TECHNISCHE UNIVERSITÄT DRESDEN
FAKULTÄT ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Diplomarbeit

Adaption of Cluster-Tree Routing Protocol to IEEE 802.15.4/ZigBee Standard

Vorgelegt von: Thomas Liske
Geboren am: 17. Dezember 1982 in: Dresden

zur Erlangung des akademischen Grades

Diplomingenieur
(Dipl.-Ing.)

Betreuer:	Dipl.-Ing. Dimitri Marandin,
	Dipl.-Ing. Rico Radeke
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. Ralf Lehnert
Tag der Einreichung:	16. Oktober 2007

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag an der Fakultät Elektrotechnik und Informationstechnik eingereichte Diplomarbeit zum Thema

Adaption of Cluster-Tree Routing Protocol to IEEE 802.15.4/ZigBee Standard

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 16. Oktober 2007

Thomas Liske

Kurzfassung

In dieser Arbeit wurde das Cluster-Tree-Routing-Protokoll auf dem ZigBee-Protokollstapel unter Verwendung eines gesonderten Adressierungsschemas umgesetzt. Durch das Betreiben der IEEE 802.15.4 MAC-Schicht im so genannten *beacon enabled mode* ist es möglich, die Knoten mit definierten Wach- und Schlafzyklen zu betreiben und so den Energieverbrauch zu minimieren. Für das daraus entstehende *Beacon-Scheduling-Problem* wurden drei unterschiedliche Lösungsansätze entworfen.

Das Protokoll wurde für Simulationszwecke im Netzwerksimulationsprogramm ns-2 implementiert. Die Fehler der bereits vorhandene Implementierung der IEEE 802.15.4 MAC-Schicht wurden entsprechend dem Standard, soweit möglich, korrigiert. Die zugehörige Simulationsdarstellung im Programm nam wurde durch zusätzliche Daten aus der Netzwerkschicht des Protokolls ergänzt. Eine endgültige Simulation und Evaluierung kann auf Grund der entdeckten konzeptionellen Fehler im Programm ns-2 jedoch nicht präsentiert werden.

Abstract

The objective of this master theses was the adaption of the cluster tree routing protocol to the IEEE 802.15.4/ZigBee stack. The adaption was done with the use of a specific addressing scheme. The MAC layer is running in *beacon enabled mode*. Thus it is possible to define wakeup and idle cycles for the nodes to minimize energy consumption. Three approaches where developed to solve the resulting *beacon scheduling problem*.

For simulation purpose the protocol has been implemented into the network simulator program ns-2. Errors of the existing IEEE 802.15.4 MAC layer implementation were corrected if possible. The simulation output for the network animator program nam was expanded by additional informations of the network layer. A final simulation and evaltion could not be conducted due conceptional errors in ns-2.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AODV	Ad-hoc On-demand Distance Vector
APL	Applicaton Layer
APS	Application Support Sublayer
ARP	Address Resolution Protocol
BI	Beacon Interval
BO	Beacon Order
CBR	Constant Bit Rate
CH	Cluster Head
CRUISE	CReating Ubiquitous Intelligent Sensing Environments
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DD	Designated Device
ED	Energy Detection
FFD	Full Functional Devices
GTS	Guaranteed Time Slots
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Medium Access Control
MLME	Medium Access Control Sub-Layer Management Entity
nam	Network Animator
NLDE	Network Layer Data Entity
NLME	Network Layer Management Entity
NP	Non-deterministic Polynomial-time
ns-2	Network Simulator 2
NWK	Network
PHY	Physical
RFD	Reduced Functional Devices

SD ..... Superframe Duration
SO ..... Superframe Order
Tcl ..... Tool command language
UML ..... Unified Modeling Language
VBR ..... Variable Bit Rate
WLAN ..... Wireless Local Area Network
ZDO ..... ZigBee Device Object

1 Einleitung

Mit dem Standard IEEE[1] 802.15.4 wurde 2003 die erste Version eines Standards für Funknetze veröffentlicht. Dieser Standard bietet nur geringe Datenraten im Vergleich zu anderen existierenden WLAN[2]-Standards. Anwendungsgebiete sind hier Bereiche, die Anforderungen an eine hohe Verfügbarkeit, hohe Sicherheit sowie ein geringes Datenaufkommen besitzen. Ein weiterer wichtiger Aspekt ist ein möglichst geringer Energieverbrauch in Verbindung mit batteriebetriebenen Netzknoten. Der Standard bietet eine Reihe von Funktionen, die sich für den Aufbau von maschen-, stern- oder baumartige Netztopologien eignen.

Die ZigBee-Spezifikation basiert auf dem IEEE 802.15.4 Standard und definiert die Funktion von Netzwerk- und Anwendungsschicht. Die ZigBee-Spezifikation wird von der ZigBee-Allianz entwickelt. Dieser Zusammenschluss von verschiedenen Industriepartnern hat das Ziel, eine Basis für kompatible Geräte zu schaffen.

Als Anwendungsgebiete sind hier unter anderem Sensoren von Alarmanlagen und *Home Automation*-Systeme sowie verteilte Überwachungsaufgaben wie Temperaturmessung oder Rauchdetektierung zu nennen.

Das Verwenden von Baumtopologien ermöglicht bei geschicktem Adressieren ein sehr einfaches Routing. Die Knoten müssen nur erkennen, ob ein zu sendender Datenrahmen an einen ihrer Kind- oder den Vaterknoten geroutet werden muss. Diese Topologie scheint für stationäre und langlebige Netze besonders geeignet, während für Netze mit mobilen und wechselnden Knoten eine Maschentopologie geeigneter scheint. Wenn ein Knoten in einer Baumtopologie das Netz verlässt, müssen dessen Kindknoten eine neue Verbindung zu dem Netz aufbauen. In einer Maschentopologie ist dies nicht zwangsläufig nötig, da Knoten mehrere Verbindungen zum Netz herstellen können.

[1] Institute of Electrical and Electronics Engineers
[2] Wireless Local Area Network

Die MAC[3]-Schicht des IEEE 802.15.4 Standards bietet außerdem die Möglichkeit der Verwendung von definierten Wach- und Schlafzyklen im so genannten *beacon enabled mode*. Dieser Modus kann jedoch nur in Stern- oder Baumtopologien verwendet werden.

In vorangegangenen Arbeiten wurden bereits Netze mit Baumtopologien auf Basis von IEEE 802.15.4 umgesetzt. Dabei basiert die Umsetzung aber weder auf den von der ZigBee-Spezifikation definierten Rahmenformaten, noch wurden die speziellen Funktionen der MAC-Schicht für eine solche Topologie beachtet.

Ziel dieser Arbeit war es deshalb, das Cluster-Tree-Protokoll an die ZigBee-Spezifikation unter Beachtung des IEEE 802.15.4 Standards anzupassen. Dabei sollten, soweit möglich, existierende Funktionen und Rahmentypen verwendet werden. Änderungen sollten nur im Rahmen der ZigBee-Spezifikation erfolgen, Änderungen am IEEE 802.15.4 als Standard kamen vorerst nicht in Betracht.

Zur Bewertung dieser Umsetzung sollte das neue Protokoll im Netzwerksimulationsprogramm ns-2 implementiert und evaluiert werden.

Gliederung

Diese Arbeit ist in folgende abgegrenzte Kapitel unterteilt. Zu Beginn werden im Kapitel 2 die grundlegenden Standards, Spezifikationen und Arbeiten, auf denen aufgebaut wird, erläutert. Das Kapitel 3 behandelt die Definition des in dieser Arbeit entwickelten Cluster-Tree-Protokolls auf ZigBee-Basis. Die Implementierung des Protokolls in ns-2 wird in Kapitel 4 abgehandelt. Überlegungen zur Simulation und Bewertung der Ergebnisse sind im Kapitel 5 beschrieben. Das letzte Kapitel 6 fasst die Arbeit zusammen und gibt Anregungen für weitere mögliche Entwicklungen und Verbesserungen.

Begriffsdefinition

Zu Beginn sollen einige Begriffe und ihre Unterschiede definiert werden, um die folgenden Kapitel korrekt zu verstehen. Die Abgrenzung der Begriffe ist in Abbildung 1.1 dargestellt.

[3]Medium Access Control

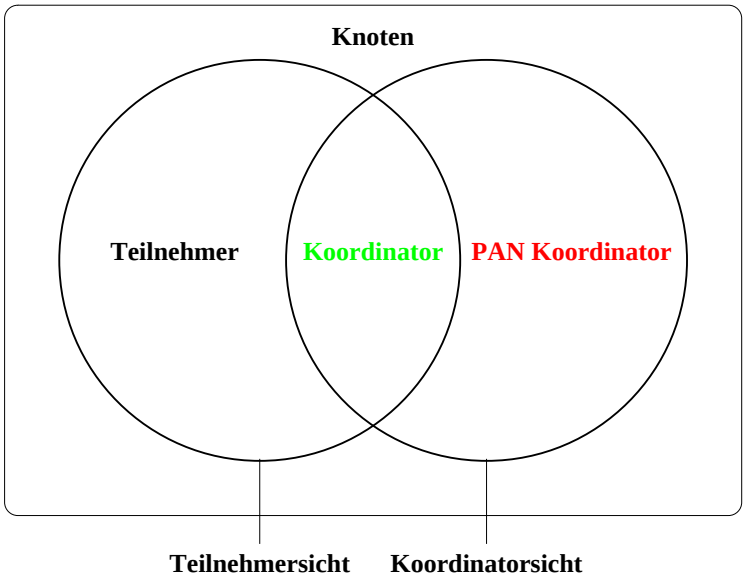

Abbildung 1.1: Abgrenzung der Begriffsdefinitionen

Unter dem Begriff *Knoten* versteht man einen beliebigen Netzknoten. Knoten können zwei unterschiedliche Funktionalitäten besitzen: die *Teilnehmersicht* und die *Koordinatorsicht*. Unter Teilnehmersicht werden dabei alle Funktionen verstanden, die benötigt werden, um in einem Baum als Kindknoten zu agieren. Unter Koordinatorsicht fallen alle Funktionen eines Knoten, um Kindknoten zu verwalten.

Knoten, die nur den Funktionsumfang der Teilnehmersicht besitzen, heißen hier Teilnehmer. Teilnehmer können nur Blattknoten werden, Blattknoten müssen aber keine Teilnehmer sein.

Ein Knoten, der sowohl Teilnehmer- als auch Koordinatorsicht beherrscht, wird als *Koordinator* bezeichnet. Der Wurzelknoten einer Baumstruktur wird als *PAN-Koordinator* bezeichnet und besitzt nur die Funktionen der Koordinatorsicht.

Bei der Adressierung unterscheidet man zwischen *Knoten-ID* und *Knotenadresse*. Die Knoten-ID ist die logische Adresse der Netzwerkschicht. Die Knotenadresse ist die eineindeutig vergebene feste Adresse der MAC-Schicht des Gerätes.

2 Grundlagen

Dieses Kapitel liefert eine Übersicht über die dieser Arbeit zu Grunde liegenden Standards und Spezifikationen. Außerdem werden die an der TU Dresden bereits abgeschlossenen thematisch verwandten Arbeiten angeführt.

2.1 IEEE 802.15.4

Der Standard 802.15.4 des IEEE ist ein Standard für ein so genanntes LR-WPAN[1]. Er spezifiziert sowohl die Funktion der PHY[2]- als auch der MAC-Schicht in einem solchen Netz.

Der Standard wurde für Netze mit geringem Energieverbrauch der Teilnehmer sowie mit hoher Verfügbarkeit und Sicherheit konzipiert. Die zur Verfügung gestellten kryptografischen Funktionen des Standards werden hier jedoch nicht näher betrachtet, da sie keinen direkten Einfluss auf den Netzaufbau haben.

Bei den Knotentypen wird zwischen den so genannten RFD[3] und den FFD[4] unterschieden. FFD-Knoten decken alle Funktionen des Standards ab, RFD-Knoten sind nur auf die wichtigsten Funktionen zur Teilnahme an einem Netz begrenzt. RFDs können im Falle eines Cluster-Tree-Netzwerkes nur die Position von Blattknoten einnehmen.

Es folgt eine kurze Übersicht über die durch den Standard spezifizierte PHY- und MAC-Schichten.

[1] Low-Rate Wireless Personal Area Network
[2] Physical
[3] Reduced Functional Devices
[4] Full Functional Devices

2.1.1 PHY-Schicht

Die PHY-Schicht ist für die drei Frequenzbänder in Tabelle 2.1 spezifiziert. Neben dem weltweit zulässigen 2,4 GHz-Band wurden die zwei regional verwendeten Bänder 868 MHz für Europa und 902 MHz für Nordamerika in die Spezifikation aufgenommen.

Frequenzband (MHz)	Modulation	Bitrate (kb/s)	Symbolrate (ksym/s)	Anzahl Kanäle
2400-2483,5	Q-QPSK	250	62,5	16
902-928	BSK	40	40	10
868-868,6	BSK	20	20	1

Tabelle 2.1: Frequenzbänder IEEE 802.15.4

Da das 2,4 GHz-Band bereits durch viele andere Anwendungen wie WLAN und Bluetooth belegt ist, wurde in der 2006 erschienenen überarbeiteten Version des Standards [IEE06] die Datenrate des 915 MHz und des 868 MHz-Bandes optional auf bis zu 250kb/s angehoben. Damit können Netze auf die beiden anderen Bänder bei gleicher Datenrate ausweichen. Die auf MAC-Schicht verwendeten logischen Kanalnummern sind dabei im Standard auf die Frequenzbänder, wie in Abbildung 2.1 dargestellt, verteilt.

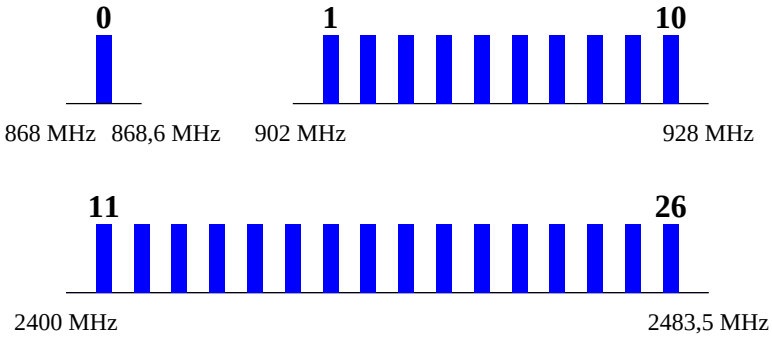

Abbildung 2.1: Verteilung der Kanalnummern auf die Frequenzbänder

2.1.2 MAC-Schicht

Die MAC-Schicht kann entweder im *non-beacon enabled mode* oder im *beacon enabled mode* betrieben werden.

Im Falle des *non-beacon enabled mode* wird das Zugriffsverfahren CSMA/CA[5] verwendet.

Im *beacon enabled mode* kommt eine *Superframe*-Struktur zum Einsatz. Als Zugriffskontrolle wird dann *slotted CSMA/CA* verwendet. Dies ermöglicht es die PHY-Schicht in den Schlafmodus zu versetzen, wenn keine Daten ausgetauscht werden.

Superframe-Struktur

Im *beacon enabled mode* sendet jeder Koordinator periodisch ein so genannte *Beacons*, die von so genannten *Superframes* gefolgt werden, aus (siehe Abbildung 2.2). Der Beacon enthält Informationen über den aussendenden Knoten und das zugehörige Netz, im *Superframe* können Daten zwischen dem Koordinator und seinen Teilnehmern ausgetauscht werden. Nach Ende der SD und Beginn des nächsten *Beacon* wechselt die PHY-Schicht in den Schlafmodus, um Energie zu sparen.

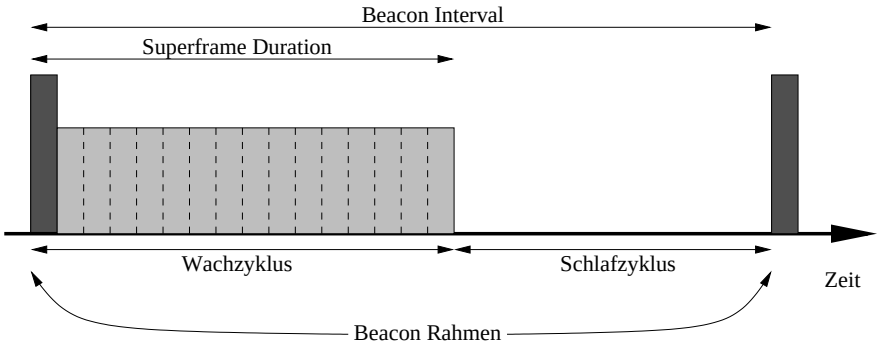

Abbildung 2.2: Definition *Superframe*-Struktur

Der *Beacon*-Rahmen enthält die in Tabelle 2.2 aufgelisteten Informationen über den *Superframe* und den Koordinator.

[5] Carrier Sense Multiple Access with Collision Avoidance

Feld
Superframe specification
GTS fields
Pending address fields
Beacon payload

Tabelle 2.2: Felder im *Beacon*-Rahmen

Superframe specification

Die Abbildung 2.3 zeigt den Aufbau der *Superframe specification*.

Bitbreite	4	4	4
Feld	Beacon Order	Superframe Order	Final CAP Slot

Bitbreite	1	1	1	1
Feld	Battery Life Extension	reserviert	PAN Coordinator	Association Permit

Abbildung 2.3: Aufbau *Superframe specification*-Feld

Sie enthält Informationen über die Abstände zwischen den Beacons und über die Länge des Superframes. Diese Informationen werden als BO[6] und SO[7] angegeben. Aus den beiden Parametern berechnet sich dann die Dauer des BI[8] nach Gleichung 2.1.

$$BI = aBaseSuperframeDuration * 2^{BO} \tag{2.1}$$

Die Dauer der SD[9] berechnet sich analog nach der Gleichung 2.2.

$$SD = aBaseSuperframeDuration * 2^{SO} \tag{2.2}$$

[6]Beacon Order
[7]Superframe Order
[8]Beacon Interval
[9]Superframe Duration

Die BO ist ein 4 bit-Wert und darf im Bereich $0 \leq BO \leq 15$ liegen. Für die SO ist der Bereich von $0 \leq SO \leq BO$ erlaubt. Die Konstante *aBaseSuperframeDuration* ist im Standard mit 960 sym festgelegt.

