BLUETOOTH
SCATTERNETZE

SEMINARARBEIT

im Rahmen des Teleseminars

zu Ubiquitous Computing im Wintersemester 2001/2002

vorgelegt am

Lehrstuhl für Praktische Informatik IV
Universität Mannheim
Prof. Dr. W. Effelsberg

Betreuer: Dr. Hartmut Ritter
Institut für Telematik
Forschungsgruppe von Prof. Dr. M. Zitterbart
Universität Karlsruhe

Christian Lochert
Sandrain 73
68219 Mannheim
Tel.: (0621) 8 76 26 34
lochert@rumms.uni-mannheim.de
9. Fachsemester
Wirtschaftsinformatik

Inhaltsverzeichnis

1 Einleitung

Bluetooth[1] ist eine Netzwerktechnologie, welche ein kabelloses Kommunizieren mehrerer Endgeräte ermöglicht. Haupteinsatzgebiet von Bluetooth-Geräten sind in erster Linie Geräte in unmittelbarer Umgebung des PC. Dazu zählen z.B. Drucker, Modem, Scanner, etc.

1.1 Einsatzbereiche von Bluetooth

Warum benötigt man Bluetooth-Geräte? Wie oben erwähnt dient Bluetooth einerseits der Schaffung von Übersichtlichkeit am Arbeitsplatz durch Wegfall der Kabelverbindungen zwischen PC und Drucker, PC und Modem, etc. Des weiteren steht aber nicht nur die Kommunikation eines PC mit einem Drucker im Vordergrund, sondern ein beliebiger PC möchte einen Druckauftrag an einen Farbdrucker schicken. Dazu stellt der PC eine Anfrage mittels Broadcasting an alle Geräte in seiner Umgebung, ob sie diesen gewünschten Dienst „Drucken in Farbe“ zur Verfügung stellen. Leicht kann so ein Druckerpool aufgebaut werden, ohne dass dazu eine verkabelte Netzstruktur aufgebaut werden müsste.

1.2 Technische Grundlagen

Bluetooth ist eine kabellose, preisgünstige und energiesparsame Technologie für den Einsatz in einem Personal Area Network (PAN) in einem Bereich von etwa 10 m um die zentrale Station (Master). Der Master kann mit maximal sieben Slaves in Verbindung treten. Durch Benutzung dieser strikten Master-Slave-Beziehung kann eine Kommunikation nur von Master zu Slave geführt werden, d.h. eine Kommunikation zweier Slaves bzw. zweier Master ist direkt nicht möglich. Bluetooth arbeitet in dem weltweit freien und unlizenzierten ISM (Industrial, Scientific, Medical) Band von 2,4 bis 2,5 GHz[2] [3]. Um gegen Störungen anderer Geräte unempfindlich zu sein, benutzt Bluetooth ein Frequenzsprungverfahren in Kombination mit Zeitmultiplexing (Frequency Hopping Spread Spectrum with Time Division Duplex). Dazu wird das Band in 79 (bzw. 23 in Frankreich, Spanien und Japan) Kanäle mit jeweils 1 MHz Größe aufgeteilt und darüber 1600 mal

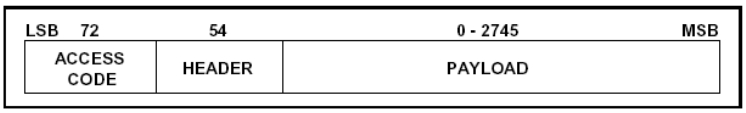
[1] Siehe auch [Blue01].

[2] In den USA sowie Europa von 2400 bis 2483,5 MHz mit Ausnahme von Frankreich und Spanien, welche ein kleineres Band zur freien Verfügung stellen. In Japan steht ebenso nicht das gesamte Band zur freien Verfügung.

[3] Siehe auch [Haar00] sowie [Bhag01].

pro Sekunde gesprungen. Dies ergibt eine Zeitschlitzgröße von 625 μs. Um einen möglichst hohen Datendurchsatz zu erhalten wird dies ergänzt durch schnelle, automatische Wiederholungsanfragen (ARQ/ Automatic Repeat Request), durch Prüfsummen (CRC/ Cyclic Redundancy Check) sowie durch eine Vorwärtsfehlerkorrektur (FEC/ Forward Error Correction).

In Bluetooth besteht eine statische Aufteilung der Zeitschlitze. Der Master kann jeweils in geraden Zeitschlitzen, ein Slave jeweils in ungeraden Zeitschlitzen Daten senden. Offensichtlich ist deshalb, dass Pakete eine ungerade Länge haben müssen. Es gibt Pakete, welche eine Länge von 1, 3 oder 5 Zeitschlitzen haben. Um die Einteilung der Zeitschlitze bei den Slaves zu erhalten, müssen sich diese mit dem Master synchronisieren, dazu wird zu der Uhr des Slaves ein Offset dazugezählt. Danach wird durch den Master die Frequenzsprungsequenz festgelegt[4].



Quelle: [Blue01].

Abbildung 1: Standard packet format

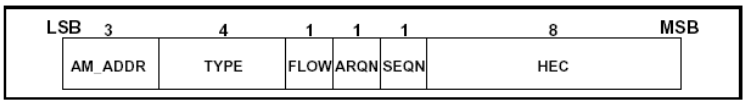
1.3 Kommunikation zwischen Endgeräten

Ein nicht verbundener Bluetooth-Knoten A befindet sich in seiner Ausgangssituation in einem *Standby-Zustand*. Um eine Verbindung zu einem anderen Knoten aufzubauen, wechselt er in den *Verbindungsaufbauzustand*. Entweder ist Knoten A bereits die Adresse des anderen Knoten bekannt, oder er muss erst durch eine INQUIRY-Broadcastnachricht, mittels einer Inquiry-Frequenzsprungssequenz innerhalb von 32 Kanälen[5] mit 3200 Sprüngen pro Sekunde, erfragen, ob sich Knoten mit dem gewünschten Dienst in Reichweite aufhalten. Existieren diese und sind ihrerseits in einem INQUIRY SCAN-Zustand, schicken sie, nach einer zufälligen Backoffzeit T_{b_0} innerhalb 0 bis 1023 Zeitschlitze, ihre Adresse (BD_ADDR[6]) zurück. Wenn nach einem Timeout keine Nachrichten zurück kamen, existieren keine solchen Knoten in Reichweite und der Sendewunsch schlägt fehl. Sonst kennt Knoten A jetzt die Adresse der möglichen Empfänger und sendet an eines von diesen, die sich in einem PAGE SCAN-Zustand befinden, eine PAGE-Nachricht. Knoten A wird dann Master in diesem Netz und der andere Knoten wird Slave und erhält vom Master eine 3-bit lange Mitgliedsadresse (AM_ADDR; Abbildung 2). Beide Knoten wechseln nun in den *aktiven Zustand*.

[4] Eine Sequenz dauert länger als 23 Stunden.

[5] In Japan und anderen Ländern werden nur 16 Kanäle benutzt.

[6] BD_ADDR ist eine 48 bit weltweit eindeutige Adresse eines Bluetooth-Gerätes.



Quelle: [Blue01].

Abbildung 2: Header format

Nur in diesem können Datenpakete versandt und empfangen werden. Bei den Datenpaketen besteht die Möglichkeit diese synchron und verbindungsorientiert (SCO) oder asynchron und verbindungslos (ACL) zu übertragen. Dabei werden zuerst die SCO-Pakete bedient und dann bei noch freier Bandbreite die ACL-Pakete übertragen (Abbildung 5).

