HOCHSCHULE FÜR
TECHNIK UND WIRTSCHAFT
DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Untersuchung von Stromspar-Mechanismen in Sensornetzen

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Fachbereich Angewandte Informationstechnologien
Intelligente Informations- und Kommunikationstechnologien

eingereicht von:

Alexander Gräb

Betreuer und Erstgutachter: Herr Prof. Dr.-Ing. Jörg Vogt
Zweitgutachter: Herr Prof. Dr. Jens Schönthier

Themenübergabe: 24.05.2017 Eingereicht: 01.11.2017

Danksagung

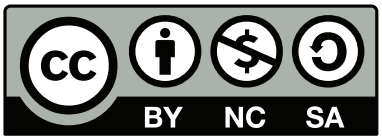
An dieser Stelle möchte ich mich bei all jenen bedanken, die zum Gelingen dieser Masterarbeit beigetragen haben.

Vor allem danke ich Herrn Prof. Dr.-Ing. Jörg Vogt, der diese Masterarbeit als Fortsetzung des Forschungsseminars „Sensornetze“ möglich gemacht hat. Das Themengebiet hat mir stets viel Freude bereitet. Außerdem danke ich ihm für seine fachliche Unterstützung und die Bereitstellung von Hardware und Messgeräten, ohne welche der praktische Teil dieser Arbeiten nicht möglich gewesen wäre.

Ich danke Herrn Prof. Dr. Jens Schönthier, der sich freundlicherweise als Zweitgutachter bereit erklärt hat.

Mein Dank gilt auch Marie Eckert, Manuel Würpel und Johannes Schaffrath, die einen Teil ihrer wertvollen Zeit für die Korrektur meiner Arbeit geopfert haben. Dank ihnen ist mir so mancher Fehler erspart geblieben.

Besonderen Dank gilt meinen Eltern und Christin Ziska, die mich im Laufe meines Studiums begleitet und unterstützt haben.



Inhalt

Abkürzungsverzeichnis

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

6TiSCH IPv6 over the TSCH mode of IEEE 802.15.4e

ACK Acknowledgement

API Application Programming Interface

APP Application

BPSK Binary Phase Shift Keying

CCA Clear Channel Assessment

CCR Channel Check Rate

CSS Chirp Spread Spectrum

CoAP Constrained Application Protocol

CSL Coordinated Sampled Listening

CSMA Carrier Sense Multiple Access

CSMA-CA Carrier Sense Multiple Access/Collision Avoidance

F Farad

FFD Full Function Device

GTS Guaranteed Time Slot

HART Highway Addressable Remote Transducer

IEC International Electrotechnical Commission

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

IPC Interprocess Communication

IPv6 Internet Protocol Version 6

ISM Industrial, Scientific and Medical Band

ITU International Telecommunication Union

LLC Logical Link Control

LPL Low Power Listening

LPP Low Power Probing

LR-WPAN Low-Rate Wireless Personal Area Network

LwMAC Lightweight MAC

MAC Media Access Control

MPPT Maximum Power Point Tracking

NFC Near Field Communication

O-QPSK Offset Quadrature Phase Shift Keying

PAN Personal Area Network

PDR Packet Delivery Rate

PHY Physical Layer

PPDU PHY Protocol Data Unit

ppm parts per million

PSSS Parallel Sequence Spread Spectrum

PRR Packet Reception Rate

RDC Radio Duty Cycling

RFC Requests for Comments

RFD Reduced Function Device

RIT Receiver Initiated Transmissions

RPL Routing Protocol for Low power and Lossy Networks

RSSI Received Signal Strength Indicator

RTT Real Time Timer

SIG Special Interest Group

SoC System-on-a-Chip

SRD Short Range Devices

TCP Transmission Control Protocol

TDMA Time Division Multiple Access

TSCH Time-Slotted Channel Hopping

UDP User Datagram Protocol

WSN Wireless Sensor Network

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

1.1. Sensornetze

Als Sensornetze, im weiteren Verlauf dieser Arbeit Wireless Sensor Networks (WSNs) genannt, bezeichnet man ein Netzwerk aus Sensoren und Aktoren[1], welche drahtlos über Funk kommunizieren (vgl. [34, S. 7]). Natürlich gibt es auch noch andere Möglichkeiten der drahtlosen Kommunikation, wie Infrarot, Induktivität, Ultraschall usw. Jedoch bietet die Funkübertragung die meisten Vorteile, da kein Sichtkontakt erforderlich ist und sich mit verhältnismäßig wenig Energie große Reichweiten überbrücken lassen (vgl. [34, S. 10]). Die Vorteile einer drahtlosen Kommunikation liegen vorallem in der Einsparung der Kosten für die Verkabelung, welche teilweise enorm ausfallen kann, sowie in der flexibleren Wahl des Standortes der Sensoren und Aktoren. Ein Endschalter[2] bspw. kostet nur wenige Euro, wohingegen die Kosten einer Verkabelung um mehr als den Faktor 100 überwiegen kann (vgl. [34, S. 8]). Verwendet man bspw. zum Schalten des elektrischen Lichts Sensoren in Form von Schaltern, welche lediglich an die Wand geschraubt oder geklebt werden, statt der herkömmlichen Unterputzschalter, spart dies nicht nur Material und Arbeitszeit für das Verlegen der Kabel, sondern es lassen sich jederzeit kostengünstig weitere Schalter hinzufügen oder die Platzierung vorhandener Schalter lässt sich mit sehr geringem Aufwand ändern.
Aber es gibt auch Einsatzgebiete, bei denen eine Verkabelung einfach nicht möglich oder sehr schwierig ist, z. B. bei großflächigen Messungen auf dem Meer (vgl. [39, S. 1]).
Auf der Kehrseite jedoch bringen WSNs die typischen Herausforderungen der drahtlosen Datenübertragung mit sich, wie die Datensicherheit, Authentifizierung, Auslastung der Funkfrequenzen und der schwankenden Qualität des Übertragungsmediums. Um die Vorteile drahtgebundener und drahtloser Sensornetze zu kombinieren, werden drahtgebundene Sensornetze oft an geeineten Stellen mit WSNs erweitert.

1.1.1. Anwendungsgebiete von WSNs

Die Anwendungsgebiete von WSNs sind vielfältig. In diesem Kapitel werden nur einige davon genannt.

WSNs finden Anwendung in der Überwachung des Zustandes von Bauwerken wie Brücken (Structural Health Monitoring, vgl. [23, S. 17 ff.]). Trotz regelmäßiger Kontrollen kommt es immer wieder dazu, dass Brücken einstürzen. Risse und andere Beschädigungen können durch permanent installierte Sensoren, welche bspw. die Schwingung des Bauwer-

[1] Obwohl ein solches Netzwerk nicht nur aus Sensoren bestehen kann, hat sich der Einfachheit halber die kurze Bezeichnung Wireless Sensor Network, statt z. B. „Wireless Sensor and Actuator Network“, durchgesetzt.

[2] Dabei handelt es sich um einen Sensor, welcher erkennt, ob ein bewegter Gegenstand eine bestimmte Position erreicht hat. Z. B., ob das Rolltor einer Tiefgarage den Zustand „Geschlossen“ erreicht hat (vgl. [65]).

kes messen, frühzeitig erkannt werden, um Maßnahmen zu ergreifen, bevor es zur Katastrophe kommt.

Sensoren (z. B. induktive Schleifen), welche den Verkehrsfluss in Städten messen, können helfen Staus zu reduzieren. Fahrern können Empfehlungen zu einer angemessenen Fahrweise oder für alternative Routen gegeben werden (vgl. [23, S. 26]).

Zur Überwachung der Gesundheit von Patienten können Sensoren Parameter wie Puls, Blutdruck, Sauerstoffsättigung des Blutes usw. messen (vgl. [23, S. 30 ff.]). Durch das Übertragen dieser Daten an eine Station und deren Aufzeichnung, bleiben dem Patienten Arztbesuche erspart. Alte Menschen können, statt im Pflegeheim untergebracht zu sein, in ihrer vertrauten Umgebung bleiben. Sensoren alamieren das Pflegepersonal oder den Arzt, wenn sich der Gesundheitszustand verschlechtert (vgl. [34, S. 22 ff.]). Auch das Überwachen von Einsatzkräften bei der Arbeit ist eine Aufgabe für WSNs.

Wie schon in Kap. 1.1 erwähnt, werden WSNs verwendet, um die Installationskosten von Sensoren und Aktoren durch Einsparung der Verkabelung zu senken. Dies spielt insbesondere in der Industrie, wegen den dort herrschenden Anforderungen an die Verkabelung und den damit einhergehenden hohen Kostendruck, eine wichtige Rolle (vgl. [34, S. 17]).

In intelligenten Stromnetzen (Smart Grids), sind Energieerzeuger, -speicher und -verbraucher miteinander vernetzt, mit dem Ziel die Auslastung des Energienetzes zu verbessern, indem Lastspitzen vermieden werden (vgl. [34, S. 18 ff.]). Für den Verbraucher hat dies außerdem den Vorteil der Kosteneinsparung, indem „smarte“ Geräte zu den Zeiten arbeiten, an denen der Strom am günstigsten ist. Durch den immer weiter ansteigenden Energiebedarf, werden Smart Grids zunehmend an Relevanz gewinnen.

Auch das Thema Heimautomatisierung (Smart Home) erfreut sich immer größerer Beliebtheit (vgl. [34, S. 19 ff.]). Dadurch lässt sich der Komfort und die Sicherheit zu Hause steigern, sowie Energie sparen, in dem diese effizienter eingesetzt wird. Lichtschalter und Dimmer können durch Sensoren ersetzt werden, wie bereits in Kap. 1.1 erwähnt. Die Aktoren in diesem Zusammenhang sind die Lichtsteuerungen. Sensoren in jedem Raum überwachen die Temperatur und Heizregler (Aktoren) steuern die Heizung, damit sich eine Solltemperatur einstellt. Sensoren an den Fenstern erkennen, wenn diese geöffnet sind und die Heizung im entsprechenden Raum kann dann heruntergeregelt werden. Bewegungsmelder oder Präsenzmelder erkennen, ob jemand zu Hause ist und können die Heizung veranlassen, bei Abwesenheit herunterzuregeln. Oder diese Sensoren werden genutzt, um Einbrüche zu erkennen. Rauchmelder lösen Alarm aus und rufen automatisch Hilfe.

Bei der Precision Agriculture geht es darum, Felder gezielter zu bestellen (vgl. [34, S. 25 ff.]). Dadurch lässt sich die Quantität sowie die Qualität der Ernte bei gleichzeitiger Senkung der Kosten (durch Einsparung von Betriebsmitteln, menschlichen Einsatz etc.) senken. Voraussetzung dafür ist, dass die Qualität der Böden bekannt ist. Dafür kommt ein Netzwerk aus einer Vielzahl an Sensoren zum Einsatz, welche Informationen sammeln. Informationen über den Feuchtegehalt, der Nitrogenkonzentration und den pH-Wert des Bodens. Weitere Sensoren erfassen den Regenfall, die Temperatur, die Luftfeuchtigkeit und den Luftdruck. All diese Informationen nutzen den Farmern, effizientere Landwirtschaft zu betreiben.

1.2. Motivation und Zielstellung

Um den in Kap. 1.1 beschriebenen Vorteil der Kostenersparnis aufrecht zu erhalten, müssen die Knoten, also die Geräte in einem WSN, so wartungsarm wie möglich bleiben. Diese Masterarbeit beschäftigt sich mit dem Aspekt des Energiesparens in WSNs. Ein Knoten sollte mit einer Batterie so lange wie möglich betrieben werden können, d. h. je nach Anwendungsfall, Tage, Monate, Jahre, bis hin zu Jahrzehnten. Im besten Fall ist der Betrieb gar völlig ohne Batterie möglich, indem die Energie aus der Umgebung bezogen wird (Energy Harvesting, siehe Kap. 6). In der Praxis kommen oft erschwerende Randbedingungen hinzu, die das Energiesparen umso wichtiger machen. Bspw., dass ein Knoten nur sehr wenig Platz einnehmen darf. Damit ist auch der Platz eingeschränkt, den eine Batterie oder Solarzelle einnehmen darf.

Die weitere Arbeit ist wie folgt aufgebaut: In Kap. 2 werden die Grundlagen erläutert, welche zum Verständnis dieser Arbeit notwendig sind. In Kap. 3 wird eine Auswahl energiesparender Funkstandards betrachtet und es wird analysiert, mit welchen Strategien diese einen energiesparenden Betrieb erreichen. Anschließend wird in Kap. 4 eine Reihe von sog. *MAC-Protokollen* untersucht, welche mittels Radio Duty Cycling (RDC) Energie sparen. Das Besondere an diesen MAC-Protokollen ist, dass die Knoten im WSN praktisch ständig erreichbar sind, obwohl sie sich die meiste Zeit in einem Energiesparmodus befinden und dass sie dadurch für Multi-Hop-Netzwerke geeignet sind, in welchen selbst Router Energie sparen können. Die Möglichkeit der Implementation eines der effizientesten MAC-Protokolle aus Kap. 4 in das freien Open Source Betriebssystem RIOT für WSN-Knoten, wird in Kap. 5 untersucht. In Kap. 6 wird auf Energy Harvesting als umweltfreundliche und wartungsarme Alternative zur Batterie eingegangen. Die Arbeit schließt in Kap. 7 mit der Zusammenfassung und dem Ausblick ab.

2. Grundlagen

2.1. Die ISM- und SRD-Frequenzbänder

Die Industrial, Scientific and Medical Band (ISM)- und Short Range Devices (SRD)-Frequenzbänder beschreiben eine Reihe von Frequenzbereichen mit Allgemeinzuteilung. Sie sind sehr beliebt, da sie ohne gesonderte Lizenz verwendet werden dürfen und dadurch keine Lizenzkosten anfallen. Jedoch unterliegen die ISM- und SRD-Frequenzbänder länderspezifischen Regularien, welche die Frequenzbänder vorallem in der Reichweite einschränken. In Deutschland bspw. ist für das 2,4-GHz-ISM-Frequenzband (2,4 - 2,4835 GHz, im Folgenden 2,4-GHz-Band genannt) bei Verwendung eines Isotropstrahlers eine maximale Sendeleistung von 100 mW für WLAN und 10 mW für sonstige Anwendungen erlaubt (vgl. [14]). Mit dieser Sendeleistung und einem Empfänger mit einer Empfindlichkeit von -90 dBm, kann eine Reichweite bis etwa 300 m im Freien[1] erzielt werden. Unter realistischeren Bedingungen, ohne Sichtkontakt und mit vielen Hindernissen (z. B. in Gebäuden) fällt die Reichweite noch deutlich geringer aus. Für WSNs wird jedoch oft ganz bewusst eine noch niedrigere Sendeleistung gewählt, da sich so erheblich Energie sparen lässt (vgl. [23, S. 12]). Das 2,4-GHz-Band spielt eine wichtige Rolle, weil es in (fast) allen Teilen der Welt zugelassen ist. Dies senkt die Fertigungskosten, da in großen Stückzahlen produziert werden kann und vereinfacht die Zulieferketten, da weniger Acht auf die Unterschiede der Regularien in den Zielländern genommen werden muss.

Allerdings bringen diese Frequenzbänder auch einen großen Nachteil mit sich: Da sie so beliebt sind und von vielen Technologien verwendet werden (die bekanntesten davon sind wohl WLAN und Bluetooth, welche das 2,4-GHz-Band nutzen), sind sie potentiell stark ausgelastet. Technologien, welche ISM- bzw. SRD-Bänder nutzen, müssen also gute Koexistenz-Eigenschaften aufweisen. Um diesen Nachteil auszugleichen, verwenden viele Technologien Channel-Hopping (siehe Kap. 2.2).

2.2. Channel-Hopping

Channel-Hopping bezeichnet das fortlaufende Wechseln des Kanals, d. h. der Funkfrequenz. Dies erhöht die Zuverlässigkeit der Datenübertragung, da Störungen auf einzelnen Kanälen ausgewichen wird. Channel-Hopping kann auch helfen Energie zu sparen. Auf einem stark ausgelasteten Kanal kann häufig nicht gesendet werden. Viele Technologien wiederholen dann den Sendevorgang, bis dieser erfolgreich war (oder eine maximale Anzahl an Versuchen unternommen wurde). Jeder Sendeversuch kostet Energie. Da selten alle Kanäle stark ausgelastet sind, wird die Anzahl der Sendeversuche durch das Channel-Hopping reduziert. Außerdem erschwert das Channel-Hopping einem Angreifer das Mithören der Datenübertragung, was wiederum die Sicherheit erhöht.

[1] Im Freien bedeutet bei Sichtkontakt und ohne größere Störfaktoren, wie z. B. Reflektionen der Funkwellen an nahen Objekten.

2.3. Energiesparmodus

Ein Energiesparmodus im Sinne dieser Arbeit bezeichnet alle Betriebsmodi, welche ein Knoten nutzen kann, um Energie zu sparen. Während ein Energiesparmodus genutzt wird, kann der Knoten weder Daten senden noch empfangen.

Eine Form eines Energiesparmodus besteht oft darin, die Funkeinheit zu deaktivieren, da diese meist nicht unerheblich zum Energieverbrauch beiträgt. Auch Schlafmodi zählen dazu, bei denen nicht nur die Funkeinheit deaktiviert wird, sondern auch weite Teile des Systems abgeschaltet werden. Dies soll am Beispiel des ATmega 128RFA1 von Atmel – ein Bauteil, welches einen Mikrocontroller und eine IEEE 802.15.4-Funkeinheit in einem Chip integriert (ein sog. System-on-a-Chip (SoC)) – verdeutlicht werden. Abb. 2.1 zeigt den durchschnittlichen Stromverbrauch des Chips bei einem Prozessortakt von 16 MHz in verschiedenen Betriebszuständen und bei unterschiedlichen Betriebsspannungen. Man erkennt, dass der Chip bei eingeschalteter Funkeinheit, also, wenn er in Empfangsbereitschaft ist (RX_ON) und beim Senden (BUSY_TX) mit Abstand am meisten Strom verbraucht. Ist die Funkeinheit abgeschaltet (TRX_OFF), verbraucht der Chip bereits weniger als ein Drittel des Stromes. Im Schlafmodus (SLEEP) sinkt der Stromverbauch weiter. Die höchste Stufe der Energiesparmodi des ATmega 128RFA1 stellt der Deep Sleep-Modus dar, bei welchem der Stromverbrauch nur noch sehr geringe 250 nA beträgt. Im Deep Sleep-Modus wird sogar der Prozessor und dessen Taktgeber angehalten. Nur ein sekundärer Taktgeber bleibt aktiv, welcher einen Timer steuert, damit der ATmega 128RFA1 den Deep Sleep-Modus nach einer vorgegebenen Zeit wieder verlassen kann.

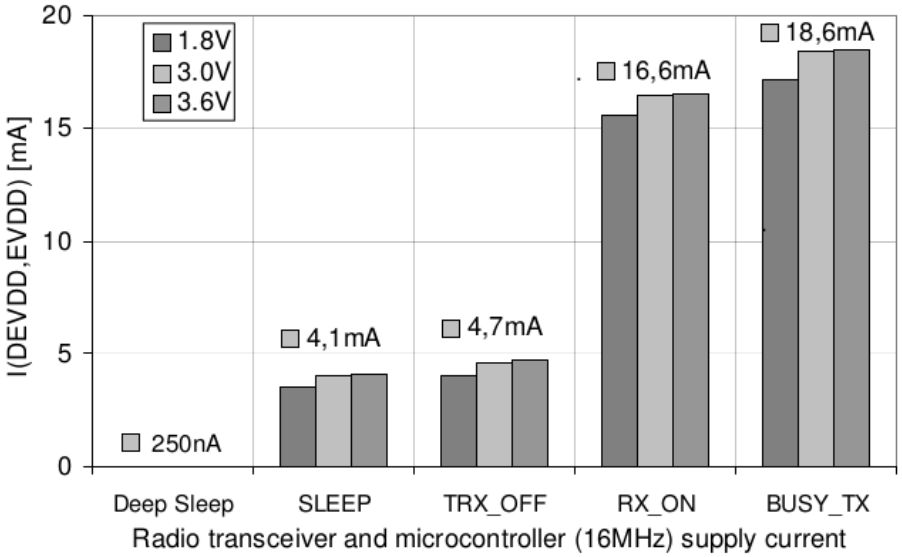


Abbildung 2.1.: Stromverbrauch des ATmega 128RFA1 bei verschiedenen Betriebszuständen

Quelle: [11, S. 4]

2.4. Netzwerktopologien

Dieses Kapitel beschreibt die typischen Netzwerktopologien eines WSNs.

2.4.1. Punkt-zu-Punkt

Die Punkt-zu-Punkt-Verbindung (Abb. 2.2) ist die einfachste Form einer Netzwerktopologie. Sie umfasst genau zwei Knoten. Davon kann bspw. einer ein Aktor sein und der andere ein Sensor. Der Sensor steuert dann in Abhängigheit von seinen Sensorwerten den Aktor an. Auch kann ein Knoten eine Basisstation sein und der andere ein Aktor oder Sensor. Die Basisstation steuert den Aktor nach einem bestimmten Regelwerk an oder sammelt die Daten, welche von einem Sensor geliefert werden. Diese Netzwerktopologie bildet oft die Basis für komplexere Netzwerktopologien.

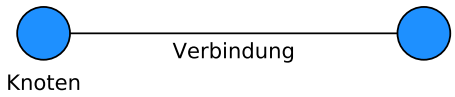


Abbildung 2.2.: Beispiel einer WSN-Punkt-zu-Punkt-Topologie
Quelle: Eigene Darstellung

2.4.2. Stern

In einer Stern-Topologie (Abb. 2.3) kommunizieren alle Knoten nur mit einem zentralen Knoten im Netzwerk. Der zentrale Knoten ist oft ein *Border-Router* (auch Edge-Router oder Gateway genannt), welcher den Übergang vom drahtlosen WSN in ein anderes, oft drahtgebundenes, Netzwerk darstellt. Der Border-Router nimmt dabei die Konvertierung zwischen den unterschiedlichen Protokollen und Datenformaten der Netzwerke vor. In dem anderen Netzwerk kann sich bspw. ein Computer befinden, welcher die Daten der Sensor-Knoten entgegennimmt, verarbeitet und Steueranweisungen an die Aktor-Knoten im WSN sendet. Der Border-Router kann auch gleichzeitig ein sog. *Personal Area Network (PAN)-Koordinator* sein. Der PAN-Koordinator hat die Aufgabe das Netzwerk zu verwalten und sich bspw. um die Adressierung der Knoten und um das Routing im WSN zu kümmern.

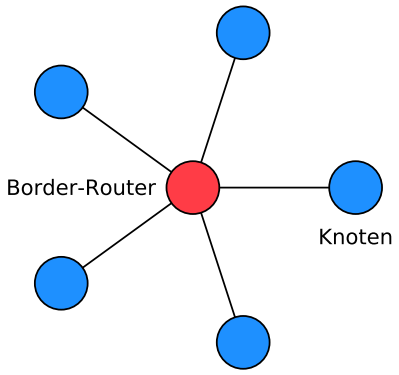


Abbildung 2.3.: Beispiel einer WSN-Stern-Topologie
Quelle: Eigene Darstellung

2.4.3. Masche

Die Maschen-Topologie (auch Mesh-Topologie genannt, Abb. 2.4) ist für WSNs von großer Bedeutung. Wie in Kap. 2.1 bereits erläutert, wird die Sendeleistung und damit die maximale Reichweite, oft gering gehalten, um Energie zu sparen. Damit Knoten, welche sich nicht in gegenseitiger Reichweite zueinander befinden, dennoch kommunizieren können, dienen andere Knoten als Vermittler (*Router*) zwischen diesen. Man spricht dabei auch von *Multi-Hop-Netzwerken*. Auf diese Weise kann ein WSN riesige Gebiete abdecken, trotz sehr geringer Sendeleistung der einzelnen Knoten.

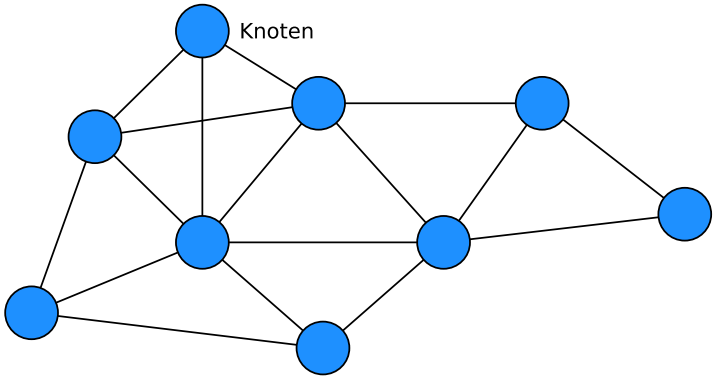
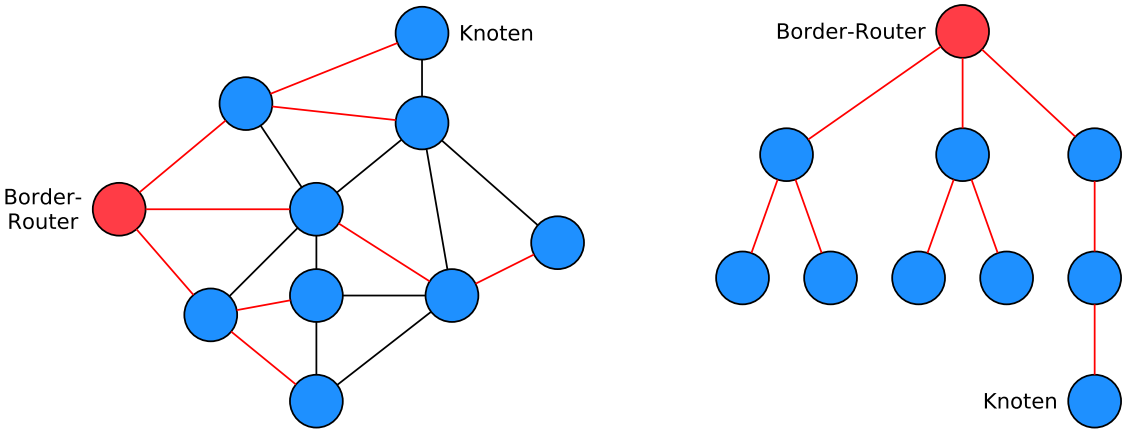


Abbildung 2.4.: Beispiel einer WSN-Maschen-Topologie
Quelle: Eigene Darstellung

In einem WSN mit Maschen-Topologie gibt es typischerweise einen Knoten, welcher als Border-Router bzw. PAN-Koordinator fungiert (vgl. Kap. 2.4.2). Das Maschen-Netzwerk zeichnet sich dadurch aus, dass es ein oder mehrere Routen zwischen zwei beliebigen Knoten gibt. Algorithmen entscheiden dann über die optimale Route zwischen diesen Knoten. Bei der Entscheidung über die Routen werden die Pfadkosten in Betracht gezogen. Die Pfadkosten können sich zusammensetzen aus der Latenz, dem Datendurchsatz, der Effizienz und der Stabilität der Einzelpfade, sowie dem aktuellen Stromverbrauch[2] der Knoten. Z. B. kann ein Knoten mit stationärer Stromversorgung gegenüber einem batteriebetriebenen Knoten als Router bevorzugt werden, um die Batterie zu schonen. Fällt ein Knoten mit Routing-Funktionalität aus, versucht der Routing-Algorithmus alternative Routen zu finden. Diese Eigenschaft wird auch *Self-Management* oder *Self-Healing* genannt (vgl. [23, S. 11] und [34, S. 82]).

Abb. 2.5 (a) zeigt beispielhaft alle Routen von einem beliebigen Knoten zum Border-Router, als rote Linien dargestellt. Der resultierende Routing-Graph ist in Abb. 2.5 (b) dargestellt. Die schwarzen Linien stellen alternative Pfade da.

[2]Ein Knoten, welcher als Router für viele andere Knoten dient, ist stärker ausgelastet, da er viele Pakete weiterleiten muss und verbraucht folglich mehr Strom.



(a) WSN-Maschen-Topologie (b) Routing-Hierarchie

Abbildung 2.5.: Maschen-WSN mit Border-Router als Wurzel (a) und der dazugehörigen Routing-Hierarchie (b)
Quelle: Eigene Darstellung

2.4.4. Baum

Bei der Baum-Topologie (siehe Abb. 2.6) handelt es sich um einen Spezialfall der Maschen-Topologie. Der Nachteil der Baum- gegenüber der Maschen-Topologie ist, dass es nur eine Route zwischen zwei Knoten gibt. Fällt ein Knoten mit Routing-Funktionalität aus, gibt es keine alternativen Routen, sodass Knoten, welche den ausgefallenen Knoten als Router nutzten, vom Netzwerk abgeschnitten werden. Der Vorteil der Baum-Topologie ist, dass die Routing-Algorithmen einfacher ausfallen und weniger Ressourcen benötigen.

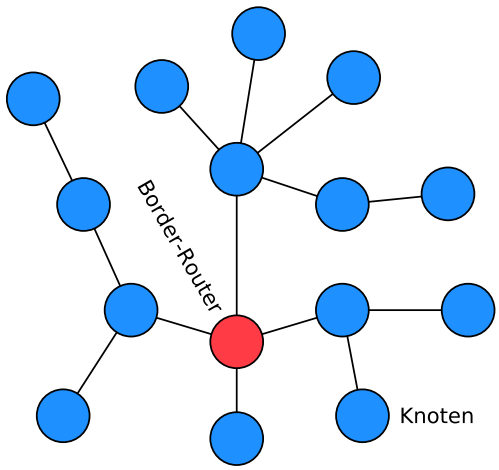


Abbildung 2.6.: Beispiel einer WSN-Baum-Topologie
Quelle: Eigene Darstellung

3. Funkstandards

In diesem Kapitel werden ausgewählte Funkstandards betrachtet, welche für WSNs in Frage kommen. Dabei werden insbesondere der Standard IEEE 802.15.4 (siehe Kap. 3.1) — da er besonders auf Energiesparsamkeit ausgelegt ist und sich dank seiner Offenheit gut untersuchen lässt — und auf IEEE 802.15.4 aufbauende Standards betrachtet (siehe Kap. 3.2 bis 3.4). Der Funkstandard EnOcean (siehe Kap. 3.5) ist insofern interessant, als dass er (fast komplett) auf Energy Harvesting (siehe Kap. 6) setzt. Ferner werden wegen ihrer Verbreitung noch Z-Wave (siehe Kap. 3.6) und Bluetooth Low-Energy (siehe Kap. 3.7) betrachtet. In Anlage A.1 ist eine Zusammenfassung der Eigenschaften aller in diesem Kapitel betrachteten Funkstandards zu finden.

3.1. IEEE 802.15.4

IEEE 802.15.4 ist ein offener Standard für Low-Rate Wireless Personal Area Networks (LR-WPANs) mit dem Ziel, propritäre Lösungen überflüssig zu machen und die Interoperabilität zwischen verschiedenen Geräten und Herstellern zu fördern (vgl. [34, S. xxiii ff.]). IEEE 802.15.4 wurde erstmals 2003 vom Institute of Electrical and Electronics Engineers (IEEE) herausgegeben. 2006 folgte die erste Revision IEEE 802.15.4b, auch bekannt als IEEE 802.15.4-2006, welche den Standard grundlegend überarbeitete und die ursprüngliche Version von 2003 (auch bekannt als IEEE 802.15.4-2003) obsolet machte. Jedoch ist IEEE 802.15.4-2006 abwärtskompatibel zu IEEE 802.15.4-2003. Weitere Änderungen (sog. *Amendments*[1]) erfuhr der Standard durch IEEE 802.15.4a von 2007, welche die IEEE 802.15.4-Familie um die beiden optional nutzbaren Fähigkeiten einer höheren Datenrate und Präzisions-Lokalisierung bzw. -Entfernungsmessung bereicherte und durch IEEE 802.15.4c/d von 2009, welche die PHY-Schicht (siehe Tabelle 3.1) um weitere Frequenzbänder und Modulationsverfahren für den chinesischen und japanischen Markt erweiterten. Die Revision IEEE 802.15.4-2011 aus dem Jahre 2011 führt die Version IEEE 802.15.4-2006 und die Amendments IEEE 802.15.4a/c/d zu einer Version zusammen (vgl. [50, S. IV]). Die wichtigste Änderung in IEEE 802.15.4-2011 besteht in einem vereinheitlichten Dokument und nicht in technischen Neuerungen.

Es gibt noch weitere Änderungen in der 802.15.4-Familie, die signifikantesten davon kamen wohl durch IEEE 802.15.4e, ein Amendment zur Version IEEE 802.15.4-2011, welches am 6. Februar 2012 von der IEEE Standards Association zugelassen wurde. Eine wesentliche Schwäche von IEEE 802.15.4-2011 ist, dass das WSN zur Laufzeit an einen einzigen Funkkanal gebunden ist und es kein Channel-Hopping (siehe Kap. 2.2) unterstützt. Ist der Funkkanal stark ausgelastet, z. B. weil das WSN eine große Anzahl an Knoten umfasst, leidet die Qualität der Verbindungen der Knoten untereinander. Dies führt zu höheren Latenzen, niedrigeren Datendurchsätzen und zu einer insgesamt schlechteren Stabilität des

[1] Ein Amendment ist eine Art Nebenversion. Oft werden die Neuerungen aus den Amendments in die späteren Hauptversionen übernommen.

WSNs. Zudem müssen dann vermehrt fehlgeschlagene Sendeversuche wiederholt werden und das kostet zusätzliche Energie. Zu den für diese Arbeit interessanten Neuerungen in IEEE 802.15.4e, gehören außerdem die Low-Energy-Mechanismen *Time-Slotted Channel Hopping (TSCH)*, *Coordinated Sampled Listening (CSL)* und *Receiver Initiated Transmissions (RIT)* in der MAC-Schicht, welche später in den Kapiteln 4.2.2, 4.2.7 und 4.2.8 ausführlich beschrieben werden.

Die aktuelle Revision ist IEEE 802.15.4-2015, welche auch die Änderungen des Amendments IEEE 802.15.4e beinhaltet. Sofern nicht anders angegeben, beziehen sich alle weiteren Ausführungen zu IEEE 802.15.4 in dieser Arbeit, auf die Version IEEE 802.15.4-2015.

Die Bestrebung von IEEE 802.15.4 ist es, so einfach wie möglich zu sein, um den Anforderungen von Geräten mit beschränkten Ressourcen und extrem niedrigem Stromverbrauch gerecht zu werden. Zwar gibt es schon einige Standards, welche dies anstreben, z. B. Bluetooth Smart (siehe Kap. 3.7) oder low-power 802.11 (auch bekannt als IEEE 802.11ah, ermöglicht die Verwendung existierender WLAN-Infrastrukturen, vgl. [63]), jedoch können diese nicht mit der Einfachheit von IEEE 802.15.4 mithalten. Einfachheit bedeutet, es werden weniger komplexe Schaltkreise benötigt, die Software-Implementierung fällt einfacher aus und benötigt weniger Rechenleistung. All dies spart Energie und Herstellungskosten. Natürlich hat dies auch eine Kehrseite, welche sich z. B. in einer geringen Datenrate (maximal 1000 kbit/s) und relativ hohen Latenzen bei der Datenübertragung bemerkbar macht. Der Trend neuer Standards für Funkdatenübertragung geht eher in Richtung immer höherer Datenraten und der Stromverbrauch ist dabei sekundär (vgl. [34, S. 4]). IEEE 802.15.4 schließt die dadurch entstehende Lücke (siehe Abb. 3.1).

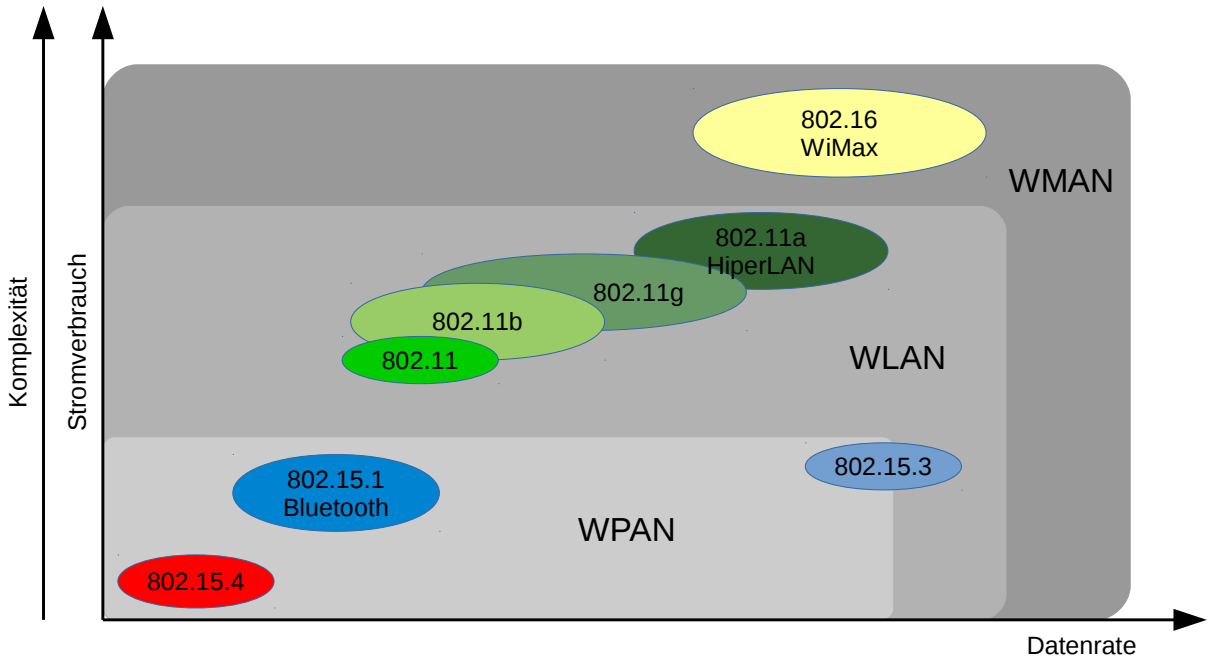


Abbildung 3.1.: Vergleich verschiedener IEEE 802-Standards
Quelle: In Anlehnung an [34, S. 5]

3.1.1. IEEE 802.15.4 im Vergleich mit dem ISO-OSI-Referenzmodell

Bezogen auf das ISO-OSI-Referenzmodell, definiert IEEE 802.15.4 die beiden untersten Schichten, die Bitübertragungs- und die Sicherungsschicht (siehe Tabelle 3.1). Die Sicherungsschicht wird aus der Sicht des IEEE nochmals in die beiden Schichten MAC und Logical Link Control (LLC) unterteilt (vgl. [64]). Die höheren Schichten müssen durch andere Standards, wie ZigBee, WirelessHART und 6LoWPAN (siehe Kap. 3.2 bis 3.4), spezifiziert werden.

	ISO-OSI-Schicht	**IEEE 802-Schicht**
7	Anwendung	Höhere Schichten
6	Darstellung	
5	Sitzung	
4	Transport	
3	Vermittlung	
2	Sicherung	Logical Link Control (LLC)
		Media Access Control (MAC)
1	Bitübertragung	Bitübertragung (Physical Signaling, PHY)

Tabelle 3.1.: Das ISO-OSI-Referenzmodell und IEEE 802
Quelle: In Anlehnung an [34, S. 7]

3.1.2. Full Function Devices und Reduced Function Devices

IEEE 802.15.4 definiert zwei Typen von Geräten, *Full Function Devices (FFDs)* und *Reduced Function Devices (RFDs)* (vgl. [34, S. 35]). Während ein FFD die komplette MAC-Funktionalität bereitstellt, ist es einem RFD erlaubt, nur einen Teil der MAC-Funktionalität bereitzustellen. Daher können RFDs mit extrem niedrigen Ressourcenanforderungen implementiert werden, welche außerdem billiger produziert werden können. FFDs sind meist ständig in Empfangsbereitschaft und können daher keinen Energiesparmodus nutzen. RFDs auf der anderen Seite können nur mit FFDs kommunizieren und können z. B. nicht die Rolle eines (PAN-)Koordinators (siehe folgendes Kap. 3.1.3) übernehmen. RFDs sind daher oft Sensoren oder Aktoren.

3.1.3. IEEE 802.15.4-Netzwerktopologien

IEEE 802.15.4 unterstützt die Netzwerktopologien Punkt-zu-Punkt (siehe Kap. 2.4.1) und Stern (siehe Kap. 2.4.2). Erstere Topologie ist gerade für industrielle Anwendungen interessant, da diese die Grundlage für komplexere Netzwerktopologien, wie der Maschen-Topologie, bildet (vgl. [34, S. 80]). Wie bereits in Kap. 2.4.3 erwähnt, können mit der Maschen-Topologie großflächige WSNs realisiert werden. Die Implementierung komplexerer Netzwerktopologien bleibt jedoch höheren Standards (siehe Kap. 3.2 bis 3.4 für einige Beispiele) vorbehalten. Typischerweise gibt es in einem IEEE 802.15.4-Netzwerk einen speziellen Knoten, welcher als PAN-Koordinator fungiert und das Netzwerk verwaltet.

Nicht direkt Bestandteil von IEEE 802.15.4 aber häufig in der Literatur zu finden, ist die Cluster-Tree-Netzwerktopologie (siehe Abb. 3.2). Dabei wird eine baumartige Hierarchie aus Knoten mit einem PAN-Koordinator als Wurzel aufgebaut. Es entstehen so

Eltern-Kind-Beziehungen zwischen den Knoten. Die Cluster-Tree-Netzwerktopologie setzt sich auf der Ebene von IEEE 802.15.4 aus lauter voneinander unabhängigen Punkt-zu-Punkt-Netzwerktopologien zusammen. Die Koordinatoren leiten den Datenverkehr zwischen verschiedenen Knoten weiter (Multi-Hop-Netzwerk). Jeder Knoten ist einem sog. *Cluster* zugeteilt. In jedem Cluster übernimmt ein Koordinator die Rolle des *Cluster-Heads*, welcher das Cluster verwaltet. Nur die Blatt-Knoten dürfen RFDs sein.

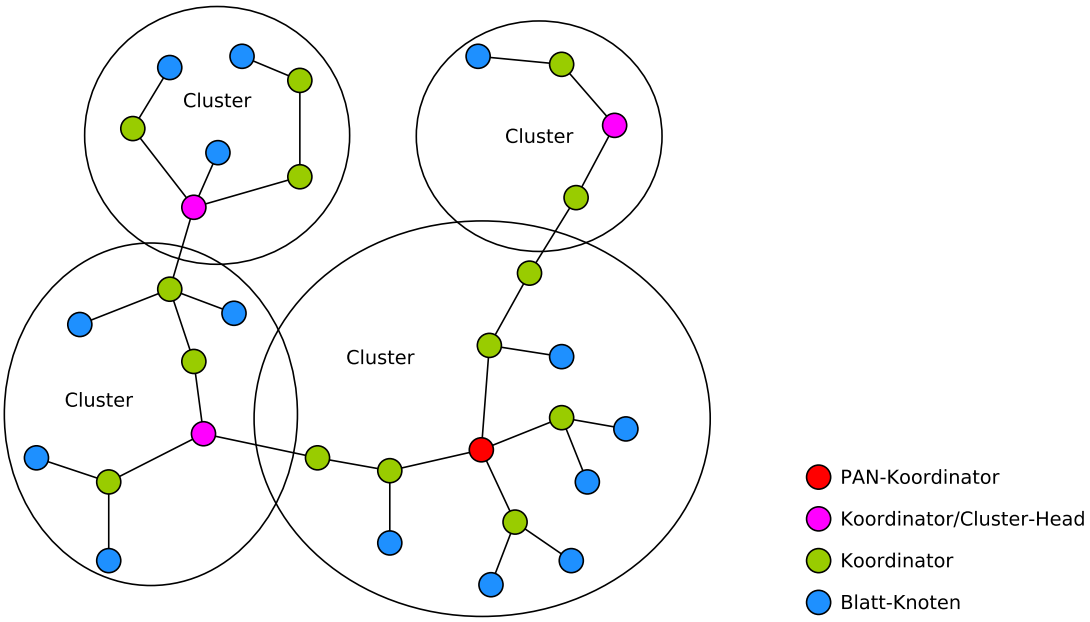


Abbildung 3.2.: Cluster-Tree-Netzwerktopologie
Quelle: In Anlehnung an [34, S. 36]

Zur eindeutigen Identifizierung hat jeder Knoten eine 64-Bit-Adresse[2]. Alternativ können 16-Bit-Kurzadressen verwendet werden. Diese benötigen weniger Speicher, setzen aber einen PAN-Koordinator voraus, welcher die Adressen vergibt und dafür sorgt, dass jeder Knoten netzwerkweit eine eindeutige Adresse hat.

3.1.4. Funkfrequenzen und Reichweite

Die Bitübertragungsschicht von IEEE 802.15.4 definiert die Frequenzbänder, Modulationsverfahren und Datenraten. IEEE 802.15.4 verwendet ausschließlich geeignete Frequenzen aus den ISM-Bändern, welche keiner gesonderten Lizenz bedürfen. Die wichtigsten davon sind in Tabelle 3.2 zusammengefasst.

IEEE 802.15.4 sieht eine Mindestsendeleistung von -3 dBm (0,5 mW) vor (vgl. [34, S. 68]). Nach oben hin ist die Sendeleistung nur durch die ortsüblichen Regularien begrenzt, z. B. 20 dBm (100 mW) für das 2,4-GHz-Band in Europa. Für den Empfänger sieht der Standard eine Sensitivität von -85 dBm für die Frequenzbänder aus Tabelle 3.2 vor, respektive -92 dBm in den niedrigeren Bändern, bei denen die BPSK-Modulation verwendet wird. Das bedeutet bspw. für das 2,4-GHz-Band, dass ein Empfänger, welcher ein Signal mit einer Stärke von -85 dBm empfängt, gerade noch in der Lage sein muss, dieses zu dekodieren. Am Beispiel des IEEE 802.15.4-Funk-Chips ATmega 128RFA1 von Atmel

[2]Das sind mehr als 184 Trillion oder $1,8 \cdot 10^{19}$ mögliche Adressen.

Frequenzband	Kanal-nummer	Datenrate	Modulationsverfahren
868 MHz[a] (868 - 868,6 MHz)	0	20 kbit/s	Binary Phase Shift Keying (BPSK)
		100 kbit/s	Offset Quadrature Phase Shift Keying (O-QPSK)
		250 kbit/s	Parallel Sequence Spread Spectrum (PSSS)
915 MHz[b] (902 - 928 MHz)	1 - 10	40 kbit/s	BPSK
		250 kbit/s	O-QPSK
		250 kbit/s	PSSS
2,4 GHz[c] (2,4 - 2,4835 GHz)	11 - 26	250 kbit/s	O-QPSK
		1 Mbit/s	Chirp Spread Spectrum (CSS)
		2 Mbit/s[d]	O-QPSK

[a] In Europa nutzbares SRD-Band.
[b] In den USA nutzbares ISM-Band.
[c] Weltweit (bis auf wenige Ausnahmen) nutzbares ISM-Band.
[d] Ab Amendment 4 (IEEE 802.15.4t-2017) zu IEEE 802.15.4-2015.

Tabelle 3.2.: Die wichtigsten Frequenzbänder, Kanalnummern und Datenraten von IEEE 802.15.4

und unter Verwendung des 2,4-GHz-Bandes, ließe sich eine Reichweite von ca. 665 m[3] im Vakuum, also ohne jegliche Störquellen, erziehlen. In der Praxis herrschen jedoch keine so idealen Bedingungen wie im Vakuum. Hier ist mit Reflektion, Beugung, Streuung und anderen störenden Faktoren zu rechnen. Eine realistische Reichweite für den Einsatz innerhalb von Gebäuden beträgt etwa 30 m.[4]

3.1.5. IEEE 802.15.4 Superframe

IEEE 802.15.4 definiert einen optionalen *Superframe* im sog. *Beacon-Enabled-Mode*. Dieser ist unterteilt in 16 gleich lange Zeitschlitze, den sog. *Slots*. Der Superframe wird vom PAN-Koordinator verwaltet. Dieser sendet periodisch ein *Beacon* genanntes Datenpaket aus. Der erste Slot eines Superframes beginnt mit dem Beginn eines Beacons. Die Länge eines Superframes erstreckt sich bis zum Beginn des nächsten Beacons (siehe Abb. 3.3).

Abbildung 3.3.: Aufbau eines IEEE 802.15.4-Superframes
Quelle: [34, S. 84]

Knoten, welche mit dem PAN-Koordinator kommunizieren möchten, müssen dies innerhalb eines Slots zwischen zwei aufeinanderfolgender Beacons, der sog. *Contention Access Period*, tun. Die *Beacon*-Datenpakete helfen außerdem, alle Knoten synchron zu halten. Dies senkt die Anforderung an die Genauigkeit der Zeitgeber (meist Schwingquarze) und führt dadurch zu niedrigeren Produktionskosten. Das *Beacon*-Datenpaket hat im 2,4-

[3] Errechnet mit Hilfe des Friis-Modells (siehe [34, S. 70])
[4] Hier ist die Luftlinie zwischen zwei Knoten gemeint. Errechnet wurde sie mit Hilfe des Log-Distance Path Loss Modells (vgl. [34, S. 71]).

GHz-Band eine Länge von 544 μs (vgl. [34, S. 30])[5]. Der zeitliche Abstand zwischen zwei Beacons ist variabel und kann im 2,4-GHz-Band zwischen 15,36 ms und 251.66 s betragen. Für den PAN-Koordinator bedeutet dies, dass er zwischen 2,3 % und 0,000216 % seiner Zeit mit dem Senden von Beacons beschäftigt ist. Die Länge eines Slots beträgt ein Sechzehntel der Länge eines Superframes. Die Länge eines Superframes ist gleich dem Abstand zweier aufeinanderfolgender Beacons.

Auf Anfrage eines Knotens kann der PAN-Koordinator diesem auch Slots des Superframes dediziert zuweisen. In diesen, als Guaranteed Time Slot (GTS) bezeichneten Slots, darf nur dieser Knoten senden. Normalerweise konkurrieren die sendewilligen Knoten um einen Slot (außer um den ersten Slot, da in diesem der Beacon gesendet wird). Dafür kommt das Carrier Sense Multiple Access/Collision Avoidance (CSMA-CA)-Verfahren zum Einsatz. D. h. jeder Knoten, welcher senden möchte, muss zunächst prüfen, ob der Kanal frei ist. Ist der Kanal nicht frei, muss der Knoten den Versuch im nächsten möglichen Slot wiederholen. Die GTS sind sinnvoll, wenn ein Knoten eine garantierte Bandbreite oder Latenz benötigt. Tatsächlich bedeutet dies aber nicht, dass ein Knoten während seines GTS garantiert senden darf. Konkurrierende Technologien, wie WLAN oder andere IEEE 802.15.4-Netzwerke, welche auf demselben Kanal senden, können dies verhindern. In einem Superframe kann es bis zu sieben GTS geben. Der Teil des Superframes, welchen die GTS einnehmen, nennt man *Contention Free Period*. Sie folgt unmittelbar nach der Contention Access Period (siehe Abb. 3.4).