Zusätzlich ist in der *Superframe specificaton* die Anzahl der Slots für den Zugriff nach *slotted CSMA/CA* im Feld *Final CAP Slot* festgelegt.

Durch einen Wert von 1 im *Battery Life Extension* Feld wird angegeben, dass sich der Koordinator nach dem Senden des Beacons in den Schlafzustand versetzt.

Das Bit *PAN Coordinator* enthält den Wert 1, falls der sendende Knoten ein PAN-Koordinator ist, sonst den Wert 0.

Wenn der Koordinator Assoziierung mit Knoten erlaubt, wird dies durch den Wert 1 im *Association Permit*-Feld angegeben.

GTS fields

Es ist möglich, Zeitfenster während der SD für Übertragungen zu reservieren. Diese so genannten GTS[10] werden hier nicht weiter berücksichtigt, da sie keinen Einfluss auf den Netzaufbau haben.

Pending address fields

Der Beacon enthält eine Liste von Knotenadressen, für die der Koordinator Rahmen bereit hält. Die betroffenen Knoten müssen während der SD beim Koordinator Rahmen anfordern. Ein Koordinator sendet niemals selbst Daten an einen Teilnehmer. Wenn ein Teilnehmer eigene Rahmen an den Koordinator gesendet und Rahmen vom Koordinator abgefragt hat, darf er sich bereits vor Ablauf der SD in den Schlafmodus versetzen. Die direkte Kommunikation zwischen den Teilnehmern eines Koordinators ist nicht vorgesehen.

[10] Guaranteed Time Slots

Beacon payload

Die höhere Schicht im Koordinator kann im *Beacon payload* zusätzliche Informationen ablegen, die für den Netzaufbau erforderlich sein können. Die Verarbeitung dieser Informationen erfolgt ausschließlich auf der Netzwerkschicht.

2.1.3 Datenübertragung

Im *beacon enabled mode* werden alle Datenübertragungen als Transaktionen durchgeführt. Dies bedeutet, dass zu sendende Daten in Rahmen verpackt und in einem so genannten Transaktionspuffer zwischengespeichert werden.

Wenn ein Koordinator einen Beacon sendet, enthält das *Pending address list*-Feld die Liste der Teilnehmer, für die Transaktionen vorliegen. Die Teilnehmer fragen dann diese Rahmen während der SD über einen MAC-Rahmens ab.

Wenn ein Teilnehmer den Beacon seines Koordinators empfangen hat, versucht er während der SD die Transaktionen abzuarbeiten, die an den Koordinator gesendet werden sollen.

Transaktionen bleiben einen begrenzten Zeitraum gültig. Bei Ablauf der Gültigkeit bzw. Überlauf des Transaktionspuffers wird die Netzwerkschicht des Knotens informiert. Dadurch lassen sich für die Netzwerkschicht Verbindungsabbrüche erkennen.

Das direkte Senden von Rahmen wie im *non-beacon enabled mode* ist nicht möglich, da sich Knoten nur während der eigenen SD oder teilweise während der SD ihres Koordinators im Wachzustand befinden.

2.1.4 Netzverwaltung

Die MAC-Schicht bietet bereits Funktionen, um im *beacon enabled mode* Knoten miteinander zu assoziieren. Diese Funktionen werden in dieser Arbeit verwendet und hier erläutert.

Assoziierung

Die Assoziierung eines Knotens an einen Koordinator folgt dem in Abbildung 2.4 dargestellten Schema.

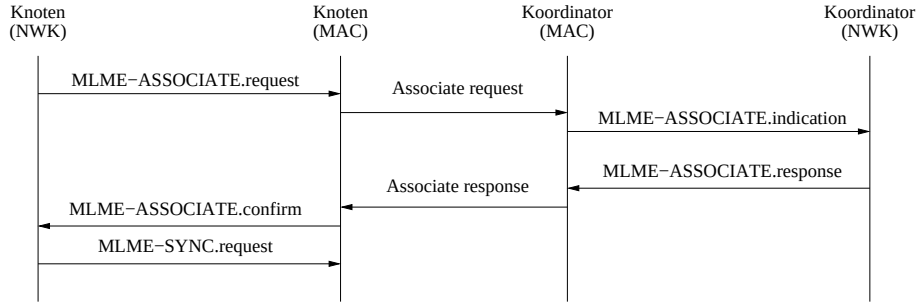

Abbildung 2.4: Ablauf Assoziierung

Durch die Verwendung der MAC-Primitiven `MLME-ASSOCIATE.request` auf dem Teilnehmer wird ein entsprechender MAC-Rahmen an den Koordinator gesendet. Die NWK-Schicht des Koordinators wird über die Assoziierung mit der `MLME-ASSOCIATE.indicaton`-Primitive informiert. Die Assoziierungsanfrage wird mit einer `MLME-ASSOCIATE.confirm`-Primitive beantwortet. Dabei kann die NWK-Schicht des Koordinators bereits die logische Adresse der NWK-Schicht im neuen Knoten mitteilen. Diese wird durch die `MLME-ASSOCIATE.response`-Primitive über das Ergebnis der Assoziierung informiert.

Scannen

Bevor eine Assoziierung durchgeführt werden kann, muss ein Knoten nach verfügbaren Koordinatoren scannen können. Dafür gibt es die MAC-Primitiven `MLME-SCAN.request` und `MLME-SCAN.confirm`. Dabei stehen die vier in Tabelle 2.3 aufgelisteten Scan-Typen zur Verfügung.

Typ	Beschreibung
ED Scan	dient zur Ermittlung der Kanalauslastung, wird vom PAN-Koordinator zur Kanalauswahl durchgeführt
Active Scan	sendet einen *Beacon request*-Rahmen und scannt dann nach Beacon-Rahmen. Wird nur von FFD unterstützt und hier nicht verwendet
Passive Scan	scannt nach Beacon-Rahmen
Orphan Scan	scannt nach Beacon-Rahmen des eigenen Koordinators, wird bei Synchronitätsverlust angewendet

Tabelle 2.3: Scan-Typen

Ein *ED*[11] *Scan* wird vom PAN-Koordinator bei der Formierung des PAN-Clusters durchgeführt. Er wird genutzt, um einen Kanal mit einer möglichst geringen Belegung zu ermitteln.

Bei einem *Active Scan* sendet der Knoten einen *Beacon Request*-Rahmen aus und scannt nach Beacons. Damit können sowohl Koordinatoren im *beacon enabled mode* als auch im *non-beacon enabled mode* entdeckt werden.

Beim *Passive Scan* versucht der Knoten Beacons zu empfangen. Damit können nur Koordinatoren im *beacon enabled mode* gescannt werden.

Ein *Orphan Scan* wird bei Synchronisationsverlust des Teilnehmers auf einen Koordinator angewendet. Bei Erfolg ist die MAC-Schicht mit dem Beacon des Koordinators wieder synchron.

2.2 ZigBee-Spezifikation

Die ZigBee-Allianz ist ein Zusammenschluss von mehreren Unternehmen, um einen offenen Netzwerkstandard für Funknetze zu schaffen. Die ZigBee-Spezifikation ist dabei ein auf IEEE 802.15.4 aufsetzender Protokollstack. Ziel ist es, die Interoperabilität zwischen sowohl kostengünstigen als auch hochverfügbaren Geräten verschiedener Hersteller in Netzen sicherzustellen.

[11] Energy Detection

Als Anwendungsbereiche gelten unter anderem die Heim- und Gebäudeautomatisierung, Patientenüberwachung in der Medizin sowie die Automatisierungstechnik.

Der Aufbau der ZigBee-Schichten [Zig06] ist in Abbildung 2.5 dargestellt und im Folgenden kurz erläutert.

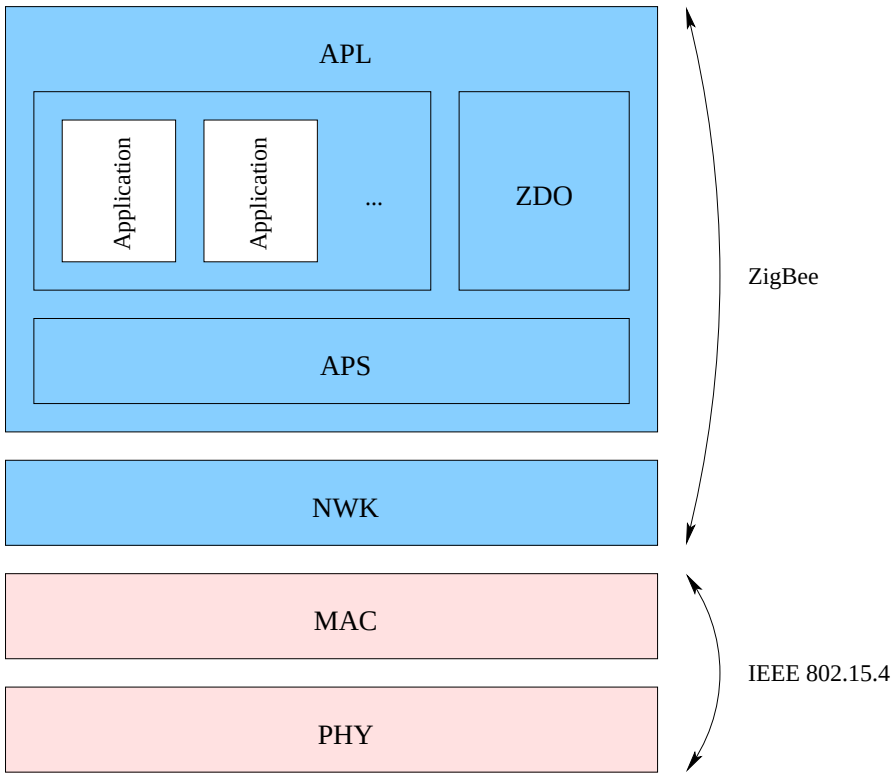

Abbildung 2.5: Aufbau ZigBee-Schichten

Network Layer

Die NWK[12]-Schicht bietet alle nötigen Funktionen zur:

- Formierung neuer Netze
- Vergabe von Adressen
- Beitreten und Verlassen von Netzen
- Anwendung von Kryptografie auf Rahmen

[12]Network

- Routing von Rahmen
- Scannen nach Nachbarknoten

Da die Adressvergabe und das Routing auf die NWK-Schicht begrenzt ist, soll für die höheren Schichten nur eine Übersicht gegeben werden.

Die Spezifikation gibt weder ein festes Routingprotokoll noch eine feste Netzwerktopologie vor, legt jedoch den Schwerpunkt auf Maschenstrukturen und AODV[13]-Routing.

Application Layer

Der APL[14] sitzt direkt über der NWK-Schicht des ZigBee Standards und ist in mehrere Einheiten unterteilt.

- Der APS[15] ist verantwortlich für die Verwaltung von logischen Verknüpfungen zwischen Geräten basierend auf ihren Diensten und Anforderungen.
- Das ZDO[16] definiert die Rolle des Gerätes im Netzwerk (Koordinator oder Endgerät), verarbeitet und startet Verknüpfungsanfragen zwischen Geräten und handelt kryptografische Parameter aus.
- Das *Application Framework* liegt im Verantwortungsbereich des Herstellers und wird nicht von der ZigBee-Spezifikation vorgegeben.

2.3 Cluster-Tree-Protokoll

Der Entwurf des *Cluster-Tree-Protokoll* [Cal01] stellt eine Spezifikation der NWK-Schicht für selbstorganisierende Ad-hoc-Netzwerke dar. Unter selbstorganisierend ist hier eine Realisierung der Netzwerktopologie ohne menschliche Steuerung zu verstehen.

Das Protokoll definiert zum einen die Formierung von einzelnen Clustern, zum anderen auch die Verbindung von mehreren Clustern. Diese Arbeit beschränkt sich auf einzelne

[13] Ad-hoc On-demand Distance Vector
[14] Applicaton Layer
[15] Application Support Sublayer
[16] ZigBee Device Object

Cluster, die keine Verbindung zu anderen Clustern besitzen. Es werden deshalb die Funktionen zur Inter-Cluster-Kommunikation nicht weiter betrachtet.

2.3.1 Knotenarten

Knoten

Normale Knoten können einem existierenden Cluster beitreten. Falls kein Cluster erreicht werden kann, kann ein Knoten einen neuen Cluster formieren.

Cluster Head

Um ein Cluster zu formieren, werden Knoten zu so genannten CH[17]s. Ein CH vergibt jedem am Cluster teilnehmenden Knoten eine eindeutige Knoten-ID. Die Auswahl des CH wird dabei entweder vorher festgelegt, oder Knoten werden nach einer definierten Zeit selbst zu CH, falls sie keine Verbindung zu einem Cluster aufbauen können.

Designated Device

Zur Verbindung von mehreren Clustern wird ein so genanntes DD[18] benötigt. Das DD ist verantwortlich für die Vergabe eindeutiger-IDs an die Cluster, um so clusterübergreifende Kommunikation zu ermöglichen.

2.3.2 Adressierung

Zur Adressierung werden den Knoten je eine 16 bit breite logische Adresse zugeordnet. Diese setzt sich aus einer 8 bit breite Cluster-ID und einer 8 bit breite Knoten-ID zusammen.

Das DD hat dabei immer die Cluster-ID 0. Cluster, die noch keine direkte oder indirekte Verbindung zum DD besitzen, erhalten die temporäre Cluster-ID 254. Um Broadcastnachrichten an alle Cluster zu senden wird die Cluster-ID 255 verwendet.

[17] Cluster Head
[18] Designated Device

Nach dem gleichen Schema werden die Konten-IDs innerhalb eines Clusters vergeben. Der CH hat immer die Knoten-ID 0. Knoten, die noch keine ID zugewiesen bekommen haben, verwenden die temporäre Knoten-ID 254, Broadcastnachrichten an alle Knoten in einem Cluster verwenden die Knoten-ID 255.

2.3.3 Intra-Cluster-Rahmen

Das Protokoll definiert für die Cluster-Formierung verschiedene Rahmentypen, die in Tabelle 2.4 aufgelistet sind.

Rahmentyp
HELLO
CONNECTION REQUEST
CONNECTION RESPONSE
NODE ID REQUEST
NODE ID RESPONSE
DISCONNECTION REQUEST
DISCONNECTION RESPONSE
LINK-STATE REPORT
TOPOLOGY UPDATE
ACKNOWLEDGEMENT

Tabelle 2.4: Rahmen-Typen

Im folgenden Abschnitt wird die Verwendung der Rahmen erläutert. Der Aufbau der Rahmenstrukturen ist in [Cal01] nachzulesen. Rahmen für die Inter-Cluster-Kommunikation werden hier nicht weiter betrachtet, da Inter-Cluster-Kommunikation nicht zum Aufgabenumfang dieser Arbeit gehört.

2.3.4 Cluster-Formierung

CH Auswahl

Nach Einschalten eines Knotens wartet dieser auf den Empfang eines `HELLO`-Rahmens. Wenn nach einer bestimmten Zeit kein `HELLO`-Rahmen empfangen wurde, wird der Knoten zu einem neuen CH. Alternativ erlaubt das Protokoll die Festlegung von CH-Knoten anhand beliebiger anderer Parameter. Diese und die vorangegangenen Arbeiten (siehe Abschnitt 2.5) gehen von einer Festlegung der CH-Knoten aus.

Stern-Cluster

Nach Auswahl des CH sendet dieser periodisch `HELLO`-Rahmen aus.

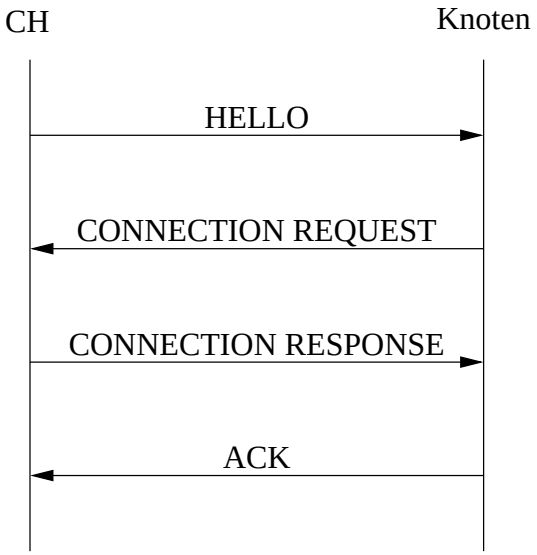

Abbildung 2.6: Stern-Cluster-Formierung

In Abbildung 2.6 ist der Ablauf beim Beitritt eines Knotens in einen Cluster dargestellt. Knoten, die einen `HELLO`-Rahmen empfangen, treten dem Cluster über einen `CONNECTION REQUEST`-Rahmen bei. Der CH bestätigt den Beitritt mit einem entsprechenden `CONNECTION RESPONSE`-Rahmen, der dem Knoten eine ID zuweist. Der Empfang des Rahmens wird mit einem `ACKNOWLEDGEMENT`-Rahmen bestätigt.

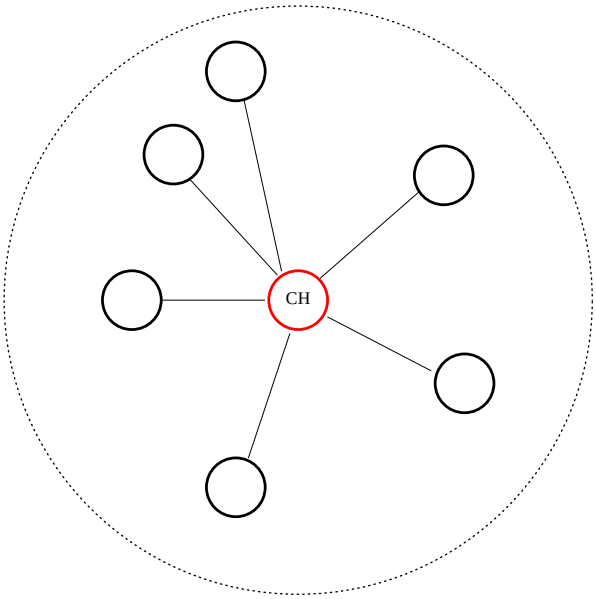

Abbildung 2.7: Beispiel eines Stern-Clusters

Durch dieses Vorgehen bildet sich, wie in Abbildung 2.7 dargestellt, ein Stern-Cluster.

Baum-Cluster

Der Standard sieht vor, dass aus einem Stern- ein Baum-Cluster werden kann. Dafür muss ein Teil der Knoten mehrere Verbindungen eingehen können. In Abbildung 2.8 ist der Beitritt eines Knotens in einem Baum-Cluster dargestellt.