SCO-Pakete werden häufig dazu benutzt, um Audiodaten zu übertragen, da hier eine feste Bandbreite durch eine Punkt-zu-Punkt-Verbindung garantiert werden kann. D.h. es wird ein Zeitslot bestimmt, in dem der Slave senden darf. Dazu schickt der Master ein Paket, das der Slave im nächsten Slot beantworten darf. SCO-Links können, aufgrund ihrer Bestimmung zur Verschickung von 1-Slot-Audiopaketen, diese mit einer maximalen symmetrischen Datenrate von 64 kbit/s übermitteln. SCO-Pakete werden ohne Prüfsummen, aber je nach Pakettyp mit einer Vorwärtsfehlerkorrektur, verschickt.

ACL-Links können ihre Bandbreite symmetrisch oder asymmetrisch aufteilen. Dabei können ACL-Pakete mit einer Länge von maximal 3 Slots symmetrisch mit jeweils 108,8 kbit/s oder asymmetrisch mit 721,0 bzw. 57,6 kbit/s übertragen werden. Die Übertragung der ACL-Pakete geschieht i.d.R. mittels Prüfsummen und Vorwärtsfehlerkorrektur in den Paketen. Der Standard sieht hier sieben Pakettypen vor. Pakete mit CRC und FEC haben die Bezeichnungen DM1, DM3 und DM5, wohingegen Pakete mit CRC aber ohne FEC die Bezeichnungen DH1, DH3 und DH5. Pakete des Typs AUX1 haben weder CRC noch FEC (siehe auch Abbildung 3 bzw. analog für SCO-Pakete Abbildung 4).

Die letzte Kategorie von Paketen existiert sowohl als SCO als auch als ACL Pakettyp. Es sind dies die NULL und POLL Pakete. Das NULL Paket dient mit einer fixen Länge von 126 bit der Bestätigung von erfolgreich verlaufenen Übertragungen bzw. zur Flusskontrolle. Die POLL Pakete werden im Gegensatz zu den NULL Paketen wiederum bestätigt. Die Aufgabe der POLL Pakete ist es, die Slaves zu einer Antwort aufzufordern, auch selbst dann, wenn diese keine Nachrichten zu senden haben.

Um eine Verbindung zu beenden, können beide Knoten in den Standby-Zustand zurückwechseln.

Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)	Asymmetric Max. Rate (kb/s)	
						Forward	Reverse
DM1	1	0-17	2/3	yes	108.8	108.8	108.8
DH1	1	0-27	no	yes	172.8	172.8	172.8
DM3	2	0-121	2/3	yes	258.1	387.2	54.4
DH3	2	0-183	no	yes	390.4	585.6	86.4
DM5	2	0-224	2/3	yes	286.7	477.8	36.3
DH5	2	0-339	no	yes	433.9	723.2	57.6
AUX1	1	0-29	no	no	185.6	185.6	185.6

Quelle: [Blue01].

Abbildung 3: ACL-Pakete

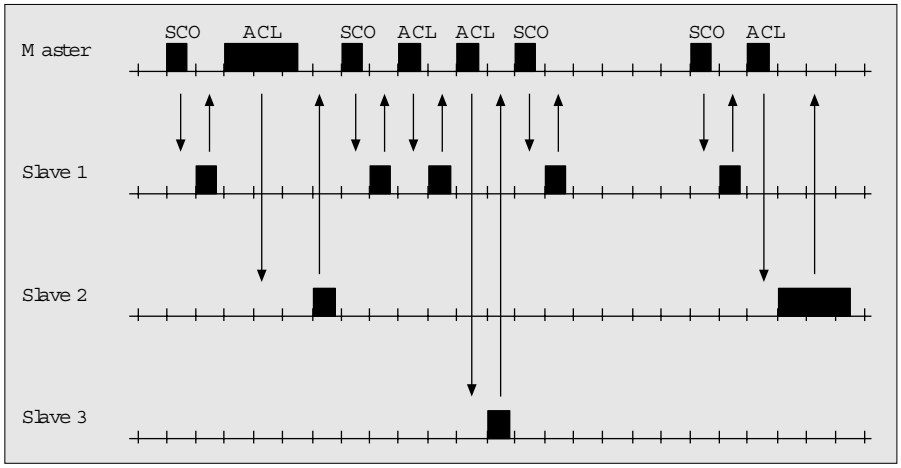
Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)
HV1	na	10	1/3	no	64.0
HV2	na	20	2/3	no	64.0
HV3	na	30	no	no	64.0
DV*	1 D	10+(0-9) D	2/3 D	yes D	64.0+57.6 D

Quelle: [Blue01].

Abbildung 4: SCO-Pakete

Soll die Verbindung allerdings nicht deshalb getrennt werden, weil die Kommunikation abgeschlossen ist, sondern um Energie zu sparen, kann ein Knoten in den *Stromsparmodus* wechseln. Im *Sniff-Zustand* wird zwischen Master und Slave eine Periode vereinbart. Der Slave wacht zu Beginn jeder Periode auf, um angesprochen werden zu können. Im *Hold-Zustand* wird zwischen Master und Slave eine Zeitspanne vereinbart. In dieser ist der ACL-Link getrennt, d.h. der Slave ist während dieser Zeitspanne nicht für ACL-Pakete erreichbar. Im *Park-Zustand* wird der Slave geparkt und muss seine AM_ADDR abgeben. Er erhält dafür vom Master eine eindeutige PM_ADDR und eine, nicht notwendigerweise eindeutige, AR_ADDR. Die PM_ADDR wird vom Master benutzt, um den Slave wieder aufzuwecken. Möchte der Slave selbst wieder aufwachen, benutzt er dafür seine AR_ADDR, um nach einer empfangenen Broadcastnachricht in einem Halbintervall zu senden.

Die Arbeit wird sich zuerst mit der Frage beschäftigen, welche Möglichkeiten existieren, um Teilnetze zu bilden. Danach wird anhand der diversen Papers darauf eingegangen, welche konkreten Vorschläge zur Bildung von Scatternetzen gemacht worden sind. Diese werden kurz einer kritischen Würdigung unterworfen. Im Anschluss daran werden Probleme und Möglichkeiten erörtert, wie in gebildeten Scatternetzen konkret Pakete weitergeleitet werden können. Zum Schluss gibt eine Zusammenfassung der Arbeit das Wesentliche wider und zeigt weiteren Forschungsbedarf auf.



Quelle: [Haar00].

Abbildung 5: SCO und ACL Verbindung mit einem Master und drei Slaves

1.4 Bildung von Teilnetzen

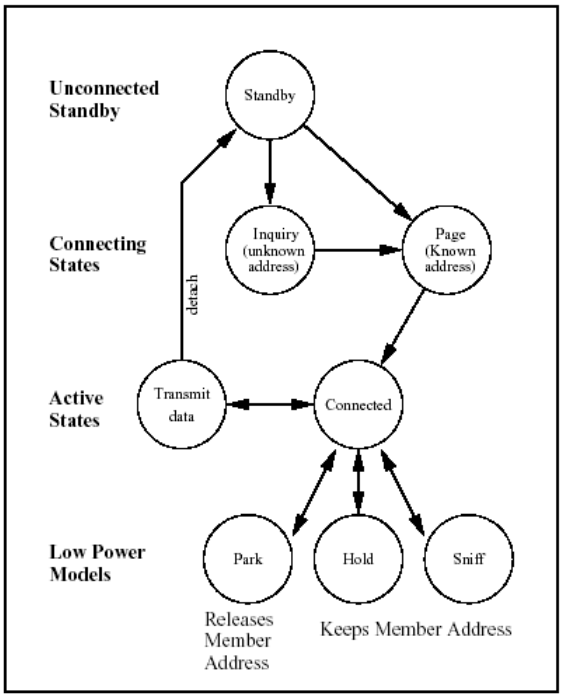
Ein Netz bestehend aus einem Master und maximal sieben Slaves wird Piconetz genannt. Die starke Einschränkung der Kommunikation, nur mit maximal sieben Slaves kommunizieren zu können, wird durch die 3-bit AM_ADDR im Header bestimmt. Um nun mit mehr als sieben Knoten zu kommunizieren, sieht der Standard vor, mehrere Piconetze zu einem Scatternetz zusammenzufügen. Dabei müssen folgende Bedingungen gelten:[7]

