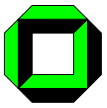

Institut für Biomedizinische Technik
Fakultät für Elektrotechnik
Universität Karlsruhe (TH)

ScatterNetz-Routing (SNR)

Multihopkommunikation für medizinische Bluetooth ad hoc Netzwerke

Diplomarbeit

vorgelegt von

cand. inform. Andreas L. Kuntz

Erstbetreuung:	Prof. Dr. Armin Bolz Dipl. Inform. Moritz Gmelin Institut für Biomedizinische Technik
Zweitbetreuung:	Prof. Dr. Wilfried Juling Dipl. Ing. Albert Krohn Telecooperation Office, Institut für Telematik Fakultät Informatik, Universität Karlsruhe (TH)
Tag der Abgabe:	31.03.2006

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

Karlsruhe, den 31.03.2006

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Durch die Entwicklung immer kleinerer Sensoren, leistungsfähigerer Mikroprozessoren und stromsparender, drahtloser Kommunikationstechniken ist es möglich geworden, Netzwerke aus mobilen Sensor- und Sensor-Aktor-Knoten aufzubauen. Im weiten Feld der potentiellen Einsatzszenarien widmet sich das DaumeN Projekt der Forschungsgruppe um Professor Bolz an der Universität Karlsruhe dem medizinischen Live-Monitoring physiologischer Patientendaten. Das Projekt soll eine umfassende Langzeitüberwachung unterschiedlichster physiologischer Daten eines Patienten ermöglichen. Dabei sollen Sensorknoten wie Körpertemparatursensoren, Pulsoxymeter, EKG, EEG, Bewegungssensoren, Herzschallsensoren, Atmungssensoren und andere in ein einheitliches Sensornetzwerk integriert werden. Die anfallenden Daten sollen per Bluetooth sowohl zwischen den einzelnen Sensoren, als auch kontinuierlich zur Diagnose und Archivierung in eine zentrale Patientenakte übertragen werden.

In einem dynamischen Netzwerk kann nicht davon ausgegangen werden, dass sich zwei kommunikationswillige Geräte zu jeder Zeit in gegenseitiger Funkreichweite befinden. Vielmehr wird es zu Problemen wie Überschreitung der maximalen Funkreichweite und temporären Abschattungen durch Hindernisse kommen, sodass sich die Kommunikationspartner nicht direkt erreichen können. In diesem Fall kann eine Weiterleitung der Daten über Zwischenknoten die Kommunikation ermöglichen. Diese Art der Kommunikation (Multihopkommunikation) wird jedoch vom Bluetooth Standard [Blue04] nicht unterstützt. Daher soll in dieser Arbeit eine Multihopkommunikation für Bluetooth-Netzwerke (Scatternetze) entwickelt und evaluiert werden.

1.2 Zielsetzung der Arbeit

Durch das geplante Einsatzszenario ergeben sich folgende Anforderungen an die Arbeit:

Es soll eine Multihopkommunikation für Bluetooth Netzwerke realisiert werden. Diese soll eine Verbindung zweier beliebiger Kommunikationspartner im Sensornetz ermöglichen. Zur Kompensation der durch Dynamik entstehenden Probleme wie temporäre Abschattung und Überschreiten der maximalen Funkreichweite soll ein automatischer Reparaturmechanismus für die Wiederherstellung unterbrochener Verbindungen sorgen.

Das System soll unabhängig von Infrastrukturkomponenten (wie z.B. Access Points) sein. Umgekehrt soll die Anwesenheit anderer, nicht zum Sensor-Netzwerk gehörender Teilnehmer die Funktion des Netzwerkes nicht beeinträchtigen.

Sowohl während der Entwicklung als auch im späteren Einsatz sollen viele, unterschiedliche Plattformen unterstützt werden, von PCs und Laptops über PDAs, Mobiltelefone bis hin zu mikrocontroller basierten Sensorknoten. Zu diesem Zweck soll auf die JSR-82 Spezifikation [Moto02] aufgesetzt werden, welche einen plattformunabhängigen Zugriff auf Bluetooth spezifiziert und für alle gewünschten Zielplattformen vorhanden ist.

1.3 Gliederung der Arbeit

Kapitel 2 vermittelt Grundlagen zum Thema Routing, zum später eingesetzten Routingalgorithmus AODV und zum Design von Bluetooth. Kapitel 3 gibt einen Überblick über den Stand der Technik bezüglich Routing in Bluetooth Netzwerken.

In Kapitel 4 wird das Problem der Multihopkommunikation analysiert und in charakteristische Teilprobleme aufgegliedert. Es werden mögliche Lösungsansätze vorgestellt, sowie auf Besonderheiten eingegangen, die sich durch den Einsatz von Bluetooth ergeben. Kapitel 5 stellt den eigentlichen Entwurf vor. Nach einem kurzen Überblick wird auf die zahlreichen Details des entwickelten Kommunikationsprotokolls eingegangen. Kapitel 6 zeigt die Struktur der Implementierung mit ihren wesentlichen Komponenten. Daneben wird auf die Realisierung des Flutens von Nachrichten in Bluetooth Netzwerken eingegangen. Die Evaluierung in Kapitel 7 zeigt die Praxistauglichkeit der vorgestellten Lösung. Abschließend gibt Kapitel 8 eine Zusammenfassung der Arbeit.

Das Abkürzungsverzeichnis auf Seite 73 schlüsselt die Bedeutung der verwendeten Abkürzungen auf. Ab Seite 75 sind wichtige Begriffe in einem Glossar zusammengestellt und erklärt.

Kapitel 2

Grundlagen

2.1 Ad hoc Routing

Routing beantwortet die Frage nach einem verbindenden Weg zwischen zwei kommunikationswilligen Knoten in einem Netzwerk. Die Einschränkung auf den „ad hoc“ Fall bedeutet, dass das Netz unabhängig von Infrastruktur und völlig selbstorganisiert arbeitet. Infrastrukturlos bedeutet insbesondere, dass jeder Knoten sämtliche Aufgaben des Routings beherrschen muss. Der Einsatzfall solcher Netze hebt sich im Allgemeinen außerdem durch hohe Dynamik von klassischen Netzen ab. Daneben unterliegen die Netzwerkknoten weiteren Einschränkungen wie geringe Energiereserven, geringe Rechenleistung und Speicherkapazität, sowie geringe Übertragungsraten.

Die für kabelgebundene Netze konzipierten Routingprotokolle können hier nicht zum Einsatz kommen, da die Knoten häufig zu wenig Speicherplatz für große, umfassende Routingtabellen haben und die Protokolle aufgrund der Dynamik des Netzes zu langsam oder gar nicht konvergieren.

Zahlreiche Strategien wurden in den letzten Jahren entwickelt und untersucht, um die Besonderheiten dynamischer ad hoc Netzwerke zu berücksichtigen und geeignet zu unterstützen. [HoXG02] und [AKKa04] ordnen die erforschten Verfahren in drei Kategorien ein:

- Flache Routing Protokolle
- Hierarchische Routing Protokolle
- Positionsgestützte Routing Protokolle

Die Kategorien sollen im Folgenden kurz vorgestellt werden. Zur Demonstration der Vielfältigkeit der unterschiedlichen Ansätze, wird jeweils auf einen Vertreter näher eingegangen.

2.1.1 Flache Protokolle

Die „Flachen“ Routing Protokolle zeichnen sich durch ihr flaches Adressierungsschema aus. Das bedeutet, Knoten werden *nicht* durch ihre Adressen in Gruppen oder sog. Cluster zusammengefasst. Alle Knoten im Netz erfüllen die selben Aufgaben in Bezug auf die Wegewahl. Weiter kann die Kategorie in proaktive und reaktive Protokolle untergliedert werden.

Proaktive Protokolle

Proaktive Protokolle teilen die Eigenschaft, Informationen über die Netzstruktur zu sammeln, schon bevor eine Anfrage nach einem Weg aufgelöst werden muss. Dazu tauschen die Geräte regelmäßig Informationen über Verbundenheit und andere zur Wegfindung notwendige Daten aus. Darauf basierend werden Pfade berechnet und Routingtabellen erstellt. Einer der wesentlichen Vorteile daraus ist die geringe Latenz zur Beantwortung einer Weganfrage. Alle notwendigen Informationen sind bereits vorhanden und die Anfrage kann umgehend beantwortet werden.

Zu Klasse der proaktiven Protokolle gehören unter anderen „Destination- Sequenced Distance-Vector Routing“ (DSDV) [PeBh94], „Fisheye State Routing“ (FSR) [PeGC00], „Optimized Link State Routing Protocol“ (OLSR) [ClJa03] und „Topology Broadcast Based on Reverse Path Forwarding“ (TBRPF) [OgLT03]. Repräsentativ soll hier OLSR kurz vorgestellt werden.

Das **Optimized Link State Routing Protocol** (OLSR) [ClJa03] benötigt als „Link State“ Protokoll globale Informationen über die Netzwerktopologie. Dazu werden periodisch Nachrichten über die Nachbarschaftsbeziehungen in das gesamte Netz geflutet. Daraufhin kann von jedem Knoten die Netzwerktopologie und damit kürzeste Pfade berechnet werden. Die Bedeutung von OLSR besteht in seiner Effizienz Nachrichten zu fluten. Jeder Knoten bestimmt eine minimale Anzahl von Nachbarknoten die ausreichen, um alle Netzwerkteilnehmer zu erreichen, die genau 2 Hops entfernt liegen. Zu flutende Nachrichten müssen nur noch an diese „Multipoint Relays“ übermittelt werden. Dadurch wird das verursachte Datenaufkommen reduziert.

Reaktive Protokolle

Die Kategorie der reaktiven Routing Protokolle zeichnet sich dadurch aus, dass Wege erst auf Anfrage bestimmt werden. Dadurch erhöht sich einerseits die bei Wegeanfragen zu erwartende Latenz, andererseits wird das durch das Routingprotokoll verursachte Datenaufkommen minimiert.

Vertreter sind „Ad Hoc On-Demand Distance Vector Routing“ (AODV) [PeBR03], „Dynamic Source Routing“ (DSR) [JoMH04], „Lightweight Mobile Routing“ (LMR) [CoEp95], „Temporally Ordered Routing Algorithm“ (TORA) [PaCo01], „Associativity-Based Routing“ (ABR) [Toh97]. Unter den vielen existierenden Protokollen wurden AODV und DSR von der „Internet Engineering Task

Force (IETF) MANET Working Group“ als favorisierte Kandidaten zur Standardisierung erklärt [HoXG02]. Da auf AODV in Abschnitt 2.2 noch genauer eingegangen wird, soll hier nur DSR erläutert werden.

Beim **Dynamic Source Routing** Protokoll (DSR) wird, sobald eine Weganfrage vorliegt, das Netz mit entsprechenden Anfrage-Nachrichten geflutet. Diese „sammeln“ auf ihrem Weg durch das Netz die Adressen aller Knoten über die sie weitergeleitet wurden ein. Der Zielknoten kann der Anfrage den vollen Pfad entnehmen und diesen zur Übermittlung einer Antwort-Nachricht nutzen. Datenpakete tragen im Paketkopf den vollen Pfad von der Quelle bis zum Ziel in Form einer Sequenz von Adressen. Zwischenknoten speichern bereits gelernte Pfade zwischen. DSR ermöglicht die Unterstützung mehrerer alternativer Pfade zum selben Ziel.

2.1.2 Hierarchische Protokolle

Hierarchische Protokolle fassen Netzwerkknoten geschickt zu Hierarchien von Gruppen zusammen. Damit lassen sich z.B. Probleme eindämmen, die entstehen, wenn man Nachrichten in sehr großen Netzwerken flutet. Hierarchische Protokolle führen damit Asymmetrien unter den Netzwerkknoten ein, indem Geräte innerhalb und außerhalb der Gruppe bestimmte Funktionalitäten zugewiesen bekommen. Die Größe ausgetauschter Routinginformationen kann so reduziert werden. Ebenfalls sinkt die Größe der benötigten Routingtabellen. Über die zu erwartende Latenz zur Beantwortung einer Weganfrage kann keine allgemeingültige Aussage gemacht werden. Sie hängt stark vom gewählten Ansatz ab.

Repräsentanten dieser Kategorie sind „Clusterhead-Gateway Switch Routing“ (CGSR) [ChGe97], „Zone Routing Protocol“ (ZRP) [HaPe01] und das „Landmark Ad Hoc Routing Protocol“ (LANMAR) [PeGH00].

Auf das **Zone Routing Protocol** (ZRP) soll im Folgenden kurz eingegangen werden. ZRP ist ein hybrides Routing Protokoll, d.h. es kombiniert proaktive und reaktive Strategien miteinander. Jeder Knoten sammelt innerhalb eines vordefinierten Radius proaktiv Informationen, um für diesen Bereich eine vollständige Routingtabelle vorhalten zu können. Für Knoten außerhalb dieses Bereichs kommt eine reaktive Strategie zum Tragen. Sobald ein Pfad zu einem Knoten unbekannt ist, muss dieser außerhalb der proaktiven Zone liegen. Solche Anfragen werden an den Rand der Zone befördert von wo aus sie — der reaktiven Strategie entsprechend — in das restliche Netz geflutet werden. Damit wird der Overhead des proaktiven Routings jeweils auf das lokale Umfeld eines Knoten beschränkt. Die hohen Latenzen reaktiver Strategien müssen nur für Ziele außerhalb der proaktiven Zone in Kauf genommen werden.

2.1.3 Positionsgestützte Protokolle

Kennen Netzwerkknoten ihre geographische Position, so kann diese Information die Effizienz des Routings steigern, denn geographische Nähe bedeutet häufig auch kurze verbindende Pfade im Netz. Entfernungen zwischen Knoten können z.B. durch

eine Messung der Funksignalstärke geschätzt werden. Durch Austausch dieser Informationen lassen sich relative Koordinaten bestimmen. Alternativ kann mit GPS[1] auch auf absoluten Koordinaten geroutet werden. Adressiert werden Knoten durch ihre Position.

Zu den Positionsgestützten Routing Protokollen gehören „Geographic Addressing and Routing“ (GeoCast) [Nalm97], „Location-Aided Routing“ (LAR) [KoVa00], „Distance Routing Effect Algorithm for Mobility“ (DREAM) [BCSW98] und „Geographic Adaptive Fidelity“ (GAF) [XuHE01].

GeoCast erlaubt es Nachrichten an sämtliche Knoten einer geographischen Region zu senden. Es existieren drei Möglichkeiten Regionen zu spezifizieren: als Punkt, als Kreis oder als Polygon. Ein geographischer Router (GeoRouter) ist für die Region zuständig, die sich aus der Vereinigung der Regionen seiner angeschlossenen Netze ergibt. GeoRouter tauschen Informationen über die von ihnen abgedeckten Gebiete untereinander aus und bilden daraus Routingtabellen. Diese Hierarchie bildet einen Baum, der Basis für die Wegfindung ist. Sollen Daten in eine Zielregion ausgeliefert werden, überprüft jeder GeoRouter ob sich die Zielregion mit seinem Zuständigkeitsbereich überschneidet. Solange die Zielregion durch den Router noch nicht vollständig abgedeckt ist, gibt er die Daten zur weiteren Bearbeitung außerhalb seines Zuständigkeitsbereiches eine Ebene im Baum nach oben. Ist der Router für einen Teil der Zielregion zuständig, sendet er die Daten an die entsprechenden Knoten.

2.2 AODV

1999 stellte Perkins in [PeRo99] das „Ad hoc On demand Distance Vector Routing“ Protokoll (AODV) vor. Es wurde als Routingprotokoll für mobile Knoten in ad hoc Netzen entworfen und seither stetig weiterentwickelt. Seit Juli 2003 steht es als RFC3561 [PeBR03] zur Verfügung.

AODV zeichnet sich durch seine schnelle Anpassungsfähigkeit an sich ändernde Verbindungsstrukturen, geringem Rechenaufwand und Speicherbedarf, sowie durch geringe Netzwerkauslastung aus. Es garantiert schleifenfreie Pfade und löst die bei Distanzvektorprotokollen üblichen Probleme wie das „Count to Infinity“-Problem [PeBR03].

AODV ist ein reaktives Routingprotokoll, d.h. Pfade werden erst auf Anfrage berechnet. Dies führt dazu, dass Knoten, die an keinem aktiven Pfad beteiligt sind weder Routinginformation pflegen müssen, noch periodische Kontrollnachrichten austauschen. Erst wenn ein Knoten mit einem anderen sprechen möchte, wird der Prozess zur Wegfindung angestoßen. Dies geschieht auf der Basis von Broadcasts. Die den gefundenen Pfad betreffenden Informationen sind dabei über alle beteiligten Knoten verteilt. Durch den Einsatz von Zielsequenznummern („Destination Sequence Numbers“) in den Signalisierungsnachrichten wird sichergestellt, dass die im Netz verteilten Routinginformationen aktuell sind.

[1] Global Positioning System

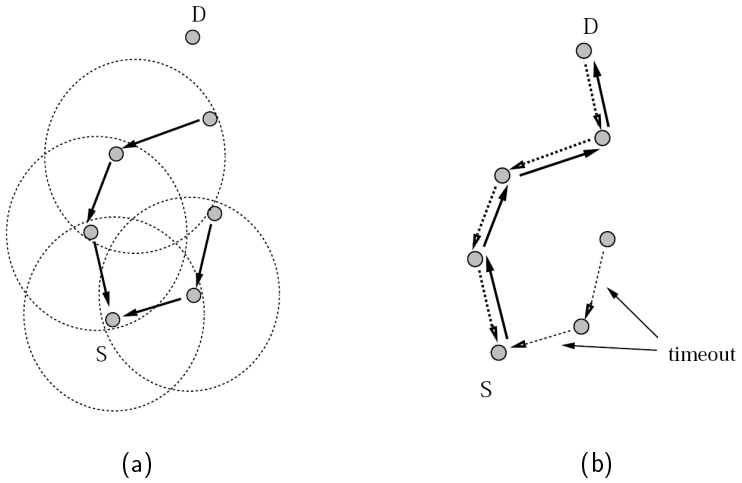

Abbildung 2.1: AODV Wegfindung. (a) Fluten einer „Route Request" Nachricht (RREQ) vom Quellknoten S in das gesamte Netz. Damit verbunden ist der Aufbau des Rückweges („Reverse Path") indem sich jeder Knoten seinen direkten Vorgänger merkt. Der Rückweg ist durch Pfeile angedeutet. (b) Der Zielknoten D beantwortet die Anfrage und sendet unicast einen RREP entlang des Rückweges zum Quellknoten S. Dabei wird der Vorwärtspfad („Forward Path") aufgebaut, der hier durch die dicken Pfeile angedeutet ist. Erreicht einen Knoten keine RREP Nachricht, werden die überflüssigen Rückweg Einträge nach Ablauf eines Timeout gelöscht.
(Quelle: [PeRo99])

Im Folgenden soll als Quelle immer derjenige Knoten bezeichnet werden, der die Wegfindung initial ausgelöst hat, als Ziel der Knoten, zu dem ein Weg gesucht wird.

2.2.1 Wegfindung

Das Auffinden eines Weges zu einem Zielknoten teilt sich in zwei Schritte:

1. Will ein Knoten mit einem anderen Knoten kommunizieren und es existieren bisher noch keine gültigen Routinginformationen für dieses Ziel, so wird der Prozess der Wegfindung angestoßen. Der Quellknoten flutet eine Route Request Nachricht (RREQ) an alle benachbarten Knoten. Jeder Empfänger, der die Anfrage nicht direkt beantworten kann, reicht sie an alle Nachbarknoten weiter. Dadurch durchdringt die Nachricht das gesamte Netz, erreicht also potentiell jeden Knoten. Um ein Kreisen der Nachrichten zu vermeiden, merkt sich jeder Zwischenknoten anhand einer Identifikationsnummer (ID) in der Nachricht, welche Anfrage er bereits bearbeitet hat. Außerdem merkt sich jeder Knoten woher er die RREQ Nachricht bekommen hat. Dieser Verweis auf den Vorgängerknoten bildet den Rückweg („Reverse Path") und wird als solcher in die Routingtabelle eingetragen. Abbildung 2.1(a) zeigt wie der Quellknoten S eine RREQ Nachricht in das Netz flutet.

   Die RREQ Nachricht enthält neben der ID noch die Sequenznummer der Quelle und die des Ziels. Sie sind für die Aktualität der Pfadinformationen zuständig. Die Sequenznummer der Quelle („Source Sequence Number")

spiegelt die Aktualität des Rückweges wieder und wird mit diesem gemeinsam gespeichert. Die letzte dem Quellknoten bekannte Sequenznummer des Zielknotens („Destination Sequence Number") spiegelt die Aktualität des Pfades zum Ziel wieder. Ein Pfad wird dann als ausreichen aktuell erachtet, wenn die mit ihm gespeicherte Sequenznummer mindestens so groß ist, wie die in der Anfrage enthaltene Zielsequenznummer.

2. Trifft der Route Request während des Weiterleitens auf einen Zwischenknoten, der einen ausreichend aktuellen Weg zum Ziel kennt oder direkt auf den gewünschten Zielknoten, antwortet dieser mit einer Route Reply Nachricht (RREP). Route Replies werden unicast entlang des in Schritt 1 aufgebauten Reverse Path geschickt, also immer nur an denjenigen Knoten, der den zugehörigen Request zuletzt weitergeleitet hat. Jeder Knoten auf diesem Pfad trägt den Nachbarn, von dem er den Route Reply erhalten hat, in seine Routingtabelle ein. So entsteht Stück für Stück der Vorwärtspfad. Abbildung 2.1(b) veranschaulicht diesen Vorgang. Die RREP Nachricht wandert vom Zielknoten D zum Quellknoten S.

2.2.2 Pflege aktiver Pfade

Die Bewegung von Knoten, die an keinem aktiven Pfad beteiligt sind, beeinflusst das Routing nicht. Bewegt sich der Knoten, der ursprünglich die Wegewahl angestoßen hat, kann der Verbindungsabbruch einer aktiven Route einfach durch erneutes Senden eines RREQ behoben werden.

Fällt auf einem aktiven Pfad eine Verbindung zwischen zwei Knoten aus, sendet der auf dem Pfad stromaufwärts liegende Knoten eine spontane RREP Nachricht an den nächsten Knoten stromaufwärts. Um die Route als ungültig zu markieren, enthält das Feld `Hop Counter`[2] den Wert unendlich. Die verwendete Sequenznummer muss dabei größer als die letzte bekannte Sequenznummer sein, sodass jeder Empfänger der Nachricht seinen entsprechenden Routingtabelleneintrag aktualisiert.

Sobald der Quellknoten die Benachrichtigung über den Ausfall einer Teilverbindung erhalten hat, kann dieser — falls die Route immer noch benötigt wird — erneut den Wegfindungsprozess anstoßen. Dazu wird eine RREQ Nachricht mit einer Zielsequenznummer, die um eins größer ist als die zuletzt bekannte, verschickt. Dies stellt sicher, dass wirklich eine neue Route gesucht wird und nicht ein Zwischenknoten, der die alte Route noch als gültig erachtet, die Anfrage beantwortet.

Anstatt den Ausfall einer aktiven Verbindung an die Nachbarknoten zu melden, kann ein Knoten unter Umständen[3] auch versuchen den Ausfall lokal zu beheben. Bei diesem optionalen Leistungsmerkmal wird eine neue RREQ Nachricht in das Netz geflutet. Um das bei Distanz Vektor Algorithmen typische „Count to Infinity" Problem zu lösen, wird als Zielsequenznummer der um eins vergrößerte Wert aus der Routing Tabelle übernommen. Falls die RREQ Nachricht unbeantwortet bleibt, löst der Knoten die zuvor beschriebene Fehlerbehandlung aus.

[2] Die AODV Nachrichtenformate sind in Anhang A dargestellt.

[3] Falls der Zielknoten nicht weiter als eine bestimmte Anzahl von Hops entfernt ist.

2.3 Bluetooth

Bluetooth ist eine Funktechnik für drahtlose ad hoc Netze. Ursprünglich zum Ersatz von Kabeln entworfen, findet Bluetooth heute breite Anwendung in Kommunikationselektronik und Computern jeder Größe. Zunehmend wird die Technik auch in Sensor Netzwerken eingesetzt.

2.3.1 Netzbildung

Bluetooth sendet im 2,4 GHz ISM-Band[4] und kann daher weltweit eingesetzt werden. Das Band ist in 79 Kanäle mit einer Bandbreite von je 1 MHz aufgeteilt. Durch das FHSS[5] Frequenz-Sprung-Verfahren werden alle Kanäle zur Kommunikation genutzt und die Robustheit gegenüber schmalbandigen Störsignalen verbessert. Dabei werden 1600 Frequenzsprünge pro Sekunde ausgeführt. Deshalb müssen die Geräte untereinander auf eine Sequenz von Frequenzen synchronisiert sein.

Piconetze

Die Abfolge der Frequenzen und die Position in der Sprungfolge definieren die Zugehörigkeit der Geräte zum selben Piconetz. Ein ausgezeichnetes Gerät, der Master, gibt die Sprungfolge und die Phasenlage in der Folge vor. Die Sprungfolge berechnet sich aus einem Teil der Bluetooth Geräteadresse des Masters; die Phasenlage richtet sich nach dessen interner Uhr. Ein Master kann weitere Geräte (Slaves) in das durch ihn definierte Netz einladen. Dies geschieht durch den Iquiry- und den Paging-Prozess (siehe Abschnitt 2.3.2). Bis zu acht aktive Knoten können im selben Piconetz kommunizieren, wobei der Master für die Koordination des Datenaustauschs verantwortlich ist. Alle Geräte in einem Piconetz teilen die selbe Sprungfolge und damit die vorhandene Übertragungsbandbreite.