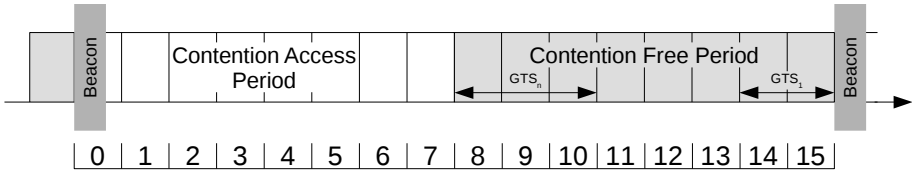


Abbildung 3.4.: Beispiel eines IEEE 802.15.4-Superframes mit GTS
Quelle: [34, S. 85]

Wenn eine hohe Bandbreite und niedrige Latenz nicht erforderlich sind, kann die Zeit zwischen dem Beginn zweier aufeinanderfolgender Beacons auch in zwei Teile, der *Active Portion* und der *Inactive Portion*, unterteilt werden (siehe Abb. 3.5). Die Active Portion enthält die 16 Slots des Superframes, welche zur Kommunikation genutzt werden können. Während der Inactive Portion findet keine Kommunikation statt. Diese Zeit können die Knoten in einem Energiesparmodus verbringen, um Energie zu sparen. Diese Funktion ist besonders für batteriebetriebene PAN-Koordinatoren interessant.

Im 2,4-GHz-Band hat ein Superframe eine minimale Länge von 15,36 ms. Der maximale Abstand zwischen dem Beginn zweier aufeinanderfolgender Beacons respektive Superframes beträgt 246 s. Im Extremfall können die Knoten im Netzwerk also mehr als 99,99 % der Zeit in einem Energiesparmodus verweilen.

Der Beacon-Enabled-Mode mit Inactive Portion kann auch genutzt werden, um die Koexistenz mehrerer IEEE 802.15.4-Netzwerke zu verbessern, indem jedes Netzwerk seine Active Portion dorthin legt, wo die anderen Netzwerke ihre Inactive Portion haben (sie-

[5]Die Länge eines Beacons bzw. eines Datenpaketes allgemein hängt vom verwendeten Modulationsverfahren und der Sendefrequenz ab. Im 2,4-GHz-Band ist die Übertragungsdauer derselben Datenmenge am kürzesten.

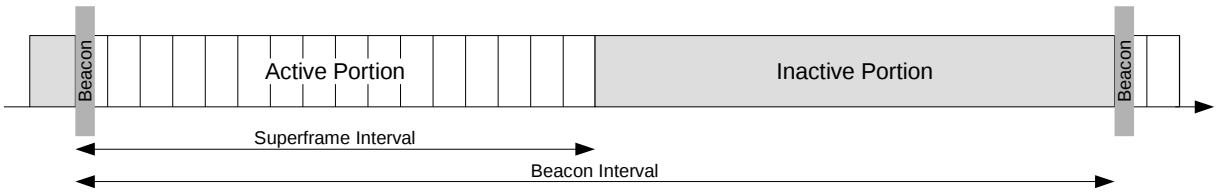


Abbildung 3.5.: Beispiel eines IEEE 802.15.4-Superframes mit anschließender Inactive Portion

Quelle: [34, S. 86]

he Abb. 3.6). Auch Multihop-Netzwerke im Beacon-Enabled-Mode, welche die Cluster-Tree-Netzwerktopologie nutzen (siehe [34, S. 82 ff.]), können so realisiert werden. Bei der Cluster-Tree-Netzwerktopologie kommt ein hierarchischer Routing-Algorithmus zum Einsatz. Jeder Kind-Knoten richtet den Beginn seines Superframes am Beginn der Inactive Portion seines Eltern-Knotens aus.

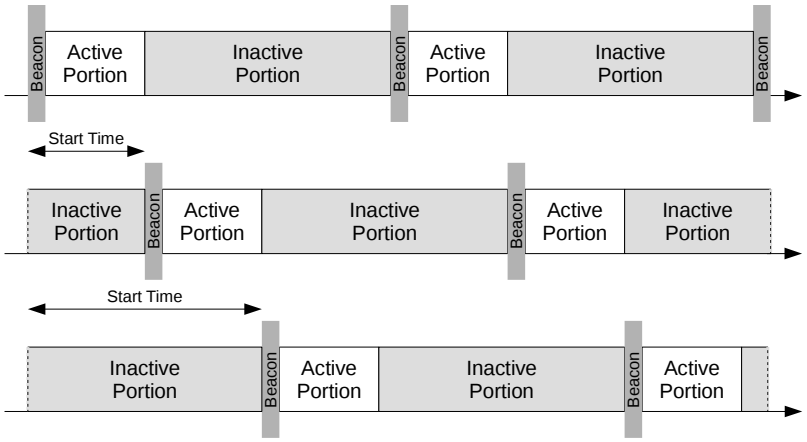


Abbildung 3.6.: Koexistenz mehrerer IEEE 802.15.4-Netzwerke, welche Superframes verwenden

Quelle: [34, S. 86]

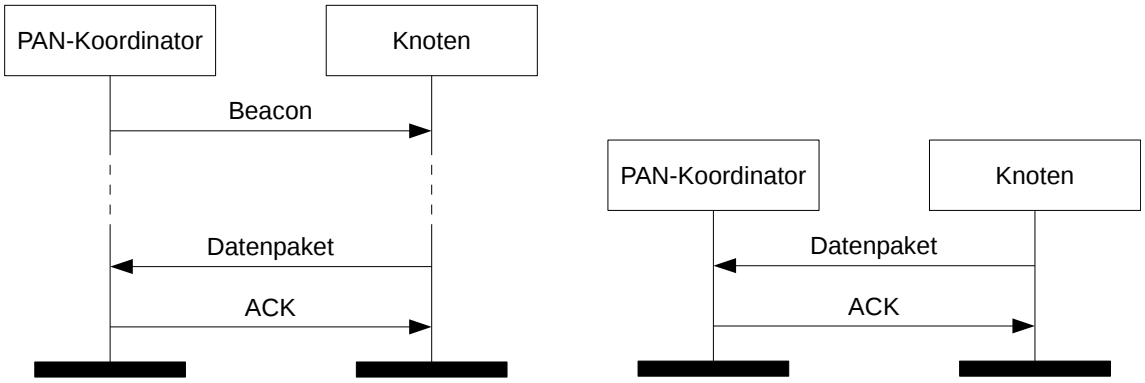
Dem *Beacon-Enabled-Mode* gegenüber steht der *Non-Beacon-Enabled-Mode*, wenn auf IEEE 802.15.4-Superframes verzichtet werden kann. Im Non-Beacon-Enabled-Mode können alle Knoten, bis auf den PAN-Koordinatoren, welcher ständig empfangsbereit sein muss, beliebig lange in einem Energiesparmodus verweilen (vgl. [34, S. 30 ff.]).

3.1.6. Direkte Datenübertragung

Die direkte Datenübertragung (im IEEE 802.15.4-Jargon auch *Direct Data Transfer* genannt) bezeichnet das Vorgehen, bei dem der Sender unter Verwendung des CSMA-CA-Verfahrens ein Datenpaket an den Empfänger übermittelt. Der Empfänger sendet ein optionales Acknowledgement (ACK)-Paket an den Sender zurück, um den erfolgreichen Erhalt des Datenpaketes zu bestätigen. Ob ein ACK erforderlich ist, teilt der Sender dem Empfänger in seinem Datenpaket über ein spezielles Flag mit.

Bezogen auf den oben beschriebenen Beacon-Enabled-Mode bedeutet dies, dass wenn ein Knoten Daten an den PAN-Koordinator senden will, er den Beacon abwarten muss und die Daten dann in einem geeigneten Slot des Superframes senden muss (siehe Abb. 3.7 (a)).

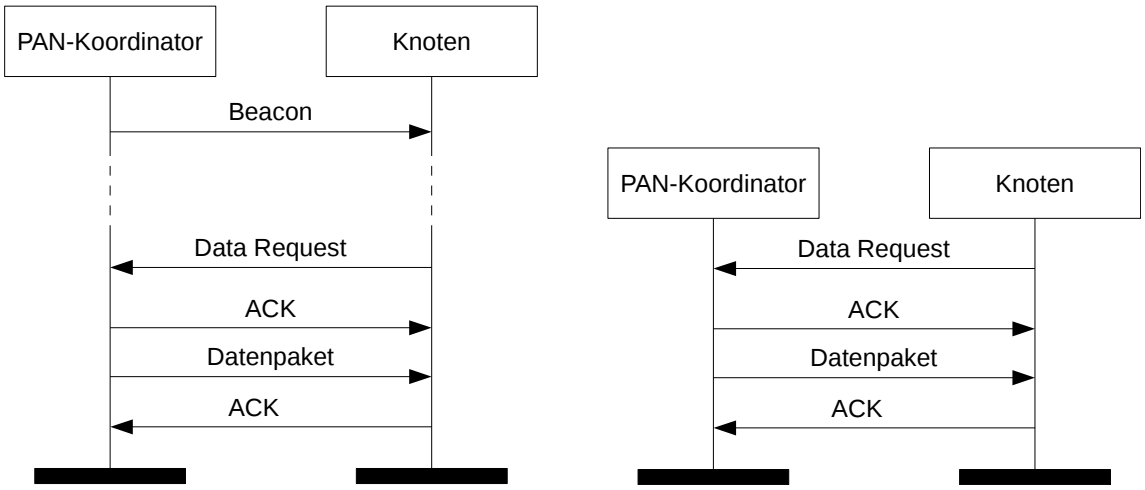
Im Non-Beacon-Enabled-Mode sendet ein Knoten die Daten (unter vorheriger Anwendung des CSMA-CA-Verfahrens) zu einem beliebigen Zeitpunkt an den PAN-Koordinator (siehe Abb. 3.7 (b)).



(a) Beacon-Enabled-Mode (b) Non-Beacon-Enabled-Mode

Abbildung 3.7.: Direkte Datenübertragung im Beacon- (a) und Non-Beacon-Enabled-Mode (b)

Quelle: [34, S. 88]



(a) Beacon-Enabled-Mode (b) Non-Beacon-Enabled-Mode

Abbildung 3.8.: Indirekte Datenübertragung im Beacon- (a) und Non-Beacon-Enabled-Mode (b)

Quelle: [34, S. 89 ff.]

3.1.7. Indirekte Datenübertragung

Bei der indirekten Datenübertragung (*Indirect Message Transfer*) speichert der Sender eines Datenpaketes dieses zwischen, bis der Empfänger es anfordert. Dies kann man sich vorstellen wie bei einem Briefkasten. Der Postbote wirft einen Brief ein, welcher im Briefkasten „zwischengespeichert“ wird. Der Empfänger, der Briefkastenbesitzer in diesem Fall, schaut in regelmäßigen Abständen nach, ob Post (ein Datenpaket) für ihn vorliegt. Dieser Modus wird genutzt, wenn der Empfänger, um Energie zu sparen, nicht ständig empfangsbereit ist.

Im Beacon-Enabled-Mode wird die indirekte Datenübertragung verwendet, wenn der PAN-Koordinator Daten an einen Knoten senden will. Für welchen Knoten Datenpakete vorliegen, übermittelt der PAN-Koordinator im Beacon. Der entsprechende Knoten sendet dann in einem geeigneten Slot ein *Data Request* an den PAN-Koordinator. Dieser sendet daraufhin ein ACK, gefolgt von dem Datenpaket an den Knoten. Der Knoten bestätigt den Erhalt des Datenpaketes wiederum mit einem ACK, falls dieses gefordert wurde (siehe Abb. 3.8 (a)). Im Non-Beacon-Enabled-Mode sendet der Knoten zu einem beliebigen Zeitpunkt und meist periodisch ein Data Request an den PAN-Koordinator. Dieser bestätigt den Erhalt des Data Requests mit einem ACK. Liegen Daten vor, sendet der PAN-Koordinator diese gleich nach dem ACK an den Knoten. Der Knoten bestätigt den Erhalt wiederum mit einem ACK (siehe Abb. 3.8 (b)).

Im Beacon-Enabled-Mode muss ein Knoten im Idealfall nur die Beacons empfangen und kann die restliche Zeit, sofern er keine Daten empfangen oder senden möchte, in einem Energiesparmodus zubringen. Bei einer Beacon-Dauer von 544 μs[6] im 2,4-GHz-Band und einer maximalen Beacon-Periode von 246 s bedeutet dies, dass der Knoten über 99.999 % der Dauer einer Beacon-Periode einen Energiesparmodus nutzen kann.

Höhere Netzwerktopologien, wie die Maschen-Netzwerktopologie, welche nicht von IEEE 802.15.4 definiert werden, können auch Gebrauch von Superframes und dem Non-Beacon- und Beacon-Enabled-Mode machen. Die konkrete Strategie der Datenübertragung bleibt dann jedoch den höheren Netzwerkschichten bzw. Standards vorbehalten.

3.2. WirelessHART

WirelessHART ist eine Erweiterung des drahtgebundenen Highway Addressable Remote Transducer (HART)-Protokolls. Es ermöglicht den Austausch digitaler Informationen zwischen intelligenten Geräten zur Überwachung und Steuerung industrieller Anlagen und ist aufgrund seiner Verbreitung mittlerweile ein De-facto-Standard auf diesem Gebiet (vgl. [34, S. 167]). HART wurde in den späten 1980er-Jahren entwickelt und wird von der FieldComm Group[7] verwaltet (vgl. [31]).

WirelessHART wurde im September 2007 veröffentlicht. Im März 2010 wurde WirelessHART als Norm IEC 62591 von der International Electrotechnical Commission (IEC) und im Juni 2010 von den europäischen Normungsorganisationen als EN IEC 62591 übernommen. WirelessHART legt besonders großen Wert auf die hohen Anforderungen der Industrie an Zuverlässigkeit und Sicherheit[8] (gegen unerlaubtes Mithören und Manipulation der Daten, Penetration des Netzwerkes usw.). Gleichzeitig versucht WirelessHART so einfach wie möglich zu bleiben und auch benutzbar für nicht spezialisierte Arbeitskräfte zu sein. In industriellen Anlagen werden oft tausende von Sensoren und Aktoren benötigt (vgl. [34, S. 166]). Die Übermittlung von Daten in Echtzeit und die Anforderungen an niedrige Latenzen von typischen 1 bis 100 Sekunden ist dabei eine große Herausforderung.

[6] In der Praxis muss ein Knoten mindestens etwas vor dem erwarteten Zeitpunkt des Beacons in Empfangsbereitschaft gehen, um die Ungenauigkeit zwischen seinem und dem Zeitgeber des PAN-Koordinators zu kompensieren.

[7] `https://fieldcommgroup.org/`

[8] Die Daten drahtlos kommunizierender Geräte können nicht einfach wie ein Kabelnetz durch physische Maßnahmen geschützt werden.

WirelessHART verwendet das 2,4-GHz-Band und schreibt eine maximale Sendeleistung von 10 dBm (10 mW) vor (vgl. [34, S. 207]). Da sich industrielle Anlagen häufig über ein großes Areal erstrecken, verwendet WirelessHART eine Maschen-Topologie (siehe Kap. 2.4.3), um die Reichweite zu erhöhen.

WirelessHART hat IEEE 802.15.4 als Basis gewählt, wegen seiner guten Koexistenz-Eigenschaften zu anderen Technologien, welche das 2,4-GHz-Band nutzen. Dies gilt insbesondere für die Koexistenz zu WLAN, da dieses häufig als Backbone-Netzwerk vieler industrieller Anwendungen dient (vgl. [34, S. 167]).

Um die Lebensdauer der Batterie eines Knotens zu erhöhen, schreibt WirelessHART ein intelligentes Kommunikationsverhalten vor. Demnach soll ein Sensor-Knoten nur dann Daten liefern, wenn er dazu aufgefordert wurde oder wenn sich eine Prozessvariable (z. B. ein Temperaturwert) geändert hat. Ferner kommt, ähnlich wie beim IEEE 802.15.4-Superframe, ein Zeitmultiplexverfahren (eng. Time Division Multiple Access (TDMA)) zum Einsatz, bei welchem jeder Knoten einen kurzen Zeitabschnitt zugeteilt bekommt, währenddessen er Daten übertragen darf. Den Rest der Zeit kann der Knoten in einem Energiesparmodus verbringen. Das Zeitmultiplexverfahren von WirelessHART ist konfigurierbar und kann so gewählt werden, dass die Batterie nur wenig beansprucht wird und somit jahrelang hält (vgl. [34, S. 170]). Die Zuweisung der einzelnen Zeitschlitze zu den Knoten beim TDMA-Verfahren, sowie die Steuerung vieler anderer WirelessHART-Funktionen, wie die Organisation des Maschen-Netzwerkes oder die Auswahl der Funkkanäle, übernimmt ein zentraler Netzwerkkoordinator.

Das kontinuierliche Wechseln des Funkkanals (Channel-Hopping) und das Sperren (Blacklisting) bestimmter Kanäle, welche besonders stark gestört sind, erhöht die Stabilität und die Sicherheit des Netzwerkes.

3.2.1. Die Komponenten eines WirelessHART-Netzwerkes

Ein WirelessHART-Netzwerk besteht aus folgenden Komponenten (siehe Abb. 3.9):

Das **Field Device** erfasst Werte eines industriellen Prozesses oder steuert diesen. Beide Aufgaben können auch in einem einzelnen Field Device kombiniert werden. Es kommuniziert bidirektional und kann als Router für andere WirelessHART-Geräte dienen. Die Stromversorgung kann durch unterschiedliche Arten erfolgen, eine davon ist die Batterie.

Adapter dienen dazu nicht drahtlos kommunizierende Geräte WirelessHART-fähig zu machen. Diese Geräte erscheinen dann im Netzwerk wie ein natives Field Device und haben auch die gleichen Fähigkeiten, einschließlich des Routings.

Router dienen der Paketvermittlung zwischen anderen Geräten im WirelessHART-Netzwerk. Da die anderen Geräte ebenfalls als Router fungieren, sind reine Router optional. Sie können eingesetzt werden, um ein Netzwerk weiter auszudehnen, dessen Zuverlässigkeit durch Redundanz zu erhöhen und um andere Field Devices von Routing-Aufgaben zu entlasten, um dessen Batterien zu schonen.

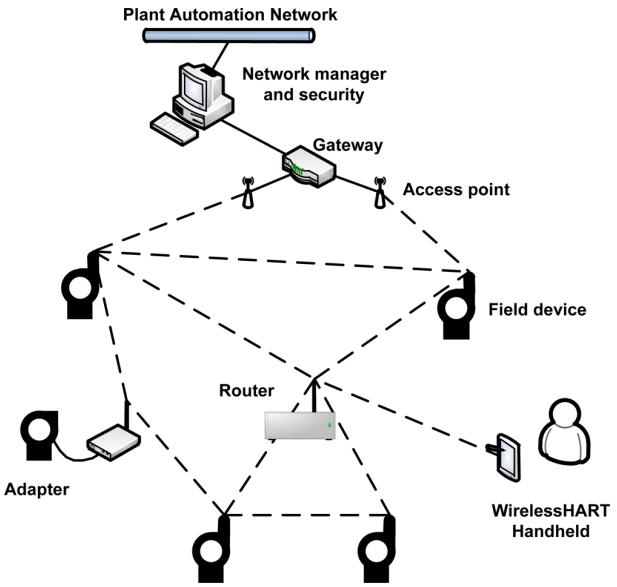


Abbildung 3.9.: Die Komponenten eines WirelessHART-Netzwerkes
Quelle: [49]

In einem WirelessHART-Netzwerk gibt es genau ein (logisches) **Gateway**. Diese verbindet das WirelessHART-Netzwerk mit einem anderen (z. B. kabelgebundenen IP-) Netzwerk, dem sog. *Plant Automation Network*. Dabei nimmt das Gateway die Konvertierung zwischen den unterschiedlichen Protokollen und Datenformaten beider Netzwerke vor. Das Gateway übernimmt auch noch andere Aufgaben. Es dient z. B. als Zeitgeber für die Knoten im Netzwerk, um das TDMA-Verfahren synchron zu halten.

Der **Access Point** dient als physicher Übergang vom drahtlosen WirelessHART-Netzwerk zum Gateway. Um die Zuverlässigkeit und den Durchsatz des Netzwerkes zu erhöhen, kann ein Gateway über mehrere Access Points verfügen.

Das **Handheld Device** wird verwendet, um das WirelessHART-Netzwerk einzurichten, zu warten und zu überwachen. Es gibt verschiedene Arten von Handheld Devices. Das in Abb. 3.9 dargestellte Handheld Device ist portabel und kommuniziert direkt mit dem WirelessHART-Netzwerk wie ein Field Device.

Der **Network Manager** kommuniziert nicht direkt mit dem WirelessHART-Netzwerk. Er gehört logisch zum Gateway und ist entweder an dieses über ein anderes Netzwerk angeschlossen oder kann physisch dasselbe Gerät wie das Gateway sein. Der Network Manager verwaltet das WirelessHART-Netzwerk. Er entscheidet im Rahmen des TDMA-Verfahrens, wann welcher Knoten im WirelessHART-Netzwerk kommunizieren darf, überwacht das WirelessHART-Netzwerk und erstattet Bericht über dessen Gesundheitszustand, z. B. darüber, dass die Batterie eines Knotens zur Neige geht. Außerdem verwaltet er die Routing-Tabellen und ist dafür verantwortlich zur Laufzeit alternative Routen zu konfigurieren. Bei der dynamischen Konfiguration des WirelessHART-Netzwerkes, berücksichtigt der Network Manager auch den Stromverbrauch der Knoten (vgl. [34, S. 189]).

Der **Security Manager** generiert, verwaltet und verteilt die Schlüssel (Encryption Keys) zur sicheren Kommunikation auf die Knoten im WirelessHART-Netzwerk. Wie

auch der Network Manager ist der Security Manager kein Knoten, welcher sich direkt im drahtlosen WirelessHART-Netzwerk befindet, sondern er ist entweder über ein anderes Netzwerk an das Gateway angebunden oder kann physisch dasselbe Gerät wie dieses sein.

3.2.2. Das TDMA-Verfahren von WirelessHART

WirelessHART definiert sog. Superframes, welche in gleich viele, zehn Millisekunden lange Zeitscheiben unterteilt sind (siehe Abb. 3.10). Jeder Knoten im Netzwerk fordert Zeitscheiben, je nach seinen Anforderungen an die Latenz und den Datendurchsatz, vom Network Manager an. Jeder Knoten erhält somit dedizierte Zeitscheiben zur Datenübertragung. Dadurch werden Kollisionen vermieden und die Bandbreite optimal genutzt. Die Zeit, während ein Knoten nicht senden darf und empfangen muss, kann dieser in einem Energiesparmodus verbringen.

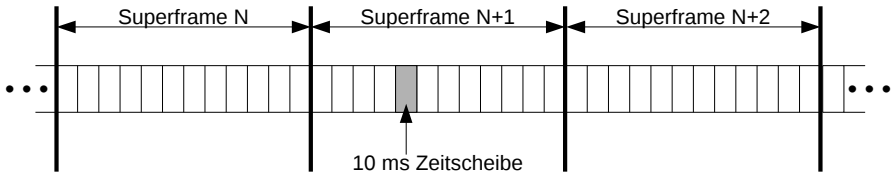


Abbildung 3.10.: WirelessHART-Superframe
Quelle: [34, S. 208]

Typischerweise ist eine Zeitscheibe einem Kanal und zwei Knoten zugewiesen, dem einen zum Senden und dem anderen zum Empfangen von Daten. Mehrere Knoten können gleichzeitig senden, indem sie Zeitscheiben unterschiedlicher Kanäle nutzen. Werden alle Kanäle des 2,4 GHz-ISM-Bandes genutzt (vgl. Tabelle 3.1), können also 16 Knoten gleichzeitig senden.

Abb. 3.11 zeigt den Ablauf einer Datenübertragung zwischen Sender und Empfänger innerhalb einer Zeitscheibe. Der Sender führt zunächst im Empfangsmodus einen sog. CCA durch, bei welchem er prüft, ob der Kanal frei ist. Wie in Kap. 3.2.1 bereits beschrieben, verteilt der Network Manager die Zeitscheiben so, dass es nicht zu Kollisionen kommen kann. Jedoch kann der Network Manager keinen Einfluss auf andere Technologien (z. B. WLAN oder andere WirelessHART-Netzwerke) nehmen, welche denselben Kanal verwenden. Ist der Kanal frei, wird gesendet. Anderenfalls kann der Sender bis zur nächsten, ihm zugewiesenen, Zeitscheibe warten und es dann erneut versuchen. Der Empfänger erwartet währenddessen die Daten. Ist der Kanal frei, schaltet der Sender um in den Sendemodus und sendet seine Daten. Wurden die Daten empfangen, überprüft der Empfänger diese und sendet anschließend ein ACK. Währenddessen schaltet der Sender wieder in den Empfangsmodus und wartet auf das ACK. Wurde nach einer bestimmten Zeit kein ACK empfangen, wiederholt der Sender den Vorgang bei der nächsten, ihm zugewiesenen, Zeitscheibe. Bei speziellen Datenpaketen wie Broadcasts, entfällt das ACK. Beim gesamten Vorgang kommt es auf ein sehr exaktes Timing an, weshalb die Uhren aller Knoten möglichst synchron sein müssen. Da absolute Synchronität jedoch schwer zu erreichen ist, erlaubt WirelessHART einen Spielraum von einigen Mikrosekunden, wie auch in Abb. 3.11 angedeutet. Außerdem werden die Uhren aller Knoten regelmäßig mit dem Referenzzeitgeber des Gateways synchronisiert.

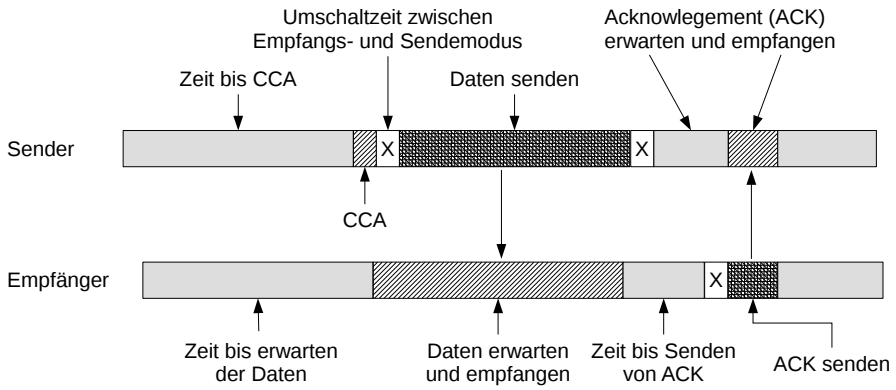


Abbildung 3.11.: Ablauf innerhalb eines Slots eines WirelessHART-Superframes
Quelle: In Anlehnung an [34, S. 212 ff.]

3.3. 6LoWPAN

IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) erlaubt den Einsatz von Internet Protocol Version 6 (IPv6) und anderen bewährten Netzwerkprotokollen wie TCP und UDP über IEEE 802.15.4 und damit auf Geräten mit beschränkten Ressourcen (vgl. [1]). Diese Geräte sind billig und können sparsam genug implementiert werden, damit sie über Jahre hinweg mit Batterien betrieben werden können. 6LoWPAN wird von der gleichnamigen Arbeitsgruppe der Internet Engineering Task Force (IETF) verwaltet.[9] Der Einsatz von IPv6 in WSNs ermöglicht die bessere Integration der WSNs in bestehende Netzwerke wie dem Internet und ist damit besonders gut für Internet of Things (IoT)-Anwendungen geeignet. Weitere Gründe für den Einsatz von IPv6 in WSNs sind (vgl. [40, S. 4]):

- Internet Protocol (IP)-basierte Technologien existieren zahlreich, sind gut verstanden und erprobt.
- Die Spezifikationen IP-basierter Technologien sind offen und frei verfügbar und so, im Gegensatz zu propritären Lösungen, einer breiteren Masse zugänglich.
- Werkzeuge zur Diagnose, Verwaltung usw. von IP-Netzwerken sind bereits vorhanden.
- Zur Integration in vorhandene IP-Netzwerke sind keine komplexen Geräte, wie spezielle Gateways oder Proxies, notwendig.

Hervorzuheben sind der große Adressraum von IPv6 sowie die schon vorhandene Autokonfigurationsfähigkeit, welche IPv6 für den Einsatz in WSNs prädestinieren.

6LoWPAN wurde erstmals in RFC[10] 4919 vom August 2007 vorgestellt und schließlich in RFC 4944 vom September 2007 spezifiziert. Es folgten Verbesserungen in Form von weiteren RFCs, zuletzt durch RFC 6775, welches die sog. Neighbor Discovery von 6LoWPAN optimiert.

[9] `https://datatracker.ietf.org/wg/6lowpan/about/`
[10] RFC steht für Requests for Comments.

Ein Hauptziel von 6LoWPAN ist die Verschlankung von IPv6-Datenpaketen. Z. B. ist das kleinst mögliche Datenpaket von IPv6 mit 1280 Bytes viel größer, als die maximale Größe eines IEEE 802.15.4-Datenpaketes (127 Bytes). Dazu wird unter anderem der IPv6-Header mittels verschiedener Verfahren komprimiert. Diese und andere Optimierungen von IPv6 machen den Einsatz von IPv6 auf Geräten mit beschränkten Ressourcen erst möglich und gestalten ihn effizient.

3.3.1. 6LoWPAN und das ISO-OSI-Referenzmodell

Aus der Sichtweise des ISO-OSI-Referenzmodells erweitert 6LoWPAN den IEEE 802.15.4-Standard um die Vermittlungsschicht, den sog. *6LoWPAN-Adaption-Layer* (vgl. [57], siehe Abb. 3.12). Die darüberliegenden Protokolle, wie TCP und UDP, sind in der Transportschicht angesiedelt.

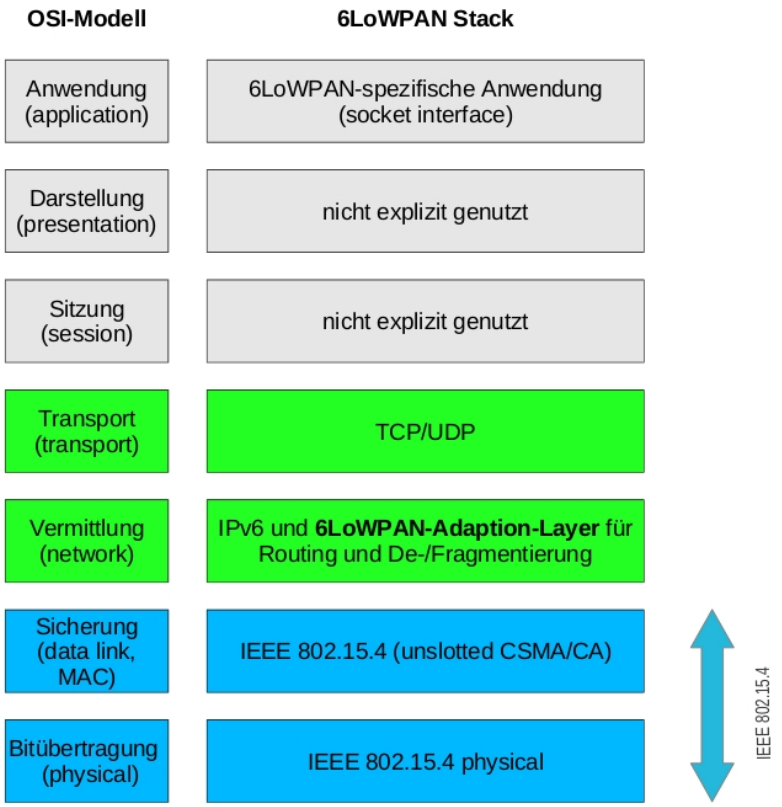


Abbildung 3.12.: Der 6LoWPAN-Stack im Vergleich mit dem ISO-OSI-Referenzmodell
Quelle: [57]

3.3.2. 6LoWPAN-Netzwerktopologien

6LoWPAN schreibt keine Netzwerktopologien vor und lässt sich mit verschiedensten Netzwerktopologien, darunter Baum und Masche, verwenden (vgl. [40, S. 5]). Die verwendeten Routing-Protokolle sollten Rücksicht auf die Anforderungen von WSNs nehmen und z. B. auf geringen Energieverbrauch ausgelegt sein. Router in einem WSN sind typischerweise FFDs mit mehr Ressourcen. Dennoch können sowohl FFDs als auch RFDs in einem WSN batteriebetrieben sein. Beim Routing sollten auch Knoten mit einbezogen werden, welche nicht Bestandteil des WSNs sind, sondern z. B. Ethernet nutzen. Ein nahtloser Übergang

des WSNs in Netzwerke auf Basis anderer Technologien ist daher, wie bereits zu Beginn von Kap. 3.3 beschrieben, ein Ziel von 6LoWPAN. Für das Routing schreibt 6LoWPAN ebenfalls kein bestimmtes Routingprotokoll vor, arbeitet aber in RFC 6606 die Anforderungen an ein solches auf. In der Praxis kommt z. B. das von der IETF-Arbeitsgruppe *Routing Over Low power Lossy networks* entwickelte *Routing Protocol for Low power and Lossy Networks (RPL)* zur Anwendung (vgl. [66]).

3.3.3. Implementierungen und Standards mit 6LoWPAN

Im Folgenden werden einige Implementierungen von 6LoWPAN und Standards, welche 6LoWPAN nutzen, beispielhaft genannt.

- ISA100.11a: Hierbei handelt es sich um einen Standard zur Überwachung und Steuerung industrieller Prozesse, welcher IPv6 nutzt. Er kann als Konkurrenz zu WirelessHART (siehe Kap. 3.2) gesehen werden.
- OpenWSN: Open Source Implementierung eines Protokoll-Stapels (eng. Protocol Stack). Er enthält unter anderem IEEE 802.15.4, 6LoWPAN, RPL, UDP/TCP und das Constrained Application Protocol (CoAP).
- Linux: Es gibt eine Implementierung von 6LoWPAN für das Betriebssystem Linux. Dies ist z. B. für Border-Router interessant, welche auf der Hardware von Mini-Computern, wie dem Raspberry Pi basieren.
- mbed OS: Open Source Betriebssystem der Firma ARM für Cortex-M Mikrocontroller.
- ZigBee-IP: Siehe Kap. 3.4.

3.3.4. Open Source Betriebssysteme für WSNs und dem IoT

Tabelle 3.3 führt einige Open Source Betriebssysteme für WSNs auf, welche 6LoWPAN unterstützen und für Mikrocontroller mit beschränkten Ressourcen konzipiert sind, da diese typischerweise das Herzstück eines Knotens in einem WSN bilden. Ein Beispiel eines solchen Mikrocontrollers ist der MSP430F1611 von Texas Instruments, mit einer maximalen Prozessor-Taktfrequenz von 8 MHz, 10 Kilobyte Arbeitsspeicher und 48 Kilobyte Festspeicher für Programme und Daten.

Alle aufgeführten Betriebssysteme nutzen RPL als Routing-Protokoll und realisieren so Maschen-Netzwerke. Weiterhin unterstützen die meisten dieser Betriebssysteme energiesparende MAC-Protokolle (manchmal auch als RDC-Protokolle bezeichnet, siehe Kap. 4). Dies ermöglicht sogar FFDs wie Routern einen sparsamen Betrieb.

3.4. ZigBee

Der ZigBee-Standard baut auf IEEE 802.15.4 auf und wird von der ZigBee Allianz verwaltet. „Die ZigBee Allianz (Gründung 2002) ist ein Industriekonsortium, welches sich zum Ziel gesetzt hat, zuverlässige, kostengünstige, energieeffiziente und für Funknetzwerke optimierende Produkte basierend auf einem offenen weltweiten Standard zu ermöglichen, die

Name	PS[a]	Erscheinungsjahr	Aktuelle Version	Beispielhaftes MAC-Protokoll
Contiki	C	2003	3.1 (Januar 2017)	ContikiMAC
RIOT	C/C++	2013	2017.04 (April 2017)	LwMAC
TinyOS	nesC[b]	1999	2.1.2 (August 2012)	BoX-MAC

[a] PS steht hier für Programmiersprache.
[b] Ein Dialekt von C. nesC wurde eigens für TinyOS entwickelt.

Tabelle 3.3.: Open Source Betriebssysteme mit Unterstützung für 6LoWPAN

für Überwachung- und Steuerungsaufgaben eingesetzt werden sollen." (Vgl. [39, S. 2].) Die erste Version der ZigBee-Spezifikation wurde 2004 herausgegeben. Diese gilt mittlerweile als veraltet und wurde von der komplett überarbeiteten Version *ZigBee 2006* abgelöst. Mit der nächsten Version *ZigBee 2007*[11] wurden eine automatische Fragmentierung von Datenpaketen, welche zu groß sind, um mit einem Mal versendet zu werden, in mehrere kleinere Datenpakete und ein Netzwerkfunkkanalmanager eingeführt. Bei letzterer Neuerung wird ein Knoten im ZigBee-Netzwerk als Netzwerkfunkkanalmanager eingesetzt, welcher den aktuellen Funkkanal überwacht. Gibt es zu viele Interferenzen auf dem aktuellen Kanal, veranlasst der Netzwerkfunkkanalmanager einen Wechsel des Kanals für das gesamte Netzwerk. Seit ZigBee 2007 gab es keine wesentlichen Änderungen mehr im ZigBee-Standard (vgl. [68]).

ZigBee ist für alle Anwendungsgebiete eines WSNs geeignet (siehe Kap. 1.1.1 für einige Beispiele). Sehr verbreitet ist die Anwendung von ZigBee vorallem für die Gebäudeautomation bzw. Heimautomatisierung (vgl. [39, S. 27]).

ZigBee-Geräte gibt es für alle Frequenzbereiche, welche der unterliegende Standard IEEE 802.15.4 (siehe Kap. 3.1) definiert. Aus den in Kap. 2.1 genannten Gründen, findet man jedoch meistens Geräte für das 2,4-GHz-Band (vgl. [39, S. 27]). Die Reichweite ist nur durch die gesetzlich vorgeschriebene maximale Sendeleistung von 10 mW (10 dBm) und der Empfindlichkeit des Empfängers beschränkt. Der ZigBee-Standard schlägt jedoch, je nach Anwendungsgebiet, unterschiedliche Reichweiten vor. Oft, z. B. für die Gebäudeautomation, sind Reichweiten von 70 m im Gebäude und 400 m im Freien angegeben.

3.4.1. ZigBee im Vergleich mit dem ISO-OSI-Referenzmodell

Wie bereits in Kap. 3.1 beschrieben, definiert IEEE 802.15.4 aus Sicht des ISO-OSI-Referenzmodells die unteren beiden Schichten Bitübertragung und Sicherung. ZigBee definiert die höheren Schichten *Vermittlung* und *Anwendung*. (Wobei die Anwendungsschicht von ZigBee die Schichten Transport bis Anwendung des ISO-OSI-Referenzmodells mit einschließt.) Es ergibt sich ein Modell mit vier Schichten, welches dem ISO-OSI-Referenzmodells in Abb. 3.13 gegenübergestellt ist.

Die Vermittlungsschicht von ZigBee kümmert sich vorallem um das Routing und ermöglicht so Maschen- und Baum-Netzwerktopologien. Die Implementation der eigentlichen Anwendung ist der Anwendungsschicht zuzuordnen.

[11] Oft wird ZigBee PRO fälschlicherweise synonym mit ZigBee 2007 verwendet.

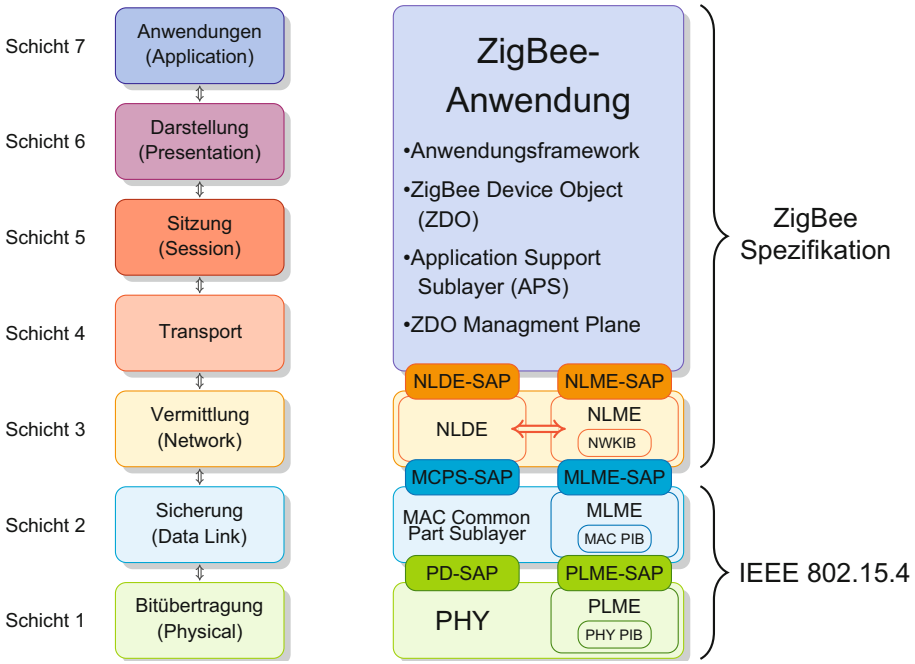


Abbildung 3.13.: Das ISO-OSI-Referenzmodell und IEEE 802.15.4/ZigBee im Vergleich
Quelle: [39, S. 217]

3.4.2. ZigBee-Netzwerktopologien und Knoten-Typen

Da ZigBee auf dem Standard IEEE 802.15.4 basiert, unterstützt es die Netzwerktopologien Punkt-zu-Punkt und Stern und führt die Baum-Netzwerktopologie ein (siehe Kap. 2.4). Das Stackprofil *ZigBee PRO* fügt zudem noch die Netzwerktopologien Masche hinzu. In einem ZigBee-Netzwerk, gibt es drei Typen von Knoten:

Es gibt genau einen **ZigBee-Koordinator** in einem ZigBee-Netzwerk. Er verwaltet das Netzwerk, initialisiert es, legt wichtige Parameter fest usw. Er dient auch gleichzeitig als ZigBee-Router. In den in Kap. 2.4 dargestellten Netzwerktopologien, nimmt der ZigBee-Koordinator die Position des Border-Routers ein.

ZigBee-Router dienen als Vermittler zwischen anderen ZigBee-Knoten und helfen so, die Reichweite des Netzwerkes auszudehnen.

ZigBee-Endgeräte haben keine Routing-Funktionalität und kommunizieren nur mit ihren Eltern-Knoten (den zuständigen ZigBee-Routern). Die Eltern-Knoten speichern den Datenverkehr zwischen, welcher an ihre Kind-Knoten adressiert ist, bis diese die Daten anfordern. Dies entspricht der indirekten Datenübertragung von IEEE 802.15.4 (siehe Kap. 3.1). ZigBee-Endgeräte können daher sehr lange in einem Energiesparmodus verweilen, was zu einer sehr langen Lebensdauer im Batteriebetrieb führt. ZigBee-Endgeräte können, wenn sie einen Energiesparmodus verwenden, ihren Empfänger entweder periodisch aktivieren oder bei Stimulierung, z. B. wenn die Taste eines ZigBee-Lichtschalters betätigt wird (vgl [6, S. 71]).

Die ZigBee-Spezifikation erwähnt außerdem den Begriff *Low Power Router* und verweist dabei auf batteriebetriebene Router, welche ihren Empfänger periodisch deaktivieren (vgl. [6, S. 362]). Router, welche eine stationäre Stromversorgung haben und durchge-

hend empfangsbereit sind, werden hingegen als *High Power Router* bezeichnet. Konkrete Hinweise zur Implementierung eines Low Power Routers liefert die ZigBee-Spezifikation nicht. Denkbar wäre die Verwendung von Superframes, mit einer möglichst langen Inactive Portion (siehe Kap. 3.1). Dies ist praktikabel für Stern- und Baum- bzw. Cluster-Tree-Netzwerktopologien, nicht aber für Maschen-Netzwerktopologien, da die Algorithmen zur Verwaltung der Superframes dort recht komplex ausfallen würden.

3.4.3. ZigBee Green Power

Mit ZigBee 2012 wurde eine neue, *Green Power* genannte, Erweiterung eingeführt (vgl. [8]). Green Power ZigBee-Geräte beziehen ihre Energie aus der Umgebung (siehe Kap. 6), z. B. über das Licht, einer Bewegung oder über Vibrationen. Um ein Green Power ZigBee-Gerät in ein ZigBee-Netzwerk zu integrieren, wird mindestens ein Green Power Proxy benötigt. Green Power Proxies sind typischerweise FFDs und verfügen über eine feste Stromversorgung. Abb. 3.14 zeigt die Integration eines ZigBee Green Power-Lichtschalters in ein ZigBee-Netzwerk, welcher eine Lampe steuert.

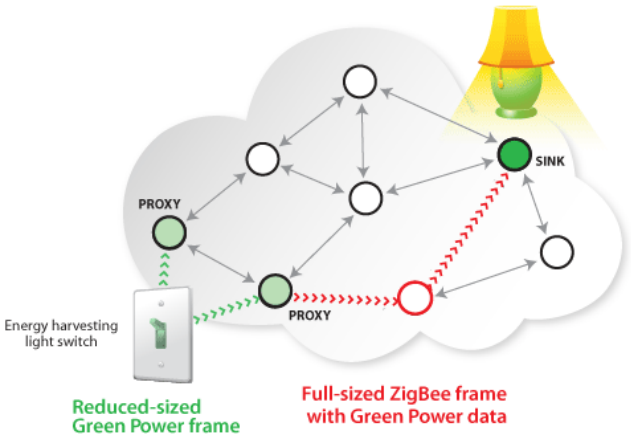


Abbildung 3.14.: ZigBee Green Power Maschen-Netzwerk
Quelle: [8]

Da das Energie-Budget eines Green Power ZigBee-Gerätes sehr gering ausfallen kann, wird ein spezielles Datenformat zur Übertragung verwendet, welches um einiges schlanker ist, als das normale ZigBee-Datenformat. ZigBee-Hosts, welche Daten von einem Green Power ZigBee-Gerät empfangen (in Abb. 3.14 die Steuerung der Lampe), müssen die *Green Power Sink* Funktionalität beherrschen, um das schlankere Datenformat auswerten zu können. Auf dem Weg der Daten von einem Green Power ZigBee-Gerät zum Ziel, können diese in das volle ZigBee-Datenformat eingekapselt werden, um in einem Multi-Hop-Netzwerk transportiert werden zu können, wie in Abb. 3.14 gezeigt. Des Weiteren ist in Abb. 3.14 ersichtlich, dass es mehrere Green Power Proxies geben kann. Ein Green Power ZigBee-Gerät bindet sich an keinen bestimmten Proxy. Die Daten werden von allen Proxies in Reichweite empfangen und die Proxies machen unter sich aus, wer die Daten weiterleitet (vgl. [5, S. 24]).

Verfahren wie das CSMA-CA sind für Green Power ZigBee-Geräte optional. Auch auf das Senden eines ACKs wird häufig verzichtet. Um dennoch zu gewährleisten, dass die Daten übertragen werden, kann ein Datenpaket stattdessen einfach mehrfach gesendet werden (vgl. [5, S. 65]). Ebenso ist eine bidirektionale Datenübertragung, also von einer

Green Power Sink zum Green Power ZigBee-Geräte, optional. Ein Green Power ZigBee-Lichtschalter bspw., welcher seine Energie aus der Bewegung im Moment seiner Betätigung bezieht, hat nur in diesem Moment Energie und ist sonst nicht in der Lage Daten zu empfangen.

ZigBee-IP

ZigBee-IP ist ein weiterer Standard der ZigBee Allianz, welcher im Hinblick auf das IoT und besonders auf intelligente Stromnetze, entwickelt wurde (vgl. [7] und [39, S. 219]). ZigBee-IP verwendet 6LoWPAN (siehe Kap. 3.3) und formt mit Hilfe des RPL eine Maschen-Netzwerktopologie. Ein Beispiel für eine ZigBee-IP-Netzwerktopologie, mit allen möglichen Knoten-Typen, ist an Abb. 3.15 dargestellt. Der Border-Router stellt die Verbindung mit anderen Netzwerken (in Abb. 3.15 dem Internet) her. Der Koordinator (Coordinator) verwaltet das Netzwerk. D. h., er kümmert sich unter anderem um das Routing, den Aufbau der Netzwerktopologie und um die Sicherheit des Netzwerkes. Router dienen dazu den Datenverkehr weiterzuleiten und helfen so, das Netzwerk auszudehnen. Ein Host (auch End Device genannt) ist ein Aktor oder Sensor. Es muss nicht jeder Knoten-Typ ein separates Gerät sein. Z. B. kann es sich beim Border-Router und Koordinator um dasselbe Gerät handeln, oder ein Sensor bzw. Aktor ist auch ein Router.

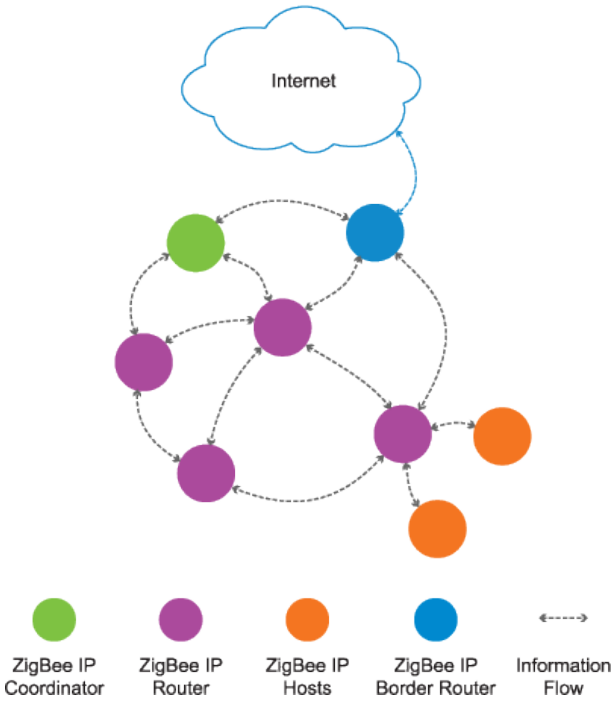


Abbildung 3.15.: Beispiel einer ZigBee-IP-Netzwerktopologie
Quelle: [7]

Zur Adressierung eines ZigBee-IP-Knotens werden 64-Bit-Adressen oder 32-Bit-Kurzadressen verwendet (siehe Kap. 3.1). Jedes Gerät muss mit einer vom Hersteller vergebenen und weltweit eindeutigen EUI-64[12]-Adresse vorkonfiguriert sein (vgl. [4, S. 9]).

[12]EUI steht hier für Extended Unique Identifier. Dabei handelt es sich um ein vom IEEE standardisiertes 64 Bit MAC-Adress-Format (vgl [61]).

Um Energie zu sparen, können Knoten einen Teil ihrer Zeit in einem Energiesparmodus verbringen. Solche Knoten werden in der ZigBee-IP-Spezifikation *Sleepy Nodes* genannt (vgl. [4, S. 40]). Ein Knoten, welcher ein Router ist, darf kein Sleepy Node sein. Ein Sleepy Node nutzt zum Empfangen die indirekte Datenübertragung von IEEE 802.15.4 (siehe Kap. 3.1). D. h. also, wenn Daten an einen Sleepy Node gesendet werden, speichert der zuständige Router die Datenpakete zwischen, bis der Sleepy Node diese „abholt". Dazu schickt der Sleepy Node in regelmäßigen Abständen Data Requests an seinen Router. Diesen Vorgang nennt man *Polling*. Da das Polling-Intervall, also der zeitliche Abstand zwischen den Data Requests, sehr groß sein kann, sollte die Initiierung einer Verbindung in der Regel von dem Sleepy Node ausgehen. Erwartet ein Sleepy Node Daten, kann sie das Polling-Intervall vorübergehend verkürzen. Sind größere eingehende Datenmengen zu erwarten, z. B. wenn ein neues Firmware-Update übertragen wird, kann der Sleepy Node sogar temporär zu einer Non-Sleepy Node werden und die direkte Datenübertragung nutzen. Das Polling-Intervall sollte so gewählt werden, dass es optimal für den Anwendungszweck ist. Läuft z. B. auf dem Sleepy Node ein Webserver, zu dem sich ein Benutzer mit einem Webbrowser verbinden kann, sollte das Polling-Intervall so gewählt werden, dass der Benutzer nicht all zu lange warten muss, bis die Seite geladen ist oder es zu einem Timeout kommt. Das maximale Polling-Intervall ist in der ZigBee-IP-Spezifikation mit einem Tag angegeben (vgl. [4, S. 41 u. 54]).

