HAW HAMBURG

BACHELOR THESIS
Christoph Busch

Ein ESP-NOW basiertes Mesh-Netzwerk für Smart-Home-Anwendungen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Christoph Busch

Ein ESP-NOW basiertes Mesh-Netzwerk für Smart-Home-Anwendungen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr. Martin Becke

Eingereicht am: 30.11.2022

Christoph Busch

Thema der Arbeit

Ein ESP-NOW basiertes Mesh-Netzwerk für Smart-Home-Anwendungen

Stichworte

ESP32, ESP-NOW, ESP-WIFI-MESH, BATMAN, Mesh Netzwerk, Smart Home

Kurzzusammenfassung

Im Zuge dieser Bachelorarbeit wurde eine Analyse von ESP-WIFI-MESH – einer Mesh-Netzwerk-Lösung für die ESP32-Plattform – durchgeführt und ein alternativer Ansatz zur Konzeption eines Mesh-Netzwerks für die ESP32-Plattform formuliert. Durch die bei ESP-WIFI-MESH verwendete Baumtopologie kann ein Knoten nur mit seinem Eltern- und Kindknoten direkt kommunizieren, nicht jedoch mit anderen Knoten, die sich in Funkreichweite befinden. Zudem führt der Ausfall eines Knotens im Netzwerk auch zum Ausfall seines Teilbaums. Damit kann der Root-Knoten kurzfristig einen Single Point of Failure darstellen. Als Alternative dazu wurde ein dezentral organisiertes Mesh-Netzwerk mit einer Mesh-Topologie entwickelt, das aufgrund der Mesh-Topologie kürzere Wege zwischen benachbarten Knoten ermöglichen soll. Zudem soll die dezentrale Organisation des Netzwerks Ausfälle einzelner Knoten schneller kompensieren können. Dazu wurde eine geeignete Kommunikationstechnologie mit großer Kompatibilität innerhalb der ESP32-Mikrocontroller-Familie (ESP-NOW) sowie ein geeignetes Routing-Protokoll (BATMAN IV) gewählt. Auf Grundlage dessen wurde ein Protokoll für den Austausch von Routing-Informationen und Nutzdaten im Mesh-Netzwerk entwickelt. Die Ergebnisse zeigen, dass die entwickelte Lösung kürzere Wege bei benachbarten Knoten finden kann, eine geringere Round Trip Time sowie Startzeit des Netzwerks aufweist und darüber hinaus Ausfälle einzelner Knoten schneller kompensieren kann als ESP-WIFI-MESH.

Christoph Busch

Title of Thesis

An ESP-NOW based Mesh Network for Smart Home Applications

Keywords

ESP32, ESP-NOW, ESP-WIFI-MESH, BATMAN, Mesh Network, Smart Home

Abstract

As part of this bachelor thesis, an analysis of ESP-WIFI-MESH – a mesh network solution for the ESP32 platform – was performed and an alternative approach to designing a mesh network for the ESP32 platform was proposed. Due to the tree topology used in ESP-WIFI-MESH, a node can only communicate directly with its parent and child nodes, but not with other nodes that are within its wireless range. Moreover, the failure of a node in the network also leads to the failure of its subtree. Thus, the root node can represent a single point of failure that results in the short-term failure of the entire network if it fails. As an alternative, a decentrally organized mesh network with a mesh topology has been developed, in which the mesh topology allows for shorter paths between neighboring nodes. In addition, the decentralized organization of the network should be able to compensate for failures of individual nodes more quickly. For this purpose, a suitable communication technology with high compatibility within the ESP32 microcontroller family (ESP-NOW) and a suitable routing protocol (BATMAN IV) were chosen. Based on these, a protocol for the exchange of routing information and payload data in the mesh network was developed. The results show that the developed solution can find shorter paths across neighboring nodes, has a lower round trip time and network startup time, and can also compensate for failures of individual nodes more quickly when compared to ESP-WIFI-MESH.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Seit Jahren nimmt die drahtlose Vernetzung von Haushaltsgeräten und Haustechnik zu, um ein höheres Level an Komfort und Automatisierung sowie effizienteres Energiemanagement für die Benutzenden zu ermöglichen [LR21] [Res18]. Werden diese Geräte direkt oder indirekt, zum Beispiel durch ein Gateway an das Internet angebunden, werden sie als Internet of Things (IoT) Geräte bezeichnet [AIM10]. Um eine größere Flexibilität bei Positionierung der IoT-Geräte im Haushalt zu ermöglichen, können diese per Funknetzwerk an ein bereits vorhandenes WLAN oder über ein Gateway an das Internet angebunden werden. Da jedoch nicht in jeder Umgebung eine optimale Funkabdeckung durch Gateways oder Access Points bereitgestellt werden kann, müssen Lücken in der Funkabdeckung überwunden werden, um dennoch eine Erreichbarkeit herzustellen. Mesh-Netzwerke stellen diese Funktionalität bereit, indem sich die Geräte untereinander verbinden, um Nachrichten auszutauschen und diese für andere Geräte weiterzuleiten. Dadurch kann die Funkabdeckung ohne Nutzung zusätzlicher oder spezialisierter Hardware erweitert werden. [LTQ+17]

Die ESP32-Plattform ist eine kostengünstige Mikrocontroller-Familie des chinesischen SoC-Entwicklers Espressif, die sich durch einen geringen Stromverbrauch und integrierte WLAN-Funktionalität auszeichnet und damit – laut Hersteller – insbesondere für Internet of Things Geräte geeignet sein soll. Mit ESP-WIFI-MESH stellt der Hersteller eine Mesh-Netzwerk-Lösung für die ESP32-Plattform bereit, um die Erreichbarkeit der Geräte trotz unzureichender Funkabdeckung herzustellen. Diese Technologie ist auf die ESP32-Plattform beschränkt und macht sich das Prinzip der Infrastruktur-WLANs für den Aufbau eines Mesh-Netzwerks zunutze. Dabei ist jeder Knoten im Mesh-Netzwerk Access Point und Station zugleich. [Esph] [Espb]

Aufgrund dieser hierarchischen Organisation ist ein ESP-WIFI-MESH-Netzwerk allerdings an eine Baum-Topologie gebunden, wodurch einige Nachteile entstehen. So können Ausfälle einzelner Knoten möglicherweise Ausfälle für weite Teile des Netzwerks nach

sich ziehen. Eine teil- oder vollvermaschte Topologie würde hingegen direktere und damit möglicherweise kürzere Wege zwischen den Knoten ermöglichen sowie weitere Wege schaffen, die im Falle eines Ausfalls genutzt werden können.

Ziel der Bachelorarbeit ist die Entwicklung eines Mesh-Netzwerks für die ESP32-Plattform. Dazu wird ESP-NOW zur Kommunikation zwischen den Teilnehmern genutzt. ESP-NOW ist ein Kommunikationsprotokoll des Herstellers Espressif und ermöglicht verbindungslose Kommunikation über WLAN [Espa]. Mit dieser Technologie als Grundlage zur Kommunikation soll ein Mesh-Netzwerk entwickelt werden, das teil- und vollvermaschte Topologien ermöglicht und dadurch als Alternative zu ESP-WIFI-MESH Vorteile bieten soll. Der Betrieb ohne Root-Knoten soll eine höhere Widerstandsfähigkeit ermöglichen, da ein Single Point of Failure entfällt. Außerdem soll die Mesh-Topologie im Vergleich mit der Baum-Topologie von ESP-WIFI-MESH zu einer direkteren Vernetzung der Knoten untereinander führen und damit kürzere Wege ermöglichen. Dazu muss ein geeignetes Routing-Protokoll gewählt werden. Da es sich bei der ESP32-Plattform um Mikrocontroller handelt, sollte das Routing-Protokoll wenig Rechenzeit in Anspruch nehmen und einen geringen Speicherbedarf aufweisen.

Zudem soll eine API bereitgestellt werden, die Konfiguration und Nutzung der Mesh-Funktionalität mit wenigen Zeilen Code ermöglichen soll. Der Quellcode wird zur Nachvollziehbarkeit und Erweiterbarkeit nach Fertigstellung vollständig veröffentlicht.

Im zweiten Kapitel wird der WLAN-Standard 802.11 als Grundlage für die Technologien ESP-NOW und ESP-WIFI-MESH ausgeführt sowie die Grundlagen zu Routing-Algorithmen und Mesh-Netzwerken erklärt. Darauf folgt eine Analyse im dritten Kapitel, die sich mit ESP-WIFI-MESH auseinandersetzt und damit die bereits adressierten Probleme aus der Einleitung vertieft. Verwandte Arbeiten, sowohl zu ESP-WIFI-MESH als auch zu (Mesh-)Netzwerken auf Basis von ESP-NOW werden im vierten Kapitel behandelt. Die Kapitel fünf und sechs behandeln die Konzeption und Umsetzung des zu entwickelnden Mesh-Netzwerks. Im siebten Kapitel werden einige Eigenschaften des entwickelten Mesh-Netzwerks mit ESP-WIFI-MESH verglichen. Abschließend wird im achten Kapitel ein Fazit gezogen.

2 Grundlagen

Im diesem Kapitel wird der WLAN-Standard 802.11 als Grundlage für die Technologien ESP-NOW und ESP-WIFI-MESH behandelt. Zudem werden Grundlagen und Eigenschaften von Mesh-Netzwerken und Routing-Verfahren dargelegt.

2.1 IEEE 802.11

802.11 ist ein Standard des *Institute of Electrical and Electronics Engineers* (IEEE) für die drahtlose Datenübertragung in lokalen Netzwerken (WLAN) und wurde 1997 in der ersten Version veröffentlicht und ist auf den Webseiten der IEEE erhältlich[1]. Seit der ersten Version (802.11-1997), die Datenraten von 1-2 MBit/s ermöglicht hat, wurde der Standard kontinuierlich weiterentwickelt und erweitert. Mit der Erweiterung 802.11ac aus dem Jahr 2013 sind Datenraten im Gigabit-Bereich möglich. [Rec12]

802.11 spezifiziert zwei Schichten für die Datenübertragung im WLAN. Der Medium Access Control Layer (MAC Layer) entspricht im OSI-Referenzmodell der Sicherungsschicht und definiert ein Zugriffsverfahren zur Kollisionsvermeidung auf dem geteilten Übertragungsmedium sowie die Verwaltung der Frames (Pakete). Der Physical Layer (PHY Layer) wird im OSI-Referenzmodell als Bitübertragungsschicht bezeichnet und definiert unterschiedliche Verfahren zur Datenübertragung auf dem Übertragungsmedium in verschiedenen Frequenzbereichen. [Rec12]

[1] https://standards.ieee.org/ieee/802.11/7028/

2.1.1 Betriebsmodi

Der Standard umfasst zwei Modi für den Betrieb eines Funknetzwerks. Im Infrastruktur-Modus wird das WLAN durch einen oder mehrere Access Points (AP) verwaltet, die zudem den Datenaustausch der verbundenen Geräte koordinieren. Der Ad-Hoc-Modus wird zur spontanen Vernetzung mehrerer Geräte untereinander genutzt. Beide Modi sind seit 802.11-1997 im Standard enthalten. [Gas05] [IEE97]

Infrastruktur-Modus

Im Infrastruktur-Modus wird zwischen zwei Gerätetypen unterschieden.

Station (STA)
: Ein 802.11-kompatibles Gerät mit drahtlosem Netzwerk-Interface. [IEE21] [Gas05]

Access Point (AP)
: Ein Access Point ist eine Station, die als Bridge den Zugang zu einem weiteren Netzwerk für die verbundenen Stations bereitstellen kann.
Zudem können APs an ein (Wireless) Distribution System angebunden sein, um Roaming zwischen mehreren APs zu ermöglichen und die Funkabdeckung zu erweitern. [IEE21] [Rec12]

Im Infrastruktur-Modus können Stations die Verbindung zu einem AP herstellen, sich jedoch nicht untereinander verbinden. Dadurch entsteht eine Stern-Topologie, wie die Abbildung 2.1 zeigt. Wenn eine Verbindung zu einem AP besteht, können Daten ausgetauscht werden. Dabei läuft die Kommunikation zwischen den Stations über den AP. [Rec12] [Gas05]

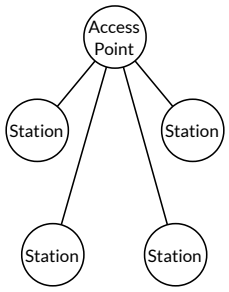

Abbildung 2.1: Stern-Topologie eines WLAN im Infrastruktur-Modus

Ein Access Point und die verbundenen Stations stellen eine Funkzelle dar, die man als Basic Service Set (BSS) bezeichnet. Das BSS wird durch die MAC-Adresse des AP identifiziert. Innerhalb des BSS verwaltet ein AP das drahtlose Netzwerk und leitet Frames der Stations weiter. Zudem können Access Points an weitere Netzwerke angebunden sein und somit als Bridge zwischen den Netzwerken agieren. So kann ein AP zum Beispiel den Zugang zum Internet bereitstellen. [AST12] [Gas05]

Um die Reichweite eines WLANs zu erweitern, können mehrere AP innerhalb eines Netzwerks verbunden werden. Dies bezeichnet man bei einem kabelgebundenen Netzwerk als Distribution System (DS) und bei einem kabellosen Netzwerk als Wireless Distribution System (WDS). Durch die Anbindung mehrerer AP an einem (W)DS entsteht ein Extended Service Set (ESS), das alle BSS der angebundenen AP umfässt. Damit lässt sich die Funkreichweite des Netzwerks erweitern. Ein ESS ermöglicht den Stations im Betrieb zwischen BSS zu wechseln. Dieses Verfahren wird als Roaming bezeichnet. [AST12] [Gas05]

Die Abbildung 2.2 zeigt einen Überblick der erwähnten Komponenten eines WLAN im Infrastruktur-Modus.

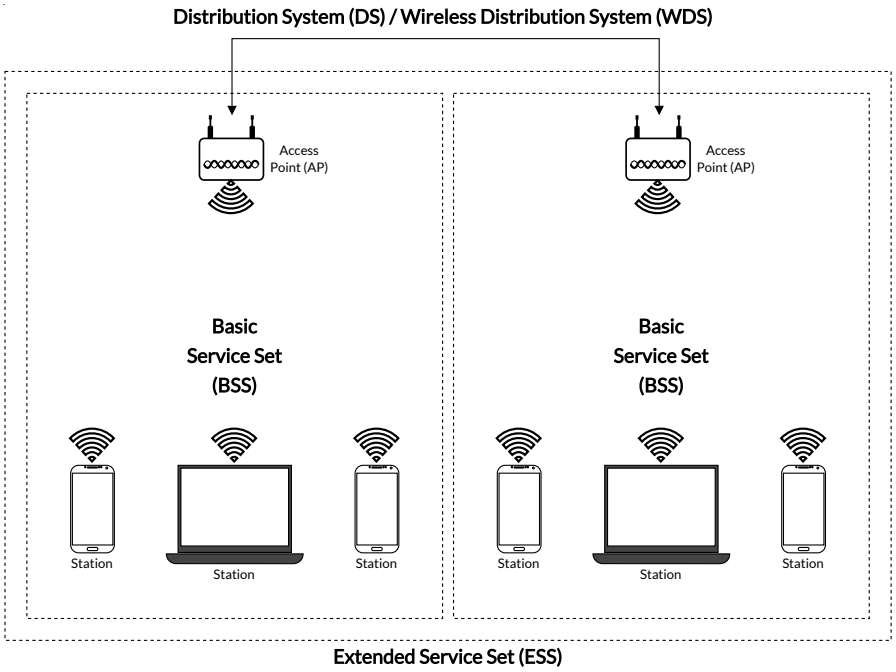

Abbildung 2.2: Übersicht: WLAN im Infrastruktur-Modus, nach [Rec12, S. 46]

Ad-Hoc-Modus

Im Ad-Hoc-Modus können sich Stations, die im Ad-Hoc-Modus konfiguriert sind, untereinander spontan verbinden, um Teil eines Ad-Hoc-Netzwerks zu werden. Dazu wird kein Access Point als zentrale Instanz des BSS benötigt. Die verbundenen Stations stellen eine Funkzelle dar, die im Ad-Hoc-Modus als Independant Basic Service Set (IBSS) bezeichnet wird. [Rec12] [Gas05]

Die Abbildung 2.3 zeigt einen Vergleich von BSS im Infrastruktur-Modus und IBSS im Ad-Hoc-Modus.

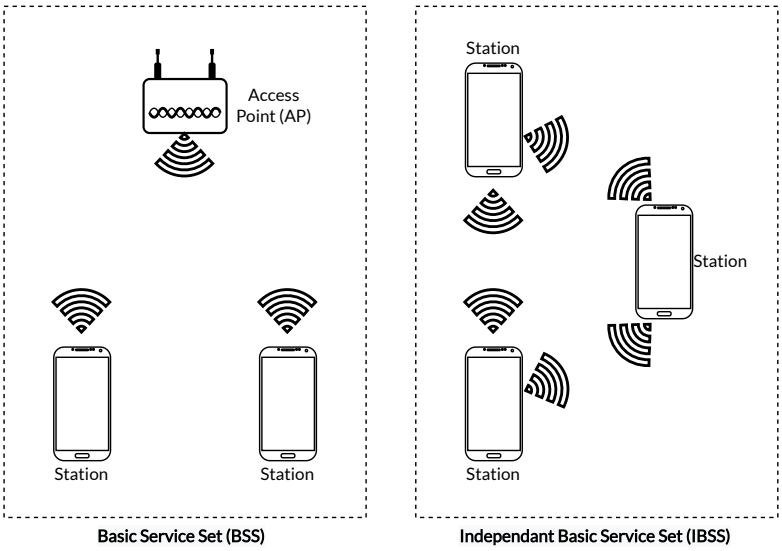

Abbildung 2.3: Übersicht: BSS im Infrastruktur-Modus und IBSS im Ad-Hoc-Modus, nach [AST12, S. 350]

Eine Station kann im Ad-Hoc-Modus mit den direkt erreichbaren Stations kommunizieren (Single-Hop) [Gas05].

2.1.2 Frames

Vor der Datenübertragung werden die Daten im MAC-Layer in Pakete aufgeteilt, die als Frames oder *MAC Protocol Data Unit* (MPDU) bezeichnet werden [Rec12]. Die Abbildung 2.4 zeigt den generellen Aufbau eines 802.11-Frames.

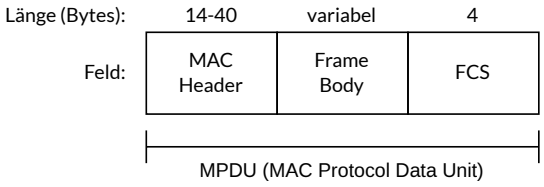

Abbildung 2.4: 802.11 Frame, nach [IEE21, S. 756]

MAC-Header

Mit dem MAC-Header werden Informationen über das Netzwerk sowie über die Zustellung, das Ziel, den Ursprung und den Typ des Frames übermittelt. Je nach Art der Information, die übertragen werden soll, muss einer der im Standard spezifizierten Frame-Typen gewählt werden, siehe 2.1.2. [Rec12] [IEE21]

Frame Body

Im Frame Body können sowohl Nutzdaten als auch Informationen zur Verwaltung des Netzwerks übertragen werden. Der Frame-Typ zeigt die Art des Inhalts an. [IEE21] [Gas05]

Frame Check Sequence (FCS)

Das FCS-Feld enthält eine 4-Byte Cyclic Redundancy Check (CRC) Prüfsumme, die vom Sender über MAC-Header und Frame-Body berechnet wird. Wenn eine FCS nicht mit der aus den empfangenen Daten errechneten FCS übereinstimmt, kann von einem Fehler bei der Datenübertragung ausgegangen werden. [IEE21] [Rec12]

Jeder 802.11-Frame enthält einen MAC-Header und die Frame Check Sequence. Wenn Nutzdaten übertragen werden, ist außerdem ein Frame Body vorhanden. [Rec12]

Ein Empfänger bestätigt den Erhalt eines Frames durch das Senden eines Acknowledgement-Frames. Erhält der Absender eines Frames nach einer festgelegten Zeit kein Acknowledgement, wird der Frame erneut gesendet. Nur der Empfang per Unicast übertragener Frames wird von dem Empfänger bestätigt. Zudem gibt es einige Frame-Typen, die explizit keine Bestätigung erfordern. [Rec12] [IEE21]

MAC-Header

Der MAC-Header ist das erste Feld des Frames und enthält je nach Typ des Frames, unterschiedliche Felder. Dieser Abschnitt erklärt die Bedeutung einiger relevanter Felder.

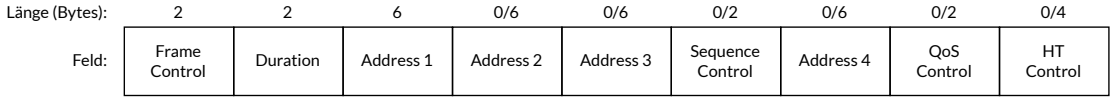

Abbildung 2.5: 802.11 MAC-Header, nach [IEE21, S. 756]

Frame Control

Das Frame-Control-Feld enthält Informationen zu der Art und dem Aufbau des Frames und wird auf der folgenden Seite im Detail dargelegt.

Address 1-4

Die Adressen 1-4 können Empfänger und Absender des Frames enthalten. Die Bedeutung und Anzahl der Adressfelder hängt von den To-/From-DS-Bits im Frame-Control-Feld ab. [IEE21] [Rec12]

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	Empfänger	Sender	BSSID	entfällt
0	1	Empfänger	BSSID	Sender	entfällt
1	0	BSSID	Sender	Empfänger	entfällt
1	1	Empfänger (AP)	Sender (AP)	Empfänger (STA)	Sender (STA)

Tabelle 2.1: Bedeutung der Adressfelder, nach [Rec12, S. 50]

Frame Control

Das Frame-Control-Feld ist das erste Feld des MAC-Headers und enthält weitere Eigenschaften des Frames. Dieser Abschnitt erklärt die Bedeutung einiger relevanter Felder.

Länge (Bits):	2	2	4	1	1	1	1	1	1	1	1
Feld:	Version	Type	Subtype	From DS	To DS	More Fragments	Retry	Power Management	More Data	Protected	HT Control

Abbildung 2.6: 802.11 Frame Control Feld, nach [IEE21, S. 757]

Version

Dieses Feld zeigt die Version des verwendeten Protokolls an. Derzeit sind die Versionen 0 und 1 im Standard vorgesehen. [IEE21]

Type & Subtype

Der Type gibt einen der folgenden Frame-Typen an. Zusätzlich werden die Typen in Subtypen unterteilt. [Rec12]

Management Frame

Management-Frames übernehmen unterschiedliche Funktionen der Netzwerk-Verwaltung. Sie werden unter anderem für den Beitritt zu einem Infrastruktur-Netzwerk, die Bekanntmachung eines Access Points oder dem Wechsel des Funkkanals genutzt. [Rec12]

Control Frame

Mit Control Frames werden Empfangsbestätigungen für erhaltene Frames gesendet sowie Informationen zum Zugriffsverfahren ausgetauscht, um die Sende-/Empfangsbereitschaft signalisieren. [Rec12]

Data Frame

Data Frames werden für die Übertragung der Nutzdaten eingesetzt. [Rec12]

Im weiteren Verlauf des Kapitels wird auf den Management-Frame näher eingegangen, da dieser Typ für die Kommunikation mit ESP-NOW relevant ist.

From/To DS

Da Frames nicht nur innerhalb eines (I)BSS übertragen werden können, sondern zum Beispiel auch über das (W)DS weitergeleitet oder verteilt werden, muss in diesen Feldern der Übertragungsweg vermerkt werden. Davon hängt auch die Bedeutung der Adressfelder ab. [IEE21] [Rec12]

To DS	From DS	Bedeutung
0	0	Distribution System wird weder betreten noch verlassen. Der Frame wird innerhalb eines IBSS gesendet. Das Netzwerk befindet sich im Ad-Hoc-Modus. [Rec12]
0	1	Der Frame verlässt das Distribution System und wird von einem Access Point an eine Station weitergeleitet. [Rec12]
1	0	Der Frame wird von einer Station gesendet und von einem Access Point in das Distribution System weitergeleitet. [Rec12]
1	1	Der Frame wird innerhalb des Distribution System weitergeleitet. [Rec12]

Tabelle 2.2: Bedeutung der From/To-DS-Felder, nach [Rec12, S. 50]

Retry

Das Retry Feld wird gesetzt, wenn auf einen gesendeten Frame keine Empfangsbestätigung erhalten wurde und der Frame erneut übertragen wird. [Gas05]

Protected

Wenn der Frame Body verschlüsselt ist, wird das Protected Feld gesetzt. [Gas05]

2.1.3 Management-Frame

Da der Management-Frame für die Kommunikation mit ESP-NOW relevant ist, wird ausschließlich auf diesen Frametypen näher eingegangen.