1. **Knotentypbeschränkung**
   Jeder Knoten ist entweder Master oder Slave.
2. **Gradbeschränkung**
   Ein Master kann maximal sieben Slaves versorgen
3. **Verbindungsbeschränkung**
   (a) Zwei Slaves können nicht direkt miteinander kommunizieren
   (b) Zwei Master sollten nicht direkt miteinander kommunizieren

Unabhängig von der Einhaltung dieser Bedingungen können aber dennoch Probleme bei der Bildung der Scatternetze entstehen. So existieren bei nur zehn Knoten bis zu 820800 Möglichkeiten ein Scatternetz zu bilden[8].

Um eine übersichtlichere Darstellung zu geben, können folgende drei Möglichkeit, ein Scatternetz zu bilden, als Hauptgliederungsmerkmale genannt werden:[9]

[7] Nach [BR00].
[8] Vgl. [BR00].
[9] Nach [KGS00].



Quelle: [HNIJ98].

Abbildung 6: Connection State Machine

1. **Single Piconet Model**
   Es wird genau ein Piconetz gebildet. Sind mehr als acht Geräte vorhanden, müssen sich Geräte, die nicht an der momentanen Kommunikation teilnehmen wollen, ausbuchen, um anderen die Kommunikation zu ermöglichen.

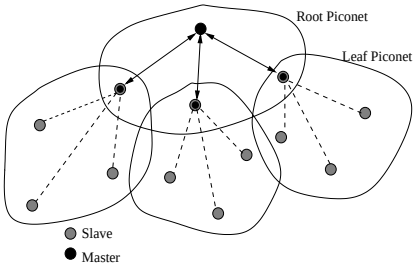
2. **Two-Level hierarchy of Piconets**
   Es wird eine Baumstruktur geschaffen, wobei die Söhne der obersten Ebene die Master der darunterliegenden Piconetze sind (siehe Abbildung 7).

3. **Shared Master/Slave Piconets**
   Knoten, die kommunizieren möchten, verbinden sich zu Piconetzen. Innerhalb dieser Piconetze gibt es Brückenknoten, welche in zwei Piconetzen eingebucht sind, um so Pakete weiterleiten zu können (siehe Abbildung 8),

2 Bildung von Scatternetzen

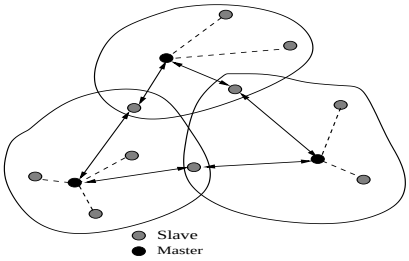
2.1 Bluetooth Topology Construction Protocol (BTCP)

Um ein Scatternetz aufzubauen, das bspw. in einer Konferenzsituation benötigt werden könnte, beschreiben [SBTL01] das Bluetooth Topology Construction Protocol (BTCP). Dazu werden einige Vorausbedingungen gestellt:



Quelle: [KGS00].

Abbildung 7: Two Level hierarchy of Piconets



Quelle: [KGS00].

Abbildung 8: Shared Slave Piconets

1. Ein Brückenknoten kann nur mit zwei Piconetzen verbunden sein.
2. Ein Scatternetz sollte aus der minimalen Anzahl an Piconetzen bestehen.
3. Das resultierende Scatternetz sollte vollständig verbunden sein.
4. Zwei Piconetze teilen sich nur eine Brücke.

BTCP basiert auf einem Konzept, welches genau einen Knoten auswählt, der der Koordinator für das gesamte Scatternetz ist. Dieser könnte in Erweiterung des Papers so ausgewählt werden, dass er genügend Rechenleistung für diese Aufgabe hat, oder nicht zu sehr mit anderen Aufgaben/ Anwendungen beschäftigt ist.

Der genaue Ablauf des Algorithmus gliedert sich in drei Phasen:

Die erste Phase bestimmt den zentralen Knoten. Dazu wird in jedem Knoten eine Variable VOTES$= 1$ nach dem Startvorgang initialisiert. Die Knoten gehen dann in einen alternierenden Modus, bestehend aus INQUIRY und INQUIRY-SCAN. Wenn ein Knoten x eine Verbindung mit Knoten y eingeht, vergleichen beide Knoten ihre Variable VOTES. Ist VOTES(x)>VOTES(y) wird Knoten x der Gewinner, sonst Knoten y. Ist VOTES(x)==VOTES(y) gewinnt derjenige Knoten, der eine größere Bluetooth-Adresse

besitzt. Sei oBdA Knoten x der Gewinner, so ist VOTES$(x)+ =$ VOTES(y). Existieren n Knoten bei der Formierung, so ist nach n-1 VOTES-Abgleichen der Zentralknoten bestimmt und alle anderen Knoten wechseln in den PAGE-SCAN Zustand. Ein Knoten weiß, dass er Zentralknoten ist, durch Ablauf des Timers ALT_TIMEOUT, der zusammen mit VOTES anfangs gesetzt wird und bei jeder gewonnenen Kommunikation zurückgesetzt wird. Durch Experimente wurde ein Wert von 2527,223 ms ermittelt.

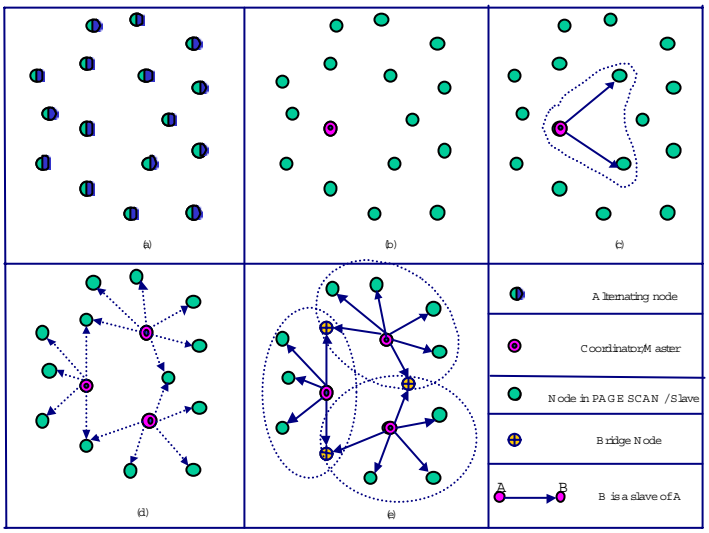
In der zweiten Phase weiß der Zentralknoten nun, wie viele Knoten das Scatternetz formen wollen. Bei mehr als sieben Knoten kann kein einziges Piconetz mehr geformt werden, so dass über Brückenknoten mehrere Piconetze das Scatternetz bilden müssen. Diese werden ebenso von dem Zentralknoten aufgrund seines Gesamtwissens ausgewählt. [SBTL01] gehen davon aus, dass eine maximale Anzahl von 36 Knoten ein vollständiges Scatternetz formen. Denn durch $P = \lceil \frac{17-\sqrt{289-8N}}{2} \rceil, \quad 1 \leq N \leq 36$ wird die minimale Anzahl der Master berechnet. Nach dieser Zuweisung hat der Koordinator nun eine Verbindungsliste bestehend aus (SLAVELIST(x), BRIDGELIST(x)), in denen alle Identitätsnummern und Uhren abgelegt sind. Am Ende der zweiten Phase verbindet sich der Zentralknoten nun mit den ausgewählten Master der zu erstellenden Piconetze und übermittelt ihnen die Verbindungsliste. Danach trennt er die Verbindung zu ihnen.

