzeit auf dem Drittserver neu gestartet wird. Dies führt zu einem Fehlerfall oder Livelock, den die Anwendung ohne Eingreifen des Benutzers nicht beheben kann. In dem Baum wird dies als Kreis im Graphen dargestellt. Ein weiterer Fall wäre, dass die Datenquelle der Anwendung die Sperre meldet, die Anwendung dies jedoch ignoriert. Der korrekte Ablauf ist in der Abbildung blau gekennzeichnet, die Datenquelle meldet die Sperre, die Anwendung nimmt diese zur Kenntnis, informiert den Benutzer und wartet bis die Sperre abgelaufen ist. Erst danach startet sie den Neuversuch, welcher zum Erfolg führt.

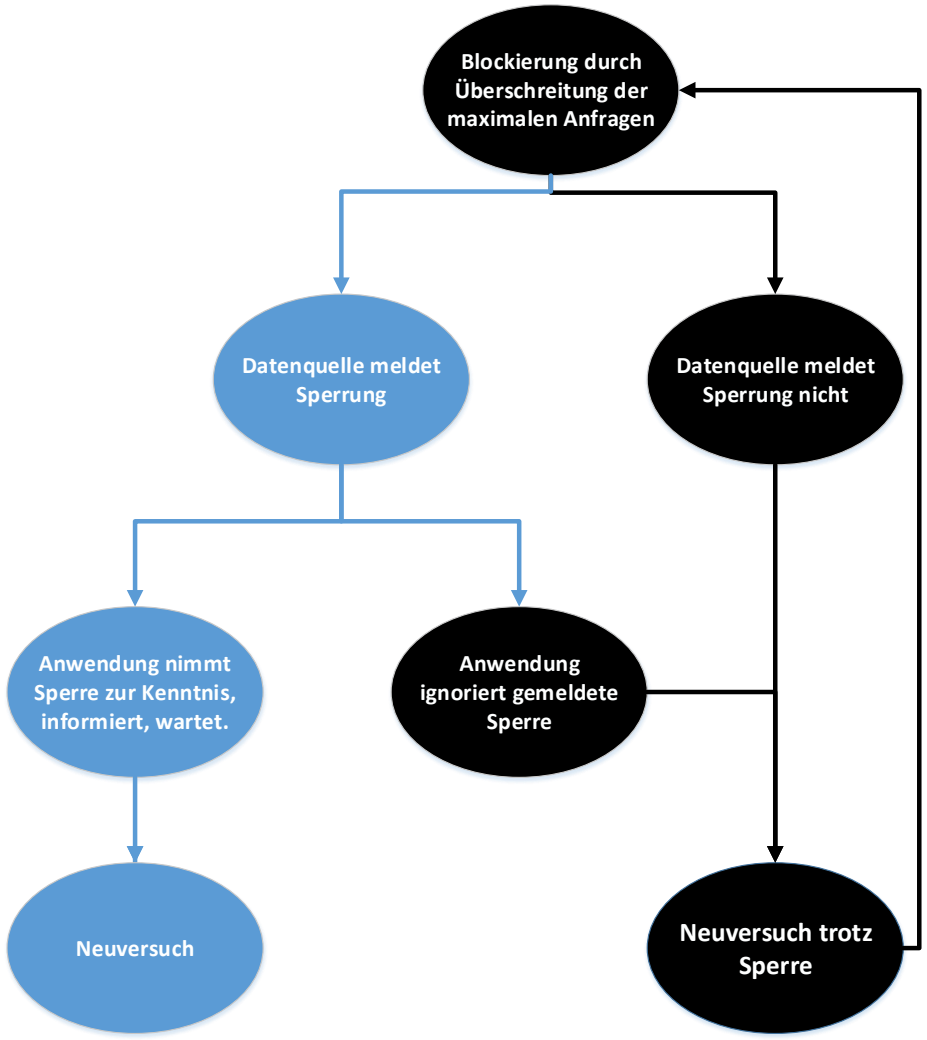

Abbildung 4.3: Darstellung des Fehlerverlaufs bei Drittserver-Limitierungen

4.2.1.2 Naive Implementierung

Eine naive Implementierung bedeutet, dass der Application Server keine oder unzureichende Techniken zur Reduzierung von Verbindungsabbrüchen beherrscht. Diese wären z.B. das dynamische Anpassen der Bandbreite, Reconnection-Unterstützung oder das Wiederaufnehmen von unterbrochenen Downloads.

4.2.2 Datenübertragung

Die Datenübertragungs- oder Internetschicht bezeichnet den Weg der Daten durch das betreffende Netzwerk. Damit sind beispielsweise auch Intranet-Netzwerke eingeschlossen, wenn die Kommunikation der beteiligen Datenquellen und Endgeräte diese nicht verlässt.

4.2.2.1 Blockierung oder Benachteiligung von Diensten in Subnetzen

Dies beschreibt das Blockieren oder die Benachteiligung von bestimmten Diensten durch die am Routing beteiligten Internetknoten. Dies wird auch als Verletzung der Netzneutralität bezeichnet. Meist werden hierbei Anwendungen mit hohem Bandbreitenverbrauch geblockt, wie z.B. Skype.

4.2.2.2 Routing

Die Daten müssen über mehrere Teile der Internetinfrastruktur übertragen werden. Dies schließt Router in verschiedenen Subnetzen mit ein, woraus Probleme entstehen können, z.B. durch Caching hervorgerufene alte Daten. Außerdem kann sich durch unvorteilhaftes Routing der Übertragungsweg verlängern und damit die Latenz erhöhen.

4.2.3 Physische Verbindung des Endgeräts

Auf dem Endgerät befinden sich üblicherweise zwei Funkmodule, eines für die mobile Datenverbindung zum Mobilfunknetz sowie zum WLAN. Daher werden die Fehlerfälle der entsprechenden Technik seperat behandelt. Unterschiede bei der physischen Verbindung unterteilen sich auf zwei Bereiche, den Zugang zum Netz per Funkmasten/WLAN-Hotspot zum einen und zum anderen die Funktechnik im mobilen Endgerät. Die Funktechnik im mobilen Endgerät wird wiederum in zwei Teilbereiche unterteilt. Diese sind generell die unterstützten Mobilfunktechniken im (2G, 3G, 3.5G, 4G) und WLAN (a,b,g,n) sowie die Güte des Empfangs. Die unterstützten Mobilfunktechniken unterscheiden

sich in Signalstärke und Frequenz sowie Fehlererholung und damit besserer Nutzung der zur Verfügung stehenden Empfangsqualität. Die Empfangsqualität hängt von den allgemeinen Antenneneigenschaften sowie der verwendeten Funkstärke ab.
Auf der Netzzugangsseite stehen die Mobilfunkmasten, welche sich in den unterstützten Mobilfunktechniken und Sendestärke unterscheiden. WLAN-Hotspots unterscheiden sich in Verschlüsselung und Authentifizierung sowie in der Tatsache, ob sie privat oder öffentlich sind.

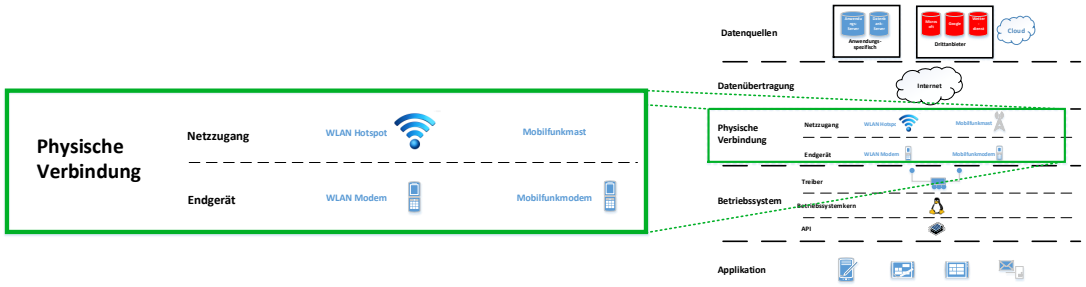

Abbildung 4.4: Physische Verbindung des Endgeräts

4.2.3.1 Kein 3G Netz verfügbar

In diesem Kapitel werden Probleme beschrieben, die dazu führen dass das Gerät keine Verbindung zum Internet über das 3G Modem herstellen kann. Dies ist ein grundsätzliches Problem, das verschiedene Ursachen haben kann, wie z.B. die Umgebungseinflüsse, unzureichende Netzabdeckung sowie die Kapazität des Datennetzes. Diese drei Fälle werden folgend beschrieben.

4.2.3.1.1 Unterbrechung durch Einfluss der Umgebung

Die hier beschriebene Unterbrechung wird durch die Blockierung der Funkverbindung zum Netzzugangspunkt (3G Funkmast) durch Gebäude verursacht. Dieses Szenario tritt oft in urbanem Gelände auf, vor allem aufgrund mehrstöckiger Gebäude aus Stahlbeton. Ein solcher Fehler tritt meist entweder kurzfristig bei der Bewegung an diesen vorbei auf oder permanent, wenn man sich in ihnen befindet. Eine mögliche Lösung innerhalb des Gebäudes sind Verstärker für Mobilfunknetze oder WLAN-Hotspots.