Management-Frames übertragen Informationen zur Verwaltung des Netzwerks und umfassen 14 Subtypen, um verschiedene Funktionalitäten zu realisieren [IEE21] [Rec12].

Ein Management-Frame folgt dem Aufbau eines 802.11-Frames aus Abbildung 2.4, enthält jedoch nicht alle Felder des generellen MAC-Headers aus Abbildung 2.5. Die Abbildung 2.7 zeigt den Aufbau eines Management-Frames.

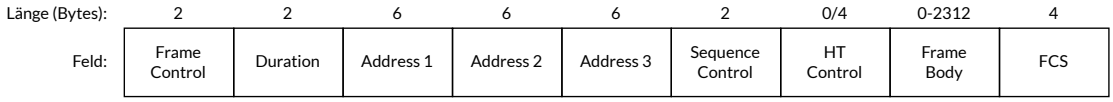

Abbildung 2.7: 802.11 Management Frame, nach [IEE21, S. 855]

Das vierte Adressfeld des MAC-Headers wird nur für Übertragungen innerhalb des (W)DS genutzt. Da Management-Frames das BSS nicht verlassen, entfällt für diesen Frame-Typ das vierte Adressfeld. [Rec12] [IEE21]

In dem Frame Body eines Management-Frames sind je nach Subtyp unterschiedliche Informationen enthalten, die in einem oder mehreren Feldern und/oder Information Elements (IE) vorliegen. Für jeden Subtyp des Management-Frame legt der Standard erforderliche und optionale Felder/IE fest. Als Felder werden Daten mit fester, durch den Standard vorgegebener Länge bezeichnet. Information Elements (IE) hingegen können eine variable Länge haben. [Rec12]

Die Abbildung 2.8 zeigt den Aufbau eines IE.

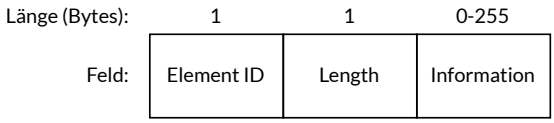

Abbildung 2.8: 802.11 Information Element, nach [IEE21, S. 939]

Ein IE hat einen Header, der eine *Element ID* enthält, um die Art des IE zu identifizieren, sowie ein Feld für die Länge des Inhalts. Darauf folgen bis zu 255 Byte Informationen. [IEE21]

2.2 ESP-NOW

ESP-NOW ist ein Kommunikationsprotokoll des chinesischen SoC-Entwicklers Espressif, das für die ESP32- und ESP8266-Plattform entwickelt wurde. Die Datenübertragung erfolgt verbindungslos über WLAN. Daher müssen die Geräte vor der Datenübertragung keinen Handshake durchführen und keine Nachrichten austauschen, um eine Verbindung aufrechtzuerhalten. Das Protokoll setzt auf den 802.11 MAC-Layer auf. So soll zusätzlicher Overhead durch weitere Protokolle vermieden werden. Die Nutzdaten aus der Anwendung werden um einen ESP-NOW-Header ergänzt und in einen WLAN-Frame eingesetzt. Dabei können mit einer Nachricht bis zu 250 Bytes Nutzdaten per Uni- oder Broadcast übertragen werden. [Espd] [Espa]

2.2.1 Datenübertragung

Auf Unicast-Nachrichten antwortet das Ziel mit einem Acknowledgement-Frame. Empfängt der Sender kein Acknowledgement, wird der Frame mit gesetztem Retry-Bit erneut gesendet. Die Anzahl der Retries ist nicht dokumentiert und kann nicht verändert werden. Für Broadcast-Nachrichten erhalten die Absender kein Acknowledgement von den Empfängern. Um den mehrfachen Empfang eines Frames zu vermeiden, fügt ESP-NOW ein 4 Byte Feld mit zufälligen Werten in jeden Frame ein. Erhält ein Empfänger einen Frame mit dem gleichen Zufallswert des zuletzt empfangenen Frames, wird dieser verworfen. [Espe] [Fla]

2.2.2 Verschlüsselung

Counter with CBC-MAC Protocol (CCMP) ist ein Verschlüsselungsprotokoll, das Teil des WLAN-Standards 802.11 ist und auf 802.11-Frames angewendet werden kann. Dabei werden die Nutzdaten des Frames mit AES im Counter-Mode (AES-CTR) verschlüsselt, um die Vertraulichkeit zu wahren. Den Kommunikationspartnern muss der Schlüssel bekannt sein, da Advanced Encryption Standard (AES) ein symmetrisches Verschlüsselungsverfahren ist und damit den gleichen Schlüssel zur Ver-/Entschlüsselung nutzt. AES wird als Blockchiffre bezeichnet, da die Ver-/Entschlüsselung blockweise erfolgt. Im Counter-Modus können einzelne Blöcke unabhängig voneinander ver- und entschlüsselt werden, wodurch sich dieses Verfahren gut parallelisieren lässt. Zusätzlich wird ein Message Authentication Code (MAC) über MAC-Header und Nutzdaten erzeugt, um Integrität und

Authentizität der Nachricht zu sichern. Der MAC wird zwischen Nutzdaten und Frame Check Sequence (FCS) in den Frame eingefügt. [Rec12] [Gas05] [AST12]

ESP-NOW setzt CCMP für die Verschlüsslung von Unicast-Nachrichten ein. Um die Verschlüsselung zu aktivieren, muss zunächst ein Primary Local Key (PMK) festgelegt werden. Zusätzlich muss für jeden Peer ein Local Master Key (LMK) hinterlegt werden. Sowohl PMK als auch LMK haben jeweils eine Länge von 16 Byte. Nachdem ein LMK für einen Peer hinzugefügt wurde, wird dieser mit dem PMK durch AES-128 in einem unbekannten Modus verschlüsselt. Der PMK und LMK sind pre-shared Keys und müssen den beteiligten Geräten zum Zeitpunkt der Kommunikation bekannt sein. [Espa]

2.2.3 Peer-Liste

Um eine Nachricht zu senden, muss das Ziel mit seiner MAC-Adresse und einigen weiteren Eigenschaften in eine Peerliste eingetragen werden. Für den Empfang von Broadcasts muss die Broadcast-MAC-Adresse in die Peerlist eingetragen werden. Die Liste kann bis zu 20 Peers enthalten, davon maximal 6 mit aktivierter Verschlüsselung. Für jeden Peer kann die Verschlüsselung per Flag aktiviert und deaktiviert werden. Zeigt das Flag eine aktive Verschlüsselung an, muss ein LMK für den Peer hinterlegt sein. Zudem muss der für die Datenübertragung genutzte WLAN-Kanal angegeben werden. Dieser muss mit dem genutzten Kanal des Empfängers übereinstimmen. [Espa] [Espe]

2.3 Wireless Mesh Networks

Als Wireless Mesh Network (WMN) wird ein Netzwerk bezeichnet, dessen Teilnehmer ein drahtloses Kommunikationsmedium nutzen und in einer Mesh-Topologie angeordnet sind. Die Teilnehmer, auch Knoten (engl.: Node) genannt, können als Router agieren, indem sie Nachrichten für andere Knoten in Richtung des Ziels weiterleiten. Knoten können spontan das Netzwerk betreten und verlassen und damit jederzeit die Struktur des Netzwerks verändern. Die Mesh-Topologie sowie das Routing sollen diese Änderungen dynamisch kompensieren. [Hel05]

In den folgenden Kapiteln werden Knoten, die sich in Funkreichweite befinden, als direkt erreichbare Knoten oder (Single-Hop-)Nachbarn bezeichnet. Knoten, die sich nicht in Funkreichweite befinden, jedoch durch Weiterleitung der Nachricht über andere Knoten erreichbar sind, werden als indirekt erreichbare Knoten bezeichnet. Die Abbildung 2.9 zeigt den Unterschied zwischen direkt und indirekt erreichbaren Knoten.

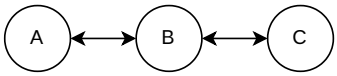

Abbildung 2.9: A und B sind Single-Hop-Nachbarn. A kann über B mit C kommunizieren. C ist damit für A indirekt erreichbar.

Eine Mesh-Topologie besteht, wenn jeder Knoten mit allen Nachbarn spontan kommunizieren oder eine Verbindung herstellen kann. Sind alle Knoten in einem Netzwerk untereinander verbunden, wird es als vollvermascht (Full-Mesh) bezeichnet, wenn das nicht der Fall ist, wird es als teilvermascht bezeichnet. [Aic07] [Hel05]

Abbildung 2.10 zeigt die Stern-Topologie eines Infrastruktur-WLAN im Vergleich zu teil- und vollvermaschten Netzwerken.

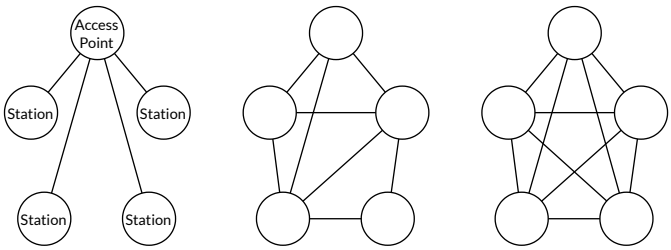

Abbildung 2.10: Netzwerk-Topologien. Links: Stern-Topologie am Beispiel Infrastruktur-WLAN. Mitte: Teilvermaschtes Netzwerk. Rechts: Vollvermaschtes Netzwerk.

In einem Wireless Mesh Network können unterschiedliche Geräteklassen vertreten sein, die sich grob in die drei folgenden Kategorien einteilen lassen. [AWW05]

Router

Ein Router empfängt und sendet Nachrichten, nimmt aktiv am Routing teil und leitet Nachrichten für andere Teilnehmer des Mesh-Netzwerks weiter. Diese Geräte sind auf den Einsatzzweck als Wireless (Mesh) Router zugeschnitten und werden gezielt platziert, um eine optimale räumliche Abdeckung zu erreichen und weisen daher eine geringe Mobilität auf.

Client

Clients senden und empfangen Nachrichten, nehmen optional am Routing teil und leiten optional Nachrichten weiter. Diese Geräte sind nicht speziell für den Einsatz als Mesh Router gebaut und weisen allgemein eine höhere Mobilität auf.

Gateway

Ein Gateway ist ein Router mit Zugang zu einem oder mehreren anderen Netzwerken und kann somit die Schnittstelle zwischen Netzwerken bilden.

Zudem lässt sich die Architektur eines WMN in die drei folgenden Kategorien unterteilen. [AWW05]

Infrastructure WMN

In einem Infrastruktur WMN sind Router und Clients präsent. Zusätzlich können Gateways eine Schnittstelle zu anderen Netzwerken, wie zum Beispiel dem Internet, bereitstellen. Router und Gateways sind untereinander verbunden und bilden den Backbone des WMN. Die Clients sind mit Mesh Routern verbunden und übernehmen keine Routing-Aufgaben.

Client WMN

Ein Client WMN besteht ausschließlich aus Clients. Daher müssen die Clients auch Aufgaben anderer Gerätetypen übernehmen.

Hybrid WMN

Ein hybrides WMN kombiniert Infrastruktur und Client WMN zu einem Netzwerk, in dem die Clients sowohl über Router als auch über andere Clients einen Zugang zum Netzwerk herstellen und kommunizieren können.

Zusätzlich zu der Topologie und Architektur werden WMN durch einige vorteilhafte Eigenschaften ausgezeichnet. [Hel05]

Skalierbarkeit

Knoten können dem Netzwerk spontan beitreten und damit die Reichweite des Netzwerks erweitern.

Zuverlässigkeit

In einem WMN existieren aufgrund der Mesh-Topologie oft mehrere Wege zu einem Ziel. Dadurch wird die Anfälligkeit des Netzwerks gegenüber Ausfällen einzelner Knoten reduziert.

Self-Healing

Fällt ein Knoten eines aktiv genutzten Wegs aus, kann spontan ein anderer Weg zum Ziel gewählt werden.

Self-Configuring

Durch das Routing-Protokoll werden die direkt und indirekt erreichbaren Knoten bekannt gemacht. Neue Knoten machen sich selbst den anderen Knoten bekannt, um Teil des Netzwerks zu werden. Der Aufwand einer Vorkonfiguration soll damit verringert werden.

2.4 Routing in WMN

In einem WMN sind in der Regel nicht alle Ziele direkt erreichbar (Single-Hop). Um dennoch mit einem entfernten Ziel zu kommunizieren, müssen die Daten über andere Teilnehmer des Mesh-Netzwerks weitergeleitet werden, bis sie am Ziel ankommen. Dazu muss dem Absender und/oder den weiterleitenden Knoten ein (Teil-)Pfad zu dem Ziel bekannt sein. Routing-Protokolle ermöglichen die Ermittlung der Wegkosten, um einen möglichst günstigen Weg zum Ziel zu finden. Dazu müssen die Wegkosten der möglichen Wege berechnet werden. Faktoren für die Wegkosten können unter anderem die Anzahl der Hops, Übertragungsdauer, Paketverlust, Datendurchsatz und Lastverteilung sein. Ist der Pfad zu einem Ziel bekannt, wird der Austausch von Informationen mit dem Ziel ermöglicht. [AST12] [SBP08]

Routing-Protokolle für WMN können grob in drei Kategorien eingeteilt werden.

Proaktiv

Proaktive Routing-Protokolle ermitteln kontinuierlich die günstigsten Wege zu den Zielen im Netzwerk. Da zum Zeitpunkt des Datentransfers bereits ein Weg zu dem adressierten Ziel ermittelt ist, entsteht keine zusätzliche Verzögerung vor Beginn des Datentransfers. Die kontinuierliche Ermittlung der besten Route zu den Zielen erfordert allerdings den Austausch von Routing-Informationen, auch wenn Ziele nur selten oder nie adressiert werden. Die Grundlast auf dem Netzwerk ist damit höher als bei den reaktiven Routing-Protokolle. [SBP08] [AWD04]

Reaktiv

Reaktive Routing-Protokolle ermitteln bei Bedarf den günstigsten Weg zu einem Ziel. Dadurch können höhere Latenzen entstehen, da die Ermittlung der Route kurz vor dem Datentransfer abgeschlossen werden muss. Die Routen-Ermittlung findet nur bei Bedarf statt, wodurch – im Gegensatz zu den proaktiven Routing-Protokolle – weniger Nachrichten ausgetauscht werden müssen. [SBP08] [AWD04]

Hybrid

Hybride Routing-Protokolle kombinieren sowohl den proaktiven als auch den reaktiven Ansatz, um Routen zu ermitteln. Für häufig adressierte Ziele können die Routen proaktiv ermittelt werden, während Routen zu selten adressierten Zielen erst bei Bedarf, also reaktiv, ermittelt werden. Bei hybriden Routing-Protokollen werden die Ziele durch den Routingalgorithmus üblicherweise hierarchisch eingeteilt, um unter anderem zwischen proaktiv und reaktiv zu ermittelnden Zielen zu unterscheiden. Daher müssen die Knoten zusätzliche Informationen zur Topologie des Netzwerks vorliegen haben, was zu einem erhöhten Speicherbedarf führen kann. [SBP08] [AWD04] [GSV11]

2.4.1 Routing-Algorithmen

Zusätzlich können die Routing-Protokolle durch die Art des Routing-Algorithmus unterschieden werden. Im Folgenden werden einige grundlegende Routing-Algorithmen erklärt.

Distanzvektor

Beim Routing mit Distanzvektor-Algorithmen bestimmt jeder Knoten zunächst die Wegkosten zu seinen Single-Hop-Nachbarn leitet die Werte an diese weiter. Empfängt ein Knoten diese Informationen von einem Nachbarn, werden die Wegkosten zu dem weiterleitenden Nachbarn auf die empfangenen Wegkosten aufaddiert und die Nachricht anschließend an die Nachbarn weitergeleitet. Mit jedem zusätzlichen Hop werden so die Wegkosten für einen Pfad zusammengefügt. Somit erfährt der Empfänger, welches Ziel über welchen Nachbarn mit welchen Wegkosten erreichbar ist. Der Nachbarn mit den günstigsten Wegkosten zu einem Ziel wird als *nächster Hop* für dieses Ziel vermerkt. Ein Vorteil der Distanzvektor-Algorithmen ist, dass die Router nicht selbst den Weg zum Ziel berechnen müssen, sondern nur den besten Wert aus den erhaltenen Nachrichten in die Routing-Tabelle übernehmen müssen. [AST12]

Link-State

Beim Routing mit Link-State-Algorithmen bestimmt jeder Knoten regelmäßig die Wegkosten zu seinen Single-Hop-Nachbarn und verbreitet sie im Netzwerk. Anhand dieser Informationen kann jeder Knoten ein Abbild der Netzwerk-Topologie mit den ihm bekannten Knoten und Wegkosten erstellen. Mittels eines Wegfindungs-Algorithmus kann der kürzeste Weg zum Ziel auf dem Abbild berechnet werden. Link-State-Algorithmen haben den Nachteil, dass jeder Knoten die Wege zu den Zielen berechnen muss und damit mehr Rechenaufwand auf den Knoten entsteht als bei Distanzvektor-Algorithmen. Zudem kann – durch Verzögerungen oder Ausfälle – das Topologie-Abbild zwischen den Knoten variieren. Bei größeren Änderungen in der Topologie des Netzwerks können Link-State-Algorithmen schneller neue Routen finden als Distanzvektor-Algorithmen, da die Wegkosten der Knoten im Netzwerk direkt ausgetauscht werden. Distanzvektor-Algorithmen weisen eine größere Verzögerung bei diesen Änderungen auf, da sich die Informationen erst über alle Routen verbreiten müssen, bis ein Knoten den günstigsten Weg zu Ziel kennt. [AST12] [SBP08]

Flooding

Flooding ist ein simples Verfahren, bei dem jede Nachricht gleichzeitig an alle Nachbarn weitergeleitet wird, wie die Abbildung 2.11 zeigt. In kabellosen Netzwerken können diese als Broadcast gesendet werden, den alle Geräte in Funkreichweite empfangen. Um Duplikate durch Flooding zu verhindern, können Nachrichten mit einer Sequenznummer ausgestattet werden. Weiterleitende Knoten stellen dann anhand des ursprünglichen Absenders und der Sequenznummer fest, ob sie ein Paket bereits weitergeleitet haben. [AST12]

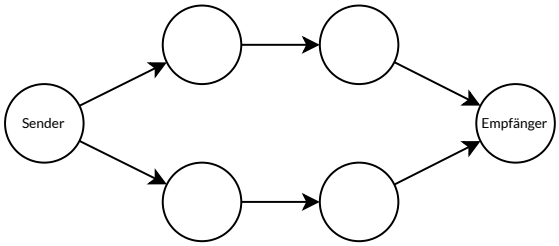

Abbildung 2.11: Message-Flooding

Flooding ist ein robustes Verfahren, da alle Wege des Netzwerks genutzt werden und somit auch Ausfälle mehrerer Knoten tolerierbar sein können, sofern weiterhin ein Weg zum Ziel besteht. Somit könnte eine Nachricht möglicherweise ohne erneutes Senden trotz mehrerer Ausfälle zugestellt werden. Im Gegensatz zu den bereits erwähnten Routing-Verfahren, die für eine Datenübertragung einen möglichst günstigen Weg wählen, wird durch Weiterleitung an alle Knoten eine höhere Last auf dem Netzwerk und den einzelnen Knoten erzeugt. [AST12]

3 Analyse

Dieses Kapitel behandelt die Mesh-Netzwerk-Lösung ESP-WIFI-MESH sowie einige Nachteile dessen und baut damit auf die bereits erwähnten Probleme aus der Aufgabenstellung in Kapitel 1 auf.

Mit ESP-WIFI-MESH stellt der SoC-Entwickler Espressif eine Mesh-Netzwerk-Lösung für die ESP32-Mikrocontroller-Plattform bereit. Zur Kommunikation zwischen den Teilnehmern des Netzwerks kommen WLANs im Infrastruktur-Modus zum Einsatz. [Espf]

3.1 Topologie

Die Knoten eines ESP-WIFI-MESH Netzwerks agieren gleichzeitig als Station und Access Point. Dadurch entsteht eine Baum-Topologie, da eine Station nur mit einem Access Point, ein Access Point aber mit mehreren Stations verbunden sein kann. Daraus folgt, dass ein Knoten mehrere Kindknoten haben kann, sich selbst jedoch nur mit einem weiteren Knoten verbinden kann, wie die Abbildung 3.1 zeigt. [Espb]

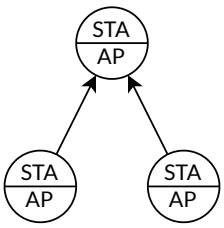

Abbildung 3.1: Jeder Knoten ist gleichzeitig Access Point (AP) und Station (STA). Eine Station kann sich mit einem AP verbinden. Ein Access Point kann mit mehreren Stations verbunden sein. Dadurch entsteht eine Baum-Topologie. Nach [Espb].

Die Knoten eines ESP-WIFI-MESH Netzwerks werden in Layer unterteilt, die der Tiefe des Knotens im Baum entsprechen. Dabei wird die Tiefe ausgehend von Layer 0, der nur den Root-Knoten umfasst, hochgezählt, wie die Abbildung 3.2 zeigt. [Espb]

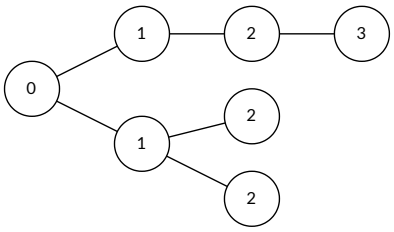

Abbildung 3.2: Ein Beispiel für Layer bzw. Tiefe der Knoten in einem ESP-WIFI-MESH Netzwerk.

Mit der hierarchischen Organisation in einer Baum-Topologie weicht ESP-WIFI-MESH von der für Mesh-Netzwerke typischen Mesh-Topologie ab, weist aber einige der in 2.3 beschriebenen Eigenschaften eines Mesh-Netzwerks auf. So wird durch das Protokoll der spontane Beitritt und die selbstständige Organisation des Mesh-Netzwerks (self-configuring) ermöglicht. Zudem wird auf Ausfälle durch Reorganisation und/oder Wahl eines neuen Weges reagiert (self-healing). [Espb]

Durch die Baum-Topologie entsteht der Nachteil, dass nicht für alle Knoten eine direkte Kommunikation mit in Funkreichweite befindlichen Nachbarn möglich ist, wie die Abbildungen 3.3 und 3.4 anhand eines Beispiels zeigen.

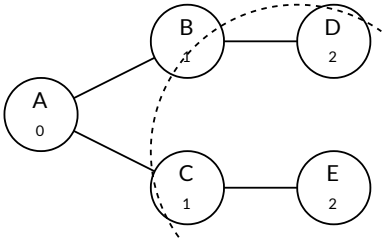

Abbildung 3.3: Der Knoten E ist dem Netzwerk beigetreten. Die gestrichelte Linie stellt die Funkreichweite des Knotens E dar. Aus den Knoten mit geringster Tiefe wählt E den Knoten mit der besten Funkverbindung. Somit sind sowohl B als auch C mögliche Elternknoten für E. Da jedoch eine bessere Funkverbindung zu C besteht, wählt E den Knoten C als seinen Elternknoten.

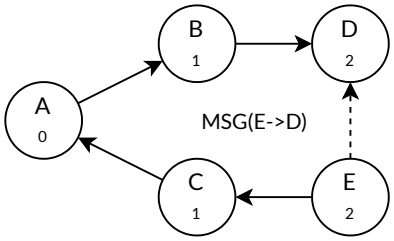

Abbildung 3.4: E möchte eine Nachricht an D senden. D befindet sich in Funkreichweite von E, womit eine direkte Kommunikation möglich wäre. Die Topologie des Netzwerks verhindert die direkte Kommunikation zwischen E und D, daher muss die Nachricht über 3 Knoten weitergeleitet werden.

Das Beispiel zeigt, dass die gewählte Organisation-Methode des Netzwerks zu längeren Pfaden bei der Datenübertragung führen und damit eine erhöhte Latenz erzeugen kann. Zudem entsteht durch die Weiterleitung über mehrere Knoten eine zusätzliche Belastung auf den weiterleitenden Knoten.