Die Master verbinden sich nun aufgrund der ihnen übermittelten Listen zu den anderen Knoten. Wird ein Knoten als Brückenknoten ausgewählt, wartet er sofort auf den Verbindungswunsch des anderen Masters und versendet dann ein CONNECTED zu beiden Master. Hat ein Master alle CONNECTED Bestätigungen erhalten, ist die Verbindung des Netzes hergestellt und der Algorithmus terminiert.

2.2 Bildung von Scatternetzen mittels Baumstruktur

2.2.1 Einleitung

Ein Algorithmus, der Piconetze zu einer Baumstruktur verknüpft, ist der von [TMGB01] vorgestellte Tree Scatternet Formation (TSF) Algorithmus. Aufgrund der Tatsache, dass die bislang verwendeten Routingalgorithmen einen enormen Overhead an Daten versenden sowie um die interne Kommunikation zu vereinfachen, wurde auf diese Struktur zurückgegriffen. Des weiteren wird so die Anzahl an Verknüpfungen zwischen einzelnen Knoten minimiert. Gleichwohl soll der Algorithmus einerseits das Szenario abdecken, dass der Verbindungswunsch aller oder sehr vieler Knoten gleichzeitig bzw. in naher Zeitdistanz auftritt (bspw. bei Konferenzen), andererseits, dass sich einzelne Knoten an ein bestehendes Netz anhängen. Bei all diesen Vorgängen soll dabei stets die Baumstruktur erhalten bleiben.



Quelle: [SBTL01].

Abbildung 9: BTCP für N=16 Knoten. (a)Start der Phase 1: Alle Knoten beginnen alternierend ihre Nachbarschaft zu erkunden. (b)Am Ende der Phase 1 wird der Koordinator ausgewählt. Für N=16 berechnet der Koordinator P=3 Master und wählt Master, Brücken und Slaves aus. (c)Phase 2: Der Koordinator formt das temporäre Piconetz mit den ausgewählten Master, um ihnen die Verbindungsliste zu übersenden. (d)Phase 3: Jeder Master verbindet sich mit den zugeteilten Knoten. (e)Das entstandene Scatternetz.

2.2.2 Algorithmus

Zu Anfang des Algorithmus ist jeder Knoten entweder Teil eines vorhandenen Baums oder ist freier Knoten. Jeder Knoten des Netzes hat zwei Zustände, FORM und COMM. FORM, welches zwei Unterzustände FORM:INQUIRY und FORM:INQUIRY-SCAN besitzt, dient dem Aufbau eines Bluetooth-Link eines Knoten zu einem anderen Teilbaum. Der COMM-Zustand dient der Kommunikation innerhalb eines bestehenden Links.

Um eine möglichst homogene Baumstruktur herzustellen, werden den Wurzelknoten eines jeden Teilbaums stärkere Bildungskompetenzen eingeräumt. Die Zeit, in der ein Knoten in einem der beiden Zustände verweilt, bestimmt eine Zufallsfunktion, welche im FORM- und COMM-Zustand von der Initialisierungszeit eines Links $T_{\text{inq}} = 2 \cdot T_{\text{sync}} + T_{b_0}$ und einem festen Parameter D abhängt. Im COMM-Zustand wird weiterhin durch die Beschäftigkeit f_{comm} eines Knoten innerhalb der Kommunikation die Verweildauer bestimmt. Die Zufallsfunktion spielt eine wichtige Rolle in dem Formierungsprozess, da erreicht werden soll, dass sich Teilbäume zu einem Gesamtbaum verbinden. [TMGB01] implementieren f_{comm} als Funktion über das Alter eines Knoten seit Eintritt in das Scat-

ternetz und über den Anteil an Kindern in dem Baum[10].

$$f_{\text{comm}} = \begin{cases} 0 & : \quad \text{wenn freier Knoten} \\ d & : \quad \text{nicht freier Knoten \& Alter} < \epsilon \\ Ad & : \quad \text{nicht freier Knoten \& Alter} > \epsilon,\ A > 1 \end{cases} \tag{1}$$

Zur Verbindungsherstellung im Rahmen einer Baumstruktur wechselt jeder Knoten zu Beginn seinen Status, d.h. von INQUIRY zu INQUIRY-SCAN und vice versa. Empfängt ein Knoten ein INQUIRY und ist in INQUIRY-SCAN wird sofort ein Link hergestellt. Dabei wird der Master Root und der Slave wird Blattknoten bzw., wenn es sich ebenfalls um einen Root eines Teilbaumes gehandelt hat, wird der gesamte Teilbaum dem Master/ Root unterstellt. Pakete werden dann auch in dieser Struktur weitergeleitet. Um Schleifen zu verhindern, werden drei Regeln festgelegt:

1. Freie Knoten können nur mit freien Knoten oder mit Nicht-Wurzelknoten Links herstellen. Dabei werden sie im zweiten Fall automatisch Slave.

2. Wurzelknoten von Bäumen mit mehr als einem Knoten können nur zu anderen Wurzelknoten Verbindungen wie bereits beschrieben herstellen.

3. Nicht-Wurzelknoten versuchen nicht größere Bäume mit nicht freien Knoten zu bilden.

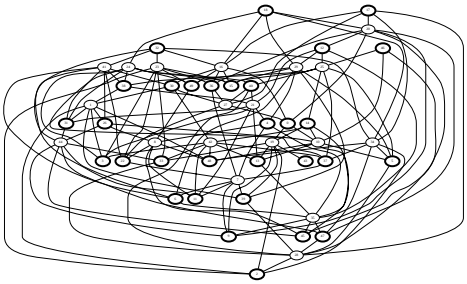
Die Aufrechterhaltung der Baumstruktur bei Verlassen von einzelnen Knoten wird ebenso gewährleistet. Dabei werden zwei Fälle unterschieden: ein Master verliert die Verbindung zu einem Slave und ein Slave verliert die Verbindung zu einem Master. Im ersten Fall muss der Master lediglich überprüfen, ob er nun freier Knoten ist. Ein Slave muss sein Alter auf Null setzen (siehe Gleichung (1)). Blattknoten werden freie Knoten, innere Knoten werden Wurzelknoten des Teilbaums.

Letztlich wird sichergestellt, dass es nicht zu der Situation kommen kann, dass Wurzelknoten, die sich zusammenschließen wollen, bereits sieben Slaves besitzen und somit keine weitere Verbindung eingehen können. Es wird hier vor Erreichen der Maximalanzahl an Slaves ein Slaveknoten zu einem Rootknoten gemacht und ehemaliger Master und Slave wechseln ihre Rollen.

[10] Dies entspricht etwa der Beschäftigkeit, da viele Kinder in einem Baum viel Kommunikation hervorrufen.

