Masterarbeit

Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard

im Studiengang

Angewandte Informationstechnologien

Hochschule für Technik und Wirtschaft Dresden

Fakultät Informatik/Mathematik

Eingereicht von:	Thomas Bettermann
	Mat.-Nr.: 32897
Eingereicht am:	10.09.2014
Betreuer:	Prof. Dr.-Ing. Jörg Vogt

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit mit dem Titel

Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard

selbständig und ausschließlich unter Verwendung der im Quellenverzeichnis aufgeführten Literatur- und sonstigen Informationsquellen verfasst zu haben.
Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.
Ich bin damit einverstanden, dass ein Exemplar meiner Bachelorarbeit in der Bibliothek ausgeliehen werden kann.

Dresden, den 10. September 2014

Thomas Bettermann

Inhaltsverzeichnis

1

Einleitung

In der heutigen Zeit gewinnen drahtlose Sensornetze zunehmend an Bedeutung. Sie kommen in einem weiten Spektrum zum Einsatz, wie beispielsweise bei Umweltapplikationen zur Überwachung der Ausbreitung von Verschmutzungen in Luft und Wasser, beim Katastrophenschutz zur Erkennung und Überwachung von Waldbränden oder bei Heimapplikationen zur Steuerung und Überwachung des Hauses (vgl. ASSC01).
Für die Knoten eines drahtlosen Sensornetzes werden meist Mikrocontroller mit begrenzten Ressourcen genutzt. Diese Mikrocontroller werden mit unterschiedlichster Software betrieben. Dies reicht von einfachen Applikationen zur Erfassung und Übertragung von Sensorwerten, bis hinzu komplexeren internetfähigen Betriebssystemen. Um eine einheitliche Kommunikation zwischen den Knoten eines drahtlosen Sensornetzes auch mit unterschiedlicher Software zu gewährleisten, wurde der IEEE 802.15.4 Standard entwickelt (vgl. IEE11). Er beschreibt im speziellen die physikalische Schicht und die Media Access Control (MAC) Schicht bezüglich des ISO/OSI Referenzmodells.
Zur Realisierung solcher drahtlosen Sensornetze kommen unterschiedliche Betriebssysteme zum Einsatz. Ein Beispiel hierfür ist das Betriebssystem Contiki. Dabei handelt es sich um ein freies, internetfähiges Betriebssystem mit einem ereignisgesteuerten Betriebssystemkern (vgl. Conc). Es unterstützt eine Vielzahl unterschiedlicher Protokolle, beispielsweise das Internet Protocol version 6 (IPv6), das User Datagram Protocol (UDP) und das Routing Protocol for Low power and Lossy Networks (RPL). Weiterhin unterstützt Contiki unterschiedliche Hardware Plattformen (vgl. Conc). Nachteilig ist jedoch, dass Contiki keine vollständige Kompatibilität zum IEEE 802.15.4 Standard besitzt.
Ziel der Arbeit ist es, zu prüfen inwieweit Contiki mit dem IEEE 802.15.4 Standard, speziell in Bezug auf sie MAC Schicht, kompatibel ist. Des Weiteren ist zu klären, welche Schritte benötigt werden, um eine Kompatibilität mit dem Standard zu erreichen und inwiefern dies in Contiki umgesetzt beziehungsweise integriert werden

kann.

Um die Aufgabenstellung zu lösen, wird in Kapitel 2 der IEEE 802.15.4 Standard und die grundlegenden Aspekte zum Betriebssystem Contiki näher beschrieben. In Kapitel 3 werden derzeit bestehende Implementierungen bezüglich des IEEE 802.15.4 Standards vorgestellt. Um die Kompatibilität Contikis zum IEEE 802.15.4 Standard zu prüfen, wird in Kapitel 4 eine Quellcodeanalyse Contikis bezüglich des Standards vorgenommen. Unter Verwendung der Ergebnisse aus der Analyse, werden die konkrete Anforderungen an das Konzept aufgestellt. Aufbauend darauf wird im Kapitel 5 ein Konzept erarbeitet. Die Details zur Realisierung des Konzepts und dem Test der Implementierung sind Gegenstand von Kapitel 6. Den Abschluss der Arbeit bildet das Kapitel 7, indem die wichtigsten Aspekte der Arbeit zusammengefasst dargestellt werden und ein Ausblick für weiterführende Arbeiten gegeben wird.

2

Theoretische Grundlagen

Um das Thema „Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard“zu bearbeiten, werden nachfolgend die Grundlagen zum IEEE 802.15.4 Standard und zum Betriebssystem Contiki erläutert. Dabei beschränkt sich die Betrachtung des Standards im Wesentlichen auf die MAC Schicht. Beim Betriebssystem Contiki wird neben den allgemeinen Erklärungen speziell auf den Protokollstack eingegangen.

2.1 Der Funkstandard IEEE 802.15.4

Der IEEE 802.15.4 Standard beschreibt eine drahtlose Funktechnologie, die konzipiert wurde, um eine Kommunikation zwischen Maschinen beziehungsweise Maschinen und Menschen zu ermöglichen. Dies erlaubt die Entwicklung von drahtlosen Sensor und Aktor-Systemen, sogenannten Wireless Sensor Networks (WSNs). Die verschiedenen Anforderungen an WSNs unterscheiden sie von den konventionellen drahtlosen Kommunikationstechnologien. Die meisten Anwendungen für drahtlose Sensornetze werden teilweise bestimmt durch Energieverbrauch, technische Anforderungen (Datendurchsatz, Latenz, Sicherheit) und anderen Herausforderungen, wie zum Beispiel Hardwarekosten (vgl. GWCB10). Der IEEE 802.15.4 Standard für Low-Rate Wireless Personal Area Networks (LR-WPANs) definiert hierfür ein einfaches und flexibles Protokoll, welches eine drahtlose Kommunikation über kurze Reichweite für Applikationen mit begrenzter Energieversorgung und niedrigen Datenraten ermöglicht (vgl. IEE11). In Abbildung 2.1 wird ein Überblick über das Einsatzspektrum von LR-WPANs und anderen Kommunikationsstandards in Bezug auf Komplexität, Energieverbrauch, sowie der Datenrate gegeben.

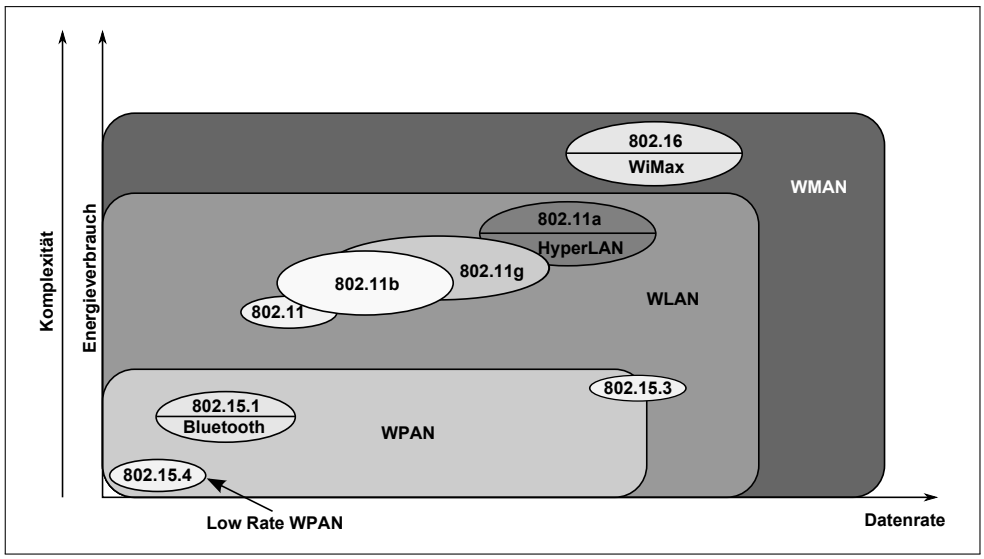

Abbildung 2.1: Einsatzspektrum von WPAN, WLAN und WPAN Standards (vgl. GWCB10)

2.1.1 ISO/OSI Referenzmodell

Die IEEE 802 Kommunikationsstandards definieren nur die zwei untersten Schichten des ISO/OSI Referenzmodells. Es handelt sich dabei um die Bitübertragungsschicht (engl. Physical Layer) und die Sicherungsschicht (engl. Data Link Layer). Die anderen Schichten werden im Standard nicht näher spezifiziert. Die Sicherungsschicht ist in zwei Unterschichten unterteilt, die Logical Link Control (LLC) Schicht und die MAC Schicht. Der IEEE 802.15.4 Standard definiert neben der Bitübertragungsschicht nur die MAC Schicht der Sicherungsschicht. Die Abbildung 2.2 zeigt den IEEE 802.15.4 Standard in Bezug auf das ISO/OSI Referenzmodell.

7-Schichten ISO/OSI Modell		IEEE 802 Modell	
7	Anwendung (Application)	Höhere Schichten	
6	Darstellung (Presentation)		
5	Sitzung (Session)		
4	Transport (Transport)		
3	Vermittlung (Network)		
2	Sicherung (Data Link)	Logical Link Control (LLC)	
		Media Access Control (MAC)	IEEE 802.15.4
1	Bitübertragung (Physical)	Physical signaling (PHY)	

Abbildung 2.2: IEEE 802.15.4 Standard im ISO/OSI Referenzmodell

Die IEEE 802.15.4 MAC Schicht Definition enthält eine erweiterte Funktionalität die normalerweise von der LLC Schicht bereitgestellt wird. Um eine einfache Implementation der drahtlosen Netzknoten zu gewährleisten, ist es daher zweckmäßig die MAC Schicht mit der Netzwerkschicht zu verbinden (vgl. GWCB10).

2.1.2 Komponenten

Der IEEE 802.15.4 Standard unterscheidet zwischen zwei Typen von Netzknoten, die Full Function Devices (FFDs) und Reduced Function Devices (RFDs).
Ein FFD besitzt den vollen Funktionsumfang und kann somit als Personal Area Network (PAN) Koordinator oder Koordinator für das Netz fungieren. Er kann sowohl mit RFDs sowie mit anderen FFDs kommunizieren. FFDs dienen sowohl zur Verwaltung des Netzes als auch zur Weiterleitung von Nachrichten innerhalb des PAN.
Ein RFD hingegen besitzt nur eine Teilmenge des Funktionsumfanges. Dieser kann nur mit einem FFD zur gleichen Zeit kommunizieren. RFDs sind für einfache Einsatzmöglichkeiten konzipiert. Typischerweise sind sie Aktoren oder Sensoren im Netzwerk, die meist keine großen Datenmengen senden oder empfangen und sich größtenteils in einem stromsparenden Zustand befinden. Mit RFDs können somit Applikationen implementiert werden, die minimale Ressourcen und Speicherplatz benötigen (vgl. IEE11).

2.1.3 Netzwerktopologien

Der IEEE 802.15.4 Standard beschreibt zwei Arten der Netzwerktopologie, die unterstützt werden (siehe Abbildung 2.3):

- Stern Topologie
- Peer-to-Peer Topologie

Die Verwaltung dieser Netzwerke obliegt der Netzwerkschicht und ist nicht Bestandteil des IEEE 802.15.4 Standards. Es werden jedoch die benötigten Funktionen von der MAC Schicht bereit gestellt (vgl. GWCB10).

Stern Topologie
Bei der Stern Topologie wird die Kommunikation durch einen einzelnen PAN-Koordinator kontrolliert. Dieser ist zuständig für die Verwaltung des Netzes. Dazu zählt die Initialisierung des PAN sowie das Hinzufügen und Entfernen von Netzteilnehmern. Bei dieser Topologie findet die Kommunikation der Netzteilnehmer ausschließlich mit dem PAN-Koordinator statt. Die Netzteilnehmer können somit nicht direkt untereinander, sondern nur über den PAN-Koordinator Nachrichten austauschen.
Bei der Initialisierung eines neuen PAN muss der PAN-Koordinator eine Netz Identifikationsnummer, die sogenannte PAN-ID wählen. Dabei ist zu Beachten, dass diese nicht in einem in Funkreichweite befindlichen Netz genutzt wird. Dies wird erreicht, indem in allen verfügbaren oder gewählten Übertragungskanälen nach anderen PANs gesucht und eine noch nicht genutzte PAN-ID gewählt wird (vgl. GWCB10).

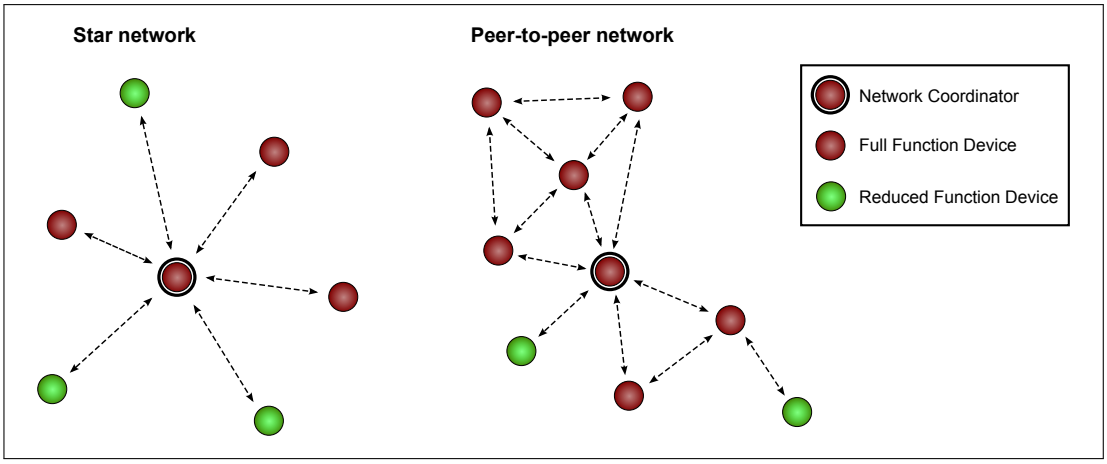

Abbildung 2.3: Netzwerktopologien

Peer-to-Peer Topologie
Die Peer-to-Peer Topologie erlaubt es jedem FFD mit jedem anderen FFD, welches in Funkreichweite ist, direkt zu kommunizieren. Mittels Multihop-Routing ist es zudem möglich Nachrichten zwischen FFDs auszutauschen, die keine unmittelbaren Nachbarn sind. Dabei werden die Nachrichten über dazwischenliegende FFDs weitergeleitet. Diese Topologie ermöglicht somit größere und komplexere Netzwerke, die sich selbst organisieren. RFDs können ausschließlich als Peripheriegeräte genutzt werden, da sie nicht die Funktionalität besitzen Nachrichten weiterzuleiten. Das Starten eines neuen Netzes verhält sich ähnlich wie beim Stern-Netz. Das heißt, der PAN-Koordinator wählt eine PAN-ID, welche sich von bestehenden in Funkreichweite befindlichen PANs und deren PAN-IDs unterscheidet (vgl. GWCB10).
Ein spezieller Typ des Peer-to-Peer Netzes ist das Cluster-Tree Netz, welches in Abbildung 2.4 dargestellt wird. Es besteht aus mehreren miteinander verbundenen PANs mit jeweils eigenem PAN-Koordinator und PAN-ID. Das Cluster-Tree Netz besitzt einen primären PAN-Koordinator, dem die gesamten Informationen über den Aufbau des Netzes vorliegen (vgl. IEE11).

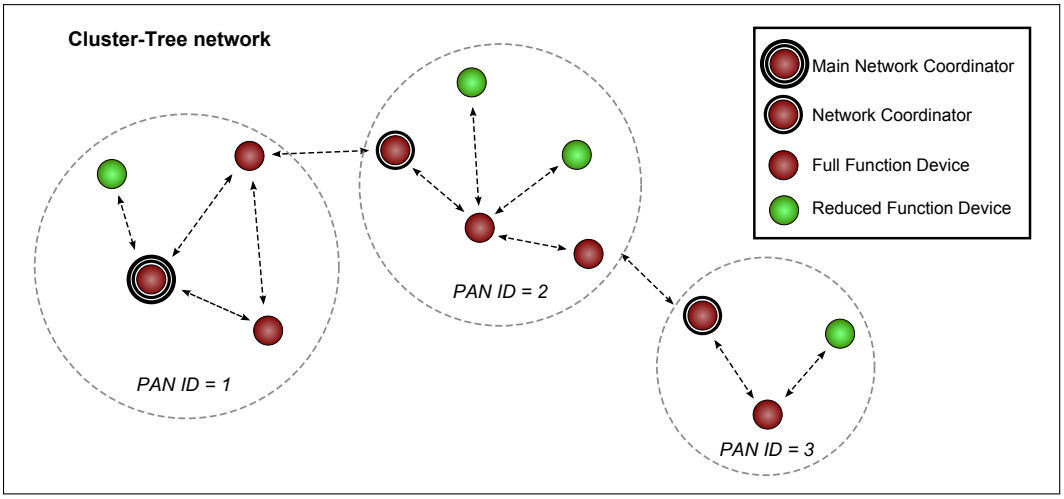

Abbildung 2.4: Cluster-Tree Netz Beispiel

2.1.4 Übertragungsverfahren

Neben den beiden Netzwerktopologien beschreibt der IEEE 802.15.4 Standard zwei grundsätzliche Verfahren zur sicheren Datenübertragung, welches vom PAN-Koordinator bestimmt werden kann (vgl. KAT05):

- *Beacon-enabled*: Es werden periodisch Beacon Frames vom Koordinator gesendet, um die Netzteilnehmer zu synchronisieren. Die Datenübertragung findet zwischen zwei Beacon Frames, in einem sogenannten Superframe, statt. Diese Superframe Struktur wird im nachfolgenden Kapitel näher beschrieben.
- *Non-Beacon-enabled*: Bei diesem Verfahren werden keine Beacon Frames verwendet. Ein Netzteilnehmer kann jederzeit seine Daten übertragen. Falls jedoch mehrere Netzknoten zur gleichen Zeit senden wollen, können dabei Kollisionen auftreten. Um dies zu vermeiden wird das Protokoll Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) genutzt.

Abbildung 2.5 zeigt eine Übersicht der im IEEE 802.15.4 Standard definierten Übertragungsverfahren.

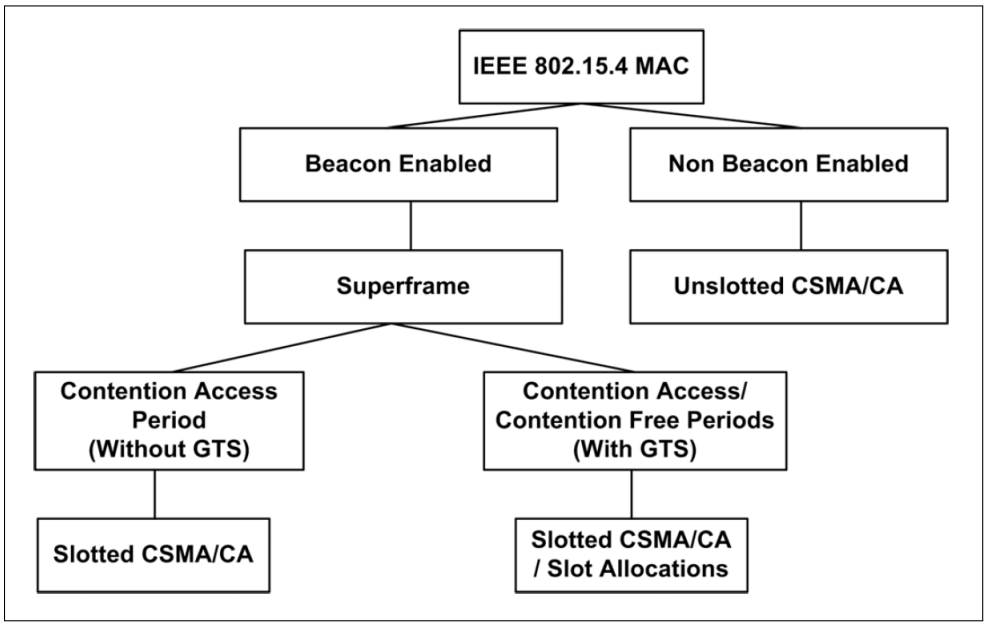

Abbildung 2.5: IEEE 802.15.4 Übertragungsverfahren(KAT05)

2.1.4.1 CSMA/CA Protokoll

Zur Datenübertragung über ein gemeinsames Medium definiert der IEEE 802.15.4 Standard zwei Methoden. Das slotted CSMA/CA Verfahren, welches in *beacon-enabled* PANs verwendet wird und das unslotted CSMA/CA Verfahren, das in *non-beacon-enabled* PANs zur Anwendung kommt. Bei beiden Verfahren wird das

Konzept des Interframe Spacing (IFS) genutzt. Aufgrund der Zeit die zur Verarbeitung der empfangen Daten benötigt wird, werden die Frames mit einem zeitlichem Abstand versendet. Die Zeitspanne richtet sich nach der Größe des gesendeten Frames. Es wird zwischen Long Interframe Spacing (LIFS) und Short Interframe Spacing (SIFS) unterschieden (vgl. IEE11).

Das unslotted CSMA/CA Verfahren nutzt laut dem IEEE 802.15.4 Standard zwei Variablen zum Zugriff auf den Übertragungskanal (vgl. LSC10):

NB: Diese Variable beschreibt wie oft der Algorithmus aufgrund eines belegten Kanals zurückgesetzt werden musste. Im Englischen spricht man von der Anzahl der *back-offs*.

BE: Diese Variable ist der *back-off* Exponent, welcher in Zusammenhang mit der Zeit steht, die abgewartet werden muss, bevor ein Clear Channel Assessment (CCA) ausgeführt werden kann. Beim CCA wird geprüft ob der Übertragungskanal frei ist.

Die Abbildung 2.6 zeigt ein Ablaufdiagramm des unslotted CSMA/CA Protokolls. Es kann zusammengefasst in fünf Schritten beschrieben werden (vgl. LSC10):

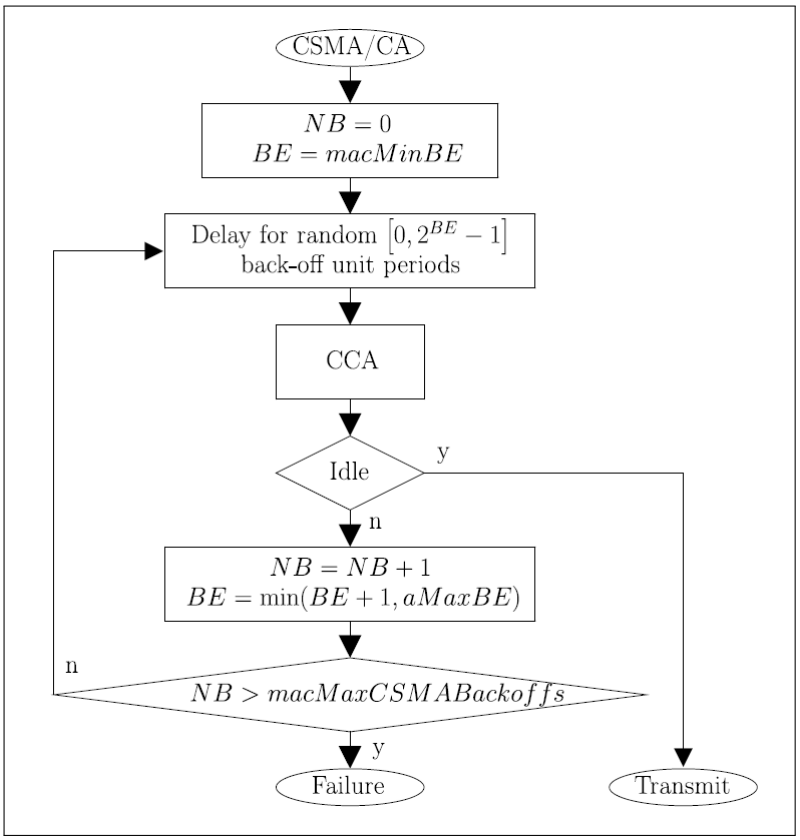

Abbildung 2.6: Unslotted CSMA Protokoll(LSC10)

1. Initialisierung der Variable *NB* mit dem Wert 0 und der Variable *BE* mit dem Wert der Konstante *macMinBE*, welche standardmäßig den Wert 3 besitzt.

2. Vermeiden von Kollisionen durch das Abwarten einer Zeitspanne, welche zufällig generiert wird und innerhalb von $[0, 2^{BE} - 1]$ *back-off* Zeiteinheiten liegt.
3. Ausführung des Clear Channel Assessment (CCA) direkt nach dem Abwarten der zufälligen Zeitspanne. Ist der Übertragungskanal belegt, fährt der Algorithmus mit Schritt 4 fort, andernfalls wird Schritt 5 ausgeführt.
4. Falls der Übertragungskanal belegt ist, werden die Variablen *NB* und *BE* um 1 erhöht. *BE* kann den Wert der Konstante *macMaxBE*(standardmäßig 5) nicht überschreiten. Falls *NB* kleiner als *macMaxCSMABackoffs* (standardmäßig 4) ist, dann kehrt der Algorithmus zu Schritt 2 zurück, andernfalls terminiert der Algorithmus mit dem Status Zugriff fehlgeschlagen.
5. Falls der Übertragungskanal nicht belegt war, übermittelt die MAC Schicht augenblicklich den zu versendenden Frame.

Beim slotted CSMA/CA Verfahren kommt zu den Variablen *NB* und *BE* eine weitere Variable *CW* hinzu. Der Wert dieser Variable beinhaltet die Anzahl der sogenannten Contention Window Slots für das CCA mit dem standardmäßigen Wert 2. Der Unterschied zum unslotted CSMA/CA Verfahren besteht darin, dass nach dem Schritt der Initialisierung der Variablen, die Grenzen der Slots ermittelt werden. Falls mittels CCA festgestellt wurde, dass der Übertragungskanal frei ist, wird im Gegensatz zum unslotted Verfahren nicht direkt mit der Übermittlung des Frames begonnen, sondern der Wert von *CW* um 1 verringert. Wenn der Wert gleich 0 ist, wird mit der Übertragung des Frames begonnen, andernfalls beginnt die Prozedur erneut. Das heißt, es wird eine zufällige Zeitspanne abgewartet und mittels CCA geprüft ob der Kanal belegt ist. Dies sorgt dafür, dass der Übertragungskanal für die Dauer von zwei Slots frei sein muss, bevor eine Übermittlung der Daten erfolgen kann (vgl. IEE11).

2.1.4.2 Superframe Struktur

Bei der Verwendung des *beacon-enabled* Modus sieht der IEEE 802.15.4 Standard die Verwendung von Superframes vor. Ein Superframe ist umschlossen von zwei Beacon Frames, die periodisch mit einem bestimmten Abstand, dem sogenannten Beacon Intervall, vom PAN-Koordinator gesendet werden. Die Struktur des Superframes kann an die verschiedensten Bedürfnisse angepasst werden. Dies reicht von kleinen Stern-Netzen mit niedriger Latenz bis zu großen Multihop-Netzen mit größerer Latenz.
Jedes Beacon Frame enthält wichtige Informationen, um den Netzteilnehmer die Synchronisation zu ermöglichen. Diese Informationen beinhalten die PAN-ID, das Beacon Intervall und die Struktur des Superframes. Ein Superframe ist in 16 aufeinanderfolgende Zeitschlitze aufgeteilt. Der erste Zeitschlitz beginnt mit dem Beacon Frame. Die weiteren 15 Zeitschlitze werden als sogenannte Contention Access Period (CAP) bezeichnet. Abbildung 2.7 zeigt den Aufbau eines solchen Superframes. Netzeilnehmer die Daten senden wollen, dürfen dies innerhalb der CAP unter Verwendung von slotted CSMA/CA.

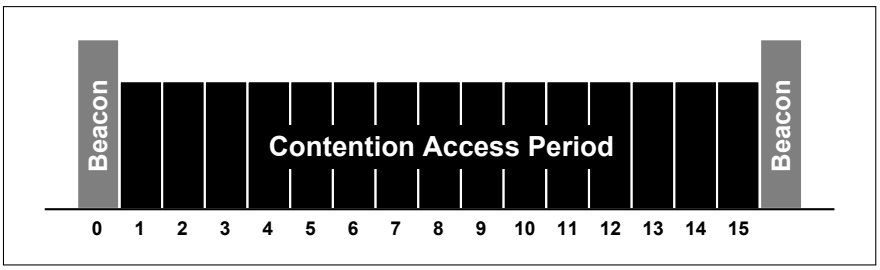

Abbildung 2.7: Aufbau eines Superframe (vgl. GWCB10)

Auf Anfrage eines Netzteilnehmers besteht die Möglichkeit, dass der PAN-Koordinator ihm bestimmte Zeitschlitze fest zur Verfügung stellt. Diese Zeitsegmente werden als sogenannte Guaranteed Time Slots (GTSs) bezeichnet. Alle GTSs werden am Ende des Superframes in der sogenannten Contention Free Period (CFP) zusammengefasst. In der Abbildung 2.8 wird der Aufbau eines Superframes mit CFP dargestellt. Da die GTSs immer jeweils einem bestimmten Netzteilnehmer zugeteilt sind, findet auch kein Wettbewerb innerhalb der CFP per CSMA/CA statt. Die Zeit, die durch die CFP genutzt wird, kann einen großen Teil des Superframes beanspruchen. Daher sieht der IEEE 802.15.4 Standard vor, dass ein Minimum (440 Symbole) für die CAP reserviert bleiben muss, damit andere Netzteilnehmer, welche nicht die CFP nutzen, kommunizieren können.
Die Möglichkeit der festen Zuteilung von Zeitschlitzen ist insbesondere nützlich für Applikationen, die eine bestimmte Bandbreite oder eine geringere Latenz benötigen. Dies bedeutet jedoch nicht, dass ein bestimmter Datendurchsatz garantiert wird oder Echtzeitbedingungen eingehalten werden, da beispielsweise ein fremder Netzwerkteilnehmer, der keine Informationen über den Superframe besitzt, innerhalb der CFP senden kann und somit die Kommunikation stört. Außerdem kann die zugeteilte Übertragungskapazität niedriger als die geforderte sein, wenn die Netzteilnehmer mehr nachfragen, als der PAN-Koordinator verteilen kann.

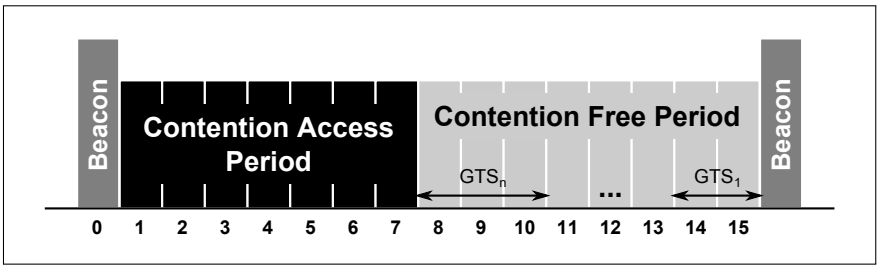

Abbildung 2.8: Aufbau eines Superframe mit GTS (vgl. GWCB10)

Eine weitere Anpassung des Superframes besteht darin, dass er in eine aktive und inaktive Phase aufgeteilt werden kann. In der aktiven Phase findet die Kommunikation wie zuvor beschrieben statt. Die inaktive Phase wird nicht zur Kommunikation zwischen den Netzteilnehmer genutzt, sodass der PAN-Koordinator in einen stromsparenden Zustand wechseln kann. Dies ermöglicht die Nutzung von einem batteriebetriebenen PAN-Koordinator, dessen reduzierter Arbeitszyklus zur Erhöhung der Lebenszeit der Batterie beiträgt. Je nach benötigtem Datendurchsatz oder

Latenz kann das Verhältnis zwischen aktiver und inaktiver Phase reguliert werden. Abbildung 2.9 zeigt ein Beispiel, in der die aktive und inaktive Phase gleich lang sind.