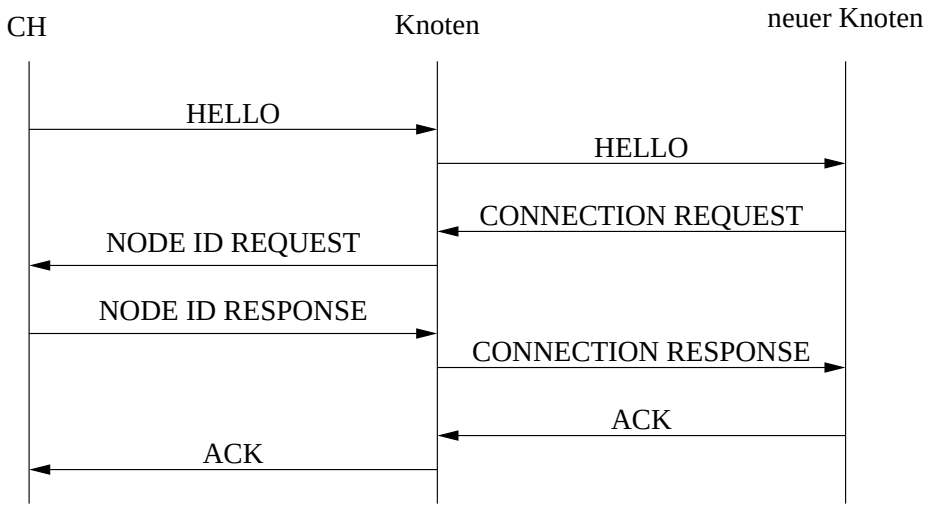

Abbildung 2.8: Baum-Cluster-Formierung

Wenn ein Knoten die Verbindung mit dem CH aufgebaut hat, beginnt er vom CH empfangene `HELLO`-Rahmen weiter zu senden. Neue Knoten, die die `HELLO`-Rahmen der aussenden Knoten empfangen, verbinden sich zu diesen mittels `CONNECTION REQUEST`-Rahmen. Falls er sich nicht mit dem CH verbindet, muss dieser Knoten einen `NODE ID REQUEST`-Rahmen in Richtung CH senden, um dem neuen Knoten eine Knoten-ID zuweisen zu können. Nach Erhalt eines `NODE ID REPSONSE`-Rahmens vom CH kann der neue Knoten per `CONNECTION RESPONSE`-Rahmen in das Cluster aufgenommen werden. Der Empfang wird wiederum mit `ACKNOWLEDGEMENT`-Rahmen bestätigt.

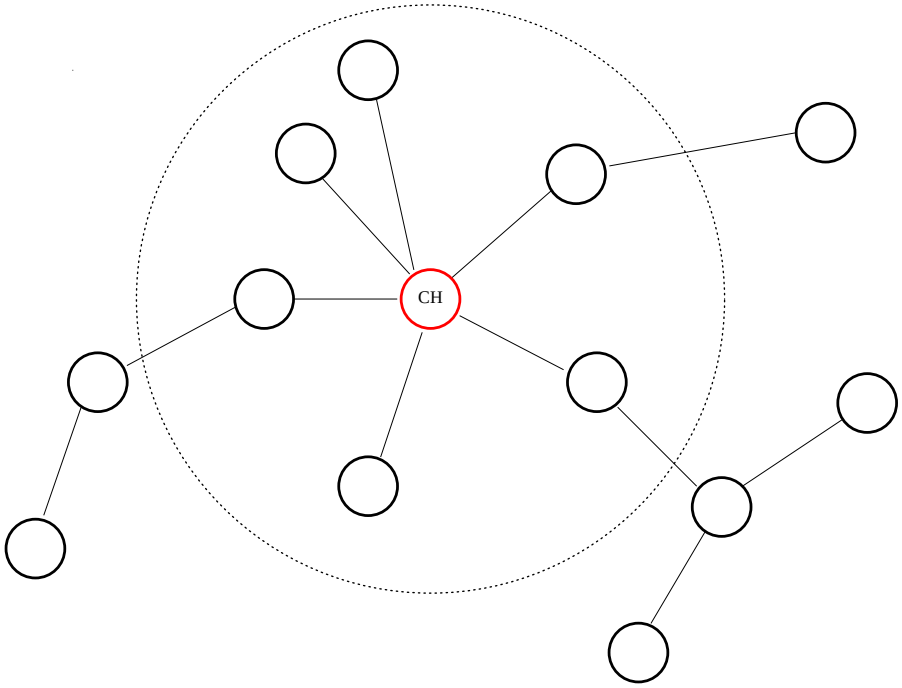

Abbildung 2.9: Beispiel eines Baum-Cluster

Daraus entstehen wie in Abbildung 2.7 dargestellt Baum-Cluster. Sie decken eine größere räumliche Fläche ab als Stern-Cluster, besitzen jedoch auch eine größere Verzögerung beim Aufbau des Netzes.

2.3.5 Netzfunktionen

Datenübertragung

Jeder Knoten führt eine Tabelle mit seinen Nachbarknoten. Die Existenz von Nachbarknoten lässt sich durch die `HELLO`-Nachrichten feststellen. Die Kommunikation zwischen benachbarten Knoten erfolgt direkt.

Falls ein Rahmen an einen Knoten gesendet werden soll, der sich nicht in direkter Nachbarschaft befindet, wird der Rahmen an den übergeordneten Knoten bzw. den CH weitergeleitet. Der CH besitzt die Routen zu allen Knoten im Cluster.

Netzoptimierung

Je nach Anforderung an den Cluster senden Knoten periodisch `LINK-STATE REPORT`-Rahmen an den CH. Dieser berechnet daraus den kürzesten Pfad zu allen Knoten des Clusters und kann über entsprechende `TOPOLOGY UPDATE`-Rahmen den einzelnen Knoten explizit neue Vaterknoten zuweisen.

Bei Ausbleiben von `LINK-STATE REPORT`-Rahmen wird von einem Verbindungsverlust des betroffenen Knotens ausgegangen. Daraufhin kann der CH durch entsprechende `TOPOLOGY UPDATE`-Rahmen betroffene Kindknoten neu an das Netz anschließen.

2.4 Network Simulator 2

Das Programm ns-2[19] ist ein Netzwerksimulator zur Simulation von IP[20]-basierenden Transport- und Routingprotokollen. Der Simulator gehört zur Klasse der *Discret Event Simulator*. Neben klassischen kabelgebundenen Medien unterstützt er auch die Simulation von Funknetzen unter Verwendung konfigurierbarer Ausbreitungsmodelle. Die Simulationsszenarien und der Simulationsablauf wird per Tcl[21]-Skript konfiguriert und gesteuert.

[19] Network Simulator 2
[20] Internet Protocol
[21] Tool command language

Zur Simulationszeit werden all diese Ereignisse in einer so genannten *trace*-Datei aufgezeichnet. Dies ermöglicht eine spätere (statistische) Auswertung der Simulation. Für eine grafische Darstellung der Simulation wird außerdem eine Animationsdatei erstellt. Mittels des für ns-2 entwickelten nam[22] kann die Simulation grafisch verfolgt und untersucht werden.

Der Simulator wurde in C++ geschrieben und bietet bereits die Unterstützung des IEEE 802.15.4 Protokolls auf der Basis der Fassung von 2003 [IEE03]. Wie in dieser Arbeit wurden dort alle kryptografischen Funktionen der MAC-Schicht nicht implementiert.

Durch seine Modularität kann ns-2 durch Einführung weiterer C++-Klassen und die Einbindung von Tcl-Schnittstellen leicht um neue Protokolle erweitert werden.

2.5 Arbeiten am Lehrstuhl

Am Lehrstuhl für Telekommunikation an der TU Dresden gab es bereits im Vorfeld für diese Arbeit bedeutende Diplomarbeiten. Im Folgenden wird auf den Inhalt dieser Arbeiten und die Verwendung in dieser Arbeit eingegangen.

2.5.1 Cluster-Tree-Topologie

In der Diplomarbeit von Negro *Investigation of Cluster-Tree Topology in ZigBee Networks* [Neg06] wurde das Cluster-Tree-Protokoll auf IEEE 802.15.4 für einzelne Cluster angewendet. Abweichend von dem in Abschnitt 2.3 vorgestellten Protokoll ist die maximale Baumtiefe und die maximalen Anzahl der Teilnehmer pro Koordinator fest vorgegeben.

Dadurch ist die Adressvergabe für jeden Teilnehmer eineindeutig von seinem Koordinator zuweisbar. Ein Ermitteln der Knotenadresse beim CH mittels der in Unterpunkt 2.3.3 vorgestellten `NODE ID REQUEST`-Rahmen ist damit nicht mehr nötig. Das Schema zur Adressvergabe wird im Folgenden erläutert.

Ein Baum ist durch durch die maximale Tiefe L_m und die maximale Anzahl von Kindknoten pro Knoten C_m definiert. Für einen Baum der Tiefe L lässt sich die maximale Anzahl von Knoten n_L nach Gleichung 2.3 berechnen.

[22] Network Animator

$$n_L = \frac{1 - C_m L + 1}{1 - C_m} \tag{2.3}$$

Der CH des Baumes bekommt die Adresse 0 zugeordnet, sein erster Kindknoten die Adresse 1. Es kann nach der Formel 2.3 die Anzahl der möglichen Knoten unterhalb von Knoten 1 als n_{L_m-1} berechnet werden. Folglich erhält der zweite Kindknoten vom Kopfknoten die Adresse $1+n_{L_m-1}$. Nach diesem Schema werden alle Adressen vergeben. Abbildung 2.10 zeigt einen Baum mit den Parametern $C_m = 3$ und $L_m = 2$.

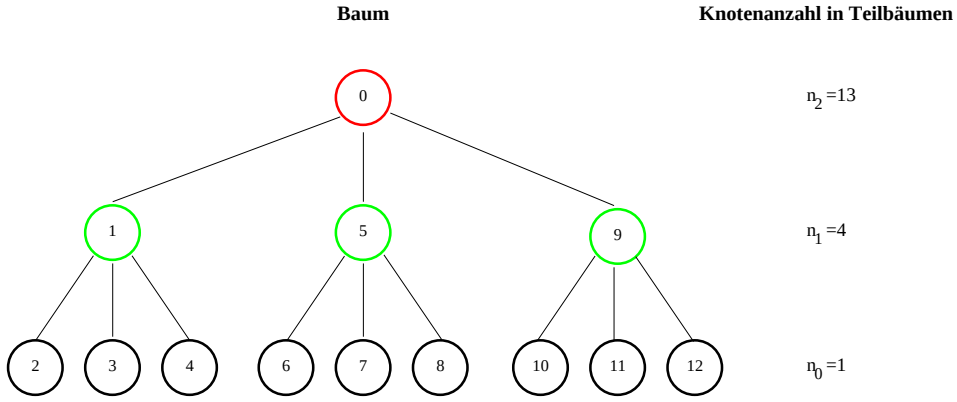

Abbildung 2.10: Adressvergabe im Baum

Das in [Neg06] verwendete Adressierungsschema wurde auch in dieser Arbeit angewandt. Die Implementierungen im ns-2 konnten jedoch nicht weiter verwendet werden, da in der Arbeit von Negro keine der Funktionen der MAC-Schicht für den Aufbau von Cluster-Tree-Netzen berücksichtigt wurden.

2.5.2 Rekonfigurationsalgorithmen

Die Diplomarbeit *Implementation of Reconfiguration Algorithms in Cluster Tree Networks for IEEE 802.15.4/ZigBee* [Sam06] von Samperio erweitert die Arbeit von Negro um Ansätze zur Rekonfiguration eines Cluster-Tree-Netzes. Unter der Rekonfiguration wird verstanden, dass Knoten, die bereits zu einem Cluster gehören, sich zu einem anderen Cluster im gleichen PAN verbinden. Ziel ist es, eine höhere Konnektivität zu erreichen.

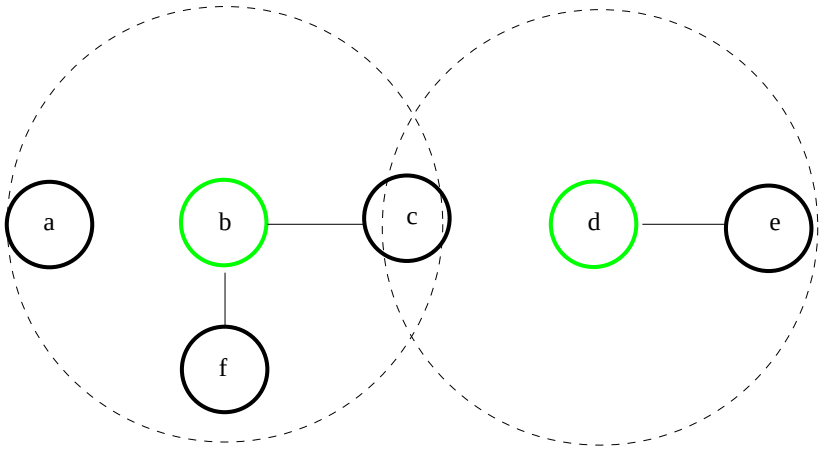

Abbildung 2.11: Rekonfiguration

Ein typisches Szenario ist in Abbildung 2.11 dargestellt. Die gestrichelten Kreise stellen die Reichweite der Beacons der Koordinatoren da. Der Knoten `a` ist in Kommunikationsreichweite mit dem Koordinator `b`. Dieser besitzt jedoch bereits die maximale Anzahl von Kindknoten, und Knoten `a` kann deshalb nicht dem PAN-Cluster beitreten. Die Algorithmen zur Rekonfiguration lösen das Problem durch das Umhängen von Knoten. Im Beispiel könnte Knoten `c` sich mit Koordinator `d` assoziieren, und Knoten `a` kann dem Cluster beitreten.

Der in der Diplomarbeit von Samperio vorgestellte Lösungsansatz kann nicht direkt auf diese Arbeit übertragen werden, da ein Knoten, der einmal mit dem Cluster assoziiert ist, nicht mehr kontinuierlich auf die Beacons der in Reichweite befindlichen Koordinatoren hören kann. Diese Erweiterung geht über den Rahmen dieser Arbeit hinaus und wurde daher hier nicht bearbeitet.

3 Umsetzung Cluster-Tree-Protokoll auf ZigBee

Dieses Kapitel beschreibt die in dieser Arbeit durchgeführte Umsetzung des in Abschnitt 2.3 beschriebenen Cluster-Tree-Protokolls auf die entsprechenden Schichten des IEEE 802.15.4 Standards und der ZigBee Spezifikation. Implementierungsspezifische Details sind im Kapitel 4 beschrieben.

3.1 Betriebsmodus IEEE 802.15.4

Wie in den Grundlagen in Kapitel 2.1 beschrieben kann die IEEE 802.15.4 MAC-Schicht sowohl im *beacon enabled mode* als auch im *non-beacon enabled mode* betrieben werden.

Im *non-beacon enabled mode* wäre eine Implementierung des in Kapitel 2.3 vorgestellten Cluster-Tree-Protokolls direkt möglich. Die hier zur Verfügung stehende MAC-Schicht kann jedoch wesentlich energiesparsamer im *beacon enabled mode* betrieben werden. Der Grund dafür sind die fest definierten Schlaf- und Wachzyklen im *beacon enabled mode*.

In der MAC-Schicht sind außerdem Funktionen für eine Verwendung in baumstrukturierten Netzen [IEE03] vorgesehen. Als Betriebsmodus wird deshalb der *beacon enabled mode* festgelegt.

Die Beacons nehmen dabei die Rolle der HELLO-Nachrichten im *Cluster-Tree-Protokoll* ein. Die Rahmen zur Netzformierung werden, soweit nötig, wie in Tabelle 3.1 durch entsprechende Rahmen der MAC-Schicht ersetzt.

Cluster-Tree-Protokoll-Rahmen	IEEE 802.15.4 MAC-Rahmen
`HELLO`	Beacon
`CONNECTION REQUEST`	Association request
`CONNECTION RESPONSE`	Association response
`NODE ID REQUEST`	*nicht benötigt*
`NODE ID RESPONSE`	*nicht benötigt*
`DISCONNECTION REQUEST`	*Realisierung auf NWK-Schicht*
`DISCONNECTION RESPONSE`	Disassociation notification
`LINK-STATE REPORT`	*nicht benötigt*
`TOPOLOGY UPDATE`	*nicht benötigt*
`ACKNOWLEDGEMENT`	Acknowledge

Tabelle 3.1: Überführung der Cluster-Tree-Rahmen

3.2 Adressierung

Jeder Knoten wird in einem Cluster durch die 16 bit breite logische Knoten-ID adressiert. Die Adressvergabe erfolgt dabei dezentral nach dem in Kapitel 2.5.1 vorgestellten Adressierungsschema.

Auf MAC-Ebene wird mit den vorhandenen 64 bit breiten und eineindeutig vergebenen Knotenadressen gearbeitet. Dies bietet eine große Robustheit bei der Überlagerung von mehreren PAN-Clustern. Dadurch wird sichergestellt, dass bei einer Überlagerung von Clustern keine Daten ungewollt zwischen den Clustern ausgetauscht werden.

Jeder PAN-Cluster hat außerdem eine 16 bit breite PAN-ID. Diese wird grundsätzlich vom PAN-Koordinator vergeben. In dieser Arbeit wird davon ausgegangen, dass sie vordefinitiert bzw. nach einem hier nicht näher spezifizierten Algorithmus bestimmt wird. Die PAN-ID wird nur im Falle von Inter-Cluster-Kommunikation benötigt, die jedoch nicht Umfang dieser Arbeit ist.

3.3 Rahmenstruktur

Die Zigbee-Spezifikation [Zig06] definiert eine Liste von Kommandorahmen, die für die eigene Umsetzung erweitert wurde. Es folgt eine Erläuterung der eingeführten Rahmenstrukturen. Die Verwendung der Rahmen ist in den folgenden Abschnitten erläutert.

Rahmentypen

Es wurden die von ZigBee in Tabelle 3.2 aufgelisteten unterstützten Kommandorahmen um die Rahmen *Beacons scanned* und *Beacon config* ergänzt.