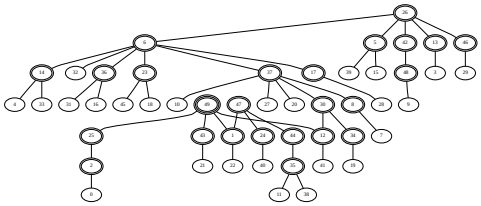
2.2.3 Simulationsresultate

Eine Evaluation des Algorithmus wurde durch Simulation mittels des Netzwerksimulators ns-2[11] unter Verwendung der Bluetooth-Erweiterung von IBM[12] durchgeführt. Dabei wurde der entwickelte Algorithmus mit einem probabilistischen Modell verglichen (siehe zum Vergleich Probabilistisches Modell (PROB) vs. Tree Scatternet Formation (TSF) Abbildungen 10 und 11).



Quelle: [TMGB01].

Abbildung 10: Ein Scatternetz mit 50 Knoten erzeugt mit PROB



Quelle: [TMGB01].

Abbildung 11: Ein Scatternetz mit 50 Knoten erzeugt mit TSF

Durch die Simulation ließ sich zeigen, dass der Verbindungsaufbau, durchschnittlich 3 Sekunden, dem probabilistischen Ansatz (PROB) überlegen ist. Die Bildung eines einzigen Scatternetz aus den diversen Bäumen gelingt im Gegensatz zu PROB immer. Jedoch zeigt sich, dass die Zeit des Scatternetzaufbaus stark mit Anzahl der Knoten zunimmt. So ist bis zu einer Anzahl von 40 Knoten eine ähnliche Aufbauzeit von ≈ 30 s zu erkennen, ab 50 Knoten aber benötigt TSF mehr als die doppelte Zeit (≈ 70 s), um das Scatternetz aufzubauen. Die durchschnittliche Verzögerung eines Paketes von einem Knoten zu einem anderen ist bei TSF vergleichbar mit der von PROB, obwohl PROB mehrere Links im Scatternetz erstellt und so ein kürzerer Weg als in TSF entstehen kann. Die Auswahl der Routingalgorithmen beeinflusst hier die Effekte stark. Bei dem Vergleich benutzten [TMGB01] all pairs shortest-path Routinginformationen. Da aber auch ein Knoten nur für genau zwei Piconetze verantwortlich zur Durchleitung ist, bei PROB aber Verbindungsknoten in zwei und mehr Piconetzen eingebucht sein können, ist die durchschnittliche Verzögerung der Pakete bei TSF geringer.

[11] `http://www.isi.edu/vint/nsnam/.`
[12] `http://oss.software.ibm.com/developerworks/opensource/bluehoc/.`

2.3 Bildung von Scatternetzen durch Benutzung des SNIFF-Modes

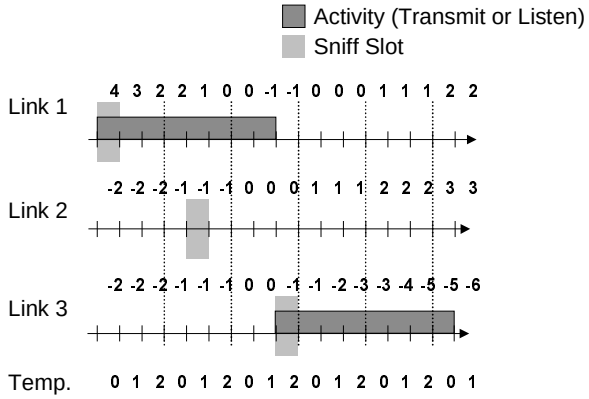
2.3.1 Einleitung

In dem Paper von [BFKM+01] werden Scatternetze gebildet, indem der jeweilige Router-Knoten mit mehreren Mastern verbunden sein kann. Jedoch immer nur zu einem eine aktive Verbindung hält. Zu den anderen Mastern wird der Zustand von aktiv auf sniff gesetzt. Möglich wird die gemeinsame Verbindung zu mehreren Mastern dadurch, dass Bluetooth frequency hopping benutzt, wobei die entsprechenden Sequenzen abhängig von den Uhren der Master sind. Dadurch kann gleichzeitig die Bandbreite des Scatternetzes in Vergleich eines einzelnen Piconetzes gesteigert werden. Der Routerknoten hört dann die Mastersendungen ab, die durch eine sniff-slot Zeit getrennt verschickt werden.

Die unterschiedlich laufenden Uhren der Master, die die frequency-hopping-Sequenz bestimmen, werfen allerdings auch das Problem auf, das keine einheitlichen slot-Grenzen bestehen. Wenn also der Routingknoten das Netz gewechselt hat, muss er mindestens bis zu dem Beginn eines geraden slots warten. Die so maximal verstrichene Zeit von zwei slots, in der keine Kommunikation stattfinden kann, wird *guard time* genannt. Um eine möglichst hohe Bandbreite zu erreichen, sollte deshalb möglichst selten das Netz gewechselt werden, ohne allerdings die Verzögerung der Pakete zu groß werden zu lassen. Des weiteren besteht die Gefahr, dass die Uhren der Master mit unterschiedlicher Geschwindigkeit laufen. Bluetooth erlaubt eine maximale Abweichung von ± 20 ppm (parts per million) bzw. 250 ppm während des Stromsparmodus. Daraus folgt, dass zwei Piconetze bereits eine maximale Abweichung von 40 ppm haben können. Um dies zu verhindern, muss eine permanente Beobachtung der Uhrenabweichung sowie eine Anpassung dieser erfolgen.

2.3.2 Kreditbasierte Scatternetz-Planung

Die eigentliche Abwicklung der Scatternetz-Formierung nennen [BFKM+01] *Kreditbasierte Scatternetz-Planung*. Dazu werden *„presence points“* definiert, welche den Startpunkt der Inter-Piconetz-Kommunikation bilden. Um dieses Verfahren zu vereinfachen, werden als presence points die sniff slots verwandt. Wenn sichergestellt ist, dass ein Knoten weiß, ob ein Netzknoten demselben Piconetz angehört, kann die Kommunikation mit diesem beginnen. Falls nicht, muss ein anderer presence point abgewartet werden, um wieder festzustellen, ob es von demselben Piconetz stammt. Es muss aber darauf geachtet werden, nicht zu viel Bandbreite zu verbrauchen. Um nicht eine statische Formierung vorzugeben, wird diese online für jede Kommunikationsperiode durchgeführt, so dass simple



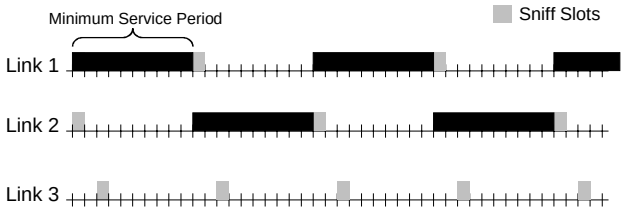
Quelle: [BFKM+01].

Abbildung 12: Credit scheme example

Kalkulationen durchgeführt werden müssen, wie lange ein Knoten einem bestimmten Piconetz zugeordnet bleiben soll. Aufgrund dieser Tatsache wird lediglich die Abweichung der presence points der Master beobachtet.