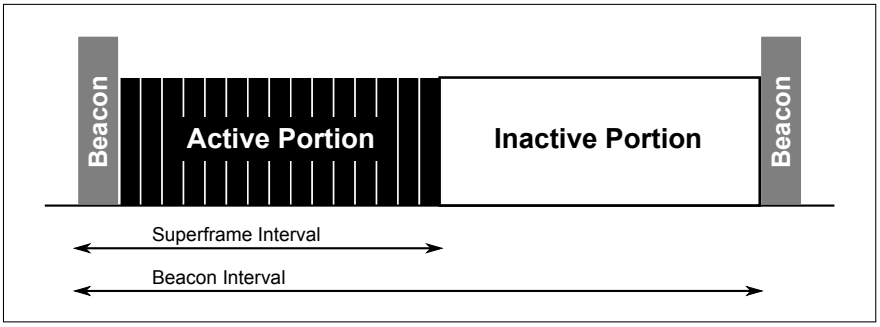

Abbildung 2.9: Aufbau eines Superframe mit mit aktiven/inaktiven Phase (vgl. GWCB10)

Als eine weitere Besonderheit definiert der IEEE 802.15.4 Standards eine MAC Funktion, die es einem Koordinator erlaubt seine Beacon Frames zeitlich auf die Beacon Frames seines übergeordneten Koordinators anzupassen, sodass es zu keinen Überschneidungen der aktiven Phasen der Superframes kommt. Dies erlaubt es Superframes nicht nur in Stern-Netzen mit einem Koordinator zu verwenden, sondern auch in größeren Multihop-Netzen mit mehreren Koordinatoren. Ein solcher Aufbau eines Superframes in Multihop-Netzen wird in Abbildung 2.10 dargestellt (vgl. GWCB10).

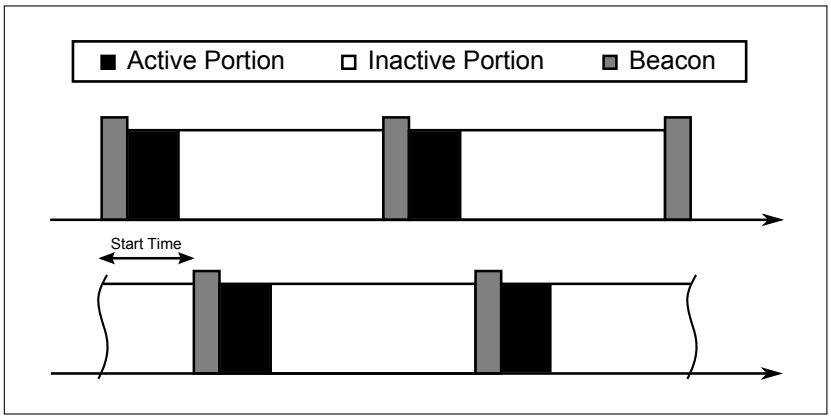

Abbildung 2.10: Aufbau eines Superframe in Multihop-Netzen (vgl. GWCB10)

2.1.4.3 Datenübertragungsmodell

Der IEEE 802.15.4 Standard definiert drei Arten der Datenübertragung, die abhängig von der Netzwerk Topologie Anwendung finden. Bei der Stern Topologie kommuniziert ausschließlich der PAN-Koordinator mit anderen Netzteilnehmern. Bei der Peer-to-Peer Topologie jedoch dürfen auch die Netzteilnehmer untereinander Nachrichten austauschen. Das heißt, in dieser Topologie müssen die folgenden drei Übertragungsarten unterstützt werden:

- Datenübertragung zum Koordinator
- Datenübertragung vom Koordinator
- Peer-to-Peer Datenübertragung

Im Folgenden werden die möglichen Übertragungsarten näher beschrieben. Dabei wird unterschieden zwischen *beacon-enabled* und *non-beacon-enabled* PANs.

Datenübertragung zum Koordinator
Möchte ein Netzteilnehmer in einem *beacon-enabled* PAN Daten zum Koordinator übertragen, muss er sich mit Hilfe der Beacons, die periodisch vom Koordinator gesendet werden, synchronisieren. Falls der Netzteilnehmer schon ein Teil des PAN ist und ihm ein GTS zugeteilt wurde, wartet er auf den bestimmten Zeitpunkt innerhalb des Superframes, um seine Daten ohne CSMA/CA zu übermitteln. Andernfalls überträgt der Netzteilnehmer seine Daten innerhalb der CFP des Superframes mittels slotted CSMA/CA. Nach dem der Koordinator die Daten empfangen hat, kann dieser, falls gefordert, eine Bestätigung (Acknowledgement) an den Netzteilnehmer senden. An dieser Stelle ist der Datentransfer abgeschlossen. Der Prozess hierzu, wird im linken Sequenzdiagramm der Abbildung 2.11 dargestellt. Der IEEE 802.15.4 Standard bezeichnet diese Prozedur als *direkten Datentransfer* (vgl. GWCB10).
In einem *non-beacon-enabled* PAN überprüft der Netzteilnehmer mittels CSMA/CA, ob der Übertragungskanal belegt ist. Ist dies nicht der Fall, überträgt er seine Daten an den Koordinator. Der Koordinator bestätigt daraufhin den Erhalt der Daten, falls gefordert. Im rechten Sequenzdiagramm der Abbildung 2.11 wird der Ablauf dargestellt (vgl. GWCB10).

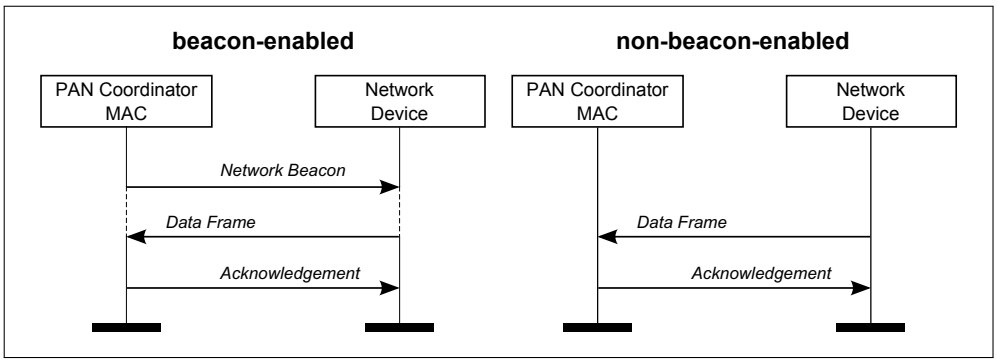

Abbildung 2.11: Datentransfer zum Koordinator (vgl. GWCB10)

Datenübertragung vom Koordinator
Wenn der Koordinator in einem *beacon-enabled* PAN Daten zu einem Netzteilnehmer übertragen möchte, dann sendet er nicht sofort, sondern speichert die Daten zwischen und übermittelt die Empfängeradresse in seinem Beacon Frame. Die Netzteilnehmer lauschen periodisch den Beacons und erkennen somit ob Daten für sie beim Koordinator vorliegen. Sobald die Daten vorliegen, werden sie mit Hilfe eines MAC Command Frame (Data Request) an den Koordinator angefordert. Die verschiedenen MAC Frames werden in Abschnitt 2.1.5.1 näher beschrieben. Nach dem

der Koordinator die Anforderung erhalten hat, schickt dieser eine Bestätigung zurück. Diese ist im Gegensatz zu den anderen Bestätigungsnachrichten nicht optional. Nach der Bestätigung beginnt der Koordinator mit der Übertragung der zwischengespeicherten Daten. Der Netzteilnehmer bestätigt die erfolgreiche Übermittlung der Daten, falls dies gefordert ist. Damit ist der, laut IEEE 802.15.4 Standard bezeichnete, *indirekte Datentransfer* abgeschlossen. Falls keine zu übertragenden Daten mehr für den Netzteilnehmer beim Koordinator vorliegen, wird die Empfangsadresse nicht mehr im Beacon übermittelt (vgl. IEE11). Im linken Sequenzdiagramm der Abbildung 2.12 wird der Prozess der Datenübertragung vom Koordinator zu einem Netzteilnehmer dargestellt.
Der indirekte Datentransfer findet auch bei *non-beacon-enabled* PANs Verwendung. Da es keinen Mechanismus gibt um dem Netzteilnehmer zu signalisieren, dass Daten für ihn beim Koordinator vorliegen, muss der Netzteilnehmer in regelmäßigen Abständen nachfragen. Dies geschieht mit Hilfe eines MAC Command Frames (Data Request). Der Koordinator bestätigt auch wieder den Erhalt der Anfrage und prüft ob Daten für den Netzteilnehmer vorliegen. Ist dies der Fall, übermittelt er seine Daten. Andernfalls wird ein Data Frame ohne Nutzdaten gesendet. Der Netzteilnehmer bestätigt den erfolgreichen Empfang des Data Frames, falls dies gefordert wurde (vgl. IEE11). Im rechten Sequenzdiagramm der Abbildung 2.12 wird dieser Prozess zusammengefasst dargestellt.

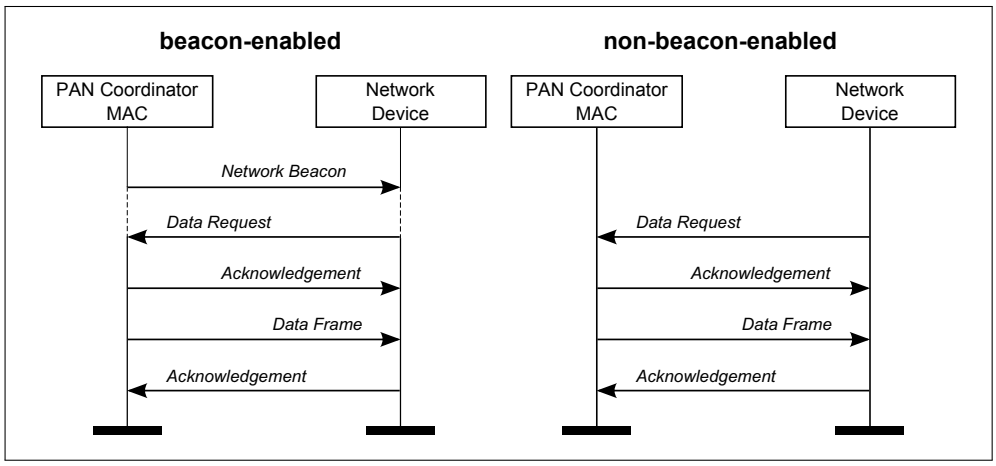

Abbildung 2.12: Datentransfer vom Koordinator (vgl. GWCB10)

Peer-to-Peer Datenübertragung
In einem Peer-to-Peer PAN kommunizieren alle Netzteilnehmer in Funkreichweite direkt miteinander. Um eine sichere Datenübertragung zu gewährleisten, müssen sie sich immer empfangsbereit halten oder sich synchronisieren. Im ersten Fall können die Daten einfach mittels CSMA/CA übertragen werden. Der zweite Fall wird im IEEE 802.15.4 Standard nicht weiter beschrieben (vgl. IEE11).

2.1.5 MAC Schicht

Die MAC Schicht kontrolliert den Zugriff auf einen geteilten Kanal und bietet die Möglichkeit einer sicheren Datenübertragung. Zudem stellt sie weitere Funktionen bereit (vgl. IEE11):

- Generierung und Synchronisierung von Beacons
- Unterstützung der An- und Abmeldung von einem PAN
- Unterstützung der Sicherheit im PAN
- Verwaltung des GTS Mechanismus

In den folgenden Unterkapiteln werden die MAC Frame Struktur und die Dienste der Mac Schicht näher beschrieben.

2.1.5.1 MAC Frame Struktur

Die IEEE 802.15.4 MAC Frame Struktur ist mit dem Anspruch entworfen wurden einem simplen und flexiblen Protokoll für LR-WPANs gerecht zu werden. Ein MAC Frame, welches auch als MAC Protocol Data Unit (MPDU) bezeichnet wird, besteht daher grundsätzlich aus 3 Teilen: dem *MAC Header (MHR)*, dem *MAC Payload* und dem *MAC Footer (MFR)*. In Abbildung 2.13 ist der generelle Aufbau eines MAC Frames dargestellt.

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

Abbildung 2.13: Generelles MAC Frame Format (IEE11)

Der MHR besteht im Wesentlichen aus Steuerdaten und Adressen. Im Folgenden werden die einzelnen Felder des MHR aufgelistet und näher beschrieben (vgl. IEE11).

- *Frame Control Field (FCF)*: Dieses Feld ist zwei Byte groß und enthält wichtige Steuerdaten.
- *Sequenz Number Field*: Dieses Feld beinhaltet die Sequenznummer des Frames, die sich mit jedem übermittelten Frame erhöht.
- *Addressing Fields*: Die Adressfelder beinhalten die PAN-IDs und die 64-bit beziehungsweise 16-bit langen Adressen des Senders und Empfängers.

- *Auxiliary Security Header Field*: Dieses Feld beinhaltet alle Informationen die für die Sicherheit benötigt werden. Je nach Sicherheitslevel kann die Länge des Feldes variieren.

Das FCF des MHR ist wiederum, wie in Abbildung 2.14 dargestellt, in folgende Felder unterteilt (vgl. IEE11):

- *Frame Type Field:* Es spezifiziert den Frame Typ (Beacon, Data, Acknowlegement, Command).
- *Security Enabled Field*: Das Frame kann durch die MAC Schicht gesichert werden, falls der Wert dieses Feldes auf 1 gesetzt wird. Das Auxiliary Security Header Field des MHR sollte nur dann im Frame vorhanden sein, falls dieses Feld auf den Wert 1 gesetzt ist.
- *Frame Pending Field*: Falls weitere Daten für den Empfänger vorliegen, erhält das Feld den Wert 1, andernfalls 0.
- *Acknowledgment Request (AR) Field*: Dieses Feld legt fest, ob ein Acknowlegement Frame bei erfolgreicher Übermittlung gefordert wird.
- *PAN ID Compression Field*: Falls die PAN-ID des Senders und Empfängers identisch sind, wird festgelegt, ob nur das Feld für die PAN-ID des Empfängers gesendet wird.
- *Destination Addressing Mode Field*: Dieses Feld legt fest, ob das Empfänger Adressfeld kurze Adressen (16-bit) oder lange Adressen (64-bit) beinhaltet.
- *Frame Version Field*: Dieses Feld spezifiziert die Versionsnummer des Frames.
- *Source Addressing Mode Field*: Dieses Feld legt fest, ob das Sender Adressfeld kurze Adressen (16-bit) oder lange Adressen (64-bit) beinhaltet.

Bits: 0–2	**3**	**4**	**5**	**6**	**7–9**	**10–11**	**12–13**	**14–15**
Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Abbildung 2.14: Frame Control Field (IEE11)

Der MAC Payload, der auch als MAC Service Data Unit (MSDU) bezeichnet wird, beinhaltet die eigentlichen Nutzdaten des Frames. Die Länge des Feldes ist variabel. Falls das *Security Enabled Field* des FCF auf den Wert 1 gesetzt ist, besteht die Möglichkeit die Daten durch kryptologische Verfahren zu sichern.
Der MFR besteht aus einer 16-bit langen Prüfsumme, der sogenannten Frame Check Sequence (FCS), die auf dem standardisierten ITU-T 16-bit Cyclic Redundancy

Check (CRC) Algorithmus basiert.

Im IEEE 802.15.4 Standard wird zwischen vier Frame Typen unterschieden: Beacon Frame, Data Frame, Acknowlegement Frame, MAC Command Frame. Im Folgenden werden die diese verschiedenen Typen näher beschrieben.

Beacon Frame Format
In einem *beacon-enabled* PAN (siehe Absch. 2.1.4) hat der Koordinator die Möglichkeit Beacon Frames zu versenden. In einem Beacon Frame beinhaltet das Adressfeld des MHR die PAN-ID und Adresse des Senders. Der MAC Payload eines Beacon Frames ist in folgende vier Felder unterteilt (vgl. GWCB10)

- *Superframe Specification Field*: Enthält die Parameter zur Spezifizierung der Superframe Struktur.
- *Pending Address Fields*: Beinhaltet die Anzahl und den Typ der Adressen, die im Address List Field aufgeführt sind.
- *Address List Field*: Beinhaltet eine Liste von Empfängeradressen für die Daten, die beim PAN-Koordinator vorliegen.
- *Beacon Payload*: Ist ein optionales Feld, welches von höheren Schichten genutzt werden kann.

Das Format eines Beacon Frame wird in Abbildung 2.15 dargestellt.

Octets: 2	**1**	**4/10**	**0/5/6/10/14**	**2**	**variable**	**variable**	**variable**	**2**
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Specification	GTS fields	Pending address fields	Beacon Payload	FCS
MHR				MAC Payload				MFR

Abbildung 2.15: Beacon Frame Format (IEE11)

Data Frame Format
Der Data Frame wird von der MAC Schicht genutzt, um Daten zu übertragen. Die Adressfelder enthalten, abhängig von den Einstellungen im FCF, die Adressfelder des Empfängers und/oder die Adressfelder des Senders. Das Format eines Data Frame wird in Abbildung 2.16 dargestellt.

Octets: 2	**1**	**variable**	**0/5/6/10/14**	**variable**	**2**
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Data Payload	FCS
MHR				MAC Payload	MFR

Abbildung 2.16: Data Frame Format (IEE11)

Acknowlegement Frame Format
Acknowlegement Frames werden zur Bestätigung einer erfolgreichen Übertragung eines Data oder Command Frame gesendet. Ein Acknowlegement Frame wird jedoch nur versendet, falls das Acknowledgment Request Field des FCF beim eingegangen Frame gesetzt war und somit eine Bestätigung gefordert wurde. Das Acknowlegement Frame enthält keine Adressfelder im MHR sowie keinen MAC Payload. Sobald ein Netzteilnehmer ein Acknowlegement Frame empfängt, überprüft er, ob eine Bestätigung von ihm angefordert wurde und vergleicht ob die erhaltene mit der von ihm erwarteten Sequenznummer übereinstimmt. Ist dies nicht der Fall, wird das Acknowlegement Frame verworfen. Das Format eines Acknowlegement Frame wird in Abbildung 2.17 dargestellt.

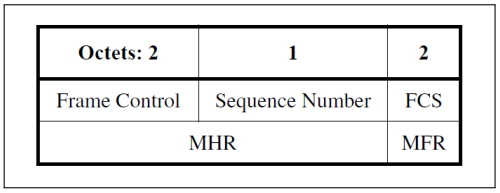

Abbildung 2.17: Acknowlegement Frame Format (IEE11)

MAC Command Frame Format
MAC Command Frames werden durch die MAC Schicht generiert und dienen der Verwaltung eines PAN. Es gibt verschiedene Command Typen, welche in Bezug auf ihre Identifikationsnummer in Tabelle 2.1.5.1 zusammengefasst dargestellt werden.

Command ID	Command Type
1	Association request
2	Association response
3	Disassociation notification
4	Data request
5	PAN ID conflict notification
6	Orphan notification
7	Beacon request
8	Coordinator realignment
9	GTS request
10-255	Reserved

Tabelle 2.1: MAC Command Frame Typen (vgl. IEE11)

Der MAC Payload setzt sich aus zwei Feldern zusammen:

- *Command Type Field*: Enthält den Typ des Befehls.
- *MAC Command Payload Field*: Enthält spezielle Informationen für den jeweils verwendeten Befehl.

Das Format eines Command Frames wird in Abbildung 2.18 dargestellt.

Octets: 2	1	variable	0/5/6/10/14	1	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Command Frame Identifier	Command Payload	FCS
MHR				MAC Payload		MFR

Abbildung 2.18: MAC Command Format (IEE11)

2.1.5.2 MAC Services

Der IEEE 802.15.4 Standard spezifiziert zwei Dienste der MAC Schicht auf welche die nächst höhere Schicht Zugriff hat. Dies ist der MAC Data Service und der MAC Management Service, auch als MAC Sublayer Management Entity (MLME) bezeichnet. Die Schnittstellen zu den Diensten sind der sogenannte MAC Common Part Sublayer Service Access Point (MCPS-SAP) und der MAC Sublayer Management Entity Service Access Point (MLME-SAP). Die Abbildung 2.19 zeigt die Schnittstellen in Bezug auf die IEEE 802.15.4 Protokoll Stack Architektur. Dabei bilden die Schnittstellen die Grundlage für komplexere Funktionalitäten auf den höheren Schichten. Hierfür werden im IEEE 802.15.4 Standard ein Set an Protokoll Primitiven für die Dienste definiert(vgl. IEE11).

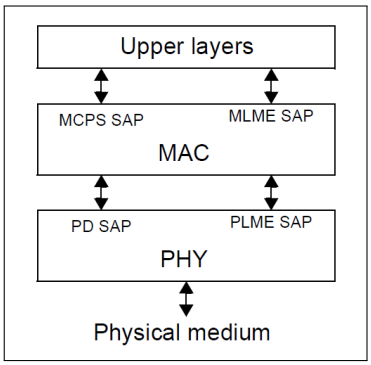

Abbildung 2.19: IEEE 802.15.4 Protokoll Stack Architektur (IEE11)

Konzept der Primitive
Die Dienste und somit die Funktionalität die eine Schicht der nächst höheren Schicht anbietet, werden laut IEEE 802.15.4 Standard durch die Beschreibung der Primitive und deren Parameter spezifiziert. Dabei werden vier generische Typen der Primitive unterschieden:

- Request: Beim Request Primitiv handelt es sich um eine Anfrage zum Start eines Dienstes.
- Indication: Das Indication Primitiv signalisiert dem Nutzer des Dienstes, dass ein internes Event ausgelöst wurde.
- Response: Das Response Primitiv beschreibt die Reaktion auf ein vorangegangenes Indication Primitiv.

- Confirm: Das Confirm Primitiv übermittelt die Ergebnisse eines vorher ausgelösten Request Primitives.

MAC Data Service
Der MAC Data Service stellt drei Primitive für den Datentransfer bereit: *MCPS-DATA.request*, *MCPS-DATA.confirm* und *MCPS-DATA.indication*. Der indirekte Datentransfer (siehe Absch. 2.1.4.3) zwischen zwei Netzknoten wird in Abbildung 2.20 in einem Sequenzdiagramm dargestellt. Der MAC Data Service benötigt keine Antwort von der MAC Schicht des Empfängers und somit existiert auch kein *response* Primitiv. Zudem unterstützt der MAC Data Service zwei weitere Primitive zum Löschen einer MSDU aus der MAC Warteschlange: *MCPS-PURGE.request* und *MCPS-PURGE.confirm*. Diese werden beim indirekten Datentransfer genutzt, falls beispielsweise ein Data Frame beim Koordinator auf eine Auslieferung an einen Netzteilnehmer wartet, der Netzteilnehmer jedoch kein *Data Request* in einer bestimmten Zeit schickt (vgl. GWCB10).

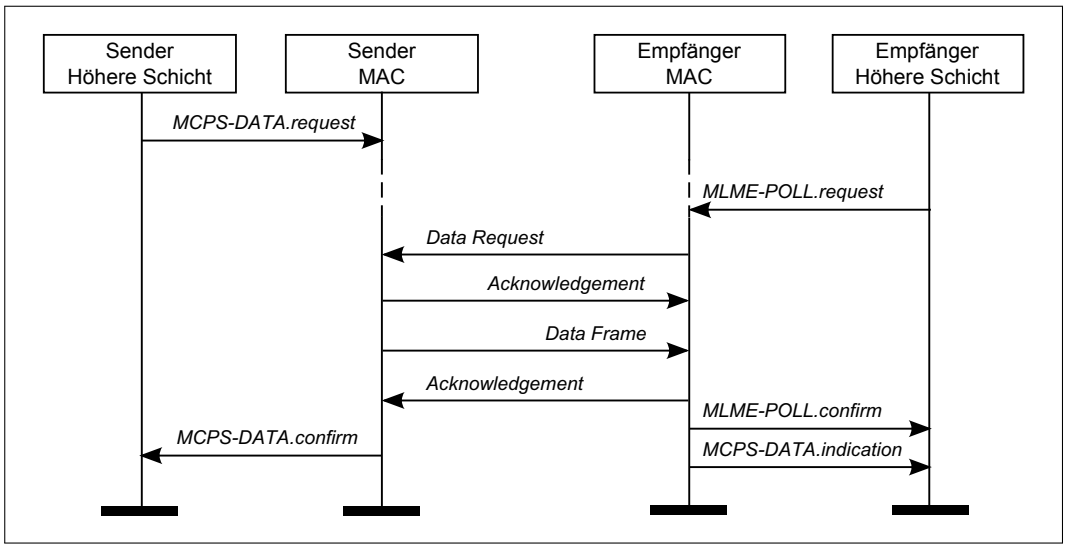

Abbildung 2.20: Sequenzdiagramm zum Datenaustausch mittels MCPS-DATA Primitiven (vgl. GWCB10)

MAC Management Service
Der MAC Management Service stellt eine Reihe von Primitiven zur Kontrolle der Kommunikationseinstellungen und der Funkeinheit sowie andere bestimmte Dienste für das Netzwerk zur Verfügung. Die Tabelle 2.2 zeigt eine Auflistung der verschiedenen Primitive.
Um eine geringere Komplexität bei Netzknoten zu erreichen, sind manche MAC Management Primitive optional. Diese Primitive sind *MLME-GTS*, *MLME-RX-ENABLE* und *MLME-SYNC*. Zudem gibt es Primitive die optional für RFDs sind: *MLME-ASSOCIATE.indication*, *MLME-ASSOCIATE.response*, *MLME-ORPHAN* und *MLME-START* (vgl. GWCB10). Im Folgenden werden die unterschiedlichen Primitive und deren Verwendung näher beschrieben.

Dienst	Request	Confirm	Response	Indication
Communication Settings:				
GET	X	X		
SET	X	X		
RESET	X	X		
Radio Control:				
RX-ENABLE	X	X		
SCAN	X	X		
Networking:				
ASSOCIATE	X	X	X	X
DISASSOCIATE	X	X		X
GTS	X	X		X
ORPHAN			X	X
SYNC	X			
SYNC-LOSS				X
START	X	X		
BEACON-NOTIFY				X
POLL	X	X		
COMM-STATUS				X

Tabelle 2.2: Überblick zu den MAC Management Primitiven (vgl. IEE11)

Mit Hilfe der Primitive *MLME-GET*, *MLME-SET* und *MLME-RESET* können die Attribute der MAC Schicht gelesen, geschrieben oder auf ihren Standardwert zurückgesetzt werden. Diese Attribute dienen zur Verwaltung der MAC Schicht und werden in der sogenannten MAC PAN Information Base (PIB) zusammengefasst.

Die Funkeinheit kann mittels des *MLME-RX-ENABLE* Primitiv aktiviert beziehungsweise auch deaktiviert werden. Diese Aktion kann sofort ausgeführt oder für einen späteren Zeitpunkt geplant werden. Dies erlaubt den höheren Schichten den Energieverbrauch zu senken.

Das *MLME-SCAN* Primitiv erlaubt einem Netzknoten das Scannen der verfügbaren Übertragungskanäle. Es gibt vier verschiedene Arten von Scans, die in unterschiedlichen Szenarien zur Anwendung kommen (vgl. GWCB10):

- *Energy Detection Scan*: Dieser erlaubt das Messen der Energie der Radiofrequenz jedes einzelnen Übertragungskanals. Genutzt wird dieses Verfahren vom PAN-Koordinator, um einen freien Übertragungskanal zu finden und um somit ein neues PAN zu starten.
- *Active Channel Scan*: Dieser Scan sucht PAN-Koordinatoren oder Koordinato-

ren in Funkreichweite, welche schon Teil eines PAN sind. Der Netzteilnehmer, der diesen Scan durchführt, um PANs in der Nähe ausfindig zu machen, schickt hierfür auf jedem Übertragungskanal ein *Beacon Request Command Frame* (siehe Absch. 2.1.5.1), welches jeden PAN Koordinator oder Koordinator dazu auffordert ein Beacon Frame zu senden.

- *Passive Channel Scan*: Im Unterschied zu einem aktiven Scan, wird beim passiven Scan kein *Beacon Request Command Frame* gesendet, sondern nur auf eingehende Beacon Frames gelauscht. Dies kommt dabei nur in *beacon- enabled* PANs zur Anwendung, da dort periodisch Beacons versendet werden.

- *Orphan Channel Scan*: Dieser erlaubt einem Netzteilnehmer, der die Verbindung zu seinem Koordinator verloren hat, eine lokale Suche durchzuführen, um einen Koordinator des PAN zu finden. Diese Suche beschränkt sich auf eine spezielle Liste von Übertragungskanälen.

Ein Netzknoten, der mit keinem PAN verbunden ist, aber einem Beitreten möchte, startet zuallererst einen aktiven beziehungsweise passiven Scan um potentielle PANs zu finden. Nach einem erfolgreichen Scan werden alle Informationen aus den erhaltenen Beacon Frames an die höhere Schicht übergeben. Auf der höheren Schicht wird anhand der Informationen entschieden welchem PAN der Netzknoten beitreten möchte. In einem *beacon-enabled* PAN wird zunächst das *MLME-SYNC.request* Primitiv verwendet, um sich mit dem PAN-Koordinator zu synchronisieren. Nach der Synchronisierung erfolgt ein *MLME-ASSOCIATE.request* zur MAC Schicht. Bei einem *non-beacon-enabled* PAN wird keine Synchronisation benötigt und der Aufruf des *MLME-ASSOCIATE.request* Primitiv kann sofort erfolgen. Nachdem die MAC Schicht ein *MLME-ASSOCIATE.request* Primitiv erhalten hat, wird ein *Association Request Command Frame* an den gewählten Koordinator gesendet. Das Verfahren zur Wahl eines Koordinators wird im IEEE 802.15.4 Standard nicht beschrieben, da es die Aufgabe der höheren Schichten ist eine Strategie zur Assoziierung zu implementieren. Bei einer erfolgreichen Übermittlung des *Command Frames*, wird eine Bestätigung (*Acknowledgement Frame*) vom Koordinator zum Netzknoten geschickt. Diese Bestätigung sagt nicht aus, dass der Beitritt zum PAN erfolgreich verlaufen ist, sondern bestätigt lediglich den Erhalt der Assoziierungsanfrage. Nach dem der PAN-Koordinator das *Association Request Command Frame* erhalten hat, entscheidet dieser, ob seine Ressourcen ausreichen um einen neuen Netzknoten zum PAN hinzuzufügen. Beim PAN-Koordinator wird daraufhin das *MLME-ASSOCIATE.response* Primitiv verwendet, um ein *Association Response Command Frame*, mit der Information ob der Beitritt zum PAN akzeptiert oder abgelehnt wird, zum Netzknoten zu übertragen. Eine Besonderheit beim Assozierungsprozess ist die Möglichkeit beim PAN-Koordinator eine 16-bit Adresse anzufordern. Dies ermöglicht eine bessere Bandbreite, da der MHR kürzer ist und mehr Nutzdaten in einem Frame übertragen werden können (vgl. GWCB10 und vgl. KAT05). Die Abbildung 2.21 stellt den Anmeldeprozess in einem Sequenzdiagramm dar.