Ein Vorteil der Baumtopologie ist, dass der Austausch von Routing-Informationen aufgrund der Hierarchie simpel gestaltet werden kann. Wie in 3.4 erklärt wird, reichen dabei die Knoten in einem ESP-WIFI-MESH Netzwerk die Routing-Informationen zu den bekannten Kindknoten an die Elternknoten weiter.

3.2 Datenübertragung

ESP-WIFI-MESH nutzt zwei unterschiedliche Mechanismen zur Übertragung von Nutzdaten und Netzwerk-Verwaltungsinformationen.

Informationen zur Netzwerkverwaltung werden als zusätzliche hersteller-spezifische Information mit Beacon Frames übertragen. Dies ist ein Subtyp des Control Frame, der zur Bekanntmachung eines Access Point genutzt wird. Mit diesem Verfahren machen sich die Knoten untereinander bekannt, organisieren das Netzwerk und tauschen Informationen zur Wahl eines Root-Knotens aus. Um die Zugehörigkeit zu einem ESP-WIFI-MESH Netzwerk zu identifizieren, übertragen die Knoten eine *Mesh ID* in den Beacon Frames. [Espb]

Zur Datenübertragung in einem ESP-WIFI-MESH Netzwerk agiert jeder Knoten sowohl als Access Point (AP) als auch als Station. Über das Station-Interface verbindet sich ein

Knoten mit einem Elternknoten. Das AP-Interface wird zur Verbindung mit Kindknoten bereitgestellt. Dadurch entsteht zwischen jedem Knoten und seinen Kindknoten ein WLAN im Infrastruktur-Modus, das für die Übertragung von Nutzdaten und Routingtabellen genutzt wird. Nutzdaten aus der Anwendung werden mit einem ESP-WIFI-MESH Header versehen, der unter anderem die Quell- und Zieladresse enthält und anschließend in einen Data Frame eingefügt. Dabei können bis zu 1472 Byte Nutzdaten mit einem Data Frame durch ESP-WIFI-MESH übertragen werden. Optional können die Data Frames verschlüsselt werden. Dazu muss zuvor ein Passwort festgelegt werden und auf allen Knoten übereinstimmen (pre-shared Key). Bei der Unicast-Kommunikation mit ESP-WIFI-MESH werden Data Frames durch alle Knoten entlang des Pfads zum Ziel mit einer Empfangsbestätigung an den jeweils letzten Absender bzw. weiterleitenden Knoten beantwortet. Erhält der Sender bzw. weiterleitende Knoten keine Empfangsbestätigung, wird der Data Frame erneut übertragen. Die Dokumentation enthält keine Hinweise zur Konfiguration von Empfangsbestätigungen. [Espb]

3.3 Organisation

ESP-WIFI-MESH sieht sowohl einen automatischen als auch einen manuellen Modus zur Organisation des Mesh-Netzwerks vor. Der Modus wird vorkonfiguriert und muss auf allen teilnehmenden Geräten übereinstimmen. Zur Organisation des Netzwerks werden Beacon Frames zum Datenaustausch genutzt, wie im vorherigen Abschnitt beschrieben. [Espb]

Im automatischen Modus wird zuerst ein Root-Knoten durch die Teilnehmer des Netzwerks gewählt. Dazu tauschen die Knoten untereinander Signalstärken zu einem manuell festgelegten Access Point aus. Die SSID des AP muss allen Knoten bekannt sein und der AP muss sich in Reichweite für mindestens einen Knoten befinden, damit ein Mesh-Netzwerk entstehen kann. Damit ist die Organisation im automatischen Modus von einem AP außerhalb des Mesh-Netzwerks abhängig. [Espb]

Nach einer festgelegten Zeit sendet jeder Knoten eine Nachricht mit der MAC-Adresse des Knotens, von dem er die höchste Signalstärke übermittelt bekommen hat. Diese Nachricht zählt als die Stimme des Knotens in der Abstimmung. Daraufhin zählt jeder Knoten, wie viele Stimmen er bekommen hat. Liegt das Ergebnis für einen Knoten über einem vorher definierten Limit, wird er zum Root-Knoten, teilt dies den anderen Knoten mit und beendet damit den Aufbau des Netzwerks im automatischen Modus. Was passiert, wenn

die Abstimmung für keinen Knoten über dem Limit liegt, wird in der Dokumentation nicht weiter ausgeführt. [Espb]

Der manuelle Modus setzt voraus, dass alle Knoten eine vorkonfigurierte Rolle erhalten haben oder diese zur Laufzeit eigenständig festlegen. Dabei wird zwischen Root- und Nicht-Root-Knoten entschieden. Wenn mehr als ein Knoten die Root-Rolle übernehmen können soll, muss die Anwendung die Rolle zur Laufzeit dynamisch festlegen, da nicht mehr als ein Root-Knoten gleichzeitig existieren kann. Während im automatischen Modus ein Access Point zur Wahl eines Root-Knotens mittels der Signalstärke benötigt wird, entfällt diese Voraussetzung im manuellen Modus. [Espb]

Tritt ein Knoten einem bereits existierenden Netzwerk im automatischen oder manuellen Modus bei, wartet dieser zunächst auf Beacon Frames aus dem Netzwerk. Anhand der erhaltenen Informationen wird nun der Knoten mit niedrigstem Layer (Tiefe im Baum) und höchster Signalstärke als Elternknoten ausgewählt und eine Verbindung hergestellt. Indem ein Elternknoten mit niedrigem Layer gewählt wird, soll die Anzahl der Hops zum Root-Knoten gering gehalten werden. Nach dem Start wird dieses Verfahren laufend durchgeführt, um auf Ausfälle zu reagieren oder neue, kürzere Wege zu nutzen, wie die Abbildung 3.5 zeigt. [Espb]

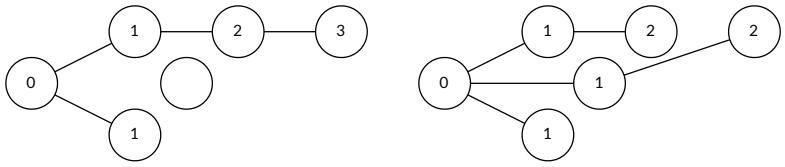

Abbildung 3.5: Ein neuer Knoten tritt dem Netzwerk bei und wird gleichzeitig wegen des niedrigeren Layers zum Elternknoten für einen anderen Knoten im Netzwerk.

Da jeder Knoten, außer dem Root-Knoten, von funktionierende Elternknoten abhängig ist, um Nachrichten zu senden/empfangen, führt der Ausfall eines Elternknotens auch zu einem kurzfristigen Ausfall weiterer Knoten im Netzwerk. Dabei können die betroffenen Knoten keine Nachrichten senden/empfangen, bis die Verbindung zu einem anderen Elternknoten hergestellt wird. Ist ein Knoten mit geringer Tiefe betroffen, fällt möglicherweise kurzfristig ein großer Teil des Netzwerks aus. Fällt der Root-Knoten aus, muss das gesamte Netzwerk durch die Wahl eines neuen Root-Knotens aufgebaut werden. Damit stellt der Root-Knoten einen Single Point of Failure dar.

3.4 Routing

Um Nutzdaten über mehrere Hops zum Ziel zuzustellen, tauschen die Knoten regelmäßig Routing-Informationen aus [Espb]. Aufgrund des regelmäßigen Austauschs dieser Informationen ist das Routing-Protokoll bei ESP-WIFI-MESH als proaktiv einzuordnen.

Jeder Knoten verwaltet eine Routingtabellen, die alle direkten Kindknoten beinhaltet. Regelmäßig werden diese an den Elternknoten weitergeleitet, der sie wiederum in der Baumstruktur nach oben weiterleitet, bis sie beim Wurzelknoten angekommen ist. Dadurch erfährt ein Knoten über weitere Knoten unterhalb der eigenen Kindknoten und über welche Kindknoten diese erreichbar sind. Die folgende Abbildung 3.6 veranschaulicht den beschriebenen Austausch der Routingtabellen in mehreren Schritten. [Espb]

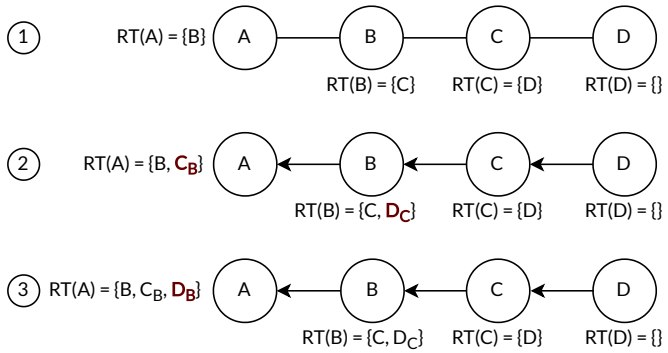

Abbildung 3.6: Im ersten Schritt sind die direkten Kindknoten bereits in die Routingtabellen (RT) der Knoten eingetragen. Die Schritte 2 und 3 zeigen den Austausch, bis die Routingtabelle des Root-Knoten A vollständig ist. Die Unterschiede zwischen den Schritten sind farblich hervorgehoben.

Empfängt ein Knoten eine Nachricht, dessen Ziel-Adresse in seiner Routingtabelle enthalten ist, leitet er die Nachricht direkt an das Ziel oder an den Kindknoten in Richtung des Ziels weiter. Ist das Ziel nicht bekannt, wird die Nachricht über den Elternknoten weitergeleitet, bis sie einen Knoten erreicht, dem das Ziel bekannt ist, um die Nachricht in Richtung des Ziels weiterzuleiten. [Espb]

Ein Nachteil dieses Routing-Ansatzes ist, dass einem Knoten nur die Knoten seines Teilbaums sowie sein Elternknoten bekannt ist, da die Routing-Tabellen im Baum nur nach oben weitergeleitet werden. Daraus folgt, dass Nachrichten im Worst Case bis zum Root-Knoten weitergeleitet werden müssen, da nur dieser alle Knoten im Netzwerk kennt.

Sollte der Root-Knoten feststellen, dass das Ziel nicht bekannt ist, wird der Absender der Nachricht nicht darüber informiert.

Vorteilhaft für diese Verfahren ist, dass der Speicherbedarf für Routing-Informationen von der Größe des Teilbaums abhängig ist und damit nicht alle Knoten Speicher für die Routing-Informationen zu allen anderen Knoten reservieren müssen.

3.5 Aufgabenstellung

Ziel der Bachelorarbeit ist die Entwicklung eines Mesh-Netzwerks für die ESP32-Plattform als Alternative zu ESP-WIFI-MESH.

Um kürzere Wege bei der Übertragung von Nachrichten im Mesh-Netzwerk zu ermöglichen, sollen die Teilnehmer mit allen direkten Nachbarn spontan kommunizieren können. Durch kürzere Wege, gemessen an der Anzahl der Hops, soll die Auslastung durch Weiterleitungen einer Nachricht im Netzwerk reduziert werden sowie die Übertragungsdauer von Absender bis Ziel reduziert werden. Zudem sollen durch die spontane Kommunikation mit allen direkten Nachbarn mehrere Pfade zu einem Ziel im Netzwerk gefunden werden, um die Widerstandsfähigkeit des Netzwerks gegen Ausfälle zu erhöhen und damit die Dauer einer Unterbrechung zu verringern. Zusätzlich soll eine dezentrale Organisation des Netzwerks ohne Root-Knoten zu einer höheren Ausfallsicherheit führen, da ein Root-Knoten als Single Point of Failure entfällt.

Zur Umsetzung dieser Anforderungen soll eine teil- oder vollvermaschte Topologie im Mesh-Netzwerk genutzt werden. Dazu muss eine geeignete Kommunikationstechnologie ausgewählt werden, die eine spontane Kommunikation mit allen direkten Nachbarn eines Knotens ermöglicht.

Außerdem ist ein geeignetes Routing-Protokoll auszuwählen, das die Pfade zu den Zielen ermittelt und diese Informationen austauscht, sodass den Knoten alle erreichbaren Ziele bekannt sind. Somit soll einem Knoten bereits vor dem Senden einer Nachricht bekannt sein, ob ein Ziel erreichbar ist.

4 Verwandte Arbeiten

Im Zuge der Recherche wurden drei verwandte Arbeiten gefunden, die ESP-NOW zur Kommunikation in Multi-Hop- oder Mesh-Netzwerken einsetzen.

A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation [KS19] beschäftigt sich mit der Kommunikationstechnologie ESP-NOW für den Datenaustausch in drahtlosen Sensornetzwerken. Dabei werden zuerst Bluetooth Low Energy (BLE) Beacons zum Auffinden weiterer Knoten genutzt und anschließend ESP-NOW zur Übertragung von Nutzdaten innerhalb des Netzwerks genutzt. Das Netzwerk wurde in Versuchen hinsichtlich Datenrate und Latenz über einen und mehrere Hops untersucht. Dabei wurde jedoch nur ein kleines Netzwerk mit 6 Knoten in einer Linientopologie betrachtet. Aus den Ergebnissen wurde ein positives Fazit bezüglich der Nutzung von ESP-NOW in drahtlosen Sensornetzwerken gezogen. Weitere Untersuchungen, wie die Ergänzung durch Mesh-Routing, wird angeregt.

In *An efficient networking solution for extending and controlling wireless sensor networks using low-energy technologies* [LEGA21] wird ein drahtloses Sensornetzwerk mit ESP-NOW in einer Spider-Mesh-Topologie umgesetzt, wie die Abbildung 4.1 zeigt.

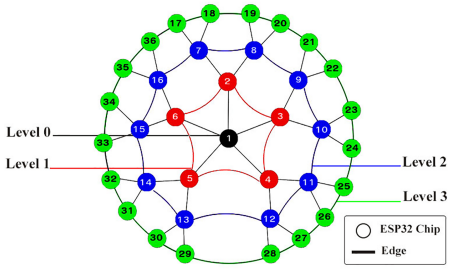

Abbildung 4.1: Die Spider-Mesh-Topologie des Netzwerks. [LEGA21]

Das Routing-Verfahren ist nicht beschrieben. Das drahtlose Sensornetzwerk wurde hinsichtlich Stromverbrauch und Reichweite der Kommunikationstechnologie in Versuchen

ausgewertet. Die Ergebnisse zeigen, dass auf der ESP32-Plattform bei der Kommunikation mit ESP-NOW im Innenbereich in etwa die doppelte Reichweite im Vergleich zu BLE erreicht werden konnte, dabei aber der Stromverbrauch um ca. 60% höher lag. Die Relevanz der Ergebnisse, bezogen auf diese Arbeit, kann nicht eingeschätzt werden, da genaue Angaben zur Durchführung der Versuche fehlen.

Ähnlich zu ESP-WIFI-MESH wird in *Dynamic Mesh Network Implemented in Micropython on Top of ESP-NOW Protocol* [Jin22] die Umsetzung eines Mesh-Netzwerks mit WLAN im Infrastruktur-Modus beschrieben. Dabei kommt zusätzlich ESP-NOW zum Austausch von Netzwerkinformationen und zur Wahl eines Root-Knotens zum Einsatz. Die Knoten stellen anhand dieser Informationen WLAN-Verbindungen zu anderen Knoten her und bilden damit das Mesh-Netzwerk. Dabei entsteht eine Baum-Topologie wie bei ESP-WIFI-MESH. Im Gegensatz dazu soll das Mesh-Netzwerk aber auch ohne Access Point automatisch aufgebaut werden können. Die Ergebnisse der Arbeit zeigen, dass ein Betrieb mit mehr als 6 Knoten aufgrund von Speicher- und Stabilitätsproblemen bei der entwickelten Software nicht erzielt werden konnte. Zeitliche Parameter wie die Dauer des Netzwerk-Aufbaus und Single-Hop-Latenz wurden gegen ESP-WIFI-MESH verglichen. Dabei handelt es sich jedoch – bedingt durch die instabile Umsetzung – um simulierte Werte. Die Relevanz dieser Vergleichswerte, bezogen auf diese Arbeit, kann daher nicht eingeschätzt werden.

5 Konzeption

Dieses Kapitel beschreibt die Konzeption des zu entwickelnden Mesh-Netzwerks. Dazu wurde zuerst eine Methode zur Kommunikation sowie ein Routing-Protokoll für Wireless Mesh Networks (WMNs) gewählt. Im Anschluss wird das Protokoll Design erklärt.

5.1 Kommunikation

Die direkte Kommunikation der Knoten mit allen in Funkreichweite befindlichen Nachbarn soll im Vergleich zu der Baumtopologie bei ESP-WIFI-MESH kürzere Wege zu den Zielen und den spontanen Wechsel des Wegs zu einem Ziel ermöglichen. Durch die kürzeren Wege soll die Round Trip Time (RTT) bei der Kommunikation reduziert werden und damit die Verzögerung beim Einsatz in einer Smart-Home-Anwendung reduziert werden. Zudem soll die Verzögerung durch Ausfall eines Knotens verringert werden, indem den Teilnehmern mehrere mögliche Wege zu einem Ziel bekannt gemacht werden. Um die spontane Kommunikation mit allen in Funkreichweite befindlichen Nachbarn zu ermöglichen, soll eine teil- oder vollvermaschte Topologie zum Einsatz kommen. Um diese Anforderungen umzusetzen, muss eine geeignete Kommunikationstechnologie gewählt werden.

Um eine größtmögliche Kompatibilität auf der ESP32-Plattform zu erreichen, wird WLAN zur Kommunikation eingesetzt, da dies die aktuell einzige Funktechnologie ist, die von allen Varianten der ESP32-Mikrocontroller unterstützt wird [Espi].

Ein WLAN im Ad-hoc-Modus, beschrieben in Kapitel 2.1.1, wäre eine geeignete Methode zur spontanen Vernetzung der Teilnehmer. Aktuell wird durch das Entwicklungsframework für die ESP32-Plattform (ESP-IDF) nur der Betrieb eines WLAN im Infrastruktur-Modus, nicht jedoch im Ad-hoc-Modus unterstützt [Espc]. Allerdings bietet der Hersteller für die Kommunikation via WLAN das bereits in Kapitel 2.2 erwähnte *ESP-NOW*. Bis

zur Größenbegrenzung der Peerlist erlaubt ESP-NOW die spontane Kommunikation mit – möglicherweise wechselnden – Nachbarn per Uni- und Broadcast.

Aufgrund der Kompatibilität zu allen Varianten der ESP32-Mikrocontroller und der Möglichkeit des spontanen Austauschs mit Nachbarn, wird ESP-NOW zur Kommunikation im Mesh-Netzwerk genutzt.

5.2 Mesh-Routing

Wie bereits in Kapitel 2.4 erwähnt, lassen sich Routing-Protokolle für WMN in die drei Kategorien proaktiv, reaktiv und hybrid einteilen.

Reaktive Routing-Protokolle ermittelt die Route zu einem Ziel erst bei Bedarf, was in einer erhöhten Latenz bei der Übertragung von Nutzdaten resultieren kann. Im Gegensatz zu den proaktiven Routing-Protokollen findet kein kontinuierlicher Austausch von Routing-Informationen statt, weshalb das Nachrichtenaufkommen vergleichsweise geringer ist.

Im Gegensatz zu den reaktiven und proaktiven Routing-Protokollen werden die Teilnehmer eines Netzwerks bei hybriden Routing-Protokollen oft hierarchisch eingeordnet. Daher müssen die Teilnehmer Kenntnis über die Topologie des Netzwerks haben, was sich nachteilig auf den Speicherbedarf auswirken kann.

In einer Smart-Home Anwendung kann eine geringere Latenz von Vorteil sein, um die Reaktionszeit des Systems – beispielsweise zwischen dem Auslösen einer Aktion durch die Benutzenden und der Reaktion eines entfernten Geräts – zu verkürzen. Daher fällt die Wahl auf ein proaktives Routing-Protokoll für das Mesh-Netzwerk, da diese den Vorteil haben, dass eine Route zum Zeitpunkt der Übermittlung bereits bekannt ist und deshalb eine geringere Latenz als die reaktiven Routing-Protokolle aufweisen. Dabei kann der kontinuierliche Austausch von Routing-Informationen allerdings zu einem höheren Nachrichtenaufkommen führen.

Wie bereits in Kapitel 2.4 erwähnt, gibt es einige grundlegende Routing-Algorithmen. Dabei ist Flooding ein simples und zugleich robustes Verfahren. Beim Flooding wird jede Nachricht mit Nutzdaten durch jeden Teilnehmer des Netzwerks an alle seine Nachbarn weitergeleitet, bis die Nachricht ihr Ziel erreicht. Um Wiederholungen zu vermeiden, kann dabei eine Sequenznummer eingesetzt werden. Da die Nachricht über alle Pfade

des Netzwerks geleitet wird, können Ausfälle eines oder mehrerer Knoten kompensiert werden, solange noch ein Weg zum Ziel besteht. Das hohe Nachrichtenaufkommen, das durch die Weiterleitung der Nachrichten über alle Knoten entsteht, ist dabei allerdings von Nachteil.

Beim Routing mit Link-State-Algorithmen tauschen die Knoten Informationen zu ihren Nachbarn untereinander aus. Anhand dieser Informationen können die Teilnehmer ein topologisches Abbild des Netzwerks erzeugen und den günstigsten Weg zu einem Ziel durch einen Wegfindungs-Algorithmus berechnen. Durch die Berechnung der Wege benötigen Link-State-Algorithmen mehr Rechenzeit als Distanzvektor-Algorithmen. Insbesondere auf Mikrocontrollern wie dem ESP32 könnte die Berechnung aufgrund der geringen Rechenleistung mehr Rechenzeit in Anspruch nehmen als bei einem Distanzvektor-Algorithmus.

Bei den Distanzvektor-Algorithmen tragen alle Knoten eines Pfads gleichermaßen zur Bestimmung der gesamten Pfadkosten bei, indem die lokal ermittelten Wegkosten aller einzelnen Knoten auf dem Weg zu Ziel durch die Übertragung von Routing-Nachrichten summiert werden. Damit weisen Distanzvektor-Algorithmen eine geringere Rechenzeit als Link-State-Algorithmen auf und sollten sich damit besser für die ESP32-Mikrocontroller eignen. Zudem muss kein Abbild der Topologie angelegt werden, da lediglich die Wegkosten für jedes Ziel über jeden Nachbarn bekannt sind, weshalb der Speicherbedarf geringer als bei Link-State-Algorithmen ausfallen kann. Daher fällt die Wahl auf ein Routing-Protokoll, dass einen Distanzvektor-Algorithmus einsetzt oder ein ähnliches Verhalten zeigt.

5.3 BATMAN

Better Approach To Mobile Ad Hoc Networking (BATMAN) ist ein proaktives Routing-Protokoll für Mesh-Netzwerke, das von der Freifunk-Community seit 2006 zur Ablösung des Optimized Link State Routing Protocol (OSLR) im Freifunk-Netzwerk entwickelt wird. [Aic07]

BATMAN verfolgt einen dezentralen Ansatz für die Speicherung der Routing-Informationen, wobei die BATMAN-kompatiblen Knoten (Originator) jeweils nur den nächstbesten Nachbarn in Richtung eines Ziels kennen [Aic07]. Damit weist das Protokoll eine Ähnlichkeit zu den Distanzvektor-Algorithmen auf.

Um die Wegkosten zu ermitteln, setzt BATMAN ein simples Verfahren ein, bei dem die Originator regelmäßig Nachrichten mit einer Sequenznummer und ihrer Adresse an alle erreichbaren Nachbarn senden. Durch die Anzahl empfangener Nachrichten in einem vorher festgelegten Zeitfenster können die Originator eine Qualitätsmetrik für Nachbarn oder entfernte Ziele ermitteln. Dieses Verfahren wurde mit den Versionen I bis IV weiter verbessert und schließlich durch die Messung des Datendurchsatzes mit der Version V abgelöst. [NAL07] [NALW08] [Opea]

Da es sich bei BATMAN um ein proaktives Routing-Protokoll handelt, erfüllt es die Anforderung, die Reaktionszeit bei der Übertragung von Nutzdaten gering zu halten. Zudem nutzt BATMAN ein Routing-Verfahren, das den Distanzvektor-Algorithmen ähnelt und damit zu den Anforderungen an Speicherbudget und Rechenleistung der ESP32-Mikrocontroller aus 5.2 passt.

In den folgenden Abschnitten werden die Ergebnisse einiger vergleichender Arbeiten zusammengefasst, die BATMAN und weitere Routing-Protokolle in simulierten oder realen WMN untersucht haben.

