Universität Rostock

Traditio et Innovatio

Cross-Layer-Optimierungen für WLAN-Mesh-Netzwerke

Der Fakultät für Informatik und Elektrotechnik der Universität Rostock
zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr.-Ing.)

vorgelegte Dissertation von
Michael Rethfeldt
geb. am 14.06.1987 in Rostock

Rostock, 26. Februar 2021

Gutachter:

Prof. Dr.-Ing. Dirk Timmermann
Institut für Angewandte Mikroelektronik und Datentechnik
Universität Rostock
Richard-Wagner-Str. 31
18119 Rostock-Warnemünde

Prof. Dr.-Ing. habil. Falko Dressler
Institut für Telekommunikationssysteme
Technische Universität Berlin
Einsteinufer 25
10587 Berlin

Dissertation
Fakultät für Informatik und Elektrotechnik

Michael Rethfeldt
Institut für Angewandte Mikroelektronik und Datentechnik

https://doi.org/10.18453/rosdok_id00002999

Gutachter:

- Prof. Dr.-Ing. Dirk Timmermann (Universität Rostock, Fakultät für Informatik und Elektrotechnik, Institut für Angewandte Mikroelektronik und Datentechnik)

- Prof. Dr.-Ing. habil. Falko Dressler (Technische Universität Berlin, Fakultät für Elektrotechnik und Informatik, Institut für Telekommunikationssysteme)

Tag der Einreichung: 28.09.2020
Tag der Verteidigung: 26.02.2021

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die Dissertation mit dem Titel

Cross-Layer-Optimierungen für WLAN-Mesh-Netzwerke

selbständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die den verwendeten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Rostock, den 28. September 2020

_____

Michael Rethfeldt

Danksagung

Während der Entstehung der vorliegenden Arbeit, zunächst im Rahmen des DFG-Graduiertenkollegs MuSAMA und später als Mitarbeiter am Institut für Angewandte Mikroelektronik und Datentechnik der Universität Rostock, haben mich viele Menschen begleitet und unterstützt. Ich möchte an dieser Stelle die Gelegenheit nutzen, meinen Dank auszusprechen.

Mein besonderer Dank gilt meinem Doktorvater Herrn Prof. Dr.-Ing. Dirk Timmermann, der mich seit Beginn meiner Tätigkeit am Institut für Angewandte Mikroelektronik und Datentechnik kontinuierlich unterstützt und bestärkt hat. Das entgegengebrachte Vertrauen in die gewählte Thematik sowie seine kritischen und motivierenden Hinweise haben maßgeblich zum Gelingen der Arbeit beigetragen. Weiterhin gilt mein Dank Herrn Prof. Dr.-Ing. habil. Falko Dressler von der TU Berlin für sein großes Engagement als externer Gutachter.

Darüber hinaus möchte ich den Mitarbeiterinnen und Mitarbeitern des Instituts für Angewandte Mikroelektronik und Datentechnik der Universität Rostock danken, die nicht zuletzt aufgrund des regen Austauschs, des kollegialen Zusammenhalts und der hohen Bereitschaft zur gegenseitigen Hilfestellung für eine hervorragende Arbeitsatmosphäre sorgen. Insbesondere danke ich meinen Kollegen Benjamin Beichler, Björn Konieczek, Peter Danielis und Tim Brockmann für ihre wertvolle Unterstützung, sei es durch fachliche Diskussionen und Anregungen oder auch das Korrekturlesen der Arbeit. Darüber hinaus danke ich allen Kollegen und ehemaligen Studenten, die an dieser Arbeit beteiligt waren. Hier möchte ich Felix Uster, Hannes Raddatz und Arne Wall stellvertretend hervorheben.

Besonders danke ich auch meinen Eltern Heike und Uwe Rethfeldt, meinem Bruder Carsten Rethfeldt und meiner Lebenspartnerin Pia Wilsdorf für ihre vielfältige Unterstützung in allen Lebenslagen.

Kurzreferat

Mesh-Netzwerke auf Basis der WLAN-Technologie (Wireless Local Area Network) bieten im Vergleich zu herkömmlichen, zentralisierten WLAN-Installationen eine erhöhte Flexibilität und Ausfallsicherheit. Durch die Fähigkeit der Spontanvernetzung und Datenweiterleitung über mehrere Zwischenstationen ermöglichen sie eine robuste und kostengünstige Abdeckung großer Areale, z. B. in Form von Zugangsnetzen, Backbones oder Service-Infrastruktur in Smart-City-Szenarien.

Der Trend hin zu proprietären, zueinander inkompatiblen Mesh-Lösungen behindert jedoch derzeit ihren praktischen Einsatz im großen Maßstab. Für den Langzeitbetrieb komplexer Mesh-Installationen ist eine flexible Austauschbarkeit von Geräten und somit die Überwindung der Herstellerbindung zwingend erforderlich. Eine logische Konsequenz ist die Etablierung von Standardtechnologien, die Kompatibilität bereits auf tiefster Ebene garantieren.

Die WLAN-Standarderweiterung IEEE 802.11s integriert erstmals Mesh-Funktionalität direkt in die WLAN-Sicherungsschicht (MAC Layer) und liefert damit eine vielversprechende Basis für interoperable Mesh-Lösungen. Oberhalb der standardisierten Grundfunktionen existieren jedoch zahlreiche Anforderungen und Problemstellungen im Hinblick auf die skalierbare Orchestrierung und Verwaltung des Netzwerks sowie die effiziente Nutzung der Kommunikationsressourcen durch übergeordnete Protokolle und Anwendungen, die durch IEEE 802.11s nicht behandelt werden.

Gegenstand dieser Arbeit ist es, das Verhalten von IEEE-802.11s-Mesh-Netzwerken in der Praxis zu untersuchen und Strategien und Lösungen zu entwickeln, durch die einerseits die Administrierbarkeit und Skalierbarkeit komplexer Mesh-Backbones erhöht werden und andererseits verteilte Anwendungen die darunter liegende Netzwerkstruktur gezielt berücksichtigen können, um das vorhandene Datendurchsatzpotential effizient zu nutzen. Übergeordnetes Ziel ist der Entwurf standardkonformer Lösungen, welche die durch IEEE 802.11s bereitgestellten Mechanismen integrieren und somit keine Abhängigkeiten von Spezial-Hardware oder proprietären Mesh-Protokollen aufweisen.

Dazu werden in dieser Forschungsarbeit zwei dezentrale Cross-Layer-Ansätze entwickelt, die Verbindungsinformationen der WLAN-Sicherungsschicht auch auf der Anwendungsschicht berücksichtigen. Dort werden diese genutzt, um Kommunikationsentscheidungen zu optimieren und die Netzwerkstruktur durch Partitionierung und Kanalselektion über vorhandene Konfigurationsschnittstellen geeignet anzupassen. Die Lösungen werden prototypisch implementiert und in einer realen Testumgebung untersucht, die im Rahmen der Arbeit entwickelt wurde. Diese erlaubt es durch einen Miniaturisierungsansatz, Mesh-Szenarien der Fläche von mehr als $300.000\,m^2$ auch im Labormaßstab abzubilden. Die Ergebnisse der Arbeit belegen die Praktikabilität und Wirksamkeit der Ansätze, welche eine nennenswerte Steigerung der Kommunikationseffizienz erzielen, ohne dabei Änderungen an den IEEE-802.11s-Standardmechanismen vorzunehmen.

Abstract

Mesh networks based on the WLAN (Wireless Local Area Network) technology offer an increased flexibility and reliability compared to conventional centralized WLAN installations. Due to the ability of spontaneous interconnection and data forwarding over multiple intermediate nodes they enable a robust and cost-effective coverage of large areas in the form of wireless access networks, backbones, or service infrastructure in smart city scenarios.

However, the trend towards proprietary and incompatible mesh solutions currently hinders their practical use on a large scale. For the long-term operation of complex mesh installations it is essential to overcome the vendor lock-in to allow for a flexible exchangeability of devices. A logical consequence is the establishment of standard technologies that guarantee compatibility at the lowest level.

The WLAN standard amendment IEEE 802.11s introduces mesh functionality to the WLAN MAC layer and provides a promising basis for interoperable mesh solutions. However, beyond the standardized functionality, there are numerous requirements and problems with regard to scalable network orchestration and administration as well as the efficient use of communication resources by overlying protocols and applications that are not covered by IEEE 802.11s.

The aim of this thesis is to investigate the practical behavior of IEEE 802.11s mesh networks and to develop strategies and solutions that, on the one hand, increase the scalability and manageability of complex mesh backbones and, on the other hand, enable distributed applications to explicitly consider the underlying network structure, allowing them to utilize the available network capacity efficiently. The overall goal is to design standard-compliant solutions that integrate the mechanisms provided by IEEE 802.11s and avoid any dependencies on specialized hardware or proprietary mesh protocols.

For this purpose, two decentralized cross-layer approaches are developed in this thesis, which consider MAC-layer connection information at the application layer. There they are used to optimize communication decisions and to adapt the network structure appropriately by means of partitioning and channel selection, using only available configuration interfaces. The solutions are implemented prototypically and investigated in a real testbed developed in the course of this work. Based on a miniaturization approach the testbed allows to reproduce mesh network scenarios of the area of more than $300,000\,m^2$ also on a laboratory scale. The results prove the practicability and effectiveness of the approaches, which notably increase communication performance without modifying the IEEE 802.11s standard mechanisms.

Inhaltsverzeichnis

Abbildungsverzeichnis **xv**

Tabellenverzeichnis **xvii**

Abkürzungsverzeichnis **xix**

1 Einleitung **1**
1.1 Problemstellung und Zielsetzungen der Arbeit . . . 3
1.2 Aufbau und Zitierweise der Arbeit . . . 5

2 Grundlagen **7**
2.1 ISO/OSI- und TCP/IP-Referenzmodell . . . 7
2.2 Drahtlose Mesh-Netzwerke . . . 9
2.2.1 Architektur . . . 9
2.2.2 Mesh Routing . . . 10
2.3 Gegenüberstellung von WLAN-Mesh-Lösungen . . . 11
2.3.1 Offene Protokolle . . . 11
2.3.2 Proprietäre Lösungen . . . 14
2.4 WLAN-Standard IEEE 802.11 . . . 14
2.4.1 Übersicht der Teilstandards . . . 15
2.4.2 WLAN-Betriebsmodi . . . 17
2.4.3 Nachrichtentypen und -formate . . . 18
2.4.4 Frequenzspektrum und Datenraten . . . 19
2.4.5 Medienzugriff und Flusskontrolle . . . 21
2.5 WLAN-Mesh-Erweiterung IEEE 802.11s . . . 25
2.5.1 Übersicht der Mesh-Funktionen . . . 25
2.5.2 Knotenrollen . . . 26
2.5.3 Medienzugriff . . . 26
2.5.4 Discovery und Verbindungsaufbau . . . 26
2.5.5 Pfadselektion mittels HWMP/ALM . . . 28
2.5.6 Adressierung und Frame-Weiterleitung . . . 30
2.5.7 Implementierung im Linux-Kernel . . . 31

3 Entwurf einer realen WLAN-Mesh-Testumgebung **35**
3.1 Motivation . . . 35
3.2 Stand der Technik . . . 36
3.3 Miniaturisierte IEEE-802.11n/s-Testumgebung *Mini-Mesh* . . . 37
3.3.1 Anforderungsbeschreibung und Auswahl der Geräteplattform . . . 37
3.3.2 Zusammensetzung und Geometrie des Aufbaus . . . 38
3.3.3 Performance-Analyse der Geräteplattform . . . 41
3.3.4 Konzept zur Reichweitenskalierung . . . 45
3.3.5 Umsetzung des Skalierungskonzepts . . . 48
3.4 Validierung des Skalierungsansatzes . . . 52
3.4.1 Vergleich des TCP-/UDP-Durchsatzes . . . 52
3.4.2 Vergleich der Kostenmetrik ALM . . . 54
3.5 Zwischenfazit . . . 55

4 Kollaborativer Datenaustausch in IEEE-802.11s-Netzwerken **57**
4.1 Motivation . . . 57

4.2 P2P-Protokoll BitTorrent . . . 59
4.2.1 Grundprinzip . . . 59
4.2.2 Choking-Algorithmus zur Peer-Auswahl . . . 61
4.3 Stand der Technik . . . 61
4.4 Cross-Layer-Optimierung *MeNTor* . . . 65
4.4.1 Ausgangsszenario . . . 65
4.4.2 Optimierter Algorithmus zur Peer-Auswahl . . . 66
4.5 Prototypische Umsetzung . . . 72
4.6 Evaluation in der Testumgebung . . . 74
4.6.1 Gerätekonfiguration . . . 74
4.6.2 Netzwerkszenarien und Testfälle . . . 75
4.6.3 *MeNTor*-Konfigurationsvarianten . . . 75
4.6.4 Experimentvorbereitung und -ablauf . . . 76
4.6.5 Ergebnisübersicht für die Kombination aller Optimierungen . . . 79
4.6.6 Ergebnisse bei Worst-Case-Platzierung des initialen Seeds . . . 82
4.6.7 Ergebnisse bei Best-Case-Platzierung des initialen Seeds . . . 84
4.7 Zwischenfazit . . . 87

5 Clustering und Kanalwahl in IEEE-802.11s-Netzwerken 89
5.1 Motivation . . . 89
5.2 Stand der Technik . . . 90
5.3 Clustering-Verfahren *CHaChA* . . . 93
5.3.1 Überblick und Terminologie . . . 93
5.3.2 Metriken, Kontrollnachrichten und Parameter . . . 96
5.3.3 Initiale Cluster-Bildung und Kanalzuweisung . . . 98
5.3.4 Cluster-Anpassung zur Laufzeit . . . 103
5.4 Prototypische Umsetzung . . . 111
5.5 Evaluation in der Testumgebung . . . 113
5.5.1 Gerätekonfiguration . . . 114
5.5.2 Clustering-Ergebnisse in statischen Topologien . . . 114
5.5.3 Erprobung der dynamischen Cluster-Anpassung . . . 127
5.5.4 Performance-Vorteile am Anwendungsbeispiel . . . 131
5.6 Zwischenfazit . . . 135

6 Zusammenfassung 137
6.1 Cross-Layer-Optimierungsansätze für WLAN-Mesh-Netzwerke . . . 137
6.2 Ausblick . . . 139

Literaturverzeichnis I

A Liste der Veröffentlichungen und Fachvorträge auf Tagungen XIX

B Liste der betreuten studentischen Arbeiten XXIII

Abbildungsverzeichnis

1.1 WLAN-Mesh-Netzwerk für Smart-City-Anwendungen . . . 2
1.2 Cross-Layer-Optimierungsvarianten . . . 4
2.1 ISO/OSI-Modell und hybrides TCP/IP-Modell [42] . . . 8
2.2 Architekturbeispiel für ein Backbone-WMN [44] . . . 10
2.3 Realisierungsvarianten für WLAN-Mesh-Netzwerke . . . 12
2.4 Übersicht der WLAN-Betriebsmodi . . . 17
2.5 Allgemeines WLAN-Frame-Format [5] . . . 18
2.6 Überlappungsfreie WLAN-Kanäle im 2,4- und 5-GHz-Band (Europa) [99] . . . 20
2.7 Beispiel-Datenübertragung mit EDCA [16] . . . 24
2.8 Hidden- und Exposed-Node-Problem [42] . . . 24
2.9 Übersicht der Hauptbestandteile von IEEE 802.11s [B 17] . . . 25
2.10 Mesh Configuration Element in Beacon- und Probe-Nachrichten [5] . . . 27
2.11 Mesh Peering mittels 4-Wege-Handshake . . . 27
2.12 Beispiel einer HWMP-Pfadermittlung . . . 29
2.13 6-Adress-Modus am Beispiel [114] . . . 31
2.14 Architektur der WLAN-Implementierung unter Linux . . . 31
3.1 Intel Galileo Board . . . 38
3.2 Testumgebung im Labor . . . 40
3.3 Geometrie und Abmessungen der Testumgebung [B 6] . . . 40
3.4 Leistungs- und Dämpfungskomponenten der Funkstrecke . . . 46
3.5 Indoor- und Outdoor-Reichweitenmessungen [B 6] . . . 50
3.6 TCP/UDP-Durchsatz im skalierten und unskalierten Aufbau [B 6] . . . 53
3.7 ALM im skalierten und unskalierten Aufbau . . . 54
4.1 Kollaborative Datenausbringung im Smart-City-Szenario . . . 58
4.2 BitTorrent-Grundprinzip . . . 60
4.3 Mismatch zwischen P2P-Overlay und Mesh-Underlay [B 17] . . . 66
4.4 Cross-Layer-Optimierungsansatz *MeNTor* [B 4] . . . 67
4.5 Pseudo-Code der BitTorrent-Peer-Auswahl inkl. *MeNTor*-Optimierungen . . . 68
4.6 Software-Architektur der *MeNTor*-Implementierung . . . 73
4.7 Testfälle für Szenario 1 [B 4] . . . 76
4.8 Testfälle für Szenario 2 [B 4] . . . 76
4.9 Mittlere Download-Zeit bei Kombination aller Optimierungen [B 4] . . . 80
4.10 Zeitverlauf der Empfangsdatenraten des Knotens G25 im Testfall 1e [B 4] . . . 81
4.11 Ergebnisse für Szenario 1 [B 4] . . . 83
4.12 Ergebnisse für Szenario 2 [B 4] . . . 85
5.1 Verteiltes Status-Monitoring im Smart-City-Szenario . . . 90
5.2 Klassifikation von Kanalselektionsverfahren nach [39] . . . 91
5.3 Phasenablauf der initialen Cluster-Bildung und Kanalzuweisung [B 3] . . . 95
5.4 Zwischenergebnisse der Phasen im Beispielnetzwerk . . . 100
5.5 Zustandsdiagramm der Mechanismen zur dynamischen Cluster-Anpassung . . . 105
5.6 Roaming-Einschränkungen am Beispielnetzwerk . . . 108
5.7 Sequenzdiagramm für das Roaming zwischen Clustern . . . 108
5.8 Balancierung am Beispielnetzwerk . . . 110
5.9 Software-Architektur der *CHaChA*-Implementierung . . . 111
5.10 Zeitkomponenten zur Abschätzung der Clustering-Dauer . . . 115
5.11 Geschätzte Clustering-Dauer für Parametrisierung P1 . . . 116
5.12 Cluster-Konstellationen im 5x5-Knoten-Gitter für Parametrisierung P1 [B 3] . . . 119
5.13 Cluster-Konstellationen im 2x2-Knoten-Gitter für Parametrisierung P2 . . . 121
5.14 Cluster-Konstellationen im 3x3-Knoten-Gitter für Parametrisierung P2 . . . 123

5.15 Cluster-Konstellationen im 4x4-Knoten-Gitter für Parametrisierung P2 124
5.16 Cluster-Konstellationen im 5x5-Knoten-Gitter für Parametrisierung P2 125
5.17 Kommunikations-Overhead und Clustering-Dauer 126
5.18 Cluster-Balancierung im 5x5-Knoten-Gitter 128
5.19 Virtuelle Nachinstallation und Umplatzierung eines Knotens 129
5.20 Verglichene Monitoring-Szenarien 132
5.21 Monitoring-Dauer im zentralisierten Referenzszenario [B 3] 133
5.22 Vergleich von zentralisiertem und dezentralem Monitoring [B 3] 134

Tabellenverzeichnis

2.1 Übersicht offener Mesh-Routing-Protokolle . . . 13
2.2 Übersicht der Überarbeitungsdokumente des IEEE-802.11-Standards [104] . . . 16
2.3 Datenrate der Bitübertragungsschicht für IEEE 802.11n (HT MCS 0–15) [109] . . . 22
2.4 Standardwerte der EDCA-Parameter für 802.11n im 5-GHz-Band [16] . . . 23
2.5 Übersicht wichtiger Mesh-Parameter [5, 127] . . . 32
3.1 Gegenüberstellung von *Mini-Mesh* mit verwandten Forschungsarbeiten [B 6] . . . 36
3.2 Hardware- und Software-Eigenschaften der Testumgebung [B 6] . . . 39
3.3 Transceiver-Eigenschaften des WLAN-Moduls Compex WLE200NX [B 6] . . . 42
3.4 TCP/UDP-Datendurchsatz eines Mesh-Links und erzeugte CPU-Auslastung [B 6] . . . 44
3.5 Ergebnisse der Indoor (ID)- und Outdoor (OD)-Reichweitenmessungen [B 6] . . . 51
4.1 Übersicht der BitTorrent-Nachrichten . . . 60
4.2 Gegenüberstellung von *MeNTor* mit verwandten Forschungsarbeiten [B 4] . . . 62
4.3 WLAN-Parameter der Testumgebung [B 4] . . . 75
4.4 Übersicht der *MeNTor*-Konfigurationen [B 4] . . . 77
5.1 Gegenüberstellung von *CHaChA* mit verwandten Forschungsarbeiten [B 3] . . . 93
5.2 Knotenrollen in *CHaChA* [B 3] . . . 94
5.3 Metriken in *CHaChA* . . . 96
5.4 Unicast (UC)- und Broadcast (BC)-Nachrichten in *CHaChA* [B 3] . . . 98
5.5 Timing-Parameter und Schwellenwerte in *CHaChA* [B 3] . . . 99
5.6 Aufbau der Unicast (UC)- und Broadcast (BC)-Nachrichten . . . 113
5.7 *CHaChA*-Parametrisierungsvariante P1 . . . 118
5.8 *CHaChA*-Parametrisierungsvariante P2 . . . 121
5.9 Vergleich von Clustering-Dauer und Kommunikations-Overhead . . . 126

Abkürzungsverzeichnis

AC	Access Category
ACI	Adjacent Channel Interference
ACK	Acknowledgement
AIFS	Arbitration Inter-Frame Space
ALM	Airtime Link Metric
ALTO	Application-Layer Traffic Optimization
ANI	Ambient Noise Immunity
AODV	Ad-Hoc On-Demand Distance Vector Routing
AP	Access Point
API	Application Programming Interface
ARP	Address Resolution Protocol
ARQ	Automatic Repeat ReQuest
BATMAN	Better Approach to Mobile Ad-Hoc Networking
BE	Best Effort
BEB	Binary Exponential Back-Off
BEP	BitTorrent Enhancement Proposal
BPSK	Binary Phase Shift Keying
BSS	Basic Service Set
BT	BitTorrent
CCA	Clear Channel Assessment
CCI	Co-Channel Interference
CENT	Centrality
CFN	Cluster-Free Node
CH	Cluster Head
CHaChA	Clustering Heuristic and Channel Assignment
CM	Cluster Member
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSA	Channel Switch Announcement
CSMA	Carrier-Sense Multiple Access
CSMA/CA	Carrier-Sense Multiple Access / Collision Avoidance
CSMA/CD	Carrier-Sense Multiple Access / Collision Detection
CTS	Clear-to-Send
CW	Contention Window
CWND	Congestion Window
DCF	Distributed Coordination Function
DFS	Dynamic Frequency Selection
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DIFS	Distributed Coordination Function Inter-Frame Space
DNS	Domain Name System
DS	Distribution System
DSDV	Destination-Sequenced Distance Vector Routing
DSR	Dynamic Source Routing
DSSS	Direct Sequence Spread Spectrum

EBSS Extended Basic Service Set
ED Energy Detection
EDCA Enhanced Distributed Channel Access
EIFS Extended Inter-Frame Space
EIRP Equivalent Isotropic Radiated Power
ETT Expected Transmission Time
ETX Expected Transmission Count

FCS Frame Check Sequence
FEC Forward Error Correction
FHSS Frequency Hopping Spread Spectrum
FTP File Transfer Protocol

GANN Gate Announcement
GATS Group Addressed Transmission Service
GI Guard Interval

HCCA HCF Controlled Channel Access
HCF Hybrid Coordination Function
HT High Throughput
HTTP Hypertext Transfer Protocol
HWMP Hybrid Wireless Mesh Protocol

IBSS Independent Basic Service Set
ICMP Internet Control Message Protocol
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
IFS Inter-Frame Space
IoT Internet of Things
IP Internet Protocol
ISM Industrial, Scientific and Medical
ISO International Organization for Standardization
ISP Internet Service Provider
ITU International Telecommunication Union

JVM Java Virtual Machine

KV Konfigurationsvariante

LAN Local Area Network
LoS Line-of-Sight
LPWAN Low Power Wide Area Network
LTE Long Term Evolution

MAC Media Access Control
MANET Mobile Ad-Hoc Network
MBSS Mesh Basic Service Set
MCCA MCF Controlled Channel Access
MCF Mesh Coordination Function
MCH Master Cluster Head
MCS Modulation and Coding Scheme

MeNTor Mesh-Network-Aware BitTorrent
MIMO Multiple Input Multiple Output
MSS Maximum Segment Size
MTU Maximum Transmission Unit

NAV Network Allocation Vector
NC Neighbor Count
NH2CH Next-Hop-to-Cluster-Head
NIC Network Interface Card
NLoS Non-Line-of-Sight
NP Neighbor Preference
NPR NC-to-PCHNC-Ratio
NTP Network Time Protocol

OFDM Orthogonal Frequency Division Multiplexing
OLSR Optimized Link State Routing
OS Operating System
OSI Open Systems Interconnection

P2P Peer-to-Peer
PC Personal Computer
PCAP Packet Capture
PCF Point Coordination Function
PCH Proposed Cluster Head
PCHNC Proposed Cluster Head Neighbor Count
PCI Peripheral Component Interconnect
PCIe PCI-Express
PDU Protocol Data Unit
PERR Path Error
PEX Peer Exchange
PREP Path Reply
PREQ Path Request
PTP Precision Time Protocol

QAM Quadrature Amplitude Modulation
QoE Quality of Experience
QoS Quality of Service
QPSK Quadrature Phase Shift Keying

RAM Random Access Memory
RANN Root Announcement
RCA Rate Control Algorithm
RFC Request for Comments
RSSI Received Signal Strength Indicator
RTS Request-to-Send
RTT Round Trip Time
RX Receive

SAE Simultaneous Authentication of Equals
SIFS Short Inter-Frame Space
SISO Single Input Single Output

SM	Spatial Multiplexing
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio
SPoF	Single Point of Failure
SS	Spatial Stream
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TF	Testfall
TPC	Transmit Power Control
TTL	Time-to-Live
TX	Transmit
UDP	User Datagram Protocol
USB	Universal Serial Bus
VHT	Very High Throughput
WDS	Wireless Distribution System
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WNPR	Weighted NC-to-PCHNC-Ratio
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

1 Einleitung

Die Visionen des *Ubiquitous Computing* und des *Internet of Things* (IoT) werden durch die rasanten Entwicklungen im Bereich der Hardware- und Software-Technologien immer greifbarer. Sie beschreiben eine intelligente Alltagsumgebung, in der miteinander vernetzte Geräte im Hintergrund eine flexible Unterstützung des Nutzers in vielfältigen Situationen und Lebenslagen anbieten. Verteilte eingebettete Systeme bilden dabei hochkooperative Geräteensembles, die im jeweiligen Umfeld individuell zugeschnittene Funktionen realisieren [1, 2]. Einsatzumgebungen finden sich z. B. im *Smart Home* und in der Gebäudeautomation, bei Informations- und Unterhaltungsdiensten, im industriellen Bereich (Industrial IoT) oder auch im Medizinsektor. Ebenso vielfältig wie die Anwendungsmöglichkeiten sind dabei die Leistungsklassen der einzelnen Geräte sowie die Ausprägungen ihres kooperativen Zusammenwirkens. Das Spektrum reicht von unterschiedlichsten Sensoren, Kameras, Erkennungs- und Erfassungssystemen über Aktoren, Antriebe und Anzeigesysteme bis hin zu komplexen Verarbeitungs- und Steuerungseinheiten.

Im verwandten Szenario der *Smart City* ermöglicht eine drahtlose Kommunikationsinfrastruktur (Wireless Backbone) intelligente Dienste auf öffentlichen Plätzen und in Einrichtungen, wie z. B. Universitäten, Behörden, Bahnhöfen, Flughäfen oder Einkaufszentren, bei denen einer Vielzahl wechselnder Nutzer automatisch kontextbezogene Informationen und Interaktionsmöglichkeiten bereitgestellt werden [3, 4]. Zu den möglichen Aufgaben gehört auch die flächendeckende Vernetzung von Parks, Stadien und Messegeländen bei Großveranstaltungen oder die Überbrückung der „letzten Meile“ zur Breitbandanbindung von Wohngebieten. Weitere Anwendungen umfassen intelligente Verkehrs- und Parkleitsysteme oder verteilte Sicherheitslösungen zur Überwachung von Häfen, Industriegebieten oder Gewerbeflächen.

Unter den geeigneten Funktechnologien, die entsprechend hohe Datenraten für diese Anwendungsfelder ermöglichen, sind *Wireless Local Area Networks (WLANs)* der Standardfamilie IEEE 802.11 bereits heute omnipräsent im Consumer-Bereich [5]. Im Gegensatz zum zellulären Mobilfunk nutzt WLAN lizenzfreie Frequenzbänder und kann providerunabhängig mit kostengünstiger Hardware bereitgestellt werden. Bei herkömmlichen, zentralisierten WLAN-Installationen mit einem Access Point (AP) als Basisstation existieren jedoch klare Beschränkungen hinsichtlich der Reichweite und Teilnehmerzahl. Eine kostenintensive und unflexible Erweiterung des Netzwerks wird vielfach immer noch kabelgebunden oder alternativ mittels statischer Repeater-Lösungen vorgenommen. Gerade für die robuste und flächendeckende Vernetzung im Außenbereich ergeben sich jedoch immer höhere Anforderungen. Durch steigende Knotenzahlen und Störungen der Funkstrecke können sich Netzwerkzustände zudem schnell ändern, z. B. in Bezug auf die Erreichbarkeit der Kommunikationsknoten oder die Verbindungsqualität zwischen ihnen. Die Geräte und ihr Zusammenwirken müssen sich daher selbständig an veränderte Bedingungen anpassen.

Eine Alternative bieten WLAN-basierte Mesh-Netzwerke (engl. *Wireless Mesh Networks (WMNs)*), die neben der Spontanvernetzung von Teilnehmern auch einen Multi-Hop-Datenaustausch über mehrere Zwischenstationen hinweg ermöglichen. Realisiert wird dieser durch ein verteiltes Routing-Verfahren, das auf jedem Gerät die aktuellen Kommunikationskosten bei der dynamischen Wahl der Pfade durch das Netzwerk (Mesh-Pfade) berücksichtigt. Die Vergrößerung bestehender Netzwerke kann somit durch einfaches Hinzufügen von Mesh-Knoten erreicht werden. Im Vergleich zum zentralisierten Betriebsmodus sind WLAN-Mesh-Netzwerke leichter erweiterbar, deutlich kostengünstiger bei der flächendeckenden Ausbringung und flexibler in der Anpassung des Netzwerks auf Änderungen [6, 7]. Im Gegensatz zu drahtlosen Sensornetzwerken und aktuellen Low Power Wide Area Network (LPWAN)-Technologien, die anwendungsbedingt starken Ressourcenbeschränkungen unterliegen, vereinen sie wiederum die Vorteile der Vermaschung und hohen Abdeckung mit deutlich höheren Datenraten. Bei der Realisierung von Smart-City-Anwendungen, wie sie in Abb. 1.1 skiz-

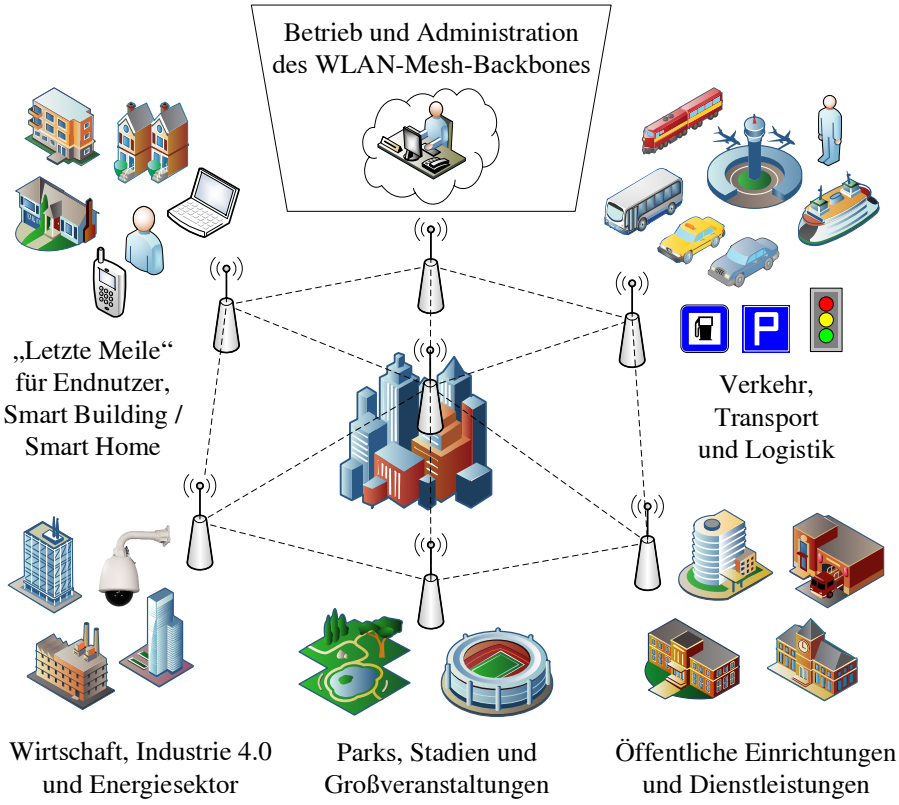


Abbildung 1.1: WLAN-Mesh-Netzwerk für Smart-City-Anwendungen

ziert sind, können WLAN-Mesh-Netzwerke dem Aufbau drahtloser Backbones zur Unterstützung und Ergänzung vorhandener Netzwerkinfrastruktur dienen [8].

Während mittlerweile verschiedene proprietäre Mesh-Produkte für den Heimbereich angeboten werden, die jedoch inkompatibel zueinander gehalten und oft in ihrer Routing-Funktionalität beschränkt sind [9–11], werden für den Aufbau von Zugangsnetzen, Backbones und Service-Infrastrukturen im Maßstab eines Stadtszenarios standardisierte und skalierbare Ansätze benötigt [3]. So muss für den Langzeitbetrieb und die Wartung komplexer Mesh-Backbone-Installationen eine grundlegende herstellerübergreifende Interoperabilität garantiert werden, um die flexible Austauschbarkeit von Geräten gewährleisten zu können.

Die Standarderweiterung IEEE 802.11s ist seit 2012 ein neuer Bestandteil der WLAN-Spezifikation und ermöglicht die herstellerunabhängige, plattformübergreifende Einrichtung eines WMN mit herkömmlicher WLAN-Hardware [5, 12]. Im Gegensatz zu bisherigen Lösungen der *Freifunk*-Initiative und verwandter Community-Projekte [13, 14], in denen zumeist unterschiedliche Routing-Protokolle oberhalb des WLAN-*Ad-Hoc*-Modus eingesetzt werden, integriert IEEE 802.11s grundlegende Mesh-Funktionen bereits direkt in die WLAN-Sicherungsschicht (Media Access Control (MAC) Layer). Um Kompatibilität auf niedrigster Ebene des Protokollstapels zu garantieren, wird die Unterstützung des Routing-Verfahrens *Hybrid Wireless Mesh Protocol (HWMP)* und der Kostenmetrik *Airtime Link Metric (ALM)* vorgeschrieben. Somit bildet IEEE 802.11s eine vielversprechende Basis für interoperable Mesh-Lösungen im Smart-City-Kontext [8].

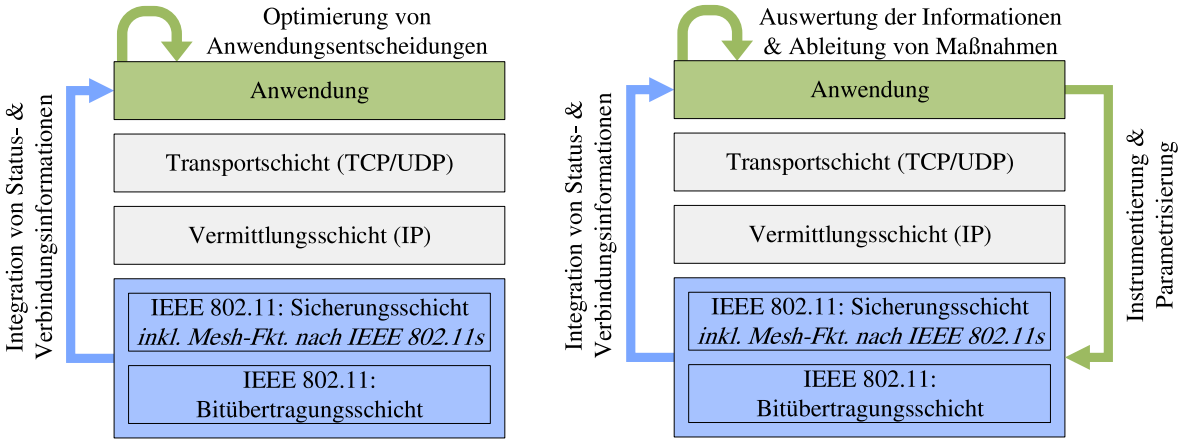
1.1 Problemstellung und Zielsetzungen der Arbeit

Bekannte Performance-Limitierungen von WLAN-Umgebungen, die inhärent durch das geteilte Funkmedium entstehen, gelten auch für WLAN-Mesh-Netzwerke und kommen in diesen aufgrund höherer Teilnehmerzahlen und des zusätzlichen Kommunikationsaufkommens für das Routing von Nachrichten sogar noch stärker zum Tragen [6, 15, 16]. Die Skalierbarkeit komplexer Mesh-Installationen muss daher durch eine geeignete Orchestrierung und Gestaltung der Netzwerkstruktur sowie durch die effiziente Nutzung der vorhandenen Kommunikationsressourcen sichergestellt werden. Dafür benötigte Optimierungsstrategien sind nicht Teil der IEEE-802.11s-Spezifikation und müssen in Form von separaten Lösungen realisiert werden. Diese müssen zudem die Standardfunktionen der WLAN-Sicherungsschicht berücksichtigen, um Interoperabilität mit IEEE 802.11s zu gewährleisten [17]. Der Großteil bestehender Optimierungsansätze wurde jedoch unter der Annahme früherer Mesh-Protokolle und Routing-Verfahren ohne Berücksichtigung der neuen Spezifikation entwickelt [18–21].

Gegenstand dieser Forschungsarbeit ist es, das Verhalten von IEEE-802.11s-WMNs in der Praxis zu untersuchen und Strategien abzuleiten, durch die einerseits die Administrierbarkeit und Skalierbarkeit des Netzwerks erhöht werden und andererseits verteilte Anwendungen die darunter liegende vermaschte Netzwerkstruktur effizient nutzen können. Bezogen auf die zuvor erläuterten Smart-City-Szenarien heißt dies, dass Betreiber von komplexen WLAN-Mesh-Backbones notwendige Management-Aufgaben wie Statusüberwachung, Software-Ausbringung oder Rekonfiguration des Netzwerks realisieren können sollen, ohne dabei die Dienstgüte für Nutzer merklich zu beeinträchtigen. Weiterhin sollen individuelle Anwendungen in die Lage versetzt werden, die ansonsten vor ihnen verborgene Netzwerkstruktur berücksichtigen zu können, um ihre Kommunikationsabläufe anforderungsbezogen zu optimieren. Gleichzeitig besteht die Herausforderung, die theoretisch zur Verfügung stehende Netzwerkkapazität, z. B. in Form von unabhängigen Netzwerkschnittstellen und WLAN-Kanälen, für die Anwendungskommunikation möglichst effizient zu nutzen. Übergeordnetes Ziel ist der Entwurf von standardkonformen Lösungen, welche die durch IEEE 802.11s bereitgestellten Mechanismen gezielt integrieren und somit keine Abhängigkeiten zu Spezial-Hardware oder proprietären Mesh-Protokollen aufweisen. Nach dem Aufbau einer realen Entwicklungs- und Testumgebung werden zwei auf IEEE 802.11s zugeschnittene Cross-Layer-Optimierungslösungen erarbeitet, die prinzipiell kombinierbar sind und ein Zusammenspiel der WLAN-Sicherungsschicht und der Anwendungsschicht realisieren.

Die in der Arbeit entwickelten Optimierungen folgen zwei unterschiedlichen Ansätzen, die in Abb. 1.2 dargestellt sind. In Variante 1 (Abb. links) erfolgt die Cross-Layer-Integration von Mesh-Verbindungsinformationen in eine darüber liegende Anwendung, sodass diese Kommunikationsentscheidungen unter Berücksichtigung der Netzwerkstruktur treffen kann. In Variante 2 (Abb. rechts) wird durch die Anwendungsschicht, ebenfalls basierend auf integriertem Verbindungswissen, zudem eine Instrumentierung der Sicherungsschicht zur effizienten Orchestrierung des Netzwerks vorgenommen. Nachfolgend wird ein kurzer Überblick über die Forschungsbeiträge dieser Arbeit gegeben. Dabei werden die entwickelten Lösungen den genannten Optimierungsvarianten zugeordnet.

Entwicklung einer realistischen WLAN-Mesh-Testumgebung [B 6]: Zur Erprobung und Bewertung der im Rahmen der Dissertation entwickelten Optimierungsansätze bedarf es einer geeigneten Entwicklungs- und Testumgebung. Im Gegensatz zu praktischen Messungen liefern Netzwerksimulationen nur eine vereinfachte Abbildung realer Effekte wie konkurrierenden Medienzugriff und Interferenzen [22, 23]. In der Regel werden Modelle sowohl für die Umgebungseffekte, den Netzwerkprotokollstapel als auch für die darüber liegenden Anwendungen genutzt. Zudem bieten etablierte freie Simulatoren bislang nur unvollständige oder veraltete Modelle für die Mesh-Spezifikation IEEE 802.11s [24, 25]. Im Rahmen der Arbeit wird daher der Weg einer praktischen Untersuchung gewählt. Aufgrund des hohen finanziellen, zeitlichen und organisatorischen Aufwands für das



(a) Variante 1 (b) Variante 2

Abbildung 1.2: Cross-Layer-Optimierungsvarianten

Betreiben einer im Stadtgebiet verteilten Geräteanordnung wird die Möglichkeit geschaffen, realistische Multi-Hop-Topologien im Labormaßstab abzubilden. Dazu wird die reale Testumgebung *Mini-Mesh* entworfen, die durch den Einsatz von Dämpfungsgliedern im Antennenweg und einer reduzierten Sendeleistung die Miniaturisierung eines großflächigen Mesh-Netzwerks darstellt [B 6].

Kollaborativer Datenaustausch in IEEE-802.11s-Netzwerken [B 4, 17]: Für die Verwaltung komplexer Mesh-Installationen ergibt sich die Anforderung des effizienten Austauschs großer Datenmengen, wie z. B. die gezielte Verteilung von Software-/Firmware-Updates an ausgewählte Gerätegruppen oder das Synchronisieren zuvor gesammelter Statusinformationen zwischen mehreren Management-Anwendungen [26, 27]. Das *Peer-to-Peer (P2P)*-Paradigma bietet hier besondere Vorteile, da es die positiven Eigenschaften von Mesh-Netzwerken hinsichtlich Ausfallsicherheit und Skalierbarkeit auch auf der Anwendungsschicht realisiert [28–30]. Als einer der meistgenutzten Vertreter für den kollaborativen Datenaustausch im Internet sticht in diesem Zusammenhang das P2P-Protokoll *BitTorrent* heraus [31, 32].
Als integraler Bestandteil des WLAN-Standards wird die Mesh-Funktionalität von IEEE 802.11s vollständig auf der Sicherungsschicht realisiert und ist daher unsichtbar für höhere Protokollschichten und Anwendungen wie BitTorrent. Zwar müssen dort nicht mehr zwingend Anpassungen für den Mesh-Betrieb vorgenommen werden und eine Interoperabilität wird bereits auf tiefster Ebene erreicht, jedoch wird die Struktur des physischen Netzwerks nicht ohne Weiteres berücksichtigt. Aufgaben wie die Wahl der logischen Kommunikationsendpunkte erfolgen dadurch in der Regel nicht optimal, da die Mehrzahl bestehender Protokolle und Anwendungen nicht für WMNs konzipiert ist. Im Fall von Standard-BitTorrent entsteht so eine ineffiziente Verteilungsreihenfolge der Daten und somit ein unnötig hoher Zeitbedarf für deren Ausbringung [33–35].
Die gezielte Nutzung der verfügbaren Kommunikationsressourcen im WMN durch aufsetzende Dienste ist entscheidend für deren Leistungsfähigkeit und Zuverlässigkeit. Ein Beitrag der vorliegenden Forschungsarbeit ist daher die Optimierung verteilter Anwendungen für den Einsatz in IEEE-802.11s-Netzwerken. Es wird gemäß der Optimierungsvariante 1 in Abb. 1.2 der Cross-Layer-Ansatz *Mesh-Network-Aware BitTorrent (MeNTor)* entwickelt, der die Mesh-Verbindungsinformationen der WLAN-Sicherungsschicht in die BitTorrent-Anwendungsschicht integriert, um deren Peer-Auswahl mit Wissen über die Mesh-Topologie anzureichern [B 4, 17].

Verteiltes Clustering und Kanalwahl in IEEE-802.11s-Netzwerken [B 3]: Beim Betrieb von komplexen Mesh-Backbones ist das Monitoring des aktuellen Netzwerkzustands eine grundlegende Voraussetzung für die Erkennung von Leistungsdefiziten und Fehlverhalten, sodass entsprechende Maßnahmen durch den Netzwerkadministrator eingeleitet werden können [26]. Im Rahmen der

Masterarbeit des Autors wurde eine zentralisierte Monitoring-Lösung oberhalb von IEEE 802.11s entwickelt, die eine Statusabfrage aller Mesh-Teilnehmer durch einen dedizierten Knoten ermöglicht [B 5, 18]. Erste Versuche mit einer steigenden Anzahl von Geräten zeigten jedoch, dass eine Verteilung der zentralisierten Lösung unabdingbar für deren Skalierbarkeit und Ausfallsicherheit ist [B 16]. Eine etablierte Methodik ist die Partitionierung des Mesh-Netzwerks in unabhängige *Cluster* durch Wahl geeigneter *Cluster Heads*, die jeweils nur für die Verwaltung ihrer Region verantwortlich sind [36–38]. In Kombination können Kanalwahlstrategien eine effizientere Ausnutzung des verfügbaren Frequenzspektrums bewirken [39–41].
Vor diesem Hintergrund wird der dezentrale Ansatz *Clustering Heuristic and Channel Assignment (CHaChA)* entwickelt, der unter Berücksichtigung der Mesh-Topologie eine logische Partitionierung des Netzwerks in Cluster auf überlappungsfreien Funkkanälen vornimmt [B 3]. Die vorgestellte Lösung wird gemäß der Optimierungsvariante 2 in Abb. 1.2 prototypisch auf der Anwendungsschicht realisiert und integriert über vorhandene Schnittstellen des Betriebssystems gezielt das Netzwerkwissen der WLAN-Sicherungsschicht, auf dessen Basis die Cluster-Bildung durch gezielte Nutzung der IEEE-802.11s-Mechanismen erfolgt.

1.2 Aufbau und Zitierweise der Arbeit

Die vorliegende Dissertation unterteilt sich in 6 Kapitel. Nach Darlegung der Problemstellung und Zielsetzungen der Arbeit in Kapitel 1 beschreibt das Kapitel 2 die technischen Grundlagen und gibt eine Übersicht bestehender WLAN-Mesh-Lösungen und -Protokolle. In Kapitel 3 wird der Entwurf der realen Testumgebung Mini-Mesh erläutert. Die Kapitel 4 und 5 präsentieren die Cross-Layer-Optimierungsansätze MeNTor und CHaChA samt deren praktischer Evaluation in Mini-Mesh. Abschließend fasst das Kapitel 6 die Ergebnisse der Dissertation zusammen und gibt einen Ausblick auf künftige Forschungsthemen.

Die vorliegende Arbeit verfügt über mehrere Literaturverzeichnisse. Ohne Präfix gekennzeichnete Referenzen verweisen auf das Literaturverzeichnis mit allen externen Quellen, die ohne Beteiligung des Autors entstanden sind (siehe Anhang „Literaturverzeichnis“). Eigene Arbeiten des Autors, als Erst- oder Co-Autor, die in wissenschaftlichen Journalen oder auf wissenschaftlichen Konferenzen veröffentlicht wurden, beginnen mit dem Präfix „A“. Studentische Arbeiten, die vom Autor betreut oder co-betreut wurden, sind mit dem Präfix „B“ gekennzeichnet. Die entsprechenden Listen befinden sich in den Anhängen A und B.

2 Grundlagen

Dieses Kapitel vermittelt die für das Verständnis der weiteren Arbeit notwendigen netzwerktheoretischen Grundlagen. Zunächst werden in den Kapiteln 2.1 und 2.2 Schichtenmodelle und Protokollstapel diskutiert sowie die allgemeinen Architektur- und Routing-Konzepte in drahtlosen Mesh-Netzwerken beschrieben. Nachfolgend gibt Kapitel 2.3 einen Überblick bestehender offener und proprietärer WLAN-Mesh-Lösungen. Die Kapitel 2.4 und 2.5 erläutern die WLAN-Standardfamilie IEEE 802.11 und deren Mesh-Erweiterung IEEE 802.11s im Detail.

In dieser Arbeit werden, soweit möglich, gängige Übersetzungen für englische Bezeichnungen gewählt. Oft ist jedoch die Verwendung der etablierten englischen Fachbegriffe zu bevorzugen. Für eine bessere Lesbarkeit werden zudem die Namensformate *IEEE 802.** und *802.** synonym genutzt.

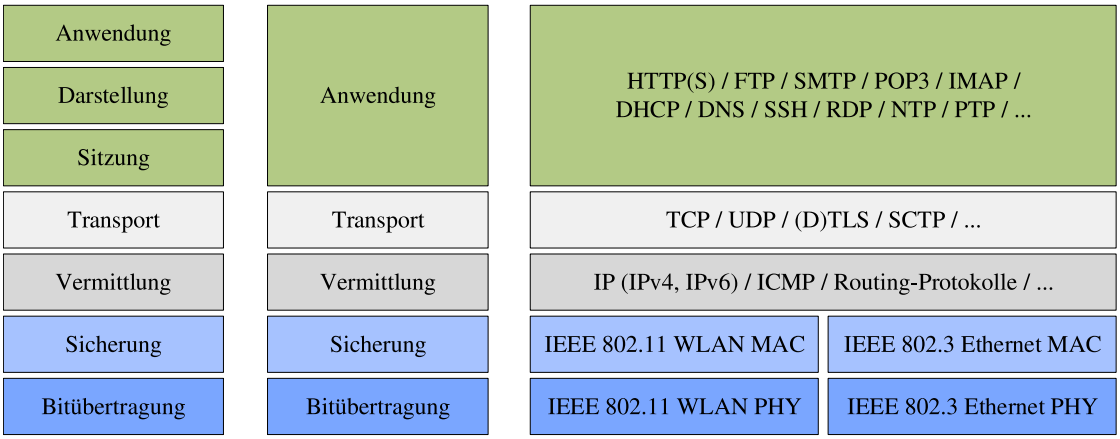
2.1 ISO/OSI- und TCP/IP-Referenzmodell

Das International Organization for Standardization (ISO) / Open Systems Interconnection (OSI)-Referenzmodell, kurz *OSI-Modell*, definierte bereits 1983 eine Trennung der Aufgaben zur Netzwerkkommunikation in sieben aufeinander aufbauende Protokollschichten, um der stetig steigenden Komplexität in diesem Bereich gerecht zu werden [42, 43]. Nach dem OSI-Modell erfolgt eine Abstraktion durch Datenkapselung, sodass tiefere Schichten für die darüber liegenden transparent sind. Dabei wird die zu übertragende *Protocol Data Unit (PDU)* schrittweise um Header- und (optionale) Trailer-Informationen der jeweiligen Schicht ergänzt, um z. B. Adressierung zu realisieren. Kommunikation erfolgt nur zwischen benachbarten Schichten über entsprechende Schnittstellen.