3.5. EnOcean

EnOcean ist ein Funkstandard für batterielose Sensoren[13] im Bereich Gebäudeautomation, Smart Home und IoT-Anwendungen (vgl. [30]). Die Sensoren (damit sind z. B. auch Lichtschalter gemeint) beziehen ihre Energie komplett aus der Umgebung (siehe Energy Harvesting in Kap. 6). Da die Energie, welche der Umgebung entnommen werden kann, oft sehr gering ausfällt, muss die Energie in einem Speicher gesammelt werden, während sich das Gerät in einem Energiesparmodus befindet, damit dieses in der Wachphase genügend Energie zum Senden bzw. Empfangen eines Datenpaketes hat. Es gibt daher EnOcean-Geräte mit einem Strombedarf von weniger als 100 nA im Energiesparmodus (vgl. [28, S. 4]).

EnOcean wurde von der 2001 gegründeten Firma EnOcean GmbH[14] erfunden (vgl. [60]). Seit 2008 wird EnOcean von der Enocean Alliance, bestehend aus über 170 Mitgliedsunternehmen (vgl. [36]) und mit Sitz in San Ramon (USA), weiterentwickelt. Die EnOcean GmbH hält jedoch noch Patente auf die EnOcean-Technologie.

EnOcean wurde 2012 zum internationalen Standard ISO/IEC 14543-3-10:2012 („Information technology – Home electronic systems (HES) architecture – Part 3-10: Wireless short-packet (WSP) protocol optimized for energy harvesting – Architecture and lower layer protocols") erhoben.

[13] Es gibt auch EnOcean-Aktoren. Diese benötigen jedoch meist eine Batterie oder stationäre Stromversorgung.

[14] Die EnOcean GmbH ist ein Ableger der Siemens AG und hat ihren Sitz in Deutschland.

3.5.1. Eigenschaften von ISO/IEC 14543-3-10:2012

EnOcean nutzt die Frequenzbänder 868 MHz (Europa und China), 902 MHz (Nordamerika) und 928 MHz (Japan), welche in diesen Regionen ohne gesonderte Lizenz verwendet werden dürfen. Diese Frequenzbänder haben den Vorteil, dass sie potentiell nicht so stark ausgelastet sind, wie das 2,4-GHz-Band, welches von weit verbreiteten Technologien wie WLAN und Bluetooth benutzt wird. Zudem haben niedrigere Frequenzen den Vorteil, dass sich höhere Reichweiten mit weniger Energie überbrücken lassen (vgl. [62]). EnOcean nennt eine Reichweite von bis zu 300 m im Freien und bis zu 30 m innerhalb von Gebäuden.

Die Übertragung eines Datenpaketes dauert nur etwa eine Millisekunde, was sich in zweierlei Hinsicht positiv auf den Energieverbrauch auswirkt: Kurze Datenpakete benötigen weniger Sendezeit und damit weniger Energie. Außerdem ist die Wahrscheinlichkeit von Kollisionen bei kurzen Datenpaketen geringer, was die Anzahl von wiederholt durchgeführten Sendeversuchen reduziert. Die Datenrate beträgt bis zu 125 kbit/s und die Kommunikation kann uni- oder bidirektional erfolgen.

3.5.2. ISO/IEC 14543-3-10:2012 im Vergleich mit dem ISO-OSI-Referenzmodell

Aus Sicht des ISO-OSI-Referenzmodells definiert ISO/IEC 14543-3-10:2012 die drei untersten Schichten, Bitübertragung, Sicherung und Vermittlung (siehe Abb. 3.16). Die Anwendungsschicht ist die vierte und oberste Schicht des ISO/IEC 14543-3-10:2012-Modells. Die EnOcean Alliance definiert Kommunikationsprofile, die sog. *EnOcean Equipment Profiles*. Diese stellen die Interoperabilität zwischen Geräten unterschiedlicher Hersteller sicher und sorgen bspw. dafür, dass ein EnOcean-Lichtschalter des Herstellers A den Schaltaktor (an dem eine Lampe angeschlossen ist) des Herstellers B steuern kann.

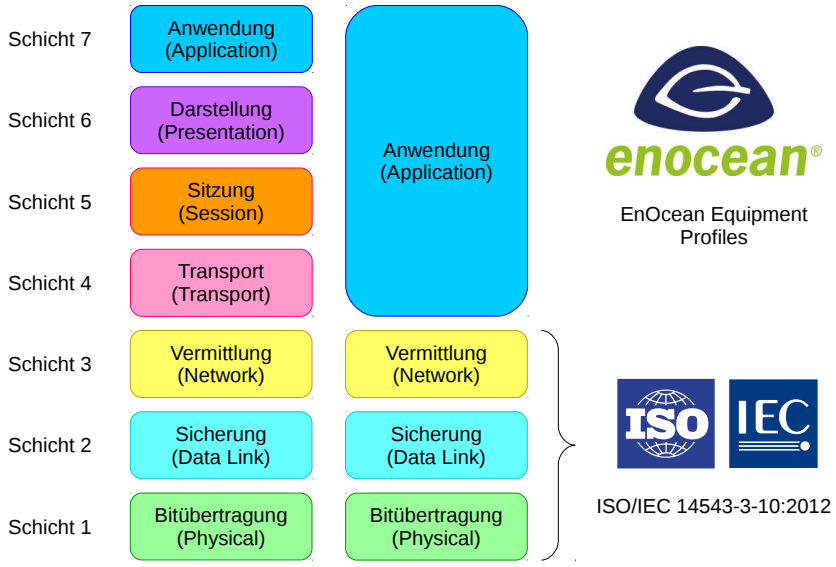


Abbildung 3.16.: Das ISO-OSI-Referenmodell und ISO/IEC 14543-3-10:2012 im Vergleich
Quelle: In Anlehnung an [35]

3.5.3. ISO/IEC 14543-3-10:2012-Netzwerktopologie

Die ISO/IEC 14543-3-10:2012-Netzwerktopologie lässt sich am ehesten mit einer Maschen-Netzwerktopologie vergleichen. Abb. 3.17 zeigt ein Beispiel einer ISO/IEC 14543-3-10:2012-Netzwerktopologie. Ein Sensor wie ein Lichtschalter bezieht seine Energie bspw. aus der Bewegungsenergie im Moment seiner Betätigung. In diesem Fall ist es sinnvoll, dass der Lichtschalter die unidirektionale Kommunikation nutzt und lediglich als Sender fungiert und ausschließlich im Moment seiner Betätigung ein Datenpaket versendet. Um die Größe eines Datenpaketes gering zu halten, wird nur die vier Bytes[15] lange ID des Senders (nicht aber die ID des Empfängers) übertragen (vgl. [37, S. 24]). Das Datenpaket richtet sich also nicht an einen bestimmten Empfänger, sondern an alle Empfänger, welche sich in Reichweite befinden (ähnlich wie beim Broadcast). Die Empfänger müssen dann anhand der ID des Senders entscheiden, ob sie das Datenpaket weiter verarbeiten oder nicht. Für Sensoren, welche ihre Energie auf ähnliche Weise wie der oben beschriebene Lichtschalter beziehen, gibt es ferner den speziellen Datenpaket-Typ *Switch Telegram*[16]. Dieser Datenpaket-Typ ist noch kürzer, da Informationen eingekürzt oder weggelassen werden. Die vier Bytes lange ID des Senders ist jedoch noch enthalten.

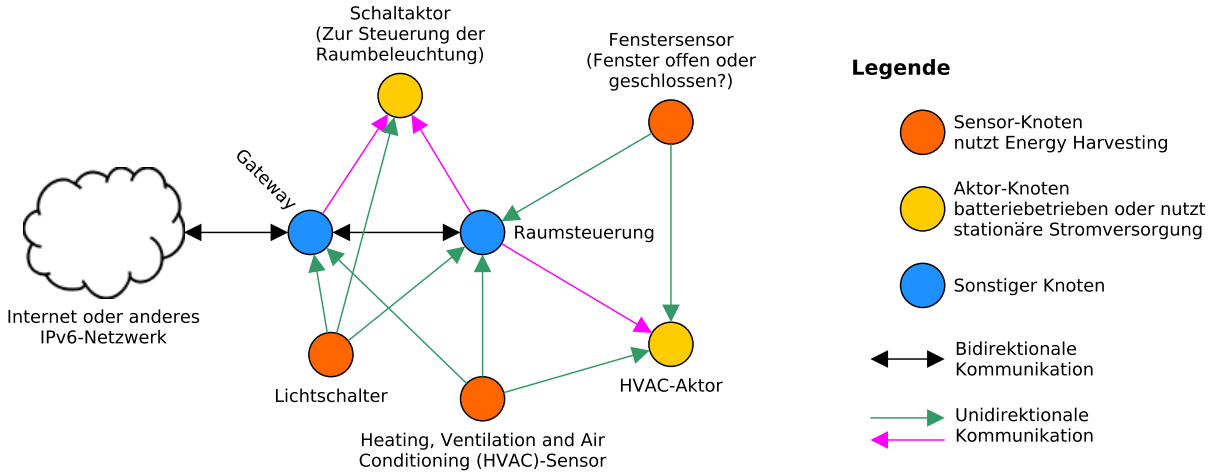


Abbildung 3.17.: Beispiel für eine ISO/IEC 14543-3-10:2012-Netzwerktopologie
Quelle: In Anlehnung an [29, S. 4]

Ein Schaltaktor betreibt typischerweise kein Energy Harvesting, denn meist sind die Lasten (z. B. Raumbeleuchtungen), welche er steuert, ohnehin an eine stationäre Stromversorgung angeschlossen. Da ein Schaltaktor nicht zwingend Daten senden muss, kann er die unidirektionale Kommunikation nutzen und als reiner Empfänger implementiert sein.

Mit einem Gateway können Sensoren und Aktoren in ein anderes Netzwerk, z. B. IPv6-Netzwerk, integriert werden. Da insbesondere Sensoren nicht immer ansprechbar sind, sondern die meiste Zeit in einem Energiesparmodus zubringen können, erzeugt das Gateway eine virtuelle Repräsentation eines jeden Knotens (Sensoren und Aktoren). Schickt ein Knoten aus einem IPv6-Netzwerk bspw. ein Datenpaket an einen Sensor, speichert das Gateway dieses zwischen, bis der Sensor das Datenpaket in seiner nächsten Wachphase

[15] Mit vier Bytes lassen sich 4.294.967.295 (über vier Milliarden) unterschiedliche Adressen darstellen. Dementsprechend viele Knoten kann ein Netzwerk umfassen.

[16] Dieser Datenpaket-Typ wurde so genannt, da er ursprünglich für die oben beschriebene Art von Lichtschaltern entwickelt wurde.

„abholt“ (ähnlich der indirekten Kommunikation von IEEE 802.15.4, siehe Kap. 3.1). Ein solches Datenpaket kann z. B. eine Steuerinformation beinhalten, welche einem Temperatursensor ein neues Intervall für die Übermittlung von Temperaturwerten mitteilt.

Ferner gibt es noch Repeater, welche dazu dienen die Reichweite zu erhöhen. Repeater werden in der Regel über eine stationäre Stromversorgung betrieben.

3.6. Z-Wave

Z-Wave ist ein Funkstandard mit Fokus auf die Hausautomatisierung bzw. Smart Home, welcher ursprünglich 2001 von zwei dänischen Ingenieuren entwickelt wurde (vgl. [69]). Mittlerweile hat die 2005 gegründete Z-Wave Allianz die Weiterentwicklung von Z-Wave übernommen. 2012 wurde die Bitübertragungs- und die Sicherungsschicht des Z-Wave-Modells (siehe Kap. 3.6.1) von der ITU-T[17] unter der Kennzeichnung G.9959 standardisiert.

Alle Z-Wave-Produkte, egal von welchem Hersteller, basieren auf einem SoC, welcher von den Firmen Sigma Designs und Mitsumi hergestellt wird. Es gibt verschiedene Typen dieser SoCs für unterschiedliche Anwendungsbereiche. Gemein ist allen SoCs, dass sie einen Mikrocontroller und eine Z-Wave-Funkeinheit in einem integrierten Schaltkreis vereinen. Um die Interoperabilität von Geräten verschiedener Hersteller zu fördern, definiert die Z-Wave Allianz Geräteklassen. Ein Gerät einer bestimmten Geräteklasse muss bestimmte Kommandos und Funktionen implementieren. Darüber hinaus darf jeder Hersteller in seine Geräte zusätzliche herstellerspezifische Kommandos und Funktionen implementieren.

3.6.1. ITU-T G.9959/Z-Wave im Vergleich mit dem ISO-OSI-Referenmodell

Abb. 3.18 stellt das vierschichtige ITU-T G.9959/Z-Wave-Modell dem siebenschichtigen ISO-OSI-Referenzmodell gegenüber. Wie auch bei IEEE 802.15.4 (siehe Kap. 3.1), ist die Sicherungsschicht des ITU-T G.9959/Z-Wave-Modells nochmals in die beiden Unterschichten MAC und LLC unterteilt.

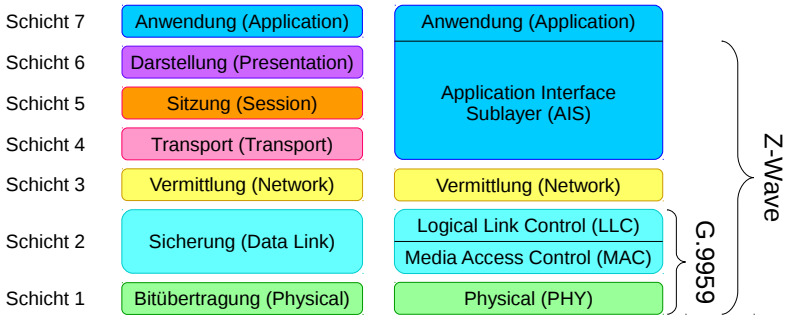


Abbildung 3.18.: Das OSI-Schichtenmodel und ITU-T G.9959/Z-Wave im Vergleich
Quelle: In Anlehnung an [38, S. 70]

[17] ITU-T bezeichnet den Standardisierungssektor für Telekommunikation der International Telecommunication Union (ITU).

Die oberste Schicht des ITU-T G.9959/Z-Wave-Modells ist in zwei Teile, in die Anwendungsschicht und den Application Interface Sublayer, unterteilt. Die anwendungsspezifischen Implementationen, welche die Hersteller in ihren Z-Wave-Geräten vornehmen, sind der Anwendungsschicht zuzuordnen. Der Application Interface Sublayer unterliegt der Z-Wave Allianz. Er vereinigt die Funktionen der Sitzungs-, Darstellungs- und Transport-Schicht des ISO-OSI-Referenzmodells. Die Vermittlungsschicht kümmert sich um das Routing, d. h. um die Weiterleitung von Datenpaketen zwischen Knoten. Im Folgenden wird näher auf die Schichten LLC, MAC und Physical Layer (PHY) eingegangen.

LLC- und MAC-Schicht

Gemäß des ISO-OSI-Referenzmodells kümmert sich die MAC-Subschicht um die Datenübertragung zwischen zwei oder mehreren (Broadcast) Knoten (vgl. [38, S. 27]). Über Prüfsummen wird sichergestellt, dass Datenpakete korrekt übertragen wurden. Ein Sender kann ein ACK vom Empfänger verlangen, wenn dieser ein Datenpaket erfolgreich empfangen hat. Erhält der Sender kein ACK, sendet er das Datenpaket erneut. Bis zu drei Sendeversuche werden unternommen, bis ein Übertragungsfehler an die höheren Schichten gemeldet wird. Die MAC-Schicht unterstützt Echtzeitanwendungen, mit allerdings eher lockeren Anforderungen an die Latenz.

Interessant ist die Unterstützung der MAC-Schicht für batteriebetriebene Knoten. Knoten können die meiste Zeit über in einem Energiesparmodus zubringen und diesen periodisch wieder verlassen, um eine Datenanfrage an andere Knoten zu senden. Batteriebetriebene Knoten, welche jederzeit mit vertretbarer Latenz erreichbar sein müssen, verwenden den sog. *Frequently Listening Modus*. Auf den Frequently Listening Modus wird in Kap. 3.6.3 genauer eingegangen.

Aufgabe der LLC-Subschicht ist es, sich um die Fragmentierung von Datenpaketen zu kümmern und dafür zu sorgen, dass die Fragmentierung transparent für die höheren Schichten ist (vgl. [38, S. 56]). Die LLC-Subschicht kann zwischen 6LoWPAN- und Standard-Z-Wave-Datenpaketen unterscheiden, damit 6LoWPAN-Datenpaketen gesondert behandelt werden können. Dies ist ein Hinweis darauf, dass ITU-T G.9959 bereits die Integration in das IoT vorsieht.

PHY-Schicht

Die PHY-Schicht definiert unter anderem die Modulationsverfahren, Datenraten, Funkfrequenzen und das Format des Datenrahmens (vgl. [38, S. 8]).

ITU-T G.9959 setzt, entsprechend der Regularien im Einsatzgebiet, auf ISM- und SRD-Frequenzbänder zwischen 850 und 950 MHz, welche keiner gesonderten Lizenz bedürfen. Damit geht ITU-T G.9959 respektive Z-Wave dem potenziell stark ausgelasteten 2,4-GHz-Band aus dem Weg und es können größere Reichweiten mit weniger Energie erzielt werden, da es sich um niedrigere Frequenzbereiche handelt. In Europa werden bspw. die SRD-Frequenzen 868,4 und 869,85 MHz verwendet (vgl. [9, S. 3]). Die Sendeleistungen müssen die Regularien im Einsatzgebiet einhalten. Typische Reichweiten für Z-Wave-Geräte sind 150 m im Freien und 40 m in Gebäuden.

Es sind Datenraten von 9,6, 40 und 100 kbit/s möglich. Diese hängen unter anderem von der Funkfrequenz ab. In Europa beträgt die Datenrate 9,6 oder 40 kbit/s im 869,85 MHz-Band und 100 kbit/s im 868,4 MHz-Band.

Die maximale Größe eines Datenpaketes der PHY (PPDU) ist unter anderem abhängig von der Datenrate und kann zwischen 66 und 211 Bytes liegen (vgl. [38, S. 15 ff.]). Die maximale Größe der Nutzdaten (Daten der übergeordneten Schichten) beträgt 170 Bytes bei einer Datenrate von 100 kbit/s und 64 Bytes bei niedrigeren Datenraten.

3.6.2. ITU-T G.9959/Z-Wave-Netzwerktopologie

ITU-T G.9959/Z-Wave unterstützt eine Art Maschen-Netzwerktopologie (vgl. [69]). Das Netzwerk lässt sich flächenmäßig vergrößern, indem Knoten als **Router** dienen, um Datenpakete weiterzuleiten. Aber nicht alle Knoten fungieren als Router, sondern typischerweise nur solche, mit einer stationären Stromversorgung, da batteriebetriebene Knoten die meiste Zeit in einem Energiesparmodus zubringen, währenddessen sie für andere Knoten nicht erreichbar sind. Jeder Knoten in einem Z-Wave-Netzwerk ist logisch genau einer sog. *Domain* zugeordnet (vgl. [38, S. 4 ff.]). Zur Identifizierung der zugehörigen Domain, ist jedem Knoten eine vier Bytes große *HomeID* zugeteilt. Das erlaubt die Koexistenz von theoretisch mehr als vier Milliarden Z-Wave-Netzwerken. Zudem ist jeder Knoten durch eine ein Byte lange NodeID innerhalb einer Domain eindeutig gekennzeichnet. Eine Domain kann bis zu 232 Knoten umfassen.[18] Jede Domain benötigt genau einen *Domain Master*. Er ist für die Koordination des Netzwerkes zuständig. Ferner kann es noch eine sog. *Interdomain Bridge* in einem Z-Wave-Netzwerk geben. Interdomain Bridges verbinden mehrere Domains miteinander, damit Knoten unterschiedlicher Domains miteinander kommunizieren können. Interdomain Bridges können eine Domain auch mit einem Netzwerk ganz anderer Art, z. B. mit einem drahtgebundenen IP-Netzwerk, verbinden. Ein solches Netzwerk, also ein Nicht-Z-Wave-Netzwerk, wird im Z-Wave-Jargon als *Alien Domain* bezeichnet.

3.6.3. Der Frequently Listening Modus

Wie in Kap. 3.6.1 bereits beschrieben, hilft der Frequently Listening Modus einem Knoten dabei Energie zu sparen, bei gleichzeitig relativ niedriger Antwortzeit (Latenz). Will ein Sender Daten an einen Empfänger senden, welcher den Frequently Listening Modus nutzt, muss der Sender zuvor sog. *Beam-Pakete* (eng. *Beam Frames*) aussenden, um die „Aufmerksamkeit“ des Empfängers zu erwecken (vgl. [38, S. 47 ff.]). Der Empfänger, also ein Knoten, welcher den Frequently Listening Modus nutzt, überprüft den Funkkanal periodisch auf Beam Frames. Wurde kein Beam Frame empfangen oder war der Beam Frame für einen anderen Knoten bestimmt, kann der Knoten unverzüglich wieder in den Energiesparmodus wechseln. Anderenfalls bleibt der Knoten wach, um die Daten des Senders zu empfangen. Ein Beam Frame ist so beschaffen, dass ein Knoten lediglich die ersten 9 oder 21 Bytes (je nach verwendetem Funkprofil) auswerten muss, um ein Beam Frame von einem anderen ITU-T G.9959/Z-Wave-Datenpaket zu unterscheiden. Dies erlaubt dem Knoten, schneller wieder in den Energiesparmodus zu wechseln. Hat der Knoten ein

[18] Mit einer ein Byte großen NodeID könnten bis zu 256 Knoten adressiert werden. Einige NodeIDs sind jedoch reserviert und können daher nicht verwendet werden.

Beam Frame erkannt, muss er auch die restlichen ein oder zwei[19] Bytes auswerten, um zu bestimmen, ob der Beam Frame für ihn bestimmt ist. Ein Beam Frame hat somit eine Größe zwischen 10 und 23 Bytes.

Beim Senden der Beam Frames wird zwischen dem fortlaufenden und dem fragmentierten Beaming unterschieden (vgl. [38, S. 49 ff.]).

Fortlaufendes Beaming

Beim fortlaufenden Beaming (in ITU-T G.9959 *Continuous Beam Format* genannt) werden fortlaufend und lückenlos Beam Frames für eine bestimmte Dauer gesendet. Das lückenlose Senden verhindert, dass ein anderer Knoten den Sendevorgang unterbricht. ITU-T G.9959 unterscheidet zwischen langen und kurzen Beams. Die Dauer eines langen Beams beträgt maximal 1160 ms und die eines kurzen Beams maximal 300 ms. Als Empfehlung gibt ITU-T G.9959 für einen langen Beam eine Dauer von 1100 ms und für einen kurzen Beam eine Dauer von 275 ms an. Nach dem Sendevorgang der Beam Frames folgt unmittelbar das Datenpaket. Der fortlaufende Sendevorgang soll bei einer Datenrate von 40 kbit/s verwendet werden. Ein Knoten, welcher den Frequently Listening Modus nutzt, muss in der Lage sein, einen solchen Beam zu detektieren, d. h. er muss den Kanal oft genug auf Beam Frames überprüfen.

Fragmentiertes Beaming

Beim fragmentierten Beaming werden mehrere Beam Frames zu einem sog. *Beam Fragment* zusammengefasst (siehe Abb. 3.19). Die Beam Frames innerhalb eines Beam Fragments werden, wie beim fortlaufenden Beaming, lückenlos hintereinander gesendet. Die Dauer eines Beam Fragments beträgt zwischen 110 ms und 115 ms.

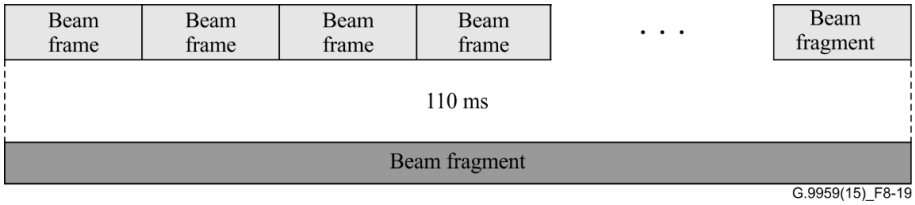


Abbildung 3.19.: ITU-T G.9959/Z-Wave Beam Fragment
Quelle: [38, S. 49]

Es werden mehrere solcher Beam Fragments nacheinander im Abstand von 190 ms bis 200 ms und jeweils gefolgt von einer kurzen Pause, gesendet (siehe Abb. 3.20). Die Gesamtdauer des fragmentierten Beamings beträgt 3 s. Im Anschluss daran wird unmittelbar das Datenpaket gesendet. Der Ziel-Knoten, also der Knoten, für welchen die Beam Frames bestimmt sind, kann den Sendevorgang des fragmentierten Beamings jederzeit unterbrechen, indem er den Erhalt eines Beam Frames mit einem ACK bestätigt. Das ACK muss innerhalb der Pausenzeit zwischen zwei Beam Fragments gesendet werden. Der Sender kann das fragmentierte Beaming daraufhin früher abbrechen und bereits das Datenpaket senden.

[19] Das letzte Byte eines Beam Frames, *HomeID Hash* genannt, ist optional. Es identifiziert die Domain, ist jedoch viel kürzer als die vier Bytes lange HomeID.

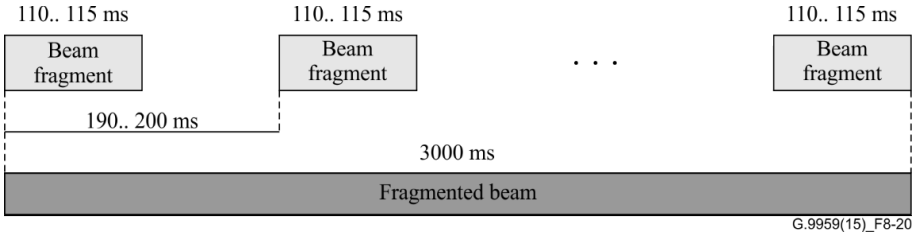


Abbildung 3.20.: Fragmentiertes Beaming von ITU-T G.9959/Z-Wave
Quelle: [38, S. 49]

Der ITU-T G.9959 empfiehlt für Knoten im Frequently Listening Modus ein Aufwachintervall von 100 ms, 150 ms, 250 ms, 500 ms, 700 ms oder 900 ms. Dabei ist das gewählte Intervall stets ein Kompromiss aus Energiesparen und möglichst kurzen Antwortzeiten.

3.7. Bluetooth Smart

Bluetooth Smart, auch bekannt als Bluetooth Low Energy, ist ein optionaler Bestandteil von Bluetooth, seit der, im Dezember 2009 herausgegebenen, Bluetooth-Spezifikation 4.0 (vgl. [46]). Weiterentwickelt und herausgegeben wird die Bluetooth-Spezifikation von der Bluetooth Special Interest Group (SIG). Die aktuelle Bluetooth-Spezifikation liegt in der Version 5.0 vor und wurde im Dezember 2016 herausgegeben. Obwohl das herkömmliche Bluetooth, auch *Bluetooth Classic* genannt, und Bluetooth Smart in derselben Spezifikation gepflegt werden, sind die untersten Schichten beider Technologien so grundverschieden, dass keine Kompatibilität besteht. Geräte, welche Bluetooth Classic der Version 4.0 oder höher unterstützen, müssen nicht zwangsläufig auch Bluetooth Smart unterstützen und umgekehrt. Es gibt zwei Kategorien von Bluetooth Smart Geräten: *Bluetooth Smart Ready* und *Bluetooth Smart*. Bluetooth Smart Ready Geräte unterstützen Bluetooth Classic und Smart. Typischerweise gehören Geräte wie Notebooks, Smartphones und Tablets zu dieser Kategorie. Aufgrund der Unterschiede zwischen Bluetooth Classic und Smart werden unterschiedliche Funkeinheiten benötigt, welche aber in der Praxis oft in einem integrierten Schaltkreis vereint sind. *Bluetooth Smart* Geräte hingegen unterstützen ausschließlich Bluetooth Smart. Dabei handelt es sich oft um sog. *Wearables*, wie Smartwatches und Fitness-Tracker, theoretisch aber auch um jede andere Art von Sensor und Aktor. Bluetooth Smart ist besonders geeignet für Geräte, welche nicht kontinuierlich Daten übertragen müssen, bei denen es aber dafür auf eine lange Batterielaufzeit ankommt (vgl. [47]). Damit soll es auch ideal für IoT-Anwendungen geeignet sein.

3.7.1. Bluetooth Smart im Vergleich mit dem ISO-OSI-Referenmodell

Die untersten drei Schichten werden oft zu einem *Controller* und die obersten Schichten, außer der Anwendungsschicht, zu einem *Host* genannten Subsystem gruppiert (siehe Abb. 3.21). Die Schicht *Host Controller Interface* ist optional und definiert eine einheitliche Schnittstelle zwischen verschiedenen Bluetooth Hosts und Controllern. Im Folgenden sollen die beiden Schichten PHY und Link Layer näher betrachtet werden, da diese im Hinblick auf den Energieverbrauch eine große Rolle spielen.

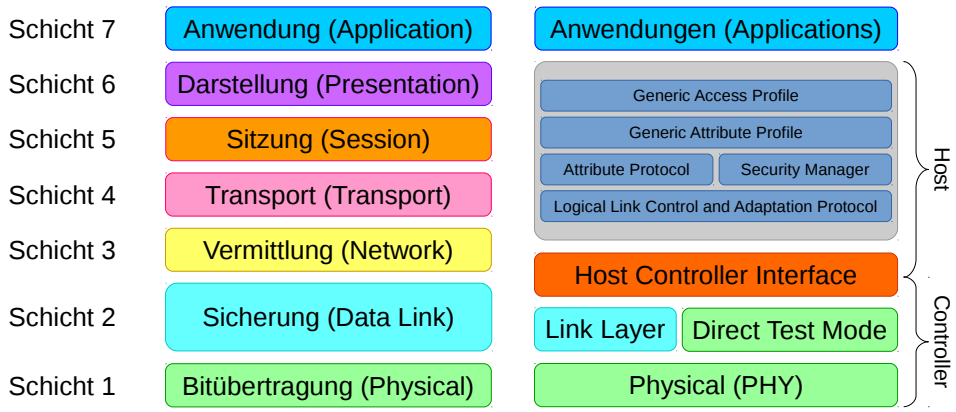


Abbildung 3.21.: Das ISO-OSI-Referenmodell und Bluetooth Smart im Vergleich
Quelle: In Anlehnung an [47]

PHY-Schicht

Bluetooth Smart verwendet das 2,4-GHz-Band und unterteilt dieses in 40 Kanäle (vgl. [48, S. 2535]).[20] Jede Bluetooth Smart Funkeinheit muss eine Datenrate von 1 Mbit/s unterstützen (vgl. [48, S. 169]). Wird eines der beiden optionalen Fehlerkorrekturverfahren[21] verwendet, sinkt die Datenrate auf 500 kbit/s oder 125 kbit/s. Zum Preis eines aufwändigeren aber optionalen Modulationsverfahrens, kann auch eine Datenrate von 2 Mbit/s erreicht werden. Für letztere stehen keine Fehlerkorrekturverfahren zur Verfügung.

Für die Ausgangsleistung des Senders sieht die Bluetooth-Spezifikation Sendeleistungen zwischen 0,01 mW (-20 dBm) und 100 mW (20 dBm) vor. Die örtlichen Regularien haben jedoch immer Vorrang. Ältere Bluetooth-Spezifikation als Version 5.0 sahen noch eine maximale Sendeleistung von 10 mW (10 dBm) vor. Für den Empfänger sieht die Bluetooth-Spezifikation eine Sensivität von -70 dBm oder besser vor. Bei Verwendung eines der beiden Fehlerkorrekturverfahren ist die Sensivität mit mindestens -75 dBm oder -82 dBm (für das bessere Verfahren) angegeben.

Mit einer Sendeleistung von 0,01 mW dürfte die Reichweite lediglich wenige Zentimeter betragen. Dies ist für Anwendungen im Nahbereich interessant. Bluetooth könnte damit eine Art Konkurrenz zu Technologien wie Near Field Communication (NFC) darstellen. Bei maximaler Sendeleistung von 100 mW beträgt die Reichweite ca. 100 m im Freien und lässt sich unter Verwendung der Fehlerkorrekturverfahren noch weiter steigern (vgl. [71]).

Link Layer

Der Link Layer stellt einen logischen Verbindungskanal zwischen zwei Geräten her. Von den 40 verfügbaren physischen Kanälen werden drei zur initialen Verbindung (sog. *Primary Advertising Channels*) und 37 für die eigentliche Datenübertragung (sog. *Secondary Advertising Channels* oder *Data Channels*) verwendet (vgl. [48, S. 2560]). Um robuster gegenüber Störungen zu sein und um andere Funkdatenübertragungen nicht zu stören,

[20] Bluetooth Classic nutzt ebenfalls das 2,4-GHz-Band, unterteilt dieses aber in 79 Kanäle.
[21] Fehlerkorrekturverfahren machen die Datenübertragung robuster, indem Fehler bei der Datenübertragung bis zu einem gewissen Grad korrigiert werden können. Je nach Güte des Fehlerkorrekturverfahrens sinkt dann aber die tatsächlich nutzbare Datenrate.

verwendet Bluetooth Channel-Hopping (siehe Kap. 2.2). Kanäle, welche besonders gestört sind, können von der Verwendung ausgeschlossen werden. Dies verleiht Bluetooth gute Koexistenz-Eigenschaften.

Die Nutzdaten, welche in einem Link Layer Datenpaket eingekapselt sind, können eine Größe von bis zu 257 Bytes haben (vgl. [48, S. 2562]). Die Übertragung eines Datenpaketes dauert zwischen 40 μs (kleinstmögliches Datenpaket bei einer Datenrate von 2 Mbit/s) und 2120 μs (größt mögliches Datenpaket bei einer Datenrate von 1 Mbit/s). Bei Verwendung eines der Fehlerkorrekturverfahren beträgt eine Paketdauer zwischen 462 μs und 17040 μs.

Die Datenübertragung zwischen zwei Geräten, welche im Link Layer spezifiziert wird, wird in Kap. 3.7.2 näher beschrieben.

3.7.2. Verbindungsaufbau und Datenübertragung

Geräte, welche eine Verbindung zu einem anderen Gerät aufbauen möchten, werden *Initiators* genannt. Ein Initiator wartet darauf, dass das Zielgerät, der sog. *Advertiser*, auf einem der drei Primary Advertising Channels ein sog. *Advertising Event* aussendet. Wurde ein Advertising Event empfangen, antwortet der Initiator auf dieses mit einer Verbindungsanfrage. Akzeptiert der Advertiser die Verbindungsanfrage, nimmt er die Rolle des *Slaves* und der Initiator die Rolle des *Masters* ein. Slave und Master können dann innerhalb der sog. *Connection Events* Daten austauschen. Jedes Connection Event findet auf einem anderen Kanal statt. Der Vorgang ist in Abb. 3.22 illustriert.

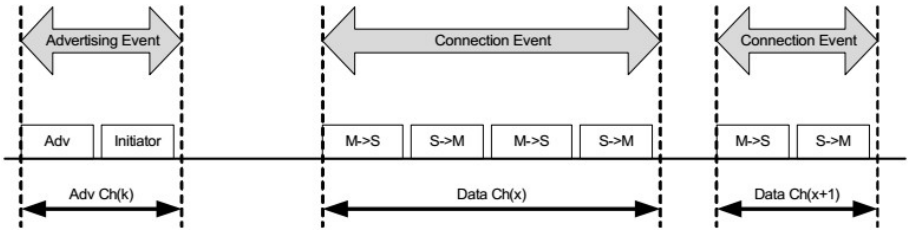


Abbildung 3.22.: Verbindungsaufbau und Datenaustausch zwischen zwei Bluetooth Smart Geräten

Quelle: [48, S. 171]

Geräte in der Rolle des Advertisers bzw. Slaves sind häufig batteriebetriebene Geräte, wie Wearables, Eingabegeräte für Computer, wie Mäuse und Tastaturen, Sensoren, Aktoren usw.

Ein Advertising Event besteht aus ein oder mehreren Advertising-Datenpaketen eines bestimmten Typs (im Bsp. vom Typ ADV_IND), welche nacheinander im Abstand von maximal 10 ms auf allen genutzten Primary Advertising Channels gesendet werden. Abb. 3.23 veranschaulicht ein Advertising Event, welches alle drei Primary Advertising Channels nutzt. Empfängt ein Advertiser unmittelbar nach einem Advertising-Datenpaket eine Verbindungsanfrage von einem Initiator, kann er das Advertising Event vorzeitig beenden und die Verbindungsanfrage beantworten. Mehrere Advertising Events werden im *Low Duty Cycle Mode* in Intervallen von 20 ms bis 10,25 s gesendet. Ferner gibt es noch

das Extended Advertising Event, bei welchem ein Intervall sogar über 7 Tage betragen kann. Aus längeren Intervallen der Advertising Events resultiert eine längere Batterielaufzeit, allerdings auch eine längere Zeit zum Verbindungsaufbau.

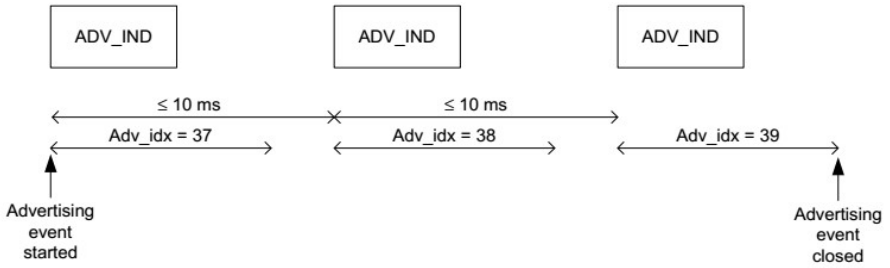


Abbildung 3.23.: Bluetooth Smart Advertising Event
Quelle: [48, S. 2614]

Wurde eine Verbindung aufgebaut, entscheiden die Parameter für das Connection Event über die tatsächlich nutzbare Datenrate und Latenz der Verbindung und somit auch über die Batterielaufzeit. Connection Events können in einem Intervall von 7,5 ms bis 4 s erfolgen (vgl. [48, S. 2638]). Die *Slave Latency* gibt an, wie viele Connection Events der Slave auslassen darf und kann maximal 499 betragen. Dadurch kann der Slave Energie sparen, wenn innerhalb eines Connection Events keine Notwendigkeit besteht, Daten zu übertragen. Eine bestehende Verbindung wird als verloren gegangen betrachtet, wenn innerhalb des *Supervision Timeouts* keine Daten übertragen wurden. Das Supervision Timeout beträgt eine Zeit zwischen 100 ms und 32 s. D. h. also, ein Slave muss spätestens nach 32 s ein Connection Event beantworten (auch, wenn diese Zeit bei einem Connection Event Intervall von 4 s und einer Slave Latency von 499 theoretisch auf über 33 Minuten ausgedehnt werden könnte).

Das Gerät in der Master-Rolle ist für die Initiierung jedes einzelnen Connection Events zuständig. D. h., der Slave sendet immer nur Daten nach Aufforderung durch den Master, wie in Abb. 3.22 dargestellt. Daten, z. B. Sensorwerte, können jedoch auch mit den Advertising-Datenpaketen gesendet werden, d. h. ohne dass zuvor eine Verbindung aufgebaut wurde. Das Advertising-Datenpaket vom Typ ADV_IND bietet dafür bis zu 31 Bytes Platz (vgl. [48, S. 2569]).

3.7.3. Bluetooth Smart Netzwerktopologien

Hat ein Master eine Verbindung zu einem oder mehreren Slaves aufgebaut, wird dies in der Bluetooth-Terminologie als *Piconet* bezeichnet. Ein Piconet kann bis zu sieben aktive Slaves umfassen (vgl. [48, S. 354]). Ein Gerät kann die Rollen Master und Slave gleichzeitig einnehmen. Direkte Verbindungen zwischen zwei Slaves innerhalb eines Piconets sind nicht möglich. Mehrere Piconets bilden ein sog. *Scatternet*. Ein Piconet ist Bestandteil eines Scatternets, wenn es mindestens ein Gerät enthält, welches an mehr als einem Piconet partizipiert. Ferner kann ein Gerät die Slave-Rolle in mehreren Piconets gleichzeitig aber nur die Master-Rolle für ein einzigen Piconet einnehmen (vgl. [48, S. 468]). Der Grund dafür ist, dass der Master das Channel Hopping kontrolliert. So würden alle Piconets, welche der Master kontrolliert, dieselbe Channel Hopping-Sequenz verwenden und

wären damit wieder ein einziges Piconet. Abb. 3.24 zeigt Beispiele für Bluetooth Smart Netzwerktopologien.

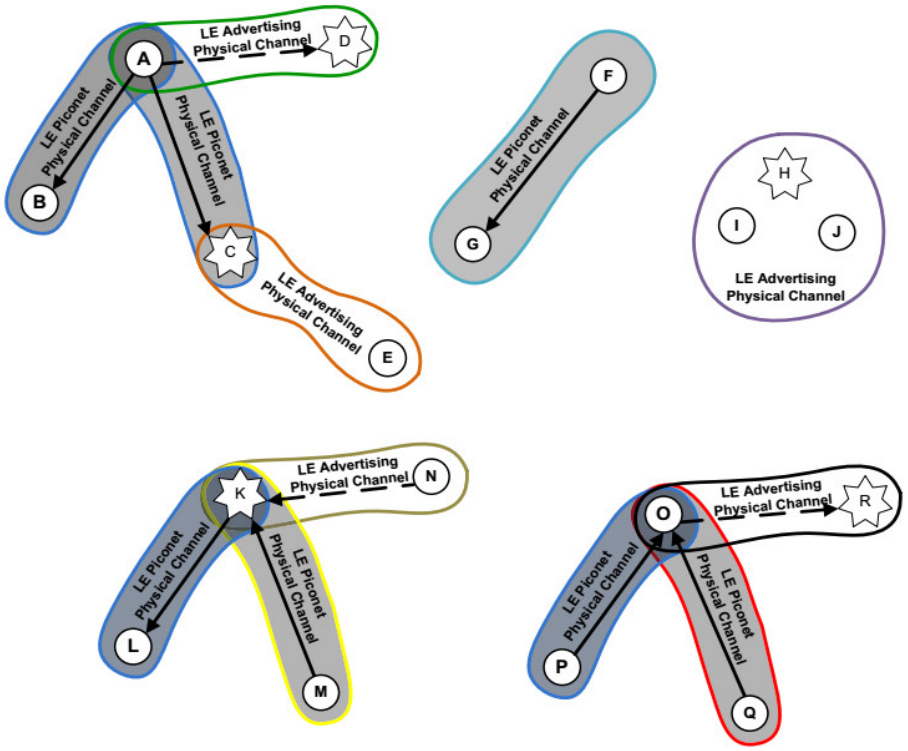


Abbildung 3.24.: Bluetooth Smart Netzwerktopologien
Quelle: [48, S. 228]

In Abb. 3.24 zeigen durchgezogene Pfeile vom Master zum Slave. Eine Verbindungsinitiierung wird durch einen gestrichelten Pfeil dargestellt, welcher vom Initiator zum Advertiser zeigt. Geräte, welche als Stern dargestellt sind, sind Advertiser und senden Advertising Events aus.
Gerät A ist ein Master und bildet mit den Geräten B und C als Slaves ein Piconet. F (Master) und G (Slave) stellen die einfachste Form eines Piconets dar. K befindet sich in einem Scatternet. Es ist Master für den Slave L und bildet mit diesem ein Piconet. Gleichzeitig ist K aber auch ein Slave und bildet mit M als Master ein weiteres Piconet. O befindet sich ebenfalls in einem Scatternet. Es ist Slave für die Master P und Q und bildet mit diesen jeweils ein Piconet.
Dargestellt sind auch einige Verbindungsinitiierungen. Z. B. versucht A (Initiator) eine Verbindung zu D (Advertiser) herzustellen. H ist ein Advertiser und I und J sind sog. *Scanner*. Scanner verhalten sich passiv und empfangen Advertising-Datenpakete, um Informationen über sich in der Nähe befindliche Advertiser zu sammeln.

Ein Piconet kann als Punkt-zu-Punkt-Topologie (ein Master und ein Slave) oder Stern-Topologie (ein Master und mehrere Slaves) aufgefasst werden. Zur flächenmäßigen Ausdehnung eines WSNs auf Basis von Bluetooth, könnten Scatternets dienen. Jedoch macht die Bluetooth-Spezifikation weder Angaben darüber, wie Scatternets geformt werden, noch wie das Routing (welches Sache der Anwendungsschicht ist) zu geschehen hat (vgl. [48, S. 226]). Dies war daher Anlass zahlreicher Forschungsarbeiten. Die Autoren Methfessel u. a. entwickelten einen SFX genannten Algorithmus, welcher eine Baum-Netzwerktopologie

aus Scatternets aufbaut (siehe [41]). Auf den einzelnen Knoten kam Linux als Betriebssystem zum Einsatz. Das Routing wurde via des TCP/IP-Stacks von Linux realisiert. Auf dieser Grundlage wurde ein WSN, bestehend aus 39 Knoten, welches eine Photovoltaikanlage über mehrere Monate hinweg überwachte, erfolgreich getestet. In der Praxis jedoch, scheinen Scatternets bisher kaum Anwendung zu finden.

3.8. Zusammenfassung

Die in Kap. 3 beschriebenen Funkstandards erreichen einen energiesparenden Betrieb vorallem durch Minimalismus. D. h. die Sendedauer von Datenpaketen wird möglichst kurz gehalten, was durch einfache Datenformate und der Einsparung von Informationen erzielt wird. Die Konsequenz des Minimalismus sind jedoch geringe Datenraten (siehe Tabelle A.3 in Anlage A.1). Am ausgeprägtesten ist der Minimalismus bei ZigBee Green Power und EnOcean. Beide Standards nutzen Energy Harvesting und sind darum auf einen sehr sparsamen Betrieb angewiesen. Daher verwenden sie sehr kurze Datenpakete und verzichten teilweise auf den Austausch von ACKs und übertragen Daten nur unidirektional. Die (maximale) Datenrate von EnOcean beträgt gar nur 125 kbit/s. Als weitere Maßnahme zum Energiesparen reduzieren viele Funkstandards die Reichweite auf das Nötigste, denn die zum Senden benötigte Energie wächst superlinear mit steigender Entfernung (vgl. [58]). EnOcean und Z-Wave setzen auf die niedrigen Frequenzbereiche unterhalb von 1 GHz, da Funkwellen mit niedrigeren Frequenzen bessere Ausbreitungseigenschaften haben, d. h. es lässt sich dieselbe Reichweite mit einem geringeren Energieaufwand überbrücken.

IEEE 802.15.4 ist ein beliebter Funkstandard, auf welchen viele andere Standards, wie 6LoWPAN, fußen. Viele große Betriebssysteme für WSN-Knoten, wie die Open Source Betriebssysteme Contiki und RIOT, nutzen 6LoWPAN und ermöglichen so die Entwicklung von Anwendungen für das Internet of Things.

Bluetooth Smart ist insofern eine attraktive Technologie, als dass sie weit verbreitet und die Hardware damit sehr günstig ist. Zudem flossen sicherlich einige der Erfahrungswerte, welche man mit Bluetooth Classic gesammel hat, in Bluetooth Smart ein. Bluetooth Classic gibt es schon sehr lange auf dem Markt, weshalb es ausgereift und robust ist. Moderne Smartphones oder tragbare Computer sind meist Bluetooth Smart Ready und können daher als Handhelts in das WSN integriert werden. Günstige Minicomputer wie der Raspberry Pi 3 verfügen ab Werk über einen Ethernet-Anschluss, WLAN, sind Bluetooth Smart Ready und eignen sich damit hervorragend als Border-Router bzw. PAN-Koordinator. Ein Wermutstropfen bleibt jedoch die fehlende Unterstützung für Netzwerktopologien, welche über die Stern-Topologie hinausgehen. Zwar sind komplexere Netzwerktopologien theoretisch möglich, diese wurden bisher allerdings nur prototypisch realisert (siehe [41]).

4. MAC-Protokolle

Um zu verstehen wie MAC-Protokolle dabei helfen können Energie zu sparen, ist es wichtig, vorab den Begriff Radio Duty Cycling (RDC) zu definieren.

RDC ist das zyklische Aktivieren und Deaktivieren der Funkeinheit, wobei der Inaktiv-Zustand (Funkeinheit deaktiviert) meist einen signifikanten Teil der Zeit einnimmt. Der Anteil der Aktivzeit wird in Prozent angegeben. Ein RDC-Wert von 1 % sagt bspw. aus, dass die Funkeinheit während einer Betriebszeit von einer Stunde, insgesamt 36 s aktiv und die restliche Zeit inaktiv ist. Es gibt zwei Motivationen RDC einzusetzen:

- Zur Einhaltung von Regularien: Für die Nutzung einiger ISM- und SRD-Frequenzbänder bspw., darf die Gesamtsendedauer, gemessen über eine Betriebsdauer von einer Stunde, nur maximal 0,1 % der Zeit betragen (siehe z. B. [14]).
- Zum Sparen von Energie.

In diesem Kapitel geht es um letztere Motivation, dem Sparen von Energie. Die Funkeinheit ist oftmals die Komponente des Gesamtsystems, mit dem höchsten Energieverbrauch, wie auch schon in Kap. 2.3 für den Funk-Chip ATmega 128RFA1 deutlich gemacht wurde. Gängige Funk-Chips kennen folgende Betriebsmodi, welche sich auf den Energieverbrauch auswirken (vgl. [23, S. 133 ff.]):

Transmit (Senden) Daten werden gesendet. Dieser Modi hat meist den höchsten Energieverbrauch.

Receive (Empfangsbereitschaft) Daten werden empfangen bzw. die Funkeinheit ist in Empfangsbereitschaft.

Idle (Inaktiv) Funkeinheit ist eingeschaltet, jedoch nicht empfangsbereit.

Sleep (Schlafen) Die Funkeinheit befindet sich in einem Schlafmodus. Der Energieverbrauch ist hier am niedrigsten, jedoch dauert der Übergang in den Receive- und Transmit-Modus länger als vom Idle-Modus aus.

Tabelle 4.1 führt beispielhaft die Energieverbräuche einiger Funk-Chips in den verschiedenen Betriebsmodi auf. Bei allen Chips handelt es sich um SoCs, welche Mikrocontroller und Funkeinheit in einem integrierten Schaltkreis vereinen. Der Energieverbrauch im Transmit-Modus hängt von der, bei allen Chips variabel einstellbaren, Ausgangsleistung ab und bezieht sich auf den in der Spalte TX_PWR angegebenen Wert. Der ATmega 128RFA1 von Atmel und der CC2530 von Texas Instruments sind IEEE 802.15.4-kompatible Chips. Der nRF51822 von Nordic Semiconductor ist ein Bluetooth Smart-Chip und der ZM5101 von Sigma Designs schließlich, ist ein Z-Wave-Chip.

Wie Tabelle 4.1 zu entnehmen ist, kann also ein erheblicher Anteil an Energie gespart werden, wenn die Funkeinheit möglichst oft und möglichst lange im Idle-Modus verharrt.