In *Comparative analysis of routing protocols for wireless mesh networks* [PZOW16] wird BATMAN IV gegen die reaktiven Routing-Protokolle AODV, DYMO, das hybride Routing-Protokolle HWMP sowie das proaktive Routing-Protokoll OSLR verglichen. Dabei konnte in einer WMN-Simulation mit BATMAN ein vergleichsweise geringer Paketverlust sowie eine geringere Verzögerung beim Senden und Weiterleiten von Paketen festgestellt werden.

In *Towards benchmarking routing protocols in wireless mesh networks* [FdRG11] wird BATMAN IV gegen die proaktiven Routing-Protokolle AODV und BABEL sowie das reaktive Routing-Protokoll AODV verglichen. Dabei weist BATMAN den geringsten Paketverlust auf und zeigt gemeinsam mit BABEL die geringste Round Trip Time (RTT).

In *Real-world performance of current proactive multi-hop mesh protocols* [AHW09] wird BATMAN IV gegen die proaktiven Routing-Protokolle OSLR und BABEL verglichen. Dabei weist BATMAN den geringsten Paketverlust auf. Die RTT von BATMAN ist mit BABEL vergleichbar, im Vergleich zu OSLR aber erheblich geringer.

In *Characterization of Wireless Mesh Network performance in an experimental test bed* [JSAT15] wird BATMAN IV gegen die proaktiven Routing-Protokolle OSLR und BABEL verglichen und weist dabei die geringste RTT sowie einen vergleichbar geringen Paketverlust auf.

Aufgrund der erwähnten Übereinstimmung mit den Anforderungen sowie den vorteilhaften Ergebnissen aus den aufgelisteten Arbeiten zu Routing-Protokollen in WMN, fällt die Wahl auf das Routing-Protokoll BATMAN.

5.3.1 Originator Message (OGM)

Jeder BATMAN-kompatible Knoten in einem Netzwerk wird als Originator bezeichnet. Diese tauschen regelmäßig Routing-Informationen über Originator Messages (OGMs) aus, die per Flooding im Netzwerk verbreitet werden. Zentrale Informationen der OGM sind die Originator-Adresse, die Sequenznummer und die Übertragungsqualität. [NALW08]

Die Versionen I-IV des BATMAN-Protokolls setzen auf die Vermittlungsschicht (Layer 3 im OSI-Referenzmodell) auf und übertragen OGMs per UDP-Broadcast in einem IPv4-Netzwerk. Daher werden die Originator in einer OGM anhand ihrer IPv4-Adresse adressiert. Ab Version V setzt das Protokoll auf die Sicherungsschicht (Layer 2 im OSI-Referenzmodell) auf, um zusätzlichen Overhead durch IP- und UDP-Pakete zu reduzieren und unabhängig von den verwendeten Protokollen in der Vermittlungsschicht zu sein. Die OGM für Version 5 (OGMv2) setzt daher MAC-Adressen zur Adressierung der Originator ein. [NALW08] [Opea] [WL]

Originator mit Zugang zu einem weiteren Netzwerk können als Gateways agieren und dies über die Gateway-Felder der OGM bekannt geben. Auf eine OGM können mehrere Host Network Announcements (HNA) folgen, die Details zu den angebundenen Netzwerken des Gateways enthalten. [NALW08]

Im folgenden Abschnitt werden die Felder der OGM beschrieben. Die Abbildung 5.1 zeigt die Felder einer BATMAN IV OGM.

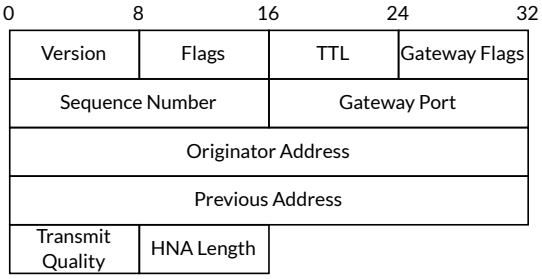

Abbildung 5.1: BATMAN IV OGM, nach [Opea].

Version

Dieses Feld enthält die Versionsnummer des Protokolls. [NALW08]

Flags

In diesem Feld sind zwei Flags enthalten. Das *Is-direct-link* Flag wurde bis Version III genutzt und ist mit Version IV nicht mehr relevant. Wenn eine OGM über ein unidirektionales Netzwerkinterface gesendet wird, muss das *Unidirectional*-Flag gesetzt werden. Die Flags sind jeweils ein Bit groß. Die restlichen sechs Bits sind ungenutzt. [NALW08] [Opea]

Time To Live (TTL)

Mit dem *TTL*-Feld kann die Ausbreitung einer OGM im Netzwerk begrenzt werden. Bei jeder Weiterleitung wird der Wert um eins verringert. Ist die TTL null, wird die OGM nicht mehr weitergeleitet. [NALW08]

Gateway Flags

In den *Gateway Flags* sind drei Felder enthalten. Das erste Feld mit der Länge von einem Bit zeigt an, ob der Originator als Gateway agiert. Darauf folgt ein 4-Bit Feld, dass die geschätzte Downlink-Geschwindigkeit des Gateways zu dem angebundenen Netzwerk angibt. Die letzten 3 Bit sind für die geschätzte Uplink-Geschwindigkeit reserviert. [NALW08]

Sequence Number

Die Sequenznummer des Originators startet bei Null und wird nach jeder gesendeten OGM um eins erhöht. [NALW08]

Gateway Port

Ist der Originator der OGM ein Gateway, kann dieser einen Port für einen UDP-

Tunnel bekannt geben, um den anderen Teilnehmern des Netzwerks Zugang zu dem angebundenen Netzwerk des Gateways bereitzustellen. [NALW08] [NAL07]

Originator Address

Die *Originator Address* ist die IPv4-Adresse des Netzwerkinterfaces, das von dem Originator der OGM für BATMAN genutzt wird. [NALW08]

Previous Sender Address

In dieses Feld trägt ein Knoten vor der Weiterleitung die IPv4-Adresse des Nachbarn ein, der die OGM an ihn weitergeleitet hat. Damit werden Wiederholungen der gleichen Nachricht verhindert, wie in dem folgenden Abschnitt zu BATMAN IV beschrieben wird. [Opea]

Transmit Quality

Dieses Feld wird zur Ermittlung der Übertragungsqualität eines Multi-Hop-Pfads genutzt, wie in dem folgenden Abschnitt zu BATMAN IV näher beschrieben wird. [Opea]

5.3.2 BATMAN IV

Um den günstigsten Weg zu einem Ziel im Netzwerk zu bestimmen, muss zuerst eine Qualitätsmetrik für die möglichen Pfade ermittelt werden. Dabei betrachtet BATMAN IV die Übertragungsqualität in Richtung des Ziels, die als Transmit Quality (TQ) bezeichnet wird. Zusätzlich unterscheidet BATMAN IV zwischen Local TQ und Global TQ. Die Local TQ beschreibt die Sendequalität zu den Nachbarn eines Originators. Die Global TQ beschreibt die Sendequalität zu einem entfernten Ziel (Multi-Hop-Ziel) und setzt sich aus den Local TQs der Knoten entlang eines Pfads zusammen. [Opea]

Local TQ

Um die *Local TQ* zu einem Nachbarn zu bestimmen, muss zuerst die Receive- und Echo-Quality zu diesem Nachbarn gemessen werden. Die *Receive Quality (RQ)* entspricht dabei der Anzahl empfangener OGM eines Nachbarn, wie die folgende Abbildung 5.2 zeigt. [Opea]

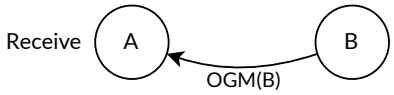

Abbildung 5.2: Receive Quality: Durch das Zählen der erhaltenen OGM, kann Originator A die Receive Quality zu dem Nachbarn B ermitteln. Nach [Opea].

Als *Echo Quality (EQ)* wird das Verhältnis zwischen gesendeten OGM und von den Nachbarn weitergeleiteten eigenen OGM bezeichnet, wie die Abbildung 5.3 zeigt. [Opea]

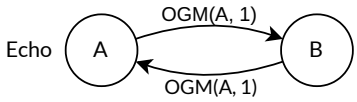

Abbildung 5.3: Echo Quality: Der Originator A zählt, wie viele seiner eigenen OGM von dem Nachbarn B weitergeleitet werden. Nach [Opea].

Zur Datenübertragung ist jedoch die Transmit Quality für die Wahl des Pfads entscheidend, da sie die Übertragungsqualität von Originator bis Ziel abbildet, wie die Abbildung 5.4 zeigt. [Opea]

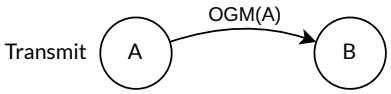

Abbildung 5.4: Transmit Quality: Aus Sicht von Originator A stellt die Übertragungsqualität beim Senden zu dem Nachbarn B die Transmit Quality zu diesem Nachbarn dar. Nach [Opea].

Aus der Receive- und Echo-Quality (RQ und EQ) kann die Transmit Quality (TQ) mit folgender Formel berechnet werden. [Opea]

$$EQ = RQ \cdot TQ \qquad \Rightarrow \quad TQ = \frac{EQ}{RQ}$$

Um die Aktualität der gemessenen Receive- und Echo-Quality zu beschränken, werden die erhaltenen Sequenznummern in ein Sliding Window eingetragen [NALW08] [Opea]. Die folgende Abbildung 5.5 zeigt ein Beispiel zur Berechnung der TQ.

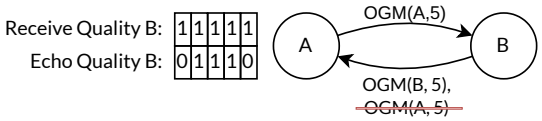

Abbildung 5.5: Berechnung der Local TQ aus Sicht von Knoten A.
Receive Quality: A hat von B 5 OGM erhalten.
Echo Quality: Von den eigenen OGM sind über B jedoch nur 3 der letzten 5 zurückgekommen.
Damit kann A die Transmit Quality von $A \rightarrow B$ berechnen.
$TQ_{A \rightarrow B} = \frac{EQ_{A \rightarrow B}}{RQ_B} = \frac{3}{5} = 0,6$

Global TQ

Die Global TQ setzt sich aus den einzelnen Local TQs entlang eines Pfads zusammen. Dazu wird das TQ-Feld der OGM durch den Originator mit dem Maximalwert initialisiert. [Opea]

Wenn eine OGM empfangen wird, muss die erhaltene TQ mit der lokalen TQ zu dem Nachbarn, der die OGM weitergeleitet hat, multipliziert werden. Vor der Weiterleitung wird die berechnete TQ in das entsprechende Feld der OGM eingetragen. [Opea]

Somit entsteht entlang des Pfades eine TQ, wie die Abbildung 5.6 zeigt.

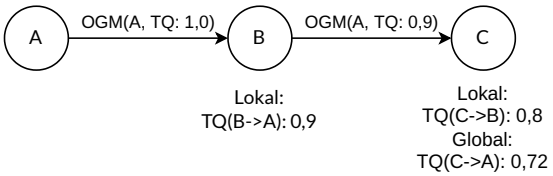

Abbildung 5.6: Die lokale TQ von B zu A und von C zu B ist bereits bekannt. A sendet eine OGM, die entlang des Pfads mit den lokalen TQ multipliziert wird. C empfängt damit die Global TQ für die Strecke $C \rightarrow A$.
$TQ_{C \rightarrow A} = TQ_{B \rightarrow A} \cdot TQ_{C \rightarrow B} = 0,9 \cdot 0,8 = 0,72$

Da die OGM per Flooding im Netzwerk verbreitetet werden, muss die Weiterleitung einer OGM mit Regeln beschränkt werden, um Wiederholungen der gleichen Nachricht zu

vermeiden. Daher werden nur OGM mit einer unbekannten Kombination aus Sequenznummer, Originator und weiterleitendem Nachbar von den Originators verarbeitet. Dabei können jedoch Echos entstehen, wie die Abbildung 5.7 zeigt. [Opea]

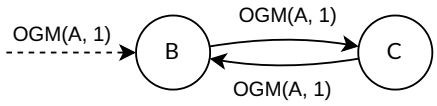

Abbildung 5.7: Entstehung eines Echos: B empfängt eine OGM von A und leitet sie an C weiter. C leitet sie ebenfalls weiter. B empfängt die weitergeleitete OGM von C und verarbeitet sie, um sie weiterzuleiten.

Um diese Wiederholungen zu verhindern, merkt sich ein Knoten beim Empfang einer OGM die Adresse des Knotens, der die OGM an ihn weitergeleitet hat, und trägt diese Adresse in das *Previous Sender* Feld vor der nächsten Weiterleitung ein. Empfängt ein Originator eine OGM mit seiner Adresse im Previous Sender Feld, wird sie verworfen und damit nicht weitergeleitet. Das in der Abbildung 5.7 dargestellte Szenario wird somit vermieden. [Opea]

5.3.3 BATMAN V

BATMAN V führt ein neues Verfahren zur Ermittlung der Übertragungsqualität ein. Dabei bleibt die Unterteilung in Local TQ und Global TQ bestehen. Als Metrik für die Local TQ wird die geschätzte Übertragungsgeschwindigkeit genutzt. Die Referenzimplementierung nimmt dazu Schätzungen der Übertragungsgeschwindigkeit zu den Nachbarn aus dem Treiber für das WLAN-Netzwerkinterface. Steht diese Schätzung nicht bereit, kann alternativ eine Messung des Datendurchsatzes erfolgen, um den Wert zu ermitteln. Wie bei Version IV wird auch mit BATMAN V die Transmit Quality eines Pfads durch die Verbreitung von OGM und das Zusammenfügen der einzelnen Local TQs ermittelt. [Opeb]

Zudem führt BATMAN V ein neues Nachrichtenformat zusätzlich zur OGM ein. Das Echo Location Packet (ELP) ähnelt der OGM, wird jedoch nur für die Neighbor Discovery eingesetzt und nicht über die Nachbarn des Originators hinaus weitergeleitet. Damit sollen Änderungen der lokalen Umgebung schneller festgestellt werden. [Opeb]

Während bisherige Versionen auf die Vermittlungsschicht aufsetzen, nutzt BATMAN V die Sicherungsschicht um den Protokoll-Overhead bei OGM und ELP zu reduzieren

und unabhängig von den verwendeten Protokollen in der Vermittlungsschicht zu sein. [Opeb]

Da die ESP32-Plattform keine Schätzung der Übertragungsgeschwindigkeit zu anderen WLAN-Geräten bereitstellt [Espc], müsste diese regelmäßig gemessen werden und würde damit zu einem erhöhten Nachrichtenaufkommen sowie höherer Auslastung der beteiligten Knoten führen. Die mit Version V eingeführte ELP soll Änderungen in der Verfügbarkeit von Nachbarn eines Originators schneller feststellen. Es wird empfohlen, zwei ELP pro OGM-Intervall zu senden [Opeb], was in einem höheren Nachrichtenaufkommen resultiert.

Aufgrund des höheren Nachrichtenaufkommens der Version V wird die Version IV für den Einsatz in dem Mesh-Netzwerk gewählt.

5.4 Protokoll Design

Für das Mesh-Netzwerk werden Nachrichtentypen zur Übertragung von Routing-Informationen sowie zum Austausch von Nutzdaten benötigt. Um den Typ der Nachricht sowie die Version des Mesh-Protokolls zu unterscheiden, wird ein gemeinsamer Header genutzt, wie die Abbildung 5.8 zeigt. In der Implementierung wird der gemeinsame Header als *Common Header* bezeichnet.

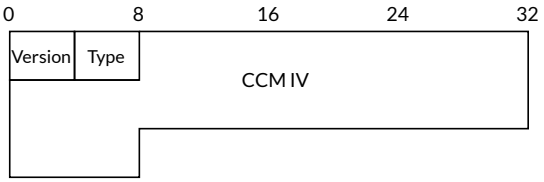

Abbildung 5.8: Gemeinsamer Header. Die Felder Version und Type sind jeweils 4 Bit groß und können damit bis zu 16 verschiedene Nachrichtentypen bzw. Versionen abbilden. Zudem ist der IV für die CCM-Verschlüsselung enthalten.

Da Broadcast-Nachricht bei ESP-NOW nicht verschlüsselt werden können und ESP-NOW zudem eine Schwachstelle bei der Verschlüsselung von Nachrichten aufweist, wie in dem folgenden Kapitel 6.2.1 detailliert erklärt wird, ist die Verschlüsselung beim Versand mit ESP-NOW nicht aktiviert. Daher wird als Workaround das von ESP-NOW verwendete Verschlüsselungsverfahren (CCM: Counter with CBC-MAC) auf die Nachrichten des Mesh-Netzwerks angewandt. Dazu wird ein Pre-Shared-Key genutzt und für jede Nachricht ein zufälliger Initialisierungsvektor (IV) generiert, um eine Wiederverwendung der Kombination aus Schlüssel und IV zu verhindern. Bei einer Länge des IV von 8 Bytes liegt die Wahrscheinlichkeit der Wiederverwendung eines IV bei 2^{64}. Weitere Details zu dem Verschlüsselungsverfahren werden in dem Kapitel 6.2.1 behandelt.

Das in Kapitel 6.2.1 beschriebene Problem mit Replay-Angriffen wird durch diesen Workaround allerdings nicht verhindert. Deshalb ist weitere Arbeit zur Verhinderung dieser Schwachstelle notwendig.

Die in den folgenden Abschnitten beschriebenen Nachrichten entsprechen der Version 0. Empfängt ein Teilnehmer eine Nachricht mit einer Versionsnummer, die nicht der eigenen entspricht, wird die Nachricht verworfen. Entspricht der Nachrichtentyp im gemeinsamen Header nicht einem der in diesem Unterkapitel beschriebenen Nachrichtentypen, wird die Nachricht verworfen.

5.4.1 Routing-Informationen

Für den Austausch von Routing-Informationen wird die BATMAN IV OGM in einer angepassten Variante verwendet. Diese entspricht dem Nachrichtentypen 0. Die folgende Abbildung 5.1 zeigt die unmodifizierte BATMAN IV OGM.

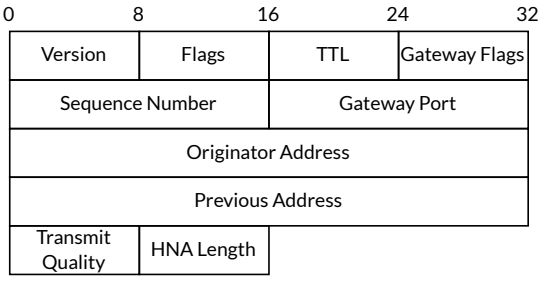

Abbildung 5.9: BATMAN IV OGM mit einer Länge von 18 Bytes, nach [Opea].

Zur Adressierung werden bei ESP-NOW MAC-Adressen genutzt, weshalb die Adressfelder der BATMAN IV OGM von 4 Byte für IPv4-Adressen auf 6 Byte für MAC-Adressen angepasst werden.

Da es sich um ein Mesh-Netzwerk für Smart-Home-Anwendungen handelt, in denen ein Smart-Home-Gateway eingesetzt werden kann, das die Nachrichten der Sensoren und Aktoren zentral verarbeitet, ist für die Knoten die Down- und Uplink-Geschwindigkeit zu weiteren Netzwerken, an die das Gateway angebunden ist, nicht relevant. Deshalb entfallen die *Gateway Flags* sowie die Host Network Announcements (HNA) und damit auch das Feld *HNA Length*. Um ein Smart-Home-Gateway im Netzwerk bekannt zu machen, wird das höchste Bit des TTL-Felds als Gateway Flag bereitgestellt. Zudem entfällt der *Gateway Port*, da die Kommunikation auf der Sicherungsschicht stattfindet.

Um die Größe der OGM weiter zu reduzieren, wird die Sequence Number von 2 auf 1 Byte verkürzt.

In den *Flags* ist die *Is-Direct-Link* Flag und die *Unidirectional* Flag enthalten. *Is-Direct-Link* entfällt, da diese Flag nur bis BATMAN III genutzt wird. Da es sich bei den WLAN-Netzwerkinterfaces der ESP32-Mikrocontroller immer um bidirektionale Interfaces handelt, entfällt auch die *Unidirectional* Flag, womit keine Flag in diesem Feld übrig bleibt und das *Flags* Feld damit entfällt.

Die Abbildung 5.10 zeigt die BATMAN IV OGM mit den beschriebenen Anpassungen.

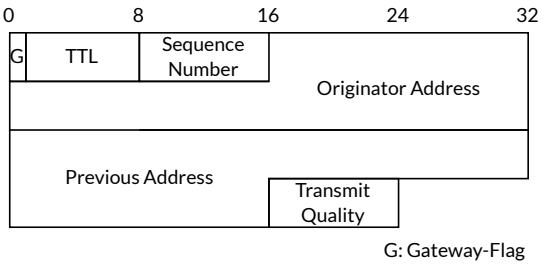

Abbildung 5.10: Angepasste BATMAN IV OGM ohne gemeinsamen Header. Länge: 15 Bytes

5.4.2 Nutzdaten

Zur Übertragung von Nutzdaten kommt eine *Data Message* (DM) zum Einsatz. Die Abbildung 5.11 zeigt das Format des Headers einer DM.

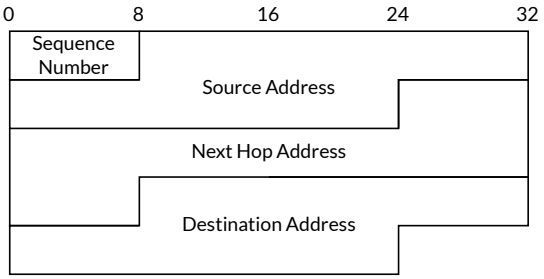

Abbildung 5.11: Header einer Data Message ohne gemeinsamen Header

Da ESP-NOW MAC-Adressen zur Adressierung nutzt, sind jeweils 6 Byte für die Quell- und Ziel-Adresse sowie für die Adresse des nächsten Hops reserviert. Zusätzlich ist eine Sequenznummer enthalten, um Wiederholungen bei Broadcasts zu vermeiden.

Um die Längenbegrenzung der ESP-NOW-Peerlist zu umgehen, werden alle Nachrichten via ESP-NOW per Broadcast versendet. Damit nur der nächste Hop eine DM weiterverarbeitet, ist die *Next Hop Address* enthalten. Da OGM immer an alle Nachbarn adressiert sind, muss für diesen Nachrichtentyp kein weiteres Feld zur Angabe des nächsten Hops hinzugefügt werden.

Empfängt ein Teilnehmer eine DM, deren nächster Hop nicht mit der eigenen MAC-Adresse übereinstimmt, wird die Nachricht verworfen. Stimmen die Adressen überein, wird die Nachricht weiterverarbeitet.

Entspricht die *Destination Address* der eigenen MAC-Adresse, wird die Nachricht an die Anwendung weitergereicht und nicht weitergeleitet. Entspricht die *Destination Address* nicht der eigenen Adresse, wird der nächste Hop zu dem Ziel aus der BATMAN-Tabelle ermittelt und die Nachricht an diesen weitergeleitet. Ist das Ziel nicht bekannt, wird die Nachricht verworfen.

Zudem wird bei den Data Messages unterschieden, ob der Absender einer DM eine Empfangsbestätigung des Empfängers erwartet. Erhält ein Empfänger eine DM vom Nachrichtentyp 1, sendet dieser eine Empfangsbestätigung an den Absender der Nachricht. Beim Nachrichtentyp 2 wird keine Empfangsbestätigung durch den Empfänger gesendet. Der Nachrichtentyp 3 entspricht einer Empfangsbestätigung und enthält nur den DM-Header ohne weitere Daten.

Auf den DM-Header folgen die Nutzdaten. Da die Länge der Nachricht bereits mit ESP-NOW übermittelt wird, enthält der DM-Header kein zusätzliches Feld für die Länge der Daten. Der Empfänger errechnet die Länge der Nutzdaten aus der von ESP-NOW übermittelten Länge abzüglich der Größe des DM-Headers.

6 Umsetzung

6.1 Implementierung

Die Mesh-Netzwerk-Library, genannt *ENMesh*, wurde in C entwickelt und nutzt für plattformspezifische Funktionalität der ESP32-Mikrocontroller-Familie das Framework *ESP-IDF* von Espressif.