Als Basis der späteren Internetprotokollfamilie wurde ab 1970 das Transmission Control Protocol (TCP) / Internet Protocol (IP)-Referenzmodell, kurz TCP/IP-Modell, mit nur vier Schichten entwickelt [42, 43]. Bei diesem werden die *Sitzungs-* und *Darstellungsschicht* des OSI-Modells in der *Anwendungsschicht* und die *Bitübertragungs-* und *Sicherungsschicht* wiederum in der *Netzzugangsschicht* zusammengefasst, da diese in der Praxis zumeist eine starke Kopplung bzw. Abhängigkeit zueinander aufweisen. In der Literatur wird daneben oft das *hybride Referenzmodell* mit fünf Schichten diskutiert, bei dem die Aufgaben der Bitübertragungs- und Sicherungsschicht erneut separat betrachtet werden [42, 43]. Für jede Schicht existieren zahlreiche mögliche Protokolle. Eine Kombination von Protokollimplementierungen der verschiedenen Schichten wird auch als *Protokollstapel* bezeichnet.

Abb. 2.1 stellt das OSI-Modell und das hybride TCP/IP-Modell gegenüber und nennt Beispiele für gängige Protokollvertreter. Die folgenden Schichten werden im hybriden Modell unterschieden:

- Die *Bitübertragungsschicht* (engl. Physical Layer) ist zuständig für die Übertragung des Bitstroms der PDU aus der Sicherungsschicht und zugeschnitten auf das genutzte Kommunikationsmedium. Sie definiert die physikalischen Signale zur Informationsdarstellung, die Modulation und Kodierung sowie weitere Parameter, wie z. B. die Symboldauer, den Kommunikationsmodus (z. B. simplex, duplex) oder die mechanische Ausführung von Konnektoren.
- Die *Sicherungsschicht* (engl. Data Link Layer) organisiert und sichert den Datenaustausch zwischen verschiedenen Geräten in einem logischen Netzwerk. Dazu werden die Pakete der Vermittlungsschicht in sog. *Frames* eingebettet. Typischerweise werden eindeutige Geräteadressen auf dieser Schicht eingesetzt, diese sind aber nicht zwingend erforderlich. Optional können verschiedene Mechanismen für eine zuverlässige Übertragung und Flusskontrolle mit einem gewünschten Maß an Fehlerbehandlung realisiert werden. So ermöglicht z. B. die Bestätigung von Frames durch *Acknowledgements (ACKs)* eine senderseitige Fehlererkennung anhand von Timeouts und ggf. eine Neuübertragung der Daten. Ebenso können Fehler über die



(a) ISO/OSI-Modell (b) Hybrides TCP/IP-Modell (c) Beispiel-Protokollstapel

Abbildung 2.1: ISO/OSI-Modell und hybrides TCP/IP-Modell im Vergleich [42]

Prüfsumme eines Frames festgestellt werden. Auch Schutzmechanismen wie *Forward Error Correction (FEC)* sind auf der Sicherungsschicht angesiedelt. Da eine weitere Aufgabe dieser Schicht der Medienzugriff der Geräte ist, wird sie oft auch als *Media Access Control (MAC) Layer* bezeichnet. Beispiele hierfür sind das im ursprünglichen Ethernet eingesetzte Zugriffsverfahren *Carrier-Sense Multiple Access / Collision Detection (CSMA/CD)* und das in vielen Funktechnologien genutzte *Carrier-Sense Multiple Access / Collision Avoidance (CSMA/CA)*.

- Die *Vermittlungsschicht* (engl. Network Layer) realisiert die paketorientierte Nachrichtenweiterleitung (Routing) zwischen logisch adressierten Netzen sowie die Verbindung heterogener Netze. Die segmentierten Daten der Transportschicht werden auf der Vermittlungsschicht in *Pakete* gefasst und ggf. zusätzlich fragmentiert. Die maximale Paketgröße (auch *Maximum Transmission Unit (MTU)*) ist dabei abhängig von der zulässigen Frame-Größe der Sicherungsschicht. Im TCP/IP-Modell übernimmt diese Aufgaben das *Internet Protocol (IP)* (IPv4 bzw. IPv6). Neben IP sind auch das *Internet Control Message Protocol (ICMP)* sowie IP-basierte Routing-Protokolle der Vermittlungsschicht zuzuordnen. Werden Routing-Funktionen in lokalen Netzen alternativ bereits auf der Sicherungsschicht realisiert, wird dies zur Unterscheidung als *Pfadselektion* bezeichnet.

- Die *Transportschicht* (engl. Transport Layer) realisiert den Ende-zu-Ende-Transport von *Segmenten* zwischen Prozessen, die über *Ports* adressiert werden. Die Übertragung erfolgt entweder zuverlässig und verbindungsorientiert oder unzuverlässig und verbindungslos. Die erste Variante bietet Zustellungsgarantien und eine gesicherte Reihenfolge der Daten durch Flusskontrolle, erfordert jedoch einen Verbindungsauf- und -abbau sowie die Bestätigung von Segmenten. Optional kann zudem eine senderseitige Überlastkontrolle erfolgen, um proaktiv Engpässe an Zwischenstationen zu vermeiden. Im Fall der verbindungslosen Übertragung müssen derartige Funktionen je nach Bedarf auf der Anwendungsschicht realisiert werden. Bekannter Vertreter der verbindungsorientierten Transportprotokolle ist das *Transmission Control Protocol (TCP)*. Zur zweiten Kategorie zählt das *User Datagram Protocol (UDP)*.

- Auf der *Anwendungsschicht* (engl. Application Layer) befinden sich Nutzerprogramme und mit diesen verbundene Protokolle. Klassische Vertreter sind das Hypertext Transfer Protocol (HTTP) und das File Transfer Protocol (FTP), aber auch das Dynamic Host Configuration Protocol (DHCP) zur IP-Adressvergabe oder das Domain Name System (DNS) zur Namensauflösung. Die Aufgaben der Nachrichtendarstellung und (optionalen) Sitzungsverwaltung (siehe OSI-Modell) werden in der Praxis ebenfalls durch Anwendungsprotokolle abgedeckt.

Die strikte Trennung in Protokollschichten bietet sowohl Vor- als auch Nachteile, da die so gewonnene Abstraktion von der physischen Netzwerktechnologie stets mit Performance-Einbußen einhergeht. Diese sind bei einer Nichtbeachtung von Informationen tieferer Schichten umso größer, je stärker sich die Netzwerkstruktur vom ursprünglichen Anwendungsszenario unterscheidet, für das ein Protokoll oder Dienst auf höherer Schicht ausgelegt wurde. So existiert insbesondere im Bereich der drahtlosen Mesh-Netzwerke ein großer Optimierungsspielraum, da viele Protokolle und Anwendungen der höheren Schichten nicht für diese entwickelt wurden [6, 15, 16]. Die Herausforderung besteht in der Schaffung einer Cross-Layer-Kooperation, ohne Interoperabilität zu beeinträchtigen [44]. Die vorliegende Arbeit widmet sich dieser Problemstellung in Kap. 4.

2.2 Drahtlose Mesh-Netzwerke

Unter dem Begriff der *drahtlosen Mesh-Netzwerke* (engl. Wireless Mesh Networks (WMNs)) werden selbstorganisierende Netzwerke zusammengefasst, deren Knoten sich automatisch miteinander verbinden und, im Gegensatz zu reinen Ad-Hoc-Netzwerken, zusätzlich Routing-Funktionen übernehmen. Die dynamische Nachrichtenweiterleitung zwischen Knoten, die außerhalb ihrer gegenseitigen Funkreichweite liegen, bietet große Vorteile hinsichtlich der kostengünstigen und flexiblen Abdeckung großer Areale bei gleichzeitig hoher Ausfallsicherheit [44].

Vergleichbare Konzepte finden sich auch im Bereich der Wireless Sensor Networks (WSNs) und Wireless Personal Area Networks (WPANs), welche jedoch nur geringe Datenraten unterstützen und anwendungsbedingt starken Ressourcenbeschränkungen unterliegen. Hierzu gehören z. B. auf der Standardfamilie IEEE 802.15.4 [45] basierende Technologien wie 6LoWPAN, Zigbee, Thread oder WirelessHART. Insbesondere das begrenzte Energiebudget der Knoten erfordert hier speziell zugeschnittene Weiterleitungs- und Scheduling-Strategien. Im Vergleich dazu kommen in WMNs für Breitbandanwendungen deutlich leistungsstärkere Geräteklassen zum Einsatz, wodurch die Nutzung anspruchsvollerer Routing-Protokolle möglich ist [44].

2.2.1 Architektur

Zur Architekturbeschreibung von WMNs werden zumeist statische *Mesh Router*, möglicherweise mobile *Mesh Clients* sowie *herkömmliche Clients* unterschieden. Mesh Router bilden dabei ein sogenanntes *Wireless Backbone* und stellen den Zugang für Mesh Clients und mobile Teilnehmer ohne Mesh-Fähigkeit bereit [44]. Abb. 2.2 zeigt ein solches Backbone-WMN am Beispiel. Während sowohl Mesh Router als auch Mesh Clients mit Routing-Funktionen ausgestattet sind, können erstere zusätzlich den Betrieb als Basisstation oder Gateway zur Anbindung an das Internet und andere externe Netzwerke unterstützen, wie z. B. IEEE 802.3 Ethernet Local Area Networks (LANs), IEEE 802.11 Wireless Local Area Networks (WLANs), IEEE 802.16 „WiMAX“ und weitere Technologien. Dazu konfigurieren Mesh Router freie, für die Backbone-Kommunikation nicht benötigte Funkschnittstellen z. B. als WLAN Access Point (AP). Alternativ können Mesh Clients untereinander ein eigenständiges Netzwerk ohne Verwaltung eines Backbones aufbauen. Solche Netzwerke aus rein mobilen Mesh-Knoten werden oft gesondert als *Mobile Ad-Hoc Networks (MANETs)* bezeichnet [46]. Mögliche Anwendungen von MANETs finden sich z. B. bei der Spontanvernetzung von Einsatzkräften in Katastrophenszenarien [7].

Backbone-WMNs stellen jedoch bei Weitem die Architektur mit dem größten Anwendungspotential dar. Sie können die Zuverlässigkeits- und Reichweitenbeschränkungen bisheriger drahtloser Netzwerkinstallationen überwinden und diese kostengünstig ergänzen oder sogar ersetzen, um flächendeckende Informationsdienste zu realisieren. Ihre Fähigkeit zur Selbstorganisation erlaubt zudem eine inkrementelle Ausbringung durch schrittweises Hinzufügen neuer Knoten, womit sich die Robustheit und Konnektivität des Netzwerks erhöhen. Zu den möglichen Aufgaben gehören die flexible Breitbandvernetzung im Stadtbereich, Verkehrsüberwachung und -steuerung, Gebäudeautomatisierung,

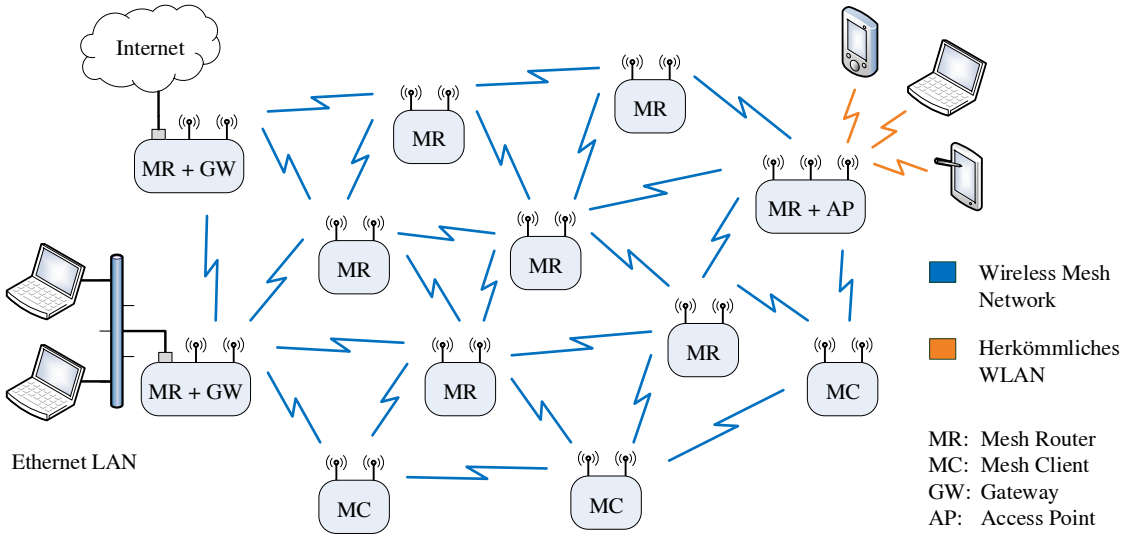


Abbildung 2.2: Architekturbeispiel für ein Backbone-WMN [44]

die Realisierung verteilter Sicherheitslösungen, und viele mehr [44]. Der Fokus der vorliegenden Arbeit lag somit auf der Entwicklung von Optimierungslösungen für diesen Netzwerktyp.

Beim Backbone-WMN handelt es sich in der Regel um eine durch einen Netzwerkbetreiber orchestrierte und verwaltete Infrastruktur aus Mesh Routern. Mögliche Betreiber sind z. B. kommunale Institutionen und Behörden, öffentliche Einrichtungen, Universitäten, Schulen, Vereine, Internet Provider oder Industrieunternehmen. Je nach Anwendungsfall stellen diese das Mesh-Netzwerk als Dienst bereit oder fungieren selbst auch als Nutzer, die entweder direkt als Mesh Clients oder über Mesh Router mit Gateway- oder AP-Funktionalität angebunden sind. Die Nutzung des Netzwerks auf der einen Seite sowie dessen gleichzeitige Überwachung und Administration auf der anderen Seite erfolgen dabei über dieselbe Infrastruktur. Mit zunehmender Netzwerkgröße und -komplexität wächst somit auch die Herausforderung, notwendige Wartungsaufgaben wie Status-Monitoring oder die Ausbringung von Konfigurationsdaten und Software-Updates so effizient zu gestalten, dass eine hohe Dienstgüte für die Nutzer gewährleistet bleibt [26].

2.2.2 Mesh Routing

Ein Hauptaspekt in WMNs ist das verteilte dynamische Routing zwischen den Teilnehmern. Dazu existiert eine Vielzahl an Routing-Protokollen, die sich verschiedenen Kategorien zuordnen lassen [42]. Zum einen werden sie durch die Art der Routen-Aktualisierung charakterisiert:

- *Proaktive* Verfahren ermitteln regelmäßig Pfadinformationen zu Zielknoten, unabhängig davon, ob mit diesen kommuniziert wird. Dies reduziert einerseits die Latenz vor einem Verbindungsaufbau, führt jedoch oft zu unnötigen Kontrollnachrichten für die Aktualisierung ungenutzter Routen. Die Gültigkeit der Informationen ist zudem abhängig von der Update-Periode.

- *Reaktive* Verfahren erstellen Pfadinformationen erst bei Bedarf (on-demand), in der Regel initiiert durch einen Verbindungsaufbau auf der Anwendungsschicht. Der dadurch reduzierte Verwaltungsaufwand geht mit einer höheren Kommunikationslatenz einher.

- *Hybride* Verfahren kombinieren die Vorteile des proaktiven und reaktiven Routings. Dabei werden Routen zu bevorzugten Knoten proaktiv erstellt und je nach Bedarf um reaktiv ermittelte Routen zu weiteren Knoten ergänzt.

Gleichzeitig können die Protokolle auf verschiedenen Routing-Algorithmen basieren, die in der Regel einem der folgenden Grundtypen angehören [42]:

- *Link State Routing* nutzt globales Topologiewissen, um den kürzesten Weg zu einem Ziel zu bestimmen. Dazu lernen Router zunächst ihre Nachbarn kennen, ermitteln die Link-Kosten zu diesen und tauschen ihre Informationen mit allen anderen Routern aus. Auf Basis des vollständigen Netzwerkgraphen können die kürzesten Wege z. B. mittels des Algorithmus von Dijkstra gefunden werden. Die schnelle Konvergenz von Link-State-Verfahren geht mit einem hohen Speicher- und Verwaltungsaufwand einher, was die Skalierbarkeit in großen Netzwerken beeinträchtigen kann.
- Beim *Distance Vector Routing* verwaltet jeder Router eine Tabelle (Vektor) mit den geringsten Distanzen zu Zielknoten und dem jeweiligen Link („Next Hop“), über welche diese erreicht werden. Somit speichert jeder Router nur einen Teil der Netzwerktopologie. Routen werden durch den Informationsaustausch zwischen Nachbarknoten ermittelt und aktualisiert, wobei die Distanzen zu diesen bekannt sind und sich die Gesamtdistanz zum Ziel durch Aufsummierung der Link-Kosten ergibt. Die Aktualität und Zyklenfreiheit von Routen wird mittels Sequenznummern realisiert, während die Time-to-Live (TTL) von Routing-Nachrichten verhindert, dass diese unbeschränkt im Netzwerk kursieren. Der geringere Verwaltungsaufwand wird mit einer im Vergleich zu Link-State-Verfahren langsameren Konvergenz erkauft.
- *Source Routing* unterscheidet sich von den vorherigen Verfahren in der Art der Informationsspeicherung. Hierbei wird die komplette Knotensequenz einer Route durch den Sender (Source) vorgegeben und innerhalb von Paketen als Header-Information mitgeschickt. Weiterleitende Stationen müssen somit nicht zwingend eigene Routing-Tabellen vorhalten. Durch Beobachten des Datenverkehrs können weitere Routen kennen gelernt und für spätere Sendevorhaben genutzt werden.

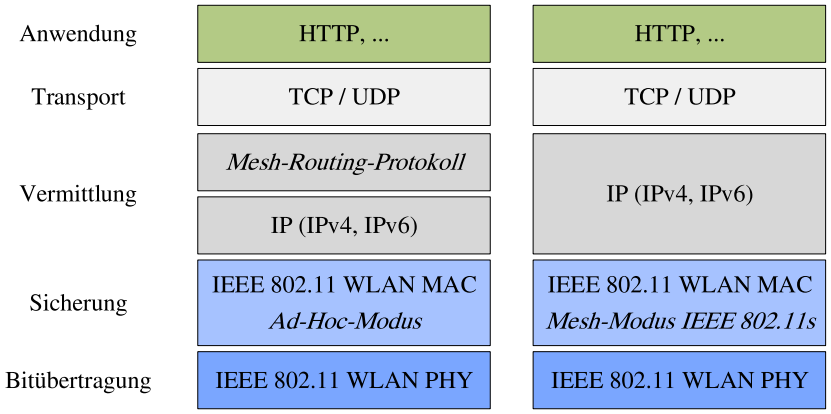
Als Kostenmaß zur Bewertung von Routen kommen neben dem *Hop Count*, also der Anzahl durchlaufener Links zwischen Quelle und Ziel, auch andere Kriterien infrage, wie z. B. die Datenrate oder das Delay der Verbindung, aber auch auf WMNs zugeschnittene Metriken wie *Expected Transmission Count (ETX)*, *Expected Transmission Time (ETT)* oder die *Airtime Link Metric (ALM)* der WLAN-Mesh-Spezifikation IEEE 802.11s [20, 21, 47]. Nachfolgend wird eine Übersicht bestehender Routing-Protokolle und Lösungen für WLAN-basierte Mesh-Netzwerke gegeben.

2.3 Gegenüberstellung von WLAN-Mesh-Lösungen

Für den Aufbau von WMNs im städtischen Umfeld eignet sich als Basistechnologie insbesondere die WLAN-Standardfamilie IEEE 802.11 [5], da diese mit kostengünstiger Hardware auf lizenzfreien Frequenzbändern betrieben werden kann [8]. Wie in Abb. 2.3 dargestellt kann eine Realisierung von WLAN-Mesh-Netzwerken einerseits auf der Vermittlungsschicht durch zumeist IP-basierte Routing-Protokolle und somit oberhalb der 802.11-Spezifikation erfolgen. Zum anderen bietet der mit IEEE 802.11s standardisierte Mesh-Modus eigene Mechanismen zur Frame-Weiterleitung und „Pfadselektion“ bereits direkt auf der WLAN-Sicherungsschicht (MAC Layer), sodass keine Anpassungen auf höheren Protokollebenen erforderlich sind.

2.3.1 Offene Protokolle

WLAN-Mesh-Installationen der *Freifunk*-Initiative [13] und verwandter Community-Projekte wie des Rostocker *Opennet*-Vereins [14] nutzen bislang verschiedene Routing-Protokolle oberhalb des WLAN-*Ad-Hoc*-Modus (siehe Kap. 2.4.2). Dieser ermöglicht lediglich eine Punkt-zu-Punkt-Kommunikation zwischen benachbarten Stationen, sodass die Multi-Hop-Weiterleitung von Paketen darauf aufsetzend realisiert werden muss. Die Grundidee der Freifunk-Initiative besteht darin, dass



(a) Routing oberhalb des Ad-Hoc-Modus

(b) Pfadselektion auf der MAC-Schicht

Abbildung 2.3: Realisierungsvarianten für WLAN-Mesh-Netzwerke

Mitglieder ihre WLAN-Router und APs vernetzen, um einer größeren Gemeinschaft einen Internetzugang bereitzustellen. Die Geräte erhalten dazu neue Firmwares, die eine Mesh-Vernetzung ermöglichen. So werden z. B. das Linux-basierte Router-Betriebssystem *OpenWRT* oder dessen Derivate entsprechend angepasst und es kommen freie Umsetzungen verschiedener Mesh-Routing-Protokolle zum Einsatz. Tab. 2.1 gibt eine Übersicht bekannter Protokolle und ihrer Linux-Implementierungen.

Oft wird das von der Internet Engineering Task Force (IETF) spezifizierte Routing-Protokoll *Optimized Link State Routing (OLSR)* [48] bzw. dessen Nachfolger *OLSRv2* [49] genutzt. Als proaktives Link-State-Verfahren besitzt OLSR die Eigenschaft, dass jeder Knoten alle Routen samt Zwischenstationen im Netzwerk vorhält und regelmäßig aktualisiert, um aus dem globalen Wissen den kürzesten Weg für Pakete ermitteln zu können. So lassen sich zwar geringe Reaktionszeiten erreichen, diese werden jedoch mit einem hohen Kommunikations- und Speicheraufwand zur Verwaltung der Pfadinformationen erkauft [50]. Als Kostenmaß dienen wahlweise der Hop Count oder ETX bei OLSRv1 im Gegensatz zu einer ETT-basierten Metrik bei OLSRv2. Mit `olsrd` [51] bzw. `olsr2d` [52] existieren Userspace-Implementierungen für Linux, welche die IP-Routing-Tabellen des Systems entsprechend konfigurieren.

Die OLSR-Weiterentwicklung *Better Approach to Mobile Ad-Hoc Networking (BATMAN)* [53, 54] ist ebenfalls ein proaktives Protokoll. Dieses nutzt jedoch ein Distance-Vector-Verfahren und so werden nur Weiterleitungsregeln anstatt vollständiger Routen gespeichert. Dazu broadcasten Knoten *Originator Messages (OGMs)* an ihre Nachbarn, welche diese ebenfalls als Broadcasts weiterleiten. Die empfangenen OGMs werden gezählt und auftretende Paketverluste dienen indirekt als Metrik. Dabei wird ausgenutzt, dass OGMs naher Knoten zuverlässiger empfangen werden als OGMs weiter entfernter Knoten. Im Ergebnis speichert jeder Router nur denjenigen Nachbarn als *Next Hop* einer Weiterleitungsregel, von dem die meisten OGMs des zugehörigen Zielknotens empfangen wurden. In der aktuellen Protokollversion BATMAN V können auch Informationen zum Link-Durchsatz („Transmit Quality“ (TQ)) berücksichtigt werden [55]. Die Skalierbarkeit von BATMAN ist im Allgemeinen stark abhängig von der Sendeperiode und Zustellungswahrscheinlichkeit der OGM-Broadcasts [50]. Auch für BATMAN existiert mit `batmand` [56] eine IP-basierte Umsetzung für den Linux Userspace. Eine leichtgewichtigere Implementierungsvariante namens *BATMAN Advanced* wurde in Form des Linux-Kernelmoduls `batman-adv` entwickelt [57]. Dieses wird auf der OSI-Schicht „2,5“ [58] zwischen der Software-Netzwerkschnittstelle und der IP-Vermittlungsschicht integriert und realisiert die Routing-Funktionalität auf Basis von MAC-Adressen.

Tabelle 2.1: Übersicht offener Mesh-Routing-Protokolle
(LS: Link State, DV: Distance Vector, SR: Source Routing, HC: Hop Count, TQ: Transmit Quality, ETX: Expected Transmission Count, ETT: Expected Transmission Time, ALM: Airtime Link Metric, *: Kernelmodul)

Protokoll	**OSI-Schicht**	**Kategorie**	**Metrik**	**Spezifikation**	**Linux-Implementierung**
OLSRv1	3	proaktiv LS	HC / ETX	RFC 3626 [48]	`olsrd` [51]
OLSRv2	3	proaktiv LS	ETT	RFC 7181 [49]	`olsr2d` [52]
BATMAN	3	proaktiv DV	TQ	BATMAN V [55]	`batmand` [56]
BATMAN Adv.	2,5	proaktiv DV	TQ	BATMAN V [55]	`batman-adv`* [57]
BMX6	3	proaktiv DV	TQ	Neumann et al. [60]	`bmx6` [61]
Babel	3	proaktiv DV	ETX	RFC 6126 [62]	`babeld` [63]
DSDV	3	proaktiv DV	HC	Perkins et al. [64]	*Grid*-Projekt [66]
DSR	3	reaktiv SR	HC	RFC 4728 [59]	*Grid*-Projekt [66]
AODV	3	reaktiv DV	HC	RFC 3561 [65]	`AODV-UU` [67]
HWMP	2	hybrid DV	ALM	IEEE 802.11s [5]	`mac80211`*

Weitere offene Protokolle sind das reaktive Source-Routing-Verfahren *Dynamic Source Routing (DSR)* [59], die BATMAN-Weiterentwicklung *BMX6* [60, 61] oder das proaktive Verfahren *Babel* [62] (Linux-Implementierung `babeld` [63]), welches Ansätze der Protokolle *Destination-Sequenced Distance Vector Routing (DSDV)* [64] und *Ad-Hoc On-Demand Distance Vector Routing (AODV)* [65] kombiniert.

Die Vielzahl heterogener Protokolle und Protokollversionen erschwert in der Praxis die Entwicklung einheitlicher Konzepte für Netzplanung und -ausbau sowie die flexible Austauschbarkeit von Hardware-/Software-Komponenten zwischen bestehenden Installationen. Die unterschiedlichen Eigenschaften der Routing-Protokolle, wie z. B. der Wissensumfang über die Netzwerktopologie pro Knoten, die eingesetzte Routing-Metrik oder die Ansiedlung im Protokollstapel, benötigen zudem individuell zugeschnittene Optimierungslösungen [18]. Darüber hinaus bestehen oft technische Limitierungen bei der Implementierung des WLAN-Ad-Hoc-Modus. Dieser dient vielen der genannten Protokolle als Grundlage, lässt sich jedoch geräteabhängig nicht immer mit anderen WLAN-Modi kombinieren und wird durch Treiber neuerer Hardware-Generationen oft nicht unterstützt [68, 69].

Nicht zuletzt aus diesen Gründen besteht mittlerweile im Freifunk-Umfeld ein Trend zum schrittweisen Umstieg auf die Mesh-Spezifikation IEEE 802.11s des WLAN-Standards [70–72]. Bislang werden vielfach nur dessen Mechanismen zur Spontanvernetzung zur Ersetzung des Ad-Hoc-Modus genutzt und bisherige Routing-Protokolle zunächst beibehalten. Ein vollständiger Umstieg auf 802.11s wird jedoch bereits diskutiert [73]. Im Gegensatz zu den bisherigen Lösungen definiert 802.11s erstmals grundlegende Mesh-Funktionen direkt als Bestandteil der WLAN-Sicherungsschicht [5, 12]. Um Kompatibilität auf niedrigster Ebene des Protokollstapels zu garantieren, wird die Unterstützung des Pfadselektionsverfahrens *Hybrid Wireless Mesh Protocol (HWMP)* und der *Airtime Link Metric (ALM)* vorgeschrieben. Das auf AODV basierende HWMP erlaubt zudem die Kombination von reaktiven und proaktiven Betriebsmodi, sodass verschiedene Anwendungsfälle abgedeckt werden können (siehe Kap. 2.5.5).

2.3.2 Proprietäre Lösungen

In den letzten Jahren sind zahlreiche proprietäre WLAN-Lösungen für den Office- und Heimbereich erschienen, die mit der Unterstützung von Mesh-Funktionalität werben. In der Regel handelt es sich bei diesen Produkten jedoch um Repeater-Systeme bestehend aus einem WLAN-Router und einigen Satelliten-Stationen, welche durch die zentrale Basisstation koordiniert werden. Die Funktion der Repeater ist stark herstellerabhängig und reicht von der einfachen Wiederholung von Frames auf physikalischer Ebene bis hin zur teilautomatischen Wahl des Frequenzbands und Synchronisation von Netzwerkeinstellungen, um Nutzern ein möglichst lückenloses Roaming zwischen den Repeatern zu ermöglichen. Zu den bekannten Produktserien gehören AVM *Fritz! Mesh* [74], Ubiquiti *AmpliFi* [75], Amazon *eero* [76], TP-Link *Deco* [77], Netgear *Orbi* [78], Asus *Lyra* [79], Linksys *Velop* [80] oder Samsung *SmartThings Wifi* [81]. Zwar existiert bei vielen Produkten die Option für ein „Daisy Chaining“, also das Aneinanderreihen von Repeatern, um eine größere Wohnfläche abzudecken. Die bestehenden Lösungen sind jedoch inkompatibel zueinander und hinsichtlich der erlaubten Netzwerkgröße sowie Verkettungslänge oft stark beschränkt [9–11].

Mittlerweile existieren mit Google *Nest WiFi* [82, 83] und Tenda *Nova MW6* [84] erste Produkte, die IEEE 802.11s als Basistechnologie angeben, sodass diese zunächst einen kombinierten Einsatz vermuten lassen. Praktisch können diese Systeme jedoch nicht zusammen eingesetzt werden [85]. Abseits davon werden einige Produkte basierend auf offener Firmware und der Linux-802.11s-Implementierung angeboten, darunter die Open-Source-Router *MeshPoint.One* [86] und *FreeMesh* [87] sowie die WLAN-Modulfamilie *WiLink 8* von Texas Instruments [88].

Auch im Enterprise-Segment sind seit Längerem verschiedene WLAN-Produkte erhältlich, die jedoch ebenfalls proprietäre und zueinander inkompatible Mesh-Routing-Protokolle einsetzen. Lösungen existieren z. B. von Cisco/Meraki [89], ABB/Tropos [90], Aruba [91], Firetide [92], Ruckus Wireless [93], Juniper Networks [94] oder Datto Networking (ehem. Open-Mesh) [95]. Hervorzuheben ist die *RouterBoard*-Serie der Firma Mikrotik, deren *RouterOS* ein an die 802.11s-Spezifikation angelehntes Routing-Protokoll namens *HWMP+* unterstützt. Dieses ist jedoch nicht kompatibel zu Standard-HWMP [96].

Zusammengefasst ist in allen Produktbereichen eine erhebliche Herstellerbindung (Vendor Lock-In) zu beobachten. Obgleich einige Lösungen mit der Unterstützung von IEEE 802.11s werben, können diese nicht kombiniert werden. Um der Inkompatibilität zumindest im Heimbereich entgegenzuwirken, versucht die *Wi-Fi Alliance* derzeit die Etablierung des Zertifizierungsprogramms „EasyMesh“ [97]. Zertifizierte Produkte verschiedener Hersteller sollen sich in einer Sternstruktur über einen zentralen Controller brücken lassen. Die individuellen Mesh-Funktionen bleiben dabei jedoch in den verschiedenen Teilnetzen voreinander verborgen, sodass auch auf diesem Weg keine übergreifende Interoperabilität erreicht wird [85, 98].

Für den schrittweisen Aufbau und Langzeitbetrieb komplexer Mesh-Backbones und Service-Infrastrukturen im Maßstab künftiger Smart-City-Szenarien werden standardkonforme Mesh-Systeme benötigt, die echte herstellerübergreifende Interoperabilität garantieren und eine flexible Austauschbarkeit von Geräten gewährleisten können. In diesem Zusammenhang bildet die WLAN-Mesh-Spezifikation IEEE 802.11s eine vielversprechende Basis [3, 8]. Um den Anforderungen dieser Szenarien gerecht zu werden, sind darüber hinaus auf IEEE 802.11s zugeschnittene Optimierungslösungen erforderlich, wie sie im Rahmen dieser Arbeit exemplarisch vorgestellt werden.

2.4 WLAN-Standard IEEE 802.11

Die Standardfamilie IEEE 802.11 [5] beschreibt die herstellerunabhängige Kommunikation in einem Wireless Local Area Network (WLAN) auf lizenzfreien Frequenzbändern. Dafür definiert sie, analog zu Ethernet, die Implementierung der ersten beiden Schichten des OSI-Modells. Während die

Bitübertragungsschicht (engl. *Physical Layer*) insbesondere Übertragungs- und Modulationstechniken vorgibt, werden auf der Sicherungs- bzw. Medienzugriffsschicht (engl. *MAC Layer*) unter anderem der Kanalzugriff auf Basis von CSMA/CA, Mechanismen zur Flusskontrolle sowie die verschiedenen Frame-Formate spezifiziert [99–103].

2.4.1 Übersicht der Teilstandards

Die erste Version des 802.11-Standards erschien bereits 1997 und erlaubte Datenraten von bis zu 2 Mbit/s auf dem 2,4-GHz-Industrial, Scientific and Medical (ISM)-Band. Auf der Bitübertragungsschicht waren die Frequenzspreizverfahren Frequency Hopping Spread Spectrum (FHSS) und Direct Sequence Spread Spectrum (DSSS) in Kombination mit unterschiedlichen Modulationsarten vorgesehen. Der Standard befindet sich in ständiger Entwicklung und wurde bereits durch zahlreiche Überarbeitungsdokumente (engl. *Amendments*) ergänzt. Diese führen Neuerungen auf der Bitübertragungs- und/oder Sicherungsschicht ein oder entfernen überholte Bestandteile der Standard-Spezifikation. Die Beiträge einer Überarbeitung werden üblicherweise mit dem Buchstaben der zugehörigen IEEE-Arbeitsgruppe (Task Group) zusammengefasst. Im Rahmen dieser Arbeit werden die Begriffe „Überarbeitungsdokument“, „Standarderweiterung“ und „Teilstandard“ synonym benutzt.

Tab. 2.2 zeigt eine Übersicht der wichtigsten 802.11-Teilstandards [104]. Neben den Hauptversionen („Wi-Fi 1–6“) werden ausgewählte Erweiterungen der Sicherungsschicht aufgeführt, die für diese Arbeit relevant sind. Dazu zählen insbesondere die Quality of Service (QoS)-Erweiterung 802.11e und die Mesh-Erweiterung 802.11s, die in Kap. 2.4.5 und 2.5 genauer erläutert werden.

Im Jahr 1999 wurden mit 802.11a und 802.11b zwei neue Spezifikationen der Bitübertragungsschicht verabschiedet. Das auf dem 2,4-GHz-Band arbeitende 802.11b führt zusätzliche Direct Sequence Spread Spectrum (DSSS)-Datenraten von bis zu 11 Mbit/s ein. Im Gegensatz dazu definiert 802.11a die Datenübertragung auf dem 5-GHz-Band. Da hier ebenfalls Wetterradar und andere Funkdienste angesiedelt sind, ist in Teilen des Frequenzbands die Nutzung von Koexistenztechniken erforderlich (ausweichender Kanalwechsel und Sendeleistungskontrolle). Als Übertragungstechnik kommt erstmals Orthogonal Frequency Division Multiplexing (OFDM) zum Einsatz und es werden Datenraten von bis zu 54 Mbit/s unterstützt. Im Jahr 2003 übernahm die Erweiterung 802.11g OFDM auch für das 2,4-GHz-Band.

Die Erweiterung 802.11n wurde 2009 veröffentlicht. Sie unterstützt sowohl 2,4- als auch 5-GHz-Band und definiert einen erweiterten Umfang von OFDM-basierten Datenraten. Zusätzlich werden Multiple Input Multiple Output (MIMO)-Modi eingeführt, die den gleichzeitigen Einsatz von jeweils bis zu vier Sende- und Empfangsantennen ermöglichen. Auf diese Weise lassen sich Effekte der Mehrwegeausbreitung positiv zur Erhöhung des Signal-Rausch-Verhältnisses nutzen. Weiterhin unterstützt 802.11n eine Verdopplung der Kanalbandbreite von 20 auf 40 MHz und die gleichzeitige Übertragung unabhängiger Datenströme über verschiedene Antennenwege, wodurch sich auf der Bitübertragungsschicht Bruttodatenraten von bis zu 600 Mbit/s ergeben. Neben diesen Erweiterungen wird auf der Sicherungsschicht das Zusammenfassen von Datenframes ermöglicht.

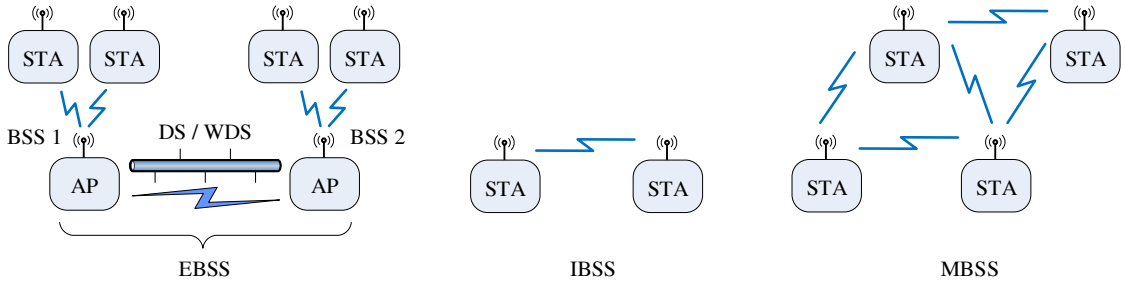
Der Nachfolger 802.11ac, verabschiedet im Jahr 2013 (erste Generation „Wave 1“) bzw. 2015 (zweite Generation „Wave 2“), arbeitet ausschließlich auf dem 5-GHz-Band. Er führt den Trend von 802.11n hinsichtlich höherer Modulationsordnungen, Kanalbandbreiten und der Nutzung von MIMO-Technologie fort, sodass in höchster Ausbaustufe Bruttodatenraten von mehr als 6 Gbit/s erreicht werden können. Die Kombination von 802.11s und 802.11ac (und neuer) ist jedoch in der Praxis bislang nur für ausgewählte Hardware der Hersteller Qualcomm/Atheros und MediaTek möglich. Die zugehörigen Treiber befinden sich zudem noch in der Entwicklung und die Konfigurationsmöglichkeiten sind im Vergleich zu ihren 802.11n-Vorläufern limitiert [105]. Im Rahmen dieser Arbeit wurde sich daher auf die Nutzung von 802.11n-fähiger Hardware beschränkt.

Tabelle 2.2: Übersicht der Überarbeitungsdokumente des IEEE-802.11-Standards [104]
(*: voraussichtliches Erscheinungsdatum)

Dokument	**Jahr**	**Inhalte und Neuerungen**
802.11-1997	1997	FHSS und DSSS im 2,4-GHz-Band, max. Bruttodatenrate 2 Mbit/s
802.11b („Wi-Fi 1“)	1999	DSSS im 2,4-GHz-Band, max. Bruttodatenrate 22 Mbit/s
802.11a („Wi-Fi 2“)	1999	OFDM im 5-GHz-Band, max. Bruttodatenrate 54 Mbit/s
802.11-1999	1999	Zusammenfassung der bisherigen Dokumente
802.11d	2001	regulatorische Bestimmungen zur Frequenznutzung
802.11g („Wi-Fi 3“)	2003	OFDM im 2,4-GHz-Band, max. Bruttodatenrate 54 Mbit/s
802.11e	2004	QoS-Erweiterung (Kanalzugriff und Arbitrierung)
802.11i	2004	Sicherheitsmechanismen (WPA2)
802.11h	2006	Radar-Koexistenzmechanismen im 5-GHz-Band
802.11-2007	2007	Zusammenfassung der bisherigen Dokumente
802.11n („Wi-Fi 4“)	2009	2,4-/5-GHz-Band, Kanalbreite bis 40 MHz, Single-User-MIMO, max. Bruttodatenrate 600 Mbit/s
802.11p	2010	Car-2-Car-Kommunikation
802.11s	2011	Mesh-Kommunikation auf der Sicherungsschicht
802.11-2012	2012	Zusammenfassung der bisherigen Dokumente
802.11ad („WiGig“)	2013	60-GHz-Band, wenige Meter Reichweite, max. Bruttodatenrate 6,9 Gbit/s
802.11ac („Wi-Fi 5“) Wave 1	2013	5-GHz-Band, Kanalbreite bis 80 MHz, Single-User-MIMO, max. Bruttodatenrate 3,4 Gbit/s
802.11ac („Wi-Fi 5“) Wave 2	2015	5-GHz-Band, Kanalbreite bis 160 MHz, Multi-User-MIMO, max. Bruttodatenrate 6,9 Gbit/s
802.11ah („HaLow“)	2016	Sub-GHz-Band, geringe Datenrate/Energiebedarf, hohe Reichweite
802.11-2016	2016	Zusammenfassung der bisherigen Dokumente
802.11ax („Wi-Fi 6“)	2019	2,4-/5-GHz-Band, direkte Weiterentwicklung von 802.11n/ac
802.11ay („NG60“)	2020 *	60-GHz-Band, direkte Weiterentwicklung von 802.11ad
802.11be („Wi-Fi 7“)	2024 *	2,4-/5-/6-GHz-Band, direkte Weiterentwicklung von 802.11ax

Das im Jahr 2019 erschienene 802.11ax baut die bisherigen Entwicklungen von 802.11n/ac weiter aus und ist für das 2,4- sowie 5-GHz-Band definiert. Der Fokus liegt zudem auf der effizienten gleichzeitigen Datenübertragung zu mehreren benachbarten Teilnehmern, wovon künftig auch Mesh-Anwendungen in Kombination mit 802.11s profitieren könnten. Ein direkter Nachfolger ist mit 802.11be bereits ebenfalls in Planung, der erneut höhere Modulationsordnungen, Kanalbandbreiten und MIMO-Modi mit sich bringen wird. Neben dem 2,4- und 5-GHz-Band ist die Nutzung zusätzlicher Frequenzen im 6-GHz-Bereich vorgesehen.

Mit 802.11ah („HaLow“), 802.11ad („WiGig“) sowie dessen Nachfolger 802.11ay („NG60“) existieren weitere jüngere Spezifikationen, deren Fokus jedoch auf Spezialanwendungen liegt und die im Rahmen dieser Arbeit nicht weiter betrachtet werden.



(a) Infrastruktur-Modus (b) Ad-Hoc-Modus (c) Mesh-Modus

Abbildung 2.4: Übersicht der WLAN-Betriebsmodi
(STA: Station, AP: Access Point, (W)DS: (Wireless) Distribution System,
(E)BSS: (Extended) Basic Service Set, IBSS: Independent BSS, MBSS: Mesh BSS)

2.4.2 WLAN-Betriebsmodi

Der WLAN-Standard unterstützt verschiedene Betriebsmodi, welche in Abb. 2.4 gegenübergestellt sind. In einem herkömmlichen *Infrastruktur*-WLAN kommunizieren alle Teilnehmer, auch *Stationen* genannt, über einen *Access Point (AP)*, der als zentrale Basisstation die Weiterleitung von Frames zwischen Kommunikationspartnern übernimmt. Die Netzwerkstruktur entspricht somit einer Sterntopologie. Ein Infrastruktur-WLAN bestehend aus einem AP und einem oder mehreren Stationen bildet ein sogenanntes *Basic Service Set (BSS)*, das typischerweise durch die MAC-Adresse der Basisstation eindeutig identifiziert wird. Die Zusammenschaltung mehrerer APs zur Netzwerkausdehnung erfolgt zumeist über ein kabelgebundenes *Distribution System (DS)*) basierend auf Ethernet. Mehrere miteinander verbundene BSSs bilden wiederum ein *Extended Basic Service Set (EBSS)*. Um den automatischen Wechsel von Stationen zwischen den APs zu ermöglichen, muss auf diesen der gleiche *Service Set Identifier (SSID)* konfiguriert werden, welcher eine menschenlesbare Kennung des Netzwerks darstellt. Darüber hinaus besteht die Möglichkeit der drahtlosen Verbindung von APs über ein statisch konfiguriertes *Wireless Distribution System (WDS)*. Dieser Modus bildet die Grundlage für Repeater-Lösungen, ist jedoch im 802.11-Standard nur grob definiert und Implementierungen sind oftmals nur zwischen Geräten des gleichen Herstellers kompatibel.

Eine besondere Variante des Infrastruktur-Modus stellt der auch als „Wi-Fi Direct“ vermarktete Peer-to-Peer (P2P)-Modus dar. Mit diesem lassen sich WLAN-Stationen, oftmals TVs und Anzeigegeräte, neben ihrer Client-Funktionalität vorübergehend als AP konfigurieren. Sie fungieren dann als *P2P Group Owner* und können direkt Daten von anderen Geräten empfangen. Der P2P-Modus bildet unter anderem die Grundlage für Medien-Streaming-Lösungen wie „Miracast“ [106].

Neben den genannten Infrastruktur-Modi existiert zudem ein *Ad-Hoc*-Modus, bei dem sich WLAN-Stationen ohne AP als gleichberechtigte Teilnehmer auf demselben Kanal spontan miteinander verbinden. Ein solches Netzwerk wird als *Independent Basic Service Set (IBSS)* bezeichnet und unterstützt lediglich die Kommunikation zwischen Stationen in direkter Funkreichweite. Wie in Kap. 2.3 erläutert können verschiedene Routing-Protokolle oberhalb des Ad-Hoc-Modus eine Datenweiterleitung über mehrere Zwischenstationen realisieren. Die bestehenden Lösungen sind jedoch in der Regel inkompatibel zueinander.

Mit der Standarderweiterung IEEE 802.11s wurde erstmals ein offizieller *Mesh*-Modus mit Mechanismen zur Spontanvernetzung und dynamischen Pfadselektion direkt auf der WLAN-Sicherungsschicht spezifiziert. Dabei formen gleichberechtigte Mesh-Stationen ein sogenanntes *Mesh Basic Service Set (MBSS)*. Das Kap. 2.5 beschreibt die Mesh-Erweiterungen im Detail.

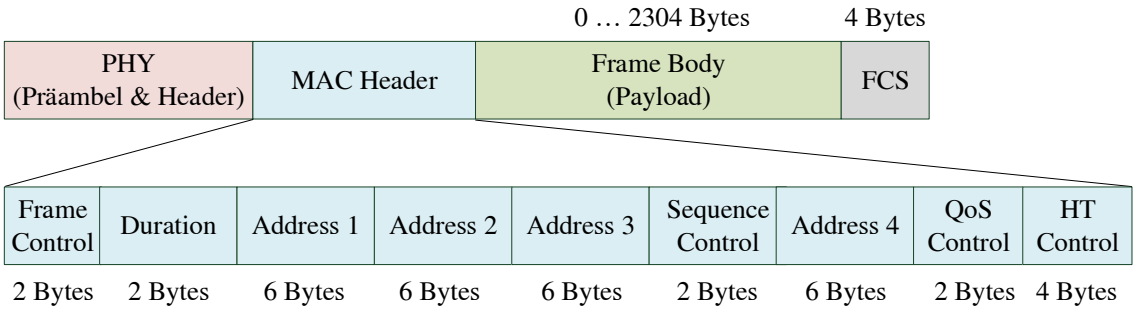


Abbildung 2.5: Allgemeines WLAN-Frame-Format [5]

2.4.3 Nachrichtentypen und -formate

In WLANs werden drei Grundtypen von Frames unterschieden: *Daten-Frames*, *Management-Frames* und *Control-Frames*. Während Daten-Frames die eigentlichen Anwendungsinformationen transportieren, realisieren Management-Frames insbesondere Verwaltungsaufgaben im Zusammenhang mit dem Netzwerkbeitritt und -austritt. Control-Frames unterstützen dagegen die Zustellung von Daten- und Management-Frames, z. B. im Zuge der Flusskontrolle. In der Regel müssen als Unicasts gesendete Frames durch *ACK*-Frames bestätigt werden. Infolge von ACK-Timeouts werden entsprechend Neuübertragungen ausgelöst. Dieser Mechanismus wird auch als *Stop-and-Wait Automatic Repeat ReQuest (ARQ)* bezeichnet. Demgegenüber erfolgt der Versand von Broadcast-Frames immer unzuverlässig und ohne Bestätigung. Zu den wichtigsten Control-Frames gehören neben ACKs auch die Handshaking-Nachrichten *Request-to-Send (RTS)* und *Clear-to-Send (CTS)*. Um eine hohe Empfangsreichweite zu garantieren, werden Management- und Control-Frames zudem im Allgemeinen mit der geringsten im Netzwerk genutzten Datenrate gesendet (siehe Kap. 2.4.5).