Der Abmeldeprozess von einem PAN kann sowohl von einem Netzknoten als auch vom PAN-Koordinator initiiert werden. Hierzu dienen die *MLME-DISASSOCIATE*

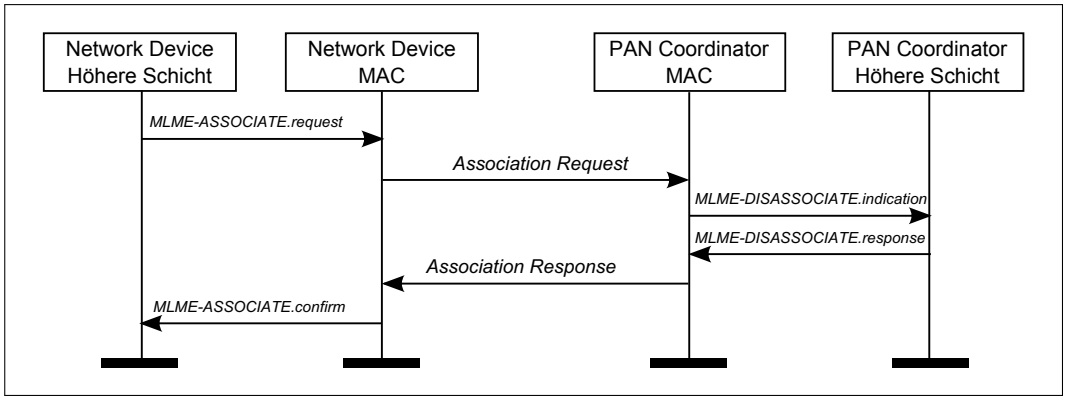

Abbildung 2.21: Sequenzdiagramm für den Anmeldeprozess (vgl. GWCB10)

Primitive. Bei der Verwendung des *MLME-DISASSOCIATE.request* wird ein *Disassociation Request Command Frame*, abhängig davon wer den Abmeldeprozess gestartet hat, an den Koordinator beziehungsweise den Netzknoten geschickt. Der Empfänger des *Disassociation Request Command Frame* bestätigt daraufhin dessen Erhalt mit einem *Acknowledgement Frame*. Nachdem die Abmeldung erfolgreich abgeschlossen ist, werden alle Informationen des Netzknoten beim PAN-Koordinator sowie alle Informationen über das PAN beim Netzknoten gelöscht (vgl. GWCB10 und vgl. KAT05). Die Abbildung 2.22 stellt den Abmeldeprozess in einem Sequenzdiagramm dar.

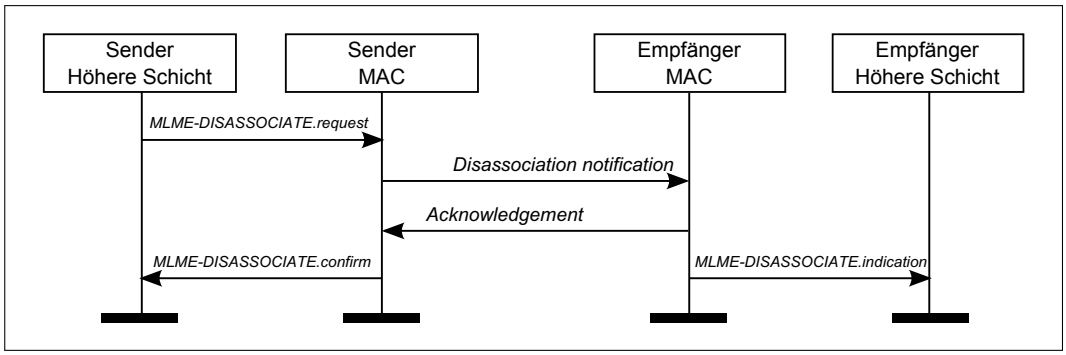

Abbildung 2.22: Sequenzdiagramm für den Abmeldeprozess (vgl. GWCB10)

Wie bereits im Abschnitt 2.1.4.2 beschrieben, erlaubt der IEEE 802.15.4 Standard die optionale Verwendung von Superframes, welche GTS (garantierte Zeitschlitze) nutzen. Die *MLME-GTS* Primitive werden genutzt, um einen neuen GTS zu erzeugen, einen bestehenden GTS freizugeben oder einen GTS neu zuzuteilen (vgl. GWCB10).

Die *MLME-ORPHAN* Primitive werden bei einem *Orphan Channel Scan* genutzt. Falls ein Netzknoten einen solchen Scan initiiert hat, erhält der PAN-Koordinator oder der Koordinator ein *Orphan Notification Command Frame*. Die MAC Schicht des Koordinators nutzt das *MLME-ORPHAN.indication* Primitiv, um die erhaltenen Informationen an die höheren Schichten weiterzuleiten. Auf den höheren Schichten wird dann überprüft, ob der Netzknoten zum PAN gehörte. Ist dies der Fall, wird

das *MLME-ORPHAN.response* Primitiv genutzt, um ein *Coordinator Realignment Command Frame* an den Netzknoten zu senden (vgl. GWCB10).

Zur Synchronisation in einem *beacon-enabled* PAN werden die *MLME-SYNC* und *MLME-SYNC-LOSS* Primitive genutzt. Das *MLME-SYNC* Primitiv erlaubt einem Netzknoten das Suchen und Erkennen von Beacon Frames. Verliert ein Netzknoten die Synchronisation mit dem PAN-Koordinator, so generiert die MAC Schicht ein *MLME-SYNC-LOSS.indication* Primitiv (vgl. GWCB10).

Das Generieren von Beacons in einem *beacon-enabled* PAN wird durch das *MLME-START* Primitiv initiiert. Die Parameter des Primitivs erlauben es dem Netzknoten, der dieses Primitiv aufruft, sich als PAN-Koordinator zu konfigurieren und damit bestimmte Aufgaben zu übernehmen. Dazu gehören die Auswahl eines Übertragungskanals, das Senden der periodischen Beacons sowie die Konfiguration des Superframes. Die MAC Schicht reagiert auf das *MLME-START.request* Primitiv mit einem *MLME-START.confirm*. Empfängt ein Netzknoten ein Beacon Frame, welches einen *Payload* von einem oder mehreren Bytes besitzt, so startet die MAC Schicht ein *MLME-BEACON-NOTIFY.indication* Primitiv (vgl. GWCB10).

Das *MLME-POLL* wird beim indirekten Datentransfer (siehe Absch. 2.1.4.3) genutzt. Da ein Netzknoten in einem *non-beacon-enabled* PAN keinerlei Anhaltspunkte besitzt, ob Daten beim Koordinator für ihn vorliegen, ist es den höheren Schichten zu überlassen periodisch nachzufragen. Hierfür existiert das *MLME-POLL.request*, um der MAC Schicht zu signalisieren ein *Data Request Command Frame* an den Koordinator zu senden. Der Koordinator erwidert dies mit einem *Acknowledgement Frame*, in welchem das *frame pending field* entweder auf den Wert 0 (keine Daten vorhanden) oder den Wert 1 (Daten vorhanden) gesetzt ist. Das Ergebnis des *MLME-POLL.request* wird mit Hilfe des *MLME-POLL.confirm* von der MAC Schicht an die höheren Schichten übermittelt (vgl. GWCB10).

Mit dem *MLME-COMM-STATUS.indication* Primitiv werden verschiedene Kommunikationsstatus Meldungen von der MAC Schicht für die höheren Schichten generiert.

2.2 Betriebssystem Contiki

Contiki ist ein leichtgewichtiges Betriebssystem und sehr gut für Netzknoten eines WSN geeignet, dessen Ressourcen bezüglich Speicher und Energie begrenzt sind. Es unterstützt das dynamische Laden und Ersetzen von individuellen Programmen und Diensten. Contiki besteht aus einem ereignisgesteuerten Kernel, bietet aber auch die Möglichkeit des präemptiven Multithreading für individuelle Prozesse. Prozesse sind durch sogenannte Protothreads realisiert und werden in Abschnitt 2.2.2 näher erläutert(vgl. DGV04).
Contiki bietet die Möglichkeit der IP Kommunikation mit Hilfe des uIP TCP/IP Stacks. Im Jahr 2008 wurde uIPv6, der weltweit kleinste IPv6 Stack mit ungefähr 11kB Speicherplatzbedarf, für Contiki veröffentlicht. Dadurch ist es möglich, dass

Contiki Applikationen direkt mit anderen IP-basierenden Applikationen und Internetdiensten kommunizieren können (vgl. VD10). Zudem werden die Protokolle TCP, UDP und ICMP unterstützt (vgl. FK11).
Contiki ist in der Programmiersprache C implementiert und ist frei verfügbar (Open Source) und steht unter der BSD-Lizenz.

2.2.1 Architektur

Das Contiki Betriebssystem beinhaltet im Kern die IP-Kommunikation via uIP, dem Laden und Entfernen von Programmen zur Laufzeit(Loader) und die Unterstützung von Multithreading mittels Protothreads, die auf dem ereignisgesteuerten Kernel aufbauen. Des Weiteren definiert Contiki die Schnittstellen zu den verschiedenen Komponenten der Plattformen. Dazu zählen unter anderem die CPU, Funkeinheit, Sensoren und Oszillatoren. Jede von Contiki unterstützte Plattform besitzt hierbei eine hardwarespezifische Implementation ihrer einzelnen Komponenten. Dabei sind die Applikationen für Contiki in der Regel plattformunabhängig, da sie nur auf die Schnittstellen der Komponenten zugreifen.
Bisher wurde Contiki auf über 12 verschiedene Mikroprozessoren und Mikrocontrollern portiert (vgl. VD10). Die Abbildung 2.23 zeigt eine vereinfachte Darstellung der Architektur des Betriebssystems Contiki.

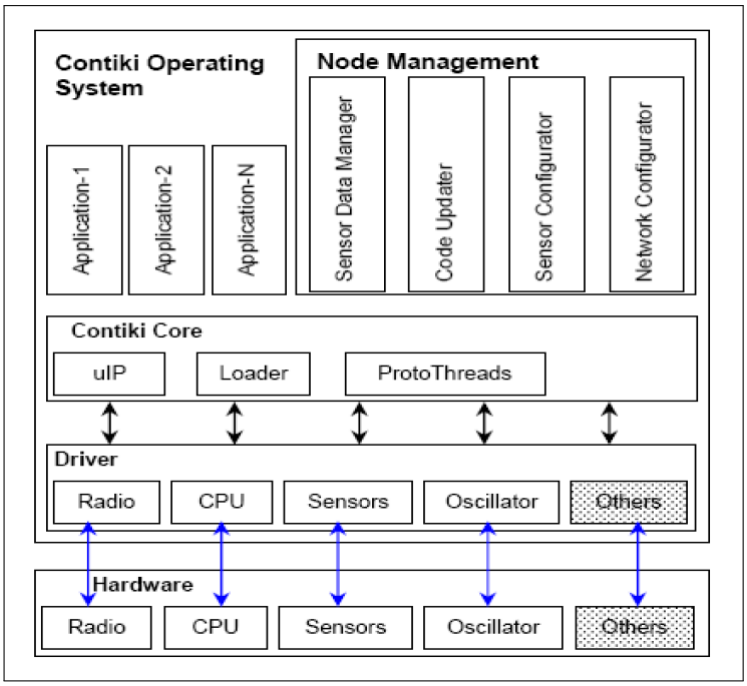

Abbildung 2.23: Vereinfachte Architektur des Betriebssystems Contiki (DTV09)

Die grundsätzliche Architektur spiegelt sich auch in der Organisation des Quellcodes und seiner Verzeichnisstruktur wieder. Der plattformabhängige Code befindet sich in den Ordnern */contiki-2.x/cpu* sowie in */contiki-2.x/platform*. Alle anderen Verzeichnisse, insbesondere der Ordner für den Kern Contiki's */contiki-2.x/core* beinhalten in der Regel plattformunabhängigen Code. Diese Struktur ermöglicht eine sehr ein-

fache Portierung Contiki's auf andere Mikroprozessoren beziehungsweise Mikrocontrollern.

2.2.2 Multithreading und Protothreads

Contiki basiert auf einem ereignisgesteuerten Kernel, bei dem die Prozesse im Gegensatz zu threadbasierten Systemen alle einen gemeinsamen Stack nutzen. Prozesse werden als sogenannte *Event Handler* implementiert und vom Kernel innerhalb einer Warteschlange verwaltet. Einmal gestartet, werden die Events bis zu ihrer vollständigen Abarbeitung nicht unterbrochen. Der Kernel unterstützt zwei Arten von Events: asynchrone und synchrone. Synchrone Events werden sofort ausgelöst und somit auch abgearbeitet. Asynchrone Events hingegen, werden in die Warteschlange eingereiht und später abgearbeitet. Zusätzlich zu den Events bietet der Kernel einen *polling* Mechanismus. *Polling Events* können als hoch priorisierte Events angesehen werden, die zwischen den asynchronen Events eingeplant werden. Dies wird meist von hardwarenahen Prozessen genutzt, die regelmäßig den Status einer Hardwarekomponente, beispielsweise des Funkmoduls, kontrollieren.
Des Weiteren unterstützt Contiki das präemptive Multithreading, welches als Systembibliothek implementiert ist und sich optional mit Applikationen verlinken lässt. Die Contiki Multithreading Bibliothek ist in einen plattformunabhägigen Teil mit der Schnittstelle zum Event Kernel und einen plattformabhägigen Teil unterteilt, welcher die Funktionalität zum Wechseln des Stacks sowie die Funktionen zum Unterbrechen beinhaltet (vgl. DGV04 und vgl. FK11).
Das ereignisgesteuerte Modell unterstützt nicht das Konzept des blockierendes Wartens, da ein Event während der Abarbeitung nicht unterbrochen werden kann. Komplexe Programme, welche nicht in einem einzigen Event implementiert werden können, müssen mit Hilfe von Zustandsmaschinen implementiert werden. Dies ist zum einen unübersichtlich und somit fehleranfälliger und zum anderen vergrößert sich der auszuführende Code. Aufgrund dessen wurde in Contiki das Konzept der Protothreads entwickelt. Protothreads können als eine Kombination von Events und Threads gesehen werden. Von den Threads wurde die Möglichkeit des blockierenden Wartens und von den Events der minimale Speicherplatzbedarf, aufgrund der Verwendung eines gemeinsamen Stacks, übernommen. Zum blockierenden Warten wird das Statement *PT_WAIT_UNTIL(boolscher Ausdruck)* zur Verfügung gestellt. Bei jedem Aufruf des Protothread wird die Bedingung erneut ausgewertet und blockiert diesen solange bis die Bedingung erfüllt ist. Des Weiteren gibt es ein Statement zum blockierenden Warten ohne Bedingung: *PT_YIELD()*. Hierbei wird der Protothread einmalig blockiert bis er das nächste Mal aufgerufen wird. Der Anfang und das Ende eines Protothreads wird mit Hilfe der *PT_BEGIN* und *PT_END* Statements markiert. Es gibt zudem die Möglichkeit der hierarchischen Protothreads. Innerhalb eines Protothreads kann mit Hilfe des *PT_SPAWN()* Statements ein Kind Protothread erzeugt werden, der den derzeitigen Protothread blockiert bis der Kind Protothread mittels *PT_EXIT* oder *PT_END* beendet wird.
Zusammenfassend lässt sich feststellen, dass Protothreads mehrere kooperative Threads mit nur geringem Speicherplatzbedarf ermöglichen, aufgrund der Verwendung nur eines Stacks. Die Verwendung der Protothread Statements ist relativ einfach,

jedoch müssen einige Punkte bei der Programmierung beachtet werden. Kontextwechsel müssen explizit angegeben werden durch *PT_WAIT_UNTIL()* oder *PT_YIELD()*, da kein präemptiver Entzug der Ressourcen stattfindet. Prothothreads mit langer Laufzeit sollten daher regelmäßig diese Funktionen aufrufen. Die Variablen eines Protothreads bleiben bei einem Kontextwechsel nur erhalten, falls diese als *static* oder *global* deklariert werden. Außerdem ist der Kontextwechsel nur innerhalb des Protothreads zulässig und nicht etwa in einer ausgelagerten Funktion. Wird die Auslagerung jedoch benötigt, so bietet sich die Möglichkeit der Verwendung des Konzepts der hierarchischen Protothreads an (vgl. DSV05 und vgl. DSVA06).

2.2.3 Protokollstack

Das Betriebssystem Contiki unterstützt eine ganze Reihe unterschiedlicher Protokolle zur Kommunikation, beispielsweise stellt es eine Implementation von uIP bereit, einem leichtgewichtigen TCP/IP Stack für 8bit Mikrocontroller. Dieser Stack setzt direkt auf den Netzwerk Stack, dem sogenannten *NETSTACK*, von Contiki auf. Im Weiteren wird dieser näher beschrieben.

2.2.3.1 Netzwerk Stack Architektur

Der Contiki *NETSTACK* ist modular aufgebaut, sodass er sehr flexibel an die verschiedensten Bedürfnisse angepasst werden kann. Hierfür werden fünf verschiedene Treiber definiert: *Radio Driver*, *Radio Duty Cycle (RDC) Driver* (radio duty cycle), *MAC Driver*, *Network Driver* und der *Framer*. Diese Treiber werden repräsentiert durch Strukturen, die in der Datei */contiki-2.x/core/net/netstack.h* deklariert sind (siehe Listing 2.1).

```
extern const struct network_driver NETSTACK_NETWORK;
extern const struct rdc_driver     NETSTACK_RDC;
extern const struct mac_driver     NETSTACK_MAC;
extern const struct radio_driver   NETSTACK_RADIO;
extern const struct framer         NETSTACK_FRAMER;
```

Listing 2.1: NETSTACK Strukturdeklaration (/contiki-2.x/core/net/netstack.h)

Die Strukturen der einzelnen Treiber sind jeweils in ihren Header-Dateien definiert. Dieser umfasst einen Zeiger auf den Namen des Treibers sowie die Deklaration der Funktionszeiger. Im Listing 2.2 wird beispielsweise die Struktur des MAC Treibers dargestellt.

```
/**
 * The structure of a MAC protocol driver in Contiki.
 */
struct mac_driver {
  char *name;

  /** Initialize the MAC driver */
  void (* init)(void);

  /** Send a packet from the Rime buffer  */
  void (* send)(mac_callback_t sent_callback, void *ptr);
```

```
  /** Callback for getting notified of incoming packet. */
  void (* input)(void);

  /** Turn the MAC layer on. */
  int (* on)(void);

  /** Turn the MAC layer off. */
  int (* off)(int keep_radio_on);

  /** Returns the channel check interval, expressed in clock_time_t
      ticks. */
  unsigned short (* channel_check_interval)(void);
};
```

Listing 2.2: Struktur des MAC Treibers (/contiki-2.x/core/net/mac.h)

Die Struktur dient gewissermaßen als Schablone für eine konkrete Implementierung eines *NETSTACK* Treibers. Das heißt, es wird eine Variable der Struktur angelegt und die Funktionszeiger mit den konkret implementierten Funktionen initialisiert. Ein Beispiel für eine Initialisierung des MAC Treibers mit einer konkrete Implementierung (CSMA) ist in Listing 2.3 dargestellt.

```
const struct mac_driver csma_driver = {
  "CSMA",
  init,
  send_packet,
  input_packet,
  on,
  off,
  channel_check_interval,
};
```

Listing 2.3: CSMA Treiber Initialisierung (/contiki-2.x/core/net/csma.c)

Die konkreten *NETSTACK* Treiber werden für ein Projekt und eine bestimmte Plattform in einer Konfigurations Header-Datei mittels #define-Anweisungen festgelegt (/contiki-2.x/platform/.../contiki-conf.h). Dies wird im Listing 2.4 für die Plattform *avr-atmega128rfa1* beispielhaft dargestellt.

```
#define NETSTACK_CONF_MAC         nullmac_driver
#define NETSTACK_CONF_MAC         csma_driver
#define NETSTACK_CONF_RDC         cxmac_driver
#define NETSTACK_CONF_FRAMER      framer_802154
#define NETSTACK_CONF_RADIO       rf230_driver
```

Listing 2.4: Festlegung der NETSTACK Treiber (/contiki-2.x/platform/ avr-atmega128rfa1/contiki-conf.h)

Durch das modulare Design des *NETSTACK* besteht die Möglichkeit eigene Implementierungen der Treiber in Contiki hinzuzufügen und einzubinden.

2.2.3.2 Netzwerk Stack im ISO/OSI Referenzmodell

Der Contiki *NETSTACK* setzt sich, wie bereits in Abschnitt 2.2.3.1 beschrieben, aus fünf Treibern zusammen. Jeder Treiber operiert auf einer der, in Bezug auf das ISO/OSI Modell, unteren drei Schichten und stellt einen Teil der Funktionalität für die Kommunikation bereit (vgl. Conb). Contiki bietet für jeden Treiber ein oder mehrere Implementationen an, die je nach Bedarf gewählt werden können. In Abbildung 2.24 ist eine mögliche Einordnung des Contiki Protokoll Stacks in Bezug auf das ISO/OSI Referenzmodell dargestellt.

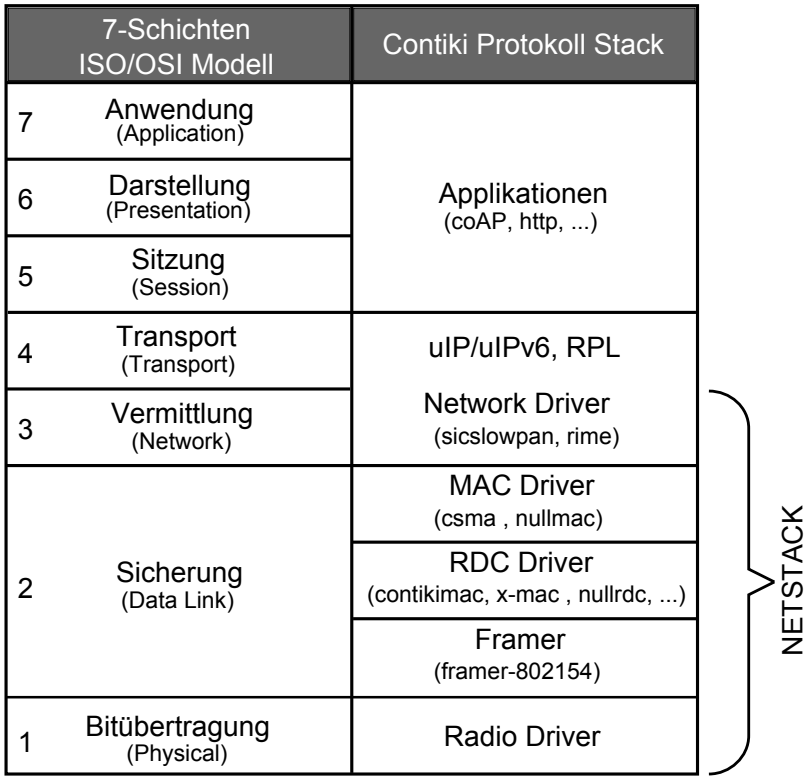

Abbildung 2.24: Contiki Protokoll Stack

2.2.3.3 Funktionale Beschreibung des Netzwerk Stack

Im Folgenden werden die Funktionen der einzelnen Treiber näher beschrieben:

- *Radio Driver*: Der Radio Driver greift direkt auf das Funkmodul zu und ist im Wesentlichen zuständig für die Funkübertragung. Dazu zählt neben dem Senden und Empfangen von Daten auch das Prüfen, ob ein Kanal offen für Übertragungen ist. Dieser Prozess wird als CCA bezeichnet. Dabei wird die Stärke des eingehenden Signals, auch als Received Signal Strength Indication (RSSI) bezeichnet, für einen Kanal gemessen. Liegt dieser unter einem bestimmten Wert, gilt der Kanal als frei und steht zur Übertagung zur Verfügung (vgl. EGN07). Des Weiteren bietet der Radio Driver die Funktionalität

zum ein- beziehungsweise ausschalten des Funkmoduls. Da der Radio Driver direkt auf die Hardware zugreift, ist eine Implementation plattformabhängig.

- *Framer*: Der Framer ist zuständig für die Erzeugung der MAC Frames. Dazu erhält er die notwendigen Informationen für den MAC Header sowie die zu übertragenden Daten von den höheren Schichten. Diese MAC Frames werden dann dem Radio Driver übergeben und versendet. Beim Empfang von Nachrichten besteht die Aufgabe für den Framer darin, die Informationen aus dem empfangenen MAC Frame zu extrahieren. Das heißt, die Informationen aus dem MAC Header und die eigentlichen Daten werden den höheren Schichten zur Verfügung gestellt. Der Austausch der Daten zwischen Framer und den höheren Schichten wird in Contiki durch Puffer realisiert (vgl. Conb).

- *RDC Driver*: Der RDC Driver bietet die Möglichkeit einer energieeffizienten Kommunikation von Netzknoten. Das heißt, er bestimmt zu welcher Zeit das Funkmodul aktiv ist, um entweder emfangsbereit zu sein oder Daten zu versenden. Der RDC Driver ist die Verbindung zwischen dem Radio Driver und dem MAC Driver. In Contiki gibt es mehrere implementierte Protokolle für den RDC Driver die unterschiedliche Methoden zur Datenübertragung bereitstellen. Standardmäßig wird in Contiki hierfür ContikiMAC verwendet. Neben diesem existiert noch das Low Power Probing (LPP) Protokoll, das X-MAC Protokoll und das CX-MAC Protokoll. Des Weiteren besteht die Möglichkeit der Verwendung des nullrdc Treibers, der keinen *radio duty cycle* nutzt und das Funkmodul somit immer aktiv bleibt (vgl. Cond).

- *MAC Driver*: Der MAC Driver ist für den Austausch der Nachrichten zwischen dem RDC Driver an dem Network Driver verantwortlich. Es stehen zwei Implementationen zur Verfügung: nullmac und csma. Die csma Implementierung setzt das CSMA/CA Verfahren um. Dazu wird die CCA Funktionalität des Radio Treibers genutzt.

- *Network Driver*: Der Network Driver übernimmt die Funktionalität der Vermittlungsschicht. Contiki bietet hierfür eine 6LoWPAN Implementation, die IPv6 unterstützt und als Adaptionsschicht zwischen dem uIP-Stack und dem MAC Driver dient. Dies bedeutet, dass große IPv6 Pakete vor dem Versenden in kleinere Pakete fragmentiert werden müssen. Beim Empfang einer Nachricht werden mehrere kleinere Pakete dann wieder zu einem IPv6 Paket zusammengefügt. Außerdem ist 6LoWPAN für die Kompression des Headers eines Paketes zuständig, um den Nutzdatenanteil eines Paketes zu erhöhen (vgl. Cona). Als eine alternative zu 6LoWPAN existiert eine zweite Implementierung: der RIME-Stack. Dieser bietet eine Reihe leichtgewichtiger Kommunikationsprimitive, welche eine Vielzahl von Anwendungsmöglichkeiten für Netze mit begrenzten Ressourcen findet. Es wird jedoch kein IPv6 unterstützt (vgl. Cone).

3

Stand der Technik

In diesem Kapitel werden zwei bestehende Implementierungen des in Kapitel 2.1 erläuterten IEEE 802.15.4 Standards vorgestellt. Dabei wird die grundlegende Architektur sowie die bereitgestellten Funktionalitäten der MAC Software Stacks näher beschrieben.

3.1 Atmel IEEE 802.15.4 MAC Software Stack

Der MAC Software Stack von Atmel[1] wurde gemäß des IEEE 802.15.4 Standards entwickelt und bietet alle Mechanismen und Funktionalitäten die dieser vorsieht. Es werden verschiedene Plattformen, wie Mikrocontroller, Boards und Funkchips der Firma Atmel unterstützt. Die Architektur des MAC Software Stacks erlaubt eine einfache Portierung zwischen den verschiedenen Plattformen und die Anpassung an unterschiedliche Bedürfnisse bezüglich der vorhandenen Ressourcen. Es wird die Kommunikation in Stern Netzwerken sowie in Peer-to-Peer Netzwerken unterstützt. Zudem besteht die Möglichkeit der Nutzung des *beacon-enabled* beziehungsweise des *non-beacon-enabled* Übertragungsverfahren (siehe Absch. 2.1.4).
Die Architektur des MAC Software Stacks ist in verschiedene Schichten unterteilt und wird in Abbildung 3.1 dargestellt.
Die einzelnen Module des Stacks, von der untersten Schicht angefangen, sind im Folgenden aufgelistet und werden im Weiteren näher beschrieben(vgl. Atm12):

- Platform Abstraction Layer (PAL)
- Tranceiver Abtraction Layer (TAL)
- MAC - beinhaltet MAC Core Layer(MCL) und MAC-API

[1] http://www.atmel.com/tools/ieee802_15_4mac.aspx

- Security Abstraction Layer (SAL) und Security Toolbox (STB)
- Ressource Management

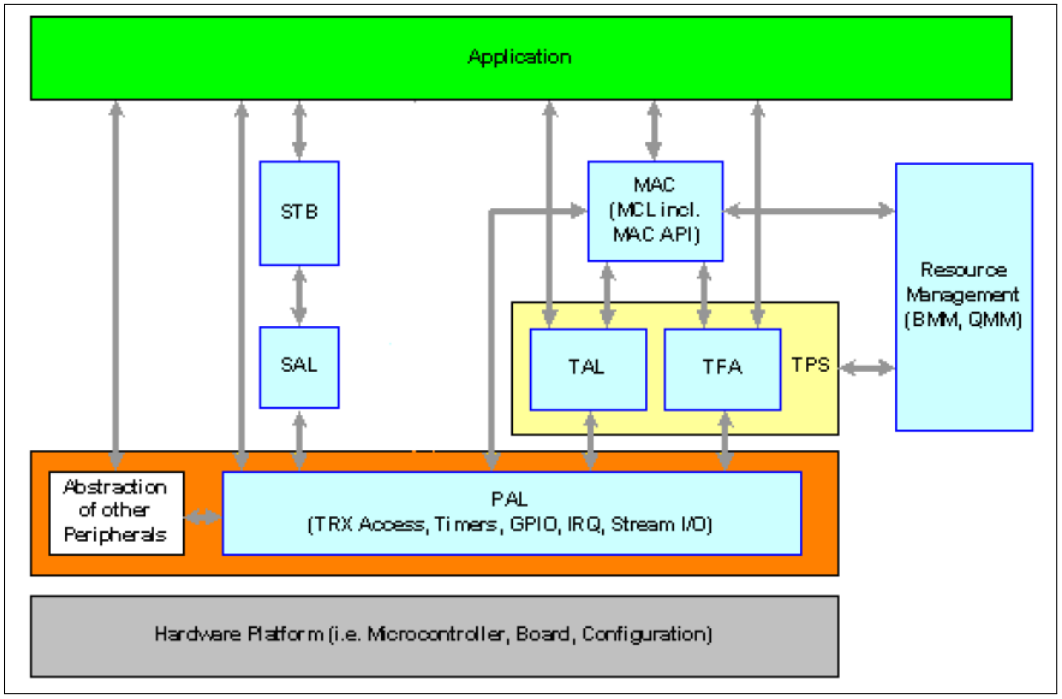

Abbildung 3.1: Atmel MAC Architektur (Atm12)

Platform Abstraction Layer (PAL)
Diese Schicht beinhaltet alle plattformspezifischen Funktionalitäten und bietet die Schnittstellen für die höheren Schichten. Für jeden unterstützten Mikrocontroller existiert eine eigene Implementation innerhalb der PAL. Zudem kann das verwendete Board mit Hilfe einer Konfigurationsdatei angepasst werden. Die PAL stellt den Zugriff auf die verschiedenen Komponenten der Hardware zur Verfügung. Dazu zählt der Zugriff auf das Funkmodul mittels Serial Peripheral Interface (SPI) oder direktem Speicherzugriff, der Zugriff auf den persistenten Speicher (EEPROM), die Ein- und Ausgabe via USB oder UART, die Low-Level Interrupt Behandlung sowie die Ansteuerung und Kontrolle der Timer, LEDs und Buttons (vgl. Atm12).