4.2.3.1.2 Keine Netzabdeckung

Keine Netzabdeckung durch den jeweiligen Netzbetreiber. Dies bedeutet, dass sich das mobile Endgerät sich außerhalb der Reichweite eines Netzzugangs (3G Funkmast) befindet. Hier wird unterschieden, ob eine Netzabdeckung gar nicht besteht oder ob sie besteht, jedoch nicht in einer Bandbreite die die benötigten Daten in einer akzeptablen Zeitspanne übertragen werden können. Der Die Ursache hierfür ist ein unzureichender Ausbau von Mobilfunkmasten.

4.2.3.1.3 Netzüberlastung

Eine *Netzüberlastung* findet statt, wenn zu viele Signale von zu vielen Netzteilnehmern die Router der Mobilfunkmasten überfordern. Dies wird *Intermittent internet interruption* genannt, was einen Ausfall oder einen extrem hohen Paketverlust beschreibt, das Internetmodem beim Endgerät jedoch die Verbindung als gut und stabil angibt. Dieses Problem ist mit fehlender Netzabdeckung zu vergleichen, jedoch mit dem wichtigen Unterschied, dass es vom verwendeten Betriebssystem nicht erkannt wird.

4.2.3.2 Kein WLAN-Netz verfügbar

In diesem Kapitel werden Probleme beschrieben, die dazu führen dass das Gerät keine Verbindung zum Internet über das WLAN Modem herstellen kann. Dies ist vor allem wichtig, da heute verstärkt WLAN-Offloading eingesetzt wird, um die Überlastung im Mobilfunknetz zu reduzieren. Die Probleme ähneln hier den im Kapitel 4.2.3.1 vorgestellten.

4.2.3.2.1 Unterbrechung durch Einfluss der Umgebung

Die hier beschriebene Unterbrechung wird durch die Blockierung der Funkverbindung zum Netzzugangspunkt (WLAN Router) durch Gebäude verursacht. WLAN-Netze werden stärker von der Umgebung beeinflusst als 3G Netze, was dazu führt, dass der Empfang bereits von Raum zu Raum deutlich schlechter wird.

4.2.3.2.2 Keine Netzabdeckung

Keine Netzabdeckung durch den WLAN-Router verhält sich genauso wie bei 3G-Netzen,

mit dem Unterschied, dass sich im urbanen Bereich mehr WLAN-Hotspots als 3G-Netzzugangspunkte überlagern. Auch ist es abhängig von den Sicherheitseinstellungen des Hotspot-Betreibers, ob ein Zugang möglich ist.

4.2.4 Betriebssystem

Hier werden Fehler beschrieben, die im Betriebssystem des mobilen Endgeräts auftreten Das Betriebssystem hat je nach Implementierungstiefe unterschiedliche Aufgaben, die auch von einer Middleware oder der Anwendung selbst übernommen werden können.
Die heutigen mobilen Betriebssysteme bieten vielfältige und mehr oder weniger tiefgreifende Möglichkeiten eine konsistente Verbindung herzustellen. Hierbei gibt es Unterschiede bei den implementierten Techniken als die Schnittstellen, welches den für Applikationen zur Verfügung steht. Die verschiedenen Betriebssysteme für mobile Endgeräte weisen Unterschiede in ihrer Architektur auf [TL11]. Vor allem bei Frameworks für die Kommunikation gibt es Unterschiede, die vom Programmierer beachtet werden müssen [App11].
Alle hier behandelten Betriebssysteme besitzen umfangreiche APIs für das Caching sowie ein Framework für Push-Notifications. Mobile Betriebssysteme können hierbei kooperativ oder hierarchisch agieren, wobei hier entschieden wird wieviel Freiheit dem Entwickler gegeben wird. Die Betriebssysteme bieten umfangreiche APIs zum Netzwerkmanagement an und machen damit die Anwendungs-Entwicklung so einfach wie möglich. Zum Beispiel sind die meisten Anfragen, die über die Netzwerk-API *CFNetwork Framework* von iOS laufen, bereits asynchron implementiert. iOS besitzt auch eine *Task Completion API*, die Operationen eines Programms ausführt, auch wenn das Programm nicht mehr aktiv ist. Bei *Android* heißt die API für Asynchronität *AsyncTask*. Dadurch kann relativ einfach ein konsistenter Datenfluss trotz einer Unterbrechung der Funkverbindung hergestellt werden. Außerdem bieten die meisten Betriebssysteme für mobile Endgeräte eine API-Methode, die auf eine aktive Internetverbindung prüft. Für den Offline-Modus implementiert der *Android Connectivity Manager* viele Methoden, um Details über die Internetverbindung zu erkennen sowie darauf zu reagieren.

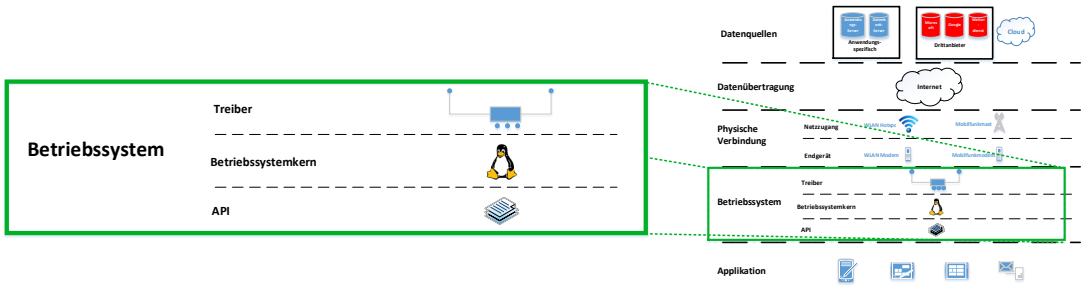

Abbildung 4.5: Betriebssystem

4.2.4.1 Wechsel von WLAN auf 3G

Der Wechsel von 3G auf WLAN und zurück wird vom Betriebssystem durchgeführt. Das kann naiv geschehen, jedoch können vom Betriebssystem Techniken zum reibungsfreien Vertical Handover umgesetzt werden. Dies hat je nach Umsetzung Einfluss auf die Anwendungsimplementierung, da der Handover von der Applikation abgefangen werden muss, wenn er nicht im Betriebssystem abgehandelt wird.

4.2.4.2 Wechseln von 3G auf WLAN: Flattern

Flattern ist ein Zustand, bei dem von einem Funkmodul ein anderes gewechselt wird und anschließend direkt zurück gewechselt wird. Ein Beispiel hierfür ist, wenn das Betriebssystem die 3G-Verbindung verliert oder aufgrund der geringen Bandbreite der Verbindung versucht, sich in einen WLAN-Hotspot einzubuchen. Dieser WLAN-Hotspot hat eine der Eigenschaften, die ihn für WLAN-Offloading unbenutzbar machen. Dies wäre beispielsweise, dass der Router ein Webinterface zum Login bereitstellt, welches nicht automatisch ausgefüllt werden kann. Ein weiteres Beispiel wäre ein Router, der generell nicht zum Internet weiterleitet. Vielerorts ist es auch der Fall, dass durch Einflüsse der Umgebung allgemein schlechter Empfang für Funktechniken herrscht. Wenn dies vom Betriebssystem erkannt wird, wechselt es in die immer noch unzureichende 3G-Verbindung und die Prozedur beginnt von vorne. Dies wird in Abbildung 4.6 illustriert. Wichtig hier ist vor allem, dass obwohl die Reaktion des Betriebssystems auf die einzel-

nen Fehler die richtige ist, sich ein Kreis im Baum bildet, der erst beendet werden kann, wenn eine der Verbindungen ausreichend verfügbar ist, um die andere zu überlagern. Auch kann es passieren, dass der Zyklus beim Login-Webinterface des Routers endet, ohne den Benutzer davon in Kenntnis zu setzen.