Scatternetze

Je mehr Slaves im selben Piconetz kommunizieren, desto kleiner ist die dem einzelnen Gerät zur Verfügung stehende Datenrate. Daher gibt es die Möglichkeit mehrere Piconetze parallel in räumlicher Nähe zueinander zu betreiben. Jedes Piconetz hat seinen eigenen Master, der die für das Piconetz characteristische Frequenzfolge definiert. Da die Sprungfolge eines Piconetzes pseudozufällig gewählt ist, wird es nur vereinzelt zu Frequenz-Kollisionen benachbarter Piconetze kommen. So können Geräte, die keine Daten miteinander austauschen müssen, in unterschiedliche Piconetze eingeteilt werden, womit sich die Übertragungsrate pro Piconetzteilnehmer vergrößert [Schi03].

Jedes Gerät kann Mitglied in mehreren Piconetzen sein, was zu einer Verbundenheit der Piconetze untereinander führt. Die entstehende Struktur wird als

[4] Industrial, Scientific, and Medical Band. Siehe Glossar ab S. 75

[5] Frequency Hopping Spread Spectrum

Scatternetz bezeichnet. Die Spezifikation [Blue04] sieht jedoch ausschließlich eine Kommunikation innerhalb der Piconetze vor.

2.3.2 Verbindungsaufbau

Geräteadressen

Bluetooth Geräte tragen eine weltweit eindeutige Kennung, ihre Geräteadresse. Sie hat eine Länge von 48 Bit und wird vom Hersteller des Gerätes fest vergeben. Die Adressen folgen der in Abbildung 2.2 dargestellten Struktur.

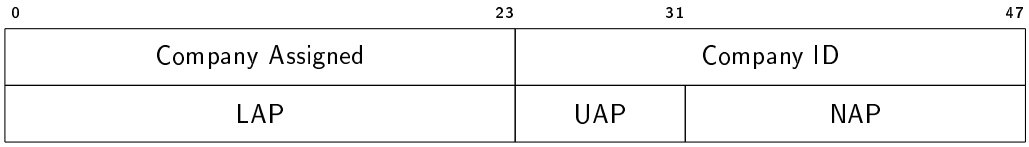

Abbildung 2.2: Struktur einer Bluetooth Geräteadresse

Die Adresse gliedert sich in den „Lower Address Part“ (LAP), den „Upper Address Part“ (UAP) und den „Non-Significant Address Part“ (NAP). Die 24 hohen Bits (UAP und NAP) der Adresse tragen eine, dem Hersteller fest zugeordnete Signatur. Der LAP wird vom Hersteller vergeben. Dabei ist darauf zu achten, dass jede Adresse nur ein einziges Mal zugeordnet werden darf.[6]

Inquiry

Bevor zwei Geräte miteinander kommunizieren können, müssen sie sich zunächst gegenseitig kennen lernen. Dazu sind Informationen über die interne Uhr und die Geräteadresse notwendig, aus der sich die Frequenz-Sprung-Folge zur späteren Kommunikation berechnet. Der Inquiry-Prozess dient der Gerätefindung. Abbildung 2.3 veranschaulicht den Vorgang in einem Weg-Zeit-Diagramm. Auf dem senkrecht nach unten verlaufenden Zeitstrahl ist die, der Sprungfolge jeweils zugrunde liegende Geräteadresse und -uhr angedeutet. Folgende Abkürzungen werden dabei verwendet:

- IAC Inquiry Access Code (anstatt einer Adresse)
- MA Master Address
- SA Slave Address
- MC Master Clock
- SC Slave Clock

Da sich die Geräte während des Inquiry noch nicht kennen, leitet sich die Frequenz-Sprungfolge nicht aus der Geräteadress sondern dem sog. Inquiry Access Code (IAC) ab. Die Synchronisation richtet sich jeweils nach der lokalen Uhr (MC bzw. SC).

[6] 64 der ca. $16 \cdot 10^6$ möglichen Werte des LAP sind fest reserviert.

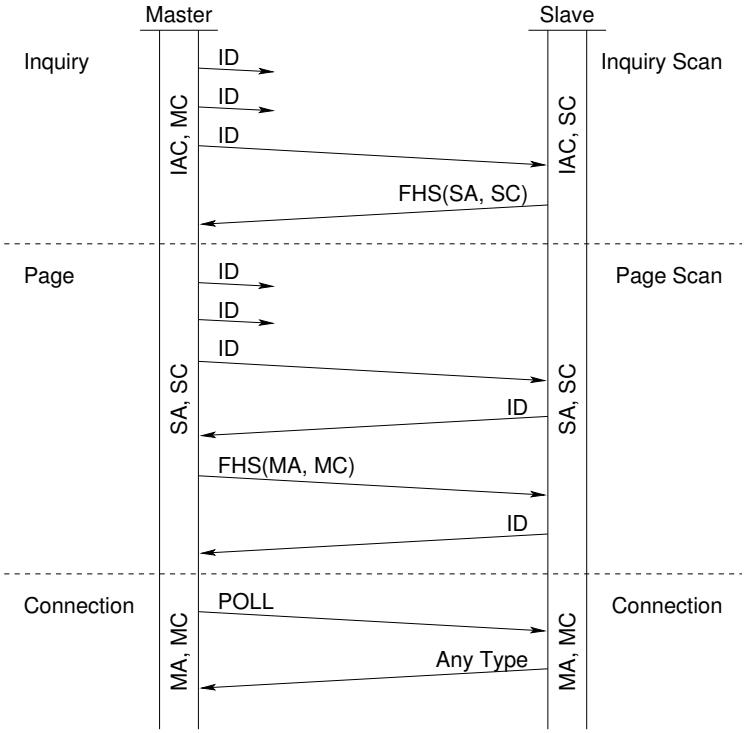

Abbildung 2.3: Ablauf von Inquiry und Paging

Ein Gerät, welches entdeckt werden möchte, führt regelmäßig einen „Inquiry Scan“ aus. Dabei wechselt es auf die durch den IAC vorgegebene Frequenz Folge. Frequenzwechsel finden nur alle 1,28 s statt.

Das suchende Gerät führt einen Inquiry aus. Es wechselt ebenfalls auf die durch den IAC und die lokale Uhr (MC) vorgegebene Sprungfolge. Dabei verschickt es auf allen Frequenzen ID-Nachrichten. Erst wenn Sender und Empfänger die selbe Frequenz gewählt haben, erreicht die Nachricht ihr Ziel. Ein vollständiger Inquiry dauert 10,24 s[7], kann aber vorzeitig abgebrochen werden.

Das im „Inquiry Scan“ befindliche Gerät antwortet mit einer FHS-Nachricht, welche die Bluetooth Geräteadresse des antwortenden Knotens, Informationen über die lokale Uhrzeit und weitere Informationen zum Gerät enthält. [Blue04]

Paging

Hat ein Gerät während des Inquiry-Prozesses Informationen über seine Nachbarn gesammelt, kann es eine Verbindung zu diesen aufbauen. Von den gesammelten Informationen ist allein die Geräteadresse für einen Verbindungsaufbau notwendig;

[7] Die Sprungfolge zum Inquiry umfasst zwei Züge zu je 16 Frequenzen. Bei einer Dauer von 625 μs pro Frequenz, benötigt das druchlaufen eines Zuges 10 ms. Bevor der Zug gewechselt werden darf, muss dieser 256 mal vollständig durchlaufen werden. Die Spezifikation empfielt mindestens drei Zug-Wechsel, um alle Antworten in einer störungsfreien Umgebung aufzusammeln: 4·(256·(16·625μs))=10,24 s

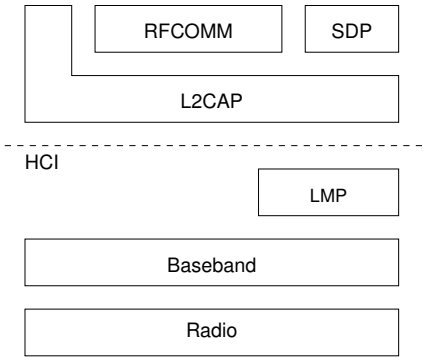

Abbildung 2.4: Bluetooth Protokollstack. Es sind nur die für Datennetze relevanten Protokolle der Bluetooth Core Spezifikation aufgeführt.

die Information über die Uhr des Kommunikationspartners beschleunigt den Prozess lediglich. Abbildung 2.3 veranschaulicht den Vorgang des Pagings.

Ein Gerät, zu dem eine Verbindung aufgebaut werden kann, führt regelmäßig einen „Page Scan“ aus. Die Frequenz-Sprungfolge richtet sich nach der eigenen Geräteadresse (SA) und Uhr (SC). Frequenzwechsel erfolgen alle 1,28 s.

Der spätere Master kann einen Verbindungsaufbau anstoßen. Dazu führt er ein „Paging“ aus. Er versucht eine Überschneidung mit dem „Page Scan“ des späteren Slave herbeizuführen. Durch die beim Inquiry erlangte Schätzung über die Uhr des Slaves und dessen Geräteadresse kann der Master die Frequenz, auf welcher der Slave eine Nachricht erwartet, vorhersagen[8]. Der Master überträgt eine ID-Nachricht, dessen Erhalt der Slave bestätigt. Diese erste Nachricht ermöglicht bereits eine erste, grobe zeitliche Synchronisation des Slaves. Anschließend überträgt der Master in einer FHS-Nachricht Informationen über seine lokale Uhr, den Code zur Berechnung der Sprungfolge, sowie eine 3 Bit Kennung[9] zur Identifikation des Slaves im Piconetz. Nachdem der Slave den Erhalt der Nachricht ebenfalls bestätigt hat, wechseln beide Kommunikationspartner auf die neue, sich am Master orientierende Sprungfolge. Zur Bestätigung des erfolgten Wechsels sendet der Master eine POLL-Nachricht, die der Slave beantwortet. [Blue04]

2.3.3 Protokollstack

Im Folgenden soll kurz auf die einzelnen Schichten des Bluetooth Protokollstack eingegangen werden. An der Luftschnittstelle beginnend, werden bis zu den anwendungsnahen Protokollen die wichtigen Eigenschaften der Schichten hervorgehoben. Abbildung 2.4 veranschaulicht die Lage der Protokolle zueinander.

[8] Um Fehler in der Schätzung zu kompensieren, werden ebenfalls die Nachbarfrequenzen, sowie eine zeitliche Verschiebung berücksichtigt.

[9] Diese Kennung heißt auch „Active Member Address“ und ist die Ursache für die Beschränkung auf 8 aktive Teilnehmer pro Piconetz.

Radio

Die Radio-Schicht bildet die Luftschnittstelle. Hier werden Frequenzbereich, Modulation und Sendeleistung festgelegt. Drei Leistungsklassen sind definiert: Klasse 1 mit einer maximalen Leistung von 100 mW und einer zu erwartenden Reichweite von 100 m bei Sichtverbindung. Die Leistungsklasse 2 strahlt maximal 2,5 mW ab. Die erwartete Reichweite bei Sichtverbindung beträgt etwa 10 m. Geräte der Klasse 3 haben eine maximale Sendeleistung von 1 mW. [Blue04, Schi03]

Basisband

Die Basisband-Schicht regelt die grundlegende Kommunikation der Geräte untereinander. Sie legt fest wie und wann auf die Funkschnittstelle zugegriffen werden darf, bietet grundlegende Mechanismen zum Verbindungsaufbau, definiert Paketformate und Dienstgüteparameter.

Wie schon erwähnt steuert der Master des Piconetzes die Kommunikation. Seine Geräteadresse und Uhr legen die Folge von pseudozufälligen Frequenzsprüngen fest. Die Kommunikation ist in Zeitschlitze unterteilt. In jedem Zeitschlitz wird eine neue Frequenz zur Datenübertragung gewählt. Ein Zeitschlitz hat die Dauer von 625 μs, womit sich 1600 Frequenzwechsel pro Sekunde ergeben. Bluetooth implementiert ein „Time Division Duplex"-Verfahren (TDD). Bidirektionale Kommunikation ist somit nicht gleichzeitig, sondern nur durch zwei aufeinander folgende unidirektionale Kommunikationen möglich. Ein Slaveknoten muss deshalb die Sendeerlaubnis des Masters abwarten. Sendet der ihm Daten, so darf der Slave im darauffolgenden Zeitschlitz antworten.

Das Paketformat der Basisband-Schicht besteht aus dem „Access Code", gefolgt von einem Paketheader. Daran schließen sich die zu übertragenden Nutzdaten an. Der „Access Code" ist charakteristisch für das Piconetz und berechnet sich aus dem LAP der Geräteadresse des Masters. Der Paketheader trägt die „Active Member Address" des an der Kommunikation beteiligten Slaves. Desweiteren enthält der Paketkopf den Typ der zu übertragenen Nutzdaten, Informationen zur Flusssteuerung[10] und eine Prüfsumme. Der Paketkopf wird immer redundant mit 1/3 Coderate kodiert. Diese Vorwärtsfehlerkorrektur (FEC) verringert die Wahrscheinlichkeit, dass der Paketkopf zerstört und erneut übertragen werden muss.

Für den Nutzdatenteil sind Pakettypen unterschiedlicher Länge und unterschiedlicher Redundanz definiert. Pakete mit einer Länge von 1, 3 und 5 Zeitschlitzen und unterschiedlichem FEC-Anteil erlauben eine Anpassung der Datenübertragung.

LMP

Das Link Manager Protokoll (LMP) dient der Verbindungsverwaltung. Es bietet Funktionen zur Authentifizierung, Paarbildung und Verschlüsselung an. Ihm ob-

[10] Bluetooth implementiert das Alternating Bit Protokoll zur Flusssteuerung

liegt die Aushandlung der verwendeten Datenübertragungsmerkmale wie die Unterstützung von Multi-Slot-Paketen, Stromsparmodi etc. Es kann Einfluss auf die Dienstgüte, Leistungssteuerung und Verbindungsüberwachung genommen werden. Außerdem koordiniert das LMP den Rollenwechsel zwischen Master- und Slave-Knoten.

HCI

Das „Host Controller Interface“ (HCI) ist eine Schnittstelle, welche höheren Protokollen den Zugriff auf wichtige Strukturen der Baseband-Schicht, des Link-Managers, sowie auf Konfigurationsparameter ermöglicht. Bluetooth ist so entworfen, dass die Funktionalität unter dem HCI häufig in Hardware, die darüber in Software implementiert wird.

L2CAP

Das „Logical Link Control and Adaptation Layer Protocol“ (L2CAP) ist ein Protokoll der Sicherungsschicht. Es bietet verbindungslose und verbindungsorientierte Datendienste, sowie die Segmentierung und Reassemblierung von Datenpaketen. Damit ermöglicht L2CAP höheren Protokollen Paketgrößen von bis zu 64 kByte Länge. L2CAP bietet durch das Multiplexen von Protokollen und logischen Kanälen eine wichtige Voraussetzung zur anwendungsorientierten Kommunikation. Alle höheren Datendienste wie SDP, RFCOMM, TCS oder BNEP setzen auf L2CAP auf. Darüber hinaus wird für jeden Kanal eine fensterbasierte Flusskontrolle, Fehlerbehandlung und Quality of Service-Aushandlung und -Überwachung angeboten.

SDP

Das „Service Discovery Protokoll“ (SDP) bietet die Möglichkeit, das Vorhandensein von Diensten und deren charakteristische Merkmale zu überprüfen. Sämtliche Informationen werden von einem auf dem Gerät laufenden SDP-Server in Form von „Service Records“ zur Verfügung gestellt. Dabei stellt SDP einen reinen Informationsdienst zur Verfügung. Es kümmert sich weder um die Aushandlung von Dienstparametern noch um den Zugriff auf den Dienst an sich.

Dienste werden durch Mengen von „Universally Unique Identifier“s (UUID) beschrieben. Ein Dienst wird durch eine Liste von UUIDs angefragt. Die Anfrage ist durch einen Dienst nur dann erfüllt, wenn der zugehörige „Service Record“ jede einzelne UUID der Anfrage aufweist. Eine gültige Anfrage muss mindestens eine UUID enthalten.

Möchte sich ein Client ohne jegliches a priori Wissen über die verfügbaren Dienste eines Knotens informieren, steht die Möglichkeit des stöberns („browsing“) zur Verfügung. Dies wird durch eine UUID ermöglicht, die allen Einträgen gemeinsam ist. Der Client sendet also eine Anfrage, welche ausschließlich die gemeinsame UUID enthält und empfängt daraufhin eine Übersicht aller verfügbaren Dienste.

RFCOMM

RFCOMM emuliert serielle RS232 Schnittstellen. Damit können Anwendungen, die auf die Kommunikation über die serielle Schnittstelle ausgelegt sind, leicht auf drahtlose Kommunikation per Bluetooth umgestellt werden.

RFCOMM emuliert alle 8 Signale der RS232 Schnittstelle und unterstützt bis zu 60 simultane Verbindungen. Eine „Null Modem“ Emulation wird ebenfalls angeboten.

2.4 JSR-82

Der „Java Specification Request“ JSR-82 [Moto02] definiert eine Bluetoothschnittstelle für die Java 2 Plattform Microedition (J2ME). Ziel der Spezifikation ist es, eine Architektur und die zugehörige API zu schaffen, um eine offene und plattformunabhängige Bluetooth-Anwendungsentwicklung zu ermöglichen. Die Entwickler hatten dabei hauptsächlich batteriebetriebene Geräte mit geringer Rechenleistung und wenig Speicher im Blick.

Die Bluetooth Spezifikation wächst durch Hinzufügen neuer Profile immer weiter. Um dieser Dynamik Rechnung zu tragen, ist das JSR-82 Design so gehalten, dass neue Bluetooth Profile auf die JSR-82 API aufsetzen und diese dadurch erweitern können. Zu diesem Zweck sind die Schnittstellen zum „Object Exchange Protocol“ (OBEX) und zum „Logical Link Control and Adaptation Protocol“ (L2CAP) vorgesehen.

JSR-82 nimmt eine Auswahl von wichtigen Protokollen vor, die einen Kompromiss zwischen umfangreicher Funktionalität und den Leistungsbeschränkungen der Zielplattform darstellt. Dazu gehört, dass ausschließlich Protokolle zur Übertragung asynchroner Daten und nicht die zur Übertragung synchroner Daten (wie z.B. Sprache) unterstützt werden. Folgende Protokolle sind verfügbar: L2CAP, RFCOMM, SDP, OBEX[11]. Die angebotenen Profile sind: GAP, SDAP, SPP, GOEP[12].

Die API stellt dabei folgende Fähigkeiten zur Verfügung: Geräte suchen, Services registrieren und suchen, Geräte verwalten, RFCOMM-, L2CAP- und OBEX-Verbindungen aufbauen. All dies geschieht in gesicherter Art und Weise. Damit spaltet sich der gesamte Umfang der API in drei Kategorien auf:

Entdeckung: Diese Kategorie bezieht sich sowohl auf Geräte als auch auf Dienste. Da es sich bei Bluetooth um einen Standard zur drahtlosen Kommunikation handelt, wird ein Mechanismus benötigt, um alle Geräte in Funkreichweite kennenzulernen. Darüber hinaus muss abfragbar sein, welche Funktionalität

[11] L2CAP: Logical Link Control and Adaptation Protocol, RFCOMM: (keine Abkürzung) Protokoll zur Emulation einer seriellen Verbindung, SDP: Service Discovery Protocol, OBEX: Object Exchange Protocol

[12] GAP: Generic Access Profile, SDAP: Service Discovery Application Profile, SPP: Serial Port Profile, GOEP: Generic Object Exchange Profile

ein bestimmtes Gerät anbietet. Die API ermöglicht zu diesem Zweck Zugriff auf das SDP Protokoll. Damit werden Funktionen zum Auffinden von Geräten und Diensten sowie zum Registrieren von Diensten angeboten.

Gerätemanagement: Das Gerätemanagement regelt, wie sich das lokale Gerät gegenüber entfernten Geräten verhält. Dies umfasst die Repräsentation des lokalen und des entfernten Gerätes durch entsprechende Klassen, Repräsentation der Bluetooth Geräteadressen, Methoden zum Zugriff auf wichtige Geräteeigenschaften, sowie zur Manipulation des Gerätestatus. Ferner umfasst dies die Kontrolle über die Bluetooth eigenen Sicherheitsfunktionen wie Verschlüsselung und Authentifikation.

Kommunikation: Hier werden die Protokollschnittstellen zu RFCOMM, L2CAP und OBEX sowie zu den unterstützten Profilen definiert. Dies umfasst unter anderem eine protokollunabhängige, einheitliche Schnittstelle zum Öffnen von Verbindungen.

[Moto02] spezifiziert ausschließlich Architektur und API. Eine Referenzimplementierung wird nicht angeboten. Daher wird in dieser Arbeit auf die auf Sourceforge frei verfügbare JSR-82 Implementierung[13] der Firma Avetana[14] zurückgegriffen.

[13] http://sourceforge.net/projects/avetanabt/
[14] Avetana Health Telematic Services, http://www.avetana-gmbh.de

Kapitel 3

Stand der Technik

3.1 RVM

Pravin Bhagwat und Adrian Segall stellten 1999 in ihrer Arbeit „A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets“ [BhSe99] einen Ansatz zum Routing in Bluetooth Scatternetzen vor. Sie setzen dabei auf einen, an die Bedürfnisse von Bluetooth angepassten „Source Routing“-Algorithmus. Jedes Paket wird mit einer Liste von Geräteadressen versehen, die den Pfad durch das Netz definiert. Die Autoren gehen dabei insbesondere auf den knappen zur Verfügung stehenden Platz in Bluetooth Basisband-Nachrichten ein. Sie ersetzen die 6 Byte langen Bluetooth Adressen durch kurze Kennungen.

Die Pfadinformation wird durch das Fluten einer SEARCH-Nachricht erlangt, welche alle Kennungen auf dem Weg zum Ziel akkumuliert. Der Zielknoten antwortet mit einer REPLY-Nachricht, welche unicast entlang des durch die Anfrage gefundenen Pfades gesendet wird. Mit der REPLY-Nachricht erhält der anfragende Knoten die gewünschten Pfadinformationen.

Der Ansatz baut auf der Bluetooth Basisband-Schicht auf und lässt höhere Protokolle außer Acht. Die Vernetzung der Bluetooth Geräte in Pico- und Scatternetze wird als gegeben angesehen und nicht weiter betrachtet. Die Arbeit enthält weder Simulationsergebnisse noch praktische Messwerte.

3.2 BlueTree

Min-Te Sun, Chung-Kuo Chang und Ten-Hwang Lai stellen in „A Self-Routing Topology for Bluetooth Scatternets“ [SuCL02] einen Ansatz vor, Bluetooth-Geräte anhand ihrer Geräteadressen in einen Baum (BlueTree) einzuordnen. Die Wegfindung ist daraufhin durch informierte Suche auf einem Baum zu realisieren.

Kommunikation kann nur entlang der Kanten im Baum stattfinden. Daten werden entweder an Knoten im selben Teilbaum „nach unten“ oder über den Wurzelknoten des Teilbaums „nach oben“ geleitet. Neue Knoten werden vom Wurzelknoten des Baumes in das Scatternetz aufgenommen. Dieser ist daher der einzige

Knoten, der Inquiries ausführen darf. Dies impliziert auch, dass sich das Netzwerk nur soweit ausdehnen kann, dass alle Knoten in Funkreichweite des Wurzelknotens bleiben. Für eine Lösung dieser Einschränkung wird auf zukünftige Arbeiten verwiesen. Sun et al. legen den Fokus auf die effiziente Konstruktion des Baumes und dessen Eigenschaften wie Balanciertheit und die Elimination unerwünschter Eigenschaften.

3.3 Scatternet Route Structure

Yong Liu, Myung J. Lee und Tarek N. Saadawi betrachten in „A Bluetooth Scatternet-Route Structure for Multihop Ad Hoc Networks“ [LiLS03] eine Reihe von Teilproblemen, die sich bei der Multihopkommunikation in Bluetooth-Netzwerken ergeben. Dabei gehen sie darauf ein, Geräte nur bei Bedarf dynamisch miteinander zu verbinden. Die Uhren der im Scatternetz eingebundenen Master sind miteinander synchronisiert, was zu einem verbesserten Durchsatz aufgrund geringerer Latenzen beim Weiterleiten der Daten führt. Mit dem selben Ziel wird ein Scheduling Algorithmus für Zwischenknoten vorgeschlagen. Liu et al. untersuchen darüber hinaus unterschiedliche Möglichkeiten, Informationen in Bluetooth Netzen effizient zu fluten. Zur Wegfindung wird ein AODV ähnliches Protokoll eingesetzt.

Da die meisten vorgestellten Lösungsansätze nur durch Veränderungen gegenüber der Bluetooth Spezifikation möglich sind, werden die Überlegungen durch Simulationsergebnisse anstatt einer Implementierung abgestützt.

3.4 ZRP

Im März 2003 stellen Rohit Kapoor und Mario Gerla von der University of California in „A Zone Routing Protocol for Bluetooth Scatternets“ [KaGe03] einen zweistufigen Ansatz für die Wegfindung in Bluetooth Scatternetzen vor. Innerhalb eines konfigurierbaren Radius von bis zu 4 Hops werden proaktiv Routinginformationen vorgehalten, außerhalb dieses Radius kommt eine reaktive Strategie zum Tragen.

Mit Hilfe von Simulationen zeigen sie, dass die Leistung ihres Ansatzes kritisch vom Radius des proaktiven Routings abhängt. Je größer dieser Wert, desto mehr Speicherplatz wird auf den Geräten benötigt und desto größer ist der durch das Routing erzeugte Kommunikationsoverhead. Dagegen sinkt die Latenz im Fall, dass eine neue Route benötigt wird.

Zur automatischen Bestimmung und Anpassung eines optimalen Wertes des proaktiven Radius, wird auf zukünftige Arbeiten verwiesen. Die Topologie der Bluetooth Scatternetze wird als gegeben angenommen, soll in Zukunft aber mit der Wegfindung einher gehen.