Noch dramatischer wird der Energiespareffekt, wenn nicht nur der Idle-Modus der Funkeinheit genutzt wird, sondern weite Teile des Gesamtsystems während der Inaktivzeit abgeschaltet werden. Für die SoCs in Tabelle 4.1 entspricht dieser Zustand dem Sleep-Modus.

	CPU	**Sleep**	**Idle**	**Receive**	**Transmit**	**TX_PWR**
ATmega 128RFA1	16 MHz	4,1 mA / 250 nA[a]	4,7 mA	16,6 mA	18,6 mA	3,5 dBm
CC2530	16 MHz	1 μA	3,4 mA	24 mA	29 mA	1 dBm
nRF51822	16 MHz	0,6 μA	4,4 mA	17,4 mA	20,4 mA	4 dBm
ZM5101	–	1 μA	–	32 mA	32 mA	0 dBm

[a] Deep-Sleep-Modus

Tabelle 4.1.: Die Betriebsmodi einiger Funk-Chips und ihre Energieverbräuche

MAC-Protokolle steuern den Zugriff auf das Medium. Die Implementierungen von RDC-Algorithmen sind daher den MAC-Protokollen bzw. der MAC-Schicht zuzuordnen. In Kap. 3 wurden bereits im Rahmen der Funkstandards MAC-Protokolle vorgestellt. Z. B. die direkte und indirekte Datenübertragung von IEEE 802.15.4, jeweils im Non-Beacon-Enabled-Mode und Beacon-Enabled-Mode mit Superframe (siehe Kap. 3.1). Dieses Kapitel behandelt weitere MAC-Protokolle, welche es den Knoten eines WSNs erlauben, möglichst energiesparend zu operieren, aber gleichzeitig quasi verzögerungsfrei auf Anfragen reagieren zu können. Dies ist eine wichtige Schlüsseleigenschaft, insbesondere für Aktoren und Router, welche sich abseits von einer stationären Stromversorgung befinden, das Verlegen von Kabeln aber aus unterschiedlichen Gründen (siehe Kap. 1.1) nicht gewünscht ist. Als Beispiel können im Bereich Smart Home Heizkörperthermostate[1] angeführt werden. Zwar sind Steckdosen in den meisten Räumen verfügbar, ein Kabel würde aber von den Benutzern als störend empfunden werden. Ein weiteres Beispiel ist der Einsatz eines großflächigen WSNs, z. B. auf dem Meer, in der Landwirtschaft (siehe Precision Agriculture in Kap. 1.1.1) oder auf dem Industriegelände. Hier werden Router-Knoten[2] zur Ausdehnung des Netzwerkes eingesetzt. Hier ist es wünschenswert, das Wechselintervall der Batterien dieser Router so weit wie möglich auszudehnen.

Ein energiesparendes MAC-Protokoll versucht den RDC-Wert so niedrig wie möglich zu halten. Dabei gilt es einen Kompromiss zwischen dem Energiesparen auf der einen und einer niedrigen Latenz und einen hohen Datendurchsatz auf der anderen Seite zu finden. Wenn das Heizkörperthermostat einen neuen Sollwert für die Temperatur übernehmen soll, stört eine Latenz von einer Minute nicht. Wenn der Benutzer aber das Licht einschaltet, ist für den Schaltaktor eine Reaktionszeit von einigen hundert Millisekunden gerade noch tolerierbar. Der RDC-Wert wird gerne als neutrales Maß, um den Energieverbrauch eines Knotens zu messen und MAC-Protokolle hinsichtlich ihres Energieverbrauchs miteinander zu vergleichen, verwendet.

[1] Hierbei handelt es sich um Sensor-Aktor-Kombinationen, welche nicht nur die Temperatur messen, sondern auch das Ventil des Heizkörpers regeln, damit sich eine gewünschte Solltemperatur einstellt.

[2] In der Praxis kann es sich anbieten, Knoten zu entwerfen, welche Router und Sensor und/oder Aktor in einem Gerät vereinen.

4.1. Eigenschaften von MAC-Protokollen

4.1.1. Synchron und Asynchron

Bei synchronen MAC-Protokollen synchronisieren die Knoten ihre Schlaf- und Wachphasen aufeinander. Dies soll vorallem das Problem des Idle-Listenings (ein Knoten befindet sich in Empfangsbereitschaft, obwohl es keine, für ihn relevanten, Daten zu empfangen gibt) mindern. Allerdings steigt dadurch der Kommunikationsoverhead, da zusätzliche Daten zur Synchronisation ausgetauscht werden müssen.

4.1.2. TDMA-basiert

TDMA-basiert bedeutet, dass das Medium, also der Funkkanal, in Zeitschlitze fester Länge unterteilt wird. Das Senden und Empfangen von Daten spielt sich innerhalb dieser Zeitschlitze ab. Bei nicht TDMA-basierten MAC-Protokollen gibt es diese Einteilung des Mediums nicht. Stattdessen darf ein Sender zu einem beliebigen Zeitpunkt, oft unter Verwendung des CSMA-CA-Verfahrens, um Kollisionen zu vermeiden, senden. TDMA-basierte MAC-Protokolle sind gleichzeitig synchron.

4.1.3. Contention-based und Contention-free

Contention ist das englische Wort für Wettstreit. Bei Contention-based MAC-Protokollen konkurrieren die Sender um das Medium. In der Regel prüft der Sender dann vor dem Senden, ob der Kanal frei ist. Ist dies der Fall, sendet der Sender seine Daten. Ist der Kanal belegt, versucht es der Sender zu einem späteren Zeitpunkt erneut. Dennoch kann es zu Kollisionen kommen, wenn mehrere Sender genau zum selben Zeitpunkt mit dem Sendevorgang beginnen oder, wenn sich das *Hidden-Station-Problem*[3] auswirkt. Daher bedient man sich zusätzlichen Mechanismen, wie dem Versand von ACKs, um die Kommunikation zuverlässiger zu gestalten. Bei Contention-free (auch *Schedule-based* genannt) MAC-Protokollen wird versucht dies zu vermeiden, indem jedem Sender ein dedizierter Zeitpunkt bzw. Zeitschlitz zum Senden zugeteilt wird. Contention-free MAC-Protokolle sind daher meist auch TDMA-basiert. In der Praxis sind dennoch Kollisionen möglich, da es andere Funknetzwerke in Reichweite geben kann, die denselben Kanal benutzen. Beide Varianten haben Vor- (+) und Nachteile (–):

- Contention-based MAC-Protokolle
  - \+ Sehr geringe Latenzen möglich, da die Sender nicht auf ihren Zeitschlitz warten müssen.
  - \+ Einfache und dezentrale Implementierung.
  - \+ Skaliert gut mit Änderungen (z. B. der Netzwerktopologie) im WSNs.
  - – Unfaire Aufteilung des Mediums setzt den Latenzen nach oben hin keine Grenzen. Dadurch können Knoten im Extremfall gar nicht mehr senden, da der Kanal ständig belegt ist.

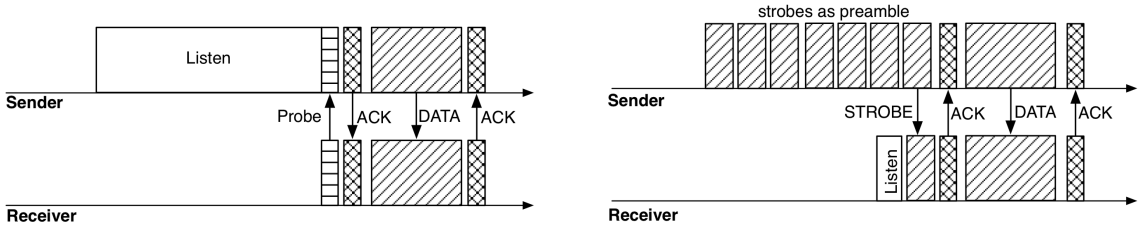
[3] Wenn sich mehrere Sender außer Reichweite zueinander befinden, registrieren sie nicht, dass ein anderer Sender bereits sendet. Unter der Annahme, dass der Kanal frei sei, beginnen sie dann zu senden. An den Empfängern, welche sich in Reichweite zu all den sendenden Sendern befinden, kommt es dann zur Kollision und die Daten werden unbrauchbar.

  - Tendenziell höherer Energieverbrauch aufgrund von *Idle-Listening*.

- Contention-free MAC-Protokolle
  + Latenz und Datendurchsatz für einen Knoten im WSN sind berechenbar.
  - Setzt einen aufwändigen Planungsalgorithmus voraus.
  - Skaliert schlecht mit Änderungen im WSNs, da der Planungsalgorithmus den Plan dann neu berechnen und über alle Knoten verteilen muss.
  - Setzt ein sauberes Medium, d. h. wenige bis keine konkurrierende Funktechnologien auf dem Kanal, voraus.
  - Unter Umständen nicht geeignet für Knoten mit sehr wenig Arbeitspeicher, da sich die Knoten ihre zugewiesenen Zeitschlitze merken müssen.

4.1.4. Low Power Probing und Low Power Listening

Asynchrone MAC-Protokolle arbeiten nach dem Prinzip der Kanalabtastung. Grundsätzlich wird dabei zwischen dem Low Power Probing (LPP) und dem LPL unterschieden. Die prinzipielle Funktionsweise beider Ansätze ist in Abb.4.1 dargestellt.



(a) LPP (b) LPL

Abbildung 4.1.: LPP und LPL bei MAC-Protokollen
Quelle: [12]

Beim LPP sendet der Empfänger nach Verlassen des Energiesparmodus einen *Probe* an den oder die Sender. Hat ein Sender Daten für den Empfänger in seiner Warteschlange, sendet er dann das Datenpaket an den Empfänger. Voraussetzung ist, dass die Sender empfangsbereit sind, um die Probes eines Empfängers zu empfangen. Im einfachsten Fall sind die Sender ständig in Empfangsbereitschaft.
Beim LPL verhält sich der Empfänger passiv. Nach Verlassen des Energiesparmodus wird lediglich für eine kurze Zeit auf ein Signal eines Senders (z. B. in Form eines *Strobes*) gewartet. Hat ein Empfänger ein solches Signal empfangen, behält er den Empfangsmodus eine gewisse Zeit lang bei, um die Daten zu empfangen.
Die konkrete Implementierung beider Verfahren kann, je nach MAC-Protokoll, unterschiedlich sein. Der wesentliche Unterschied ist, dass der Empfänger beim LPP stets mindestens einen Sendevorgang, nämlich das Senden des Probes, durchführt.

4.2. Ausgewählte MAC-Protokolle

Im Folgenden soll eine Reihe von MAC-Protokollen analysiert werden, welche hinsichtlich der obigen Forderungen (Energie sparen bei quasi ständiger Erreichbarkeit) interessant sind. In Anlage A.2 werden die Eigenschaften dieser MAC-Protokolle nochmals zusammengefasst.

4.2.1. WirelessHART-MAC-Protokoll

Das in Kap. 3.2.2 beschriebene MAC-Protokoll von WirelessHART, nutzt den Non-Beacon-Enabled-Mode von IEEE 802.15.4. WirelessHART verwendet seine eigene Superframe-Struktur, welche nicht auf 16 Slots begrenzt ist, sondern tausende von Slots umfassen kann. Statt sich auf die Stern- bzw. Cluster-Tree-Netzwerktopologie zu beschränken, unterstützt dieses MAC-Protokoll die Maschen-Netzwerktopologie. Außerdem führt WirelessHART Channel-Hopping ein. Eine Fähigkeit, welche sich in der Praxis — besonders bei WSNs mit vielen Knoten — als sehr wichtig erwiesen hat. Durch das Channel-Hopping sinkt nicht nur der Einfluss von Störungen auf das WSN, sondern es können auch mehrere Knoten im selben Slot senden, wenn jeder der sendenden Knoten einen anderen Kanal verwendet.

WirelessHART legt Wert auf Berechenbarkeit. D. h. jeder Knoten weiß, wann er senden darf. Damit ergibt sich eine gesicherte Bandbreite und Latenz für jeden Knoten. Erreicht wird dies, indem einem Knoten-Paar, bestehend aus Sender und Empfänger, stets ein Slot und ein Kanal zugeteilt wird. Eine Kombination aus Sender, Empfänger, Slot und Kanal wird als Link bezeichnet. Ist ein Superframe in 9000 Slots unterteilt und werden 15 Kanäle des 2,4-GHz-Bandes verwendet, ergeben sich 135.000 Links, von denen stets 15 gleichzeitig verwendet werden können.

Der Nachteil liegt darin, dass ein guter Planungsalgorithmus von Nöten ist. WirelessHART verwendet hier einen zentralen Ansatz, bei welchem die Planung vom Network Manager übernommen wird. Den Planungsalgorithmus selbst spezifiziert WirelessHART nicht, sondern überlässt dies den Geräteherstellern. Außerdem kommt es bei WirelessHART auf ein sehr exaktes Timing an. Die Zeitgeber müssen eine Genauigkeit von mindestens zehn parts per million (ppm) (vier mal besser, als von IEEE 802.15.4 vorgeschrieben) haben (vgl. [34, S. 215]).

Ein Knoten kann unter folgenden Bedingungen im Energiesparmodus verweilen:

- Der aktuelle Slot gehört zu keinem ihm zugewiesenen Link.
- Wenn in Sender-Rolle: Wenn es keine Daten zu senden gibt.
- Wenn in Empfänger-Rolle: Bis 2,12 ms[4] nach Beginn des zehn Millisekunden langen Slots und wenn spätestens 3,32 ms nach Beginn des Slots keine Daten empfangen wurden.

Zu beachten ist aber, dass die Zeitgeber der Knoten über Zusatzinformationen in den Datenpaketen, meist in den ACKs, synchronisiert werden. D. h. ein Knoten sollte nicht zu lange keine Daten austauschen. Wenn es keine Daten zu senden gibt, können stattdessen sog. *Keep-Alive-Pakete* gesendet werden. WirelessHART empfiehlt die Genauigkeit der Zeitgeber so auszulegen, dass Daten zur Synchronisation nicht öfter als alle 30 s ausgetauscht werden müssen, um die geforderte Genauigkeit bei einem Temperaturunterschied von 2 $°C$ pro Minute zu halten (vgl. [34, S. 215]).

[4] Dies ist die Zeit, welche dem Sender vor dem Senden der Daten für Berechnungen und den CCA-Check eingeräumt wird (vgl. [34, S. 211 f.]).

4.2.2. IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH)

Der TSCH-Modus von IEEE 802.15.4 ähnelt in seiner Funktionsweise stark dem MAC-Protokoll von WirelessHART, welches ein paar Jahre zuvor erschienen ist. Das kommt nicht von ungefähr, bediente man sich doch bei IEEE 802.15.4 an Ideen anderer Funkstandards. Beim TSCH synchronisieren sich alle Knoten auf einen sog. *Slotframe*, welcher sich zyklisch wiederholt (vgl. [33, S. 7 ff.]). Jeder Slotframe ist wiederum in sog. *Timeslots* unterteilt. Abb. 4.2 zeigt beispielhaft einen Slotframe mit vier Timeslots. Die Anzahl der Timeslots in einem Slotframe ist nicht in IEEE 802.15.4 festgeschrieben und kann durchaus einige tausend betragen (vgl. [56, S. 15]). Die Anzahl der Timeslots sollte daher optimal für die jeweilige Anwendung gewählt werden.

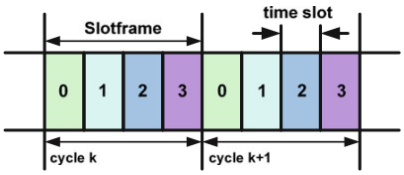


Abbildung 4.2.: Exemplarischer IEEE 802.15.4-Slotframe
Quelle: [33, S. 11]

Auch die Länge eines Timeslots ist nicht festgeschrieben. Ein Timeslot sollte lang genug sein, damit der Sender ein Datenpaket senden und das (optionale) ACK des Empfängers empfangen kann (siehe Abb. 4.3). Zusätzlich sollte noch etwas Zeit zum Ausgleich der Ungenauigkeit[5] der Zeitgeber, die sog. *Guardtime* und zur Datenverarbeitung eingeplant werden. Ein typischer Wert für die Länge eines Timeslots beträgt zehn ms (vgl. [56, S. 15]). Die Übertragung des größtmöglichen IEEE 802.15.4-Datenpaketes dauert im 2,4-GHz-Band ungefähr vier ms. Die Übertragung eines ACKs dauert etwa eine ms. Daher bleiben also fünf ms für die Datenverarbeitung und als Guardtime, um die Ungenauigkeit der Zeitgeber auszugleichen.

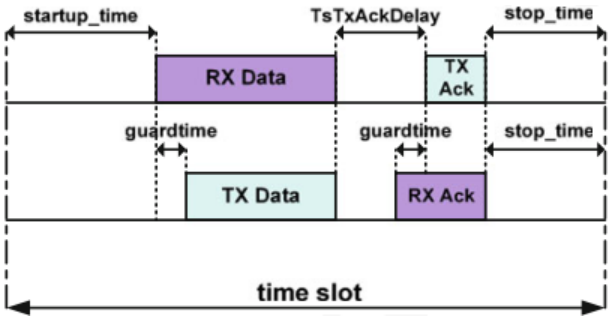


Abbildung 4.3.: IEEE 802.15.4-Timeslot
Quelle: [33, S. 11]

Mit *TX* wird das Senden und mit *RX* das Empfangen eines Datenpaketes respektive eines ACKs bezeichnet. Während der Zeit *TsTxAckDelay* überprüft der Empfänger das Datenpaket anhand einer Prüfsumme und generiert, wenn die Prüfsumme korrekt war, das ACK.

[5] Selbst, wenn die Zeitgeber der Knoten eines WSNs ständig synchronisiert werden, kommt es zu kleineren Abweichungen. Bei höheren Anforderungen an die Synchronität, können genauere Zeitgeber verbaut werden. Diese sind aber teuerer und haben oft einen höheren Stromverbrauch.

TSCH unterstützt Channel-Hopping. Im 2,4-GHz-Band kann die Kommunikation also über bis zu 16 Kanälen verteilt werden. Mehrere Knoten können so gleichzeitig innerhalb eines Timeslots senden, ohne dass es zu Interferenzen kommt, da jeder Knoten eine andere Frequenz verwendet. Die gerichtete Kommunikation zwischen zwei Knoten wird *Link* genannt. Jedem Link ist ein Timeslot und eine Frequenz zugeordnet. Es gibt also einen Link, damit A ein Datenpaket an B senden kann und einen anderen Link, damit B ein Datenpaket an A senden kann. Jeder Knoten muss eine Liste führen, damit er weiß, in welchem Timeslot er Daten an einen bestimmten Empfänger senden darf und wann er, in der Rolle des Empfängers, möglicherweise Daten zu erwarten hat. Die Frequenz, welche für den aktuellen Timeslot zu verwenden ist, wird mit Formel 4.1 berechnet.

$$f = F[(ASN + channelOffset)\ mod\ N_{Channels}] \tag{4.1}$$

Wobei ASN (Absolute Slot Number) die absolute Nummer des Timeslots seit der Initiierung des Netzwerkes ist. Es handelt sich also um einen Zähler, welcher mit jedem vergangenen Timeslot, auch über Slotframe-Grenzen hinweg, inkrementiert wird. channelOffset ist eine Nummer, welche für den Index des Kanals (nicht für die Kanal-Nummer selbst) steht und $N_{Channels}$ ist die Anzahl der Kanäle. Die Funktion F kann als Lookup-Tabelle implementiert sein. Dies erlaubt das Ausschließen (Blacklisting) einzelner Kanäle, weil diese z. B. eine schlechte Qualität haben. Von den 16 verfügbaren Kanälen im 2,4-GHz-Band, können bspw. nur 14 verwendet werden. Demnach hätte dann $N_{Channels}$ den Wert 14. Derselbe Link verwendet in aufeinanderfolgenden Timeslots stets einen anderen Kanal, damit die Qualität eines Links nicht von der Qualität eines einzelnen Kanals abhängt.

Zur Verteilung der Links ist ein spezieller Planungsalgorithmus notwendig, welcher nicht von IEEE 802.15.4 spezifiziert wird und daher den Implementationen bzw. der Standards, welche auf IEEE 802.15.4 aufbauen, überlassen ist. Abb. 4.4 zeigt eine mögliche Verteilung der Links in einem WSN mit Baum-Netzwerktopologie, welche auf das Sammeln von Daten mittels Sensoren optimiert ist. Im Beispiel besteht ein Slotframe aus vier Timeslots und es werden fünf Kanäle (mit den Indizes 0 bis 5) verwendet. In Abb. 4.4 rechts zu sehen, ist die Planungs-Matrix, welche acht Links in einem Slotframe unterbringt.

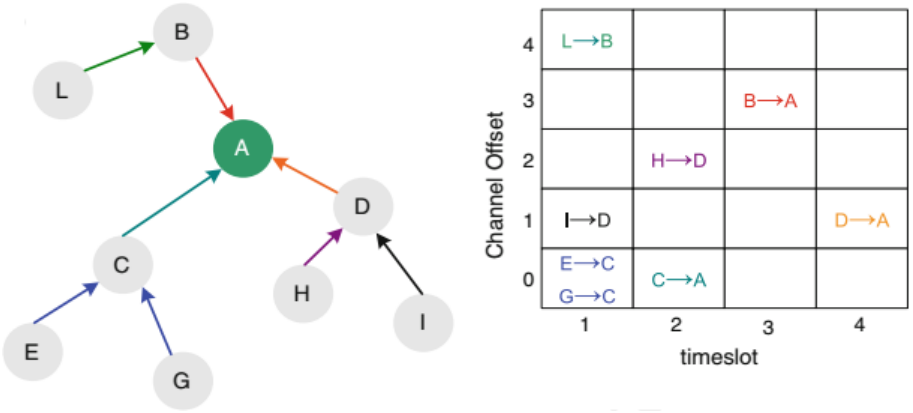


Abbildung 4.4.: Beispiel IEEE 802.15.4-TSCH-Kommunikationsplan in einem sternförmigen WSN

Quelle: [33, S. 12]

Ferner unterscheidet IEEE 802.15.4 TSCH zwischen *Dedicated Links* und *Shared Links*. Dedicated Links bleiben der Übertragung von Datenpaketen von einem bestimmten Quell- zu einem bestimmten Ziel-Knoten vorbehalten. Z. B. dient der Link an Position [1,1] der Matrix aus Abb. 4.4 ausschließlich zur Datenübertragung von Knoten I zu Knoten D. Bei einem Shared Link konkurrieren mehrere Knoten um das Medium. Der Link an Position [1,0] der Matrix ist ein Shared Link, bei welchem die Knoten E und G darum konkurrieren, wer senden darf. Im Beispiel ist C der einzige Empfänger, prinzipiell kann es aber mehrere Empfänger in einem Shared Link geben. Da es bei Shared Links zu Kollisionen kommen kann, definiert IEEE 802.15.4 einen sog. *Backoff-Algorithmus*, um die Wahrscheinlichkeit von erneuten Kollisionen zu reduzieren. Im Beispiel hätte man den Link $E \rightarrow C$ bspw. auch an Position [3,1] der Planungs-Matrix legen können, um Shared Links zu vermeiden. Dann gäbe es allerdings einen Timeslot weniger, in welchem der Knoten C Energie sparen könnte. Konnte ein Knoten seine Daten nicht erfolgreich übertragen, sei es, weil er den Konkurrenzkampf in einem Shared Link verloren hat oder wegen einer Störung des Kanals, muss der Knoten den Sendeversuch im nächst möglichen Slot bzw. Link wiederholen. Im Gegensatz zu einem Dedicated Link, bei welchem ein Knoten entweder Sender oder Empfänger ist, kann ein Knoten in einem, ihm zugewiesenen, Shared Link auch beides sein (vgl. [56, S. 17]). D. h. er wird zum Sender, wenn sich Datenpakete in seiner Warteschlange für ausgehenden Datenverkehr befinden und andernfalls zum Empfänger, um auf mögliche eingehende Datenpakete zu warten.

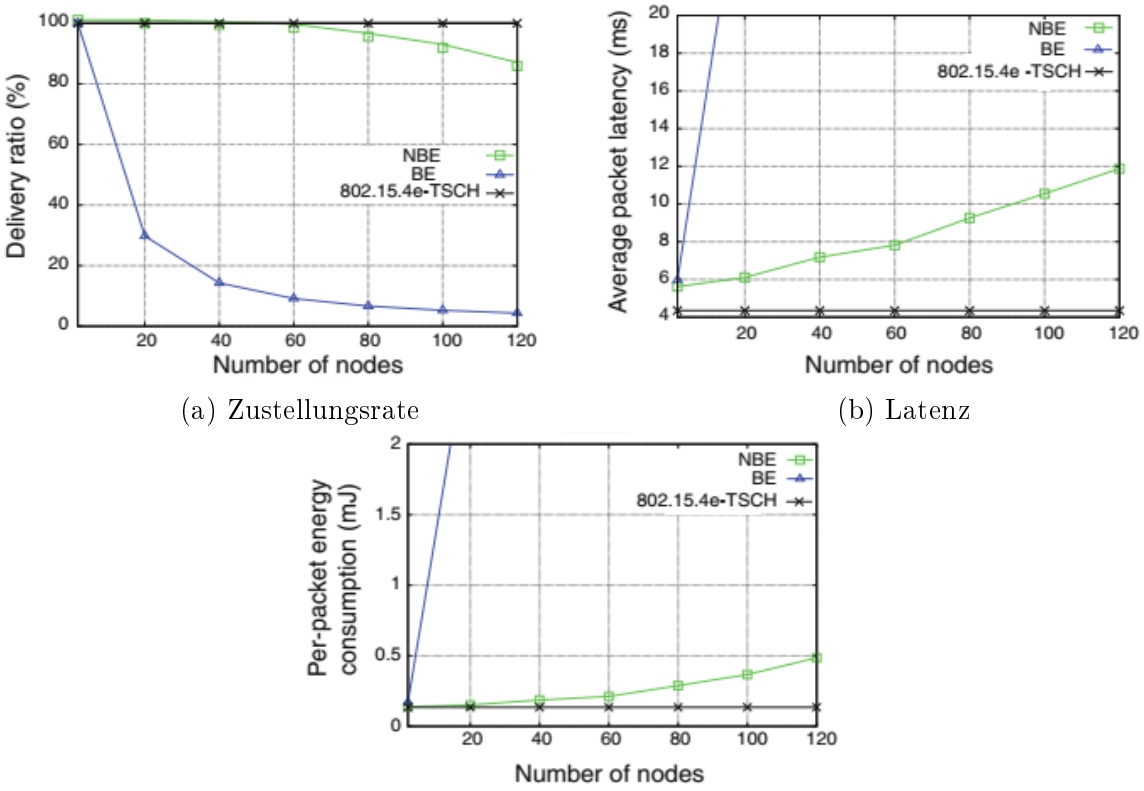
Es wird unterschieden in zentrale[6] und dezentrale[7] Planungsalgorithmen (vgl. [10, S. 13]). Bei einem zentralen Planungsalgorithmus übernimmt ein Knoten, typischerweise der PAN-Koordinator, die Planung. Dieser Knoten kennt die Struktur und den Zustand (z. B. die Auslastung einzelner Links) des gesamten Netzwerkes und kann die Links so verteilen, dass eine sehr effiziente Kommunikation im WSN möglich wird. Jedoch muss bei jeder Änderung der Netzwerkstruktur neu geplant werden. Bei Netzwerken, in welchen mit einer starken Dynamik zu rechnen ist, ist der zentrale Ansatz daher eher ungeeignet. Bei einem dezentralen Planungsalgorithmus handelt jeder Knoten die Links mit seinen Nachbarn aus. Da jeder Knoten nur einen Teil des Netzwerkes kennt (sich und seine Nachbarn), fällt die Planung dann unter Umständen nicht so effizient wie bei einem zentralen Planungsalgorithmus aus, jedoch kann besser auf Änderungen reagiert werden. Die diversen Planungsalgorithmen, welche in der Literatur zu finden sind, sind oft für eine bestimmte Netzwerktopologie und einen bestimmten Anwendungszweck, z. B. das Einsammeln und Transportieren von Daten verteilter Sensoren zu einer gemeinsamen Senke (*Convergecast*), konzipiert. Eine interessante Lösung bietet der Planungsalgorithmus Orchestra, welcher flexibel und an keinen Anwendungszweck gebunden ist (siehe Kap. 4.6). Er setzt weder auf den zentralen, noch auf den dezentralen Ansatz, sondern jeder Knoten plant seine Links autonom (ohne spezifische Informationen mit anderen Knoten austauschen zu müssen).

TSCH vs. Beacon-Enabled-Mode und Non-Beacon-Enabled-Mode

Guglielmo, Anastasi und Seghetti führten ein simuliertes Experiment durch, um die Performance des Beacon-Enabled-Mode (BE) mit Superframe und Non-Beacon-Enabled-Mode (NBE) von IEEE 802.15.4-2011 mit dem neuen TSCH-Modus von IEEE 802.15.4e

[6] Beispiele für zentrale Planungsalgorithmen sind in [44] und [52] zu finden.
[7] Beispiele für dezentrale Planungsalgorithmen sind in [3] und [53] zu finden.

respektive IEEE 802.15.4-2015 zu vergleichen (siehe [33, S. 13 ff.]). Es wurde die Latenz, die Zustellungsrate[8] und die aufgewendete Energie je Datenpaket über eine steigende Anzahl an Knoten im simulierten WSN ermittelt. Um den Vergleich fair zu halten, wurde TSCH ohne Channel-Hopping betrieben, d. h., es wurde nur ein Kanal verwendet.
Die Ergebnisse für den TSCH-Modus blieben mit steigender Anzahl an Knoten konstant und fielen durchweg gut aus. Die Zustellungsrate betrug sogar konstant 100 %. Der Non-Beacon-Enabled-Modus hingegen verschlechterte sich in allen drei Bereichen mit steigender Anzahl an Knoten. Am schlechtesten jedoch fielen die Ergebnisse für den Beacon-Enabled-Modus aus. Abb. 4.5 fasst die Ergebnisse des Experiments nochmal zusammen.



(a) Zustellungsrate

(b) Latenz

(c) Aufgewendete Energie je Datenpaket

Abbildung 4.5.: Performance-Analyse des Beacon-Enabled- und Non-Beacon-Enabled-Modus von IEEE 802.15.4-2011 und des TSCH-Modus von IEEE 802.15.4 von Guglielmo, Anastasi und Seghetti
Quelle: [33, S. 15 ff.]

4.2.3. X-MAC

2006 stellten Buettner u. a. in [13] das MAC-Protokoll X-MAC vor, welches einige Nachteile von älteren asynchronen LPL-MAC-Protokollen wie B-MAC und WiseMAC, beseitigt. Diese Nachteile sind die immer gleich lange Präambel mit relativ langer Sendedauer, die fehlende Skalierung bei sich änderndem Datenaufkommen und die fehlende Unterstützung für paketorientierte Funkeinheiten.

[8] Prozentualer Anteil der Gesamtheit aller versendeter Datenpakete, welche erfolgreich zugestellt werden konnten.

Will ein Sender Daten an einen Empfänger senden, muss der Sender diesen zunächst durch das Senden einer Präambel „aufwecken“. D. h. der Empfänger verlässt periodisch den Energiesparmodus, um den Kanal für kurze Zeit auf eine Präambel zu prüfen. Wird keine Präambel empfangen, geht der Sender wieder in den Energiesparmodus. Anderenfalls bleibt er für eine definierte Zeit in Empfangsbereitschaft, um die eigentlichen Daten des Senders zu empfangen. Bei klassischen LPL-MAC-Protokollen beträgt die Sendedauer der Präambel immer gleich lange und mindestens etwas länger als eine Schlafphase des Empfängers, um sicherzugehen, dass dieser die Präambel empfängt. Dies ist schlecht für die Energiebilanz von Sender und Empfänger. Selbst, wenn der Empfänger die Präambel relativ früh empfängt, muss dieser bis zum Ende der Präambel auf Empfang bleiben, um die Daten, welche auf die Präambel folgen, zu empfangen (siehe Abb. 4.6 oben). Ferner gilt dies auch für alle Empfänger, für welche die Daten nicht bestimmt sind, welche die Präambel aber empfangen haben (eng. *Overhearing Problem*). Denn erst, wenn die Daten empfangen wurden, also nach der Präambel, kann ein Empfänger feststellen, ob die Daten an ihn adressiert sind. Dies bedeutet, dass sich der Energieverbrauch für jede Datenübertragung proportional zu der Anzahl der Empfänger in Reichweite verhält.

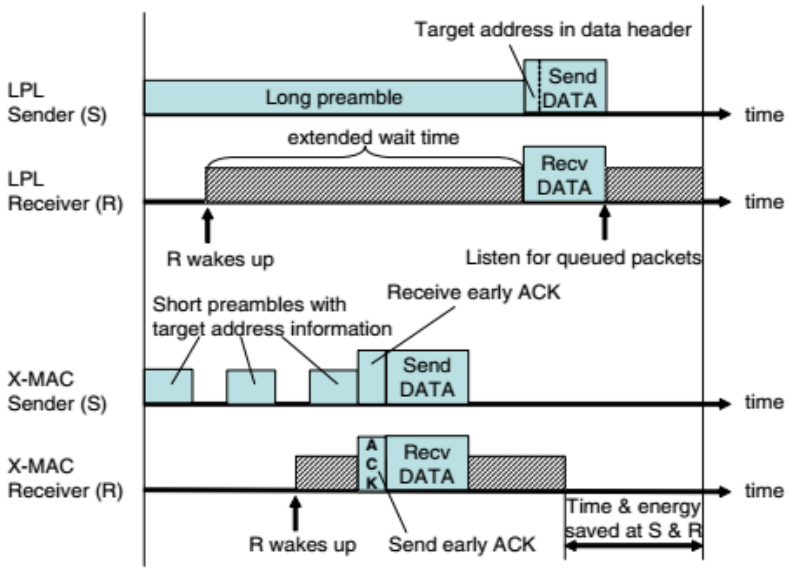


Abbildung 4.6.: Vergleich der klassischen Funktionsweise von LPL-MAC-Protokollen mit der von X-MAC

Quelle: [13, S. 310]

Die Latenz pro Hop ist nach unten hin durch die Dauer der Präambel begrenzt. D. h. also, in einem Multi-Hop-WSN ergibt sich die Mindest-Latenz aus der Dauer der Präambel multipliziert mit der Anzahl der Hops, welche zwischen einem Sender und Empfänger liegen. Dadurch kann die Latenz in einem Multi-Hop-WSN schnell sehr hohe Werte annehmen.

X-MAC begegnet diesen Problemen mit dem Einbetten der Adresse des Empfängers in eine gepulste Präambel (eng. *strobed preamble*). Das bedeutet, statt einer langen Präambel werden viele kurze Präambel-Pakete, welche jeweils die Adresse des Empfängers enthalten, nacheinander und mit einer kurzen Pausen zwischen jedem Präambel-Paket, gesendet (siehe Abb. 4.6 unten). Dadurch genügt es, dass der Empfänger ein kurzes Präambel-Paket empfängt, um festzustellen, ob er Ziel der Datenübertragung ist. Wenn nicht, kann er so-

fort wieder in einen Energiesparmodus wechseln. Dies behebt das Overhearing Problem. Hat ein Empfänger ein Präambel-Paket mit seiner Adresse empfangen, sendet er ein ACK zurück an den Sender. Der Sender weiß daraufhin, dass der Empfänger die Präambel empfangen hat und kann die eigentlichen Daten senden. Der Sender muss die gepulste Präambel also nicht für die gesamte Dauer senden. Dadurch, dass die Präambel gepulst ist, also aus vielen kurzen Präambel-Paketen mit Pausen besteht, kann der Empfänger bereits ein ACK in den Pausen zwischen zwei Präambel-Paketen senden, um dem Sender seine Empfangsbereitschaft zu signalisieren. Der Sender überträgt die eigentlichen Daten dann bereits zu einem früheren Zeitpunkt. Dies behebt das Problem der langen Präambeln und spart Energie auf Seiten des Senders und Empfängers und verringert die Latenzen, besonders in Multi-Hop-WSNs, erheblich. Die Zeit nach dem Empfangen eines Datenpakets bleibt der Empfänger noch eine kurze Zeit in Empfangsbereitschaft, für den Fall, dass mehrere Datenpakete kurz aufeinander folgen, z. B. bei einem Strom zusammenhängender Datenpakete. Dies wiederum erhöht den Datendurchsatz.
X-MAC verfügt noch über eine weitere Optimierung: Will ein Sender Daten an einen Empfänger übermitteln und stellt fest, dass der Kanal belegt ist, sendet er nicht die Präambel, sondern wartet darauf, dass der Kanal frei wird. Während der Sender wartet, befindet er sich im Empfangsmodus. Empfängt er währendessen ein ACK seines Ziels, also des gewünschten Empfängers, weiß er, dass der Empfänger in Empfangsbereitschaft ist. Der Sender muss nun keine eigene Präambel senden, sondern kann seine Daten direkt senden. Der Sender wartet damit eine zufällige Zeit, um Kollisionen mit anderen Sendern zu vermeiden. Dies funktioniert nur, weil der Empfänger, wie oben bereits beschrieben, nach dem Empfangen eines Datenpaketes noch eine kurze Zeit in Empfangsbereitschaft bleibt.

Bei der Wahl der Parameter für die Länge der Schlaf- und Wachphasen, geht es darum einen Kompromiss zwischen der Latenz bzw. dem Datendurchsatz und dem Energieverbrauch zu finden. Eine optimale Einstellung für jede Art von Datenaufkommen in einem WSN gibt es nicht. In ihrem Paper ermittelten Buettner u. a. optimale Werte für die Schlaf- und Wachphasen für Paketankunftsraten zwischen 0.01 und 100 Pakete pro Sekunde. Die Werte lagen zwischen ca. 1750 ms und 0 ms für die Schlafphasen bzw. zwischen 7,5 ms und 2 ms für die Wachphasen (vgl. [13, S. 311 ff.]). Jedoch kann das Datenaufkommen in einem WSN variabel sein, d. h., sich während des Betriebs ändern. X-MAC präsentiert daher ein Modell, mit dem sich optimale Werte für die Dauer der Schlaf- und Wachphasen in guter Näherung bestimmen lassen. Allerdings setzt dies das Verständnis der Dynamik der Anwendung vorraus.

Ältere LPL-MAC-Protokolle sind für Bit-Streaming-Funkeinheiten ausgelegt und funktionieren nicht mit den neueren und heutzutage üblichen paketorientierten Funkeinheiten (eng. Packetizing Radios). Bit-Streaming-Funkeinheiten erlauben dem Prozessor sozusagen jedes Bit zu kontrollieren. So können Präambeln beliebiger Länge erzeugt werden. Die paketorientierten Funkeinheiten erlauben eine so feine Kontrolle nicht. Der Prozessor teilt der paketorientierten Funkeinheiten einfach die Nutzdaten der PHY-Schicht mit und die Funkeinheit kümmert sich um die Generierung des Paketes, d. h., um das Voranstellen einer kurzen Präambel, mit welcher jedes Datenpaket beginnt, das Erzeugen der Prüfsumme usw. Beim Empfangen von Datenpaketen überprüft die paketorientierte Funkeinheit automatisch anhand der Prüfsumme, ob die Daten korrekt übertragen wurden und übermittelt dem Prozessor nur noch die Nutzdaten. Der Vorteil liegt hier klar in

der Entlastung des Prozessors. Da X-MAC keine langen Präambeln sendet, sondern viele kurze Präambel-Pakete, wobei es sich lediglich um ganz normale Datenpakete handelt, ist X-MAC für paketorientierte Funkeinheiten geeignet. Da sich das Verhalten von paketorientierten Funkeinheiten mit Bit-Streaming-Funkeinheiten per Software implementieren lässt, ist X-MAC folglich für beide Arten von Funkeinheiten geeignet.

X-MAC ist bereits ein betagtes MAC-Protokoll. Aber wie wir noch sehen werden, wurde vieles der grundlegenden Funktionsweisen für modernere MAC-Protokolle beibehalten.

4.2.4. Contiki X-MAC

Contiki X-MAC ist eine Abwandlung von X-MAC für das freie Open Source Betriebssystem Contiki und erweitert X-MAC um Fähigkeiten, welche im Folgenden erläutert werden (vgl. [20]).

Broadcast-Datenübertragung

Es gibt zwei Möglichkeiten, Broadcast-Datenübertragung in X-MAC zu implementieren:

1.) In die Präambel-Pakete wird die Broadcast-Adresse als Ziel-Adresse geschrieben, welche alle Knoten adressiert. Ein einzelner Knoten, welcher ein solches Präambel-Paket empfangen hat, bestätigt dies nicht mit einem ACK, behält aber die Empfangsbereitschaft bei. Um sicherzugehen, dass alle Knoten mindestens ein Präambel-Paket empfangen haben, wird ein Strobe aus Präambel-Paketen für die Dauer von etwas länger als einer Schlafphase gesendet. Anschließend folgt das eigentliche Datenpaket. Diese Methode verschwendet allerdings Energie, da die Empfänger statistisch gesehen bereits zur Hälfte der Sendezeit des Präambel-Strobes ein Präambel-Paket empfangen haben und bis zum Empfang des eigentlichen Datenpaketes in Empfangsbereitschaft bleiben müssen.

2.) Die zweite Methode besteht darin, bei Broadcast-Datenpaketen anstelle der Präambel-Pakete gleich das Datenpaket zu senden. So kann ein Knoten, welcher ein Broadcast-Datenpaket empfangen hat, sofort wieder in den Energiesparmodus zurückkehren. Da die Empfänger die Broadcast-Datenpakete bei dieser Methode allerdings zu unterschiedlichen Zeiten empfangen, kann es problematisch werden, Broadcast-Datenpakete für bestimmte Synchronisierungs-Zwecke einzusetzen.

In Contiki X-MAC wurde die zweite Methode implementiert.

Phasen-Optimierung (Phase Awareness)

X-MAC ist ein asynchrones MAC-Protokoll. D. h., da der Sender nicht über die Informationen verfügt, wann der Empfänger in Empfangsbereitschaft sein wird, beginnt er sofort mit dem Senden der Präambel-Datenpakete, wenn ein zu sendendes Datenpaket anfällt. Hat der Empfänger ein Präambel-Datenpaket empfangen, sendet er ein ACK zurück an den Sender. Contiki X-MAC merkt sich den Zeitpunkt, wann ein Empfänger ein Präambel-Datenpaket mit einem ACK bestätigt hat. Beim nächsten Sendevorgang an denselben Empfänger, beginnt Contiki X-MAC erst kurz vor der Wachphase des Empfängers mit dem Senden der Präambel-Datenpakete. Im Idealfall muss der Sender nur ein einziges Präambel-Datenpaket senden, bevor das eigentliche Datenpaket gesendet werden

kann. Sind die Zeitgeber von Sender und Empfänger seit dem letzten Datenaustausch zu weit auseinandergedriftet, muss der Sender wieder mehr Präambel-Datenpakete senden. Im schlimmsten Fall, wird wieder ein Präambel-Strobe maximaler Länge gesendet. Beim Empfangen des ACKs des Empfängers, aktualisiert der Sender dann einfach wieder den Zeitpunkt der Wachphasen des Empfängers. Dies macht Contiki X-MAC zu einem effizienteren Protokoll als X-MAC, da nicht so viel Energie für das Senden der Präambel-Pakete benötigt wird und der Kanal tendenziell weniger stark ausgelastet ist. Jedoch müssen die Knoten genügend Speicher bereithalten, um eine Tabelle über die Wachphasen, also die Zeitpunkte der Empfangsbereitschaften, ihrer Nachbar-Knoten zu pflegen.

Optimierung der zuverlässigen Datenübertragung

Die höheren Schichten können der MAC-Schicht, also Contiki X-MAC, mitteilen, dass ein Datenpaket vom Empfänger durch ein ACK zu bestätigen ist. Contiki X-MAC deaktiviert die Funkeinheit dann nicht sofort nach dem Senden des Datenpaketes, sondern hält sie noch eine Zeit lang in Empfangsbereitschaft, um das ACK des Empfängers zu empfangen. Der Empfänger wiederum muss dann nicht das ganze Prozedere, beginnend mit dem Senden der Präambel-Datenpakete usw., wiederholen, sondern kann das ACK unmittelbar nach Erhalt des Datenpaketes senden. Desweiteren unternimmt Contiki X-MAC selbstständig, ohne Anweisung durch höhere Schichten, das erneute Senden des Datenpaketes, wenn das ACK des Empfängers nicht empfangen wurde (optional MAC-Layer Retransmissions). Dies entlastet die höhereren Schichten.

Streaming-Fähigkeit

Von höheren Schichten gehen manchmal mehrere Datenpakete aus, welche zusammengehörig sind und damit einen *Stream* bilden. In Contiki X-MAC lassen sich Datenpakete als zugehörig zu einem Stream kennzeichnen. Der Empfänger eines Streams bleibt dann nach Erhalt eines Datenpaketes (und dem Senden des ACKs) etwas länger in Empfangsbereitschaft, um unmittelbar folgende Datenpakete zu empfangen.

Kompatibilität zu IEEE 802.15.4

Da das Betriebssystem Contiki den Standard IEEE 802.15.4 zur Funkdatenübertragung nutzt, macht es Sinn, Contiki X-MAC auf IEEE 802.15.4 zu optimieren. D. h. zum Senden der Präambel-Pakete wird das IEEE 802.15.4-Format für Datenpakete verwendet usw.

4.2.5. ContikiMAC

ContikiMAC wurde explizit für das Betriebssystem Contiki entwickelt und ist seit Contiki 2.5 das Standard-MAC-Protokoll[9] des Betriebssystems. Es wurde im Dezember 2011 von Adam Dunkels, dem Gründer des Contiki-Projektes, vorgestellt (siehe [24]). Dunkels behauptet in seinem Bericht, dass ContikiMAC signifikant effizienter Energie spart, als vorherige MAC-Protokolle, wie z. B. Contiki X-MAC (vgl. [24, S. 1]). Dies soll unter anderem durch ein präzises Timing erreicht werden.

[9] Im Contiki-Jargon spricht man auch von einem RDC-Protokoll. Contiki führt eigens eine RDC-Unterschicht in der MAC-Schicht ein, um RDC-Algorithmen besser zu kapseln.

Wie bei Contiki X-MAC (siehe Kap. 4.2.4) tastet der Empfänger periodisch den Kanal ab. Anstatt Präambel-Pakete zu senden, wie bei Contiki X-MAC, sendet ContikiMAC direkt Kopien der Datenpakete, solange, bis der Empfänger den Erhalt mit einem ACK bestätigt hat oder eine obere Zeitgrenze (Timeout) erreicht wurde (siehe Abb. 4.7). Zwischen den einzelnen Datenpaketen ist wieder eine Pause, lange genug, damit der Empfänger ein ACK senden kann. Nach dem Erhalt des ersten ACKs ist der Sendevorgang bereits beendet.

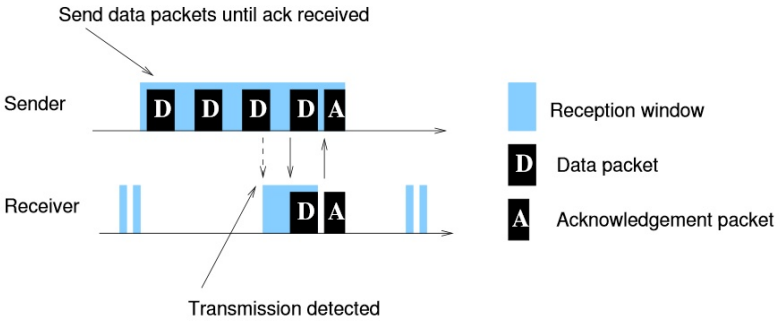


Abbildung 4.7.: Prinzipielle Funktionsweise von ContikiMAC
Quelle: [24, S. 2]

Hier ist bereits eine Verbesserung gegenüber Contiki X-MAC zu erkennen: Bei Contiki X-MAC würde der Empfänger erst das Präambel-Datenpaket mit einem ACK bestätigen, dann würde der Sender das eigentliche Datenpaket senden, welches der Empfänger wiederum (optional) durch ein ACK bestätigen würde. Da ContikiMAC das Datenpaket direkt sendet, entfällt das Senden der letzten beiden Pakete (Datenpaket und ACK), da der Empfänger nach dem ersten ACK ja bereits die Nutzdaten empfangen hat. Da ein Präambel-Pakete unter X-MAC bzw. Contiki X-MAC im Gegensatz zu einem IEEE 802.15.4-Datenpaket in voller Länge relativ kurz ausfällt, kann dieser Vorteil wieder zunichte gemacht werden, wenn die Datenpakete allgemein sehr groß ausfallen. Die Wahl des richtigen MAC-Protokolls könnte hier also wieder eine Frage des Kommunikationsverhaltens im konkreten WSN sein.

Die Broadcast-Datenübertragung unterscheidet sich im Prinzip nicht von der unter Contiki X-MAC: Datenpakete mit demselben Inhalt werden fortlaufend wiederholt gesendet und zwar etwas länger als die Dauer der längst möglichen Schlafphase eines Knotens, um sicherzugehen, dass alle Knoten das Datenpaket empfangen haben (siehe Abb. 4.8). Ebenfalls wird der Erhalt eines Broadcast-Datenpaketes nicht durch ein ACK bestätigt.

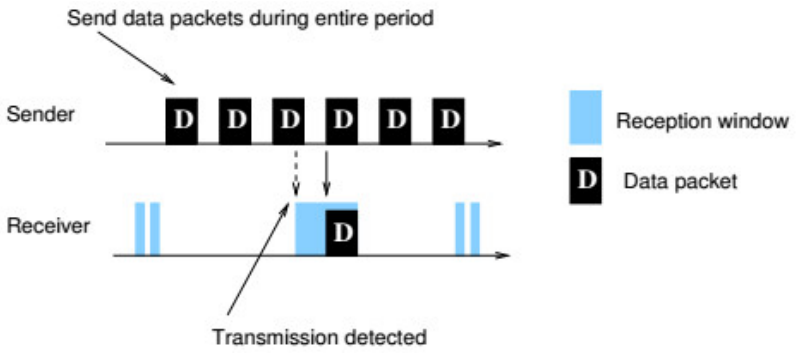


Abbildung 4.8.: Broadcast-Datenverkehr unter ContikiMAC
Quelle: [24, S. 2]

CCA-Phase

Wie oben bereits beschrieben, tastet der Empfänger periodisch den Kanal ab. Wie oft der Kanal abgetastet wird, wird über die sog. *Channel Check Rate (CCR)* angegeben. Die CCR hat die Einheit Herz und ist üblicherweise eine Potenz von zwei.[10] Jedes Abtasten, im Zusammenhang mit ContikiMAC, wird in dieser Arbeit als CCA-Phase bezeichnet. Eine CCA-Phase besteht aus zwei einzelnen CCAs der Länge t_r, welche in einem Abstand von t_c zueinander liegen (siehe Abb. 4.9).