Für die richtige Funktionsweise spielt die Frage nach der Synchronisation eine entscheidende Rolle. Wann muss ein sniff-Event abgebrochen werden, um auf einen neuen sniff slot aufzuspringen? Deshalb wird jedem verbundenen Knoten eine Priorität zugewiesen. Wenn ein Knoten unfair behandelt wird, erhöht sich damit seine Priorität. Da für jeden verbundenen Knoten eine Priorität berechnet werden muss, bekommt auch jeder verbundene Knoten einen Eintrag in einer Liste. Dabei entspricht jeder Kredit genau einem Slot. Wenn ein Knoten den Link benutzt, um Daten zu verschicken oder zu empfangen, sinkt die jeweilige Priorität. Um eine konstante Anzahl von Krediten innerhalb des Systems zu halten, wird eine weiter temporäre Liste generiert, die sobald ein Kredit eines Knotens abgezogen wird, ein Kredit erhält. Die gerechte Verteilung wird nun so erreicht, dass sobald die temporäre Liste genau so viele Kredite wie verbundene Knoten enthält, diese sofort auf alle verbundenen Geräte gleich aufgeteilt werden. Die aktuelle Verbindung wird unterbrochen, sobald ein anderes verbundenes Gerät eine höhere Priorität als das sendende bzw. empfangende besitzt. Der Vorgang wird unterbrochen und der nächste Slot wird als sniff slot verwendet (siehe Abbildung 12). Das regelmäßige Senden von POLL-NULL-Nachrichten ist von dieser Regelung jedoch nicht betroffen. Ebenso bricht ein verbundenes Gerät, das seine poll-Grenze erreicht, also alle T_{poll} slots, das sniff event ab. Sobald ein sniff event vorzeitig abgebrochen worden ist, wird sofort der nächste slot als sniff slot verwandt.

Probleme können nur dadurch auftreten, dass es zu häufig zu Verbindungswechseln kommt, da dadurch die guard time ein sehr störender Faktor wäre. Gelöst wird dieses Problem



Quelle: [BFKM+01].

Abbildung 13: Starvation scenario

durch Garantierung einer Mindestanzahl an slots, welche ungestört zur Kommunikation benutzt werden können. [BFKM+01] haben sich so aber ein neues Problem geschaffen, welches sie „Verhungern“ (starvation) von verbundenen Knoten nennen. Durch diese Mindestgarantierung nämlich können sniff slots anderer Knoten innerhalb der garantierten Zugriffszeit eintreten, die aber keine Kommunikationszuteilung zur Folge haben. So kann es passieren, dass obwohl die Kreditpriorität ständig zunimmt, keine sniff slots dieses Knoten genommen werden (siehe Abbildung 13). Lösung dieses Problems bringt die Zuteilung unterschiedlicher Startkredite auf die verbundenen Knoten sowie die Einführung einer Verbindungswechselschwelle $N_{\text{switch\_th}}$, welche als untere Grenze bzw. $2 \cdot N_{\text{switch\_th}}$ als obere Grenze der sniff events dient[13]. Bei dieser Zuordnung der Kredite, gleichmäßig auf alle verbundenen Knoten, wird aber nicht berücksichtigt, ob evtl. alle Knoten überhaupt senden wollen. Um eine bessere Aufteilung der Kredite zu gewähren, werden diese umverteilt. Des weiteren wird nicht die absolute Anzahl an Krediten betrachtet, sondern letztlich nur die Differenz zwischen den Knoten. Eingeleitet wird die Umverteilung durch eine POLL-NULL-Sequenz, da hier der Datenverkehr als sehr gering angenommen werden kann.

2.3.3 Simulationsresultate

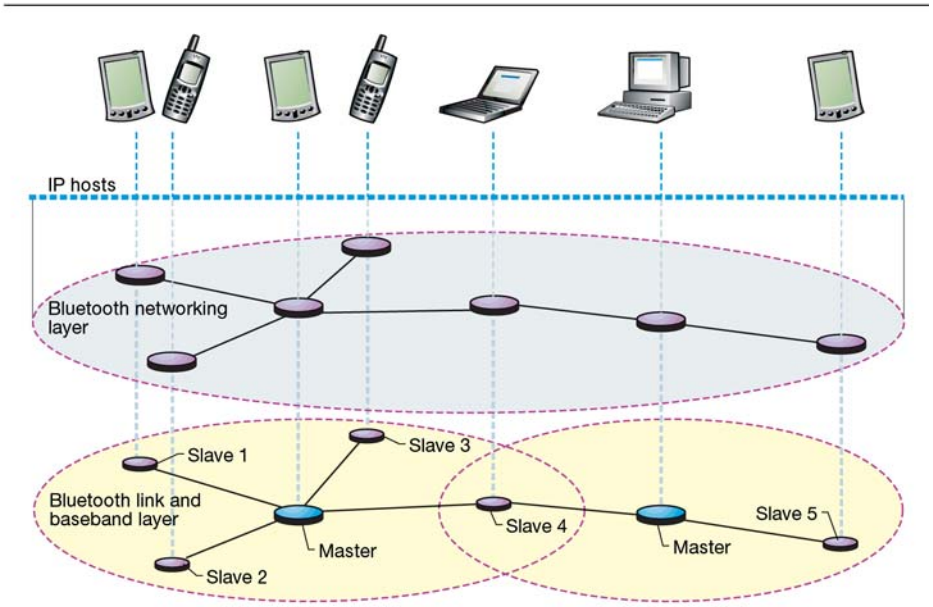
Für die Simulation des Algorithmus verwenden [BFKM+01] den Netzwerksimulator ns-2. In ihrer Simulation existiert ein Scatternetz, bestehend aus sechs Knoten, welche drei Piconetze bilden. Ein Knoten nimmt die Rolle eines Hubs ein. Es kann gezeigt werden, dass ein maximaler Datendurchsatz von $800s^{-1} \cdot 23$ Byte $\cdot\, 8 = 147.2$ kbit/s erreicht wird. Demzufolge sendet jeder äußere Knoten mit 29.44 kbit/s. Stoppen die Knoten nacheinander die Datenübertragung und Kredite werden nicht umverteilt, wird die freie Kapazität nicht auf die restlichen Knoten verteilt. Im Gegensatz dazu wird bei Umverteilung der Kredite die Gesamtkapazität immer auf alle sendenden Knoten verteilt.

[13] Die Einführung der $N_{\text{switch\_th}}$ hat zur Folge, dass jeder verbundene Knoten die gleiche durchschnittliche Kommunikationszeit zur Verfügung stehen hat, nämlich $\frac{n}{n-1} \cdot N_{\text{switch\_th}}$.

3 Routing in Scatternetzen/ Inter-Piconetzen

3.1 Unterschiede zu IP-Routing

Um Pakete in einem Scatternetz zwischen zwei Knoten übere mehrere hops hinweg auszutauschen, ist es nicht nur notwendig, eine funktionierende Scatternetzstruktur zu haben. Denn auch mit einer solchen existieren eine Vielzahl von Fragen bezüglich der genauen Wegewahl der einzelnen Pakete, dem Routing. Normalerweise wird Routing auf Schicht 3 durchgeführt. Wenn Bluetooth-Geräte auch im Internet verwendet werden, müssen diese IP-fähig sein. Deshalb könnte auch hier unter IP auf Schicht 3 geroutet werden. [FJL00] fühen aber als Gegenargument an, dass durch die Mobilität der Geräte, die die Zuweisung ihrer IP-Adressen durch DHCP-Server erhalten, welche aber wiederum mehrere hops entfernt stehen können, die Zuweisung nicht erfolgen kann. Denn IP-Router lassen Pakete ohne gültige Absenderadresse nicht zu und verwerfen diese. Deshalb sollte das Routing auf einer Zwischenschicht unter der Schicht 3 mit in die Formierungsfunktionen eingebunden werden. Diese Zwischenschicht sollte nahe mit dem Eingehen und Lösen von Bluetooth-Verbindungen agieren sowie ein broadcast-segmentenähnliche Schnittstelle zu IP bieten.