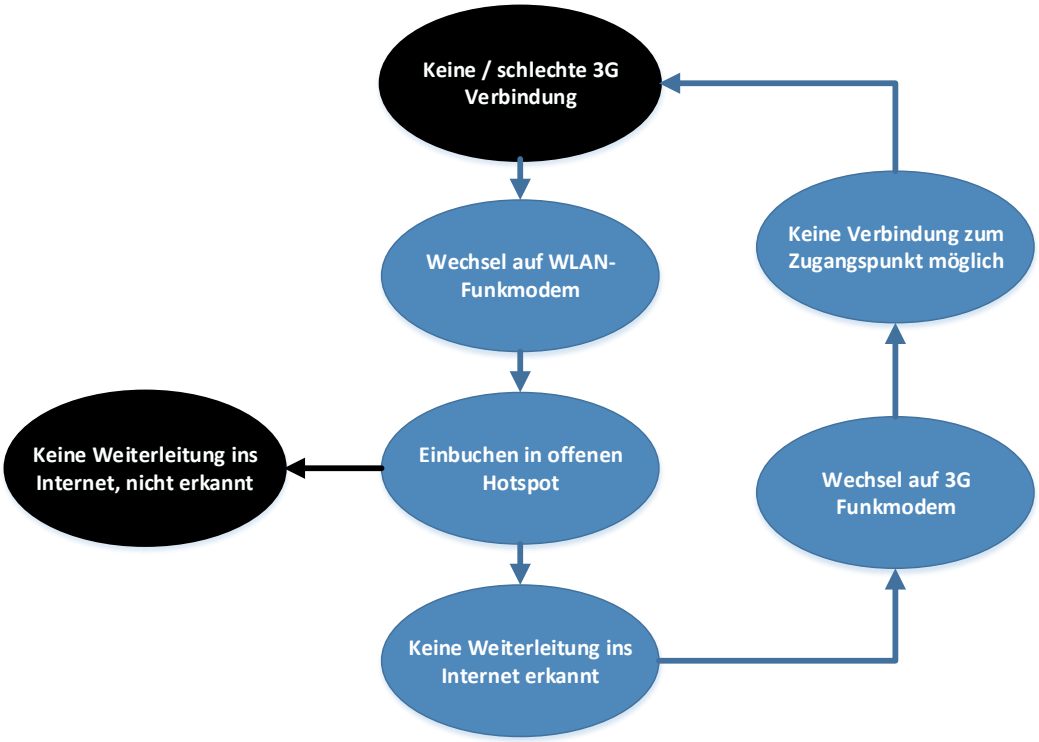

Abbildung 4.6: Fehlerbaum Flattern

4.2.4.3 Timeouts

TCP/IP - Timeouts werden bei schlechter Verbindung zu klein gesetzt bzw. nicht angepasst. Dadurch kann extremer Paketverlust auftreten, der zu einem Kombinationsfehler bei Netzüberlastung führen kann. Ein Beispiel hierfür wäre ein Netz, das eine geringe Bandbreite und hohe Netzlast hat und dadurch eine hohe Latenz aufweist. Daten können über ein solches Netz theoretisch übertragen werden, jedoch sind die Timeouts so

gering gesetzt, dass sie höher als die Latenz sind. Daher wird jedes Paket als verloren betrachtet, auch wenn es übertragen wird.

4.2.4.4 Caching im Betriebssystem

Je nach Implementierung wird Caching von Anwendungen oder dem Betriebssystem realisiert. Dies kann vor allem beim inkorrekten Anwenden der angebotenen APIs oder durch Caching in der Anwendung zu einem Kombinationsfehler führen.

4.2.5 Middleware

Bei einer Implementierung mit einer Middleware können hauptsächlich Kombinationsfehler auftreten, diese entstehen durch Wechselwirkungen mit dem Betriebssystem und der Anwendung. Dabei ist die Wahl und korrekte Konfiguration von Betriebssystem, Middleware und Anwendung wichtig für einen reibungslosen Ablauf.

4.2.6 Anwendung

In der Anwendung können viele Probleme auftreten, die oft auf unzureichende Implementierung von Offline-Funktionalität zurückzuführen ist. Auch kommt es vor, dass die Datenverbindung als robust angenommen wird, was jedoch nicht der Fall ist. Bei den implementierten Techniken in der Anwendung muss auf eine saubere Umsetzung geachtet werden. Ein grundlegendes Problem ist, dass die Anwendung alle Fehler lösen muss, die nicht in den Schichten zwischen ihr selbst und ihrer Datenquelle gelöst wurde.

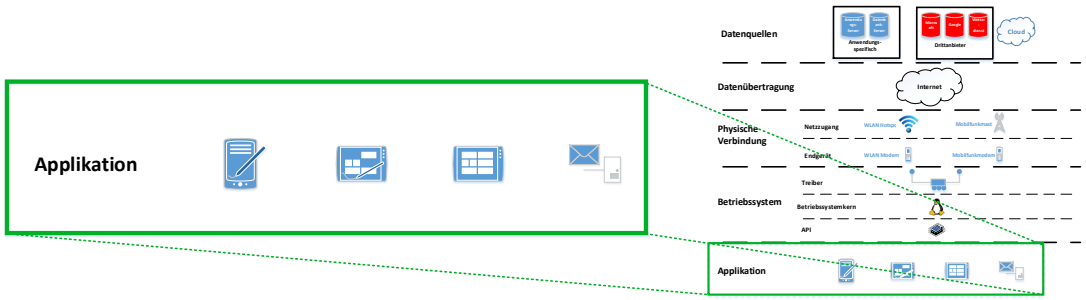

Abbildung 4.7: Anwendung

4.2.6.1 Verbindungsstatus in der Anwendung

Wichtig bei einer Anwendung ist das Beobachten und die Reaktion auf den aktuellen Verbindungsstatus. Wenn dieser nicht beachtet wird, laufen Anfragen an die Netzwerk-API ins Leere oder es treten Fehler auf. Außerdem muss das Verhalten der Anwendung bei gewissen Szenarien angepasst werden, z.B. beim Wechsel von WLAN auf 3G.

4.2.6.2 Caching und Datenspeicherung in der Anwendung

Eine Anwendung sollte, um eine verlässliche Versorgung mit Daten zu gewährleisten, Caching-Techniken implementieren. Das das einfache Herunterladen von Daten, welche für die Anwendung in einem gewissen Zeitraum gleich bleiben sein. Ein komplexeres Beispiel ist das sogenannte *Predictive Caching*, das abhängig von den angeforderten Daten weitere Daten anfordert, die mit diesen in Verbindung stehen. Dies dient dazu, Daten lokal zur Verfügung zu haben, von denen erwartet werden kann dass sie in naher Zukunft angefordert werden. Ein Beispiel hierfür sind bei einer Kartenanwendung die Umgebung des angeforderten Kartenausschnitts.
Wichtig ist hier die Unterscheidung zwischen Caching und lokalem(Offline-) Speicher zu machen, da der Cache vom Betriebssystem als flüchtig angesehen wird und nicht als permanenter Datenspeicher gedacht ist. Dies wird vor allem wichtig, wenn das Betriebssystem bei unzureichendem Speicherplatz auf dem jeweiligen Speichermedium zuerst den Cache leert.

4.2.6.3 Umfangreiche Datenübertragung

Dies beschreibt die Übertragung großer Datenmengen durch die Anwendung. Große Downloads werden häufig mit der *HTTP partial content* Methode durchgeführt, da es mit dieser einfach ist, Downloads wiederaufzunehmen. Geschieht dies nicht oder ist unzureichend implementiert, muss bei einer einfachen Verbindungsunterbrechung ein Download neu begonnen werden. Durch die zeitweise begrenzte Bandbreite und Downloads von größeren Dateien wird dies noch verstärkt. Beim Media-Streaming werden die Daten meist eine gewisse Zeit im Voraus geladen (Caching), jedoch ist oft nicht klar was geschieht, wenn eine Verbindung so lange unterbrochen ist, dass der Inhalt im Cache aufgebraucht ist. Dies muss gesondert behandelt werden, andernfalls muss hier ebenso die Verbindung komplett neu aufgebaut werden.

4.2.6.4 Verlust der Session beim Reconnect

Eine Session ist hier die Gesamtheit der benötigten Daten für eine Verbindung mit der Datenquelle, was Authentifizierungs-, Verschlüsselungs- und Identifizierungsdaten einschließt. Beim Verlust der Session muss die Anwendung die Authentifizierung und die bisher gespeicherten Daten von selbst neu laden, um die Session wiederaufzunehmen. Dies benötigt eine korrespondierende Methode bei der Datenquelle. Geschieht dies nicht, muss sie sich neu authentifizieren und die aktuell benötigten Daten neu laden (siehe Kapitel 2.2.8).
Wenn als Problemursache eine fehlende 3G-Verbindung angenommen wird, ist der Fehlerbaum in Abbildung 4.8 wie folgt definiert. Am Anfang ist kein 3G Netz verfügbar. Dies kann 3 verschiedene Ausprägungen haben. Eine davon ist die Netzüberlastung, bei der entweder eine *Intermittent internet interruption* vorliegt, beim dem der Status des Netzes nicht festgestellt werden und daher nichts unternommen werden kann, oder die Netzüberlast erkannt werden kann und durch das Betriebssystem ein Vertical Handover durchgeführt wird. Umgebungseinflüsse können nicht überwunden werden, daher muss auch hier ein Vertical Handover durchgeführt werden. Bei mangelnde Abdeckung durch den Netzzugangspunkt (3G-Funkmast) ist entweder ein anderer in Reichweite, wodurch ein Horizontal Handover, auch Roaming genannt, ausgeführt wird,

oder ebenfalls ein Vertical Handover durchgeführt wird. Der Vertical Handover kann auch durch eine Richtlinie des Betriebssystems durchgeführt werden, z.B. wenn das eigene Heimnetzwerk in Reichweite ist. Einem Vertical Handover folgt meist ein Neuaufbau der IP-Verbindung und damit einem Wechsel der IP-Adresse. Da Datenquellen die IP als Identifizierung der Clients nutzen und Anfragen nicht mehr von der IP des 3G-Netzes kommen, muss eine Verbindung neu aufgebaut werden und damit die aktuelle Session neu aufgebaut werden.