Wichtigste Vertreter aus der Gruppe der Management-Frames sind *Beacons*. So kündigen sich APs wie auch Stationen im Ad-Hoc- oder Mesh-Modus periodisch (üblicherweise im Bereich von 100 ms–1 s) durch den Versand von Beacons an. Diese erreichen als Broadcasts alle Stationen im Empfangsbereich, werden jedoch anders als herkömmliche Broadcast-Frames von diesen nicht weitergeleitet. Beacons enthalten unter anderem einen Zeitstempel zur Synchronisation, den Netzwerknamen (SSID), eine Liste unterstützter Datenraten und die Art der Verschlüsselung. Nach erfolgreichem Empfang können Stationen mithilfe dieser Informationen einen Verbindungsaufbau einleiten. Weitere Management-Frames umfassen z. B. Nachrichten zur aktiven Suche nach APs. So übermittelt ein AP nach Anfrage durch ein *Probe Request* die ansonsten auch in den Beacons enthaltenen Informationen mittels *Probe Response* an suchende Client-Stationen. Darüber hinaus existieren verschiedene Management-Frames für den AP-Verbindungsaufbau sowie die Untergruppe der *Action*-Frames für Spezialaufgaben (siehe Kap. 2.5.4 und 2.5.5).

Abb. 2.5 zeigt das allgemeine WLAN-Frame-Format. Es besteht aus dem *MAC-Header*, gefolgt vom *Frame Body* (Payload) und der *Frame Check Sequence (FCS)*, welche die CRC-Prüfsumme des Frames enthält. Der MAC-Header setzt sich wiederum aus bis zu sieben Feldern zusammen, wobei die Bestandteile *Frame Control* und *Duration* sowie auch die abschließende FCS in allen Frame-Typen vertreten sind. Während das Frame-Control-Feld unter anderem Informationen über den Frame-Typ enthält und auch angibt, ob es sich um eine Neuübertragung handelt, trägt das Duration-Feld den berechneten Zeitbedarf für die Frame-Übertragung inklusive Bestätigung (siehe Kap. 2.4.5). Das Feld *Sequence Control* dient der optionalen MAC-Layer-Fragmentierung. In Daten-Frames werden je nach Betriebsmodus drei oder vier MAC-Adressen unterschieden. In herkömmlichen Infrastruktur-WLANs werden *Source*, *Destination* und weiterleitender AP (*Transmitter*) einer

Übertragung adressiert. Quelle und Ziel sind dabei entweder per WLAN oder Ethernet angebundene Stationen. Im WDS- und Mesh-Modus sind dagegen Weiterleitungen über mehrere APs bzw. WLAN-Knoten möglich. Hierbei agieren Zwischenknoten als *Transmitter* und *Receiver* im jeweiligen Weiterleitungsschritt (Hop). Der Mesh-Modus definiert darüber hinaus ein erweitertes Adressierungsschema (siehe Kap. 2.5.6). Dem MAC-Header vorangestellt sind Präambel und Header der Bitübertragungsschicht. Die Präambel wird dabei mit der geringsten Datenrate der Bitübertragungsschicht gesendet. Dadurch ist diese einerseits abwärtskompatibel, andererseits oft noch in höherer Reichweite dekodierbar als die restlichen Frame-Bestandteile. Dies wird im Zuge des Medienzugriffs ausgenutzt (physischer Carrier Sense, siehe Kap. 2.4.5).

2.4.4 Frequenzspektrum und Datenraten

Der WLAN-Standard definiert die Kommunikation auf lizenzfreien Frequenzbändern. Neben einigen Spezifikationen für Spezialanwendungen (vgl. „WiGig“ bei 60 GHz und „HaLow“ im Sub-GHz-Bereich) werden im Allgemeinen das 2,4- und 5-GHz-ISM-Band genutzt. Die Bänder sind generell in 5 MHz-Abschnitte unterteilt und die Mittenfrequenzen der Kanäle aufsteigend nummeriert. Abhängig vom Frequenzband und weiteren länderspezifischen Beschränkungen, definiert als *Regulatory Domains* in der Teilspezifikation 802.11d, stehen unterschiedlich viele Kanäle zur Verfügung. Abb. 2.6 zeigt eine vereinfachte Übersicht der in Europa überlappungsfrei nutzbaren Kanäle im 2,4- und 5-GHz-Band für eine Kanalbandbreite von 20 MHz.

Auf dem 2,4-GHz-Band sind in Europa bis zu 13 Kanäle zur WLAN-Nutzung freigegeben. In Nordamerika sind dagegen nur die Kanäle 1–11 zugelassen, der Kanal 14 ist ausschließlich in Japan mit der älteren DSSS-Modulation (802.11b) nutzbar. Typischerweise wird kommerziell erhältliche WLAN-Hardware mit einer Kanalbandbreite von 20 MHz oder höher betrieben, sodass sich im 2,4-GHz-Band mindestens vier aufeinander folgende Kanäle überlappen. Da die Frequenzfilter realer WLAN-Transceiver keine perfekten Bandpasseigenschaften besitzen, ist zudem oftmals ein Mittenfrequenzabstand von mehr als 20 MHz zwischen Kanälen notwendig, um Interferenzen zuverlässig zu vermeiden [99]. In diesem Zusammenhang definiert der WLAN-Standard sogenannte *Spektralmasken* mit Dämpfungsvorschriften für die verschiedenen Frequenzbänder und Kanalbandbreiten, die angeben, bis zu welchem Leistungspegel das Signal bei welchem Abstand von der Mittenfrequenz abgefallen sein muss [5]. Im 2,4-GHz-Band sind in der Regel nur bis zu drei 20-MHz-Kanäle (z. B. Kanal 1, 6 und 11) gleichzeitig überlappungsfrei nutzbar. Zudem teilen sich WLAN-Geräte dieses Band mit verschiedenen anderen Technologien, wie Mikrowellenherde, DECT, Bluetooth oder IEEE-802.15.4-basierten Protokollen. Die in Europa zulässige Sendeleistung beträgt im 2,4-GHz-Band 100 mW. Dabei bezieht sich diese Angabe auf die *Equivalent Isotropic Radiated Power (EIRP)* eines idealen Kugelstrahlers (Isotropstrahlers) unter Berücksichtigung des Antennengewinns.

Auf dem deutlich größeren 5-GHz-Band ist bei gleichem 5-MHz-Kanalabstand für eine Kanalbreite von 20 MHz nur jeder vierte Kanal freigegeben, sodass benachbarte Kanäle theoretisch überlappungsfrei sind. Aufgrund der zuvor genannten Beschränkungen realer Frequenzfilter kann es jedoch auch hier noch geräteabhängig zu Störungen kommen [99]. In Europa sind insgesamt bis zu 19 Kanäle nutzbar, von denen 8 Kanäle (36–64) nur für Indoor-Anwendungen und weitere 11 Kanäle (100–140) sowohl für Indoor- als auch Outdoor-Anwendungen freigegeben sind. Aufgrund der Koexistenz mit Wetter- und Flugradar in den vier oberen Indoor- sowie allen Outdoor-Kanälen gelten weitere Vorschriften für die Einhaltung einer maximalen Sendeleistung sowie die Umsetzung von Mechanismen zur Radarsignalerkennung und Kanalvermeidung. Diese sind als *Transmit Power Control (TPC)* und *Dynamic Frequency Selection (DFS)* im WLAN-Standard definiert [5]. Während die Sendeleistung in Europa im Indoor-Bereich bis zu 200 mW betragen darf, sind für Outdoor-Anwendungen bei aktivierter TPC und DFS bis zu 1000 mW möglich. Die Nutzung vier weiterer

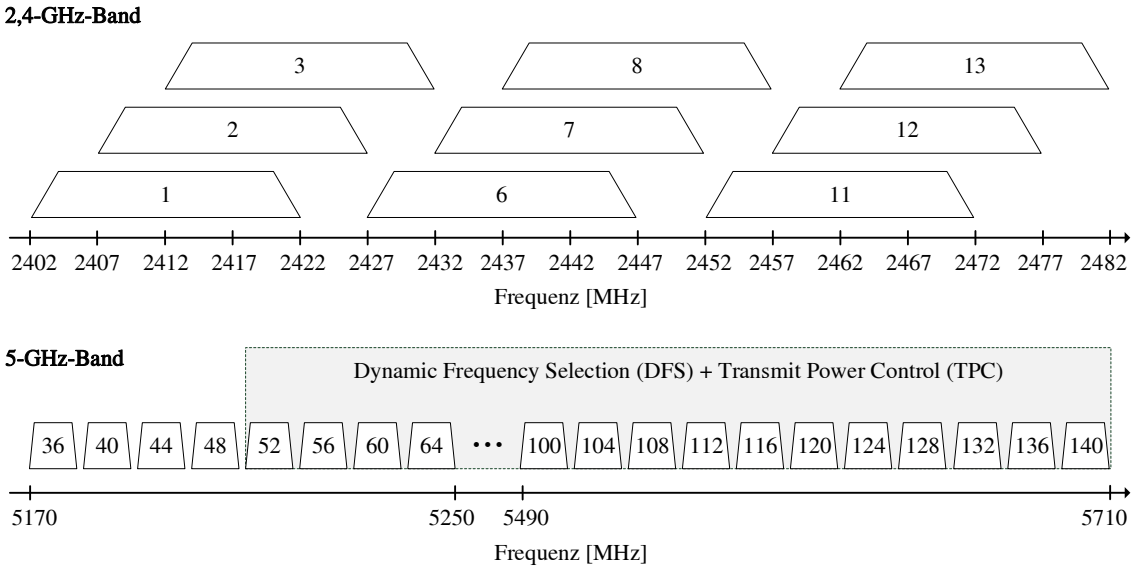


Abbildung 2.6: Überlappungsfreie WLAN-Kanäle im 2,4- und 5-GHz-Band (Europa) [99]

Kanäle (149–165) ist in Europa nur für sogenannte *Short Range Devices* mit einer maximalen Sendeleistung von 25 mW erlaubt [107].

Der Großteil der WLAN-Versionen für den universellen Datentransfer („Wi-Fi 2–6“ bzw. 802.11a/g/n/ac/ax) nutzt als Übertragungstechnik das Frequenzmultiplexverfahren *Orthogonal Frequency Division Multiplexing (OFDM)*. Hierbei wird die Nutzinformation mit hoher Datenrate auf mehrere Teildatenströme mit geringer Datenrate aufgeteilt. Diese werden separat durch herkömmliche Modulationsverfahren wie Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK) und Quadrature Amplitude Modulation (QAM) mit einer geringeren Bandbreite moduliert. Die so erzeugten, dicht beieinander liegenden Subträger werden zu einem gemeinsamen Hochfrequenzsignal addiert und beeinflussen sich gegenseitig kaum (sind im Funktionenraum „orthogonal“), was ihre Unterscheidung am Empfänger ermöglicht. Der Vorteil dieser Technik ist, dass schmalbandige Störungen nur Einfluss auf einzelne Subträger haben, während die Übertragung über die verbleibenden Subträger weiterhin erfolgen kann. Im Randbereich der Kanalbandbreite liegende Subträger bleiben als *Guard Band* zu benachbarten Kanälen ungenutzt. Zum Schutz vor Inter-Symbol-Interferenz werden zudem aufeinander folgende Informationssymbole in einem gewissen Zeitabstand, dem sogenannten *Guard Interval (GI)*, gesendet.

Mit der Einführung des *High Throughput (HT)*-Modus für die Bitübertragungsschicht mit 802.11n („Wi-Fi 4“) wurde der Anteil der zur Datenübertragung genutzten OFDM-Subträger gegenüber 802.11a/g von 48 auf 52 der insgesamt 64 Subträger unter Kürzung des Guard Bands vorgenommen. Zudem erlaubte 802.11n erstmals die Nutzung einer erhöhten Kanalbreite von 40 MHz. Diese Option wurde mit 802.11ac („Wi-Fi 5“, *Very High Throughput (VHT)*-Modus) auf 80 bzw. 160 MHz breite Kanäle erweitert. Dadurch erhöht sich einerseits die erreichbare Bruttodatenrate, andererseits sinkt jedoch die Anzahl überlappungsfreier Kanäle.

Eine weitere Neuerung von 802.11n war die Definition einheitlicher *Modulation and Coding Schemes (MCSs)*, welche aufsteigend indiziert sind. Jeder MCS-Index repräsentiert eine mögliche Kombination aus Subträger-Modulationsverfahren und Coderate der FEC. Bei dieser handelt es sich um eine häufig in Funktechnologien eingesetzte Fehlerkorrekturtechnik, bei der die Informationen senderseitig redundant kodiert werden, sodass Bitfehler am Empfänger ohne Neuanforderung der Daten

kompensiert werden können. Die FEC-Rate wird als Verhältnis *k*/*n* angegeben, d.h. für *k* Bits Nutzdaten werden *n* Bits zur Kodierung aufgewendet [99].

Darüber hinaus existiert seit 802.11n die Möglichkeit der Nutzung von *MIMO*-Techniken. So können mehrere Antennenwege einerseits zum Senden und Empfangen des gleichen Datenstroms eingesetzt werden, wodurch die Zuverlässigkeit der Datenübertragung erhöht werden kann. Dieser als Antennendiversität (engl. *Diversity MIMO*) bezeichnete Modus existierte in der Praxis schon vor der 802.11n-Spezifikation und ist auch mit früheren Varianten der Bitübertragungsschicht nutzbar. Andererseits können mittels *Spatial Multiplexing MIMO* unabhängige Datenströme über verschiedene Antennen übertragen werden, wodurch sich die Gesamtdatenrate steigern lässt. Dieser Modus erfordert gute Kanalbedingungen und Umgebungen mit ausgeprägter Mehrwegeausbreitung [99, 100, 102].

Der MCS-Index bietet gleichzeitig Rückschluss darauf, wie viele unabhängige Datenströme (engl. *Spatial Streams*) übertragen werden. Daraus ergibt sich, welcher MIMO-Modus zum Einsatz kommt. Insgesamt existieren bei 802.11n acht verschiedene Modulations-FEC-Kombinationen. Dabei indizieren MCS 0–7 diese Kombinationen für den Fall, dass nur ein Datenstrom über eine Antenne bzw. redundant über alle Antennenwege mittels Diversity MIMO übertragen wird. Entsprechend fassen MCS 8–15 die Kombinationen für den Fall mit zwei Datenströmen und Spatial Multiplexing MIMO zusammen. In der 802.11n-Spezifikation setzt sich die MCS-Indizierung für Spatial Multiplexing mit drei (MCS 16–23) und vier Antennen (MCS 24–31) fort.

Zusammengefasst ergibt sich die WLAN-Bruttodatenrate auf der Bitübertragungsschicht abhängig vom genutzten MCS (Modulationsverfahren, FEC-Coderate und Anzahl der Spatial Streams), der GI-Länge sowie der gewählten Kanalbreite. Tab. 2.3 zeigt die Bruttodatenraten für die 802.11n-MCS 0–15 unter Annahme verschiedener GI-Längen (Long vs. Short GI) und Kanalbandbreiten (20 vs. 40 MHz). In der Praxis variieren WLAN-Stationen ihren MCS-Index zumeist durch Nutzung dynamischer *Rate Control Algorithms (RCAs)*. Dabei können verschiedene Kriterien berücksichtigt werden, wie z. B. die Link-Signalstärke zu anderen Teilnehmern oder die beobachtete Erfolgshäufigkeit von Übertragungen mit dem jeweiligen MCS [108].

2.4.5 Medienzugriff und Flusskontrolle

Die *Kollisionsdomäne* einer WLAN-Station wird durch alle anderen Stationen in Empfangsreichweite gebildet, da sich diese das Kommunikationsmedium teilen. Die Gesamtheit aller in einem Netzwerk verbundenen Teilnehmer wird gesondert als *Broadcast-Domäne* bezeichnet, da diese beim Versand von Broadcast-Nachrichten vollständig durchlaufen wird. Da es sich bei WLAN um eine Halb-Duplex-Technologie handelt, also nicht gleichzeitig gesendet und empfangen werden kann, muss der Zugriff auf das Medium außerdem exklusiv erfolgen. Ein gleichzeitiger Zugriff führt hingegen zu einer *Kollision*, also der Überlagerung der Signale am Empfänger. Die gegenseitige Beeinflussung von Stationen auf demselben Kanal wird auch als *Co-Channel Interference (CCI)* bezeichnet. Dagegen werden Störungen von Teilnehmern auf benachbarten und überlappenden Kanälen auch unter dem Begriff *Adjacent Channel Interference (ACI)* zusammengefasst. Eine vollständig parallele Kommunikation kann nur auf zueinander überlappungsfreien Kanälen erfolgen.

Zur Koordination des Medienzugriffs nutzt WLAN das Verfahren *Carrier-Sense Multiple Access / Collision Avoidance (CSMA/CA)*. Dabei prüfen Stationen mittels *Carrier Sense* vor einem Sendevorgang, ob der Kanal belegt ist. Der Aspekt der *Collision Avoidance*, also die proaktive Vermeidung gleichzeitiger Sendevorgänge nach Erkennung eines freien Mediums, wird durch die Einhaltung zufälliger Wartezeiten realisiert. Beim Carrier Sense wird unterschieden zwischen dem *physischen* und dem *virtuellen* Carrier Sense. Der *physische* Carrier Sense berücksichtigt einerseits WLAN-Übertragungen auf demselben Kanal (Mechanismus *Clear Channel Assessment (CCA)* zur Erkennung der Physical-Layer-Präambel von Frames) und andererseits WLAN-Übertragungen auf überlap-

Tabelle 2.3: Datenrate der Bitübertragungsschicht für IEEE 802.11n (HT MCS 0–15) [109]
(MCS: Modulation & Coding Scheme, FEC: Forward Error Correction, GI: Guard Interval, LGI: Long GI, SGI: Short GI)

				Kanalbreite 20 MHz		**Kanalbreite 40 MHz**	
				LGI 800 ns	**SGI 400 ns**	**LGI 800 ns**	**SGI 400 ns**
MCS-Index	**# Datenströme**	**Modulation**	**FEC-Rate**	**Datenrate [Mbit/s]**	**Datenrate [Mbit/s]**	**Datenrate [Mbit/s]**	**Datenrate [Mbit/s]**
0	1	BPSK	1/2	6,5	7,2	13,5	15
1	1	QPSK	1/2	13	14,4	27	30
2	1	QPSK	3/4	19,5	21,7	40,5	45
3	1	16-QAM	1/2	26	28,9	54	60
4	1	16-QAM	3/4	39	43,3	81	90
5	1	64-QAM	2/3	52	57,8	108	120
6	1	64-QAM	3/4	58,5	65	121,5	135
7	1	64-QAM	5/6	65	72,2	135	150
8	2	BPSK	1/2	13	14,4	27	30
9	2	QPSK	1/2	26	28,9	54	60
10	2	QPSK	3/4	39	43,3	81	90
11	2	16-QAM	1/2	52	57,8	108	120
12	2	16-QAM	3/4	78	86,7	162	180
13	2	64-QAM	2/3	104	115,6	216	240
14	2	64-QAM	3/4	117	130	243	270
15	2	64-QAM	5/6	130	144,4	270	300

penden Kanälen bzw. Interferenz anderer Funktechnologien (Mechanismus *Energy Detection (ED)*). Der *virtuelle* Carrier Sense nutzt darüber hinaus Zeitinformationen gesendeter WLAN-Frames (*Duration*-Feld im MAC-Header), um die Kanalbelegungszeit auf empfangenden Stationen zu aktualisieren. Der zugrunde liegende Timer wird auch als *Network Allocation Vector (NAV)* bezeichnet. Dieser bietet den Vorteil, dass während der bekannten Belegungszeit der physische Carrier Sense zum Energiesparen vorübergehend deaktiviert werden kann.

Der WLAN-Standard definiert verschiedene Implementierungsvarianten des CSMA/CA-Prinzips. Zugrunde liegendes Standardverfahren ist die *Distributed Coordination Function (DCF)*, welche in Netzwerken mit und ohne AP nutzbar ist. Mit der *Point Coordination Function (PCF)* wurde zudem ein optionales Verfahren spezifiziert, bei dem der AP die Planung der Zugriffe aller Teilnehmer zentral koordiniert. Die mit der Teilspezifikation IEEE 802.11e im Jahr 2005 eingeführte *Hybrid Coordination Function (HCF)* erweiterte die bestehenden Mechanismen um Quality of Service (QoS)-Funktionen [16]. Sie umfasst die Teilverfahren *Enhanced Distributed Channel Access (EDCA)* und *HCF Controlled Channel Access (HCCA)*. Während EDCA direkt auf der DCF basiert und diese um eine Unterscheidung von Prioritätsklassen beim Senden von Frames ergänzt, definiert das optionale HCCA diese Erweiterungen analog für die PCF. Bislang wurden PCF und HCCA jedoch nicht in Produkten implementiert. Hingegen ist die Unterstützung von EDCA in WLAN-Installationen

Tabelle 2.4: Standardwerte der EDCA-Parameter für 802.11n im 5-GHz-Band [16]
(*: Annahme einer ACK-Sendezeit von 44 μs mit geringster Basisdatenrate)

AC-Priorität	**Slot Time**	**SIFS**	**AIFS**	**EIFS***	CW_{min}	CW_{max}
Background	9 μs	16 μs	79 μs	139 μs	15	1023
Best Effort	9 μs	16 μs	43 μs	103 μs	15	1023
Video	9 μs	16 μs	34 μs	94 μs	7	15
Voice	9 μs	16 μs	34 μs	94 μs	3	7

basierend auf 802.11n (und neuer) sowie für 802.11s-basierte Mesh-Netzwerke verpflichtend [5]. Mit der QoS-Erweiterung werden auf höherer Schicht konfigurierte Prioritätsklassen für Datenübertragungen auf vier *Access Categories (ACs)* abgebildet. Unterschieden werden die AC-Prioritäten *Background*, *Best Effort* (Standardpriorität), *Video* und *Voice*, welche jeweils ihre eigene Sendewarteschlange besitzen. Die Untersuchung verschiedener AC-Prioritäten für konkurrierende Kommunikationsanwendungen stand nicht im Fokus der vorliegenden Arbeit. Daher wurde in allen praktischen Experimenten stets die Standardpriorität *Best Effort* für Applikationsdaten genutzt.

Der verteilte Medienzugriff mittels EDCA nutzt feste und zufällige Wartezeiten, deren konkrete Werte abhängig von der Variante der Bitübertragungsschicht und dem Frequenzband sind. Tab. 2.4 gibt eine Übersicht der Standardwerte für 802.11n im 5-GHz-Band. Die kleinste Zeitkonstante ist die sogenannte *Slot Time*. Je nach gesendetem Frame-Typ werden zudem verschiedene *Inter-Frame Spaces (IFSs)* genutzt. Der Mindestabstand zwischen zwei Frames ist der sogenannte *Short Inter-Frame Space (SIFS)*. Dieser wird vor dem Versand von ACKs und anderen Control Frames eingehalten, wodurch diese die höchste Priorität erhalten. Aus den festen Werten für Slot Time und SIFS ergeben sich weitere Wartezeiten. Diese werden außerdem entsprechend den AC-Prioritäten skaliert. So dient der sogenannte *Arbitration Inter-Frame Space (AIFS)* der Priorisierung von Daten-Frames. Der *Extended Inter-Frame Space (EIFS)* wird wiederum von empfangenden Stationen abgewartet, die eine fehlerhafte Datenübertragung z. B. anhand der Frame-Prüfsumme erkennen.

Nach dem Abwarten der entsprechenden IFS-Zeit folgt eine weitere zufällige Wartezeit, das sogenannte *Contention Window (CW)*. Dazu wird ein gleichverteilter ganzzahliger Zufallswert aus einem Intervall zwischen 0 und CW_{min} gewählt. Das CW ergibt sich anschließend aus der Multiplikation dieses Werts mit der Slot Time. Bei jedem neuen Übertragungsversuch des gleichen Frames, ausgelöst durch ACK Timeouts, erhöht sich die obere Intervallschranke CW_{min} nach der Formel $[CW_{min+1} = 2 \cdot CW_{min} - 1]$ bis zu einem Maximalwert CW_{max}. Die zulässige Anzahl an Übertragungsversuchen ist in der Regel als *Retry Limit* konfigurierbar [110]. Bei Übertragungsfehlern steigt somit die durchschnittliche Zugriffszeit, wodurch im Gegenzug die Kollisionswahrscheinlichkeit sinkt. Dieses Verfahren wird auch als *Binary Exponential Back-Off (BEB)* bezeichnet. Die Dimensionierung von CW_{min} und CW_{max} ist erneut abhängig von der AC-Priorität. Die so bestimmte Backoff-Zeit wird in Slot-Time-Schritten dekrementiert, solange das Medium als frei erkannt wird. Kommt es zwischenzeitlich zu anderen Sendevorgängen, wird der Timer pausiert. Nach Ablauf der Wartezeit beginnt die Station ihre Übertragung.

Abb. 2.7 skizziert den prinzipiellen Zeitablauf einer Daten-Frame-Übertragung inklusive Bestätigung mittels ACK-Frame. Im Beispiel nehmen Station A (Sender) und Station B (Empfänger) das Medium aufgrund einer laufenden Übertragung zunächst als belegt war. Nachdem der Carrier Sense den Kanal als frei erkennt, wartet Station A für die Zeit [AIFS+CW] und sendet den Daten-Frame. Station B muss vor dem Versand des zugehörigen hochprioren ACK-Frames lediglich einen SIFS abwarten. Zur Effizienzsteigerung aufeinanderfolgender Übertragungen definieren 802.11e und 802.11n zudem das

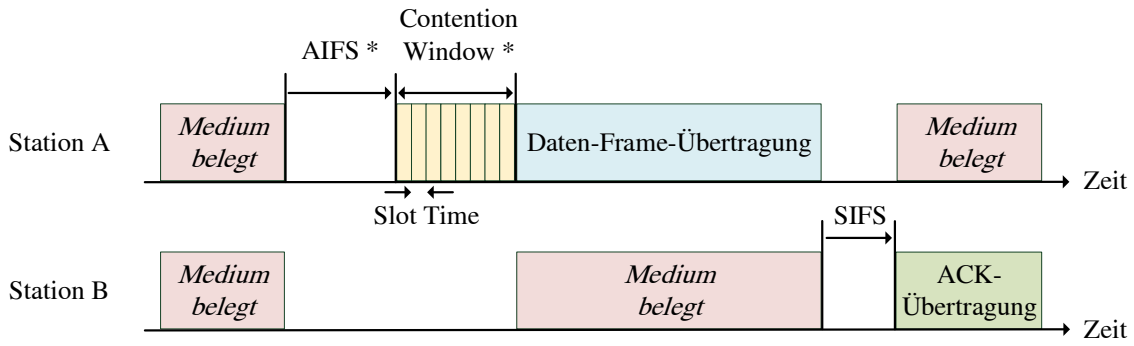
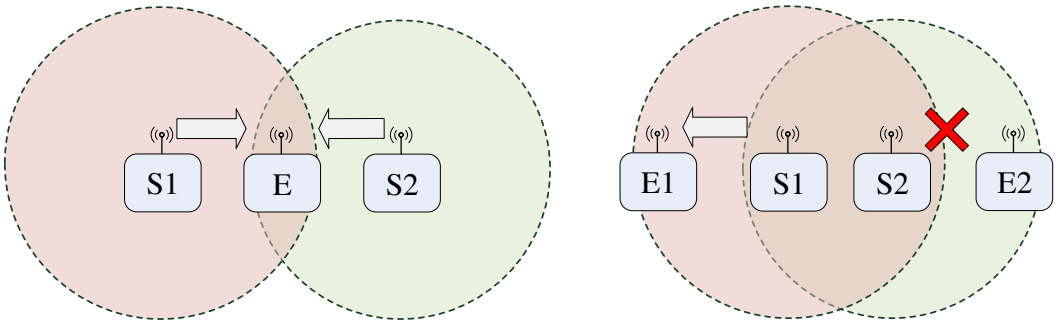


Abbildung 2.7: Beispiel-Datenübertragung mit EDCA [16]
(*: Skalierung der Wartezeit abhängig von der Access Category)



(a) Hidden-Node-Problem (b) Exposed-Node-Problem

Abbildung 2.8: Hidden- und Exposed-Node-Problem [42]

optionale Zusammenfassen mehrerer Daten-Frames derselben Prioritätsklasse (engl. *Frame Aggregation*) sowie deren gesammelte Bestätigung mittels *Block ACK*. Auf diese Weise können Backoff- und IFS-Wartezeiten für ansonsten einzeln gesendete Frames eingespart werden [111].

Im Kontext des Medienzugriffs existieren darüber hinaus bekannte Phänomene wie das *Hidden-Node-* und das *Exposed-Node*-Problem [42]. Abb. 2.8 zeigt diese am Beispiel. Erkennen zwei außer Reichweite befindliche Knoten (auch „Hidden Nodes“) zeitgleich das Medium als frei und senden Frames zu einem dazwischen liegenden Knoten, kommt es an diesem zu einer Kollision (Abb. 2.8 (a)). In der Folge greifen Fehlerbehandlungen auf der Sicherungsschicht oder höheren Protokollebenen, sodass der Datendurchsatz sinkt. Eine andere Situation entsteht, wenn benachbarte Sender zu verschiedenen Empfängern übertragen wollen, die wiederum außerhalb einer gegenseitigen Reichweite liegen (Abb. 2.8 (b)). Durch das CSMA/CA-Prinzip erhält nur einer der Sender Zugriff, obwohl in diesem Szenario gleichzeitige Übertragungen möglich wären und nicht an den jeweiligen Empfängern kollidieren würden. Die hierbei in ihrer Kommunikation blockierten Stationen werden als „Exposed Nodes“ bezeichnet. Zur Linderung des Hidden-Node-Problems definiert der WLAN-Standard den optionalen Mechanismus *Request-to-Send (RTS) / Clear-to-Send (CTS)*. Dieser realisiert ein zusätzliches, bidirektionales Handshaking zwischen Sender und Empfänger vor Beginn einer Frame-Übertragung. Dabei sind RTS und CTS als separate Control Frames definiert. Das CTS, als Antwort auf ein RTS, wird wie ein ACK-Frame nach Abwarten eines SIFS ohne zufällige Wartezeit verschickt und besitzt daher eine sehr hohe Priorität. Das Mithören eines oder beider Nachrichten setzt andere Stationen über einen Übertragungsvorgang und dessen Dauer in Kenntnis, sodass Hidden-Node-Kollisionen reduziert werden können. Das Exposed-Node-Problem ist nur unter besonderen Voraussetzungen durch RTS / CTS lösbar [112].

2.5 WLAN-Mesh-Erweiterung IEEE 802.11s

Die Standarderweiterung IEEE 802.11s zur Spezifikation von WLAN-Mesh-Netzwerken wurde im September 2011 verabschiedet und ist seit 2012 integraler Bestandteil des Basisstandards IEEE 802.11 [5, 12, 113, 114]. Im Gegensatz zu bisherigen Lösungen (siehe Kap. 2.3) werden alle Mesh-Grundfunktionen direkt auf der WLAN-Sicherungsschicht definiert. Sie sind daher transparent zu höheren Protokollschichten, während die Kompatibilität zu existierenden Versionen der Bitübertragungsschicht uneingeschränkt bestehen bleibt. Prinzipiell sind oberhalb der Sicherungsschicht keine Anpassungen für den Mesh-Betrieb erforderlich und es wird Interoperabilität bereits auf tiefer Ebene ermöglicht. Zudem können Eigenschaften der WLAN-Technologie leichter berücksichtigt werden. So stehen beispielsweise für die gezielte Weiterleitung von Frames über mehrere Zwischenstationen genaue Statusinformationen der einzelnen Funkverbindungen zur Verfügung.

2.5.1 Übersicht der Mesh-Funktionen

IEEE 802.11s spezifiziert verschiedene Grundfunktionen für die Realisierung von WLAN-Mesh-Netzwerken. Abb. 2.9 zeigt eine Übersicht der Hauptbestandteile. Zunächst beschreibt die *Mesh Coordination Function (MCF)* die Mechanismen für den Medienzugriff. Weiterhin werden Funktionen für das gegenseitige Finden und Vernetzen von Mesh-Knoten (Peering) sowie Verfahren zur Frame-Weiterleitung und dynamischen Pfadselektion definiert. Neben diesen Pflichtmechanismen sind weitere optionale Funktionen vorgesehen, darunter Sicherheitsmechanismen, Energiesparmodi, Staukontrolle sowie die Ankündigung von Kanaländerungen.

Zur Integration der Mesh-Funktionen wird die bestehende WLAN-Sicherungsschicht beibehalten und an geeigneten Stellen erweitert. So erhält beispielsweise das Daten-Frame-Format ein erweitertes Adressierungsschema sowie Ergänzungen um Informationen über die genutzte Mesh-Konfiguration. Nachfolgend werden die wichtigsten Bestandteile der Mesh-Spezifikation genauer erläutert. Weiterführende Informationen finden sich im WLAN-Standard [5] und in der Literatur [12, 113, 114]. Abschließend wird ein Überblick über die 802.11s-Implementierung unter Linux gegeben.

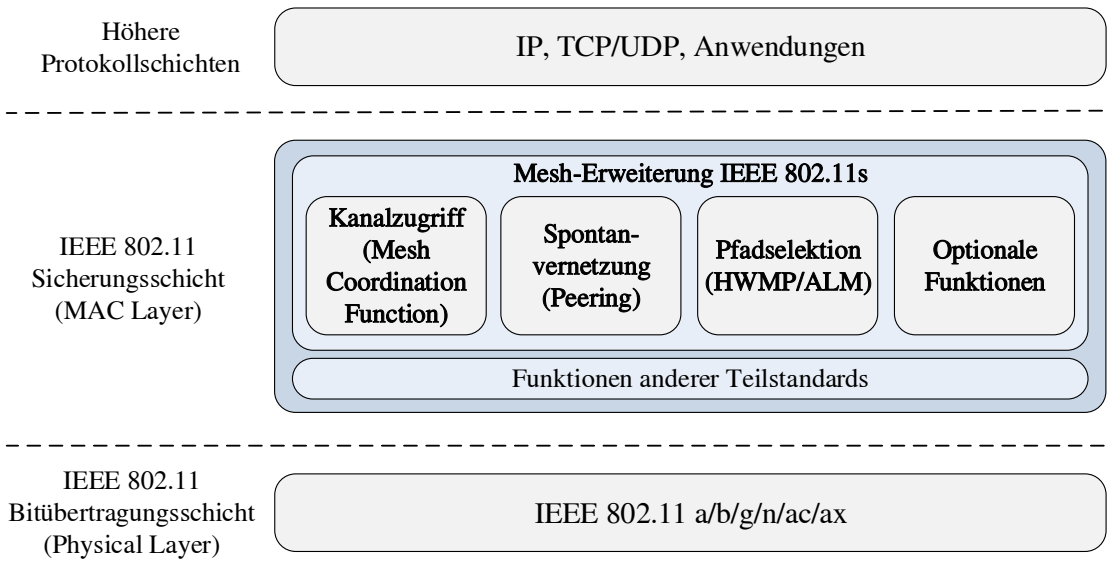


Abbildung 2.9: Übersicht der Hauptbestandteile von IEEE 802.11s [B 17]

2.5.2 Knotenrollen

In der Terminologie des WLAN-Standards bilden gleichberechtigte *Mesh Stations* ein sogenanntes *Mesh Basic Service Set (MBSS)*. In der Literatur werden Mesh Stations oftmals auch *Mesh Points* genannt. Im Rahmen der vorliegenden Arbeit wird synonym zu diesen Bezeichnungen der Begriff „Mesh-Knoten“ genutzt. Im Gegensatz zu herkömmlichen WLAN-Stationen besitzen diese die Fähigkeit der Frame-Weiterleitung und Pfadselektion.

Abweichend von der in Kap. 2.2 beschriebenen Architektur eines Backbone-WMN wird in 802.11s-Netzwerken technisch nicht zwischen statischen Mesh Routern und mobilen Mesh Clients unterschieden. Stattdessen wird die erweiterte Rolle *Mesh Gate* für Knoten mit Gateway-Funktionalität definiert [5]. Solche können die Verbindung zu einem beliebigen anderen Netzwerk aus der IEEE-802-Standardfamilie herstellen und z. B. als Bridge in ein Ethernet-LAN, Infrastruktur-WLAN oder separates 802.11s-MBSS fungieren. In der Literatur finden sich gelegentlich auch die Bezeichnungen *Mesh Access Point* und *Mesh Portal*, um die Anbindung von WLAN- oder kabelgebundenen Teilnehmern klarer zu unterscheiden. Die Möglichkeit zur Bekanntgabe von Gateway-Knoten im Netzwerk ist im Zuge der Pfadselektion vorgesehen (siehe Kap. 2.5.5).

2.5.3 Medienzugriff

Für den Medienzugriff von Mesh-Knoten definiert die Mesh Coordination Function (MCF) zwei verschiedene Mechanismen. Als Standardverfahren ist das aus der Teilspezifikation IEEE 802.11e stammende *EDCA* vorgeschrieben und muss in jedem Fall unterstützt werden. Bei diesem konkurrieren alle Teilnehmer nach dem klassischen CSMA/CA-Prinzip um die Nutzung des Kanals (siehe Kap. 2.4.5). Optional kann das Verfahren *MCF Controlled Channel Access (MCCA)* unterstützt werden. Dieses definiert, analog zu den bereits bestehenden optionalen Verfahren PCF und HCCA für Infrastruktur-WLANs, die Aushandlung von Zeitfenstern, in denen der Medienzugriff konkurrenzfrei erfolgen kann. Es sind bislang jedoch keine praktischen Implementierungen von MCCA bekannt.

Generell beschränkt sich IEEE 802.11s auf die Spezifikation von Mechanismen zur Kommunikation von Mesh-Knoten auf einem gemeinsamen Kanal. Darüber hinaus definiert der WLAN-Standard lediglich optionale Kontrollinformationen, sogenannte *Channel Switch Announcements (CSAs)*, für die Ankündigung eines Wechsels von APs auf einen neuen Kanal. Zwar ist die Nutzung von CSAs auch für Mesh-Netzwerke vorgesehen, es existieren jedoch keine Vorgaben oder Empfehlungen, anhand welcher Kriterien ein solcher Kanalwechsel durch Knoten einzuleiten ist. Ebenfalls werden keine weiterführenden Strategien vorgeschlagen, die eine gezielte Nutzung und Koordination mehrerer Netzwerkschnittstellen auf verschiedenen Kanälen realisieren. Insbesondere in Mesh-Netzwerken mit einer hohen Teilnehmerzahl ist die effiziente Nutzung des Frequenzspektrums durch überlappungsfreie Kanäle jedoch essentiell für die Dienstgüte. Die vorliegende Arbeit widmet sich dieser Problemstellung in Kap. 5.

2.5.4 Discovery und Verbindungsaufbau

Im WLAN-Standard existieren bereits für den herkömmlichen Betriebsmodus mit einem AP zwei Varianten von Nachrichten für das Finden der Basisstation. Einerseits erlauben periodisch durch den AP gesendete *Beacons* dessen passives Auffinden durch Client-Stationen. Als zweite Variante existieren sogenannte *Probe-Request*- und *Probe-Response*-Nachrichten für die aktive Suche nach Basisstationen (siehe Kap. 2.4.3). In einem 802.11s-Netzwerk agieren wiederum alle Mesh-Teilnehmer als Kommunikationspartner und können sich direkt miteinander verbinden. Die beschriebenen Varianten zur Knotensuche finden auch hier Anwendung. Die Frame-Formate der Beacons und Probe-

Element ID	Length	Active Path Selection Protocol ID	Active Path Selection Metric ID	Congestion Control Mode ID	Synchroniz. Method ID	Authentic. Protocol ID	Mesh Formation Info	Mesh Capability
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

Abbildung 2.10: Mesh Configuration Element in Beacon- und Probe-Nachrichten [5]

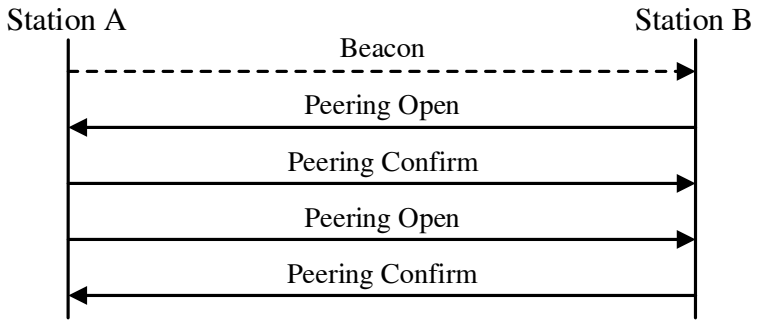


Abbildung 2.11: Mesh Peering mittels 4-Wege-Handshake

Nachrichten besitzen dabei variable Felder, die im 802.11s-Modus mit Informationen über die Mesh-Konfiguration der Knoten, das sogenannte *Mesh Profile*, belegt werden. Alle Teilnehmer eines MBSS müssen sich auf demselben Kanal befinden und ein übereinstimmendes Profil besitzen. Dieses besteht aus der *Mesh ID*, analog zur SSID herkömmlicher WLANs, und dem *Mesh Configuration Element* (siehe Abb. 2.10). Letzteres gibt Aufschluss über das verwendete Verfahren zur Pfadselektion, die dabei genutzte Kostenmetrik sowie optionale Mechanismen zur Staukontrolle, Zeitsynchronisation zwischen Nachbarknoten sowie Authentifizierung und Verschlüsselung. Mithilfe der Felder *Mesh Formation Info* und *Mesh Capability* können Knoten außerdem Informationen zum aktuellen Netzwerkstatus (Anzahl verbundener Knoten, Vorhandensein von Mesh Gates) austauschen und weitere Fähigkeiten (Energiesparmodi, Unterstützung des Medienzugriffs mittels MCCA) signalisieren.

Haben sich Knoten mit übereinstimmender Konfiguration z. B. durch den Empfang von Beacons gefunden, können diese eine aktive Verbindung aufbauen. Diese wird auch als *Link* bezeichnet. Für das *Mesh Peering* wurden neue Management-Frames in der Untergruppe der Action-Frames definiert. Dabei werden, wie in Abb. 2.11 dargestellt, in einem Vier-Wege-Handshake sogenannte *Peering-Open*- und *Peering-Confirm*-Nachrichten ausgetauscht. Nach erfolgreichem Verbindungsaufbau wird der gebildete Link regelmäßig durch Beacons sowie durch anderweitige Frame-Übertragungen zwischen den Nachbarknoten aktualisiert. Infolge von Beacon-Verlusten, Übertragungsfehlern oder sich ändernder Mesh-Konfiguration können Links wieder entfernt bzw. explizit abgebaut werden (*Peering-Close*-Nachricht) [114].

Neben einem unsicheren Verbindungsaufbau spezifiziert 802.11s auch die optionale Authentifizierung und Verschlüsselung von Mesh-Links. Als Standardverfahren wird der Mechanismus *Simultaneous Authentication of Equals (SAE)* [115] vorgeschlagen, bei dem sich Mesh-Knoten basierend auf einem vordefinierten Passwort gegenseitig authentifizieren. Im Rahmen dieser Arbeit standen Sicherheitsmechanismen jedoch nicht im Vordergrund und es wurde in allen Untersuchungen ein unverschlüsseltes Netzwerk genutzt.

2.5.5 Pfadselektion mittels HWMP/ALM

Die zielgerichtete Nachrichtenweiterleitung wird als *Routing* bezeichnet und ist üblicherweise eine Aufgabe der Vermittlungsschicht (siehe Kap. 2.1). Erfolgt das Routing wie bei 802.11s auf der Sicherungsschicht, wird oft zur Unterscheidung von *Pfadselektion* gesprochen. Im weiteren Verlauf dieser Arbeit werden beide Begriffe jedoch auch für 802.11s-Netzwerke synonym benutzt. Um Interoperabilität zu garantieren, werden im WLAN-Standard mit dem *Hybrid Wireless Mesh Protocol (HWMP)* und der *Airtime Link Metric (ALM)* bereits Mechanismen zur Pfadselektion vorgegeben, die von jedem Mesh-Knoten unterstützt werden müssen.

Die ALM (siehe Gl. 1) ist eine speziell für den Einsatz auf der Sicherungsschicht ausgelegte Kostenmetrik und wird auf jedem Mesh-Knoten für seine an aktiven Routen beteiligten Links berechnet. Die Metrik gibt die Zeitkosten c_a für die Übertragung eines Frames der Bitgröße B auf einem Peer Link an und berücksichtigt neben dem Overhead O der Funktechnologie (z. B. Zeitbedarf durch Kanalzugriff, Flusskontrolle und 802.11-Protokoll-Header) auch die geschätzte Frame-Fehlerwahrscheinlichkeit e (zwischen 0 und 1) und die WLAN-Sendedatenrate r [116, 117].

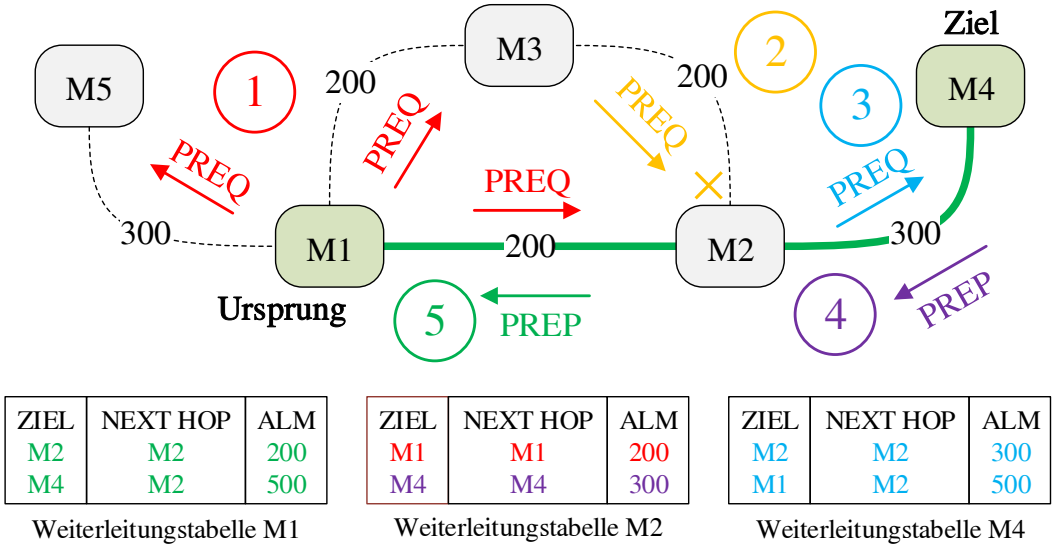
$$c_a = \left[O + \frac{B}{r} \right] \cdot \frac{1}{1-e} \tag{1}$$

Für die Frame-Größe B wird im WLAN-Standard ein Wert von 8192 bits (1 KiB) angenommen. Angegeben wird die ALM in der Einheit $10\,\mu$s [5]. Da die ALM eine kumulative Metrik ist, ergeben sich die Gesamtkosten eines Mesh-Pfades aus der Summe der Teilkosten seiner Links (Hops).

Beim HWMP handelt es sich um ein hybrides Protokoll, das eine Kombination reaktiver und proaktiver Routing-Ansätzen erlaubt. In seinen Grundzügen basiert HWMP auf dem Distance-Vector-Verfahren *AODV*, arbeitet jedoch auf der Sicherungsschicht und es werden MAC-Adressen statt IP-Adressen verwendet. Jeder Knoten verwaltet stets nur das Wissen über den besten Nachbarknoten (auch *Next Hop*), an den Frames zu einem bestimmten Zielknoten weitergeleitet werden müssen. Wie für das Peering wurden auch für die Pfadselektion neue Management-Frames in der Untergruppe der Action-Frames definiert. Diese umfassen die Nachrichtentypen *Path Request (PREQ)*, *Path Reply (PREP)*, *Path Error (PERR)* und *Root Announcement (RANN)*.

Der im Standardfall genutzte *reaktive* HWMP-Modus muss von jedem Knoten unterstützt werden. Pfade zu unbekannten Zielknoten werden in diesem Modus erst bei Bedarf ermittelt. In der Praxis wird dieser Vorgang in der Regel durch Kommunikationsvorhaben auf höheren Protokollschichten ausgelöst. Bei IP-basierten Anwendungen muss zunächst die Auflösung der IP-Adresse des Zielknotens zu seiner MAC-Adresse erfolgen. Im Fall von IPv4 dient hierzu das *Address Resolution Protocol (ARP)*. Für die Pfadermittlung sendet ein Mesh-Knoten im reaktiven Modus eine PREQ-Nachricht als Broadcast an alle seine Nachbarn, welche den Zielknoten als Anfrage enthält. Besitzen Zwischenknoten bereits einen gültigen Pfad zum Ziel, antworten sie dem anfragenden Knoten direkt mit einem PREP per Unicast. Ansonsten wird das PREQ so lange weitergeleitet, bis entweder ein anderer Zwischenknoten mit Pfadkenntnis oder schließlich der Zielknoten selbst erreicht wird und mit einer PREP-Unicast-Nachricht antwortet. Wird nicht rechtzeitig eine PREP-Nachricht empfangen, wird der Zielknoten als nicht erreichbar gewertet.

Abb. 2.12 zeigt die HWMP-Pfadermittlung am Beispiel. In diesem wird ein Pfad von Knoten M1 (Ursprung) zu Knoten M4 (Ziel) aufgebaut. Der Vorgang ist in fünf Schritte unterteilt, welche farbig hervorgehoben sind. Neben den pro Schritt gesendeten HWMP-Nachrichten sind auch die Pfadtabellen der Knoten M1, M2 und M4 dargestellt. Die Farbe einer Weiterleitungsregel (bestehend aus Ziel, Next Hop und ALM-Kosten) entspricht dabei dem jeweiligen Schritt, in dem diese angelegt wird. In den Schritten 1–3 wird das Netzwerk ausgehend von M1 mit PREQ-Nachrichten geflutet. Im Schritt 2 wird angenommen, dass der Pfad über M3 eine schlechtere Metrik aufweist und dessen



ZIEL	NEXT HOP	ALM
M2	M2	200
M4	M2	500

Weiterleitungstabelle M1

ZIEL	NEXT HOP	ALM
M1	M1	200
M4	M4	300

Weiterleitungstabelle M2

ZIEL	NEXT HOP	ALM
M2	M2	300
M1	M2	500

Weiterleitungstabelle M4

Abbildung 2.12: Beispiel einer HWMP-Pfadermittlung

weitergeleitetes PREQ bei M2 später eintrifft als das von M1 zuerst gesendete PREQ. Daher wird die durch M3 weitergeleitete Nachricht von M2 verworfen. Im Schritt 4 erreicht das durch M2 weitergeleitete PREQ des kürzeren Pfades den Zielknoten M4. In den Schritten 4 und 5 wird die PREP-Antwort von M4 über M2 an M1 gesendet und die bidirektionale Pfaderstellung abgeschlossen.