Kern der Implementierung ist die Logik zur Verarbeitung und Weiterleitung der *BATMAN OGM* und *Data Messages* (DM). Eingehende Nachrichten werden in eine FIFO Queue (*Receive Queue*) geschrieben und sequenziell von einem *Receive Thread* verarbeitet, der die Nachrichten an die BATMAN- oder DM-Logik delegiert. Dort werden die Nachrichten nach den Beschreibungen aus Kapitel 5 verarbeitet und anschließend entweder an andere Knoten weitergeleitet oder über eine FIFO Queue der Anwendung bereitgestellt. Bei Implementierung der Library wurde der Fokus auf Plattformunabhängigkeit gelegt, sodass alle plattformspezifischen Funktionen wie Senden und Empfangen der Nachrichten sowie einige weitere Funktionen über eine als *Platform Adapter* bezeichnete Komponente bereitgestellt werden. Damit soll der Aufwand für eine mögliche Portierung zu anderen Plattformen reduziert werden. Die folgende Abbildung 6.1 zeigt einen Überblick der Architektur von ENMesh als UML-Komponentendiagramm.

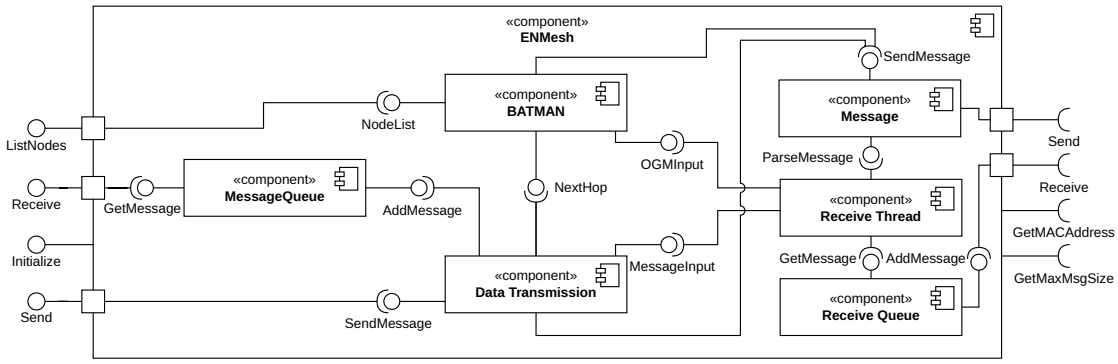

Abbildung 6.1: UML-Komponentendiagramm: ENMesh Architektur

Über eine API stellt ENMesh Funktionen zur Initialisierung, zum Senden und Empfangen von Nachrichten und zum Auflisten bekannter Nodes bereit. Dabei war das Ziel, eine simple API bereitzustellen, die mit geringem Aufwand in bestehende Anwendungen integriert werden kann, wie der Quellcode 1 zeigt.

```
1  enm_platform_adapter_t esp32_platform_adapter = ESP32_PLATFORM_ADAPTER_DEFAULT;
2  enm_crypto_key_t key = {...};
3  enm_mac_t mac_addr = {...};
4  enm_handle_t *enm_handle = enm_init(esp32_platform_adapter, mac_addr, key);
5  enm_start(enm_handle);
```

Quellcode 1: Initialisierung von ENMesh (ohne Fehlerbehandlung) in 5 Codezeilen

Nachrichtenempfang

In diesem Abschnitt wird der Ablauf bei Erhalt einer Nachricht beschrieben.

Bei Empfang einer Nachricht durch ESP-NOW wird ein zuvor registriertes Receive Callback aus dem Platform Adapter aufgerufen und die MAC-Adresse des Absenders, die Daten und deren Länge in Bytes durch ESP-NOW als Argumente übergeben.

Da der Buffer für die empfangenen Daten nach Beenden des Callbacks freigegeben wird, muss in dem Callback eine Kopie dieser Daten angelegt werden. Diese wird anschließend mit dem Absender und der Länge der Daten zur weiteren Verarbeitung der Receive Queue hinzugefügt. Dabei handelt es sich um eine FIFO Queue, aus der ein Receive Thread sequenziell Nachrichten entnimmt und verarbeitet. Im Receive Thread wird zuerst der unverschlüsselte Common Header ausgelesen und validiert. Ist der Header gültig, wird die Nachricht mit dem Pre-Shared Key und dem mitgesendeten Initialisierungsvektor (IV) entschlüsselt, um den Inhalt der Nachricht auszulesen und die Integrität der Nachricht anhand des *Message Authentication Code* (MAC) zu überprüfen.

Handelt es sich bei der erhaltenen Nachricht um eine OGM, wird diese an die BATMAN-Logik weitergegeben. Dort wird die OGM validiert und, wenn sie gültig ist, nach der Beschreibung aus Kapitel 5.3.2 verarbeitet. Anhand dieser Regeln wird anschließend entschieden, ob die Nachricht weitergeleitet werden soll. Soll eine Weiterleitung erfolgen, wird die OGM an die *enm_message_send_ogm* Funktion der Message-Komponente zur Weiterleitung übergeben.

Handelt es sich um eine Data Message, wird zunächst der Data Message Header validiert. Da bei ENMesh alle Nachrichten via ESP-NOW per Broadcast übertragen werden, wird zuerst das Next Hop Feld ausgelesen. Entspricht diese MAC-Adresse nicht der eigenen, wird das Paket verworfen. Stimmt das Next Hop Feld mit der eigenen MAC-Adresse überein, wird das Destination-Feld der Nachricht ausgelesen. Entspricht dieses Feld nicht der eigenen MAC-Adresse, wird der nächste Hop für das Ziel aus der BATMAN-Komponente ermittelt und in das Next Hop Feld eingetragen. Daraufhin wird die Nachricht verschlüsselt und in Richtung des Ziels weitergeleitet. Handelt es sich um die eigene MAC-Adresse im Destination-Feld, wird die Nachricht durch die Data Transmission Logik weiter verarbeitet.

Bei einer Nachricht vom Typ *ENM_MESSAGE_TYPE_DATA_ACK* wird zunächst eine Empfangsbestätigung an den Absender gesendet, um den erfolgreichen Empfang zu bestätigen. Anschließend wird der Inhalt der Nachricht zur Message Queue hinzugefügt und damit der Anwendung zur Verfügung gestellt.

Message-Komponente

Die Message Komponente enthält mehrere Funktionen zum Senden und Parsen von Nachrichten sowie Funktionalität zum Ver- und Entschlüsseln der Nachrichten. Im Folgenden Abschnitt wird auf die Funktionsweise der Message-Komponente eingegangen und der Ablauf zweier Funktionen anhand des Quellcodes erläutert.

Die Funktion enm_message_send_dm erstellt aus dem Payload, dem Ziel und dem nächsten Hop eine Nachricht, verschlüsselt diese und sendet sie anschließend über die Send Funktion des Platform Adapters. Zudem kann per Parameter die Empfangsbestätigung ein- und ausgeschaltet werden. Quellcode 2 zeigt die Signatur dieser Funktion. Der vollständige Code dieser Funktion befindet sich auf der beiliegenden CD-ROM.

```
1  bool enm_message_send_dm(enm_handle_t *enm_handle, const enm_mac_t next_hop, const
   ↪  enm_mac_t destination, const uint8_t *data, enm_message_length_t data_length,
   ↪  bool ack)
```

Quellcode 2: Signatur der Funktion enm_message_send_dm

Zuerst überprüft die Funktion, ob die Größe der Nachricht, bestehend aus den Längen von Common- und Data-Header sowie dem Payload und dem MAC, über der maximalen Nachrichtengröße der Send Funktion aus dem Platform Adapter liegt. Da es sich um ESP-NOW handelt, liegt diese 250 Bytes pro Nachricht.

```
21  // check size
22  size_t common_header_size = sizeof(enm_common_header_t);
23  size_t dm_header_size = sizeof(enm_data_message_header_t);
24  size_t total_message_size = common_header_size + dm_header_size + data_length +
    ↪  ENM_CRYPTO_TAG_LENGTH;
25  if (total_message_size > enm_handle->platform_adapter.get_max_msg_size())
26  {
27          enm_log_err(fn_enm_message_send_dm, "message too large");
28          return false;
29  }
```

Anschließend wird der Data Message Header erzeugt. Dazu wird die intern verwaltete *Sequence Number* übernommen und anschließend hochgezählt. Zudem werden die Felder *Source*, *Next Hop* und *Destination* mit den Argumenten, die der Funktion übergeben wurden, befüllt. Daraufhin wird der konstruierte Data Message Header zusammen mit den Daten in einen Buffer kopiert. Der Inhalt des Buffers stellt den Plaintext der Nachricht dar und wird im weiteren Verlauf verschlüsselt.

```
31  // create data message header
32  enm_data_message_header_t dm_header = {.sequence_number =
    ↪  enm_handle->dm_sequence_number};
33  enm_handle->dm_sequence_number++;
34
35  memcpy(dm_header.source, enm_handle->mac_addr, ENM_MAC_ADDRESS_LENGTH);
36  memcpy(dm_header.next_hop, next_hop, ENM_MAC_ADDRESS_LENGTH);
37  memcpy(dm_header.destination, destination, ENM_MAC_ADDRESS_LENGTH);
38
39  // create buffer for dm header and dm data
40  size_t dm_length = dm_header_size + data_length;
41  uint8_t dm_buffer[dm_length];
42  memcpy(dm_buffer, &dm_header, dm_header_size);
43  memcpy(dm_buffer + dm_header_size, data, data_length);
```

Anschließend wird der Common Header erzeugt, der die Version des Protokolls, den Typ der Nachricht sowie den zufällig generierten Initialisierungsvektor (IV) für die Verschlüsselung enthält. Abhängig von dem Wert des *ack* Parameters, der an die Funktion übergeben wurde, wird entweder eine ENM_MESSAGE_TYPE_DATA_ACK oder ENM_MESSAGE_TYPE_DATA_NO_ACK Nachricht erzeugt. Damit wird dem Ziel angezeigt, ob eine Empfangsbestätigung bei erfolgreichem Empfang erwartet wird.

```
45  // create common header
46  enm_message_type_t message_type = ack ? ENM_MESSAGE_TYPE_DATA_ACK :
    ↪  ENM_MESSAGE_TYPE_DATA_NO_ACK;
47  enm_common_header_t header = enm_message_create_common_header(message_type);
```

Im vorletzten Schritt wird zuerst ein Buffer für den Ciphertext angelegt. Die Länge des Ciphertexts setzt sich aus den Längen von Common Header, Data Message Header, Payload und MAC zusammen. Da der Common Header nicht verschlüsselt wird, wird dieser unverschlüsselt in den Ciphertext-Buffer kopiert. Mit der Funktion enm_crypto_encrypt, die einen Wrapper für die mbedtls_ccm_encrypt_and_tag Funktion von der MbedTLS Library ist, wird der Plaintext im *Counter with CBC-MAC* Verfahren verschlüsselt und mit dem MAC in den Ciphertext Buffer nach dem Common Header geschrieben. Dabei wird der MAC sowohl über den Plaintext als auch den Common Header generiert.

```
49  // create buffer for ciphertext
50  size_t ciphertext_length = common_header_size + dm_length + ENM_CRYPTO_TAG_LENGTH;
51  uint8_t ciphertext[ciphertext_length];
52
53  // copy common header into ciphertext buffer
54  memcpy(ciphertext, &header, common_header_size);
55
56  // encrypt message
57  if (!enm_crypto_encrypt(
58          enm_handle->crypto_context, // contains key
59          dm_buffer, dm_length, // plaintext
60          (const uint8_t *)&header, common_header_size, // use common header for
    ↪  mac
61          ciphertext + common_header_size, // write ciphertext after common header
62          header.ccm_iv)) // use randomly generated iv from common header
63  {
64          enm_log_err(fn_enm_message_send_dm, "enm_crypto_encrypt failed");
65          return false;
66  }
67
```

Abschließend wird der Ciphertext mit der Ziel-Adresse zum Senden an die Send Funktion des Platform Adapters übergeben.

```
66  // send message
67  if (!enm_handle->platform_adapter.send(ENM_BROADCAST_MAC, ciphertext,
    ↪  ciphertext_length))
68  {
69          enm_log_err(fn_enm_message_send_dm, "failed to send data");
70          return false;
71  }
```

Auf der Empfänger-Seite wird die Funktion enm_message_receive genutzt, um den Inhalt einer Nachricht zu entschlüsseln. Dazu werden die verschlüsselten Daten *data* sowie deren Länge *data_length* und der Buffer *message* für die entschlüsselte Nachricht als Parameter übergeben. Quellcode 3 zeigt die Signatur dieser Funktion. Der vollständige Code dieser Funktion befindet sich auf der beiliegenden CD-ROM.

```
1  bool enm_message_receive(enm_handle_t *enm_handle, const uint8_t *data,
   ↪  enm_message_length_t data_length, uint8_t *message)
```

Quellcode 3: Signatur der Funktion enm_message_receive

Zuerst wird der unverschlüsselte Common Header in den Message Buffer kopiert. Anschließend wird die Nachricht mit der Funktion enm_crypto_decrypt entschlüsselt, die ein Wrapper für mbedtls_ccm_auth_decrypt von der MbedTLS Library ist. Dazu wird der Ciphertext und der Buffer für die entschlüsselte Nachricht sowie der IV an die Funktion übergeben. Schlägt die Entschlüsselung fehl, beispielsweise weil der MAC nicht zu dem Plaintext und dem Common Header passt, terminiert die Funktion und schreibt keine Daten in den Message Buffer.

```
21  // copy common header into decrypted message buffer
22  const enm_common_header_t *common_header = (const enm_common_header_t *)data;
23  memcpy(message, data, sizeof(enm_common_header_t));
24
25  // decrypt message
26  if (!enm_crypto_decrypt(enm_handle->crypto_context,
27  data + sizeof(enm_common_header_t), data_length - sizeof(enm_common_header_t) -
    ↪  ENM_CRYPTO_TAG_LENGTH, // plaintext
28  data, sizeof(enm_common_header_t),
29  message + sizeof(enm_common_header_t), common_header->ccm_iv))
30  {
31          enm_log_err(fn_enm_message_receive, "enm_crypto_decrypt failed");
32          return false;
33  }
```

Wurde die Nachricht erfolgreich entschlüsselt, werden die Werte des Common Headers validiert. Schlägt die Validierung fehl, wird die Nachricht im weiteren Verlauf verworfen.

```
35  // validate common header
36  if (common_header->version != ENM_COMMON_HEADER_VERSION)
37  {
38          enm_log_err(fn_enm_message_receive, "version mismatch");
39          return false;
40  }
41  if (common_header->type >= ENM_MESSAGE_TYPE_MAX)
42  {
43          enm_log_err(fn_enm_message_receive, "invalid message type");
44          return false;
45  }
```

Der folgende Abschnitt beschreibt einen Ansatz zur Ermittlung des Speicherbedarfs eines Knotens im Netzwerk für eine gegebene Anzahl Originators und direkte Nachbarn. Der Speicherbedarf der Library für die Routing-Informationen eines Originators entspricht $27 + 2 \cdot ((k/8) + 1)$ Bytes, wobei k die Größe der Bitmap in Bits darstellt, die den Empfang von OGM für Receive- und Echo-Quality (siehe 5.3.2) speichert. Da BATMAN eine Qualitätsmetrik für die Kombination aus Originator und Nachbar speichert, entsteht pro direktem Nachbar ein zusätzlicher Speicherbedarf von $17+((k/8)+1)$ Bytes für jeden Originator.

Mit folgender Formel lässt sich der Speicherbedarf eines Knotens im Mesh-Netzwerk in Bytes berechnet werden, wobei $n_{originators}$ die Anzahl der bekannten Originators und $n_{neighbors}$ die Anzahl direkter Nachbarn darstellt.

$$(n_{originators} - 1) \cdot (27 + 2 \cdot ((k/8) + 1) + n_{neighbors} \cdot (27 + 2 \cdot ((k/8) + 1)))$$

Somit entsteht in einem Mesh-Netzwerk mit 20 Knoten, von denen 6 direkte Nachbarn des betrachteten Knotens sind, der folgende Speicherbedarf:

$$n_{originators} = 20$$
$$n_{neighbors} = 6$$
$$k = 16$$
$$(20 - 1) \cdot (27 + 2 \cdot ((16/8) + 1) + 6 \cdot (27 + 2 \cdot ((16/8) + 1))) = 4389 \text{ Bytes.}$$

Anhand des Speicherbudgets der Mikrocontroller aus der ESP32-Mikrocontroller-Familie (272 KB bis 520 KB [Espi]) sowie dem Speicherbedarf der Anwendung kann mit dieser Formel die maximale Anzahl Knoten im Mesh-Netzwerk bestimmt werden.

6.2 Qualitätssicherung

Im Rahmen der Qualitätssicherung wurde eine Analyse des ESP-NOW-Protokolls durchgeführt und einige Probleme mit dem Frameformat, der Validierung von Headern, Replay-Angriffen und der Verschlüsselung entdeckt, die in den folgenden Abschnitten ausgeführt werden.

6.2.1 Frameformat

ESP-NOW setzt zur Datenübertragung Vendor Specific Action Frames (VSAF) ein [Espa], die im 802.11-Standard für den Austausch hersteller-spezifischer Informationen vorgesehen sind [IEE21]. Da der VSAF ein Subtyp des Action Frame ist, der wiederum Subtyp des Management Frame ist, folgt ein VSAF der Struktur eines Management Frames (siehe 2.1.3). Die Abbildung 6.2 zeigt einen Vendor Specific Action Frame.

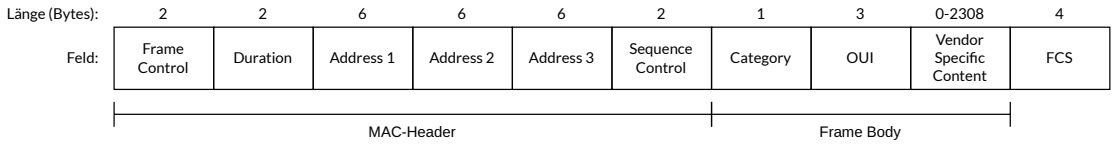

Abbildung 6.2: 802.11 Vendor Specific Action Frame

In dem Frame Body eines VSAF sind die folgenden Felder enthalten.

Category

Das Category-Feld zeigt den Subtyp des Action Frames an und enthält den Wert 127 für unverschlüsselte VSAF oder 126 für verschlüsselte VSAF. [IEE21]

Organizationally Unique Identifier (OUI)

Ein Organizationally Unique Identifier (OUI) ist 3 Byte groß und wird von der IEEE an Hersteller vergeben, um diese zu identifizieren. [Rec12]
Empfänger können den OUI nutzen, um auf VSAF von Geräten eines spezifischen Herstellers zu filtern.

Vendor Specific Content

Das Format und die Inhalte dieses Felds sind dem Hersteller überlassen. Die maximale Länge des Felds beträgt 2308 Bytes. [IEE21]

ESP-NOW fügt in den *Vendor Specific Content* eines VSAF maximal ein *Vendor Specific Information Element* (VSIE) ein [Espa]. Dies ist ein *Information Element* (IE, siehe 2.1.3) mit der Element ID 221 für hersteller-spezifische Informationen. Die Abbildung 6.3 zeigt ein VSIE.

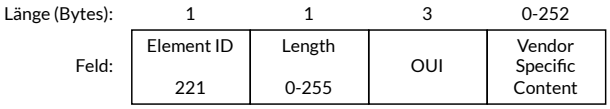

Abbildung 6.3: 802.11 Vendor Specific Information Element, nach [IEE21, S. 1059]

Inhalt und Struktur des Vendor Specific Content eines VSIE werden durch den Standard nicht vorgegeben. ESP-NOW setzt die Nutzdaten zusammen mit einem Header in den Vendor Specific Content eines VSIE ein, wie die Abbildung 6.4 zeigt [Espa].

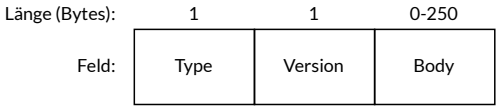

Abbildung 6.4: ESP-NOW-Header, nach [Espa]

Zu den Feldern *Type* und *Version* des ESP-NOW-Headers liegen keine Informationen in der Dokumentation vor. Eine Analyse mit einem Disassembler hat gezeigt, dass diese Felder jeweils einen statischen Wert nutzen.

Die Abbildung 6.5 zeigt einen vollständigen ESP-NOW-Frame.

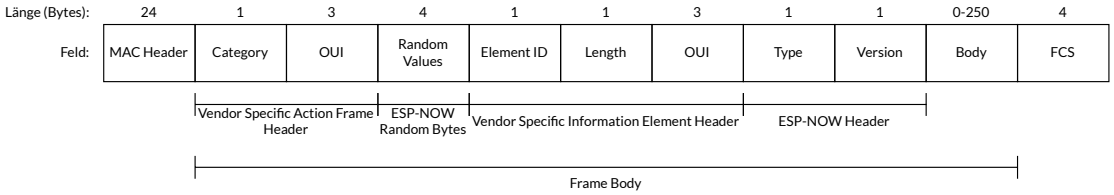

Abbildung 6.5: ESP-NOW-Frame

Wie die Abbildung 6.5 zeigt, werden durch ESP-NOW zusätzlich 4 *Random Bytes* zwischen dem Header des VSAF und dem Header des VSIE in den Frame eingefügt, die dem Schutz vor Replay-Angriffen dienen sollen und in dem folgenden Abschnitt 6.2.1 weiter behandelt werden.

Da die Länge des *Vendor Specific Content* eines VSIE auf 252 Bytes begrenzt ist und durch ESP-NOW maximal ein VSIE pro Frame genutzt wird, können abzüglich des ESP-NOW-Headers maximal 250 Bytes Nutzdaten pro Frame übertragen werden. Zudem werden bei maximaler Länge des VSIE lediglich 257 von den 2308 Bytes des VSAF Frame Body durch ESP-NOW genutzt. Mögliche Gründe für diese Design-Entscheidung werden nicht genannt.

Zudem liegen einige Informationen der VSAF und VSIE Header doppelt vor. Der OUI ist sowohl im VSAF-Header als auch im VSIE-Header enthalten. Zudem zeigen sowohl das Category-Feld des VSAF Headers, als auch die Element ID des VSIE an, dass es sich um hersteller-spezifische Informationen handelt.

Da keine weitere Dokumentation vorliegt, bleibt unklar, wieso ein VSIE als Inhalt des VSAF gewählt wurde und weshalb nur eins enthalten ist. Der Hersteller wurde darauf hingewiesen, eine Antwort steht aus.

Replay Angriff

Bei einem Replay Angriff nutzt ein Angreifer Daten, die in der Vergangenheit übertragen wurden und spielt diese auf dem Übertragungsmedium erneut ein, um so eine Aktion bei dem Empfänger auszulösen. [AST12]

Wie bereits im vorherigen Abschnitt erwähnt, überträgt ESP-NOW 4 zufällige Bytes in dem *Random Values* Feld mit jedem Frame. Dies soll als Schutz vor Replay Angriffen bei verschlüsselter Kommunikation dienen [Espa]. Die Dokumentation enthält dazu keine weiteren Details. Daher wurde mit einem Disassembler eine Analyse der ESP-NOW Library durchführt und die folgenden Abläufe rekonstruiert.

Mit der Initialisierung der ESP-NOW-Library wird ein Speicherbereich mit einer Größe von vier Bytes reserviert, um sich die zuletzt erhaltenen oder gesendeten Random Bytes zu merken.

Der Sender generiert 4 zufällige Bytes für das Random Byte Feld, die jedoch nicht den Random Bytes der zuletzt gesendeten oder empfangenen Nachricht entsprechen dürfen.

Der Empfänger merkt sich nach Erhalt eines ESP-NOW-Frames die Random Bytes. Nur bei einem erneut übertragenen Frame, erkennbar an dem gesetzten Retry-Flag im MAC-

Header des Frames, wird überprüft, ob die Bytes mit den zuletzt empfangenen Random Bytes übereinstimmen. Stimmen die Random Bytes überein, wird der Frame verworfen.

Damit verhindert die Implementierung lediglich, dass ein wiederholt gesendeter Frame erneut empfangen wird. Sollte jedoch ein weiterer Peer in der Zwischenzeit senden und damit die zuletzt empfangenen Random Bytes überschreiben, könnte danach ein Retry-Frame erneut empfangen werden oder ein Angreifer den vorletzten oder ältere Frames erfolgreich erneut einspielen.

Dieses Verfahren ist als Schutz gegen Replay-Angriffe ungeeignet, insbesondere auf Geräten mit geringem Speicherbudget wie den ESP32-Mikrocontrollern. Das empfangende Gerät müsste sich alle empfangenen zufälligen Bytes merken, um einen Replay-Angriff effektiv zu verhindern, da nur so ausgeschlossen werden kann, dass eine Kombination der 4 zufälligen Bytes schon empfangen wurde. Zusätzlich müsste die Liste der erhaltenen zufälligen Bytes über einen Neustart hinweg bestehen, um Replay-Angriffe auch nach dem Neustart zu verhindern.