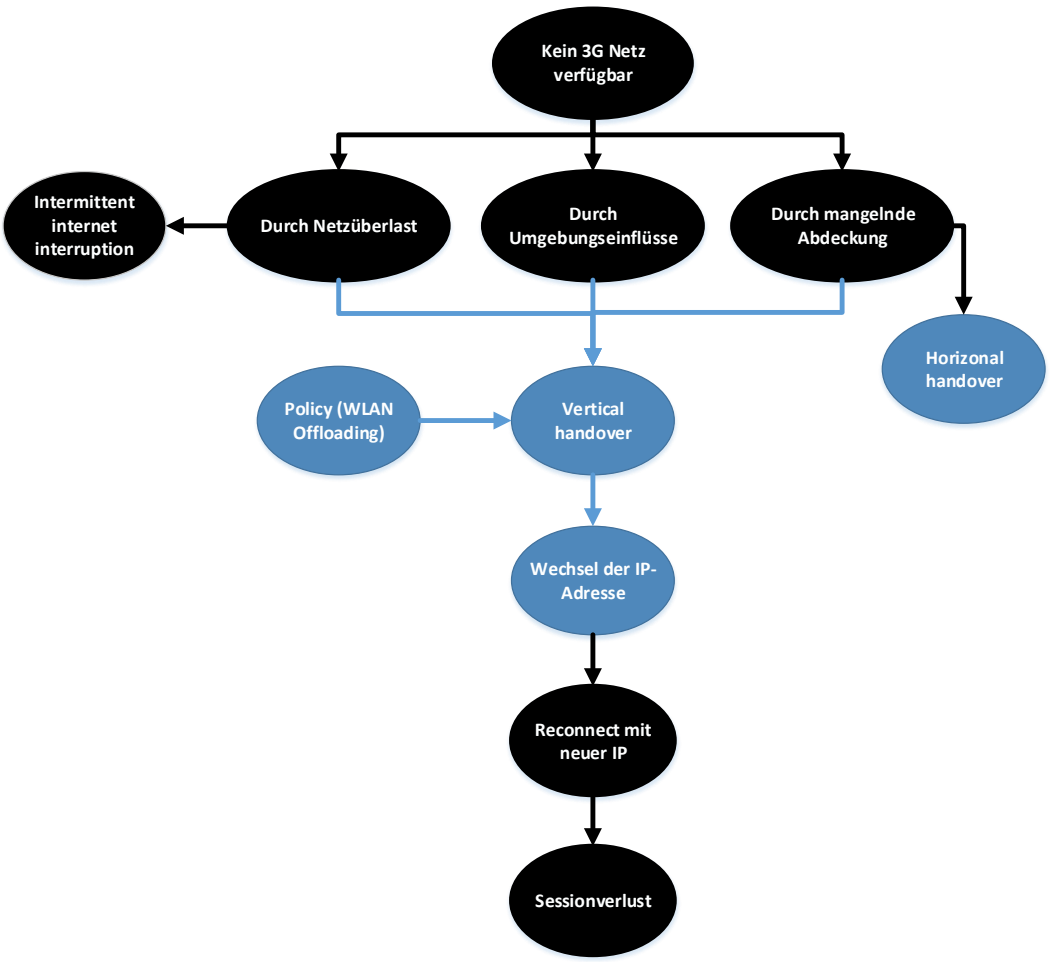

Abbildung 4.8: Fehlerbaum Sessionverlust

4.3 Zwischenfazit

Aufgrund der vielen beteiligten Schichten und ihrer heterogenen Struktur gibt es eine Vielzahl an möglichen Fehlern und auch eine nicht überschaubare Menge an Kombinationsfehlern bzw. Wechselwirkungen. Jedoch lassen sich viele Fehler auf einige wenige Ursachen zurückführen. Dadurch lassen sich Fehler meist eindeutig einer Ursache zuteilen. Über diese Zuteilung kann über eine mögliche Lösung entschieden werden.
Es ist wichtig, einen auftretenden Folgefehler korrekt zu seinem übergeordneten Fehler zurückzuführen und der richtigen Schicht nach Abbildung 1.2 einzuordnen. Durch eine inkorrekte Folgerung kann ein Fehler übersehen werden und im schlimmsten Fall Wechselwirkungen hervorrufen, die nicht zugeordnet werden können. Im nächsten Kapitel werden einige Lösungen vorgestellt und mit einer Rückverfolgung durch die Fehlerbäume versucht, einen ganzen Fehlerbaum an einer größtmöglichen Wurzel (Ursache) zu lösen.

5

Lösungen

Lösungen können entweder mit dem Fokus auf die Applikation oder das komplette mobile Endgerät und seine Datenverbindung implementiert werden. Dazwischen steht eine Implementierung mit einer MEAP mit Middleware, die meist angewandt wird, wenn für mehrere verschiedene Endgeräte verschiedene Anwendungen mit einem gemeinsamen Datenpool entwickelt werden. Die Lösungen werden nach den in Abbildung 1.2 vorgestellten Schichten eingeteilt. Auch gibt es Lösungen, die über mehrere Schichten hinausgehen, jedoch ist dabei eine klare Trennung von Zuständigkeiten seitens des umgebenden Frameworks wichtig, um Kombinationsfehler zu vermeiden und Synergieeffekte zu verstärken.

5.1 Methodik und Rückverfolgung einer Lösung im Fehlerbaum

Hierbei wird der Weg einer Lösung auf einem Fehlerbaum aus Kapitel 4 soweit zurückverfolgt, bis ein Fehler gefunden wird, der von dieser Lösung behoben werden kann. Jedoch ist es sinnvoll für einen Fehler eine Lösung zu finden, die auf eine höchstmögliche Wurzel zurückführt, um einen möglichst großen Teilbaum und damit viele Folgefehler in einem Fehlerbaum zu lösen. Hierbei wird der Fehlerbaum von einem Blatt zur Wurzel hin nach oben durchgegangen und eine Lösung gesucht. Wenn dies geschehen ist, sollte bei jedem Knoten entweder eine Lösung oder keine Lösung existieren. Hierbei wird die Lösung ausgewählt, die dem ursprünglichen Problem am nächsten kommt. Eine wurde am Beispiel von Kapitel 4.2 eine Lösung gesucht. Diese wird nun in Abbildung 5.1 dargestellt. Lösungen werden im Baum rot markiert.

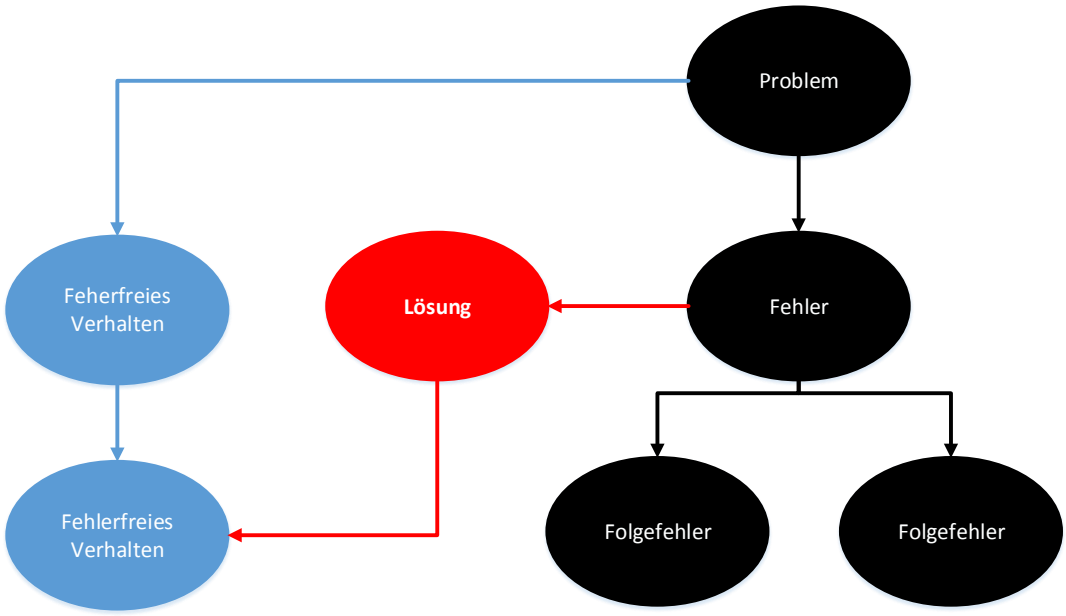

Abbildung 5.1: Beispiel eines Lösungsbaums

5.2 Global und Anwendungsspezifische Lösungen

Globale Lösungen schließen MEAP-Frameworks und das Betriebssystem mit ein. Sie können auf alle Anwendungen bezogen sein oder auch auf eine Unterklasse von Anwendungen, die z.B. die gleiche Middleware benutzen. In der Applikation können sehr viele der Probleme gelöst werden. Wichtig ist hier insbesondere die korrekte Rück-/Fehlermeldung an den Benutzer auf eine fehlende Verbindung und/oder veraltete Daten. Anwendungsspezifische Lösungen benötigen eine gute Kenntnis der API und der verwendeten Techniken seitens des Entwicklers.