Im Zuge der Weiterleitung der PREQ- und PREP-Nachrichten aktualisieren Zwischenknoten das enthaltene Metrik-Feld durch Addition der ALM ihres Hops. Während der PREQ-Phase erstellen alle Knoten (inkl. Ziel) einen Pfadeintrag mit den kumulativen Kosten in Richtung des Ursprungsknotens der Anfrage (auch *Reverse Path*). Die Antwort des Ziels mittels Unicast-PREP erfolgt auf dem in der PREQ-Phase ermittelten Pfad und erstellt wiederum auf allen Knoten (inkl. Ursprung) den entsprechenden Pfadeintrag in Richtung Ziel (auch *Forward Path*). Die individuellen ALM-Kosten können sich daher für Forward und Reverse Path unterscheiden. Um das Aussortieren von Duplikaten und die Vermeidung von Zyklen zu ermöglichen, besitzen PREQ- und PREP-Nachrichten eine Sequenznummer sowie eine Time-to-Live (TTL). Dabei aktualisieren Knoten einen Pfadeintrag nur nach Empfang neuerer Informationen oder dem Kennenlernen geringerer ALM-Kosten (bei gleicher Sequenznummer). Die auf den Knoten gespeicherten Weiterleitungsregeln besitzen darüber hinaus eine konfigurierbare Gültigkeitsdauer. Sie müssen daher vor einem Timeout wie zuvor beschrieben aktualisiert werden. Verlieren Zwischenknoten zudem aufgrund eines Link-Abbruchs ihren Pfadeintrag zwischen Quelle und Ziel, generieren sie vorzeitig eine *PERR*-Nachricht in Richtung des Ursprungsknotens.

Neben dem reaktiven Modus unterstützt HWMP auch die *proaktive* Pfadermittlung. Dieser Modus kann als Ergänzung zum reaktiven Modus optional hinzugezogen werden. Dadurch reduziert sich einerseits die Latenz vor Datenübertragungen, andererseits werden mehr HWMP-Kontrollnachrichten gesendet. Je nach Anwendungsfall können sich auf diesen Modus konfigurierte Knoten periodisch als Wurzelknoten (engl. *Root Node*) im Netzwerk ankündigen und die Erstellung uni- oder bidirektionaler Pfade in einer Baumstruktur erzwingen. Ein mögliches Szenario ist das Vorhalten von Pfadeinträgen zu *Mesh Gates*, über die möglicherweise häufiger kommuniziert wird. Diese können ihre Gateway-Funktion explizit ausweisen und als Proxy für Zielknoten außerhalb des Mesh-Netzwerks dienen. Es existieren drei proaktive Betriebsvarianten:

- Im Modus *Proactive PREQ without PREP* sendet ein Wurzelknoten periodisch PREQ-Broadcast-Nachrichten adressiert an alle Teilnehmer im Netzwerk. Es wird jedoch keine Antwort mittels PREP erwartet, sodass alle Knoten lediglich den Reverse Path zur Wurzel erstellen. Ist dies für den Anwendungsfall ausreichend, erzeugt dieser Modus den geringsten Overhead.
- Der Modus *Proactive PREQ with PREP* gleicht der vorherigen Variante, es wird jedoch ein PREP von jedem Knoten verlangt. Somit werden stets bidirektionale Pfade zwischen dem Wurzelknoten und allen Teilnehmern vorgehalten.
- Die Variante *Proactive RANN* stellt einen Mittelweg zwischen den ersten beiden Modi dar. Sie unterscheidet sich dahingehend, dass periodisch RANN-Nachrichten gesendet werden, die jedoch nur die Kostenmetrik zum Wurzelknoten im Netzwerk verbreiten bzw. aktualisieren sollen. Eine bidirektionale Pfaderstellung mittels Unicast-PREQ und PREP erfolgt ausgehend von den empfangenden Mesh-Knoten nur bei Änderungen der Pfadinformationen.

Die explizite Ankündigung von Knoten als Gateway erfolgt entweder über separate *Gate Announcements (GANNs)* oder alternativ durch Nutzung eines *Gate-Announcement*-Feldes innerhalb der proaktiven PREQ- bzw. RANN-Nachrichten. Wird der Zielknoten einer Pfadermittlung nicht gefunden und befindet sich möglicherweise außerhalb des Mesh-Netzwerks, d.h. es wird keine PREP-Nachricht als Antwort auf ein PREQ empfangen, wird der Frame erneut an bekannte Gateway-Knoten geschickt, die diesen an das jeweilige externe Netzwerk weiterleiten. Dabei wird, wie nachfolgend in Kap. 2.5.6 beschrieben, eine erweiterte Adressierung genutzt. Sind keine Gateways bekannt oder das Ziel ist auch über diese nicht erreichbar, wird die Übertragung verworfen.

Die Thematik der Designation von Gateway-Knoten lag nicht im Fokus der vorliegenden Arbeit. Der gezielte Einsatz der proaktiven HWMP-Modi erfolgt jedoch im Rahmen des in Kap. 5 vorgestellten Konzepts zur Optimierung von WLAN-Mesh-Backbones mittels Clustering und Kanalselektion.

2.5.6 Adressierung und Frame-Weiterleitung

Nach der Ermittlung eines Mesh-Pfades können Frames über mehrere Zwischenstationen weitergeleitet werden. Ist die Kommunikation auf dasselbe Mesh-Netzwerk beschränkt, kommen lediglich vier Adressen zum Einsatz. Der *4-Adress-Modus* ist bereits mit dem herkömmlichen WLAN-Frame-Format realisierbar und findet auch in Installationen Anwendung, bei denen mehrere APs durch ein WDS verbunden sind (siehe Kap. 2.4.3). Unterschieden werden die Adressen *Source*, *Destination*, *Transmitter* und *Receiver*. Beim Transmitter und Receiver handelt es sich im Mesh-Netzwerk um die aktuellen Zwischenknoten, die den Frame auf dem Weg von der Quelle zum Ziel weiterleiten. Diese Adressen ändern sich somit in jedem Weiterleitungsschritt.

Liegen Quelle oder Ziel dagegen außerhalb des MBSS und werden über Gateway-Knoten angebunden, ist die Unterscheidung von bis zu sechs Adressen nötig. Dazu besitzen Frames im Mesh-Modus ein sogenanntes *Mesh Control Field* als Teil der Payload (Frame Body) direkt nach dem MAC-Header [114]. Neben Feldern für die Adresserweiterung enthält dieses auch die Mesh-Sequenznummer und -TTL des Frames. Die zusätzlichen Adressen 5 und 6 werden stets für Quell- bzw. Zielknoten außerhalb des MBSS genutzt. Dies bietet den Vorteil, dass deren Behandlung theoretisch nur an Gateway-Knoten erforderlich ist, während die Weiterleitung im Mesh-Netzwerk nur unter Berücksichtigung der herkömmlichen Header-Informationen erfolgen kann. Abb. 2.13 zeigt ein Adressierungsbeispiel, in dem sowohl Quelle als auch Ziel außerhalb des Mesh-Netzwerks liegen. Dabei werden die Gateways als Quell- und Zielknoten innerhalb des Mesh-Netzwerks adressiert, die externen Knoten wiederum als finale Endpunkte.

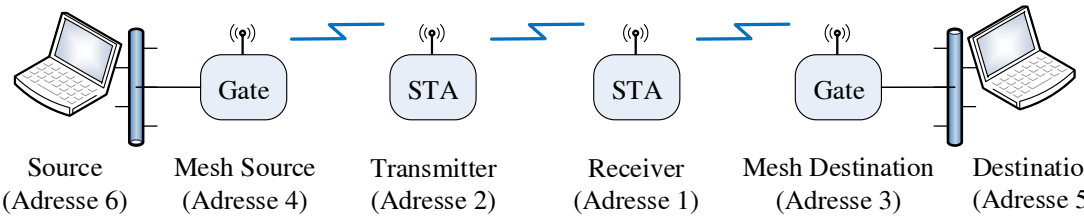


Abbildung 2.13: 6-Adress-Modus am Beispiel [114]

2.5.7 Implementierung im Linux-Kernel

Die Linux-Implementierung von 802.11s ist direkter Bestandteil des Kernels und bildet im Rahmen der vorliegenden Arbeit die Software-Basis für die Testumgebung *Mini-Mesh* (siehe Kap. 3) sowie alle entwickelten Prototypen (siehe Kap. 4 und 5). Sie befindet sich in stetiger Entwicklung, deckt jedoch bereits alle Pflichtmechanismen sowie einige optionale Funktionen der Mesh-Spezifikation ab [118]. So werden z. B. die Authentifizierung und Verschlüsselung mittels SAE-Verfahren, aber auch verschiedene Energiesparmodi für Mesh-Knoten unterstützt. Begonnen wurde die Entwicklung unter dem Projektnamen *open80211s* von der Firma *Cozybit* mit dem Ziel, eine erste Open-Source-Referenzimplementierung bereitzustellen [119]. Mittlerweile wird sie als integraler Teil des WLAN-Protokollstapels durch das *Linux-Wireless*-Team gepflegt [120]. Neben der Linux-Unterstützung für 802.11s existiert eine unabhängige Implementierung für das Betriebssystem *FreeBSD* [121].

Abb. 2.14 zeigt vereinfacht die Architektur des WLAN-Protokollstapels im Linux-Kernel. Eine zentrale Komponente ist das Kernelmodul `mac80211` [122], das einen Großteil von Funktionen der WLAN-Sicherungsschicht inklusive der 802.11s-Mesh-Funktionen in Software realisiert. Im Zusam-

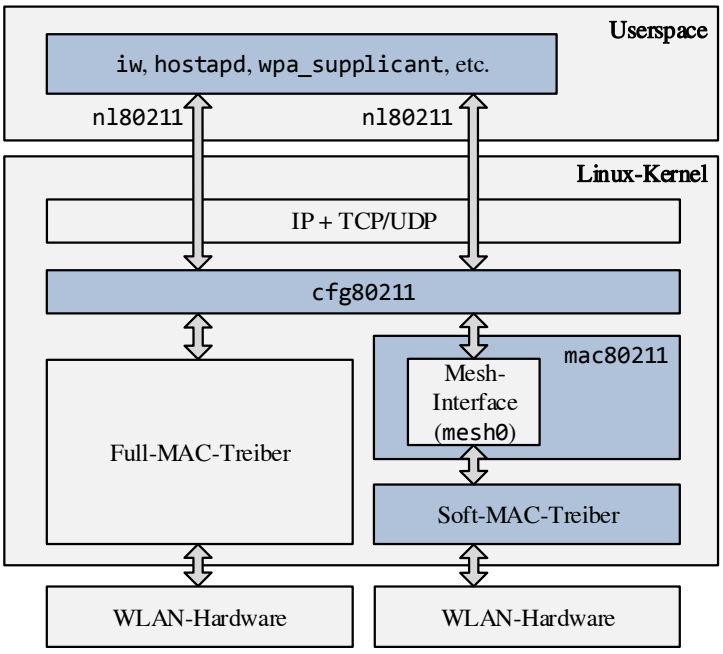


Abbildung 2.14: Architektur der WLAN-Implementierung unter Linux

Tabelle 2.5: Übersicht wichtiger Mesh-Parameter [5, 127]
(*: Linux-spezifischer Parameter)

Parameter	**Default-Wert**	**Beschreibung**
MaxRetries	3	Anzahl konsekutiver Peering-Versuche
RetryTimeout	100 ms	Timeout für Peering-Open-Nachrichten
ConfirmTimeout	100 ms	Timeout für Peering-Confirm-Nachrichten
HoldingTimeout	100 ms	Timeout beim Verbindungsabbau
Forwarding	true	Aktivierung der Frame-Weiterleitung
TTL	31	Max. Hop-Anzahl bei der Frame-Weiterleitung
GateAnnouncements	false	Ankündigung als Mesh Gate
HWMPmaxPREQretries	4	Anzahl konsekutiver PREQ-Versuche
HWMPpreqMinInterval	10 ms	Min. Wartezeit zwischen PREQ-Nachrichten
HWMPperrMinInterval	10 ms	Min. Wartezeit zwischen PERR-Nachrichten
HWMPactivePathTimeout	5 s	Lebensdauer von Pfadeinträgen
HWMProotMode	off	Aktivierung des proaktiven Modus (drei Modi)
HWMProotInterval	5 s	Sendeperiode proaktiver HWMP-Nachrichten
beacon_interval *	1 s	Sendeperiode für Beacon Frames
path_refresh_time *	1 s	Zeitabstand zur Pfadaktualisierung vor einem Timeout
plink_timeout *	30 min	Lebensdauer inaktiver Peer Links

menspiel mit Gerätetreibern werden zwei Implementierungsarten unterschieden. Sind die Funktionen der Sicherungsschicht bereits auf der Hardware des WLAN-Adapters herstellerseitig integriert und somit unabhängig von `mac80211`, spricht man von einem *Full-MAC*-Gerät. In diesem Fall steht der Mesh-Betriebsmodus bislang meist nicht zur Verfügung. Wird die Sicherungsschicht dagegen größtenteils oder komplett in Software realisiert, handelt es sich um ein *Soft-MAC*-Gerät. Auf der WLAN-Hardware werden im Gegenzug meist nur noch zeitkritische Aufgaben ausgeführt, wie z. B. Medienzugriff und Flusskontrolle der Sicherungsschicht sowie die Funktionen der Bitübertragungsschicht. Das Verhältnis der Aufgabenteilung zwischen Hardware, Treiber und Kernel ist dabei von Produkt zu Produkt unterschiedlich. Es existieren verschiedene Soft-MAC-Treiber, die im Zusammenspiel mit `mac80211` den Mesh-Modus nach 802.11s unterstützen. Dazu zählen Treiber für Chipsätze der Firmen Atheros (`ath5k`, `ath9k`, `ath10k`), Broadcom (`b43`) oder Ralink/MediaTek (`rt2x00`, `mt76`) [105]. Im Rahmen dieser Arbeit wurde der Atheros-Treiber `ath9k` genutzt [123].

Oberhalb von `mac80211` ermöglicht das Kernelmodul `cfg80211` Programmen im Linux Userspace den Zugriff auf die Konfiguration und den Zustand von Sicherungsschicht- und Geräteparametern. Die Kommunikation zwischen Userspace und Kernel erfolgt dabei nachrichtenbasiert über die Socket-Schnittstelle *Netlink* [124]. Im Application Programming Interface (API) `nl80211` sind wiederum die WLAN-spezifischen Netlink-Nachrichten definiert [125]. Anwendungen nutzen `nl80211` oftmals über die Bibliothek `libnl` [126]. Bekannte Beispiele sind `iw` zur Gerätekonfiguration, `hostapd` zur Realisierung von AP-Authentifizierungsdiensten oder `wpa_supplicant` zur Unterstützung von Sicherheitsmechanismen auf WLAN-Clients.

Unter Linux werden Netzwerkgeräte durch virtuelle Netzwerk-Interfaces repräsentiert. Vorhandene Geräte werden durch den Linux-Kernel erkannt und es werden automatisch Repräsentationen angelegt, die oftmals aufsteigend nummeriert sind. Typische Bezeichnungen sind z. B. „ethX“ für Ethernet- oder „wlanX“ für WLAN-Interfaces. Im Fall eines Ethernet-Gerätes existiert als Repräsentation in der Regel nur ein Netzwerk-Interface, bei WLAN-Geräten sind hingegen mehrere virtuelle Interfaces auf einem physischen Adapter explizit vorgesehen. Daher existieren hier auch separate Repräsentationen der physischen Geräte (z. B. „phyX“). Ein solcher WLAN-Adapter kann verschiedene virtuelle Interface-Typen unterstützen. Beispiele dafür sind der Typ *managed* für herkömmliche WLAN-Clients, *ap* für Interfaces im AP-Modus, *ibss* für Ad-Hoc-Stationen oder *mesh* für Interfaces im 802.11s-Modus. Die Möglichkeit, dass auf einem physischen WLAN-Adapter gleichzeitig mehrere virtuelle Interfaces gleichen oder unterschiedlichen Typs angelegt und aktiviert werden können, muss durch den jeweiligen Gerätetreiber unterstützt werden. Generell gilt jedoch die Beschränkung, dass sich gleichzeitig aktivierte virtuelle Interfaces auf demselben Kanal befinden müssen.

Eine im Rahmen der Arbeit besonders relevante Anwendung ist das Kommandozeilen-Tool `iw` [128]. Es dient der Verwaltung von physischen WLAN-Adaptern sowie deren virtuellen Interfaces. Als eines von wenigen Tools unterstützt es die umfangreiche Konfiguration von 802.11s-Mesh-Interfaces. Neben grundlegenden Einstellungen wie der Festlegung der Mesh ID oder des WLAN-Kanals können zahlreiche weiterführende Optionen angepasst werden. Der gezielte Netzwerkbei- und austritt sind ebenso möglich wie die Analyse der lokalen Mesh-Interfaces. So können Informationen über die sichtbaren Nachbarknoten in der *Station List* oder die aktuellen HWMP-Weiterleitungsregeln in der *Mesh Path List* ausgelesen werden. Ebenso lassen sich Nachbarknoten individuell blockieren oder statische Weiterleitungsregeln hinzufügen. Darüber hinaus können zahlreiche Mesh-Parameter angepasst werden. Während einige davon Linux-spezifisch sind, ist der Großteil bereits im WLAN-Standard definiert [5, 127]. Tab. 2.5 fasst die wichtigsten Parameter und deren Linux-Default-Werte zusammen. Diese galten für alle praktischen Untersuchungen im Rahmen dieser Arbeit.

3 Entwurf einer realen WLAN-Mesh-Testumgebung

3.1 Motivation

Die Wireless Local Area Network (WLAN)-Teilspezifikation IEEE 802.11n führte grundlegende technologische Neuerungen auf der Bitübertragungsschicht ein, die seitdem durch die Nachfolger 802.11ac und 802.11ax logisch fortgesetzt werden. Dazu zählen insbesondere höhere Modulationsordnungen, die Bündelung von Kanälen sowie der Einsatz von Multiple Input Multiple Output (MIMO)-Techniken [102, 129]. Im Vordergrund dieser Arbeit stand die Untersuchung von Optimierungsstrategien für 802.11s-Mesh-Netzwerke unter der Annahme von mindestens 802.11n-Funktionalität auf der Bitübertragungsschicht. Bei der Frage nach einer geeigneten Entwicklungs- und Evaluationsumgebung mit 802.11n/s-Unterstützung wurden mehrere Möglichkeiten betrachtet.

Ein kostengünstiger Weg zur Untersuchung auch komplexer Szenarien mit hoher Teilnehmerzahl ist der Einsatz von Netzwerksimulatoren. Im Gegensatz zu praktischen Messungen liefern diese jedoch oft nur eine vereinfachte Abbildung realer Effekte wie konkurrierenden Medienzugriff und Interferenzen [22, 23]. In der Regel werden Modelle sowohl für die Kanal- und Umgebungseffekte als auch den Netzwerkprotokollstapel und darüber liegende Anwendungen genutzt. In diesem Zusammenhang bieten etablierte freie Simulatoren wie *ns-3* [130] und *OMNeT++* [131] nur unvollständige oder veraltete Modelle für 802.11s [24, 25]. Für die Entwicklung interoperabler Optimierungsansätze ist zudem die Berücksichtigung real vorhandener Hard- und Software-Schnittstellen von hoher Bedeutung. Auch diese werden in Simulatoren nicht oder nur stark eingeschränkt abgebildet.

Eine weitere Möglichkeit besteht in einem System, in dem Umgebungseffekte und geeignete Abstraktionen der Hardware von Netzwerkteilnehmern simuliert werden. Die Software der simulierten Zielsysteme bleibt dabei unverändert. So lassen sich z. B. Netzwerkprotokollstapel und Anwendungsschicht mehrerer Teilnehmer durch Virtualisierungslösungen getrennt voneinander auf einem Simulationssystem ausführen. Zu diesem Zweck können virtuelle Maschinen oder *Linux Containers* (LXC) und darauf aufsetzende Systeme zum Einsatz kommen [132]. Die Herausforderung besteht jedoch in der Verknüpfung der virtualisierten Teilnehmer mit einer Simulation der Kanaleffekte zwischen ihnen. Moderne Emulations- und Virtualisierungslösungen erreichen bei der Ausführung der Ziel-Software oft eine ähnliche Geschwindigkeit wie die Originalsysteme. Im Gegensatz dazu besitzen komplexe Kanalsimulationen einen hohen Rechenaufwand, der bei der zeitlichen Synchronisation solcher Simulationssysteme beachtet werden muss [133–135]. Diese Problematik stellt ein eigenes, komplexes Forschungsfeld dar, in dem der Autor im Rahmen von Kooperationen am Institut für angewandte Mikroelektronik und Datentechnik bereits erste Konzepte veröffentlichen konnte [B 10, 11].

Für die Entwicklung und Evaluation der in dieser Arbeit vorgestellten Optimierungsansätze wurde daher der Weg einer realen 802.11n/s-Testumgebung gewählt. Als Basis diente die praktische Referenzimplementierung von 802.11s im Linux-Kernel. Die entwickelten Lösungen wurden prototypisch auf Standard-Hardware und -Software implementiert und nutzen ausschließlich vorhandene Schnittstellen auf der Anwendungsschicht. Dadurch sind keinerlei Modifikationen am Protokollstapel des Betriebssystems notwendig.

Im Hinblick auf Einsatzszenarien im Smart-City-Kontext ergab sich die Anforderung, komplexe Mesh-Strukturen untersuchen zu können, wie sie bereits für die Abdeckung kleiner Stadtviertel oder öffentlicher Plätze benötigt werden. Der Aufbau und Betrieb einer im realen Stadtgebiet verteilten Geräteanordnung ist jedoch mit einem hohen finanziellen und zeitlichen Aufwand verbunden. Ebenso wäre eine reproduzierbare Durchführung von Experimenten nur sehr schwer zu gewährleisten. Daher bestand die Zielstellung darin, realistische Multi-Hop-Topologien auch im Labormaßstab abzubilden. Bisherige Ansätze zur gezielten Miniaturisierung realer Testumgebungen umfassen hingegen keine Forschungsarbeiten, die 802.11s und 802.11n (oder neuer) kombinieren [B 6].

Nachfolgend wird das im Rahmen der Arbeit entwickelte, miniaturisierte 802.11n/s-Testbed *Mini-Mesh* vorgestellt. Mithilfe eines Ansatzes zur Skalierung der Kommunikationsreichweite wird der Aufbau einer ebenen quadratischen Geräteanordnung mit 6x6 Knoten auf einer Laborfläche von lediglich 1 m^2 realisiert. In einem Aufbau ohne Reichweitenbeschränkung entspricht dies einer Fläche von mehr als 300.000 m^2, vergleichbar mit einem Stadtviertel, Park oder Campusgelände.

Das Skalierungskonzept und dessen experimentelle Validierung wurden in [B 6] veröffentlicht. Die Auswertung der in Kap. 4 und 5 beschriebenen Optimierungsansätze erfolgte jeweils unter Nutzung der miniaturisierten Testumgebung. Diese Ergebnisse wurden ebenfalls veröffentlicht [B 3, 4]. Ferner wurde die Hardware-Plattform der Testumgebung in verschiedenen Kooperationen des Autors am Institut für angewandte Mikroelektronik und Datentechnik genutzt [B 7–9, 13, 14, 19].

3.2 Stand der Technik

Die Surveys [136–139] geben einen Überblick praktischer Testumgebungen im Forschungsumfeld. Die darunter diskutierten Wireless Mesh Network (WMN)-Aufbauten nutzen jedoch nicht 802.11s und nur wenige von ihnen verfolgen einen Ansatz zur Miniaturisierung. Diese und weitere verwandte Arbeiten sind in Tab. 3.1 zusammengefasst. Sie sind zeilenweise gruppiert basierend auf den unterstützten 802.11-Modi, der Skalierungsmethode (falls zutreffend) sowie den Umgebungen (*Indoor* oder *Outdoor*), in denen Experimente durchgeführt wurden. Für jede Gruppe sind weitere Informationen zur Knotenanzahl und zu eingesetzten Routing-Protokollen angegeben.

Als erste Kategorie (Zeilen 1–3) sind Arbeiten aufgeführt, die 802.11n und 802.11s nutzen, jedoch keinen Skalierungsansatz verfolgen. Lediglich [146] und [147] reduzieren in Experimenten mit fliegenden Drohnen die Sendeleistung, um die Direktverbindung zu einer Bodenstation zu verhindern.

Tabelle 3.1: Gegenüberstellung von *Mini-Mesh* mit verwandten Forschungsarbeiten [B 6]
(*: 802.11s-Vorversion, A: Antennendämpfung, K: Kabeldämpfung, S: Sendeleistung)

Arbeiten	**802.11-Modi**	**Mesh Routing**	**# Knoten**	**Skalierungs-methode**	**Umgebung**
Lin et al. [140], Imboden et al. [141]	n + s*	statisch / N/A	6–9	—	Indoor
Chakraborty et al. [142], Sajjadi et al. [143], Krug et al. [144], Makris et al. [145]	n + s	HWMP	2–120	—	Indoor / Outdoor
Pojda et al. [146], Hayat et al. [147]	n + s	HWMP	3–6	S	Indoor / Outdoor
EWANT [148], MiNT [149], Bonsai [150], Meraka [151], IvyNet [152], ScaleMesh [153], Bialkowski et al. [154], FloorNet [155], Intel Research, Seoul National University [137]	a / b / g	AODV, OLSR, DSR, u.a.	3–210	A / K / S	Indoor
MeshTest [156], ORBIT [136], WHYNET / Castadiva / MNE [138]	a / b / g	AODV, OLSR	10–64	A / K / S	Indoor / Outdoor
Mini-Mesh [B 6]	a / n + s	HWMP	36	A + S	Indoor (Vergleich Outdoor)

Einige der Testumgebungen verwenden zudem eine präfinale Version von 802.11s oder unterstützen noch keinen gleichzeitigen Betrieb mit 802.11n, sondern nur mit früheren WLAN-Modi [140, 141].

Die zweite Kategorie (Zeilen 4–5) umfasst ältere Arbeiten, die sich bereits gezielt mit Skalierungsmethoden für WLAN-Mesh-Testbeds beschäftigen. Neben üblichen Ansätzen wie der Reduzierung der Sendeleistung oder der Nutzung von Dämpfungsgliedern im Antennenweg werden auch kabelgebundene Aufbauten beschrieben. Hier sind die Geräte über Koaxialkabel und Multiplexer direkt miteinander verbunden und die Signalqualität der künstlichen Funkstrecken wird mittels Dämpfung der Kabelwege variiert. Alle Arbeiten der zweiten Gruppe unterstützen nur die WLAN-Modi 802.11a/b/g und weder 802.11n noch 802.11s. Das Mesh-Routing wird oberhalb des WLAN-Ad-Hoc-Modus durch Protokolle auf der Vermittlungsschicht realisiert.

Der in diesem Kapitel vorgestellte Ansatz *Mini-Mesh*, aufgeführt in der letzten Zeile, verfolgt hingegen die Miniaturisierung einer 802.11n/s-Testumgebung. Durch den Einsatz von Dämpfungsgliedern im Antennenweg und einer Reduktion der Sendeleistung wird die Kommunikationsreichweite der Knoten auf einen Labormaßstab skaliert. Zur Validierung des Ansatzes werden Vergleichsmessungen auf einer miniaturisierten Indoor- und einer nicht-miniaturisierten Outdoor-Kommunikationsstrecke durchgeführt und Kalibrierungswerte für die verwendete Hardware abgeleitet. Für verschiedene Konfigurationen wird der Skalierungsfaktor experimentell bestimmt und das Protokollverhalten auf Sicherungs- und Transportschicht untersucht. Basierend auf den gemessenen Reichweiten erfolgt die Parametrisierung eines Umgebungsmodells [153]. Mithilfe des Modells lassen sich korrespondierende reale Abmessungen für verschiedene miniaturisierte Laboranordnungen bestimmen.

3.3 Miniaturisierte IEEE-802.11n/s-Testumgebung *Mini-Mesh*

Im Folgenden wird der Entwurf der miniaturisierten Testumgebung *Mini-Mesh* beschrieben. Nach Auswahl einer geeigneten Geräteplattform in Kap. 3.3.1 wird in Kap. 3.3.2 der grundlegende Aufbau der Testumgebung skizziert. Basierend auf einer Performance-Analyse werden in Kap. 3.3.3 geeignete Arbeitspunkte für die weiteren Untersuchungen ermittelt. Das Konzept zur Reichweitenskalierung und dessen technische Umsetzung sowie Kalibrierung werden in Kap. 3.3.4 und 3.3.5 beschrieben.

3.3.1 Anforderungsbeschreibung und Auswahl der Geräteplattform

Beim Entwurf der Testumgebung musste ein Kompromiss zwischen der möglichst guten Annäherung an die Komplexität und Teilnehmerzahl städtischer WLAN-Mesh-Backbones einerseits und den positiven Eigenschaften kontrollierter Laborbedingungen andererseits gefunden werden. Folgende Hauptanforderungen stellten sich an den zu entwickelnden Laboraufbau:

1. Nutzung kostengünstiger Standard-Hardware
2. Unterstützung von IEEE 802.11n & IEEE 802.11s
3. Hohe Konfigurierbarkeit (Betriebssystem, Treiber)
4. Hohe Reproduzierbarkeit von Experimenten
5. Geringer Flächenbedarf & leichte Erweiterbarkeit
6. Übertragbarkeit auf Szenarien realer Dimension

Zur Erfüllung der Punkte 1–3 kamen Einplatinencomputer mit dem Betriebssystem Linux zum Einsatz. Diese sind aufgrund ihres geringen Kostenfaktors gegenüber anderen Plattformen zu bevorzugen. Darüber hinaus ist die Referenzimplementierung von 802.11s integraler Bestandteil des Linux-

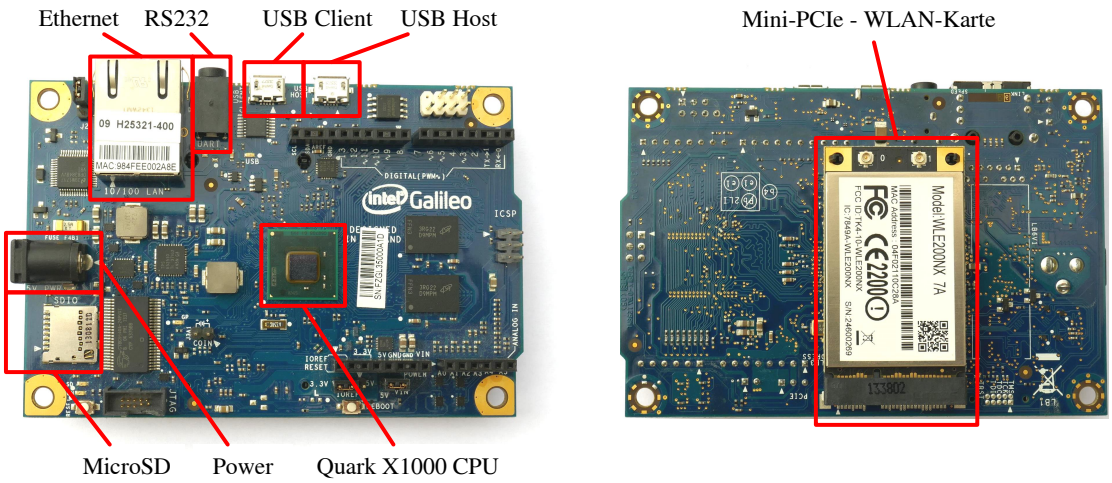


Abbildung 3.1: Intel Galileo Board: Vorderseite (links) und Rückseite (rechts) mit WLAN-Modul

Kernels. Zum anderen bieten die meist quelloffenen WLAN-Treiber unter Linux einen sehr hohen Konfigurationsspielraum. Vorrangig ist hier der Treiber `ath9k` für die 802.11n-Produktgeneration der Firma Atheros zu nennen [123]. Dieser nutzt zur Implementierung der nicht-zeitkritischen MAC-Funktionen die Software-Bestandteile des Linux-Kernels, während auf dem WLAN-Modul nur eine sehr schlanke, vorprogrammierte Firmware ausgeführt wird. Treiberarchitekturen für Chipsätze anderer Hersteller lagern häufig einen größeren Teil der Funktionen in eine proprietäre Firmware aus, die auf die Hardware geladen wird und mit dem Treiber interagiert. Im Fall von `ath9k` lassen sich, neben üblichen Parametern wie der Datenrate oder Sendeleistung, auch tiefgreifende Mechanismen der Hardware beeinflussen. Beispielsweise können automatische Kalibrierungsvorgänge auf dem WLAN-Modul deaktiviert werden [157]. Dies reduziert vorhandene Freiheitsgrade und erhöht die Reproduzierbarkeit von Experimenten (Anforderung 4). Aufgrund seines Funktionsumfangs und der guten Erweiterbarkeit wird `ath9k` oft in Forschungsarbeiten eingesetzt.

Um den Anforderungen 5 und 6 gerecht zu werden, musste die Kommunikationsreichweite geeignet auf einen Labormaßstab skaliert werden. Da eine reine Reduzierung der Sendeleistung hierfür nicht ausreichte, war der Einsatz von Dämpfungsgliedern im Antennenweg erforderlich. WLAN-Module mit Atheros-Chipsatz und Anschlüssen für externe Antennen sind jedoch praktisch nur als PCI-Express (PCIe)-Karten erhältlich. Dieser Umstand schränkte die Auswahl der Geräteplattform stark ein. Im Zeitrahmen dieser Arbeit verfügbare, kostengünstige Einplatinencomputer wie der Raspberry Pi und Derivate boten keine PCIe-Schnittstelle [158]. Geräte aus der ALIX-/APU-Board- oder Hummingboard-Familie unterstützten zwar PCIe, wurden jedoch zu Stückpreisen jenseits der 100 € angeboten [159, 160]. Eine Alternative stellte das von Intel entwickelte Galileo Board dar [161]. Die Leistungsklasse dieser Geräte liegt leicht unterhalb der eines Raspberry Pi v1. Sie bieten jedoch in der gleichen Preiskategorie eine Mini-PCIe-Schnittstelle und ebenfalls Unterstützung für Standard-Linux. Durch ein Intel-Förderprogramm standen für diese Arbeit 40 Galileo Boards zur Verfügung.

3.3.2 Zusammensetzung und Geometrie des Aufbaus

Der in dieser Arbeit entwickelte Testaufbau *Mini-Mesh* besteht aus 36 der verfügbaren Galileo Boards. Bei der Anordnung handelt es sich um ein reguläres 6x6-Knoten-Gitter, das die Untersuchung verschiedener statischer Szenarien und Teiltopologien erlaubt. Abb. 3.1 zeigt die Vorder- und Rückseite des Galileo Boards unter Hervorhebung der wichtigsten Hardware-Schnittstellen. Ansichten der gesamten Laboranordnung finden sich in Abb. 3.2. Eine Übersicht der grundlegenden Hardware- und

Tabelle 3.2: Hardware- und Software-Eigenschaften der Testumgebung [B 6]

Komponente	**Beschreibung**
Geräteplattform	Intel Galileo Board (1. Generation) [161]
CPU	Quark X1000 (x86 Single-Core 400 MHz)
RAM	256 MB DDR3-800
Betriebssystem	Debian 8 (Linux-Kernel v4.9.27)
Interfaces	PCIe 2.0, USB 2.0 Host, Fast Ethernet
WLAN-Modul	Compex WLE200NX 802.11a/b/g/n Dual-Band [162]
WLAN-Chipsatz	Atheros AR9280 (Treiber `ath9k` [123])
Antennen	2 x 5 dBi Dual-Band Rundstrahler
Antennenkabel	2 x 20 cm U.FL zu RP-SMA
Dämpfungsglieder	2 x Mini-Circuits VAT-30+ (30 dB) [163]

Software-Eigenschaften der Geräte wird in Tab. 3.2 gegeben. Als Betriebssystem dient im Rahmen dieser Arbeit Debian-Linux 8 mit Kernel-Version 4.9.27 [3.1] [3.2]. Alle Boards sind über ihre rückseitige Mini-PCIe-Schnittstelle mit einem Atheros-WLAN-Modul bestückt. Die Nutzung eines sekundären WLAN-Moduls ist mittels USB-Host-Schnittstelle möglich. Zusätzlich sind alle Geräte über Ethernet mit einem separaten Kontrollnetzwerk verbunden, das der skriptbasierten Steuerung und Automatisierung [3.3] von Experimenten dient. Über dieses erfolgt auch die Zeitsynchronisation der Geräte mittels Network Time Protocol (NTP) [164] und Precision Time Protocol (PTP) [165]. Dabei dient ein Labor-PC als NTP-Server zur Vorgabe einer absoluten Systemzeit sowie als PTP-Server für den relativen Zeitabgleich. Die Software-Implementierung `PTPd` verspricht bereits eine Synchronisationsgenauigkeit im Millisekundenbereich ohne Nutzung von Hardware-Zeitstempeln [166]. Dies erleichtert die zeitliche Korrelation von Paketmitschnitten und Log-Datei-Einträgen verschiedener Mesh-Knoten bei der Evaluation verteilter Anwendungen in der Testumgebung.

Das auf dem Atheros-Chipsatz AR9280 basierende, 802.11n-fähige WLAN-Modul unterstützt zwei Antennen mit den entsprechenden MIMO-Modi. So kann einerseits Antennendiversität (engl. *Diversity MIMO*) zum Senden und Empfangen des gleichen Datenstroms und andererseits *Spatial Multiplexing MIMO* zur Übertragung unabhängiger Datenströme genutzt werden (vgl. Kap. 2.4.4).

Durch die Dual-Band-Fähigkeit der Karte lassen sich sowohl Kanäle im 2,4- als auch im 5-GHz-Band wählen. Dies ermöglicht die Untersuchung von Kanalwahlstrategien und erlaubt störungsfreie Experimente auf Frequenzen, die nicht im Universitätsnetz verwendet werden. Um externer Interferenz bestmöglich vorzubeugen, ist auf den Geräten der Kanal 149 (5745 MHz) als Standardwert konfiguriert. Im Gegensatz zu den USA, Kanada und einigen anderen Ländern ist die Nutzung der Kanäle 149–165 in Europa für sogenannte *Short Range Devices* mit einer maximalen Sendeleistung von 25 mW zulässig [107]. Aufgrund dieser und anderer technischer Beschränkungen des WLAN-Moduls lässt sich die Sendeleistung pro Antennenweg mit einer Granularität von 1 dBm zwischen 0 dBm (1 mW) und 14 dBm (25 mW) wählen [3.4]. Für die summierte Gesamtleistung von zwei Antennen ergibt sich demnach ein Minimum von 3 dBm (2 mW) sowie ein Maximum von 17 dBm (50 mW).

[3.1] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/galileo-kernel`
[3.2] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/galileo-kernel-artifacts`
[3.3] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/mini-mesh-scripts`
[3.4] dBm: relativer Leistungspegel bezogen auf 1 mW

Abbildung 3.2: Testumgebung im Labor (oben: Gesamtansicht; unten links: Teilansicht Board-Einpassung im Gehäuse; unten rechts: Teilansicht Antennen-Dämpfungsglieder)

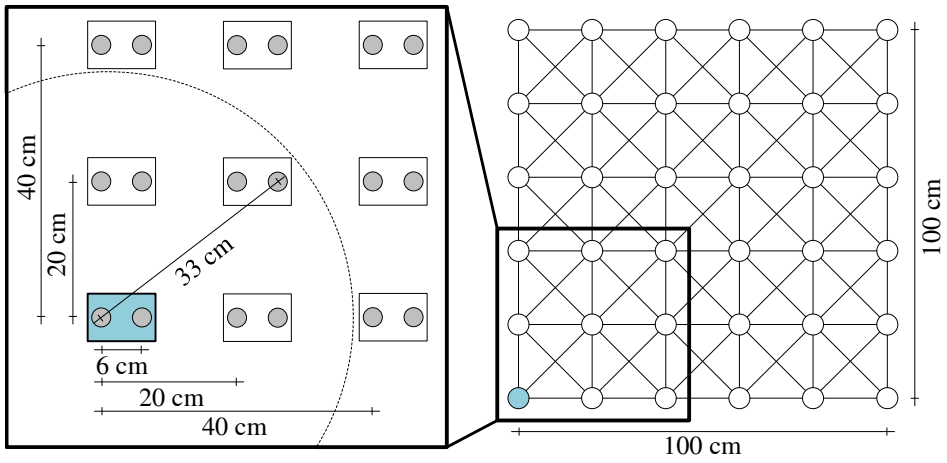


Abbildung 3.3: Geometrie und Abmessungen der Testumgebung [B 6]

Jedes Gerät ist mit zwei Rundstrahlantennen ausgestattet, die im Abstand von 6 cm zueinander auf der Oberseite des Gehäuses befestigt sind. Dies erfüllt den empfohlenen Minimalabstand einer halben Wellenlänge im 2,4-GHz- (ca. 12,5 cm) und 5-GHz-Band (ca. 6 cm) [167]. Der zusätzliche Antennengewinn beträgt 5 dBi [3.5]. Zur maßgeblichen Reduzierung der Kommunikationsreichweite ist ein HF-Dämpfungsglied mit einer festen Dämpfung von 30 dB pro Antennenweg verschraubt [163]. Eine Feinabstimmung der Reichweite erfolgt über die Einstellung der Sendeleistung. Kap. 3.3.4 erläutert den Skalierungsansatz im Detail.

Abb. 3.3 zeigt einen vergrößerten Ausschnitt des schematischen Testaufbaus. Hierbei sind die korrespondierenden Antennen benachbarter Knoten im horizontalen und vertikalen Abstand von 20 cm zueinander platziert. Aufgrund der Ausstattung mit zwei Antennen ergibt sich zudem ein Abstand von 33 cm in diagonaler Richtung zwischen den jeweils äußeren Antennen des Knotenpaars. Durch Beschränkung der Kommunikationsreichweite d auf 33 cm $<$d$<$ 40 cm ist ein direkter Verbindungsaufbau nur zwischen benachbarten Geräten möglich. Multi-Hop-Pfade zwischen weiter entfernten Knoten werden bei Bedarf automatisch durch das 802.11s-Routing-Protokoll HWMP ermittelt. Die so dimensionierte Testumgebung benötigt mit den bislang 36 Knoten eine Laborfläche von nur 1 m^2. Dadurch ist künftig eine flexible Erweiterung der Anordnung auf mehr als 100 Knoten möglich.

3.3.3 Performance-Analyse der Geräteplattform

Im Vorfeld der Inbetriebnahme der Testumgebung wurden zunächst mögliche Arbeitspunkte der Plattform identifiziert. Im Vordergrund stand die Parametrisierung des WLAN-Adapters derart, dass eine einheitliche, reduzierte Kommunikationsreichweite der Geräte im Laboraufbau erreicht wird. Ebenfalls wurden praktische Leistungsgrenzen der Geräte ermittelt. Nachfolgend wurden Arbeitspunkte deutlich unterhalb dieser Grenzen gewählt, um Leistungsreserven zu garantieren und verfälschte Ergebnisse bei der Untersuchung der in dieser Arbeit entwickelten Optimierungsansätze zu vermeiden.

3.3.3.1 Technische Eigenschaften der WLAN-Hardware

In der Praxis unterstützt der Großteil handelsüblicher WLAN-Hardware nicht den sogenannten *Greenfield*-Modus von 802.11n, bei dem ausschließlich dessen High Throughput (HT)-Datenraten zum Einsatz kommen [100]. Zwar erlaubt dieser Modus eine Erhöhung des Datendurchsatzes im Netzwerk, schränkt die Netzwerkteilnehmer jedoch strikt auf 802.11n-fähige Geräte ein und verschlechtert das Zusammenspiel mit Netzwerken, die 802.11n noch nicht unterstützten. Zwei benachbarte WLAN-Zellen auf demselben Kanal, eine im 802.11n-Greenfield-Betrieb und eine ohne 802.11n-Fähigkeit, berücksichtigen sich im Zuge des Medienzugriffs nicht gleichermaßen. Da die nicht-HT-fähigen Geräte die Frame-Präambel des HT-Netzwerks nicht dekodieren können, erkennen sie eine Kanalbelegung durch die Nachbarzelle nur auf Basis des Carrier-Sense-Teilmechanismus *Energy Detection (ED)* (siehe Kap. 2.4.5). Dessen im Vergleich zur Präambel-Erkennung geringere Sensitivität erhöht in diesem Szenario die Wahrscheinlichkeit für Kollisionen und Interferenz zwischen den WLAN-Zellen. Aus diesem Grund wird der Greenfield-Modus oft nicht unterstützt und von dessen Nutzung abgeraten [100].

Dies gilt auch für die WLAN-Module der Testumgebung. Stattdessen arbeiten diese in einem sogenannten *Mixed*-Modus. Alle Frames besitzen hier eine abwärtskompatible Präambel, welche die für den Rest des Frames geltenden Übertragungsparameter angibt [100]. Zudem werden lediglich Unicast-Daten-Frames mit 802.11n-Datenraten übertragen. Im Gegensatz dazu werden 802.11-Steuerinformationen (Management, Control und Action Frames) sowie jegliche Multicast-Frames

[3.5] dBi: relativer Antennengewinn bezogen auf einen idealen Isotropstrahler

Tabelle 3.3: Transceiver-Eigenschaften des WLAN-Moduls Compex WLE200NX [B 6]
(MCS: Modulation & Coding Scheme, FEC: Forward Error Correction, P_{tx}: Sendeleistung, P_{rx}: Empfangsleistung; max. P_{tx} (pro Antenne; 14 dBm-Limit auf Kanal 149) und min. P_{rx} (Empfangssensitivität) laut Datenblatt [162])

802.11n-Eigensch. (5 GHz, 20 MHz Bandbreite, Langes Guard-Intervall)							**802.11a-Eigenschaften**		
MCS-Index	**# Datenströme**	**Modulation**	**FEC-Rate**	**Datenrate [Mbit/s]**	**Max. P_{tx} [dBm]**	**Min. P_{rx} [dBm]**	**Datenrate [Mbit/s]**	**Max. P_{tx} [dBm]**	**Min. P_{rx} [dBm]**
0	1	BPSK	1/2	6,5	17 (14)	-93	6	17 (14)	-94
1	1	QPSK	1/2	13	17 (14)	-91	12	17 (14)	-93
2	1	QPSK	3/4	19,5	17 (14)	-87	18	17 (14)	-90
3	1	16-QAM	1/2	26	17 (14)	-85	24	17 (14)	-86
4	1	16-QAM	3/4	39	17 (14)	-82	36	15 (14)	-83
5	1	64-QAM	2/3	52	16 (14)	-77	48	13	-79
6	1	64-QAM	3/4	58,5	12	-75	54	12	-75
7	1	64-QAM	5/6	65	8	-72	54	12	-75
8	2	BPSK	1/2	13	17 (14)	-93	6	17 (14)	-94
9	2	QPSK	1/2	26	17 (14)	-91	12	17 (14)	-93
10	2	QPSK	3/4	39	17 (14)	-87	18	17 (14)	-90
11	2	16-QAM	1/2	52	17 (14)	-85	24	17 (14)	-86
12	2	16-QAM	3/4	78	17 (14)	-82	36	15 (14)	-83
13	2	64-QAM	2/3	104	16 (14)	-77	48	13	-79
14	2	64-QAM	3/4	117	12	-75	54	12	-75
15	2	64-QAM	5/6	130	8	-72	54	12	-75

(auch Daten-Frames) mit *Non-HT*-Datenraten versandt, die durch frühere Spezifikationsversionen der Bitübertragungsschicht eingeführt wurden. Für die genutzten WLAN-Module sind dies die Datenraten von 802.11g im 2,4-GHz- bzw. 802.11a im 5-GHz-Band [101]. In der Testumgebung sollen bevorzugt Kanäle im 5-GHz-Band genutzt werden, um Störungen im oft überfüllten 2,4-GHz-Band zu vermeiden. Daher sind die technischen Eigenschaften für 802.11n und 802.11a relevant.

Die Tab. 3.3 gibt eine Übersicht der Transceiver-Eigenschaften des WLAN-Moduls WLE200NX, die dem Datenblatt des Herstellers entnommen sind [162]. Der linke Tabellenbereich enthält die 802.11n-Informationen. Die ersten fünf Spalten zeigen die unterstützten 802.11n-Datenraten, die als sogenannte Modulation and Coding Schemes (MCSs) aufsteigend indiziert werden. Dabei bezeichnet ein MCS die Kombination aus Modulationsverfahren, Kodierungsrate der Forward Error Correction (FEC) sowie Anzahl der MIMO-Datenströme (vgl. Kap. 2.4.4). Die korrespondierenden Informationen für 802.11a sind im rechten Tabellenbereich zu finden. Dessen erste Spalte ordnet jedem 802.11n-MCS die hinsichtlich Modulationsverfahren und FEC-Rate äquivalente 802.11a-Datenrate zu. Lediglich für die höchste Kombination aus 64-QAM und FEC-Rate 5/6 existiert keine Entsprechung in 802.11a. In diesem Fall wird auf 802.11a-Seite wie beim vorherigen MCS ebenfalls die FEC-Rate 3/4 angenommen.

Die in dieser Arbeit genutzte Hardware unterstützt mit zwei Antennen die Übertragung von bis zu zwei MIMO-Datenströmen (engl. *Spatial Streams*). Der MIMO-Modus ergibt sich durch Konfiguration der MCS 0–7 (ein Datenstrom mittels Diversity MIMO) bzw. MCS 8–15 (zwei Datenströme mittels Spatial Multiplexing MIMO). Für alle mit 802.11a-Datenraten gesendeten Frames gilt der Modus Diversity MIMO.

Allen Datenraten in Tab. 3.3 sind weitere Informationen zur maximal konfigurierbaren Leistung der Sendeeinheit (max. P_{tx}) sowie zur Empfindlichkeit der Empfangseinheit (min. P_{rx}) zugeordnet. Letztere entspricht dem minimalen Leistungspegel eines empfangenen Signals, der für dessen erfolgreiche Erkennung und Demodulation benötigt wird [103]. Wie zuvor erläutert gilt bei Nutzung der oberen 5-GHz-Kanäle ab Kanal 149 in Europa eine Beschränkung von P_{tx} auf 14 dBm (siehe Tabellenwerte in Klammern), selbst wenn durch die Hardware prinzipiell höhere Werte unterstützt werden. Die 802.11n-Angaben gelten für eine Kanalbandbreite von 20 MHz und eine Wartezeit zwischen aufeinander folgenden OFDM-Symbolen von 800 ns (engl. *Long Guard Interval (GI)*). Dies entspricht grundsätzlich den Parametern der früheren Spezifikationen 802.11a/g. Einziger Unterschied ist die Verwendung des mit 802.11n eingeführten HT-Modus. Bei gleicher Kanalbandbreite werden im HT-Modus anteilig mehr OFDM-Subträger zur Datenübertragung verwendet, was zu einer leicht höheren Datenrate gegenüber 802.11a/g führt. Der sich in dieser Parametrisierung ergebende Spielraum für die Konfiguration der Sendeleistung sowie die Empfindlichkeit der Empfangseinheit entsprechen jedoch nahezu den Werten im 802.11a-Modus.

Einerseits ermöglicht 802.11n die Nutzung von 40 MHz breiten Kanälen und damit eine theoretische Verdopplung der Bruttodatenrate. Zudem lässt sich mit der Wahl eines *Short GI* von 400 ns die Datenrate um etwa 10 % steigern, was jedoch mit einer erhöhten Anfälligkeit für Inter-Symbol-Interferenz einhergeht [99]. Da beide Optionen zu stark unterschiedlichen Transceiver-Eigenschaften gegenüber dem 802.11a-Modus führen, wurde von ihrer Nutzung explizit abgesehen.