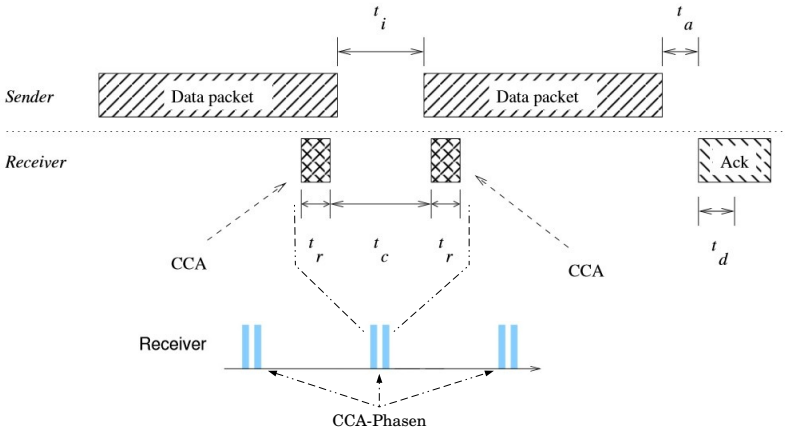


Abbildung 4.9.: Eine CCA-Phase in ContikiMAC
Quelle: In Anlehnung an [24, S. 2]

Bei einem CCA wird der Received Signal Strength Indicator (RSSI) herangezogen. Er liefert einen Wert über das Energieniveau des aktuell empfangenen Signals und wird häufig in dBm angegeben. Hat der RSSI einen konfigurierbaren Wert, z. B. -80 dBm, überschritten, fällt das CCA negativ aus, d. h. der Kanal ist nicht frei, also übermittelt ein potenzieller Sender gerade Daten. Dabei ist es allerdings egal, welchen Ursprung das Signal hat. Auch ein Signal von einer inkompatiblen Technologie, wie WLAN oder Bluetooth, kann zu einem negativen CCA führen. (Der IEEE 802.15.4-kompatible Chip Atmega 128RFA1 besitzt ein *Carrier Sense* genanntes Feature. D. h. beim CCA wird nicht der RSSI herangezogen, sondern es wird geprüft, ob tatsächlich ein IEEE 802.15.4-konformes Signal vorliegt.) Fallen beide CCAs positiv aus, bedeutet dies, dass niemand Daten sendet

[10] In der aktuellen Implementierung in Contiki 3.1 beträgt die CCR 8 Hz (siehe auch Tabelle 4.2). D. h. also, der Kanal wird acht mal in der Sekunde bzw. alle 125 ms abgetastet.

und der Empfänger kann wieder in den Energiesparmodus wechseln. Fällt einer der beiden CCAs negativ aus, bleibt der Empfänger auf Empfang, da er ein Datenpaket erwartet.

Die Zeitparameter von ContikiMAC unterliegen einer Reihe von Restriktionen:

- Die Zeit zwischen zwei Datenpaketen muss kleiner sein, als die Zeit zwischen den beiden CCAs, um sicherzustellen, dass entweder der erste oder der zweite CCA ein Datenpaket detektieren kann: $t_i < t_c$
- Die kürzeste Sendedauer t_s, die ein Datenpaket haben darf, muss lang genug sein, damit ein Datenpaket nicht zwischen zwei CCAs fallen kann (siehe Abb. 4.10), d. h. die Restriktion lautet: $t_s > t_r + t_c + t_r$
- Die Zeit, welche benötigt wird, um ein ACK zu senden (t_a) und zu empfangen (t_d) legt die untere Zeitgrenze (t_c) zwischen zwei CCAs fest: $t_a + t_s < t_c$
- Die Restriktionen können nun zusammengefasst werden zu: $t_a + t_s < t_i < t_c < t_c + 2t_r < t_s$

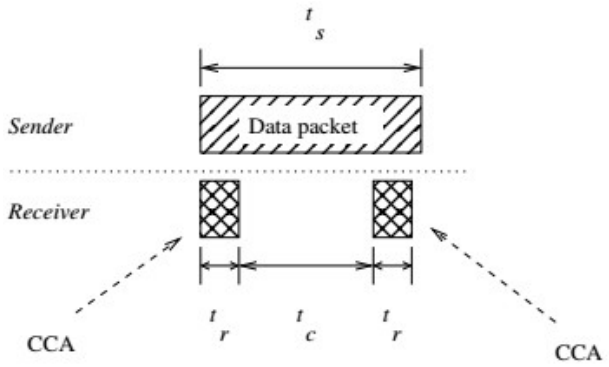


Abbildung 4.10.: Die Sendedauer t_s eines Datenpaketes muss lang genug sein, um von den CCAs sicher detektiert werden zu können
Quelle: [24, S. 3]

Tabelle 4.2 fast die Zeitparameter von ContikiMAC zusammen. Diese können zum Größteil frei gewählt werden, jedoch sind einige Werte durch IEEE 802.15.4 vorgegeben. Diese Vorgabewerte und die aktuellen Standardwerte von Contiki 3.1 sind in Tabelle 4.2 in der letzten Spalte angegeben.

Eine CCA-Phase hat mit den Werten aus Tabelle 4.2 eine Dauer von 1,036 ms (t_c+2t_r). Der Standardwert für die Dauer einer Wachphase von Contiki X-MAC unter Contiki 3.1 beträgt 6,25 ms. Hinzu kommt, dass die Aktivzeit der Funkeinheit unter Contiki X-MAC die gesamte Wachphase über andauert. Bei ContikiMAC hingegen beträgt diese Zeit nur $2t_r$. Mit diesen Parametern erzielt ContikiMAC gegenüber (Contiki) X-MAC einen signifikant besseren Energieverbrauch.

	Bedeutung	**Wert**
t_i	Zeitlicher Abstand, mit welchem der Sender die Datenpakete sendet.	0,4 ms[a]
t_r	Zeit, welche für einen CCA benötigt wird.	0,268 ms[b]
t_c	Zeit zwischen zwei aufeinanderfolgenden CCAs innerhalb einer CCA-Phase.	0,5 ms[a]
t_a	Zeit zwischen dem Empfangen eines Datenpakets und dem Senden des zugehörigen ACKs.	**0,192 ms**[c]
t_d	Benötigte Zeit, um ein ACK des Empfängers erfolgreich zu erkennen.	**0,16 ms**
t_s	Kürzeste Sendedauer, welche ein Datenpaket haben darf, um in einer CCA-Phase sicher detektiert zu werden.	$t_s > 2t_r + t_c$
t_l	Sendedauer, welche für das längst mögliche IEEE 802.15.4-Datenpaket benötigt wird.	**4 ms**
CCR	Gibt die Wiederholungsrate der CCA-Phase an.	8 Hz

[a] Standardwert der Implementierung in Contiki 3.1.
[b] Der Wert ist für jede Funkeinheit spezifisch. Dieser Wert gilt für den Atmega 128RFA1 und das 2,4-GHz-Band (vgl. [11, S. 74]).
[c] **Fett** dargestellte Werte sind durch IEEE 802.15.4 festgelegt und gelten für das 2,4-GHz-Band.

Tabelle 4.2.: Die Zeitparameter von ContikiMAC

Fast Sleep

Wie im Abschnitt *CCA-Phase* oben beschrieben, sorgt jegliche Energie auf dem entsprechenden Kanal, welche den Empfänger erreicht und den RSSI-Schwellwert überschreitet dafür, dass Daten detektiert werden. Egal, ob es sich um eine valide IEEE 802.15.4-Datenübertragung oder ein nicht verwertbares Signal handelt. Durch *Fast Sleep* ist ContikiMAC in der Lage, ein nicht verwertbares Signal zu erkennen und umgehend wieder in einen Energiesparmodus zu wechseln. Ein nicht verwertbares Signal wird anhand folgender Bedingungen erkannt:

- Die detektierte Energie hält länger an, als die Sendedauer t_l des längsten IEEE 802.15.4-Datenpaketes. (Auf jedes Datenpaket muss eine Pause folgen.)
- Der Zeitpunkt seit das letzte Mal Energie detektiert wurde, ist länger her als t_i, die Zeit, welche zwischen zwei aufeinanderfolgenden Datenpaketen liegt.
- Wenn auf den Zeitpunkt der letzten Detektierung von Energie eine korrekt lange Pause t_i folgt, nach der wieder Energie detektiert wird, welche gemäß IEEE 802.15.4 aber keinen gültigen Anfang eines Datenpaketes darstellt.

Die prinzipielle Funktionsweise von Fast Sleep ist nochmal in Abb. 4.11 dargestellt.

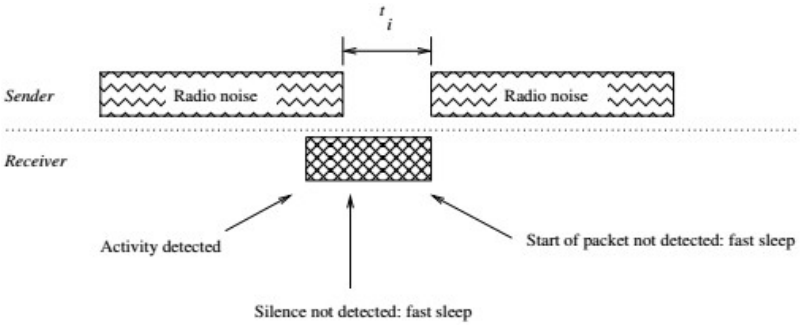


Abbildung 4.11.: Die prinzipielle Funktionsweise von ContikiMACs Fast Sleep
Quelle: [24, S. 4]

Phasen-Optimierung (Transmission Phase-Lock)

Wie bei Contiki X-MAC, gibt es auch eine Phasen-Optimierung. Die Funktionsweise ist die gleiche: Bei der ersten Datenübertragung kennt der Sender den Zeitpunkt der nächsten CCA-Phase des Empfängers noch nicht und muss ein Datenpaket öfters senden, bevor es der Empfänger erkennt und ein ACK zurück sendet (siehe Abb. 4.12 oben). Anhand des Zeitpunktes des Empfangens des ACKs vom Empfänger, kann der Sender bei der nächsten Datenübertragung den Zeitpunkt der nächsten CCA-Phase des Empfängers berechnen und muss im Idealfall nur noch zwei Datenpakete senden (siehe Abb. 4.12 unten). Das erste Datenpaket dient dazu den Empfänger zu „wecken“ und das Zweite wird schließlich empfangen. Hardwareseitig setzt dieses Verfahren wieder genügend Speicher voraus, um eine Tabelle der CCA-Phasen aller Nachbarknoten zu pflegen.

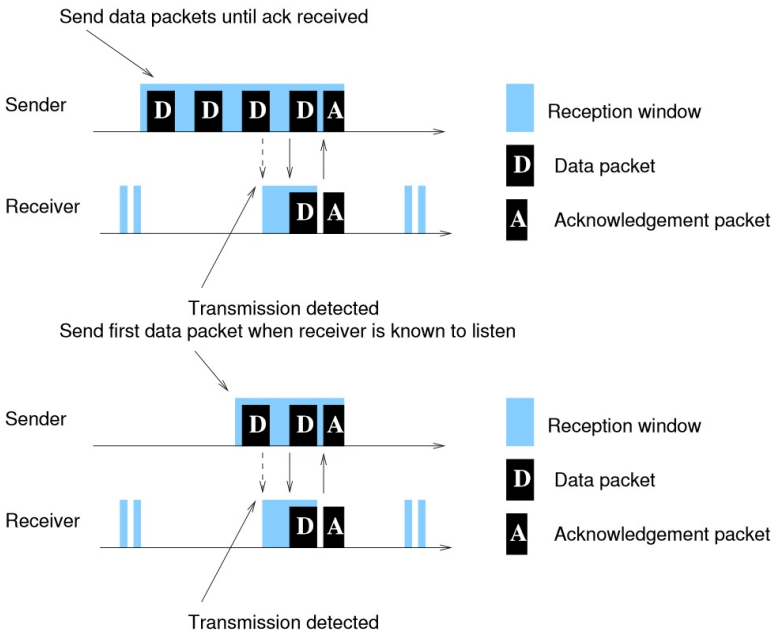


Abbildung 4.12.: Die prinzipielle Funktionsweise von ContikiMACs Phasen-Optimierung
Quelle: [24, S. 4]

4.2.6. MiCMAC

In ihrer Arbeit vom Mai 2014 stellten Nahas u. a. *MiCMAC* vor, ein MAC-Protokoll, welches auf ContikiMAC aufbaut und dieses um Channel-Hopping erweitert (siehe [42]). Die Autoren entschieden sich für ContikiMAC als Basis für MiCMAC, da es sich bereits als sehr effizient beim Einsatz nur eines Kanals erwiesen hat (vgl. [42, S. 3]). Channel-Hopping und dessen Vorteile waren zuvor schon lange bekannt, dennoch ist der Einsatz von Channel-Hopping in WSNs nicht selbstverständlich. Das ist daran zu erkennen, dass die Standard-MAC-Protokolle vieler Softwares für WSNs, wie RIOT OS oder Contiki, kein Channel-Hopping nutzen (vgl. [42, S. 2]). Als Gründe dafür sehen die Autoren von MiCMAC, dass die Implementierung solcher MAC-Protokolle entweder zu komplex, einen idealen Kommunikationsplan (und damit einen guten Planungsalgorithmus) voraussetzen oder zu anwendungsspezifisch sind. MiCMAC ist jedoch ein recht einfaches und anwendungsunabhängiges Channel-Hopping-MAC-Protokoll, welches ohne Planungsalgorithmus auskommt.

CCA-Phase und Channel-Hopping

Der Ablauf der CCA-Phasen gestaltet sich wie bei ContikiMAC, mit dem Unterschied, dass ein Knoten jede nachfolgende CCA-Phase auf einem anderen Kanal durchführt (vgl. [42, S. 3]). Das Intervall der CCA-Phasen im WSN ist für alle Knoten gleich. (Die CCA-Phasen der einzelnen Knoten können aber zueinander versetzt sein und sind dies in der Praxis auch häufig.) Nimmt ein Sender zum ersten Mal Kontakt mit einem Empfänger auf, lernt der Sender die CCA-Phasen des Empfängers, sodass er bei der nächsten Kontaktaufnahme den Zeitpunkt und den Kanal auf dem der Empfänger seine CCA-Phase durchführen wird, vorausberechnen kann. Ohne diese Phasenoptimierung wäre jede Kontaktaufnahme, gegenüber ContikiMAC, sehr ineffizient, wie sich weiter unten noch zeigen wird, da neben dem Zeitpunkt noch der zusätzliche Parameter des Kanals hinzukommt. Anders als bei ContikiMAC, ist die Phasen-Optimierung bei MiCMAC daher kein optionales Feature, sondern Pflicht.

Der Kanal, auf dem die nächste CCA-Phase stattfindet, wird von einer gleichverteilten Pseudozufallszahl bestimmt, welche mittels eines linearen Kongruenzgenerators (eng. *Linear Congruential Generator*) erzeugt wird (vgl. [42, S. 4]). Diesem Generator liegt die Formel 4.2 zugrunde.

$$X_{n+1} = (aX_n + c)\ mod\ N, \quad n \geq 0 \tag{4.2}$$

Wobei X_{n+1} der nächste Kanal ist, N die Anzahl der verwendeten Kanäle, a ein Faktor mit $0 \leq a < N$, c ein Inkrement mit $0 \leq c < N$ und X_n der letzte Kanal ist. Die Wahl der Parameter a, c und N bestimmen die Eigenschaften der Pseudozufallszahlenfolge und können so gewählt werden, dass die Kanäle in zufällig erscheinender Folge und immer nur genau einmal auftreten, bevor sich die Pseudozufallszahlenfolge wiederholt (kurze Channel-Hopping-Sequenz). Bei der ersten Verwendung der Generators wird X_n mit einem Wert X_0 mit $0 \leq X_0 < N$ initialisiert. Wenn nun das 2,4-GHz-Band verwendet werden soll, wird zu der Kanalnummer noch 11 addiert, da dies der erste Kanal in diesem Frequenzband ist. Sollen alle Kanäle des 2,4-GHz-Bandes benutzt werden, wird N auf 16 gesetzt.

Jeder Knoten im WSN erhält seine eigene Channel-Hopping-Sequenz, indem ihm ein Tupel $\{< a, c >\}$ zugewiesen wird. Die Länge der Channel-Hopping-Sequenz beträgt daher $\|\{< a, c >\}\| \times N$ (lange Channel-Hopping-Sequenz). Die Wahl einer kurzen (mit Länge N) oder langen (Kombination aus mehreren kurzen Sequenzen) Channel-Hopping-Sequenz, beeinflusst das Verhalten der ersten Kontaktaufnahme und des Broadcast-Sendevorgangs, wie weiter unten beschrieben wird.

Andere MAC-Protokolle, wie IEEE 802.15.4 CSL (siehe Kap. 4.2.7), bieten noch die Möglichkeit des Blacklistings einzelner Kanäle, statt wie hier, wo alle Kanäle aus einem bestimmten Bereich verwendet werden. Da sich aber gezeigt hat, dass blindes Channel-Hopping, wie bei MiCMAC, die Verbindungsqualität gegenüber der Verwendung nur eines Kanals bereits signifikant verbessert, haben die Autoren von MiCMAC zugunsten einer einfachereren Implementierung auf ein Blacklisting verzichtet ([42, S. 4]).

Erste Kontaktaufnahme zwischen einem Sender und einem Empfänger

Bei der ersten Kontaktaufnahme weiß der Sender noch nicht, wann und auf welchem Kanal der Empfänger seine nächste CCA-Phase durchführen wird. Daher wählt er einen beliebigen Kanal aus, auf dem er beginnt, die Datenpakete fortlaufend zu senden, bis der Sender ein ACK vom Empfänger empfangen hat. Um sicherzugehen, dass eines der Datenpakete von der nächsten CCA-Phase des Empfängers detektiert wird, muss der Sender die Datenpakete maximal für die Dauer $W = I \cdot N$ senden, mit I der Dauer eines CCA-Phasen-Intervalls und N der Anzahl der verwendeten Kanäle. Abb. 4.13 veranschaulicht dies an einem Beispiel mit $N = 4$. Jede Farbe kennzeichnet einen anderen Kanal mit der Ausnahme von Blau, welches den Empfang eines Datenpaketes respektive ACKs kennzeichnet.

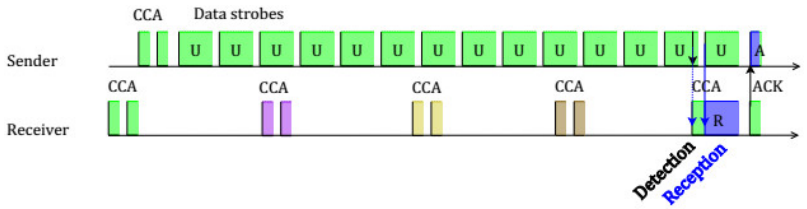


Abbildung 4.13.: Erste Kontaktaufnahme zwischen Sender und Empfänger bei $N = 4$
Quelle: [42, S. 4]

Das ACK des Empfängers beinhaltet dessen Parameter a, c und X_0. Der Sender speichert diese Parameter und den aktuellen Kanal und Zeitpunkt in einer internen Tabelle und kann somit den Zeitpunkt und Kanal der CCA-Phase des Empfängers für jede folgende Kontaktaufnahme vorherberechnen. Im Umkehrschluss bedeutet dies, dass ein Knoten für die Phasen-Optimierung und derselben Anzahl an Nachbarn mehr Speicher bereithalten muss, als unter ContikiMAC.

Broadcast-Datenübertragung

Wie bei ContikiMAC werden Broadcast-Datenpakete von den Empfängern nicht durch ein ACK bestätigt, sondern es werden fortlaufend für die gesamte Dauer eines Zyklus

Kopien des Broadcast-Datenpaketes gesendet, sodass theoretisch jeder Empfänger eine Kopie empfangen haben muss. MiCMAC bietet zwei Varianten für die Broadcast-Datenübertragung:

1.) Die Standardvariante von MiCMAC: Wie bei der ersten Kontaktaufnahme, werden die Datenpakete für die Dauer $W = I \cdot N$ (bei einer kurzen Channel-Hopping-Sequenz) oder $W = I \cdot (2N - 1)$ (bei einer langen Channel-Hopping-Sequenz) wiederholt gesendet, mit dem Unterschied, dass der Sendevorgang nicht frühzeitig durch ein ACK des Empfängers eingestellt werden kann. Die Sendedauer ist also stets gleich lang. Dies hat den Nachteil, dass der Sender viel Energie benötigt und der entsprechende Kanal für die Dauer des Broadcast-Sendevorgangs relativ lange ausgelastet ist. Abb. 4.14 zeigt ein Beispiel für eine Broadcast-Datenübertragung, wenn vier Kanäle benutzt werden.

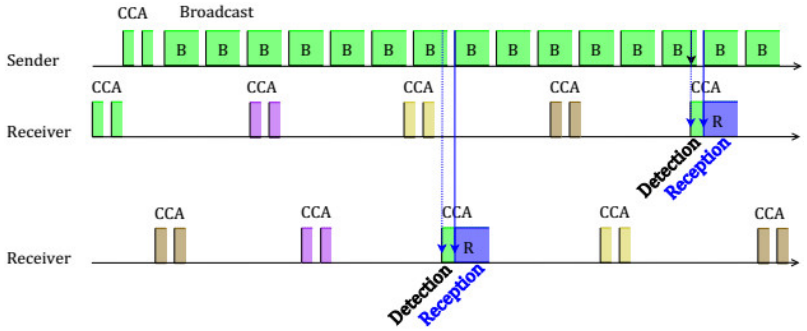


Abbildung 4.14.: Klassische Broadcast-Datenübertragung unter MiCMAC bei $N = 4$
Quelle: [42, S. 5]

2.) MiCMAC-BC: Eine abgewandelte Variante für die Broadcast-Datenübertragung von MiCMAC stellt MiCMAC-BC dar. Unmittelbar auf die regulären CCA-Phasen folgt bei MiCMAC-BC eine weitere CCA-Phase, welche stets auf einem dedizierten Broadcast-Kanal stattfindet. Ein Sender muss eine Broadcast-Datenübertragung dann nur für die Dauer eines CCA-Phasen-Intervalls auf diesen dedizierten Kanal durchführen. Der Vorteil ist hier ganz klar, dass der Sender weniger Energie benötigt. Die Nachteile sind jedoch, dass die Knoten einen höheren Energieverbrauch wegen der doppelt so langen CCA-Phasen haben und die Broadcast-Datenübertragung weniger robust ist, da sie nur auf einem Kanal stattfindet. Ein Beispiel für eine Broadcast-Datenübertragung unter MiCMAC-BC mit vier Kanälen für die Unicast-Datenübertragung und einem Kanal für die Broadcast-Datenübertragung ist in Abb. 4.15 veranschaulicht.

Als Mittelweg zwischen MiCMAC und MiCMAC-BC wäre eine Variante des MAC-Protokolls denkbar, welches ebenfalls mehrere Kanäle für die Broadcast-Datenübertragung benutzt (vgl. [42, S. 5]). Die Anzahl der verwendeten Kanäle für die Broadcast-Datenübertragung wäre dann geringer, als die Anzahl der Kanäle für die Unicast-Datenübertragung, wenn man davon ausgeht, dass es in einem WSN mehr Unicast- als Broadcast-Datenverkehr gibt.

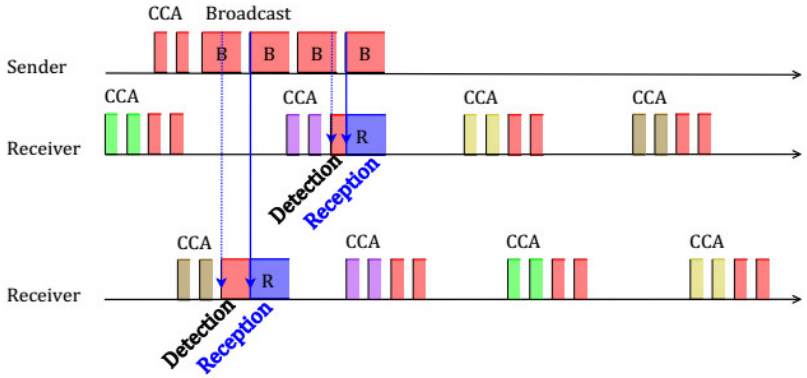


Abbildung 4.15.: Broadcast-Datenübertragung unter MiCMAC-BC bei $N = 4$
Quelle: [42, S. 5]

Always-on Nodes

Manche Knoten (typischerweise Border-Router) verfügen über eine stationäre Stromversorgung. Da das Energiesparen in diesem Fall eine untergeordnete Rolle spielt, lassen diese Knoten ihre Funkeinheit zugunsten einer besseren Performance ständig aktiv (always on). Um auf allen Kanälen Daten empfangen zu können, wechseln Always-on Nodes fortlaufend den Kanal. Die Autoren von MiCMAC haben sich für ein Kanalwechsel-Intervall von 10 ms entschieden, da es die Sendedauer von zwei vollen IEEE 802.15.4-Datenpaketen umfasst. Sender, welche an eine Always-on Node senden, benutzen keine Phasen-Optimierung.

Verwendung vordefinierter Channel-Hopping-Sequenzen

Statt die Channel-Hopping-Sequenz von einem Zufallszahlengenerator vorgeben zu lassen, bietet MiCMAC auch die Möglichkeit, diese statisch in einer Tabelle zu hinterlegen. Die MAC-Adresse eines Knotens bestimmt dann, welche Sequenz verwendet wird.

Verwendung kurzer Channel-Hopping-Sequenzen

Bei der Verwendung kurzer Channel-Hopping-Sequenzen mit der Länge N, kann der Empfänger auf das Einbetten der Parameter a, c und X_0 in seine ACKs verzichten. Da jeder Kanal in einer Sequenz nur einmal vorkommt, reicht der aktuelle Kanal und die MAC-Adresse des Empfängers aus, damit der Sender dessen Sequenz eindeutig bestimmen, d. h. vorausberechnen kann.

MiCMAC im Vergleich mit ContikiMAC

In einem Experiment verglichen Nahas u. a. die Performance von MiCMAC und ContikiMAC miteinander (siehe [42, S. 5 ff.]). Das Ergebnis war, dass MiCMAC in einer interferenzfreien Umgebung eine fast genau so gute Performance bot, wie ContikiMAC. In einer Umgebung mit vielen Interferenzen, wie es in echten Einsatzszenarien oft der Fall ist, lieferte MiCMAC jedoch eine signifikant bessere Performance ab. Das Experiment konstruierte ein Sezenario, bei dem es um das Dateneinsammeln ging. D. h., mehrere Sensoren übertragen in gewissen Abständen Daten zu einer gemeinsamen Senke. Nicht getestet wurde, inwiefern sich MiCMAC schlägt, wenn es darum geht, dass mehrere Knoten gleichzeitig auf unterschiedlichen Kanälen Daten austauschen. Das Verwenden mehrerer

Kanäle ist ein Weg, den Datendurchsatz in einem WSN durch simultane Verbindungen zu erhöhen.

4.2.7. IEEE 802.15.4 CSL

Abb. 4.16 stellt die prinzipielle Funktionsweise des CSL-Modus von IEEE 802.15.4 dar.

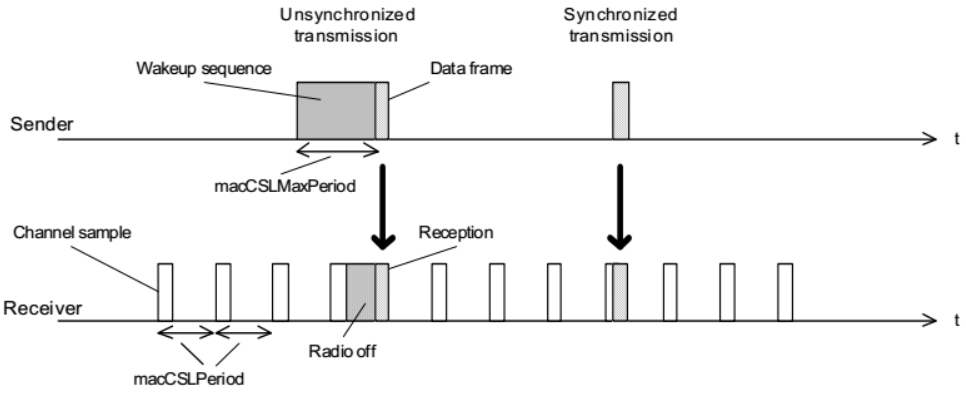


Abbildung 4.16.: IEEE 802.15.4 CSL
Quelle: [51, S. 51]

Empfangen von Datenpaketen

Der Empfänger (Receiver) überprüft periodisch, im Abstand von *macCSLPeriod*, das Energieniveau bzw. den RSSI-Wert des Kanals. Liegt der RSSI-Wert über einem bestimmten Schwellwert, wird die Funkeinheit für eine längere Zeit im Empfangsmodus gelassen, um mögliche *Wake-Up-Pakete* zu empfangen. Die Zeit zwischen den einzelnen Abtastpunkten kann die Funkeinheit deaktiviert werden oder gar der gesamte Knoten in einen Energiesparmodus wechseln. Die Zeit, in welcher keine Wake-Up-Datenpakete detektiert werden, wird als *Idle Listening* bezeichnet.

Wurde ein Wake-Up-Paket empfangen, wird geprüft, ob die Zieladresse im Wake-Up-Pakete mit der eigenen Adresse übereinstimmt. Ist dies der Fall, wird die Funkeinheit bis zur, im Wake-Up-Paket mitgeteilten, *Rendezvous Time* deaktiviert. Zur Rendezvous Time wird die Funkeinheit wieder in den Empfangsmodus versetzt, um das Datenpaket mit den Nutzdaten zu empfangen.
Enthält das Wake-Up-Paket eine fremde Adresse als Ziel, wird die Funkeinheit für die Zeit bis zur Rendezvous Time zuzüglich der Zeit, welche für die Übertragung eines Datenpaketes mit maximaler Länge und anschließendem ACK benötigt wird, deaktiviert gehalten. Dies ist eine weitere Optimierung hinsichtlich des Energiesparens. Denn der Knoten berechnet, zu welchen der nächsten Abtastzeitpunkten der Kanal mit, für ihn irrelevanten, Datenverkehr belegt ist. Steht im Wake-Up-Paket die eigene Adresse und wurde ein Datenpaket zur Rendezvous Time empfangen, antwortet der Empfänger dem Sender mit einem (optionalen) ACK. In diesem ACK kann der Empfänger dem Sender außerdem Informationen über sein Abtastverhalten (CSL-Phasen) mitteilen, sodass der Sender die folgenden Sendevorgänge im Falle einer Unicast-Datenübertragung (siehe weiter unten) optimieren kann.

Es kann vorkommen, dass die Nutzdaten größer sind, als das größte IEEE 802.15.4-Datenpaket. In diesem Fall werden die Nutzdaten auf mehrere IEEE 802.15.4-Datenpakete aufgeteilt und nacheinander übertragen. Um den Durchsatz zu erhöhen und um Energie zu sparen, damit also die Prozedur nicht für jedes Datenpaket wiederholt werden muss und Wake-Up-Pakete gesendet werden müssen usw., kann der Sender in jedem Datenpaket, auf welches ein weiteres Datenpaket folgt, ein *Pending Bit* setzen. Ist das Pending Bit gesetzt, hält der Empfänger seine Funkeinheit im Empfangsmodus, um weitere Datenpakete, welche unmittelbar folgen, zu empfangen.

Die Abtast-Periode macCSLPeriod wird in Einheiten zu je zehn Symbolen angegeben und kann maximal *macCSLMaxPeriod* betragen (vgl. [51, S. 181]). macCSLMaxPeriod kann zwischen 0 und 65535 liegen und wird ebenfalls in Einheiten zu je zehn Symbolen angegeben. Zehn Symbole haben im 2,4-GHz-Band eine Dauer von 160 μs. Folglich kann der Kanal in Abständen zwischen 160 μs (macCSLPeriod = 1) und 10,4856 s (macCSLPeriod = 65.535) abgetastet werden. Hat macCSLPeriod den Wert 0, bedeutet dies, dass kein CSL verwendet wird. Die macCSLMaxPeriod wird für das gesamte WSN festgelegt.

Beim Senden von Datenpaketen wird unterschieden zwischen der *Unicast*- und der *Broadcast*-Datenübertragung, welche im Folgenden erläutert werden.

Unicast-Datenübertragung

Bei der Unicast-Datenübertragung gibt es nur einen bestimmten Empfänger. Der Sender sendet zunächst eine Reihe, unmittelbar aufeinander folgender, Wake-Up-Pakete, die sog. *Wakeup-Sequence*. Sendet der Sender zum ersten Mal an das Ziel, beträgt die Dauer der Wakeup-Sequence macCSLMaxPeriod (siehe oben), da der Sender die CSL-Phase des Empfängers noch nicht kennt. Dies wird als unsynchronisierte Datenübertragung (eng. *Unsynchronized Transmission*) bezeichnet (siehe Abb. 4.16). Die Wake-Up-Pakete enthalten die Rendezvous Time, zu welcher der Sender beabsichtigt, das Datenpaket mit den Nutzdaten zu senden. Der restliche Vorgang wurde bereits oben beschrieben.
Kennt der Sender die CSL-Phase des Empfängers bereits, genügt es, die Wakeup-Sequence kurz bevor der nächste Abtastzeitpunkt des Senders erwartet wird, zu senden. Dies wird als synchronisierte Datenübertragung (eng. *Synchronized Transmission*) bezeichnet (siehe Abb. 4.16). Die Wakeup-Sequence bei der synchronisierten Datenübertragung sollte lang genug sein, um das Auseinanderdriften der Zeitgeber von Sender und Empfänger zu kompensieren.

Der Algorithmus der Unicast-Datenübertragung lautet wie folgt (vgl. [51, S. 52]):

1. Mittels des CSMA-CA-Verfahrens wird versucht auf den Kanal zuzugreifen.
2. War das Pending Bit im letzten, durch ein ACK bestätigten, Datenpaket an den Empfänger gesetzt und ist die Zeit macCSLFramePendingWaitT[11] noch nicht abgelaufen, gehe zu Schritt 5.
3. Wenn es sich um eine synchronisierte Datenübertragung handelt, warte bis kurz vor den nächsten erwarteten Abtastzeitpunkt des Empfängers.
4. Für die Dauer der Wakeup-Sequence:

[11] Maximal etwas über eine Sekunde im 2,4-GHz-Band. Siehe [51, S. 181].

a) Berechne das nächste Wake-Up-Paket mit der Zieladresse des Empfängers und der Zeit bis zur Übertragung des Datenpaketes mit den Nutzdaten (Rendezvous Time).

b) Sende das Wake-Up-Paket.

5. Übertrage das Datenpaket mit den Nutzdaten.
6. Warte auf das ACK des Empfängers, wenn gefordert, aber maximal macEnhAck-WaitDuration[12] lang.
7. Wurde ein ACK empfangen, merke dir die CSL-Phase des Empfängers, welche in diesem ACK übermittelt wurde.
8. Wurde kein ACK empfangen, starte die Übertragung erneut (gehe zu Schritt 1).

Broadcast-Datenübertragung

Bei der Broadcast-Datenübertragung richtet sich ein Datenpaket an alle (sich in der Nähe befindlichen) Knoten. Der Ablauf ist derselbe wie bei der Unicast-Datenübertragung, mit dem Unterschied, dass es sich stets um eine unsynchronisierte Datenübertragung handelt und die Zieladresse die Broadcast-Adresse (16 Bit Kurzadresse 0xffff) ist.

Optional kann der Sender im Header seiner Datenpakete Informationen über seine CSL-Phase und -Periode übermitteln, damit auch der Empfänger sein Sendeverhalten optimieren, sprich die synchronisierte Datenübertragung nutzen kann, wenn Empfänger und Sender die Rollen tauschen. Diese Möglichkeit besteht sowohl bei der Unicast- als auch bei der Broadcast-Datenübertragung.

CSL und Channel-Hopping

Der CSL-Modus kann auch mit mehreren Kanälen gleichzeitig verwendet werden. In diesem Fall legt eine 32 Bit Bitmap WSN-weit die zu verwendenden Kanäle fest. Zu jedem Abtastzeitpunkt tastet der Sender im Round-Robin-Verfahren[13] einen anderen Kanal ab. Bei der unsynchronisierten Datenübertragung beträgt die Dauer der Wakeup-Sequence dann [Anzahl der verwendeten Kanäle] x [macCSLMaxPeriod]. Bei der synchronisierten Datenübertragung berechnet der Sender nicht nur den nächsten Abtastzeitpunkt, sondern auch den Kanal, den der Empfänger abtasten wird. In diesem Fall gibt die CSL-Phase die Dauer zwischen einem Abtastzeitpunkt und dem nächsten Abtastzeitpunkt auf demselben Kanal an.

Im Gegensatz zu IEEE 802.15.4 TSCH ist der CSL-Modus dezentral, erfordert keinen aufwändigen Planungsalgorithmus und die Implementierung sollte verhältnismäßig einfach ausfallen. Die Knoten benötigen jedoch genügend Speicher, um sich die CSL-Phasen und -Perioden ihrer Nachbar-Knoten zu merken (oder verwenden einfach stets die unsynchronisierte Datenübertragung). Zudem ist der CSL-Modus an keine bestimmte Netzwerktopologie gebunden.

[12] Zwischen 0 μs und 65535 μs (unabhängig vom Modulationsverfahren und Frequenzband). Standardwert ist 864 μs (siehe [51, S. 171]).

[13] D. h. es wird jedesmal der Reihe nach ein anderer Kanal verwendet. Wurde der letzte Kanal abgetastet, wird wieder von vorn begonnen.

4.2.8. IEEE 802.15.4 RIT

RIT steht für Receiver Initiated Transmissions und ist nicht zu verwechseln mit der indirekten Datenübertragung von IEEE 802.15.4 (siehe Kap. 3.1.7). Abb. 4.17 stellt die prinzipielle Funktionsweise des RIT-Modus von IEEE 802.15.4 dar.

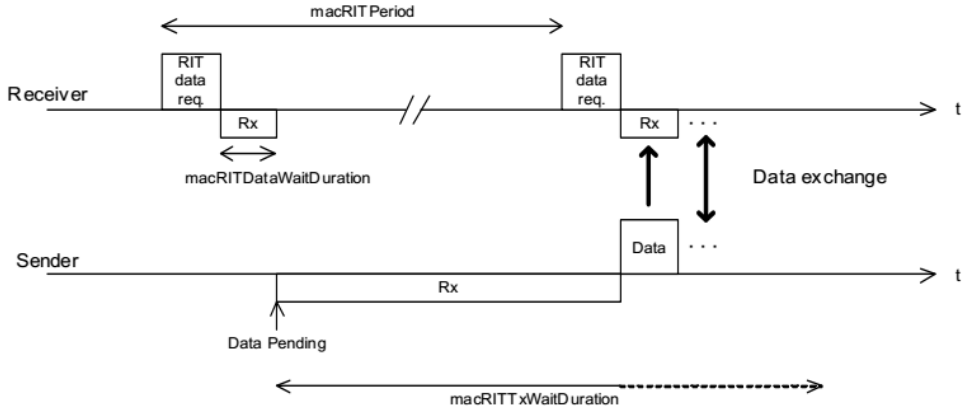


Abbildung 4.17.: IEEE 802.15.4 RIT
Quelle: [51, S. 54]

Empfangen von Datenpaketen

Im RIT-Modus sendet der Empfänger unter Verwendung des CSMA-CA-Verfahrens periodisch und im Abstand von *macRITPeriod* sog. *RIT-Data-Requests* (siehe Abb. 4.17). Empfänger der RIT-Data-Requests ist entweder ein bestimmtes Ziel oder die Broadcast-Adresse, mit welcher alle in Reichweite befindlichen Ziele angesprochen werden. macRITPeriod wird in Einheiten von *aBaseSuperframeDuration* angegeben und kann Werte zwischen 0 und 16.777.215 annehmen (vgl. [51, S. 182]). aBaseSuperframeDuration entspricht im 2,4-GHz-Band einer Dauer von 15,36 ms. Folglich kann die RIT-Periode, also der zeitliche Abstand von Anfang zu Anfang zweier aufeinanderfolgender RIT-Data-Requests, zwischen 15,36 ms und über 71,58 Stunden betragen. Hat macRITPeriod den Wert 0, bedeutet dies, dass der RIT-Modus nicht verwendet wird.

Nach dem Senden jedes RIT-Data-Requests bleibt die Funkeinheit des Empfängers noch für die Dauer von *macRITDataWaitDuration* im Empfangsmodus, um auf potentielle Daten eines Senders zu warten. macRITDataWaitDuration wird in Einheiten von aBaseSuperframeDuration angegeben und kann Werte zwischen 0 und 255 annehmen, was im 2,4-GHz-Band einer Dauer zwischen 0 s und etwa 3,92 s entspricht. Der Zeitraum nach macRITDataWaitDuration und vor dem Senden des nächsten RIT-Data-Requests kann wiederum zum Energiesparen genutzt werden. Wird ein Datenpaket während der Zeit macRITDataWaitDuration detektiert, wird dieses empfangen. RIT-Data-Requests von anderen Knoten werden jedoch ignoriert.

Optional kann der Empfänger mit einem RIT-Data-Request noch vier Bytes Daten übermitteln. Diese vier Bytes enthalten Informationen über die Zeit T_0 bis zur ersten Empfangsperiode, die Anzahl N der Wiederholungen und das Intervall T der Empfangsperioden (siehe Abb. 4.18). Eine Empfangsperiode hat die Dauer macRITPeriod, währenddessen der Empfänger seine Funkeinheit in den Empfangsmodus versetzt und auf

potentielle Datenpakete wartet. Die Zeit unmittelbar nach einem RIT-Data-Request bis zur ersten Empfangsperiode und die Zeit zwischen zwei aufeinanderfolgenden Empfangsperioden kann wieder zum Energiesparen genutzt werden. Selbiges gilt natürlich auch für die Zeit zwischen der letzten Empfangsperiode und dem nächsten RIT-Data-Request.

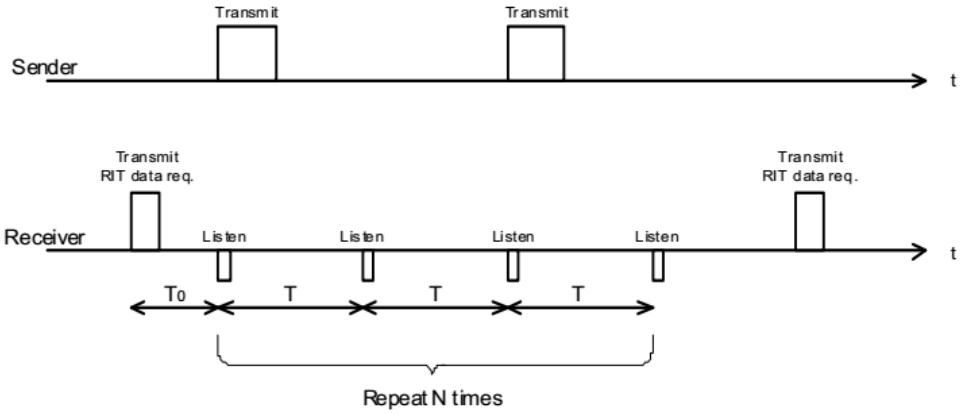


Abbildung 4.18.: IEEE 802.15.4 RIT bei einem Data-Request mit vier Bytes Payload
Quelle: [51, S. 54]

Senden von Datenpaketen

Hat ein Sender ein Datenpaket zu übertragen, stoppt er zunächst die Übertragung von eigenen RIT-Data-Requests und versetzt seine Funkeinheit in den Empfangsmodus und wartet anschließend für die Dauer von *macRITTxWaitDuration* auf ein RIT-Data-Request des Empfängers. macRITTxWaitDuration wird in Einheiten von aBaseSuperframeDuration angegeben und kann Werte zwischen macRITPeriod und 16.777.215 annehmen. Während dieser Zeit werden alle anderen einkommenden Datenpakete ignoriert. Wurde ein entsprechendes RIT-Data-Request empfangen, versetzt der Sender seine Funkeinheit in den Sendemodus und sendet das Datenpaket (siehe Abb. 4.17). Enthielt das RIT-Data-Request zusätzliche vier Bytes (siehe oben bei *Empfangen von Datenpaketen*), kann der Sender seine Daten nur während der übermittelten Empfangsperioden senden (siehe Abb. 4.18). Die Zeit zwischen den Empfangsperioden kann auch vom Sender zum Energiesparen genutzt werden.

Broadcast-Datenübertragungen sind im RIT-Modus (mit Ausnahme der RIT-Data-Requests) nicht vorgesehen.

Wie IEEE 802.15.4 CSL, ist RIT dezentral und verhältnismäßig einfach zu implementieren und an keine bestimmte Netzwerktopologie gebunden. Jedoch wird für RIT kein Speicher benötigt, um sich die Phasen und Perioden der Nachbar-Knoten zu merken. Im Gegensatz zu LPL-MAC-Protokollen wie ContikiMAC und CSL, führt RIT immer Sendeoperationen aus, auch dann, wenn eigentlich keine Daten anfallen. Sendeoperationen sind, wie bereits erwähnt, meist teurer als das reine Empfangen. Außerdem belegen Sendeoperationen den Kanal und verhindern, dass andere Sender während dieser Zeit senden können. Um effizient zu sein, müssen sich die Sender bei LPL-MAC-Protokollen auf die Phasen, also die Kanalabtastperioden der Empfänger synchronisieren. Das setzt exakte Zeitgeber und die regelmäßige Kommunikation vorraus, um die Synchronizität zu erhalten. Bei LPP-MAC-Protokollen wie RIT, ist dies entspannter. Die Zeitgeber können hier

weiter auseinanderdriften, ohne dass die Sendeoperationen länger werden müssen, um die Kanalabtastzeitpunkte der Empfänger nicht zu verfehlen.

4.3. Vergleich von ContikiMAC und IEEE 802.15.4 CSL

ContikiMAC und IEEE 802.15.4 CSL sind beides LPL-MAC-Protokolle, welche von ihrer Funktionsweise her sehr ähnlich sind. CSL bietet Channel-Hopping, welches ContikiMAC von Haus aus nicht bietet. MiCMAC (siehe Kap. 4.2.6) erweitert ContikiMAC jedoch um diese fehlende Fähigkeit. Ein weiterer Vorteil von CSL ist, dass es standardisiert ist. Im Folgenden werden ContikiMAC und CSL für den Einkanalbetrieb, also ohne Berücksichtigung von Channel-Hopping, miteinander verglichen.

4.3.1. Die erste Kontaktaufnahme

Sendet ein Sender zum ersten Mal an einen Empfänger, kennt er das Abtastverhalten, also die CCA- respektive die CSL-Phasen des Empfängers noch nicht. Die Phasen-Optimierung von ContikiMAC greift dann nicht bzw. spricht man bei CSL dann von der unsynchronisierten Datenübertragung. Abb. 4.19 veranschaulicht beispielhaft die erste Kontaktaufnahme für beide MAC-Protokolle.

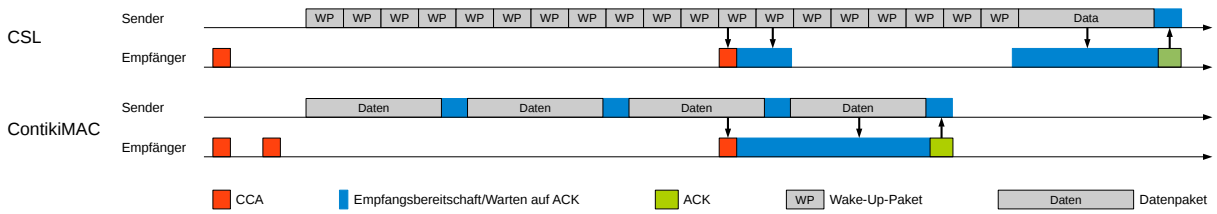


Abbildung 4.19.: Vergleich der ersten Kontaktaufnahme zwischen einem Sender und Empfänger bei IEEE 802.15.4 CSL und ContikiMAC
Quelle: Eigene Darstellung

Bei CSL sendet der Sender zunächst eine Wake-Up-Sequenz fester Länge, bestehend aus mehreren einzelnen Wake-Up-Paketen, welche lückenlos gesendet werden (siehe Abb. 4.19 oben). Im Beispiel registriert der Empfänger bei der Kanalabtastung mit Hilfe eines CCAs das 11. Wake-Up-Paket von links. Der Empfänger bleibt daraufhin eine längere Zeit in Empfangsbereitschaft, um das nachfolgende Wake-Up-Paket (das 12. von links) zu empfangen, da das 11. Wake-Up-Paket nicht vollständig empfangen werden konnte. Der Empfänger liest aus den Informationen des 12. Wake-Up-Pakets, dass er Empfänger des Datenpaketes ist, welches auf die Wake-Up-Sequenz folgt. Außerdem entnimmt er dem 12. Wake-Up-Paket die Rendezvous Time, also die Zeit, wann der Sender beabsichtigt, das Datenpaket zu senden. Der Empfänger wechselt dann wieder in den Energiesparmodus und verlässt diesen kurz vor der Rendezvous Time wieder, um in den Empfangsmodus zu gehen und das Datenpaket zu empfangen. Anschließend sendet er ein ACK zur Bestätigung an den Sender zurück. Der Sender errechnet aus dem ACK die CSL-Phase des Empfängers, um beim nächsten Mal eine synchronisierte Datenübertragung vornehmen zu können.

Anstatt Wake-Up-Pakete zu senden, sendet ContikiMAC fortlaufend Kopien des Datenpaketes (siehe Abb. 4.19 unten). Auf jedes Datenpaket folgt eine kurze Pause. Im Beispiel registriert das erste CCA der nächsten CCA-Phase des Empfängers das dritte Datenpaket von links. Da der Empfänger das dritte Datenpaket nicht vollständig empfangen konnte, bleibt er noch eine Zeit lang in Empfangsbereitschaft, um das vierte Datenpaket vollständig zu empfangen. Da die Zieladresse in den Datenpaketen mit der Adresse des Empfängers übereinstimmt, nutzt der Empfänger die Pause nach dem vierten Datenpaket, um ein ACK an den Sender zu senden. Der Sender stellt daraufhin die Übertragung der Datenpakete ein.

Bei der ersten Kontaktaufnahme gestaltet sich ContikiMAC aus Sicht des Senders um einiges effizienter: Statt einer Wakeup-Sequenz stets fester Länge zu senden, ermöglichen die Pausen zwischen den Datenpaketen dem Empfänger, den erfolgreichen Empfang eines Datenpaketes mit einem ACK zu bestätigen, sodass der Sender das Senden der Datenpakete dann vorzeitig abbrechen kann. Die Dauer einer Wake-Up-Sequenz bei CSL bzw. die maximale Sendedauer der Datenpakete bei ContikiMAC, entspricht einem Zyklus des Empfängers, d. h. der Zeit, von Anfang zu Anfang zwei aufeinanderfolgender Kanalabtastungen. Statistisch gesehen tastet ein Empfänger den Kanal zur Hälfte der Sendezeit der Wake-Up-Sequenz respektive der Datenpakete ab (vgl. [13, S. 310]). Das bedeutet, dass der Sender unter ContikiMAC im statistischen Mittel das Senden der Datenpakete bereits nach etwas mehr als der Hälfte der Dauer eines Zyklus einstellen kann.

Aus Sicht des Empfängers, läuft das Empfangen eines Datenpaketes bei der ersten Kontaktaufnahme unter CSL im Durchschnitt effizienter ab. Das Empfangen eines Datenpaketes ist hier ähnlich zur Broadcast-Datenübertragung. Berechnungen dazu folgen in Kap. 4.3.3.