5.3 Caching

Caching kann auf viele Arten durchgeführt werden. Die zwei wichtigsten sind das einfache *On-Demand Caching* und das Anticipatory Caching für das Caching von Daten, welche mit den aktuell verwendeten Daten verwandt sind.

5.3.1 In der Applikation

Beim Caching innerhalb der Anwendung ist insbesondere die Synchronisierung nach einer Verbindungsunterbrechung wichtig. Außerdem sollten die Daten zu jedem Zeitpunkt zeitsynchron sein. Des Weiteren ist es wichtig, die Caching-Hierarchie dem Datenmodell anzupassen.

Ein Beispiel für das Caching als Lösung von zu vielen Anfragen ist das Beispiel aus den Fehlern von Kapitel 4.2.1.1, welches in Abbildung 5.2 dargestellt wird. Hierbei werden zwei Lösungen dargestellt, die in der Applikationsschicht umgesetzt werden können und das Fehlverhalten der Applikation korrigieren. Selbstverständlich kann dieses Fehlverhalten nicht korrigiert werden, wenn die Datenquelle die Sperrung nicht meldet. Dies müsste man nun auf der Ebene der Datenquelle lösen, um den Fehler gar nicht erst entstehen zu lassen.

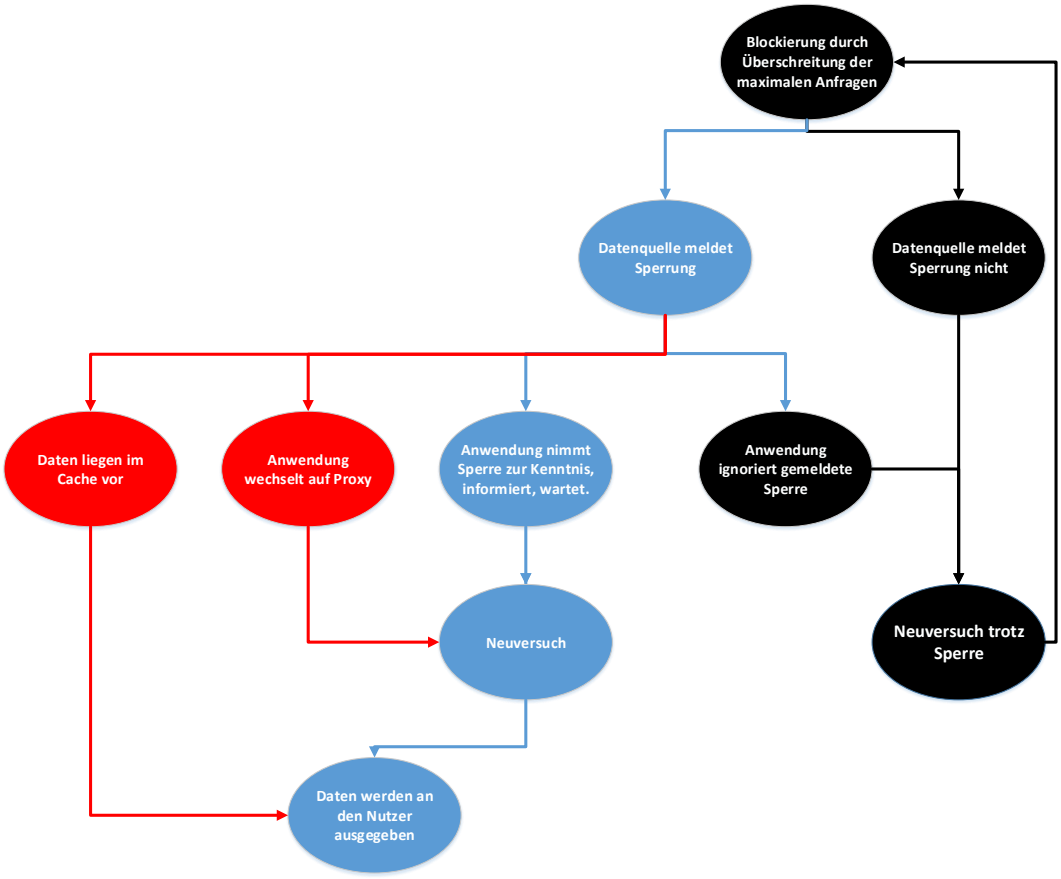

Abbildung 5.2: Darstellung des Lösungsverlaufs bei Drittserver-Limitierungen

5.3.2 Middleware

Eine Middleware bzw. Anwendungen zum Caching von Daten können in vielen Formen zu einer stabilen Datenversorgung beitragen, eine wichtige Technik ist hier das *Distributed Cooperative Caching*, bei welcher der Cache zwischen Endgeräten synchronisiert wird und damit bei globaler oder lokaler Verfügbarkeit einer 3G-Funkverbindung eine Versorung mit aktuelleren Daten ermöglicht [Dou11] [JJ12] [RKC12].

5.4 Connection Managing

Eine wichtiger Bereich ist das Aufrechterhaltung der Verbindung über mehrere Verbindungsarten hinweg. Auch ist es wichtig, über den jeweiligen Netzwerkmanager des Betriebssystems Informationen über den Verbindungsstatus an die Applikationen weiterzugeben.

5.4.1 Heatmap

Eine Idee für das Management der Verbindung ist eine Heatmap, die Informationen über die Datenverbindung an einem spezifischen Standort kombiniert mit einer Bewegungsrichtung angibt. Damit kann die Netzwerkverfügbarkeit anhand von vorhandenen Daten vorausgesagt werden und darauf basierend Einstellungen angepasst werden, z.B. das Caching-Verhalten der Anwendung oder des Betriebssystems.

5.4.2 OpenGarden Connection Manager

Eine weitere Möglichkeit des Connection Managing ist ein *Mobiles Kooperatives Ad-Hoc Netzwerk*, welches Nutzern ermöglicht ihre Mobilfunknetz-Verbindung mit anderen Nutzern über WLAN zu teilen und damit ein Mesh-Netzwerk zur besseren Überdeckung zu erzeugen. Außerdem ist es bei einem Doppelten WLAN-Interface eines Endgeräts möglich, das Mesh-Netzwerk zu einem WLAN-Hotspot mit hoher Verfügbarkeit und Bandbreite zu verbinden. Ein Beispiel für eine solche Anwendung ist die Applikation OpenGarden [Ope13a].

5.4.3 Hybridmodus für Funkmodule

Eine weitere Idee ist das gleichzeitige Nutzen von beiden Funkmodulen, um eine verlässliche Verbindung herzustellen. Jedoch müssen dabei Masking-Techniken angewandt werden, um die IP-Adresse für die Anwendungen auf dem Endgerät sowie zum Anwen-

dungsserver gleich zu halten [leo13]. Andernfalls müssen von der Anwendung sowie dem Anwendungsserver mehrere IP-Verbindungen aktzeptiert werden.

5.4.4 WLAN-Offloading

Das Wechseln von der 3G Verbindung zur WLAN-Verbindung ist eine der einfachsten Techniken, die eine Verbindung mit höherer Verfügbarkeit und Bandbreite möglich machen. Dabei wird der Datenverkehr von der 3G-Verbindung auf eine WLAN Verbindung umgeleitet, wenn die vom Benutzer gesetzten Umstände gegeben sind. Ein Beispiel ist das Einbuchen in einen WLAN-Hotspot in der eigenen Wohnung oder in der eigenen Firma. Jedoch muss auch hier die IP-Adresse mit Masking angepasst werden, um beim Wechsel eine stetige Verbindung ohne Neuverbindung herzustellen [SDA13].