Tranceiver Abtraction Layer (TAL)
Diese Schicht beinhaltet die Funktionalität bezüglich des Funkmoduls. Sie bietet eine Schnittstelle zu dem MAC Core Layer (MCL), der unabhängig vom verwendeten Funkmodul verwendet werden kann. Die TAL API kann auch direkt von der Applikationsschicht genutzt werden, falls keine MAC Funktionalitäten benötigt werden. Die folgenden Komponenten sind in der TAL implementiert: Frame Übertragungseinheit, Frame Empfangseinheit, Zustandsmaschine, TAL PIB Speicher, CSMA Modul, Energy detect Scan, Power Management, Interrupt Behandlung, Initialisierung und Reset. Jedes unterstützte Funkmodul besitzt eine eigene Implementierung (vgl. Atm12).

Media Access Control (MAC)
Diese Schicht beinhaltet den MCL und die MAC-API. Die MCL implementiert das Verhalten für *non-beacon-enabled* beziehungsweise *beacon-enabled* Netzwerke bezüglich des IEEE 802.15.4 Standards. Dies beinhaltet unter anderem den MAC Data Service, MAC Management Service, MAC Beacon Manager, MAC Incoming Frame Processor und das MAC PIB Modul. Die MAC-API stellt die Schnittstelle zwischen der Applikation und dem MAC Stack dar. Sie sendet Anfragen(Requests) und Antworten(Responses) zum Stack, indem die Funktionen, die von der MAC-API bereitgestellt werden, aufgerufen werden. Diese Anfragen beziehungsweise Antworten werden von der MAC-API in eine Warteschlange eingeordnet und nacheinander abgearbeitet (vgl. Atm12).

Security Abstraction Layer (SAL) und Security Toolbox (STB)
Die SAL stellt eine Schnittstelle zu low-level AES Funktionen zur Verschlüsselung beziehungsweise Entschlüsselung von Frames bereit. Diese Funktionen wurden in Abhängigkeit der verwendeten Hardware implementiert. Die STB bietet für die Applikationsschicht eine einfach zu nutzende Schnittstelle zur Ver- und Entschlüsselung an. Sie liegt auf der SAL und abstrahiert und implementiert vom Funkmodul beziehungsweise Mikrocontroller abhängige Sicherheitsfunktionalitäten (vgl. Atm12).

Ressource Management
Das Ressource Management stellt den Zugriff auf die Ressourcen des MAC Stacks oder der Applikation bereit. Es unterteilt sich in das Buffer Management zur dynamische Zuteilung von Speicher für verschiedene Puffer und in das Queue Management zur Erzeugung und Verwaltung von Warteschlangen (vgl. Atm12).

3.2 TIMAC - IEEE 802.15.4 MAC Software Stack

Der TIMAC IEEE802.15.4 MAC Software Stack[2] wurde basierend auf dem IEEE 802.15.4 Standard von der Firma Texas Instruments[3] entwickelt. Er unterstützt bisher drei Plattformen von Texas Instruments. Das heißt, drei verschiedene Mikrocontroller und die dazugehörigen Boards. TIMAC ermöglicht zudem eine einfache Applikationsentwicklung sowie eine einfache Portierung auf andere Plattformen (vgl. TI).
Das System besteht grundsätzlich aus 2 Schichten: der Operating System Abstraction Layer (OSAL) und der Hardware Abstraction Layer (HAL). Die OSAL beinhaltet ein Modul für die Applikationsebene (MAC Sample Application) sowie ein Modul für die MAC Schicht. In Abbildung 3.2 wird dieser Systemaufbau dargestellt. Im Folgenden werden die einzelnen Komponenten näher beschrieben.

Operating System Abstraction Layer
OSAL stellt die Ereignisbehandlung, das Message-Passing, die Timer, die Zuteilung von Speicher und andere Dienste bereit (vgl. TIM11).

[2]http://www.ti.com/tool/timac
[3]http://www.ti.com/ww/de/

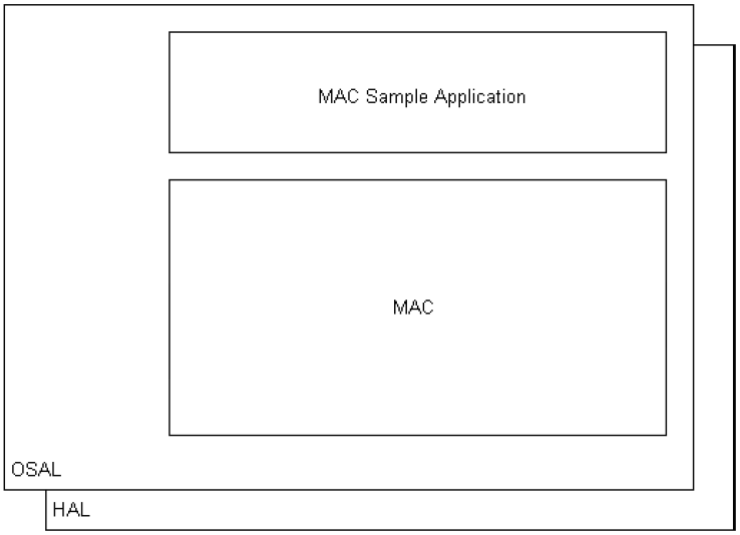

Abbildung 3.2: TIMAC Systemübersicht (TIM11)

MAC Sample Application
Die MAC Sample Application beinhaltet die grundlegenden Protokolle und Funktionen für den Nutzer, um die 802.15.4 MAC Funktionalität zu testen und zu verifizieren. Sie dient somit als Beispiel für eine eigene Implementation der Applikationsschicht. Die grundlegenden Funktionen sind (vgl. TIM11):

- Starten des Devices und Festlegung des Device Typs: Koordinator oder Device (*beacon-enabled* oder *non-beacon-enabled*)
- Beitreten beziehungsweise Erzeugen eines PANs
- Scannen nach einem PAN-Koordinator
- Senden/Empfangen von Daten
- Beacon Unterstützung

MAC
Das MAC Modul verarbeitet die Events und Nachrichten der OSAL, wie beispielsweise empfangene Command Frames oder den Status der Nachrichtenübertragung. Zudem werden weitere grundlegende Funktionalitäten bereitgestellt (vgl. TIM11):

- MAC Funktionen: Scannen, Association/Disassociation, Start des Netzes, Indirekter Datentransfer, Polling, Beacons
- PIB Management
- Erstellen und Parsen der Frames
- Funkmodul Management
- MAC Timer Management

- Empfang und Senden der Frames (beinhaltet CSMA)
- Empfangsbestätigung von Frames (Acknowledgements)
- Ver-/Entschlüsselung
- Power Management

Hardware Abstraction Layer
HAL stellt eine plattformunabhängige Schnittstelle zu den Hardwarekomponenten und deren Diensten bereit.

4

Analyse

Die Analyse beginnt mit einer Problemdefinition. Hierzu werden die wichtigen Kriterien zur Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard herausgestellt. Anhand dessen, folgt eine genauere Analyse des Betriebssystems Contiki bezüglich des IEEE 802.15.4 Standards. Die Ergebnisse der Analyse werden anschließend zusammenfassend dargestellt, um die Anforderungen hinsichtlich der Konzeption und Implementation genauer zu spezifizieren.

4.1 Problemdefinition

Das Thema der Arbeit ist die Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard.
Mediumzugriffsverfahren bestimmen wie und wann ein Netzteilnehmer auf ein gemeinsames Medium zugreifen darf um Daten zu übertragen. Der IEEE 802.15.4 Standard legt hierfür das CSMA/CA-Verfahren fest. Dies ist jedoch nur ein Aspekt den der Standard definiert. Um eine Kompatibilität zum Standard zu erreichen, müssen weitere Punkte erfüllt sein. Dies beginnt mit der Unterscheidung verschiedener Typen von Netzteilnehmern (siehe Absch. 2.1.2) und Netzwerktopologien (siehe Absch. 2.1.3). Zudem muss das Protokoll zur Datenübertragung eingehalten werden. Dazu sind mehrere Arten der Übertragung im Standard definiert (siehe Absch. 2.1.4). Die MAC Schicht ist außerdem zuständig für die Bereitstellung bestimmter Dienste für die höheren Schichten (siehe Absch. 2.1.5.2). Bei diesen Diensten, auch als MAC Services bezeichnet, wird unterschieden zwischen dem MAC Data Service und dem MAC Management Service. Der MAC Data Service stellt verschiedene Funktionen zur Übertragung von Daten bereit und der MAC Management Service ist zuständig für den Aufbau und die Verwaltung des Netzes.
Um eine Kompatibilität zum IEEE 802.15.4 Standard zu erreichen, müssen die eben

genannten Punkte und deren Funktionalitäten in Contiki integriert werden. Dies setzt eine genaue Analyse des Betriebssystems voraus. Ziel der Analyse ist es durch die Erfassung bestehender Strukturen in Contiki zum einen zu zeigen, ob bereits Funktionalitäten bezüglich des Standards existieren und zum anderen sollen im Hinblick auf die Konzeption genaue Informationen über die internen Abläufe erhalten werden, um die Integration neuer Funktionalitäten zu ermöglichen.
Um den Umfang der Arbeit zu begrenzen, werden die Aspekte zur Sicherheit und Verschlüsselung bezüglich des IEEE 802.15.4 Standard nicht betrachtet.

4.2 Untersuchung des Betriebssystems Contiki bezüglich des Standards 802.15.4

Die Untersuchung des Betriebssystems Contiki in Bezug auf den IEEE 802.15.4 Standard basiert im Wesentlichen auf einer Quellcodeanalyse. Es sind speziell die Quellcodedateien zur Datenübertragung untersucht worden, die sich im Ordner */contiki2.x/core/net* befinden. Für die Datenübertragung auf der Bitübertragungsschicht ist der plattformabhängige Quellcode für die AVR Funkmodule der Firma Atmel zur Analyse herangezogen worden. Dieser befindet sich im Ordner */contiki-2.x/cpu/avr/radio*.

4.2.1 Komponenten und Topologie

Das Betriebssystem Contiki unterscheidet nicht zwischen einem Reduced Function Device (RFD) und einem Full Function Device (FFD) (siehe Absch. 2.1.2). Zudem besteht keine Möglichkeit ein Netzteilnehmer als PAN-Koordinator zu definieren. Die PAN-ID ist im Quellcode vorhanden. Jedoch wird sie nur als Konstante definiert, kann zur Laufzeit nicht geändert werden und dient lediglich zur Filterung der Nachrichten auf der Hardwareebene. Das heißt, die PAN-ID wird in ein spezielles Register des Funkmoduls geschrieben, sodass eingehende Nachrichten mit der richtigen PAN-ID im Kopf der Nachricht an die höheren Schichten weitergeleitet werden können. Dies hat zur Folge, dass Netzteilnehmer mit verschiedenen vordefinierten PAN-IDs nur über die Broadcast PAN-ID kommunizieren können.
Aufgrund der fehlenden Unterscheidung zwischen RFD, FFD und PAN-Koordinator besitzen alle Netzteilnehmer den gleichen Funktionsumfang, weswegen eine Stern-Topologie nicht möglich ist. Folglich handelt sich um eine Peer-to-Peer Topologie, in der alle Netzteilnehmer als FFDs agieren. Die höheren Schichten von Contiki stellen dabei eine Routing Funktionalität zur Verfügung, damit auch nicht direkt benachbarte Knoten miteinander kommunizieren können.

4.2.2 Übertragungsverfahren

Da das Betriebssystem Contiki nicht zwischen RFD und einem FFD unterscheidet, folgt es dem Peer-to-Peer Datenübertragungsmodell. Zudem werden in Contiki keine Beacon Frames unterstützt, sodass nur ein *non-beacon-enabled* Übertragungsverfahren möglich ist (siehe Absch. 2.1.4).

Im Hinblick auf die Konzeption wird im Weiteren der typische Ablauf zum Senden beziehungsweise Empfangen von Nachrichten in Contiki genauer analysiert. Hierbei soll festgestellt werden, welche Funktionalität bereitgestellt wird und welche Datenstrukturen dabei zum Einsatz kommen.
Wie bereits im Abschnitt 2.2.3 beschrieben bietet Contiki einen leichtgewichtigen TCP/IP Stack zur Kommunikation über IPv4 sowie IPv6. Dieser setzt direkt auf den NETSTACK von Contiki auf, der aus fünf Modulen besteht und durch seinen flexiblen Aufbau an die verschiedensten Bedürfnisse angepasst werden kann. Um eine genauere Analyse bezüglich des Datentransfers in Contiki durchzuführen, wurde eine typische Konfiguration des NETSTACK bestimmt. Das heißt, für jedes NETSTACK Modul wurde eine von Contiki angebotene Implementierung ausgewählt. Diese Konfiguration wird in Tabelle 4.1 aufgeführt und dient als Grundlage für die folgende Analyse des Ablaufs zum Versenden beziehungsweise Empfangen eines IP-Paketes in Contiki.

NETSTACK Modul	Verwendete Implementierung
NETWORK	sicslowpan
MAC	csma
RDC	nullrdc
FRAMER	framer-802154
RADIO	rf230bb

Tabelle 4.1: Konfiguration des Contiki NETSTACK

Wichtige Datenstrukturen im Contiki NETSTACK
Alle Schichten des NETSTACKs von Contiki nutzen einen Paketpuffer, den sogenannten *packetbuf*. Dieser hat die Größe, um genau ein Paket zu fassen. Der Paketpuffer enthält das aktuelle Paket, welches von NETSTACK verarbeitet wird. Ein Paket umfasst hierbei den Header sowie die Nutzdaten. Zudem können die Sender- und Empfängeradressen sowie weitere Attribute des Pakets gespeichert werden. Um mehrere Pakete in Warteschlangen zu speichern, stellt Contiki den *queuebuf* bereit. Dieser Puffer wird durch eine doppelt verlinkte Liste realisiert. Contiki bietet zudem Funktionen an, um ein Paket aus dem *paketbuf* in einen *queuebuf* oder umgekehrt, ein Paket aus einem *queuebuf* in den *paketbuf* zu kopieren.

Versenden eines IP-Pakets in Contiki
Die Abbildung 4.1 zeigt den kompletten Ablauf des Versendens eines IP-Pakets in Contiki mit Bezug auf die verwendeten Datenstrukturen. Sie dient als Grundlage für die weiteren Erklärungen.

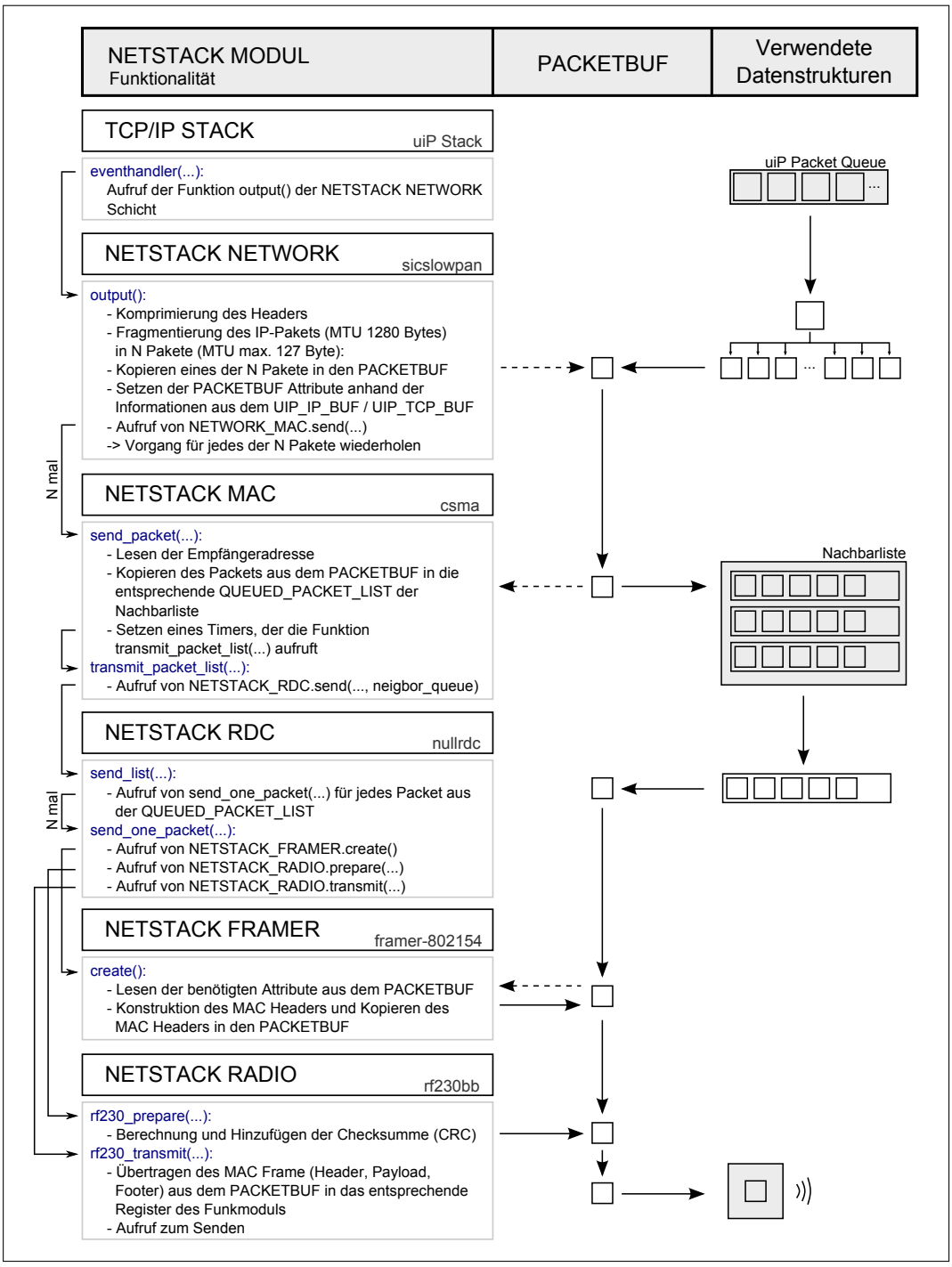

Abbildung 4.1: Senden von Nachrichten im Contiki NETSTACK

Zu Anfang befinden sich die zu versendenden IP-Pakete in der Warteschlange des TCP/IP Stacks (uiP Packet Queue). Durch ein Event an den TCP/IP Prozess wird das Verschicken eines IP-Paketes aus der Warteschlange initiiert. Dabei wird die *output()* Funktion des NETSTACK Network Moduls, einer 6LoWPAN Implementation, aufgerufen.
6LoWPAN dient als Adaptionsschicht zwischen der Transportschicht und der Sicherungsschicht. Hier wird der Header des IP-Pakets komprimiert und eine Fragmentierung vorgenommen, da die Maximum Transmission Unit (MTU) eines IPv6 Paketes laut Standard mindestens 1280 Bytes beträgt, jedoch der IEEE 802.15.4 Standard eine maximale Paketgröße von 127 Bytes vorsieht(vgl. DH98 und vgl. IEE11). Nach der Fragmentierung wird ein Paket in den Paketpuffer(packetbuf) kopiert. Nach dem die benötigten Informationen für das zu versendende Paket aus den Puffern des TCP/IP Stacks gelesen wurden, werden die entsprechenden Attribute im Paketpuffer gesetzt. Danach wird die *send()* Funktion des NETSTACK MAC Moduls aufgerufen. Dieser Vorgang wird für jedes durch die Fragmentierung entstandene Paket wiederholt.

Für das NETSTACK MAC Modul stellt Contiki eine CSMA Implementierung bereit. Diese besitzt eine Nachbarliste, die für jeden Nachbarn einen Eintrag besitzt. Solch ein Eintrag beinhaltet unter anderem die Adresse des Nachbarn, einen Timer und eine Packet Warteschlange(QUEUED_PACKET_LIST). Beim Aufruf der *send_packet()* Funktion durch das NETSTACK Network Modul, wird die Empfängeradresse des Pakets aus dem Paketpuffer gelesen und anhand dieser der Inhalt des Paketpuffers in die jeweilige Warteschlange der Nachbarliste kopiert. Ist das erste Paket in der Warteschlange, wird ein Timer gestartet, dem ein Funktionspointer durch die *transmit_packet_list()* Funktion zugeordnet ist. Beim Ablauf des Timers und dem Aufruf der *transmit_packet_list()* Funktion, ruft diese wiederum die *send()* Funktion des NETSTACK RDC Moduls auf. Als Parameter wird ein Pointer auf den entsprechenden Eintrag in der Nachbarliste übergeben.

Das NETSTACK RDC Modul ist in Contiki zuständig für den Duty Cycle. Das heißt, es wird geregelt in welchem zeitlichen Verhältnis das Funkmodul aktiviert beziehungsweise deaktiviert ist, um Energie zu sparen. Der Fokus der Analyse liegt jedoch auf der bereitgestellten Funktionalität zum eigentlichen Versenden der Pakete und nicht dem zeitlichen Ablauf. Die Kernfunktionalität ist in der *nullrdc* Implementierung vorhanden, sodass diese für die Analyse herangezogen werden kann. In der *send_list()* Funktion wird ein Paket aus der Warteschlange(QUEUED_PACKET_LIST) in den Paketpuffer kopiert und daraufhin die *send_one_packet()* Funktion aufgerufen. Dies wird für jedes Paket aus der Warteschlange durchgeführt. In der *send_one_packet()* Funktion wird die *create()* Funktion des NETSTACK Framer Moduls aufgerufen. In dieser Funktion werden die benötigten Attribute sowie die Adressen aus dem Paketpuffer gelesen und daraus der MAC Header konstruiert. Dieser Header wird anschließend in den Paketpuffer kopiert. Des Weiteren ruft die *send_one_packet()* Funktion die *prepare()* und die *transmit()* Funktion des NETSTACK Radio Moduls auf. In der *prepare()* Funktion wird die Checksumme(CRC) des Pakets berechnet und an den Payload angehangen. In der *transmit()* Funktion

wird der Inhalt (Header, Payload) aus dem Paketpuffer gelesen und in das entsprechende Register des Funkmoduls geschrieben, um ihn anschließend über Funk zu versenden.

Jedes Modul besitzt zudem eine Callback-Funktion, die beim Aufruf einer Funktion eines anderen Moduls als Funktionspointer übergeben wird. Über diese Callback-Funktionen kann der Status beziehungsweise das Ergebnis der Paketübermittlung zurückgegeben werden. Dies wird benötigt, falls es beispielsweise bei einer Übertragung zu einer Kollision kommt. Dann kann das NETSTACK MAC Modul(CSMA) darüber informiert werden, um eine erneutes Übertragen der Pakete einzuleiten. Zur besseren Übersicht wurde in Abbildung 4.1 auf die Darstellung der Callback-Funktionen verzichtet.

Empfangen eines IP-Pakets in Contiki
Die Abbildung 4.2 zeigt den kompletten Ablauf des Empfangs eines IP-Pakets in Contiki mit Bezug auf die verwendeten Datenstrukturen. Sie dient als Grundlage für die weiteren Erklärungen.

Der Empfang eines Pakets beginnt mit einem Event an den Prozess des NETSTACK Radio Moduls. Der Prozess ruft die *read()* Funktion auf, in der die Checksumme überprüft und anschließend das Paket aus dem Register des Funkmoduls in den Paketpuffer(PAKETBUF) kopiert wird. Danach wird die *input()* Funktion des NETSTACK RDC Moduls aufgerufen.
In der *input()* Funktion wird die *parse()* Funktion des NETSTACK Framer Moduls gestartet, in der der MAC Header aus dem Payload des Paketpuffers extrahiert und geparst wird. Daraufhin wird überprüft, ob die Empfänger PAN-ID mit der eigenen übereinstimmt. Die Informationen aus dem Header werden mittels Attributen im Paketpuffer gespeichert. Des Weiteren wird im NETSTACK RDC Modul anhand der Sequenznummer des Pakets, die beispielsweise als Attribut im Paketpuffer gespeichert wurde, überprüft ob, es sich um ein doppeltes Paket handelt beziehungsweise bereits empfangen wurde. Als nächstes wird die *input()* Funtion des NETSTACK MAC Moduls aufgerufen, welche außer dem Aufruf der *input()* Funktion des NETSTACK Network Moduls keinerlei Funktionalität besitzt.
In der *input()* Funktion der 6LoWPAN Implementierung wird das IP Paket wieder zusammengefügt. Hierzu werden die entsprechenden Pakete nacheinander aus dem Paketpuffer in den Puffer des Moduls(SICSLOWPAN_IP_BUF) kopiert. Liegt das IP Paket vollständig in diesem Puffer vor, wird der Inhalt in den Puffer des uiP Stacks(UIP_IP_BUF) kopiert und die *tcpip_input()* Funktion des TCP/IP Stacks aufgerufen.
Die *tcpip_input()* Funktion signalisiert dem TCP/IP Prozess durch ein Event den Erhalt eines neuen IP Paketes.

CSMA/CA-Verfahren in Contiki
Nachdem gezeigt wurde, wie die Übertragung eines IP-Pakets in Contiki erfolgt, wird im Folgenden das CSMA/CA Verfahren in Contiki etwas genauer beleuchtet. Dabei werden nicht alle Funktionen der Module erwähnt, sondern nur die, die im

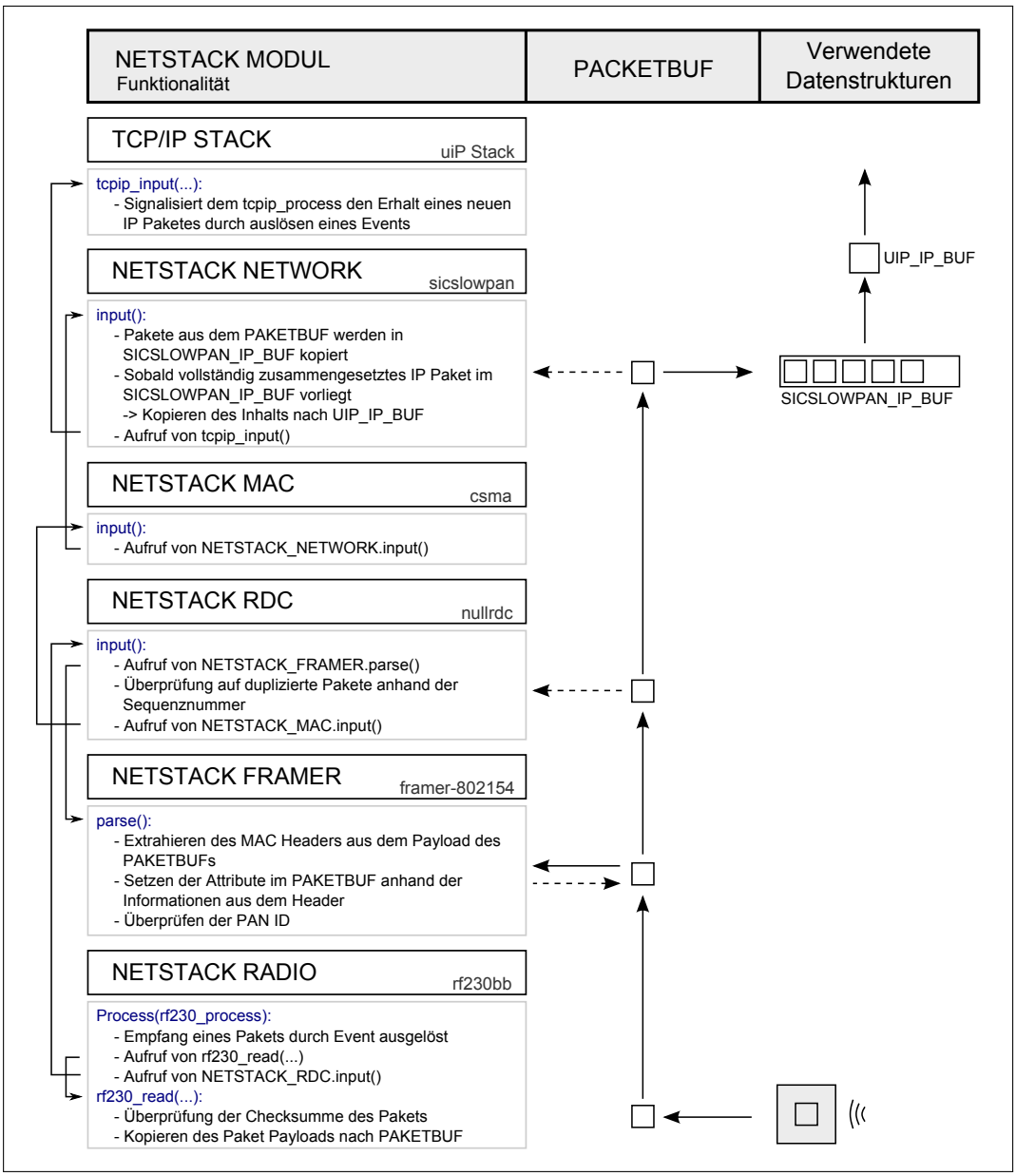

Abbildung 4.2: Empfangen von Nachrichten im Contiki NETSTACK

Wesentlichen für das CSMA/CA Verfahren von Bedeutung sind. Die Abbildung 4.3 zeigt den Ablauf zur Übertragung von Paketen mittels des CSMA/CA Verfahren in Contiki und dient als Visualisierung für die weiteren Erklärungen.