3.5 XHop

Jan Beutel, Oliver Kasten, Matthias Ringwald, Frank Siegemund und Lothar Thiele stellen in „Bluetooth Smart Nodes for Mobile Ad-hoc Networks“ [BKRS+03] unter anderem die Fähigkeit ihrer Plattform für Sensorknoten, „BTNodes“, zur Multihopkommunikation vor. Der Ansatz basiert auf einer funktional reduzierten Version des „CMU Dynamic Source Routing“ Protokolls. Allerdings beschränkt sich die Implementierung auf die Paketweiterleitung. Es wird angenommen, dass der Mechanismus zur Wegfindung und Pflege der Pfadinformationen unabhängig von der vorgestellten Lösung zur Paketweiterleitung realisiert ist.

Dass für die Übertragung jeder einzelnen Nachricht eine eigene Verbindung aufgebaut und nach der Übertragung wieder abgebaut wird, schlägt sich in langen Latenzen von etwa 1 Sekunde pro Verbindung nieder. Zur Behebung schlagen Beutel et al. ein zweites Bluetooth-Interface vor, womit eine Ende-zu-Ende-Verzögerung von unter 40 Millisekunden pro Hop gemessen wurde.

3.6 Übersicht

Tabelle 3.1 gibt einen Überblick über die vorgestellten Arbeiten.

Name	Sim/Impl	Routing	Scatternet Topologie	Plattform	Besonderheiten
RVM [BhSe99]	keine Angabe	Source Routing	gegeben	BT Radio	Labels statt Bluetooth Adressen
BlueTree [SuCL02]	keine Angabe	Baum-Suche	BlueTree (statisch)	keine Angabe	Baum auf BT Adressen, Wurzelknoten verwaltet Scatternetz, alle Knoten in Funkreichweite
SRS [LiLS03]	Simulation	ähnlich AODV	On Demand	keine Angabe	Verändert BT Protokoll, effizienter Broadcast, Piconetzweite Synchronisation
ZRP [KaGe03]	Simulation	Zone Routing	gegeben	oberhalb des Link Managers	Im Nahbereich proaktiv, außerhalb reaktiv (AODV)
XHop [BKRS+03]	Implementierung	minimales Source Routing	gegeben	L2CAP BTNodes	Multihop Message Passing statt Routing, mehrere BT Interfaces zur Senkung der Latenz

Tabelle 3.1: Verwandte Arbeiten

Kapitel 4

Analyse

4.1 Anforderungen

Durch den geplanten Einsatz im DaumeN Projekt ergeben sich eine Reihe von Anforderungen an die Arbeit. Diese sollen im Folgenden kurz erläutert werden.

Es kann nicht vorausgesetzt werden, dass sich zwei kommunikationswillige Knoten in einem mobilen, dynamischen Sensor Aktor Netzwerk zu jedem Zeitpunkt in gegenseitiger Funkreichweite befinden. Gründe dafür sind die beschränkte Sendereichweite der Geräte sowie die Dynamik der Netzknoten bzw. deren Umwelt. Um dennoch eine Verbindung zu ermöglichen ist eine Multihopkommunikation notwendig. Daten von nicht direkt miteinander verbundenen Knoten werden von Zwischenknoten zum Ziel weitergeleitet.

Desweiteren bedingt die Dynamik von Netz und Umwelt die Notwendigkeit unterbrochene Verbindungen reparieren zu können. Wird ein Datenpfad unterbrochen, muss ein neuer, alternativer Weg zum Ziel gefunden werden.

Das Sensor Aktor Netzwerk soll neben dem klinischen Umfeld auch im „Home Care“ Bereich eingesetzt werden können. Dabei soll das Netz unabhängig von jeglichen Infrastrukturkomponenten, wie „Access Points“ oder ähnlichen, voll funktionsfähig sein. Neu hinzukommende Knoten sollen automatisch gefunden werden und Knoten, welche das Netz verlassen, dürfen die Funktionsfähigkeit desselben nicht gefährden.

Die zugrunde liegende Funktechnologie ist weit verbreitet. Bluetooth ist bereits in viele Geräte integriert und es ist zu erwarten, dass dieser Trend weiter zunimmt. Es kann nicht davon ausgegangen werden, dass jedes bluetoothfähige Gerät zum Sensor Aktor Netzwerk gehört und die entsprechenden Dienste anbietet. Dadurch ergibt sich ein heterogenes Umfeld.

Schon während der Entwicklung, aber auch im späteren Einsatz, soll eine Reihe von unterschiedlichen Plattformen unterstützt werden. Unterschiedlichste Geräte, angefangen bei Laptops und PCs, über PDAs, Mobiltelefone, bis hin zu kleinen Microcontroller basierten Sensorknoten, sollen an der Kommunikation in einheitlicher Form teilnehmen können. Dies erfordert eine plattformunabhängige Lösung, die eine unkomplizierte Portierbarkeit gewährleistet.

Ebenfalls durch die breite Unterstützung und Einbindung unterschiedlicher Plattformen ergibt sich, dass keine Veränderungen am Bluetooth Protokollstack vorgenommen werden können.

4.2 Bluetooth versus IP Netzwerke

Im Gegensatz zu anderen Funktechniken (wie z.B. die 802.11) weist Bluetooth einige Besonderheiten auf, auf die hier kurz eingegangen werden soll.

Einer der wichtigsten Unterschiede, die Bluetooth in Bezug auf Routing mitbringt, ist der fehlende Broadcast. Zwar ist mit dem „Active Member Broadcast" eine ähnliche Funktionalität für Piconetze definiert, doch enthält dieser wichtige Einschränkungen, sodass er im Hinblick auf Multihopkommunikation nicht ausreichend ist. Ein Broadcast ist ein Rundruf an alle Geräte in Funkreichweite. Der „Active Member Broadcast" stellt ausschließlich eine Kommunikationsmöglichkeit vom Master zu allen Slaves eines Piconetzes dar. Ein Slave hat nicht die Möglichkeit des Broadcasts. Insbesondere wenn ein Knoten an mehreren Piconetzen gleichzeitig beteiligt ist, bringt der „Active Member Broadcast" nicht das gewünschte Resultat. Darüber hinaus werden in Funkreichweite befindliche Knoten, die nicht am Piconetz beteiligt sind, nicht erreicht.

Bluetooth unterstützt nur die Kommunikation innerhalb eines Piconetzes. Während andere, insbesondere IP basierte Netze, eine Weiterleitung über mehrere Hops auf höheren Protokollschichten definieren, wird darauf in der Bluetooth Spezifikation nicht eingegangen.

Eine weitere Besonderheit ist die Strategie des Medienzugriffs. Das verbreitete WLAN setzt mit CSMA/CA[1] eine probabilistische Zugriffsmethode ein, bei der es zu Kollisionen der Pakete auf dem Medium kommen kann. Der bei Bluetooth gewählte Ansatz setzt dagegen auf einer expliziten Zuteilung eines exklusiven Senderechtes auf, welches vom Master vergeben wird. Nach Erhalt des Senderechtes ist das Medium für den entsprechenden Knoten reserviert, sodass keine Kollisionen auftreten können.

Der letzte Unterschied, der hier Erwähnung finden soll, ist die Verbindungsorientiertheit von Bluetooth. Bevor zwei Geräte miteinander kommunizieren können, müssen sie eine Verbindung aufbauen. Dies gilt nicht nur für die hohen Protokollschichten, sondern gilt schon für die von der Basisband-Schicht koordinierten physischen Verbindungen. Zwei Geräte, die miteinander kommunizieren möchten, müssen sich untereinander auf den selben physischen Kanal verständigen. Dieser ist durch die Frequenz-Sprungfolge und die Phasenlage in der Sprungfolge definiert. Zum koordinierten Austausch der notwendigen Informationen sind die in Abschnitt 2.3.2 erläuterten Prozesse des Inquiry und Paging notwendig. Demgegenüber können bei anderen Funkstandards, die ein und dieselbe Frequenz für die gesamte Kommunikation verwenden, zwei kommunikationswillige Geräte direkt miteinander Daten austauschen, ohne sich vorab auf eine gemeinsame Synchronisation verständigt zu haben.

[1] Carrier Sense Multiple Access with Collision Avoidance.

4.3 Teilaufgaben und Lösungsansätze

Im Folgenden wird das Problem der Multihopkommunikation in charakteristische Teilaufgaben aufgeteilt. Neben der Beschreibung der Teilprobleme werden übliche Lösungswege angeführt. Damit soll der potentielle Lösungsraum grob skizziert werden. Falls notwendig, wird darüber hinaus auf Besonderheiten im Zusammenhang mit Bluetooth eingegangen. Abschliessend wird jeweils der für diese Arbeit relevante Lösungsansatz aufgezeigt. Diese Analyse dient als Grundlage für das im nächsten Kapitel erstellte Design.

4.3.1 Schritte zur Multihopkommunikation

Bluetooth bringt durch seine Verbindungsorientiertheit einen wichtigen Unterschied zu vielen anderen Netzwerken mit sich. Bevor zwei benachbarte Geräte miteinander kommunizieren können, müssen sie sich aufeinander synchronisieren und einen Verbindungsaufbau vollziehen. Dabei gibt Bluetooth die Sterntopologie, genannt Piconetz, vor. Der den Verbindungsaufbau initiierende Knoten wird Master. Dieser koordiniert das Piconetz und kann weitere Geräte in das Netz einladen. Mehrere Piconetze lassen sich dann zu einem Scatternetz zusammenfügen (siehe Abschnitt 4.3.6).

Nach dem Ausprägen der Topologie besteht der nächste Schritt darin, einen Pfad zwischen zwei kommunikationswilligen Knoten zu finden; also eine Kette von Geräten, die Nachrichten zwischen den beiden weiterleiten. Dieser Prozess wird mit „Wegfindung“ bezeichnet und soll in 4.3.7 näher betrachtet werden.

Im Allgemeinen wird eine Kommunikation nicht einfach zwischen zwei Geräten, sondern vielmehr zwischen zwei auf den Geräten angebotenen Diensten stattfinden. Dazu ist es einerseits notwendig die Dienste eindeutig zu identifizieren, andererseits ist einem kommunikationswilligen Knoten möglicherweise nur der Dienst, nicht aber die Adresse des dienstgebenden Knotens bekannt. In diesem Fall muss der Wegfindung eine Dienstsuche vorangestellt werden. Diese liefert die Adresse des diensterbringenden Knotens.

Ist der dienstgebende Knoten und der Pfad dort hin bekannt, kann eine Multihopverbindung aufgebaut werden. Dazu wird eine Kette von Einzelverbindungen aufgebaut und auf den entsprechenden Zwischenknoten eine Weiterleitungsfunktion eingerichtet.

4.3.2 Verbindungstyp

Kommunikationswege zwischen Netzwerkknoten können verbindungslos oder verbindungsorientiert sein. Bei einem verbindungslosen Kommunikationsweg wird jede Nachricht für sich alleine und unabhängig von vorhergehenden oder folgenden Nachrichten behandelt. Jede einzelne Nachricht muss somit sämtliche zu ihrer Interpretation und Weiterleitung notwendigen Informationen wie z.B. Adressen und Dienstbeschreibung tragen.

Bei verbindungsorientierten Kommunikationswegen existiert ein Kontext zur Verbindung. Datenpakete werden einer Verbindung und damit den im Kontext enthaltenen Informationen zugeordnet. Jede Nachricht steht demnach im Kontext aller vorangegangenen und evtl. auch der folgenden Nachrichten. Eine verbindungsorientierte Kommunikation wird durch die folgenden drei Phasen charakterisiert:

1. Verbindungsaufbauphase: hier wird durch spezielle Nachrichten einerseits der Verbindungsaufbauwunsch, andererseits die im Kontext gespeicherte Information übermittelt.

2. Kommunikationsphase: hier findet die eigentliche Kommunikation, also der Austausch der Nutzdaten, statt.

3. Verbindungsabbauphase: Einer der Kommunikationspartner baut die Verbindung ab. Der gesamte Verbindungskontext wird entfernt und die Verbindung damit geschlossen.

Bluetooth bietet ausschließlich verbindungsorientierte Kommunikationsformen an. So wird auch die hier vorgestellte Lösung diesem Ansatz folgen und auf Verbindungsorientiertheit ausgerichtet sein.

4.3.3 Signalisierung

Neben den reinen Nutzdaten müssen auch die Verbindung betreffende Informationen übertragen werden. Diese Steuer- oder Signalisierungsnachrichten tragen alle notwendigen Informationen das Kommunikationsziel, den angeforderten Dienst etc. betreffend. Im Fall einer verbindungsorientierten Kommunikation sind dies insbesondere Nachrichten zum Verbindungsaufbau und -abbau. Dabei ist es notwendig, Datenpakete von Signalisierungspaketen unterscheiden zu können. Prinzipiell stehen dafür zwei Möglichkeiten zur Verfügung: „In-Band Signalisierung“ und „Out-Of-Band Signalisierung“.

Im Fall der „In-Band Signalisierung“ werden die Signalisierungsinformationen auf dem selben Kanal wie die Nutzdaten übertragen. Die Unterscheidung der beiden Nachrichtenklassen kann zum Beispiel durch eine Markierung im Header des Pakets geschehen, welche das Paket entweder der Klasse „Datenpaket“ oder der Klasse „Signalisierungspaket“ zuordnet. Alternativ können die notwendigen Signalisierungsinformationen auch ausschließlich im Kopf der Datenpakete übermittelt werden. Damit würden eigene Signalisierungsnachrichten entfallen (dies ist z.B. bei den im Internet eingesetzten Protokollen TCP, UDP und IP der Fall). In beiden Fällen macht die „In-Band Signalisierung“ den Einsatz von Paketheadern erforderlich.

Bei der „Out-Of-Band Signalisierung“ werden die Signalisierungsinformationen in einem gesonderten Datenkanal übertragen. Dabei ergibt sich die Unterscheidung zwischen Signalisierungsnachricht und Datenpaket einfach durch den zur Übertragung verwendeten Kanal. Es kann somit vollkommen auf die Verwendung von Paketköpfen verzichtet werden. Umgekehrt müssen hier die Signalisierungsdaten

korrekt den zugehörigen Datenkanälen zugeordnet werden. Dies kann auf einfache Weise durch Kanalkennungen geschehen.

Im Fall von Bluetooth bietet sich, bedingt durch die ohnehin vorgegebene Verbindungsorientiertheit, die „Out-Of-Band Signalisierung“ an. Dadurch können insbesondere Paketheader für Nutzdaten vollständig vermieden werden. Außerdem ermöglicht die Kombination von Verbindungsorientiertheit und „Out-Of-Band Signalisierung“ den Übergang vom Routing zum Switching, was eine ausgesprochen effiziente Weiterleitung ermöglicht.

4.3.4 Dienstcharakteristik

In der Telematik unterscheidet man zuverlässige und unzuverlässige Dienste. Ein zuverlässiger Dienst ist dadurch gekennzeichnet, dass Nachrichten genau so beim Empfänger ankommen, wie sie vom Sender losgeschickt wurden. Dies bezieht sich insbesondere auf Paketverluste, Reihenfolgevertauschung, Phantompakete, Bitfehler und Duplikate von Nachrichten. Diese Fehler müssen bei einem zuverlässigen Dienst erkannt und behandelt werden. Bluetooth stellt in diesem Sinne nur unzuverlässige Kommunikationsdienste zur Verfügung. Zwar bietet Bluetooth die Möglichkeit Nachrichten mit Redundanz zu versehen, um die Resistenz gegenüber Bitfehlern und Phantompaketen zu erhöhen, doch werden Paketverluste und Duplikate nicht erkannt. Reihenfolgevertauschung tritt erst im Fall von Multihopkommunikation auf, wenn Datenpakete unterschiedliche Pfade durch das Netz nehmen. Dieser Fehlertyp ist daher für die Singlehopkommunikation innerhalb von Bluetooth Piconetzen nicht relevant.

Der Tatsache, dass Bluetooth nur unzuverlässige Dienste anbietet, muss in unterschiedlicher Weise Rechnung getragen werden. Einerseits soll die Charakteristik des angebotenen Multihop Datendienstes der des Singlehop Dienstes entsprechen. Andererseits müssen Fehler in der Übertragung von Signalisierungsnachrichten erkannt werden. Daher werden Paketverluste, Duplikate und Reihenfolgevertauschung für Signalisierungsverbindungen gesondert betrachtet.

Paketverluste werden dadurch abgefangen, dass für jede einzelne Signalisierungsnachricht eine Ankunftsbestätigung vom Empfänger gefordert wird. Duplikate können auf unterschiedliche Weise behandelt werden. Entweder jeder Knoten kann erkennen, ob eine neu ankommende Nachricht bereits behandelt wurde, oder die Nachrichtenbehandlung ist so ausgelegt, dass die wiederholte Behandlung derselben Nachricht unkritisch ist. So müssen Duplikate garnicht erst detektiert werden. Hier muss zwischen Speicherbedarf und Rechenaufwand abgewogen werden. Ebenfalls unkritisch ist der Fehlertyp der Reihenfolgevertauschung. Signalisierungsnachrichten können, ob ihrer konstanten Größe, stets in einem einzigen Datenpaket übertragen werden. Somit sind Reihenfolgevertauschungen für Signalisierungsnachrichten ausgeschlossen.

Datenpakete unterliegen der selben Problematik. Insbesondere ist bei der Weiterleitung von Datenströmen mit einer hohen Anzahl von Paketverlusten zu rechnen. Dies liegt darin begründet, dass Zwischenknoten zur Weiterleitung abwechselnd an unterschiedlichen Piconetzen teilnehmen müssen. Eine Datenübertragung

an einen „abwesenden“ Knoten schlägt fehl, ohne dass der Sender den Fehler bemerkt. Diese Paketverluste können jedoch durch Bestätigungsnachrichten abgefangen werden.

4.3.5 Erkennen der Nachbargeräte

Möchte ein Knoten mit einem anderen Knoten im Netz kommunizieren, so muss der erstgenannte zumindest ein weiteres, direkt erreichbares Gerät kennen, welches ihn mit dem restlichen Netz verbindet. Im Allgemeinen handelt es sich jedoch um mehr als nur ein Gerät. Direkt erreichbar heißt hier, dass es sich in Funkreichweite befinden muss.

Viele Netzwerke sind broadcastfähig[2], d.h. sie stellen eine Möglichkeit bereit eine einzige Nachricht an alle erreichbaren Knoten zu senden. Dies kann z.B. durch eine Broadcastadresse gelöst sein, also eine dedizierte Adresse durch die alle Teilnehmer gleichzeitig angesprochen werden können. Damit ist es möglich Geräte anzusprechen, deren Adresse a priori nicht bekannt ist.

So ergeben sich zwei Möglichkeiten bislang unbekannte, aber erreichbare Nachbarknoten zu entdecken:

1. Jeder Knoten informiert in regelmäßigen Abständen alle anderen Knoten über die eigene Existenz. Dazu reicht eine Nachricht aus, welche die eigene Adresse enthält und per Broadcast übermittelt wird.

2. Alternativ kann ein Knoten auch alle anderen Knoten dazu auffordern, sich bei ihm zu melden. Wieder wird eine Nachricht — in diesem Fall die Aufforderung sich zu melden — per Broadcast übermittelt. Daraufhin meldet sich jeder Empfänger beim Initiator der Aufforderung.

Im Fall von Bluetooth greifen diese Lösungen jedoch nicht, da Bluetooth aufgrund des Frequenzsprungverfahrens nicht broadcastfähig ist. Zu keinem Zeitpunkt ist sichergestellt, dass beliebige, bislang unbekannte Geräte auf derselben Funkfrequenz empfangsbereit sind. Stattdessen stellt Bluetooth das Verfahren des Inquiry bereit, um erreichbare Geräte einander bekannt zu machen (siehe Abschnitt 2.3.2). Dabei werden Informationen über die Geräteadressen, sowie zur geräteeigenen Uhr übermittelt, was eine zeitliche Synchronisation der Kommunikation ermöglicht. Schon die beim Inquiry übermittelte Information reicht aus, um eine Liste direkt erreichbarer Nachbarn zusammenzustellen.

Weiterhin stehen bei Bluetooth zwei Möglichkeiten zur Verfügung die Liste der benachbarten Geräte aktuell zu halten:

1. Zum einen kann ein Knoten in regelmäßigen Abständen immer wieder erneut Inquiries durchführen. Je nach erwarteter Dynamik des Netzes kann die Zeitspanne zwischen den erneuten Erkundigungen angepasst werden.

[2]Broadcast wird gelegentlich bildlich mit „Rundruf“ übersetzt. Hier wird allerdings weiterhin der englische Begriff verwendet, da er gebräuchlicher ist.

2. Zum anderen steht seit Version 1.1 der Bluetooth Spezifikation die Möglichkeit zur Verfügung ein Gerät in den sogenannten „Periodic Inquiry Mode“ zu versetzen[3]. Dabei führt der Controller selbständig und regelmäßig Inquiries aus. Die Besonderheit dabei ist, dass der Inquiry nicht in seiner vollen Länge durchlaufen wird, sondern nach einer vorgegebenen Zeitspanne unterbrochen wird. Dadurch finden die Inquiries im Hintergrund statt und die normale Kommunikation muss nicht für die Dauer eines gesamten Inquiries unterbrochen werden.

Die JSR-82 Spezifikation stellt keinen Zugriff auf den „Periodic Inquiry Mode“ zur Verfügung, sodass für diese Arbeit Alternative 1 gewählt wurde.

4.3.6 Ausprägung der Netztopologie

Bluetooth arbeitet verbindungsorientiert, d.h. bevor zwei Geräte miteinander kommunizieren können, müssen sie eine Kommunikationsverbindung aufbauen. Dabei organisieren sich die Knoten in Piconetze, die von jeweils einem dedizierten Gerät, dem Master, verwaltet werden. Jedes Gerät innerhalb des Piconetzes kann ausschliesslich mit seinem Master, nicht aber mit anderen Slaves, kommunizieren. Piconetze lassen sich zu größeren Gebilden, den sogenannten Scatternetzen, zusammenfügen. Dies geschieht indem ein Knoten Mitglied in mehr als einem Piconetz ist. Dieser Knoten kann entweder eine Doppelrolle erfüllen, indem er Master in einem und Slave im andere Piconetz ist oder er fungiert als Slave/Slave Brücke und hat in beiden Piconetzen die Rolle eines Slaves.

Bei der Ausprägung der Netztopologie wird einerseits die Lage der Kanten im Verbindungsgraph festgelegt, also welche Knoten direkt miteinander verbunden sind. Dabei entscheidet sich die Anzahl und die Länge der zwischen zwei Knoten existierenden möglichen Pfade.

Andererseits wird die Rollenverteilung (Master/Slave) zwischen den Knoten bestimmt, was einen wichtigen Einfluss auf die Anzahl und Lage der Piconetze hat. Die dem Piconetz zugrunde liegende Hoppingsequenz[4] ist durch den jeweiligen Master vorgegeben. Damit hängt die Wahrscheinlichkeit, dass zur selben Zeit in zwei Piconetzen die selbe Funkfrequenz verwendet wird und somit eine Kollision der Datenpakete stattfindet, direkt mit der Anzahl in Funkreichweite befindlicher Piconetze zusammen. Kollisionen stören die Kommunikation und vermindern den Datendurchsatz. Um die Kollisionswahrscheinlichkeit (bei gegebener Knotenverteilung) zu minimieren, sollte daher eine möglichst kleine Anzahl von Piconetzen angestrebt werden.

Daneben ist eine möglichst robuste Topologie wünschenswert. Veränderungen in der Netzstruktur durch neu hinzukommende oder wegfallende Geräte sollten

[3] Der zugehörige Befehl wird am Host Controller Interface bereit gestellt. Siehe [Blue04] Band 3, Kapitel 7.1.3

[4] Sequenz von Frequenzen die nacheinander zur Kommunikation ausgewählt werden. Siehe Glossar ab S. 75

wenig Einfluss auf die Topologie und wenig Änderungen an der Rollenverteilung der einzelnen Knoten nach sich ziehen.

Yong Liu et al. diskutieren zu diesem Thema in [LiLS03] zwei unterschiedliche Ansätze der Rollenverteilung: Bei dem als „Single Role Approach“ bezeichneten Ansatz hat jedes Gerät genau eine Rolle. Es ist entweder nur Master oder in allen Piconetzen Slave. Damit gelingt die Verbindung zweier Piconetze ausschließlich durch Slave/Slave Brücken. Die beschriebene Rollenverteilung hat den Vorteil eine minimale Anzahl an Piconetzen aufzuweisen und damit die Kollisionen von Frequenzen gering zu halten. Weiter fanden Kalia et al. in [KaGS00] heraus, dass diese Rollenverteilung einen hohen Datendurchsatz und geringe Verzögerungszeiten aufweist.

Daneben wird ein weiterer Ansatz, der „Double Role Approach“ betrachtet. Hier ist ein Knoten sowohl Master in einem, als auch Slave in einem weiteren Netz. Jeder Knoten hat zwei Rollen. Diese Struktur zeichnet sich durch eine wesentlich höhere Rubustheit gegenüber Veränderungen im Netzwerk aus.

Aufgrund der geringeren zu erwartenden Interferrenzen und des höheren Durchsatzes, soll hier der „Single Role Approach“ angestrebt werden. Allerdings scheint es nicht sinnvoll diese Struktur in jeder Situation aufrecht erhalten zu wollen. Denn dann würden sich — durch das Hinzukommen eines neuen Knoten oder den Wegfall eines vorhandenen Knoten — immer wieder Rollenwechsel und Strukturänderungen ergeben, die sich im schlimmsten Fall durch das gesamte Netz propagieren. Vielmehr soll der Ansatz verfolgt werden, beim Aufbauen von neuen Verbindungen auf die alternierende Rollenverteilung entlang eines Pfades zu achten. Anschließende Korrekturen werden zugunsten der Robustheit unterlassen.