Zum Zeitpunkt der Analyse konnten mehrere aufeinander folgende verschlüsselte Frames mit den gleichen zufälligen 4 Bytes übertragen werden. Dazu wurde ein Proof-of-Concept (PoC) in Python mit der *Scapy* Library entwickelt und dem Hersteller zur Verfügung gestellt. Der Fehler wurde durch den Hersteller bestätigt. Zu diesem Zeitpunkt wurde das Problem noch nicht behoben.

Header Validierung

Im Rahmen der Qualitätssicherung wurde die Validierung der Header-Felder innerhalb des Frame Body geprüft. Dabei wurde ein Disassembler zur Analyse der ESP-NOW Library genutzt und eine Analyse der Frames mit *Wireshark* durchgeführt.

Die folgende Abbildung 6.6 zeigt die Felder eines ESP-NOW-Frames.

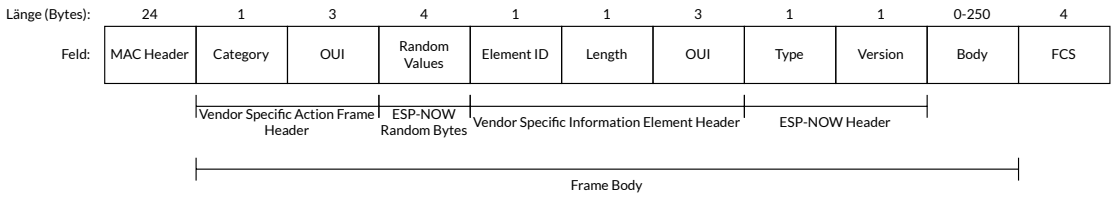

Abbildung 6.6: ESP-NOW-Frame

Die Analyse hat gezeigt, dass die OUI Felder, sowohl im Header des Vendor Specific Action Frame (VSAF) als auch im Header des Vendor Specific Information Element (VSIE) nicht geprüft werden. Damit besteht die Möglichkeit, dass auch VSAF anderer Hersteller verarbeitet werden, wenn das Format des Frame Body ähnlich oder gleich ist.

Zudem wird die Element ID des VSIE nicht geprüft. Lediglich das Length Feld des VSIE wird ausgelesen.

Wie bereits in 6.2.1 erklärt, besteht der ESP-NOW-Header aus den Feldern Type und Version. Diese werden vor dem Senden mit statischen Werten belegt, beim Empfang aber nicht geprüft. Daher könnten Probleme mit der Rückwärtskompatibilität auftreten, falls zukünftig weitere Nachrichtentypen oder Versionen des Protokolls genutzt werden.

Mit einem PoC konnte gezeigt werden, dass ESP-NOW-Frames mit beliebigen Werten in den erwähnten Felder befüllt werden können und trotzdem von der Anwendung empfangen werden. Die folgende Abbildung 6.7 zeigt, welche Felder des Frames von ESP-NOW validiert werden.

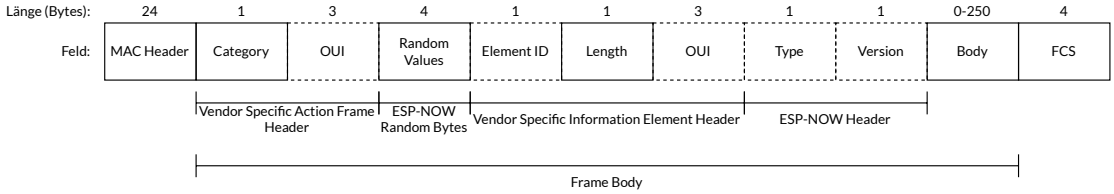

Abbildung 6.7: ESP-NOW-Frame. Felder mit gestricheltem Rand werden nicht validiert.

Die Problembeschreibung wurde an den Hersteller übermittelt. Eine Antwort steht zu diesem Zeitpunkt noch aus.

Schlüsselstrom Reuse

In einem Versuchsaufbau mit zwei ESP32-Mikrocontrollern, bei dem jeweils einer als Sender und Empfänger agiert, wurden vom Sender nach einem Neustart jeweils eine Nachricht per ESP-NOW an den Empfänger übertragen. Dabei war die Verschlüsselung für den Peer in ESP-NOW aktiviert und auf beiden Geräten identische PMK und LMK festgelegt, wie in 2.2 beschrieben. Zudem war der Inhalt der Nachricht statisch, sodass bei jeder Übertragung die gleiche Nachricht gesendet wurde. Dabei wurde die verschlüsselten ESP-NOW-Frames mit *Wireshark* untersucht.

Dieser Versuch wurde mehrfach durchgeführt und hat gezeigt, dass die Ciphertexte zu einem großen Teil übereinstimmen. Ein Abgleich der mit dem Header-Format (siehe 6.5) hat gezeigt, dass lediglich die Random Bytes unterschiedliche Werte aufweisen.

Bei der Verschlüsselung mit AES im Counter-Mode wird aus einem Schlüssel, einem Initialisierungsvektor (IV) und einem Zähler ein Schlüsselstrom erzeugt, der blockweise per XOR auf den Klartext angewandt wird. Dabei muss für jede Nachricht eine neue Kombination aus Schlüssel und IV genutzt, um Rückschlüsse auf den Plaintext zu vermeiden. Dazu kann, bei gleichem Schlüssel, der IV mit jeder Nachricht einen neuen Wert zugewiesen bekommen. Das folgende Beispiel zeigt die möglichen Auswirkungen bei mehrfacher Verwendung einer Kombination aus Schlüssel und IV. [Wä18]

In diesem Beispiel nach [Wä18, S. 69f] werden die Klartexte M_i und M_i' durch XOR mit demselben Schlüsselstrom E_K, bestehend aus dem Schlüssel K, dem Initialisierungsvektor IV und dem Zähler Z_i verschlüsselt. Das Ergebnis dieser Operation sind die beiden Ciphertexte C_i und C_i'.

$C_i = M_i \oplus E_K(IV, Z_i)$
$C_i' = M_i' \oplus E_K(IV, Z_i)$

Nun kann ein XOR auf die beiden Ciphertexte C_A und C_B angewandt werden.

$C_i \oplus C_i' = M_i \oplus E_K(IV, Z_i) \oplus M_i' \oplus E_K(IV, Z_i)$

Da es sich bei XOR um eine selbstinverse Funktion handelt [Wä18], hebt sich der per XOR angewandte identische Schlüsselblock auf und es bleibt ein XOR der beiden Plaintexts übrig.

$C_i \oplus C_i' = M_i \oplus M_i'$

Ist einer der Plaintexts bekannt, so kann durch eine XOR-Operation auch der andere Plaintext entschlüsselt werden.

$M_i' = M_i \oplus M_i' \oplus M_i$

Dieses Problem wird als *Schlüsselstrom-Reuse* bezeichnet [AST12].

Das von ESP-NOW eingesetzte Verschlüsselungsprotokoll *Counter with CBC-MAC Protocol* (CCMP) sieht vor, dass eine Packet Number (PN) pro Empfänger gespeichert wird und, beginnend bei Null, mit jedem Frame hochgezählt wird. Da die PN zusammen mit

der Empfängeradresse in den Initialisierungsvektor einfließt, wird die gleiche Kombination aus IV und Schlüssel in aufeinanderfolgenden Nachrichten vermieden. Zudem soll bei 802.11 für jede Verbindung zwischen Station und Access Point ein neuer Key ausgehandelt werden, um das mehrfache Auftreten der gleichen Kombination aus Schlüssel und IV über mehrere Verbindung hinweg zu verhindern. [IEE21]

Da ESP-NOW jedoch den gleichen Schlüssel für einen Peer nutzt und die Packet Number nach jedem Neustart bei Null beginnt, wird nach jedem Neustart die gleiche Abfolge des Schlüsselstroms genutzt.

Aus zeitlichen Gründen wurde dieses Problem dem Hersteller noch nicht gemeldet.

7 Evaluation

Zur Evaluation der entwickelten Mesh-Netzwerk-Lösung wurde ein Vergleich von ENMesh zu ESP-WIFI-MESH unter den Gesichtspunkten aus Aufgabenstellung und Analyse durchgeführt. Dazu wurden drei Szenarien entwickelt.

Im ersten Szenario wird ein Netzwerk mit zwei Teilnehmern zum Vergleich der Single-Hop-Kommunikation aufgebaut. Dabei wird die Round Trip Time (RTT) und Datenrate gemessen.

Im zweiten Szenario wird ein Netzwerk mit drei Teilnehmern aufgebaut, um den Vergleich der Startzeit und die Verzögerung bei Ausfall eines Knotens zu messen. Dabei befinden sich die drei Geräte in unmittelbarer Nähe zueinander und sind mit einem PC verbunden, um zur Messung der Startzeit einen gleichzeitigen Neustart durchzuführen. Wie bereits in der Aufgabenstellung beschrieben, sollen aufgrund der Mesh-Topologie mit ENMesh mehr direkte Wege zwischen den Teilnehmern bestehen. Zudem sind ENMesh-Netzwerke dezentral organisiert. Daher ist bei ENMesh eine geringere Ausfalldauer bei Entfernung eines Knotens zu erwarten. Weiter sollte die dezentrale Organisation des Netzwerks mit ENMesh schneller erfolgen als mit ESP-WIFI-MESH, da kein Root-Knoten zur Organisation des Netzwerks gewählt werden muss, weshalb eine geringere Startzeit zu erwarten ist.

Das dritte Szenario orientiert sich grob an dem Aufbau eines Smart-Home-Netzwerks. Die Geräte werden mit größerer räumlicher Entfernung als bei den vorherigen Szenarien platziert. Dabei soll Knoten 1 ein Smart-Home-Gateway darstellen. Die Knoten 2-5 stellen Sensoren und Aktoren dar. Dieses Szenario soll die Kommunikation zwischen einem Sensor und einem Aktor sowie einem Sensor und dem Smart-Home-Gateway darstellen, wobei RTT und Datenrate für beide Kommunikationswege gemessen werden. Damit wird sowohl bei ENMesh als auch bei ESP-WIFI-MESH zusätzlich zur Single-Hop-Kommunikation auch eine Multi-Hop-Kommunikation gemessen. Zudem wird die

Ausfalldauer beim Entfernen zweier Knoten gemessen. Durch die dezentrale Organisation sowie direkterer Verbindungen durch die Mesh-Topologie von ENMesh ist auch hier mit einer geringeren Ausfalldauer zu rechnen.

Die folgende Abbildung 7.1 zeigt die drei Szenarien.

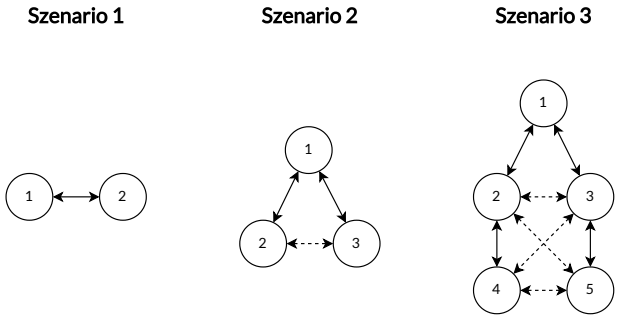

Abbildung 7.1: Drei Szenarien für den Vergleich. Durchgezogene Linien stellen eine Verbindung bei ESP-WIFI-MESH und ENMesh dar. Gestrichelte Linien stellen eine Verbindung bei ENMesh dar.

Da die Kommunikation mit 802.11 auf einem Shared Medium stattfindet, wurden die Versuche außerhalb der Bürozeiten an dem Hamburger Standort der inovex GmbH durchgeführt, um Interferenzen durch andere WLAN-Geräte zu minimieren.

Bei den Vergleichen wurde ESP-WIFI-MESH mit der automatischen Organisation (siehe 3.3) eingesetzt, da sich ein Mesh-Netzwerk mit ENMesh ebenfalls automatisch konfiguriert und aktuell keine Möglichkeit zur manuellen Konfiguration bietet. Da die automatische Organisation von ESP-WIFI-MESH einen Access Point (AP) zur Wahl des Root-Knotens erfordert, wurde ein AP vorgegeben. Bei ESP-WIFI-MESH findet die Kommunikation im Netzwerk auf dem gleichen WLAN-Kanal statt, der auch durch den AP genutzt wird. Daher wurde zur besseren Vergleichbarkeit auch für ENMesh der gleiche WLAN-Kanal für die Kommunikation gewählt.

Wie bereits in Kapitel 3.2 erwähnt wurde, werden die Nachrichten bei ESP-WIFI-MESH bei Paketverlust zwischen den einzelnen Knoten erneut übertragen. Da dieses Verfahren nicht deaktivierbar ist, kann kein Vergleich des PDR (Packet Delivery Ratio) durchgeführt werden.

Im Folgenden werden die Knoten 1-5 mit N1-N5 abgekürzt.

Als Testgeräte für die Versuche wurden ESP32-C3-DevKitC-02 Entwicklungsboards von Espressif eingesetzt, auf denen der ESP32-C3 Mikrocontroller verbaut ist.

Szenario 1

Für das erste Szenario wurden zwei Entwicklungsboards in geringer Entfernung gegenüber platziert und zur Stromversorgung sowie zur Datenauswertung per USB-Kabel mit einem PC verbunden. Die Abbildung 7.2 zeigt den Versuchsaufbau.

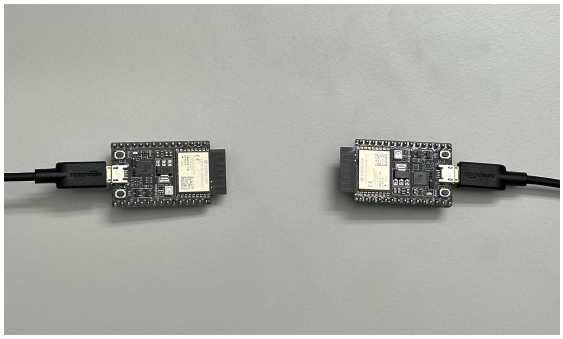

Abbildung 7.2: Szenario 1: Versuchsaufbau

Da die Entwicklungsboards die Protokollversionen 802.11b, 802.11g und 802.11n mit jeweils unterschiedlichen Datenraten unterstützen, wurde in diesem Szenario zusätzlich der Einfluss durch 802.11-Protokollversionen und deren Datenrate auf die Ergebnisse ermittelt. Da die Dokumentation von ESP-WIFI-MESH keine Hinweise zur verwendeten Protokollversion und Datenrate enthält und diese durch die API nicht konfigurierbar sind, wird im Folgenden nur ENMesh betrachtet. In den folgenden Abschnitten zur RTT und Datenrate werden die 802.11-Protokollversionen bei ENMesh verglichen und anhand der Ergebnisse eine der drei Versionen für die weiteren Szenarien gewählt.

Für die Versuche wurde jeweils die höchste Datenrate der Protokollversionen eingestellt.

Szenario 1: Round Trip Time

Zur Messung der Round Trip Time (RTT) wurden Nachrichten mit einem Byte Payload alle 100 Millisekunden von N1 zu N2 übertragen, worauf N2 mit Empfangsbestätigungen antwortete. Um daraus die RTT zu errechnen, ermittelte N1 die Dauer zwischen dem Senden einer Nachricht und dem Erhalt der Empfangsbestätigung. Dieser Versuch wurde jeweils 1000 Mal mit ESP-WIFI-MESH und ENMesh durchgeführt.

Für die Darstellung der Ergebnisse wurde ein Boxplot-Diagramm gewählt, da dieses den Median und die Spanne sowie das 3. und 1. Quartil angibt und damit Auskunft über die Verteilung der gemessenen Werte gibt. Die Abbildung 7.3 zeigt die Ergebnisse.

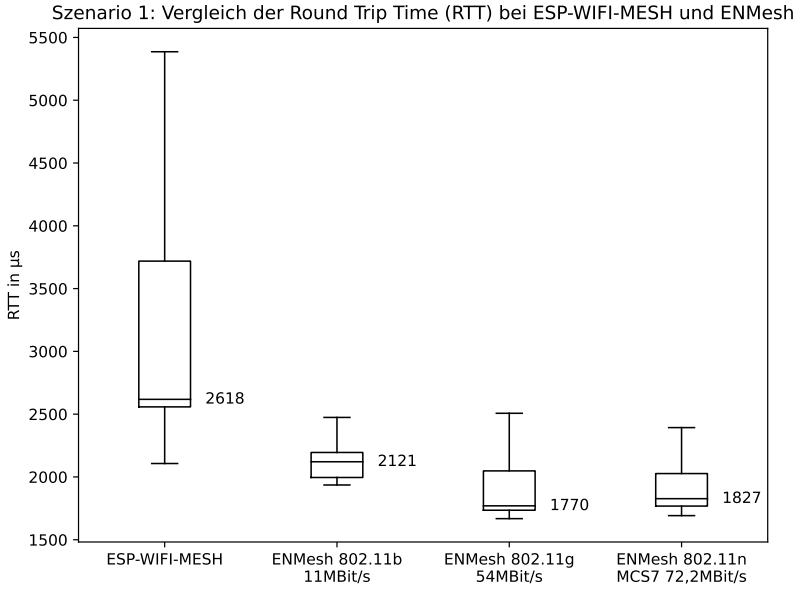

Abbildung 7.3: Szenario 1: Gemessene Single-Hop-RTT bei ESP-WIFI-MESH und ENMesh

Bei Betrachtung des Medians und der Quartile wird deutlich, dass ENMesh im Vergleich zu ESP-WIFI-MESH eine geringere Single-Hop-RTT aufweist. Zudem ist die Spannweite im Vergleich zu ESP-WIFI-MESH deutlich geringer.

Espressif gibt die Single-Hop-RTT bei ESP-WIFI-MESH mit 10-30 ms pro Hop an [Espb]. Die gemessenen Werte liegen damit deutlich unter den Angaben des Herstellers. Diese

Differenz ist möglicherweise auf die geringe Distanz zwischen N1 und N2 zurückzuführen (siehe 7.2). Zudem wurden die Messungen zu einem Zeitpunkt durchgeführt, an dem sich wenige WLAN-Geräte im Umfeld befanden, die das Ergebnis durch Nutzung des WLAN-Kanals hätten beeinflussen können.

Wie bereits in Kapitel 3 beschrieben, werden die Unicast Data Frames, die mit ESP-WIFI-MESH gesendet werden, durch Empfangsbestätigungen von jedem Knoten entlang des Pfads bestätigt. Erhält der Sender oder weiterleitende Knoten keine Empfangsbestätigung, wird der Data Frame erneut übertragen. Die ESP-WIFI-MESH Library gibt keine Auskunft über erhaltene Empfangsbestätigungen oder die Anzahl der Retransmissions. Daher könnte die Abweichung bei der RTT zwischen ESP-WIFI-MESH und ENMesh auf zusätzliche Acknowledgements oder die Verzögerung durch Retransmissions zurückzuführen sein.

Ein Vergleich der bei ENMesh gemessenen RTTs zeigt, dass die Protokollversionen 802.11g mit 54 MBit/s und 802.11n mit 72,2 MBit/s sowohl beim Median als auch bei den 1. und 3. Quartilen geringe RTTs erzielen konnten.

Szenario 1: Datenrate

Zur Messung der Datenrate sendet N1 100 Nachrichten an N2. Für ENMesh und ESP-WIFI-MESH wurde die Messung jeweils zehnmal durchgeführt. Dabei wurden die Nachrichten schnellstmöglich hintereinander gesendet und anschließend die Zeitspanne für den Erhalt der Nachrichten bei dem Empfänger N2 gemessen. Anhand von Anzahl und Größe der Nachrichten sowie der Zeitspanne wurde die Datenrate in MBit/s berechnet.

Für den Versuch senden ESP-WIFI-MESH und ENMesh Nachrichten mit der maximalen Payloadgröße. Da diese bei ESP-WIFI-MESH 1472 Bytes und bei ENMesh 218 Bytes beträgt, wurde ein zusätzlicher Versuch mit 218 Bytes Payloadgröße bei ESP-WIFI-MESH durchgeführt.

Zusätzlich wurde, wie bereits zuvor, ein Vergleich der 802.11 Protokollversionen bei ENMesh hinsichtlich Datenrate durchgeführt.

Für die Darstellung der Ergebnisse wurde ein Boxplot-Diagramm gewählt, da dieses den Median und die Spanne sowie das 3. und 1. Quartil angibt und damit Auskunft über die Verteilung der gemessenen Werte gibt. Die Abbildungen 7.4 und 7.5 zeigen die Ergebnisse des Versuchs.

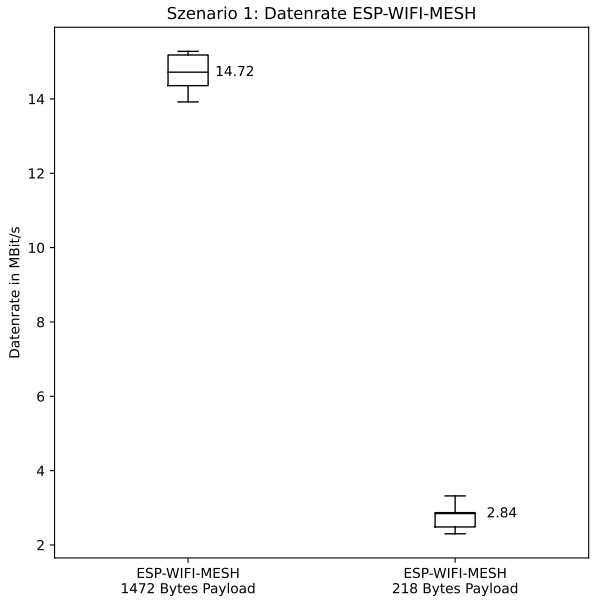

Abbildung 7.4: Szenario 1: Gemessene Datenrate bei ESP-WIFI-MESH

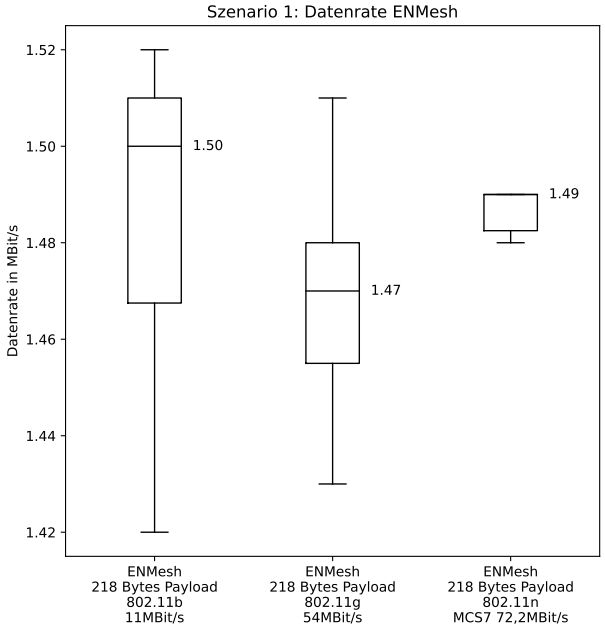

Abbildung 7.5: Szenario 1: Gemessene Datenrate bei ENMesh

Espressif gibt die Single-Hop-Datenrate bei ESP-WIFI-MESH mit ca. 10 MBit/s an [Espb]. Die Ergebnisse zeigen, dass diese Angabe mit einem Median von 14,72 MBit/s sowie einem Maximalwert von 16,43 MBit/s übertroffen wurde. Die höheren Messwerte könnten, wie bereits in dem letzten Abschnitt zur RTT erwähnt wurde, auf die geringe Distanz sowie die geringe Auslastung des WLAN-Kanals zurückzuführen sein.

Die Ergebnisse zeigen, dass ENMesh eine deutlich geringere Datenrate aufweist. Bei gleicher Payloadgröße erreicht ESP-WIFI-MESH eine Datenrate von 2,84 MBit/s (Median). Der Vergleich mit ESP-WIFI-MESH bei geringerer Payloadgröße zeigt, dass die niedrige Datenrate bei ENMesh teilweise auf die geringere Payloadgröße zurückzuführen sein könnte.

Die Ergebnisse des Vergleichs mit unterschiedlichen 802.11-Protokollversionen unter ENMesh haben gezeigt, dass die Datenraten vergleichbar sind. Zur Wahl einer 802.11-Protokollversion für die weiteren Versuche wird zusätzlich eine Tabelle mit Sendestärken aus dem Datenblatt des auf dem Entwicklungsboard enthaltenen Moduls *ESP32-C3-WROOM-02* herangezogen. Die Abbildung 7.6 zeigt die Sendestärken.