Name	ID
Route request	1
Route reply	2
Leave	3
Route record	4
Rejoin request	5
Rejoin response	6
Beacons scanned	7
Beacon config	8

Tabelle 3.2: Ergänzte ZigBee-Rahmentypen

Beacons scanned-Rahmen

Dieser Rahmen enthält eine Liste von Knoten aus dem PAN-Cluster, von denen ein Beacon empfangen wurde. Die Knotenadressen werden dabei von der höchsten zur schlechtesten Signalqualität des empfangenen Signals sortiert. Die Anzahl der maximalen Knotenadressen ist durch die maximale Größe eines Rahmens begrenzt.

Bitbreite	1	7	64	n*64
Feld	isCoord	rxBeacons	devAddr	coordNodes

Abbildung 3.1: Aufbau *Beacons scanned*-Rahmenformat

Die Felder des Rahmen haben folgende Bedeutung:

Feldname	Typ	Beschreibung
isCoord	Boolean	Hat den Wert 1, wenn der Knoten ein FFD ist, ansonsten den Wert 0.
rxBeacons	Integer	Die Anzahl der Knotenadressen im *coordNodes* Feld.
devAddr	Geräteadresse	Die IEEE Adresse des sendenden Knotens.
coordNodes	Liste von Geräteadressen	Liste von Knotenadressen, von denen ein Beacon empfangen wurde.

Tabelle 3.3: Beschreibung *Beacons scanned*-Felder

Der Inhalt des Feldes *isCoord* gibt an, ob der Knoten die Rolle eines Koordinators einnehmen kann. Für FFD-Knoten ist dieses Feld auf 1, für RFD-Knoten immer auf den Wert 0 gesetzt.

Das Feld *rxBeacons* beschreibt die Anzahl der Knotenadressen, die im Feld *coordNodes* enthalten sind an.

Die Knotenadresse des ursprünglichen Senders des Rahmens ist im Feld *devAddr* abgelegt.

Die Liste der Knotenadressen, von denen ein Beacon empfangen wurde, ist im Feld *coordNodes* hinterlegt. Die maximale Größe des Feldes ist durch die maximale Größe der Rahmen auf der MAC-Schicht definiert.

Beacons config-Rahmen

Dieser Rahmen teilt einem Knoten die Startzeit seines Beacons relativ zum Beacon seines Koordinators mit.

Bitbreite	1	1	6	24	8
Feld	sendBeacon	doRescan	reserviert	startTime	rescanOrder

Abbildung 3.2: Aufbau *Beacon config*-Rahmenformat

Die Felder im *Beacon config*-Rahmen haben folgende Bedeutung:

Feldname	Typ	Beschreibung
sendBeacon	Boolean	Wert 1, falls Beacons gesendet werden dürfen, sonst den Wert 0.
doRescan	Boolean	Wert 1, falls zyklische Scans nach Beacons durchgeführt werden sollen, sonst den Wert 0.
startTime	Integer	Die relative Startzeit in Symbolen für die `MLME-START.request`-Primitive.
rescanOrder	Integer	Zeit bis Beacon Scan, falls *doRescan* den Wert 1 hat. Die Zeit Berechnet sich aus $BI * 2^{rescanOrder}$.

Tabelle 3.4: Beschreibung *Beacon config*-Felder

Für den Fall, dass *sendBeacon* den Wert 0 hat, ist es dem Knoten nicht erlaubt Beacons auszusenden. Er hat sich somit als Teilnehmer zu verhalten.

Mit dem Feld *doRescan* kann der Knoten angewiesen werden, zyklisch Beacon Scans durchzuführen. Bei dem Wert 0 erfolgen keine Scans.

Das Feld *startTime* gibt die Zeit in Symbolen an, zu der der Beacon des Knoten gesendet werden soll. Die Zeit ist relativ zum Beginn des Beacons des Koordinators des Knotens. Dieses Feld wird nur ausgewertet, falls *isCoord* den Wert 1 besitzt.

Falls *doRescan* den Wert 1 hat, wird aus *rescanOrder* die Zeit berechnet, nach der nächste Scan durchgeführt werden soll. Nach dem Ablauf von $2^{rescanOrder}$ *Beacon Intervals* erfolgt dann der nächste Scan.

3.4 PAN-Cluster

3.4.1 Parameter

Als PAN-Koordinator kommt prinzipiell jedes FFD in Frage. Es sollte jedoch ein Gerät mit genügend Rechenleistung und Energieversorgung ausgewählt werden. Die Auswahl ist nicht Bestandteil dieser Arbeit, da sie abhängig vom Anwendungsfall ist.

Für die IEEE 802.15.4 MAC-Schicht müssen die in Tabelle 3.5 aufgezählten Parameter festgelegt werden.

Parameter	Typ	Beschreibung
Channels	Integer	Bitmaske der erlaubten Kanalnummern für das PAN.
BO	Integer	*Beacon Order* des PANs ($0 \leq BO \leq 14$).
SO	Integer	*Superframe Order* des PANs ($0 \leq SO < BO$).

Tabelle 3.5: IEEE 802.15.4-Parameter

Channels ist eine Bitmaske für die erlaubten logischen Kanäle, von denen der PAN-Koordinator auswählen darf (siehe Abbildung 2.1).

BO und SO sind die Parameter für die Dauer des BI und der SD der MAC-Schicht (siehe Kapitel 2.1).

Für das Cluster-Tree-Protokoll müssen entsprechend Kapitel 2.5.1 die Parameter in Tabelle 3.6 festgelegt werden.

Parameter	Typ	Beschreibung
L_m	Integer	Maximale Baumtiefe.
C_m	Integer	Maximale Anzahl von Kindknoten an einem Koordinator.

Tabelle 3.6: Cluster-Tree-Parameter

Beacon Order und *Superframe Order*

Eine wichtige Festlegung sind die zu verwendenden Werte für BO und SO, da sie direkten Einfluss auf die Metriken des Netzes haben (siehe Kapitel 2.1). Durch die Parameter BO und SO wird die maximale Übertragungsverzögerung über eine Anzahl von Hops bestimmt.

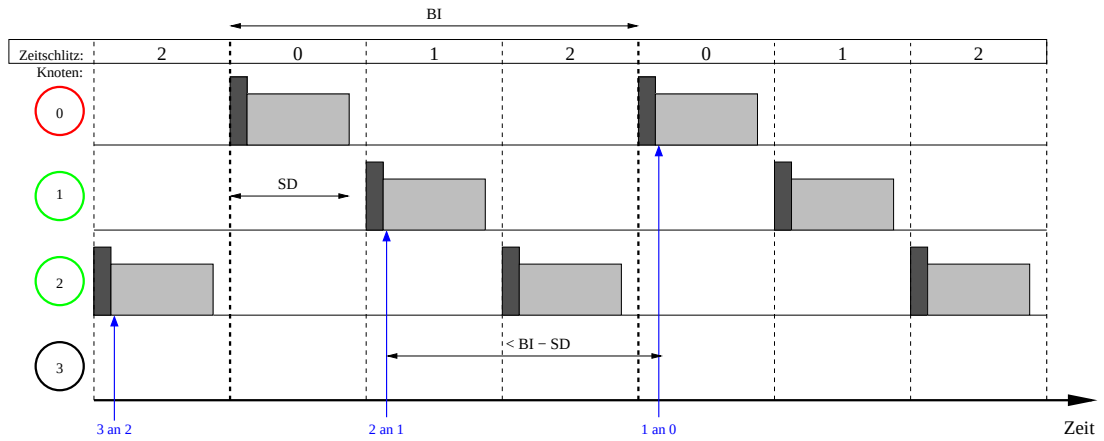

Abbildung 3.3: Auswirkung *Beacon Order* und *Superframe Order*

In Abbildung 3.3 ist die Beaconverteilung eines Baumes mit der Tiefe 4 dargestellt. Vom Blattknoten 3 soll an den Kopfknoten 0 ein Rahmen gesendet werden. Im ungünstigsten Fall muss maximal jeweils die Dauer eines *Beacon Intervals* abzüglich der Dauer einer *Superframe Duration* pro Hop gewartet werden, bis der Rahmen den nächsten Knoten erreicht.

Es wird deutlich, dass neben der *Beacon* und *Superframe Order* auch die Verteilung der *Superframe Durations* der Knoten über das *Beacon Interval* maßgeblichen Einfluss auf die Latenz und den Durchsatz des Netzwerkes haben. Diese Problematik wird in Abschnitt 3.5 näher erläutert.

3.4.2 Cluster-Formierung

In Abbildung 3.4 ist der Ablauf beim Start eines Clusters dargestellt. Zu Beginn scannt der PAN-Koordinator mittels der `MLME-SCAN.request`-Primitive der MAC-Schicht die Kanäle mittels eines *ED Scans*. Aus dem Ergebnis wird dann ein Kanal mit möglichst geringer Belegung ausgewählt. Mittels der MAC-Primitive `MLME-START.request` wird der PAN-Cluster gestartet.

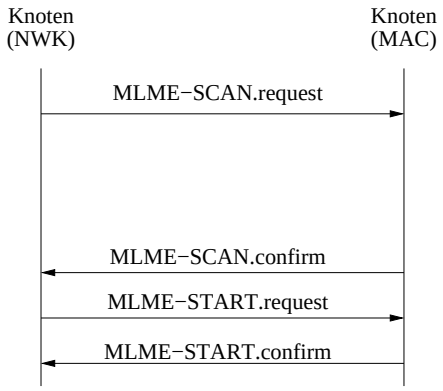

Abbildung 3.4: Ablauf Cluster-Formierung

3.4.3 Cluster-Beitritt

Das Beitreten zu einem PAN-Cluster erfolgt über die existierenden Funktionen der MAC-Schicht. In Abbildung 3.5 ist der Ablauf eines erfolgreichen Beitritts eines Knoten zu einem PAN dargestellt.

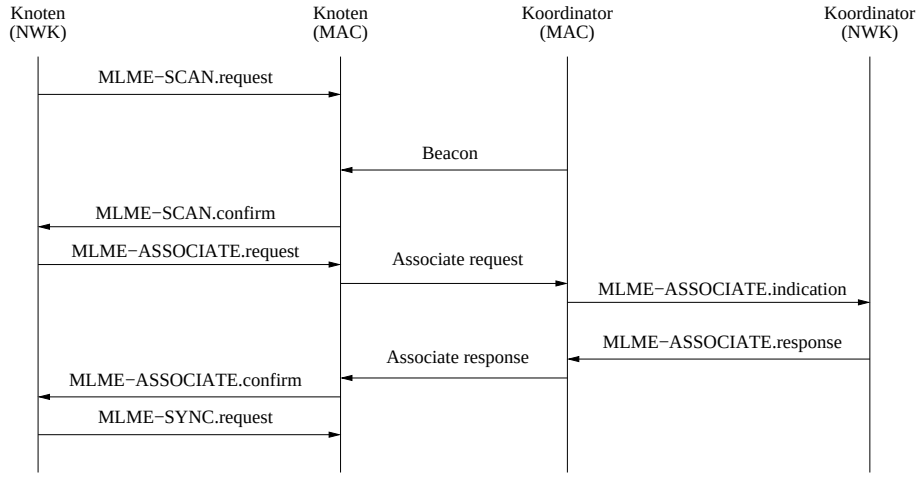

Abbildung 3.5: Ablauf Cluster-Beitritt

Zu Beginn startet die Netzwerkschicht einen *Passive Scan* auf der MAC-Schicht. Als Ergebnis bekommt die Netzwerkschicht von der MAC-Schicht eine Liste der empfangenen Beacons. In dieser Liste sind die Informationen aus den Beacons und die Signalqualität enthalten.

Der Knoten wählt den Koordinator mit der besten Signalqualität aus und leitet eine Assoziierung auf der MAC-Schicht ein. Bei Erfolg bekommt die Netzwerkschicht des Koordinators die Indikation `MLME-ASSOCIATE.indication`. Darauf ruft diese wiederum die MLME-ASSOCIATE.response-Primitive der MAC-Schicht auf. Als Paramter wird hier der Status der Assoziierung und die vergebene logische Adresse übergeben.

Die MAC-Schicht empfängt dann einen entsprechenden Rahmen vom Koordinator und beendet den Assoziierungsvorgang mit einer MLME-ASSOCIATE.response-Primitive. Damit ist die Assoziierung beendet. Als letzter Schritt ist es nötig, dass der Knoten sich auf den Beacon des Koordinators durch Verwendung der `MLME-SYNC.request`-Primitive synchronisiert, damit der Knoten zum Empfang des Beacons aktiv werden und den Rest der Zeit im Schlafmodus verbringen kann.

Sollte die Assoziierung nicht erfolgreich sein, versucht der Knoten sich mit dem nächsten Koordinator auf der Liste zu assoziieren. Gründe für ein Scheitern können unter anderem eine Störung der Übertragung oder die Erschöpfung des logischen Adressvorrates im Koordinator sein.

Falls keine Assoziierung mit einem der Koordinatoren erfolgreich war, ist der Assoziierungsvorgang endgültig gescheitert. Der Knoten kann dann nach einer Wartezeit eine erneute Assoziierung versuchen.

3.4.4 Verbindungsverlust

Es ist wichtig, dass der Verbindungsverlust zwischen zwei assoziierten Knoten auf Grund der begrenzten Baumgröße bzw. Adressvorrates erkannt wird. Es wird hier nach Teilnehmersicht und Koordinatorsicht unterschieden.

Die Teilnehmersicht beschreibt die Erkennung des Verbindungsverlustes eines Knotens zu seinem Koordinator. Die Koordinatorsicht behandelt den Verlust der Verbindung von einem Koordinator zu einem seiner Teilnehmer.

Teilnehmersicht

Die MAC-Schicht überwacht die Beacons des assoziierten Koordinators. Falls mehr als drei aufeinanderfolgende Beacons nicht empfangen werden, gilt die Verbindung als verloren. Darauf wird die Netzwerkschicht mit einer `MLME-SYNC-LOSS.indication` informiert.

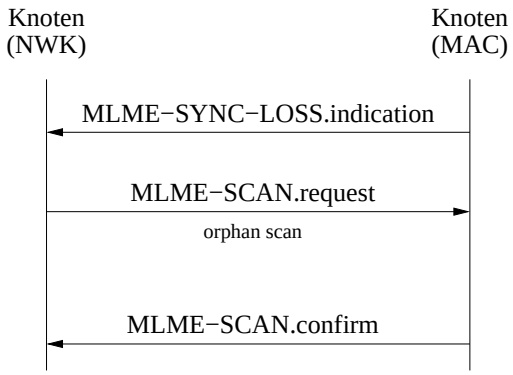

Abbildung 3.6: Verbindungsverlust aus Teilnehmersicht

Als Reaktion darauf führt die Netzwerkschicht einen *Orphan Scan* mittels der `MLME-SCAN.request`-Primitive durch. Über die `MLME-SCAN.confirm`-Primitive wird die Netzwerkschicht informiert, ob der Koordinator wiedergefunden werden konnte.

Falls der Knoten selbst auch ein Koordinator ist, werden die Kindknoten mittels der MAC-Primitive `MLME-DISASOCIATE.request` vom Knoten getrennt und somit der Teilbaum aufgelöst. Durch Auflösen des Teilbaumes binden sich alle betroffenen Knoten neu an das Cluster an, und es entstehen keine vom Cluster isolierten Teilbäume.

Danach findet ein normales Beitreten in den PAN-Cluster statt. Wenn die Verbindung zum PAN wiederhergestellt ist, wird der ursprüngliche Koordinator über den Verlust seines Kindes informiert.

Koordinatorsicht

Der Verbindungsverlust zu einem Teilnehmer kann auf die folgenden zwei Wege festgestellt werden.

Falls ein Kindknoten die Verbindung verliert und sich neu assoziiert, wird die Netzwerkschicht des alten Koordinators über einen entsprechenden `Leave`-Rahmen darüber informiert.

Falls ein Knoten keine neue Verbindung zu dem Cluster herstellen kann, wird dies wie folgt erkannt. Wenn Rahmen für den Knoten bei seinem Koordinator vorliegen und diese nach einem Timeout nicht abgeholt wurden, wird die Netzwerkschicht des Koordinators darüber von der MAC-Schicht mit einer `MLME-COMM-STATUS.indication`-Primitive und dem Ereigniscode `TRANSACTION EXPIRED` informiert.

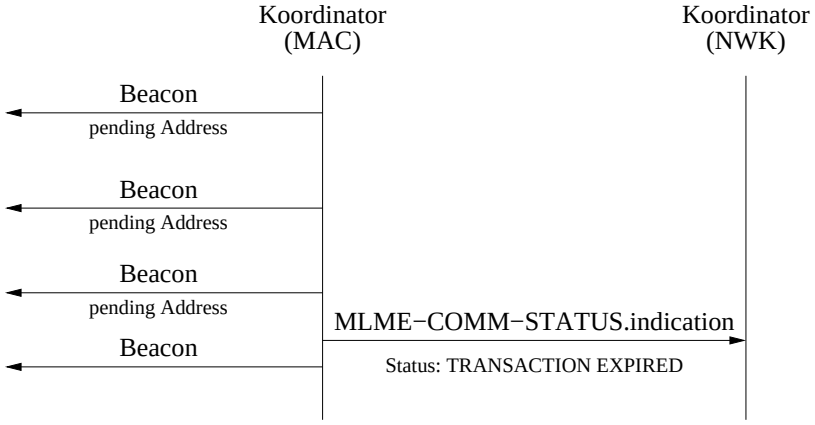

Abbildung 3.7: Verbindungsverlust aus Koordinatorsicht

Es wird deutlich, dass Knoten, die für immer die Verbindung zum PAN-Cluster verlieren und an die keine Daten gesendet werden, für ihren Koordinator weiterhin Teil des Netzwerkes bleiben. Dies kann nur durch ein Polling detektiert werden, in dem ein Koordinator regelmäßig an seine Kindknoten Rahmen sendet und bei Ablauf der Transaktionen ein Verbindungsabbruch feststellt. Dies bedeutet jedoch eine höheren Energieverbrauch der Kinder und eine zusätzliche Netzwerkbelastung und muss deshalb bei Bedarf von höheren Schichten realisiert werden.

3.5 Beacon-Scheduling-Problem

Je mehr Koordinatoren in einem Netzwerk existieren, um so wahrscheinlicher wird es, dass sich die ausgesendeten Beacons gegenseitig stören.