Die Topologie des Netzes wird nicht statisch a priori festgelegt, sondern während des Prozesses der Wegfindung dynamisch und nur bei Bedarf ausgeprägt. Dies hat den Vorteil, dass Knoten, die momentan weder primär (als Datenquelle oder -senke) noch sekundär (als Zwischenknoten auf einer aktiven Route) am Datenaustausch beteiligt sind, in einen Stromsparmodus wechseln können. Außerdem sind hierdurch die Geräte genau so miteinander verbunden, wie es die aktuelle Kommunikationssituation erfordert. Dadurch werden zusätzliche Kosten (Energiereserven, Rechenzeit) eingespart, die notwendig wären, um unbenutzte Verbindungen aufrecht zu erhalten.

4.3.7 Wegfindung

Die Wegfindung ist das Kernproblem bei der Kommunikation über mehrere Hops. Sie beantwortet die Frage nach einem verbindenden Pfad zwischen zwei Geräten.

Zahlreiche Routingprotokolle wurden dazu entwickelt und untersucht. Einen Überblick über unterschiedliche Klassen von Protokollen gab bereits Abschnitt 2.1. Für diese Arbeit sind davon jedoch nur die infrastrukturlos arbeitenden und darunter die reaktiven Protokolle von Interesse, da letztere sehr schnell auf Dynamik im Netz reagieren können.

Vielen reaktiven Protokollen ist dabei der Ansatz gemeinsam, Weganfragen aufgrund mangelnder Informationen in das Netz zu fluten und Antworten unicast auf dem selben Weg, den die Anfrage genommen hat, zurück zum Initiator der Anfrage zu senden. AODV [PeRo99] und DSR [JoMH04] arbeiten nach diesem Prinzip. Der entscheidende Unterschied zwischen diesen Protokollen liegt darin, wo später die eigentlichen Informationen über gefundene Pfade liegen. DSR bringt als „Source Routing" Protokoll die gesamte Information über den Pfad zum anfragenden Knoten. Jedem Datenpaket muss demzufolge wieder sämtliche Pfadinformation beigefügt werden. Bei AODV dagegen sind die Knoten im Netz zustandsbehaftet. Sie halten jeweils die lokal relevanten Teile der Pfadinformation vor, welche von allen Pfaden zum selben Ziel gemeinsam genutzt werden. Ein Datenpaket trägt nur die Zieladresse und kann somit auf die Reise geschickt werden, ohne dass vorher der gesamte Pfad zum Ziel bekannt sein muss. Dadurch lassen sich in die Wegsuche noch weitere Aufgaben integrieren, was ausschlaggebend für die Entscheidung war, die Wegfindung sehr stark an AODV zu orientieren. Allerdings muss auf Bluetooth spezifische Besonderheiten eingegangen werden. Hierzu gehören insbesondere die Verbindungsorientiertheit und die Bluetooth eigene Adressierung von Diensten.

4.3.8 Dienste

Ein Knoten im Netz kann und soll häufig mehr als nur eine einzige Aufgabe erfüllen. Die Aufgaben werden als Dienst bezeichnet. Eine Kommunikation soll daher eigentlich nicht mit einem Knoten, sondern mit dem dort angebotenen Dienst geführt werden. Bluetooth charakterisiert Dienste anhand von „Universally Unique Identifiers" (UUID). Aufgrund des großen Wertebereichs[5] dieser Dienstidentifikatoren können sie verteilt und dennoch praktisch kollisionsfrei gewählt werden. Sobald ein Dienst auf einem Knoten gestartet wird, bekommt dieser einen weiteren Identifikator zugewiesen. Diese Dienstnummer ist nur für die Laufzeit des Dienstes und auch nur auf dem lokalen Knoten gültig. Dafür kann der Wertebereich bedeutend kleiner sein.

Ein kommunikationswilliger Knoten wird im Allgemeinen nur den Bezeichner des Dienstes kennen mit dem er kommunizieren möchte, nicht aber die Adresse des Knotens, der den Dienst anbietet. Daher ist — wie eingangs schon erwähnt — eine Dienstsuche notwendig, um den diensterbringenden Knoten und damit dessen Adresse ausfindig zu machen. Ein weiteres Resultat der Dienstsuche kann die lokal auf dem Zielknoten gültige Dienstnummer sein.

Bluetooth bietet mit dem „Service Discovery Protokol" die Möglichkeit einer Dienstsuche an, die allerding auf Piconetze beschränkt ist. Da eine zusätzliche Erweiterung des gesamten „Service Discovery Protocols" auf Multihop Dienstsuche den Umfang dieser Arbeit sprengen würde, soll hier nur eine rudimentäre Dienstsuche basierend auf einer einzigen UUID als Dienstbezeichner betrachtet werden. Dabei wird der Dienstbezeichner in die Geräteadresse des diensterbringenden Knotens und der dort lokal gültigen Dienstnummer aufgelöst.

[5] Eine UUID hat eine Länge von 128 Bit und kann damit etwa $3{,}4 \cdot 10^{38}$ unterschiedliche Werte annehmen.

4.3.9 Datenkanäle

Entscheidet man sich für eine verbindungsorientierte Kommunikation, so besteht die Notwendigkeit, Datenkanäle zwischen den Diensten der Kommunikationspartner aufzubauen. Insbesondere bei der „Out-Of-Band Signalisierung“ kann der bereits zur Signalisierung aufgebaute Kanal nicht zum Übertragen der Daten herangezogen werden. Vielmehr muss für eine Datenverbindung ein eigener Kanal geschaffen werden.

Eine Designentscheidung bleibt, die Daten mehrerer Verbindungen zum selben Ziel über den selben Kanal zu schicken oder aber für jede Verbindung einen eigenen, dedizierten Kanal zwischen den beiden Kommunikationspartnern aufzubauen. Beide Möglichkeiten bieten Vor- und Nachteile. Vorteil der gemeinsam genutzten Datenkanäle ist, dass insgesamt weniger Datenkanäle aufgebaut und verwaltet werden müssen. Dafür besteht die Notwendigkeit, die Daten auf den geteilten Kanälen derart zu markieren, dass sie eindeutig der Quelle bzw. dem Ziel zugeordnet werden können. Dadurch ist wieder die Einführung von Headern zu den Datenpaketen notwendig, was die Verarbeitung durch Auswerten der Header verlangsamt. Kommen dedizierte Kanäle zwischen Quelle und Ziel zum Einsatz, kann auf Header gänzlich verzichtet werden. Allein die Zuordnung der Daten zum Kanal erlaubt die Identifikation von Ziel- und Quellknoten. Zugunsten dieses Vorteils soll hier der Nachteil der größeren Anzahl an notwendigen Kanälen in Kauf genommen werden.

Ein Kanalaufbau kann zu unterschiedlichen Zeitpunkten geschehen. Eine strenge Trennung der Aufgaben im Sinne des Schichtenmodells würde vorschreiben, zunächst einen Pfad zum Zielknoten zu finden und erst anschließend den Verbindungsaufbau zum gesuchten Dienst des Knotens zu initiieren. Lässt man dieses Ziel außer Acht, kann man durch Zusammenfassen von Wegfindung und Verbindungsaufbau die Anzahl der zu übertragenden Signalisierungsnachrichten verringern. Ebenfalls verringert wird dabei die zu erwartende Latenz beim Verbindungsaufbau für den Fall, dass noch keine Pfadinformation vorliegt.

4.3.10 Weiterleitung der Daten

Ist eine Datenverbindung zwischen zwei Knoten etabliert, müssen ankommende Daten in Richtung Ziel weitergeleitet werden. Diese Aufgabe ist unabhängig von der Wegfindung und sollte aus Effizienzgründen möglichst weit „unten“ im Protokollstack untergebracht sein.

Zum ersten Mal softwareseitig zugreifbar werden ankommende Datenpakete im Fall von Bluetooth am Host Controller Interface (HCI). Im Kopf von ACL[6]-Datenpaketen auf HCI-Ebene befindet sich ein „Connection Handle“ Feld, welches das Datenpaket eindeutig einer Verbindung zuordnet (siehe [Blue04]). Weiterleitung auf dieser Ebene — ohne die vorgegebenen Strukturen zu verändern — würde bedeuten, den „Connection Handle“ in ankommenden Paketen anzupassen und wieder über das Host-Controller-Interface an den Controller zurückzugeben, der das

[6] ACL=Asynchronous Connection-oriented transport Layer. ACL ist der hier relevante, zur Übertragung von Nutzdaten eingesetzte Verbindungstyp im Bluetooth Basisband.

Paket dann entsprechend der Angabe im „Connection Handle“ Feld an den nächsten Knoten weiterleitet. Diese Vorgehensweise wird in der Literatur gemeinhin als Switching bezeichnet, da nicht eine beliebige Geräteadresse in einer großen Tabelle (ggf. durch Prefixmatching) nachgeschlagen werden muss, sondern einfach eine Nummer aus einem kompakten Zahlenraum durch eine neue ersetzt werden kann.

Das Weiterleiten kann auch eine Ebene höher im Bluetooth Protokollstack, auf L2CAP-Ebene vorgenommen werden. Dadurch ergibt sich eine zusätzliche Verzögerung, die aus der weiteren Verarbeitung der Daten innerhalb der L2CAP-Schicht herrührt. Insbesondere kann L2CAP von höheren Schichten kommende Datenpakete vor dem Versenden segmentieren und muss diese im Gegenzug nach dem Empfangen zunächst puffern, dann zusammensetzen, bevor sie an höhere Schichten weitergegeben werden können. Die Datenpakete sind — wie zuvor auf HCI-Ebene — fest einer Verbindung zugeordnet. Somit muss nach dem Empfang eines Pakets nur die zugehörige Verbindung ermittlt werden. Da es sich hier um eine bijektive Abbildung der Verbindungen zueinander handelt, können die beiden einander zugeordneten Verbindungen auch gemeinsam verwaltet werden, womit sich das Ermitteln der zugehörigen Verbindung erübrigt. Wieder ist die Vorgehensweise in Bezug auf die Komplexität eher mit Switching als mit Routing zu vergleichen.

Bluetooth Anwendungen setzen meist auf L2CAP oder der darüber gelegenen RFCOMM Schicht auf. Weiterleitung auf HCI Ebene würde daher erfordern, am „Host Controller Interface“ eine Zwischenschicht einzuziehen, welche ankommende, an das Gerät adressierte Daten an die L2CAP Schicht weiterreicht, welche die Daten der Anwendung zur Verfügung stellt. Leider ist eine Zwischenchicht auf HCI Ebene nicht plattformunabhängig realisierbar, sodass der Weiterleitung auf Basis von L2CAP Verbindungen der Vorzug gegeben wird.

Beim Weiterleiten der Datenpakete lässt es sich ab einer Pfadlänge von 2 Hops nicht mehr vermeiden, dass ein Knoten Daten von einem Piconetz in ein anderes transportieren muss. In jedem der Piconetze wird die Kommunikation vom Master gesteuert. Dabei werden anhand der Uhr des Masters Zeitschlitze festgelegt, die den einzelnen Geräten zur Kommunikation zugewiesen werden. Die Uhren der Master benachbarter Piconetze sind allerdings nicht miteinander synchronisiert, sodass ein weiterleitender Knoten an der Grenze zwischen zwei Piconetzen mitunter lange auf den Beginn des nächsten Zeitschlitzes und dann auf die Kommunikationserlaubnis des entsprechenden Masters warten muss. [LiLS03] schlägt zur Behebung des Problems vor, die Uhren der Master scatternetzweit miteinander zu synchronisieren. Dieser Vorgang ist allerdings nicht innerhalb der Bluetooth Spezifikation möglich und erfordert Veränderungen an der Hardware. Der Ansatz wird hier daher nicht weiter verfolgt werden.

4.3.11 Pflege gefundener Pfade

Ad hoc Netzwerke weisen typischerweise eine hohe Dynamik auf. Neue Knoten werden in das Netz aufgenommen, andere verlassen die Funkreichweite des Netzes oder werden abgeschaltet. Auch das physikalische Umfeld der Knoten kann sich

verändern, was unter Umständen Auswirkungen auf die Struktur des Netzes haben kann.

Die Dynamik im Netz kann also dazu führen, dass existierende Verbindungen plötzlich und unvorhersehbar ausfallen. Damit werden Teile der gesammelten Routinginformationen ungültig, weil sie Wege beschreiben, die nicht mehr existieren. Umgekehrt können auch neue Pfade entstehen, die kürzer oder effizienter als die bisher bekannten sind. Routingtabelleneinträge können daher nicht statisch sein, sondern müssen regelmäßig aktualisiert werden. Gleichzeitig besteht nach einem Ausfall die Notwendigkeit neue Routen zu finden, um unterbrochene Datenverbindungen wieder herzustellen und anfallende Daten ans Ziel zu bringen.

Reparaturmechanismen sind zum Teil Bestandteil der Routingprotokolle. Routen werden mit Zeitstempeln versehen, die sie bei Nichtbenutzung altern lassen. AODV bietet darüber hinaus die Möglichkeit des Reparaturversuchs bei Verbindungsausfall. Fällt irgendwo auf einer aktiven Route eine Verbindung zwischen zwei Knoten aus, so wird ausgehend von der Stelle des Ausfalls eine neue Verbindung zum Ziel gesucht. Schlägt der Reparaturversuch fehl, werden Informationen über den Verbindungsausfall und über nicht mehr erreichbare Knoten in die Gegenrichtung (vom Ausfall weg) propagiert, um die Routingtabellen entsprechend aktualisieren zu können.

Daneben existieren Ansätze, welche schon vor einem konkreten Verbindungsausfall alternative Pfade suchen, um dann bei einem Ausfall schneller reagieren zu können. M. Marina stellt in [MaDa01] eine Erweiterung des Routingprotokolls AODV vor, die anstatt eines einzigen Pfades gleich mehrere, möglichst disjunkte Pfade zum Ziel vorhält. Falls eine Verbindung ausfällt, kann der Fehler durch das Wechseln auf einen der alternativen Pfade mit minimaler Latenz behoben werden.

Für diese Arbeit sollen die AODV-eigenen Reparaturmechanismen herangezogen werden, wie sie in [PeBR03] beschrieben sind. Dies umfaßt Timeouts für Routingtabelleneinträge und den Reparaturversuch bei Verbindungsausfall.

Kapitel 5

Entwurf

5.1 Protokollüberblick

Zur Multihopkommunikation sind die in Abschnitt 4.3 analysierten Teilaufgaben zu lösen. Die für den Kommunikationsaufbau wichtigen sind Topologiekontrolle, Wegfindung, Dienstsuche und Verbindungsaufbau. Die Behandlung von Verbindungsausfällen stellt die dauerhafte Verfügbarkeit einer bestehenden Verbindung sicher.

Dabei kombiniert SNR die zum Kommunikationsaufbau notwendigen Schritte so miteinander, dass sie nahtlos ineinander über gehen. Ein Verbindungsaufbau löst Dienstsuche, Wegfindung und falls notwendig auch die Topologieausprägung aus. Wobei die Prozesse nicht nacheinander, sondern miteinander ablaufen.

Die Topologieausprägung wird durch die Wegfindung gesteuert, d.h. alle zur Wegfindung notwendigen Verbindungen werden aufgebaut und automatisch wieder abgebaut, falls keine Kommunikation darüber stattfindet. So ist sichergestellt, dass langfristig nur Verbindungen existieren, welche aktiv am Datenaustausch beteiligt sind.

Die Wegfindung des SNR Protokolls ist sehr stark an das in [PeBR03] definierte AODV Routing Protokoll angelehnt. Dabei wurden jedoch einige Veränderungen zur besseren Anpassung an die Bluetooth eigenen Gegebenheiten vorgenommen. Hier soll zunächst in einem kurzen Überblick die Idee vermittelt werden, bevor in den darauf folgenden Abschnitten das Protokoll detailliert beschrieben wird.

Eine Anfrage zum Verbindungsaufbau wird so lange in das Netz geflutet, bis sie auf einen Knoten trifft, der über valide Pfadinformationen verfügt. Von hier an wird die Anfrage direkt auf dem gefundenen Pfad in Richtung Ziel weitergeleitet. Der Zielknoten überprüft, ob der angeforderte Dienst zur Verfügung steht und beantwortet die Anfrage dementsprechend. Fällt die Antwort positiv aus, so wird gemeinsam mit dem Propagieren der Antwort zum anfragenden Knoten ein dedizierter Datenkanal aufgebaut. Ein Datenkanal besteht aus einer Kette von 1-Hop Einzelverbindungen. Sobald die Verbindungsaufbau-Antwort ihr Ziel erreicht hat, steht der volle Kommunikationskanal zur Verfügung.

Routinginformationen werden dabei — genau wie bei AODV — während des Flutens der Anfrage in das Netz, sowie während des Weiterleitens der Antwort durch die Knoten im Netz gesammelt und verwaltet. Die bei AODV typischen Vorteile wie eine schnelle Anpassungsfähigkeit an sich verändernde Verbindungssituationen, geringer Speicherbedarf und Rechenaufwand bleiben dabei gewahrt.

5.2 Bezeichnungen

Quellknoten/Zielknoten: Die Bezeichnung Quellknoten („originator") und Zielknoten („destination") bezieht sich ausnahmslos *immer* auf die Quelle bzw. das Ziel der CREQ Nachricht. Dies gilt insbesondere auch für CREP Nachrichten, die vom gesuchten Zielknoten ausgesandt und in Richtung CREQ Quelle weitergeleitet werden. Diese Konvention vermeidet Bezeichnungsverwirrung wenn nacheinander Nachrichten unterschiedlicher Flussrichtung betrachtet werden.

Hinweg/Rückweg: Die beiden Bezeichnungen beziehen sich auf die Richtung der Routingtabelleneinträge, die während der Verarbeitung der Signalisierungsnachrichten angelegt werden. Wieder orientiert sich die logische Richtung an der Ausbreitungsrichtung der Verbindungsaufbauanfrage (CREQ). Der Hinweg (oder „forward path") verweist damit vom Quellknoten ausgehend in Richtung Zielknoten, während der Rückweg in die entgegengesetzte Richtung zeigt.

Verbindung/Pfad/Kanal: Während der Begriff „Verbindung" im Folgenden für Singlehop Verbindungen zwischen direkt benachbarten Knoten reserviert sein soll, bezeichnen die Begriffe „Pfad" und „Kanal" Kommunikationswege über mehrere Hops bzw. Multihop Verbindungen.

stromabwärts/stromaufwärts: Durch den Datenkanalaufbau wird jedem Datenkanal eine Richtung aufgeprägt. Diese Richtung hat nichts mit dem späteren Fluss der Nutzdaten zu tun, da alle Verbindungen bidirektional sind. Gemeint ist die logische Richtung der Initiative zum Kanalaufbau, die vom Clientknoten in Richtung Serverknoten ging (entspricht der Richtung der CREQ-Nachricht). Verläufe in Richtung dieser Initiative sollen als „stromabwärts" bezeichnet werden („downstream"); die entgegengesetzte Richtung als „stromaufwärts" („upstream").

5.3 Datenstrukturen

Dieser Abschnitt stellt die von SNR verwendeten Datenstrukturen vor. Insbesondere wird auf den Aufbau der Signalisierungsnachrichten eingegangen, welche alle wichtigen Informationen des Signalisierungsprozesses tragen und übermitteln.

SNR-Nachricht	Name	AODV Äquivalent
CREQ	Connection Request	RREQ
CREP	Connection Reply	RREP
CREL	Connection Release	keine Entsprechung
ACK	Acknowledge	keine Entsprechung

Tabelle 5.1: SNR Signalisierungsnachrichten

SNR kennt vier Signalisierungsnachrichten. Aufgrund der starken Verwandtschaft mit AODV sind die SNR-Nachrichten zum Datenkanalaufbau den AODV-Nachrichten sehr ähnlich. Die Verbindungsorientiertheit von SNR bringt allerdings gravierende Unterschiede in der Signalisierung von Fehlersituationen mit sich. So kommt SNR ohne ein Pendant zur AODV RERR-Nachricht aus, benötigt dafür jedoch eine Möglichkeit, den ordnungsgemäßen Abbau von Datenkanälen mitzuteilen. Zusätzlich definiert SNR eine Bestätigungsnachricht (ACK), welche bei AODV in dieser Form nicht benötigt wird. Tabelle 5.1 listet die Nachrichten auf und stellt sie dem jeweiligen AODV Äquivalent gegenüber.

5.3.1 Connection Request Nachricht (CREQ)

Die Connection Request Nachricht signalisiert einen Datenkanalaufbauwunsch. Die Verbindung soll zu einem Dienst aufgebaut werden, der auf unterschiedliche Weise in der Nachricht spezifiziert sein kann. Drei Varianten der Adressierung sind möglich:

1. Unter Angabe der UUID des Dienstes,
2. unter Angabe der UUID des Dienstes und der Bluetooth Geräteadresse des gewünschten Zielknotens oder
3. unter Angabe der Bluetooth Adresse des Zielknotens und des PSM[1].

Die drei Möglichkeiten werden im Abschnitt 5.4.1 genauer erläutert. Nachfolgende Abbildung 5.1 fasst den Aufbau der Nachricht zusammen, wobei die Felder folgende Bedeutung tragen:

Type: 8 Bit Typfeld. Legt den Nachrichtentyp fest, hat immer den Wert 1.

D: 1 Bit Unknown-Destination-Sequence-Number-Flag. Ist gesetzt, falls dem Quellknoten noch keine Sequenznummer des Zielknotens bekannt ist.

U: 1 Bit UUID-Search-Flag. Zeigt an, dass der gesuchte Dienst durch die UUID anstatt durch den PSM identifiziert wird.

Reserved:
Reserviertes, 14 Bit breites Feld. Alle Bits werden beim Senden zu 0 gesetzt und beim Empfang ignoriert.

[1]Protocol Service Multiplexer. Bluetooth Dienstbezeicher. Siehe Glossar ab Seite 75

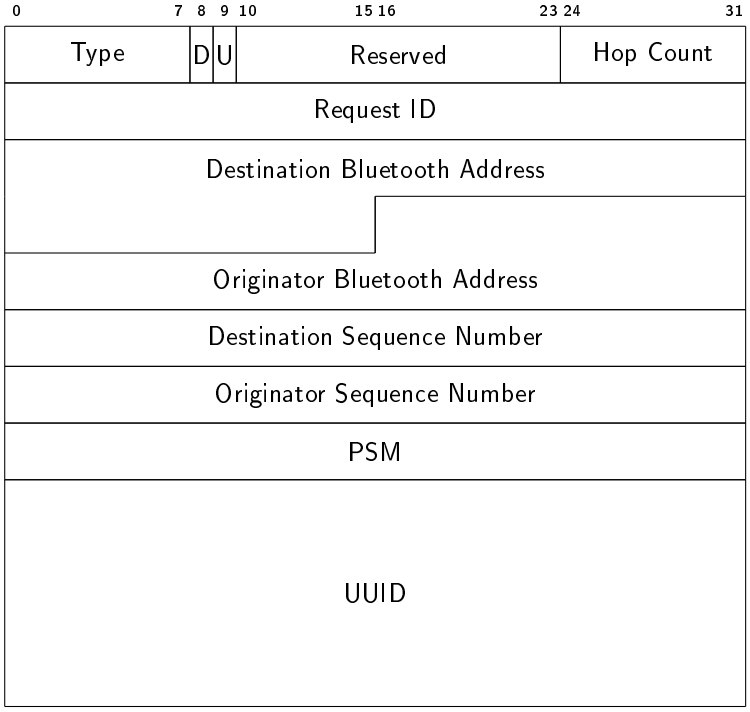

Abbildung 5.1: SNR Connection Request (CREQ) Nachrichtenformat

Hop Count:
8 Bit Zähler. Enthält die Anzahl der Hops von der Quelle des CREQ bis zum aktuell bearbeitenden Knoten.

Request ID:
32 Bit Sequenznummer, die — zusammen mit der Absenderadresse (Originator Bluetooth Address) — die Nachricht eindeutig identifiziert.

Destination Bluetooth Address:
48 Bit Bluetooth Adresse des Zielknotens, falls diese bekannt ist. Andernfalls *muss* das UUID Feld ausgefüllt und das U-Flag gesetzt sein. Siehe Abschnitt 5.4.1.

Originator Bluetooth Address:
48 Bit Bluetooth Adresse des Quellknotens.

Destination Sequenz Number:
32 Bit Sequenznummer. Die neuste bekannte Zielsequenznummer. Sie wird von demselben Knoten ausgefüllt, der auch die Zieladresse einträgt. Ist keine Zielsequenznummer bekannt, so muss das D-Flag gesetzt sein.

Originator Sequence Number:
32 Bit Sequenznummer. Aktuelle Sequenznummer des Quellknotens.

PSM: 32 Bit Protocol Service Multiplexer. Identifiziert zusammen mit der Zieladresse den Dienst zu dem eine Verbindung aufgebaut werden soll. Dieses

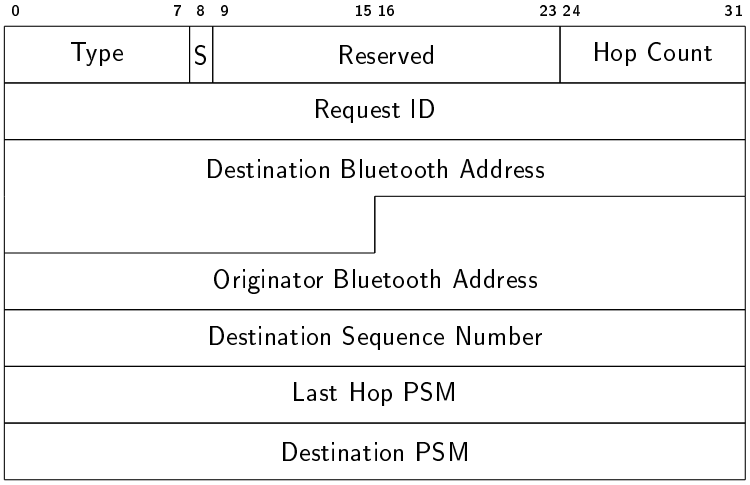

Abbildung 5.2: SNR Connection Reply (CREP) Nachrichtenformat

Feld ist nur ausgefüllt, wenn auch die Bluetoothadresse des Zielknotens angegeben wurde. Andernfalls wird es bei der Verarbeitung ignoriert.