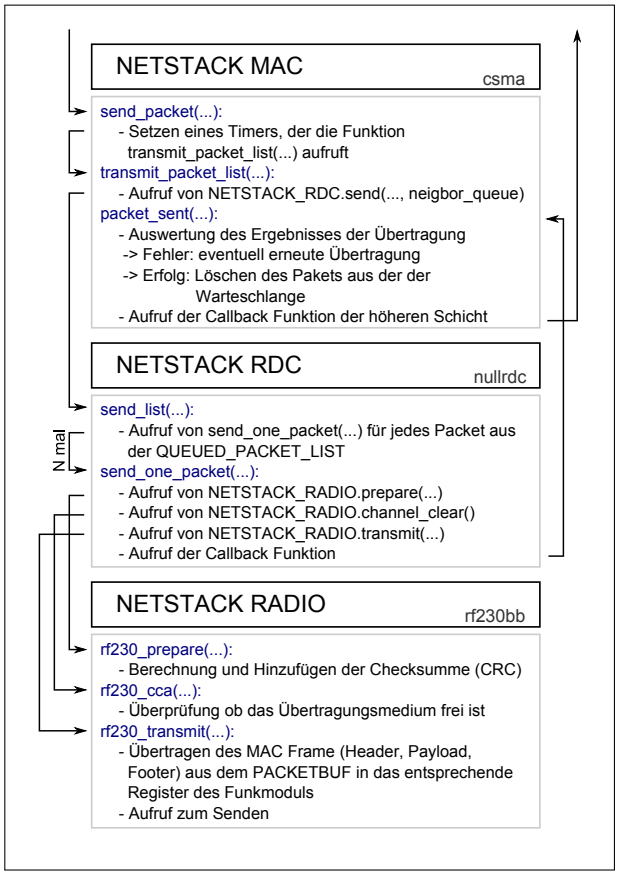

Abbildung 4.3: CSMA/CA im Contiki NETSTACK

In der *send_packet()* Funktion des NETSTACK MAC Moduls wird der Timer der Paketwarteschlange gesetzt, falls dies noch nicht geschehen ist. Nach Ablauf des Timers wird die *send_list()* Funktion des NETSTACK RDC Moduls aufgerufen, die neben einem Pointer auf die Paketwarteschlange, den Pointer auf die Callback-Funktion *packet_sent()* als Parameter erhält.
Die *send_list()* Funktion ruft für jedes Paket aus der Warteschlange die *send_one_packet()* Funktion auf und übergibt den Pointer auf die Callback-Funktion als Parameter. Hier wird die *channel_clear()* Funktion aufgerufen, welche auf die *rf230_cca()* Funktion des NETSTACK Radio Moduls gemappt ist. Diese prüft, ob der Übertragungskanal frei ist. Falls dies der Fall ist, wird mit der Übertragung der Daten durch die *rf230_transmit()* Funktion begonnen. Das Ergebnis der Übertragung wird in einer Variablen gespeichert und der Callback-Funktion als Parameter übergeben. In der Funktion *packet_sent()* wird das Ergebnis der Übertragung ausgewertet. Falls die Übertragung erfolgreich war, wird das Paket aus der Warteschlange entfernt. Bei einer fehlerhaften Übertragung oder einem belegten Übertragungskanal wird geprüft, ob die maximale Anzahl der Übertragungsversuche erreicht ist, um gegebenenfalls den Timer der Paketwarteschlange neu zusetzen und somit eine

erneute Übertragung zu initiieren. Die Zeit für den Timer wird dabei zufällig und in Abhängigkeit der Anzahl der bisherigen Übertragungsversuche bestimmt. Im Falle einer erfolgreichen oder einer endgültig fehlgeschlagenen Übertragung wird die Callback-Funktion der nächst höheren Schicht aufgerufen und somit das Ergebnis der Übertragung mitgeteilt.
Bei der CSMA/CA Implementierung des Betriebssystems Contiki handelt es sich um das unslotted CSMA/CA Verfahren (siehe Absch. 2.1.4.1).

4.2.3 MAC Frames

Die im IEEE 802.15.4 Standard definierte Struktur für MAC Frames, wird von Contiki unterstützt. Zur Erzeugung beziehungsweise zum Parsen des MAC Headers dient die Implementation *framer-802154* des NETSTACK Framer Moduls. Hierbei werden außer dem *Auxiliary Security Header Field* (siehe Absch. 2.1.5.1) alle benötigten Felder des Headers unterstützt. Das heißt, die benötigte Datenstruktur ist vorhanden.
Bei einer Betrachtung der Funktionalität bezüglich des IEEE 802.15.4 Standards sind diverse Punkte nicht erfüllt. So werden derzeit nur Data Frames verschickt und empfangen. Die Verwendung unterschiedlicher Adresstypen (*Short Address* und *Extended Address*) ist vorgesehen, jedoch durch die interne Speicherung der Adressen eingeschränkt. Dies bedeutet, dass der verwendete Adresstyp im Quelltext fest definiert wird und sich zur Laufzeit nicht ändern lässt. Ähnlich verhält es sich bei den PAN-IDs. Sie werden für den Sender und Empfänger vorher im Quelltext fest definiert, so dass keine Änderung zur Laufzeit erfolgen kann.
Der MAC Footer, welcher die Checksumme des MAC Frames enthält, wird durch das NETSTACK Radio Module berechnet und angefügt beziehungsweise überprüft.

4.2.4 MAC Services

Wie im Abschnitt 2.1.5.2 beschrieben, definiert der IEEE 802.15.4 Standard mehrere Dienste für die MAC Schicht. Diese beinhalten die Definition der Primitive und deren Parameter, sozusagen die Schnittstelle zwischen der MAC Schicht und einer höheren Schicht. Zudem wird die grundlegende Funktionalität der Dienste beschrieben. Im Weiteren wird analysiert inwiefern die Schnittstellen der Dienste beziehungsweise deren Funktionalität in Contiki umgesetzt sind. Die Tabelle 4.2 gibt hierzu einen Überblick der MAC Services und deren Umsetzung in Contiki bezüglich der Schnittstellen sowie der Funktionalität.

MAC Data Service

Die im IEEE 802.15.4 Standard definierten Primitive *MCPS-DATA.request*, *MCPS-DATA.confirm* und *MCPS-DATA.indication* existieren in Contiki nicht in dieser Form. Jedoch besteht die Funktionalität zum Versenden und Empfangen von Daten/Paketen wie bereits im Abschnitt 4.2.2 dargestellt wurde. Die Primitive zum Entfernen eines Pakets aus der Warteschlange *MCPS-PURGE.request* und *MCPS-PURGE.confirm* sind nicht definiert und es wird keine Funktionalität in der Form von Contiki bereitgestellt.

Kriterien	Schnittstelle	Funktionalität
Datentransfer:		
DATA	nein	ja
PURGE	nein	nein
Kommunikationseinstellungen:		
GET	nein	nein
SET	nein	nein
RESET	nein	nein
Funkmodulsteuerung:		
RX-ENABLE	nein	ja
SCAN	nein	nein
Netzwerkverwaltung:		
ASSOCIATE	nein	nein
DISASSOCIATE	nein	nein
GTS	nein	nein
ORPHAN	nein	nein
SYNC	nein	nein
SYNC-LOSS	nein	nein
START	nein	nein
BEACON-NOTIFY	nein	nein
POLL	nein	nein
COMM-STATUS	nein	nein

Tabelle 4.2: Umsetzung der MAC Services in Contiki

MAC Management Service
Die im IEEE 802.15.4 Standard definierten Mac Management Service Primitive (siehe Absch. 2.1.5.2) existieren in Contiki bislang nicht. Das heißt, weder die Schnittstellen noch die Funktionalitäten der Dienste sind implementiert. Eine Ausnahme gilt jedoch für den Dienst zur Funkmodul Steuerung (*MLME-RX-ENABLED*). Hierzu existiert zwar keine standardkonforme Schnittstelle, jedoch ist die Funktionalität vorhanden.

MAC Information Base Management Primitives
Eine einheitliche Basis zur Speicherung und Verwaltung der Attribute der MAC Schicht ist in Contiki nicht vorhanden. Die Konstanten und Variablen, welche die Kommunikationseinstellungen beinhalten, verteilen sich auf mehrere verschiedene Quell- und Headerdateien.

4.2.5 Zusammenfassung der Untersuchung

Die Analyse des Betriebssystems Contiki zeigt, dass wesentliche Punkte des IEEE 802.15.4 Standards überhaupt nicht beziehungsweise nur teilweise umgesetzt sind. Die Tabelle 4.3 veranschaulicht nochmals die Ergebnisse der Untersuchung.

Kriterien	Umsetzung	Beschreibung
Komponenten	nein	keine Unterscheidung zwischen FFD, RFD, PAN-Koordinator
Topologie	ja	Peer-to-Peer Topologie
Übertragungsverfahren	teilweise	non-beacon-enabled, kein indirekter Datentransfer möglich
CSMA/CA	ja	unslotted CSMA/CA Implementierung vorhanden
MAC Frame Struktur	teilweise	Struktur zur Erzeugung eines gültigen MAC Frames vorhanden, keine vollständige Nutzung der Felder
MAC Data Service	teilweise	Funktionalität teilweise vorhanden, keine standardkonformen Schnittstellen
MAC Managment Service	teilweise	Funktionalität teilweise vorhanden, keine standardkonformen Schnittstellen
MAC PIB	nein	keine einheitliche Verwaltung der Kommunikationseinstellungen

Tabelle 4.3: Ergebnisse der Untersuchung

4.3 Anforderungsspezifikation

Die Unterscheidung der Netzteilnehmer in RFD, FFD und PAN-Koordinator ermöglicht die Unterstützung der Sterntopologie. Dabei kommunizieren die Netzteilnehmer nicht untereinander, sondern nur mit einem Koordinator. Contiki unterstützt derzeit nur Peer-to-Peer, da alle Knoten den gleichen Funktionsumfang besitzen und somit als FFDs agieren. Die Verbindung beider Topologien ermöglicht den Aufbau spezieller Netze, zum Beispiel das Cluster-Tree Netz (siehe Absch. 2.1.3). Zur Übertragung von Daten in solchen Netzen definiert der IEEE 802.15.4 Standard zwei Verfahren. Dies ist zum einen das *beacon-enabled* Übertragungsverfahren, bei dem der PAN-Koordinator die komplette Verwaltung des Netzes mit Hilfe sogenannter Superframes übernimmt, und zum anderen das *non-beacon-enabled* Übertragungsverfahren, bei dem keine zentrale Verwaltung des Netzes durch einen Knoten erfolgt (siehe Absch. 2.1.4). Um eine einfache Integration in die bestehenden Strukturen Contikis zu gewährleisten, wurde sich für die Implementierung des *non-beacon-enabled*

Übertragungsverfahren entschieden. Dabei sollen drei Arten der Datenübertragung unterstützt werden. Hierbei handelt es sich um die Datenübertragung zum Koordinator, die Datenübertragung von einem Koordinator und die Peer-to-Peer Datenübertragung (siehe Absch. 2.1.4.3). Eine Umsetzung des Datentransfer zum Koordinator ist in Contiki bereits möglich, da die Daten einfach mittels CSMA/CA übertragen werden können. Der Datentransfer von einem Koordinator muss indirekt erfolgen, da der Empfänger möglicherweise das Funkmodul deaktiviert hat. Die Peer-to-Peer Datenübertragung erfolgt zwischen FFDs, die als Koordinatoren agieren und so ständig empfangsbereit sein müssen. Dies ist in Contiki bereits möglich, da es sich um eine einfache Datenübertragung mittels CSMA/CA handelt.Die Umsetzung des *non-beacon-enabled* Übertragungsverfahren reduziert zudem den Aufwand zur Verwaltung des PAN-Netzes, da die Synchronisation zwischen PAN-Koordinator und Netzteilnehmern entfällt. Folglich werden nicht alle Dienste des MAC Management Service benötigt.

Zur Integration der MAC Services soll ein Grundkonzept entworfen werden, welches die Integration aller MAC Services auf einfache Weise ermöglicht. Ein MAC Service besteht aus einem bis maximal vier Primitiven beziehungsweise Funktionen, die eine Schnittstelle zwischen der MAC Schicht und den höheren Schichten bilden und die Funktionalität des Dienstes kapseln. Anhand der Analyse Contikis wurde festgestellt, dass die Nutzung einer standardkonformen Schnittstelle für den MAC Data Service aufgrund der bisherigen Struktur Contikis nicht möglich ist. Nur durch erhebliche Anpassungen auf den höheren Schichten wäre eine solche Nutzung möglich. Aus diesem Grund, wird auf die Integration einer standardkonformen Schnittstellen zur Datenübertragung zwischen MAC Schicht und der höheren Schicht verzichtet. Dies beeinträchtigt jedoch nicht das Versenden und Empfangen standardkonformer MAC Pakete, da die Funktionalität diesbezüglich durch Contiki bereitgestellt wird (siehe Absch. 4.2.2).

Die Integration der MAC PAN Information Base zur Verwaltung der Kommunikationseinstellungen an einem zentralen Ort, könnte konzipiert und umgesetzt werden. Jedoch hat sich durch die Analyse Contikis herausgestellt, dass die Variablen und Konstanten, welche die Kommunikationseinstellungen beinhalten, auf mehrere verschiedene Quell- und Headerdateien verteilt sind. Die Verwendung einer MAC PAN Information Base wäre nur sinnvoll, wenn die Variablen und Konstanten einheitlich verwaltet werden. Um die MAC PAN Information Base zu integrieren, müssten daher zahlreiche Änderungen an Contiki vorgenommen werden, die sich aktuell noch nicht überblicken lassen. Deshalb wurde von der Konzeption einer solchen MAC PAN Information Base abgesehen. Die Schnittstellen zur Verwaltung sollen im Konzept allerdings vorgesehen werden, um die Nutzung einer MAC PAN Information Base nicht vollständig auszuschließen.

Da in Contiki kaum Umsetzungen bezüglich der MAC Management Services existieren (siehe Tab. 4.2), gibt es folglich auch kein Modul auf der höheren Schicht Contikis, welches die Dienste nutzen könnte. Daher sollte ein Modul zur Verwaltung des Netzes konzipiert werden, das die MAC Services nutzt.

Eine weitere Anforderung zur Integration besteht darin, durch eine einfache Konfiguration Contikis die erweiterten MAC Funktionalitäten nutzen beziehungsweise nicht nutzen zu können. Das bedeutet, es sollen keinerlei Einschränkungen für andere Applikationen entstehen, die diese Implementation der MAC Schicht nicht verwenden. Das heißt, weder die Funktionalität soll beeinträchtigt werden, noch soll sich der Speicherplatzbedarf wesentlich erhöhen.
Ausgehend von den vorhergehenden Betrachtungen und den Erkenntnissen aus der Analyse ergeben sich verschiedene Anforderung an die Konzeption, die in Tabelle 4.4 zusammengefasst dargestellt sind.

Anforderung	Beschreibung
ANF-1	**Komponenten**: • Möglichkeit zur Konfiguration eines Netzknoten als FFD, RFD oder PAN-Koordinator • Möglichkeit zur Konfiguration des Funktionsumfanges eines FFD, RFD oder PAN-Koordinator • Möglichkeit der Konfiguration zur Aktivierung/Deaktivierung der zusätzlichen MAC Funktionalität
ANF-2	**MAC Frame Struktur**: • Möglichkeit zum Erzeugen und Parsen aller verfügbaren MAC Frame Typen (Data Frame, Acknowledgement Frame, Command Frame, Beacon Frame) • Möglichkeit zur Nutzung der PAN-ID, die sich zur Laufzeit ändern kann
ANF-3	**MAC Services**: • Erarbeitung eines Konzepts zur Integration der MAC Services in die MAC Schicht Contikis • Konzeption eines Moduls zur Verwaltung des Netzes, welches die MAC Services nutzt
ANF-4	**Übertragungsverfahren**: Konzept zur Integration des indirekten Datentransfers
ANF-5	**MAC Management Service**: Konzept zur Integration der für das *non-beacon-enabled* Übertragungsverfahren benötigten Dienste für die Kommunikationseinstellungen, Funkmodulsteuerung und die Netzwerkverwaltung. Diese Primitive werden in Tabelle 4.5 dargestellt.
ANF-6	**Erweiterbarkeit**: Möglichkeit zur Integration weiterer MAC Management Services

Tabelle 4.4: Anforderungen an das Konzept zur Integration

Dienst	Request	Confirm	Response	Indication
Kommunikationseinstellungen:				
GET	X	X		
SET	X	X		
RESET	X	X		
Funkmodulsteuerung:				
RX-ENABLE	X	X		
SCAN	X	X		
Netzwerkverwaltung:				
ASSOCIATE	X	X	X	X
DISASSOCIATE	X	X		X
POLL	X	X		
ORPHAN			X	X
COMM-STATUS				X

Tabelle 4.5: Zu integrierende MAC Management Primitive

5

Konzeption

In diesem Kapitel wird ein Konzept zur Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard vorgestellt. Hierzu werden die Anforderungen zugrunde gelegt, die sich aus dem Analysekapitel ergaben (siehe Absch. 4.3). Zu Beginn wird eine Möglichkeit zur Definition der Komponenten erläutert. Danach folgt ein Vorschlag zur Verwendung aller MAC Frames und die Vorstellung eines Grundkonzepts, welches die Integration der einzelnen MAC Services in Contiki beschreibt. Weiterhin wird gezeigt, wie die für das *non-beacon-enabled* Übertragungsverfahren benötigten Dienste integriert werden können und wie das Gesamtkonzept aufgebaut ist.

5.1 Definition der Komponenten

Die unterschiedlichen Komponenten FFD, RFD und PAN-Koordinator definieren sich über die Art und den Umfang der Funktionen, die sie besitzen. Das heißt, es müssen neben der Definition der Komponenten auch die zugehörigen Funktionen definiert werden. Hierzu bietet die Programmiersprache *C* die Möglichkeit der Verwendung von Makros. Durch die Definition eines Makros (#define) und der bedingten Kompilierung (#if) kann die Übersetzung bestimmter Funktionen oder Teile eines Programms vom Zutreffen einer bestimmten Bedingung abhängig gemacht werden. Da beispielsweise ein RFD einen geringeren Funktionsumfang als ein FFD besitzt, sollten die nicht benötigten Funktionen auch nicht kompiliert werden. Auf diese Weise kann die Größe des Programms und somit der benötigte Speicherplatz auf der jeweiligen Zielplattform verringert werden. Aufgrund der begrenzten Ressourcen eines Mikrocontroller, ist dies von Vorteil.
Die Definition der Makros für die Komponenten sowie der zugehörigen Funktionen wird in der Konfigurationsdatei *contiki-conf.h* der jeweiligen Plattform vorgenommen. Dort befinden sich alle Konfigurationen, wie beispielsweise die des Contiki

NETSTACKs. Die Makros zur bedingten Übersetzung werden in den verschieden Modulen, in denen sich die Funktionalität für die Komponenten befindet, eingefügt. Zusätzlich wird ein weiteres Makro definiert, welches zur bedingten Kompilierung der gesamten hinzugefügten Funktionalität ient, ähnlich einem Schalter, der es dem Entwickler erlaubt die erweiterte Funktionalität bezüglich des IEEE 802.15.4 Standards hinzuzufügen oder zu entfernen.
In Abbildung 5.1 wird ein Schema zur Definition der Komponenten gezeigt. Durch die Definition der Komponenten und deren Funktionsumfang mit Hilfe von Makros sowie die Definition eines übergeordneten Makros zur bedingten Kompilierung wird die Anforderung *ANF-1* in Tabelle 4.4 erfüllt.

Abbildung 5.1: Schema zur Definition der Komponenten

5.2 MAC Frame Struktur

Für das Erzeugen und Parsen eines MAC Frames, welches der Definition des IEEE 802.15.4 Standards entspricht, ist in Contiki das NETSTACK Framer Modul und speziell die Implementierung *framer-802154* zuständig. Bei ausgehenden Nachrichten werden in der *create()* Funktion die benötigten Informationen aus dem Paket-

puffer (paketbuf) gelesen, um ein gültiges MAC Frame zu erzeugen. Bei eingehenden Nachrichten werden in der *parse()* Funktion die Header Informationen aus dem MAC Frame extrahiert und als Attribute in den Paketpuffer geschrieben.
Der Paketpuffer besitzt bisher keine Attribute, um die PAN-ID des Senders und Empfängers eines MAC Frames zu speichern. Zudem ist die PAN-ID derzeit als Konstante definiert und wird für das Feld des Senders sowie des Empfängers im MAC Frame verwendet. Dies hat zur Folge, dass Netzteilnehmer mit verschiedenen vordefinierten PAN-IDs nicht miteinander kommunizieren können. Um das Problem zu lösen, werden Attribute für die PAN-ID des Senders und die des Empfängers definiert, ähnlich der Attribute der Adressen für den Paketpuffer. Des Weiteren müssen Anpassungen in der *create()* und *parse()* Funktion vorgenommen werden, um die PAN-IDs des Paketpuffers zur Erzeugung eines MAC Frames zu verwenden beziehungsweise beim Parsen eines MAC Frames die PAN-IDs in den Paketpuffer zu schreiben.
Für die Unterscheidung der MAC Frame Typen existiert zwar ein Attribut im Paketpuffer, jedoch wird dieses in der *create()* und *parse()* Funktion nicht genutzt. Daher müssen zur Verwendung des Attributs für den MAC Frame Typ Anpassungen in den eben genannten Funktionen vorgenommen werden. Dies ermöglicht beispielsweise das Versenden von Command Frames, was für die MAC Services benötigt wird.
Die benötigten Anpassungen am Paketpuffer und am NETSTACK Framer Modul werden in Abbildung 5.2 zusammengefasst dargestellt. Diese Anpassungen ermöglichen das Erzeugen und Parsen der verschiedenen MAC Frame Typen und die Nutzung einer nicht statischen PAN-ID. Damit wird die Anforderung *ANF-2* in Tabelle 4.4 erfüllt.

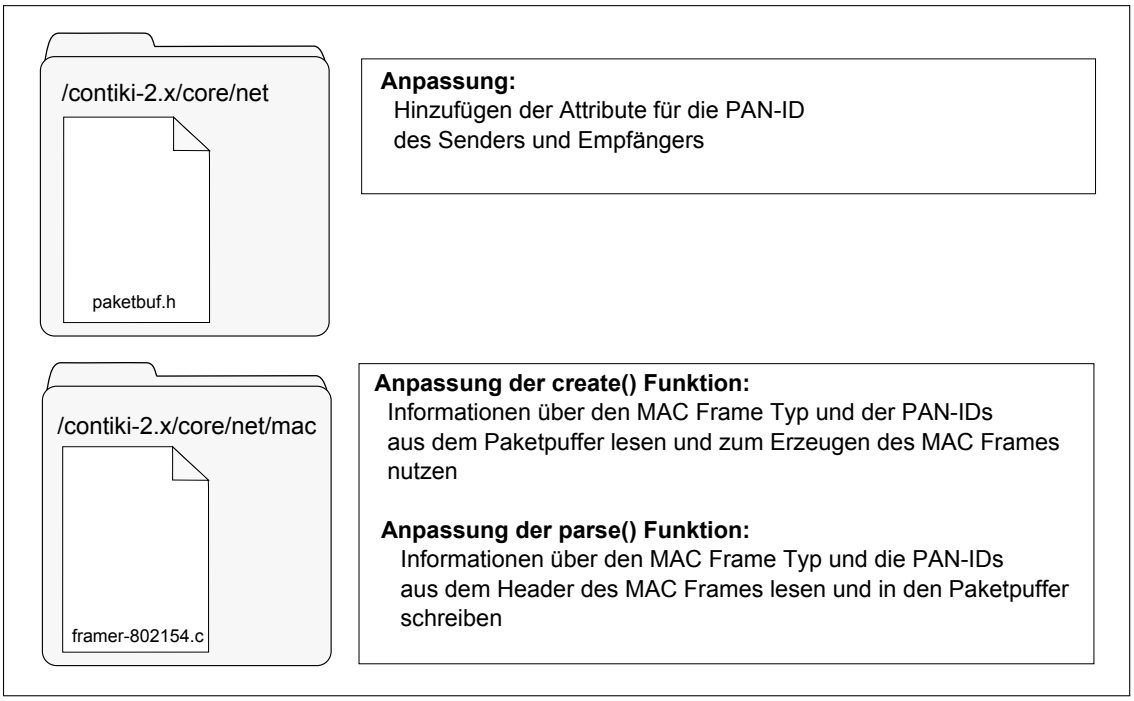

Abbildung 5.2: Anpassungen des Paketpuffers und des Framers

5.3 Integration der MAC Services

Zur Integration der MAC Services wird ein Grundkonzept benötigt, um jeden Dienst im einzelnen nach einem bestimmten Schema zu Contiki hinzuzufügen. Des Weiteren sollten die Dienste in einer bestimmten Weise von einer höheren Schicht genutzt werden können, um beispielsweise das PAN zu verwalten oder eine sichere Datenübertragung zwischen den Netzteilnehmern zu gewährleisten.

5.3.1 Grundkonzept

Zur Integration der MAC Services wurde sich für die Verwendung von zwei Modulen entschieden. Dazu zählt ein Kern Modul, welches die Funktionalitäten der Dienste der MAC Schicht beinhaltet und die Schnittstellen für die höheren Schichten bereitstellt. Dieses Modul wird im Weiteren als MAC Core Modul bezeichnet. Außerdem wird ein Modul auf der höheren Schicht benötigt, welches die Schnittstellen des MAC Core Moduls nutzt und im Wesentlichen die Logik zur Verwaltung eines PANs beinhaltet. Dieses Modul wird im Weiteren als Device Manager bezeichnet werden.
Der Device Manager (siehe Absch. 5.3.1.1), der MAC Core (siehe Absch. 5.3.1.2) und die Interaktion zwischen beiden Modulen (siehe Absch. 5.3.1.3) bilden die Grundstruktur zur Integration der MAC Services in Contiki und erfüllen die *ANF-3* in Tabelle 4.4.

5.3.1.1 Device Manager

Das Betriebssystem Contiki besitzt kein Modul zur Verwaltung eines PANs. Daher muss für diesen Zweck ein neues Modul konzipiert und einige Designentscheidungen getroffen werden.
Die Verwaltung eines PANs ist ein andauernder Prozess. Das bedeutet, die Verwaltung ist zeitlich nicht an irgendeinem Punkt abgeschlossen, da beispielsweise ständig neue Netzteilnehmer hinzukommen beziehungsweise entfernt werden können. Um dies zu erreichen, wurde sich für die Verwendung eines Prozesses, welche in Contiki implementiert sind (siehe Absch. 2.2.2), entschieden. Damit ist es möglich CPU Ressourcen abzugeben, falls keine Aufgaben für das Modul vorliegen. Durch die Implementierung eines Event Handlers wird zudem die Verarbeitung eingehender Events ermöglicht. Der Device Manager Prozess soll so früh wie möglich während der Initialisierung des Betriebssystems Contiki gestartet werden. Dies sollte daher in der Datei *contiki-main.c* der jeweiligen Plattform geschehen.
Der Device Manager beinhaltet die Verhaltenslogik für die unterschiedlichen Komponenten (FFD, RFD, PAN-Koordinator). Durch die Definition der Makros und der bedingten Kompilierung (siehe Absch. 5.1) wird die Trennung der Verhaltenslogik für die Komponenten erreicht, sodass nur die benötigten Programmteile übersetzt werden. Diese Verhaltenslogik ist nicht Bestandteil des IEEE 802.15.4 Standards, wird jedoch für den Aufbau und die Verwaltung eines PANs benötigt. Um später die Funktionsfähigkeit der Umsetzung zu zeigen, wird daher eine grundlegende Logik definiert, die in Tabelle 5.1 dargestellt wird. Diese Logik kann und sollte zuvor an die verschiedenen Anforderungen, die an ein PAN bestehen, angepasst werden.

Phase	RFD/FFD	PAN-Koordinator
Initialisierung:	Initialisierung der Merkmale des Netzknotens (capability informations)	Initialisierung der Liste zur Verwaltung der PAN Teilnehmer
Anmeldephase:	1. Aktiver Scan nach PAN-Koordinatoren 2. Wählen des favorisierten PAN-Koordinators 3. Anmeldeanfrage an den PAN-Koordinator stellen	1. Anmeldeanfragen entgegennehmen und Netzknoten gegebenenfalls in die Liste aufnehmen 2. Antwort schicken
Betriebsphase:	Netzknoten mit begrenzten Energieressourcen, die nicht die ganze Zeit emfangsbereit sin,d müssen durch periodisches Pollen beim PAN-Koordinator nachfragen, ob Daten für sie vorliegen	Verwaltung des PANs: • Eingehende Anmelde- bzw. Abmeldeanfragen bearbeiten • Bereitstellen der Informationen über die Netzteilnehmer für die MAC Schicht

Tabelle 5.1: Verhaltenslogik

5.3.1.2 MAC Core

Um die bestehenden Strukturen des Betriebssystems Contiki zu verwenden, wurde sich für eine eigene Implementierung des NETSTACK MAC Drivers entschieden. Da Contiki bereits eine funktionierende CSMA/CA Implementierung besitzt, wird diese als Basis für die neue Implementierung verwendet. Das heißt, ausgehend von dieser Implementierung, die sich in der Datei *csma.c* befindet, werden neue Funktionalitäten hinzugefügt beziehungsweise bestehende angepasst. Dabei ist zu beachten, dass neue Funktionen, die von anderen Modulen aufgerufen werden, als Funktionszeiger in der Datei mac.h definiert werden müssen (siehe Absch. 2.2.3.1).
In der *input()* Funktion, die vom NETSTACK RDC Modul aufgerufen wird und die MAC Schicht über den Erhalt einer neuen Nachricht informiert, muss die Filterung bezüglich des MAC Frame Typs erfolgen. Damit können beispielsweise die Informationen eingehender Command Frames an den Device Manager weitergeleitet werden. Für Data Frames hingegen wird die *input()* Funktion des NETSTACK Network Moduls aufgerufen.
Genau wie beim Device Manager wird durch die bedingte Kompilierung sichergestellt, dass je nach Konfiguration nur die benötigten Programmteile übersetzt werden.

5.3.1.3 Interaktion zwischen den Modulen

Die Kommunikation zwischen den Modulen findet durch die Aufrufe der Primitive der verschiedenen Dienste statt. Diese Primitive sind Funktionen mit Parameterlisten, welche im Standard sehr genau definiert sind. Es wird zwischen vier Primitiven unterschieden (siehe Abschnitt 2.1.5.2). Die Request und Response Primitive sind Funktionen, die im MAC Core Module implementiert und vom Device Manager aufgerufen werden sollten, da sie verwendet werden um eine Anfrage beziehungsweise eine Antwort von der höheren Schicht an die MAC Schicht zu senden. Bei den Indication und den Confirm Primitiven verhält es sich anders. Diese werden genutzt um der höheren Schicht zu signalisieren, dass ein bestimmtes Ereignis eingetreten ist, auf das in einer bestimmten Weise reagiert werden sollte. Deshalb werden diese Funktionen im Device Manager implementiert und vom MAC Core in gewisser Weise aufgerufen. Im Gegensatz zu den Request und den Response Primitiven handelt es sich dabei nicht um einfache Funktionsaufrufe. Stattdessen wird vom MAC Core ein Event generiert, welches alle benötigten Informationen im Event Kontext bereitstellt und durch den Event Handler des Device Managers verarbeitet wird. Der Vorteil der Events ist die einfache Umsetzung des nicht blockierenden Wartens. Das heißt, das MAC Core Modul generiert ein Event und kann mit der Abarbeitung fortfahren. Es muss nicht, wie beim Aufrufen einer Funktion, warten bis diese zurückkehrt.
In Abbildung 5.3 wird die Interaktion zwischen den Modulen dargestellt.