Table 13: TX Power with Spectral Mask and EVM Meeting 802.11 Standards

Rate	Min (dBm)	Typ (dBm)	Max (dBm)
802.11b, 1 Mbps	—	20.5	—
802.11b, 11 Mbps	—	20.5	—
802.11g, 6 Mbps	—	20.0	—
802.11g, 54 Mbps	—	18.0	—
802.11n, HT20, MCS0	—	19.0	—
802.11n, HT20, MCS7	—	17.5	—
802.11n, HT40, MCS0	—	18.5	—
802.11n, HT40, MCS7	—	17.0	—

Abbildung 7.6: Sendestärken des ESP32-C3-WROOM-02 Moduls nach 802.11-Protokollversion und Datenrate [Espg, S. 16]

ENMesh mit 802.11g (54 MBit/s) und 802.11n (72,2 MBit/s) zeigen in den Ergebnissen eine geringere RTT als 802.11b(11 MBit/s) sowie eine vergleichbare Datenrate. Aufgrund der höheren Sendeleistung bei 802.11g (54 MBit/s) im Vergleich zu 802.11n (72,2 MBit/s) (siehe 7.6), wird 802.11g mit 54 MBit/s für die weiteren Versuche verwendet.

Szenario 1: Energie

Zur Messung der durchschnittlichen Stromstärke der Entwicklungsboards wurde mit dem *Power Profiler Kit II* von *Nordic Semiconductors* bei einer Versorgungsspannung von 3,3 V über 60 Sekunden die Stromstärke gemessen. Die Abbildung 7.7 zeigt den Versuchsaufbau.

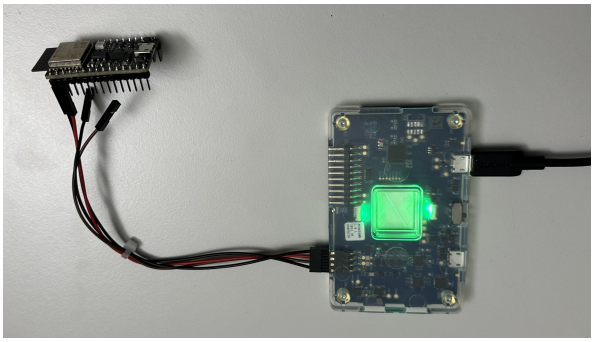

Abbildung 7.7: Szenario 1: Versuchsaufbau zur Strommessung an einem ESP32-C3-DevKitC-02 mit dem Power Profiler Kit II von Nordic Semiconductors

Dabei agiert N1 als Sender und N2 als Empfänger. In einem 200 ms Intervall sendet N1 Nachrichten mit einer Größe von 100 Byte an N2. Der Versuch wurde für ESP-WIFI-MESH und ENMesh durchgeführt. Die Abbildung 7.8 zeigt die Ergebnisse.

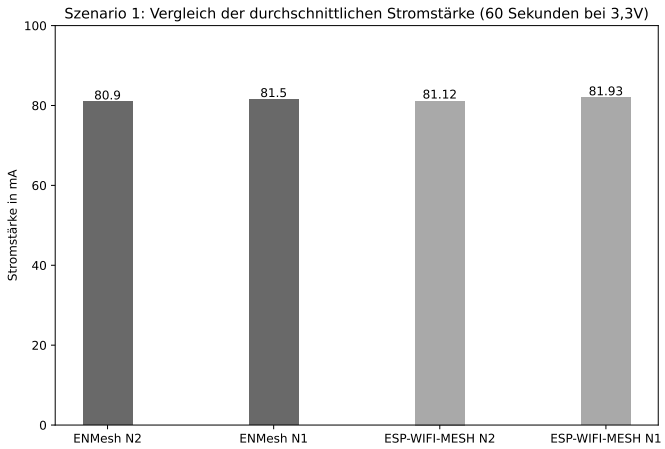

Abbildung 7.8: Szenario 1: Gemessene durchschnittliche Stromstärke über 60 Sekunden bei 3,3 V mit ESP-WIFI-MESH und ENMesh

Die Ergebnisse des Versuchs zeigen, dass die gemessenen Werte bei ENMesh und ESP-WIFI-MESH vergleichbar sind. Sowohl mit ENMesh als auch ESP-WIFI-MESH ist die durchschnittliche Stromstärke bei den Sendern geringfügig höher als bei den Empfängern. Der geringe Unterschied zwischen ENMesh und ESP-WIFI-MESH könnte auf den bereits hohen Strombedarf bei aktiviertem WLAN-Modul ohne Sende- oder Empfangsaktivität zurückzuführen sein. Die folgende Abbildung 7.9 zeigt die Stromstärke bei 3,3 V über 60 Sekunden eines ESP32-C3-DevKitC-02 mit aktiviertem WLAN-Modul ohne Sende- und Empfangsaktivität.

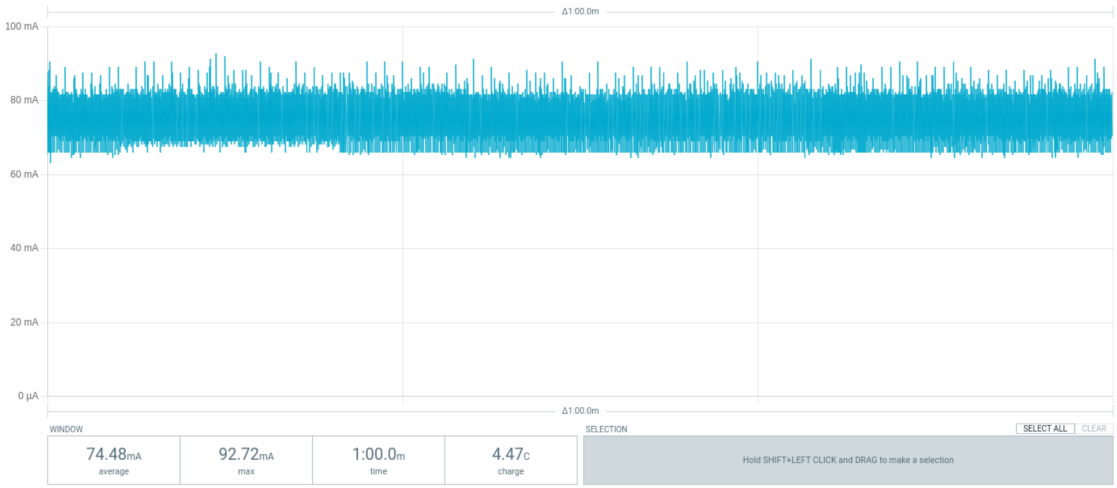

Abbildung 7.9: Szenario 1: Gemessene durchschnittliche Stromstärke mit aktiviertem WLAN-Modul eines ESP32-C3-DevKitC-02

Szenario 2

Für das zweite Szenario wurden drei Entwicklungsboards in geringer Entfernung gegenüber platziert und zur Stromversorgung sowie zur Datenauswertung per USB-Kabel mit einem PC verbunden. Die Abbildung 7.10 zeigt den Versuchsaufbau.

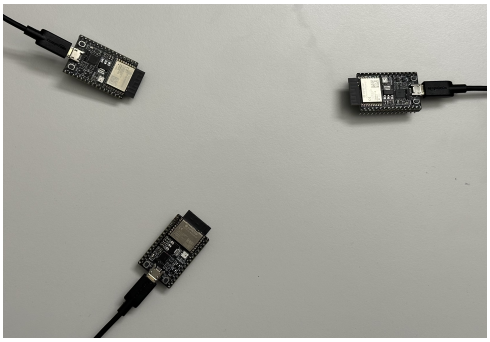

Abbildung 7.10: Szenario 2: Versuchsaufbau

Szenario 2: Startzeit

Als Startzeit wird im Folgenden die Zeitspanne von Initialisierung der Mesh-Software bis zum Empfang mindestens einer Nachricht durch jeden anderen Knoten im Netzwerk bezeichnet. Dazu senden die Knoten alle 50 ms eine Nachricht mit einem 1 Byte Payload. Jeder Knoten protokolliert die Startzeit des Mesh-Netzwerks sowie den Zeitpunkt bei Erhalt einer Nachricht. Der Versuch wurde jeweils 5 Mal mit ESP-WIFI-MESH und ENMesh durchgeführt. Die durchschnittliche Startzeit sowie die Spanne der Messwerte wird für jeden Knoten (N1-N3) auf der folgenden Abbildung 7.11 dargestellt.

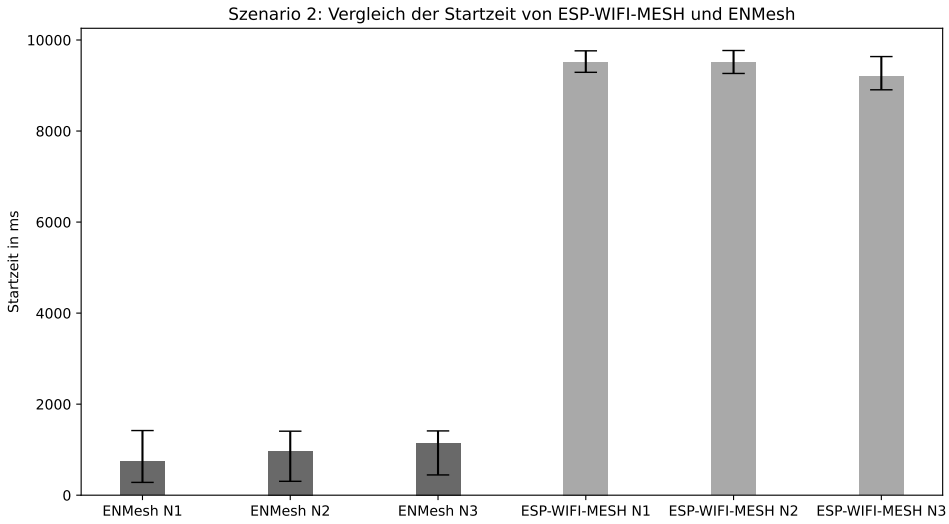

Abbildung 7.11: Szenario 2: Gemessene durchschnittliche Startzeit des Netzwerks

Der Versuch hat gezeigt, dass ENMesh eine deutlich geringere Startzeit aufweist. Dies könnte auf die dezentrale Organisation des Netzwerks zurückzuführen sein. Während ENMesh bereits ab Start kontinuierlich Routen-Informationen austauscht, muss mit ESP-WIFI-MESH zuerst ein Root-Knoten gewählt werden. Da ENMesh direkt nach Start die erste BATMAN OGM sendet und daraufhin in einem Intervall von einer Sekunde weitere OGM gesendet werden, können die Originator bei Single-Hop-Kommunikation bereits direkt nach dem Start bekannt sein, spätestens aber nach zwei OGM Intervallen.

Szenario 2: Ausfall eines Knotens

Um die Dauer der Unterbrechung bei Ausfall von N1 zu messen, wurden von N2 und N3 Nachrichten in einem 50 ms Intervall an den jeweils anderen Knoten gesendet. Die Unterbrechungsdauer entspricht dabei der Differenz zwischen dem Zeitpunkt der zuletzt erhaltenen Nachricht vor der Unterbrechung und dem Zeitpunkt nach der Unterbrechung, zu dem die erste Nachricht des jeweils anderen Knoten empfangen wurde. Bei diesem Versuchsaufbau stellt N1 dem Root-Knoten bei ESP-WIFI-MESH dar. Ein Ausfall dieses Knotens hat die Neuorganisation des Netzwerks zur Folge. Der Versuch wurde jeweils fünf Mal mit ESP-WIFI-MESH und ENMesh durchgeführt. Die folgende Abbildung 7.12 zeigt die Ergebnisse des Versuchs.

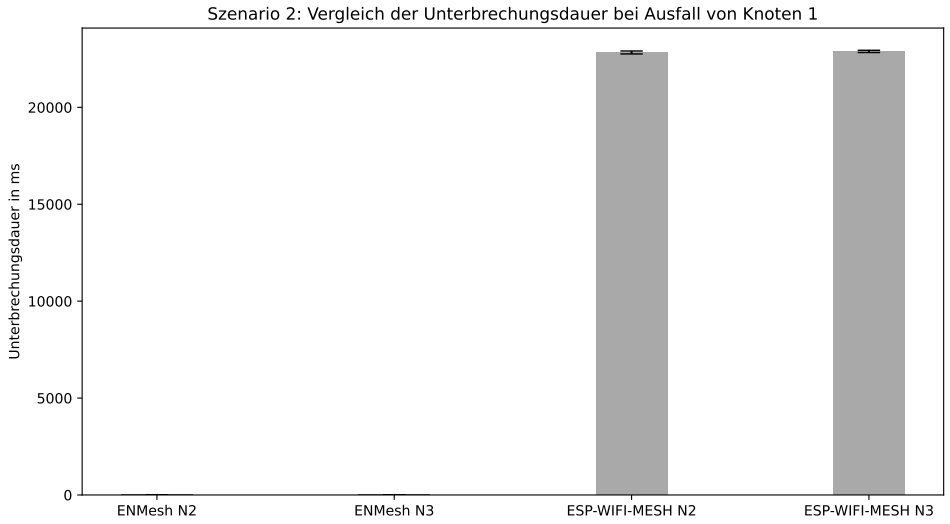

Abbildung 7.12: Szenario 2: Gemessene durchschnittliche Unterbrechungsdauer bei Ausfall von N1

Da ENMesh in diesem Szenario ein vollvermaschtes Mesh-Netzwerk ist und damit eine direkte Verbindung zwischen N2 und N3 besteht, entsteht keine Verzögerung durch den Ausfall von N1. Da das ESP-WIFI-MESH-Netzwerk in einer Baum-Topologie organisiert ist, wobei N1 der Root-Knoten und zugleich der Elternknoten von N2 und N3 ist, läuft die Kommunikation zwischen den beiden Knoten über N1. Der Ausfall von N1 unterbricht daher die Übertragung, bis sich die Knoten neu organisiert haben und N2 oder N3 zum neuen Root-Knoten gewählt haben.

Szenario 3

Für das dritte Szenario wurden fünf Entwicklungsboards nach dem Aufbau aus 7.1 am Hamburger Standort der inovex GmbH platziert. Die Abbildung 7.10 zeigt den Versuchsaufbau.

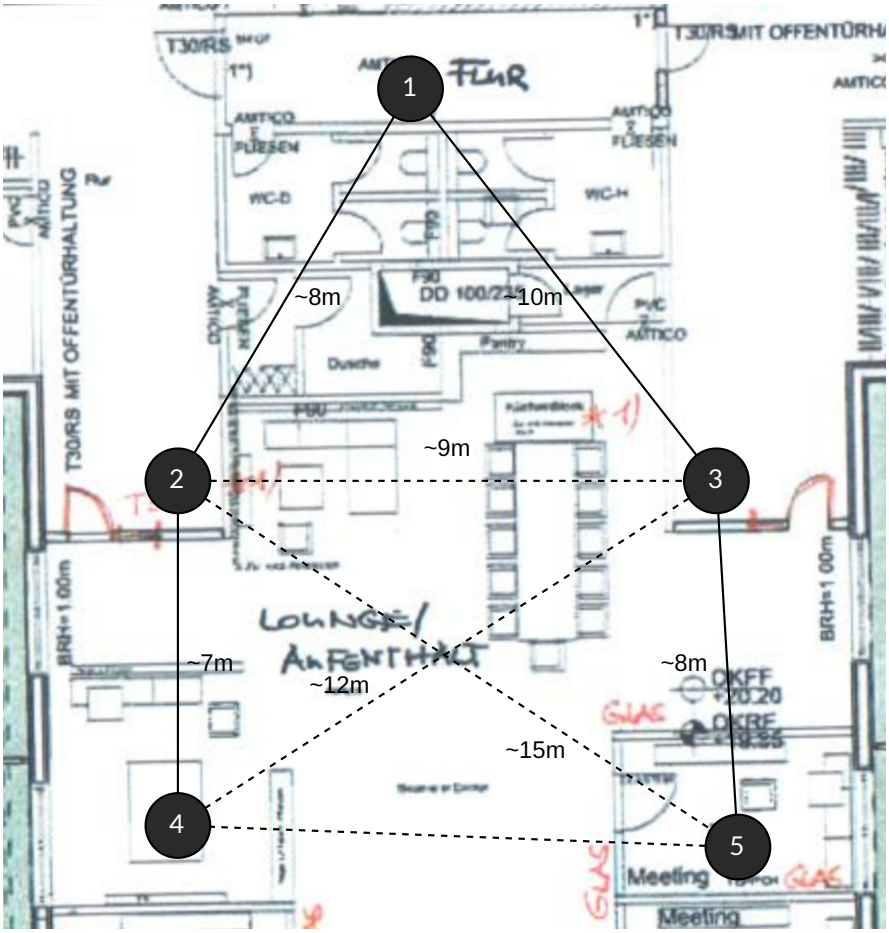

Abbildung 7.13: Szenario 3: Versuchsaufbau der Knoten 1-5 mit ungefähren Abständen. Durchgezogenen Linien stellen eine Verbindung bei ESP-WIFI-MESH und ENMesh dar. Gestrichelte Linien stellen eine Verbindung bei ENMesh dar.

Wie die Abbildung 7.10 zeigt, wird mit ESP-WIFI-MESH ein Netzwerk in einer Baumtopologie aufgebaut, während das Netzwerk mit ENMesh eine teilvermaschte Topologie aufweist.

Szenario 3: Round Trip Time

Die Messung der Round Trip Time (RTT) wurde nach der Beschreibung aus Szenario 1 durchgeführt. In diesem Versuch stellt N4 einen Sensor, N5 einen Aktor und N1 das Smart-Home-Gateway dar. Dabei wurde die RTT des Pfads von N4 zu N5 und von N4 bis N1 untersucht. Damit soll Single-Hop-Kommunikation zwischen Sensor und Aktor sowie die Multi-Hop-Kommunikation zwischen einem Sensor und dem Smart-Home-Gateway dargestellt und die Verzögerung durch die RTT untersucht werden.

Aus den bereits zuvor erwähnten Gründen wird für die Darstellung der RTT erneut ein Boxplot-Diagramm gewählt. Die Abbildung 7.14 zeigt die Ergebnisse des Versuchs.

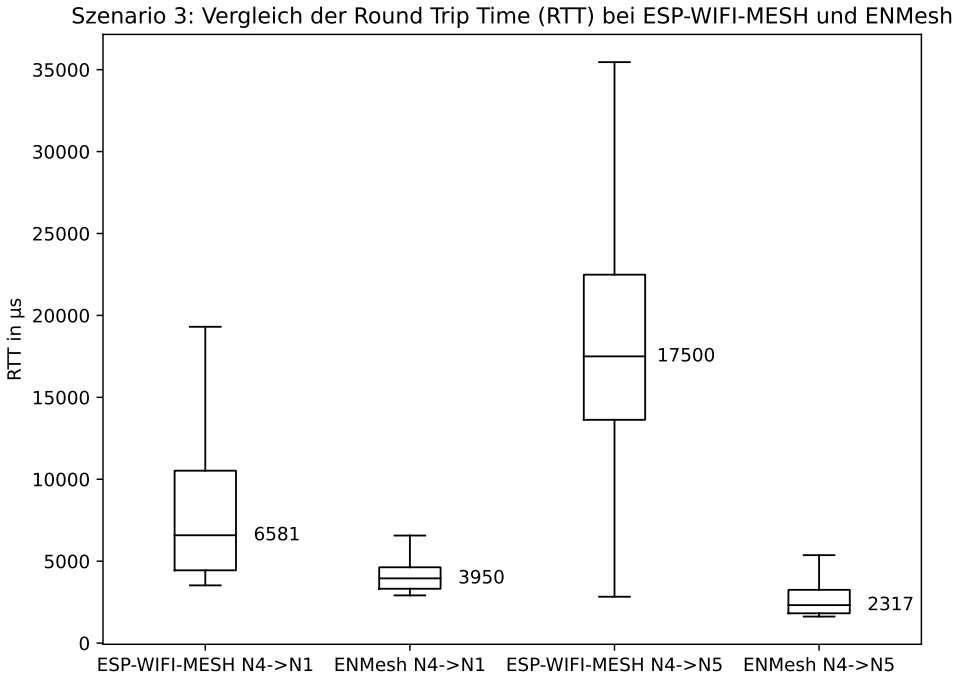

Abbildung 7.14: Szenario 3: Gemessene RTT für die Kommunikation zwischen N4 → N1 und N4 → N5

Dabei lief die Kommunikation zwischen N4 und N1 sowohl bei ESP-WIFI-MESH als auch bei ENMesh konstant über zwei Hops und stellt somit in beiden Fällen eine Multi-Hop-Kommunikation dar. Wie zuvor bereits bei der Single-Hop-RTT im ersten Szenario zu

beobachten war, ist auch hier die RTT bei ENMesh geringer. Dabei lag der Median der RTTs bei ENMesh ca. 40% unterhalb des gemessenen Medians bei ESP-WIFI-MESH.

Die Kommunikation zwischen N4 und N5 lief bei ENMesh konstant über einen Hop, während sie bei ESP-WIFI-MESH über 4 Hops lief. Dabei ist ein erheblicher Unterschied bei der RTT zwischen den beiden Mesh-Netzwerken zu beobachten. Im Vergleich zu ENMesh liegt der Median der RTT bei ESP-WIFI-MESH ca. 7,5 Mal höher. Dieser könnte auf die, durch die Baumtopologie bedingte, höhere Anzahl Hops bei ESP-WIFI-MESH zu erklären sein. Durch die teilvermaschte Topologie bei ENMesh findet eine direkte Kommunikation (Single-Hop) zwischen N4 und N5 statt.

Szenario 3: Datenrate

Die Messung der Datenrate wurde nach der Beschreibung aus Szenario 1 durchgeführt. Dabei wurde die Datenrate von zwei Pfaden untersucht. Wie bereits im vorherigen Versuch stellt N4 einen Sensor, N5 einen Aktor und N1 das Smart-Home-Gateway dar. Zuerst wurde die Datenrate des Pfads von N4 zu N5 untersucht, der die direkte Kommunikation zwischen einem Sensor und Aktor darstellen soll. Der zweite Pfad, N4 bis N1, stellt die Kommunikation zwischen einem Sensor und dem Smart-Home-Gateway dar.

Aus den bereits zuvor erwähnten Gründen wird für die Darstellung der Datenrate erneut ein Boxplot-Diagramm gewählt. Die Abbildungen 7.15 und 7.16 zeigen die Ergebnisse des Versuchs.

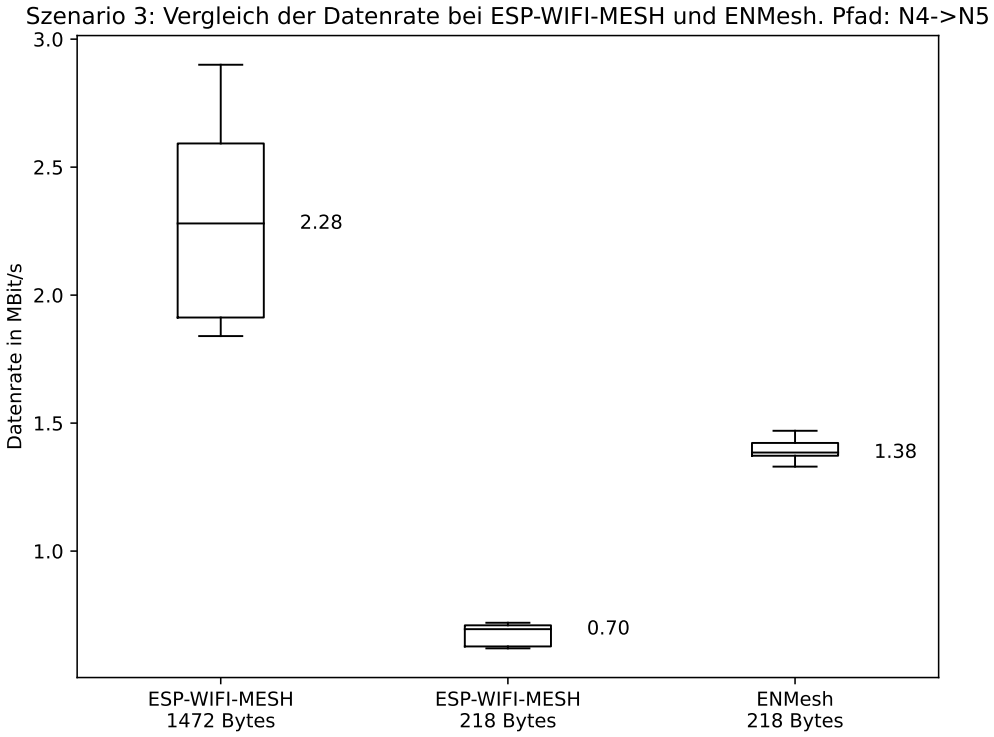

Abbildung 7.15: Szenario 3: Gemessene Datenrate für den Pfad N4 → N5

Die Ergebnisse des Versuchs bei der Kommunikation zwischen N4 und N5 zeigen, dass ESP-WIFI-MESH bei maximaler Payloadgröße eine höhere Datenrate als ENMesh erreicht. Bei gleicher Payloadgröße wie ENMesh ist die Datenrate bei ESP-WIFI-MESH deutlich geringer. Dies könnte auf die Länge des Pfades zurückzuführen sein. Aufgrund der Topologie müssen die Daten bei ESP-WIFI-MESH für die Kommunikation zwischen N4 und N5 über 4 Hops weitergeleitet werden, während die Knoten bei ENMesh direkt kommunizieren können.