Eine Beaconkollision führt potentiell zu einer schlechteren Verfügbarkeit der Verbindungen zwischen Teilnehmern und Koordinatoren im Netzwerk:

- Teilnehmer können durch Kollisionen ihre Verbindung zum Koordinator verlieren, wenn der Beaconempfang gestört wird
- Assoziierung kann nicht durchgeführt werden, obwohl sich ein Koordinator mit freier Kapazität in Reichweite befindet, der jedoch durch Beaconkollision verdeckt wird
- zwei Koordinatoren stören potentiell nur ihre eigenen Teilnehmer, sich gegenseitig jedoch nicht

Die folgenden Betrachtungen begrenzen sich auf Kollisionen im PAN-Cluster.

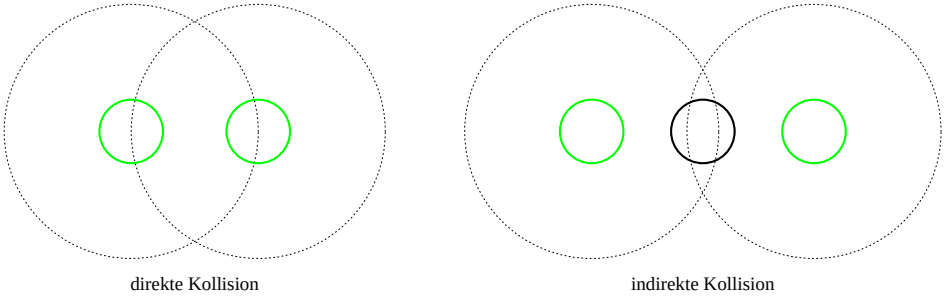

Abbildung 3.8: Darstellung direkte und indirekte Kollision

Man unterscheidet zwischen der in Abbildung 3.8 dargestellten direkten und indirekten Kollision [KAAN07]. Die gestrichelten Kreise stellen die Reichweite der Beacons der Koordinatoren dar.

Direkte Kollision

Bei der direkten Kollision senden zwei Koordinatoren, die sich in Reichweite befinden, ihre Beacons zur gleichen Zeit. Das stört die Kommunikation zwischen Teilnehmern und Koordinatoren. Zwischen einem Koordinator und dessen Teilnehmern tritt diese Art der Kollision nicht auf, da das gleichzeitige Senden von Beacons zweier miteinander assoziierter Knoten von der MAC-Schicht verhindert wird.

Indirekte Kollision

Bei der indirekten Kollision können sich die Koordinatoren gegenseitig nicht empfangen. Es gibt jedoch Knoten, die sowohl den Beacon des einen als auch des anderen Koordinators empfangen. Als Resultat können die Knoten entweder nur einen oder keinen Beacon durch die Kollision dekodieren.

Um indirekte Kollisionen zu vermeiden, benötigt man offensichtlich Informationen über die Reichweite der Beacons eines jeden Koordinators - eine dezentrale Kollisionsvermeidung ist somit nicht möglich.

Lösungsansätze

Es ist offensichtlich, dass je nach Wahl der PAN Parameter unterschiedliche Lösungsansätze für das Beacon-Scheduling-Problem nötig sind. Im folgenden werden drei Ansätze vorgestellt, die für unterschiedliche Szenarien geeignet sind.

Zeitliche Gleichverteilung

Als erster Ansatz wird hier die stochastische Gleichverteilung der Koordinatoren auf die Zeitfenster angeführt. Es ist durch die MAC-Schicht sichergestellt, dass zwei assoziierte Knoten nicht gleichzeitig ihren Beacon senden. Weitere Zusicherungen sind nicht vorhanden.

Damit ist keine Verhinderung von Beaconkollisionen garantiert. Ein solcher PAN-Cluster sollte die untere Grenze der erreichbaren, erfolgreich assoziierten Knoten in einem Cluster markieren.

Zeitmultiplex

Durch die Wahl des *Beacon Intervals* und der *Superframe Duration* ist die maximale Zahl von Zeitfenster über das *Beacon Interval* vorgegeben.

Wenn für den Anwendungsfall größere Latenzen und ein geringerer Durchsatz genügt, kann bei entsprechender Wahl von *Beacon Order*, *Superframe Order* und der Baumparamter L_m und C_m jedem Koordinator des PANs ein separates Zeitfenster zugeordnet werden.

Die Anzahl der Koordinatoren n_{Coord} lässt sich nach Gleichung 3.1 analog nach Anzahl der Knoten in einem Baum [Neg06] berechnen.

$$n_{Coord} = \frac{1 - C_m^{L_m}}{1 - C_m} \tag{3.1}$$

Die Anzahl der maximal verfügbaren Zeitfenster n_{TS} während eines *Beacon Intervals* berechnet sich nach Gleichung 3.2 aus dem Verhältnis der Dauer des *Beacon Intervals* zur Dauer der *Superframe Duration*.

$$
\begin{aligned}
n_{TS} &= \frac{BI}{SD} \\
&= \frac{aBaseSuperframeDuration * 2^{BO}}{aBaseSuperframeDuration * 2^{SO}} \\
&= 2^{BO-SO}
\end{aligned}
\tag{3.2}
$$

$aBaseSuperframeDuration$ ist eine Konstante, die für die MAC-Schicht definiert ist. Für ein reines Zeitmultiplex der Beacons muss die Anzahl der Koordinatoren kleiner gleich der Anzahl der Zeitfenster sein. Dies führt zu der Ungleichung 3.3.

$$
\begin{aligned}
n_{Coord} &\leq n_{TS} \\
&\leq 2^{BO-SO}
\end{aligned}
\tag{3.3}
$$

Die größtmögliche *Beacon Order* ist mit 14 spezifiziert. Somit ergibt sich bei einer minimalen *Superframe Order* von 0 die maximale Anzahl von $2^{14} = 16384$ Koordinatoren, die mit reinem Zeitmultiplex in einem Netzwerk betrieben werden können.

Zu beachten bleibt, dass für eine höhere Anzahl von erlaubten Koordinatoren größere Latenzen und ein geringerer Durchsatz durch größere *Beacon Intervals* bzw. kürzere *Superframe Durations* in Kauf genommen werden müssen.

Für den Extremfall einer maximalen BO im 2,4 GHz-Band lässt sich mit der Gleichung 3.4 die größtmögliche Dauer für ein BI berechnen.

$$
\begin{aligned}
BI_{max} &= aBaseSuperframeDuration * 2^{BO_{max}} \\
&= 960\,sym * 16384 \\
&= \frac{15728,64k\,sym}{62,5\,\frac{ksym}{s}} \\
&\approx 251,2\,s
\end{aligned}
\tag{3.4}
$$

$aBaseSuperframeDuration$ ist mit 960 Symbolen definiert. Über die Symbolrate für das 2,4 GHz-Band aus Tabelle 2.1 kann dann die maximal mögliche Dauer eines *Beacon Interval* BI_{max} bestimmt werden.

Bei einem *Beacon Interval* von über 4 Minuten treten große Verzögerungen in der Datenübertragung und beim Netzaufbau auf. Der Verbindungsverlust eines Teilnehmers zu einem Koordinator wird zum Beispiel erst nach über 12 Minuten erkannt.

Ein weiteres wichtiges Kriterium für das Zeitmultiplex-Verfahren, ist die Verteilung der Beacons über die Zeitfenster. Eine ungünstige Verteilung kann zu einer hohen Latenz bei Multihop-Übertragungen führen, da im schlechtesten Fall nur ein Hop pro *Beacon Interval* übertragen werden kann.

Es wird davon ausgegangen, dass voranging Verkehr von den Knoten zum Koordinator gesendet wird. Somit ist es günstig, wenn die vom PAN-Koordinator am weitest entfernten Koordinatoren vor den übergeordneten Koordinatoren in einem *Beacon Interval* senden. Damit ist sichergestellt, dass ein Rahmen von einem beliebigen Knoten innerhalb eines *Beacon Intervals* bei fehlerfreier Übertragung den PAN-Koordinator erreicht.

Zeit- und Raummultiplex

Um die offensichtlichen Nachteile eines reinen Zeitmultiplex zu kompensieren soll die Sendereichweite der Koordinatoren mit in Betracht gezogen werden. Damit ist es möglich, dass zwei Koordinatoren gleichzeitig einen Beacon ohne Kollision senden, solange sie sich räumlich nicht überlagern.

Die räumliche Abdeckung der Koordinatoren ist nicht bekannt und muss erst ermittelt werden. Die Reichweite der Koordinatoren wird empirisch durch den Empfang von Beacons angenähert.

Ausgehend von diesen Informationen kann die Verteilung der Beacons über das *Beacon Interval* auf das *Vertex Coloring Problem* überführt werden.

Das *Vertex Coloring Problem* beschreibt einen Graphen, in dem jeder Knoten eine andere Farbe erhält als seine durch Kanten verbundenen Nachbarn.

Für den Fall des *Beacon-Scheduling* soll also jeder Koordinator ein unterschiedliches Zeitfenster erhalten als die Koordinatoren, mit dessen Beacons er sich direkt oder indirekt überschneiden kann.

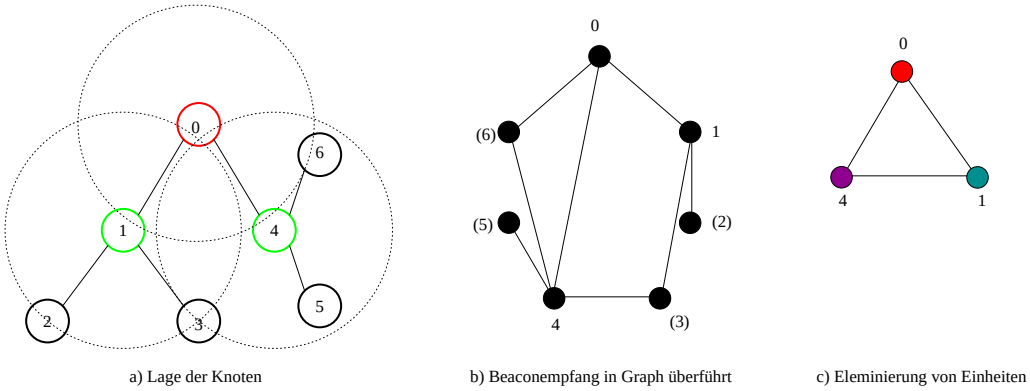

Abbildung 3.9: Überführung eines Clusters auf das *Vertex Coloring Problem*

In Abbildung 3.9 ist in Bild a) die physische Lage der Knoten mit dem logischen Cluster-Tree dargestellt. Bei der Überführung in einen Graph in Bild b) sind die Netzknoten als Knoten im Graph wiederzufinden. Die Kanten des Graphes stehen für den Empfang eines Beacons von einem Knoten durch einen anderen. Der Graph besteht also aus Kanten des Cluster-Trees und zusätzlichen Kanten.

Dieser Graph kann im weiteren Schritt reduziert werden, denn Teilnehmer benötigen kein Zeitfenste zum Senden eines Beacons. Teilnehmer sind in Bild b) eingeklammert und werden entfernt, anliegende Kanten werden miteinander verbunden.

Nach dieser Transformation ist der Graph in Bild c) mittels eines Algorithmus zur Lösung des *Vertex Coloring Problems* zu behandeln. Die Farben stellen hier die unterschiedlichen Zeitfenster über das *Beacon Interval* dar.

Dieser Ansatz arbeitet unabhängig von der Größe des Baumes, erfordert jedoch einen erhöhten Aufwand und kann eine vollständige Kollisionsfreiheit nur für stationäre Knoten garantieren. Damit der Baum in den Graphen abgebildet werden kann, müssen die Knoten im Cluster Informationen über die von ihnen empfangenen Beacons mittels eine *Beacons scanned*-Rahmen an den PAN-Koordinator senden. Außerdem muss der PAN-Koordinator den Koordinatoren im Netz Zeitschlitze mittels *Beacon config*-Rahmen zuweisen.

Das *Vertex Coloring Problem* gehört zur (aus der Komplexitätstheorie bekannten) Klasse der NP[1]-schweren Probleme. Es existiert also kein effizienter Algorithmus mit dem sich das Problem berechnen lässt. In der Implementierung wurde somit auf eine Heuristik zurückgegriffen. Die Beschreibung der Heuristik ist in Abschnitt 4.2.2 zu finden.

[1]Non-deterministic Polynomial-time

3.6 Energieverbrauch

Ein tragendes Argument für die Verwendung eines Cluster-Tree-Protokolls auf Grundlage von IEEE 802.15.4 war die zu erreichende Energieeffizienz.

Dabei haben die verschiedenen Knoten in einem PAN unterschiedliche Wach- und Schlafzyklen:

Knotentyp	Wachzeiten
PAN-Koordinator	eigene SD
Koordinator	eigene SD und die ihres Koordinators
Teilnehmer	partiell die SD ihrer Koordinatoren

Tabelle 3.7: Wachzeiten der Knotentypen

Dabei ist zu beachten, dass Teilnehmer von ihrem Koordinator nicht die gesamte SD wach sein müssen. Der Wachzyklus endet, sobald die Transaktionen von Teilnehmern und Koordinator abgearbeitet sind. Falls keine Transaktionen vorhanden sind bedeutet dies, dass sofort nach Empfangen des Beacons der Teilnehmer wieder in den Ruhezustand übergehen kann. Bei dieser Betrachtung wird davon ausgegangen, dass das Netz aufgebaut ist und sich in einem stabilen Zustand befindet. Bei Verbindungsverlusten und den darauf folgenden Abläufen gelten diese Betrachtungen nicht mehr.

Hier wird deutlich, dass der Energieverbrauch maßgeblich durch das *Beacon Interval* und die *Superframe Duration* gesteuert wird.

Für den PAN-Koordinatoren kann der Wachanteil w_{PCoord} aus dem Verhältnis von *Superframe Duration* zu *Beacon Interval* nach Gleichung 3.5 berechnet werden.

$$\begin{aligned} w_{PCoord} &= \frac{SD}{BI} \\ &= \frac{aBaseSuperframeDuration * 2^{SO}}{aBaseSuperframeDuration * 2^{BO}} \\ &= 2^{SO-BO} \end{aligned} \tag{3.5}$$

Daraus lässt sich der maximale Wachanteil von 100 % für den Fall, dass *Beacon Order* gleich der *Superframe Order* ist, ermitteln. Der minimale Wachanteil tritt bei größtmöglicher *Beacon Order* und kleinstmöglicher *Superframe Order* auf:

$$
\begin{aligned}
w_{PCoord_{min}} &= 2^{SO_{min}-BO_{max}} \\
&= 2^{0-14} \\
&= 0,006\,\%
\end{aligned}
\tag{3.6}
$$

Die ermittelten Ergebnisse lassen sich näherungsweise auch für reine Teilnehmer übernehmen, wenn man vom ungünstigen Fall ausgeht, dass sie für die komplette Dauer des *Superframes* Daten zu senden und zu empfangen haben. Damit lässt sich der minimale Wachanteil für Koordinatoren wie folgt bestimmen:

$$
\begin{aligned}
w_{Coord_{min}} &= 2 * w_{PCoord_{min}} \\
&= 0,012\,\%
\end{aligned}
\tag{3.7}
$$

Das Erreichen des minimalen Wachanteils hat einen negativen Einfluss auf die Dienstgüte des Netzes. Die nötige Zeit für den Aufbau des Netzes und die Latenz steigt (siehe Abschnitt 3.5). Außerdem sinkt die Datenrate, die das Netz transportieren kann.

4 Implementierung

In diesem Kapitel wird beschrieben, wie das Protokoll aus Kapitel 3 im ns-2 umgesetzt wurde und welche Anpassungen des Programmes nötig waren.

4.1 Tcl-Schnittstelle

Um das implementierte Protokoll zu verwenden, wurde das Routingprotokoll mit dem Namen *ZBR* in ns-2 implementiert. Das lässt sich dann über die Knotenkonfiguration verwenden:

```
$ns_ node-config -adhocRouting ZBR ...
```

Das Routingprotokoll ZBR stellt eine allgemeine Schnittstelle zu Routingprotokollen, die auf ZigBee-Netzen verwendet werden, dar. Das hier beschriebene Cluster-Tree-Protokoll ist das erste darauf implementierte Protokoll.

Für die unterschiedlichen Ansätze im Beacon-Scheduling-Problem (siehe Kapitel 3.5) werden die in Tabelle 4.1 aufgelisteten Cluster-Tree-Routing-Ansätze zur Verfügung gestellt. Die Unterschiede beziehen sich ausschließlich auf das Beacon-Scheduling.

Protokollname	Beschreibung
`CTUS`	Cluster-Tree-Routing, Beacons zeitlich gleichverteilt (siehe Abschnitt 3.5)
`CTPS`	Cluster-Tree-Routing, Beacons im Zeitmultiplex (siehe Abschnitt 3.5)
`CTCS`	Cluster-Tree-Routing, Beacons im Zeit- und Raummultiplex (siehe Abschnitt 3.5)

Tabelle 4.1: Implementierte Cluster-Tree-Routing-Ansätze

Natürlich müssen alle Knoten eines Clusters den gleichen Algorithmus für das Beacon-Scheduling verwenden. Dieser lässt sich zentral mit folgendem Befehl einstellen:

```
Agent/ZBR rt CTxx
```

Dabei ist für `CTxx` das entsprechende Protokoll aus Tabelle 4.1 zu wählen. Für alle drei implementierten Protokolle sind noch die Baumparameter C_m und L_m (siehe Kapitel 2.3) festzulegen:

```
Agent/ZBR rp $Cm $Lm
```

Um die ZigBee-Routing-Erweiterung der Knoten zu steuern, wurden die Knoten um die in Tabelle 4.2 aufgelisteten Funktionsaufrufe erweitert.

Funktionssyntax	Beschreibung
`zbr startPANCoord [BO=6] [SO=3]`	Startet einen Knoten als PAN-Koordinator. Als optionale Parameter können die BO und die SO angegeben werden.
`zbr startDevice [isFFD=1]`	Startet einen Knoten. Falls der Parameter den Wert 0 hat, handelt es sich um einen Teilnehmer-, für den Wert 1 um einen Koordinatorknoten.
`zbr disassociate`	Lässt den Knoten sich explizit disassoziieren.

Tabelle 4.2: Knotenfunktionen der ZigBee-Routing-Erweiterung

Ein Beispiel für ein komplettes Tcl-Skript ist im Anhang B zu finden.