5.3.2 MAC Data Service

Wie bereits in der Anforderungsspezifikation (siehe Absch. 4.3) beschrieben, ist die Integration der im IEEE 802.15.4 Standard definierten Schnittstellen nicht ohne Weiteres möglich. Zwar kann die Funktionalität zum Versenden standardkonformer MAC Frames genutzt werden, jedoch muss die *send_packet()* Funktion des MAC Core Moduls angepasst werden. Da die höhere Schicht die PAN-ID eines Empfängers nicht in den Paketpuffer schreibt, muss dies in der Funktion erfolgen.

5.3.3 MAC Management Service

Die MAC Management Services umfassen die An- und Abmeldung zu einem PAN, das Polling und dem damit verbundenen indirekten Datentransfer, dem Scannen der Übertragungskanäle, der Orphan Verwaltung und weitere weniger komplexe Dienste. Die Integration der MAC Management Services wird in den folgenden Abschnitten näher erläutert.

5.3.3.1 Anmeldeprozess

Zur Anmeldung eines RFDs oder FFDs zu einem PAN beim PAN-Koordinator werden mehrere Schritte benötigt. Der Anmeldeprozess wird durch den Prozess des Device Managers mit dem Aufruf der *AssociateRequest()* Funktion des MAC Core Moduls gestartet. Es werden im Wesentlichen die Kanalnummer, die PAN-ID und Adresse des PAN-Koordinators sowie die Knoteninformationen (capability information) als Parameter übergeben. Diese Informationen beinhalten unter anderem den

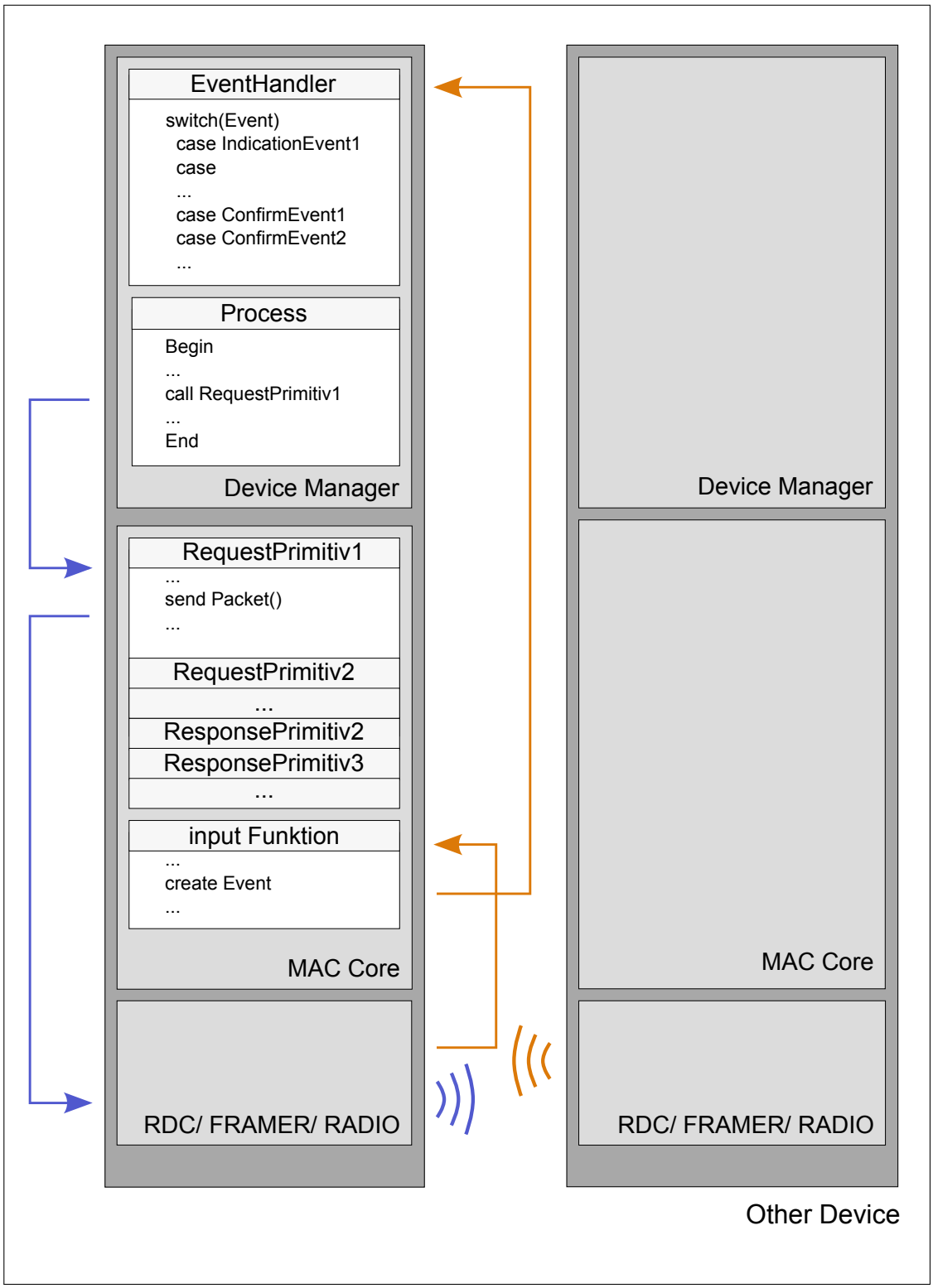

Abbildung 5.3: Interaktionskonzept zwischen den Modulen

Typ (RFD/FFD) des Knotens, die Informationen, ob das Funkmodul zwischen dem Senden deaktiviert wird und ob eine kurze Adresse (16 bit) angefordert werden soll. In der *AssociateRequest()* Funktion werden alle für die Generierung des Associate Request Command Frames benötigten Informationen in den Paketpuffer geschrieben. Das heißt, der Typ des zu versendenden Frames, die PAN-ID und die Adresse des PAN-Koordinators werden festgelegt sowie der Typ des Command Frames und die Knoteninformationen in den Payload kopiert. Danach wird die *send()* Funktion aufgerufen, woraufhin das Associate Request Command Frame erzeugt und über Funk verschickt wird. Beim PAN-Koordinator angekommen, wird das Command Frame geparst, alle Informationen in den Paketpuffer kopiert und die *input()* Funktion des MAC Core Moduls aufgerufen. Hier werden alle benötigten Informationen aus dem Paketpuffer in eine Datenstruktur kopiert. Es wird ein Associate Indication Event mit dem Verweis auf die Datenstruktur generiert, welches dem Device Manager des PAN-Koordinator das Eintreffen eines Associate Requests signalisiert. Der Device Manager prüft, ob ein Beitritt zum PAN möglich ist und nimmt ihn gegebenenfalls in seine Liste auf. In dieser wird die Adresse und die Netzknoteninformationen gespeichert. Um die Entscheidung über den Beitritt zum PAN dem anfragenden Netzknoten mitzuteilen, wird die *AssociateResponse()* Funktion des MAC Core Moduls aufgerufen. Es werden alle Informationen in den Paketpuffer geschrieben, die zur Generierung des Associate Response Command Frame benötigt werden. Danach wird wieder die *send()* Funktion aufgerufen, sodass das Frame erzeugt und über Funk verschickt werden kann. Beim anfragenden Netzknoten angekommen, wird das Frame geparst, die Informationen in den Paketpuffer geschrieben und die *input()* Funktion des MAC Core Moduls aufgerufen. Hier werden die benötigten Informationen in eine Datenstruktur kopiert und das Associate Confirm Event mit dem Verweis auf die Datenstruktur generiert. Im Event Handler des Device Managers wird anhand der Antwort des PAN-Koordinators geprüft, ob die Anmeldung zum PAN erfolgreich war. Daraufhin wird gegebenenfalls die PAN-ID und Adresse des PAN-Koordinators gespeichert und der Status auf „Angemeldet“geändert. Der komplette Anmeldeprozess wird in Abbildung 5.4 dargestellt. Die für den Anmeldeprozess umzusetzenden Funktionalitäten sind durch grüne Schrift hervorgehoben.

5.3.3.2 Abmeldeprozess

Die Abmeldung eines RFDs oder FFDs von einem PAN kann durch den PAN-Koordinator oder dem Knoten selbst initiiert werden. Hierfür muss der Device Manager Prozess die *DisassociateRequest()* Funktion des MAC Core Moduls aufrufen. Als Parameter wird die Empfangsadresse, die PAN-ID, der Grund der Abmeldung sowie die Information darüber, ob indirekte Datenübertragung benötigt wird, übergeben. In der *DisassociateRequest()* Funktion müssen die Informationen in den Paketpuffer kopiert und die *send()* Funktion aufgerufen werden. Außerdem sollte anschließend das Disassociate Confirm Event generiert werden, da laut IEEE 802.15.4 Standard keine Antwort zur Abmeldeanfrage benötigt wird. Auf den unteren Schichten wird das Disassociate Notification Command Frame mit Hilfe der Informationen aus dem Paketpuffer erzeugt und über Funk versendet. Auf der Empfängerseite wird das Command Frame geparst, alle Informationen in den Paketpuffer geschrieben und die *input()* Funktion des MAC Core Moduls aufgerufen. In der *input()* Funktion müs-

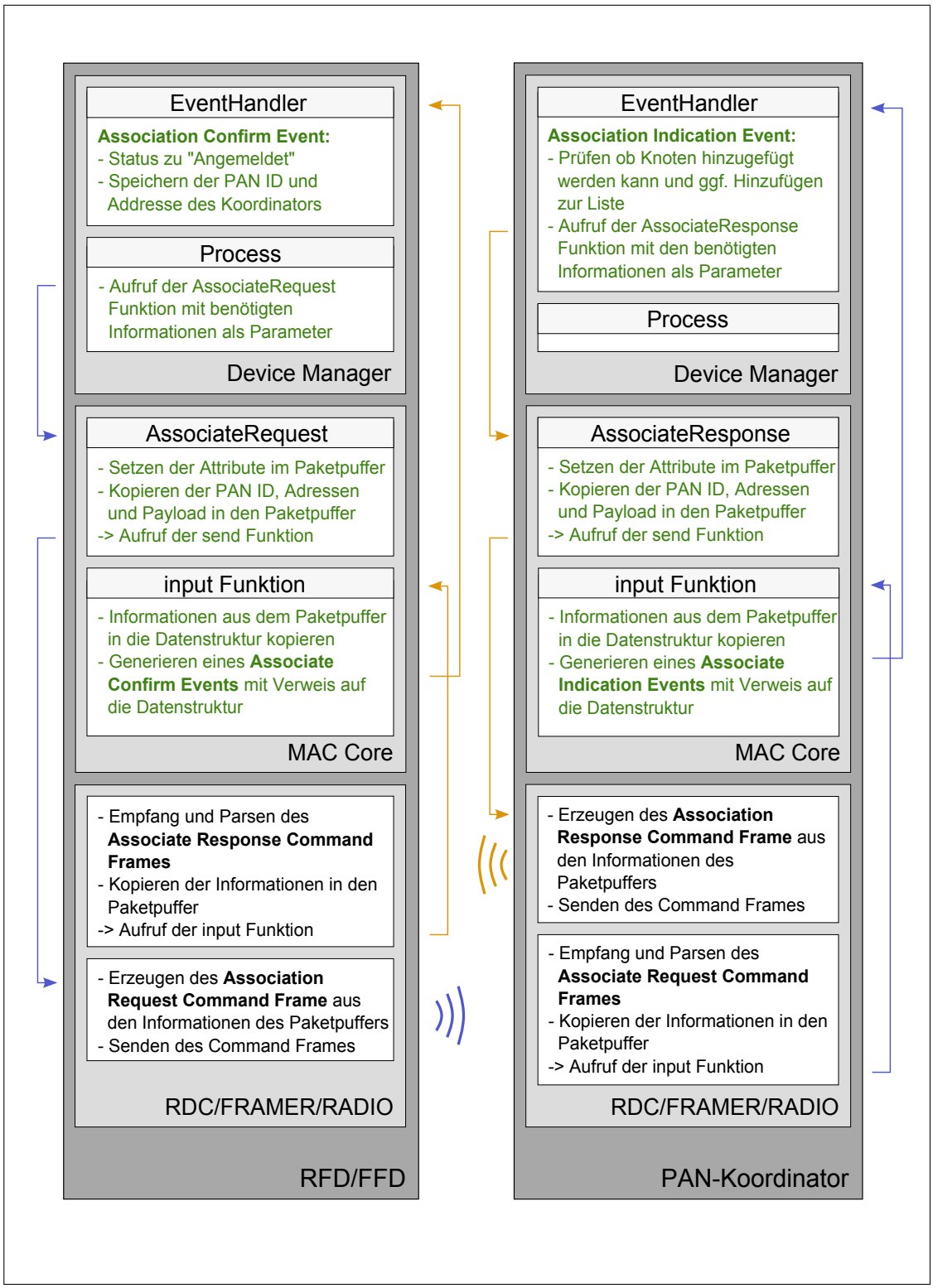

Abbildung 5.4: Anmeldeprozess

sen die benötigten Informationen aus dem Paketpuffer in eine Datenstruktur kopiert werden. Daraufhin sollte ein Disassociate Indication Event mit dem Verweis auf die Datenstruktur generiert werden. Durch das Disassociate Confirm Event und das Disassociate Indication Event werden die Device Manager des PAN-Koordinators und des abzumeldenden Knoten darüber informiert, dass der Knoten sich vom PAN abgemeldet hat. Der Device Manager des PAN-Koordinators entfernt anschließend den Eintrag des Knotens aus seiner Liste und der abgemeldete Knoten ändert seinen Status auf „Abgemeldet“. Wie bereits erwähnt, wird im Gegensatz zum Anmeldeprozess keine Antwort in Form eines Command Frames vom Empfänger des Disassociation Notification Command Frames erzeugt. Hierfür sieht der IEEE 802.15.4 Standard optional die Bestätigung durch ein Acknowledgement Frame vor.
Der Abmeldeprozess, initiiert von einem RFD beziehungsweise FFD, wird in Abbildung 5.5 dargestellt. Die für den Abmeldeprozess umzusetzenden Funktionalitäten sind durch grüne Schrift hervorgehoben.

5.3.3.3 Polling und indirekter Datentransfer

Um eine sichere Übertragung vom Koordinator zu einem Knoten zu gewährleisten, welcher nicht ständig emfangsbereit ist, wird der indirekte Datentransfer verwendet (siehe Absch. 2.1.5.2). Die Grundlage dafür bildet die Verwendung des POLL Primitivs des MAC Management Services.
Die Übertragung der Daten beginnt mit dem Aufruf der *send_packet()* Funktion des MAC Core Moduls. Dabei werden die Daten aus dem Paketpuffer anhand der Empfängeradresse in die entsprechende Paketwarteschlange der Nachbarliste des MAC Core Moduls kopiert und anschließend der Transmit Timer gesetzt. Nach Ablauf des Timers wird die *transmit_packet_list()* Funktion für die Warteschlange des Nachbarn aufgerufen um die Daten an die unteren Schichten weiterzuleiten und per Funk zu übertragen (siehe Absch. 4.2.2). Beim indirekten Datentransfer sollte dieser Transmit Timer jedoch nicht sofort gesetzt werden, da nicht eindeutig ist, ob der Empfänger emfangsbereit ist. Die Information, ob ein Knoten im PAN einen indirekten Datentransfer benötigt, sollte der Device Manager des PAN-Koordinators liefern. Dieser besitzt eine Liste über alle Knoten die dem PAN beigetreten sind. Die Liste beinhaltet, die beim Anmeldeprozess mitgelieferten Informationen über die Knoten (siehe Absch. 5.3.3.1).
Ein Empfängerknoten mit einem nicht dauerhaft aktivierten Funkmodul muss regelmäßig beim Koordinator nachfragen, ob Daten für ihn vorliegen. Dazu sollte der Device Manager periodisch die *pollRequest()* Funktion des MAC Moduls aufrufen. Diese Funktion füllt den Paketpuffer mit den notwendigen Informationen, damit auf den unteren Schichten ein Data Request Command Frame erzeugt und an den PAN-Koordinator verschickt werden kann. Nach dem Empfang des Command Frames durch den PAN-Koordinator wird das Frame geparst, die Informationen in den Paketpuffer kopiert und die *input()* Funktion des MAC Core Moduls aufgerufen. Hier muss überprüft werden, ob Daten in der Nachbarliste für den Knoten vorliegen. Ist dies nicht der Fall, wird ein Acknowledgement Frame mit dem Frame Pending Field gleich 0 erzeugt und verschickt, andernfalls ein Frame Pending Field mit dem Wert 1 erzeugt und verschickt. Außerdem muss der Transmit Timer für die Paketliste des Nachbarn gestartet werden, um ein Frame zu übermitteln. Da nach dem Aufruf

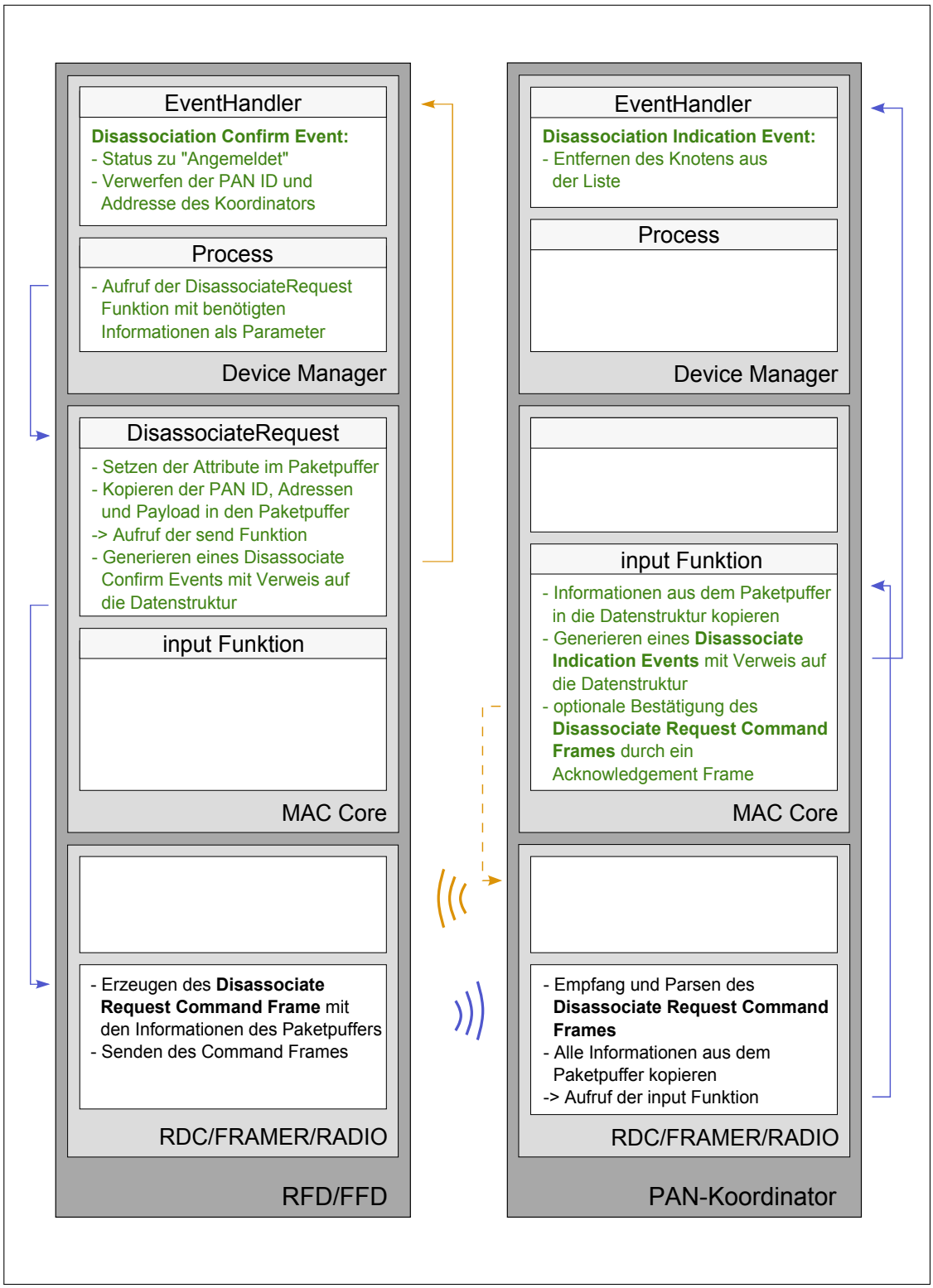

Abbildung 5.5: Abmeldeprozess

der *transmit_packet_list()* Funktion durch den Timer alle Frames übertragen werden, müssen Anpassungen vorgenommen werden, um nur ein Frame zu verschicken. Zudem muss überprüft werden, ob weitere Frames zum Versenden vorliegen, um gegebenenfalls das Frame Pending Field im Paketpuffer zu setzen. In der *input()* Funktion des Empfängerknotens sollte das Acknowledgement Frame ausgewertet und überprüft werden. Ist das Frame Pending Field gleich 0, wird ein Poll Confirm Event für den Device Manager generiert, welches signalisiert, dass das Polling abgeschlossen ist und keine Daten zur Übertragung vorlagen. Hat das Frame Pending Field des Acknowledgement Frames den Wert 1, so wird auf die Übertragung eines Data Frames eine bestimmte Zeit gewartet. Wenn ein Data Frame erhalten wurde, wird ein Poll Confirm Event für den Device Manager generiert. Falls beim erhaltenen Data Frame das Frame Pending Field gesetzt war, sollte direkt ein neues Poll Request gestartet werden. Der Standard sieht zudem vor, dass auf die Nutzung der Acknowledgement Frames verzichtet werden kann, sodass nur die Frame Pending Fields der Data Frames ausgewertet werden müssen.
Der Ablauf des indirekten Datentransfers mit Polling wird in Abbildung 5.6 dargestellt. Durch das Konzept zur Integration des indirekten Datentransfers ist die Anforderung *ANF-4* in Tabelle 4.4 erfüllt.

5.3.3.4 Scannen der Übertragungskanäle

Der IEEE 802.15.4 Standard unterscheidet vier verschiedene Scan-Arten: Energy Detection Scan, Active Channel Scan, Passive Channel Scan und Orphan Channel Scan (siehe Absch. 2.1.5.2). Bei allen Arten wird die gleiche Schnittstelle und die gleichen Primitive verwendet. Zum Starten des Scans im Device Manager, wird die *scanRequest()* Funktion des MAC Core Moduls aufgerufen. Als Parameter werden unter anderem der Scan-Typ, die zu scannenden Kanäle in Form einer Liste und die Scan-Dauer übergeben. Ist das Scannen abgeschlossen, sollte ein Scan Confirm Event mit einem Verweis auf eine Datenstruktur, die die Ergebnisse des Scans beinhaltet, generiert werden. Die verschiedenen Scans sollten entsprechend der in Abschnitt 2.1.5.2 beschriebenen Funktionalitäten umgesetzt werden. Dabei ist zu beachten, dass das Warten auf eine Antwort beim Scannen der einzelnen Übertragungskanäle nicht blockierend ist, da sonst während des Wartens beispielsweise keine Frames empfangen und verarbeitet werden können. Weiterhin werden zur Umsetzung der Scan Funktionen die Funktionen des Funkmoduls benötigt, um beispielsweise den Übertragungskanal des Funkmoduls zu ändern. Hierfür bietet das NETSTACK Radio Modul keine plattformunabhängige Schnittstelle. Um bestehenden Funktionen des Funkmoduls als plattformunabhängige Schnittstelle zur Verfügung zu stellen, müssen Anpassungen am NETSTACK Radio Modul vorgenommen werden. Das heißt, es müssen Funktionszeiger definiert werden, welche auf die konkrete Implementierung des jeweiligen Funkmoduls verweisen. Die Umsetzung und Verfügbarkeit der Funktionalitäten bezüglich der Implementierung der verschiedenen Funkmodule standen nicht im Fokus der Analyse und werden im Konzept nicht genauer betrachtet.
Ein spezieller Scan ist der Orphan Channel Scan, da er ein Teil der Orphan Verwaltung ist und im nächsten Abschnitt betrachtet wird.

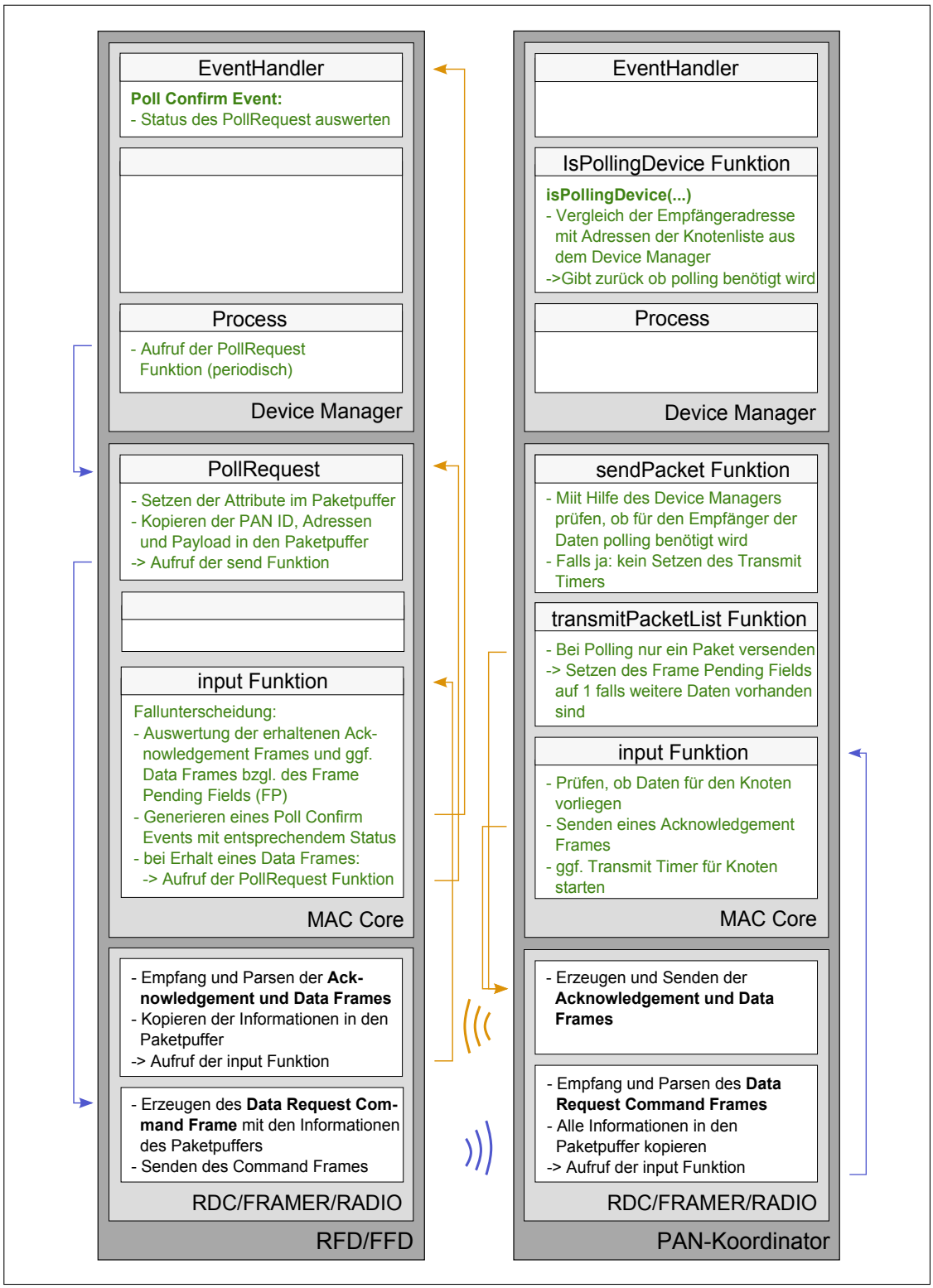

Abbildung 5.6: Indirekter Datentransfer mit Polling

5.3.3.5 Orphan Verwaltung

Ein Orphan Channel Scan erlaubt es einem Knoten, der die Verbindung zu seinem PAN-Koordinator verloren hat, einen Scan durchzuführen, um seinen PAN-Koordinator zu finden. Dieser Scan wird durch den Device Manager mit dem Aufruf der *scanRequest()* Funktion des MAC Core Moduls gestartet. Als Parameter wird der Typ des Scans (Orphan), eine Liste der zu scannenden Kanäle und die Scan-Dauer übergeben. Anhand dieser Informationen muss ein Orphan Notification Command Frames erzeugt werden, indem die entsprechenden Attribute, Adressen und der Payload im Paketpuffer gesetzt werden. Der Übertragungskanal wird daraufhin auf den ersten Kanal in der Liste geändert und die *send()* Funktion aufrufen. Weiterhin muss ein Timer gesetzt werden. Nach dem Ablauf des Timers wird der Übertragungskanal auf den zweiten Kanal in der Liste geändert. Bevor die *send()* Funktion aufgerufen werden kann, muss der Paketpuffer mit den entsprechenden Informationen neu befüllt werden, da durch eingehende Pakete der Paketpuffer sich verändert haben könnte. Der Vorgang muss für alle gewählten Kanäle wiederholt werden, bis jeder der Kanäle gescannt oder ein Coordinator Realignment Command Frame empfangen wurde. Ein PAN-Koordinator der ein Orphan Notification Command Frame erhält, sollte in der *input()* Funktion des MAC Core Moduls ein Orphan Indication Event generieren. Aus den mitgelieferten Informationen prüft der Device Manager, ob der Knoten sich in seiner Liste befindet. Falls dies nicht der Fall ist, gibt er keine Antwort auf das Orphan Notification Command Frame. Andernfalls sollte die *orphanResponse()* Funktion des MAC Core Moduls aufgerufen werden. Dabei müssen die benötigten Adressen, Attribute und der Payload aus der Parameterliste im Paketpuffer gesetzt beziehungsweise kopiert und die *send()* Funktion aufgerufen werden. In den unteren Schichten wird das Coordinator Realignment Command Frame aus den Informationen des Paketpuffers erzeugt und per Funk übermittelt. Beim Empfang des Command Frames muss der Scan weiterer Kanäle abgebrochen und ein Scan Confirm Event für den Device Manager generiert werden.
Die Orphan Verwaltung wird in Abbildung 5.7 dargestellt. Die für den Prozess umzusetzenden Funktionalitäten sind durch grüne Schrift hervorgehoben.

5.3.3.6 Weitere Dienste

In den Anforderungen an die Konzeption wurden weitere Dienste der MAC Schicht festgelegt. Die folgenden Dienste sind weniger komplex als die vorhergehenden, da keine Kommunikation mit anderen Knoten erfolgt, sondern die Dienste nur intern Verwendung finden.