3.3.3.2 TCP- und UDP-Performance eines Mesh-Links

Um die Leistungsgrenzen der Geräte zu bestimmen und geeignete Arbeitspunkte abzuleiten, wurde der erreichbare mittlere Datendurchsatz eines Mesh-Links für jede MCS-Einstellung unter Optimalbedingungen gemessen. Dazu wurden zwei Knoten bei jeweils maximaler Sendeleistung (siehe Tab. 3.3) und ohne Dämpfungsglieder im 30 cm-Abstand voneinander platziert.

Die Durchsatzmessungen erfolgten mit dem Client/Server-Programm `iperf3` (v3.2) [168] sowohl für Transmission Control Protocol (TCP) als auch User Datagram Protocol (UDP) auf der Transportschicht. Es wurden bis auf folgende Ausnahmen die Standardeinstellungen des Linux-Kernels beibehalten. In der Praxis wird für WLAN meist auch eine Maximum Transmission Unit (MTU) von 1500 Bytes genutzt, da dies eine fragmentierungsfreie Weiterleitung in angebundene Ethernet-Netzwerke und das Internet erlaubt. Aus diesem Grund wurde in den TCP-Messungen eine Maximum Segment Size (MSS) von 1448 Bytes gewählt. Als TCP-Überlastkontrollverfahren kam *CUBIC* [169] zum Einsatz, das seit Linux-Kernel v2.6.19 den Standardalgorithmus darstellt. Es wurde jedoch dessen Teilmechanismus *HyStart* deaktiviert, der über WLAN-Verbindungen ein normales Wachstum des TCP-Sendefensters verhinderte. Dieses Problem war in Entwicklerkreisen für neuere Kernelversionen bereits bekannt und bestätigt [170].

Im Fall von UDP wurden Datagramme der festen Größe von 1448 Bytes versendet. Für den UDP-Socket der `iperf3`-Applikation wurde die statische Buffergröße von 160 kB auf 1 MB erhöht. Dadurch kann eine größere Anzahl an Nachrichten im Linux-Userspace vorgehalten werden. Dies reduziert die Prozessorlast und den zeitlichen Overhead durch Kontextwechsel in den Kernel Space beim Versand und Empfang von Nachrichten. Im Fall von TCP wurde der Socket Buffer automatisch vom Betriebssystem bis zu einer Größe von 2 MB angepasst.

Für jede Kombination aus Transportprotokoll und MCS wurden 5 Durchsatzmessungen durchgeführt. Jeder Durchlauf hatte eine Dauer von 60 s und es wurden Messwerte in 1 s-Intervallen direkt in der `iperf3`-Applikation aufgenommen und gemittelt. Pro Konfiguration wurden somit 300 Samples berücksichtigt. Parallel zum Datendurchsatz wurde mithilfe der `iperf3`-internen Reporting-Funktion auch die mittlere Prozessorauslastung gemessen. Tab. 3.4 zeigt den TCP-/UDP-Durchsatz, die Paketverlustrate (nur UDP) und die durch `iperf3` erzeugte Prozessorauslastung für

Tabelle 3.4: TCP/UDP-Datendurchsatz eines Mesh-Links und erzeugte CPU-Auslastung [B 6]
(*: WLAN-Vergleichstest mit zwei Laptops; ETH: Fast-Ethernet-Vergleichstest mit zwei Galileo Boards)

	TCP			UDP			
MCS Index	**Durchsatz [Mbit/s]**	**Sender CPU [%]**	**Empfänger CPU [%]**	**Durchsatz [Mbit/s]**	**Paket-verluste [%]**	**Sender CPU [%]**	**Empfänger CPU [%]**
0	4,7	2,5	9,7	5,8	0	7,8	9,7
1	9,2	4,9	22,5	11,5	0	15,1	18,4
2	13,4	7,6	25,5	17,2	0	22,5	28,0
3	17,8	11,2	46,7	22,9	0	29,6	38,2
4	26,7	20,5	57,1	26,5	23	37,5	89,5
5	16,7	98,2	43,1	23,5	0	98,6	57,4
6	16,6	98,1	43,6	23,4	0	98,5	56,9
7	16,6	98,2	43,6	23,3	0	98,4	56,2
8	9,1	4,9	20,0	11,5	0	12,9	25,6
9	17,4	10,8	42,1	22,9	0	26,4	56,5
10	26,8	20,8	54,9	26,9	22	37,7	88,9
11	16,9	98,1	45,5	23,5	0	98,4	57,6
12	16,9	98,0	44,2	23,3	0	98,3	55,5
13	17,3	98,1	44,6	23,1	0	98,2	54,5
14	17,0	98,4	45,3	23,1	0	98,3	54,3
15	16,9	98,5	44,8	22,5	0	98,5	57,4
15 *	71,5	1,3	7,5	96,8	0	11,5	13,0
ETH	94,1	11,2	49,0	96,1	0	30,9	32,8

jede MCS-Konfiguration auf Sender- und Empfängerseite. Bei Wahl der grün markierten MCS wurde ein für die jeweilige WLAN-Bruttodatenrate der Bitübertragungsschicht typischer TCP- und UDP-Nettodurchsatz bei einer Prozessorauslastung von weniger als 60 % erreicht. Dies betrifft die Einstellungen MCS 0–3 (ein Datenstrom, Diversity MIMO) und MCS 8–9 (zwei Datenströme, Spatial Multiplexing MIMO).

Bei Wahl höherer MCS, hervorgehoben in rot, stießen Empfänger oder Sender an ihre Leistungsgrenzen. Mit MCS 4 und 10 war der Sender noch in der Lage, Übertragungsraten nahe der Bruttodatenrate zu erreichen. Im UDP-Fall konnten Senderaten von über 34 Mbit/s gemessen werden. Hier offenbarten eine hohe Prozessorauslastung von fast 90 % und eine Paketverlustrate von 23 % jedoch die empfängerseitige Leistungsgrenze. Diese führte zu einem deutlich geringeren effektiven Datendurchsatz von nur 26,5 Mbit/s. Im TCP-Fall stellte sich der Sender durch Fluss- und Überlastkontrolle auf die Verarbeitungsgeschwindigkeit des Empfängers ein. Bei MCS 5–7 und 11–15 wurde schließlich die Leistungsgrenze des Senders erreicht. Hier mussten eine vollständige Prozessorauslastung sowie stagnierende Durchsatzwerte weit unterhalb der jeweiligen Bruttodatenrate festgestellt werden. Dass diese sogar unterhalb der mit MCS 4 bzw. 10 erzielten Ergebnisse lagen, weist auf Beschränkungen der Galileo-Plattform im Zusammenspiel mit den höheren MCS-Einstellungen hin.

Um ein generelles Fehlverhalten der WLAN-Hardware oder des Software-Protokollstapels als Ursachen auszuschließen, wurden Messungen mit MCS 15 bei gleicher Konfiguration der WLAN- und TCP-/UDP-Parameter auf einer stärkeren Geräteplattform wiederholt. Hierzu wurden zwei Laptops vom Typ Dell Latitude E4300 (Intel Core 2 Duo 2,5 GHz, 4 GB RAM, Linux-Kernel v4.10) mit den WLAN-Adaptern der Galileo Boards genutzt. Es konnte ein UDP-Nettodurchsatz von mehr als 90 Mbit/s bei einer Prozessorauslastung unterhalb von 15 % erreicht werden. In einem weiteren Experiment wurden TCP- und UDP-Messungen mit den Galileo Boards über deren integrierte Fast-Ethernet-Schnittstelle durchgeführt. Gegenüber der WLAN-Konfiguration wurden demnach der Netzwerkadapter, dessen I/O-Interface sowie der Code-Pfad für Treiber und Sicherungsschicht ausgetauscht. Im Ergebnis wurde ein TCP- und UDP-Nettodurchsatz von mehr als 90 Mbit/s bei moderater Prozessorauslastung erreicht. Dies unterstützt die Vermutung, dass Probleme in der Kombination von Linux-WLAN-Stack, `ath9k`-Treiber und WLAN-Hardware bestehen, die spezifisch für die Galileo-Plattform sind.

Zusammengefasst stellen MCS 0–3 bzw. 8–9 (und die jeweils zugehörigen 802.11a-Datenraten) den Bereich der geeigneten Datenraten für das Testbed dar. Für diese Einstellungen bieten die Geräte noch genügend Leistungsreserven, um Applikationen auszuführen und Optimierungsansätze zu erproben. Nachfolgend werden daher nur diese Arbeitspunkte betrachtet.

3.3.4 Konzept zur Reichweitenskalierung

3.3.4.1 Methodik

Die Grundidee der Skalierung der Kommunikationsreichweite im Testbed *Mini-Mesh* basiert auf dem Ansatz *ScaleMesh* von ElRakabawy et al. [153]. Dessen Autoren beschreiben einen miniaturisierten Mesh-Aufbau mit 20 Knoten und Unterstützung für 802.11a/b/g. Die WLAN-Adapter der Geräte in ScaleMesh waren mit ihrer maximal möglichen Datenrate und Sendeleistung konfiguriert. Die Reichweitenbeschränkung erfolgte ausschließlich über variabel einstellbare Dämpfungsglieder im Antennenweg. Wie in [153] werden auch in Mini-Mesh die Antennenwege gedämpft. Anstatt kostenintensiver variabler Dämpfungsglieder wurden jedoch preiswerte Komponenten mit festem Dämpfungswert gewählt. Experimentell wurde ein Dämpfungswert von 30 dB pro Antenne ermittelt. Bei maximaler Sendeleistung des WLAN-Adapters reduziert dieser die effektive Sende- bzw. Empfangsleistung bereits um mehrere Größenordnungen auf eine Reichweite im Labormaßstab von wenigen Metern. Die anschließende Feineinstellung der Reichweite erfolgt durch geeignete Reduktion der WLAN-Sendeleistung.

Wie eingangs in Tab. 3.3 dargestellt, gilt für jede der im Testbed nutzbaren Datenraten (MCS 0–3 bzw. 8–9) eine bestimmte Empfangsempfindlichkeit. Daher wird für jede Einstellung der Datenrate eine unterschiedliche Sendeleistung benötigt, um die gleiche anvisierte Reichweite im Laboraufbau zu erhalten. Bei der in Kap. 3.3.2 motivierten Gitteranordnung sind die Knoten im horizontalen und vertikalen Abstand von 20 cm platziert (siehe Abb. 3.3). Um eine Direktverbindung nur zu den Gitternachbarn in horizontaler, vertikaler und diagonaler Richtung zu ermöglichen, muss der Kommunikationsradius zwischen 33 und 40 cm liegen. Dieser Distanzbereich gilt als Zielvorgabe bei der Kalibrierung der WLAN-Adapter, beschrieben in Kap. 3.3.5.

3.3.4.2 Mathematisches Modell

Um von beliebigen Laboranordnungen auf die zugehörigen Dimensionen eines Aufbaus ohne Miniaturisierung zu schließen, bedarf es einer mathematischen Beschreibung der Skalierungsmethode. Dazu wurden Modellbetrachtungen von ElRakabawy et al. [153] einbezogen und erweitert. Diese kombinieren den mathematischen Ausdruck der Leistungsdifferenz zwischen Sender- und Empfängerknoten mit einem frequenz- und distanzabhängigen Modell für den Pfadverlust der Funkstrecke.

ElRakabawy et al. [153] untersuchten eine Reichweitenskalierung lediglich innerhalb ihres Labors mittels variabler Dämpfungsglieder. Als Umgebungsparameter des Pfadverlustmodells diente daher der theoretische Wert für ein Indoor-Szenario. Die WLAN-Hardware arbeitete mit einer festen Sendeleistung von 18 dBm und einer Datenrate von 54 Mbit/s. Mit dem Modell konnten aus diesen bekannten Parametern die erforderlichen Werte der Dämpfungsglieder für gewünschte Reichweiten im Laboraufbau berechnet werden. Im Gegensatz zu [153] werden bei Mini-Mesh Dämpfungsglieder mit festem Dämpfungswert genutzt. Statt einer variablen Dämpfung wird die Sendeleistung experimentell variiert, sodass eine gewünschte Reichweite im Labor eingehalten wird. Mithilfe des Modells soll nun für eine bekannte Laborreichweite die korrespondierende Reichweite abgeschätzt werden, die ohne Dämpfungsglieder unter Beibehaltung der Datenrate und Sendeleistung entsteht.

Um das Pfadverlustmodell aus [153] nutzen zu können, besteht in allen Aufbauten eine Sichtverbindung (Line-of-Sight (LoS)) zwischen benachbarten Knoten. Als dominierende Signalkomponente erlaubt diese eine Vernachlässigung unterschiedlicher Ausprägungen der Mehrwegeausbreitung im Aufbau mit und ohne Miniaturisierung. Kap. 3.3.5 beschreibt die praktische Validierung dieser Annahme. Für die im Labor ermittelten Konfigurationen wurden korrespondierende Reichweiten ohne Dämpfungsglieder im Freien gemessen und der resultierende Umgebungsparameter des Modells bestimmt. Das Ergebnis liegt in der Größenordnung des in [153] gewählten Referenzwerts.

Nachfolgend wird das mathematische Modell der Skalierungsmethode von Mini-Mesh erläutert. Beschrieben wird die Funkstrecke zwischen einem Sender und einem Empfänger, wie sie in Abb. 3.4 dargestellt ist. Die Kommunikationsreichweite im Laboraufbau erhält das Attribut *skaliert* (S). Die korrespondierende Reichweite ohne Dämpfungsglieder wird als *unskaliert* (U) bezeichnet.

Die Equivalent Isotropic Radiated Power (EIRP) des Senders wird ausgedrückt als

$$EIRP = P_{tx} + G_{ant}. \tag{2}$$

Sie bezeichnet die vom Transceiver abgegebene Sendeleistung P_{tx} verstärkt durch den Antennengewinn G_{ant}. Bei Zwei-Antennen-Systemen mit gleicher Polarisation bezeichnet P_{tx} die Leistungssumme beider Antennenwege [167]. Für eine gewünschte Reichweite im Labor wird P_{tx} experimentell

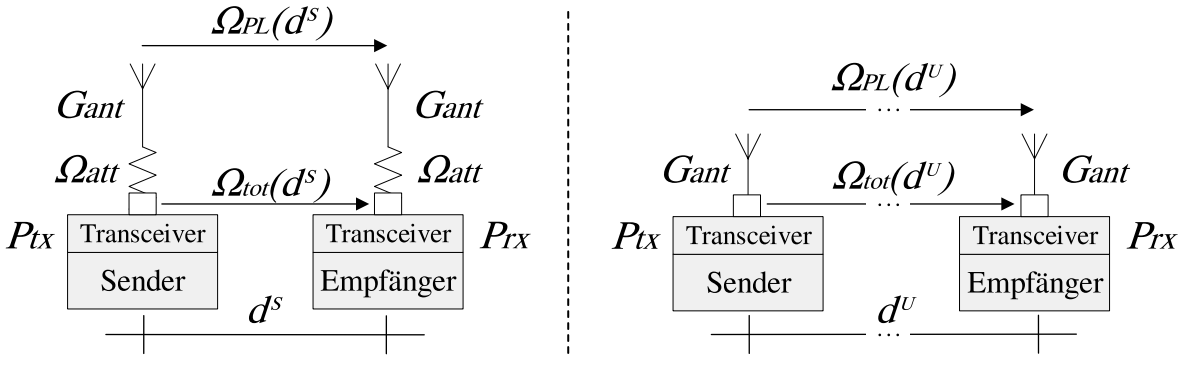


Abbildung 3.4: Leistungs- und Dämpfungskomponenten der Funkstrecke
(links: skaliert (S) mit Dämpfungsgliedern, rechts: unskaliert (U) ohne Dämpfungsglieder)

ermittelt. Die Gesamtdämpfung Ω_{tot} zwischen Sender und Empfänger ist in Gl. 3 als Differenz der Sende- und Empfangsleistung gegeben.

$$\Omega_{tot} = P_{tx} - P_{rx} \tag{3}$$

Die Empfangsleistung P_{rx} bezeichnet den am Empfänger wahrgenommenen Leistungspegel nach Einwirkung von Antennengewinn, möglicher Dämpfungsglieder und Freiraumdämpfung. Für die verschiedenen WLAN-Datenraten besteht eine untere Schranke für P_{rx} in Form der Empfangsempfindlichkeit der Hardware (siehe Tab. 3.3). Da sich bei Erhöhung der Modulationsordnung die Energie pro Bit verringert, wird auch eine höhere Signal-to-Noise Ratio (SNR) für die erfolgreiche Dekodierung des Signals benötigt [99]. Soll dieselbe Kommunikationsreichweite für verschiedene Einstellungen der Datenrate erreicht werden, muss daher eine Anpassung der Sendeleistung P_{tx} erfolgen.

Gl. 4 zeigt Ω_{tot} als Funktion des distanzabhängigen Pfadverlusts (engl. *Path Loss*) $\Omega_{PL}(d)$, des Antennengewinns G_{ant} sowie der Antennendämpfung Ω_{att} im Fall der skalierten Laboranordnung mit Dämpfungsgliedern. Hierbei ist Ω_{att} der feste Dämpfungswert von je 30 dB auf Sender- und Empfängerseite. Die durch Verbindungskabel zwischen Netzwerkadapter und Antennen hervorgerufene Dämpfung ist im Verhältnis zu den anderen Größen vernachlässigbar und wird nicht betrachtet.

$$\begin{aligned} \Omega^{U}_{tot} &= \Omega_{PL}(d^U) - 2 \cdot G_{ant} \\ \Omega^{S}_{tot} &= \Omega_{PL}(d^S) - 2 \cdot G_{ant} + 2 \cdot \Omega_{att} \end{aligned} \tag{4}$$

Durch Einsetzen von Gl. 3 in Gl. 4 erhält man die in Gl. 5 gegebenen Ausdrücke für den Pfadverlust Ω_{PL} im skalierten und unskalierten Fall.

$$\begin{aligned} \Omega_{PL}(d^U) &= P_{tx} - P_{rx} + 2 \cdot G_{ant} \\ \Omega_{PL}(d^S) &= P_{tx} - P_{rx} + 2 \cdot G_{ant} - 2 \cdot \Omega_{att} \end{aligned} \tag{5}$$

Skalierter und unskalierter Aufbau unterscheiden sich allein im Vorhandensein der Dämpfungsglieder (Ω_{att}) und in den resultierenden Reichweiten (d^U, d^S). Daher besitzen P_{tx} und G_{ant} in beiden Fällen jeweils gleiche Werte. Ebenso ist P_{rx} jeweils im Punkt der maximalen stabilen Kommunikationsreichweite durch die Empfangsempfindlichkeit der WLAN-Hardware gegeben und damit identisch.

Der Einsatz von Diversity MIMO findet implizit in P_{rx} Berücksichtigung. Insbesondere in Szenarien ohne direkte Sichtverbindung (Non-Line-of-Sight (NLoS)) und mit vielen unabhängigen Ausbreitungswegen durch Hindernisse und reflektierende Objekte erlaubt die Ausnutzung der räumlichen Freiheitsgrade mittels mehrerer Antennen eine deutliche Verbesserung der SNR [99]. Die Charakteristik der Mehrwegeausbreitung ist jedoch stark umgebungsabhängig. Um eine Vergleichbarkeit zwischen skaliertem und unskaliertem Aufbau zu ermöglichen, wird in dieser Arbeit ein Kanal mit dominierender LoS-Komponente angenommen.

Gl. 6 zeigt das Ausbreitungsmodell der International Telecommunication Union (ITU) [153]. Es beschreibt den Pfadverlust Ω_{PL} zwischen Sender und Empfänger als Funktion der Distanz d, der Mittenfrequenz f_c des genutzten Kanals sowie eines Umgebungsparameters p. Die Wahl von p ist abhängig von der angenommenen Umgebung und liegt im Bereich von 2 (Freiraum-Szenario) bis 5 (stark gestörtes Indoor-Szenario). ElRakabawy et al. treffen eine Wahl von $p = 3$ [153]. Dies entspricht laut ITU dem Szenario einer Büroumgebung mittlerer Größe [171].

$$\Omega_{PL}(d) = 20 \cdot log_{10}(f_c) + 10 \cdot p \cdot log_{10}(d) \tag{6}$$

Wird Ω_{PL} in Gl. 5 nun mit dem ITU-Modell aus Gl. 6 ersetzt, können Ausdrücke für die Distanzen d^U und d^S abgeleitet werden:

$$\begin{aligned} log_{10}(d^U) &= \frac{P_{tx} - P_{rx} + 2 \cdot G_{ant} - 20 \cdot log_{10}(f_c)}{10 \cdot p} \\ log_{10}(d^S) &= \frac{P_{tx} - P_{rx} + 2 \cdot G_{ant} - 2 \cdot \Omega_{att} - 20 \cdot log_{10}(f_c)}{10 \cdot p} \end{aligned} \tag{7}$$

Durch Umformung erhält man das Skalierungsverhältnis r, gegeben in Gl. 8, das den Miniaturisierungsmaßstab der Testumgebung darstellt. Sind der Umgebungsparameter p und eine der Reichweiten d^U bzw. d^S bekannt, kann die jeweils andere Reichweite berechnet werden.

$$r = \frac{d^U}{d^S} = 10^{\frac{2 \cdot \Omega_{att}}{10 \cdot p}} \tag{8}$$

Umgekehrt kann mit bekannten Reichweiten d^U und d^S der Parameter p nach Gl. 9 bestimmt und somit das Modell überprüft werden:

$$p = \frac{2 \cdot \Omega_{att}}{10 \cdot log_{10}(r)} \tag{9}$$

Unter der Voraussetzung einer Funkstrecke mit dominierender LoS-Komponente können das vorgestellte Modell und der Parameter p zur Umrechnung zwischen skaliertem und unskaliertem Aufbau genutzt werden. Zur Validierung dieser Annahme wurde, wie in Kap. 3.3.5 erläutert, p experimentell bestimmt und mit dem in [153] gewählten Wert verglichen.

3.3.5 Umsetzung des Skalierungskonzepts

Im Folgenden werden die Realisierung der Reichweitenskalierung und die Parametrisierung des mathematischen Modells beschrieben. Für alle Messungen wurde ein Knotenpaar der Testumgebung mit den in Tab. 3.2 gegebenen Eigenschaften verwendet. Mit Ausnahme der in Kap. 3.3.3 diskutierten Anpassungen galten die Standardeinstellungen des Linux-Kernels und WLAN-Treibers. Aufgrund der Performance-Limitierungen der Geräte wurden nur die 802.11n-Datenraten MCS 0–3 und 8–9 sowie ihre zugehörigen 802.11a-Datenraten untersucht.

3.3.5.1 Technische Vorbereitungen

Um reproduzierbare Ergebnisse für die Kommunikationsreichweite im Labor zu erhalten, musste eine Reihe praktischer Voraussetzungen identifiziert und sichergestellt werden. Zunächst war es notwendig, die Geräte der Testumgebung gegen elektromagnetische Einflüsse zu schirmen. Die Leiterplatten der Galileo-Boards und der WLAN-Karten weisen selbst eine Antennencharakteristik auf. Da die regulären Antennenwege um jeweils 30 dB gedämpft sind, besitzt diese zudem einen höheren Einfluss. Ohne Schirmung wurden bei Laborexperimenten im Bereich unterhalb eines Meters deutliche Reichweitenschwankungen trotz statischer Konfiguration der Datenrate und Sendeleistung beobachtet. Als Gegenmaßnahme wurden die Kunststoffgehäuse der Geräte mit selbstklebender Kupferfolie umschlossen und lediglich Öffnungen für Antennenanschlüsse, Stromversorgung und Ethernet ausgespart. Die Wirksamkeit der Schirmung wurde mit einem Knotenpaar überprüft, bei dem die WLAN-Module nicht mit den Kabeln der externen Antennen verbunden waren. Eine Kommunikation beider Knoten musste somit durch die Gehäusewände hinweg erfolgen. Es wurde eine minimale Datenrate (MCS 0 / 6 Mbit/s) und maximale Sendeleistung (17 dBm) konfiguriert. Dies entspricht den

WLAN-Adapter-Einstellungen für eine maximale Reichweite. Selbst bei einer Platzierung in unmittelbarer Nähe zueinander empfingen beide Knoten jedoch aufgrund der Schirmung keine Beacon-Frames und bauten somit keine Mesh-Verbindung auf.

Neben der Antennencharakteristik der Platinen wurden weitere Ursachen identifiziert, die eine Reproduzierbarkeit der Reichweite erschwerten. Dabei handelte es sich um Mechanismen, die spezifisch für den Atheros-Chipsatz der WLAN-Adapter sind. Zunächst musste die Funktion *Ambient Noise Immunity (ANI)* [172] deaktiviert werden. Andernfalls passte diese zur Laufzeit automatisch Signalerkennungs- und Verstärkungsparameter der Empfängerstufe an. Des Weiteren führt die Atheros-Hardware bei Standardeinstellungen eine periodische Rekalibrierung des *Noise Floors* durch. Dieser bezeichnet den Leistungspegel des Hintergrundrauschens und dient als Referenzwert, auf den sich die relative empfangene Signalstärke und verschiedene Schwellenwerte beim Kanalzugriff beziehen. Dazu zählt beispielsweise die sogenannte *Clear Channel Assessment (CCA) Threshold*, die der Erkennung eines belegten Mediums dient [16]. Durch die Option *NF override* des `ath9k`-Treibers [157] konnte die automatische Rekalibrierung deaktiviert und stattdessen ein statischer Noise Floor beibehalten werden.

3.3.5.2 Kalibrierung im Labor

Die Parameter der WLAN-Hardware wurden für die gewünschte Reichweite im Labor experimentell kalibriert. Beide Mesh-Knoten wurden entsprechend der vertikalen Richtung der in Abb. 3.3 gezeigten Gitteranordnung zueinander positioniert. Die 30 dB-Dämpfungsglieder pro Antenne begrenzten den maximal möglichen Geräteabstand bei geringster Datenrate und maximaler Sendeleistung bereits auf weniger als 10 m.

Für jedes 802.11a/n-Datenratenpaar wurde die Sendeleistung P_{tx} ermittelt, die zur Einhaltung der Zielreichweite d nötig war. Es wurde unterschieden zwischen den Reichweiten d_{11a} und d_{11n}. Erstere gilt für die Übertragung von Management- und Multicast-Frames. Dazu zählen z. B. die periodischen Beacons der Mesh-Knoten, Acknowledgement (ACK)-Frames, aber auch die Kontrollnachrichten zum Verbindungsaufbau (Link Handshake) und Routing. Aufgrund des Betriebs der WLAN-Hardware im *Mixed*-Modus werden diese Frames mit der geringeren 802.11a-Datenrate versandt. Die Reichweite d_{11n} gilt hingegen für die Übertragung von Unicast-Daten-Frames über den Link. Diese werden mit der höheren 802.11n-Datenrate (MCS) im HT-Modus versandt.

Bestimmt wurde die Reichweite d_{11a} als diejenige Distanz, bei der die empfangene Leistung P_{rx} gerade noch oberhalb der Sensitivitätsschwelle des WLAN-Adapters für die jeweilige 802.11a-Datenrate lag. In den Messungen wurde der Mesh-Link hier noch kontinuierlich durch die jede Sekunde gesendeten Beacons der Gegenstelle aufrecht erhalten. Analog stellte sich die Reichweite d_{11n} kurz vor dem Abstand ein, in dem P_{rx} die Sensitivitätsschwelle für die jeweilige 802.11n-Datenrate erreichte. Zur Überprüfung wurden ICMP-Ping-Pakete (Daten-Frames) der Größe 1400 Bytes in einem Intervall von 1 s verschickt. Die Kalibrierung der Sendeleistung wurde so vorgenommen, dass in der gewünschten Reichweite durchgängig eine Round Trip Time (RTT) unter 10 ms für eine Dauer von mindestens einer Minute beobachtet werden konnte. Die Distanzbestimmung erfolgte mit einer Granularität von 1 cm.

Wie zuvor in Tab. 3.3 gezeigt, besitzen 802.11n-HT-Datenraten gegenüber ihren korrespondierenden 802.11a-Datenraten höhere Anforderungen an die SNR. Ein Mesh-Link, der durch Management-Frames in einer bestimmten Reichweite noch stabil hergestellt wird, garantiert in dieser daher nicht gleichzeitig auch eine stabile Datenübertragung im HT-Modus. Generell gilt der Zusammenhang $d_{11n} \leq d_{11a}$. Die Sendeleistung P_{tx} (Leistungssumme beider Antennen) wurde daher im Konfigurationsspielraum 3–17 dBm so variiert, dass sich eine stabile Daten-Frame-Reichweite im Bereich $33\,\text{cm} < d_{11n} < 40\,\text{cm}$ ergab. Dies entspricht der anvisierten Nachbarknotendistanz in der Gitteranordnung des Laboraufbaus in Abb. 3.3.

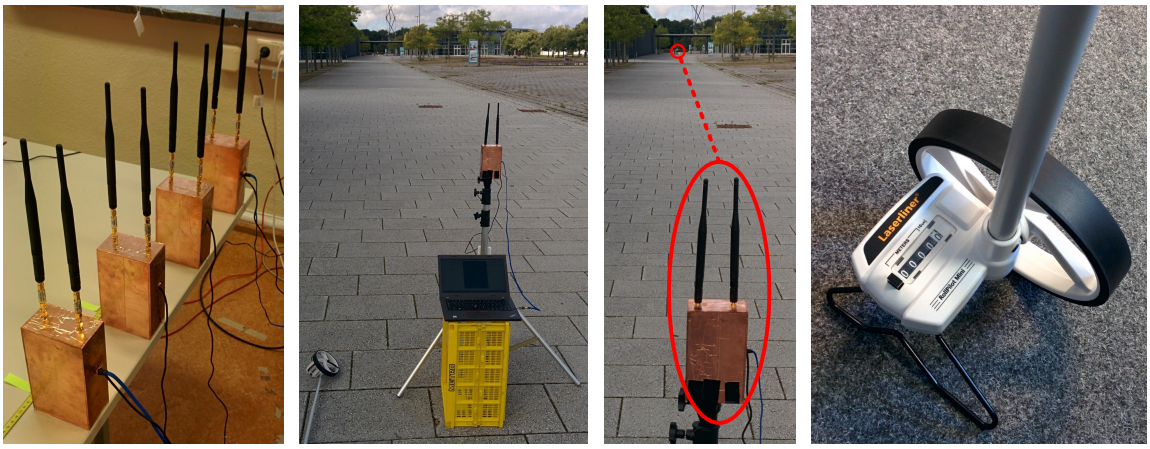


Abbildung 3.5: Indoor- und Outdoor-Reichweitenmessungen [B 6] (links: 3-Hop-Pfad Labor; Bilder Mitte: 1-Hop-Teilstrecke Messegelände; rechts: Messrad zur Outdoor-Distanzbestimmung)

3.3.5.3 Vergleichende Reichweitenmessungen

Im Anschluss an die Laborexperimente wurden diese im Freien wiederholt. Hierzu diente das Gelände der Hanse-Messe beim IGA-Park Rostock. Abb. 3.5 zeigt die für die Reichweitenmessungen sowie die praktische Validierung in Kap. 3.4 genutzten Aufbauten im Labor und auf dem Messegelände. Die Freifläche neben der Messehalle bot eine uneingeschränkte Sichtverbindung (LoS) über 450 m. Beide Geräte wurden mit Stativen auf 1 m Höhe befestigt und wie im Labor gegenüberliegend positioniert. Die zuvor ermittelten Wertepaare für Datenrate und Sendeleistung wurden beibehalten, die Dämpfungsglieder an den Antennen jedoch entfernt. Die sich so ergebende, korrespondierende Outdoor-Reichweite wurde mit einer Granularität von 1 m bestimmt.

Tab. 3.5 zeigt die Gegenüberstellung der Indoor- und Outdoor-Ergebnisse. Jeder der sechs Konfigurationen der 802.11a/n-Datenrate ist die experimentell bestimmte P_{tx} zugeordnet, die im Labor zur gewünschten Reichweite d_{11n} führte. Wie für den kombinierten 802.11a/n-Betrieb erwartet, lag d_{11n} in allen Messungen (Indoor und Outdoor) unterhalb von d_{11a}. Bei zwei der Datenraten-Konfigurationen wurden jedoch Abweichungen festgestellt. Wie nachfolgend erläutert, liegen diese im sich unterscheidenden MIMO-Verfahren begründet.

Für Management- und Multicast-Frames, stets mit 802.11a-Datenraten gesendet, ist der Modus *Diversity MIMO* treiberseitig vorgegeben. Bei diesem werden die Informationen redundant über beide Antennenwege übertragen. In einem Szenario mit dominierender Sichtverbindung und nahezu gleicher SNR auf allen Signalwegen ergibt sich nur ein geringer Reichweitengewinn gegenüber einem Aufbau mit einer Antenne (Single Input Single Output (SISO)) [99].

Der MIMO-Modus für die mit 802.11n-Datenraten gesendeten Daten-Frames ergibt sich wiederum aus dem MCS-Index. Die ersten vier Konfigurationen, MCS 0–3, definieren ebenfalls die Nutzung von Diversity MIMO. Hier lagen d_{11a} und d_{11n} daher stets dicht beieinander. Alle auf Diversity MIMO basierenden Ergebnisse sind in Tab. 3.5 grün hervorgehoben. Für MCS 0–3 und die zugehörigen 802.11a-Datenraten wurden im Labor Reichweiten d_{11a} bzw. d_{11n} von 36–40 cm erreicht. Die korrespondierenden Reichweiten ohne Dämpfungsglieder lagen im Bereich von 200–225 m. Nach Kap. 3.3.4 Gl. 8 ergibt sich ein mittleres Skalierungsverhältnis $r \approx 562$. Aus Gl. 9 resultiert der Umgebungsparameter $p \approx 2,18$. Im Vergleich mit ITU-Referenzwerten entspricht dies nahezu einem Freiraum-Szenario, das durch einen Wert von p=2 repräsentiert wird [153].

Auch in den letzten beiden Konfigurationen (Zeilen 5–6 in Tab. 3.5) werden die Management-Frames mit 802.11a-Datenraten und Diversity MIMO gesendet. Hinsichtlich der Link-Reichweite d_{11a} gelten

Tabelle 3.5: Ergebnisse der Indoor (ID)- und Outdoor (OD)-Reichweitenmessungen [B 6]
(Ω_{att}: Antennendämpfung, r: OD/ID-Skalierungsverhältnis, p: Umgebungsparameter, P_{tx}: Sendeleistung, d_{11a}: 11a-Reichweite (Non-HT), d_{11n}: 11n-Reichweite (HT), DM: Diversity MIMO, SM: Spatial Multiplexing)

			ID ($2 \cdot \Omega_{att}$: 60 dB)		OD ($2 \cdot \Omega_{att}$: 0 dB)		r		p	
11n-MCS	11a-Datenrate [Mbit/s]	P_{tx} [dBm]	d_{11a} [m]	d_{11n} [m]	d_{11a} [m]	d_{11n} [m]	11a	11n	11a	11n
0	6	7	0,40	0,38	224	210	560	553	2,18	2,19
1	12	9	0,40	0,38	223	202	558	532	2,18	2,20
2	18	11	0,38	0,36	216	212	568	589	2,18	2,17
3	24	14	0,39	0,36	213	210	546	583	2,19	2,17
8	6	13	0,72	0,38	412	23	572	61	2,18	3,37
9	12	15	0,74	0,38	410	23	554	61	2,19	3,37
							$\varnothing r_{DM}$:	562	$\varnothing p_{DM}$:	2,18
							$\varnothing r_{SM}$:	61	$\varnothing p_{SM}$:	3,37

daher die zuvor ermittelten Werte für r und p. Allerdings definieren die 802.11n-Datenraten MCS 8 und 9 den Modus *Spatial Multiplexing MIMO* für die Daten-Frames. Hier werden zwei unabhängige Datenströme übertragen, wodurch eine hohe SNR für beide Signale benötigt wird. Gleichzeitig müssen die Signalwege möglichst unkorreliert sein. Spatial Multiplexing MIMO benötigt Umgebungen mit starker Mehrwegeausbreitung, in denen die einzelnen Datenströme auf Empfangsseite klar unterschieden werden können [99]. Aufgrund der Sichtverbindung zwischen den Geräten waren diese Voraussetzungen jedoch nur bedingt gegeben.

Die Ergebnisse für Spatial Multiplexing MIMO sind in Tab. 3.5 blau hervorgehoben. Es sind starke Abweichungen zwischen d_{11a} und d_{11n} zu erkennen. Dies bestätigt frühere Simulationsstudien eines LoS-Szenarios, die für Spatial Multiplexing MIMO eine deutlich geringere Reichweite als für SISO und Diversity MIMO unter gleichen Kanalbedingungen ergaben [B 11]. Im Labor musste die Sendeleistung P_{tx} erhöht werden, um die gewünschte Daten-Frame-Reichweite d_{11n} auch mit Spatial Multiplexing MIMO zu erreichen. Die mit Diversity MIMO übertragenen Management-Frames führten bei dieser Einstellung jedoch gleichzeitig zu einer deutlich höheren Link-Reichweite. Gegenüber den Ergebnissen der vorherigen Konfigurationen wurde d_{11a} mit 72–74 cm (Indoor) bzw. 410–412 m (Outdoor) nahezu verdoppelt.

Der Vergleich der Indoor- und Outdoor-Ergebnisse für d_{11n} offenbarte zudem weitere Effekte. Trotz der Sichtverbindung in beiden Fällen zeigte Spatial Multiplexing MIMO eine hohe Empfindlichkeit gegenüber Umgebungsunterschieden. Innerhalb des Labors konnte die anvisierte d_{11n} durch Erhöhung von P_{tx} noch erreicht werden. Im Freien war die für diesen MIMO-Modus ungünstige LoS-Dominanz durch Abwesenheit reflektierender Wände und Oberflächen jedoch vermutlich verstärkt. Zudem bestanden Unterschiede im Abstand der Geräte zum Boden sowie im Antennenabstand pro Gerät im Verhältnis zur Distanz zwischen den Geräten. Diese lassen sich praktisch nicht vermeiden, verändern jedoch die Charakteristik der Mehrwegeausbreitung. Die in den Outdoor-Messungen erzielte d_{11n} war mit 23 m somit deutlich geringer als in den Konfigurationen mit Diversity MIMO. Zwar konnte wie auch im Labor ein reproduzierbarer Wert für d_{11n} gemessen werden. Das resultierende Skalierungsverhältnis $r \approx 61$ zwischen unskaliertem und skaliertem Aufbau liegt jedoch eine Größenordnung unter dem mit Diversity MIMO erzielten Wert von $r \approx 562$.

Im Rahmen der Arbeit wurde auf eine umfangreiche Erweiterung des mathematischen Skalierungsmodells für Spatial Multiplexing MIMO verzichtet. Die unterschiedlichen Umgebungsbedingungen lassen sich stattdessen im veränderten Parameter $p \approx 3,37$ zusammenfassen (siehe Tab. 3.5), der ebenfalls im vorgesehenen Wertebereich des ITU-Modells liegt [153]. Allerdings bezieht sich diese Parametrisierung speziell auf die Beschaffenheit der aktuellen Laborumgebung. Im weiteren Verlauf der Arbeit wird sich daher auf die Datenraten-Konfigurationen beschränkt, die Diversity MIMO für alle Frame-Typen nutzen (Zeilen 1–4 in Tab. 3.5). Mithilfe des für Diversity MIMO parametrisierten Skalierungsmodells können für beliebige LoS-Szenarien im miniaturisierten Laboraufbau die korrespondierenden Abmessungen für einen Aufbau ohne Dämpfungsglieder bestimmt werden.

3.4 Validierung des Skalierungsansatzes

Zum Nachweis, dass die miniaturisierte Testumgebung aussagekräftige Ergebnisse auf verschiedenen Ebenen des Netzwerkprotokollstapels liefert und sich damit zur Analyse praktischer Optimierungsansätze eignet, wurden weitere Messungen im Labor (mit Dämpfungsgliedern) sowie im Freien (ohne Dämpfungsglieder) durchgeführt. Zum einen wurde der Ende-zu-Ende-Datendurchsatz auf Transportschicht untersucht. Zum anderen wurde die sich bei verschiedenen Pfadlängen einstellende 802.11s-Routing-Metrik Airtime Link Metric (ALM) auf der WLAN-Sicherungsschicht verglichen.

3.4.1 Vergleich des TCP-/UDP-Durchsatzes

Unter Nutzung von vier Geräten der Testumgebung wurde zunächst der UDP- und TCP-Durchsatz einer Mesh-Verbindung von 1–3 Hops ermittelt. Alle grundlegenden Geräteparameter entsprachen der Konfiguration aus Kap. 3.3.2. Auf den Endpunkten der Strecke kam erneut `iperf3` (v3.2) als Client-/Server-Anwendung zum Einsatz und es galten dieselben UDP- und TCP-Einstellungen wie in der Performance-Analyse in Kap. 3.3.3.

Wie in Kap. 3.3.5 beschrieben, beträgt die mit dieser Datenrate erzielte Übertragungsreichweite d_{11n} im Freien mehr als 200 m. Das Gelände am Rostocker IGA-Park bot jedoch nur eine Strecke von maximal 450 m für die Ausbringung des Geräte. Um dennoch den Aufbau eines 3-Hop-Pfades zu ermöglichen, wurde für alle Messungen ein reduzierter Platzierungsabstand zwischen aufeinander folgenden Knoten von $0,6 \cdot d_{11n} \approx 125\,m$ gewählt. Für die zugehörigen Messungen im Labor wurde analog dazu ein Platzierungsabstand von $22\,cm$ genutzt. Dadurch war weiterhin die Multi-Hop-Weiterleitung von Frames garantiert, ohne dass Knoten auf dem Pfad übersprungen werden konnten.

Bereits in den Indoor-Messungen wurde ein unerwartet starker Einbruch des Datendurchsatzes mit zunehmender Hop-Anzahl festgestellt. Als Ursache wurde das *Hidden-Node*-Problem identifiziert, das ab einer Pfadlänge von zwei Hops auftreten kann (siehe Grundlagen, Kap. 2.4.5). Als Gegenmaßnahme wurde der optionale Handshaking-Mechanismus *Request-to-Send (RTS) / Clear-to-Send (CTS)* aktiviert. Die beobachtete Linderung des Problems widerspricht früheren Forschungsarbeiten mit Mesh-Netzwerken auf 802.11a/b/g-Basis, die eine Nutzung von RTS/CTS aufgrund seines Nachrichten-Overheads nicht empfehlen [153]. Sie bestätigt dagegen Empfehlungen für die Nutzung von RTS/CTS mit der neueren Spezifikation 802.11n, da diese das Zusammenfassen von Frames erlaubt. Hier überwiegen die Kosten der Neuübertragung großer Frame-Blöcke den Mehraufwand der RTS/CTS-Signalisierung [111].

Wie in der Performance-Analyse in Kap. 3.3.3 wurden 300 Durchsatzmesswerte (5 Messungen über jeweils 60 s, mit sekündlich aufgenommenen Messwerten) für jede Konfiguration bestehend aus WLAN-Datenrate (MCS 0–3), Transportprotokoll (UDP/TCP), Pfadlänge (1–3 Hops) und RTS/CTS-Verwendung (an/aus) gemittelt. Für alle MCS-Einstellungen zeigte sich eine ähnliche Abhängigkeit des Datendurchsatzes von der Pfadlänge. Nachfolgend werden daher nur die Ergebnisse der höchsten

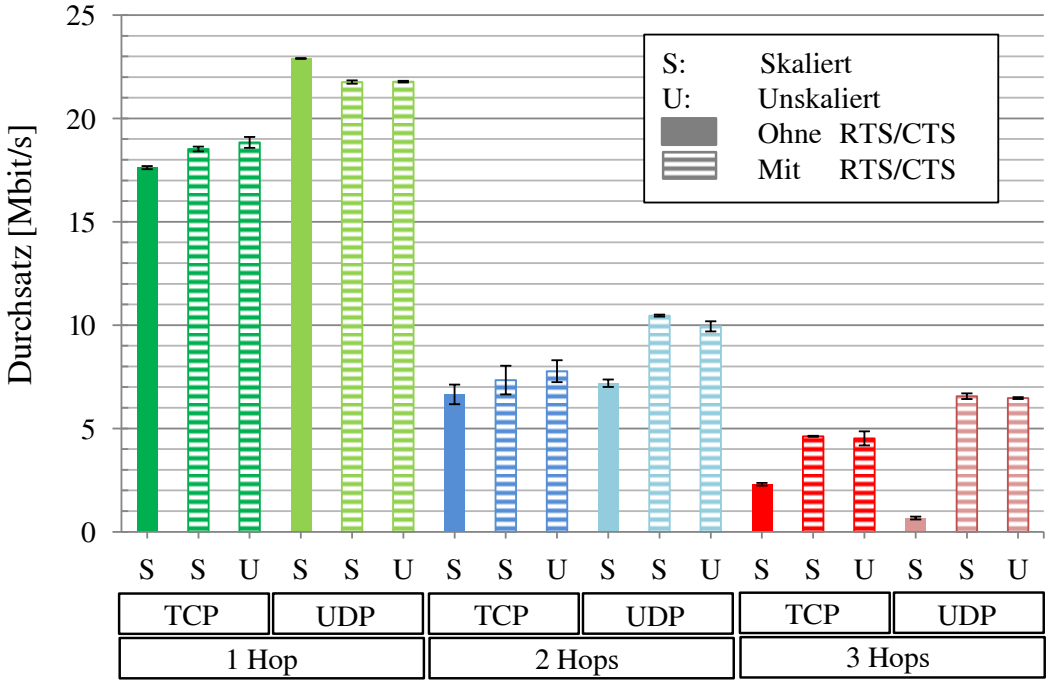


Abbildung 3.6: TCP/UDP-Durchsatz im skalierten und unskalierten Aufbau [B 6]

Datenraten-Einstellung MCS 3 gezeigt. Abb. 3.6 stellt den im skalierten (S) Indoor- und unskalierten (U) Outdoor-Aufbau erreichten Datendurchsatz gegenüber. Die Ergebnisse jeder Pfadlänge sind farbig gruppiert, wobei die TCP-Ergebnisse jeweils in der dunkleren, die UDP-Ergebnisse in der helleren Farbschattierung dargestellt sind. Die Standardabweichung jeder Messreihe ist in schwarz dargestellt. Zur Demonstration des Hidden-Node-Problems sind die Laborergebnisse ohne Verwendung von RTS/CTS (ausgefüllte Balken) sowie mit Verwendung von RTS/CTS (schraffierte Balken) angegeben. Die Outdoor-Messungen wurden stets mit aktiviertem RTS/CTS durchgeführt.

Aufgrund der zusätzlichen Sendevorgänge zur Bestätigung empfangener Segmente besitzt TCP in WLANs eine höhere Kollisionswahrscheinlichkeit im Vergleich zu UDP [173]. Daher zeigte sich im Labor durch Aktivierung von RTS/CTS schon über einen Hop eine leichte TCP-Durchsatzerhöhung in Höhe von 5 %. Über zwei Hops betrug die Verbesserung 9 %, über drei Hops betrug sie 50 % gegenüber dem Fall ohne RTS/CTS. In beiden Fällen konnte die Fluss- und Überlastkontrolle von TCP das Hidden-Node-Problem auch ohne RTS/CTS noch teilweise kompensieren.

Im Gegensatz dazu wirkte sich RTS/CTS bei UDP erst über mehrere Hops positiv aus, dann allerdings erheblich. Durch den unidirektionalen Charakter von UDP sendet im 1-Hop-Fall nur einer der beiden Knoten Nachrichten. Hier entsteht durch das RTS/CTS-Handshaking vorwiegend Overhead. Über zwei Hops konnten jedoch bereits 31 %, über drei Hops sogar 90 % Durchsatzerhöhung mit RTS/CTS festgestellt werden. Da UDP nicht über Fluss- und Überlastkontrollmechanismen verfügt, findet auch keine senderseitige Drosselung und damit Reduzierung der Kollisionswahrscheinlichkeit statt.

Im direkten Vergleich der Indoor- und Outdoor-Ergebnisse (RTS/CTS aktiviert) betrug die mittlere Abweichung zudem stets weniger als 6 %. Dies bestätigt, dass die im miniaturisierten Testaufbau gewonnenen Ergebnisse auch für korrespondierende Topologien ohne Dämpfungsglieder gültig sind, die über das in Kap. 3.3.4 beschriebene Skalierungsmodell ermittelt werden können.

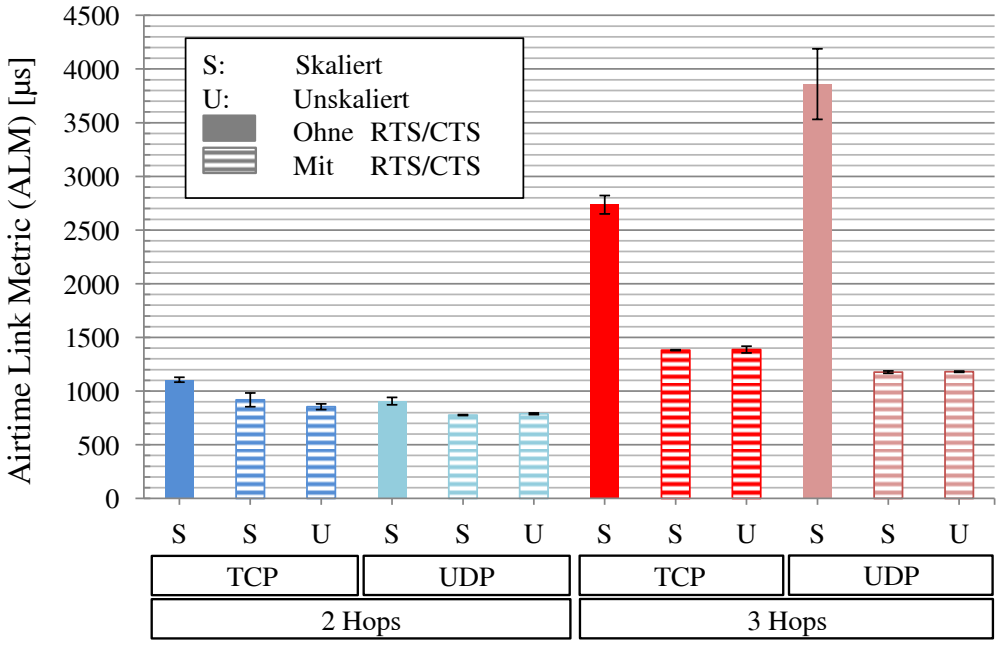


Abbildung 3.7: ALM im skalierten und unskalierten Aufbau

3.4.2 Vergleich der Kostenmetrik ALM

Die ALM ist die im WLAN-Standard spezifizierte Kostenmetrik für die Pfadselektion in 802.11s-Netzwerken (siehe Grundlagen, Kap. 2.5.5). Aufgrund ihrer kumulativen Eigenschaft stellt die ALM sowohl ein Quality of Service (QoS)- als auch ein Distanzmaß für Mesh-Verbindungen dar. Dadurch ist sie auch auf höheren Schichten als Eingangsgröße für Optimierungsansätze von Bedeutung, die Entscheidungen abhängig von der aktuellen Netzwerksituation treffen. Die Unterstützung der ALM ist durch die 802.11s-Spezifikation vorgeschrieben. Somit ist ihre Verfügbarkeit auf jedem standardkonformen Gerät garantiert. Die Integration in Optimierungsansätze ist daher insbesondere auch aus Interoperabilitätsgründen von Vorteil.