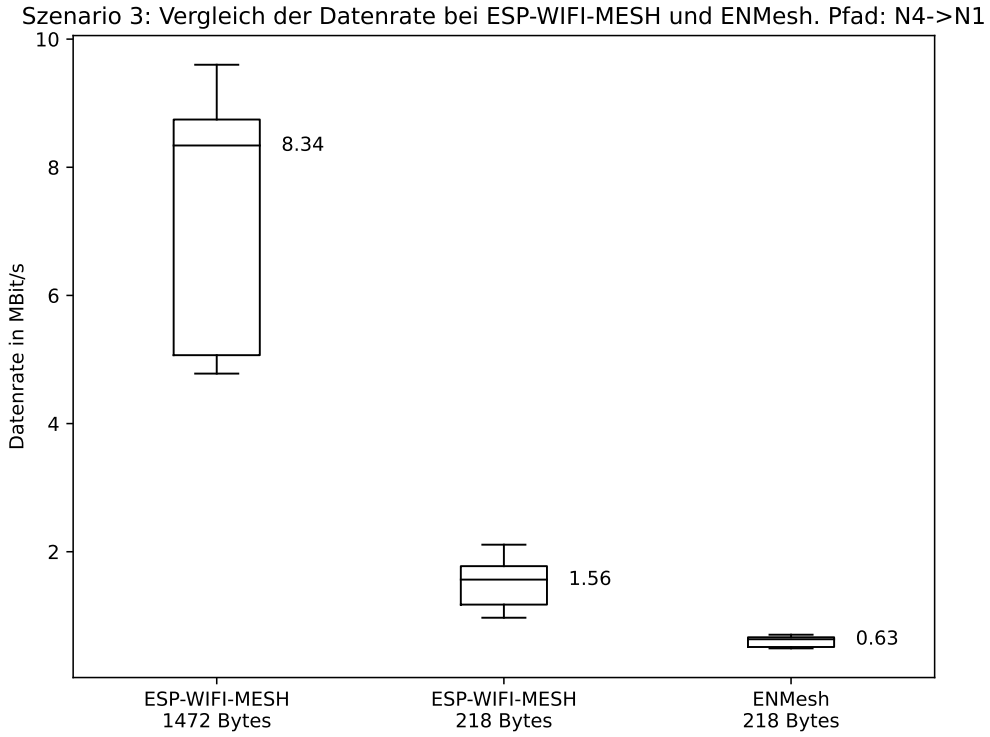

Abbildung 7.16: Szenario 3: Gemessene Datenrate für den Pfad N4 → N1

Die Ergebnisse des Versuchs bei der Kommunikation zwischen N4 und N1 zeigen, dass ESP-WIFI-MESH sowohl bei gleicher Payloadgröße wie ENMesh als auch maximaler Payloadgröße eine höhere Datenrate als ENMesh erreicht. Dabei ist die Ursache für die Differenz bei gleicher Payloadgröße unklar.

Szenario 3: Ausfall eines Knotens

In diesem Versuch wurden zuerst die Auswirkungen des Ausfalls von N1 auf die Kommunikation zwischen N4 und N5 untersucht. Dabei erfolgt die Messung der Ausfalldauer nach der Beschreibung aus Szenario 2. Wie bereits im vorherigen Versuch stellt N4 einen Sensor, N5 einen Aktor und N1 das Smart-Home-Gateway dar. Damit stellt dieser Versuch den Ausfall eines – aus Sicht von N4 und N5 – entfernten Knotens und dessen

Auswirkung auf die Kommunikation zwischen dem Sensor N4 und dem Aktor N5 dar. Die Abbildung 7.17 zeigt die Ergebnisse des Versuchs.

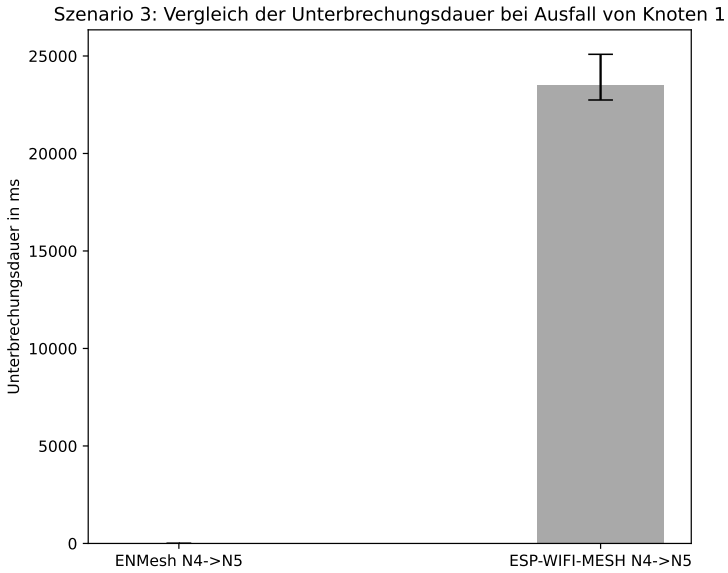

Abbildung 7.17: Szenario 3: Gemessene durchschnittliche Unterbrechungsdauer der Kommunikation von N4 → N5 bei Ausfall von N1

Die Ergebnisse des Versuchs bei Ausfall von N1 während der Kommunikation zwischen N4 und N5 zeigen, dass dieser keine Unterbrechung bei ENMesh verursacht, jedoch bei ESP-WIFI-MESH in einer Unterbrechung von durchschnittlich ca. 22 Sekunden resultiert. Dies ist auf die direkte Verbindung zwischen den Knoten bei ENMesh sowie den längeren Pfad bei ESP-WIFI-MESH, an dem der Knoten N1 beteiligt ist, zurückzuführen.

In einem weiteren Versuch werden die Auswirkungen des Ausfalls von N3 auf die Kommunikation zwischen N5 und N1 untersucht. Der Versuch stellt den Ausfall eines, aus Sicht von N1 und N5, durch Weiterleitung direkt an der Kommunikation beteiligten Knotens dar. Die Abbildung 7.18 zeigt die Ergebnisse des Versuchs.

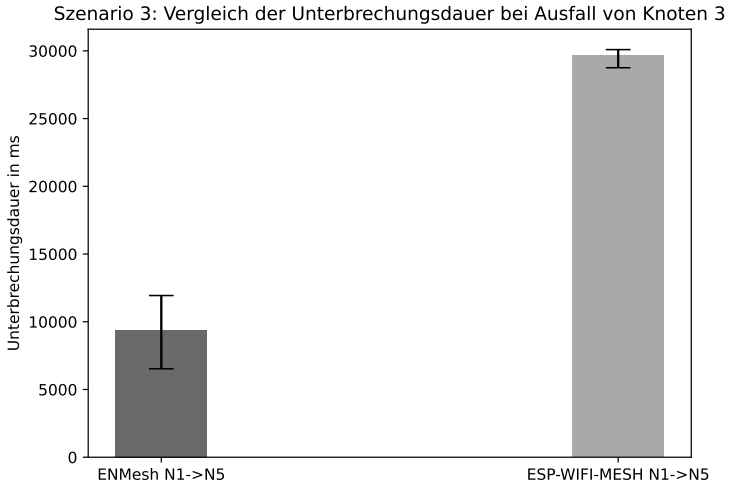

Abbildung 7.18: Szenario 3: Gemessene durchschnittliche Unterbrechungsdauer der Kommunikation von N1 → N5 bei Ausfall von N3

Die Ergebnisse des Versuchs bei Ausfall von N3 während der Kommunikation zwischen N1 und N5 zeigen, dass in beiden Fällen eine Unterbrechung verursacht wird, da sowohl bei ENMesh als auch bei ESP-WIFI-MESH der Knoten N3 die Nachrichten zwischen N1 und N5 weiterleitet. Zudem fällt auf, dass die Unterbrechungsdauer bei ENMesh deutlich geringer ist als bei ESP-WIFI-MESH.

Dies könnte auf die zum Zeitpunkt des Ausfalls bereits vorliegende Routing-Informationen zu einem alternativen Pfad bei ENMesh zurückzuführen sein. Da ESP-WIFI-MESH erst bei Verbindungsverlust zu dem Elternknoten nach einem alternativen Elternknoten sucht, um wieder Teil des Netzwerks zu werden, könnte eine Verzögerung entstehen. ENMesh hingegen erhält regelmäßig Nachrichten mit Routing-Informationen zu allen erreichbaren Zielen über alle direkten Nachbarn. Daher könnte durch die Berechnung der Transmit Quality (TQ) durch BATMAN, der alternative Pfad den zuvor gewählten Pfad durch eine bessere TQ allmählich verdrängen.

8 Fazit

Im Rahmen dieser Bachelorarbeit wurde ein Mesh-Netzwerk für die ESP32-Plattform entwickelt. Dieses stellt eine Alternative zu ESP-WIFI-MESH dar, das von Espressif, dem Hersteller der ESP32-Mikrocontrollerfamilie, bereitgestellt wird.

Bei ESP-WIFI-MESH handelt es sich um ein Netzwerk mit typischen Eigenschaften eines Mesh-Netzwerks, das allerdings eine Baumtopologie einsetzt. Aufgrund der Topologie können die in Funkreichweite befindlichen Knoten nicht immer direkt kommunizieren. Wie in der Evaluation gezeigt wurde, entstehen daher längere Wege zu den Zielen und in Folge dessen eine größere Verzögerung bei der Übertragung. Zudem ist das Netzwerk anfällig für Ausfälle einzelner Knoten, da hierbei auch der Teilbaum unterhalb des ausfallenden Knotens von dem Ausfall betroffen ist. Die zentrale Organisation des Netzwerks durch die Wahl eines Root-Knotens stellt zudem einen Single Point of Failure dar, der bei einem Ausfall auch zu einem kurzfristigen Ausfall des gesamten Netzwerks führt.

Um diese Einschränkungen zu vermeiden, wurde die Entwicklung eines dezentral organisierten Netzwerks mit Mesh-Topologie als Ziel gesetzt. Dazu wurde zuerst eine geeignete Kommunikationstechnologie gewählt, die eine spontane Kommunikation mit allen direkt erreichbaren Nachbarn ermöglicht. Aufgrund der Beschränkungen durch die Plattform sowie der großen Kompatibilität innerhalb der ESP32-Mikrocontrollerfamilie, ist die Wahl auf ESP-NOW gefallen. Anschließend wurde ein geeignetes Routing-Protokoll zur Ermittlung der Pfade sowie der erreichbaren Ziele im Netzwerk gewählt. Dabei stand die kontinuierliche Ermittlung der Routen im Fokus, um die Verzögerung bei Senden einer Nachricht zu verringern. Da es sich bei der ESP32-Plattform um Mikrocontroller handelt, wurde zudem ein Fokus auf den Speicherverbrauch gesetzt. Dabei ist die Wahl auf das BATMAN-Routing-Protokoll in der Version IV gefallen, da es sich um ein proaktives Routing-Protokoll handelt, dessen Routing-Algorithmus den Distanzvektor-Algorithmen ähnlich ist. Auf Grundlage der Kommunikationstechnologie und des Routing-Protokolls wurde ein Protokoll zur Übertragung von Routing-Informationen und Nutzdaten für

das Mesh-Netzwerk entwickelt. Zur Umsetzung der Anforderungen wurde eine Mesh-Netzwerk-Library mit dem Namen ENMesh entwickelt.

In der Evaluation wurden die beiden Mesh-Netzwerke ENMesh und ESP-WIFI-MESH unter den Gesichtspunkten aus der Aufgabenstellung und Analyse bei einem Vergleich in mehreren Szenarien gegenübergestellt. Dabei hat sich gezeigt, dass wie bereits in der Aufgabenstellung vermutet, ENMesh kürzere Wege in einer teil- oder vollvermaschten Topologie bilden kann und damit eine geringere Verzögerung in der Übertragung sowie eine geringere Beteiligung anderer Knoten bei der Weiterleitung von Nachrichten erreichen kann. So wurde im dritten Szenario der Evaluation eine – im Vergleich zu ENMesh – 7,5 Mal höhere RTT bei ESP-WIFI-MESH gemessen. Dabei konnte die Differenz auf einen kürzeren Weg zum Ziel bei ENMesh zurückgeführt werden. Zudem hat sich gezeigt, dass die dezentrale Organisation des Netzwerks bei ENMesh vorteilhaft bei der Startzeit des Netzwerks ist und zudem der Ausfall von Knoten schneller kompensiert werden kann. So wurde bei ESP-WIFI-MESH eine durchschnittlich ca. 9 Mal höhere Startzeit im zweiten Szenario der Evaluation gemessen. Darüber hinaus schneidet ENMesh unter anderem aufgrund der von ESP-NOW vorgegebenen, geringeren maximalen Payload-Größe deutlich schlechter bei der Single- und Multi-Hop-Datenrate ab. Durch die Evaluation wurden die Annahmen aus der Aufgabenstellung und Analyse für Vorteile eines dezentral organisierten Netzwerks mit voll- oder teilvermaschter Topologie bestätigt.

Während der Entwicklung wurden einige Probleme mit ESP-NOW entdeckt, die in dem Unterkapitel 6.2 erläutert wurden. Davon wurden einige im Protokolldesign berücksichtigt, allerdings bleibt Verbesserungsbedarf hinsichtlich Replay-Angriffen und einer Ende-zu-Ende-Verschlüsselung bei der Datenübertragung innerhalb des Mesh-Netzwerks. Zudem könnte für eine höhere Payloadgröße eine alternative Übertragungstechnologie gewählt oder entwickelt werden, um eine höhere Datenrate bei der Übertragung zu erzielen.

Darüber hinaus wäre die Optimierung des Energieverbrauchs für den Einsatz von ENMesh bei batteriebetriebenen Smart-Home-Geräten interessant, die in dieser Bachelorarbeit nicht betrachtet wurde. Zudem könnte weitere Funktionalität für Low-Power-Geräte entwickelt werden.

Literaturverzeichnis

[AHW09] Abolhasan, M. ; Hagelstein, B. ; Wang, J. C.-P.: Real-world performance of current proactive multi-hop mesh protocols. In: *2009 15th Asia-Pacific Conference on Communications*, 2009, S. 44–47

[Aic07] Aichele, Corinna: *Mesh - drahtlose Ad-hoc-Netze*. Open Source Press, 2007. – ISBN 978–3–937514–39–0

[AIM10] Atzori, Luigi ; Iera, Antonio ; Morabito, Giacomo: The Internet of Things: A survey. In: *Computer Networks* 54 (2010), Nr. 15, S. 2787–2805. http://dx.doi.org/10.1016/j.comnet.2010.05.010. – DOI 10.1016/j.comnet.2010.05.010. – ISSN 1389–1286

[AST12] Andrew S. Tanenbaum, David J. W.: *Computernetzwerke, 5., aktualisierte Auflage*. Pearson, Higher Education, 2012. – ISBN 978–3–86894–137–1

[AWD04] Abolhasan, Mehran ; Wysocki, Tadeusz ; Dutkiewicz, Eryk: A review of routing protocols for mobile ad hoc networks. In: *Ad Hoc Networks* 2 (2004), Nr. 1, S. 1–22. http://dx.doi.org/10.1016/S1570-8705(03)00043-X. – DOI 10.1016/S1570–8705(03)00043–X. – ISSN 1570–8705

[AWW05] Akyildiz, Ian F. ; Wang, Xudong ; Wang, Weilin: Wireless mesh networks: a survey. In: *Computer Networks* 47 (2005), Nr. 4, S. 445–487. http://dx.doi.org/10.1016/j.comnet.2004.12.001. – DOI 10.1016/j.comnet.2004.12.001. – ISSN 1389–1286

[Espa] Espressif Systems (Shanghai) Co., Ltd.: *ESP-IDF Programming Guide: ESP-NOW*, https://docs.espressif.com/projects/esp-idf/en/v4.4.2/esp32/api-reference/network/esp_now.html, Abruf: 02.09.2022

[Espb] Espressif Systems (Shanghai) Co., Ltd.: *ESP-IDF Programming Guide: ESP-WIFI-MESH*, https://docs.espressif.com/projects/esp-idf/en/v4.4.2/esp32/api-guides/esp-wifi-mesh.html, Abruf: 25.09.2022

[Espc] Espressif Systems (Shanghai) Co., Ltd.: *ESP-IDF Programming Guide: WiFi Driver*. https://docs.espressif.com/projects/esp-idf/en/v4.4.3/esp32/api-guides/wifi.html, Abruf: 30.11.2022

[Espd] Espressif Systems (Shanghai) Co., Ltd.: *ESP-NOW: Espressif's Wireless Communication Protocol*. https://www.espressif.com/en/news/ESP-NOW, Abruf: 18.09.2022

[Espe] Espressif Systems (Shanghai) Co., Ltd.: *ESP-NOW User Guide*, https://www.espressif.com/sites/default/files/documentation/esp-now_user_guide_en.pdf, Abruf: 19.09.2022

[Espf] Espressif Systems (Shanghai) Co., Ltd.: *ESP-WIFI-MESH*. https://www.espressif.com/en/products/sdks/esp-wifi-mesh/overview, Abruf: 25.09.2022

[Espg] Espressif Systems (Shanghai) Co., Ltd.: *ESP32-C3-WROOM-02 Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32-c3-wroom-02_datasheet_en.pdf, Abruf: 27.11.2022

[Esph] Espressif Systems (Shanghai) Co., Ltd.: *ESP32 Series Datasheet*, https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf, Abruf: 07.10.2022

[Espi] Espressif Systems (Shanghai) Co., Ltd.: *Espressif SoCs*. https://www.espressif.com/en/products/socs, Abruf: 22.11.2022

[FdRG11] Friginal, Jesús ; de Andrés, David ; Ruiz, Juan-Carlos ; Gil, Pedro: Towards benchmarking routing protocols in wireless mesh networks. In: *Ad Hoc Networks* 9 (2011), Nr. 8, S. 1374–1388. http://dx.doi.org/10.1016/j.adhoc.2011.03.010. – DOI 10.1016/j.adhoc.2011.03.010. – ISSN 1570–8705. – Recent advances on practical aspects of Wireless Mesh Networks

[Fla] Flayols, Thomas: *How ESP-NOW really works ?* https://hackaday.io/project/161896-linux-espnow/log/154698-how-esp-now-really-works, Abruf: 19.09.2022

[Gas05] Gast, Matthew S.: *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, Inc., 2005. – ISBN 0596100523

[GSV11] Gupta, Anuj ; Sadawarti, Harsh ; Verma, Anil: Review of Various Routing Protocols for MANETs. In: *International Journal of Information and Electronics Engineering* 1 (2011), 01, S. 251–259. http://dx.doi.org/10.7763/IJIEE.2011.V1.40. – DOI 10.7763/IJIEE.2011.V1.40

[Hel05] Held, Gilbert: *Wireless Mesh Networks*. Auerbach Publications, 2005. – ISBN 978–0–8493–2960–9

[IEE97] IEEE: IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. In: *IEEE Std 802.11-1997* (1997). http://dx.doi.org/10.1109/IEEESTD.1997.85951. – DOI 10.1109/IEEESTD.1997.85951

[IEE21] IEEE: IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. In: *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)* (2021). http://dx.doi.org/10.1109/IEEESTD.2021.9363693. – DOI 10.1109/IEEESTD.2021.9363693

[Jin22] Jindřich Šesták: *Dynamic Mesh Network Implemented in Micropython on Top of ESP-NOW Protocol*, Vysoké učení technické v Brně, Fakulta informačních technologií, Diplomová práce, 2022. https://www.fit.vut.cz/study/thesis/24845/

[JSAT15] J, Pramod ; S, Sahana K. ; A, Akshay ; Talasila, Viswanath: Characterization of Wireless Mesh Network performance in an experimental test bed. In: *2015 IEEE International Advance Computing Conference (IACC)*, 2015, S. 910–914

[KS19] Khanchuea, Kanitkorn ; Siripokarpirom, Rawat: A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation: Design and Implementation. In: *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, 2019, S. 1–6

[LEGA21] Labib, Mostafa I. ; ElGazzar, Mohamed ; Ghalwash, Atef ; AbdulKader, Sarah N.: An efficient networking solution for extending and controlling wireless sensor networks using low-energy technologies. In: *PeerJ Computer Science* 7 (2021), November, S. e780. http://dx.doi.org/10.7717/peerj-cs.780. – DOI 10.7717/peerj–cs.780

[LR21] Lasquety-Reyes, Jeremiah: Statista Smart Home Report 2021 - Market Report / Statista. Version: 2021. https://www.statista.com/study/42112/smart-home-report. 2021. – Forschungsbericht

[LTQ+17] Liu, Yu ; Tong, Kin-Fai ; Qiu, Xiangdong ; Liu, Ying ; Ding, Xuyang: Wireless Mesh Networks in IoT networks. In: *2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition*, 2017, S. 183–185

[NAL07] Neumann, Axel ; Aichele, Corinna ; Lindner, Marek: *B.A.T.M.A.N Status Report*. https://downloads.open-mesh.org/batman/papers/batman-status.pdf. Version: 2007, Abruf: 31.10.2022

[NALW08] Neumann, Axel ; Aichele, Corinna ; Lindner, Marek ; Wunderlich, Simon: Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) / Internet Engineering Task Force. Version: 2008. https://datatracker.ietf.org/doc/draft-wunderlich-openmesh-manet-routing/00/. Internet Engineering Task Force, 2008 (draft-wunderlich-openmesh-manet-routing-00). – Internet-Draft. – Work in Progress

[Opea] Open-Mesh Project: *BATMAN IV*. https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_IV, Abruf: 02.09.2022

[Opeb] Open-Mesh Project: *BATMAN V*. https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_V, Abruf: 02.09.2022

[PZOW16] Pieciak, Maciej ; Zwierzykowski, Piotr ; Owczarek, Piotr ; Wasłowicz, Michał: Comparative analysis of routing protocols for wireless mesh networks. In: *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2016, S. 1–5

[Rec12] Rech, Jörg: *Wireless LANs - 802.11-WLAN-Technologie und praktische Umsetzung im Detail, 4., aktualisierte und erweiterte Auflage*. Heise, 2012. – ISBN 978–3–936931–75–4

[Res18] Research, Bitkom: Home Smart Home: Jeder Vierte ist auf dem Weg zum intelligenten Zuhause / Bitkom Research. Version: 2018. https://www.bitkom.org/Presse/Presseinformation/Home-Smart-Home-Jeder-Vierte-ist-auf-dem-Weg-zum-intelligenten-Zuhause.html, Abruf: 07.10.2022. 2018. – Forschungsbericht

[SBP08] Sarkar, Subir K. ; Basavaraju, T. G. ; Puttamadappa, C.: *Ad Hoc Mobile Wireless Networks: Principles, Protocols, and Applications*. CRC Press, Inc., 2008. – ISBN 1466514469

[WL] Wunderlich, Simon ; Lindner, Marek: *Wireless Kernel Tweaking*. https://media.ccc.de/v/24c3-2292-en-wireless_kernel_tweaking, Abruf: 30.10.2022

[Wä18] Wätjen, Dietmar: *Kryptographie: Grundlagen, Algorithmen, Protokolle, 3. Auflage*. Springer Vieweg, 2018. http://dx.doi.org/10.1007/978-3-658-22474-5. http://dx.doi.org/10.1007/978-3-658-22474-5. – ISBN 978–3–658–22473–8

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original