4.3.2. Die synchronisierte Unicast-Datenübertragung

Im Folgenden soll die synchronisierte Unicast-Datenübertragung analysiert werden. D. h. es werden Daten an einen bestimmten Sender, unter Verwendung der Phasen-Optimierung von ContikiMAC respektive der synchronisierten Datenübertragung von CSL gesendet. Dabei werden zwei Extreme betrachtet: Der Versand des kleinstmöglichen (a) und größtmöglichen (b) IEEE 802.15.4-Datenpaketes.

a) Die Größe des PPDU-Payloads muss zwischen 9 und 127 Bytes liegen. (Bei PPDUs mit einem Payload von weniger als 9 Bytes, handelt es sich um Spezialpakete, wie ACKs.) Für die kleinstmögliche PPDU ergibt sich somit eine Größe von 15 Bytes (9 Bytes PPDU-Payload zuzüglich 6 Bytes für die Präambel der PPDU etc.). Allerdings genügt die Sendedauer $t_s = \frac{15\ Bytes*8}{250000\ kbit/s} = 48\ \mu s$ für das kürzeste Datenpaket nicht der Restriktion $t_s > t_r + t_c + t_r \Rightarrow 48\ \mu s \not> 286\ \mu s + 500\ \mu s + 286\ \mu s = 1072\ \mu s$ von ContikiMAC (siehe Kap. 4.2.5). Aus diesem Grund werden für alle weiteren Berechnungen 34 Bytes als Größe für das kleinstmögliche Datenpaket mit einer Sendedauer von $\frac{34\ Bytes*8}{250000\ kbit/s} = 1,088\ ms$ angenommen. Wenn der Sender gut auf den Empfänger synchronisiert ist, genügt bei CSL ein Wake-Up-Paket, welches zeitlich auf ein CCA des Empfängers fällt. Die Informationen im Wake-Up-Paket (Zieladresse und Rendezvous Time) sind in diesem Falle irrelevant, d. h. das Datenpaket kann unmittelbar folgen (siehe Abb. 4.20 oben). Bei ContikiMAC genügt es zwei Datenpakete zu senden. Das erste Datenpaket muss zeitlich auf einem der beiden

CCAs einer CCA-Phase des Empfängers liegen. Da dieses Datenpaket nicht vollständig empfangen werden kann, muss ein weiteres Datenpaket abgewartet werden. Im Idealfall detektiert das CCA des Empfängers gerade noch das Ende eines Wake-Up-Paketes bei CSL (siehe Abb. 4.20 oben rechts) bzw. das erste CCA einer CCA-Phase detektiert das Ende des ersten Datenpaketes bei ContikiMAC (siehe Abb. 4.20 unten rechts). Die Zeit, welche der Empfänger in Empfangsbereitschaft bleiben muss, ist dann am geringsten. Im schlechtesten Fall liegt das CCA bzw. das letzte der beiden CCAs einer CCA-Phase bei ContikiMAC genau am Anfang eines Wake-Up-Paketes respektive eines Datenpaketes (siehe Abb. 4.20 links).

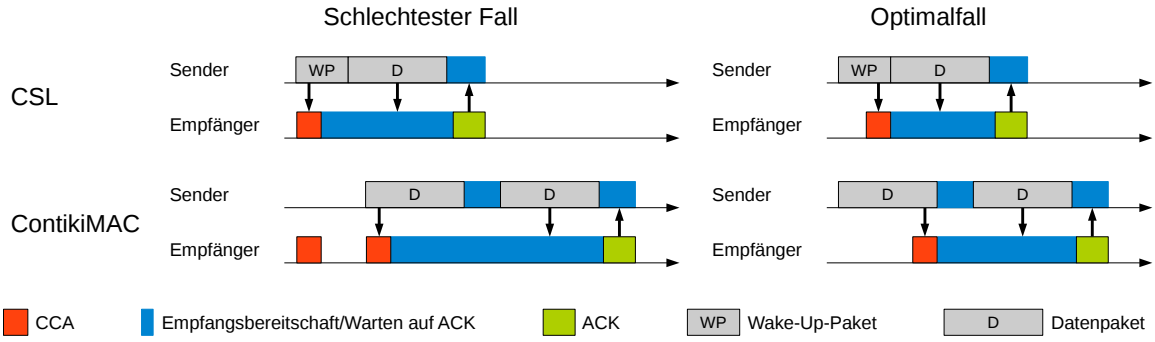


Abbildung 4.20.: ContikiMAC und CSL: Schlechtester Fall und Optimalfall beim kleinstmöglichen IEEE 802.15.4-Datenpaket
Quelle: Eigene Darstellung

Für folgende Berechnungen werden die Zeiten aus Tabelle 4.2 auf S. 59 und die Verwendung des 2,4-GHz-Bandes angenommen. Zur Vereinfachung wird außerdem das Senden und Empfangen des ACKs, welches ohnehin bei allen Varianten gleich abläuft, Pufferzeiten (Guardtimes) zur Kompensation der Ungenauigkeit der Zeitgeber, sowie Umschaltzeiten, welche die Funkeinheiten benötigt, um zwischen verschiedenen Modis, wie Empfangs- und Sendemodus, zu wechseln, vernachlässigt.

Bei CSL: Die Größe eines Wake-Up-Paketes von CSL auf PHY-Ebene (PPDU-Größe) beträgt stets 18 Bytes (entspricht einer Sendedauer von $\frac{18\ Bytes*8}{250000\ kbit/s} = 0,576\ ms$, vgl. [51, S. 79]). Der Sender sendet das 18 Bytes große Wake-Up-Paket und anschließend das 34 Bytes große Datenpaket. Der Sendevorgang dauert damit $0,576\ ms + 1,088\ ms = 1,664\ ms$. Der Empfänger benötigt im schlechtesten Fall genau so lange zum Empfangen (Dauer eines CCAs zuzüglich der Empfangsbereitschaft bis zum Ende des Datenpaketes).
Für den Optimalfall ändert sich beim Sender nichts. Der Empfänger benötigt jedoch nur noch die Dauer eines CCAs (siehe t_r in Tabelle 4.2) zuzüglich der Zeit zum Empfangen des Datenpaketes, d. h. also $0,268\ ms + 1,088\ ms = 1,356\ ms$.

Bei ContikiMAC: Der Sender benötigt stets die Zeit zum Senden der beiden Datenpakete $2 * 1,088\ ms = 2,176\ ms$ zuzüglich einer kurzen Pause von 400 μs (siehe t_i in Tabelle 4.2) zwischen den Datenpaketen, welche er in Empfangsbereitschaft verbringt, um auf das ACK zu warten (was aber genau genommen hier nicht nötig wäre, da klar ist, dass der Empfänger zu diesem Zeitpunkt noch kein ACK sendet). Daraus ergibt sich eine Gesamtaktivzeit der Funkeinheit von $0,4\ mss + 2,176\ ms = 2,576\ ms$. Im schlechtesten Fall dauert die Empfangsbereitschaft einschließlich des zweiten CCAs genau so lange. Hinzu addiert sich noch der erste CCA, der das erste Datenpaket nicht

detektieren konnte. Daraus ergibt sich für den schlechtesten Fall eine Gesamtzeit für die Empfangsbereitschaft des Empfängers von $0,268\ ms + 2,576\ ms = 2,844\ ms$. Im Optimalfall detektiert das erste CCA einer CCA-Phase gerade noch das Ende des ersten Datenpaketes. In diesem Fall ergibt sich die Gesamtdauer der Empfangsbereitschaft aus einer CCA-Dauer, Pausenzeit und aus der Zeit der Übertragung eines Datenpaketes, also $0,268\ ms + 0,4\ ms + 1,088\ ms = 1,756\ ms$.

b) Analog werden die Berechnungen jetzt für das größtmögliche IEEE 802.15.4-Datenpaket mit einer PPDU-Größe von 133 Bytes (entspricht einer Sendedauer von 4,256 ms im 2,4-GHz-Band) durchgeführt. Abb. 4.21 veranschaulicht wieder den schlechtesten Fall und den Optimalfall für beide MAC-Protokolle.

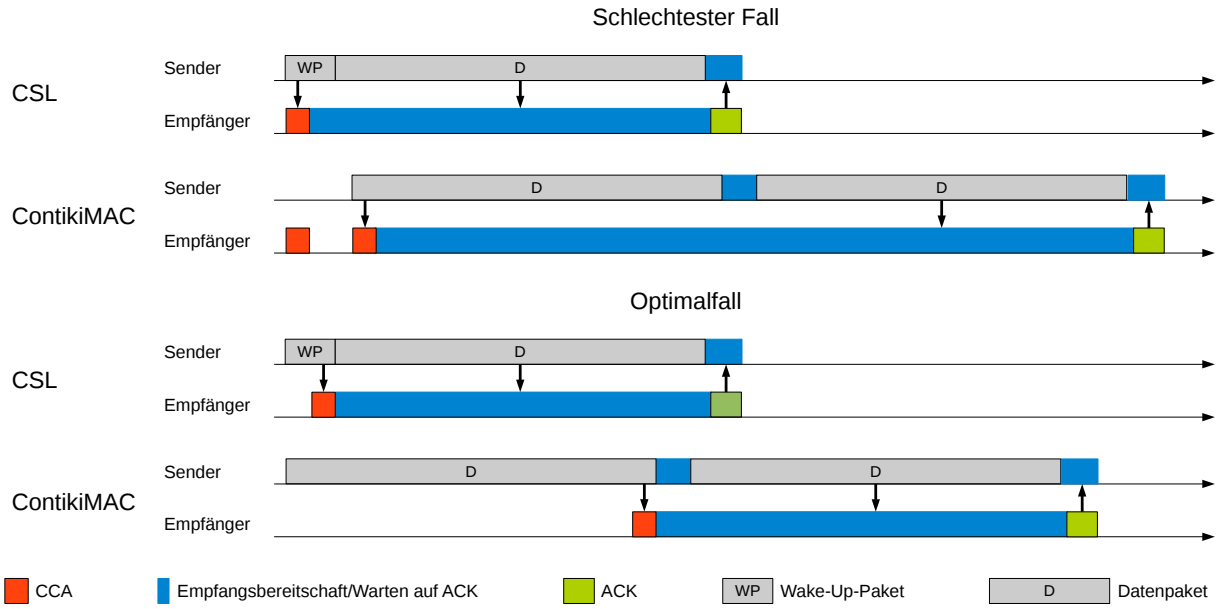


Abbildung 4.21.: ContikiMAC und CSL: Schlechtester Fall und Optimalfall bei größtmöglichen IEEE 802.15.4-Datenpaket
Quelle: Eigene Darstellung

Für CSL: Die Sendedauer für den Sender ergibt sich für beide Fälle (schlechtester Fall und Optimalfall) wieder aus der Sendedauer für ein Wake-Up-Paket und ein Datenpaket: $0,576\ ms + 4,256\ ms = 4,832\ ms$ Im schlechtesten Fall detektiert der CCA des Empfängers genau den Anfang des Wake-Up-Paketes. Die Dauer der Empfangsbereitschaft (einschließlich des CCAs) beträgt dann ebenfalls 4,832 ms. Im Optimalfall detektiert der CCA genau das Ende des Wake-Up-Paketes. Dann ergibt sich die Dauer der Empfangsbereitschaft aus der Dauer eines CCAs zuzüglich der Sendedauer des Datenpaketes und beträgt damit $0,268\ ms + 4,256\ ms = 4,524\ ms$.

Für ContikiMAC: Die Aktivzeit der Funkeinheit des Senders ergibt sich in beiden Fällen aus der Sendedauer für zwei Datenpakete zuzüglich einer Pausenzeit von t_i, also beträgt sie $2 * 4,256\ ms + 0,4\ ms = 8,912\ ms$. Im schlechtesten Fall detektiert das zweite CCA des Empfängers genau den Anfang des ersten Datenpaketes. Die Dauer der Empfangsbereitschaft ergibt sich dann aus der Dauer beider CCAs zuzüglich der Dauer vom Ende des zweites CCAs bis zum Ende des zweiten Datenpaketes und ergibt sich damit zu $0,268\ ms + 8,912\ ms = 9,18\ ms$. Im Idealfall detektiert der erste

der beiden CCAs einer CCA-Phase gerade noch so das Ende des zweiten Datenpaketes. Die Dauer der Empfangsbereitschaft setzt sich dann zusammen aus der Dauer eines CCAs, einer Pausenzeit t_i und der Sendedauer eines Datenpaketes. Daraus ergibt sich also $0,268\ ms + 0,4\ ms + 4,256\ ms = 4,924\ ms$.

Tabelle 4.3 fasst die Ergebnisse obiger Berechnungen zusammen. Die Werte in Klammern geben den Mittelwert an.

	Sender	**Empfänger**
	Für kleinstmögliches IEEE 802.15.4-Datenpaket	
CSL	1,664 ms	1,356 - 1,664 ms (1,51 ms)
ContikiMAC	2,576 ms	1,756 - 2,844 ms (2,3 ms)
	Für größtmögliches IEEE 802.15.4-Datenpaket	
CSL	4,832 ms	4,524 ms - 4,832 ms (4,678 ms)
ContikiMAC	8,912 ms	4,924 ms - 9,18 ms (7,052 ms)
	Für alle IEEE 802.15.4-Datenpaket-Größen	
CSL	1,664 - 4,832 ms (3,248 ms)	1,356 - 4,832 ms (3,09 ms)
ContikiMAC	2,576 - 8,912 ms (5,744 ms)	1,756 - 9,18 ms (5,468 ms)

Tabelle 4.3.: Ergebnisse der Berechnungen zum Vergleich der synchronisierten Unicast-Datenübertragung zwischen CSL und ContikiMAC

Die obigen Berechnungen bezogen sich auf den Fall, wenn der Sender sehr gut auf den Empfänger synchronisiert ist. Dann muss nur noch ein Wake-Up-Paket und ein Datenpaket unter CSL respektive zwei Datenpakete unter ContikiMAC gesendet werden. Lediglich leichte Abweichungen wurden angenommen, deren Extremfälle, schlechtester Fall und Optimalfall, untersucht wurden. Die Ergebnisse in Tabelle 4.3 zeigen, dass CSL in allen Fällen, insbesondere für den Empfänger, etwas besser ist, als ContikiMAC. Die Ergebnisse von CSL gegenüber ContikiMAC verbessern sich umso mehr, je größer die IEEE 802.15.4-Datenpakete werden. Unter Verwendung von 6LoWPAN liegt die Größe eines IEEE 802.15.4-Datenpaketes bei mindestens 92 Bytes (maximale Größe einer IEEE 802.15.4-PPDU abzüglich der maximalen Payload-Größe unter 6LoWPAN von 41 Bytes) und damit bereits über der durchschnittlichen Größe eines IEEE 802.15.4-Datenpaketes.

CSL ist damit aus Sicht des Senders um den Faktor $\frac{5,744\ ms}{3,248\ ms} = 1,77$ und aus Sicht des Empfängers ebenfalls um den Faktor $\frac{5,468\ ms}{3,09\ ms} = 1,77$ effizienter, als ContikiMAC.

4.3.3. Die Broadcast-Datenübertragung

Die Broadcast-Datenübertragung wird bei CSL wie eine unsynchronisierte Datenübertragung durchgeführt, mit dem Unterschied, dass die Empfänger das Datenpaket nicht mit einem ACK bestätigen. Zunächst wird also die Wake-Up-Sequenz für die Dauer eines kompletten Zyklus gesendet. Anschließend folgt das Datenpaket. Der obere Teil in Abb. 4.22 veranschaulicht dies.

ContikiMAC sendet Broadcast-Datenpakete ohne Phasen-Optimierung, wie bei der ersten Kontaktaufnahme (siehe Kap. 4.3.1), mit dem Unterschied, dass die Empfänger Broadcast-Datenpakete ebenfalls nicht mit einem ACK bestätigen. Um sicherzugehen,

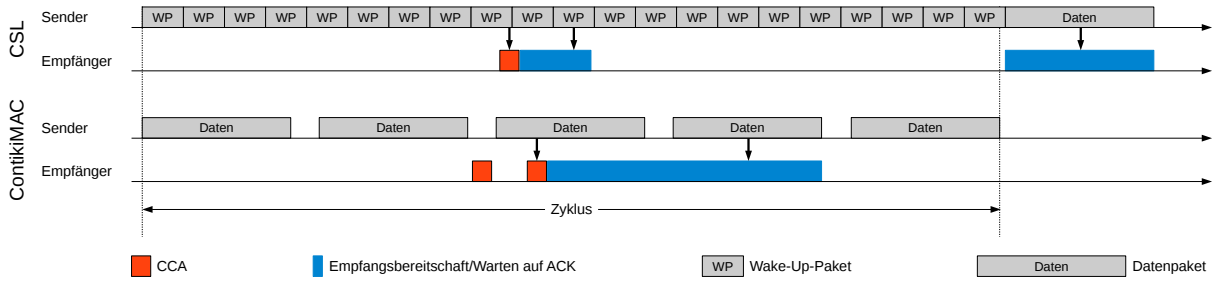


Abbildung 4.22.: Vergleich der Broadcast-Datenübertragung bei ContikiMAC und CSL
Quelle: Eigene Darstellung

dass jeder Empfänger eine Kopie des Datenpaketes erhalten hat, wiederholt ContikiMAC die Datenpakete für die komplette Dauer eines Zyklus (siehe Abb. 4.22 unten).
Aus Sicht des Senders, ist die Broadcast-Datenübertragung von ContikiMAC effizienter, da CSL eine längere Sendedauer benötigt, da das Datenpaket erst am Ende der Wake-Up-Sequenz gesendet wird. Der Faktor der Verbesserung von ContikiMAC gegenüber CSL aus Sicht des Senders soll anhand eines Beispiels berechnet werden. Für das Datenpaket wird eine mittlere Größe von 65 Bytes angenommen, was einer Sendedauer von 2,08 ms entspricht. Für die Kanalabtastrate des Empfängers wird 8 Hz (der Standardwert von ContikiMAC) angenommen. D. h. also der Empfänger tastet den Kanal alle 125 ms ab.

Bei CSL: Zunächst werden genügend Wake-Up-Pakete gesendet, um die gesamte Zykluszeit von 125 ms abzudecken. Es müssen also $\frac{125\ ms}{0{,}576\ ms} = 217,01 \Rightarrow 218$ Wake-Up-Pakete gesendet werden, was einer Sendedauer von $218 * 0,576\ ms = 125,568\ ms$ entspricht. Zieht man die Sendezeit für das Datenpaket hinzu, ergibt sich eine Gesamtsendedauer von $125,568\ ms + 2,08\ ms = 127,648\ ms$.

Bei ContikiMAC: Um mindestens auf eine Dauer von 125 ms zu kommen, muss das Datenpaket n mal gesendet werden. Die Sendedauer der Datenpakete inklusive der Pausenzeiten t_i beträgt $n(2,08\ ms + 0,4\ ms) - 0,4\ ms$. Bei obigen Annahmen beträgt n: $125\ ms \leq n(2,08\ ms + 0,4\ ms) - 0,4\ ms \Rightarrow n \geq \frac{125\ ms+0{,}4\ ms}{2{,}08\ ms+0{,}4\ ms} = 50.56$ Das Datenpaket muss also 51 mal gesendet werden, was einer Sendedauer von $51 * (2,08\ ms + 0,4\ ms) - 0,4\ ms = 126,08\ ms$ entspricht.

ContikiMAC ist also aus Sicht des Senders um den Faktor $\frac{127{,}648\ ms}{126{,}08\ ms} = 1,01$ effizienter als CSL. Wiederholt man die Berechnungen für Kanalabtastraten von 1 Hz, 2 Hz, 4 Hz, 8 Hz und 16 Hz und für das kleinstmögliche, mittlere und größtmögliche Datenpaket, stellt man fest, dass der Faktor geringfügig zwischen 1 und 1,04 schwankt.

Im Folgenden soll die Effizienz aus Sicht des Empfängers betrachtet werden.

Bei CSL: Im Beispiel in Abb. 4.22 oben registriert das CCA des Empfängers die Wake-Up-Sequenz etwa zur Hälfte ihrer Sendedauer. Das erste Wake-Up-Paket, welches das CCA registriert hat, kann noch nicht empfangen werden, daher bleibt der Empfänger so lange in Empfangsbereitschaft, bis er ein vollständiges, nämlich das nächste Wake-Up-Paket empfangen hat. Ihm entnimmt er die Rendezvous Time und kann die Empfangsbereitschaft daraufhin bis zum Beginn des Datenpaketes einstellen. Im Beispiel in Abb. 4.22 detektiert das CCA des Empfängers die Wake-Up-Sequenz über das Ende des neunten

und den Anfang des zehnten Wake-Up-Paketes von links. Weder das neunte, noch das zehnte, sondern erst das elfte Wake-Up-Paket kann damit empfangen werden. Für die Berechnung der oberen Grenze der Dauer von der Detektion der Wake-Up-Sequenz bis zum erfolgreichen Empfang eines Wake-Up-Paketes, soll ein ähnlicher Sachverhalt angenommen werden: Das CCA liegt am Ende eines Wake-Up-Paketes und gerade noch am Anfang des darauffolgenden Wake-Up-Paketes, was zur Folge hat, dass erst ein drittes Wake-Up-Paket vollständig empfangen werden kann. Die obere Grenze ergibt sich damit aus der Dauer eines CCAs und zwei Wake-Up-Paketen, also $0,268\ ms + 2 * 0,576\ ms = 1,42\ ms$. Im Optimalfall detektiert das CCA das Ende eines Wake-Up-Paketes und so kann das nachfolgende Wake-Up-Paket sofort empfangen werden. Die untere Grenze ergibt sich damit zu $0,268\ ms + 0,576\ ms = 0,844\ ms$. Die Zeit, welche ein Empfänger von der Detektion der Wake-Up-Sequenz bis zum Empfang eines Wake-Up-Paketes benötigt, beträgt damit zwischen 0,844 ms und 1,42 ms. Zieht man nun noch die Sendedauer für das kleinst- (1,088 ms) und größtmögliche (4,256 ms) IEEE 802.15.4-Datenpaket in Betracht, bewegt sich die benötigte Zeit zum Empfangen eines Broadcast-Datenpaketes im Bereich zwischen 1,932 ms und 5,676 ms und hat damit einen Mittelwert von 3,809 ms.

Bei ContikiMAC: Im schlechtesten Fall hat das erste CCA der CCA-Phase des Empfängers ein Datenpaket knapp verfehlt und liegt damit also am Anfang der Pausenzeit t_i zwischen zwei Datenpaketen, wie im Beispiel in Abb. 4.22 unten gezeigt. Das hat zur Folge, dass erst das zweite CCA, nach einer Pause von $t_c = 0,5\ ms$, ein Datenpaket detektiert. Dieses Datenpaket kann aber noch nicht empfangen werden, weshalb der Empfänger in Empfangsbereitschaft bleibt, um ein vollständiges, also das nächste Datenpaket zu empfangen. Unter der Annahme des kleinstmöglichen IEEE 802.15.4-Datenpaketes mit $t_p = 1,088\ ms$ ergibt sich die obere Grenze der benötigten Zeit zu $t_p - (t_c - t_i) + t_i + t_p = 1,088\ ms - (0,5\ ms - 0,4\ ms) + 0,4\ ms + 1,088\ ms = 2,476\ ms$. Unter der Annahme des größtmöglichen IEEE 802.15.4-Datenpaketes mit $t_p = 4,256\ ms$ ergibt sich die obere Grenze zu $4,256\ ms - (0,5\ ms - 0,4\ ms) + 0,4\ ms + 4,256\ ms = 8,812\ ms$. Im Optimalfall liegt das zweite CCA der CCA-Phase unmittelbar vor einem Datenpaket, jedoch wird das zweite CCA nicht ausgeführt, da das erste CCA bereits das vorherige Datenpaket detektiert hat (siehe Abb. 4.23).

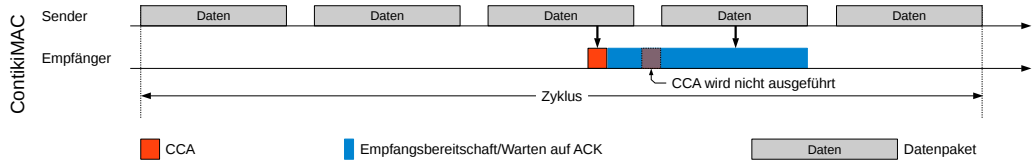


Abbildung 4.23.: Die optimale Lage einer CCA-Phase bei ContikiMAC, welche zur geringsten Dauer der Empfangsbereitschaft führt

Quelle: Eigene Darstellung

Im Optimalfall ergibt sich die untere Grenze der Dauer der Empfangsbereitschaft bei Annahme des kleinstmöglichen IEEE 802.15.4-Datenpaketes dann zu $2 * t_r + t_c + t_p = 2 * 0,268\ ms + 0,5\ ms + 1,088\ ms = 2,124\ ms$ und bei Annahme des größtmöglichen IEEE 802.15.4-Datenpaketes zu $2 * 0,268\ ms + 0,5\ ms + 4,256\ ms = 5,292\ ms$. ($t_r$ ist die Dauer eines CCAs.) Im Gesamtergebnis verbringt der Empfänger also eine Zeit zwischen 2,124 ms und 8,812 ms, d. h. eine mittlere Zeit von 5,468 ms in Empfangsbereitschaft, um ein Broadcast-Datenpaket zu empfangen. Bei CSL betrug diese Zeitspanne zwischen 1,932 ms und 5,676 ms, mit einem Mittelwert von 3,809 ms.

Vergleicht man nun die errechneten Zeiten miteinander, fällt auf, dass die Zeitspanne bei ContikiMAC deutlich größer ausfällt (siehe Abb. 4.24). Zudem liegt der Bereich, in welchem sich die Werte für CSL bewegen, weit genug links auf der Skala, sodass ContikiMAC keine besseren Werte als CSL erzielen kann. Im Mittel ist die benötigte Zeit zum Empfangen eines Broadcast-Datenpaketes mit ContikiMAC um den Faktor $\frac{5{,}292\ ms}{3{,}809\ ms} = 1{,}39$ größer, als mit CSL. Die hier berechneten Ergebnisse für den Empfänger, lassen sich so auch für die ersten Kontaktaufnahme (siehe Kap. 4.3.1) übernehmen, da das Prinzip beim Empfangen eines Datenpaketes (mit Ausnahme des fehlenden ACKs) dort dasselbe ist, wie bei der Broadcast-Datenübertragung.

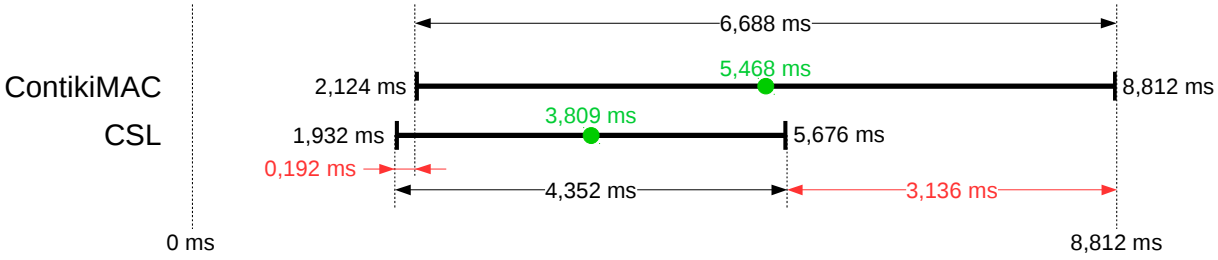


Abbildung 4.24.: Vergleich der Zeitspannen zwischen ContikiMAC und CSL, welche ein Empfänger benötigt, um ein Broadcast-Datenpaket zu empfangen
Quelle: Eigene Darstellung

4.3.4. Idle-Listening

Idle-Listening bedeutet, dass ein CCA ausgeführt, aber keine Daten detektiert wurden. Unter ContikiMAC kostet Idle-Listening mehr Energie, als unter CSL, da ContikiMAC für jedes Abtasten des Kanals zwei, also doppelt so viele CCAs verwendet. Dies ist notwendig, da ein einzelner CCA genau in die Pausenzeit zwischen zwei Datenpaketen fallen kann. Der Empfänger würde in diesem Fall die Datenübertragung nicht erkennen.

4.3.5. Zusammenfassung

Im Falle der ersten Kontaktaufnahme, d. h. bei der unsynchronisierten Datenübertragung bzw. ohne Phasen-Optimierung, ist ContikiMAC aus Sicht des Senders wesentlich effizienter. Jedoch ist der Fall der ersten Kontaktaufnahme relativ selten, es sei denn, die Nachbarn eines Knotens ändern sich häufig. Bei der Broadcast-Datenübertragung ist der Effekt nicht so dramatisch, aber auch hier ist ContikiMAC aus Sicht des Senders stets ein wenig effizienter. Bei der synchronisierten Unicast-Datenübertragung ist CSL aus Sicht des Senders jedoch das effizientere MAC-Protokoll. Wo CSL im Schnitt in allen Fällen effizienter ist als ContikiMAC, ist das Empfangen von Datenpaketen. Dies liegt in der Größe der Wake-Up-Pakete begründet, welche nicht mal ein Viertel der Größe eines durchschnittlichen IEEE 802.15.4-Datenpaketes haben.

Zusammengefasst sind die Vorteile beider MAC-Protokolle gegenüber des jeweils anderen:

- CSL
  - CSL ist standardisiert.
  - Unterstützt von Haus aus Channel-Hopping.
  - Effizienter bei der synchronisierten Unicast-Datenübertragung für den Sender (Faktor 1,77).
  - Für den Empfänger im Durchschnitt stets effizienter (Faktor 1,39 bis 1,77 bei der synchronisierten Unicast-Datenübertragung).
  - Idle-Listening kostet die Hälfte an Energie.
- ContikiMAC
  - Wesentlich effizienter für den Sender bei der ersten Kontaktaufnahme (im Mittel etwa um Faktor 2).
  - Geringfügig effizienter für den Sender bei der Broadcast-Datenübertragung (Faktor 1 bis 1,04).
  - Die Phasen-Optimierung kann deaktiviert werden, wenn die Knoten z. B. über zu wenig Speicher verfügen. Die Unicast-Datenübertragung läuft dann stets wie bei der ersten Kontaktaufnahme ab. ContikiMAC operiert dann aus Sicht des Senders effizienter, als CSL ohne die synchronisierte Datenübertragung.
  - Ändern sich die Nachbarn eines Knotens häufig, kommt es auch häufig zur ersten Kontaktaufnahme und diese ist unter ContikiMAC aus Sicht des Senders effizienter.

Die Wahl des besseren MAC-Protokolls kommt auf die Anwendung an. Ändern sich bspw. die Nachbarschaften der Knoten nur selten und ist der Datenverkehr im WSN überwiegend Unicast, so ist CSL die geeignetere Wahl.

Leider scheint es keine öffentlich zugänglichen Implementationen von CSL zu geben, wie auch schon die Autoren Guclu, Ozcelebi und Lukkien in ihrer Arbeit (siehe [32, S. 3]) festgestellt haben. Laut Guclu, Ozcelebi und Lukkien liegt dies daran, dass die gängigsten IEEE 802.15.4-Funkeinheiten kein lückenloses Versenden von mehreren Datenpaketen hintereinander unterstützen, was aber für das Senden der Wake-Up-Pakete notwendig ist.

4.4. LPL- im Vergleich zu LPP-MAC-Protokollen

Asynchronen MAC-Protokolle lassen sich in die zwei Unterarten LPL (z. B. X-MAC) und LPP (z. B. IEEE 802.15.4 RIT) unterteilen. Hinsichtlich eines niedrigen Energieverbrauchs zählen LPP-MAC-Protokolle allgemein zu der schlechteren Wahl. Dies liegt darin begründet, dass am Anfang eines jeden Zyklus zunächst ein Probe gesendet wird und anschließend für eine definierte Zeit auf eventuelle Datenpakete gewartet wird. Mit dem Probe werden die Nachbar-Knoten aufgefordert, jetzt ihre Daten zu senden. LPL-MAC-Protokolle hingegen verhalten sich passiv und benötigen pro Zyklus nur eine kurze Empfangs-Phase (oder kommen lediglich mit kurzen CCAs aus, siehe weiter unten). Bei

LPP-MAC-Protokollen müssen die Sender frühzeitig in Empfangsbereitschaft gehen, um die Probes der Empfänger zu empfangen. Bei IEEE 802.15.4 RIT geht ein Sender in Empfangsbereitschaft, sobald Daten anfallen, um auf das Probe des Empfängers zu warten (siehe Abb. 4.17 auf S. 68). Bei LPL-MAC-Protokollen würde der Sender stattdessen ein Strobe senden, d. h. eine Folge von Paketen, um die nächste Wachphase bzw. den Anfang des nächsten Zyklus des Empfängers abzupassen. Diese Strobes tragen erheblich zur Kanalauslastung bei, weshalb LPP-MAC-Protokolle einen Vorteil bieten können, wenn es darum geht, den Kanal geringstmöglich auszulasten. Dieser Vorteil ist aber erst ab einem gewissen Datenaufkommen im WSN gegeben. Ist das Datenaufkommen nur sehr gering, ist die Grundauslastung des Kanals durch die Probes wiederum höher, als bei einem LPL-MAC-Protokoll. Ein weiterer Nachteil von LPP-MAC-Protokollen ist, dass sie häufig keinen Broadcast-Datenverkehr unterstützen. Dies ist z. B. auch bei IEEE 802.15.4 RIT der Fall (vgl. [51, S. 57]).

Zwei dem Stand der Technik entsprechende LPL-MAC-Protokolle sind ContikiMAC und IEEE 802.15.4 CSL. Sie arbeiten wesentlich effizienter, als z. B. X-MAC und Contiki X-MAC. Letztere schalten die Funkeinheit am Anfang eines Zyklus für eine kurze Zeit (6,25 ms ist der Standardwert für Contiki X-MAC) in den Empfangsmodus und öffnen damit ein kurzes Zeitfenster, in welchem Pakete empfangen werden können. ContikiMAC und CSL hingegen tasten den Kanal am Anfang jedes Zyklus lediglich mit zwei (ContikiMAC) bzw. einem (CSL) kurzen CCA ab. Die Dauer eines CCAs ist von Funkeinheit zu Funkeinheit unterschiedlich. Beim ATmega 128RFA1 hat dieser eine Dauer von 0,268 ms im 2,4 GHz-Band. Vergleicht man z. B. ContikiMAC und Contiki X-MAC (beide mit ihrer jeweiligen Standardkonfiguration) miteinander, nimmt das Idle-Listening unter Coniki X-MAC fast zwölf mal mehr Zeit ($\frac{6{,}25\ ms}{2*0{,}268\ ms} = 11,66$) in Anspruch.

4.5. Synchrone im Vergleich zu Asynchronen MAC-Protokollen

Synchrone (auch TDMA-basierte) MAC-Protokolle teilen das Medium, wie bereits erwähnt, in Zeitschlitze gleicher Länge ein. Innerhalb eines Zeitschlitzes ist genau definiert, welcher Knoten senden darf (contention-free). Da es auch Anwendungsfälle gibt, bei denen Knoten nur nach Bedarf senden und die entsprechenden Zeitschlitze somit die meiste Zeit ungenutzt blieben (also Bandbreite verschenkt werden würde), unterstützen die meisten synchronen MAC-Protokolle eine weitere Art von Zeitschlitzen, innerhalb welcher mehrere Knoten um das Medium konkurrieren dürfen (contention-based). Synchrone MAC-Protokolle können daher gleichzeitig die Eigenschaften contention-free und contention-based haben, jedoch sind sie in der Regel mindestens contention-free. Asynchrone MAC-Protokolle sind hingegen stets nur contention-based. Für eine Übersicht, welches MAC-Protokoll aus Kap. 4.2 welcher Art zuzuordnen ist, siehe Anlage A.2.

Wie in Kap. 4.1.3 bereits beschrieben, haben synchrone MAC-Protokolle, wie IEEE 802.15.4 TSCH, den Nachteil, dass, im Gegensatz zu asynchronen MAC-Protokollen wie ContikiMAC oder IEEE 802.15.4 CSL, höhere Latenzen in Kauf genommen werden müssen. Dies liegt darin begründet, dass die Sender bei asynchronen MAC-Protokollen jederzeit (unter der Voraussetzung, dass der Kanal frei ist) senden dürfen. Synchrone MAC-Protokolle können diesen Nachteil mit ihren contention-based Zeitschlitzen — bei TSCH

sind dies bspw. die Shared Links — nur geringfügig aufwiegen. Auf der anderen Seite erreichen synchrone MAC-Protokolle geringere RDC-Werte, d. h. der Energiesparmodus kann noch intensiver genutzt werden. Dies liegt vorallem im deutlich reduzierten Idle-Listening begründet. Dies bedeutet also, dass der Energieverbrauch mit synchronen MAC-Protokollen tendenziell niedriger ist, als mit asynchronen MAC-Protokollen.

Die Autoren Ojo, Adami und Giordano verglichen in ihrer Arbeit (siehe [43]) die Performance der asynchronen MAC-Protokolle ContikiMAC, Contiki X-MAC, LPP[14] und des synchronen MAC-Protokolls TSCH. Für letzteres kam OpenWSN zum Einsatz, welches die Referenzimplementierung von TSCH enthält. Die Ergebnisse zeigten, dass ContikiMAC einen wesentlich niedrigeren RDC-Wert (8,73 %) als Contiki X-MAC (26,22 %) und LPP (28,35 %) aufweist (vgl. [43, S. 4]). Am besten schnitt jedoch TSCH, mit einem RDC-Wert von lediglich 2,3 % und dem damit niedrigsten Energieverbrauch, ab. Dies bestätigt die oben stehende Aussage, dass mit synchronen MAC-Protokollen ein niedrigerer Energieverbrauch erzielt werden kann.

IEEE 802.15.4 CSL und TSCH wurden in der Arbeit von Zhou u. a. verglichen (siehe [70]). Auch die Ergebnisse dieser Arbeit untermauern das oben bereits Erwähnte im Bezug auf den Unterschied zwischen synchronen und asynchronen MAC-Protokollen (geringerer RDC-Wert bzw. Energieverbrauch vs. geringere Latenz). Zhou u. a. simulierten verschiedene Szenarien für WSNs, jeweils unter Verwendung von CSL und TSCH. In allen Szenarien war der Energieverbrauch mit CSL deutlich höher, als mit TSCH (Faktor drei bis sechs). Umgekehrt war die Latenz mit TSCH stets viel größer, als mit CSL (Faktor fünf bis zehn).

Bei synchronen MAC-Protokollen stellt sich immer wieder die Frage nach dem richtigen Planungsalgorithmus und nach einer geeigneten Strategie zum Verteilen des Kommunikationsplans im WSN, damit jeder Knoten weiß, in welchem Zeitschlitz er senden darf oder empfangsbereit sein muss. 6LoWPAN über TSCH ist bspw. keine sehr triviale Sache, weshalb sich eigens die IETF-Arbeitsgruppe *IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH)* gegründet hat, welche sich diesem Problem annimmt (siehe [2] u. [56]).

Die Autoren Duquennoy u. a. implementierten eine einfache Variante des MAC-Protokolls IEEE 802.15.4 TSCH, in das Betriebssystem Contiki. In ihrer Arbeit (siehe [26]) berichteten sie über die Herausforderungen der Implementierung von TSCH in Contiki und verglichen die Performance zweier Varianten ihrer TSCH-Implementierung mit ContikiMAC als Stellvertreter eines asynchronen MAC-Protokolls und dem Einsatz von reinem Carrier Sense Multiple Access (CSMA)[15] als MAC-Protokoll (d. h. die Funkeinheit ist ständig aktiv und es wird keine Energie gespart). Die erste Variante von TSCH genügt der Minimalkonfiguration, welche von 6TiSCH spezifiziert wird und verzichtet damit komplett auf einen Planungsalgorithmus. Die Kommunikation findet in dieser Minimalkonfiguration in einem einzigen Shared Link bzw. Timeslot statt. Alle Knoten im WSN konkurrieren innerhalb dieses Shared Links um das Medium. Die zweite Variante verwendet *Orchestra*, ein autonomer Planungsalgorithmus, mit welchem jeder Knoten seinen

[14] LPP ist hier der Name eines, nicht in dieser Arbeit behandelten, asynchronen LPP-MAC-Protokolls für das Betriebssystem Contiki.

[15] Bei CSMA prüft der Sender vor dem Senden, ob der Kanal frei ist, um Kollisionen zu reduzieren. Ist der Kanal nicht frei, wird der Sendevorgang zu einem späteren Zeitpunkt wiederholt.

eigenen lokalen Kommunikationsplan berechnet. Auf Orchestra wird in Kap. 4.6 näher eingegangen.
Die Performance-Analyse in [26] beinhaltet die Stabilität, die Latenz und den Energieverbrauch. Interessant ist, dass TSCH in Verbindung mit Orchestra, wenn man die Verwendung von reinem CSMA herausnimmt, durchweg in allen Versuchen die besten Ergebnisse lieferte. Sogar der oben erwähnte Nachteil, dass synchrone MAC-Protokolle zu höheren Latenzen neigen, als asynchrone MAC-Protokolle, scheint hier nicht zu bestehen. Die detaillierten Ergebnisse der Performance-Analyse sind in Anlage A.3 zusammengefasst.

Synchrone MAC-Protokolle sind dann gut geeignet, wenn es auf niedrige Fehlerraten und berechenbare Latenzen und Datendurchsätze ankommt. Asynchrone MAC-Protokolle sind gut für Echtzeitanwendungen geeignet, wo es auf niedrigen Latenzen ankommt oder auch, wenn eine einfache Implementierung im Vordergrund steht. Zwei vielversprechende asynchrone MAC-Protokolle, ContikiMAC und CSL, wurden in Kap. 4.3 verglichen. Bei einer großen Anzahl an Knoten, kann die Latenz unter asynchronen MAC-Protokollen jedoch auch drastisch steigen. Denn steigt das Datenaufkommen, ist der Kanal häufiger belegt und es kommt vermehrt zu Kollisionen. Im schlimmsten Fall kann ein Knoten dann garnicht mehr senden. MAC-Protokolle wie CSL können diesen Nachteil mit Hilfe von Channel-Hopping etwas lindern. Für WSNs mit tausenden von Knoten ist möglicherweise jedoch ein synchrones MAC-Protokoll die bessere Wahl.

4.6. Orchestra

Orchestra ist ein autonomer Planungsalgorithmus, welcher für Maschen-WSNs auf Basis von IEEE 802.15.4 TSCH, 6LoWPAN und RPL konzipiert ist (vgl. [25]). Prinzipiell ist er aber so generisch gehalten, dass er auch mit anderen Protokollen und Standards in TDMA-basierten WSNs zusammen arbeiten kann. Die meisten Planungsalgorithmen sind an einen bestimmten Anwendungszweck (z. B. an ein bestimmtes Kommunikationsmuster) gebunden und reagieren schlecht auf Änderungen im WSN bezüglich der Topologie, sich ändernder Qualität des Mediums usw. Orchestra richtet sich insbesondere auf die dynamische Natur des IoT aus, in welchem kein bestimmtes Kommunikationsmuster vorherrscht und Änderungen der Topologie regelmäßig stattfinden können. Gleichzeitig soll die Zuverlässigkeit, welche TSCH verspricht, nicht vernachlässigt werden. Orchestra versucht also Flexibilität (wie sie z. B. ein asynchrones MAC-Protokoll bietet) und Zuverlässigkeit, zwei eigentlich gegensätzliche Dinge, zu vereinen (vgl. [25, S. 1]).

Mit Orchestra berechnet jeder Knoten seine eigenen Kommunikationspläne[16] lokal und aktualisiert diese, wenn es Änderungen in der Topologie respektive dem Routing gegeben hat. Typischerweise unterscheidet man zwischen zentralen und dezentralen Planungsalgorithmen. Bei ersteren gibt es eine zentrale Instanz, welche den Kommunikationsplan global verwaltet und im WSN über alle Knoten verteilt. Bei letzteren handelt jeder Knoten entsprechende Kommunikationspläne mit seinen Nachbarn aus. Der Ansatz von Orchestra, dass jeder Knoten seine Kommunikationspläne autonom verwaltet, ist hingegen atypisch. Orchestra kommt ohne zusätzlichen Kommunikations-Overhead aus[17] und bedient sich

[16] Hier ist tatsächlich die Mehrzahl gemeint.

[17] Das bedeutet, dass Orchestra keinen zusätzlichen Datenverkehr im WSN generiert. Beim klassischen zentralen und dezentralen Ansatz ist dies notwendig, um den Kommunikationsplan im WSN zu ver-

zur Berechnung der Kommunikationspläne an vorhandenen Informationen, welche der Netzwerk-Stack bereitstellt. In der in [25] beschriebenen Implementierung, werden vor allem Informationen verwendet, welche RPL bereitstellt. Damit geht Orchestra auch einen etwas anderen Weg, als von der 6TiSCH beschrieben, welche nur Vorschläge für zentrale und dezentrale Ansätze macht.

4.6.1. Kommunikationspläne

Wie oben bereits erwähnt, verwaltet jeder Knoten seine eigenen Kommunikationspläne autonom. Dies geschieht auf Basis seiner Nachbar-Knoten. Standardmäßig verwaltet jeder Knoten drei Kommunikationspläne, welche jeweils für eine bestimmte Art von Datenverkehr zuständig sind: Spezifischer Datenverkehr der MAC-Schicht (TSCH-Beacons), routing- (RPL) und anwendungsspezifischer Datenverkehr (siehe Abb. 4.25).

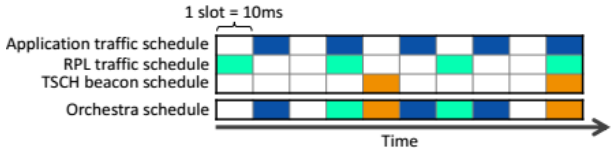


Abbildung 4.25.: Die verschiedenen, sich überlappenden Kommunikationspläne von Orchestra

Quelle: [25, S. 1]

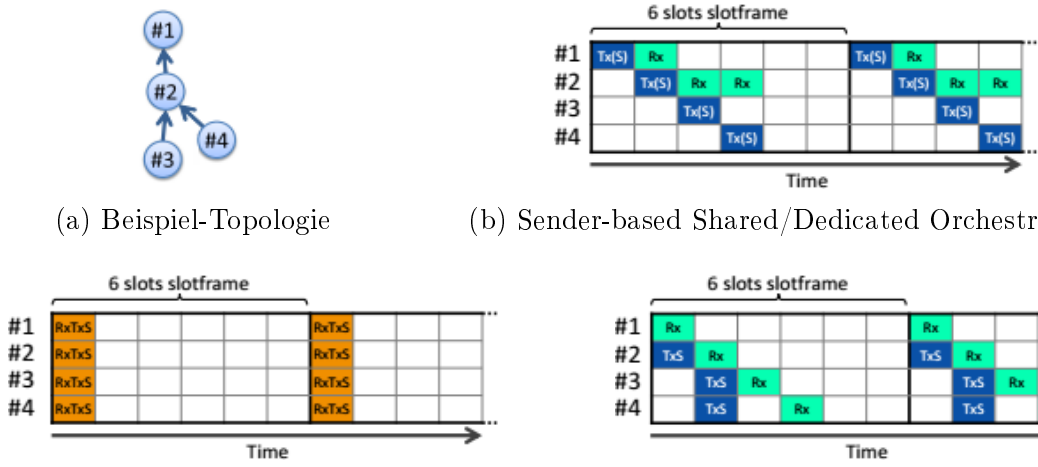
Jeder Kommunikationsplan besteht aus einem Slotframe, welcher eine bestimmte Menge an aktiven Slots enthält, welche sich mit einer bestimmten Periode wiederholen. Die Perioden müssen sich für jeden Kommunikationsplan unterscheiden und sollten einer Primzahl entsprechen. In Abb. 4.25 wiederholen sich die Slots der drei Kommunikationspläne mit den Perioden zwei (anwendungsspezifischer Datenverkehr), drei (RPL-spezifischer Datenverkehr) und fünf (Datenvehrkehr durch TSCH-Beacons). Jeder Kommunikationsplan hat eine andere Priorität. Fallen die Slots mehrerer Kommunikationspläne zusammen, hat der Slot des Kommunikationsplans mit der höchsten Priorität Vorrang. Der anwendungsspezifische Kommunikationsplan hat die niedrigste und der Kommunikationsplan der MAC-Schicht die höchste Priorität. Der daraus resultierende übergeordnete Kommunikationsplan ist in Abb. 4.25 in der letzten Zeile dargestellt.

Welches Datenpaket welcher Art von Kommunikationsplan zugeordnet wird, entscheidet Orchestra anhand eines Regelwerks (sog. *Traffic Filter*). Der Filter kann Datenpakete nach Typ (Broadcast oder Unicast), nach Protokoll (z. B. TSCH und RPL) etc. unterscheiden. Prinzipiell ist das Regelwerk erweiterbar und es kann weitere, als die oben genannten drei, Kommunikationspläne geben.

4.6.2. Slot-Typen

Im Folgenden werden die vier Haupttypen von Slots erläutert, welche Orchestra unterscheidet. Abb. 4.26 (a) zeigt eine Beispiel-WSN-Topologie, anhand derer die unterschiedlichen Slot-Typen in den Abb. 4.26 (b) bis (d) verdeutlicht werden.

teilen bzw. um Kommunikationspläne mit den Nachbar-Knoten auszuhandeln.



(a) Beispiel-Topologie

(b) Sender-based Shared/Dedicated Orchestra Slots

(c) Common Shared Orchestra Slots

(d) Receiver-based Shared Orchestra Slots

Rx - Empfangs-Slot Tx - Sende-Slot TxS - Sende-Slot indem mehrere Sender konkurrieren

Abbildung 4.26.: Die Slot-Typen von Orchestra anhand einer Beispiel-WSN-Topologie
Quelle: [25, S. 5]

Common Shared Orchestra Slots

Ein *Common Shared Orchestra Slot* ist ein TSCH Shared Link, welcher von allen Knoten im gesamten WSN sowohl zum Senden, als auch zum Empfangen benutzt werden kann. D. h. alle Sender konkurrieren innerhalb dieser Slots um das Medium. Knoten, welche nicht senden möchten, nehmen in diesen Slots die Empfängerrolle ein. Die Common Shared Orchestra Slots nehmen eine feste Position (Slot-Nummer im Slotframe und Kanal) im Slotframe ein. Nützlich ist diese Art von Slots etwa für RPL, um die Nachbar-Knoten zu ermitteln, oder für Broadcast-Datenverkehr. Abb. 4.26 (c) zeigt einen beispielhaften Kommunikationsplan, welcher auf diesen Slot-Typ basiert.

Die folgenden Slot-Typen werden ausschließlich zur Kommunikation zwischen benachbarten Knoten im WSN eingesetzt.

Receiver-based Shared Orchestra Slots

Auch *Receiver-based Shared Orchestra Slots* bauen auf TSCH Shared Links auf. Die Position des Slots wird anhand der Adresse des Empfängers (dessen MAC-Adresse oder eindeutigen ID im WSN) berechnet. D. h. für jeden Knoten ergibt sich ein Slot zum Empfangen, dessen Position durch seine Adresse bestimmt wird und einen Slot zum Senden je Nachbar-Knoten, dessen Positionen durch die Adressen der jeweiligen Nachbar-Knoten bestimmt wird. Ein typischer Anwendungsfall für die Kind-Knoten-zu-Eltern-Knoten-Kommunikation, bei welcher jeder Eltern-Knoten einen Slot zum Empfangen reserviert und seine Kind-Knoten einen Slot mit derselben Position zum Senden. Wechselt ein Knoten seinen Eltern-Knoten, aktualisiert er automatisch die Position des Slots zum Senden. Da es pro Slot nur einen Empfänger gibt, es aber mehrere Sender geben kann, ist diese Art von Slot contention-based, d. h. also, die Sender müssen um das Medium konkurrieren. Abb. 4.26 (d) zeigt einen beispielhaften Kommunikationsplan, welcher Receiver-based Shared Orchestra Slots nutzt. Wie zu erkennen ist, konkurrieren die Knoten #3 und #4 um den Empfänger #2.