Für die in Abschnitt 2.2 beschriebenen auf der Netzwerkschicht aufbauenden Schichten im ZigBee-Protokollstapel gibt es keine Entsprechungen im ns-2. Da hier der Schwerpunkt auf der Simulation der MAC- und Netzwerkschicht liegt und keine konkreten Anwendungen vorliegen, erfolgte auch keine Implementierung der Primitiven der NLDE[1]- bzw. NLME[2]-Schnittstelle. Die Konfiguration und der Start der Knoten geschieht mittels der Kommandos aus Tabelle 4.2.

[1] Network Layer Data Entity
[2] Network Layer Management Entity

4.2 Erweiterung von ns-2

In diesem Punkt sind die Erweiterungen und Änderungen an den internen Klassen von ns-2 besprochen.

4.2.1 Eingeführte Klassen

Für die Implementierung wurde eine Reihe von neuen Klassen, die in einem UML[3]-Diagramm in Abbildung 4.1 dargestellt sind, eingeführt.

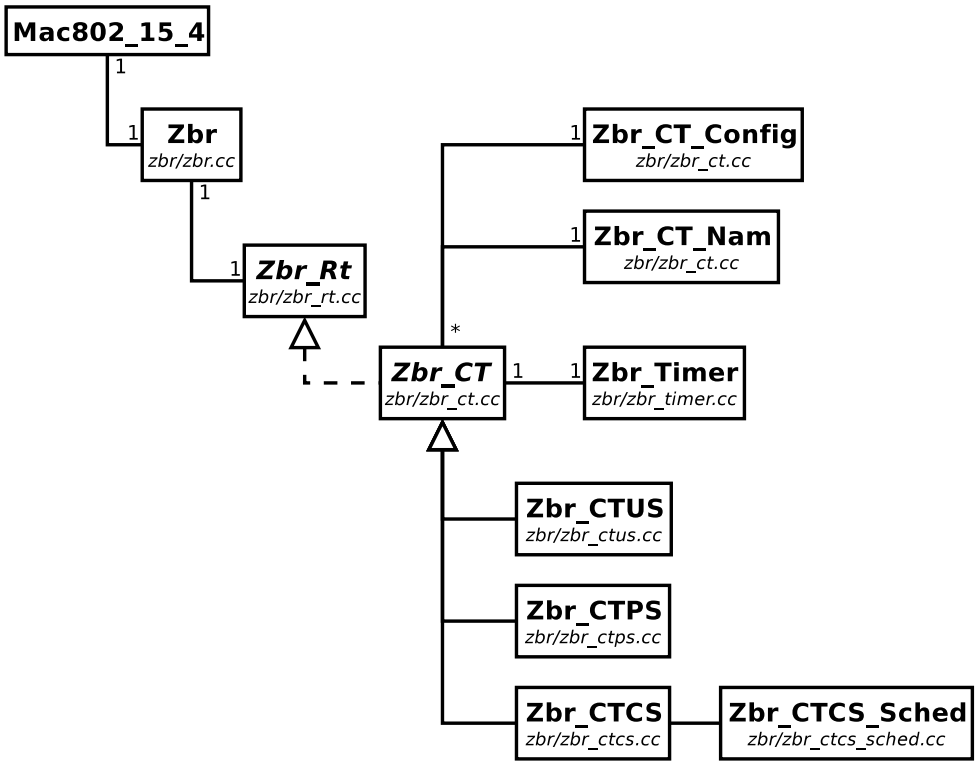

Abbildung 4.1: Klassen der Implementierung

Im Folgenden sind die Klassen und ihre Funktionen beschrieben. Der Quelltext ist komplett auf der dieser Arbeit beiliegenden CD-ROM enthalten (siehe Anhang A).

[3]Unified Modeling Language

Klasse `Zbr`

Diese Klasse stellt die Schnittstelle zu ns-2 her und wird vom existierenden WPAN-Stapel in ns-2 aufgerufen. Funktionsaufrufe aus den Simulationsskripten und eingehende Rahmen werden an die entsprechend verwendete Routing-Klasse weitergeleitet. Die Routing-Klassen sind Ableitungen von der abstrakten Klassen `Zbr_Rt`.

Klasse `Zbr_Rt`

Abstrakte Klasse, die für das Routing nötige Schnittstellenfunktionen definiert. Eine Instanz der `Zbr` Klasse benötigt eine Implementierung von `Zbr_Rt`, um Rahmen routen zu können.

Klasse `Zbr_CT`

Abstrakte Cluster-Tree-Routing-Klasse. In dieser sind bereits alle für Cluster-Tree-Routing nötigen Funktionen bis auf die Behandlung des Beacon-Scheduling implementiert.

Die Funktionen zur Ermittlung der Beaconstartzeit sind abstrakt definiert und werden je nach konfiguriertem Ansatz im Simulationsskript (siehe Kapitel 4.1) implementiert. Die folgenden drei Klassen realisieren die in Abschnitt 3.5 vorgestellten Ansätze.

Klasse `Zbr_CTUS`

Implementiert Cluster-Tree-Routing mit zeitlich gleichverteilten Beacons (siehe Kapitel 3.5). Über die Statistik-Funktion zur Gleichverteilung von ns-2 werden die Zeitfenster für die Koordinatoren ermittelt.

Klasse `Zbr_CTPS`

Implementiert Cluster-Tree-Routing mit Beacons im Zeitmultiplex (siehe Kapitel 3.5). Es wird anhand der Baumparameter überprüft, ob jeder Koordinator ein Zeitfenster erhalten kann (siehe Kapitel 3.5). Falls nicht, wird die Simulation verweigert.

Die Zeitfenster in einem Beacon Interval werden im logischen Baum ebenenweise von unten nach oben vergeben. Damit wird realisiert, dass der PAN-Koordinator innerhalb eines *Beacon Intervals* von jedem Knoten erreichbar ist.

Klasse `Zbr_CTCS`

Implementiert Cluster-Tree-Routing mit Beacons im Zeit- und Raummultiplex (siehe Kapitel 3.5).

Klasse `Zbr_CTCS_Sched`

Diese Klasse implementiert die Funktionalität zur Lösung des *Vertex Coloring Problems* für das Beacon-Scheduling-Problem.

Das *Vertex Coloring Problem* ist als NP-schwer klassifiziert. Damit gibt es keinen Algorithmus, der das Problem effizient lösen kann. Zur Lösung wird hier der *Welsh-Powell Algorithmus* [WP67] verwendet. Die Implementierung wird im Abschnitt 4.2.2 erläutert.

Klasse `Zbr_CT_Config`

Die Klasse `Zbr_CT_Config` dient zur Berechung der Adressinformationen aus den Baumparametern C_m und L_m. Dies soll zu einer schnelleren Durchführung der Simulation führen, da alle Informationen, die bei Adressvergabe eines Knotens aus dem Baummodell benötigt werden, zu Beginn zentral berechnet wurden. Zu den berechneten Werten gehören die Ebene zu einer Knoten-ID und die Größe der Teilbäume, die unter jedem Knoten dem Netz angeschlossen werden können.

Klasse `Zbr_CT_Nam`

Zur Unterstützung der Animation mittels *nam* wird die Klasse `Zbr_CT_Nam` genutzt. Mit ihr lässt sich der Assoziierungszustand zwischen den Knoten und die Überwachung von Werten des Routingprotokolls aufzeichnen.

In *nam* wird die Animation der Simulation durch die Ereignisse in der Simulationdatei erzeugt. Diese Simulationsdatei wird dabei linear ausgelesen, in *nam* selbst wird nur der Zustand der Komponenten im Arbeitsspeicher gehalten. Damit ist es möglich beliebig

große Animationsdateien zu verarbeiten. Dies erfordert jedoch, dass beim Erstellen der Zustandsübergänge in der Animationsdatei sowohl alter als auch neuer Zustand für jedes Ereignis angegeben werden. Damit ist ein beliebiges Vorwärts- und Rückwärtsabspielen der Animation möglich. Die Speicherung der Zustandsinformation zur Simulationszeit ist in diese Klasse ausgelagert.

4.2.2 Welsh-Powell Algorithmus

Für die Berechnung des Beacon-Scheduling nach dem Zeit- und Raummultiplex (siehe Kapitel 3.5) dient eine Implementierung auf Basis des Welsh-Powell Algorithmus [WP67] zur Lösung des *Vertex Coloring Problems*. Dieser Alogrithmus ist ein optimierter Greedy-Algorithmus [Die06].

Er garantiert eine Färbung eines Graphen G mit maximal $\triangle(G) + 1$ Farben ($\triangle(G)$ ist als der maximale Grad des Graphen G definiert). Er führt nicht zwangsläufig zu einer optimalen Färbung (so wenig Farben wie möglich verwenden).

Im vorliegenden Fall kann dies dazu führen, dass nicht alle Koordinatoren ein Zeitfenster zugeordnet bekommen können, da die Anzahl der Zeitfenster durch die Wahl der Parameter, wie in Abschnitt 3.4.1 beschrieben, begrenzt ist.

Der Welsh-Powell Algorithmus läuft wie folgt ab:

1. Zu Beginn sind alle Knoten ungefärbt und werden absteigend nach ihrem Grad sortiert.
2. Die Liste der Knoten durchgehend wird jedem Knoten, der keine Farbe und keinen gleich gefärbten Nachbarknoten hat, die Farbe 1 zugeordnet.
3. Der zweite Schritt ist analog mit den weiteren Farben zu wiederholen, bis alle Knoten gefärbt sind.

Nachdem der PAN-Koordinator neue *Beacon scanned*-Rahmen (siehe Abschnitt 3.3) erhalten hat, wird zur Vergabe der Zeitfenster folgender abgewandelter Algorithmus ausgeführt:

1. Liste der Koordinatoren wird absteigend nach Anzahl der erreichbaren Koordinatoren sortiert, bereits vergebene Zeitfenster bleiben erhalten. Dem PAN-Koordinator wird das Zeitfenster 0 fest zugeteilt.

2. Für jedes weitere verfügbare Zeitfenster wird die sortierte Liste der Koordinatoren abgearbeitet. Jedem Koordinator, dem noch kein Zeitfenster zugeordnet wurde, wird ein Zeitfenster zugewiesen, solange kein benachbarter Koordinator bereits das gleiche Zeitfenster belegt.

3. Falls dies der erste Durchlauf des Algorithmus ist und einem Koordinator kein Zeitfenster zugeteilt werden konnte, werden alle Zeitfenster-Zuordnungen verworfen und der Algorithmus beginnt erneut mit dem ersten Punkt.

4. Zum Schluss müssen aus den Zeitfenstern die relativen Startzeiten der Knoten zu ihren Koordinatoren ausgerechnet werden. Dafür wird die Liste der Koordinatoren, aufsteigend nach ihrer Ebene abgearbeitet und aus den absoluten Zeiten der Zeitfenster relative Startzeiten berechnet. Falls das berechnete Ergebnis der relativen Startzeit sich von der vorherigen zugeordneten Startzeit unterscheidet, wird an den Knoten ein entsprechender *Beacon config*-Rahmen (siehe Abschnitt 3.3) geschickt. Im Falle von Koordinatoren, denen kein Zeitfenster zugeordnet werden konnte, wird der Parameter `sendBeacon` in diesem Rahmen auf den Wert 0 gesetzt.

4.2.3 Network Animator

Mit Hilfe von nam lässt sich die Simulation grafisch animiert verfolgen und der Zustand der Knoten zu jedem Simulationszeitpunkt examinieren.

Die WPAN-Erweiterung von ns-2 enthält bereits die Funktionalität, um die Knotenfarbe an den Knotenzustand anzupassen und die Koordinatorenadresse jedes Teilnehmers anzuzeigen. Diese Funktionalität wurde unverändert beibehalten.

Mittels der Klasse `Zbr_CT_Nam` wurden weitere Animationen hinzugefügt. Der Zustand von Teilnehmer-Koordinator-Beziehungen wird nun über die Farbe der Verbindungen zwischen zwei Knoten dargestellt. Die verwendeten Farben und ihre Bedeutung sind in Tabelle 4.3 aufgezählt.

Farbe	Bedeutung
farblos	keine Verbindung zwischen den Knoten
orange	Beginn einer Assoziierung zwischen Teilnehmer und Koordinator
rot	Abweisung einer Assoziierung vom Koordinator an den Teilnehmer
grün	erfolgreiche Assoziierung

Tabelle 4.3: Bedeutung Verbindungsfarben

Zusätzlich wird jedem Knoten ein so genannter Agent von der verwendeten Routingklasse zur Verfügung gestellt. Dieser Agent dient zur Überwachung der in Tabelle 4.4 aufgelisteten Parameter der ihm zugeordneten Routingklasse.

Parameter	Bedeutung
AssociationStatus	Enthält den letzten Statuswert der `MLME-ASSOCIATE.confirm`-Primitive.
isFFD	Enthält den Wert 1, falls es sich um einen FFD-Knoten handelt, sonst den Wert 0.
macAssociationPermit	Gibt an, ob Assoziierungen auf der MAC-Schicht erlaubt sind oder nicht.
NodeID	Gibt die aktuelle Knoten-ID an.
ParentNodeID	Knoten-ID des Koordinators des Knotens.
StartTime	Enthält Beaconstartzeit relativ zum Beacon des Koordinators in Symbolen.

Tabelle 4.4: Überwachte Parameter

Die Abbildung 4.2 zeigt ein Bildschirmfoto einer geladenen Simulation im *nam*, wobei die überwachten Agenten angezeigt werden.

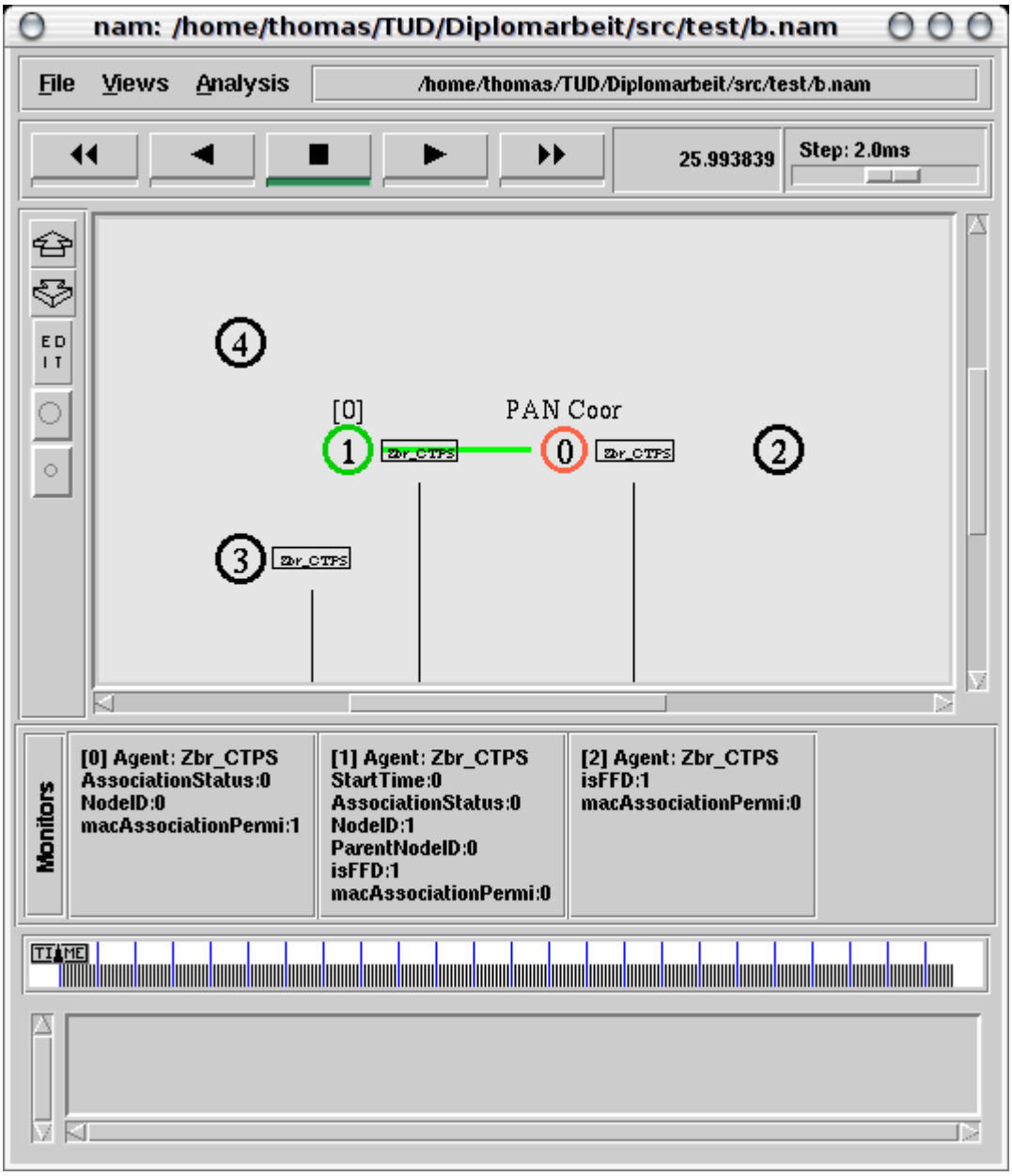

Abbildung 4.2: nam Bildschirmfoto

4.2.4 Änderungen an ns-2

Neben den neu eingeführten Klassen waren einige Änderungen im vorhandenen Quellcode von ns-2 nötig.

Abschaltung ARP

Der Simulator ns-2 ist für die Verwendung von IP ausgelegt. Die existierenden Anwendungsagenten zum Erzeugen von Netzwerkverkehr verwenden zur Adressierung IP-Adressen. Es existieren keine auf dem ZigBee-Protokollstapel aufbauenden Anwendungsagenten. Um die existierenden Agenten verwenden zu können, wird die Knotenadresse als IP-Adresse interpretiert.

Bei der Verwendung des IP-Protokolls führt ns-2 automatisch die entsprechenden Funktionen des ARP[4]s durch, um die einer IP-Adresse zugeordneten MAC-Adresse zu ermitteln. Für die Simulation mit dem *ZBR*-Protokoll muss dies jedoch deaktiviert werden, da die Adressen direkt aus den IP-Adressen abgeleitet werden. Wenn ARP aktiviert bleibt, werden sonst zusätzliche Pakete für das ARP gesendet und verfälschen damit die Simulation. Zur Abschaltung von ARP reicht folgende Änderung der Methode `ARPTable::arpresolve` in der Datei `mac/arp.cc`:

```
int
ARPTable::arpresolve(nsaddr_t dst, Packet *p, LL *ll)
{
  /* Disable ARP for ZBR */
  mac_->hdr_dst((char*) HDR_MAC(p), dst);
  return 0;
}
```

Damit werden alle ARP-Anfragen für eine IP-Adresse mit der identischen MAC- bzw. Knotenadresse beantwortet.