UUID: 128 Bit Universally Unique Identifier. Identifiziert den Dienst zu dem eine Verbindung aufgebaut werden soll. Alternativ zum Tupel (Adresse, PSM). Die UUID kann entweder alleine (Anycast Adressierung) oder gemeinsam mit der Zieladresse angegeben sein. Siehe Abschnitt 5.4.1.

5.3.2 Connection Reply Nachricht (CREP)

Eine CREP-Nachricht wird ausschließlich vom Zielknoten verschickt, d.h. von dem Knoten, der den gesuchten Dienst zur Verfügung stellt. Ist der angeforderte Dienst auf dem Zielknoten verfügbar, so wird mit der Weiterleitung dieser Nachricht der gewünschte Datenpfad zwischen Zielknoten und Quellknoten aufgebaut.

CREP-Nachrichten werden vom Zielknoten ausgehend entlang des — beim Weiterleiten des CREQ aufgebauten — Rückweges bis zum Quellknoten weitervermittelt. Dabei verbindet sich jeder Knoten zunächst mit dem angegebenen Serverdienst des letzten Knotens, startet selbst einen Serverdienst mit beliebigem PSM und leitet die CREP-Nachricht mit aktualisiertem „Last Hop PSM“-Feld weiter. So wird Stück für Stück eine Kette von dedizierten Teilverbindungen aufgebaut. Jeder Zwischenknoten hat später genau zwei einander fest zugeordnete Teilverbindungen, zwischen denen alle Nachrichten einfach weiterzuleiten sind.

Die Nachricht ist in Abbildung 5.2 dargestellt. Die Bedeutung der Felder ist im Folgenden aufgeschlüsselt:

Type: 8 Bit Typfeld. Legt den Nachrichtentyp fest, hat immer den Wert 2.

S: 1 Bit Service-Not-Available-Flag. Ist gesetzt, falls der angeforderte Dienst vom Zielknoten nicht angeboten wird.

Reserved:
Reserviertes, 15 Bit breites Feld. Alle Bits werden beim Senden zu 0 gesetzt und beim Empfang ignoriert.

Hop Count:
8 Bit Zähler. Enthält die Anzahl der Hops von der Quelle des CREQ bis zum Zielknoten.

Request ID:
32 Bit Identifikator des zugehörigen CREQ.

Destination Bluetooth Address:
48 Bit Bluetooth Adresse des Zielknotens der CREQ-Nachricht. Also der Knoten, der den gesuchten Dienst beherbergt.

Originator Bluetooth Address:
48 Bit Bluetooth Adresse des Quellknotens. Der Knoten, der die Verbindungsanfrage gestellt hat.

Destination Sequenz Number:
32 Bit Sequenznummer des Zielknotens.

Last Hop PSM:
32 Bit Protocol Service Multiplexer des Serverdienstes auf dem letzten Hop, der eine Weiterleitung zum gesuchten Dienst bereit stellt.

Destination PSM:
32 Bit Protocol Service Multiplexer des gesuchten Dienstes auf dem Zielknoten. Der Wert wird mit dem aufgebauten Datenkanal gespeichert. Diese Information wird für Local Repair benötigt.

5.3.3 Connection Release (CREL)

Bedingt durch die Möglichkeit des lokalen Reparaturversuchs eines Knotens bei Verbindungsausfall, wird eine Nachricht zum ordentlichen Verbindungsabbau benötigt. Die Nachricht wird entlang des abzubauenden Datenpfades bis zum gegenüberliegenden Kommunikationsendpunkt weitergeleitet. Dabei markiert jeder Zwischenknoten die entsprechende Teilverbindung als aufgelöst. Am anderen Ende des Pfades angekommen, schließt der Knoten die unterlagerte L2CAP Verbindung ordnungsgemäß. Dies wird vom nächsten Hop des Pfades detektiert, der aber aufgrund des zuvor gesetzten Flags zur Auflösung der Verbindung keinen Reparaturversuch ausführt. Stattdessen schließt der Knoten die Verbindung. So propagiert sich der Verbindungsabbau wieder bis zum Initiator des CREL zurück.

Abbildung 5.3 zeigt das Format der CREL-Nachricht. Die Bedeutung der Felder ist im Folgenden aufgeschlüsselt.

Type: 8 Bit Typfeld. Legt den Nachrichtentyp fest, hat immer den Wert 3.

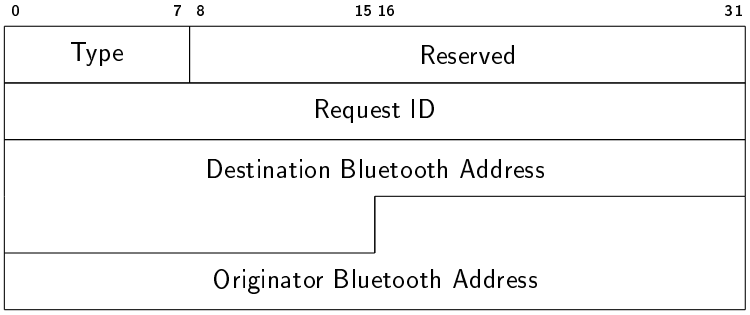

Abbildung 5.3: SNR Connection Release (CREL) Nachrichtenformat

Reserved:
Reserviertes, 24 Bit breites Feld. Alle Bits werden beim Senden zu 0 gesetzt und beim Empfang ignoriert.

Request ID:
32 Bit Identifikator der zugehörigen CREQ-Nachricht aus der Phase des Datenkanalaufbaus.

Destination Bluetooth Address:
48 Bit Bluetooth Adresse des Zielknotens aus der Phase des Datenkanalaufbaus.

Originator Bluetooth Address:
48 Bit Bluetooth Adresse des Quellknotens aus der Phase des Datenkanalaufbaus.

5.3.4 Bestätigungen (ACK)

Aufgrund des unzuverlässigen Dienstes, den Bluetooth zur Verfügung stellt, müssen Signalisierungsnachrichten bestätigt werden. Das bedeutet, jedesmal wenn eine Signalisierungsnachricht übertragen wurde, wird eine passende ACK-Nachricht auf dem selben Datenkanal erwartet. Bleibt die Bestätigung aus, so wird die Neuübertragung der Signalisierungsnachricht angestoßen. Die Bestätigung der Datenpakete funktioniert analog, wobei durch die Trennung von Daten- und Signalisierungsnachrichten die Bestätigung nicht über die selbe Verbindung wie die Daten, sondern über die zugehörige Signalisierungsverbindung übertragen wird.

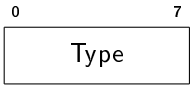

Abbildung 5.4: SNR Acknowledge (ACK) Nachrichtenformat

Abbildung 5.4 zeigt den Aufbau der Acknowledge Nachrichten. Diese bestehen aus einem einzigen Byte, welches die Nachricht als Bestätigungsnachricht identifiziert und gleichzeitig den Typ der bestätigten Nachricht mitteilt. Der Wert des Feldes ergibt sich aus einer bitweisen Oder-Verknüpfung des Type Feldes der zu

Feld	Beschreibung
Destination	Adresse des Zielknotens
Dest. Sequence Number	Sequenznummer des Zielknotens
Dest. Seq. Num. Is Valid	Flag. Gibt an ob die eingetragene Sequenznummer gültig ist.
Hop Count	Distanz zum Zielknoten
Next Hop	Adresse des nächsten Knotens auf dem Pfad zum Ziel
Expiration Time	Zeitpunkt bis zu dem die Route gültig ist.
Is Active	Flag, welches angibt ob die Route aktiv ist.

Tabelle 5.2: Struktur eines Routingtabelleneintrages

bestätigenden Nachricht mit dem Wert 0x10. Dadurch ergibt sich: 0x11 falls der Empfang einer CREQ-Nachricht bestätigt wird, 0x12 falls eine CREP-Nachricht bestätigt wird und 0x13 im Falle einer bestätigten CREL-Nachricht. Zur Bestätigung von Nutzdaten ist der Wert 0x18 reserviert.

5.3.5 Weitere Datenstrukturen

Routingtabelle

Die Routingtabelle ist die zentrale interne Datenstruktur, die von jedem Knoten im Netz gepflegt wird. Sie stellt sämtliche vorhandene Pfadinformation in Form der Abbildung *Zielknoten* $\mapsto$ *nächster Hop* zur Verfügung. Daneben müssen noch weitere Informationen in der Routingtabelle vorgehalten werden. Tabelle 5.2 zeigt alle in der Routingtabelle abgelegten Informationen.

Liste der Signalisierungsverbindungen

Um Signalisierungsnachrichten von Datenpaketen unterscheiden zu können, werden die zugehörigen Verbindungen getrennt voneinander verwaltet. Zwischen beliebigen benachbarten Knoten kann keine oder genau eine Signalisierungsverbindung bestehen. Jede Signalisierungsverbindung wird gemeinsam mit der Information zu welchem Nachbarknoten sie führt und der Gültigkeitsdauer in der Liste abgelegt.

Die Gültigkeitsdauer einer Signalisierungsverbindung verlängert sich immer dann, wenn Informationen über die Verbindung oder einen der zugehörigen Datenkanäle übertragen werden. Ist die Gültigkeitsdauer abgelaufen, so wird die Verbindung geschlossen und der Eintrag aus der Liste entfernt.

Liste der Datenverbindungen

Datenverbindungen treten in Zwischenknoten immer paarweise auf, wobei eine stromaufwärts und eine stromabwärts orientiert ist. Die beiden Verbindungen werden gemeinsam mit den Adressen der stromaufwärts bzw. stromabwärts liegenden Nachbarknoten und dem PSM des verbundenen Dienstes abgelegt. Die Informationen über die Kommunikationsendpunkte sowie den PSM des verbundenen Dienstes werden im Falle eines lokalen Reparaturversuches benötigt. Tabelle 5.3 listet die Einträge im Detail auf.

Feld	Beschreibung
Downstream Connection	stromabwärts orientierte Datenverbindung
Upstream Connection	stromaufwärts orientierte Datenverbindung
Downstream Hop	Adresse des stromabwärts liegenden Nachbarknoten
Upstream Hop	Adresse des stromaufwärts liegenden Nachbarknoten
Upstream Destination	Adresse des stromaufwärts liegenden Kommunikationsendpunktes
Downstream Destination	Adresse des stromabwärts liegenden Kommunikationsendpunktes
Destination Service PSM	PSM des verbundenen Dienstes

Tabelle 5.3: Struktur eines Eintrages in der Liste der Datenverbindungen

Liste bekannter Dienste

Die Liste bekannter Dienste stellt eine Abbildung *UUID* $\mapsto$ *Bluetooth Adresse* dar. Während die UUID den Dienst identifiziert, zeigt die Bluetooth Adresse das Gerät an, welches den Dienst anbietet. Die Liste ist anfänglich leer, Einträge werden mit der Zeit erlernt. Jedesmal wenn ein CREQ mit gesetztem UUID-Search-Flag verarbeitet wird, untersucht der Knoten, ob neben der UUID auch eine Zieladresse im Paket gegeben ist. Ist eine Adresse eingetragen, so kann der Knoten die Abbildung *UUID* $\mapsto$ *Bluetooth Adresse* lernen. Ist keine Zieladresse im Paket eingetragen, so versucht der Knoten die Angaben im Paket durch seine Liste bekannter Dienste zu ergänzen[2].

Liste lokaler Dienste

Die Liste lokaler Dienste ist eine Abbildung der Form *UUID* $\mapsto$ *PSM*. Jeder lokal laufende Dienst meldet sich mit diesen Informationen an. Erreicht ein CREQ den Knoten und die Nachricht ist entweder an den Knoten adressiert oder das UUID-Search-Flag ist gesetzt, ohne dass eine Zieladresse gegeben ist, wird überprüft, ob der gesuchte Dienst lokal verfügbar ist. Ebenfalls kann eine UUID in den aktuellen PSM übersetzt werden, der für die Signalisierung des Datenkanalaufbaus maßgeblich ist.

Liste erreichbarer Nachbarn

Die Liste der erreichbaren Nachbarn puffert die Ergebnisse der regelmäßig ablaufenden Inquiries. Sie repräsentiert die Sicht des Knotens auf das Netzwerk. Anhand dieser Liste kann ein Broadcast emuliert werden.

[2]Sobald neben der UUID auch die Zieladresse eingetragen ist, muss der CREQ nicht mehr geflutet werden, sondern kann unicast auf dem Pfad zum Ziel weitergeleitet werden.

5.4 Protokollbeschreibung

Die folgenden Abschnitte befassen sich mit dem SNR-Protokollablauf. Zunächst wird die Dienstadressierung und die Dienstsuche beschrieben. Anschließend stehen die Wegfindung und der Datenkanalaufbau bzw. -abbau im Vordergrund. Betrachtungen zur Fehlerbehandlung ausgefallener Verbindungen schließen die Protokollbeschreibung ab.

5.4.1 Dienstadressierung

Bluetooth identifiziert Dienste bzw. Dienstmerkmale typischerweise über Universally Unique Identifier (UUID). SNR unterstützt drei unterschiedliche Dienstadressierungsarten, wovon zwei auf UUIDs basieren. Ein Dienst kann adressiert werden durch:

1. eine UUID,
2. eine UUID und eine Geräteadresse oder
3. einen PSM und eine Geräteadresse.

Die erste Variante ist gedacht für den Fall, dass ein Dienst entweder nur ein einziges Mal im Netz vorhanden ist oder — falls der Dienst mehrfach, von unterschiedlichen Geräten angeboten wird — es nicht von Belang ist, welcher Knoten den Dienst erbringt. Es wird eine Verbindung zu dem am schnellsten antwortenden Gerät aufgebaut, welches den angeforderten Dienst anbietet. Diese Adressierung ist mit einem Anycast vergleichbar.

Falls der Dienst von einem bestimmten Gerät erbracht werden soll, kann zusätzlich zur UUID noch die Geräteadresse des gewünschten Zielknotens angegeben werden (Variante 2). Eine Verbindung wird dann ausschließlich zum gesuchten Zielknoten aufgebaut. Sollte dort der Dienst nicht vorhanden sein, wird das in der CREP-Nachricht durch ein gesetztes S-Flag vermerkt und es findet kein Datenkanalaufbau statt.

Die dritte Variante ist für interne Zwecke gedacht und wird im Fall eines lokalen Reparaturversuchs nach Verbindungsausfall (siehe Abschnitt 5.4.7) verwendet. Die Idee dabei ist, dass sich der PSM eines gestarteten Dienstes nicht mehr ändert. Außerdem sind PSM und Geräteadresse die eigentlichen zum Datenkanalaufbau benötigten Informationen. Besteht also die Notwendigkeit, nach einem bereits erfolgreichen Datenkanalaufbau erneut eine Verbindung zum selben Dienst aufzubauen, kann dieser direkt anhand des PSM und der Geräteadresse identifiziert werden.

Diese drei Fälle spiegeln sich im Aufbau der CREQ-Nachricht wieder, die Felder für UUID, PSM und die Bluetooth Adresse des Zielgerätes bereithält. Das U-Flag spezifiziert, ob der Dienst durch die UUID oder durch PSM und Geräteadresse gegeben ist. Die Bits nicht ausgefüllter Felder werden vor der Übertragung zu 0 gesetzt.

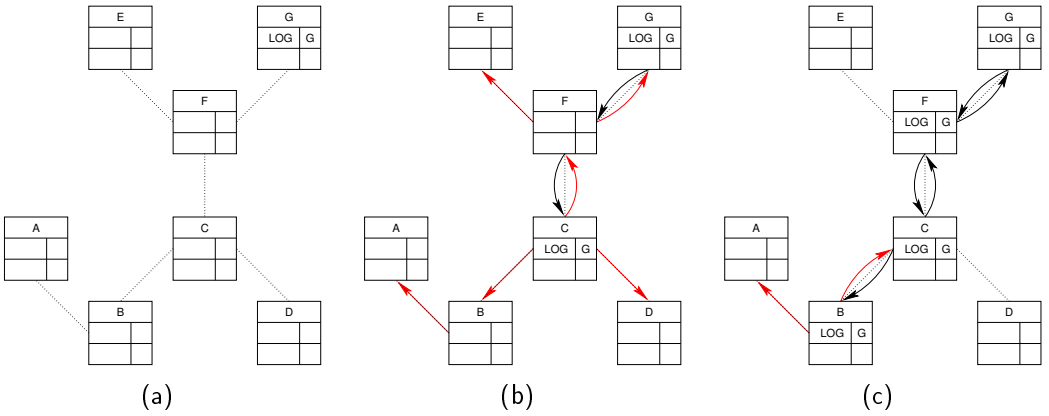

Abbildung 5.5: Ausbreitung der Information über bekannte Dienste (a) Initial kennt jeder Knoten nur die lokal angebotenen Dienste (b) Knoten C sucht den LOG-Dienst. Solange keine weitere Information verfügbar ist, wird die Anfrage geflutet. Anhand der Antwort des Diensterbringers lernt C, wer den Dienst anbietet. (c) Knoten B sucht ebenfalls den LOG-Dienst. C kann die Anfrage auflösen und direkt an den Diensterbringer weiterleiten. Dabei lernt auch der Zwischenknoten F von wem der Dienst angeboten wird. B lernt die Abbildung wieder mit der Antwort des Diensterbringenden Knotens.

Die Weiterleitung einer CREQ-Nachricht unterscheidet sich danach, ob eine Zieladresse gegeben ist oder nicht. Insbesondere sind die oben aufgeführten Varianten 1 und 2 zu unterscheiden. Solange keine Zieladresse im Signalisierungspaket eingetragen ist, muss das Paket in das Netz geflutet werden. Ein CREQ mit Zieladresse kann, falls entsprechende Routinginformationen verfügbar sind, per Unicast zum Ziel weitergeleitet werden, was wesentlich effizienter[3] ist. Steht eine Nachricht mit ausgefüllter Zieladresse und UUID zur Weiterleitung an, trägt der Knoten die Abbildung der beiden Werte aufeinander in seine Liste bekannter Dienste ein (siehe Abschnitt 5.3.5). Ist keine Zieladresse gegeben, versucht der weiterleitende Knoten den zur UUID gehörenden Zielknoten zu ermitteln und diesen in der Nachricht zu ergänzen.

Die Ausbreitung der Information geht folgendermaßen vonstatten: Initial kennt jeder Knoten nur die eigenen Dienste (Abbildung 5.5(a)). Knoten C möchte eine Verbindung zum Dienst „LOG“ aufbauen, der von Knoten G angeboten wird. C flutet eine Anfrage mit der UUID des gesuchten Dienstes. Da Knoten C die Zieladresse noch nicht kennt, bleibt das entsprechende Feld der Nachricht leer. Die Anfrage wird in das Netz geflutet bis sie auf den diensterbringenden Knoten G trifft. Dieser kann die Anfrage auflösen und sendet eine Antwortnachricht an C zurück. In der Antwortnachricht ist C als Quelle und G als Ziel vermerkt[4]. Die Antwortnachricht enthält nicht den Dienstbezeichner. Daher kann erst der Initiator der Anfrage, Knoten C, die Abbildung angefragte UUID zu antwortendem Knoten lernen. Ab diesem Zeitpunkt kennen C und G den Diensterbringer des „LOG“-Dienstes (siehe Abbildung 5.5(b)).

[3] Broadcasts stehen nur emuliert zur Verfügung.
[4] Siehe dazu Abschnitt 5.2 Bezeichnungen.

Möchte der Knoten B eine Verbindung zum selben Dienst aufbauen, so muss dieser zunächst genau wie C, ob fehlender Informationen, die Anfrage fluten. C kann die Anfrage auflösen und durch die Adresse des Diensterbringers ergänzen. Damit kann die Nachricht per Unicast zu G weitergeleitet werden. Da die Nachricht nun sowohl Dienstidentifikator als auch die Adresse des Diensterbringers enthält, kann jeder weiterleitende Knoten die Lokation des Dienstes lernen. G beantwortet die Anfrage mit einer entsprechenden Nachricht, welche es dem Anfragesteller B ermöglicht, die Lokation des gesuchten Dienstes zu lernen (siehe Abbildung 5.5(c)).

Der Aufbau des Datenkanals wird in beiden Fällen parallel zum Weiterleiten der Antwortnachricht vollzogen. Siehe dazu auch Abschnitt 5.4.5.

5.4.2 Sequenznummern

Um die Aktualität der Routinginformationen gegenüber einer Anfrage sicherzustellen, wird das aus AODV bekannte Prinzip der Sequenznummern eingesetzt. Jeder Knoten pflegt einen Zähler, der seine aktuelle Sequenznummer enthält. Beim Senden einer CREQ-Nachricht wird der Zähler erhöht und die neue Sequenznummer als Quellsequenznummer in die Nachricht eingetragen. Leitet ein Knoten die Anfrage weiter, so aktualisiert er daraufhin die im Routingtabelleneintrag zum Quellknoten gespeicherte Sequenznummer und bestätigt damit die Aktualität des bekannten Pfades.

Eine Verbindungsanfrage enthält dabei die größte, dem Quellknoten der Anfrage bekannte Sequenznummer des Zielknotens. Ein Routingtabelleneintrag ist nur dann ausreichend aktuell, wenn die mit dem Eintrag gespeicherte Sequenznummer mindestens so groß ist, wie die Zielsequenznummer der CREQ-Nachricht. Nur in diesem Fall darf der Eintrag zum Weiterleiten der Nachricht herangezogen werden. Anderenfalls wird er ignoriert und die Nachricht in das Netz geflutet.

5.4.3 Wegfindung

Neue Pfade werden durch Fluten von CREQ-Nachrichten in das Netz gefunden. Dabei kann jeder weiterleitende Knoten zwei Routingtabelleneinträge neu anlegen oder ggf. aktualisieren. Darunter fallen der Eintrag, der auf den Quellknoten der Nachricht verweist, sowie der Eintrag zu dem Nachbarknoten, von dem die Nachricht zuletzt weitergeleitet wurde. Abbildung 5.6(a) veranschaulicht diesen Sachverhalt. Ein Routingtabelleneintrag umfasst folgende Informationen: Die Adresse des Knotens auf den der Eintrag verweist, dessen Sequenznummer (falls die Sequenznummer nicht bekannt ist, wird das „destSeqNumIsValid“-Flag gelöscht), die Entfernung in Hops, den nächsten Knoten auf dem Weg dort hin, ein Verfalldatum und ein weiteres Flag, welches anzeigt ob der Eintrag gültig ist. Aktualisiert werden die zur Route gehörende Sequenznummer und die Gültigkeitsdauer. Diese Einträge können in allen drei Fällen der Dienstadressierung angelegt werden. Insbesondere auch für den Fall, dass *keine* Zieladresse angegeben ist (Abschnitt 5.4.1, Variante 1).

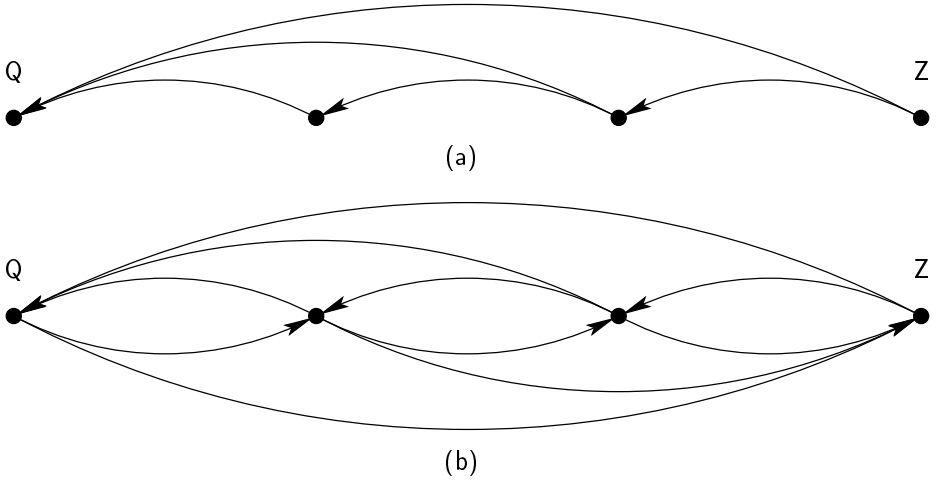

Abbildung 5.6: Routingtabelleneinträge nach Wegfindung. (a) zeigt alle Rückweg-Einträge nachdem die CREQ-Nachricht vom Quellknoten Q bis zum Zielknoten Z weitergeleitet wurde. (b) zeigt die Situation nachdem zusätzlich die CREP-Nachricht von Z nach Q übermittelt wurde.

Existiert in der Routingtabelle schon ein ausreichend aktueller[5] Eintrag zum Zielknoten, so kann die Nachricht unicast, anderenfalls per Broadcast weitergeleitet werden. Während bei AODV schon jeder Zwischenknoten, der über einen ausreichend aktuellen Pfad zum Ziel verfügt, die Anfrage beantworten darf, *muss* bei SNR ein CREQ bis zum Zielknoten durchgereicht werden, da letztendlich ein Datenpfad zu diesem aufgebaut werden muss.

Der Zielknoten Z beantwortet die Anfrage mit einer „Connection Reply"-Nachricht, welche entlang der zuvor angelegten Routingtabelleneinträge zum Quellknoten Q zurück geschickt wird. Dabei kann jeder weiterleitende Knoten wiederum zwei Einträge anlegen oder aktualisieren. Darunter fallen der Eintrag zum Zielknoten Z, sowie der Eintrag zum Nachbarknoten, von dem die Nachricht erhalten wurde.