5.4.5 Mobile IPv6

Mobile IP [APJ13] ist ein Standard der IETF, welcher das jeweilige Protokoll (IPv4 oder IPv6) erweitert, um vom einen Subnetz ins andere zu wechseln und dabei die gleiche IP Adresse im Client zu verwenden. Bei mobilen Endgeräten ist insbesondere der Wechsel vom 3G-Subnetz in ein WLAN-Subnetz oder der Wechsel zwischen zwei WLAN-Subnetzen wichtig.
Mobile IP kann entweder gewöhnlich eingesetzt werden, wobei dabei wichtig ist, dass der jeweilige Subnetz-Anbieter Mobile IP unterstützt, damit die Weitergabe von IP Adressen korrekt funktioniert. Wenn das nicht gegeben ist, kann *Proxy Mobile IP* [DCG+13] verwendet werden, um dieselbe Funktionalität zu gewährleisten. Diese ist jedoch nicht abhängig von der jeweiligen Unterstützung durch den Subnetz-Anbieter. Hierbei wird vom PIMPv6-Gerät (mobile mode) ein Tunnel über das Subnetz (Mobile Access Gateway) auf eine IPv4 oder IPv6-Home Adresse (auch LMA, local mobility anchor genannt) geöffnet und von diesem LMA auf das Internet zugegriffen.

5.4.6 Multipath TCP

Multipath TCP ist eine Erweiterung von TCP, welche die Nutzung von mehreren Interfaces für eine Verbindung erlaubt. Dabei wird bei Multipath TCP die Anwendung trotzdem mit einer Standard TCP Socket API angebunden. Dies erlaubt es, wie in den Artikeln *Exploring Mobile/WiFi Handover with Multipath TCP* [PDD+12] und *A Disruption-tolerant Transmission Protocol for Practical Mobile Data Offloading* [GMNP12] eine Möglichkeit gegeben, trotz der wechselnden Funktechnik einen gewöhnlichen TCP-Socket liefern zu können.
Im Fall eines einfachen TCP-Sockets wäre der Socket mit einem Fehler abgebrochen worden und die Anwendung müsste die Verbindung neu aufbauen. Implementierungen für Multipath TCP gibt es testweise für verschiedene Android-Geräte und für das iPhone ab iOS Version 7, speziell für die Applikation Siri (Speech Interpretation and Recognition Interface).
Nun wird am Beispiel aus Kapitel 4.2.6.4 verdeutlicht, wie diese Lösung den Fehler behebt und in Abbildung 5.3 dargestellt. Durch Multipath TCP bleibt bei einem IP-Wechsel im Betriebssystem der jeweilige TCP-Kanal bestehen, wenn ein Handover stattfindet. Jedoch geschieht dies nur, wenn im Betriebssystem eine Verbindung sowohl mit 3G als auch WLAN besteht. Andernfalls müsste die Lösung so implementiert werden dass dies auch bei einer noch nicht aufgebauten IP-Verbindung funktioniert.
In Abbildung 5.3 werden die Lösungen für den in Kapitel 4.2.6.4 vorgestellten Fehlerbaum dargestellt. Hierbei kann man sehen, dass je höher in der Schichten-Hierarchie ein Fehler gelöst wird, Folgefehler ausbleiben. Beim Session-Recovery durch die Anwendung speichert sich diese die Verbindungs-, Identitäts-, und Authentifizierungsdaten und nimmt die Verbindung mit der Datenquelle neu auf. Dies stellt je nach verwendetem Framework einen Implementierungsaufwand dar. Bei Multipath TCP wird die Datenverbindung durch mehrere IP-Interfaces aufgebaut und kann den Wegfall eines dieser Interfaces tolerieren, ohne dass dadurch die TCP-Verbindung gestört wird. Hierbei entfällt der Reconnect der Applikation. Bei Mobile IP bleibt die IP gleich und wird durch einen anderen mobile access gateway geroutet. Hierbei entfällt zusätzlich das Wechseln der IP-Adresse innerhalb der Anwendung. Bei einem Hybridmodus der Funk-

module wird innerhalb des Treibers die Funkverbindung anders geroutet, wodurch das Betriebssystem nicht involviert wird und kein Vertical Handover mehr stattfindet.

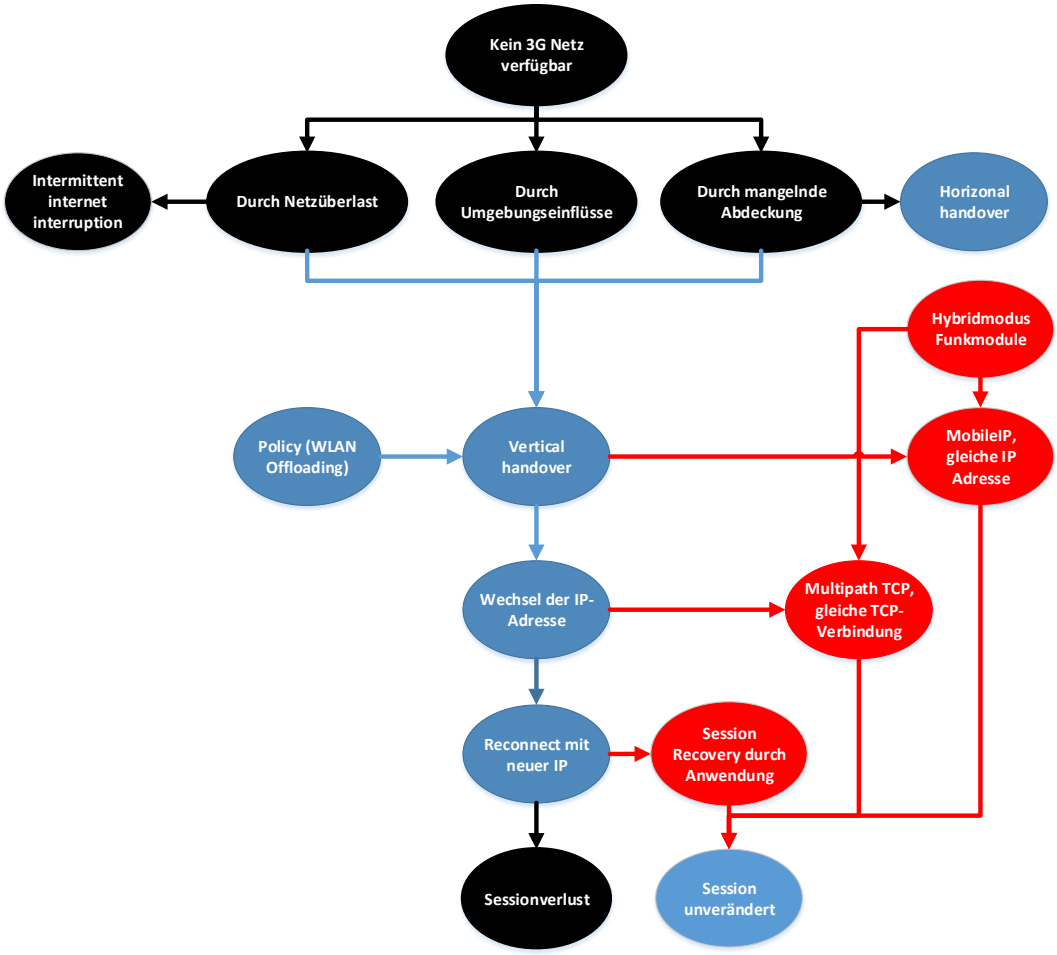

Abbildung 5.3: Anwendung von *Multipath TCP*, *MobileIP*, *Session Recovery* und *Hybridmodus für Funkmodule*

5.5 Hardware

In diesem Kapitel geht es um Hardware, welche die Physische Verbindung verbessert. Dies geschieht sowohl im Endgerät als auch auf dem Netzzugangsgerät, welches

gewöhnlich ein Mobilfunkmast oder ein WLAN-Hotspot ist. Folgend werden einige Lösungen vorgestellt.

5.5.1 Endgerät

Ein Lösung ist das gleichzeitige Nutzen aller verfügbaren Funktechniken. Dies kann auch durch das Betriebsystem geschehen, muss jedoch in diesem implementiert sein. Eine Hardware-Lösung bietet z.B. ein Chip von Qualcomm, der WLAN und 3G parallel betreibt und eine Software-Komponente auf dem Endgerät das Routing übernimmt.

5.5.2 Netz

Eine weitere Lösung ist das großflächige Ausbauen von 3G-Funkmasten. Dies ist bereits weit fortgeschritten, löst jedoch nicht die konzentrierten Datenverbrauch, den vor allem Innenstädte aufweisen. Daher gibt es verschiedene Lösungsansätze, beispielsweise das WLAN-Offloading durch den Netzbetreiber, wie z.B. die Deutsche Telekom mit Hotspot-to-go [Tel13]. Auch werden von einer EU Kommission mehr Frequenzen für die WLAN-Funktechnik angefordert, um Störungen durch Frequenzüberlagerung zu verringern [Kom13].

5.5.3 Repeater

Mobilfunkrepeater dienen dazu das Mobilfunknetz an Orten zu erweitern, die nicht durch einen Funkmasten erreichbar sind, z.B. Tiefgaragen und Stahlbetongebäude. Bisher wurden diese von Mobilfunknetzanbietern sowie freien Anbietern installiert, jedoch ist dies freien Anbieter durch ein Urteil vom 20.07.2013 des Verwaltungsgerichts Köln untersagt worden, was den Ausbau dieser Lösung hemmt.