Während der Durchsatzmessungen wurde zusätzlich die sich zwischen den Endpunkten des Mesh-Pfades einstellende ALM ermittelt. Parallel zum UDP- und TCP-Durchsatz wurde dazu sekündlich auf dem Sender die Metrik zum Empfänger ausgelesen. Wie zuvor wurde abschließend der Durchschnitt aus 300 Werten gebildet.

Abb. 3.7 stellt die im skalierten (S) Indoor- und unskalierten (U) Outdoor-Aufbau während der jeweiligen Durchsatzmessungen ermittelte ALM der 2-Hop- und 3-Hop-Strecke gegenüber. Die Intervalle der Standardabweichung sind in schwarz dargestellt. Erneut ist für die Indoor-Messungen auch die Konfiguration ohne RTS/CTS aufgeführt.

Der zuvor beobachtete positive Einfluss von RTS/CTS ist auch direkt in der ALM sichtbar. Durch Verringerung der Hidden-Node-Kollisionen sinkt die Anzahl der Übertragungsfehler und damit auch die Metrik, welche die Zeitkosten der Übertragung repräsentiert. Insbesondere über drei Hops zeigt sich ohne Nutzung von RTS/CTS eine deutlich erhöhte ALM. Im Fall von TCP misst diese fast das Doppelte, im Fall von UDP sogar mehr als das Dreifache der Metrik mit aktiviertem RTS/CTS. In der gegenüber dem UDP-Fall geringeren Metrik werden erneut die Vorteile der Fluss- und Überlastkontrolle von TCP sichtbar.

Der Vergleich der Indoor- und Outdoor-Ergebnisse (RTS/CTS aktiviert) zeigt eine mittlere Abweichung von stets weniger als 7 %. Dies bestätigt abermals, dass sich das im Laboraufbau beobachtete Verhalten auf Szenarien ohne Miniaturisierung übertragen lässt.

3.5 Zwischenfazit

In diesem Kapitel wurde die miniaturisierte 802.11n/s-Testumgebung *Mini-Mesh* vorgestellt, mit der realistische WLAN-Mesh-Szenarien in der korrespondierenden Größe eines Stadtviertels auch im Labor praktisch untersucht werden können. Mithilfe eines Ansatzes zur Skalierung der Kommunikationsreichweite wurde eine 36-Knoten-Gitteranordnung auf einer Fläche von lediglich 1 m^2 realisiert. Die Reichweitenbeschränkung basiert auf dem Einsatz von Dämpfungsgliedern im Antennenweg, kombiniert mit der Konfiguration einer reduzierten Sendeleistung und festen Datenrate [B 6].

In einer Performance-Untersuchung wurden zunächst Leistungsgrenzen der Geräte identifiziert und geeignete Arbeitspunkte abgeleitet. Die Skalierungsmethode wurde durch Messung der Reichweiten überprüft, die sich für diese Arbeitspunkte mit und ohne Dämpfungsglieder ergaben. Mit der in der Arbeit genutzten Dämpfung von 30 dB pro Antenne wird reproduzierbar ein Skalierungsmaßstab von rund 1:560 erreicht, sodass die derzeitige miniaturisierte Anordnung einer Fläche von mehr als 300.000 m^2 ohne Dämpfungsglieder entspricht. Basierend auf den Ergebnissen wurde ein mathematisches Modell parametrisiert, das für beliebige Laboranordnungen die Abschätzung korrespondierender Abmessungen ohne Dämpfungsglieder erlaubt.

In weiteren Messungen wurden Kenngrößen auf der Transport- und Sicherungsschicht zwischen miniaturisierter Laboranordnung und korrespondierendem Aufbau ohne Dämpfungsglieder verglichen. Die Ergebnisse für den TCP-/UDP-Durchsatz sowie für die Mesh-Routing-Metrik ALM belegen mit einer mittleren Abweichung von weniger als 7 % klar die praktische Tauglichkeit der Skalierungsmethode und untermauern die Aussagekraft von im Labor durchgeführten Untersuchungen. Als Nebenprodukt dieser Messungen konnte der theoretische positive Einfluss des Handshaking-Verfahrens RTS/CTS zur Linderung des in Mesh-Netzwerken verstärkt auftretenden Hidden-Node-Problems praktisch bestätigt werden. Dies zeigte sich insbesondere anhand eines deutlich erhöhten UDP-Durchsatzes über Multi-Hop-Verbindungen.

Der vorgestellte Ansatz und das mathematische Modell sind auf Szenarien mit einer dominierenden Sichtverbindung (LoS) zwischen benachbarten Knoten beschränkt. Dies erlaubt die Vernachlässigung von Mehrwegeausbreitungseffekten, sodass reproduzierbare Ergebnisse für Diversity MIMO unabhängig von der konkreten Laborumgebung entstehen. Im weiteren Verlauf der Arbeit dient *Mini-Mesh* als Umgebung zur prototypischen Entwicklung und praktischen Analyse zweier Cross-Layer-Optimierungsansätze für 802.11s-Netzwerke.

4 Kollaborativer Datenaustausch in IEEE-802.11s-Netzwerken

4.1 Motivation

Wireless Mesh Networks (WMNs) zeichnen sich durch eine flexible und robuste Netzwerkstruktur aus. Durch Spontanvernetzung und Routing ermöglichen sie eine einfache und kostengünstige drahtlose Abdeckung großer Areale [7]. Die Wireless Local Area Network (WLAN)-Standard-Erweiterung IEEE 802.11s [5, 12] spezifiziert grundlegende Mesh-Funktionen bereits als Teil der 802.11-Sicherungsschicht und bildet damit die Basis für interoperable Mesh-Lösungen zur Realisierung von Smart-City-Anwendungen [3, 8].

Peer-to-Peer (P2P)-Protokolle sind in diesem Zusammenhang ideal für den Aufbau verteilter Applikationen in WLAN-Mesh-Netzwerken, da sie deren Vorteile wie Skalierbarkeit und Ausfallsicherheit auf der Anwendungsschicht ergänzen [174, 175]. Das P2P-Prinzip beschreibt ein dezentrales Entwurfsparadigma für skalierbare und lose gekoppelte Kommunikationsdienste [176, 177]. Es bildet oberhalb einer physischen Netzwerkstruktur (Underlay) ein logisches Netzwerk (Overlay) aus gleichberechtigten Teilnehmern (Peers). Im Gegensatz zur klassischen zentralisierten Client-Server-Architektur übernimmt jeder Peer sowohl die Rolle eines Clients als auch eines Servers. Auf diese Weise wird ein Single Point of Failure (SPoF) im Netzwerk vermieden. Ein vielversprechender Schritt ist die Kombination der P2P- und Mesh-Technologie, um robuste Kommunikationsdienste zu ermöglichen, wie z. B. die kollaborative Datenverteilung und -synchronisation in zukünftigen Smart Cities. Ein Anwendungsfall besteht in der effizienten, kollaborativen Ausbringung von Software-Updates innerhalb eines Mesh-Backbones im Stadtgebiet aus der Sicht eines Netzwerkbetreibers. Vor dem Hintergrund einer drahtlosen Infrastruktur, die gleichzeitig Informationsdienste bereitstellt, ist ein weiteres Szenario die schnelle Verteilung von Medieninhalten an öffentliche Bildschirme und Anzeigetafeln. Aufsetzende P2P-Anwendungen müssen sich dabei automatisch an Änderungen im Mesh-Netzwerk anpassen, um die verfügbaren Kommunikationsressourcen effizient nutzen zu können.

BitTorrent (BT) [178] ist das meistgenutzte P2P-Protokoll für den kollaborativen Datenaustausch im Internet [32]. Auszutauschende Daten werden dabei in Teile fester Größe zerlegt und zwischen interessierten Peers weitergegeben. BT nutzt einen eigenen Mechanismus zur Auswahl von Upload-Kandidaten aus der Menge aller am Datensatz interessierten Peers. Einem Belohnungsprinzip folgend erhalten Peers mit der größten geteilten Datenmenge (Upload) wiederum Download als Gegenleistung. BT ermöglicht insbesondere die effiziente Ausbringung von Daten, die nur für eine Gruppe von Knoten im physischen Netzwerk bestimmt sind. In einem Smart-City-Szenario ist ein möglicher Einsatzzweck z. B. die Verteilung von Updates an eine konkrete Geräteklasse. Abb. 4.1 skizziert ein Beispiel, in dem verteilte drahtlose Kamerasysteme durch den Netzwerkbetreiber ausgehend von einem administrativen Zugangspunkt aktualisiert werden sollen. Nach Übertragung erster Datenteile von der Quelle an die Zielknoten (dargestellt in rot) können diese sich zusätzlich kollaborativ untereinander und über möglicherweise kürzere Mesh-Pfade versorgen (dargestellt in grün), wodurch sich der Update-Prozess beschleunigt.

Herkömmliche P2P-Protokolle wie BT wurden jedoch für die Nutzung über kabelgebundene Netzwerke und das Internet entworfen. Die Organisation des logischen Overlays erfolgt unabhängig vom physischen Underlay und ist auf zuverlässige Verbindungen und stabile Kanalbedingungen ausgelegt [174, 175, 179–182]. Die klassische BT-Peer-Auswahl berücksichtigt daher weder die physische Nähe von Teilnehmern noch die dynamischen Verbindungseigenschaften eines drahtlosen Underlays [33–35]. Im Gegensatz zu kabelgebundenen Netzwerken besitzen WLANs ein geteiltes Funkmedium und sind anfällig gegenüber externen Störungen. Konkurrierender Kanalzugriff und Mechanismen zur Fehlerbehandlung auf der Sicherungsschicht führen in der Regel zu einer höheren und dynamischeren Bitfehlerrate und Kommunikationslatenz [15]. In WLAN-Mesh-Netzwerken intensivieren sich diese Effekte mit steigender Teilnehmerzahl und Pfadlänge. Dies führt in P2P-

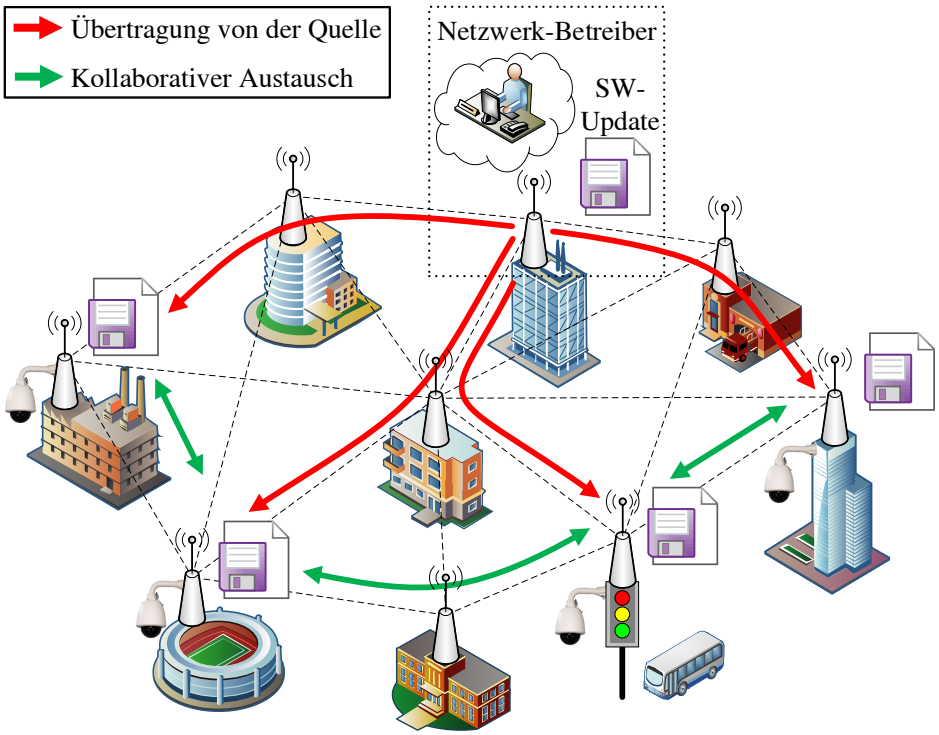


Abbildung 4.1: Kollaborative Datenausbringung im Smart-City-Szenario

Protokollen wie BT zu gravierenden Leistungseinbußen. Verschiedene Forschungsarbeiten beschäftigen sich daher mit einer Berücksichtigung der physischen Netzwerkstruktur innerhalb des BT-Overlays [28–30]. Es existieren bislang jedoch keine Arbeiten, die BT gezielt für den Einsatz in 802.11s-basierten WLAN-Mesh-Netzwerken optimieren.

Die in 802.11s definierte Routing-Metrik *Airtime Link Metric (ALM)* repräsentiert die sich aus der Gesamtlänge eines Mesh-Pfades und der Qualität seiner Teil-Links ergebenden Übertragungszeitkosten. Sie stellt somit ein besseres Kriterium zur Peer-Auswahl in WLAN-Mesh-Netzwerken dar als die für das Internet ausgelegte Belohnungsstrategie von Standard-BT. Im Rahmen dieser Arbeit wird daher der Cross-Layer-Ansatz *Mesh-Network-Aware BitTorrent (MeNTor)* entwickelt, welcher die folgenden Optimierungen umfasst:

- Die Link- und Pfadinformationen des 802.11s-Protokolls werden über bestehende Schnittstellen des Betriebssystems als Prioritätskriterien in die BT-Peer-Auswahl integriert. Dies ermöglicht eine Berücksichtigung des Underlays auf der BT-Anwendungsschicht, ohne Änderungen an unteren Schichten vornehmen zu müssen. Folglich werden die Peers mit den geringsten Kommunikationskosten bevorzugt bedient, wodurch sich der Zeitbedarf zur Ausbringung der Daten im Overlay reduziert.

- Es werden BT-Protokollanpassungen untersucht, die den gleichzeitigen Upload zu verschiedenen Peers sowie die zufällige Auswahl sogenannter *optimistischer* Peers in Mesh-Netzwerken vermeiden. Dadurch wird die Anzahl sich gegenseitig beeinträchtigender Übertragungsvorgänge im Netzwerk reduziert.

- Durch Einführung eines *Generous Mode* wird erreicht, dass Peers einander gänzlich ohne Belohnungsprinzip bedienen, d.h. rein kollaborativ und lediglich auf Basis ihrer Kommunikationskosten zueinander.

Zur praktischen Bewertung des Ansatzes wird eine prototypische Implementierung als Plugin für den bekannten BT-Client *Vuze* [183] entwickelt. Untersuchungen in der realen Testumgebung *Mini-Mesh* sollen zeigen, ob und in welchem Umfang MeNTor die mittlere Download-Zeit pro Peer und im gesamten Overlay bereits für Netzwerkgrößen von bis zu 25 Knoten reduzieren kann. Die vorgestellten Ergebnisse wurden international veröffentlicht [B 4, 17]. Die Thematik war zudem Bestandteil mehrerer studentischer Arbeiten am Institut für angewandte Mikroelektronik und Datentechnik, die durch den Autor betreut wurden [C 8, 9, 24].

4.2 P2P-Protokoll BitTorrent

BitTorrent (BT) [178] ist ein von Bram Cohen entwickeltes, auf der Anwendungsschicht angesiedeltes P2P-Protokoll. Seit seiner Referenzimplementierung aus dem Jahr 2001, nachfolgend als *Standard-BT* bezeichnet, wurde es von einer großen Entwicklergemeinde fortlaufend erweitert [31]. In diesem Zuge sind zahlreiche BT-Implementierungen entstanden, die sich in Details unterscheiden und verschiedene optionale Protokollerweiterung unterstützen können. BT ist bis heute das dominierende Protokoll für den effizienten kollaborativen Datenaustausch im Internet. Im Jahr 2019 besaß es einen Anteil von ca. 27,5 % am gesamten Upstream-Datenverkehr der weltweiten Zugangsnetze [32]. Seine Beliebtheit resultiert aus der Eigenschaft, durch die Bündelung der Upload-Kapazitäten aller Teilnehmer und deren parallelen Datenaustausch hohe Download-Datenraten erreichen zu können, die nicht durch einzelne Server begrenzt sind. Die redundante Datenhaltung bewirkt zudem eine hohe Verfügbarkeit [178]. Wie viele andere Anwendungen, die eine zuverlässige Datenübertragung realisieren, nutzt BT als Transportprotokoll typischerweise das Transmission Control Protocol (TCP).

4.2.1 Grundprinzip

Alle an einem bestimmten Datensatz interessierten BT-Nutzer (Peers) bilden ein logisches Netzwerk, den sogenannten *Schwarm*. Abb. 4.2 demonstriert die möglichen Peer-Rollen sowie die grundlegenden Schritte zur Schwarmbildung. Peers, die den Datensatz bereits vollständig bezogen haben und ausschließlich zum Download anbieten, werden *Seeds* genannt. Derjenige Peer, der den kompletten Datensatz zuerst in den Schwarm einbringt, wird als *initialer Seed* bezeichnet. Sich noch im Download befindliche Peers, die gleichzeitig bereits empfangene Dateiteile kollaborativ an andere Peers weitergeben, werden *Leecher* genannt. Im Beispiel in Abb. 4.2 besteht das P2P-Netzwerk aus einem initialen Seed und den später beitretenden Leechern A und B.

Die Suche nach einem Schwarm für einen gewünschten Datensatz erfolgt in der Regel über ein separates Webportal, das die für den Schwarmbeitritt benötigten Metadaten in Form einer **.torrent*-Datei besitzt. Diese wurde im Vorfeld beispielsweise durch einen initialen Seed erzeugt und auf dem Web Server veröffentlicht (Schritt 1 in Abb. 4.2). In den Schritten 3 und 6 der Abb. 4.2 beziehen Leecher A bzw. B die Metadaten vor ihrem Beitritt zum Schwarm. Die **.torrent*-Datei enthält unter anderem einen Hashwert als Kennung des Schwarms, Informationen über die Segmentierung des Datensatzes sowie die Adresse eines *Trackers*. Beim Tracker handelt es sich um einen Bootstrap-Server, der die Liste aller aktuellen Teilnehmern des Schwarms verwaltet und neuen Peers bei deren Anmeldung bereitstellt (Schritte 2, 4 und 7).

BT unterteilt die auszutauschenden Daten in *Pieces*, die wiederum aus sogenannten *Sub-Pieces* bestehen. Die Größe der Pieces ist konfigurationsabhängig und liegt zwischen 32 kB und 32 MB, während Sub-Pieces eine vorgegebene Standardgröße von 16 kB besitzen. Pieces werden in Form ihrer Sub-Pieces und daher in mehreren Einzelnachrichten versandt. Die Überprüfung der Checksumme der Daten wird stets für vollständige Pieces durchgeführt. Erst nach erfolgreichem Empfang eines Pieces können seine Sub-Pieces an andere Peers weitergegeben werden. In Schritt 5 der Abb. 4.2

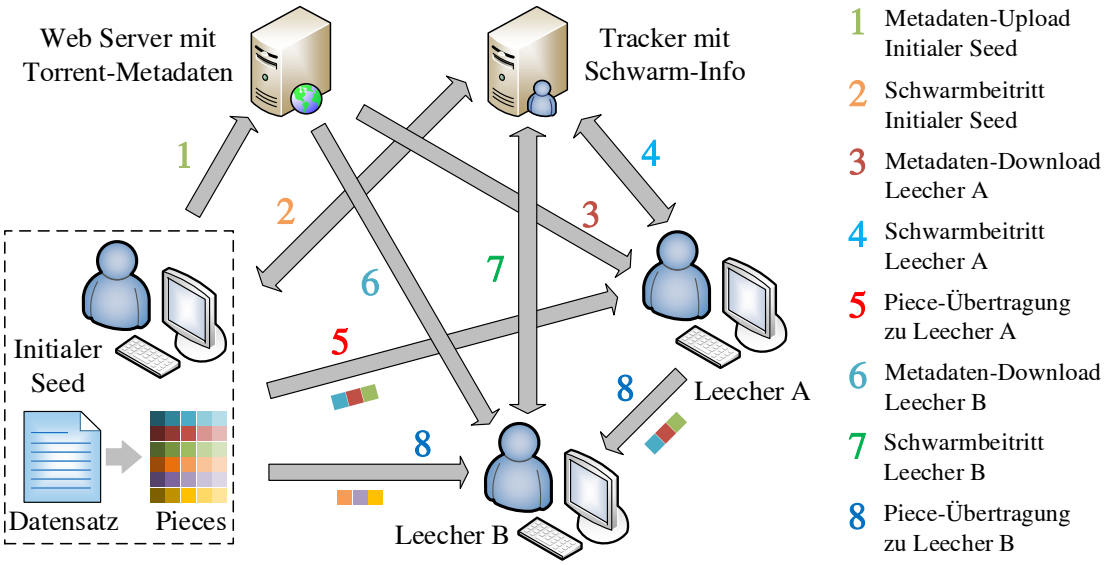


Abbildung 4.2: BitTorrent-Grundprinzip – Schwarmbildung und Datenaustausch

Tabelle 4.1: Übersicht der BitTorrent-Nachrichten

Nachricht	**Beschreibung**
`handshake`	initiiert den Verbindungsaufbau zu einem Peer, enthält Schwarm-Hash und eigene Peer-ID
`bitfield`	auf Handshake folgender Bitvektor zur Bekanntgabe schon empfangener / noch benötigter Pieces
`have`	signalisiert einem Peer den vollständigen Erhalt eines Pieces
`interested`	signalisiert einem Peer, dass dieser noch benötigte Pieces besitzt
`not interested`	signalisiert einem Peer, dass keine seiner Pieces benötigt werden
`request`	fordert ein bestimmtes (Sub-)Piece zum Download an
`piece`	überträgt ein bestimmtes (Sub-)Piece an einen anfragenden Peer
`cancel`	bricht eine begonnene (Sub-)Piece-Übertragung ab
`unchoke`	signalisiert einem Peer die vorübergehende Download-Freigabe
`choke`	signalisiert einem Peer die vorübergehende Download-Sperre

erhält Leecher A nach seinem Schwarmbeitritt erste Pieces vom initialen Seed. In Schritt 8 wird Leecher B bereits gleichzeitig von den beiden anderen Peers mit Pieces versorgt. Generell beginnen Leecher damit, bei anderen Peers zufällige Pieces anzufordern. Nach Erhalt des ersten Pieces wechseln sie auf eine Reihenfolge, die stets das im Schwarm am seltensten vorkommende Piece bevorzugt. Dabei gilt die Priorisierung, zunächst alle Sub-Pieces des aktuellen Pieces anzufragen, bevor zum nächsten Piece gewechselt wird.

Das BT-Protokoll definiert verschiedene Kontrollnachrichten, von denen die wichtigsten in Tab. 4.1 zusammengefasst werden. Neben dem Verbindungsaufbau zwischen Peers mittels `handshake`- und `bitfield`-Nachrichten dienen weitere Nachrichtentypen insbesondere der gezielten Anfrage und Übertragung von (Sub-)Pieces. Als Ergebnis der auf jedem Peer zyklisch durchgeführten Auswahl von Upload-Zielen werden zudem `unchoke`- bzw. `choke`-Nachrichten an andere Peers gesendet,

die eine Download-Freigabe bzw. -Sperre signalisieren. Der zugrunde liegende *Choking*-Algorithmus wird nachfolgend im Detail erläutert.

4.2.2 Choking-Algorithmus zur Peer-Auswahl

Eine Kernkomponente des BT-Protokolls ist die Peer-Auswahl im Zuge der kollaborativen Weitergabe der Daten. Zur Bestimmung der Peers, die eine Download-Möglichkeit erhalten, führt jeder Peer den sogenannten *Choking-Algorithmus* aus (*choke* = engl. für „drosseln“) [178, 184]. Peers, die eine Download-Erlaubnis erhalten, werden im BT-Terminus *unchoked*. Im Gegenzug werden nicht länger bevorzugte Peers bis zu ihrer Neubewertung *choked*, d.h. der Upload zu ihnen wird eingestellt. Auf einem *Leecher*, der selbst noch Pieces des Datensatzes bezieht, erfolgt die Peer-Auswahl nach dem Belohnungsprinzip. Aus seiner Sicht werden andere Peers nach ihrer Upload-Rate zu ihm bewertet und erhalten dafür entsprechend Download-Zeit. Sobald ein Leecher alle Daten vollständig bezogen hat, kann er sich entscheiden, noch für eine bestimmte Zeit (engl. *lingering time*) im Netzwerk zu verbleiben und als *Seed* zu fungieren. Abweichend von der Idee des Belohnungsprinzips bevorzugt ein Seed den Upload zu Peers, zu denen er die höchste Upload-Datenrate erreicht.

Der Choking-Algorithmus wird periodisch aufgerufen, da sich die Konstellation der Peers und deren Upload-Verhalten zueinander schnell verändern können. In diesem Zusammenhang werden vom BT-Entwickler Bram Cohen Standardwerte zur Parametrisierung des Algorithmus vorgeschlagen [178]. Demnach beträgt das Ausführungsintervall auf einem Leecher 10 s, auf einem Seed 30 s. Bei einer gegebenen Menge an Upload-Slots N (üblicherweise $N = 5$) wird $N - 1$ Peers die Download-Erlaubnis während des kommenden Intervalls gewährt. Diese Slots werden auch als *regular unchokes* bezeichnet. Als Ausnahme vergibt jeder Peer alle 30 s einen zusätzlichen Upload-Slot, den sogenannten *optimistic unchoke*. Dieser wird einem zufälligen Peer „optimistisch“, d.h. unabhängig von einem Bewertungsmaß zugewiesen. Dadurch wird das Kennenlernen neuer, möglicherweise besser geeigneter Peers erlaubt, zu denen noch keine Informationen existieren. Zusätzlich sollen so stichprobenartig auch potentiell unterversorgte Bereiche des Schwarms bedient werden, wodurch sich langfristig die Verfügbarkeit von Pieces und die Anzahl parallel stattfindender Datenübertragungen im Netzwerk erhöhen [178, 184]. Zur weiteren Unterstützung einer fairen Peer-Auswahl nutzt BT die sogenannte *Anti-Snubbing*-Strategie als Teil des Choking-Algorithmus. Sie verhindert bei der Vergabe der *regular unchokes* die Auswahl von Peers, die über einen Zeitraum von einer Minute kein einziges Sub-Piece im Gegenzug bereitgestellt haben.

Die klassische BT-Peer-Auswahl berücksichtigt weder die physische Nähe von Teilnehmern noch die dynamischen Verbindungseigenschaften in drahtlosen Mesh-Netzwerken [33–35]. Das Treffen der Unchoke-Entscheidungen ist in der BT-Spezifikation jedoch nicht fest vorgeschrieben, sodass an dieser Stelle die Möglichkeit für Optimierungen besteht [31]. Nachfolgend werden verschiedene Forschungsarbeiten diskutiert und gegenübergestellt, die Anpassungen von BT für Mobile Ad-Hoc Networks (MANETs) und WLAN-Mesh-Netzwerke untersuchen.

4.3 Stand der Technik

Die Surveys [28–30] geben einen Überblick über bisherige Forschungsarbeiten, die sich mit dem P2P-basierten Datenaustausch in drahtlosen Netzwerken beschäftigen. In der Literatur wird bereits eine Reihe von Ansätzen vorgeschlagen, die BT oder an dieses angelehnte P2P-Protokolle auch für den Einsatz in drahtlosen Ad-Hoc- und Mesh-Netzwerken optimieren [33, 180, 181, 185–196]. Der Großteil der Arbeiten basiert jedoch auf früheren, meist zueinander inkompatiblen Mesh-Routing-Protokollen und berücksichtigt nicht die Standardmechanismen der neueren 802.11s-Spezifikation. In einigen Ansätzen werden proaktive Link-State-Routing-Verfahren wie Optimized Link State Routing

Tabelle 4.2: Gegenüberstellung von *MeNTor* mit verwandten Forschungsarbeiten [B 4]

Arbeit	Kompatibel zu 802.11s	Cross-Layer-Ansatz	Kein Zusatz-Datenverkehr	Optimierte Peer-Auswahl	WLAN-spezifische Metrik	Reales Testbed
BEAN [185]	✓	✓	✓	—	—	—
BTM [186]	—	✓	—	✓	—	—
Balázsfalvi et al. [187]	✓	—	✓	✓	—	—
ElRakabawy et al. [180]	—	✓	✓	✓	✓	—
Michiardi et al. [188]	—	✓	✓	✓	—	—
BitHoc [189]	—	✓	✓	✓	—	✓
Marques et al. [181]	—	✓	✓	—	✓	—
Sbai et al. [190]	✓	—	✓	—	—	—
Quental et al. [191]	✓	—	✓	—	—	—
D'Elia et al. [192]	—	✓	—	✓	✓	✓
Di Stasi et al. [193]	✓	✓	—	—	—	✓
Mawji et al. [194]	✓	—	✓	—	—	—
BestPeer [195]	✓	✓	—	✓	—	—
Manousakis et al. [33]	✓	✓	—	✓	✓	—
Casini et al. [196]	✓	—	✓	—	—	—
MeNTor [B 4]	✓	✓	✓	✓	✓	✓

(OLSR) oder Better Approach to Mobile Ad-Hoc Networking (BATMAN) zugrunde gelegt, die ein umfassenderes Netzwerkwissen pro Knoten verwalten als das in 802.11s vorgeschriebene Distance-Vector-Verfahren Hybrid Wireless Mesh Protocol (HWMP) [188, 189, 192, 197]. Ebenso werden andere Routing-Metriken als die ALM angenommen, die in einem 802.11s-Netzwerk nicht ohne Weiteres zur Verfügung stehen. Daher lassen sich viele der bestehenden Lösungen nur eingeschränkt auf 802.11s-Netzwerke übertragen.

Tab. 4.2 stellt die verwandten Arbeiten hinsichtlich der Eigenschaften des im Rahmen dieser Dissertation entwickelten Ansatzes *MeNTor* gegenüber. Die Forschungsbeiträge sind nach dem Jahr ihrer Veröffentlichung sortiert. Einige von ihnen verfolgen jedoch keinen Cross-Layer-Ansatz und werden daher nicht im Detail diskutiert. Dies betrifft die Arbeiten [187, 190, 191, 194, 196]. Allerdings präsentiert [196] eine jüngere Untersuchung in diesem Bereich, in der das unicast-basierte Standard-BT mit einem broadcast-basierten Ansatz verglichen wird, der speziell auf MANETs zugeschnitten ist. Die Autoren schlussfolgern nach Experimenten in einem emulierten Netzwerk, dass das unmo-

difizierte BT hinter den Ergebnissen der eigenen Speziallösung zurückbleibt. Ein Vergleich mit BT-Varianten, die für WMNs optimiert sind, wird als zukünftiges Ziel genannt.

In [185] wird das Routing-Protokoll *BEAN* für MANETs vorgeschlagen. Da diese Netzwerke anfällig für häufige Topologieänderungen sind, pflegt das Protokoll mehrere unabhängige Routen zwischen Quell- und Zielknoten. Dazu kombiniert es Mechanismen der Routing-Protokolle Ad-Hoc On-Demand Distance Vector Routing (AODV) und Dynamic Source Routing (DSR) mit einer verteilten Verwaltung der Routen-Informationen auf Basis von BT. Routen sollen anhand ihres Hop Counts sowie Informationen zur Bandbreite und Latenz charakterisiert werden. Die Arbeit trifft jedoch keine Aussagen dazu, wie diese Quality of Service (QoS)-Informationen in der Praxis zu ermitteln sind. Zudem wird die eigene Lösung weder simulativ noch praktisch untersucht.

Die Arbeit [186] vergleicht eine BT-Implementierung namens *BTI*, die einige Erweiterungen für MANETs besitzt, mit einer Cross-Layer-Variante von BT namens *BTM*. Beide Varianten basieren auf dem eigens entwickelten reaktiven Routing-Protokoll *ANSI*. Die Autoren untersuchen das Verhalten hinsichtlich Durchsatz, Download-Zeitbedarf und Routing-Overhead und kommen zu dem Schluss, dass *BTM* sowohl *BTI* als auch das klassische BT schlägt. Allerdings birgt *BTM* den Nachteil eines erhöhten Ressourcen-Verbrauchs, da benötigte Zusatzinformationen in Form eines Netzwerk-Caches bereitgestellt und ständig abgefragt werden müssen.

Ein kooperatives P2P-Datenübertragungsprotokoll für WMNs wird in [180] vorgestellt. Es wählt solche Peers bevorzugt aus, die eine minimale Interferenz auf dem Download-Pfad aufweisen. Der beschriebene Cross-Layer-Ansatz arbeitet wie BT auf Ende-zu-Ende-Basis, bettet allerdings Informationen der Anwendungsschicht in Routing-Nachrichten der Vermittlungsschicht ein. Zusätzlich wird Overhead-Datenverkehr durch aktive Latenzmessungen erzeugt, auf Basis derer eine Abschätzung der Interferenz erfolgt. Im Gegensatz dazu nimmt die im Rahmen dieser Arbeit vorgestellte BT-Optimierung MeNTor keinerlei Änderungen an den Standardmechanismen von 802.11s und dessen Mesh-Routing-Protokoll vor. Auch werden durch MeNTor keine zusätzlichen Nachrichten versandt, sondern stattdessen inhärent verfügbare Informationen der Sicherungsschicht an BT weitergereicht.

In [188] wird ein an BT angelehntes, kooperatives P2P-System für den Einsatz in WMNs vorgeschlagen, das die Reichweite von Datenübertragungen beschränkt. Auf der Vermittlungsschicht wird das proaktive Link-State-Routing-Protokoll OLSR angenommen, das auf jedem Teilnehmer globales Netzwerkwissen inklusive der Hop-Distanz aller Pfade bereitstellt. Das Hauptziel besteht darin, lange Pfade im physischen Underlay und dadurch entstehende Interferenzen zu vermeiden, um insgesamt höhere Download-Geschwindigkeiten zu erreichen. Die Peer-Auswahl nutzt weiterhin ein Belohnungsprinzip ähnlich zu BT. Die Steuerung der maximalen Pfadlänge erfolgt hingegen durch Vorgabe der Time-to-Live (TTL) für IP-Pakete. Dieser Vorschlag birgt die Gefahr der Partitionierung des P2P-Netzwerks für Szenarien, in denen nicht alle Knoten des physischen Underlays auch Teilnehmer im logischen Overlay sind. Im ungünstigen Fall ist die Hop-Distanz zwischen zwei logischen Zellen größer als die gewählte TTL-Schranke, sodass keine Verbindung entsteht und ein vollständiger Datenaustausch nicht garantiert werden kann. Die Bestimmung einer optimalen TTL erfordert globales Netzwerkwissen, das jedoch in 802.11s-Netzwerken nicht ohne Weiteres zur Verfügung steht.

Das *BitHoc*-Projekt verfolgt eine Anpassung des BT-Protokolls für drahtlose Ad-Hoc-Netze [35, 189]. Auch hier wird die Kommunikationsreichweite zwischen Peers auf wenige Hops beschränkt, um die Belastung des Netzwerks durch lange Pfade zu reduzieren. Wie schon in [188] dient OLSR als Routing-Protokoll und es wird eine TTL-Vorgabe für dessen HELLO-Pakete getroffen, die dem gegenseitigen Finden von Peers dienen. Die aus dem Routing-Protokoll bekannte Hop-Distanz wird zudem auch in der BT-Peer-Auswahl genutzt, um die Anzahl berücksichtigter Peers auf nahe Knoten zu reduzieren. Ergebnisse simulativer und praktischer Experimente zeigen, dass durch die eingeführte Reichweitenbeschränkung der Zeitbedarf für die Datenverteilung sinkt und sich der mittlere Datendurchsatz erhöht. Um eine höhere Streuung von Pieces im Overlay zu

gewährleisten, erlaubt *BitHoc* weiterhin die auch in BT vorgesehene Datenübertragung zu einigen zufällig gewählten Peers, die als Ausnahme auch weiter entfernt liegen dürfen. In [197] erweitern die Autoren ihre Untersuchungen auf mobile Szenarien. Sie argumentieren, dass durch die höhere Vakanz der Knoten die Auswahl von zufälligen Peers unnötig wird. Zusammengefasst übernimmt *BitHoc* die potentiellen Probleme des Ansatzes aus [188]. Um die Konnektivität des Overlays nicht zu gefährden, sind komplizierte Heuristiken zur Wahl der TTL und Erkennung der Mobilität notwendig. Ebenso bleiben WLAN-spezifische QoS-Metriken, wie z. B. die in 802.11s definierte ALM, als Kriterien für die Peer-Auswahl unberücksichtigt. Diese bietet jedoch Vorteile, da sie im Gegensatz zum Hop Count auch Rückschlüsse auf die Verbindungsqualität zulässt. Typischerweise können so kaum genutzte längere Pfade von kürzeren, stark belasteten Pfaden unterschieden werden.

Der in [181] beschriebene Ansatz verfolgt als einziger die Cross-Layer-Optimierung von P2P-Anwendungen oberhalb eines 802.11s-WMN durch Integration der Routing-Metrik ALM. Die Autoren schlagen ein generisches Software-Framework vor, das als Schnittstelle für P2P-Anwendungen fungieren und diesen die ALM bereitstellen soll. Es werden jedoch keine spezifischen Anwendungen untersucht und die praktischen Auswirkungen einer ALM-Integration auf der Anwendungsschicht bleiben unbewertet. Zudem muss das 802.11s-Routing-Protokoll HWMP, insbesondere durch Einführung neuer Frame-Felder in Kontrollnachrichten, stark erweitert werden. Im Gegensatz dazu besteht die Zielstellung der vorliegenden Arbeit darin, keine Änderungen an der Sicherungsschicht vorzunehmen und Interoperabilität mit Standard-802.11s zu gewährleisten.

In [192] wird eine Cross-Layer-Lösung präsentiert, welche die Funktionalität des BT-Trackers erweitert. Dieser verwaltet eine Liste der aktiven Teilnehmer und unterstützt neue Peers beim Netzwerkbeitritt. Der vorgeschlagene Ansatz ergänzt das Netzwerk um eine Server-Infrastruktur auf Basis des von der Internet Engineering Task Force (IETF) spezifizierten Protokolls Application-Layer Traffic Optimization (ALTO). Der ALTO-Server versorgt den Tracker mit Informationen über die QoS der Pfade zwischen Peers. Diese gewinnt er zuvor aus dem proaktiven Routing-Protokoll OLSR, das im Zusammenhang mit der Metrik Expected Transmission Count (ETX) genutzt wird. Folglich beantwortet der Tracker nun Anfragen mit einer nach ETX sortierten Peer-Liste, anstatt eine zufällige Auswahl zu treffen. Allerdings erlaubt ETX nur eine grobe Schätzung der Verbindungsqualität, indem die Anzahl erfolgreicher und fehlgeschlagener Paketübertragungen auf der Vermittlungsschicht beobachtet werden. Die in 802.11s spezifizierte Metrik ALM berücksichtigt hingegen alle Übertragungen von Unicast-Daten-Frames auf der Sicherungsschicht und setzt Übertragungsfehler in Beziehung zur jeweils genutzten WLAN-Datenrate. Darüber hinaus ist die Unterstützung der ALM und des Routing-Protokolls HWMP in 802.11s vorgeschrieben. Im Gegensatz zu anderen Protokollen und Metriken kann ihre Verfügbarkeit daher auf standardkonformen Geräten vorausgesetzt werden.

Der Fokus der Arbeit [193] liegt in der Anbindung der WMNs-Testbeds *WiLEE* an das globale Forschungsnetz *PlanetLab*. Eine Optimierung des BT-Trackers wird als Testanwendung hinzugezogen. Diese unterscheidet die Netzgrenzen der Internet Service Providers (ISPs) und trifft dadurch grobe Aussagen über die Nähe von Peers.

In [195] wird eine neue Metrik zur BT-Peer-Auswahl in WMNs namens *BestPeer* vorgestellt, die Verbindungseigenschaften berücksichtigt. Diese führt jedoch aktive Messungen auf Basis periodisch versendeter Nachrichten durch, um eine Abschätzung der Link-Auslastung zu erhalten. Im Gegensatz dazu integriert der Ansatz der vorliegenden Arbeit direkt die 802.11s-Routing-Metrik ALM, ohne zusätzlichen Datenverkehr zu erzeugen.

Ein Framework für den verteilten, BT-basierten Datenaustausch in WMNs wird in [33] vorgeschlagen. Es benötigt keinen Tracker-Knoten, bezieht jedoch Informationen zur Hop-Distanz, Latenz und Paketverlustrate zwischen Peers aus aktiven Messungen, die das Netzwerk zusätzlich belasten.

Im Gegensatz zu den bestehenden Lösungen ist der in dieser Arbeit entwickelte Cross-Layer-Optimierungsansatz *MeNTor* direkt auf 802.11s-WMN zugeschnitten. Inhärent verfügbare Informa-

tionen der WLAN-Sicherungsschicht und des 802.11s-Routing-Protokolls HWMP werden in die BT-Anwendungsschicht gereicht, ohne dass zusätzlicher Datenverkehr entsteht. Durch Integration der 802.11s-Routing-Metrik ALM als Kriterium zur Peer-Auswahl werden die physische Nähe von Teilnehmern und die Kommunikationskosten ihrer Verbindungen berücksichtigt. Die Optimierung wird prototypisch als Plugin für den BT-Client *Vuze* entwickelt und in einem realen Testbed untersucht.

4.4 Cross-Layer-Optimierung *MeNTor*

4.4.1 Ausgangsszenario

In der Vision der intelligent vernetzten Stadt, der Smart City, bestehen für Anbieter städtischer WMNs große Herausforderungen hinsichtlich des skalierbaren Betriebs und gleichzeitigen Managements des Netzwerks. Die im Stadtgebiet verteilten Mesh-Knoten sind oft an schwer erreichbaren Stellen platziert und besitzen außer einer Stromversorgung keine Anbindung an eine kabelgebundene Kommunikationsinfrastruktur [26]. Das Mesh-Netzwerk als drahtloses Backbone für den Internetzugang und andere Dienstleistungen ist daher gleichzeitig auch sein eigener Wartungskanal. Hier fallen insbesondere administrative Aufgaben ins Gewicht, die ein hohes Datenaufkommen verursachen, wie z. B. das gezielte Ausrollen von Software-Updates an Gerätegruppen oder die Synchronisation von Statusinformationen zwischen Management-Instanzen.

Auf den ersten Blick erscheint ein Multicast-Ansatz als mögliche Lösung dieses Problems geeignet. Im Zusammenspiel mit aktueller WLAN-Hardware ergeben sich jedoch mehrere Einschränkungen [198]. Im Allgemeinen werden Multicasts in WLANs unzuverlässig als unbestätigte Broadcasts im Netzwerk verschickt. In einem herkömmlichen WLAN-Setup erfolgt die Weiterleitung von Client-Broadcasts durch die zentrale Basisstation und alle Teilnehmer werden spätestens nach zwei Hops erreicht. In einem Mesh-Szenario hingegen gibt jeder Knoten eingehende Nachrichten erneut an alle seine Nachbarn weiter, wodurch das gesamte Netzwerk geflutet wird. Dies ist insbesondere dann ein Nachteil, wenn die Daten nur für eine Teilmenge aller Knoten bestimmt sind.

Die Möglichkeit einer effizienteren und zuverlässigeren Gruppenkommunikation besteht mit der Standarderweiterung IEEE 802.11aa und deren Spezifikation eines *Group Addressed Transmission Service (GATS)* [199]. Neben der Bildung von Multicast-Gruppen auf der Sicherungsschicht definiert GATS verschiedene Arten, Daten-Frames an diese zuzustellen. In der einfachsten und gleichzeitig ineffizientesten Variante werden Multicasts in einzelne, durch die jeweiligen Empfänger bestätigte, Unicasts aufgebrochen. In zwei weiteren Varianten werden gruppenadressierte Frames entweder vorsorglich mehrfach versandt oder blockweise bestätigt. Zwar wurden erste prototypische GATS-Implementierungen vorgestellt (siehe [200, 201]), es existiert jedoch bislang keine Unterstützung durch herkömmliche Consumer-Hardware und Treiber.

Ohne GATS werden Multicast-Nachrichten weder auf der Sicherungs- noch Transportschicht bestätigt, sodass Mechanismen für eine zuverlässige Übertragung auf der Anwendungsschicht realisiert werden müssen. Darüber hinaus existieren mit üblicher WLAN-Hardware zusätzliche Einschränkungen hinsichtlich der nutzbaren Datenraten. In der Regel werden nur Unicast-Daten-Frames mit höheren Datenraten versendet, wie sie z. B. durch IEEE 802.11n (High Throughput (HT)) oder IEEE 802.11ac (Very High Throughput (VHT)) definiert sind. Der Versand von Broadcast-/Multicast- und jeglichen Management-Frames ist dagegen auf die niedrigeren *Basic Rates* der Spezifikationen IEEE 802.11a/b/g beschränkt, da typischerweise im Mixed-Modus operiert wird. Oft wird sogar nur die jeweils kleinste Datenrate (*Lowest Base Rate*) für diese Frame-Typen genutzt [198, 202].

Statt einer Multicast-Lösung wird daher in dieser Arbeit ein P2P-Ansatz auf Basis von BT verfolgt. Dieser realisiert die effiziente und zuverlässige Datenübertragung zwischen mehreren Teilnehmern

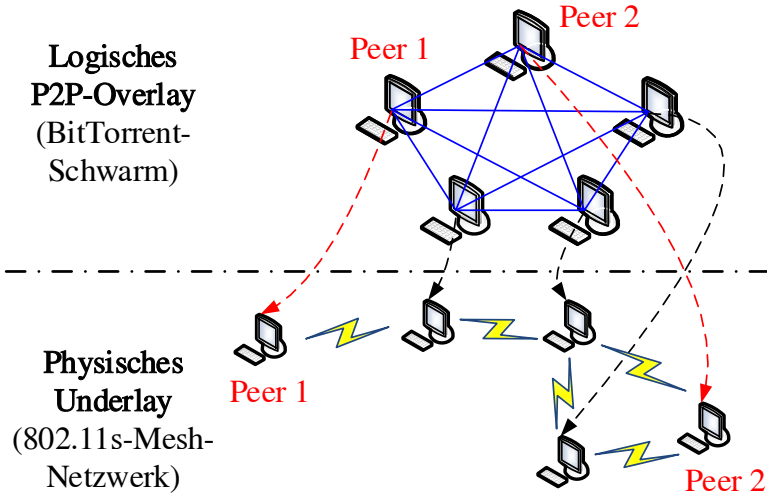


Abbildung 4.3: Mismatch zwischen P2P-Overlay und Mesh-Underlay [B 17]

(Peers) in einem logischen Overlay-Netzwerk auf der Anwendungsschicht. Da Peers in einem städtischen WMN als rein kollaborativ angesehen werden können – nicht zwangsläufig eine gültige Annahme in klassischen BT-Netzwerken – werden Anpassungen an BT und dessen Strategie zur Peer-Auswahl vorgenommen.

4.4.2 Optimierter Algorithmus zur Peer-Auswahl

Herkömmliche P2P-Protokolle wie BT berücksichtigen bei der Organisation ihres logischen Overlays nicht die Struktur des darunter liegenden physischen Netzwerks. Sie sind optimiert für kabelgebundene Netzwerke, die typischerweise durch Switches getrennte Kollisionsdomänen und eine geringe Paketfehlerrate aufweisen. In kabelgebundenen Netzwerken ergibt sich bereits durch die reine Aufgabenverteilung auf mehrere Peers eine erhöhte Skalierbarkeit und Leistungsfähigkeit einer Anwendung [176, 177]. Die Eignung von Peers als Kommunikationspartner ist dabei weniger von ihrer physischen Position im Netzwerk abhängig, sondern vielmehr von ihrem Verhalten auf der Anwendungsschicht. Im Fall von Standard-BT ist dies der Bereitschaftsgrad, Daten kollaborativ weiterzugeben. In WMNs sind die Kommunikationsbedingungen jedoch dynamischer und stark abhängig von der Link-Qualität, Kanalauslastung und Netzwerktopologie [175, 181]. Geräte in Funkreichweite und auf demselben Kanal bilden eine Kollisionsdomäne, in der sie um den exklusiven Zugriff auf das Medium konkurrieren. Bei der Kommunikation über mehrere Hops werden entsprechend viele Sendevorgänge zur Frame-Weiterleitung benötigt, die jeweils anfällig für Übertragungsfehler und Kollisionen sind.

Beim Einsatz von BT oberhalb eines Mesh-Underlays hat daher die Wahl der logischen Endpunkte einen großen Performance-Einfluss. Während auf P2P-Anwendungsschicht alle Peers als direkte Nachbarn wahrgenommen werden, können diese in der physischen Netzwerktopologie weit voneinander entfernt sein. In diesem Zusammenhang spricht man auch von einem *Mismatch* zwischen Overlay und Underlay [28, 30]. Abb. 4.3 zeigt ein vereinfachtes Beispielnetzwerk bestehend aus fünf Mesh-Knoten, die auf der Anwendungsschicht einen BT-Schwarm bilden. Zwar sehen sich Peer 1 und 2 im Schwarm als logische Nachbarn, sie besitzen in der Mesh-Topologie jedoch eine Distanz von mindestens drei Hops. Um das Mismatch-Problem zu lösen, muss die physische Netzwerkstruktur auch auf der Anwendungsebene Berücksichtigung finden. Physisch nahe BT-Peers sollten bevorzugt als Upload-Ziele gewählt werden, da auf diese Weise die Netzwerkbelastung lokal gehalten wird und gleichzeitig der kollaborative Datenaustausch gewährleistet bleibt. Die Wahl von Endpunkten

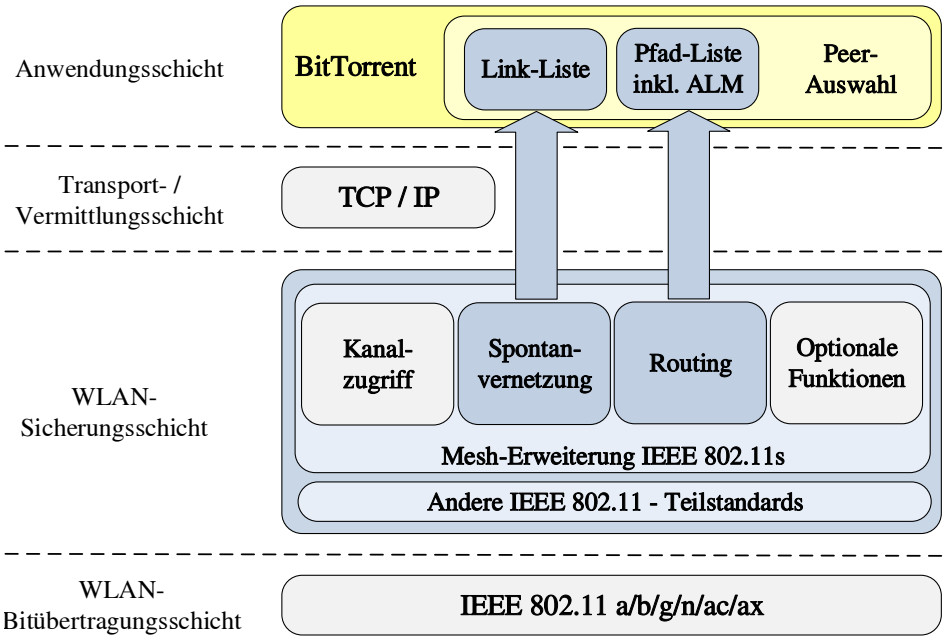


Abbildung 4.4: Cross-Layer-Optimierungsansatz *MeNTor* [B 4]

langer und ausgelasteter Mesh-Pfade sollte hingegen vermieden werden, da diese durch näher gelegene Knoten effizienter mit Pieces versorgt werden können.