Sobald der Quellknoten die Antwort auf seine Anfrage empfangen hat, enthält jeder Knoten auf dem Pfad zwischen Quelle und Ziel mindestens vier aktuelle Routingtabelleneinträge.[6] Diese sind die Einträge zu Q und Z, sowie zu den beiden auf dem Pfad liegenden Nachbarknoten. Abbildung 5.6(b) fasst die vorhandenen Routingtabelleneinträge zusammen.

5.4.4 Signalisierungs- und Datenverbindungen

Jeder Knoten unterhält zwei Arten von Verbindungen: Signalisierungsverbindungen und Datenverbindungen. Zu jedem Nachbarn kann maximal eine Signalisierungsverbindung aufgebaut sein. Diese dient zur Übertragung sämtlicher Signalisierungs-

[5] Siehe den Abschnitt über Sequenunummern 5.4.2.

[6] Die zu Z und Q benachbarten Knoten bilden eine Ausnahme. Hier fallen je zwei Einträge zusammen.

nachrichten die entweder an den entsprechenden Nachbarknoten adressiert sind oder über diesen weitergeleitet werden müssen.

Daneben verwaltet ein Knoten Datenverbindungen, die Teil eines multihop Datenkanals sind. Datenkanäle sind reserviert für die Kommunikation von Nutzdaten zwischen Initiator des Kanalaufbaus und dem angeforderten Dienst. Daher erfordert jeder Datenkanal eine Kette von reservierten Datenverbindungen.

5.4.5 Datenkanalaufbau

Der eigentliche Datenkanalaufbau findet vom Zielknoten ausgehend in Richtung Quellknoten der Anfrage statt. Ausgelöst durch den Empfang einer CREP-Nachricht, baut jeder Knoten eine Datenverbindung zu dem Knoten auf, von dem er die Nachricht erhalten hat (Last Hop). Der Dienst zu dem verbunden wird, ist durch das „lastHopPSM"-Feld der CREP-Nachricht gekennzeichnet. Anschließend wird lokal ein neuer Weiterleitungsdienst gestartet und dessen PSM als „lastHopPSM" in die CREP-Nachricht übernommen. Der CREP wird nun weitergeleitet.

Sobald sich der nächste Knoten auf dem Pfad zwischen Quelle und Ziel mit dem Weiterleitungsdienst verbindet, kann das Paar von stromaufwärts gerichteter und stromabwärts gerichteter Verbindung in die Liste der Datenverbindungen aufgenommen werden.

5.4.6 Datenkanalabbau

Um einen ordentlichen Datenkanalabbau von einem Verbindungsverlust unterscheiden zu können, ist die CREL-Nachricht notwendig. Diese kann von beiden Kommunikationsendpunkten ausgesandt werden. Zur Identifikation des abzubauenden Datenkanals trägt die Nachricht die ID des CREQ, der den Kanal aufgebaut hat und die Adresse des CREQ-Quellknotens. Die CREL-Nachricht wird entlang des Datenkanals weitergeleitet. Auf jedem Zwischenknoten wird dabei die Verbindung als aufgelöst markiert. Geschlossen werden die Verbindungen erst, wenn alle Teilverbindungen entsprechend markiert sind. D.h., erreicht die CREL-Nachricht den gegenüberliegenden Kommunikationsendpunkt, so beginnt dieser die zugehörige Datenverbindung zu schließen. Jeder Zwischenknoten, der eine geschlossene Verbindung detektiert, welche gleichzeitig als aufgelöst markiert ist, schließt auch die assoziierte zweite Datenverbindung.

5.4.7 Behandlung eines Verbindungsverlustes

Verlust lokal behandeln

Jeder Knoten kann einen detektierten Verbindungsausfall lokal behandeln. Das bedeutet, der Knoten kann versuchen einen neuen Weg zum Zieldienst zu finden. Ob dies versucht werden soll, kann für jeden Knoten einzeln konfiguriert werden.

Reparable Verbindungsverluste sind solche, die zwischen der Stelle des Verbindungsausfalls und der Client-Seite der Verbindung detektiert werden. Der Datenkanal zwischen Verbindungsausfallstelle und Server-Seite wird einfach abgebaut. In beiden Fällen werden sämtliche Routingtabelleneinträge, die über die ausgefallene Verbindung führen, deaktiviert. Gegenüber dem Löschen der Einträge bietet diese Vorgehensweise den Vorteil, dass mit den Einträgen gespeicherte Sequenznummern erhalten bleiben. Ferner wird der ausgefallene Knoten aus der Liste der sichtbaren Geräte entfernt.

Nach der Detektion eines reparablen Verbindungsverlustes wird versucht, einen neuen Kanal zum Zieldienst aufzubauen. Dazu dient der mit der Verbindung lokal gespeicherte „Destination Service PSM“ und die Geräteadresse des diensterbringenden Knotens (siehe Tabelle 5.3). Aus diesen Informationen kann eine neue CREQ-Nachricht erstellt werden. Das „Destination Sequence Number“-Feld muss dabei unbedingt größer sein als das entsprechende Feld der gerade ausgefallenen Verbindung. Nur so kann sicher gestellt werden, dass ein *neuer* Pfad zum Ziel gesucht wird und nicht ein anderer Knoten die Nachricht aufgrund veralteter Informationen in der Routingtabelle in die falsche Richtung weiterleitet.

Sobald eine positive Antwort auf die Anfrage empfangen wurde, kann eine Datenverbindung zum letzten Sender der Antwort aufgebaut werden. Damit ist der Datenkanal erfolgreich repariert. Bleibt die positive Antwort aus, wird auch die stromaufwärts weisende Datenverbindung geschlossen und damit dem Nachbarknoten der Ausfall signalisiert.

Verlust Signalisieren

Soll ein Verbindungsverlust *nicht* lokal behandelt werden, geht der Knoten genau wie oben beschrieben vor. Allerdings wird kein neuer Verbindungsaufbau initiiert, sondern direkt so verfahren, als ob keine positive Antwort empfangen worden wäre.

5.5 Veränderungen gegenüber RFC3561

Folgende wichtige Veränderungen ergeben sich gegenüber dem Protokollablauf und den Datenstrukturen des in [PeBR03] definierten AODV-Protokolls:

- SNR arbeitet verbindungsorientiert.
- Die AODV-Nachricht zur Pfadanfrage (RREQ) wurde um notwendige Informationen zur Dienstsuche und zum Verbindungsaufbau erweitert.
- Ebenfalls wurde die AODV-Antwort-Nachricht (RREP) erweitert. Das SNR-Pendant trägt mit der ID der zugehörigen Anfrage und den beiden PSM-Feldern zusätzliche Informationen zum Verbindungsaufbau bzw. für den lokalen Reparaturversuch bei Verbindungsausfall.

- Aufgrund der Verbindungsorientiertheit von SNR kann die AODV-Route-Error-Nachricht (RERR) entfallen. Im selben Zug kann auf die AODV-Precursor-Listen verzichtet werden, sodass die Routingtabelleneinträge eine feste Größe bekommen. Dies vereinfacht die Speicherverwaltung.

- Im Gegenzug ist eine zusätzliche Nachricht zum ordentlichen Abbau der Datenkanäle notwendig geworden, um Kanalabbau und Verbindungsausfall unterscheiden zu können.

- Die Bestätigungsnachricht wurde auf alle Signalisierungsnachrichten ausgeweitet. Die Notwendigkeit der Nachricht ergibt sich hier allerdings aus einem anderen Grund. AODV benötigt die Bestätigung, um bidirektionale Verbindungen sicherzustellen. Bluetooth bietet ausschließlich bidirektionale Verbindungen. Die SNR-Bestätigung sichert das Protokoll gegen Paketverluste auf unzuverlässigen Signalisierungsverbindungen ab.

- SNR integriert einen einfachen Mechanismus zur Topologiekontrolle.

- SNR behandelt neben der Wegfindung alle notwendigen Schritte zum Verbindungsaufbau und -abbau.

- SNR enthält einen integrierten Mechanismus zur Dienstsuche.

Kapitel 6

Implementierung

6.1 Voraussetzungen

Die Implementierung des im vorhergehenden Kapitel vorgestellten Entwurfs wurde in Java vorgenommen. Der Zugriff auf Bluetooth wurde durch die JSR-82 Implementierung avetanaBluetooth[1] realisiert, womit die geforderte Plattformunabhängigkeit sichergestellt ist.

Abbildung 6.1 zeigt einen vollständigen Überblick über das System, von der Luftschnittstelle bis zur Applikation. Die Radio- und Baseband-Schicht, sowie L2CAP, RFCOMM und SDP sind bereits aus Abschnitt 2.3 im Grundlagenkapitel bekannt. Darüber hinaus ist die Aufteilung in Bluetooth Adapter und Softwarestack angedeutet, welche in diesem Fall über den USB-Bus miteinander verbunden sind. Der Bluetooth Adapter ermöglicht die Kommunikation von Gerät zu Gerät, der Softwarestack den Zugriff auf die Hardware. Darüber befindet sich JSR-82, was den betriebssystemübergreifend einheitlichen Zugriff auf den Bluetooth Stack erlaubt. Darauf setzt die SNR-Implementierung auf. Eine Applikation kann über die SNR-Schnittstelle Multihop-Verbindungen aufbauen.

6.2 SNR Architektur

Dieser Abschnitt soll einen kurzen Überblick über die SNR-Architektur geben. Auf die wichtigsten Datenstrukturen wurde bereits im Abschnitt 5.3.5 eingegangen. Daher soll hier der Fokus auf den Informationsfluss und die informationsverarbeitenden Threads gelegt werden. Abbildung 6.2 zeigt die SNR-Architektur. Threads sind durch fette Einrahmung kenntlich gemacht, Datenstrukturen durch gepunktete Einrahmungen. Die Aufgaben der einzelnen Threads sollen im Folgenden kurz umrissen werden:

Device Discoverer: Der „Device Discoverer“ hält die Liste der erreichbaren Geräte aktuell. In regelmäßigen Abständen führt er Inquiries aus. Alle neu gefundenen Nachbarn, werden in die Liste der erreichbaren Geräte („Reachable

[1] http://sourceforge.net/projects/avetanabt/

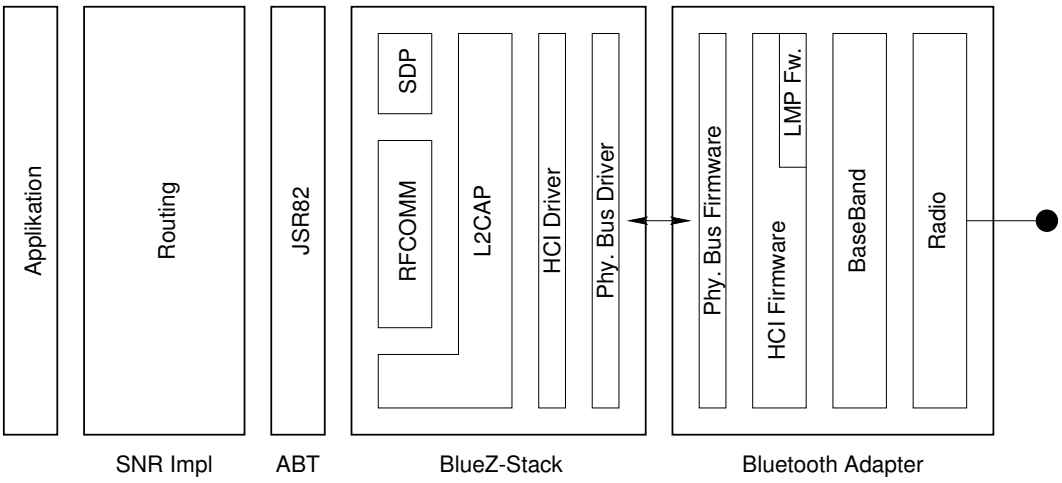

Abbildung 6.1: Systemübersicht

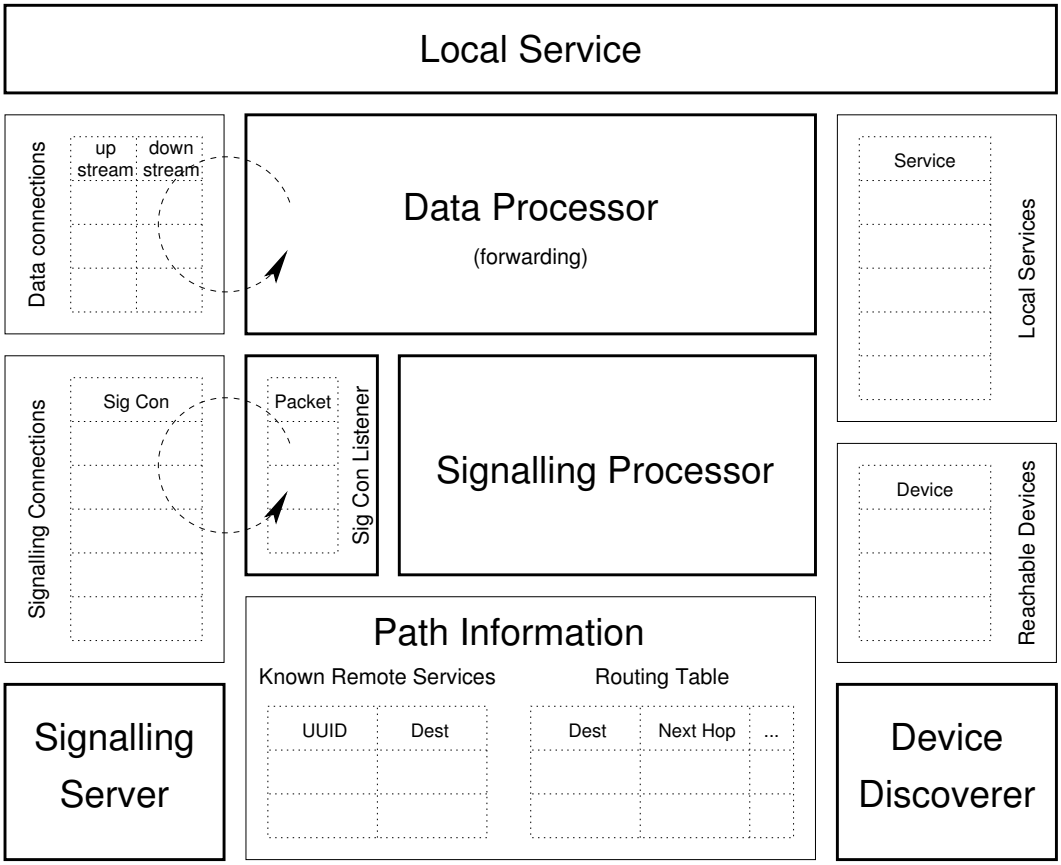

Abbildung 6.2: SNR Architektur

Devices“) aufgenommen, alte Einträge werden gelöscht. Die Liste ist Basis für das Fluten der Signalisierungsnachrichten.

Signalling Server: Der „Signalling Server“ stellt die Gegenstelle für den Aufbau der Signalisierungsverbindungen dar. Jeder Knoten bietet diesen Dienst an und kann selbst zum „Signalling Server“ anderer Knoten eine Verbindung aufbauen. Jede aufgebaute Signalisierungsverbindung wird in der Liste der Signalisierungsverbindungen („Signalling Conns“) abgelegt.

SigConListener: Diese Komponente überwacht alle aufgebauten Signalisierungsverbindungen. Diese werden regelmäßig auf neu angekommene Signalisierungsnachrichten überprüft. Sobald eine neue Signalisierungsnachricht eingetroffen ist, wird sie zwischengespeichert und dem „Signalling Processor“ zur weiteren Verarbeitung übergeben.

Signalling Processor: Der „Signalling Processor“ stellt die wichtigste Komponente für die Verarbeitung der Signalisierungs-Nachrichten dar. Hier werden die einzelnen Nachrichten ausgewertet und ggf. weitergeleitet bzw. beantwortet. Der „Signalling Processor“ pflegt außerdem die Pfadinformationen, bestehend aus Routing Tabelle und die Liste der bekannten entfernten Dienste („Known Remote Services“). Darüber hinaus wird von hier aus der Datenverbindungsaufbau initiiert und Tupel aus stromaufwärts und stromabwärts gerichteten Teilverbindungen in der Liste der Datenverbindungen („Data Conns“) abgelegt.

Data Processor: Der „Data Processor“ realisiert die Weiterleitung der Daten zwischen den einander zugeordneten Datenverbindungen. Diese werden regelmäßig auf neu angekommene Daten überprüft und diese ggf. über die zugeordnete Verbindung weitergegeben.

Local Service: Der „Local Service“ steht stellvertretend für alle höheren Server-Dienste, welche auf SNR aufbauen und bereit für einen clientseitigen Verbindungsaufbau sind. Sie melden sich beim Starten als lokaler Dienst an und werden daraufhin in die Liste der lokalen Dienste aufgenommen.

6.3 Broadcast

Bluetooth stellt einen sogenannten „Active Member Broadcast“ bereit. Damit kann ein Master Nachrichten an alle bereits verbundenen Slave-Knoten übermitteln, also sämtliche Knoten in seinem Piconetz erreichen. Slaves können auf diese Weise jedoch *nicht* mit ihrem Master kommunizieren.

CREQ-Signalisierungsnachrichten müssen bei der Wegfindung in das Netz geflutet werden. Jeder Knoten, ob Master oder Slave, muss die Nachricht an *alle* direkt erreichbaren Knoten weitergeben. Dies kann der von Bluetooth angebotene „Active Member Broadcast“ nicht leisten. Daher wurde das Fluten der CREQ-Nachrichten iterativ gelöst. Für jeden erreichbaren Nachbarn wird, falls dies nicht

bereits geschehen ist, eine Signalisierungsverbindung aufgebaut und die CREQ-Nachricht unicast übertragen. Diese Lösung soll im folgenden als „iterativer Broadcast“ bezeichnet werden. Der iterative Broadcast entspricht in guter Näherung einem „echten“ Broadcast. Ein wichtiger Unterschied besteht jedoch darin, dass die Nachrichten an die unterschiedlichen Nachbarknoten nacheinander und damit verzögert übertragen werden. Dadurch geht die Eigenschaft der AODV-Wegfindung verloren immer den kürzesten Pfad zum Ziel zu finden.

Zwei Optimierungen lassen sich an dieser Strategie vornehmen. Da beide Optimierungen zu einem veränderten Verhalten gegenüber einem echten Broadcast führen, lassen sie sich jeweils über einen Parameter deaktivieren. Im Folgenden soll der Knoten, von dem eine Nachricht erhalten wurde, als „Last Hop“ bezeichnet werden.

1. Beim iterativen Broadcast einer Nachricht kann die Übermittlung derselben an den Last Hop Knoten entfallen, da dieser die Nachricht schon erhalten hat.

2. Falls der Zielknoten in der Liste direkt erreichbarer Nachbarn enthalten ist, kann die Nachricht unicast übertragen werden.

Darüber hinaus wird eine Nachricht zuerst an die Nachbarn übertragen, zu denen bereits eine Signalisierungsverbindung besteht. Erst anschließend beginnt der Knoten mit der Übertragung an die übrigen Nachbarn. Diese Vorgehensweise führt dazu, dass bevorzugt der Pfad gefunden wird, über den der Verbindungsaufbau am schnellsten vollzogen wird.

Jeder Master kann maximal acht Slave-Knoten in sein Piconetz einladen. Damit ist die Situation denkbar, dass ein ungünstig gelegener Knoten von der Teilnahme am gesamten Scatternetz ausgeschlossen ist. Mit Sicherheit vermeidbar ist dieses Problem selbst durch eine aufwändige Topologiekontrolle nicht. Der einfache Ansatz, die Ausprägung der Verbindungen zu randomisieren, kann zumindest sicher stellen, dass ein Knoten nicht unendlich lange von der Netzteilnahme ausgeschlossen ist. Die Topologieausprägung geht mit dem Broadcast einher, da zu diesem Zeitpunkt neue Verbindungen zwischen Nachbarn aufgebaut werden. Daher wurde die Reihenfolge des Verbindungsaufbaus während des Broadcasts randomisiert, um eine ständige ungünstige Topologie weniger wahrscheinlich werden zu lassen.

6.4 Parameter

Wie im Abschnitt 6.3 bereits angedeutet, ergaben sich bei der Implementierung eine Reihe von Parametern, die das Laufzeitverhalten beeinflussen. Diese sollen im Folgenden kurz erläutert werden.

randomBroadcast: Randomisiert die Reihenfolge der beim Fluten angesprochenen Nachbargeräte. Der Parameter soll die Wahrscheinlichkeit langfristig anhaltender, ungünstiger Verbindungssituationen verringern.

preferOpenSigCon: Bevor zum Fluten einer Signalisierungsnachricht neue Verbindungen aufgebaut werden, überträgt der Knoten die Nachricht zuerst über bereits bestehende Signalisierungsverbindungen. Erst anschließend werden neue Verbindungen zur Übertragung aufgebaut. Dieser Parameter beschleunigt das Fluten in bereits verbundenen Teilen des Netzes.

intelligentBroadcast: verhindert das zurücksenden einer Signalisierungsnachricht an den letzten Knoten, der sie weitergeleitet hat (Last Hop). Kreisende Signalisierungsnachrichten werden nicht durch diesen Parameter, sondern durch die Behandlung von Duplikaten verhindert.

allowUnicastBroadcast: erlaubt einem Knoten eine zu flutende Signalisierungsnachricht unicast an einen Nachbarn weiterzuleiten, falls dieser Adressat der Nachricht ist. Dadurch kann im direkten Umfeld des Zielknotens ein Broadcast vermieden werden.

alterMasterSlave: gibt eine alternierende Reihenfolge von Master- und Slave-Knoten entlang eines neu aufzubauenden Pfades vor. Bereits bestehende Verbindungen werden nicht beeinflusst. Der Parameter vermindert die Anzahl der Knoten mit Master/Slave-Doppelrolle.

doLocalRepair: Zur Behandlung eines Verbindungsausfalls wurden in Abschnitt 5.4.7 zwei Möglichkeiten vorgestellt. Ist der Parameter aktiviert, so führt ein Knoten nach der Detektion eines Verbindungsausfalls eine Reparatur der betroffenen Datenpfade aus. Ist der Parameter deaktiviert, so baut der Knoten die entsprechenden Teilverbindungen ab. Der Teilverbindungsabbau propagiert sich bis zum Initiator der Verbindung, welcher entscheidet, ob die Verbindung repariert werden soll.

acknowledgeSignallingMessages: legt fest, ob eine Eingangsbestätigung für Signalisierungsverbindungen vom Empfänger der Nachricht gefordert wird. Bleibt die Bestätigung aus, wird die Nachricht bis zu „max_sending_attempts“ Mal neu übertragen. Da die verwendeten L2CAP Verbindungen unzuverlässig sind, kann es ohne diese Bestätigungen zum Verlust von Signalisierungsnachrichten kommen.

acknowledgeData: legt fest, ob der Empfang jedes Datenpakets vom nächsten Zwischenknoten zu bestätigen ist. Falls die Bestätigung ausbleibt wird das Paket bis zu „max_sending_attempts“ Mal neu übertragen.

max_sending_attempts: legt die Anzahl der versuchten Neuübertragungen nach Ausbleiben einer Bestätigung fest.

Kapitel 7

Evaluierung

7.1 Umgebung

Die Evaluierung der Implementierung wurde an einem PC vorgenommen. Bis zu Acht Bluetooth Adapter konnten gleichzeitig über zwei USB-Hubs angesprochen werden. Für jeden Adapter wurde jeweils eine separate Instanz der SNR-Implementierung gestartet. Da sich alle Bluetooth Geräte somit in gegenseitiger Funkreichweite befanden, musste durch die Software auf die Sichtbarkeit der Geräte so Einfluss genommen werden, dass eine beliebige Struktur vorgegeben werden konnte. Die Einhaltung dieser Vorgaben wurde mit `hcidump` überwacht.

Es kamen 8 Bluetooth Adapter der Leistungsklasse[1] 1 zum Einsatz. Da sich diese Adapter mit den verfügbaren Mitteln nicht gegeneinander abschirmen ließen, wurden zur Überprüfung der Fehlerbehandlungsroutinen vier schwächere Adapter der Leistungsklasse 2 eingesetzt. Alle Adapter wurden über die USB-Schnittstelle mit dem zur Evaluierung eingesetzten Rechner verbunden. Tabelle 7.1 gibt einen Überblick über die Adapter.

Die Evaluierung wurde auf einem Intel Celeron M Prozessor mit einer Taktfrequenz von 1500 MHz und 768 MByte Arbeitsspeicher durchgeführt. Die Implementierung wurde mit dem SDK-1.5 von Sun übersetzt und ausgeführt. Als Bluetooth Protokollstack kam der im Linux Kernel Version 2.6.12 integrierte BlueZ Stack zum Einsatz. Den Bluetooth Zugriff von Java ermöglichte die JSR-82 Implementierung avetanaBluetooth[2] (Version 1.3.6, libbluetooth1 Version 2.20).

Zur Evaluation wurden vier Applikationen erstellt. Ein Echo-Service, der als Server gestartet wurde, reflektierte ankommende Datenpakete. Die zugehörige Client-Applikation baute eine Verbindung zum Echo-Service auf und übermittelte einzelne Datenpakete. Dabei ist anzumerken, dass nach dem Versenden eines Datenpakets zuerst die Antwort des Echo-Services abgewartet wurde, bevor das

[1] [Blue04] teilt Bluetooth Adapter nach der maximalen Sendeleistung in drei Leistungsklassen („Power Class“) ein. Für Geräte der Klasse 1 ist eine maximale Sendeleistung von 100 mW vorgeschrieben, für Geräte der Klasse 2 maximal 2.5 mW und für Geräte der Klasse 3 gilt ein Maximum von 1 mW.