Pakettyp

ns-2 hat eine interne Liste mit den unterstützten Pakettypen. Für die ZigBee-Rahmen wurde in *common/packet.h* der neue Pakettyp `PT_ZBR` registriert.

[4]Address Resolution Protocol

Tcl-Schnittstelle

In der Tcl-Bibliothek, über die ns-2 angesprochen wird, musste für den neuen Routing-Agenten eine entsprechende Schnittstelle geschaffen werden.

In der Datei *tcl/lib/ns-mobilenode.tcl* wurde folgende Methode hinzugefügt:

```
Node/MobileNode instproc zbr args {
  $self instvar mac_
  eval $mac_(0) zbr $args
}
```

Die Registrierung des Routing-Agentens erfolgt in der Datei *tcl/lib/ns-lib.tcl* durch Definition einer Funktion zum Erzeugen einer neuen Instanz des Routing-Agenten:

```
Simulator instproc create-zbr-agent { node } {
  #  Create ZigBee Routing agent
  set ragent [new Agent/ZBR [$node node-addr]]
  $self at 0.0 "$ragent start"
  $node set ragent_ $ragent
  return $ragent
}
```

In der Funktion `create-wireless-node` in der gleichen Datei wird dann bei Erstellen von ZBR-Knoten die Funktion `create-zbr-agent` aufgerufen.

Kopplung WPAN und ZBR Klassen

Das entwickelte Routing-Modul ist fest an das WPAN Modul als MAC-Schicht gebunden. Dafür wurde in der Klasse `SSCS802_15_4`, bei der Verwendung von `ZBR` als Routingprotokoll, die Behandlungsroutinen für die Primitiven der Netzwerkschicht an die Methoden der Klasse `ZBR` weitergereicht.

Anpassung WPAN

Die Implementierung der MAC-Schicht IEEE 802.15.4 basiert auf dem Standard aus dem Jahre 2003. Im Jahr 2006 ist jedoch eine überarbeitete Version erschienen. Ein Großteil der Änderungen [IEE06] betrifft die kryptografischen Funktionen, die im Simulator

nicht von Belang sind. Eine wichtige Änderung hat die Primitive `MLME-START.request` erhalten. In der überarbeiteten Version ist es möglich die Startzeit des Beacons eines Knoten relativ zum Beacon des dazugehörigen Koordinators festzulegen. Dieser Mechanismus wird für das Beacon-Scheduling-Problem aus Abschnitt 3.5 benötigt und wurde implementiert.

Die der `MLME-START.request`-Primitive übergebene Startzeit des Beacons wird in der neu eingeführten Variable `beaconStartTime` der Klasse `Mac802_15_4` gespeichert. Die Methode `Mac802_15_4::recvBeacon`, die ausgeführt wird, wenn ein Beacon vom Koordinator empfangen wurde, wurde wie folgt erweitert:

```
void Mac802_15_4::recvBeacon(Packet *p)
{
  // ...

  // sync Tx beacon on Rx beacon
  if (beaconStartTime > 0) {
    if (!bcnTxT->busy()) {
      bcnTxT->start(true,true,beaconStartTime);
    }
  }

  // ...
}
```

Damit wird beim Empfang des Beacons ein Timer gestartet, der nach Ablauf das Aussenden des Beacons ausführt.

4.3 Fehler in ns-2

Bei der Implementierung wurden einige Fehler in der existierenden Implementierung des IEEE 802.15.4 Standards in ns-2 festgestellt. Diese Fehler treten im *beacon enabled mode* auf. Dieses Kapitel gibt eine Übersicht über die gefundenen Fehler und beschreibt, welche davon behoben werden konnten. Für die offenen Fehler wird beschrieben, wo sie konzeptionell begründet liegen, und welche Änderungen durchgeführt werden müssen, um eine standardkonforme Implementierung zu erhalten.

4.3.1 Geschlossene Fehler

Folgende Fehler wurden in ns-2 erkannt und konnten behoben werden.

Passive Scans

Es stellte sich heraus, dass die Implementierung der `MLME-SCAN.request`-Primitive für *Passive Scans* einen Fehler enthielt, der zu einer Nullzeiger-Dereferenzierung geführt hat, enthielt. Zur Korrektur wurde die Methode `Mac802_15_4::scanHandler` wie folgt verändert:

```
void Mac802_15_4::scanHandler(void)
{
  if ((taskP.mlme_scan_request_ScanType == 0x01) ||
      (taskP.mlme_scan_request_ScanType == 0x02))
    taskP.taskStep(TP_mlme_scan_request)++;
  dispatch(p_SUCCESS,__FUNCTION__);
}
```

Damit war es nicht mehr nötig *Active Scans* durchzuführen, die durch zusätzliche *Beacon request*-Rahmen die Simulation verfälscht hätten.

`MLME-DISASSOCIATE`-Primitiven

Die Primitiven `MLME-DISASSOCIATE` wurden in der Implementierung falsch verwendet. So war in der Klasse `Mac802_15_4`, die die MAC-Schicht repräsentiert, eine Primitive `MLME-DISASSOCIATE.indication` definiert. Eine Indikation der MLME[5]-Schnittstelle ist jedoch immer eine Primitive der Netzwerkschicht.

Die Deklaration wurde entfernt und in die Schnittstelle zur Netzwerkschicht eingefügt. In der Behandlungsroutine für die MAC-Rahmen wurde der entsprechende Aufruf der `MLME-DISASSOCIATE.indication`-Primitive der Netzwerkschicht implementiert.

[5]Medium Access Control Sub-Layer Management Entity

4.3.2 Offene Fehler

Bei Simulationstests während der Entwicklungszeit stellte sich heraus, dass die Rahmen im *beacon enabled mode* direkt wie im *non-beacon enabled mode* gesendet werden. Im *beacon enabled mode* müssen Rahmen jedoch in Transaktionen abgearbeitet werden.

`recv`-Methode

Agenten, die in ns-2 auf einem Knoten laufen, rufen dessen `recv`-Methode auf, um Rahmen zu schicken. Die existierende Implementierung dieser Methode in der Klasse `Mac802_15_4` führte jedoch immer zu einem Aufruf der Primitive `MCPS-DATA.request` für das direkte Senden. Der Funktionaufruf wurde entsprechend angepasst, um im *beacon enabled mode* ein indirektes Senden zu veranlassen:

```
void Mac802_15_4::recv(Packet *p, Handler *h)
{
        // ...

        if (p802_15_4macDA(p) == (nsaddr_t)MAC_BROADCAST)
          txop = 0;
        else
        {
          if (Mac802_15_4::ack4data)
            txop = TxOp_Acked;
          else
            txop = 0;

          txop |= Mac802_15_4::txOption;
        }

        /* Packet to the parent? */
        if (p802_15_4macDA(p) == mpib.macCoordExtendedAddress) {
          /* Parent node in beacon enabled mode? */
          if (macBeaconOrder2 < 15)
            txop |= TxOp_Indirect;
        }
        /* Node in beacon enabled mode? */
        else if (mpib.macBeaconOrder < 15)
          txop |= TxOp_Indirect;
```

```
        wph->msduHandle = 0;
        MCPS_DATA_request(0,0,0,defFrmCtrl_AddrMode16,mpib.macPANId,
                          p802_15_4macDA(p),ch->size(),p,0,txop);

        // ...
}
```

Es wird vor dem Aufruf der `MCPS_DATA_request`-Methode überprüft, ob der Rahmen an den Koordinator gesendet werden soll, und ob dieser mit Beacons arbeitet. Falls der Rahmen nicht an der Koordinator gesendet werden soll, wird geprüft, ob der Knoten selbst Beacons aussendet. In beiden Fällen wird bei Verwendung von Beacons eine indirekte Übertragung eingeleitet.

`MCPS_DATA_request`-Methode

Trotz Korrektur der `recv`-Methode konnten Daten nur teilweise korrekt übertragen werden. Es wurde festgestellt, dass bei der Handhabung der Transaktionen grundsätzliche Fehler enthalten sind.

Beim Verwalten der Transaktionen wird nicht zwischen Transaktionen aus der Teilnehmersicht und Transaktionen aus der Koordinatorsicht unterschieden. Nach der im vorherigen Punkt angesprochenen Korrektur funktioniert nur die Übertragnung von Transaktionen aus der Koordinatorsicht. Zur Lösung des Problems müssen folgende Punkte realisiert werden.

Transaktionen müssen in Transaktionen, die an die Teilnehmer, und Transkationen, die an den Koordinator des Knotens gesendet werden sollen, unterschieden werden. Es muss also ein zweiter Transaktionspuffer geschaffen oder die Funktionalität des bestehenden Puffers entsprechend erweitert werden.

In der Implementierung der `MCPS-DATA.request`-Primitive müssen die Rahmen im Falle des *beacon enabled mode* je nach Ziel entsprechend dem Transaktionspuffer zugeführt werden.

Die Methode `Mac802_15_4::recvBeacon`, die vom Koordinator empfangene Beacons verarbeitet, muss entsprechend erweitert werden, um Transaktionen, die zum Koordinator gesendet werden sollen, abzuarbeiten. Dies bedeutet, dass beim Empfang des Beacons neben einer eventuell nötigen `MLME-POLL.request`-Primitive auch noch Transaktionen

innerhalb der Dauer der SD eingeplant werden müssen. Dafür müssen die Callback-Funktionen und der Übergang in den Schlafzustand angepasst werden.

Diese nötigen Änderungen an der komplexen `Mac802_15_4`-Klasse ließen sich nicht im zeitlichen Rahmen der Arbeit durchführen. Durch das fehlerhafte Konzept bei der ursprünglichen Implementierung bedarf es vieler Anpassungen, die gründlich überprüft werden müssen.

Durch diesen Fehler ist es nicht möglich, dass Datenrahmen zwischen Knoten ausgetauscht werden. Außerdem können Knoten, die versucht haben Datenrahmen zu senden, durch die fehlerhafte Implementierung nicht mehr mit anderen Knoten kommunizieren. Aus diesen Gründen kann eine Simulation des Protokolls nur theoretisch betrachtet werden.

5 Simulation

In diesem Kapitel sind die Ansätze für die Simulation des Protokolls beschrieben. Ein Teil der in Abschnitt 4.3 beschriebenen Fehler konnte wärend dieser Arbeit korrigiert werden. Auf Grund der in Unterpunkt 4.3.2 beschriebenen konzeptionellen Fehler war es im zeitlichen Rahmen dieser Arbeit nicht möglich, eine Simulation des hier entworfenen Protokolls durchzuführen.

5.1 Szenarien

Die Simulation soll zeigen, ob das entwickelte Protokoll funktioniert und wie sich die Parameterwahl auf die Netzgüte auswirkt. Als Rahmenbedingung wären vorrangig Szenarien mit stationären Knoten auszuwählen. Bei mobilen Knoten ist damit zu rechnen, dass das Protokoll um Mechanismen erweitert werden muss, um ein schnelles Umhängen von Bäumen bei sich bewegenden Koordinatoren zu erlauben.

Im Dokument CRUISE[1] M112.3 [CRU07] wurden verschiedene spezifische Szenarien zur Simulation von Sensornetzen definiert. Das Dokument ist Vorlage für die Bearbeitung von *Task 123.3 Development of common scenarios and study cases*, welche konkrete vergleichbare Simulationszenarien schaffen soll.

Die Szenarien werden dabei mit folgenden Parametern beschrieben:

- Routing-Protokoll
- Netzwerktopologie
- Ereignisse (PAN-, Knoten- und Anwendungsstartzeiten)
- Ausbreitungsmodell

[1] CReating Ubiquitous Intelligent Sensing Environments

- Verkehrsmuster
- Superframe-Struktur bei Betrieb im *beacon enabled mode*

Die Szenarien sind dabei für unterschiedliche Protokolle und Anwendungsfälle definiert. Für dieses Protokoll sind Szenarien mit folgenden Parametern interessant:

- Stern- oder Baumtopologie
- Betrieb im *beacon enabled mode*
- keine oder nur geringe Mobilität der Knoten

Die drei ausgewählten Szenarien werden im folgenden beschrieben.

Target Tracking

Das Szenario umfasst eine Fläche von 150 m x 500 m. Auf dieser Fläche sind 100 Knoten auf feste Positionen verteilt. Die Knoten besitzen eine Reichweite von 25 m bis 50 m und sind nicht mobil. Alle Sensoren senden Daten an einen Knoten. Für die Simulation soll der PAN-Koordinator der Empfänger sein.

Flooding Detection

Bei diesem Szenario sollen 3 bis 500 Knoten auf eine $1\,m^2$ bis mehrere km^2 umfassende Fläche manuell platziert werden. Die Reichweite der Knoten soll zwischen 1 m bis 100 m liegen. Die Knoten sind stationär. Als Datenaufkommen wird ein geringer CBR[2] oder VBR[3] Verkehr angenommen. Die Datenpaketgrößen liegen bei 1 bis 5 Byte. Als Simulationszeit kommen wenigen Sekunden bis Jahre in betracht.

Fire Detection

Es sind 3 bis 500 Knoten wiederum auf eine $1\,m^2$ bis mehrere km^2 umfassende Fläche gleichmäßig zu verteilen. Die Reichweite der Knoten liegt wieder zwischen 1 m bis 100 m.

[2]Constant Bit Rate
[3]Variable Bit Rate

Die Knoten sind nicht mobil. Als Simulationszeit werden Jahre angenommen. Es ist ein nur geringes Datenaufkommen zu simulieren.

5.2 Messparameter

Es werden aus den Simulationsdurchläufen verschiedene Parameter zur Beurteilung des Protokolls aufgezeichnet. Von Interesse ist dabei unter anderem die Auswirkung der verschiedenen Beacon-Scheduling-Ansätze, die in Abschnitt 3.5 beschrieben wurden.

Konnektivität

Unter der Konnektivität wird die Anzahl der Knoten, die eine Verbindung zum PAN-Cluster besitzen, im Verhältnis zur Anzahl der Knoten insgesamt verstanden. Sie wird durch die Baumparameter L_m und C_m (siehe Kapitel 2.3) beeinflusst. Außerdem beeinflussen mögliche Störungen durch Beacon-Kollisionen die Konnektivität (siehe Abschnitt 3.5).

Mittlere Verbindungszeit

Unter der mittleren Verbindungszeit wird das Mittel über die Zeiten verstanden, die die Knoten benötigen, um dem PAN-Cluster beizutreten. Offensichtlich wird diese Zeit auch maßgeblich durch die in Abschnitt 2.1 beschriebene Dauer von BI und SD beeinflusst.

Energieverbrauch

Für batteriebetriebene Knoten ist ein möglichst geringer Energieverbrauch wichtig. Hier wird nur der Energieverbrauch durch den Transmitter beachtet - Energieverbrauch durch Rechenoperationen kann mit ns-2 nicht simuliert werden. Offensichtlich spielt hier wieder die Dauer von BI und SD eine wichtige Rolle. Hinzu kommt der zusätzliche Aufwand für das Senden von Rahmen bei einem Betrieb mit Zeit- und Raummultiplex Beacon-Scheduling.

5.3 Ergebnisse

Wie bereits beschrieben, konnte eine Simulation nicht durchgeführt werden, deshalb werden hier die Erwartungen an die Simulationsergebnisse dargestellt.

Konnektivität

Für eine möglichst hohe Konnektivität ist davon auszugehen, dass beim reinen Zeitmultiplex die besten Ergebnisse erzielt werden. Nur bei diesem Scheduling-Ansatz ist sichergestellt, dass alle FFD-Knoten auch Beacons senden. Im Zeit- und Raummultiplex ist bei kleiner werdender Anzahl von Zeitfensten mit einer geringeren Konnektivität zu rechnen, sobald nicht mehr alle potentiellen Koordinatoren Beacons senden dürfen. Bei statistischer Gleichverteilung der Beacons über die Dauer des BI ist mit Kollisionen zu rechnen, die potentiell zur schlechtesten Konnektivität führen.

Verbindungszeit

Gerade bei einer großen Dauer des BI und einer großen Tiefe der Bäume ist bei Scheduling nach Zeit- und Raummultiplex mit großen Verzögerungen zu rechnen, da hier jeder neue potentielle Koordinator erst nach Austausch eines *Beacons scanned*- und eines *Beacon config*-Rahmen mit dem PAN-Koordinator anfängt Beacons zu senden. Die Dauer für den Transport dieser Rahmen hängt außerdem von der Baumtiefe ab. Die anderen Verfahren unterliegen keinem Einfluss durch die Baumtiefe. Die Länge der Dauer des BI setzt hier die obere Grenze für die Zeit, die nötig ist, um eine Assoziierung durchzuführen.

Energieverbrauch

Beim Zeit- und Raummultiplex ist ein erhöhter Aufwand beim Cluster-Beitritt zu verzeichnen, da hier Rahmen mit dem PAN-Koordinator ausgetauscht werden. Es ist zu erwarten, dass dieses Verfahren die meiste Energie zum Aufbau des Netzes verbraucht. Bei der statistischen Gleichverteilung der Beacons über die Dauer des BI ist damit zu rechnen, dass Knoten ihre Verbindung zum Cluster durch neue Koordinatoren verlieren können, wenn sich Beacons überlagern. Es ist deshalb mit einer geringfügig schlechter

ausfallenden Energiebilanz als bei reinem Zeitmultiplex, verursacht durch erneut nötige Assoziierungen, zu rechnen.

Bewertung

Aus den Überlegungen zu den erwarteten Simulationsergebnissen soll eine theoretische Bewertung der Ergebnisse folgen.

Für Netze, die durch entsprechende Wahl der Baumparameter (siehe Kapitel 3.4.1) C_m, L_m, BO und SO ein Scheduling der Beacons nach reinem Zeitmultiplex erlauben, bietet dieses Verfahren eine möglichst hohe Konnektivität bei zugleich möglichst geringem Energieverbrauch. Die Zeit für die Anbindung neuer Knoten wird nur durch die BO direkt beeinflusst. Ein höheres Verhältnis von BO zu SO, um eine größere Anzahl von Knoten im Netz zu erlauben, führt zwangsläufig zu längeren Verbindungszeiten, zu geringerem Durchsatz, zur einer höheren Verzögerung und zu einem geringeren Energieverbrauch.