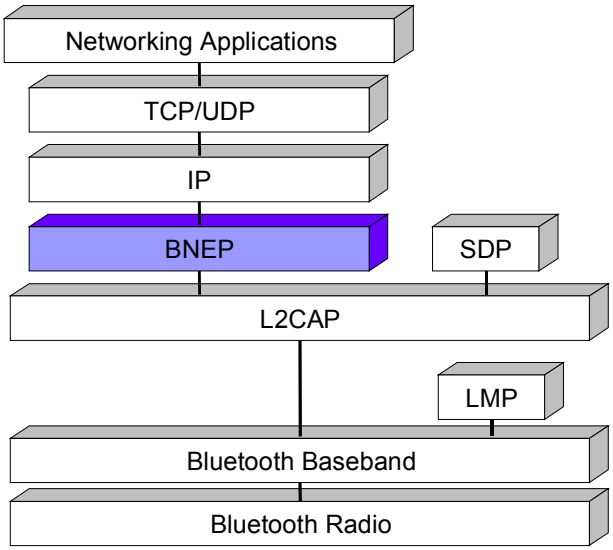
Quelle: [FJL00].

Abbildung 14: Ein Scatternetz, bei dem Pakete auf Netzwerkebene geroutet werden.

3.2 Bluetooth Network Encapsulation Protocol

Das Bluetooth Network Encapsulation Protocol (BNEP) von [JKKG01] stellt eine Schnittstelle zu IP ähnlich der des Ethernet zur Verfügung. BNEP wird auf das Logical Link and

Control Adaption Protocol (L2CAP) aufgesetzt, welches zuständig für die Segmentierung Pakete höherer Schichten ist (siehe Abbildung 15). Da L2CAP keine gesamtnetzwerkweite Adressierung kennt, benutzt BNEP die eindeutige BD_ADDR der Bluetooth-Geräte und stellt damit auch einen von IP unabhängigen Dienst zur Verfügung. Das Routen kann in einem Scatternetz, das aus Geräten mit BNEP besteht, auf gleiche Weise wie in jedem anderen ad hoc Netz durchgeführt werden. Wird anstatt der IP-Adresse die BD_ADDR als Sende- und Empfangsadresse verwendet, können alle bekannten Routingalgorithmen für mobile Netze verwendet werden. Hier bieten sich gerade durch den geringeren Datenoverhead reaktive und auch positionsbasierte Routingalgorithmen an. Durch das Wissen der Routingknoten, welche anderen Knoten sich in ihrer Umgebung aufhalten, kann BNEP ebenso zur Formierung von Scatternetzen verwandt werden.



Quelle: [JKKG01].

Abbildung 15: Der Bluetooth IP stack mit BNEP.

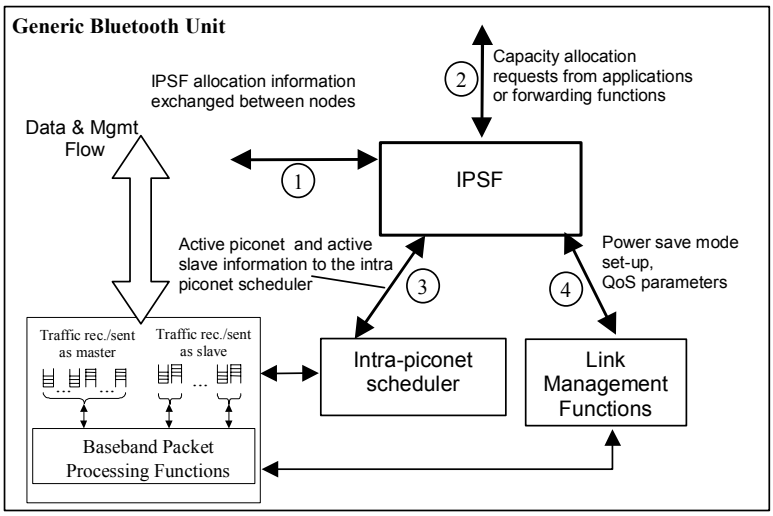
Wichtig in bezug auf das Weiterleiten von Paketen ist die faire Zuteilung der Ressourcen. [JKKG01] verwenden den *Fair Exhaustive Polling (FEP)* Algorithmus, um die Intrapikonetzzeitplanung (IRPS) durchzuführen, da dieser einfach zu implementieren ist und die gewünschte Fairness gewährleistet. Gleichwohl ist auch die Interpikonetzzeitplanung (IPSF) von enormer Wichtigkeit. IPSF muss deshalb mit folgenden Verbindungen zu anderen Bluetoothfunktionen ausgestattet sein (siehe auch Abbildung 16):

1. Austausch von Informationen über Zeitpunkte des Treffens der Master und der Interpikonetzknoten.

2. Informationen von Anwendungen, die bspw. maximale Verzögerungsbedingungen stellen.

3. Wenn der Knoten Master ist, Austausch mit IRPS zwecks Polling von Slaves.

4. Dem IPSF werden Funktionen wie die Stromsparmechanismen (SNIFF, HOLD und PARK) zur Verfügung gestellt.



Quelle: [JKKG01].

Abbildung 16: Die funktionelle Architektur der Interpikonetzzeitplanungsfunktion (IPSF) und ihre Verbindung zu anderen Funktionen.

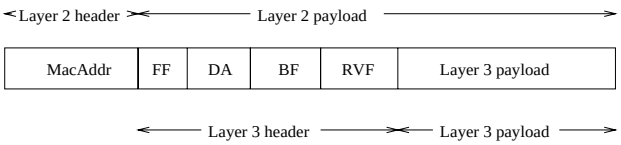
Die Interpikonetzkommunikation zweier Knoten wird dann mit Hilfe eines Rendezvous Point Algorithmus gesteuert. Zur Funktionsweise eines solchen Algorithmus siehe Seite 12.

3.3 Wegewahl mittels Routenvektoren

[BS98] gehen davon aus, dass Bluetoothnetzwerken klein, fast statisch und von kurzer Lebenszeit sind. Deshalb werden hier keine Routingalgorithmen benötigt, die nach Skalierbarkeitskriterien optimiert sind.

Für die Übertragung der Pakete wird die Paketstruktur verändert. Der Schicht 2 Datenteil wird aufgeteilt in einen Schicht 3 Header plus Schicht 3 Datenblock (siehe Abbildung 17).

FF=1 gibt an, dass ein Paket weitergeleitet werden muss, da zwei Slaves nicht direkt miteinander komunizieren können. DA bezeichnet die Empfängeradresse (Destination Address). Jedem Piconetz wird eine lokale Identifikationsnnummer (LocID) zugewiesen. LocID=000 ist dabei das eigene Netz. Der Routingalgorithmus basiert auf dem Prinzip des Dynamic Source Routing Algorithmus (DSR).



Quelle: [BS98].

Abbildung 17: Relevante Felder in einem Bluetooth-Paket.

Dazu werden SEARCH-Pakete, welche Schicht 3 Kontrollpakete darstellen, mittels Broadcast versandt. Dabei trägt der Senderknoten seine MacAddr und seine LocId in das Feld RVF ein. Jeder Knoten, der das Paket erhält, trägt seine eigene MacAddr und LocID in das Paket ein und schickt es per broadcast weiter. Dabei sieht er, wenn BF=0 gesetzt ist, dass das Paket ein Interpikonetzpaket ist bzw. bei BF=1, dass das Paket an alle Nachbarn innerhalb des Piconetztes versendet werden muss, bis das Paket sein Ziel erreicht hat. Der Zielknoten verschickt per unicast ein REPLY-Paket, welches die gesamte Route zu ihm in RVF enthält, an den ursprünglichen Sender zurück. Dieser kann anhand der RVF-Einträge Datenpakete versenden, die die genaue Route in RVF eingetragen haben.