[2] http://sourceforge.net/projects/avetanabt/

Hersteller	Geräte Adresse	Version	Klasse	Chipsatz
EPoX, BT-DG05A	00:04:61:84:4B:F0	1.1	1	Cambridge Silicon Radio
EPoX, BT-DG05A	00:04:61:84:4B:F1	1.1	1	Cambridge Silicon Radio
EPoX, BT-DG05A	00:04:61:84:4B:E9	1.1	1	Cambridge Silicon Radio
EPoX, BT-DG05A	00:04:61:84:4B:9B	1.1	1	Cambridge Silicon Radio
Delock, (unbekannt)	00:09:DD:10:68:19	1.1	1	Cambridge Silicon Radio
Delock, (unbekannt)	00:09:DD:10:68:F5	1.1	1	Cambridge Silicon Radio
Delock, M II-791	00:11:F6:04:16:A1	1.2	1	Silicon Wave
Delock, M II-791	00:11:F6:04:F5:85	1.2	1	Silicon Wave
BlueExpert, (unbek.)	00:11:B1:07:A3:4E	1.2	2	Cambridge Silicon Radio
BlueExpert, (unbek.)	00:11:B1:07:A3:5E	1.2	2	Cambridge Silicon Radio
BlueExpert, (unbek.)	00:11:B1:07:A3:68	1.2	2	Cambridge Silicon Radio
BlueExpert, (unbek.)	00:11:B1:07:A3:25	1.2	2	Cambridge Silicon Radio

Tabelle 7.1: Eingesetzte Bluetooth Adapter

nächste Datenpaket übermittelt wurde. Desweiteren kam ein reiner Empfänger-Service („Receiver-Service“) für Durchsatzmessungen zum Einsatz. Dieser Service registrierte ankommende Datenpakete und wertete diese statistisch aus. Es wurden keine Antwort-Nachrichten erzeugt. Der zugehörige Client („Sender-Client“) erfüllte die Aufgabe, einen Strom von Datenpaketen mit maximaler Größe an den Receiver-Service zu versenden.

Wie in Abschnitt 5.3.5 erwähnt, pflegt jedes Gerät eine Liste erreichbarer Nachbarn. Diese Liste spiegelt potentielle Verbindungen zwischen Geräten wieder und wird durch regelmäßige Inquiries aktuell gehalten. Da sich bei der Evaluierung sämtliche Bluetooth Adapter in gegenseitiger Funkreichweite befanden, würden bei jedem Inquiry alle im Test befindlichen Geräte gefunden werden. Daher wurde die Liste erreichbarer Geräte stattdessen regelmäßig aus einer Datei ausgelesen, was eine Vorgabe der potentiellen Topologie ermöglichte. Innerhalb dieser Vorgabe mussten die einzelnen Geräte nach wie vor selbst entscheiden, zu welchen Kommunikationspartnern eine Verbindung aufgebaut und unterhalten wurde.

Durch das zum BlueZ-Stack gehörende Werkzeug `hcidump` konnte sämtliche Kommunikation auf HCI-Ebene überwacht werden. Dadurch konnte sichergestellt werden, dass die Kommunikation nur in der vorgegebenen Weise vonstatten ging.

7.2 Durchsatz

Zur Messung des Durchsatzes wurden die Bluetooth Adapter logisch in einer Kette angeordnet. Dabei wurden Ketten von 2 – 8 Geräten betrachtet. Übertragen wurde ein Strom von 1000 Paketen maximaler Größe (672 Byte pro Paket). Die Übertragung fand unidirektional vom Client zum Server statt. Für jeden Bluetooth Adapter wurde eine separate SNR-Instanz gestartet.

7.2.1 Unzuverlässige Datenübertragung

Zunächst wurde eine Messung zur Überprüfung der Notwendigkeit von Bestätigungsnachrichten für Datenpakete durchgeführt. Dabei wurden im oben beschrie-

	Pakete			Bytes		
	versandt	empfangen	verloren	versandt	empfangen	verloren
1 Hop	1000	1000	0	672000	672000	0
2 Hop	1000	85	915	672000	57120	614880
3 Hop	1000	40	960	672000	26880	645120
4 Hop	1000	25	975	672000	16800	655200
5 Hop	1000	25	975	672000	16800	655200
6 Hop	1000	37	963	672000	24864	647136
7 Hop	1000	37	963	672000	24864	647136

Tabelle 7.2: Paket- und Byteverluste bei der Übertragung von 1000 Paketen zu je 672 Byte über eine unzuverlässige Verbindung.

	Ø [kbit/sec]	σ	Paketverluste	Bytefehler
1 Hop	51,52	0,97	0	0
2 Hop	15,95	0,10	0	0
3 Hop	14,77	0,62	0	0
4 Hop	14,62	0,66	0	0
5 Hop	13,59	0,41	0	0
6 Hop	14,03	1,05	0	0
7 Hop	14,32	0,22	0	0

Tabelle 7.3: Mittlerer Durchsatz in kbit/sec. Angegeben ist der Mittelwert über drei Messungen zu je 1000 Datenpaketen und die Standardabweichung σ.

benen Szenario Daten übertragen, wobei die Charakteristik des darunterliegenden L2CAP-Dienstes nicht verändert wurde. D.h., der jeweilige Sender erwartete keine Bestätigungsnachricht über den Erhalt eines versendeten Datenpaketes. Demzufolge wurden Paketverluste nicht detektiert und keine Neuübertragungen veranlaßt.

Tabelle 7.2 zeigt die Messergebnisse. Zu sehen ist, dass ab einer Länge von 2 Hops massive Paketverluste auftreten. Über 90% der versandten Datenpakete erreichen nicht das Ziel. Die registrierten Byteverluste, welche ebenfalls der Tabelle zu entnehmen sind, entsprechen genau den verlorenen Paketen. Fehler innerhalb der Pakete konnten nicht festgestellt werden.

Die hohe Anzahl der Paketverluste lässt sich durch die notwendigen Piconetzwechsel der weiterleitenden Knoten erklären. Werden Daten an einen vom Piconetz „abwesenden“ Knoten weitergeleitet, gehen diese verloren. Aufgrund des fehlenden Bestätigungsmechanismusses, kann der Verlust nicht detektiert werden.

7.2.2 Zuverlässige Datenübertragung

Nach Aktivierung der Bestätigungen für Datenpakete wurden weitere Messungen durchgeführt. Für jedes Datenpaket wurde eine Empfangsbestätigung des Empfängers erwartet. Blieb diese aus, wurde bis zu fünf Mal die Neuübertragung des selben Paketes angestoßen. Die maximale Wartezeit auf eine Bestätigung betrug 1 Sekunde.

Tabelle 7.3 gibt die Messergebnisse wieder. Durch die Empfangsbestätigungen für Datenpakete konnten die Paketverluste vollständig vermieden werden. Ebenfalls traten keine Verluste von einzelnen Bytes auf. Daneben wurde der Durch-

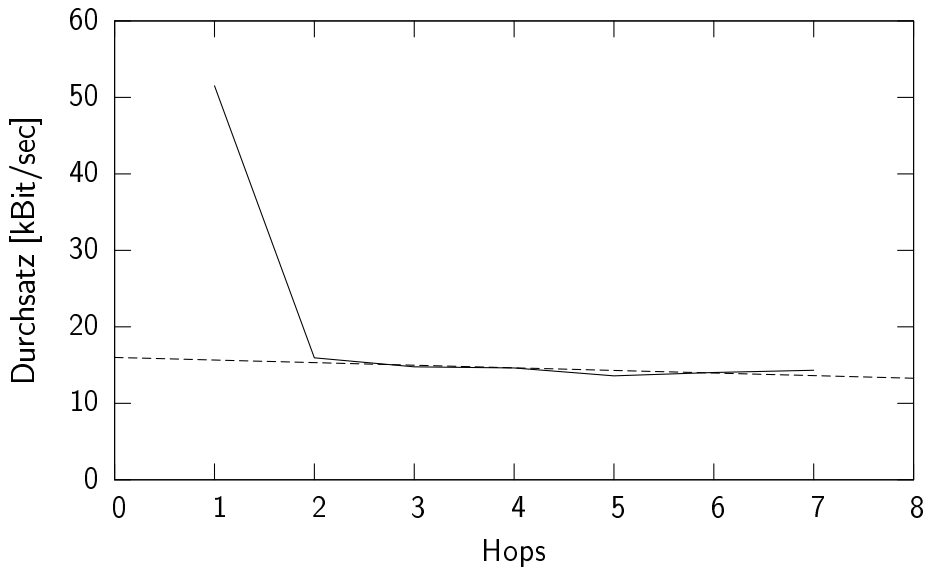

Abbildung 7.1: Mittlerer Durchsatz in kbit/sec. Aufgetragen sind die Mittelwerte über drei Messungen zu je 1000 Datenpaketen. Die eingezeichnete Gerade soll die Tendenz im Multihop-Fall verdeutlichen. Sie hat ein Gefälle von 0,34 kbit/sec pro Hop.

satz in kbit pro Sekunde gemessen. Der Durchsatz einer Multihop-Verbindung fällt im Vergleich zu einer Singlehop-Verbindung auf etwa 30% ab. Mit zunehmender Multihop-Verbindungslänge verändert sich der Datendurchsatz nur noch geringfügig. Die in Abbildung 7.1 eingezeichnete Gerade hat ein Gefälle von 0,34 kbit/sec pro Hop. Sie soll die Tendenz der Durchsatzentwicklung für Multihop-Verbindungen andeuten.

Diese und die vorangegangenen Messergebnisse zeigen deutlich, dass Bestätigungsnachrichten für die Übertragung von Datenströmen unerlässlich sind. Darüber hinaus bleibt zu untersuchen, ob eine Anpassung des Sendeverhaltens („Traffic Shaping“) oder ein kurzfristiges Sammeln und Zwischenspeichern der Datenpakete auf einem Zwischenknoten vor jedem Piconetzwechsel den Durchsatz weiter erhöhen können.

7.3 Paketumlaufzeit

Zur Messung der Paketumlaufzeit (*engl.* Round Trip Time, RTT) wurden die Adapter in einer Kette von zwei bis acht Geräten angeordnet. Bis auf die beiden Endgeräte der Kette konnte sich jeder Bluetooth Adapter mit genau zwei Nachbarn verbinden.

Am einen Ende der Kette sorgte ein als Server gestarteter Echo-Service dafür, dass auf einem Datenkanal ankommende Nachrichten sofort nach Erhalt über den selben Datenkanal an den Absender zurückgeschickt wurden. Den zweiten Kommunikationsendpunkt stellte ein Client dar, der Nachrichten an den Echo-Server

	Ø [ms]	σ	pro Hop
1 Hop	102	12	102
2 Hop	261	53	131
3 Hop	430	72	143
4 Hop	600	95	150
5 Hop	719	109	144
6 Hop	858	114	143
7 Hop	1046	114	149

Tabelle 7.4: Paketumlaufzeit (RTT) über eine unzuverlässige Datenverbindung. Angegeben ist der Mittelwert über 1000 Messungen, die Standardabweichung σ und die mittlere Paketumlaufzeit pro Hop.

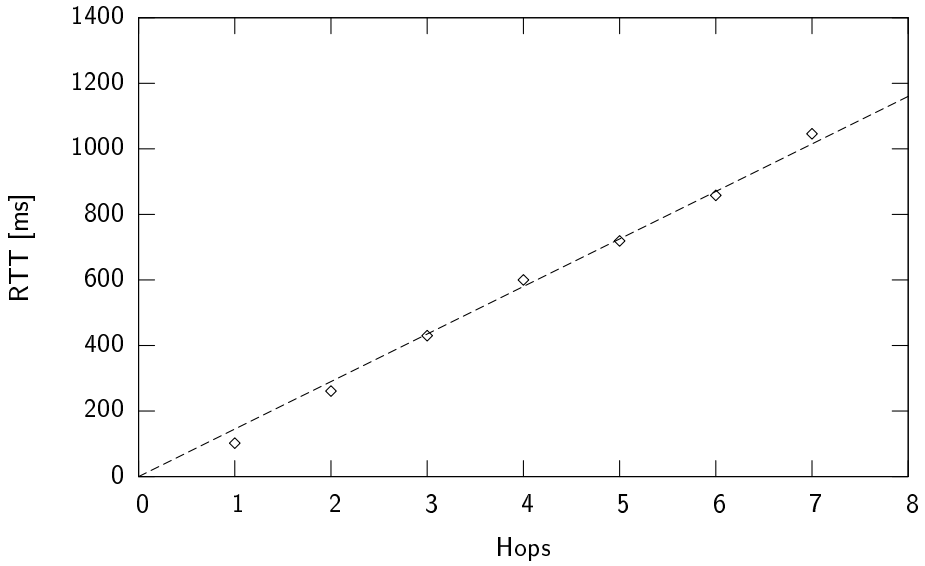

Abbildung 7.2: Paketumlaufzeit (RTT) über eine unzuverlässige Datenverbindung. Angegeben ist der Mittelwert über 1000 Messungen. Desweiteren ist eine Fitting-Gerade mit der Annahme einer mittleren Paketumlaufzeit von 145 ms pro Hop eingetragen.

sendete und die verstreichende Zeit bis zum Empfang der Antwort maß. Erst nach Erhalt der Antwort wurde eine neue Nachricht ausgesandt.

Die Paketumlaufzeit wurde für zwei unterschiedliche Fälle untersucht. Zum einen wurde die Paketumlaufzeit über eine unzuverlässige L2CAP Verbindung betrachtet, zum anderen über eine mittels Bestätigungsnachrichten gegen Paketverluste gesicherte L2CAP Verbindung.

7.3.1 unzuverlässige Datenverbindung

Tabelle 7.4 gibt die Messergebnisse im Detail wieder. Neben dem Mittelwert über 1000 Messungen ist die Standardabweichung und die mittlere Umlaufzeit pro Hop angegeben. Abbildung 7.2 fasst die Messergebnisse zusammen. Nach rechts ist die Entfernung in Hops aufgetragen, nach oben die über 1000 Messungen gemittelte Paketumlaufzeit in Millisekunden. Desweiteren ist eine Fitting-Gerade mit der Annahme einer mittleren Paketumlaufzeit von 145 ms pro Hop eingetragen. Es ist zu

	Ø [ms]	σ	pro Hop
1 Hop	246	50	246
2 Hop	370	52	185
3 Hop	521	62	174
4 Hop	707	90	177
5 Hop	793	111	159
6 Hop	932	128	155
7 Hop	1265	132	181

Tabelle 7.5: Paketumlaufzeit (RTT) über eine zuverlässige Datenverbindung. Angegeben ist der Mittelwert über 1000 Messungen, die Standardabweichung σ und die mittlere Paketumlaufzeit pro Hop.

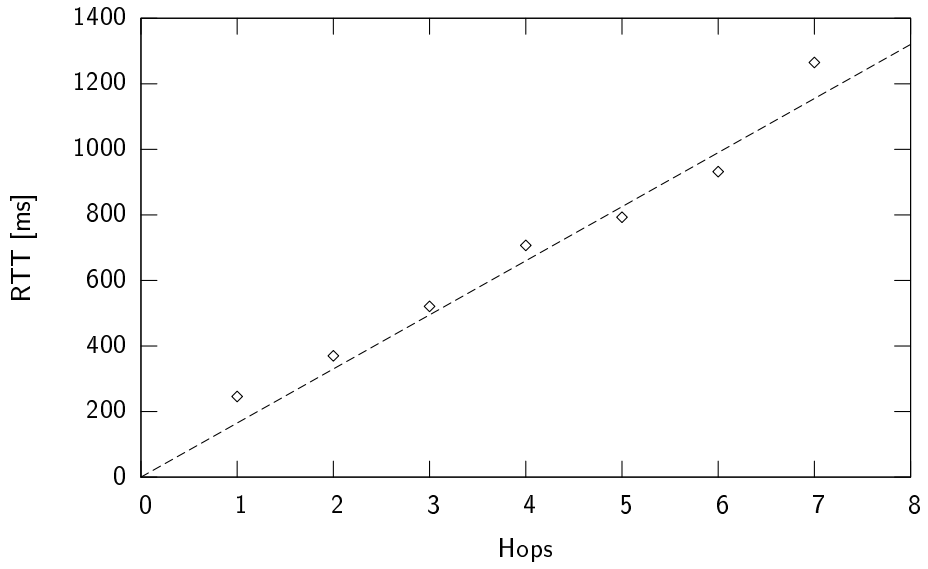

Abbildung 7.3: Paketumlaufzeit (RTT) über eine zuverlässige Datenverbindung. Angegeben ist der Mittelwert über 1000 Messungen. Desweiteren ist eine Fitting-Gerade mit der Annahme einer mittleren Paketumlaufzeit von 165 ms pro Hop eingetragen.

sehen, dass im untersuchten Bereich von 1 – 7 Hops die Annahme einer linearen Entwicklung der Paketumlaufzeit bestätigt wird.

7.3.2 zuverlässige Datenverbindung

Die selbe Messung wurde ebenfalls für eine durch Bestätigungsnachrichten abgesicherte Datenverbindung durchgeführt. Erwartungsgemäß vergrößert sich die Paketumlaufzeit. Tabelle 7.5 zeigt die über 1000 Paketumläufe gemittelten Messergebnisse. Daneben sind die Standardabweichung σ und die mittlere Paketumlaufzeit pro Hop angegeben. Abbildung 7.3 veranschaulicht die Ergebnisse graphisch. Zusätzlich ist eine Fitting-Gerade mit einer Steigung von 165 ms/Hop eingetragen.

Ein Vergleich der Paketumlaufzeiten zwischen zuverlässiger und unzuverlässiger Verbindung ergibt, dass die Bestätigungsnachrichten eine zusätzliche Verzögerung von etwa 20 ms pro Hop einführen.

7.4 Latenz beim Verbindungsaufbau

Zwei Grenzfälle sind in Bezug auf die Latenz beim Verbindungsaufbau von besonderem Interesse. Diese sind:

1. die Latenz eines Verbindungsaufbaus zu einem Knoten, für den im gesamten Netz noch keine Pfadinformationen vorhanden sind. Dies wäre zum Beispiel der Fall für einen Verbindungsaufbau zu jedem Knoten, der neu zum Netz hinzukommt. Keiner der späteren Zwischenknoten verfügt über entsprechende Routingtabelleneinträge, es ist auch nicht ist davon auszugehen, dass die notwendigen Signalisierungsverbindungen bereits aufgebaut wurden. Dieser Fall soll im Folgenden mit „Kaltstart“ bezeichnet werden.

2. die Latenz eines Verbindungsaufbaus zu einem Knoten über einen bereits bekannten Pfad. Dieser Extremfall tritt auf, wenn eine Verbindung abgebaut und anschließend wieder aufgebaut wird. Alle Zwischenknoten verfügen über aktuelle Routingtabelleneinträge und unterhalten die notwendigen Signalisierungsverbindungen, um die Verbindungsaufbaunachricht bis zum Zielknoten zu transportieren. Dieser Fall soll im Folgenden mit „Warmstart“ bezeichnet werden.

Ein Kaltstart beschreibt die Situation, in der sowohl Wegfindung als auch Verbindungsaufbau notwendig sind. Im Fall eines Warmstarts sind alle zur Wegfindung notwendigen Schritte bereits erfolgt und die gemessene Latenz ergibt sich allein aus den zum Verbindungsaufbau notwendigen Schritten.

Die in einem realen Netz zu erwartende Latenz beim Verbindungsaufbau wird sich zwischen den beiden Fällen einstellen. Einen Übergang zwischen den beiden Extremen ließe sich durch den Wegfindungsanteil formulieren. Damit ist der Anteil der Knoten des späteren Pfades gemeint, der neben dem Verbindungsaufbau noch die Wegfindung vornehmen muss. Im Fall eines Kaltstarts liegt dieser Anteil bei 100%, im Fall eines Warmstarts bei 0%.

Zur Messung der Latenz bei einem Verbindungsaufbau wurden die Adapter in einer Kette von zwei bis acht Geräten angeordnet. Bis auf die beiden endständigen Geräte konnte sich jeder Knoten mit genau zwei Nachbarn verbinden. Genau wie bei der Messung der Paketumlaufzeit waren am einen Ende ein Client und am anderen ein Echo-Server für Aufbau und die Entgegennahme eines Datenpfades verantwortlich. Über die bestehende Verbindung wurde vor Abbau derselben ein Datenpaket übertragen. Die korrekte Übertragung dieses Datenpaket bestätigte den fehlerfrei vollzogenen Verbindungsaufbau.

7.4.1 Kaltstart

Für jede Messung wurde zu jedem Bluetooth Adapter eine neue SNR-Instanz gestartet. Dies führte dazu, dass sämtliche Verbindungen neu aufgebaut werden mussten und jede Messung auf einer leeren Routingtabelle basierte. Die Messung entspricht dem Fall einer Verbindungssuche in einem Netz ohne jegliches Vorwissen.

	Ø [ms]	σ	pro Hop
1 Hop	1371	235	1371
2 Hop	2331	641	1166
3 Hop	4976	1859	1659
4 Hop	5446	557	1362
5 Hop	6951	590	1390
6 Hop	8243	1234	1374
7 Hop	10631	2048	1519

Tabelle 7.6: Latenz bei einem Kaltstart. Angegeben ist der Mittelwert über 200 Messungen, die Standardabweichung σ und die mittlere Latenz pro Hop.

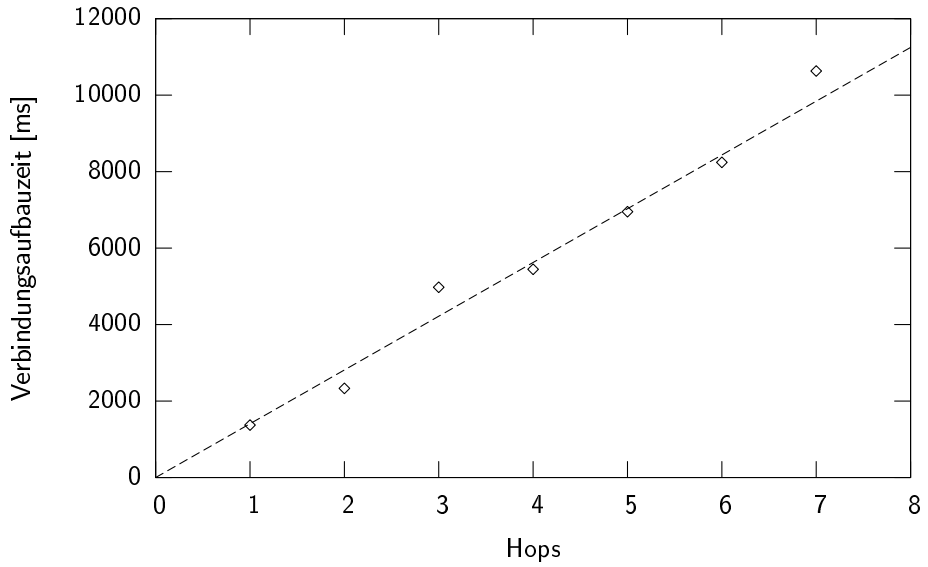

Abbildung 7.4: Latenz bei einem Kaltstart. Angegeben ist der Mittelwert über 200 Messungen. Desweiteren ist eine Fitting-Gerade mit der Annahme einer mittleren Latenz von 1400 ms pro Hop eingetragen.

Es wurden die Pfadlängen 2 – 7 Hops mit jeweils 200 Messungen untersucht. Die Mittelwerte der gemessenen Latenzen, sowie die Standardabweichung sind in Tabelle 7.6 aufgeführt. Daneben ist jeweils die mittlere Latenz pro Hop angegeben. Die Streuung der Messwerte war relativ groß, was an der Standardabweichung zu erkennen ist. Eine mögliche Ursache liegt darin, dass jedes Gerät zum Verbindungsaufbau ein Paging durchführen muss, dessen Länge nicht deterministisch vorhergesagt werden kann. Mit dem Paging werden Piconetze gebildet und die Synchronisation der Geräte hergestellt. Dieser Vorgang kann langwierig sein. Dennoch befindet sich die mittlere Latenz pro Hop über den gesamten Messbereich nie weit vom Mittelwert 1406 ms entfernt. Abbildung 7.4 veranschaulicht die Messergebnisse, sowie die Lage des Mittelwertes graphisch.

7.4.2 Warmstart

Für jeden Bluetooth Adapter wurde eine SNR-Instanz gestartet. Für jede Messung wurde eine neue Verbindung aufgebaut, eine Nachricht übertragen und an-

	Ø [ms]	σ	pro Hop
1 Hop	370	53	370
2 Hop	939	58	470
3 Hop	1405	125	483
4 Hop	1860	154	465
5 Hop	2362	348	472
6 Hop	2718	243	453
7 Hop	3157	284	451

Tabelle 7.7: Latenz bei einem Warmstart. Angegeben ist der Mittelwert über 200 Messungen, die Standardabweichung σ und die mittlere Latenz pro Hop.