Eine Unterscheidung von Peers nach diesen Gesichtspunkten kann auf Basis der in 802.11s spezifizierten Routing-Metrik ALM erfolgen, da diese sowohl die Länge als auch die Übertragungsgeschwindigkeit und Fehlerwahrscheinlichkeit von Mesh-Pfaden abbildet (siehe Kap. 2.5.5). Verglichen mit der Peer-Auswahl-Strategie von Standard-BT, welche Upload-Slots nach einem Belohnungsprinzip oder sogar zufällig vergibt, stellt die ALM ein besseres Bewertungskriterium für Peers in 802.11s-WMNs dar. Im Folgenden wird der in dieser Arbeit entwickelte Cross-Layer-Optimierungsansatz *MeNTor* beschrieben, der die ALM und weitere Informationen der 802.11s-Sicherungsschicht in die BT-Anwendungsschicht integriert.

Abb. 4.4 zeigt schematisch den Informationsfluss von MeNTor, wie er auf jedem Netzwerkteilnehmer erfolgt. Demnach werden die auf der WLAN-Sicherungsschicht verwalteten Datenstrukturen für die Direktverbindungen (Links) sowie HWMP-Routen eines Mesh-Knotens samt der ALM-Kosten zu Zielknoten an die BT-Applikation und deren Algorithmus zur Peer-Selektion gereicht. Auf Basis dieser Informationen wird die physische Mesh-Topologie im logischen Overlay berücksichtigt.

Wie in Kap. 4.2.2 beschrieben wird die BT-Peer-Auswahl durch den sogenannten *Choking-Algorithmus* realisiert. Dieser wird periodisch ausgeführt und vergibt eine vorgegebene Anzahl von Upload-Slots an interessierte Peers. Die Slot-Anzahl N setzt sich zusammen aus *regulären* und *optimistischen* Uploads, sodass $N = N_{reg} + N_{opt}$. Standard-BT vergibt $N_{reg} = 4$ reguläre Uploads nach einem Belohnungsprinzip und gewährt mit $N_{opt} = 1$ einem weiteren, zufällig gewählten Peer einen optimistischen Slot [178].

Listing 4.5 zeigt den Pseudo-Code des Choking-Algorithmus sowohl für Standard-BT als auch für den Cross-Layer-Optimierungsansatz MeNTor, der eine Sammlung von Teiloptimierungen umfasst. Innerhalb des Choking-Algorithmus wird unterschieden, ob sich der ausführende BT-Peer in der *Leecher*- oder *Seed*-Rolle befindet. Die in MeNTor vorgeschlagenen Teiloptimierungen kommen abhängig von der Konfiguration des Algorithmus zum Einsatz. Für Entwicklungs- und Testzwecke wurde dieser um zusätzliche Eingabeparameter erweitert, die eine individuelle Aktivierung der verschiedenen Optimierungen erlauben. Alle zu MeNTor gehörenden Erweiterungen sind im Pseudo-

```
1:  Inputs: InterestedPeers[*], N_reg, N_opt, GenerousMode ∈ {true, false}, Criterion ∈ {Default, ALM, ALM + NP}
2:  Results: Unchokes_reg[N_reg], Unchokes_opt[N_opt], Chokes[* − N_reg − N_opt]
3:  do every 30s:
4:      Unchokes_opt = getRandomPeers(InterestedPeers)
5:  end
6:  if (self == leecher) do every 10s else if (self == seed) do every 30s:
7:      if ((self == leecher) and GenerousMode): Candidates = getInterestingPeers(InterestedPeers)
8:      else: Candidates = InterestedPeers
9:      switch Criterion
10:         case Default
11:             if (self == leecher)
12:                 Unchokes_reg = getBestPeersRecentDownloadRate(Candidates)
13:                 fillRemaining(Unchokes_reg, getBestPeersLongTermDownloadUploadRatio(Candidates))
14:             else if (self == seed)
15:                 Unchokes_reg = getBestPeersUploadRateAndBytesSent(InterestedPeers)
16:             fillRemaining(Unchokes_reg, getRandomPeers(InterestedPeers))
17:         case ALM
18:             Unchokes_reg = getBestPeersALM(Candidates)
19:             fillRemaining(Unchokes_reg, getRandomPeers(InterestedPeers))
20:         case ALM + NP
21:             Unchokes_reg = getBestPeersNeighborALM(Candidates)
22:             fillRemaining(Unchokes_reg, getBestPeersALM(Candidates))
23:             fillRemaining(Unchokes_reg, getRandomPeers(InterestedPeers))
24:     Chokes = getDiffSet(InterestedPeers, Unchokes_reg, Unchokes_opt)
25:     notifyChokedUnchokedPeers(Chokes, Unchokes_reg, Unchokes_opt)
26: end
```

Listing 4.5: Pseudo-Code der BT-Peer-Auswahl inkl. *MeNTor*-Optimierungen [B 4]
(reg: regular, opt: optimistic, ALM: Airtime Link Metric, NP: Neighbor Preference,
alle Änderungen zu Standard-BT sind farbig hervorgehoben)

Code farblich hervorgehoben. Wie in Zeile 1 ersichtlich, werden dem Algorithmus die aktuelle Menge der interessierten Peers (Argument `InterestedPeers[]`) als Liste beliebiger Länge X sowie das konfigurierbare Bewertungskriterium für Peers (Argument `Criterion`) übergeben. Als Ergebnis eines Durchlaufs werden einige Peers *choked* (abgewürgt) und andere Peers als reguläre oder optimistische Upload-Ziele gewählt, d.h. *unchoked* (Zeilen 2 und 24–25).

In Standard-BT vergibt ein Leecher nur an solche interessierten Peers Upload-Slots, die gleichzeitig auch für ihn *interessant* sind, d.h. noch durch ihn benötigte Pieces anbieten. In MeNTor wird ein weiterer Eingabeparameter namens *Generous Mode* (großzügiger Modus) eingeführt, um dieses Verhalten wahlweise zu umgehen und die Upload-Vergabe nicht nur auf interessante Peers einzuschränken (siehe Zeile 7). Da ein Peer in der Seed-Rolle bereits alle Pieces besitzt und es daher aus seiner Sicht keine für ihn interessanten Peers gibt, berücksichtigt er stets alle interessierten Peers als potentielle Upload-Ziele (siehe Zeile 8).

Die optimistischen Upload-Slots werden stets zufällig gewählt und alle 30 s neu bestimmt (Zeilen 3–5). Die regulären Uploads hingegen werden in der Leecher-Rolle alle 10 s und in der

Seed-Rolle alle 30 s aktualisiert (Zeile 6). In MeNTor werden alle Zeitintervalle von Standard-BT beibehalten. Laut BT-Erfinder Bram Cohen wird von der Wahl kleinerer Intervalle als 10 s abgeraten, da eine Mindestzeit für das Wachstum des effektiven Sendefensters der TCP-Verbindungen für die Piece-Übertragungen benötigt wird [178].

Die unmodifizierte Peer-Auswahl-Strategie von Standard-BT ist in MeNTor weiterhin enthalten und per Konfiguration aktivierbar. Sie dient zum einen als Referenzfall zur Bewertung der neu eingeführten Optimierungen für Mesh-Netzwerke. Zum anderen ist sie eine potentielle Fallback-Strategie für kabelgebundene Teilnehmer in heterogenen Netzwerkszenarien, die z. B. eine Mischung aus WLAN-Mesh- und Ethernet-Infrastruktur darstellen.

Wird das Belohnungsprinzip von Standard-BT genutzt (Kriterium *Default*), vergibt ein Leecher reguläre Uploads an anfragende Peers basierend auf der im Gegenzug gewährten Upload-Rate (Zeile 12) sowie dem Langzeitverhältnis aus empfangener zu gesendeter Datenmenge (Zeile 13). In praktischen Implementierungen werden zumeist nur Peers berücksichtigt, deren Datenrate oberhalb eines gewissen Minimalwertes liegt [183]. Ein Seed wiederum ermittelt seine regulären Uploads, indem er Leecher nach seiner Upload-Performance zu ihnen bewertet, basierend auf der momentanen Upload-Datenrate und der Statistik der bereits übertragenen Datenmenge (Zeile 15). Direkt nach dem Schwarmbeitritt eines Peers besitzt dieser unter Umständen noch nicht genügend Informationen über andere Peers, um seine N_{reg} regulären Uploads nach einer Performance-Statistik zu vergeben. Sind ausreichend viele interessierte Peers vorhanden, werden daher ungenutzte reguläre Slots wie optimistische Uploads behandelt und mit zufälligen Peers aufgefüllt (Zeile 16).

MeNTor umfasst drei Teiloptimierungen: ein neues *Kriterium zur Peer-Auswahl* in 802.11s-Mesh-Netzwerken, eine veränderte *maximale Anzahl gleichzeitiger Uploads* sowie eine *nichtrestriktive Upload-Strategie* für Leecher namens *Generous Mode*. In der Funktion sowohl einer prototypischen Implementierung als auch eines Test-Frameworks erlaubt MeNTor die individuelle Aktivierung aller Teiloptimierungen unabhängig voneinander. Nachfolgend werden diese im Detail erläutert und es werden Erwartungshaltungen für Designentscheidungen diskutiert.

4.4.2.1 Kriterium zur Peer-Auswahl

Die Wichtigste der durch MeNTor bereitgestellten Optimierungen ist die Cross-Layer-Integration von Informationen der WLAN-Sicherungsschicht in den Choking-Algorithmus von BT. Neben dem zuvor beschriebenen Kriterium *Default*, das die herkömmliche Belohnungsstrategie von Standard-BT realisiert, werden zwei neue Kriterien für den Einsatz in 802.11s-Netzwerken eingeführt: *ALM* und *ALM mit Neighbor Preference (NP)*. Beide Varianten stützen sich allein auf lokale Informationen, die auf jedem Mesh-Knoten inhärent verfügbar sind. Sie erfordern weder zusätzlichen Datenverkehr noch eine Modifikation der WLAN-Sicherungsschicht.

Wird die 802.11s-Routing-Metrik *ALM* als Kriterium genutzt (Listing 4.5, Zeilen 17–19), ergibt sich die Priorität von BT-Peers direkt aus den Mesh-Pfadkosten zu ihnen. Die ALM schätzt den Zeitbedarf zur Frame-Übertragung an einen Zielknoten und berücksichtigt die Parameter der WLAN-Technologie sowie die momentane Datenrate und Fehlerwahrscheinlichkeit der Verbindung (siehe Kap. 2.5.5, Gl. 1). Durch den kumulativen Charakter der Metrik bildet diese indirekt die Länge eines Pfades ab, erlaubt jedoch gleichzeitig die Berücksichtigung der genannten QoS-Eigenschaften. Damit bietet die Nutzung der ALM als Unchoke-Kriterium sowohl das Potential zur Bevorzugung physisch naher Peers wie auch zur Wahl günstiger längerer Pfade. Insbesondere in ausgedehnten Mesh-Topologien mit vielen Multi-Hop-Pfaden kann so der Datenverkehr überwiegend lokal gehalten und der Overhead durch unnötige Weiterleitungsschritte reduziert werden.

In dicht besetzten Topologien mit hohem Kommunikationsaufkommen besteht jedoch die Gefahr, dass sich die Konvergenzzeit des Mesh-Routing-Protokolls aufgrund einer steigenden Fehlerrate

oder Rückstellung von Routing-Nachrichten erhöht und somit die Aktualität der ALM-Informationen beeinträchtigt ist [203–205]. Hier kann es erforderlich sein, die robuste Wahl physisch naher Peers durch Nutzung weiterer Informationen zu garantieren. Die auf der 802.11s-Sicherungsschicht verwalteten Datenstrukturen erlauben die Unterscheidung von direkt benachbarten und weiter entfernten Mesh-Knoten. In der *Link-Liste* jedes Knotens werden ausschließlich die Verbindungen zu seinen Nachbarn geführt. Der Link-Status wird beim Empfang von Beacon-Frames aktualisiert, die in der Regel sekündlich durch alle Knoten gesendet werden [118, 127]. In der *Pfad-Liste* wiederum finden sich die HWMP-Routing-Informationen, welche für jeden Zielknoten den zuständigen Weiterleitungsknoten (Next Hop) und die resultierende Ende-zu-Ende-ALM enthalten. Auch die Verbindungen zu Nachbarknoten sind hier als Single-Hop-Pfade aufgeführt. Der Nachbarknoten ist im jeweiligen Pfadeintrag sowohl Zielknoten als auch Next Hop. Die Ende-zu-Ende-ALM entspricht direkt der auf dem lokalen Knoten berechneten Metrik des Links. Die nach Bedarf ermittelten Pfade besitzen in der Praxis eine Gültigkeit von 5 s und werden stets 1 s vor Auslaufen ihres Timeouts durch Kontrollnachrichten aufgefrischt [118, 127]. Im Vergleich zu den Kontrollnachrichten des Routing-Protokolls HWMP werden Beacons in kürzeren Zeitabständen gesendet und niemals weitergeleitet. Die rechtzeitige Aktualisierung der Link-Informationen besitzt damit vor allem auch in Lastsituationen eine höhere Erfolgschance als die Verbreitung neuer Pfad-Informationen über mehrere Hops. Aus diesem Grund wird eine zweiten Variante der optimierten Peer-Auswahl untersucht, die als Kriterium die *ALM mit Neighbor Preference (NP)* nutzt (Listing 4.5, Zeilen 20–23). Diese Kombination betrachtet zunächst nur die Peers in direkter Funkreichweite und sortiert diese aufsteigend nach ALM. Im Ergebnis werden benachbarte Peers immer gegenüber solchen im Multi-Hop-Abstand bevorzugt, selbst wenn die längeren Pfade geringere ALM-Kosten aufweisen. Stehen nicht genügend direkte Nachbarn zur Besetzung der konfigurierten Menge regulärer Upload-Slots zur Verfügung, werden nachfolgend auch Peers im Multi-Hop-Abstand berücksichtigt (Listing 4.5, Zeile 22).

Neben dem in dieser Arbeit diskutierten Fall, in dem das BT-Netzwerk auf eine reine Mesh-Installation beschränkt ist, sind in der Praxis auch heterogene Szenarien denkbar, in denen ebenfalls Peers aus anschließender Ethernet-Infrastruktur oder dem Internet am Overlay teilnehmen. Hier könnte auf den Mesh-Knoten die ursprüngliche Peer-Auswahl nach dem Kriterium *Default* als potentielle Fallback-Lösung für die Bewertung der kabelgebundenen Peers genutzt werden. Da zu diesen nur ALM-Informationen bis zu einem Mesh-Gateway vorhanden sind, können sie klar von den Knoten innerhalb des Mesh-Netzwerks unterschieden werden. Um eine unnötige Belastung des Mesh-Netzwerks durch Kommunikation in die externe Infrastruktur zu vermeiden, sollten weiterhin physisch nahe Peers bevorzugt werden, die gleiche Pieces bereits im Mesh-Netzwerk anbieten. Externe Peers, die Quellen exklusiver Pieces darstellen, könnten zudem abhängig von den ALM-Kosten zum Gateway priorisiert werden, über die sie an das Mesh-Netzwerk angebunden sind.

4.4.2.2 Anzahl und Konstellation der Upload-Slots

Als weitere Optimierungsparameter werden die Anzahl regulärer und optimistischer Upload-Slots, N_{reg} und N_{opt}, betrachtet (Listing 4.5, Zeile 2). Die für Standard-BT empfohlene Konfiguration wurde von Bram Cohen mit $N_{reg} = 4$ und $N_{opt} = 1$ angegeben [178]. Diese wird als erste Upload-Slot-Konstellation untersucht. Um zu analysieren, wie sich die Deaktivierung der zufälligen Peer-Auswahl und die Beschränkung gleichzeitiger Uploads in einem Mesh-Netzwerk auswirken, werden im Rahmen dieser Arbeit neben der Standardkonfiguration zudem zwei weitere Konstellationen definiert (siehe Kap. 4.6 für die Übersicht aller untersuchten MeNTor-Konfigurationen und Testfälle).

Die zweite Upload-Slot-Konstellation stellt eine Aufteilung dar, bei der die fünf Slots von Standard-BT ausschließlich nach dem Peer-Auswahl-Kriterium vergeben werden ($N_{reg} = 5$, $N_{opt} = 0$). Sie soll den direkten Performance-Unterschied aufzeigen, der durch die Vermeidung der Auswahl möglicherweise ungünstiger, zufällig gewählter Peers entsteht. Insbesondere in Verbindung mit der ALM-basierten Peer-Auswahl wird somit der Upload zu physisch nahen Peers erzwungen. In dieser Konstel-

lation werden wie in Standard-BT pro Peer fünf quasi-parallele Datenübertragungen zu anderen Peers zugelassen. Diese werden an der WLAN-Schnittstelle serialisiert und teilen sich das Zeitintervall bis zur nächsten Ausführung des Choking-Algorithmus. Innerhalb einer Kollisionsdomäne, d.h. im Einzugsbereich jedes Mesh-Knotens, herrscht konkurrierender Medienzugriff und Sendevorgänge erfolgen exklusiv. Darüber hinaus beeinflussen Uploads zu Peers im Multi-Hop-Abstand alle auf dem Weg liegenden Kollisionsdomänen.

Hingegen ist eine Effizienzsteigerung zu erwarten, wenn jeder Peer lediglich ein nahes Ziel, dieses jedoch mit einer größeren Datenmenge pro Unchoke-Intervall versorgt. Dadurch kann eine höhere Zahl unabhängiger Kommunikationspaare entstehen, die sich nicht im Kanalzugriff beeinflussen. Insbesondere in dicht besetzten Mesh-Topologien sollte die Reduzierung konkurrierender Sendevorgänge den kollaborativen Datenaustausch begünstigen. Daher wird eine dritte Upload-Slot-Konstellation untersucht, welche die regulären Uploads auf lediglich einen Slot beschränkt ($N_{reg} = 1$, $N_{opt} = 0$). Folglich wird stets nur der günstigste Peer laut Auswahlkriterium bedient.

4.4.2.3 Nichtrestriktive Upload-Strategie (Generous Mode)

In Standard-BT verfolgen Leecher eine sogenannte *Anti-Snubbing*-Strategie (engl. *snub* = *abblitzen lassen*). Diese verhindert, dass Peers Upload erhalten, die über einen Zeitraum von einer Minute keinen Upload im Gegenzug gewährt haben. In drahtgebundenen Netzwerken und im Internet, für die Standard-BT optimiert ist, deuten derartige Upload-Lücken meist auf ein unfaires Verhalten von Peers hin. In drahtlosen Netzwerken und insbesondere WMNs mit gänzlich kooperativen Knoten liegen die Gründe jedoch vielmehr in zufälligen Übertragungsfehlern, die zu aufeinander folgenden TCP-Neuübertragungen und sporadischen Timeouts führen können. Die Entscheidung, in so einem Fall ansonsten geeignete Peers zu benachteiligen, könnte daher eine ungewollte Reaktion sein.

Ein zweiter Aspekt des Choking-Algorithmus in Standard-BT sieht vor, dass nur solche Peers Upload erhalten, die selbst fehlende Pieces anbieten und dadurch aus Sicht des lokalen Knotens ebenfalls *interessant* sind. Dabei kann es vorkommen, dass die Anzahl der interessierten Peers nach Filterung geringer ist als die Anzahl der verfügbaren regulären Upload-Slots. Damit diese dennoch besetzt werden, füllt ein Leecher übrige Slots mit zufälligen Peers auf. Im Hinblick auf die kollaborative Datenverteilung in WMNs stellt die Filterung der Peers eine Limitierung dar, die eine Wahl physisch naher Knoten als günstige Upload-Ziele potentiell unterbindet.

Im Rahmen von MeNTor wird mit dem sogenannten *Generous Mode* (engl. *generous* = *großzügig*) eine nichtrestriktive Upload-Strategie eingeführt. Demnach betrachtet ein Leecher alle interessierten Peers als geeignete Upload-Kandidaten, unabhängig von deren Interessantheit und Snubbing-Status (Listing 4.5, Zeile 7). Dieses Verhalten entspricht dem eines herkömmlichen BT-Seeds, der ebenfalls keine Filterung der interessierten Peers vornimmt (Listing 4.5, Zeile 8). Der *Generous Mode* impliziert besondere Vorteile in Overlays mit wenigen oder sogar nur einem initial vorhandenen Seed, der Daten an bestimmte Knoten im Mesh-Netzwerk verteilen will. So könnte BT in einem administrativen Szenario beispielsweise der kollaborativen Ausbringung von Software-Updates an ausgewählte Gerätegruppen eines städtischen drahtlosen Zugangsnetzes oder Backbones dienen.

Zu Beginn durch den initialen Seed bediente Leecher kennen selbst noch keine für sie interessanten Peers, sodass nach der herkömmlichen, restriktive Strategie von Standard-BT nur eine zufallsbasierte Weitergabe bereits empfangener Pieces an möglicherweise ungünstige Peers erfolgen kann. Werden die mit MeNTor eingeführten, ALM-basierten Peer-Auswahl-Varianten genutzt und zudem optimistische Upload-Slots deaktiviert, erhalten vorwiegend physisch nahe Peers die Gelegenheit zum Download. Wird jedoch auch in dieser Kombination eine Beschränkung der regulären Kandidaten auf interessante Peers vorgenommen, müssen Leecher im Umkreis des initialen Seeds ihren Download zunächst vollständig abschließen und selbst in die Seed-Rolle wechseln. Erst dann versorgen sie andere (uninteressante) Peers nicht mehr nach dem Zufallsprinzip, sondern unter Berücksichti-

gung der ALM-Kosten. Der *Generous Mode* hebt die genannten Limitierungen auf und verfolgt das Ziel, sofort einen effizienten, ALM-basierten Verteilungsprozess zu ermöglichen. Dennoch werden die durch MeNTor eingeführten Peer-Auswahl-Varianten auch mit deaktiviertem *Generous Mode* untersucht. Dazu werden wie bei Standard-BT aufgrund der Filterung unbesetzte Upload-Slots mit zufälligen Peers aufgefüllt (Listing 4.5, Zeilen 19 und 23).

4.5 Prototypische Umsetzung

Bei der Open-Source-Software *Vuze* (ehemals *Azureus*) [183] handelt es sich um einen populären BT-Client, der in Java entwickelt und plattformübergreifend für die Betriebssysteme Windows, MacOS und Linux erhältlich ist. Vuze bietet zudem eine Schnittstelle (Application Programming Interface (API)) für Plugins, durch welche das Hauptprogramm flexibel modifiziert und erweitert werden kann. Die prototypische Implementierung von MeNTor wurde als Plugin für die Vuze-Programmversion 4.3 realisiert. Als Grundgerüst für den Quellcode diente das externe Open-Source-Projekt *BASS* [206]. Es verfolgte das langfristige Ziel, die BT-Peer-Auswahl durch manuelle Prioritätenvergabe für Peers aus unterschiedlichen IP-Subnetzen beeinflussen zu können. Das Projekt wurde in einem sehr frühen Entwicklungsstadium eingestellt, bot jedoch bereits ein Plugin-Grundgerüst, um die Vuze-interne Implementierung des BT-Choking-Algorithmus zu verändern. In der erhältlichen Vorversion realisierte das Plugin lediglich eine rudimentäre Logging-Funktionalität zur Erstellung von Upload- und Download-Statistiken von Peers. Im Rahmen dieser Arbeit wurde das bestehende Code-Skelett um den in Kap. 4.4.2, Listing 4.5 als Pseudo-Code gegebenen Choking-Algorithmus MeNTor erweitert.

Abb. 4.6 zeigt die vereinfachte Architektur des entstandenen Software-Prototyps. Der BT-Client Vuze, bestehend aus dem Hauptprogramm (Vuze Core) und der Plugin-Erweiterung, wird auf einem Linux-Betriebssystem mit 802.11s-Unterstützung ausgeführt (vgl. Kap. 2.5.7). Bei Programmstart ersetzt das Plugin die ursprüngliche Implementierung der Java-Klasse `Unchoker` des Hauptprogramms, welche den Choking-Algorithmus realisiert. Abhängig vom aktuellen Download-Fortschritt im BT-Schwarm wechselt der lokale Peer automatisch zwischen den Implementierungsvarianten `DownloadingUnchoker` und `SeedingUnchoker` dieser Klasse. Dies entspricht der Fallunterscheidung zwischen den Peer-Rollen *Leecher* und *Seed* im Pseudo-Code in Kap. 4.4.2, Listing 4.5. Zu Entwicklungs- und Testzwecken können die einzelnen MeNTor-Optimierungen individuell über eine Konfigurationsdatei aktiviert werden, welche bei Programmstart durch die Klasse `ConfFileParams` des Plugins eingelesen wird. Die Parameter umfassen das Peer-Auswahlkriterium (Default, ALM oder ALM mit Nachbar-Präferenz), den Generous Mode sowie die Anzahl der regulären und optimistischen Upload-Slots. Die in der praktischen Auswertung genutzten Konfigurationsvarianten werden in Kap. 4.6 im Detail erläutert.

Der Aufruf des Choking-Algorithmus der Klasse `Unchoker` erfolgt zyklisch durch die Klasse `PeerControl` des Hauptprogramms. Diese ist Teil der Komponente `PeerManager`, welche die Verbindungen zu anderen BT-Peers und den Datenaustausch mit diesen verwaltet. Das Unchoke-Timer-Intervall entspricht den in Kap. 4.2.2 beschriebenen Standardwerten von BT, die sich in der Leecher-Rolle für reguläre und optimistische Upload-Slots mit 10 s bzw. 30 s unterscheiden. Demgegenüber werden in der Seed-Rolle beide Upload-Slot-Varianten im 30 s-Intervall aktualisiert.

Der Zugriff auf die Mesh-Informationen des lokalen Knotens erfolgt aus dem Plugin heraus über bereits bestehende Schnittstellen des Betriebssystems. Die 802.11s-Implementierung des Linux-Kernelmoduls `mac80211` verwaltet Datenstrukturen für die Mesh-Links und HWMP-Pfade, welche über das Linux-Kommandozeilenprogramm `iw` ausgelesen werden können [128]. Die *Link-Liste*, abrufbar über das `iw`-Kommando `station dump`, enthält die Statusinformationen aller Direktverbindungen zu Nachbarknoten. Die *Pfad-Liste* wird über das Kommando `mpath dump` ausgelesen und enthält die Weiterleitungsregeln der Pfade zu verschiedenen Zielknoten. Jede Regel besteht

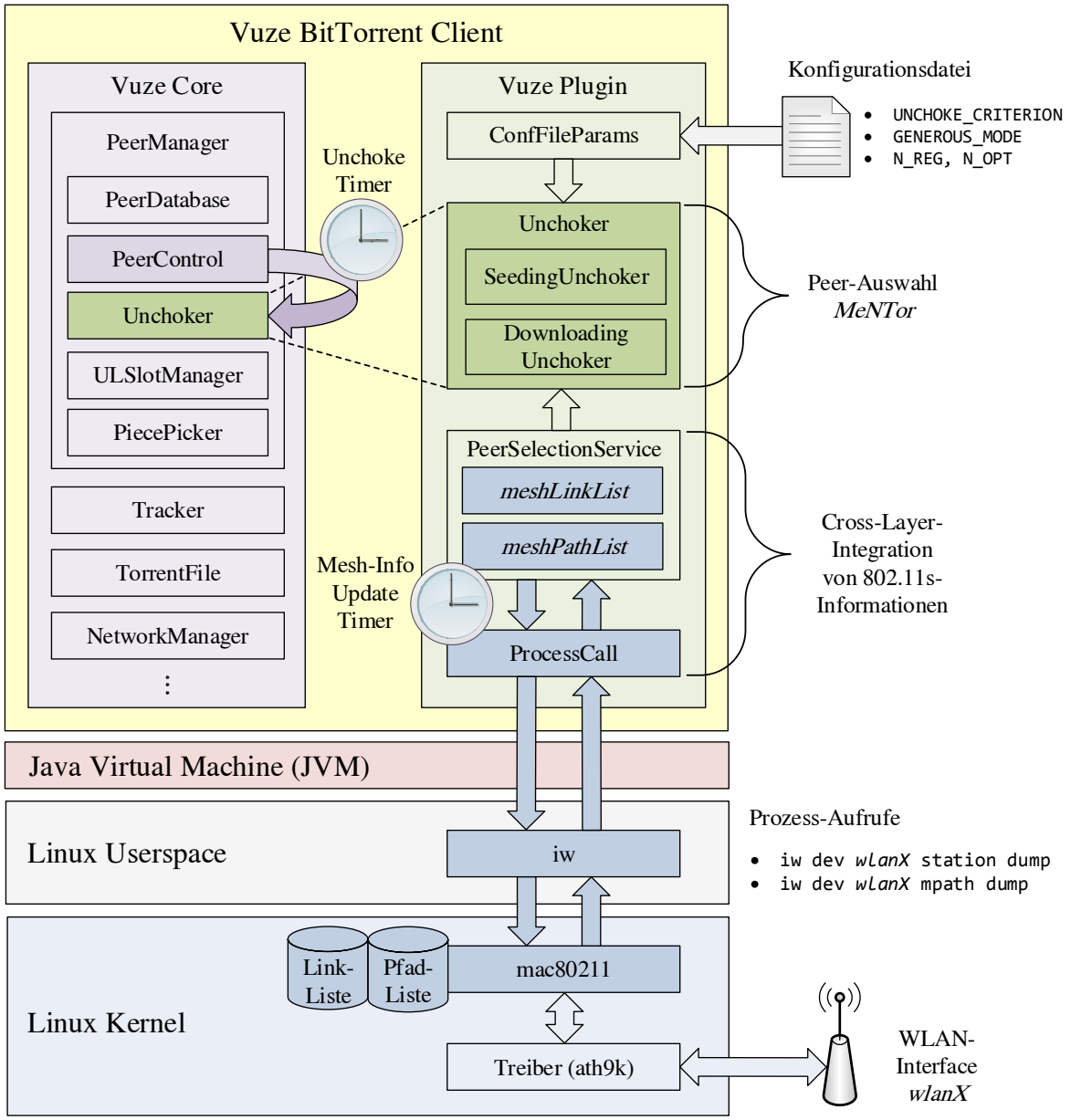


Abbildung 4.6: Software-Architektur der *MeNTor*-Implementierung

aus den MAC-Adressen des Ziel- und ersten Zwischenknotens sowie den resultierenden Pfadkosten in Form der ALM. Unter Beibehaltung der Standardwerte der Linux-Mesh-Parameter (siehe Kap. 2.5.7) werden die Link-Informationen sekündlich und die Pfad-Informationen alle 4–5 Sekunden auf Kernel-Ebene aktualisiert [118, 127].

In der Klasse `ProcessCall` des Vuze-Plugins wird `iw` über Prozessaufrufe basierend auf der `ProcessBuilder`-Funktionalität [207] der Java-Sprachbibliothek gekapselt. Hier erfolgt die Verknüpfung des ansonsten plattformunabhängigen Plugins mit der plattformabhängigen 802.11s-Implementierung unter Linux. Für mögliche künftige 802.11s-Implementierungen anderer Betriebssysteme wäre somit eine Anpassung der Klasse `ProcessCall` erforderlich. Die `iw`-Textausgaben der Link- und Pfadlisten werden an dieser Stelle eingelesen und in Java-Listenobjekte überführt. Die Prozessaufrufe werden wiederum von der Klasse `PeerSelectionService` des Plugins veranlasst, welche die aktuellen Listenobjekte verwaltet und der Klasse `Unchoker` bereitstellt. Wie der Choking-Algorithmus erfolgt auch der Abruf der Mesh-Informationen zyklisch, jedoch auf Basis

eines zeitlich versetzten Timers. Dadurch stehen die aktuellen Informationen jeweils schon vor der nächsten Ausführung des Choking-Algorithmus zur Verfügung.

Der Java-Quellcode des erweiterten Vuze-Plugins [4.1] und die im Rahmen der praktischen Evaluation zur Experimentautomatisierung entwickelten Shell-Skripte [4.2] sind frei verfügbar. Die erzielten Ergebnisse werden nachfolgend im Kap. 4.6 erläutert.

4.6 Evaluation in der Testumgebung

Um die Auswirkungen der MeNTor-Optimierungen sowohl individuell als auch im Zusammenspiel zu untersuchen, wurden umfangreiche Experimente in der Testumgebung *Mini-Mesh* (siehe Kap. 3) durchgeführt. Kap. 4.6.2 beschreibt die einzelnen Testfälle im Detail. Diese orientieren sich am Szenario eines WLAN-Mesh-Backbones in einer Smart-City, in dem BT als Protokoll zur Datensynchronisierung eingesetzt wird. Im betrachteten Kommunikationsmuster erfolgt ausgehend von einem *initialen Seed* die kollaborative Verteilung eines Datensatzes an mehrere interessierte *Leecher*. In der Praxis entspricht der Seed beispielsweise einem administrativen Knoten, der Software-Updates ausrollt oder gesammelte Statusinformationen mit anderen Management-Instanzen abgleicht.

Die gewählten Testfälle beinhalten die Worst-Case- und Best-Case-Position des initialen Seeds hinsichtlich der mittleren Pfadlänge zu einer bestimmten Anzahl und Anordnung von Leechern. Oberhalb einer statischen Mesh-Topologie wurden insgesamt 10 verschiedene Seed-/Leecher-Platzierungen untersucht. In jeder Platzierung kamen 18 Konfigurationsvarianten des BT-Choking-Algorithmus zum Einsatz, welche die verschiedenen Kombinationen der MeNTor-Optimierungen darstellen. Als Bewertungsmaß wurde der durchschnittliche Download-Zeitbedarf pro Peer und für den kompletten BT-Schwarm ermittelt.

4.6.1 Gerätekonfiguration

Alle Experimente wurden innerhalb der in dieser Arbeit entwickelten miniaturisierten Testumgebung *Mini-Mesh* durchgeführt. Durch Dämpfungsglieder im Antennenweg und eine verringerte Sendeleistung besitzt diese eine stark beschränkte WLAN-Übertragungsreichweite und ermöglicht es, realistische Multi-Hop-Szenarien in einem Laboraufbau im Maßstab von ca. $1:560$ nachzubilden [6]. Durch den Skalierungsansatz entspricht die aktuelle Gitteranordnung einer theoretischen Ausdehnung von ca. $300.000\,m^2$ auf einer Laborfläche von lediglich $1\,m^2$. Im Szenario einer Smart City korrespondiert dies z. B. mit einer Mesh-Backbone-Installation auf Hausdächern und Laternen, die mehrere Straßenzüge umfasst. Es wurden 25 der insgesamt 36 verfügbaren Knoten genutzt, da eine 5x5-Knoten-Anordnung im Gegensatz zu einem 6x6-Gitter eine symmetrisch zentrale Position im Zentrum aufweist. Diese konnte als Best-Case-Platzierung des initialen Seeds einer Worst-Case-Position in der Gitterecke gegenübergestellt werden.

Durch Nutzung freier Kanäle im oberen 5 GHz-Frequenzband, Festsetzen der Sendeleistung und -datenrate sowie treiberseitige Optimierungen bietet Mini-Mesh kontrollierte Kanalbedingungen und Verbindungseigenschaften. Die Sendedatenraten aller WLAN-Nachrichtentypen wurden auf das 802.11n Modulation and Coding Scheme (MCS) 3 bzw. dessen korrespondierende 802.11a-Datenrate konfiguriert. Bei der Kanalbreite von 20 MHz ergab sich so eine feste Bruttodatenrate von $\sim$26 Mbit/s für Unicast-Daten-Frames bzw. 24 Mbit/s für Multicast-Daten- und Management-Frames [109]. Anstatt externer Einflüsse dominierten das auf der Anwendungsschicht verursachte Kommunikationsaufkommen der Knoten und deren Konkurrenz im Medienzugriff die Performance

[4.1] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/mentor-vuze-plugin`
[4.2] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/mentor-scripts`

Tabelle 4.3: WLAN-Parameter der Testumgebung [B 4]

Parameter	**Wert**
Kanal	149 ($f_c = 5745$ MHz)
Kanalbandbreite	20 MHz
Guard Interval (GI)	800 ns (Long GI)
802.11a-Datenrate	24 Mbit/s (16-QAM, FEC 1/2)
802.11n-Datenrate	26 Mbit/s (HT MCS 3)

des Netzwerks. In der Testumgebung beobachtete Veränderungen des mittleren Zeitbedarfs für den BT-Datenaustausch konnten somit vorwiegend auf algorithmische Anpassungen der Peer-Auswahl zurückgeführt werden. Tabelle 4.3 fasst die wichtigsten WLAN-Konfigurationsparameter der Testumgebung zusammen. Ferner galten die Linux-Standardeinstellungen für 802.11s und TCP. Eine Übersicht der Mesh-Parameter des Linux-Kernels ist in Kap. 2.5 zu finden. Für weitere Details zu den Hard- und Software-Eigenschaften der Testumgebung wird auf Kap. 3 verwiesen.

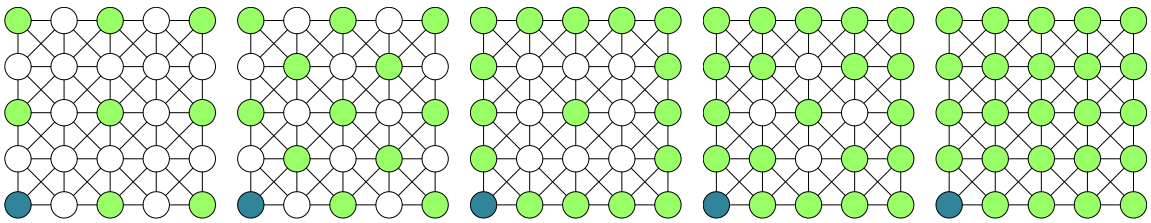
4.6.2 Netzwerkszenarien und Testfälle

Untersucht wird ein kollaboratives Kommunikationsmuster, bei dem eine bestimmte Menge von Peers (Leechern) am gleichen, von einem initialen Seed in den BT-Schwarm eingebrachten Datensatz interessiert ist. Der initiale Seed steht stellvertretend für den Zugangsknoten eines Netzwerkbetreibers oder Dienste-Anbieters beispielsweise in einer städtischen WLAN-Mesh-Installation. Bei den Daten handelt es sich in der Realität z. B. um Software-Updates oder Medieninhalte, die an eine Gruppe von Geräten verteilt werden. In den Experimenten wird ein fester Beispiel-Datensatz genutzt, bestehend aus zuvor erzeugten Zufallsdaten der festen Größe von 50 MB (siehe Kap. 4.6.4.1 für eine Diskussion zur Wahl der Datensatz- und Piece-Größe).

Es werden zwei übergeordnete Szenarien unterschieden, die den Einfluss der Position des initialen Seeds in der Mesh-Gitteranordnung demonstrieren. Dieser ist entweder in der unteren linken Ecke (Szenario 1, siehe Abb. 4.7) oder direkt in der Mitte des Gitters platziert (Szenario 2, siehe Abb. 4.8). Hinsichtlich der mittleren Hop-Distanz der Leecher (grüne Knoten) zum initialen Seed (blauer Knoten) stellt Szenario 1 den Worst Case und Szenario 2 den Best Case dar. Um gleichzeitig den Einfluss der Schwarmgröße und Position der Leecher zu untersuchen, sind beide Szenarien in jeweils 5 Testfälle gegliedert. Neben dem initialen Seed besteht der Schwarm somit aus 8, 12, 16, 20, oder 24 Leechern. Je nach Testfall wird die Implementierung von MeNTor somit auf einem bestimmten Teil der Mesh-Knoten des Testbeds ausgeführt. Die am Schwarm unbeteiligten Geräte (weiße Knoten) dienen ausschließlich als Mesh-Infrastruktur zur Datenweiterleitung.

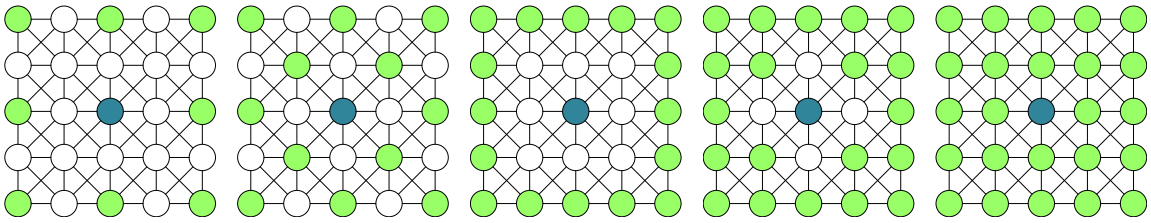
4.6.3 *MeNTor*-Konfigurationsvarianten

Wie in Kap. 4.5 beschrieben handelt es sich bei der Implementierung von MeNTor um ein konfigurierbares Plugin für den Java-basierten BT-Client *Vuze*. In jedem *Testfall (TF)* werden 18 *Konfigurationsvarianten (KV)* von MeNTor gegenübergestellt. Diese sind in Tab. 4.4 aufgeführt. Die KV sind zunächst nach dem *Peer-Auswahl-Kriterium* sortiert: dem Belohnungsprinzip von Standard-BT (KV 1–6), der Priorisierung nach ALM (KV 7–12) und der Priorisierung nach ALM mit zusätzlicher *Neighbor Preference (NP)* (KV 13–18). Jede KV kombiniert eines der drei Kriterien mit



(a) 8 Leecher (b) 12 Leecher (c) 16 Leecher (d) 20 Leecher (e) 24 Leecher

Abbildung 4.7: Testfälle für Szenario 1: initialer Seed & Tracker in der Ecke des Gitters [B 4]



(a) 8 Leecher (b) 12 Leecher (c) 16 Leecher (d) 20 Leecher (e) 24 Leecher

Abbildung 4.8: Testfälle für Szenario 2: initialer Seed & Tracker im Zentrum des Gitters [B 4]

einer von drei *Upload-Slot-Konstellationen* sowie der optionalen, nichtrestriktiven Upload-Strategie *Generous Mode*. Die in Kap. 4.4.2.2 beschriebenen Konstellationen für die Anzahl der regulären und optimistischen Upload-Slots umfassen dabei zum einen die Aufteilung von Standard-BT ($N_{reg} = 4$ und $N_{opt} = 1$). Zum anderen werden die Deaktivierung der zufälligen Peer-Auswahl ($N_{reg} = 5$ und $N_{opt} = 0$) sowie die zusätzliche Beschränkung auf einen exklusiven Upload pro Peer ($N_{reg} = 1$ und $N_{opt} = 0$) untersucht. Dabei repräsentiert KV 1 das unmodifizierte Verhalten von Standard-BT und dient als Referenz zur Bewertung der MeNTor-Optimierungen. Die letzte Konfiguration, KV 18, kombiniert alle Optimierungen mit dem Ziel, den BT-Datenverkehr in 802.11s-WMN lokal zu halten und die Kommunikations-Performance zu erhöhen.

4.6.4 Experimentvorbereitung und -ablauf

Mit der Untersuchung von 18 MeNTor-Konfigurationsvarianten in 10 Testfällen wurden insgesamt 180 verschiedene Experimente betrachtet. Jedes Experiment wurde 5 mal wiederholt, sodass insgesamt 900 Messungen durchgeführt wurden. Zur Bewertung der BT-Performance wurde die mittlere Download-Zeit bestimmt. Mit dem Begriff *Download-Zeit* wird hier die Differenz der Zeitstempel zwischen dem Upload-Beginn des initialen Seeds und dem Empfang des letzten Pieces auf einem Peer (Leecher) bezeichnet. Es wurden zwei verschiedene Zeiten ermittelt: die *mittlere Download-Zeit pro Peer* und die *mittlere Download-Zeit des gesamten Schwarms*. Die Schwarm-Download-Zeit ergab sich direkt aus der Download-Zeit desjenigen Peers, der zuletzt alle Pieces erhielt.

4.6.4.1 Wahl der Datensatz- und Piece-Größe

Vor der Durchführung der Experimente musste eine geeignete Größe für den auszutauschenden Datensatz definiert werden. Einerseits sollte diese nicht zu hoch gewählt werden, da die zu übertragende Datenmenge maßgeblich die Zeitdauer des Experiments bedingt. Andererseits galt es jedoch, eine gewisse Anzahl an Ausführungen des Choking-Algorithmus zu garantieren. Im Fall des Kriteriums *Default* (KV 1–6 in Tab. 4.4) beginnt der Datenaustausch zwangsläufig zufallsbehaftet. Im

Tabelle 4.4: Übersicht der *MeNTor*-Konfigurationen [B 4]
(KV: Konfigurationsvariante, ALM: Airtime Link Metric, NP: Neighbor Preference,
N_{reg}: Anzahl regulärer Uploads, N_{opt}: Anzahl optimistischer Uploads)

KV-Nr.	Auswahl-Kriterium	Upload-Slot-Konstellation	Generous Mode
1	Default	$N_{reg} = 4, N_{opt} = 1$	false
2		$N_{reg} = 5, N_{opt} = 0$	
3		$N_{reg} = 1, N_{opt} = 0$	
4		$N_{reg} = 4, N_{opt} = 1$	true
5		$N_{reg} = 5, N_{opt} = 0$	
6		$N_{reg} = 1, N_{opt} = 0$	
7	ALM	$N_{reg} = 4, N_{opt} = 1$	false
8		$N_{reg} = 5, N_{opt} = 0$	
9		$N_{reg} = 1, N_{opt} = 0$	
10		$N_{reg} = 4, N_{opt} = 1$	true
11		$N_{reg} = 5, N_{opt} = 0$	
12		$N_{reg} = 1, N_{opt} = 0$	
13	ALM + NP	$N_{reg} = 4, N_{opt} = 1$	false
14		$N_{reg} = 5, N_{opt} = 0$	
15		$N_{reg} = 1, N_{opt} = 0$	
16		$N_{reg} = 4, N_{opt} = 1$	true
17		$N_{reg} = 5, N_{opt} = 0$	
18		$N_{reg} = 1, N_{opt} = 0$	

Gegensatz zur Peer-Auswahl nach ALM (KV 7–18) nutzt herkömmliches BT kein Vorwissen über die Netzwerkstruktur. Peers müssen zunächst über einige Runden kennen gelernt und zufällig gewählt werden. Erst sobald Informationen über die Upload- und Download-Raten zu diesen bereitstehen, kann die Peer-Auswahl nach dem klassischen Belohnungsprinzip erfolgen. Es sollte der Großteil der Ausführungen des Choking-Algorithmus außerhalb der zufallsbehafteten Startphase stattfinden, um eine faire Gegenüberstellung von Standard-BT und den MeNTor-Optimierungen zu erlauben. Darüber hinaus kann die Anzahl der Ausführungen auf jedem Peer variieren. Befindet er sich in der Rolle *Leecher*, beträgt das Ausführungsintervall des Choking-Algorithmus 10 s. In der Rolle *Seed* erfolgt der Aufruf alle 30 s. Damit besitzt der initiale Seed stets die minimale, der zuletzt abschließende Leecher stets die maximale Anzahl an Ausführungen. Auf den restlichen Peers ist diese abhängig vom Zeitpunkt des Wechsels von der Leecher- in die Seed-Rolle.

In einer Reihe von Testläufen mit den verschiedenen KV im kleinsten und größten TF (TF 2a bzw. TF 2e, siehe Abb. 4.8) wurden 50 MB als geeignete Dateigröße bestimmt. Selbst im kleinsten Setup, TF 2a, betrug die mittlere Download-Zeit pro Peer mindestens 500 s. Während dieser wurde der Choking-Algorithmus zwischen 17 (initialer Seed) und 50 mal (zuletzt abschließender Leecher) ausgeführt. Im größten Setup, TF 2e, betrug die durchschnittliche Download-Zeit je nach KV zwischen 850 s und 1500 s. Es fanden somit ca. 28–85 bzw. 50–150 Ausführungen des Choking-Algorithmus pro Peer statt. Die Wahl der Größe der Pieces, in die der Datensatz zur Übertragung unterteilt wird, hängt normalerweise von dessen Gesamtgröße ab. Studien haben zudem gezeigt, dass eine Zahl von 1000–2000 Pieces zu bevorzugen ist [208]. Diese bietet einen guten Kompro-

miss zwischen der Anzahl gleichzeitig zur Übertragung bereitstehender Pieces im Netzwerk und dem entstehenden Nachrichten-Overhead, um diese anzufragen. Die Größe der Pieces für den 50 MB-Datensatz wurde daher auf 32 kB festgelegt, was zu einer Menge von ca. 1600 Pieces führte.

4.6.4.2 Bootstrapping des BitTorrent-Schwarms

Abhängig vom jeweiligen TF bestand der BT-Schwarm aus dem initialen Seed sowie 8, 12, 16, 20 bzw. 24 weiteren Mesh-Knoten des Testaufbaus als Leecher. Der initiale Seed übernahm stets auch die Funktion des *Trackers*. Bei diesem handelt es sich um einen Bootstrap-Knoten, der anfragenden Peers eine Liste von Kontakten aus dem Schwarm liefert (siehe Kap. 4.2). Alle zur Teilnahme am Schwarm benötigten Metadaten waren bereits vor Beginn jedes Experiments in Form einer **.torrent*-Datei auf allen Peers vorhanden. Unter anderem enthält diese Informationen über den im Schwarm ausgetauschten Datensatz, dessen Pieces sowie die Adresse des Trackers. Im Vergleich zum Datensatz besitzt die **.torrent*-Datei nur eine geringe Größe. Sie wird zudem in einer platzsparenden Kodierung namens *Bencode* gespeichert [31, 209]. Die Größe der Datei hängt dabei stark von der Größe und Aufteilung des Datensatzes ab, da sie die Prüfsummen aller Pieces enthält. Für den in dieser Arbeit genutzten Datensatz von 50 MB, unterteilt in 1600 Pieces, entstand eine 32 kB große **.torrent*-Datei.