3.4 Gesamtheitliche Optimierung des Bluetooth Stack

Da Bluetooth-Geräte abhängig von der Kapazität ihrer Baterien sind, sollten Verbindungen zwischen Knoten nur dann aufrecht erhalten werden, wenn auch Pakete versandt werden sollen. Das gleiche gilt für IP-Routen. Das bedeutet, dass untere Schichten nur dann Arbeit verrichten müssen, wenn höhere Schichten diese anfordern. Dies spricht nach Meinung von [RBS01] schon dafür, dass ein gesamtheitliche Optimierung des Stack erforderlich ist. Ein weiterer Punkt ist, dass Bluetooth-Geräte nicht nach ihrer IP-Adresse angesprochen werden sollen, sondern nach den Diensten, die sie zur Verfügung stellen. Deshalb müsste bei einer getrennten Schichtenstruktur eine untere Schicht wissen, welche Dienste eine höhere Schicht anbietet. Auch kann durch Caching verhindert werden, dass durch Broadcastnachrichten, die eine Route suchen, das gesamte Netz geflutet werden muss. Nachteilig stellt sich ein Gesamtlayer aufgrund der eingeschränkten Möglichkeiten bzgl. Modularisierung dar.

Simulationsresultate zeigen dennoch, dass ein Stack, bei dem entweder IP und Service Discovery (SD) zusammen oder IP, SD und Verbindungsherstellung zusammen eine Schicht bilden, Vorteile gegenüber Stacks besitzt, die diese Aufteilung nicht haben. So ist im Vergleich des normalen Schichtenstacks mit dem gesamtheitlichen Stack die Anzahl der versandten Nachrichten mehr als doppelt so hoch.

4 Zusammenfassung

Da Bluetooth immer noch in der Entwicklung steckt und erste Geräte erst in jüngster Zeit auf den Markt kommen, ist auch die Standardisierung der Bildung von Scatternetzen nicht vorhanden. Die Spezifikation gibt keine genaue Auskunft wie Scatternetze zu bilden sind und wie Pakete geroutet werden sollen.

Alle hier vorgestellten Algorithmen wurden nur durch Simulationsverfahren auf ihre Performanz getestet. Da auch hier zum einen keine einheitlichen Parameter gewählt worden sind, zum anderen mit verschiedenen Simulatoren gearbeitet worden ist, fällt ein Vergleich der Algorithmen untereinander äußerst schwer. Erst wenn erste Test in realer Umgebung durchgeführt worden sind, die auf die Mobilität der Geräte eingehen, lassen sich die Simulationen bestätigen oder verwerfen.

Literatur

[AFN] Ardaiz, O.; F. Freitag; L. Navarro: On Service Deployment in Ubiquitous Computing.

[AS99] Arfwedson, Henrik; Rob Sneddon: Ericsson's Bluetooth modules, Ericsson Review No. 4, 1999.

[BFKM+01] Baatz, Simon; Matthias Frank; Carmen Kühl; Peter Martini; Christoph Scholz: Adaptive Scatternet Support for Bluetooth using Sniff Mode, Proceedings of the 26th Annual Conference on Local Computer Networks, LCN 2001, Tampa, Florida, November 2001.

[BS98] Bhagwat, Pravin; Adrian Segall: A routing vector method (RVM) for routing in bluetooth scatternets, in Mobile Multimedia Communications, 1999. (MoMuC '99). 1999 IEEE International Workshop on, pp. 375-379, 1999.

[BR00] Bhagwat, Pravin; Srinivasa P. Rao: On the Characterization of Bluetooth Scatternet Topologies, Submitted for publicaton.

[Bhag01] Bhagwat, Pravin: Bluetooth: Technology for Short-Range Wireless Apps, IEEE Internet Computing, May/June 2001, pp. 96-102.

[Blue01] The Bluetooth Special Interest Group (SIG): BLUETOOTH SPECIFICATIONS, Version 1.1, http://www.bluetooth.com, 2001.

[FJL00] Frodigh, Magnus; Per Johansson; Peter Larsson: Wireless *ad hoc* networking - The art of networking without network, Ericsson Review No. 4, 2000.

[GKKJ01] Gerla, Mario; Rohit Kapoor; Manthos Kazantzidis; Per Johansson: Ad hoc networking with Bluetooth, IEEE Network Special Issue on Personal Area Networks, Sept-Oct 2001.

[Haar98] Haartsen, Jaap: BLUETOOTH - The universal radio interface for ad hoc, wireless connectivity, Ericsson Review No. 3, 1998.

[HNIJ98] Haartsen, Jaap; Mahmoud Naghshineh; Jon Inouye; Olaf J. Joeressen; Warren Allen: Bluetooth: Vision, Goals, and Architecture, Mobile Computing and Communications Review, Volume 1, Number 2, October 1998, pp. 1-8.

[Haar00] Haartsen, Jaap: The Bluetooth Radio System, in IEEE Personal Communications Volume: 7 1, February 2000, pp. 28-36.

[JKKG01] Johansson, Per; Manthos Kazantzidis; Rohit Kapoor; Mario Gerla: Bluetooth: An Enabler for Personal Area Networking, IEEE Network, September/October 2001, pp. 28-37.

[KGS00] Kalia, Manish; Sumit Garg, Rajeev Shorey: Scatternet Structure and Inter-Piconet Communication in the Bluetooth System, IEEE National Conference on Communications 2000, New Delhi, 2000.

[Kaza01] Kanzantzidis, Manthos: Locally optimal Bluetooth Scatternet formation, UCLA Computer Science WAM Lab, Technical Report No. 010033, September 2001.

[LMS01] Law, Ching; A. Mehta; K. Siu: Performance of a New Bluetooth Scatternet Formation Protocol, Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing 2001, Long Beach, California, USA, October 2001.

[MRTV00] Miklós, G.; A. Rácz; Z. Turányi; A. Valkó; P. Johansson: Performance aspects of Bluetooth scatternet formation, Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on, pp. 147-148, 2000.

[RBS01] Raman, Bhaskaran; Pravin Bhagwat; Srinivasan Seshan: Arguments for cross-layer optimizations in Bluetooth scatternets, Applications and the Internet, 2001. Proceedings. 2001 Symposium on, pp. 176-184, 2001.

[SBTL01] Salonidis, Theodoros; Pravin Bhagwat; Leandros Tassiulas; Richard LaMaire: Distributed Topology Construction of Bluetooth Personal Area Networks, Infocom 2001.

[SSM01] Son, Le Thanh; H. Schiøler; Ole Brun Madsen: Predictive scheduling approach in Inter-piconet communications, Proc. of the 4th international symposium on Wireless personal multimedia communications, Aalborg, Denmark, September 2001.

[TMGB01] Tan, Godfrey; Allen Miu; John Guttag; Hari Balakrishnan: Forming Scatternets from Bluetooth Personal Area Networks, MIT Laboratory for Computer Science, October 2001.

[Tan01] Tan, Godfrey: Interconnecting Bluetooth-like Personal Area Networks, in 1st Annual Oxygen Workshop, Gloucester, MA, 2001.

[ZS01] Zussman, Gil; Adrian Segall: Capacity Assignment in Bluetooth Scatternets - Analysis and Algorithms, CCIT Report No. 355, Center for Comunication and Information Technologies, Haifa, Israel, October 2001.