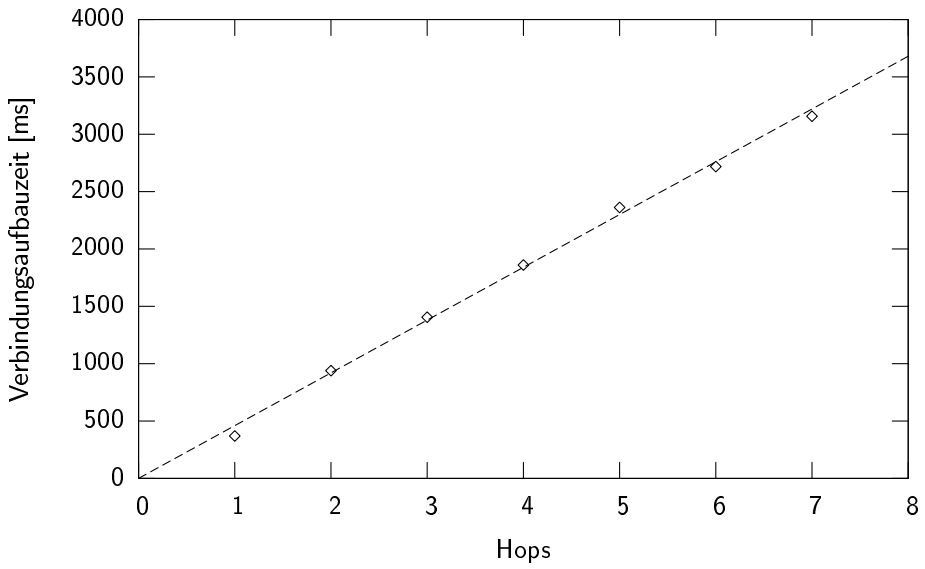

Abbildung 7.5: Latenz bei einem Warmstart. Angegeben ist der Mittelwert über 200 Messungen. Desweiteren ist eine Fitting-Gerade mit der Annahme einer mittleren Latenz von 460 ms pro Hop eingetragen.

schließend die Verbindung wieder abgebaut. Die verschiedenen Messungen wurden jedoch mit der selben SNR-Instanz durchgeführt, sodass bestehende Signalisierungsverbindungen und Routingtabelleneinträge ausgenutzt wurden. Der allererste Verbindungsaufbau nach dem Starten der Instanzen entspricht einem Kaltstart und wurde daher von der Auswertung ausgenommen. Die restlichen Verbindungsaufbauprozesse entsprechen dem Modell eines Warmstarts.

Für die Pfadlängen 2 – 7 Hops wurden jeweils 200 Messungen durchgeführt. Tabelle 7.7 zeigt die gemittelten Messwerte sowie die Standardabweichung und die mittlere Verbindungsaufbauzeit pro Hop. Abbildung 7.5 stellt die Messwerte gemeinsam mit einer Fitting-Geraden graphisch dar. Die Gerade wurde unter der Annahme einer mittleren Latenz pro Hop von 460 ms eingetragen. Dieser Wert weicht vom arithmetischen Mittel der gemessenen mittleren pro-Hop-Latenzen um nur 8 ms nach oben ab.

Abschließend ist festzustellen, dass die mittlere Latenz im Fall eines Kaltstarts um den Faktor drei größer ist als die im Fall eines Warmstarts.

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Protokoll zur Multihopkommunikation in Bluetooth Scatternetzen entwickelt. Dabei wurde nicht nur die Wegfindung betrachtet, sondern in einem umfassenden Ansatz sämtliche zur Multihopkommunikation notwendigen Teilprobleme analysiert und gelöst. Insbesondere betrifft dies die Multihopsuche nach Diensten und den Prozess des Verbindungsaufbaus. Dabei konnten Topologieausprägung, Wegfindung, Dienstsuche und Verbindungsaufbau gemeinsam gelöst werden, was den Signalisierungsaufwand auf ein Minimum reduziert. Durch den verbindungsorientierten Ansatz verbunden mit Out-Of-Band Signalisierung entsteht kein Overhead durch Routing bei der Kommunikation über eine bestehende Verbindung. Es müssen allein die Nutzdaten übertragen werden. Zwei Arten der Dienstadressierung wurden realisiert. Einerseits die klassische Adressierung anhand von Dienstidentifikator und Geräteadresse. Darüber hinaus noch eine Anycast-Adressierung für den Fall, dass der Dienstidentifikator, nicht aber die Geräteadresse des diensterbringenden Knotens bekannt ist. Zur Kompensation der Dynamik eines mobilen Sensor-Netzwerkes wurde eine Behandlung von Verbindungsausfällen realisiert, welche unterbrochene Datenverbindungen automatisch wieder in Stand setzt. Die Veränderung der Dienstcharakteristik der genutzten Bluetooth Verbindung von einem unzuverlässigen zu einem zuverlässigen Dienst ermöglicht die Übertragung von Datenströmen. Eine funktionsfähige Implementierung bestätigt die gewählten Ansätze.

Durch die Realisierung einer Multihopkommunikation für Bluetooth Netzwerke wird die Kommunikationsreichweite einzelner Geräte effektiv vergrößert. Jedes am Netzwerk beteiligte Gerät kann zu jedem anderen Gerät im gesamten Scatternetz Verbindungen aufbauen und Daten austauschen. Selbst die Stabilität einer direkten Kommunikation zwischen zwei benachbarten Geräten kann durch die Möglichkeit der Multihopkommunikation gepaart mit der Behandlung von Verbindungsausfällen erhöht werden, da bei Abschattung der Verbindung ein neuer Pfad zum Zielknoten über eine Kette erreichbarer Geräte gesucht und anschließend die Verbindung wieder hergestellt wird.

Im Hinblick auf den Einsatz im medizinischen Umfeld ist für die Zukunft sicher eine Authentifikation sämtlicher Zwischenknoten wünschenswert, da medizinische Daten immer als sensibel anzusehen sind und daher besonderen Schutzes bedürfen.

Es bleibt zu prüfen, ob sich die Authentifikation der Knoten ebenfalls in den Prozess der Wegfindung integrieren lässt.

Eine weitere sinnvolle Erweiterung stellt die Integration von „Quality of Service“ Aspekten dar. Die Sensorknoten des vorgestellten Einsatzszenarios erzeugen häufig ein gleichmäßiges Datenvolumen. Diese Information könnte bei der Pfadsuche Beachtung finden, indem nur Zwischenknoten zur Weiterleitung in Betracht gezogen weden, die über ausreichend freie Kapazitäten verfügen. Diese Vorgehensweise würde helfen Stausituationen und Datenverluste im Netz zu vermeiden. Eine Priorisierung der Datenströme könnte in Überlastsituationen eine sinnvolle Entscheidung erleichtern, welchen Daten der Vorrang zu gewähren ist. Insbesondere in einem medizinischen Sensornetz muss gewährleistet sein, dass wichtige Informationen — wie z.B. ein von einem Sensorknoten ausgelöster Alarm — nicht im Netz verloren gehen, sondern mit Sicherheit ihr Ziel erreichen.

Eine geordnete Verbindungsübergabe (Handover) zwischen weiterleitenden Knoten bei Erreichen der Grenze der Funkreichweite würde die Qualität der Datenübertragung weiter erhöhen. Bluetooth erlaubt die Überwachung der Signalpegel aus den hohen, anwendungsorientierten Schichten heraus. Ein Sensorknoten könnte damit einen bevorstehenden Verbindungsabbruch detektieren und schon im Voraus nach einem alternativen Pfad mit besserem Signalpegel suchen. Eine anschließende, geordnete Verbindungsübergabe würde helfen, durch Reparatur verursachte Wartezeiten, sowie Paketverluste zu vermeiden.

Anhang A

AODV Nachrichten

Im Folgenden sind die in [PeBR03] definierten AODV-Nachrichten wiedergegeben.

A.1 RREQ

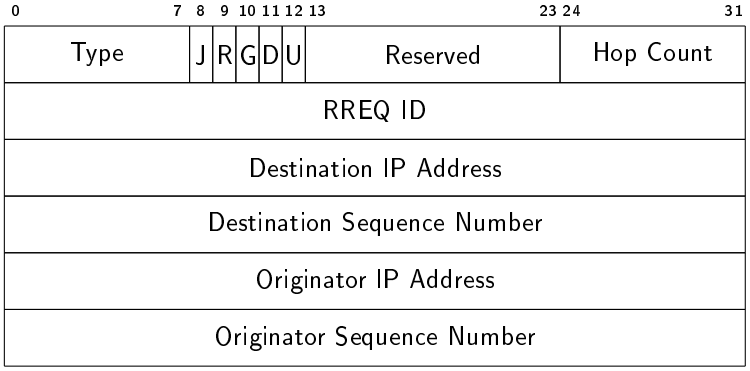

Abbildung A.1: AODV Route Request (RREQ) Nachrichtenformat

Type 1

J Join Flag, für Multicast reserviert.

R Repair Flag, für Multicast reserviert.

G Gratuituous RREP Flag,

D Destination Only Flag, nur der angegebene Zielknoten darf diesen RREP beantworten.

U Unknown Sequence Number Flag; zeigt unbekannte Zielsequenznummer an.

Reserved
Bits werden als 0 gesendet, beim Empfang ignoriert.

Hop Count
Anzahl Hops von der Quelle des RREQ bis zum bearbeitenden Knoten.

RREQ ID
Sequenznummer, die — zusammen mit der Absenderadresse — die Nachricht eindeutig identifiziert.

Destination IP Address
Adresse des Zielknotens.

Destination Sequenz Number
Die neuste, dem Sender bekannte Zielsequenznummer

Originator IP Address
Adresse des Quellknotens.

Originator Sequence Number
aktuelle Sequenznummer des Quellknoten.

A.2 RREP

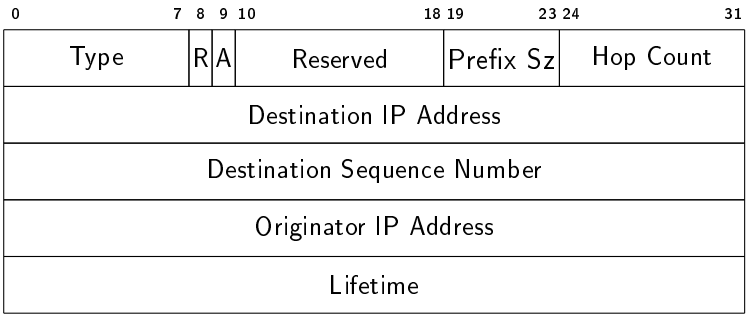

Abbildung A.2: AODV Route Reply (RREP) Nachrichtenformat

Type 2

R Repair Flag, wird für Multicast benutzt.

A Acknowledgement required Flag

Reserved
Bits werden als 0 gesendet, beim Empfang ignoriert.

Prefix Sz
falls nicht Null, zeigt das Feld an, dass der zugehörige Next Hop für alle Knoten genutzt werden kann, die den selben Routing Präfix (definiert durch PrefixSz) wie der Zielknoten haben.

Hop Count
Anzahl Hops von der Quelle des RREQ bis zum Zielknoten.

Destination IP Address
Adresse des Zielknotens zu dem eine Route aufgebaut wird.

Destination Sequenz Number
mit der Route assoziierte Sequenznummer.

Originator IP Address
Adresse des Knotens, der den zugehörigen RREQ ausgesandt hat.

Lifetime Gültigkeitsdauer der Route.

A.3 RERR

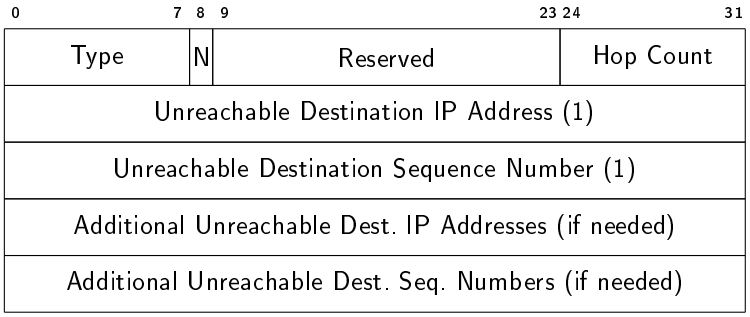

Abbildung A.3: AODV Route Error (RERR) Nachrichtenformat

Type 3

N No Delete Flag; zeigt an, dass ein Knoten die Route lokal reparieren konnte. Upstream Knoten sollen den Eintrag nicht aus ihrer Routingtabelle löschen.

Reserved
Bits werden als 0 gesendet, beim Empfang ignoriert.

Dest Count
Anzahl der durch diese Nachricht als nicht erreichbar signalisierte Knoten.

Unreachable Destination IP Address
Adresse des nicht mehr erreichbaren Zielknotens.

Unreachable Destination Sequenz Number
zugehörige Sequenznummer.

A.4 RREP-ACK

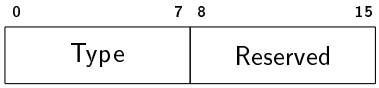

Abbildung A.4: AODV Route Reply Acknowledgement (RREP-ACK) Nachrichtenformat

Type 4

Reserved

Bits werden als 0 gesendet, beim Empfang ignoriert.

Anhang B

Inhalt der CD

Der schriftlichen Ausarbeitung der Diplomarbeit ist eine CD beigelegt, die zusätzliche Informationen und Materialien zur Dokumentation enthält.

Ausarbeitung als PDF
: Die schriftliche Ausarbeitung ist im PDF-Format auf der CD zu finden.

Java Quellcode
: Der Java Quellcode der SNR-Implementierung. Daneben befindet sich der Quellcode der Client- und Server-Applikationen aus der Evaluation. Diese können als Beispiele zur gedachten Nutzung der SNR-Schnittstelle herangezogen werden.

Dokumentation des Quellcodes
: Dokumentation der SNR-Implementierung im HTML-Format.

Evaluationsergebnisse und Protokolle aus der Evaluation
: Die Evaluation ist anhand der, durch die SNR-Implementierung und ihre Client- und Server-Applikationen erzeugten Log-Dateien nachvollziehbar dokumentiert. Die Log-Dateien sind der CD beigefügt. Ihnen sind alle im Kapitel Evaluation vorgestellten Ergebnisse entnommen.

Projektionsfolien des Diplomvortrages
: Ebenfalls sind die Projektionsfolien des am 22. März 2006 gehaltenen Diplomvortrages auf der CD enthalten.

Abkürzungsverzeichnis

ABR	Associativity-Based Routing
ABT	Avetana Bluetooth JSR-82 Implementierung
ACK	Acknowledge
ACL	Asynchronous Connection-oriented Logical transport
AODV	Ad hoc On demand Distance Vector Routing
API	Application Programming Interface, Programmierschnittstelle
BNEP	Bluetooth Network Encapsulation Protocol
CGSR	Clusterhead-Gateway Swith Routing
CREL	Connection Release
CREP	Connection Reply
CREQ	Connection Request
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DSDV	Destination-Sequenced Distance-Vector Routing
DSR	Dynamic Source Routing
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
FSR	Fisheye State Routing
GAP	Generic Access Profile
GIAC	General Inquiry Access Code
GOEP	Generic Object Exchange Profile
HCI	Host Controller Interface
IAC	Inquiry Access Code
ID	Identifikationsnummer
IETF	Internet Engineering Task Force
JSR	Java Specification Request
L2CAP	Logical Link Control and Adaptation Protocol
LANMAR	Landmark Ad Hoc Routing Protocol

LAP	Lower Address Part
LMP	Link Manager Protocol
LMR	Lightweight Mobile Routing
MANET	Mobile Ad Hoc Networks
NAP	Non-Significant Address Part
OBEX	Object Exchange Protocol
OLSR	Optimized Link State Routing
PSM	Protocol Service Multiplexer
RERR	Route Error
RFCOMM	*RFCOMM ist keine Abkürzung. Siehe Glossar.*
RFC	Request for Comments
RREP	Route Reply
RREQ	Route Request
RTT	Round Trip Time
RVM	Routing Vector Method
SCO	Synchronous Connection-Oriented logical transport
SDAP	Service Discovery Application Profile
SDP	Service Discovery Protocol
SNR	ScatterNetz-Routing
SPP	Serial Port Profile
SRS	Scatternet Route Structure
TBRPF	Topology Broadcast Based on Reverse Path Forwarding
TCS	Telephony Control protocol Specification
TORA	Temporally Ordered Routing Algorithm
UAP	Upper Address Part
UUID	Universally Unique Identifier
WLAN	Wireless LAN
ZRP	Zone Routing Protocol

Glossar

Access Code: Nachrichten der Bluetooth Basisband-Schicht beginnen mit einem meist 72 Bit (in Ausnahmefällen auch 68 Bit) langen Access Code. Diese Bitfolge dient zur Identifikation des Piconetzes und der zeitlichen Synchronisation der Geräte beim Empfang. Der Access Code leitet sich — während der regulären Kommunikation im Piconetz — aus dem LAP der Geräteadresse des Masters ab.

Active Member Address: Jeder aktive Slaveknoten eines Piconetzes bekommt von seinem Master eine kurze 3 Bit Adresse zugewiesen, die an Stelle der 48 Bit langen Bluetooth Geräteadresse zur Kommunikation innerhalb des Piconetzes eingesetzt wird.

Anycast: Adressierungsart, spricht einen beliebigen (nicht weiter spezifizierten) Knoten aus einer Gruppe von Knoten an.

AODV: Ad hoc On demand Distance Vector routing protocol. Reaktives Routing Protokoll für Ad hoc Netzwerke.

Broadcast: Auch Rundruf genannt. Datenpakete werden von einem Punkt aus gleichzeitig an alle Teilnehmer eines Netzes oder Subnetzes übertragen.

CREL: Connection Release. SNR-Signalisierungsnachricht zum ordnungsgemäßen Abbau einer Datenverbindung.

CREP: Connection Reply. SNR-Signalisierungsnachricht mit der eine Pfadanfrage und ein Verbindungsaufbauwunsch beantwortet wird. Diese Nachricht löst den Aufbau einer Datenverbindung aus.

CREQ: Connection Request. SNR-Signalisierungsnachricht die die Wegfindung initiiert und gleichzeitig einen Verbindungsaufbauwunsch mitteilt.

CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance. Verfahren zur Kollisionsvermeidung, wenn mehrere Netzteilnehmer auf das selbe Medium zugreifen. Findet bei WLAN Anwendung.

FEC: Forward Error Correction. Daten werden redundant codiert, um sie trotz fehlerhafter Datenübertragung rekonstruieren zu können.

FHM-Nachricht: Signalisierungsnachricht der Bluetooth Basisbandschicht. Es überträgt unter Anderem die Geräteadresse des Senders und dessen interne Uhrzeit.

Flusssteuerung: Stellt sicher, dass ein Knoten nur so viele Nachrichten sendet, wie sein Kommunikationspartner verarbeiten kann.

GPS: Global Positioning System. Satellitengestütztes System zur weltweiten Positionsbestimmung.

Handover: Strategie zur Mobilitätsunterstützung. Bevorstehende Verbindungsabbrüche werden detektiert und die bestehende Verbindung geordnet an einen neuen Kommunikationspartner übergeben.

Hop: *(engl. Etappe)* In Netzwerken werden Informationen häufig indirekt über eine Kette von Zwischenknoten übertragen. Einen einzelnen Sprung in dieser Kette bezeichnet man als Hop.

Hoppingsequenz: Abfolge von Frequenzen, die bei Bluetooth nacheinander zur Kommunikation verwendet werden. Die Hoppingsequenz wird anhand der Geräteadresse des Masters bestimmt und ist für das durch den Master verwaltete Piconetz charakteristisch.

ID-Nachricht: Identifikationsnachricht der Bluetooth Basisband Schicht. Es hat eine feste Länge von 68 bit und kann aus dem „Device Access Code“ oder dem „Inquiry Access Code“ bestehen.

ISM-Band: Industrial, Scientific, and Medical Band. Frequenzbereich, der lizenzfrei genutzt werden darf. Insbesondere das 2,4 GHz Band ist weltweit einheitlich lizenzfrei nutzbar.

L2CAP: Logical Link Control and Adaptation Protocol. Protokoll der Sicherungsschicht des Bluetooth Stacks. Unterstützt das Multiplexen höherer Protokolle, Segmentierung und Reassemblierung von Nachrichten, sowie die Übermittlung von Quality of Service Informationen.

LAP: Lower Address Part. Bit 0 – 23 der Bluetooth Geräteadresse. Siehe dazu Abbildung 2.2 auf Seite 10.

Multicast: Dieselbe Nachricht wird von einem Sender ausgehend an mehrere, wohldefinierte Zielknoten übermittelt.

multihop: Hier: Indirekte Verbindung über Zwischenknoten. Bezeichnet die Kommunikation von Nachrichten oder eine Verbindung die sich über eine Kette von mehreren benachbarten Knoten erstreckt.

PSM: Protocol Service Multiplexer. Identifikator für einen L2CAP basierten Dienst auf einem Bluetooth Gerät. Der Wertebereich spaltet sich in zwei Intervalle. Das erste beinhaltet die von der Bluetooth SIG fest zugeordneten Werte der Basisdienste wie SDP und RFCOMM. Werte aus dem zweiten Intervall werden dynamisch vergeben und können über das „Service Discovery Protocol“ zur Laufzeit erfragt werden.

RERR: Route Error; AODV Signalisierungsnachricht. Zeigt den Verbindungsabbruch zu einem oder mehreren Zielknoten an.

RFCOMM: Protokoll des Bluetooth-Stacks zur Emulation einer seriellen Verbindung.

RREP: Route Reply; AODV Signalisierungsnachricht. Kann ein Netzwerkknoten die durch einen RREQ signalisierte Anfrage auflösen, so antwortet dieser mit einer RREP Nachricht.

RREQ: Route Request. AODV Signalisierungsnachricht, welche die Wegfindung initiiert. Ein Knoten fragt seine Nachbarn nach einem Weg zu einem bestimmten Zielknoten. Diese antworten ggf. mit einer RREP Nachricht.

SDP: Service Discovery Protocol. Protokoll des Bluetooth-Stacks zum Auffinden von Diensten in Bluetooth Piconetzen.

Unicast: Auch Punkt zu Punkt Verbindung. Datenpakete werden von einem Punkt aus an genau einen Empfänger übertragen.

Literaturverzeichnis

[AKKa04] Jamal N. Al-Karaki und Ahmed E. Kamal. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communications*, Dezember 2004, S. 6 – 28.

[BCSW98] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk und Barry A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the 4th annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1998, S. 76–84.

[BhSe99] Pravin Bhagwat und Adrian Segall. A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets. In *IEEE International Workshop on Mobile Multimedia Communications*, 1999, S. 375 – 379.

[BKRS+03] Jan Beutel, Oliver Kasten, Matthias Ringwald, Frank Siegemund und Lothar Thiele. Bluetooth Smart Nodes for Mobile Ad-hoc Networks. TIK-Report No 167, Computer Engineering and Networks Lab, Swiss Federal Institute of Technology (ETH) Zurich, 8092 Zurich, Switzerland, April 2003.

[Blue04] Bluetooth SIG. *Specification of the Bluetooth System — Version 2.0*, 4. November 2004. http://www.bluetooth.org.

[ChGe97] Ching-Chuan Chiang und M. Gerla. Routing and Multicast in Multihop, Mobile Wireless Networks. In *IEEE 6th International Conference on Universal Personal Communications*, Band 2, Oktober 1997, S. 546–551.

[ClJa03] T. Clausen und P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC3626, http://tools.ietf.org/wg/manet/draft-ietf-manet-olsr/rfc3626.txt, Oktober 2003.

[CoEp95] M. S. Corson und A. Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, März 1995, S. 61 – 81.

[HaPe01] Z. J. Haas und M. R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. In *IEEE/ACM Transactions on Networking*, Band 9, August 2001, S. 427–438.

[HoXG02] Xiaoyan Hong, Kaixin Xu und Mario Gerla. Scalable Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, Juli 2002, S. 11 – 21.

[JoMH04] David B. Johnson, David A. Maltz und Yih-Chun Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). draft-ietf-manet-dsr-10.txt, Juli 2004.

[KaGe03] Rohit Kapoor und Mario Gerla. A Zone Routing Protocol for Bluetooth Scatternets. In *IEEE Wireless Communications and Networking*, Band 3, März 2003, S. 1459 – 1464.

[KaGS00] M. Kalia, S. Garg und R. Shorey. Scatternet structure and inter-Piconet communication in the Bluetooth system. *IEEE National Conference Communications 2000, New Delhi, India*, 2000.

[KoVa00] Young-Bae Ko und Nitin H. Vaidya. Location-aided routing (LAR) in Mobile Ad Hoc Networks. *Wireless Networks*, Band 6, Juli 2000, S. 307–321.

[LiLS03] Yong Liu, Myung J. Lee und Tarek N. Saadawi. A Bluetooth Scatternet-Route Structure for Multihop Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 21(2), Februar 2003, S. 229 – 239.

[MaDa01] Mahesh K. Marina und Samir R. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Ninth International Conference on Network Protocols*, November 2001, S. 14–23.

[Moto02] Motorola, Wireless Software, Applications & Services. *Java™ APIs for Bluetooth™ Wireless Technology (JSR-82) — Specification Version 1.0a*, 4. April 2002.

[Nalm97] Julio C. Navas und Tomasz Imielinski. GeoCast — Geographic Addressing and Routing. In *Proceedings of the 3rd annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1997, S. 66–76.

[OgLT03] R. Ogier, M. Lewis und F. Templin. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). draft-ietf-manet-tbrpf-07.txt, März 2003.

[PaCo01] V. Park und S. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1, Functional Specification. draft-ietf-manet-tora-spec-04.txt, Juli 2001.

[PeBh94] Charles Perkins und Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *(ACM) SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994, S. 234–244.

[PeBR03] Charles E. Perkins und Elizabeth M. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC3561, http://www.ietf.org/rfc/rfc3561.txt, Juli 2003.

[PeGC00] Guangyu Pei, Mario Gerla und Tsu-Wei Chen. Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks. In *IEEE International Conference on Communications (ICC)*, 2000, S. 70–74.

[PeGH00] Guangyu Pei, Mario Gerla und Xiaoyan Hong. LANMAR: Landmark Routing for Large Ad Hoc Wireless Networks with Group Mobility. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, 2000, S. 11–18.

[PeRo99] Charles E. Perkins und Elizabeth M. Royer. Ad Hoc On-demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Februar 1999, S. 90 – 100.

[Schi03] Jochen Schiller. *Mobilkommunikation*. Addison-Wesley. 2003.

[SuCL02] Min-Te Sun, Chung-Kuo Chang und Ten-Hwang Lai. A Self-Routing Topology for Bluetooth Scatternets. In *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '02)*, Mai 2002, S. 13 – 18.

[Toh97] Chai-Keong Toh. Associativity-Based Routing for Ad Hoc Mobile Networks. *Wireless Personal Communications*, März 1997.

[XuHE01] Ya Xu, John Heidemann und Deborah Estrin. Geography-informed energy conservation for Ad Hoc routing. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, S. 70–84.