Funkmodul aktivieren/deaktivieren
Da die Funktionalität zur Aktivierung beziehungsweise Deaktivierung des Funkmoduls in Contiki bereits besteht, müssen lediglich die Schnittstellen des MAC Core Moduls angepasst werden. Im MAC Core Modul muss zusätzlich ein RX-Enable Event generiert werden, wenn der Aufruf der Funktion an die unteren Schichten in der RX-EnableRequest Funktion zurückkehrt.

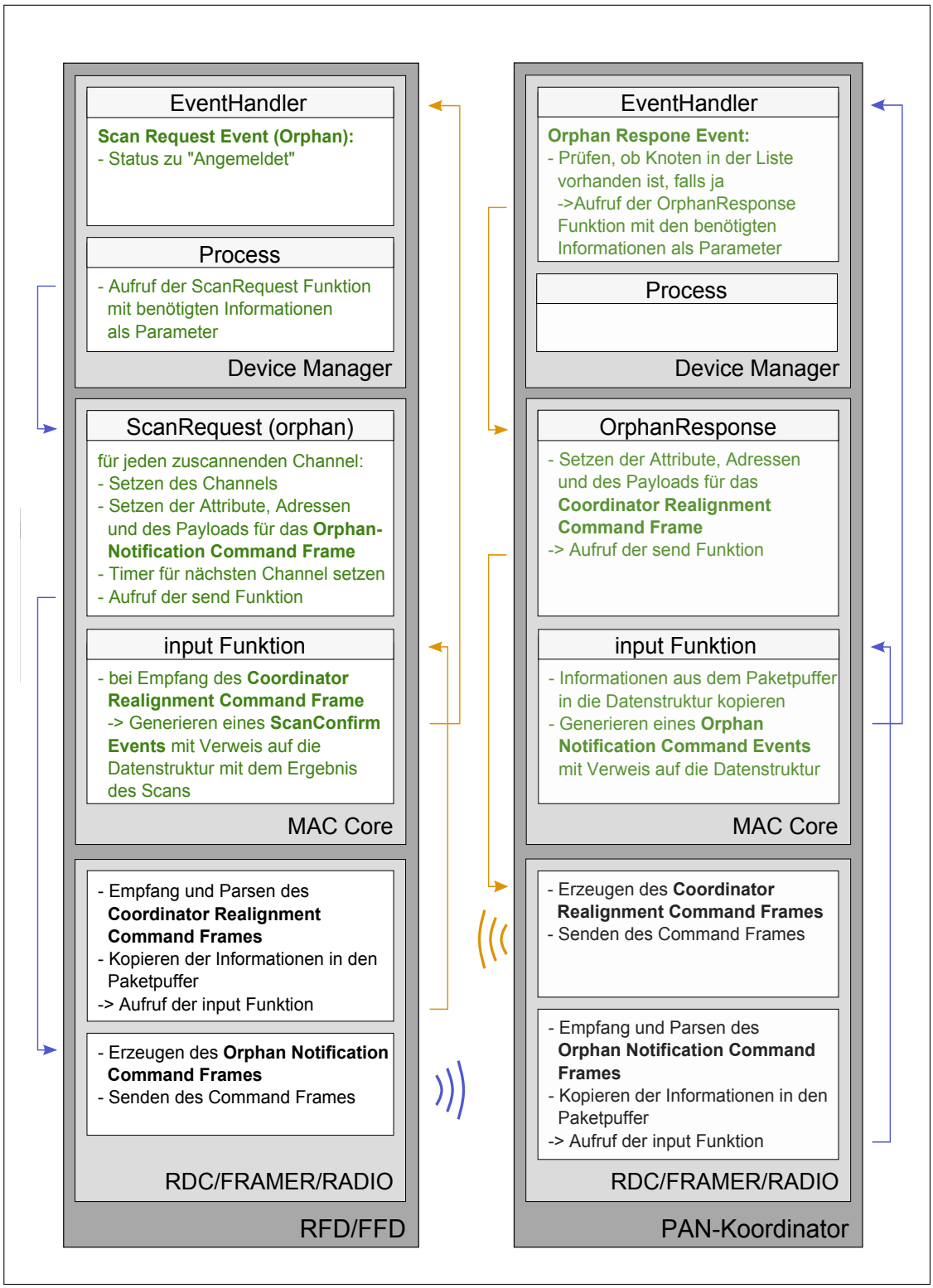

Abbildung 5.7: Orphan Verwaltung

Kommunikationsstatus
Dieser Dienst wird genutzt, um verschiedene Kommunikationsstatusmeldungen vom MAC Core Modul an den Device Manager zu schicken. Dabei werden die Meldungen durch die Erzeugung eines CommStatus Indication Events versandt.

Kommunikationseinstellungen
Zum Auslesen, Ändern und Zurücksetzen der Kommunikationseinstellungen der MAC PAN Information Base sieht der Standard die Dienste GET, SET und RESET mit jeweils einem Request und Confirm Primitiv vor. Die Integration der Schnittstellen erfolgt analog dem Grundkonzept zur Integration der MAC Services. Das heißt, die Request Primitive müssen als Funktionen im MAC Core Modul umgesetzt werden, in denen, nach Abarbeitung der Aufgabe, ein Confirm Event für den Device Manager generiert wird. Eine Konzeption und Umsetzung der MAC PAN Information Base ist aus den in Abschnitt 4.3 genannten Gründen nicht angedacht, jedoch wird die Verwendung einer solchen Datenbasis durch die Konzeption der Schnittstellen ermöglicht.

5.4 Konzeptübersicht

Zusammenfassend lässt sich festhalten, dass zur Integration der MAC Funktionalität in Contiki neue Module umgesetzt beziehungsweise bestehende angepasst oder erweitert werden müssen. In Abbildung 5.8 sind die Module des NETSTACKs und der höheren Schichten dargestellt, die zur Datenübertragung und zur Verwaltung des PANs beitragen. Dabei wird überblicksartig gezeigt, welche Funktionalitäten in welchen Modulen umgesetzt werden sollen.
Im Konzept wurden die durch die Anforderungsspezifikation bestimmten MAC Services betrachtet und eine Möglichkeit zur Integration in Contiki aufgezeigt. Dies erfüllt die Anforderung *ANF-5* in Tabelle 4.4. Für *beacon-enabled* PANs werden weitere Dienste zur Synchronisation benötigt. Diese können nach dem gleichen Schema in das MAC Core Modul integriert werden. Dafür müssen die Funktionen für die Request und Response Primitive definiert werden sowie an den gewünschten Stellen Indication und Confirm Events für den Device Manager generiert werden. Das bedeutet, durch das Grundkonzept (siehe Absch. 5.3.1) wird die Integration weiterer MAC Management Services und somit die Erweiterbarkeit der MAC Funktionalität ermöglicht. Dies erfüllt die Anforderung *ANF-6* in Tabelle 4.4.
Damit sind die Anforderungen aus der Analyse an das Konzept erfüllt, sodass eine Umsetzung erfolgen kann.

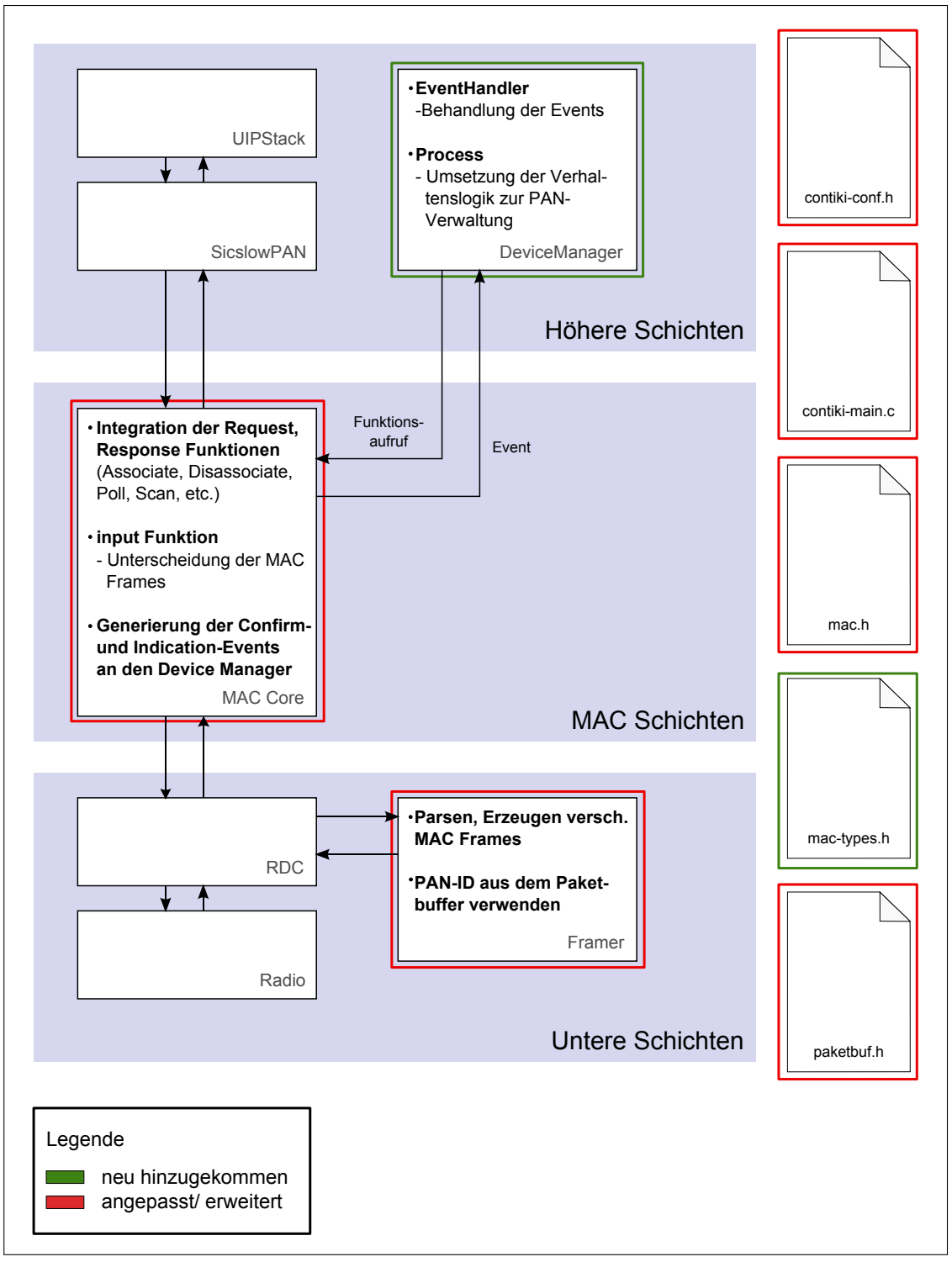

Abbildung 5.8: Konzeptübersicht

6

Realisierung und Test

Dieses Kapitel beschreibt die Realisierung des Konzepts und die durchgeführten Tests zur Validierung der Implementation. Dazu werden, die für die Umsetzung verwendete Hard- und Software beschrieben sowie grundsätzliche Aspekte der Implementierung beispielhaft erläutert. Abschließend erfolgt die Durchführung und Auswertung einiger Tests, um die Funktionsfähigkeit der Implementierung zu überprüfen.

6.1 Verwendete Hard- und Software

Zur Realisierung des Konzepts wurde verschiedene Hard- und Software genutzt. Dabei diente das Betriebssystem Contiki als Grundlage, welches unter dem Betriebssystem Linux entsprechend des Konzepts erweitert wurde. Darüber hinaus diente die Gnu Compiler Collection (GCC) der Kompilierung Contikis und das Programm *avrdude* der Übertragung des kompilierten Betriebssystems Contiki auf einen Mikrocontroller. Zur Analyse des Funkverkehrs zwischen den Mikrocontrollern kam das Programm *wireshark* sowie die Testversion des Protocol Analyzer der Firma Perytons[1] zum Einsatz. Für den Test der Implementation wurde zudem der Atmel IEEE 802.15.4 MAC Software Stack genutzt.
Als Hardware standen zwei Breakout-Boards und zwei Funkmodulen mit integriertem Funkchip zur Verfügung. Zur Stromversorgung und zur Kommunikation zwischen Computer und Mikrocontroller wurden zwei USB-to-UART Sticks verwendetet. Außerdem wurde ein Programmer zum Flashen des Mikrocontrollers benötigt. Zur Analyse des Funkverkehrs zwischen den Mikrocontrollern wurde ein USB-Funk-Stick mit einer vorinstallierten Firmware genutzt.
Die verwendete Hard- und Software und deren genaue Bezeichnung beziehungsweise Version ist in Tabelle 6.1 aufgelistet.

[1] http://www.perytons.com/

Software	
Contiki	2.7
Linux	Ubuntu 12.10
GCC	4.7.2
avrdude	5.11.1
wireshark	1.10.0
Perytons Analyzer	5.2 (Testversion)
Atmel MAC Stack	2.6.1
Hardware	
Breakout Board	deRFbreakout Board (dresden elektronik)
Funkmodul	deRFmega128-22A00 (dresden elektronik)
Funkchip	ATmega128RFA1 (Atmel)
USB-to-UART Stick	USB-A UART Bridge (In-Circuit)
Programmer	AVR Dragon (Atmel)
USB-Funk-Stick	2.4 GHz Cortex-M3 | Analyzer Stick Perytons (dresden elektronik)

Tabelle 6.1: Verwendete Hard- und Software

6.2 Umsetzung des Konzepts

Zur Umsetzung des Konzepts wurden die Komponenten FFD und RFD definiert und die MAC Services integriert. Die wesentlichen Aspekte hierzu werden in den folgenden Abschnitten näher beschrieben.

6.2.1 Definition der Komponenten

Die Unterscheidung der Komponenten FFD und RFD sowie deren Funktionsumfang erfolgt, wie im Konzept angedacht (siehe Absch. 5.1), durch Makros in der Konfigurationsdatei der jeweiligen Plattform. Aufgrund der verwendeten Hardware kam die Plattform *avr-atmega128rfa1* zum Einsatz. Das Listing 6.1 zeigt einen Ausschnitt zur Definition eines RFDs und dem zugehörigen Funktionsumfang. Das MLME_-ASSOCIATE_INDICATION und das MLME_ASSOCIATE_CONFIRM Primitiv wurden hierbei nicht definiert, da diese Primitive für ein RFD, im Gegensatz zu einem FFD, nicht benötigt werden. Die Definition des *MAC_RFD* beziehungsweise *MAC_FFD* Makros erfolgte als Compiler-Option im Makefile der Applikation. Alternativ hierzu kann die Definition in der Konfigurationsdatei der entsprechenden erfolgen.

```
/* 802.15.4 MAC setup */

/** RFD (Reduced Function Device Setup) **/
#ifdef MAC_RFD
#ifndef MAC802154
#define MAC802154
#endif

/* included mac services for RFD */
#define MLME_ASSOCIATE_REQUEST
#define MLME_ASSOCIATE_CONFIRM

#define MLME_DISASSOCIATE_REQUEST
#define MLME_DISASSOCIATE_INDICATION
#define MLME_DISASSOCIATE_CONFIRM

#define MLME_POLL_REQUEST
#define MLME_POLL_CONFIRM

#define MLME_SCAN_REQUEST
#define MLME_SCAN_CONFIRM
#endif  /* MAC_RFD */
```

Listing 6.1: Definition des RFDs und dessen Funktionsumfangs (/contiki-2.7/platform/avr-atmega128rfa1/contiki-conf.h)

6.2.2 Integration der MAC Services

Zur Integration der MAC Services erfolgte die Umsetzung des Mac Core Moduls sowie des Device Managers Dies wird im Folgenden wird näher erläutert.

MAC Core

Die Umsetzung des Mac Core Moduls basiert auf der CSMA/CA Implementierung von Contiki. Diese wurde um die *Request* und *Response* Funktionen der MAC Services erweitert. Für die Integration der MAC Services wurde im Konzept festgelegt, dass die Request und Response Primitive der Dienste als Funktionen im NETSTACK MAC Modul integriert werden. Hierzu erfolgte eine Anpassung der Definition des MAC Treibers aus der *mac.h* Datei. Das Listing 6.2 zeigt die Erweiterung des MAC Treibers um die neuen Funktionszeiger.

```
/**
 * The structure of a MAC protocol driver in Contiki.
 */
struct mac_driver {
  char *name;
  /** Initialize the MAC driver */
  void (* init)(void);
  /** Send a packet from the Rime buffer  */
  void (* send)(mac_callback_t sent_callback, void *ptr);
  /** Callback for getting notified of incoming packet. */
  void (* input)(void);
  /** Turn the MAC layer on. */
  int (* on)(void);
  /** Turn the MAC layer off. */
```

```
  int (* off)(int keep_radio_on);
  /** Returns the channel check interval, expressed in clock_time_t
      ticks. */
  unsigned short (* channel_check_interval)(void);

/*** MAC802154 Service Definitions ***/
#ifdef MLME_ASSOCIATE_REQUEST
  void (* mlme_associate_request)(mlme_associate_request_t *pData);
#endif /* MLME_ASSOCIATE_REQUEST */

#ifdef MLME_ASSOCIATE_RESPONSE
  void (* mlme_associate_response)(mlme_associate_response_t *pData
     );
#endif /* MLME_ASSOCIATE_RESPONSE */

#ifdef MLME_DISASSOCIATE_REQUEST
void (* mlme_disassociate_request)(mlme_disassociate_request_t *
    pData);
#endif /* MLME_DISASSOCIATE_REQUEST */

#ifdef MLME_POLL_REQUEST
void (* mlme_poll_request)(mlme_poll_request_t *pData);
#endif /* MLME_POLL_REQUEST */

#ifdef MLME_SCAN_REQUEST
void (* mlme_scan_request)(mlme_scan_request_t *pData);
#endif /* MLME_SCAN_REQUEST */
...
};
```

Listing 6.2: Anpassung des MAC Treibers (/contiki-2.7/core/net/mac.h)

Da die Parameterlisten der MAC Service Funktionen größer ausfallen können, sind sie als Strukturen definiert. Zudem wird nur ein Zeiger auf diese als Parameter übergeben. Das hat den Vorteil, dass spätere Veränderungen der Parameter der Funktionen an einer zentralen Stelle vorgenommen werden können. Die Strukturdefinitionen und weitere Definitionen für die 802.15.4 MAC Implementation befinden sich in einer neu hinzugefügten Datei *mac802154_types.h*. Im Listing 6.3 wird ein Beispiel der Struktur der Parameterliste für die *mlme_associate_request()* Funktion gezeigt.

```
...
/* association primitives */
typedef struct{
  uint8_t       channelNumber;
  uint8_t       channelPage;
  macAddr_t     coordAddr;
  uint16_t      coordPanId;
  uint8_t       capabilityInformation;
  macSecurity_t sec;
}mlme_associate_request_t;
...
```

Listing 6.3: Struktur für die Associate Request Funktion (/contiki-2.7/core/net/mac802154_types.h)

Die *Request* und *Response* Funktionen wurden im MAC Core Modul umgesetzt. Das Listing 6.4 zeigt beispielhaft die Umsetzung der *Associate Request* Funktion. Diese ist zuständig für das Versenden eines Association Request Command Frames an den PAN-Koordinator. Dabei werden die Informationen, die als Parameter der Funktion übergebenen werden, in den Paketpuffer kopiert beziehungsweise die Attribute und Adressen im Paketpuffer gesetzt. Nach dem Aufruf der *send_packet()* Funktion, wird das Command Frame im NETSTACK Framer Modul anhand der Informationen aus dem Paketpuffer erzeugt und durch das NETSTACK Radio Modul verschickt.

```
...
#ifdef MLME_ASSOCIATE_REQUEST
static void
mlme_associate_request(mlme_associate_request_t *pData)
{
  uint8_t payload[2];

  packetbuf_clear();

  /* generate command frame -association request command- */
  // set payload - command frame type
  payload[0] = CMDFRAME_ASSOCIATION_REQUEST;
  payload[1] = pData->capabilityInformation;
  packetbuf_copyfrom(&payload, sizeof(payload));

  // set coordinator address
  packetbuf_set_addr(PACKETBUF_ADDR_RECEIVER, (rimeaddr_t *) pData
     ->coordAddr.addr.extAddr);

  // set packetbuf attributes - Commandframe, PANID Receiver
  packetbuf_set_attr(PACKETBUF_ATTR_PACKET_TYPE,
     PACKETBUF_ATTR_PACKET_TYPE_COMMAND);
  packetbuf_set_attr(PACKETBUF_ATTR_PAN_ID_RECEIVER, pData->
     coordPanId);

  // send packet
  send_packet(packet_sent, NULL);
}
#endif /* MLME_ASSOCIATE_REQUEST */
...
```

Listing 6.4: Umsetzung der Associate Request Funktion (/contiki-2.7/core/net/mac802154.c)

Die Umsetzung der Indication und Confirm Primitive erfolgt durch die Erzeugung von Events an den Device Manager Prozess. Dabei werden die Parameter wiederum als Zeiger auf Strukturen als Event-Kontext übergeben. Das Listing 6.5 veranschaulicht die Erzeugung eines Associate Indication Events für den Device Manager Prozess, nachdem ein Associate Request Command Frame erhalten wurde.

```
...
#ifdef MLME_ASSOCIATE_INDICATION
      case CMDFRAME_ASSOCIATION_REQUEST:
        // associate indication data
        mlme_associate_indication_t associateIndicationData;
        associateIndicationData.capabilityInformation = payload[1];
        // fill with address from packetbuf
        memcpy(&a.addr.extAddr, packetbuf_addr(
           PACKETBUF_ADDR_SENDER), 8);
        a.addrMode = AM_EXTENDED_ADDRESS;
        associateIndicationData.deviceAddress = a;
        // send event
        process_post_synch(&device_management_process,
           ASSOCIATE_INDICATION_EVENT, &associateIndicationData);
        break;
#endif /* MLME_ASSOCIATE_INDICATION */
...
```

Listing 6.5: Senden eines Associate Indication Events (/contiki-2.7/core/net/mac802154.c)

Device Manager

Um die vom MAC Core bereitgestellten Dienste zu nutzen, wurde der Device Manager umgesetzt. Dieser ist hauptsächlich für die Verwaltung des PANs zuständig (siehe Absch. 5.3.1.1) und wurde unter der Verwendung eines Prozesses und einem Event Handler realisiert. Gestartet wird der Prozess in der *contiki-main.c* Datei der jeweiligen Plattform.

Das Verhalten eines RFDs beziehungsweise FFDs wurde im Device Manager Prozess wie folgt umgesetzt: In der Prozess Schleife wird periodisch überprüft, ob der Knoten bereits bei einem PAN-Koordinator angemeldet ist. Ist er nicht angemeldet, wird geprüft, ob bereits ein PAN-Koordinator durch einen Active Scan gefunden und ausgewählt wurde. Falls kein Koordinator spezifiziert ist, wird ein Active Scan durchgeführt. Andernfalls wird der Anmeldeprozess mit dem gewählten PAN-Koordinator gestartet. Sobald der Active Scan und der darauffolgende Anmeldeprozess erfolgreich abgeschlossen sind, wird periodisch durch Polling beim PAN-Koordinator nachgefragt, ob Daten für ihn vorliegen. Das Polling erfolgt jedoch nur, falls es sich um einen Knoten handelt, der das Funkmodul zum Energiesparen abschaltet. Das Listing 6.6 zeigt einen Programmcodeausschnitt des Device Manager Prozesses, der für den beschriebenen Ablauf zuständig ist. In den Funktionen *pan_association()* , *scan_for_pancoordinator()* und *poll_data()* werden die entsprechenden Request Funktionen des MAC Core Moduls aufgerufen. Hierfür werden die benötigten Informationen in der entsprechende Struktur gespeichert und der Zeiger auf die Struktur als Parameter übergeben.

Der Device Manager empfängt außerdem die Indication und Confirm Events vom MAC Core Modul. Hierzu wurde ein Event Handler realisiert, der die eingehenden Events behandelt, die Informationen auswertet und gegebenenfalls neue Aktionen auslöst. Beispielsweise wird bei dem Erhalt eines Associate Indication Events, nach der Auswertung der Informationen, die *Associate Response()* Funktion des MAC Core Moduls aufgerufen, um ein Association Response Command Frame zu verschicken.

```
PROCESS_THREAD(device_management_process, ev, data)
{
  static struct etimer et;
  /*---Process Begin---*/
  PROCESS_BEGIN();
...
  etimer_set(&et, PERIOD*CLOCK_SECOND);
  /* main loop */
  while(1){
    PROCESS_YIELD();
...
    /* periodic call */
    if(etimer_expired(&et)) {
        etimer_reset(&et);
...
      /* if not associated to PAN -> Scan + Association */
      if (!associated){
        // if pan coordinator specified
        if (pan_coordinator_specified){
          // pan associate request
          pan_association();
          // polling data if needed
          if((capinfo & (1 << CI_RECEIVERON)) == 0) poll_data();
        }
        // no pan coordinator specified
        else{
          // scan for pan coordinators, if scan not in progress
          if(!scan_processing){
            scan_for_pancoordinator();
          }
        }
      }
      // device associated to pan
      else{
        // polling data if needed
        if((capinfo & (1 << CI_RECEIVERON)) == 0) poll_data();
      } /* if else !associated */
    } /* if periodic call */
  } /* while(1) */
  PROCESS_END();
}
```

Listing 6.6: Auszug aus dem Device Manager Prozess (/contiki-2.7/core/net/ dev_-manage.c)

6.2.3 Anmerkungen zur Implementation

Bei der Umsetzung einiger Funktionen im MAC Core Modul, beispielsweise dem Active Scan, kommen die von Contiki bereitgestellten Callback Timer zum Einsatz. Das ermöglicht das Abwarten einer bestimmten Zeitspanne ohne die Ressourcen des Mikrocontrollers zu blockieren. Nach Ablauf des Timers wird eine vorher definierte Callback-Funktion gestartet. Im Gegensatz zum Device Manager besitzt das MAC Core Modul keinen Prozess, für den spezielle Funktionen zur Abgabe der Ressourcen

des Mikrocontrollers existieren (siehe Abschn. 2.2.2). Daher ist die Verwendung des Callback Timers notwendig.

6.3 Test der Implementierung

Zum Test der Implementierung wird in der Vorbereitung der Testaufbau, die verwendeten Hilfsmittel sowie verschiedene Testkonfigurationen beschrieben. Anschließend werden in der Durchführung die Funktionalitäten bezüglich der Implementierung unter Verwendung der Hilfsmittel getestet. Eine Darstellung und Auswertung der Ergebnisse der Tests bildet den Schluss.

6.3.1 Vorbereitung

Zum Test der Implementation kam die gleiche Hardware zum Einsatz, wie bereits in der Implementierung (siehe Absch. 6.1). Das heißt, zwei Mikrocontroller mit integriertem Funkchip, zwei Breakout-Boards und als Schnittstelle zum PC die USB-to-UART Sticks. Der Aufbau der Hardware zur Entwicklung sowie zum Testen der Implementation wird in Abbildung 6.1 gezeigt.

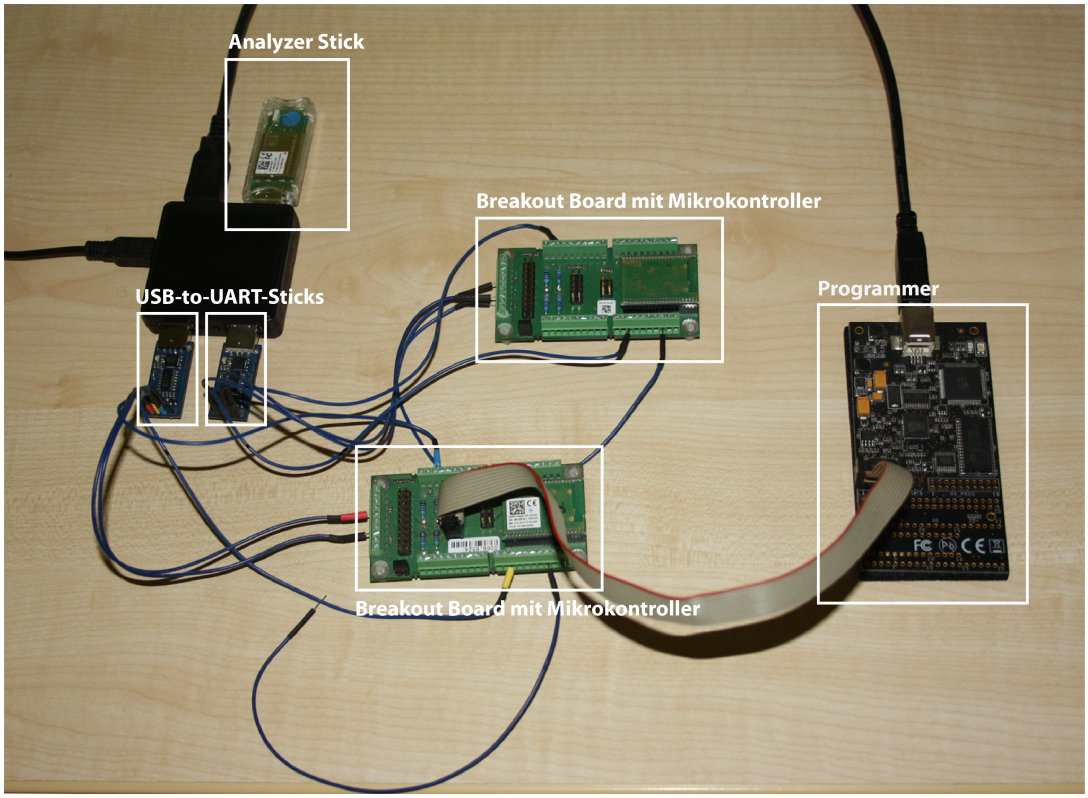

Abbildung 6.1: Testaufbau

Verwendete Hilfsmittel
Da der verwendete Mikrocontroller keinerlei Peripherie besitzt und es somit schwierig ist den Programmablauf nachzuvollziehen, wurden verschiedene Hilfsmittel zum

Test der Implementation genutzt.
Unter Verwendung der UART-Schnittstelle eines Mikrocontrollers, werden Daten in Form von Zeichenketten vom Mikrocontroller zum PC übertragen. Am PC werden die erhalten Zeichenketten auf der Konsole zur Ausgabe gebracht. Diese Zeichenketten werden durch das Einfügen bestimmter *printf*-Anweisungen im Programmcode erzeugt und können über ein Makro aktiviert beziehungsweise deaktiviert werden. Durch das Hinzufügen der *printf*-Anweisungen können die Werte der verwendeten Variablen ausgegeben und der Programmverlauf nachvollzogen beziehungsweise überprüft werden.
Als ein weiteres Mittel zur Überprüfung der Implementation dient die Analyse der über Funk übertragenen Pakete. Dazu kommt ein USB-Funk-Stick zum Einsatz, der einen integrierten Mikrocontroller, einen Funkchip sowie die benötigte Firmware besitzt. Er empfängt die Pakete, die per Funk übertragen wurden und leitet sie an ein spezielles Programm weiter. Dieses bereitet die Daten auf und visualisiert sie im Kontext der verwendeten Protokolle. Verwendet wurde hierfür eine Testversion des Protocol Analyzer Programm der Firma Perytons[2] (siehe Absch. 6.1). Hiermit lassen sich unter anderem die versendeten Pakete in einem Sequenzdiagramm visualisieren.