Bei Netzen, bei denen ein reines Zeitmultiplex nicht möglich ist, bietet das Scheduling mit Zeit- und Raummultiplex eine gute Konnektivität ohne negative Einflüsse auf Verzögerung oder Durchsatz. Nachteilig ist der hier erhöhte Energiebedarf aller beteiligten Knoten.

Das statistische Verteilen der Beacons auf die Dauer des BI bringt den Kompromiss zwischen den beiden anderen Verfahren. Im ungünstigen Fall erhält man durch Kollisionen eine schlechtere Konnektivität und einen geringfügig höheren Energiebedarf als bei Zeit- bzw. Zeit und Raummultiplex. Dafür kann das Netz jedoch mit geringerer Latenz bei passender Wahl von BO und SO betrieben werden.

6 Zusammenfassung und Ausblick

Dieses Kapitel gibt eine Zusammenfassung über die Arbeit und eine Übersicht über mögliche Themen für weitere aufbauende Arbeiten.

Zusammenfassung der Arbeit

Zielstellung dieser Arbeit war es, das existierende Cluster-Tree-Protokoll an die durch die MAC- und Netzwerkschicht von IEEE 802.15.4 und ZigBee zur Verfügung gestellten Funktionen anzupassen.

Zur Umsetzung wurde der im Cluster-Tree-Protokoll spezifizierte Austausch von Rahmen durch existierende Funktionen der IEEE 802.15.4 MAC-Schicht abgebildet. Durch die Verwendung von Beacons wurde es ermöglicht, dass Knoten definierte Schlaf- und Wachzeiten besitzen. Durch die Synchronisation aller Knoten in einem Cluster entstand das *Beacon-Scheduling-Problem*. Zur Lösung wurden drei Vorschläge zur Vergabe der Beacon-Zeitfenster gemacht.

Bei der Implementierung des Protokolls in ns-2 zeigte sich, dass die Implementationen aus vorangegangenen Arbeiten nicht sinnvoll verwendet werden konnten. Es erfolgte die Implementierung eines neuen Routing-Agentens, der an die existierende IEEE 802.15.4 MAC-Schicht in ns-2 gekoppelt wurde. Die Implementierung wurde dabei so gehalten, dass spätere Arbeiten parallel integriert und zu Simulationszeit ausgewählt werden können. Dies soll den späteren Vergleich unterschiedlicher Algorithmen vereinfachen.

Während der Implementierung zeigte sich, dass für den Betrieb im *beacon enabled mode* mehrere Fehler in der MAC-Schicht enthalten waren. Im Zeitrahmen dieser Arbeit konnten nicht alle Fehler behoben werden, da sie teilweise konzeptionell bedingt sind.

Durch die Fehler in der MAC-Schicht ist es nicht möglich, dass Protokoll im *beacon enabled mode* zu betreiben, da der Austausch von Rahmen zwischen Knoten nicht funk-

tioniert. Die Simulation verschiedener Szenarien zur Bewertung des Protokolls und der Lösungsansätze des *Beacon-Scheduling-Problems* konnten deshalb nicht durchgeführt werden.

Weiterführende Arbeiten

Das hier vorgestellte Protokoll bietet Raum für weitere Arbeiten:

- **Implementierung:** Um die Implementierung dieser Arbeit verwenden zu können, ist es nötig, die IEEE 802.15.4 MAC-Schicht in ns-2 zu korrigieren. In Abschnitt 4.3.2 wurde das Problem aufgezeigt und beschrieben, welche Schritte unternommen werden müssen, um eine standardkonforme Implementation zu erhalten.
- **Inter-Cluster-Kommunikation:** Diese Arbeit beschreibt die Umsetzung des Cluster-Tree-Protokolls in für sich isolierten Clustern. Eine weiterführende Arbeit sollte Gedanken über eine mögliche Kopplung von Clustern, basierend auf dem Cluster-Tree-Protokoll, entwickeln. In den Rahmen einer solchen Arbeit fiele auch die selbstorganisierte Cluster-Formierung.
- **Rekonfiguration:** Es wurden am Lehrstuhl bereits Rekonfigurationsansätze in der Arbeit von Samperio [Sam06] für das Cluster-Tree-Protokoll umgesetzt. Eine weiterführende Arbeit muss prüfen, inwieweit die Ansätze mit dem neuen Protokoll verwendet werden können. Offensichtlich ist eine identische Implementierung nicht möglich, da einmal auf ihren Koordinator synchronisierte Teilnehmer keine Beacons von anderen Koordinatoren in ihrer Nachbarschaft empfangen.
- **Umhängen von Teilbäumen:** Für mobile Knoten wäre es wünschenswert, wenn das Protokoll ein Umhängen von Teilbäumen in einem Baum, wie in Abbildung 6.1 dargestellt, erlauben würde.

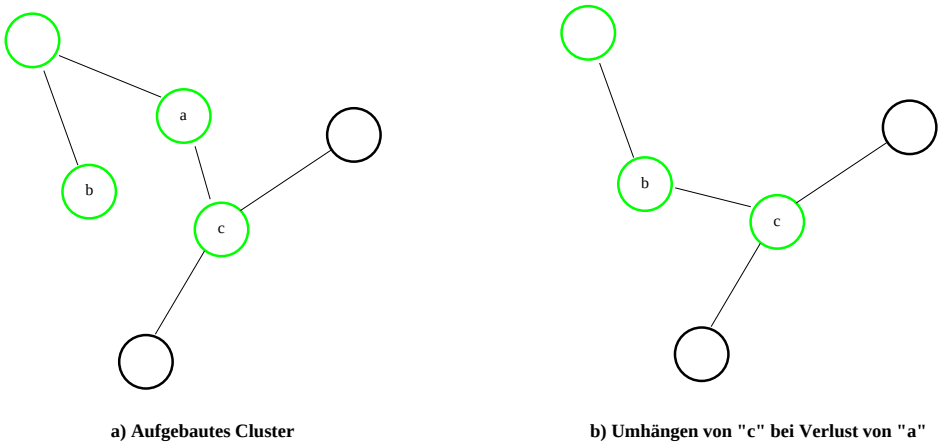

Abbildung 6.1: Umhängen von Teilbäumen

Wenn Knoten `a` den Cluster verlässt, wird in dem jetzigen Protokoll der komplette Teilbaum von `c` aufgelöst. Damit müssen alle Knoten unterhalb von `c` eine erneuten Assoziierungsprozess zum Cluster durchführen. Wenn Knoten `c` jedoch an Knoten `b` umgehangen würde und seine Kindknoten mittels eines speziellen Rahmens neue Adressen vergeben könnte, hätten die Knoten nur für kurze Zeit keine Verbindung zum Cluster und müssten keine energieaufwändigen Scans durchführen.

- **Änderungen der MAC-Schicht**

  In [KAAN07] wurde der Vorschlag einer *Beacon-Only Period* eingebracht. Vorgesehen ist, dass zu Beginn eines Superframes mehrere Zeitfenster zum Senden von Beacons vorhanden sind, erst danach verwenden alle Knoten das *slotted CSMA/CA*-Zugriffsverfahren zum Austausch von Rahmen. Das Beacon-Scheduling-Problem bleibt erhalten, hat jedoch keinen großen Einfluss mehr auf die Latenz.

  Beacon-Only Periods sind jedoch nicht Teil des IEEE 802.15.4. Die Möglichkeit einer Integration in den Standard sollte geprüft werden. Zu untersuchen ist, in wie weit sich Cluster-Tree-Topologien auf der MAC-Schicht effektiver als bisher im *beacon enabled mode* betreiben lassen und inwieweit Änderungen am IEEE 802.15.4 Standard nötig wären.

Literaturverzeichnis

[Cal01] Callaway, Ed: Cluster Tree Network / IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs). 2001. – Forschungsbericht

[CRU07] CRUISE: *M112.3 - A number of application scenarios defined and detailed in a form that can be an input for a simulation or a test-bed study.* August 2007

[Die06] Diestel, Reinhard: *Graphentheorie.* 3. Auflage. Heidelberg : Springer-Verlag, 2006. – ISBN 3-540-21391-0

[Erg04] Ergen, Sinem C.: ZigBee/IEEE 802.15.4 Summary / Wireless Sensor Networks Berkeley Lab. 2004. – Forschungsbericht

[FV06] Fall, Kevin ; Varadhan, Kannan: *The ns Manual.* `http://www.isi.edu/nsnam/ns/ns-documentation.html`: UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2006

[IEE03] IEEE: LAN/MAN Standards Committee: *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* Oktober 2003

[IEE06] IEEE: LAN/MAN Standards Committee: *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* September 2006

[KAAN07] Koubâa, Anis ; Alves, Mario ; Attia, Melek ; Nieuwenhuyse, Anneleen V.: Collision-Free Beacon Scheduling Mechanisms for IEEE 802/.15.4/ZigBee Cluster-Tree Wireless Sensor Networks / Polytechnic Institute of Porto. 2007. – Forschungsbericht

[KAT06] KOUBAA, Anis ; ALVES, Mrio ; TOVAR, Eduardo: A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks / IPP-HURRAY! Research Group, Polytechnic Institute of Porto. 2006. – Forschungsbericht

[Neg06] NEGRO, Piedad: *INVESTIGATION OF CLUSTER-TREE TOPOLOGY IN ZIGBEE NETWORKS*, TU Dresden, Diplomarbeit, Juni 2006

[Sam06] SAMPERIO, Francisco J. C.: *IMPLEMENTATION OF RECONFIGURATION ALGORITHMS IN CLUSTER TREE NETWORKS FOR IEEE 802.15.4/ZigBee*, TU Dresden, Diplomarbeit, Dezember 2006

[WP67] WELSH, D. J. A. ; POWELL, M. B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. In: *The Computer Journal* 10 (1967), Nr. 1, S. 85–86

[Zig06] ZigBee Alliance: *ZigBee-2006 Specification*. Dezember 2006

Abbildungsverzeichnis

Tabellenverzeichnis

A CD-Inhalt

Die der Arbeit beiliegenden CD-ROM enthält die Dateien die im Rahmen dieser Arbeit entstanden sind. Die Quellen von ns-2 sind sowohl im Original als auch mit den daran vorgenommenen Änderungen enthalten um einen direkten Vergleich zu ermöglichen.

Folgende Dateien sind enthalten:

- **examples.tar.gz:** Beispiel- und Testskripte, die während der Entwicklung verwendet wurden. Die Formierung von Clustern wurde mit diesen Skripten überprüft.
- **lis07_da.pdf:** Diese Diplomarbeit in digitaler Form.
- **lis07_zvert.pdf:** Folien die für die Zwischenverteidigung der Arbeit am 03. August 2007 erstellt wurden.
- **ns-2.31.tar.gz:** Unveränderter Quelltext von ns-2 in der Version 2.31. Auf dessen Basis wurde diese Arbeit implementiert.
- **ns-2.31_lis07.tar.gz:** Im Rahmen dieser Arbeit veränderter und erweiterter Quelltext von ns-2. Es sind alle Korrekturen, Änderungen und neuen Erweiterungen enthalten. Nach dem Entpacken des Quelltextes kann dieser wie im ns-2 Handbuch [FV06] beschrieben übersetzt werden.

B Simulationsbeispiel

Hier ist ein vollständiges Simulationsbeispiel gegeben. Das folgende Tcl-Skript *example.tcl* wird zur Simulation an ns-2 übergeben. Es sind 11 Knoten auf einem Raster nach Abbildung B.1 platziert. Die Platzierung erfolgt durch die Knotenpositionen in der Datei *example.scn*. Die Knoten haben eine Reichweite von 12 cm und befinden sich in 10 cm Abstand zueinander. Alle Knoten sind FFD-Knoten. Die Simulationszeit beträgt insgesamt 10 min. Die Knoten werden innerhalb der ersten 7,5 s, der PAN-Koordinator mit der Knotenadresse 0 zum Zeitpunkt 0 s gestartet.

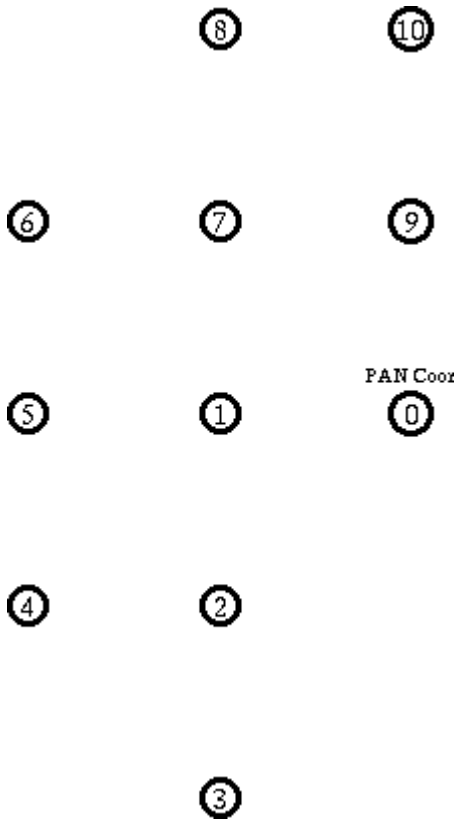

Abbildung B.1: Platzierung der Knoten

example.tcl

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# Channel Type
set val(prop)           Propagation/TwoRayGround   ;# Radio-propagation model
set val(netif)          Phy/WirelessPhy/802_15_4
set val(mac)            Mac/802_15_4
set val(ifq)            Queue/DropTail/PriQueue    ;# Interface queue type
set val(ll)             LL                         ;# Link layer type
set val(ant)            Antenna/OmniAntenna        ;# Antenna model
set val(ifqlen)         50                         ;# Max packet in ifq
set val(nn)             10                         ;# Number of mobilenodes
set val(rp)             ZBR                        ;# ZigBee Routing
set val(x)              50
set val(y)              50

set val(nam)            example.nam

# Read command line arguments
proc getCmdArgu {argc argv} {
        global val
        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
                set name [string range $arg 1 end]
                set val($name) [lindex $argv [expr $i+1]]
        }
}
getCmdArgu $argc $argv

set stopTime            600     ;# Simulation time in seconds

# Initialize Global Variables
set ns_         [new Simulator]
set tracefd     [open ./example.tr w]
$ns_ trace-all $tracefd
if { "$val(nam)" == "example.nam" } {
        set namtrace     [open ./$val(nam) w]
        $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
}
```

```
# Inform nam that this is a trace file for wpan (special handling needed)
$ns_ puts-nam-traceall {# nam4wpan #}

Mac/802_15_4 wpanCmd verbose on
Mac/802_15_4 wpanNam namStatus on

# For model 'TwoRayGround'
set dist(5m)  7.69113e-06
set dist(9m)  2.37381e-06
set dist(10m) 1.92278e-06
set dist(11m) 1.58908e-06
set dist(12m) 1.33527e-06
set dist(13m) 1.13774e-06
set dist(14m) 9.81011e-07
set dist(15m) 8.54570e-07
set dist(16m) 7.51087e-07
set dist(20m) 4.80696e-07
set dist(25m) 3.07645e-07
set dist(30m) 2.13643e-07
set dist(35m) 1.56962e-07
set dist(40m) 1.20174e-07

# Transmission range
Phy/WirelessPhy set CSThresh_ $dist(12m)
Phy/WirelessPhy set RXThresh_ $dist(12m)

# Set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create God
set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]

# Configure nodes
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
```

```
                -topoInstance $topo \
                -agentTrace OFF \
                -routerTrace OFF \
                -macTrace ON \
                -movementTrace OFF \
                -channel $chan_1_

# Setup tree parameters
set Cm 4
set Lm 3
Agent/ZBR rt CTUS
Agent/ZBR rp $Cm $Lm

puts "----------------------------\n"

for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
}

puts "----------------------------\n"

# Setup node positions
source ./example.scn

$ns_ at 0.0     "$node_(0) NodeLabel \"PAN Coor\""

# Set node start times
$ns_ at 0.0     "$node_(0) zbr startPANCoord"
$ns_ at 0.3     "$node_(1) zbr startDevice 1 1"
$ns_ at 1.3     "$node_(9) zbr startDevice 1 1"
$ns_ at 3.3     "$node_(2) zbr startDevice 1 1"
$ns_ at 3.5     "$node_(7) zbr startDevice 1 1"
$ns_ at 4.3     "$node_(3) zbr startDevice 1 1"
$ns_ at 4.5     "$node_(6) zbr startDevice 1 1"
$ns_ at 5.8     "$node_(5) zbr startDevice 1 1"
$ns_ at 6.0     "$node_(10) zbr startDevice 1 1"
$ns_ at 7.0     "$node_(4) zbr startDevice 1 1"
$ns_ at 7.3     "$node_(8) zbr startDevice 1 1"

# Defines the node size in nam
for {set i 0} {$i < $val(nn)} {incr i} {
        $ns_ initial_node_pos $node_($i) 2
}
```

```
# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stopTime "$node_($i) reset";
}

$ns_ at $stopTime "stop"
$ns_ at $stopTime "puts \"\nNS EXITING...\n\""
$ns_ at $stopTime "$ns_ halt"

proc stop {} {
    global ns_ tracefd val env
    $ns_ flush-trace
    close $tracefd
    set hasDISPLAY 0
    foreach index [array names env] {
        if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
                set hasDISPLAY 1
        }
    }

    # Start Nam if running in X11 environment
    if { ("$val(nam)" == "example.nam") && ("$hasDISPLAY" == "1") } {
        exec nam example.nam &
    }
}

puts "\nStarting Simulation..."
$ns_ run
```

example.scn

```
$node_(0) set X_ 20.0
$node_(0) set Y_ 25.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 10.0
$node_(1) set Y_ 25.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 10.0
$node_(2) set Y_ 15.0
```

```
$node_(2) set Z_ 0.0

$node_(3) set X_ 10.0
$node_(3) set Y_ 5.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 0.0
$node_(4) set Y_ 15.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 0.0
$node_(5) set Y_ 25.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 0.0
$node_(6) set Y_ 35.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 10.0
$node_(7) set Y_ 35.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 10.0
$node_(8) set Y_ 45.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 20.0
$node_(9) set Y_ 35.0
$node_(9) set Z_ 0.0

$node_(10) set X_ 20.0
$node_(10) set Y_ 45.0
$node_(10) set Z_ 0.0
```