Sender-based Shared Orchestra Slots

Sender-based Shared Orchestra Slots unterscheiden sich von Receiver-based Shared Orchestra Slots dadurch, dass die Position des Slots von der Adresse des Senders, statt des Empfängers abhängt. Somit verwaltet jeder Knoten einen Slot zum Senden, dessen Position von seiner eigenen Adresse abhängt und mehrere (einen per Nachbar-Knoten) Slots zum Empfangen, dessen Positionen von den Adressen der jeweiligen Nachbar-Knoten abhängen. Sender-based Shared Orchestra Slots tragen zu einem reduzierten Konkurrenzverhalten bei[18], da bei der Slot-Verteilung nicht mehr die Empfänger im Vordergrund stehen, wie es bei den Receiver-based Shared Orchestra Slots der Fall ist. In Szenarien mit eher gemäßigten Datenverkehr, ist die Energiebilanz jedoch schlechter, als bei der Verwendung von Receiver-based Shared Orchestra Slots. Dies liegt darin begründet, dass auf einen Sende-Slot nun mehrere Empfangs-Slots kommen und jeder Empfangs-Slot das Verlassen des Energiesparmodus voraussetzt. (Slots zum Senden können einfach übersprungen werden, wenn keine Daten zum Senden anfallen.) Bei der Kind-Knoten-zu-Eltern-Knoten-Kommunikation verwalten die Kind-Knoten einen Sende-Slot, während die Eltern-Knoten einen Empfangs-Slot je Kind-Knoten verwalten. Wechselt ein Kind-Knoten seinen Eltern-Knoten, muss dieser seine Slots nicht aktualisieren, wohl aber der neue (fügt einen Empfangs-Slot hinzu) und der alte (entfernt einen Empfangs-Slot) Eltern-Knoten.

Sender-based Dedicated Orchestra Slots

Sender-based Dedicated Orchestra Slots unterscheiden sich von Sender-based Shared Orchestra Slots dadurch, dass sie auf TSCH Dedicated Links basieren. D. h., dass Sender-based Dedicated Orchestra Slots Konkurrenz vollkommen vermeiden, also contention-free sind. Dies setzt voraus, dass ein Slotframe mindestens so viele Slots umfasst, wie es Knoten im WSN gibt und dass jeder Knoten über eine eindeutige Adresse verfügt. Eine eindeutige Adresse kann entweder in jeden Knoten hart einprogrammiert sein, oder durch den PAN-Koordinator beim Beitreten eines Knotens zum WSN zugeteilt werden. Die MAC-Adresse sollte an dieser Stelle nicht verwendet werden. Üblicherweise bildet man einen Hash-Wert über die MAC-Adresse und rechnet diesen modulo der Anzahl der Slots eines Slotframes. Obwohl jeder Knoten normalerweise eine einzigartige MAC-Adresse besitzt, kann dieses Verfahren, selbst bei unterschiedlichen MAC-Adressen, zu doppelten Slot-Belegungen führen.

4.6.3. Slot-Typen und Kommunikationspläne

Im Folgenden wird der Zusammenhang zwischen den Slot-Typen und den Kommunikationsplänen anhand zweier Beispiele hergestellt (vgl. [25, S. 6 ff.]).

Die einfachste Variante

In der einfachsten Variante (von den Autoren von Orchestra auch als *6TiSCH Minimal Schedule* bezeichnet) gibt es nur einen einzigen Kommunikationsplan, dessen Slots vom Typ Common Shared Orchestra Slot sind und welcher für sämtlichen Datenverkehr zuständig ist. Ein entsprechender Kommunikationsplan könnte dann aus einem Slotframe

[18] Auch wenn das Beispiel in Abb. 4.26 (b) keine Konkurrenzsituation zeigt, ist es dennoch möglich, dass mehrere Sender-Slots dieselbe Position einnehmen. Umfasst der Slotframe weniger Slots, als es Knoten im WSN gibt, muss dieser Fall zwangsläufig eintreten.

mit einer Größe von zehn Slots, von denen jeweils der erste aktiv ist, bestehen (siehe auch Abb. 4.26 (c)). Die Größe des Slotframes und die Anzahl der darin vorhandenen aktiven Slots können beliebig variieren. Da hier in allen aktiven Slots Konkurrenz herrscht und die Kollisionswahrscheinlichkeit damit entsprechend hoch ist, liefert diese Variante schnell eine äußerst schlechte Performance (Hinsichtlich Stabilität, Latenz und Energieverbrauch) und ist daher nicht zu empfehlen

Verwendung von Receiver-based Unicast Slots

Die Standardkonfiguration von Orchestra sieht momentan den Einsatz von drei unterschiedlichen Kommunikationsplänen, wie in Kap. 4.6.1 bereits beschrieben, vor. Der Slotframe des ersten Kommunikationsplans, welcher für den Datenverkehr der MAC-Schicht verantwortlich ist und die höchste Priorität hat, umfasst mehr Slots, als es Knoten im WSN gibt. Dies ermöglicht den Einsatz von Sender-based Dedicated Orchestra Slots. Daraus resultiert, dass jeder Knoten genau zwei Slots, zum Senden und zum Empfangen, in seinem Kommunikationsplan reserviert. Über den Empfangs-Slot empfängt der Knoten die TSCH-Beacons seines Eltern-Knotens und sendet seinerseits TSCH-Beacons an seine Kind-Knoten. Über die TSCH-Beacons wird auch die Synchronisation der Zeitgeber vorgenommen, indem sich jeder Knoten auf den Zeitgeber seines Eltern-Knotens synchronisiert. Dies wird fortgeführt bis zum obersten Knoten in der Hierarchie, welcher keine Eltern-Knoten mehr hat. Typischerweise ist dies der PAN-Koordinator, welcher dann den Referenzzeitgeber für das gesamte WSN darstellt.
Der zweite Kommunikationsplan, für das Routing-Protokoll RPL, nutzt Common Shared Orchestra Slots. Die Position des aktiven Slots im Slotframe ist im gesamten WSN, d. h. für alle Knoten, gleich, also nicht von der Adresse eines Knotens abhängig. Dies ist wichtig, damit ein Knoten den RPL-Broadcast-Datenverkehr der anderen Knoten empfangen kann (auch wenn dessen Eltern und Kinder noch nicht feststehen) und somit seine Nachbar-Knoten erkunden kann. Ohne RPL wäre die Entstehung von Eltern-Kind-Beziehungen zwischen den Knoten im WSN gar nicht möglich. Andere Slot-Typen hängen von den Informationen über die Eltern-Kind-Beziehungen des Knotens ab. Daher sind Common Shared Orchestra Slots die einzigen Slot-Typen, welche für RPL in Frage kommen.
Der dritte Kommunikationsplan, für den anwendungsspezifischen bzw. den restlichen Datenverkehr (welcher sich keinem der anderen Kommunikationspläne zuordnen lässt) und der niedrigsten Priorität, verwendet Receiver-based Shared Orchestra Slots. (Genauso gut könnten aber auch Sender-based Dedicated Orchestra Slots verwendet werden.)

5. Betrachtungen zur Implementation eines MAC-Protokolls in RIOT

Das Open Source Betriebssystem RIOT[1] ist ein noch recht junges Projekt (2013) mit Potenzial.[2] RIOT zeichnet sich durch einen minimalen Ressourcenverbrauch (CPU, RAM etc.) aus. Es läuft auf zahlreichen Hardware-Plattformen, denen ein 8-, 16- oder 32-Bit-Mikrocontroller zugrunde liegt. Die Programmierung erfolgt wahlweise in C oder C++. Für RIOT steht eine Open-Source-Mentalität à la Linux an oberster Stelle.

Lange Zeit fehlte in der Architektur von RIOT die Möglichkeit, ein energiesparendes MAC-Protokoll einzubinden. Seit einer Weile ist diese Möglichkeit jedoch gegeben und seit Ende Juni 2017 hat die Implementation eines *Lightweight MAC (LwMAC)* genannten asynchronen LPL-MAC-Protokolls Einzug in den offiziellen Quellcode von RIOT gefunden. LwMAC ist, wie auch Contiki X-MAC, eine erweiterte Version von X-MAC (vgl. [19]). Wie Contiki X-MAC besitzt LwMAC z. B. auch die Fähigkeit, sich auf die Empfänger zu synchronisieren, um die Unicast-Datenübertragung zu optimieren, sodass die Präambel an schon bekannte Empfänger deutlich kürzer ausfallen kann.

Ein MAC-Protokoll wie ContikiMAC oder IEEE 802.15.4 CSL arbeitet jedoch effizienter, da der Kanal in jedem Zyklus lediglich durch kurze CCAs abgetastet wird (siehe Kap. 7.1.2). In diesem Kapitel prüft der Autor die Möglichkeit, ContikiMAC in RIOT zu implementieren. Da viele Funkeinheiten das lückenlose Senden von Paketen nicht zu unterstützen scheinen (siehe Kap. 4.3.5), dies aber für das Senden der Wake-Up-Pakete von CSL notwendig ist, wird auf die Untersuchung der Möglichkeit CSL zu implementieren verzichtet. Außerdem ist für ContikiMAC, im Gegensatz zu CSL, eine Referenzimplementierung vorhanden, was sich als hilfreich erweist. Zu Testzwecken kam die Hardware-Plattform *SAM R21 Xplained Pro Evaluation Kit*[3] von Atmel zum Einsatz (siehe Abb. 5.1). Das Herzstück dieser Hardware-Plattform bildet der SoC *ATSAMR21E18A* von Atmel. Der SoC umfasst einen ARM Cortex-M0+ Mikrocontroller und eine AT86RF233 IEEE 802.15.4-Funkeinheit. Der Mikrocontroller verfügt über 32 KB Arbeitsspeicher, 256 KB Flash-Speicher für Programme und Daten und ist mit 16 MHz getaktet (48 MHz sind maximal möglich).

[1] Offizielle Webseite: `https://riot-os.org/` Quellcode: `https://github.com/RIOT-OS/RIOT`
[2] Das Betriebssystem Contiki, zum Vergleich, startete bereits 2003.
[3] Die interne Bezeichnung in RIOT für diese Hardware-Plattform lautet *samr21-xpro*.

Abbildung 5.1.: SAM R21 Xplained Pro Evaluation Kit von Atmel
Quelle: [22]

5.1. RIOT Netzwerk-Stack

Abb. 5.2 zeigt den Netzwerk-Stack von RIOT mit seinen fünf Ebenen. Jede Ebene wird in einem eigenen Thread ausgeführt. Jeder Thread einer Ebene hat eine niedrigere Priorität, als der Thread der darunterliegenden Ebene und eine höhere Priorität, als der Thread der darüberliegenden Ebene. D. h. die Ebene gnrc_conn hat die niedrigste Priorität (eine Prioritätsstufe höher, als die Basis-Priorität von RIOT-Threads) und die Ebene gnrc_netdev wird mit der höchsten Priorität ausgeführt. Die Threads kommunizieren über die Interprozesskommunikations-Schnittstelle (eng. Interprocess Communication (IPC)) des RIOT-Kernels oder über das *GNRC Communication Interface* (netapi). Letztere dient speziell der Kommunikation der Threads innerhalb des Netzwerk-Stacks und nutzt im Hintergrund ebenfalls IPC. Die meiste Zeit kommunizieren Threads benachbarter Ebenen miteinander, wenn Pakete den Netzwerk-Stack von unten nach oben (Paket empfangen) bzw. von oben nach unten (Paket senden) durchwandern.

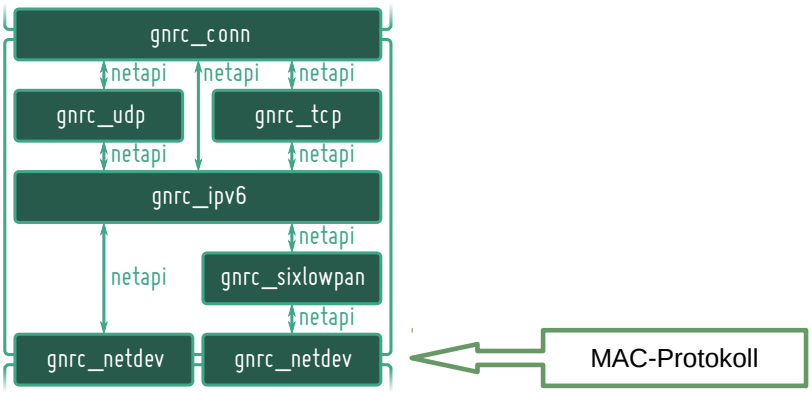


Abbildung 5.2.: RIOT Netzwerk-Stack
Quelle: In Anlehnung an [18]

Die Ebene gnrc_netdev bildet die Schnittstelle zwischen dem Netzwerk-Stack und dem Treiber der Funkeinheit und ist zugleich die Ebene, in welcher MAC-Protokolle implementiert werden. Standardmäßig wird die Implementierung hierfür durch das gleichnamige RIOT-Modul gnrc_netdev bereitgestellt.[4] Das zugehörige Modul zu LwMAC heißt

[4] Code: `sys/net/gnrc/link_layer/netdev/gnrc_netdev.c`

gnrc_lwmac. Wird es aktiviert, ersetzt es das Modul gnrc_netdev vollständig. Das Modul gnrc_netdev beinhaltet reinen Verbindungs-Code zwischen dem Treiber der Funkeinheit AT86RF233 und dem Netzwerk-Stack und beinhaltet keinerlei MAC-Funktionalität. Dies liegt darin begründet, dass der Treiber der Funkeinheit den sog. *Extended Mode* nutzt. D. h. die Funkeinheit beinhaltet bereits die nötigsten MAC-Funktionalitäten (ohne Energiesparmechanismen) und entlastet so den Mikrocontroller. Bei Empfangen sendet die Funkeinheit automatisch ein ACK zurück, wenn gefordert. Beim Senden wird automatisch das CSMA-CA-Verfahren[5] angewendet, um Kollisionen zu vermeiden und es wird nach dem Senden automatisch auf das ACK des Empfängers gewartet. Schlägt das Senden fehl, weil der Kanal gerade belegt ist oder das ACK des Empfängers nicht empfangen wurde, wiederholt die Funkeinheit den Sendeversuch automatisch bis zu einer festgelegten Maximalzahl. Der Mikrocontroller wird am Ende lediglich via Interrupt über das Ergebnis des Sendevorgangs benachrichtigt. Klassische Funkeinheiten sind für die PHY-Schicht verantwortlich (siehe Tabelle 3.1 auf S. 13). Funkeinheiten wie die AT86RF233 können zusätzlich also auch die Verantwortung für die MAC-Schicht übernehmen.

Am anderen Ende des Netzwerk-Stacks befindet sich die Ebene gnrc_conn. Sie stellt das Bindeglied zwischen der Anwendungsschicht und dem Netzwerk-Stack dar. Anwendungen greifen mittels der *Sock API*[6] von RIOT auf die Funktionalitäten des Netzwerk-Stacks zu.

Zusammengefasst bedeutet dies: Empfängt die Funkeinheit ein Paket, benachrichtigt sie die Ebene gnrc_netdev via Interrupt. Die Ebene gnrc_netdev leitet dieses Paket an die Ebene gnrc_sixlowpan weiter, welche das gemäß 6LoWPAN komprimierte Paket wieder zu einem vollwertigen IPv6-Paket extrahiert und leitet es anschließend an die Ebene gnrc_ipv6 weiter. Die Ebene gnrc_ipv6 kümmert sich um den IP-spezifischen Inhalt des Paketes und leitet dieses entweder an die Ebene gnrc_udp weiter, wenn der Payload einen User Datagram Protocol (UDP)-Header enthält, oder an die Ebene gnrc_tcp, wenn der Payload einen Transmission Control Protocol (TCP)-Header enthält. Die Ebenen gnrc_udp und gnrc_tcp werten das UDP- respektive TCP-Paket aus und leiten es weiter an die Ebene gnrc_conn. Die Ebene gnrc_conn prüft, welcher Anwendung das eingehende Paket zuzuordnen ist und leitet dessen Payload an die entsprechende Anwendung weiter. Beim Senden wird der umgekehrte Weg gegangen. Zunächst übermittelt die Anwendung den Payload des Paketes über die Sock API an die Ebene gnrc_conn. Diese prüft, ob der Payload als UDP- oder TCP-Paket versendet werden soll und leitet den Payload entsprechend an die Ebene gnrc_udp oder gnrc_tcp weiter. Diese wiederum leiten das Paket weiter an die Ebene gnrc_ipv6 usw.

5.2. Erstellen eines RIOT-Moduls für ContikiMAC

Wie oben schon beschrieben, ist LwMAC als Modul gnrc_lwmac realisiert. Das Präfix *gnrc* stellt die Zugehörigkeit des Moduls zum Netzwerk-Stack (in RIOT auch *Generic (GNRC) Network Stack* genannt) klar. Ein entsprechendes Modul für ContikiMAC könnte gnrc_contikimac heißen. Dieses Kapitel beschreibt die minimalen Schritte, welche notwendig sind, um ein solches Modul im RIOT-Quellcode-Baum zu erzeugen.

[5] D. h. es wird zunächst mit einem CCA geprüft, ob der Kanal frei ist. Nur wenn dieser frei ist, wird gesendet. Ist der Kanal belegt, wird eine zufällige Zeit gewartet, bevor der Vorgang wiederholt wird. Der Vorgang wird solange wiederholt, bis eine Obergrenze erreicht wurde oder der Kanal frei ist.

[6] Application Programming Interface

- Das Verzeichnis `sys/net/gnrc/link_layer/contikimac/` ist anzulegen. Es enthält die Quellcode-Dateien für die Implementation des Moduls sowie eine Make-Datei (`Makefile`). Letztere enthält nur zwei Zeilen und den Namen des Moduls (siehe Listing 5.1).

```
MODULE = gnrc_contikimac
include $(RIOTBASE)/Makefile.base
```

Listing 5.1: Makefile für das Modul gnrc_contikimac.

- Die Dateien `contikimac.c` im Verzeichnis `sys/net/gnrc/link_layer/contikimac/` ist anzulegen. Sie enthält die Implementation von ContikiMAC und inkludiert die Datei `sys/include/net/gnrc/contikimac/contikimac.h` (siehe weiter unten). Die Datei `contikimac.c` muss mindestens die Funktion `gnrc_contikimac_init()` implementieren, welche den Thread von ContikiMAC initialisiert.

- Die Header-Datei `sys/include/net/gnrc/contikimac/contikimac.h` ist anzulegen. Sie definiert das Interface des Moduls und enthält mindestens die Deklaration der Funktion `gnrc_contikimac_init()`. Des Weiteren ist dies eine passende Stelle, um Konstanten zu definieren, welche die Betriebsparameter für ContikiMAC (z. B. wie oft der Kanal abgetastet wird) festlegen.

- Das Modul muss in die Make-Datei `sys/net/gnrc/Makefile` durch das Hinzufügen der Zeilen aus Listing 5.2 aufgenommen werden.

```
ifneq (,$(filter gnrc_contikimac,$(USEMODULE)))
    DIRS += link_layer/contikimac
endif
```

Listing 5.2: Zusätzliche Zeilen in sys/net/gnrc/Makefile.

- Die Abhängigkeiten des Moduls von anderen Modulen und Fähigkeiten der Hardware-Plattform muss in der Datei `Makefile.dep` durch das Hinzufügen der Zeilen aus Listing 5.3 definiert werden.

```
ifneq (,$(filter gnrc_contikimac,$(USEMODULE)))
  USEMODULE += gnrc_mac
  USEMODULE += gnrc_netdev
  FEATURES_REQUIRED += periph_rtt
endif
```

Listing 5.3: Die Abhängigkeiten des Moduls müssen in der Datei Makefile.dep hinterlegt werden.

Wichtig ist, dass diese Zeilen nach den Angaben der Abhängigkeiten der Module, von denen es abhängt, eingefügt werden. D. h. also in diesem Fall nach den Abhängigkeitsangaben der Module gnrc_mac und gnrc_netdev. Die Zeile `FEATURES_REQUIRED += periph_rtt` gibt an, dass die Hardware-Plattform den *Real Time Timer (RTT)* unterstützen muss. Dabei handelt es sich um einen interrupt-gesteuerten Timer, welcher bei Vorgängen eingesetzt wird, bei denen es auf ein exaktes Timing ankommt.

- Modifikation der Datei `sys/auto_init/netif/auto_init_at86rf2xx.c`. Diese Datei ist spezifisch für die Initialisierung von Funkeinheiten der Familie AT86RF2xx und damit auch für die Funkeinheit AT86RF233. Der Code in dieser Datei ist dafür zuständig, den Treiber der Funkeinheit mit der Ebene gnrc_netdev des Netzwerk-Stacks zu verheiraten. Listing 5.4 zeigt die Anpassungen, welche für das Modul gnrc_contikimac vorgenommen wurden. (Die Zeilennummern entsprechen nicht den tatsächlichen Zeilennummern der Datei.)

```
1 #include "net/gnrc/contikimac/contikimac.h"
2 ...
3 #ifdef MODULE_GNRC_LWMAC
4             gnrc_lwmac_init(_at86rf2xx_stacks[i],
5                             AT86RF2XX_MAC_STACKSIZE,
6                             AT86RF2XX_MAC_PRIO,
7                             "at86rf2xx-lwmac",
8                             &gnrc_adpt[i]);
9 #elif defined MODULE_GNRC_CONTIKIMAC
10            gnrc_contikimac_init(_at86rf2xx_stacks[i],
11                            AT86RF2XX_MAC_STACKSIZE,
12                            AT86RF2XX_MAC_PRIO,
13                            "at86rf2xx-contikimac",
14                            &gnrc_adpt[i]);
15 #else
16            gnrc_netdev_init(_at86rf2xx_stacks[i],
17                             AT86RF2XX_MAC_STACKSIZE,
18                             AT86RF2XX_MAC_PRIO,
19                             "at86rf2xx",
20                             &gnrc_adpt[i]);
21 #endif
```

Listing 5.4: Die Ergänzungen in der Datei auto_init_at86rf2xx.c.

Die Zeilen 1 und 9 bis 15 wurden ergänzt. Für jedes Modul wird eine Konstante `MODUL_MODULNAME` definert, wenn es eingebunden wurde. Diese lautet für ContikiMAC entsprechend `MODULE_GNRC_CONTIKIMAC`. Anhand dieser Konstanten kann der Compiler mit Hilfe von Präprozessor-Anweisungen, wie z. B. in den Zeilen 3, 9, 15 und 21 zu sehen sind, entscheiden, welcher Code übersetzt wird. Modul-spezifischer Code wird somit nur übersetzt, wenn das Modul eingebunden ist. In Zeile 3 wird zunächst geprüft, ob das Modul für LwMAC eingebunden wurde und es wird gegebenenfalls die Initialisierungsroutine von LwMAC aufgerufen (Zeile 4 bis 8). Ist dies nicht der Fall, wird geprüft, ob das Modul für ContikiMAC eingebunden wurde, um dessen Initialisierungsroutine aufzurufen. Wurde keines dieser Module eingebunden, wird die Initialisierungsroutine des Standard-Moduls gnrc_netdef aufgerufen.

- Schlussendlich muss das Modul eingebunden werden, indem die Anweisung aus Listing 5.5 der Make-Datei des Projektes hinzugefügt wird. Für das Beispiel-Projekt gnrc_border_router muss diese Anweisung bspw. in die Datei `examples/gnrc_border_router/Makefile` eingefügt werden.

```
USEMODULE += gnrc_contikimac
```

Listing 5.5: Anweisung in der Make-Datei das Projektes, welche das Modul aktiviert.

Der Quellcode von RIOT, mit allen in diesem Kapitel erläuterten Dateien, ist auf der beiliegenden CD zu finden. Hinweise auf eine Internetquelle gibt Anlage A.5. Die für dieses Kapitel gemachten Änderungen, bauen auf dem Master-Entwicklungszweig[7] von RIOT, mit Stand vom 18. September 2017, auf.

5.3. Festgestellte Probleme

5.3.1. CCA-Phase

Wie in Kap. 4.2.5 beschrieben, beginnt unter ContikiMAC jeder Zyklus mit einer CCA-Phase, welche aus zwei aufeinanderfolgenden CCAs besteht. Wie in Kap. 5.1 erwähnt, verwendet der Treiber der Funkeinheit AT86RF233 den Extended Mode. Eine manuelle Ausführung eines CCAs ist nicht vorgesehen und steht im Konflikt mit dem Extended Mode der Funkeinheit (vgl. [16] und [17]). Die Konsequenz war die Deimplementierung der API-Funktion für einen CCA aus dem Treiber. Erst einige Monate später (am 4. September 2017) wurde diese API-Funktion wieder in den Treiber aufgenommen. Die neue Funktion umgeht das Problem, indem der Extended Mode vorübergehend deaktiviert wird, wenn ein manueller CCA ausgeführt wird. Die CCA-Phase ist sehr zeitkritisch, d. h. die CCA-Phasen müssen sich in möglichst exakten Zeitabständen wiederholen (alle 125 ms (8 Hz) in der Standardeinstellung von ContikiMAC). Daher muss der RTT von RIOT für das Auslösen der CCA-Phase eingesetzt werden. Wie oben bereits beschrieben, in der RTT interrupt-gesteuert. D. h. es gibt einen Hardware-Timer[8], welcher zu definierten Zeitpunkten einen Interrupt auslöst. Der aktuell ausführende Thread wird dann sofort unterbrochen und es wird die Routine für die CCA-Phase aufgerufen. Listing 5.6 zeigt den Quellcode des Prototyps der Routine für die CCA-Phase.

```
1  bool contikimac_perform_cca_phase(netdev_t *dev)
2  {
3      netopt_enable_t channel_free;
4      xtimer_ticks32_t last_wakeup;
5
6      for (int ccas = 0; ccas < GNRC_CONTIKIMAC_CCA_COUNT_MAX; ccas++) {
7          if (ccas > 0) {
8              last_wakeup = xtimer_now();
9              xtimer_periodic_wakeup(&last_wakeup,
                   GNRC_CONTIKIMAC_CCA_SLEEP_TIME_US);
10         }
11
12         dev->driver->get(dev, NETOPT_IS_CHANNEL_CLR, &channel_free, sizeof(
               channel_free));
13
14         if (!channel_free) return true;
15     }
16
17     return false;
18 }
```

Listing 5.6: Quellcode der Routine für die CCA-Phase

[7] https://github.com/RIOT-OS/RIOT

[8] Hiermit ist ein Timer gemeint, welcher nicht mit Software, sondern mit realen Schaltkreisen realisiert ist. Die meisten Hardware-Plattformen verwenden für diesen Timer einen dedizierten Schwingquarz.

Innerhalb der For-Schleife (ab Zeile 6) werden die einzelnen CCAs der CCA-Phase durchgeführt. Typischerweise besteht eine CCA-Phase aus zwei CCAs. Dies lässt sich über die Konstante `GNRC_CONTIKIMAC_CCA_COUNT_MAX` festlegen. In Zeile 12 wird die eigentliche API-Funktion des Treibers für einen CCA aufgerufen. Das Ergebnis wird in die Variable `channel_free` geschrieben. Hat der Aufruf der API-Funktion ergeben, dass der Kanal belegt ist, wurde möglicherweise ein Paket detektiert und die Routine wird mit dem Rückgabewert `true` verlassen. Zwischen jedem CCA wird eine Pause (siehe t_c in Tabelle 4.2 auf S. 4.2) eingelegt (Zeile 8 u. 9). Die If-Anweisung in Zeile 7 sorgt dafür, dass die Pausen nur zwischen zwei CCAs und nicht schon vor dem ersten CCA eingelegt wird. Hat keiner der CCAs einen belegten Kanal, d. h. ein potentielles Paket detektiert, wird die Routine mit dem Rückgabewert `false` verlassen.

An dieser Stelle wurde ein Problem mit dem Timing festgestellt: Die Pausenzeit t_c dauert viel zu lange (1,3 ms). Sie wird über die Konstante `GNRC_CONTIKIMAC_CCA_SLEEP_TIME_US` festgelegt, welche für den Test mit dem Standardwert (0,5 ms) definiert wurde. Selbst nach dem Entfernen der Verzögerung in den Zeilen 7 bis 10, dauert die Pausenzeit zu lange (sie beträgt dann noch ca. 741 μs). Die eigentliche Dauer eines CCAs beträgt für diese Funkeinheit laut Datenblatt 128 μs (im 2,4 GHz-Band). Der Aufruf der API-Funktion in Zeile 12 dauert jedoch rund 917 μs (das Siebenfache des eigentlichen CCAs). Diese Werte wurden vom Autor mit Hilfe eines Oszilloskops gemessen (siehe Anlage A.4). Möglicherweise hat dies mit dem Verlassen und Wiedereintreten in den Extended Mode vor und nach dem manuellen CCA zu tun.

ContikiMAC legt für die minimale Sendedauer eines Datenpaketes die Restriktion $t_s > t_r + t_c + t_r = 2t_r + t_c$ fest (siehe S. 58 ff.). Mit einer CCA-Dauer von 128 μs und der Standarddauer einer Pause von 0,5 ms ergibt sich damit eine minimale Sendedauer t_s von $2t_r + t_c = 2 * 128\ \mu s + 500\ \mu s = 756\ \mu s$. Um oben beschriebenes Problem auszugleichen, kann die minimale Sendedauer eines Datenpaketes erhöht werden. Für die Dauer eines CCAs werden 174 μs angenommen.[9] Die Pausenzeit lässt sich, wie oben beschrieben, auf minimal 741 μs verkürzen. Die neue minimale Sendedauer eines Paketes ergibt sich dann zu $2t_r + t_c = 2 * 174\ \mu s + 741\ \mu s = 1089\ \mu s$. Das entspricht ungefähr einem Viertel der Sendedauer des größtmöglichen IEEE 802.15.4-Datenpaketes.

Der Quellcode der prototypischen Implementation der CCA-Phase, mit allen notwendigen Dateien, ist auf der beiliegenden CD zu finden. Hinweise auf eine Internetquelle gibt Anlage A.5.

[9] Die tatsächliche Dauer eines CCAs und wann dieser genau ausgeführt wird, kann mit dem Oszilloskop nicht näher gemessen werden (siehe Anlage A.4).

5.3.2. Das Senden von Datenpaketen (die Pausenzeit t_i einhalten)

Der Algorithmus zum Senden eines Datenpaketes (ohne Berücksichtigung der Phasen-Optimierung) gestaltet sich wie folgt:

1. Schreibe die aktuelle System-Zeit in eine Variable (`start_time`).
2. Sende eine Kopie des Datenpaketes.
3. Prüfe, ob die verstrichene Zeit (aktuelle System-Zeit - `start_time`) größer ist, als die Dauer eines Zyklus und falls ja, gehe zu 6.
4. Warte bis die Zeit t_i, welche zwischen zwei aufeinanderfolgenden Datenpaketen liegen muss, abgelaufen ist.
   a) Bei Unicast: Wenn vor dem Verstreichen der Zeit t_i das ACK des Empfängers empfangen wurde, gehe zu 6.
5. Gehe zu 2.
6. Sendevorgang beendet.

In diesem Abschnitt geht es um Punkt 4, dem Verzögern des Sendens des nächsten Datenpaketes um die Zeit t_i. Dafür gibt es zwei Möglichkeiten: Die Datenpakete werden in einer Schleife gesendet, solange, bis die maximale Sendedauer (Zeit eines Zyklus) erreicht ist oder das ACK des Empfängers (bei Unicast) empfangen wurde. Die Pausenzeit t_i zwischen den einzelnen Datenpaketen wird mit Verzögerungs-Funktionen der Timer-Bibliothek Xtimer von RIOT erreicht. Die zweite Möglichkeit besteht darin, dass eine Funktion ein Datenpaket sendet und anschließend dafür sorgt, dass sie — unter der Voraussetzung, dass die maximale Sendedauer noch nicht erreicht wurde und das ACK des Empfänger noch nicht empfangen wurde — nach der Zeit t_i wieder aufgerufen wird.

LwMAC bspw. nutzt die zweite Möglichkeit. Wie Contiki X-MAC sendet LwMAC zunächst Präambel-Pakete, welche hier Wake-Up-Request-Pakete (WR-Pakete) genannt werden, um den Empfänger zu veranlassen, wach zu bleiben, um ein oder mehrere Datenpakete empfangen zu können (vgl. [21] u. [19]). Das Senden eines Unicast-Datenpaketes läuft bei LwMAC (etwas vereinfacht) wie folgt ab: Der Thread von LwMAC erhält eine Nachricht via IPC vom Thread der überliegenden Schicht, dass ein Datenpaket gesendet werden soll. Daraufhin beginnt das Sender einer Sequenz von WR-Paketen. Es wird das erste WR-Paket gesendet und LwMAC sendet dem eigenen Thread, mit Hilfe der Funktion `xtimer_set_msg()`[10], eine verzögerte Nachricht, welche den LwMAC-Thread veranlasst, das nächste WR-Paket zu senden usw. Zwischen den WR-Paketen wechselt der Sender in den Empfangsmodus. Wird zwischenzeitlich ein Paket empfangen, ruft die Funkeinheit eine Interrupt-Routine auf, welche dem LwMAC-Thread eine Nachricht schickt, dass ein Paket empfangen wurde. Handelt es sich dabei um das Antwort-Paket des Empfängers auf ein WR-Paket, wird die aktuelle Nachricht an den LwMAC-Thread für das Senden des nächste WR-Paketes, welche noch darauf wartet gesendet zu werden, gelöscht und der LwMAC-Thread sendet sich erneut eine Nachricht, welche dieses Mal das Senden des eigentlichen Datenpaketes veranlasst.

[10] `xtimer_set_msg()` ist eine Funktion der Xtimer-Bibliothek, mit welcher einem Thread zeitversetzt eine Nachricht via IPC gesendet werden kann.

Dieser Ansatz wurde auf ähnliche Weise vom Autor für eine testweise Implementation einer Senden-Funktion für ContikiMAC übernommen. Dabei wurde festgestellt, dass die Pausenzeit t_i nicht eingehalten werden konnte. Statt 0,4 ms betrug diese stets etwas mehr als 5 ms. Selbst als die Funktion `xtimer_set_msg()` durch die Funktion `msg_send_to_self()` (eine nicht-blockierende Funktion, welche der Warteschlange des Threads eine Nachricht hinzufügt) ersetzt wurde, d. h. dem eigenen Thread wurde nun ohne Verzögerung eine Nachricht gesendet, konnte eine Pausenzeit von 5 ms nicht unterschritten werden. Vermutlich ist das IPC-System von RIOT für solch kurze Zeitspannen zu träge. Bei LwMAC liegt zwischen jedem WR-Paket standardmäßig eine Zeit von 5 ms (vgl. [21]). Dies ist genau die Zeit, welche nicht unterschritten werden konnte. Daher liegt die Vermutung nahe, dass dies der Grund ist, weshalb man bei LwMAC genau diese Zeit gewählt hat.

Das Timing-Problem veranlasste den Autor dazu, eine weitere testweise Implementation einer Senden-Funktion zu entwickeln, welche die erste Möglichkeit prüft (siehe Listing 5.7).

```
1 bool contikimac_send(gnrc_netdev_t *netdev, gnrc_pktsnip_t *pkt)
2 {
3     xtimer_ticks32_t transmission_start, now;
4
5   gnrc_contikimac_state_t old_state = netdev->contikimac.state;
6     netdev->contikimac.state = GNRC_CONTIKIMAC_TRANSMITTING;
7
8     transmission_start = xtimer_now();
9
10    /* Disable things that make packet sending slow. */
11    netopt_enable_t netop = NETOPT_DISABLE;
12    netdev->dev->driver->set(netdev->dev, NETOPT_AUTOACK, &netop, sizeof(netop));
13    netdev->dev->driver->set(netdev->dev, NETOPT_CSMA, &netop, sizeof(netop));
14    netdev->dev->driver->set(netdev->dev, NETOPT_TX_START_IRQ, &netop, sizeof(
          netop));
15
16
17    do {
18        /* Don't let the packet be released yet, we want to send it again. */
19        gnrc_pktbuf_hold(pkt, 1);
20
21        netdev->send(netdev, pkt);
22
23        //xtimer_periodic_wakeup(&now,
              GNRC_CONTIKIMAC_INTER_PACKET_INTERVAL_US);
24        xtimer_usleep(GNRC_CONTIKIMAC_INTER_PACKET_INTERVAL_US);
25
26        now = xtimer_now();
27    } while((now.ticks32 - transmission_start.ticks32) <
          GNRC_CONTIKIMAC_STROBE_TIME);
28
29    gnrc_pktbuf_release(pkt);
30    netdev->contikimac.state = old_state;
31
32    return true;
33 }
```

Listing 5.7: Quellcode der testweisen Senden-Funktion

Die Funktion in Listing 5.7 sendet wiederholt Kopien des Datenpaketes für die Dauer eines Zyklus. Der nötige Code zum vorzeitigen Abbrechen des Sendevorgangs, wenn das ACK des Empfängers empfangen wurde, ist nicht enthalten. Die Datenpakete werden innerhalb der Schleife (Zeile 17 bis 27) gesendet. Das eigentliche Senden eines einzelnen Datenpaketes wird in Zeile 21, über den Aufruf der entsprechenden Routine des Treibers der Funkeinheit, realisiert. Die Pausenzeit t_i wird über die Anweisung in Zeile 24 erreicht. Die auskommentierte Anweisung in Zeile 23 bietet eine Alternative dazu. Auch hier lässt sich feststellen, dass die Pausenzeit nicht ganz eingehalten werden kann. Sie dauert ca. 60 μs länger, als geplant. Dieses Problem lässt sich umgehen, indem die Pausenzeit mit 0,36 ms, statt mit 0,4 ms, definiert wird. Dieser Versatz könnte durch die Zeit, welche die CPU für Berechnungen benötigt und durch eventuelle Ungenauigkeiten des Xtimers zustande kommen.

5.3.3. Das Senden von Datenpaketen (Der Extended-Mode der Funkeinheit)

Wie bereits in Kap. 5.1 beschrieben, übernimmt die Funkeinheit AT86RF233 im Extended-Mode die Funktionalitäten der MAC-Schicht bzw. eines einfachen MAC-Protokolls. D. h. vor dem Senden eines Datenpaketes wird geprüft, ob der Kanal frei ist und es wird automatisch auf das ACK des Empfängers gewartet. Wenn der Kanal nicht frei sein sollte oder das ACK nicht empfangen wurde, wird der Sendeversuch automatisch wiederholt. Außerdem sendet der Treiber der Funkeinheit dem ContikiMAC-Thread über eine Interrupt-Routine eine Nachricht, um zu signalisieren, dass ein Sendevorgang begonnen wurde.[11] Dies führte dazu, dass der Aufruf in Zeile 21 in Listing 5.7 über 6 ms in Anspruch nahm, also eine viel zu lange Zeit, um die kurzen Pausenzeiten zwischen den Datenpaketen einzuhalten.[12] In den Anweisungen in den Zeilen 11 bis 14 ließ sich dieses Verhalten korrigieren. In Zeile 12 wird das automatische Empfangen des ACKs deaktiviert und in Zeile 13 die automatische Prüfung, ob der Kanal frei ist. Zeile 14 schließlich deaktiviert die Benachrichtigung an den ContikiMAC-Thread, dass ein Sendevorgang begonnen hat. Letzteres ist für den Löwenanteil der Zeit verantwortlich, da hier wieder IPC zum Einsatz kommt. Weiter oben wurde bereits festgestellt, dass die IPC unter RIOT ca. 5 ms zur Verarbeitung benötigt.

Damit ContikiMAC bei der Unicast-Datenübertragung das Senden der Kopien des Datenpaketes vorzeitig einstellen kann, muss ContikiMAC in den Pausenzeiten t_i in der Lage sein, das ACK des Empfängers zu empfangen. Dies stellt ein noch ungelöstes Problem dar, da der Extended-Mode bereits für das Empfangen des ACKs zuständig ist. Eine Lösung könnte darin bestehen, die Interrupt-Routine zu verwenden, welche nach dem Senden von der Funkeinheit aufgerufen wird, um den Mikrocontroller über das Ergebnis des Sendevorgangs zu informieren.

Die komplette Implementation der prototypischen Senden-Funktion ist auf der beiliegenden CD zu finden. Hinweise auf eine Internetquelle gibt Anlage A.5.

[11] Dies ergibt an dieser Stelle keinen Sinn, da der ContikiMAC-Thread ja selbst diesen Sendevorgang ausgelöst hat.

[12] Mit einem Sniffer wurde festgestellt, dass während dieser Zeit ein Datenpaket mit 45 Bytes gesendet wurde. Die reine Sendezeit dieses Paketes beträgt somit nur $\frac{45\ Bytes*8}{250000\ kbit/s} = 1,44\ ms$.

6. Energy Harvesting

Energy Harvesting bezeichnet den Prozess, bei welchem Energie aus vorhandenen Quellen der Umgebung für den Betrieb von Geräten mit geringer Leistung (wie Knoten in einem WSN) gewonnen wird (vgl. [59]). Obwohl Batterien nicht viel kosten und zuverlässige Stromlieferanten sind, gibt es dennoch einige Gründe für Energy Harvesting. Beispiele dafür sind:

- Sensoren und Aktoren im industriellen Umfeld sollen für eine lange Zeit (mehr als 10 Jahre) unterbrechungsfrei arbeiten und der Einsatz von Technikern, welche die Batterien der zahlreichen Knoten wechseln, soll vermieden werden (vgl. [45, S. 273]).
- Das Wechseln der Batterie ist nach der Installation eines Sensors schlicht nicht mehr möglich, da dieser nicht mehr zugänglich ist.
- Der Einsatz verteilter Sensoren zur Überwachung der Umwelt, z. B. zur Vermeidung von Waldbränden oder der Erkennung von Umweltverschmutzung (vgl. [45, S. 273]).
- Im Bereich Smart Home soll der Anwender ein Gerät, z. B. einen Lichtschalter oder Feuermelder installieren können, ohne sich danach jemals Gedanken um die Wartung machen zu müssen („Install and forget.“).

Aufgrund der Anforderung an einen meist sehr geringen Platzbedarf eines Knotens in einem WSN, kommt nicht jede Energiequelle für das Energy Harvesting in Frage. Geeignete Energiequellen sind:

Mechanische Energie kann aus Wind, der Bewegung von Wellen oder Fahrzeugen bzw. aus jeder Art von Vibrationen gewonnen werden (vgl. [27, S. 8]). Die Umwandlung in elektrische Energie kann mit Hilfe des Elektromagnetismus (z. B. mit einem Gleichstromgenerator), der Piezoelektrizität (z. B. mit einem Piezoelement) oder der Elektrostatik (z. B. mit einem Bandgenerator) erfolgen.

Temperaturgradienten, also Temperaturunterschiede, können genutzt werden, um elektrische Energie zu gewinnen (vgl. [27, S. 11]). Diese Art der Energieerzeugung kommt besonders dort in Frage, wo hohe Temperaturen herrschen, wie in der Nähe eines Ofens oder an Abgasleitungen technischer Anlagen. Thermoelektrische Generatoren sind zuverlässig und erfordern wenig bis keine Wartung während ihrer Lebenszeit. Allerdings müssen die Sensorknoten den extremen Temperaturbedingungen standhalten können.

Radiowellen bzw. elektromagnetische Wellen kommen vorallem in Ballungsgebieten zur Energiegewinnung in Frage. Quellen von Radiowellen sind Antennen zur Ausstrahlung von Radio- oder TV-Sendern, WLAN- und Mobilfunkgeräte usw. (vgl. [27, S. 14]). Radiowellen sind zudem, unabhängig von der Tageszeit, ständig verfügbar. Die Energieausbeute ist jedoch sehr gering und stark von der Frequenz und Entfernung der Quelle abhängig.

Mikrobielle Brennstoffzellen nutzen die Energie, welche durch elektrochemische Reaktionen, hervorgerufen durch die Aktivität von Bakterien, entsteht (vgl. [27, S. 20]). Dieses Prinzip wird z. B. angewendet, um Unterwasser-Überwachungssysteme mit Energie zu versorgen. Dabei werden die im Wasser vorhandenen Bakterien genutzt.

Photovoltaik bezeichnet die Umwandlung von Licht in elektrische Energie (vgl. [27, S. 23]). Bei Licht handelt es sich um sichtbare elektromagnetische Wellen, welche einem kleinen Teil des elektromagnetischen Spektrums entsprechen. Solarzellen sind in der Lage, Licht direkt in elektrische Energie, ohne Umwege, umzuwandeln.

Die Photovoltaik ist einer der günstigsten und zugleich einfachsten Wege, Energie aus der Umgebung zu gewinnen (vgl. [27, S. 29]). Aus diesem Grund soll die Photovoltaik im Folgenden näher betrachtet werden.

6.1. Photovoltaik

Grundsätzlich kommen bei der Photovoltaik nicht nur Solarzellen zum Einsatz. Es gibt z. B. auch noch photosynthetisch-elektrochemische Zellen, welche Energie aus Licht nach dem Prinzip der Pflanzen gewinnen. In dieser Arbeit soll unter Photovoltaik jedoch der Einsatz von Solarzellen verstanden werden.

Licht ist die primäre Energiequelle der Erde und weist die höchste Energiedichte pro Volumen auf (vgl. [27, S. 29]). Wie viel Energie das Licht der Sonne pro Fläche (gemessen auf Höhe des Meeresspiegels) liefert, hängt von der Atmosphäre ab, welche das Licht durchdringen muss. Da das Licht der Sonne mit unterschiedlichen Winkeln auf die Erdoberfläche einfällt, variiert dieser Wert von Region zu Region. In Europa liefert das Sonnenlicht bei direkter Einstrahlung und maximaler Intensität etwa eine Energie von 100 mW/cm^2. In beleuchteten Büros beträgt die Energie nur noch etwa 0.1 mW/cm^2. Heutige, in der Praxis gängige, Solarzellen haben typische Wirkungsgrade von 14 – 20 % (Solarzellen auf Basis von polykristallinem Silicium), 16 – 22 % (monokristallines Silicium) und 5 – 10 % (amorphes Silicium, vgl. [67]). Obwohl Solarzellen auf Basis von amorphen Silicium unter den genannten Typen die geringsten Wirkungsgrade erzielen, haben sie jedoch das beste Preis-Leistungs-Verhältnis.

Die Photovoltaik bietet theoretisch eine absolut wartungsfreie Energiequelle. In der Praxis ist jedoch zu bedenken, dass sich Staub auf den Solarzellen absetzen kann und diese im Freien der Witterung ausgesetzt sind. Daher gibt es auch für Solarmodule eine Angabe für die Lebensdauer (etwa von 25 bis 30 Jahren für poly- und monokristalline Silicium-Solarmodule, vgl. [67]).

6.2. Energy Harvesting für WSN-Knoten

Die Quellen denen sich Energy Harvesting bedient, sind oft unzuverlässig. Am Beispiel der Photovoltaik bedeutet dies, dass nachts keine Energie aus dieser Quelle zur Verfügung steht. Oder, dass an sonnigen Tagen viel mehr Energie produziert wird, als an bewölkten Tagen. Aus diesen Gründen wird die überschüssige Energie, welche unter guten Bedingungen gewonnen wurde, gespeichert (vgl. [27, S. 30]).

Gängige Energiespeicher für diesen Zweck sind Superkondensatoren und Akkus. Superkondensatoren weisen, bei gleicher Größe, eine wesentlich höhere Kapazität als andere Kondensator-Typen auf. Die Kapazität kann mehrere tausend Farad (F) betragen. Superkondensatoren in einer praktikablen Baugröße für WSN-Knoten haben eine typische Kapazität zwischen 1 und 50 F (vgl. [27, S. 102]). Die Nachteile von Superkondensatoren sind ihre hohe Selbstenladung und geringe Spannungsfestigkeit von nur wenigen Volt (z. B. 2,3 V). Da die typische Betriebsspannung eines WSN-Knotens ebenfalls sehr gering ist, kann letzterer Nachteil vernachlässigt werden. Akkus hingegen haben eine noch viel höhere Energiedichte, als Superkondensatoren (vgl. [27, S. 98]). Der Nachteil von Akkus gegenüber Superkondensatoren ist die geringere Lebensdauer, welche sich vorallem in einer niedrigeren Anzahl verträglicher Lade-Entladezyklen bemerkbar macht. Die Anzahl der Lade-Entladezyklen von Akkus bewegen sich, je nach Akku-Typ, im Bereich von mehreren Hundert bis einige Tausend Zyklen, wohingegen Superkondensatoren über eine Million Zyken überstehen können. Hinsichtlich maximal verträglicher Lade-Entladezyklen, Energiedichte, Wirkungsgrad beim Laden und Entladen und einigen anderen Eigenschaften (siehe [27, S. 98 ff.]), eignen sich Li-Ion- und Li-Polymer-Akkus besonders gut. Jedoch benötigen diese Akku-Typen wiederum eine nicht triviale Vorschaltelektronik, welche die Akkus vor Über-, Unterladung und im schlimmsten Fall sogar vor Entzündung schützen.

Energiespeicher dienen auch dazu Leistungsspitzen abzufangen. Wenn ein Knoten bspw. ein MAC-Protokoll mit RDC einsetzt, treten Leistungsspitzen während der Wachphasen auf. Den größten Teil der Zeit befindet sich der Knoten jedoch im Energiesparmodus, sodass der durchschnittliche Stromverbrauch weit unter dem der Leistungsspitze liegt. Die überschüssige Energie während des Energiesparmodus wird in einem Energiespeicher akkumuliert und steht in den Wachphasen zur Verfügung. Dies erlaubt eine viel kleinere Dimensionierung des Energy Harvesting-Systems. Für Solarzellen bedeutet dies also, dass sie eine viel kleinere Fläche einnehmen können.

Die Spannung, welche bei der Umwandlung der Energie aus der Umgebung in elektrische Energie entsteht, entspricht selten der benötigten Betriebsspannung des WSN-Knotens und hängt zudem von der Intensität der genutzen Energiequelle ab. Bei Solarzellen liegt die Spannung typischerweise unterhalb der benötigsten Betriebsspannung und hängt insbesondere von der Stärke des einfallenden Lichts ab. Daher ist ein Wandler notwendig, welcher die Spannung anhebt (Step-Up-Converter) bzw. herabsenkt (Step-Down-Converter). Zu beachten bei der Dimensionierung des Energy Harvesting-Systems ist, dass diese Wandler in der Praxis einen Wirkungsgrad $< 100\ \%$ haben, also verlustbehaftet sind.

Ferner hat jeder Energieumwandler, ob Solarzelle, Gleichstromgenerator, Piezoelement etc., einen optimalen Betriebspunkt, in welchem deren Effizienz am größten ist. Daher empfiehlt sich der Einsatz einer Maximum Power Point Tracking (MPPT)-Technik (siehe [27, S. 103 ff.]), zumal damit eine Effizienzsteigerung um $65 - 90\ \%$ erzielt werden kann.

6.2.1. Ein Rechenbeispiel

Die benötigte Fläche für ein Solarmodul soll anhand eines konkreten WSN-Knotens bestimmt werden. Dieser WSN-Knoten besteht aus der Platine deRFnode des Herstellers dresden elektronik ingenieurtechnik GmbH, bestückt mit dem Funkmodul deRFmega128-

22A00 desselben Herstellers. Als Herzstück kommt der, bereits in Kap. 2.3 vorgestellte, SoC ATmega 128RFA1 von Atmel zum Einsatz. Der Prozessortakt liegt bei 16 MHz. Als Software dient Contiki OS in der Version 3.0. Außerdem wird das MAC-Protokoll ContikiMAC mit den Standardparametern (siehe Tabelle 4.2 auf S. 59) eingesetzt. Der WSN-Knoten ist also quasi ständig erreichbar und kann auch als Router eingesetzt werden. Zur Vereinfachung der Berechnungen wird der Verlust, welcher an einem Step-Up-Converter entsteht und das Speichern der Energie, um den Betrieb in der Dunkelheit zu ermöglichen, vernachlässigt.