Testkonfiguration
Zum Test der Implementierung stehen zwei Mikronotroller und somit zwei Knoten zur Verfügung. Ein Knoten wird als PAN-Koordinator und der andere als RFD konfiguriert. Zudem werden die zu testenden Funktionalitäten festgelegt: der Active Scan eines RFDs, mit dem die PAN-Koordinatoren in der Umgebung gefunden werden sollen, der Anmeldeprozess eines RFDs zu einem PAN, der Abmeldeprozess von einem PAN und der direkte/indirekte Datentransfer zwischen dem RFD und dem PAN-Koordinator.
Um zu zeigen, dass die Integration des Mediumzugriffsverfahren in Contiki kompatibel zum IEEE 802.15.4 Standard ist, wird zum Testen außerdem die Implementierung des Atmel Stacks herangezogen. Bei dem Atmel MAC Stack handelt es sich um eine Implementation des IEEE 802.15.4 Standards, der in Abschnitt 3.1 bereits vorgestellt wurde. Es werden die oben beschriebenen Funktionalitäten getestet, indem der Atmel MAC Stack als PAN-Koordinator beziehungsweise RFD und die Contiki Implementation als RFD beziehungsweise PAN-Koordinator konfiguriert werden. Im Atmel MAC Stack sind verschiedene Beispielapplikationen enthalten. Für den Test wurde die *non-beacon-enabled* Applikation mit indirektem Datentransfer genutzt (/MAC_v_2_6_1/Applications/MAC_Examples/App_2_Nobeacon_Indirekt_Traffic).
Damit ergeben sich drei Testkonfigurationen, die in der Tabelle 6.2 zusammengefasst dargestellt werden.

[2]http://www.perytons.com/

Nr.	PAN-Koordinator	RFD	Zu testende Funktionalität
1	**Contiki**	**Contiki**	Active Scan Anmeldeprozess Direkter Datentransfer Indirekter Datentransfer Abmeldeprozess
2	**Atmel Stack**	**Contiki**	Active Scan Anmeldeprozess Direkter Datentransfer Indirekter Datentransfer Abmeldeprozess
3	**Contiki**	**Atmel Stack**	Active Scan Anmeldeprozess Direkter Datentransfer Indirekter Datentransfer Abmeldeprozess

Tabelle 6.2: Testkonfiguration

6.3.2 Durchführung

Für die Durchführung der Tests wurden drei Konfigurationen festgelegt, die in Tabelle 6.2 beschrieben sind. Für jede Konfiguration werden die Funktionalitäten Active Scan, Anmeldeprozess, direkter/indirekter Datentransfer und der Abmeldeprozess getestet. Dabei bestehen die Funktionalitäten meist aus mehreren Schritten. Für den Anmeldeprozess werden beispielsweise folgende Schritte benötigt:

- Versenden und Empfangen des Association Request Command Frames
- Verarbeitung des Association Request Command Frames und Erzeugen des Association Response Command Frames
- Versenden und Empfangen des Association Response Command Frames
- Verarbeitung des Association Response Command Frames

Zur Überprüfung der einzelnen Schritte werden die in Abschnitt 6.3.1 beschrieben Hilfsmittel eingesetzt. Hierbei werden der Programmablauf, die ordnungsgemäße Erzeugung der MAC Frames und die erfolgreiche Übermittlung der MAC Frames geprüft.
Die Abbildung 6.2 zeigt beispielhaft ein Sequenzdiagramm, welches mit Hilfe des Analyzer Sticks und der Protocol Analyzer Software der Firma Perytons zur Testkonfiguration 2 erzeugt wurde. Darauf wird veranschaulicht, welche Frames zwischen

dem PAN-Koordinator und dem RFD ausgetauscht wurden. Bei den Pfeilen ohne Beschriftung handelt es sich um Data Frames zum Test des direkten und indirekten Datentransfers.

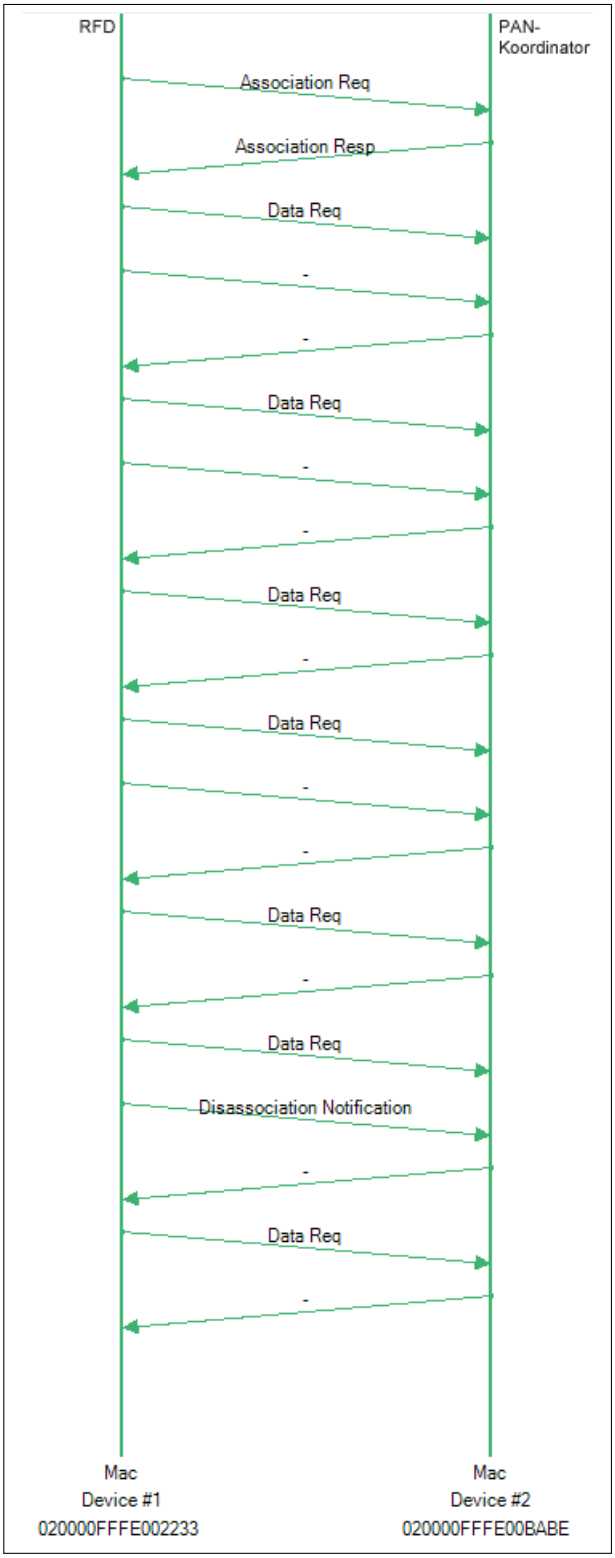

Abbildung 6.2: Sequenzdiagramm für Testkonfiguration 2

6.3.3 Ergebnisse und Auswertung

Im Folgenden werden die Ergebnisse der Tests für die drei Testkonfigurationen (siehe Tab. 6.2) dargestellt und ausgewertet.

Testkonfiguration 1
In dieser Konfiguration wurden beide Knoten mit der Contiki Implementation betrieben. Ein Knoten wurde als PAN-Koordinator und der andere Knoten als RFD konfiguriert. Getestet wurden der Active Scan, der Anmeldeprozess, der direkte/indirekte Datentransfer und der Abmeldeprozess. Beim Datentransfer wurden Data Frames vom Device Manager verschickt. Der Test der Funktionalitäten verlief erfolgreich. Das heißt, die Funktionsfähigkeit ist gegeben.
Da Contiki über einen TCP/IP Stack verfügt, wurde der direkte und indirekte Datentransfer mit dem Versenden von UDP-Paketen getestet. Dabei fiel auf, dass nicht alle Pakete übertragen wurden. Nach weiteren Tests diesbezüglich stellte sich heraus, dass die Pakete erst gar nicht an die MAC-Schicht übergeben wurden und somit auch nicht versendet werden konnten. Eine mögliche Erklärung dafür ist, dass der TCP/IP Stack Contikis zum Verschicken der Pakete ein bestimmtes Protokoll nutzt, das Neighbor Discovery Protocol (NDP). Dabei werden die Adressen der Nachbarknoten in einer Liste verwaltet. Ein Eintrag in der Liste kann jedoch veralten, wenn länger kein Datenaustausch stattfand. Durch den Austausch bestimmter Nachrichten wird die Liste aktuell gehalten. Beim indirekten Datentransfer jedoch, können die Antworten auf eine Nachbarschaftsanfrage nicht unmittelbar übertragen werden, sodass der Eintrag in der Liste als veraltet markiert wird. Wenn dann ein UDP-Paket übertragen werden soll, kann kein entsprechender Eintrag für den Nachbarn in der Liste gefunden werden, sodass das Paket nicht bei der MAC-Schicht ankommt. Dieses Problem muss noch genauer betrachtet werden.
Die Ergebnisse des Tests sind in Tabelle 6.3 zusammengefasst dargestellt. Die Ergebnisse zum direkten und indirekten Datentransfer beziehen sich dabei auf die vom Device Manager verschickten Data Frames.

Testkonfiguration 2
Bei dieser Konfiguration wurde ein Knoten mit dem Atmel Stack betrieben und als PAN-Koordinator konfiguriert. Der andere Knoten wurde mit der Contiki Implementation ausgestattet und fungierte als RFD. Es wurden die gleichen Funktionalitäten, wie auch in der Testkonfiguration 1 getestet (siehe Tab. 6.2).
Der erste Test verlief fehlerhaft. Nach genauerer Analyse fiel auf, dass der Atmel MAC Stack für die versendeten Frames eine Bestätigung verlangt. Eine mögliche Lösung hierfür war die Konfiguration des NETSTACKs von Contiki, sodass automatisch eingehende Frames auf den unteren NETSTACK Schichten automatisch mit einem Acknowledgement bestätigt werden. Dies brachte jedoch nicht den gewünschten Erfolg, da der Mikrocontroller nach Aktivierung der Option ständig abstürzte. Eine weitere Möglichkeit war das Erzeugen der Acknowledgement Frames durch die MAC Schicht. Dies brachte jedoch auch keinen Erfolg. Um die Implementation dennoch zu testen, wurde der Atmel MAC Stack so konfiguriert, dass keine Bestätigung der ausgehenden Frames nötig ist. Daraufhin wurden alle Frames ordnungsgemäß verschickt, sodass sich der RFD nach einem Active Scan beim PAN-Koordinator an-

Funktionalität	PAN-Koordinator (Contiki)	Funktionalität	RFD (Contiki)
Active Scan			
- Empfangen des Beacon Request Frames	✓	- Senden des Beacon Request Frames	✓
- Senden des Beacon Frames	✓	- Empfangen des Beacon Frames	✓
		- Verarbeitung des Beacon Frames (Scan Confirm)	✓
Anmeldeprozess			
- Empfangen des Association Request Frames	✓	- Senden des Association Request Frames	✓
- Verarbeitung des Association Requests (Associate Indication)	✓		
- Senden des Association Response Frames	✓	- Empfangen des Association Response Frames	✓
		- Verarbeitung des Association Response (Associate Confirm)	✓
Direkter Datentransfer			
- Empfangen des Data Frames	✓	- Senden des Data Frames	✓
Indirekter Datentransfer			
- Empfangen des Data Request Frames	✓	- Senden des Data Request Frames	✓
- Senden des Data Frames	✓	- Empfangen des Data Frames	✓
Abmeldeprozess			
- Empfangen des Disassociation Notification Frames	✓	- Senden des Disassociation Notification Frames	✓
- Verarbeitung der Disassociation Notification (Dissassociate Indication)	✓		

Tabelle 6.3: Ergebnisse zur Testkonfiguration 1

melden konnte. Auch der direkte und indirekte Datentransfer sowie das Abmelden vom PAN-Koordinator erfolgten anschließend problemlos. Der Test zeigte, dass bezüglich der Bestätigung von eingehenden Frames in Contiki noch Handlungsbedarf besteht.
Die Ergebnisse des Tests, unter Berücksichtigung der oben genannten Modifikation am Atmel Mac Stack, sind in Tabelle 6.4 zusammengefasst dargestellt.

Testkonfiguration 3
Bei dieser Konfiguration wurde ein Knoten mit dem Atmel Stack betrieben und als RFD konfiguriert. Der andere Knoten wurde mit der Contiki Implementation ausgestattet und fungierte als PAN-Koordinator. Es wurden auch hier wieder die gleichen Funktionalitäten, wie auch in der Testkonfiguration 1 und 2 getestet (siehe Tab. 6.2).
Das Problem mit der Bestätigung der eingehenden Frames durch Contiki bestand auch hier wieder, sodass der Atmel MAC Stack entsprechend des oberen Vorschlags konfiguriert wurde. Das heißt, für ausgehende Frames wird keine Bestätigung benötigt. Der Test zum Active Scan, Anmelden, direktem Datentransfer sowie zum Abmelden verlief anschließend erfolgreich. Beim indirekten Datentransfer traten jedoch Probleme auf. Zwar konnten das Data Request zum Pollen der Daten erfolgreich vom RFD an den PAN-Koordinator übertragen werden und der PAN-Koordinator seine Daten senden, allerdings wurden diese Daten nicht vom RFD erhalten. Einige Tests hierzu ergaben, dass das Funkmodul des RFDs (Atmel MAC Stack) nach dem Senden des Data Request Frames deaktiviert wurde und so keine Daten empfangen werden konnten. Mögliche Gründe hierfür könnten sein, dass die Zeitspanne bis zum Versenden des Data Frames vom PAN-Koordinator an den RFD zu lang ist oder aufgrund der Modifikationen am Atmel MAC STACK bezüglich der Bestätigung von ausgehenden Frames, das Funkmodul direkt nach dem Versenden des Data Request Frames deaktiviert wird. Auch nach einer intensiven Analyse des Quellcodes des Atmel MAC Stacks konnte keine befriedigende Antwort auf dieses Problem gefunden werden. Dieses Problem muss daher noch genauer betrachtet werden.
Die Ergebnisse des Tests unter Berücksichtigung der oben genannten Modifikation am Atmel Mac Stack sind in Tabelle 6.5 zusammengefasst dargestellt.

Funktionalität	PAN-Koordinator (Atmel)	Funktionalität	RFD (Contiki)
Active Scan			
- Empfangen des Beacon Request Frames	✓	- Senden des Beacon Request Frames	✓
- Senden des Beacon Frames	✓	- Empfangen des Beacon Frames	✓
		- Verarbeitung des Beacon Frames (Scan Confirm)	✓
Anmeldeprozess			
- Empfangen des Association Request Frames	✓	- Senden des Association Request Frames	✓
- Verarbeitung des Association Requests (Associate Indication)	✓		
- Senden des Association Response Frames	✓	- Empfangen des Association Response Frames	✓
		- Verarbeitung des Association Response (Associate Confirm)	✓
Direkter Datentransfer			
- Empfangen des Data Frames	✓	- Senden des Data Frames	✓
Indirekter Datentransfer			
- Empfangen des Data Request Frames	✓	- Senden des Data Request Frames	✓
- Senden des Data Frames	✓	- Empfangen des Data Frames	✓
Abmeldeprozess			
- Empfangen des Disassociation Notification Frames	✓	- Senden des Disassociation Notification Frames	✓
- Verarbeitung der Disassociation Notification (Dissassociate Indication)	✓		

Tabelle 6.4: Ergebnisse zur Testkonfiguration 2

Funktionalität	PAN-Koordinator (Contiki)	Funktionalität	RFD (Atmel)
Active Scan			
- Empfangen des Beacon Request Frames	✓	- Senden des Beacon Request Frames	✓
- Senden des Beacon Frames	✓	- Empfangen des Beacon Frames	✓
		- Verarbeitung des Beacon Frames (Scan Confirm)	✓
Anmeldeprozess			
- Empfangen des Association Request Frames	✓	- Senden des Association Request Frames	✓
- Verarbeitung des Association Requests (Associate Indication)	✓		
- Senden des Association Response Frames	✓	- Empfangen des Association Response Frames	✓
		- Verarbeitung des Association Response (Associate Confirm)	✓
Direkter Datentransfer			
- Empfangen des Data Frames	✓	- Senden des Data Frames	✓
Indirekter Datentransfer			
- Empfangen des Data Request Frames	✓	- Senden des Data Request Frames	✓
- Senden des Data Frames	✓	- Empfangen des Data Frames	✗
Abmeldeprozess			
- Empfangen des Disassociation Notification Frames	✓	- Senden des Disassociation Notification Frames	✓
- Verarbeitung der Disassociation Notification (Dissassociate Indication)	✓		

Tabelle 6.5: Ergebnisse zur Testkonfiguration 3

7

Fazit und Ausblick

In diesem abschließenden Kapitel wird zunächst die Thematik der vorliegenden Arbeit zusammenfassend betrachtet und diskutiert. Anschließend werden im Ausblick mögliche Erweiterungen des Konzepts vorgeschlagen.

7.1 Fazit

Das Thema der Arbeit war die Integration eines Mediumzugriffsverfahren in Contiki mit Kompatibilität zum IEEE 802.15.4 Standard. Dazu wurden wichtige Aspekte des IEEE 802.15.4 Standard in Bezug auf die MAC Schicht in Kapitel 2 beschrieben. Neben der Unterscheidung verschiedener Knotentypen in FFDs und RFDs, wurden die Netzwerktopologien, die Übertragungsverfahren, die unterschiedlichen Typen der MAC Frames und die MAC Services erläutert. Des Weiteren wurden die Grundlagen zum Betriebssystem Contiki beschrieben. Dabei wurde insbesondere auf den Netzwerkstack eingegangen, der die unteren Schichten bezüglich des ISO/OSI Referenzmodells in Contiki umsetzt.
Anschließend folgte in Kapitel 3 eine Vorstellung von zwei aktuellen Implementierungen des IEEE 802.15.4 Standards, den Atmel MAC Stack und den TIMAC.
In Kapitel 4 wurden zunächst in einer Problemdefinition die wichtigen Kriterien des IEEE 802.15.4 Standards zur Integration eines Mediumzugriffsverfahren in Contiki herausgestellt. Danach wurde mit Bezug auf die Kriterien, eine Quellcodeanalyse Contikis durchgeführt, um die bestehenden Strukturen Contikis genauer zu untersuchen. Dabei wurde überprüft, inwieweit die Kriterien zum IEEE 802.15.4 Standard bereits von Contiki umgesetzt sind. Zudem wurden genauere Informationen über den internen Programmablauf beim Versenden und Empfangen eines Paketes mittels des Netzwerkstacks erhalten. Hierbei stellte sich heraus, dass die Funktionalität zum Versenden und Empfangen standardkonformer MAC Frames durch eine Implementierung des NETSTACK Framer Modul gegeben ist. Allerdings können nur Data

Frames erzeugt und geparst werden. Weiterhin zeigte sich, dass für den Zugriff auf einen Übertragungskanal mittels des CSMA/CA Protokolls eine Implementierung des NETSTACK MAC Moduls besteht. Andere Kriterien des IEEE 802.15.4 Standards wurden in Contiki bisher nicht umgesetzt. Zu diesen Kriterien zählen insbesondere die Unterscheidung der Knoten in FFD und RFD sowie die Umsetzung der MAC Services, die im Wesentlichen zur Verwaltung eines PANs benötigt werden. Anhand der Ergebnisse aus der Analyse wurden abschließend eine Anforderungsspezifikation an ein mögliches Konzept zur Integration der fehlenden Funktionalität bezüglich des IEEE 802.15.4 Standards erarbeitet.
Auf Basis dieser Vorarbeiten erfolgte in Kapitel 5 die Entwicklung eines Konzepts. Es wurde gezeigt, wie durch Anpassungen im Contiki die Definition der Knotentypen als FFD beziehungsweise RFD sowie die Unterstützung der verschiedenen MAC Frames ermöglicht werden soll. Außerdem wurde ein Grundkonzept zur Integration der MAC Services in Contiki entworfen. Dazu wurden zwei Module, das MAC Core Modul und der Device Manager, sowie die Interaktion zwischen diesen konzipiert. Das MAC Core Modul basiert auf der CSMA/CA Implementierung des NETSTACK MAC Moduls von Contiki. Die Implementierung wurde hierzu um die Funktionalität der zu integrierenden MAC Services erweitert. Zudem wurden die Schnittstellen der Dienste dem NETSTACK MAC Modul hinzugefügt. Um nun die hinzugekommene Funktionalität zu nutzen, wurde der Device Manager entworfen. Dieser besteht im Kern aus einem Prozess, der bei der Initialisierung Contikis gestartet wird und hauptsächlich zur Verwaltung des PANs dient. Dazu werden die entsprechenden Request beziehungsweise Response Funktionen der MAC Services im MAC Core Modul aufgerufen. Das MAC Core Modul wiederum interagiert mit dem Device Manager durch die Generierung von Indication oder Confirm Events an den Prozess. Die eingehenden Events werden anschließend durch den Event Handler des Device Managers verarbeitet. Weiterhin wurde im Konzept die Integration der MAC Services, welche speziell für das *non-beacon-enabled* Übertragungsverfahren erstellt. Das Grundkonzept zur Integration der MAC Services wurde im Hinblick auf Erweiterbarkeit entworfen, so dass weitere Dienste hinzugefügt werden können.
Im Kapitel 6 wurde die Realisierung des Konzepts sowie der Test der Implementierung beschrieben. Hierzu wurden zu Beginn, die für die Entwicklung und der Tests verwendete Hard- und Software aufgelistet. Danach wurden die grundsätzlichen Aspekte der Implementierung erläutert und mit Programmcodebeispielen unterlegt. Anschließend folgte der Test der Implementierung. Hierzu wurde zunächst der Testaufbau, die verwendeten Hilfsmittel und drei Testkonfiguration beschrieben. In der ersten Konfiguration wurde die Contiki Implementierung getestet, indem ein Knoten als PAN-Koordinator und ein weiterer Knoten als RFD fungierte. Der Test verlief bis auf eine Einschränkung bezüglich des Versendens von UDP-Paketen mit Hilfe des TCP/IP Stacks Contikis erfolgreich. Für die zweite und dritte Testkonfiguration wurde der Atmel MAC Stack, der in Abschnitt 3.1 vorgestellt wurde, für den Test hinzugezogen. Bei beiden Tests traten Probleme in Bezug auf die automatische Bestätigung der empfangen Frames in Contiki auf. Durch eine Konfiguration des Atmel MAC Stacks konnte jedoch gezeigt werden, dass bis auf diese Einschränkung die Funktionsfähigkeit gegeben ist. Die gesamten Ergebnisse der Tests und deren Auswertung sind in Abschnitt 6.3.3 aufgeführt.

Zusammenfassend kann festgestellt werden, dass wesentliche Aspekte des IEEE 802.15.4 Standards bezüglich der MAC Schicht in Contiki bisher nicht umgesetzt waren. Anhand des entwickelten Konzepts konnte die Integration eines zum Standard kompatiblen Mediumzugriffsverfahren in das Betriebssystem Contiki erreicht werden. Dies wurde durch die Umsetzung und das Testen der Implementierung mit dem Atmel MAC Stack gezeigt.

7.2 Ausblick

Weiterführend könnten in Bezug auf die Implementierung noch einige Punkte bearbeitet werden. Durch den Test der Implementierung mit dem Atmel MAC Stack sind einzelne Probleme erkannt worden. Diese gilt es näher zu untersuchen und zu beheben. Zudem benötigen bestimmte Dienste der MAC Schicht die Funktionalität der physikalischen Schicht Contikis, genauer die des NETSTACK Radio Moduls. Um diese Funktionalitäten der MAC-Schicht bereitzustellen, wurde eine Anpassung der Schnittstelle des NETSTACK Radio Moduls im Konzept angedacht. Inwiefern die Funktionalitäten tatsächlich durch die bestehenden Implementierungen der verschiedenen Funkmodule umgesetzt sind, stand nicht im Fokus der Arbeit und sollte noch näher untersucht werden.
Weiterhin besteht die Möglichkeit zur Erweiterung der Implementierung um zusätzliche MAC Services, die beispielsweise für das *beacon-enabled* Übertragungsverfahren benötigt werden.
Darüber hinaus wurden die Aspekte zur Sicherheit und Verschlüsselung bezüglich des IEEE 802.15.4 Standards in der Arbeit nicht näher betrachtet. Diese könnten in einer weiterführenden Arbeit näher untersucht werden.

A

Anhang

Abkürzungsverzeichnis

AES Advanced Encryption Standard

CAP Contention Access Period

CCA Clear Channel Assessment

CFP Contention Free Period

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CSMA/CA Carrier Sense Multiple Access / Collision Avoidance

FCF Frame Control Field

FCS Frame Check Sequence

FFD Full Function Device

GCC Gnu Compiler Collection

GTS Guaranteed Time Slot

HAL Hardware Abstraction Layer

ICMP Internet Control Message Protocol

IEEE Institute of Electrical and Electronics Engineers

IFS Interframe Spacing

IP Internet Protocol

IPv6 Internet Protocol version 6

ISO International Organization for Standardization

LLC Logical Link Control

LIFS Long Interframe Spacing

LPP Low Power Probing

LR-WPAN Low-Rate Wireless Personal Area Network

MAC Media Access Control

MCPS MAC Common Part Sublayer

MCPS-SAP MAC Common Part Sublayer Service Access Point

MCL MAC Core Layer

MFR MAC Footer

MHR MAC Header

MLME MAC Sublayer Management Entity

MLME-SAP MAC Sublayer Management Entity Service Access Point

MPDU MAC Protocol Data Unit

MSDU MAC Service Data Unit

MTU Maximum Transmission Unit

NDP Neighbor Discovery Protocol

OSAL Operating System Abstraction Layer

OSI Open Systems Interconnection

PAL Platform Abstraction Layer

PAN Personal Area Network

PAN-ID Personal Area Network Identifier

PIB MAC PAN Information Base

RDC Radio Duty Cycle

RFD Reduced Function Device

RPL Routing Protocol for Low power and Lossy Networks

RSSI Received Signal Strength Indication

SAL Security Abstraction Layer

SIFS Short Interframe Spacing

SPI Serial Peripheral Interface

STB Security Toolbox

TAL Tranceiver Abtraction Layer

TCP Transmission Control Protocol

UDP User Datagram Protocol

uIP micro IP, open source TCP/IP-Stack

WSN Wireless Sensor Network

Abbildungsverzeichnis

Tabellenverzeichnis

Programmcodeverzeichnis

Literaturverzeichnis

[ASSC01] Akyildiz, I.F. ; Su*, W. ; Sankarasubramaniam, Y. ; Cayirci, E.: *Wireless sensor networks: a survey.* december 2001

[Atm12] Atmel Corporation: *Atmel AVR2025: IEEE 802.15.4 MAC Software Package - User Guide.* 2012. – Online verfügbar unter: `http://www.atmel.com/images/doc8412.pdf`; besucht am 20.06.2014

[Cona] *6LoWPAN implementation.* – Online verfügbar unter: `http://dak664.github.io/contiki-doxygen/a01680.html#_details`; besucht am 10.05.2014

[Conb] *The Contiki netstack.* – Online verfügbar unter: `https://eva.fing.edu.uy/pluginfile.php/76857/mod_folder/content/0/The_Contiki_Netstack.pdf`; besucht am 10.05.2014

[Conc] *Ofizielle Contiki Website.* – Online verfügbar unter: `http://www.contiki-os.org/#why`; besucht am 10.06.2014

[Cond] *Radio duty cycling.* – Online verfügbar unter: `https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling`; besucht am 10.05.2014

[Cone] *The Rime communication stack.* – Online verfügbar unter: `http://dak664.github.io/contiki-doxygen/a01678.html#_details`; besucht am 10.05.2014

[DGV04] Dunkels, Adam ; Grönvall, Björn ; Voigt, Thiemo: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors / Swedish Institute of Computer Science. Version: nov 2004. `http://dunkels.com/adam/dunkels04contiki.pdf`. 2004. – Forschungsbericht

[DH98] Deering, S. ; Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. Version: December 1998. `http://tools.ietf.org/rfc/rfc2460.txt`. 1998 (2460). – RFC. – 24 S.

[DSV05] Dunkels, Adam ; Schmidt, Oliver ; Voigt, Thiemo: Using Protothreads for Sensor Node Programming / Swedish Institute of Computer Science. Version: jun 2005. `http://dunkels.com/adam/dunkels05using.pdf`. 2005. – Forschungsbericht

[DSVA06] Dunkels, Adam ; Schmidt, Oliver ; Voigt, Thiemo ; Ali, Muneeb: Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems / Swedish Institute of Computer Science. Version: nov 2006. `http://dunkels.com/adam/dunkels06protothreads.pdf`. 2006. – Forschungsbericht

[DTV09] Dwivedi, A. K. ; Tiwari, M. K. ; Vyas, O. P.: *Operating Systems for Tiny Networked Sensors: A Survey.* 2009. – Online verfügbar unter: `http://academypublisher.com/ijrte/vol01/no02/ijrte0102152157.pdf`; besucht am 10.05.2014

[EGN07] Engstrom, J. C. ; Gray, Chase ; Nelakuditi, Srihari: *Clear Channel Assessment in Wireless Sensor Networks.* 2007. – Online verfügbar unter: `http://www.cse.sc.edu/files/reu/2007papers/EngstromPaper.pdf`; besucht am 17.05.2014

[FK11] Farooq, Muhammad O. ; Kunz, Thomas: Operating Systems for Wireless Sensor Networks: A Survey. In: *ISSN 1424-8220, DOI 10.3390/s110605900* (2011), nov

[GWCB10] Gutiérrez, José A. ; Winkel, Ludwig ; Callaway, Edgar H. ; Barrett, Raymond L.: *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors With IEEE 802.15.4.* Standards Information Network IEEE Press, 2010. – ISBN 978–0–7381–6285–0

[IEE11] IEEE ; Institute of Electrical and Electronics Engineers, Inc. (Hrsg.): *IEEE. 802.15.4-2011 - IEEE Standard for Local and metropolitan area networks Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).* : Institute of Electrical and Electronics Engineers, Inc., September 2011

[KAT05] Koubâa, Anis ; Alves, Mário ; Tovar, Eduardo: Ieee 802.15.4 for wireless sensor networks: A technical overview / IPP-HURRAY! Polytechnic Institute of Porto (ISEP-IPP). 2005 (TR-050702). – Forschungsbericht

[LSC10] Lauwens, Ben ; Scheers, Bart ; Capelle, Antoine V.: *Performance analysis of unslotted CSMA/CA in wireless networks.* 2010

[TI] Texas Instruments, Inc.: *IEEE802.15.4 Medium Access control (MAC) software stack.* `http://www.ti.com/tool/timac#descriptionArea`, . – besucht am 20.06.2014

[TIM11] Texas Instruments, Inc.: *MAC Sample Application Software Design.* 2011. – Document Number: SWRA200

[VD10] Vasseur, Jean-Phillipe ; Dunkles, Adam: *Interconnecting Smart Objects with IP: The Next Internet.* Morgan Kaufmann Publishers, 2010. – ISBN 0123751659 9780123751652