5.6 Frameworks

Komplette Frameworks bieten Gesamtlösungen für die Applikationsentwicklung und schließen viele der erwähnten Lösungen mit ein. Eine *MEAP* ist z.B. ein Framework, welches eine Schicht die zwischen zwei oder mehreren Softwaresystemen abstrahiert. Dabei bildet sie eine meist komplexe Infrastruktur auf eine einfache Schnittstellen ab. Ein Beispiel ist eine Schnittstelle zwischen Anwendung und Betriebsystem, welche eine vielfältige Logik zur Verbindung in ein Netzwerk unterstützt. Cross-Platform Development Frameworks bieten ebenfalls eine Unterstützung für von Techniken zur robusten Datenversorgung einer Applikation an.

5.6.1 Cross-Platform Development Frameworks

Cross-Platform ist eine Eigenschaft einer Applikation, auf mehreren verschiedenen Plattformen ausgeführt lauffähig zu sein. Zur Entwicklung einer solchen Applikation gibt es umfangreiche Frameworks, die mächtige IDEs und bereitstellen. Diese ermöglichen es, eine entwickelte Applikation einfach auf mehrere Softwaresysteme zu verteilen, ohne dass man sich um die konkreten nativen Eigenschaften der betreffenden Systeme kümmern muss. Hierbei gibt es verschiedene Integrationstiefen, von einer sehr abstrakten Lösung, die nur eine HTML5-Applikation in einem Frame darstellt über Frameworks, die sich mehr den nativen Funktionen des Zielsystems zu Nutze machen bis hin zu einer nativen Applikation. Die Entscheidung, welche Integrationstiefe benötigt wird, basiert auf den Anforderungen der Applikation. Dies hat unter anderem Auswirkungen auf die Möglichkeit, Einfluss auf die Verbindung der Applikation zu nehmen. Jedoch existieren hier wiederum Techniken, die es einfacher machen die Applikation anzubinden, z.B. die JavaScript Protokoll-API Socket.IO, welche es einfach macht, eine automatische Wiederverbindung ohne Implementierungsaufwand zu bewerkstelligen.

5.6.2 MEAP (Mobile Enterprise Application Platform)

Eine *Mobile Enterprise Application Platform* erweitert den Definitionsbereich der *Cross-Platform Development Frameworks*. Sie werden vor allem von Firmen benötigt, die mehrere Applikationen in ein größeres Framework integrieren wollen. Diese haben typischerweise einen Client, der Unterstützung für Offline-Verfügbarkeit oder mit Hilfe einer Middleware einen eigenen Connectivity Manager integriert hat. Eine MEAP stellt je nach Funktionsumfang vielfältige Lösungen für instabile Datenverbindungen bereit, vorrangig jedoch für das Caching und die Synchronisierung. Zum Beispiel auch Implementierungen für Kooperatives Caching sowie Synchronisierung zwischen Clients [Ope13b].

5.7 Interpolation

Diese Technik wird hauptsächlich in der Anwendungsschicht durchgeführt. Interpolation ist eine sehr effiziente Möglichkeit mit wenig verfügbaren Daten zu arbeiten. Dies kann zum Beispiel die Zeitsynchronisierung von Daten sein, z.B. Busabfahrtszeiten anhand der verfügbaren Daten. Ausserdem ist es zur Standortbestimmung wichtig, da bei fehlendem GPS-Emfpang andere Informationsquellen zur Verfügung stehen, wie z.B. WLAN-Hotspots mit jeweiliger Funkstärke oder der Richtungsvektor des Standortverlaufs, über den der aktuelle Standort interpoliert werden kann.

5.8 Zwischenfazit

Die verschiedenen Lösungen hängen von den einzelnen Schichten und dem Aufwand ab, den man aufbringen will, um Verbindungsprobleme in der Applikation zu lösen. Auch ist es sehr wichtig die Anforderungen für eine Applikation klar zu definieren und bereits existierende Techniken kooperativ mitzuverwenden. Ein vollständiger Ausfall des Mobilfunknetzes kann nur noch mit intensivem Caching oder WLAN-Offloading behoben werden. Mit der in Kapitel 5.1 beschriebenen Methodik können Fehler im Baum

identifiziert werden, für die eine Lösung existiert. Damit kann sowohl ein Fehler als auch seine Folgefehler aus dem Baum entfernt werden, wenn die Lösung den Fehler so löst, dass keine Nebeneffekte auftreten. Andernfalls müssen die Nebeneffekte als Fehler in die Fehlerliste aufgenommen werden oder das Problem in einer tieferen Baumebene gelöst werden.

6

Fazit

In diesem Kapitel wird ein Fazit über die gewonnenen Erkenntnisse gezogen. Hierbei wird zunächst wird die Idealvorstellung der Datenversorgung einer Applikation auf einem mobilen Endgerät beschrieben. Dann wird der derzeitige Stand der Datenversorgung eines mobilen Endgeräts angegeben. Danach wird darauf eingegangen, wie die hier beschriebe Methodik dazu beizutragen soll, dieser Idealvorstellung näher zu kommen. Im Ausblick wird beschrieben, wie die hier beschriebene Methodik weiterentwickelt werden kann und wie damit viele der derzeit vorliegenden Probleme identifiziert werden können.

6.1 Idealvorstellung

Die Idealvorstellung für eine Applikation ist eine robuste Datenversorgung, welche die Anwendung stetig mit Daten versorgt. Da dies durch eine Funkverbindung nicht erreicht werden kann, müssen Techniken angewandt werden um die Unzulänglichkeiten der Funkverbindung auszugleichen. Dies sollte koordiniert auf mehreren Schichten erfolgen, um Kombinationsfehler zu vermeiden und Synergieeffekte zu nutzen. Dabei können Lösungen auch auf mehreren Ebenen angewandt werden. Wichtig ist hier das Zusammenspiel aller im Prozess beteiligten Teilnehmer. Diese sind der Benutzer, der Entwickler der Applikation, der für das verwendete Betriebssystem Verantwortliche, der Hardwarehersteller und der Mobilfunknetz-Anbieter sowie teilweise die betreffende Regulierungsbehörde.

6.2 Derzeitiger Stand

Zum jetzigen Zeitpunkt existieren in der Datenversorgung eines mobilen Endgeräts viele Herausforderungen, die es zu bewältigen gibt, um die Versorgung mit Daten zu verbessern. Einfluss darauf haben viele Faktoren, die nicht nur technischen, sondern teilweise auch politischen Ursprungs sind. Beispiele hierfür sind die Störerhaftung in Deutschland oder die geringe Möglichkeit der Einflussnahme auf ein Betriebssystems wie etwa bei iOS.
Eine große Rolle spielt hier auch die Komplexität und der Funktionsumfang der Applikation und der zur Verfügung stehenden Ressourcen. Technisch sind viele Probleme lösbar, jedoch ist die (firmen-)politische Motivation dafür nicht groß genug. Allerdings können durch derzeitige Lösungsansätze viele der Fehler behoben werden und die Datenversorgung einer Applikation deutlich erhöht werden. Zum einen Teil überlagern oder ergänzen sich diese Lösungen, zum anderen geraten sie in Konflikt miteinander, was wiederum zu Fehlern führt. Das liegt unter anderem an der losen Definition der Zuständigkeit der einzelnen Schichten und beteiligen Systeme.
Wichtig ist ein modernes Betriebssystem, welches aktives Connection Management betreibt. Außerdem müssen sich Entwickler bei der Anwendungsentwicklung des instabilen

Charakters einer mobilen Datenverbindung bewusst sein und Maßnahmen ergreifen, diese zu kompensieren. Viele Lösungen arbeiten auch darauf hin, den Aufwand für Entwickler möglichst gering zu halten und den Implementierungsaufwand in höhere Schichten zu verlagern, z.B. das Betriebssystem. Dies würde viele der Probleme lösen. Am wichtigsten jedoch ist die korrekte und schnelle Weitergabe von Fehlermeldungen und Statusänderungen zwischen den einzelnen Schichten. Dies stellt sicher dass für die darunterliegende Schicht eine Möglichkeit besteht auf einen Fehler zu reagieren. Die Datenverbindung ist zwar im generellen Falle als instabil anzusehen, jedoch muss sich jeder Entwickler bewusst sein, dass viele Unzulänglichkeiten einer Datenverbindung kompensiert werden können. Entwickler in jeder Schicht sollten dies beachten und bei der Entwicklung evaluieren, welche Möglichkeiten und Hindernisse sie zu bewältigen haben, um dem Benutzer eine robuste Versorgung mit Daten zu bieten.