Der Spitzenstrom dieses WSN-Knotens liegt bei 22 mA[1]. Unter Verwendung von ContikiMAC treten diese Spitzenströme nur für einen Bruchteil der Zeit auf. Die restliche Zeit befindet sich der WSN-Knoten in einem Energiesparmodus, in welchem der Strom nur noch wenige Hundert nA beträgt. Der Autor berechnete unter dieser Konstellation einen durchschnittlichen Strom von 0,34 mA.[2] Bei einer Betriebsspannung von 3,3 V ergibt sich daraus eine durchschnittliche Leistungsaufnahme von $3,3\ V * 0,34\ mA = 1,1\ mW$.

Mit Hilfe eines kleinen Energiespeichers, welcher die Leistungsspitzen abfängt, kann ein Solarmodul nun so ausgelegt werden, dass die Durchschnittsleistung von 1,1 mW abgedeckt wird. Geht man vom Einsatz in gut beleuchteten Innenräumen aus (0.1 mW/cm^2) und legt man ein Solarmodul mit einer guten Effizient von 22 % zugrunde, müsste das Solarmodul eine Fläche von $\frac{1,1\ mW}{0,22*0,1\ mW/cm^2} = 50\ cm^2$ (z. B. 7,1 cm x 7,1 cm) einnehmen. Tabelle 6.1 zeigt weitere Ergebnisse für In- und Outdoor-Verhältnisse und unterschiedlicher Effizienzen von Solarmodulen.

Energiedichte des Lichts	**Effizienz des Solarmoduls**	**Benötigte Fläche für das Solarmodul**	
		Fläche	**äquivalente Rechteckfläche**
0.1 mW/cm^2 (gut beleuchteter Innenraum)	5 %	220 cm^2	14,9 cm x 14,9 cm
	15 %	73,34 cm^2	8,6 cm x 8,6 cm
	22 %	50 cm^2	7,1 cm x 7,1 cm
10 mW/cm^2 (Outdoor, stark bewölkter Wintertag)	5 %	2,2 cm^2	1,49 cm x 1,49 cm
	15 %	0,74 cm^2	0,86 cm x 0,86 cm
	22 %	0,5 cm^2	0,71 cm x 0,71 cm
45 mW/cm^2 (Outdoor, bewölkter Sommertag)	5 %	0,49 cm^2	0,7 cm x 0,7 cm
	15 %	0,17 cm^2	0,41 cm x 0,41 cm
	22 %	0,12 cm^2	0,34 cm x 0,34 cm
100 mW/cm^2 (Outdoor, maximale Sonneneinstrahlung)	5 %	0,22 cm^2	0,47 cm x 0,47 cm
	15 %	0,074 cm^2	0,28 cm x 0,28 cm
	22 %	0,05 cm^2	0,23 cm x 0,23 cm

Tabelle 6.1.: Weitere Ergebnisse zur Beispielrechnung

Bedenkt man, dass die Berechnungen sehr optimistisch sind (Vernachlässigung von Wandlungsverlusten, Verzicht auf das Speichern von Energie für den nächtlichen Betrieb etc.), erscheint der Indoor-Betrieb dieses WSN-Knotens mittels Solarzellen, selbst bei der Verwendung eines Solarmoduls mit einer sehr guten Effizienz, bei einer benötigten Fläche von 50 cm^2 unpraktikabel zu sein.

[1] Vom Autor gemessen.
[2] Der Strom hängt jedoch stark vom Datenverkehr im WSN ab.

6.2.2. Experiment von Wang u. a.

Bereits 2010 zeigten die Autoren Wang u. a., dass ein vollständig über Photovoltaik betriebenes Sensor-Netz auf Basis von ZigBee bzw. IEEE 802.15.4 in einem gut beleuchteten Bürogebäude möglich ist (siehe [55]). Allerdings verwendeten die Sensor-Knoten aus [55] kein energiesparendes MAC-Protokoll wie ContikiMAC, sondern übermittelten lediglich alle fünf Minuten Sensor-Werte. Die restliche Zeit nutzten sie einen Energiesparmodus und waren damit nicht quasi ständig erreichbar. Der durchschnittliche Energieverbrauch eines Knotens betrug damit 0,23 mW. (Der Knoten im obigen Rechenbeispiel hatte einen durchschnittlichen Energieverbrauch von 1,1 mW, also mehr als das Vierfache.) Die Fläche des Solarmoduls entsprach lediglich 5,81 cm x 4,86 cm (kleiner als eine Kreditkarte). Als Energiespeicher kam ein 10 F Superkondensator zum Einsatz.

6.2.3. Adaptive Strategie zur Anpassung der Leistungsaufnahme

Eine adaptive Strategie zur Anpassung der Leistungsaufnahme kann helfen die Betriebsdauer auszudehnen, wenn die gespeicherte Energie zur Neige geht oder die Energiequelle der Umgebung (z. B. das Licht) nicht mehr ausreicht (vgl. [27, S. 30]). Die Reduzierung der eigenen Leistungsaufnahme kann auf unterschiedliche Weisen erfolgen. Z. B. indem die Nutzung des Energiesparmodus, d. h. die Schlafphasen, ausgedehnt werden. Für einen Sensor würde das bedeuten, dass er die Übertragungsintervalle von Messwerten verlängert. Andere Möglichkeiten sind die Verringerung des CPU-Taktes oder des Senderadius. Welche Methode zur Reduzierung der Leistungsaufnahme infrage kommt, hängt wieder von der konkreten Anwendung ab. Ein Router in einem maschenförmigen WSN könnte bspw. die Kosten der Pfade (siehe auch Kap. 2.4.3), welche über ihn führen, anheben, damit via RPL andere Pfade für Knoten, welche diesen Router benutzen, berechnet werden und der Router somit entlastet wird.

7. Schlussbemerkungen

7.1. Zusammenfassung

Der Energieverbrauch eines Knotens hängt von vielen Faktoren ab:

- von der verwendeten Hardware wie dem Mikrocontroller, dem Funk-Chip, den angeschlossenen Sensoren und/oder Aktoren usw.
- von dem Prozessortakt mit dem der Mikrocontroller arbeitet[1]
- von der Häufigkeit der Mess- und Steuervorgänge eines Sensors bzw. Aktors
- wie oft und wie viele Daten übertragen werden
  - bei einem Sensor bzw. Aktor, wie oft Messwerte gesendet respektive Steuerbefehle empfangen werden
  - bei einem Router, wie viele Pfade über ihn führen und wie stark diese ausgelastet sind
- von der Sendeleistung
- vom Anteil der Zeit, welchen die Energiesparmodi einnehmen

7.1.1. Funkstandards

Ein energiesparendes Design beginnt bereits bei der Basis. Dazu gehört vor allem der unterliegende Funkstandard, welcher (mindestens) das Verfahren und grundlegende Datenformat zur Datenübertragung definiert (siehe Kap. 3). Um einen energiesparenden Betrieb zu erreichen, setzen Funkstandards für WSNs insbesondere auf Minimalismus (siehe Kap. 3.8). Ein beliebter vertreter dieser Funkstandards ist IEEE 802.15.4 (siehe Kap. 3.1), auf welchen unter anderen die höheren Standards ZigBee und 6LoWPAN aufbauen.

7.1.2. MAC-Protokolle

MAC-Protokolle steuern den Zugriff auf das Medium und üben damit großen Einfluss auf den Energieverbrauch aus. Ziel eines energiesparenden MAC-Protokolls ist es, die Funkeinheit eines WSN-Knotens für den größtmöglichen Anteil der Betriebszeit deaktiviert zu halten. Dabei gilt es einen Kompromiss zwischen einem möglichst geringen Energieverbrauch und einer möglichst niedrigen Latenz (Antwortzeit) zu finden. In Kap. 4 wurden energiesparende MAC-Protokolle untersucht, welche quasi ständige Erreichbarkeit, d. h. bei maximalen Latenzen von typischerweise einigen Millisekunden bis hin zu wenigen Sekunden, simulieren.

[1] Ein höherer Prozessortakt bedeutet allgemein einen höheren Energieverbrauch. Wenn ein Energiesparmodus benutzt wird, kann ein höherer Prozessortakt aber auch zu einem niedrigeren Energieverbrauch beitragen, da Berechnungen schneller abgeschlossen sind und somit insgesamt mehr Zeit im Energiesparmodus verbracht werden kann.

Es wird grob unterschieden in asynchrone und synchrone MAC-Protokolle. Asynchrone MAC-Protokolle sind leicht zu implementieren und erlauben niedrigere Latenzen (unter der Voraussetzung, dass das Medium nicht zu stark ausgelastet ist). Außerdem sind asynchrone MAC-Protokolle sehr flexibel, was die spontane Änderung der Topologie des WSNs und das Kommunikationsmuster der Knoten betrifft. Topologieänderungen und unvorhersehbare Kommunikationsmuster sind bspw. typisch für das IoT. Ein dem Stand der Technik entsprechendes asynchrones MAC-Protokoll ist ContikiMAC (siehe Kap. 4.2.5).

Bei synchronen MAC-Protokollen wird das Medium in Zeitschlitze unterteilt und es gibt einen Kommunikationsplan, welcher vorgibt, in welchem Zeitschlitz ein Knoten senden darf bzw. empfangsbereit sein muss. Dies setzt eine strikte Synchronizität der Knoten untereinander voraus, erlaubt aber auch einen größeren Energiespareffekt, vor allem da die Funkeinheit weniger Zeit mit Idle-Listening (siehe Kap. 4.1.1) verbringt. Synchrone MAC-Protokolle sind, insbesondere wegen der Notwendigkeit eines Planungsalgorithmus, sehr viel schwerer zu implementieren, als asynchrone MAC-Protokolle. Der Planungsalgorithmus berechnet den Kommunikationsplan und verteilt diesen über die Knoten im WSN. Die meisten Planungsalgorithmen in der Literatur setzen die genaue Kenntnis des Kommunikationsmusters im WSN voraus und sind damit an einen bestimmten Anwendungszweck, z. B. das Sammeln und Transportieren von Daten zu einer gemeinsamen Senke hin, gebunden. Zudem reagieren sie träge auf Topologieänderungen. Asynchrone MAC-Protokolle lasten das Medium durch das Senden von Strobes verhältnismäßig stark aus und es kommt vermehrt zu Kollisionen. Daher wird ein synchrones MAC-Protokoll bei größeren WSNs mit hunderten von Knoten schnell unumgänglich. Ein dem Stand der Technik entsprechendes synchrones MAC-Protokoll ist IEEE 802.15.4 TSCH. Es unterstützt Channel-Hopping und ist damit sehr Robust gegenüber Störungen. Synchrone MAC-Protokolle und Planungsalgorithmen sind typischerweise nicht aneinander gebunden und werden getrennt voneinander spezifiziert. Ein Planungsalgorithmus, welcher für IEEE 802.15.4 TSCH entwickelt wurde, prinzipiell aber mit jedem synchronen MAC-Protokoll zusammenarbeitet, wurde in Kap. 4.6 mit Orchestra vorgestellt. Orchestra kombiniert die Zuverlässigkeit synchroner mit der Flexibilität asynchroner MAC-Protokolle. Bei einem Vergleich von ContikiMAC mit IEEE 802.15.4 TSCH in Verbindung mit Orchestra anhand eines konkreten Anwendungsszenarios, lieferte das Gespann aus IEEE 802.15.4 TSCH und Orchestra von beiden MAC-Protokollen für alle Testmetriken (Energieverbrauch, Latenz und Stabilität) die besten Ergebnisse (siehe Anlage A.3).

7.1.3. Energy Harvesting

In Kap. 6 wurde kurz auf Energy Harvesting als theoretisch unerschöpfliche Energiequelle eingegangen. Die Photovoltaik ist dabei besonders beliebt, da sie einfach zu realisieren und günstig ist. Zudem eignet sie sich auch für den Einsatz in Gebäuden. Trotz Energy Harvesting ist der Einsatz von energiesparenden Funkstandards in Kombination mit energiesparenden MAC-Protokollen sinnvoll, da sich so Platz sparen lässt. Am Beispiel der Photovoltaik bedeutet dies, dass der Einsatz kleinerer Solarmodule und Energiespeicher, um den Betrieb in der Nacht sicherzustellen, möglich wird.

In Kap. 6.2.1 wurde gezeigt, dass sich der Einsatz von Photovoltaik in Gebäuden, selbst bei guter Beleuchtung, für quasi ständig erreichbare Knoten, aufgrund der benötigte Fläche für das Solarmodul, als schwierig gestaltet. Ein Anwendungsgebiet hierfür wäre z. B.

ein wartungsfreies Maschen-WSN, bestehend aus Sensoren, welche diverse Umgebungsparameter in Gebäuden, wie die Temperatur, überwachen.

7.2. Ausblick

IEEE 802.15.4 CSL (siehe Kap. 4.2.7) ist, neben ContikiMAC, ein weiteres dem Stand der Technik entsprechendes asynchrones MAC-Protokoll. Leider mangelt es derzeit an frei zugänglichen Implementationen des Protokolls, was auf die mangelnde Unterstützung der Funkeinheiten zurückzuführen sein könnte (siehe Kap. 4.3.5). Sollte sich dies in Zukunft ändern, wäre der Vergleich von ContikiMAC mit IEEE 802.15.4 CSL mittels einer Simulation mit konkreten Anwendungsfällen ein lohnenswertes Ziel, denn bereits der analytische Vergleich der beiden MAC-Protokolle in Kap. 4.3, spricht für das Potenzial von IEEE 802.15.4 CSL.

In Kap. 5 wurde die Möglichkeit untersucht das asynchrone MAC-Protokoll ContikiMAC, in das Open Source Betriebssystem RIOT zu portieren. Dabei wurden Schwierigkeiten aufgezeigt und wie man sie beheben kann. Die Implementierung einer voll funktionsfähigen Variante von ContikiMAC würde den Rahmen dieser Arbeit sprengen und bleibt daher Gegenstand zukünftiger Arbeiten.[2]
Weiterhin ist die Portierung von IEEE 802.15.4 TSCH mit Orchestra als Planungsalgorithmus von Contiki nach RIOT erstrebenswert. Allerdings fällt der Aufwand der Implementierung eines synchronen MAC-Protokolls, wie bereits erwähnt, deutlich größer aus. Denn nicht nur das MAC-Protokoll IEEE 802.15.4 TSCH, sondern auch Orchestra müssen portiert werden.

In dieser Arbeit nicht betrachtet wurden sog. Aufwachempfänger (siehe z. B. [54]). Dabei gibt es, neben der eigentlichen Funkeinheit, einen weiteren Empfänger, welcher auf ein bestimmtes Aufwachsignal reagiert. Der Aufwachempfänger kann eine andere Frequenz als die eigentliche Funkeinheit nutzen. Aufgrund der besseren Ausbreitungseigenschaften, würde sich z. B. eine niedrigere Frequenz anbieten. Der Knoten kann sich so permanent im Energiesparmodus befinden und diesen erst verlassen, wenn ein Aufwachsignal empfangen wurde (oder der Knoten von sich aus einen Sendevorgang initiieren möchte). Dies ermöglicht dem Knoten, praktisch beliebig lange im Energiesparmodus zu verweilen und dennoch verzögerungsfrei auf Anfragen zu reagieren.
Einen ähnlichen Ansatz bietet z. B. die Funkeinheit CC2500 von Texas Instruments mit einem *Wake On Radio* genannten Feature (siehe [15, S. 41]), welches eine Art Hardware-Implementation eines energiesparenden MAC-Protokolls ist. Mit aktiviertem Wake On Radio überprüft die Funkeinheit das Medium periodisch auf Datenpakete, ohne Interaktion des Mikrocontrollers. Zwischen den Prüfintervallen begibt sich die Funkeinheit in einen Energiesparmodus. Der Mikrocontroller aber kann die ganze Zeit über im Energiesparmodus verweilen und wird von der Funkeinheit geweckt, wenn diese Daten empfangen hat.

[2] RIOT verfügt bereits über LwMAC, ein weniger effizientes MAC-Protokoll (siehe Kap. 4.6). Die Autoren Daniel Krebs und Shuguo Zhuo haben über ein Jahr lang an LwMAC gearbeitet.

Literatur

[1] IETF working group „6LoWPAN“. *IPv6 over Low power WPAN (6lowpan)*. URL: https://datatracker.ietf.org/wg/6lowpan/about/ (besucht am 08. 06. 2017).

[2] IETF working group „6TiSCH“. *IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch)*. URL: https://datatracker.ietf.org/wg/6tisch/about/ (besucht am 25. 08. 2017).

[3] N. Accettura u. a. „Decentralized Traffic Aware Scheduling for multi-hop Low power Lossy Networks in the Internet of Things“. In: *2013 IEEE 14th International Symposium on „A World of Wireless, Mobile and Multimedia Networks“ (WoWMoM)*. Juni 2013, S. 1–6. DOI: 10.1109/WoWMoM.2013.6583485.

[4] ZigBee Alliance. *ZigBee IP Specification: ZigBee Document 095023r34, Revision 34*. März 2014.

[5] ZigBee Alliance. *ZigBee PRO Green Power feature specification: Basic functionality set, Version 1.0*. Jan. 2016.

[6] ZigBee Alliance. *ZigBee Specification*. Aug. 2015.

[7] zigbee alliance. *zigbee IP and 920IP*. URL: http://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeeip/ (besucht am 28. 05. 2017).

[8] zigbee alliance. *zigbee PRO with Green Power*. URL: http://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeepro/ (besucht am 07. 06. 2017).

[9] Z-Wave Alliance. *Z-Wave Alliance Recommendation ZAD12837-1: Z-Wave Transceivers - Specification of Spectrum Related Components*. Techn. Ber. Feb. 2015. URL: http://z-wavealliance.org/wp-content/uploads/2015/02/ZAD12837-1.pdf.

[10] Giuseppe Anastasi. *From IEEE 802.15.4 to IEEE 802.15.4e: Another Step towards the Internet of (Important) Things*. Mai 2014. URL: http://info.iet.unipi.it/~anastasi/talks/2014-Guangzhou.pdf (besucht am 16. 06. 2017).

[11] *ATmega128RFA1 Datasheet*. TJA1043. 8266F-MCU Wireless-09/14. Atmel Corporation. Sep. 2014.

[12] Universität Bern. *Implementing a LPP version of BEAM in omnet++ 4.x*. URL: http://rvs.unibe.ch/teaching/diplom/diplom_LLP_LPP.html (besucht am 02. 08. 2017).

[13] Michael Buettner u. a. „X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks“. In: *in SenSys*. Nov. 2006, S. 307–320.

[14] Bundesnetzagentur. *Allgemeinzuteilungen*. URL: https://www.bundesnetzagentur.de/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/allgemeinzuteilungen-node.html (besucht am 03. 05. 2017).

[15] *CC2500, Low-Cost Low-Power 2.4 GHz RF Transceiver*. SWRS040C. Texas Instruments. Aug. 2017.

[16] Contiki Community. *at86rf2xx: remove CCA check #6355*. URL: https://github.com/RIOT-OS/RIOT/pull/6355 (besucht am 06. 10. 2017).

[17] Contiki Community. *drivers/at86rf2xx: CCA when in Extended Mode #3169*. URL: https://github.com/RIOT-OS/RIOT/issues/3169 (besucht am 06. 10. 2017).

[18] Contiki Community. *Generic (GNRC) network stack*. URL: http://www.riot-os.org/api/group__net__gnrc.html (besucht am 05. 10. 2017).

[19] Contiki Community. *LWMAC: A simple duty cycling 802.15.4 MAC protocol (2nd try). #6554*. URL: https://github.com/RIOT-OS/RIOT/pull/6554 (besucht am 05. 10. 2017).

[20] Contiki Community. *Radio duty cycling*. URL: https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling (besucht am 03. 08. 2017).

[21] Contiki Community. *Simplest possible MAC layer*. URL: https://riot-os.org/api/group__net__gnrc__lwmac.html (besucht am 15. 10. 2017).

[22] Atmel Corporation. *http://www.atmel.com/tools/atsamr21-xpro.aspx*. URL: http://www.atmel.com/tools/atsamr21-xpro.aspx (besucht am 05. 10. 2017).

[23] Waltenegus Dargie Dargie und Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley, 2010.

[24] Adam Dunkels. *The ContikiMAC Radio Duty Cycling Protocol*. Report. 2011.

[25] Simon Duquennoy u. a. „Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH“. In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. SenSys '15. Seoul, South Korea: ACM, 2015, S. 337–350. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2809714. URL: http://doi.acm.org/10.1145/2809695.2809714.

[26] Simon Duquennoy u. a. *TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation*. Juni 2017.

[27] Carlos Manuel Ferreira Carvalho und Nuno Filipe Silva Veríssimo Paulino. *Energy Harvesting Electronic Systems*. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-21617-1. DOI: 10.1007/978-3-319-21617-1_2. URL: https://doi.org/10.1007/978-3-319-21617-1_2.

[28] EnOcean GmbH. *Energy Harvesting Wireless Power for the Internet of Things*. Techn. Ber. Aug. 2015. URL: https://www.enocean.com/fileadmin/redaktion/pdf/white_paper/White_Paper_Internet_of_Things_EnOcean.pdf.

[29] EnOcean GmbH. *EnOcean Technology – Energy Harvesting Wireless*. Techn. Ber. Juli 2011. URL: https://www.enocean.com/fileadmin/redaktion/pdf/white_paper/WP_EnOcean_Technology_en_Jul11.pdf.

[30] EnOcean GmbH. *Funktechnologie von EnOcean für batterielose Funksensor-Lösungen*. URL: https://www.enocean.com/de/technology/radio-technology/ (besucht am 22. 06. 2017).

[31] FieldComm Group. *Die Mitglieder der HART® Communication Foundation und der Fieldbus Foundation™ genehmigen die Gründung der FieldComm Group*. URL: http://de.hartcomm.org/hcf/org_mbr/news/pr2014/fieldcommgrp.html (besucht am 17. 05. 2017).

[32] S. S. Guclu, T. Ozcelebi und J. J. Lukkien. „Improving Broadcast Performance of Radio Duty-Cycled Internet-of-Things Devices“. In: *2016 IEEE Global Communications Conference (GLOBECOM)*. Dez. 2016, S. 1–7. DOI: 10.1109/GLOCOM.2016.7841888.

[33] Domenico De Guglielmo, Giuseppe Anastasi und Alessio Seghetti. „From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things“. In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2014, S. 135–152. DOI: 10.1007/978-3-319-03992-3_10. URL: https://doi.org/10.1007/978-3-319-03992-3_10.

[34] José A. Gutiérrez u. a. *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors With IEEE 802.15.4*. Standards Information Network, IEEE Press, 2010.

[35] EnOcean Alliance Inc. *EnOcean-Funkstandard*. URL: https://www.enocean-alliance.org/de/was-ist-enocean/enocean-funkstandard/ (besucht am 23. 06. 2017).

[36] EnOcean Alliance Inc. *Unsere Mitglieder*. URL: https://www.enocean-alliance.org/de/mitgliedschaft/unsere-mitglieder/ (besucht am 22. 06. 2017).

[37] ISO/IEC. *Information technology - Home Electronic Systems (HES) - Part 3-10: Wireless Short-Packet (WSP) protocol optimized for energy harvesting - Architecture and lower layer protocols*. Techn. Ber. 2011.

[38] ITU-T. *Short range narrow-band digital radiocommunication transceivers - PHY, MAC, SAR and LLC layer specifications*. Techn. Ber. Jan. 2015.

[39] Markus Krauße und Rainer Konrad. *Drahtlose ZigBee-Netzwerke: Ein Kompendium*. Springer Vieweg, 2014. ISBN: 978-3-658-05821-0.

[40] N. Kushalnagar, G. Montenegro und C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919. RFC Editor, Aug. 2007. URL: http://www.rfc-editor.org/rfc/rfc4919.txt.

[41] Michael Methfessel u. a. „Real-Life Deployment of Bluetooth Scatternets for Wireless Sensor Networks“. In: *Real-World Wireless Sensor Networks: Proceedings of the 5th International Workshop, REALWSN 2013, Como (Italy), September 19-20, 2013*. Cham: Springer International Publishing, 2014, S. 43–51. ISBN: 978-3-319-03071-5. DOI: 10.1007/978-3-319-03071-5_4. URL: https://doi.org/10.1007/978-3-319-03071-5_4.

[42] B. A. Nahas u. a. „Low-Power Listening Goes Multi-channel“. In: *2014 IEEE International Conference on Distributed Computing in Sensor Systems*. Mai 2014, S. 2–9. DOI: 10.1109/DCOSS.2014.33.

[43] M. Ojo, D. Adami und S. Giordano. „Performance evaluation of energy saving MAC protocols in WSN operating systems“. In: *2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. Juli 2016, S. 1–7. DOI: 10.1109/SPECTS.2016.7570509.

[44] M. R. Palattella u. a. „Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15.4e networks“. In: *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*. Sep. 2012, S. 327–332. DOI: 10.1109/PIMRC.2012.6362805.

[45] Shashank Priya und Daniel J. Inman. *Energy Harvesting Technologies*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 0387764631, 9780387764634.

[46] Andreas Sebayang. „Classic, Smart und Smart Ready: Warum Bluetooth nicht gleich Bluetooth ist“. In: (17. Jan. 2014). URL: https://glm.io/103934 (besucht am 04. 07. 2017).

[47] Bluetooth SIG. *Bluetooth Core Specification*. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification (besucht am 04. 07. 2017).

[48] Bluetooth SIG. *Bluetooth Core Specification v5.0*. Techn. Ber. Dezember 2016.

[49] Ivanovitch Silva u. a. „Reliability and Availability Evaluation of Wireless Sensor Networks for Industrial Applications“. In: *Sensors* 12.1 (2012), S. 806–838. ISSN: 1424-8220. DOI: 10.3390/s120100806. URL: http://www.mdpi.com/1424-8220/12/1/806.

[50] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Techn. Ber. Sep. 2011.

[51] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). Amendment 1: MAC sublayer*. Techn. Ber. Apr. 2012.

[52] R. Soua, P. Minet und E. Livolant. „MODESA: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks“. In: *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*. Dezember 2012, S. 91–100. DOI: 10.1109/PCCC.2012.6407742.

[53] Ridha Soua, Pascale Minet und Erwan Livolant. „Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15.4e TSCH networks“. In: *Transactions on Emerging Telecommunications Technologies* 27.4 (2016). ett.2991, S. 557–575. ISSN: 2161-3915. DOI: 10.1002/ett.2991. URL: http://dx.doi.org/10.1002/ett.2991.

[54] Christoph Tzschoppe. *Integrierter Aufwachempfänger in einer BiCMOS-Halbleitertechnologie*. Jörg Vogt Verlag, 2017. ISBN: 978-3-95947-008-7. URL: http://vogtverlag.de/buecher/9783959470087.html.

[55] W. Wang u. a. „Autonomous wireless sensor network based Building Energy and Environment Monitoring system design“. In: *2010 The 2nd Conference on Environmental Science and Information Application Technology*. Bd. 3. Juli 2010, S. 367–372. DOI: 10.1109/ESIAT.2010.5568311.

[56] T. Watteyne, M. Palattella und L. Grieco. *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*. RFC 7554. RFC Editor, Mai 2015. URL: http://www.rfc-editor.org/rfc/rfc7554.txt.

[57] Wikipedia. *6LoWPAN*. URL: https://de.wikipedia.org/wiki/6LoWPAN (besucht am 08. 06. 2017).

[58] Wikipedia. *Abstandsgesetz*. URL: https://de.wikipedia.org/wiki/Abstandsgesetz (besucht am 21. 07. 2017).

[59] Wikipedia. *Energy harvesting*. URL: https://en.wikipedia.org/wiki/Energy_harvesting (besucht am 25. 08. 2017).

[60] Wikipedia. *Enocean*. URL: https://de.wikipedia.org/wiki/Enocean (besucht am 22. 06. 2017).

[61] Wikipedia. *EUI-64*. URL: https://de.wikipedia.org/wiki/EUI-64 (besucht am 04. 05. 2017).

[62] Wikipedia. *Freiraumdämpfung*. URL: https://de.wikipedia.org/wiki/Freiraumd%C3%A4mpfung (besucht am 27. 10. 2017).

[63] Wikipedia. *IEEE 802.11ah*. URL: https://en.wikipedia.org/wiki/IEEE_802.11ah (besucht am 07. 05. 2017).

[64] Wikipedia. *Media Access Control*. URL: https://de.wikipedia.org/wiki/Media_Access_Control (besucht am 11. 05. 2017).

[65] Wikipedia. *Positionsschalter*. URL: https://de.wikipedia.org/wiki/Positionsschalter (besucht am 30. 04. 2017).

[66] Wikipedia. *Routing Protocol for Low power and Lossy Networks*. URL: https://de.wikipedia.org/wiki/Routing_Protocol_for_Low_power_and_Lossy_Networks (besucht am 09. 06. 2017).

[67] Wikipedia. *Solarzelle*. URL: https://de.wikipedia.org/wiki/Amorphes_Material (besucht am 27. 09. 2017).

[68] Wikipedia. *ZigBee*. URL: https://de.wikipedia.org/wiki/ZigBee (besucht am 25. 05. 2017).

[69] Wikipedia. *Z-Wave*. URL: https://de.wikipedia.org/wiki/Z-Wave (besucht am 28. 06. 2017).

[70] J. Zhou u. a. „Comparison of IEEE 802.15.4e MAC features“. In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. März 2014, S. 203–207. DOI: 10.1109/WF-IoT.2014.6803159.

[71] Dušan Živadinović. „Kleine Umwälzung: Wie Bluetooth 5 Reichweite und Datenrate erhöht“. In: *C't 01/2017*. Heise Medien GmbH & Co. KG, 2017, S. 34–35. URL: http://heise.de/-3575330.

A. Anlagen

A.1. Gegenüberstellung der Eigenschaften der Funkstandards

Die Tabellen A.1 bis A.3 auf den folgenden Seiten stellen die wichtigsten Eckdaten der Funktechnologien aus Kap. 3 gegenüber.

	Ersterscheinung	**Einsatzgebiet**
IEEE 802.15.4	2003	LR-WPANs bzw. WSNs allgemein.
WirelessHART	September 2007	Prozess-Überwachung und -Steuerung im industriellen Umfeld.
6LoWPAN	September 2007	IoT-Anwendungen jeglicher Art.
ZigBee	2004	LR-WPANs bzw. WSNs allgemein. In der Praxis jedoch häufig zur Gebäudeautomation bzw. Heimautomatisierung.
EnOcean	2001	Gebäudeautomation, Smart Home und IoT-Anwendungen
Z-Wave	2001	Hausautomatisierung bzw. Smart Home.
Bluetooth Smart	Dezember 2009	WSNs, IoT-Anwendungen und Wearables.

Tabelle A.1.: Gegenüberstellung der Funkstandards hinsichtlich Entwicklungsbeginn und Einsatzgebiet.

Tabelle A.2 zeigt die Größen, Sendedauern und Größen der Payloads eines Datenrahmens auf PHY-Ebene (PPDU). Die letzte Spalte gibt die mögliche Payload-Größe auf der obersten Schicht, etwa der Anwendungsschicht (Application (APP)-Schicht), an.

	PHY-Schicht			**APP-Schicht**
	PPDU (in Bytes)	**Sendedauer (in ms)**	**PPDU-Payload (in Bytes)**	**Payload (in Bytes)**
IEEE 802.15.4[a]	11 (ACK) 15 – 133	0,352 (ACK) 0,48 – 4,256	5 (ACK) 9 – 127	88 – 122[b]
WirelessHART	siehe IEEE 802.15.4			8 Process Val.[c]
6LoWPAN	siehe IEEE 802.15.4			41 / 33 (UPD) / 21 (TCP)
ZigBee	siehe IEEE 802.15.4			60 – 90
EnOcean	12,5 – 33	0,8 – 2,112		7 – 20
Z-Wave (bei 6,9, 40 und 100 kbit/s)	22 – 76 19 – 105 21 – 211	18,333 – 63,333 3,8 – 21 1,68 – 16,88	bis 64 bis 64 bis 170	54 54 158
Bluetooth Smart	10 – 265	0,08 – 2,12 (1 Mbit/s) 0,04 – 1,06 (2 Mbit/s)	2 – 257	0 – 251

[a] Alle Angaben beziehen sich auf das 2,4-GHz-Band.
[b] Angabe bezieht sich auf die MAC-Schicht.
[c] Angabe in Process Values (Temperatur, Druck, Füllstand, Sollwert, usw.).

Tabelle A.2.: PPDU-Größen, -Sendedauern und Payload-Größen.

	Frequenzbänder	Datenraten (kbit/s)	
IEEE 802.15.4	868 MHz	20, 100 und 250	⊙[b]
	915 MHz	40 und 250	
	2,4 GHz	250, 1000, 2000[a]	
	250 - 750 MHz[a]	110, 850, 1700, 6810, 27240	
	3,1 - 4,8 GHz[a]		
	6 - 10,6 GHz[a]		
	780 MHz[c]	250	
	950 MHz[d]	20 und 250	
WirelessHART	2,4 GHz	250	⊕
6LoWPAN	Wie IEEE 802.15.4.		
ZigBee	Theoretisch wie IEEE 802.15.4, praktisch jedoch meist 2,4 GHz mit einer Datenrate von 250 kbit/s.		⊖[e]
EnOcean	868 MHz[f]	125	⊖
	902 MHz[g]		
	928 MHz[d]		
Z-Wave	850 - 950 MHz[h]	9,6, 40, 100	⊖
Bluetooth Smart	2,4 GHz	125[i], 500[i], 1000, 2000	⊕

Unterstützt Channel-Hopping: ⊕ Ja ⊙ Bedingt ⊖ Nein

[a] Amendment IEEE 802.15.4t-2017
[b] Ja, seit dem TSCH- und CSL-Modus.
[c] Nur in China
[d] Nur in Japan
[e] Kein Channel-Hopping. Der Netzwerkfunkkanalmanager kann aber ein Kanalwechsel des gesamten Netzwerkes im Betrieb vornehmen, um Störungen auszuweichen.
[f] Europa und China
[g] Nordamerika
[h] Jenachdem, was im jeweiligen Einsatzgebiet zulässig ist. In Europa werden 868,4 MHz (9,6 und 40 kbit/s) und 869,85 MHz (100 kbit/s) verwendet.
[i] Eigentlich 1000 kbit/s. Durch den Einsatz der Fehlerkorrekturverfahren halbiert bzw. achtelt sich die Datenrate jedoch. Vorteil ist, dass sich die Reichweite erheblich steigern lässt.

Tabelle A.3.: Gegenüberstellung der Funkstandards hinsichtlich genutzer Frequenzbänder, Datenraten und Channel-Hopping

A.2. Zusammenfassung der Eigenschaften der MAC-Protokolle

Abb. A.1 stellt die Eigenschaften von MAC-Protokollen in einem hierarchischen Graph dar. Ganz unten ist die Zuordnung der MAC-Protokolle aus Kap. 4.2 zu diesen Eigenschaften zu sehen. Die Tabelle A.4 auf der nächsten Seite zeigt diese Zuordnung ebenfalls, jedoch ergänzt um weitere Eigenschaften.

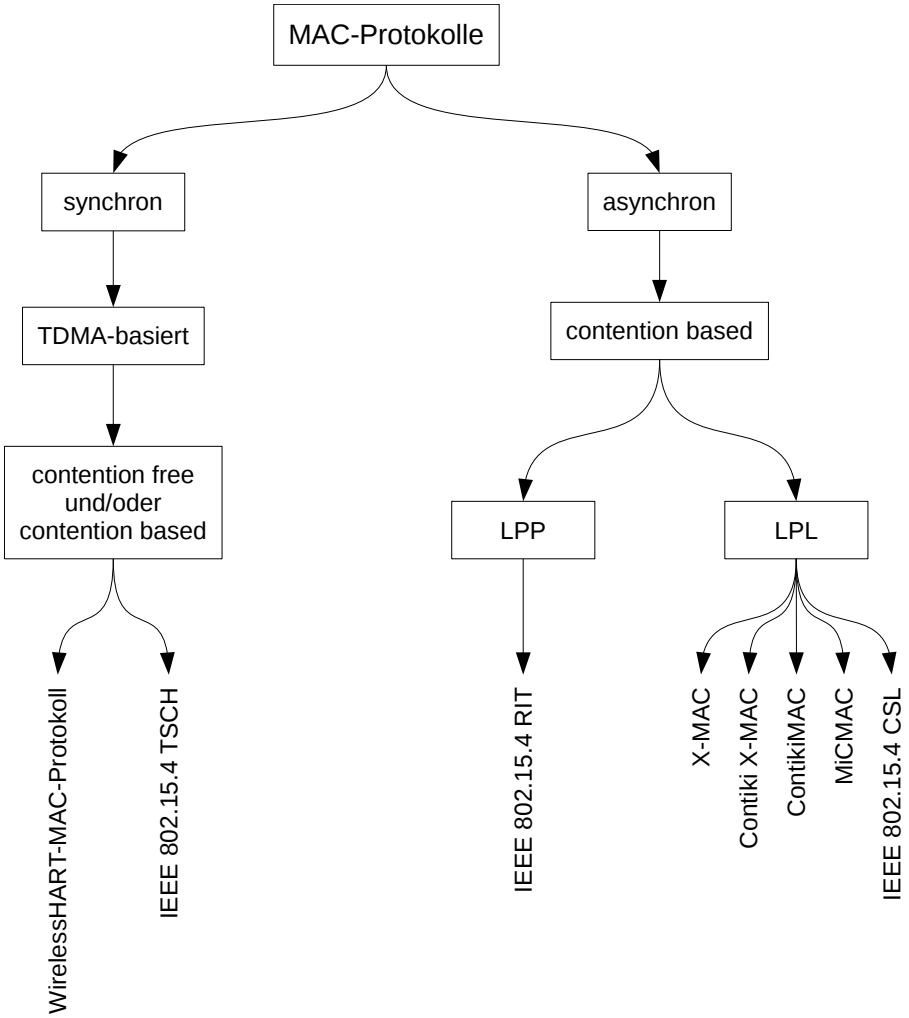


Abbildung A.1.: Die Eigenschaften von MAC-Protokollen als hierarchischer Graph
Quelle: Eigene Darstellung

	TDMA	Contention-based	LPP/LPL	(A)Synchron	CH	Netzwerktopologien
WirelessHART-MAC-Protokoll	Ja	Nein	–	synchron	Ja	Masche
IEEE 802.15.4 TSCH	Ja	Ja in Shared Links	–	synchron	Ja	keine bestimmte
X-MAC	Nein	Ja	LPL	asynchron	Nein	keine bestimmte
Contiki X-MAC	Nein	Ja	LPL	asynchron	Nein	keine bestimmte
ContikiMAC	Nein	Ja	LPL	asynchron	Nein	keine bestimmte
MiCMAC	Nein	Ja	LPL	asynchron	Ja	keine bestimmte
IEEE 802.15.4 CSL	Nein	Ja	LPL	asynchron	Ja	keine bestimmte
IEEE 802.15.4 RIT	Nein	Ja	LPP	asynchron	Nein	keine bestimmte

CH - Channel-Hopping

Tabelle A.4.: Zusammenfassung der Eigenschaften der MAC-Protokolle aus Kap. 4.2

A.3. Zusammenfassung der Ergebnisse der Performance-Analyse von Duquennoy u. a.

Die Autoren Duquennoy u. a. verglichen die Performance vor der Minimalkonfiguration von TSCH (TSCH minimal), TSCH mit Orchestra (TSCH dedicated), ContikiMAC (LPL) und bei ständig aktivierter Funkeinheit (CSMA) miteinander (siehe [26]). Die Versuche wurden mit einem realen WSN, bestehend aus einem Knoten als Senke und vier Blatt-Knoten, welche in einer Stern-Netzwerktopologie angeordnet waren, durchgeführt. Die Blatt-Knoten sendeten zu einem zufälligen Zeitpunk ein UDP-Datenpaket an die Senke. Die Versuche wurden mehrmals und mit unterschiedlichen Paket-Intervallen von 0,25 s, 1 s, 4 s und 16 s wiederholt. Abb. A.2 zeigt die Zusammenfassung der Ergebnisse der Performance-Analyse. Die Packet Reception Rate (PRR) und die Packet Delivery Rate (PDR) sind Indikatoren für die Stabilität des Netzwerkes. Je höher deren Werte, umso besser. Die PRR ist ein Maß dafür, wie viele Pakete auf der Link-Layer-Ebene beim ersten Versuch zugestellt werden konnten. Ein Versuch gilt als Fehlschlag, wenn entweder nicht gesendet werden kann, weil der Kanal belegt ist oder das ACK des Empfängers innerhalb eines Timeouts nicht empfangen wurde. Der Link-Layer wiederholt fehlgeschlagene Sendeversuche automatisch bis zu einer gewissen Anzahl an Versuchen, bevor der Sendeversuch endgültig als fehlgeschlagen gilt. Ein Großteil der Übertragungsfehler bleibt damit für höhere Schichten transparent. Die PDR gibt die Erfolgsrate der Ende-zu-Ende-Übertragung an. Die Übertragung eines Paketes wird bei der PDR erst als Fehlschlag gewertet, wenn der Link-Layer die Übertragungsfehler nicht abfangen konnte. Der Energieverbrauch wird über den RDC-Wert im Diagramm *Duty Cycle* angegeben. Ein höherer Wert bedeutet auch einen höheren Energieverbrauch. Zu beachten ist, dass die Darstellung der Y-Achse des Diagramms für die Latenz logarithmisch ist. Außerdem zu beachten ist, dass der Duty Cycle für CSMA stets 100 % beträgt und nicht vollständig im Diagramm eingezeichnet ist.

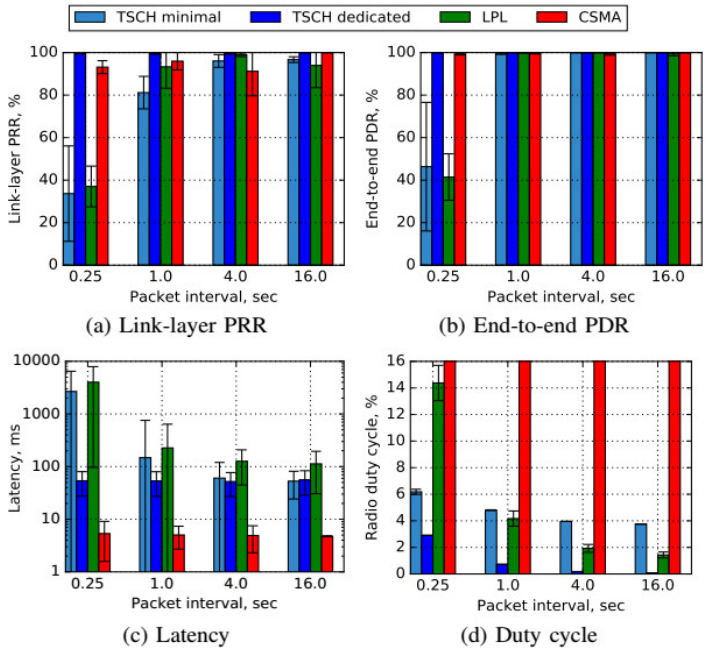


Abbildung A.2.: Ergebnisse der Performance-Analyse von TSCH, ContikiMAC (LPL) und CSMA

Quelle: [26, S. 6]

A.4. Messungen einer CCA-Phase von ContikiMAC in RIOT

Abb. A.3 auf der nächsten Seite zeigt die Messungen einer CCA-Phase der prototypischen Implementation von ContikiMAC in RIOT mit einem Oszilloskop. Die Messungen wurden an der Hardware-Plattform SAM R21 Xplained Pro Evaluation Kit (samr21-xpro) durchgeführt. Die rote Linie zeigt den Spannungsverlauf während einer CCA-Phase. Dabei wurde der Spannungsabfall an einem 5 Ohm Shunt-Widerstand gemessen, welcher anstelle des, für Strommessungen vorgesehenen, Jumpers eingesetzt wurde. (Der Stromverlauf kann dann mittels des Ohmschen Gesetzes berechnet werden.) Die blaue Linie zeigt den Spannungsverlauf, welcher während einer CCA-Phase an einem Ausgabe-Pin gemessen wurde. Der Pin wurde per Software unmittelbar vor dem API-Aufruf der CCA-Funktion des Treiber auf Betriebsspannung (high) und sofort danach wieder auf Masse (low) geschaltet. Über den Verlauf der blauen Linie lässt sich somit die Zeit, welche der API-Aufruf benötigt, messen (917 μs). Die Zeit zwischen zwei Aufrufen der CCA-Funktion des Treibers wird mit Hilfe des Xtimers von RIOT, gemäß der Standardparameter von ContikiMAC (siehe t_c in Tabelle 4.2 auf S. 4.2), um 0,5 ms verzögert. Tatsächlich beträgt die Pausenzeit jedoch 0,559 ms, was auf die Ungenauigkeit des Xtimers zurückzuführen sein könnte. Die beiden Spitzen im Verlauf der roten Linie zeigen, an welchen Stellen der eigentliche CCA der Funkeinheit durchgeführt wird. Die beiden Spitzen haben jeweils eine Dauer von ungefähr 174 μs. Die eigentliche Dauer eines CCAs beträgt laut Datenblatt 128 μs im 2,4-GHz-Band. Wo genau dieser in dem 174 μs breiten Bereich liegt, lässt sich anhand des Diagramms nicht feststellen. Zu erkennen ist auch, dass zwischen diesen beiden Spitzen respektive den beiden eigentlichen CCAs ungefährt 1,3 ms liegen. Das ist mehr als das Doppelte der vorgesehenen Pausenzeit. Lässt man die Verzögerung durch den Xtimer weg, ergibt sich immer noch eine Pause von 741 μs ($1,3\,ms - 559\,\mu s$).

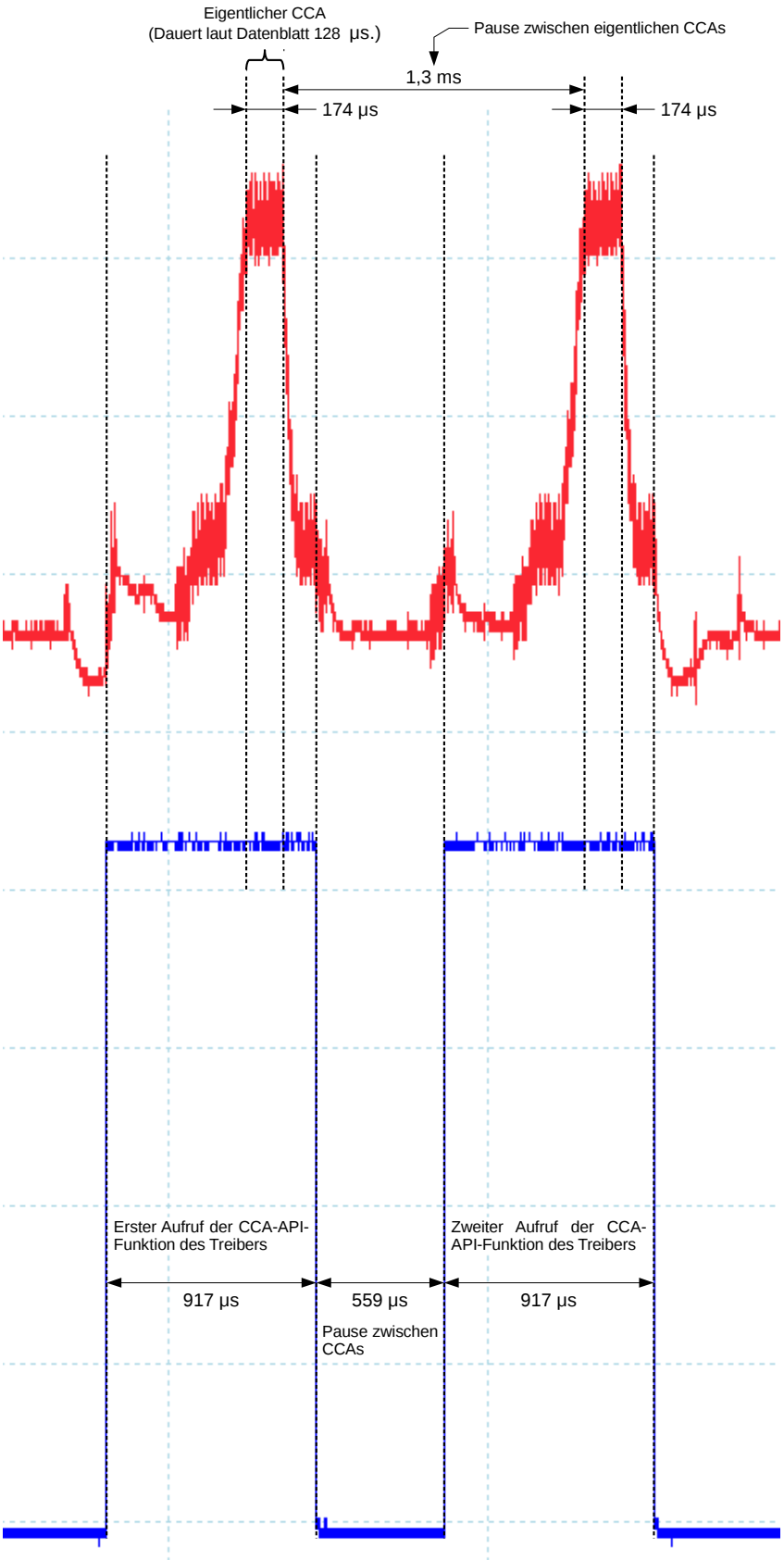


Abbildung A.3.: Messungen einer CCA-Phase der prototypischen Implementation von ContikiMAC in RIOT

Eigene Darstellung

A.5. Hinweise zur Internetquelle der Implementation von ContikiMAC in RIOT

Implementation der CCA-Phase von ContikiMAC

Enthält den notwendigen Code für das RIOT-Modul für ContikiMAC (gnrc_contikimac) und die Implementation der CCA-Phase. Die LED LED0 des Boards SAM R21 Xplained Pro Evaluation Kit (samr21-xpro) ändert jedes Mal, wenn ein Paket detektiert wurde, ihre Status.

GitHub: `https://github.com/ghavag/RIOT/tree/ContikiMAC-CCA_only`

Implementation der Senden-Funktion von ContikiMAC

Wie die Implementation der CCA-Phase, enthält jedoch zusätzlich den Code der Senden-Funktion. LED0 leuchtet jedes Mal auf, wenn ein Paket gesendet wird.

GitHub: `https://github.com/ghavag/RIOT/tree/ContikiMAC`

Liste der geänderten und hinzugefügten Dateien

Folgende Dateien wurden im RIOT-Quellcode-Baum geändert (M) und hinzugefügt (A):

(M) `Makefile.dep`
(M) `sys/auto_init/netif/auto_init_at86rf2xx.c`
(A) `sys/include/net/gnrc/contikimac/contikimac.h`
(A) `sys/include/net/gnrc/contikimac/types.h`
(M) `sys/include/net/gnrc/mac/types.h`
(M) `sys/include/net/gnrc/netdev.h`
(M) `sys/net/gnrc/Makefile`
(A) `sys/net/gnrc/link_layer/contikimac/Makefile`
(A) `sys/net/gnrc/link_layer/contikimac/contikimac.c`
(A) `tests/contikimac/Makefile`
(A) `tests/contikimac/README.md`
(A) `tests/contikimac/main.c`
(A) `tests/contikimac/udp.c`

Testen der Implementationen

Zum Testen kann die Applikation `tests/contikimac` kompiliert (`make all`) und auf das Board samr21-xpro geflasht werden (`make flash`). Soll die Applikation für anderen Boards kompiliert werden, muss die Zeile `BOARD ?= samr21-xpro` in der Datei `tests/contikimac/Makefile` zuvor angepasst werden.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ort, Datum

Unterschrift des Verfassers