6.3 Methodik

Durch das Testen einer Lösung in einem Fehlerbaum kann eine Lösung für den gesamten Baum oder einen Teilbaum gefunden werden. Durch das mehrfache Anwenden dieser Methode können alle erfassten Probleme und Fehler zumindest teilweise mit einem Lösungsvorschlag ausgestattet werden. Hierbei sind sowohl Kombinationsfehler als auch der Implementierungsaufwand der gewählten Lösungen zu beachten. Mit diesen Informationen kann nun eine Lösungsstrategie formuliert werden, mit der eine robuste Datenversorgung einer Applikation erreicht wird. Jedoch ist durch die Menge an beteiligten Systemen eine große Anzahl an Kombinationsfehler möglich, die durch falsch angewandte Lösungen noch verstärkt werden kann. Daher ist es nötig eine Lösung vor der Integration in die Applikation oder das Framework darauf zu testen, ob sie mit den anderen, bereits implementierten Lösungen zusammenarbeitet.

6.4 Ausblick

Die hier vorgestellte Methodik ist sehr einfach und man kann Fehler strukturiert darstellen. Die Granularität kann jedoch noch deutlich erhöht werden. Damit kann ein Fehler und seine Folgefehler umfassender beschrieben werden. Auch kann für die Fehler und Lösungen definiert werden, welche Techniken sie verwenden und / oder in welche sie eingreifen (z.B. Caching), um mögliche Kombinationsfehler zu vermeiden.
Eine mögliche Anwendung der vorgestellten Methodik wäre z.B. das Darstellen einiger Fehlerfälle mit dem Fokus auf Entwickler, sodass diese wissen welche Probleme sie in der Applikation lösen müssen, um den Anforderungen der Applikation zu genügen.
Auch wäre es möglich, aus einer Aufstellung von Fehlerbäumen ein Modell zu erhalten, welches gegen eine Gesamtlösung (z.B. eine SDK oder ein Cross-Platform Development Framework) verglichen werden kann. Damit könnte automatisiert für eine Menge von Fehlerbäumen nach einer Gesamtlösung gesucht werden , welche eine maximale Lösung der Fehlerbaummenge bietet (am meisten Fehler löst). Dabei können die Gesamtlösungen auch aus zur Verfügung stehenden Teillösungen erstellt werden. Damit könnte in der Planungsphase der Applikationsentwicklung bereits ermittelt werden, welches Framework zur Umsetzung der Applikation gewählt wird.
Wichtig ist jedoch vor allem, ein Bewusstsein dafür zu schaffen dass Probleme einer Datenverbindung eines mobilen Endgeräts erfassbar und strukturiert lösbar sind.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[APJ13] ARKKO, Jari ; PERKINS, Charles ; JOHNSON, Dave: *Mobility Support in IPv6*. http://tools.ietf.org/html/rfc6275. Version: 2013, Abruf: 2013-08-14

[App11] APPELQUIST, Daniel: *Smartphone Challenge: Guidelines for development of network friendl...* Version: Oktober 2011. http://www.slideshare.net/dappelquist/smartphone-challenge-guidelines-for-development-of-network-friendly-applications, Abruf: 2013-03-08

[DCG+13] DEVARAPALLI, Vijay ; CHOWDHURY, Kuntal ; GUNDAVELLI, Sri ; PATIL, Basavaraj ; LEUNG, Kent: *Proxy Mobile IPv6*. http://tools.ietf.org/html/rfc5213. Version: 2013, Abruf: 2013-08-14

[Dou11] DOUGHERTY, Brian: *Overcoming Cellular Connectivity Limitations with M2Blue Autonomic Distributed Data Caching*. http://www.cs.wustl.edu/~schmidt/PDF/M2Blue.pdf. Version: 2011, Abruf: 2013-02-21

[Fit12] FITCHARD, Kevin: *Why are mobile networks dropping like flies? — Tech News and Analysis*. http://gigaom.com/2012/07/13/why-are-mobile-networks-dropping-like-flies/. Version: 2012, Abruf: 2013-08-13

[GMNP12] GO, Younghwan ; MOON, YoungGyoun ; NAM, Giyoung ; PARK, KyoungSoo: A disruption-tolerant transmission protocol for practical mobile data offloading. In: *Proceedings of the third ACM international workshop on Mobile*

Opportunistic Networks. New York, NY, USA : ACM, 2012 (MobiOpp '12). – ISBN 978–1–4503–1208–0, 61–68

[GSM13a] GSMA: *Network Friendliness*. http://smarterappsguidelines.gsma.com/content/network-friendliness. Version: 2013, Abruf: 2013-08-12

[GSM13b] GSMA: *Smarter apps guidelines*. http://smarterappsguidelines.gsma.com/. Version: 2013, Abruf: 2013-08-15

[Jib13] JIBE MOBILE: *Jibe Whitepaper - Latency Kills*. http://www.jibemobile.com/downloads/developer/unity/Jibe_Whitepaper_Latency_Kills.pdf. Version: 2013, Abruf: 2013-09-28

[JJ12] JOY, P.T. ; JACOB, K.P.: Cooperative caching techniques for mobile ad hoc networks. In: *2012 International Conference on Data Science Engineering (ICDSE)*, 2012, S. 175 –180

[Kom13] KOMMISSION, Europäische: *EUROPA - PRESS RELEASES - Press Release - Europa liebt Wi-Fi: neue Studie empfiehlt Bereitstellung von mehr Frequenzen*. http://europa.eu/rapid/press-release_IP-13-759_de.htm. Version: 2013, Abruf: 2013-08-13

[leo13] LEO: *Let WiFi and 3G connection work together by hacking ConnectivityService.java « MobiSocial News*. http://mobisocial.stanford.edu/news/2011/03/let-wifi-and-3g-connection-work-together-by-hacking-connectivityservice-java/. Version: 2013, Abruf: 2013-08-06

[mob13] MOBITHINKING: *Global mobile statistics 2013 Part A: Mobile subscribers; handset market share; mobile operators*. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a. Version: 2013, Abruf: 2013-08-14

[Mot12] MOTOROLA: *Best practices for developing enterprise smartphone apps - Best_Practices_Dev_Ent_Smart_Apps_WP.pdf*. http://www.motorola.com/web/Business/Products/Software%20and

%20Applications/RhoMobile_Suite/_Documents/_StaticFiles/Best_Practices_Dev_Ent_Smart_Apps_WP.pdf. Version: 2012, Abruf: 2013-02-27

[Ope13a] OPENGARDEN, Inc: *You Are the Network - Open Garden*. http://opengarden.com/. Version: 2013, Abruf: 2013-08-13

[Ope13b] OPENMEAP: *OpenMEAP | Open Source Mobile Enterprise Application Platform - OpenMEAP*. http://www.openmeap.com/. Version: 2013, Abruf: 2013-08-06

[ope13c] OPENSIGNAL: *3G and 4G LTE Cell Coverage Map - OpenSignal*. http://opensignal.com/. Version: 2013, Abruf: 2013-08-14

[PDD+12] PAASCH, Christoph ; DETAL, Gregory ; DUCHENE, Fabien ; RAICIU, Costin ; BONAVENTURE, Olivier: Exploring mobile/WiFi handover with multipath TCP. In: *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*. New York, NY, USA : ACM, 2012 (CellNet '12). – ISBN 978–1–4503–1475–6, 31–36

[RKC12] RAO, A. ; KUMAR, P. ; CHAUHAN, N.: Energy efficient dynamic group caching in mobile Ad hoc networks for improving data accessibility. In: *2012 International Conference on Recent Trends In Information Technology (ICRTIT)*, 2012, S. 372 –376

[Rob11] ROBECKE, Andreas: *Development of an iPhone business application*, University of Ulm, diploma, Februar 2011. http://dbis.eprints.uni-ulm.de/716/, Abruf: 2013-09-21

[SDA13] SINGH, S. ; DHILLON, H.S. ; ANDREWS, J.G.: Offloading in Heterogeneous Networks: Modeling, Analysis, and Design Insights. In: *IEEE Transactions on Wireless Communications* 12 (2013), Nr. 5, S. 2484–2497. http://dx.doi.org/10.1109/TWC.2013.040413.121174. – DOI 10.1109/TWC.2013.040413.121174. – ISSN 1536–1276

[Sof13] SOFTWARE, Antenna: *Mobile Enterprise Magazine Report: A Guide/Case Study to Building M...* Version: Februar 2013. http://www.slides

hare.net/antenna_software/mobile-enterprise-magazine-report-a-guidecase-study-to-building-mobile-apps-with-a-meap, Abruf: 2013-08-31

[Tel13] TELEKOM: *WLAN TO GO – größtes Hotspot Netz der Welt | Telekom*. http://geschaeftskunden.telekom.de/festnetz/wlan-to-go-groesstes-hotspot-netz-der-welt/59842. Version: 2013, Abruf: 2013-08-13

[TL11] TARKOMA, S. ; LAGERSPETZ, E.: Arching over the Mobile Computing Chasm: Platforms and Runtimes. In: *Computer* 44 (2011), April, Nr. 4, S. 22 –28. http://dx.doi.org/10.1109/MC.2010.272. – DOI 10.1109/MC.2010.272. – ISSN 0018–9162

Name: Till Fischer Matrikelnummer: 655906

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Till Fischer