Auch im Anwendungsszenario der BT-basierten Update-Verteilung in einem städtischen WMN gehört es zu den Aufgaben des Netzwerkbetreibers, eine Bootstrap-Infrastruktur für die Bildung des Schwarms bereitzustellen. Der Natur des WMN entsprechend sollte diese ebenfalls robust und ausfallsicher sein. Neben der klassischen Realisierung mittels Tracker und **.torrent*-Datei werden in der Praxis meist BT-Protokollerweiterungen genutzt, die eine Abhängigkeit von zentralen Komponenten weitgehend vermeiden. Diese Erweiterungen wurden in sogenannten *BitTorrent Enhancement Proposals (BEPs)* veröffentlicht. So existieren mehrere BEPs, die Mechanismen zur Entlastung des Trackers spezifizieren. Im einfachsten Fall können nach BEP 12 mehrere Tracker in den Metadaten hinterlegt werden, sodass sich die Robustheit gegenüber Ausfällen erhöht [210]. Nach BEP 5 können sich alle BT-Peers in einem zusätzlichen P2P-Netzwerk organisieren, das nur der Suche nach Schwarm-Teilnehmern dient [211]. Dieses Such-Netzwerk basiert auf dem Prinzip einer *Distributed Hash Table (DHT)*, in der jeder Peer nur eine Teilmenge aller Kontakte speichert. Die Zuständigkeit für Kontaktinformationen wird durch ein logisches Distanzmaß zwischen Peer-Adressen bestimmt. Eine in BEP 9 beschriebene Erweiterung ermöglicht Peers den direkten Austausch der Metadaten für den Schwarmbeitritt [212]. Anstatt der **.torrent*-Datei wird nur noch ein kurzer *Magnet Link* benötigt, der einen maximal 40 Bytes langen, eindeutigen Hash-Wert für den Schwarm enthält. Mit dieser Information können Peers in der DHT gefunden werden, welche die restlichen Metadaten für den im Schwarm ausgetauschten Datensatz bereitstellen. Eine Alternative bzw. Ergänzung zur DHT stellt die in BEP 11 definierte Protokollerweiterung *Peer Exchange (PEX)* dar [213]. Diese erlaubt Peers, Informationen über aktive Verbindungen zu anderen Peers miteinander auszutauschen. Kombiniert mit der DHT-basierten Suche kann PEX die Aktualität der Schwarm-Informationen erhöhen. BEP 14 wiederum definiert eine auf Broadcast basierende Peer-Suche innerhalb von Local Area Networks (LANs) [214]. Diese eignet sich besonders für den Einsatz WLAN- und 802.11s-Mesh-Umgebungen und wird ebenfalls durch den BT-Client Vuze unterstützt [215].

Der Aspekt des Schwarmbeitritts lag im Gegensatz zur Peer-Auswahl-Strategie nicht im Fokus dieser Arbeit. Um eine mögliche Beeinflussung der Experimente zu vermeiden, wurde daher explizit auf die Nutzung der BEPs 5, 9 und 11 verzichtet. Stattdessen kamen nur ein Tracker und die in Vuze bereits integrierte LAN-Peer-Suche zum Einsatz.

4.6.4.3 Experimentablauf

Die Initialisierung des Mesh-Netzwerks und der BT-Client-Instanzen sowie die Automatisierung der Experimente wurden durch Kommandozeilenskripte [4.3] realisiert. Diese wurden teils auf den Mesh-Knoten und teils auf einem Kontroll-PC ausgeführt, der separat über Ethernet mit der Testumgebung verbunden war.

Der initiale Seed und alle Leecher wurden vor Beginn jedes Experiments in einem pausierten Zustand gestartet. Noch bevor der eigentliche Datenaustausch erfolgte, nahmen sie Kontakt mit dem Tracker auf und kündigten sich bei diesem als Schwarmteilnehmer an. Zum Start des Experiments wurden alle Peers synchron aktiviert und erhielten zunächst die komplette Kontaktliste vom Tracker. Mittels LAN-Peer-Suche wurde diese anschließend während des Experiments aktuell gehalten. Dadurch war jederzeit sichergestellt, dass Peers im Entscheidungsprozess des Choking-Algorithmus nicht aufgrund fehlender Kontaktinformationen eingeschränkt waren.

Jeder Leecher wechselte automatisch in die Rolle eines Seeds, sobald alle Pieces des Datensatzes erfolgreich empfangen wurden. Der BT-Client-Parameter *Seed Lingering Time*, der die Verweildauer als Seed nach vollständigem Download definiert, erhielt den Wert *unbegrenzt*, sodass Seeds stets bis zum Ende des Experiments im Schwarm verblieben. Der Abschluss des Datenaustauschs war somit daran festzustellen, dass sich jeder Peer in der Seed-Rolle befand.

Nach jeder Messung wurden die Log-Dateien aller BT-Client-Instanzen vom Kontroll-PC abgerufen. Wie in Kap. 3 erläutert, waren alle Geräte der Testumgebung mittels Precision Time Protocol (PTP) synchronisiert. Mithilfe eines Python-Skripts wurden die Zeitstempel der Übertragungsinformationen aus den Log-Dateien extrahiert und zueinander in Beziehung gesetzt. Die Download-Zeiten ergaben sich aus der Zeitdifferenz zwischen dem Download-Ende des jeweiligen Peers und dem Upload-Beginn des initialen Seeds. Zusätzlich wurden aus der Sicht jedes Peers die zeitlichen Verläufe der Empfangsdatenraten aller BT-Verbindungen zu anderen Peers ermittelt. Dazu wurden über die Gesamtdauer der Messung die zu jedem 1 s-Zeitfenster gehörenden Piece-Übertragungen als Momentandatenrate zusammengefasst.

4.6.5 Ergebnisübersicht für die Kombination aller Optimierungen

Bevor die Ergebnisse aller KV des Prototypen im Detail diskutiert werden, wird nachfolgend eine Übersicht der Ergebnisse bei Aktivierung aller MeNTor-Optimierungen gegeben. Dies entspricht KV 18, die sich erwartungsgemäß als beste Parametrisierung in den Experimenten herausstellte und somit die zu empfehlende Standardkonfiguration für Mesh-Anwendungen ist. Sie kombiniert alle vorgeschlagenen Erweiterungen des BT-Choking-Algorithmus für 802.11s-Netzwerke: die ALM-basierte Peer-Auswahl mit Nachbar-Präferenz, die nichtrestriktive Upload-Strategie *Generous Mode* für Leecher sowie die Begrenzung paralleler Uploads auf einen einzigen, regulär vergebenen Slot.

Abb. 4.9 zeigt die in jedem TF mit KV 18 erzielte, mittlere Schwarm- bzw. Peer-Download-Zeit bezogen auf die Referenzkonfiguration KV 1, die dem unmodifizierten Standard-BT entspricht. Alle Werte sind jeweils normalisiert auf die mit KV 1 erreichten Ergebnisse, welche pro TF den Bezugswert 100 % darstellen. Die absoluten Bezugswerte sind dabei für jeden TF und zwischen Schwarm- und Peer-Download-Zeit unterschiedlich (siehe Detaildarstellung in Kap. 4.6.6 und 4.6.7), sodass hier nur die jeweilige Verbesserung gegenüber Standard-BT ersichtlich ist.

Bei geringster Schwarmgröße von 9 Peers, bestehend aus dem initialen Seed und 8 Leechern, konnte im TF 1a bereits eine Zeitersparnis von 8 % pro Peer und 10 % für den gesamten Schwarm verzeichnet werden. Hierbei befand sich der initiale Seed entsprechend Szenario 1 (TF 1a–1e) in der Gitterecke.

[4.3] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/mentor-scripts`

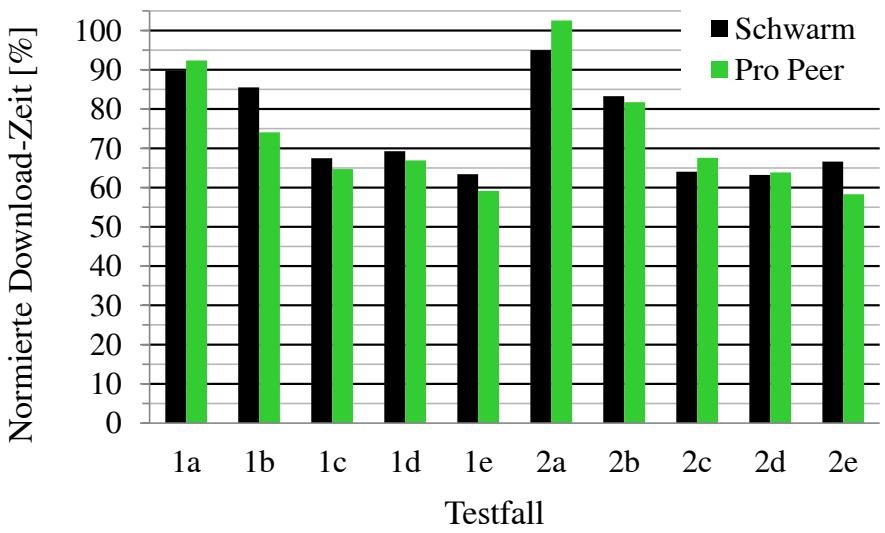


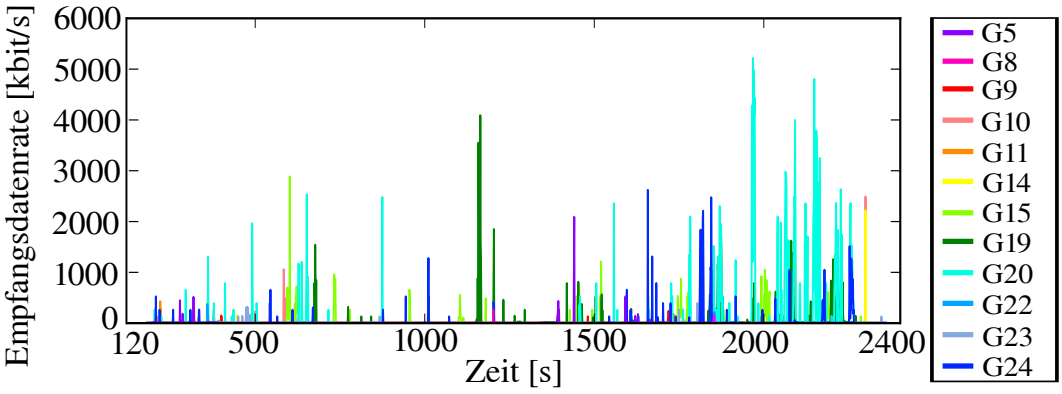
Abbildung 4.9: Mittlere Download-Zeit bei Kombination aller Optimierungen (Konfigurationsvariante 18) in %, normiert auf Standard-BT (Konfigurationsvariante 1) [B 4]

Im zugehörigen TF 2a fiel die Verbesserung geringer aus, da sich der initiale Seed hier im Zentrum der Gitteranordnung befand und alle Peers über maximal 2 Hops erreichen konnte. Somit ergab sich in der Startphase des Datenverteilung ein geringerer Optimierungsspielraum durch Bevorzugung näher gelegener Peers. Aufgrund der noch geringen Schwarmgröße war zudem noch kein positiver Einfluss des Upload-Slot-Limits erkennbar. Es ergab sich für die Schwarm-Download-Zeit nur eine geringe Zeitersparnis von 5 %, hinsichtlich der mittleren Download-Zeit pro Peer zeigte sich keine Verbesserung gegenüber Standard-BT.

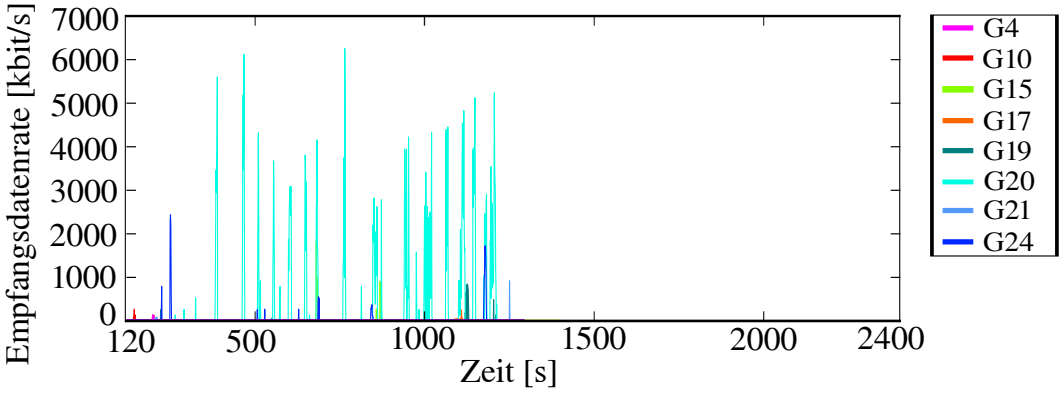
Mit zunehmender Schwarm-Teilnehmerzahl in den TF 1b–e bzw. 2b–e erhöhte sich jedoch das Kommunikationsaufkommen im Netzwerk. Insbesondere die Knoten im Inneren der Gitteranordnung konkurrierten mit jeweils 8 Nachbarknoten um den Zugriff auf das Medium. Je dichter das Gitter mit logischen Peers besetzt war, desto mehr Vorteile ergaben sich durch Berücksichtigung der ALM-Kosten bei der Peer-Auswahl. Für eine Schwarmgröße von 13 Peers in TF 1b und 2b lag die Zeitersparnis im Bereich von 15–25 %. In TF 1c und 2c mit 17 Peers im Schwarm betrug die Zeitersparnis bereits mehr als 30 %. Dieses Optimierungsergebnis konnte in den TF 1d und 2d mit 21 Peers bestätigt werden. Im größten Schwarm, bestehend aus 25 Peers, wurde eine Reduzierung der Download-Zeit pro Peer von mehr als 40 % erreicht. So ergab sich in TF 1e bzw. 2e eine Zeitersparnis in der Höhe von 41 % (42 %) pro Peer bzw. 37 % (33 %) für den gesamten Schwarm.

Die Abbildungen 4.10 (a) und (b) veranschaulichen den positiven Effekt der kombinierten MeNTor-Optimierungen (KV 18) auf den zeitlichen Verlauf des kollaborativen Datenaustauschs im Vergleich zu Standard-BT (KV 1). Gezeigt werden die Ergebnisse zweier Einzelmessungen im TF 1e mit einer Schwarmgröße von 25 Peers und dem initialen Seed in der unteren linken Ecke des Gitters (nummerierte Anordnung in Abb. 4.10 (c)). Dargestellt sind jeweils die über die Gesamtdauer der Messung aufgetragenen Empfangsdatenraten der Verbindungen zu anderen Peers aus Sicht des Knotens *Galileo 25* (G25). Dieser befand sich in der oberen rechten Ecke der Anordnung, d.h. im maximalen Hop-Abstand zum initialen Seed, und beendete in beiden Messungen als letzter Peer seinen Download. Die Experiment-Initialisierungsphase der festen Dauer von 120 s vor Aktivierung des Schwarms wurde bewusst aus den Diagrammen entfernt. Die farbigen Legenden in Abb. 4.10 (a) und (b) enthalten alle Peers, von denen G25 im jeweiligen Experiment Pieces empfangen hat.

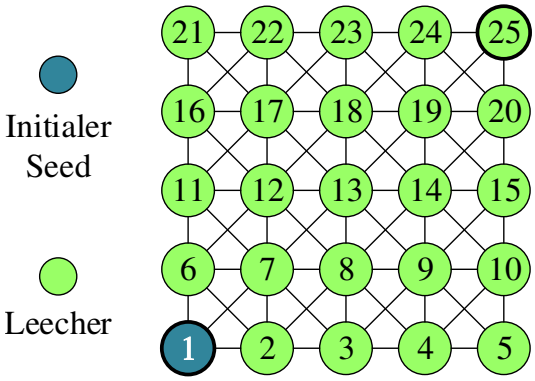
Die Abb. 4.10 (a) verdeutlicht, dass die Peer-Auswahlstrategie von Standard-BT eine längere Zeit benötigte, bis G25 vorrangig von physisch nahen Knoten versorgt wurde. Peers mussten in Standard-



(a) MeNTor-Konfigurationsvariante 1 (Standard-BT)



(b) MeNTor-Konfigurationsvariante 18 (Kombination aller Optimierungen)



(c) Nummerierte Gitteranordnung für Testfall 1e

Abbildung 4.10: Zeitverlauf der Empfangsdatenraten des Knotens G25 für Konfigurationsvariante 1 (Teilabb. (a)) und Konfigurationsvariante 18 (Teilabb. (b)) im Testfall 1e (Teilabb. (c)) [B 4]

BT zunächst rein zufällig gewählt und kennen gelernt werden, bevor ihre Bewertung basierend auf der gemessenen Upload-Datenrate (in der Seed-Rolle) bzw. dem Verhältnis aus empfangener und gesendeter Datenmenge (in der Leecher-Rolle) erfolgen konnte. Zudem nutzte jeder Peer stets einen seiner insgesamt 5 Upload-Slots für die optimistische Wahl eines zufälligen Peers. Zwar erhielt G25 im späteren Verlauf des Experiments den Großteil seiner Pieces von den Nachbarknoten G20 und G24, allerdings erst nachdem diese ihren Download abgeschlossen und den Wechsel in die Seed-Rolle

vollzogen hatten. Zuvor wurde G25 von verschiedenen Peers zeitweise zufällig gewählt, erhielt somit nur sporadisch Pieces und war nach dem Belohnungsprinzip benachteiligt. Trotz seiner maximalen Distanz zum initialen Seed wurde G25 immer wieder auch durch weit entfernte Knoten gewählt, wie z. B. G5 und G11. In der Folge wurden ineffiziente Multi-Hop-Übertragungen quer durch das Netzwerk durchgeführt und der parallele, kollaborative Datenaustausch beeinträchtigt. Insgesamt erhielt G25 Pieces von 12 verschiedenen Peers.

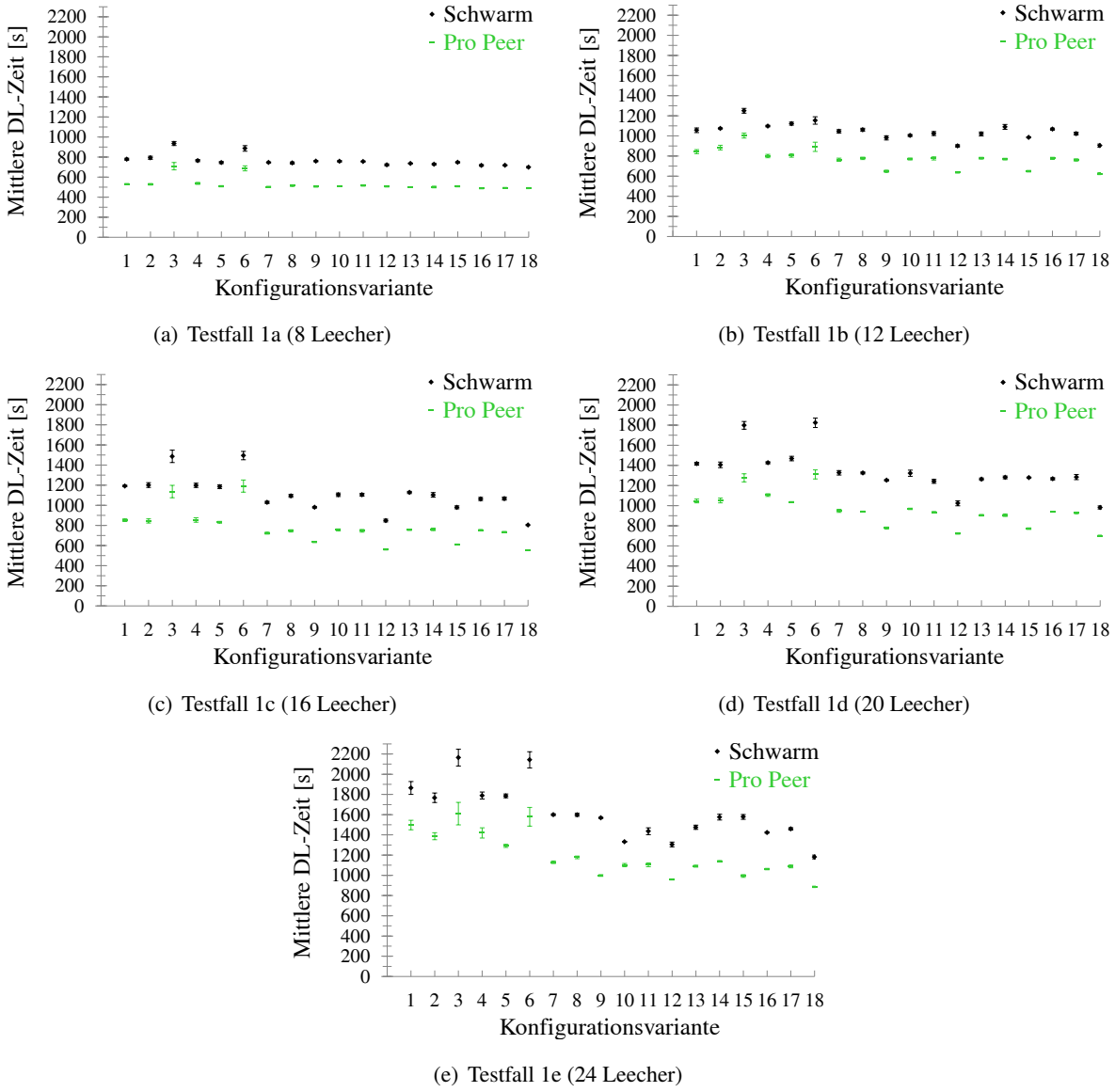
Im Gegensatz dazu waren im Fall von KV 18, dargestellt in Abb. 4.10 (b), nur 8 verschiedene Peers an der Versorgung von G25 beteiligt. Zudem vergaben viele von ihnen nur kurzzeitig Upload-Kapazität an G25, falls sie fehlende Pieces besaßen, während der Hauptteil der Daten von den Nachbarknoten G20 und G24 bereitgestellt wurde. Es zeigte sich klar, dass die Nutzung der ALM als Peer-Auswahlkriterium in Kombination mit der Nachbar-Präferenz (NP) zuverlässig eine frühzeitige Bevorzugung physisch naher Peers bewirkt. Darüber hinaus führen der *Generous Mode* und die Begrenzung auf nur einen, nach ALM vergebenen, Upload-Slot pro Peer zu einer klaren Geschwindigkeitserhöhung. An Pieces interessierte Peers werden folglich ohne Einschränkung und exklusiv, d.h. mit den gesamten über eine Unchoke-Periode zur Verfügung stehenden Kommunikationsressourcen, versorgt.

4.6.6 Ergebnisse bei Worst-Case-Platzierung des initialen Seeds

Die Ergebnisse für Szenario 1, in dessen TF sich der initiale Seed in der Ecke der Gitteranordnung befand, sind in Abb. 4.11 (a)–(e) dargestellt und werden nachfolgend erläutert. Für jeden TF ist pro KV die durchschnittliche *Peer-Download-Zeit* grün und die durchschnittliche *Schwarm-Download-Zeit* schwarz dargestellt. Alle Messpunkte sind mit 95 %-Konfidenzintervallen angegeben. Trotz der Anzahl von nur 5 Messungen pro KV konnte über alle TF hinweg eine hohe Reproduzierbarkeit der Ergebnisse festgestellt werden. Lediglich für KV 1–6 (Peer-Auswahl nach Standard-BT) ergab sich mit zunehmender Schwarmgröße eine sichtbar steigende Varianz des Gesamtzeitbedarfs. Diese fiel insbesondere mit der Begrenzung auf einen Upload-Slot in KV 3 und 6 stärker aus, da hier das schnelle zufallsbasierte Kennenlernen geeigneter Peers beeinträchtigt war.

Testfall 1a: 8 Leecher: In diesem TF wurden der initiale Seed und 8 Leecher homogen im Mesh-Gitter verteilt, sodass sich diese jeweils außerhalb der Link-Reichweite anderer Peers befanden. Die Messergebnisse sind in Abb. 4.11 (a) dargestellt. Trotz Worst-Case-Platzierung des initialen Seeds war der Vorteil der ALM als Auswahlkriterium (KV 7–18) bei dieser geringen Schwarmgröße noch marginal. Die Peer-Auswahlstrategie von Standard-BT in Kombination mit mehreren Upload-Slots (KV 1, 2, 4 und 5) war hier noch in der Lage, nach kurzer Zeit günstige Peers zu finden. Wurde jedoch bei Nutzung des Default-Kriteriums nur ein einziger Upload-Slot gewährt (KV 3 und 6), führte dies im Allgemeinen zu einer sichtbar schlechteren Performance. Hier bestand eine geringere Chance des zufälligen Kennenlernens günstiger Peers, wodurch vermehrt suboptimale Peers bedient wurden. Dieser Effekt verstärkte sich mit zunehmender Schwarmgröße in den späteren TF. Aufgrund der in TF 1a stets mindestens zwei Hops voneinander entfernten Peers zeigte die NP-Option (KV 13–18) erwartungsgemäß noch keinen Vorteil. Diese Topologie besaß im Vergleich zu den TF mit höherer Schwarm-Teilnehmerzahl und -dichte zudem eine geringere kompetitive Kanalauslastung durch die Peers. Im Gegensatz zu den dichter besetzten TF führte daher die Kombination aus ALM-Kriterium und Upload-Slot-Limit (KV 9, 12, 15 und 18) noch zu keiner erkennbaren Verbesserung gegenüber Standard-BT (KV 1).

Testfall 1b: 12 Leecher: In TF 1b wurden dem Schwarm 4 zusätzliche Leecher im inneren Quadrat des Gitters hinzugefügt. Aufgrund der erhöhten Peer-Anzahl und -Dichte zeigte die Beschränkung der BT-Upload-Slots unter Nutzung der ALM als Auswahlkriterium bereits eine sichtbare Verbesserung (siehe KV 9, 12, 15 und 18 in Abb. 4.11 (b)). Dieser positive Effekt ist jedoch an die Kombination mit der ALM gebunden, die als Vorwissen zur Wahl günstiger Peers genutzt wird. Wie zuvor in TF 1a



(a) Testfall 1a (8 Leecher)

(b) Testfall 1b (12 Leecher)

(c) Testfall 1c (16 Leecher)

(d) Testfall 1d (20 Leecher)

(e) Testfall 1e (24 Leecher)

Abbildung 4.11: Ergebnisse für Szenario 1: mittlere Download (DL)-Zeit pro Peer und für den Schwarm (langsamster Peer) über alle Konfigurationsvarianten in den Testfällen 1a–e [B 4]

stellte die Kombination aus Upload-Slot-Limit und Peer-Auswahl von Standard-BT die schlechteste Strategie dar (KV 3 und 6). Unter den ALM-basierten KV mit mehreren Upload-Slots pro Peer wies die zufällige Vergabe eines optimistischen Uploads noch keinen erkennbaren Nachteil auf. Bei dieser Schwarmgröße bestand mit der Standardanzahl von 5 parallelen Uploads immer noch eine geringe Chance, einen besonders ungeeigneten fünften Peer zufällig auszuwählen. Ebenso bewirkte die NP-Option (KV 13–18) bei dieser Peer-Dichte noch keine Verbesserung. Ein klarer Einfluss war erst für enger besetzte Topologien mit höherem Kommunikationsaufkommen zu erwarten, welches die zeitnahe und zuverlässige Aktualisierung der ALM-Informationen des 802.11s-Routing-Protokolls beeinträchtigen kann. Dort unterstützt das Hinzuziehen des lokalen Wissens über direkte Nachbarknoten die robuste Auswahl naher Peers.

Testfall 1c: 16 Leecher: In diesem TF wurden die zuvor im Inneren des Gitters hinzugefügten 4 Leecher zusammen mit 4 weiteren Leechern an den Gitterrand verschoben, sodass sich eine Ringtopologie mit einem Peer im Zentrum ergab. Jeder Peer besaß maximal 2 andere Peers in seiner Nach-

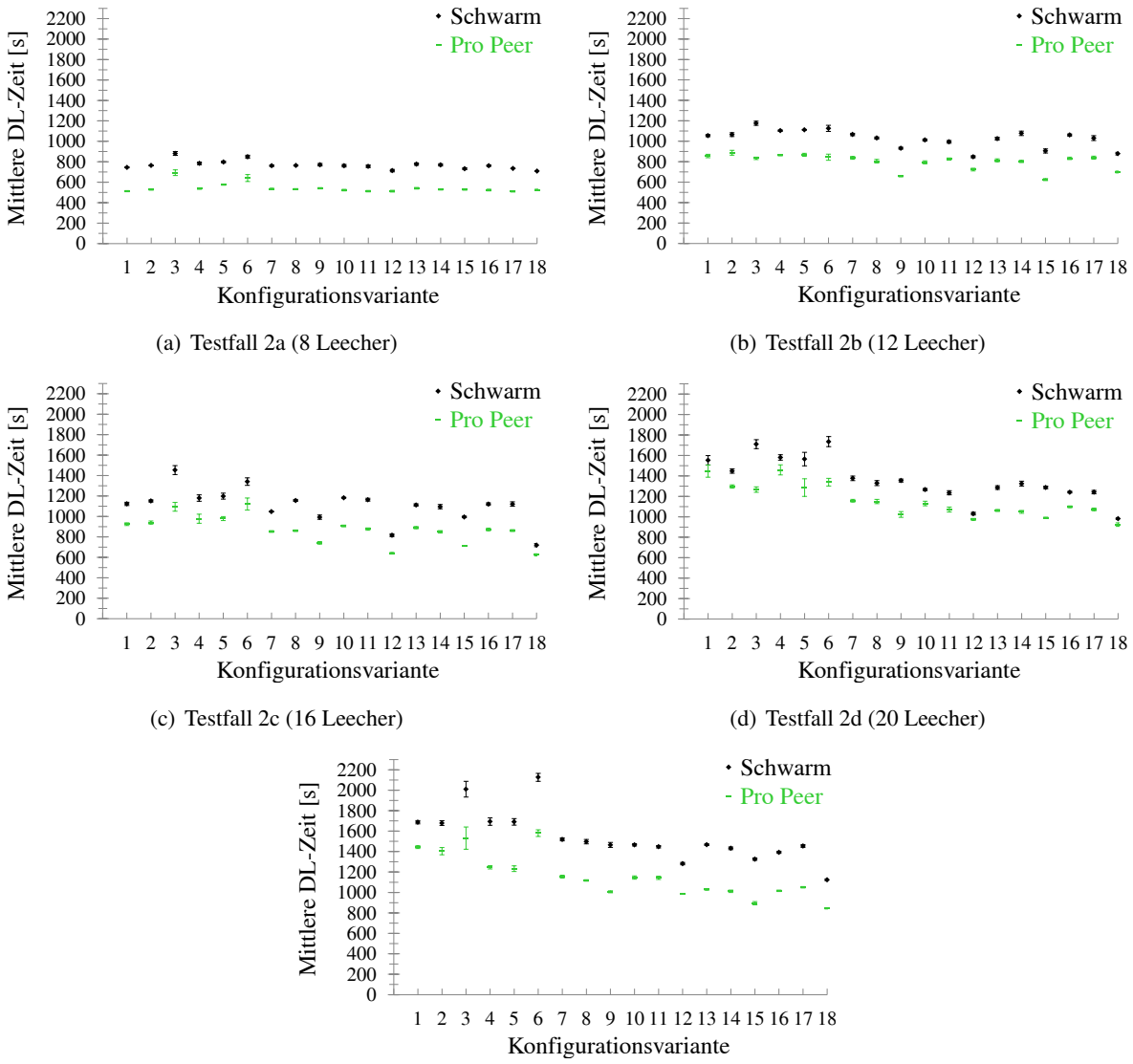
barschaft. Die auf dem inneren Ring liegenden Mesh-Knoten dienten wie in TF 1a nur der Datenweiterleitung zwischen Gitterrand und -mitte. Abgesehen vom erwarteten Anstieg der Download-Zeit für die erhöhte Schwarmgröße waren die relativen Ergebnisse aller KV mit denen des vorherigen TF vergleichbar (siehe Abb. 4.11 (c)). Die geringere Performance der KV 3 und 6 zeigte sich jedoch deutlicher, auch durch eine merklich höhere Varianz der gemessenen Download-Zeiten. Darüber hinaus war der positive Einfluss des *Generous Mode* in Kombination mit dem ALM-Kriterium und dem Upload-Slot-Limit in KV 12 und 18 nun klar erkennbar. Diese Strategie führte zu einer insgesamt höheren Upload-Parallelität im Schwarm und beschleunigte den kollaborativen Verteilungsprozess. Nachdem die ersten Leecher vom initialen Seed mit Pieces versorgt worden waren, begannen sie selbst mit dem Upload zu anderen Peers, auch wenn diese im Gegenzug keine benötigten Pieces anboten. Mit zunehmender Schwarmgröße kam dieser Vorteil immer deutlicher zum Tragen.

Testfall 1d: 20 Leecher: TF 1d stellt eine Kombination der Topologien von TF 1b und 1c dar. Auch hier entsprachen die relativen Ergebnisse aller KV im Wesentlichen dem vorherigen TF (siehe Abb. 4.11 (d)). Die NP-Option (KV 13–18) wirkte sich bereits positiv auf die mittlere Peer-Download-Zeit aus. Die Kombination aus *Generous Mode*, ALM-Kriterium und Upload-Slot-Limit in KV 12 und 18 reduzierte erneut deutlich die mittlere Download-Zeit sowohl pro Peer als auch für den gesamten Schwarm. Wie erwartet traf dies jedoch nicht in gleichem Umfang auch für KV 9 und 15 zu, die das Upload-Slot-Limit ohne *Generous Mode* verwenden. Zwar ergibt sich auch hier eine geringere Download-Zeit pro Peer durch das Upload-Slot-Limit, da Peers einander stets exklusiv mit den insgesamt verfügbaren Kommunikationsressourcen über eine Unchoke-Periode versorgen. Dennoch greift das restriktive Verhalten von Standard-BT, bei dem Leecher nur *interessante* Peers bedienen, die im Gegenzug noch benötigte Pieces bereitstellen. Diese müssen sie zuvor vom initialen Seed oder von anderen, bereits in die Seed-Rolle gewechselten Leechern erhalten haben. Dadurch ergibt sich ein wellenartiger Verteilungsprozess, der sich in einer größeren Zeitdifferenz zwischen dem Download-Ende der ersten und letzten Peers im Schwarm äußern kann. Eine solche Divergenz zwischen mittlerer Peer- und Schwarm-Download-Zeit ist klar erkennbar in den Ergebnissen der TF 1d (KV 15), 1e (KV 9 und 15), 2d (KV 9) sowie 2e (KV 9). Im Gegensatz dazu erlaubte der *Generous Mode* in KV 12 und 18 die sofortige Versorgung von *uninteressanten*, jedoch möglicherweise nach ALM günstigeren Leechern, wodurch sich diese Zeitdifferenz reduzierte.

Testfall 1e: 24 Leecher: In TF 1e waren alle 25 Mesh-Knoten als Peers im BT-Overlay beteiligt. Bei dieser Schwarmgröße zeigten die Ergebnisse für die Peer-Auswahl von Standard-BT (KV 1–6) eine erheblich höhere Varianz (siehe Abb. 4.11 (e)). Es stellte sich zudem heraus, dass die Reservierung eines optimistischen Slots neben der Vergabe mehrerer regulärer Upload-Slots mittels ALM auch vorteilhaft sein kann (KV 7, 10, 13 und 16). Bei dieser Schwarmgröße ergänzte der zufällig vergebene Slot die ansonsten ALM-basierte Peer-Auswahl anteilig um die Chance, Pieces in weiter entfernte Bereiche des Netzwerks zu streuen und dadurch insgesamt die Übertragungsparallelität zu erhöhen. Dennoch erwiesen sich KV 12 und 18 wie in den vorherigen TF als die besten Konfigurationen mit einer Zeitersparnis von mehr als 40 % im Vergleich zu Standard-BT (KV 1). Dies zeigt den Vorteil der Kombination des ALM-Kriteriums mit der Beschränkung auf einen Upload-Slot, welche die exklusive Versorgung physisch naher Peers bewirkt. Wie in TF 1d zeigte zudem die NP-Option eine klare Verbesserung der Peer-Download-Zeit für die erhöhte Schwarmgröße und -dichte.

4.6.7 Ergebnisse bei Best-Case-Platzierung des initialen Seeds

Die Ergebnisse für Szenario 2, in dessen TF sich der initiale Seed im Zentrum der Gitteranordnung befand, sind in Abb. 4.12 (a)–(e) dargestellt und werden nachfolgend erläutert. Erneut sind die durchschnittliche *Peer-Download-Zeit* grün und die durchschnittliche *Schwarm-Download-Zeit* schwarz dargestellt sowie alle Messpunkte mit 95 %-Konfidenzintervallen angegeben. Auch in Szenario 2 war eine mit der Schwarmgröße zunehmende Varianz der Ergebnisse nur für KV 1–6 (insbesondere für



(a) Testfall 2a (8 Leecher)

(b) Testfall 2b (12 Leecher)

(c) Testfall 2c (16 Leecher)

(d) Testfall 2d (20 Leecher)

(e) Testfall 2e (24 Leecher)

Abbildung 4.12: Ergebnisse für Szenario 2: mittlere Download (DL)-Zeit pro Peer und für den Schwarm (langsamster Peer) über alle Konfigurationsvarianten in den Testfällen 2a–e [B 4]

KV 3 und 6) festzustellen. Diese war aufgrund der günstigeren Positionierung des initialen Seeds jedoch weniger stark ausgeprägt als in Szenario 1.

Testfall 2a: 8 Leecher: In TF 2a, der ansonsten identisch mit TF 1a war, befand sich der initiale Seed im Zentrum des Gitters und damit in 2-Hop-Distanz zu jedem Leecher. Daher konnten zu Beginn unabhängig von der KV keine suboptimalen Peers durch den initialen Seed gewählt werden. Dies unterscheidet sich von TF 1a, in dem die Peer-Distanz bis zu 4 Hops betrug. Im Gegensatz zu Szenario 1 mit 3 benachbarten Knoten war der initiale Seed im Szenario 2 stets mit 8 Knoten direkt verbunden. Insbesondere wenn diese Nachbarschaft mit BT-Peers besetzt war, besaß sie eine hohe kompetitive Kanalauslastung in der Anfangsphase der Datenverteilung. Im TF 2a war der initiale Seed jedoch von reinen Weiterleitungsknoten umgeben, die nicht am Schwarm teilnahmen. Wie bereits in TF 1a beobachtet, zeigte die ALM-basierte Peer-Auswahl für diese kleine Schwarmgröße und Peer-Dichte noch keine signifikante Verbesserung (siehe Abb. 4.12 (a)). Dies galt ebenso für die Kombination von ALM-Kriterium und Upload-Slot-Limit.

Testfall 2b: 12 Leecher: Die Ergebnisse von TF 2b sind weitgehend mit denen des korrespondierenden TF 1b vergleichbar (siehe Abb. 4.12 (b)). Mit zunehmender Schwarmgröße wurde die Kombination aus ALM und Upload-Slot-Limit immer vorteilhafter. Eine sichtbare Divergenz der Peer- und Schwarm-Download-Zeit ergab sich in den KV 3 und 6, welche die Peer-Auswahl von Standard-BT mit dem Upload-Slot-Limit kombinieren. Hier ging eine Reduzierung der mittleren Peer-Download-Zeit mit einer Erhöhung des Zeitbedarfs für den gesamten Schwarm einher. Dabei führte der *Generous Mode* in KV 6 insgesamt zu einem leichten Zeitvorteil. Im Zusammenhang mit der Peer-Auswahl von Standard-BT kann das Upload-Slot-Limit vorteilhaft für die mittlere Peer-Download-Zeit sein, da Peers einander exklusiv mit ihren verfügbaren Kommunikationsressourcen bedienen. Die Upload-Performance hängt jedoch von der zufälligen Auswahl günstig gelegener Peers ab. Im Allgemeinen verringert das Upload-Slot-Limit bei Standard-BT die Chance, in kurzer Zeit physisch nahe Peers kennen zu lernen, was zu einer insgesamt höheren Download-Zeit für den gesamten Schwarm führt.

Testfall 2c: 16 Leecher: In TF 2c (siehe Abb. 4.12 (c)) fiel der zeitliche Abstand zwischen Peer- und Schwarm-Download-Zeit kleiner aus als im korrespondierenden TF 1c, da auch der mittlere Hop-Abstand zwischen initialem Seed und Leechern geringer war. Diese Beobachtung gilt auch für die nachfolgenden TF 2d und 2e. Wie in TF 2a nahmen die Mesh-Knoten um den initialen Seed nicht am Schwarm teil, sodass kein besonderer Vorteil durch die NP-Option entstand. Die Kombination aus ALM-Kriterium und Upload-Slot-Limit in KV 9, 12 und 18 verbesserte deutlich die Download-Zeit pro Peer und Schwarm.

Testfall 2d: 20 Leecher: Aufgrund der gestiegenen Schwarmgröße in TF 2d wurde eine deutlichere Verbesserung durch das ALM-Kriterium und den *Generous Mode* erreicht (siehe Abb. 4.12 (d)). Dabei erzielten KV 12 und 18 erneut die höchste Zeitersparnis. Wie auch in anderen TF konnte für KV, die das Upload-Slot-Limit ohne *Generous Mode* nutzen, eine teils unterschiedliche Auswirkung auf die Peer- und Schwarm-Download-Zeit festgestellt werden (siehe KV 3 und 9). Zudem bewirkte die NP-Option (KV 13–18) eine noch stärkere Verbesserung als im korrespondierenden TF 1d. Im Szenario 2 führte die dichter mit Peers besetzte Nachbarschaft um den initialen Seed bereits zu Beginn des Verteilungsprozesses zu einer höheren kompetitiven Auslastung des Kanals. Dies erhöhte die Wahrscheinlichkeit einer unzuverlässigeren Aktualisierung der ALM-Informationen durch das 802.11s-Routing-Protokoll. Das Hinzuziehen der lokalen Kenntnis von Nachbarknoten unterstützte sichtbar die robuste Auswahl naher Peers.

Testfall 2e: 24 Leecher: Die Ergebnisse in TF 2e (siehe Abb. 4.12 (e)) bestätigen den Trend des vorherigen TF 2d und sind ebenfalls mit denen des korrespondierenden TF 1e vergleichbar. Eine steigende Verbesserung wurde durch die ALM-basierten KV erreicht, insbesondere in Kombination mit *Generous Mode* und NP. In TF 1e war der positive Einfluss des *Generous Mode* jedoch stärker ausgeprägt, während in TF 2e die NP-Option aufgrund der zentralen Position des initialen Seeds eine deutlichere Reduzierung der Peer-Download-Zeit bewirkte. Wie bei TF 2d erwies sich auch hier die Nutzung lokaler Link-Informationen als hilfreich, um die Robustheit der Peer-Auswahl auch bei höherem Kommunikationsaufkommen im Inneren der Gitteranordnung zu garantieren. Wie in TF 1e reduzierte KV 18 sowohl die Peer- als auch die Schwarm-Download-Zeit um mehr als 40 % gegenüber Standard-BT (KV 1).

Zusammengefasst führt der Ansatz, den BT-Datenverkehr auf die Verbindungen mit der besten Metrik zu verteilen, bereits bei einer Mesh-Netzwerkgröße von 25 Knoten mit Pfadlängen von bis zu 4 Hops zu klaren Performance-Verbesserungen. Die Nutzung der ALM als Auswahlkriterium, die Beschränkung auf einen Upload-Slot sowie der Verzicht auf zufällig gewählte Peers sorgen dafür, dass immer der aus Sicht der Übertragungszeit kostengünstigste Peer bedient wird. Dies reduziert unnötige Multi-Hop-Übertragungen und vermeidet redundante Mesh-Weiterleitungen derselben Daten über Zwischenknoten, die als Peers auf der Anwendungsschicht ebenfalls an diesen interessiert sind. Die Verwendung des *Generous Mode* fördert darüber hinaus die nichtrestriktive, rein kollaborative Versorgung günstig gelegener Peers.

4.7 Zwischenfazit

Dieses Kapitel beschreibt mit *MeNTor* eine Sammlung von Optimierungen für den BT-basierten kollaborativen Datenaustausch in WLAN-Mesh-Netzwerken nach IEEE 802.11s [B 4, 17]. Es wird ein Cross-Layer-Ansatz verfolgt, der die Integration von Link-Informationen und der Routing-Metrik ALM aus der WLAN-Sicherungsschicht in die BT-Anwendungsschicht vornimmt. Auf diese Weise erhält das BT-Protokoll Informationen über die Kommunikationskosten zu logischen Teilnehmern, auf deren Basis eine optimierte Peer-Auswahl-Strategie umgesetzt wird. Die vorgestellte Lösung nutzt ausschließlich Standardfunktionen der 802.11s-Spezifikation. Als Pflichtmechanismus kann die Verfügbarkeit der ALM auf jedem standardkonformen Mesh-Knoten vorausgesetzt werden. Da ausschließlich inhärente Informationen des 802.11s-Routing-Protokolls HWMP genutzt werden, entsteht zudem kein zusätzlicher Datenverkehr im Netzwerk. Die ALM wird als Prioritätskriterium an den Algorithmus zur Peer-Auswahl gereicht, um das herkömmliche, für Internet-Anwendungen konzipierte Belohnungsprinzip von BT zu ersetzen. Kombiniert mit einer Bevorzugung von Nachbarknoten wird so die robuste Auswahl physisch naher Peers auch in Lastsituationen gewährleistet. Weiterhin werden die Einschränkung simultaner Uploads zu verschiedenen Peers sowie die Vermeidung zufälliger Upload-Ziele vorgeschlagen, um die Anzahl konkurrierender Multi-Hop-Flows zu reduzieren und dadurch insgesamt die Effizienz paralleler Übertragungen im Mesh-Netzwerk zu erhöhen. Der ebenfalls im Rahmen von MeNTor eingeführte *Generous Mode* bewirkt zudem ein für Mesh-Szenarien geeignetes, rein kollaboratives Upload-Verhalten von BT-Peers.

Im realen Versuchsaufbau *Mini-Mesh*, unter Nutzung einer Gitteranordnung aus 25 Mesh-Knoten, wurden 18 verschiedene Konfigurationsvarianten des MeNTor-Prototypen in 10 Beispieltopologien verglichen. In den praktischen Untersuchungen stellte sich erwartungsgemäß die Kombination aller Optimierungen als beste Konfiguration heraus. Die erzielte Verbesserung ist dabei topologieabhängig und steigt mit der Teilnehmerzahl und -dichte des P2P-Overlays. Für die in der Gitteranordnung maximale BT-Schwarmgröße von 25 Peers reduzierte MeNTor die mittlere Download-Zeit bereits um mehr als 40 % gegenüber Standard-BT.

5 Clustering und Kanalwahl in IEEE-802.11s-Netzwerken

5.1 Motivation

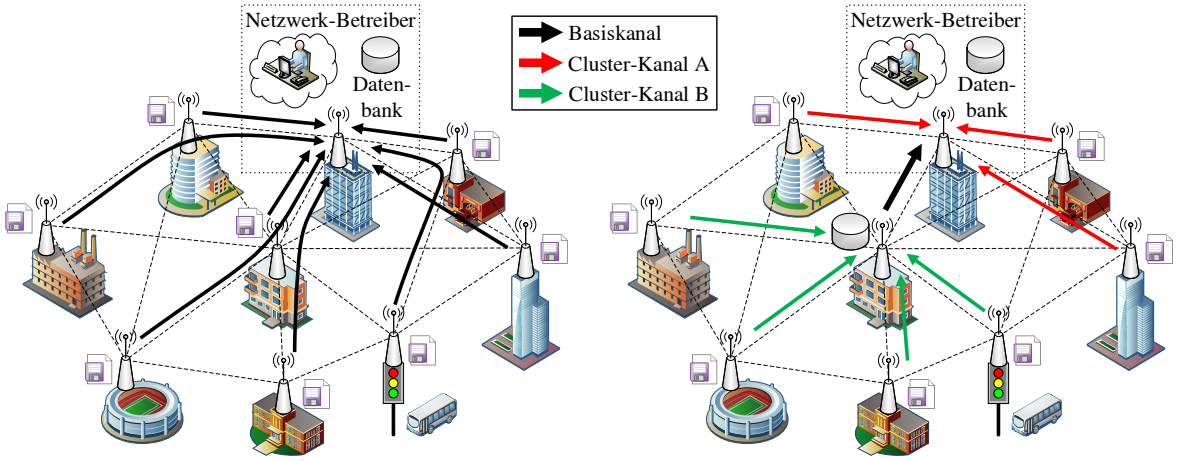
Die Standarderweiterung IEEE 802.11s [5, 12] integriert grundlegende Mesh-Funktionen direkt in die Wireless Local Area Network (WLAN)-Sicherungsschicht und liefert damit die Basis für interoperable Mesh-Lösungen zur Realisierung von Smart-City-Anwendungen [3, 8, 216]. Dazu definiert 802.11s Pflichtmechanismen für die Spontanvernetzung und Nachrichtenweiterleitung (Routing) zwischen Mesh-Knoten. Oberhalb der standardisierten Grundfunktionen existieren jedoch zahlreiche Anforderungen und Problemstellungen, um den Betrieb von 802.11s-Mesh-Netzwerken gerade bei zunehmender Komplexität effizient und zuverlässig realisieren zu können.

So ergeben sich Skalierbarkeitsbeschränkungen von WLAN-Mesh-Netzwerken aus ihrer naturgemäß begrenzten effektiven Kanalauslastung [6, 15, 16]. In Funkreichweite befindliche Geräte auf demselben Kanal konkurrieren um den exklusiven Zugriff auf das Medium und bilden dadurch eine *Kollisionsdomäne*. Die Gesamtheit aller auf dem gleichen Kanal verbundenen Teilnehmer wird gesondert als *Broadcast-Domäne* bezeichnet, da diese beim Versand von Broadcast-Nachrichten vollständig durchlaufen wird (siehe Kap. 2.4.5). Doch auch für die Unicast-Nachrichtenweiterleitung über mehrere Hops wird die entsprechende Anzahl individueller Sendevorgänge benötigt, sodass ein Mesh-Pfad mehrere Kollisionsdomänen traversiert. Dies geht einher mit dem Overhead-Datenverkehr der WLAN-Sicherungsschicht, der z. B. durch periodische Ankündigungen (Beacons) der Mesh-Knoten oder im Rahmen des Routing-Protokolls erzeugt wird. Folglich erhöhen sich mit der Teilnehmerzahl und durchschnittlichen Pfadlänge im Netzwerk unweigerlich auch die Kommunikationslatenz und Übertragungsfehlerwahrscheinlichkeit.

Diese Beschränkungen bestimmen maßgeblich die Leistungsfähigkeit von Diensten und Anwendungen im Mesh-Netzwerk. Sie müssen daher im Rahmen der Orchestrierung und Verwaltung des Netzwerks berücksichtigt und behandelt werden. Eine etablierte Methodik zur skalierbaren Netzstrukturierung und der Ansatz vieler Forschungsarbeiten ist die Partitionierung des Mesh-Netzwerks in unabhängige Regionen, sogenannte *Cluster*. Dies geschieht z. B. durch Wahl geeigneter *Cluster Heads*, die jeweils nur für die Verwaltung ihres Clusters verantwortlich sind [36–38]. In Kombination mit Clustering-Ansätzen können Kanalwahlstrategien eine effizientere Ausnutzung des verfügbaren Frequenzspektrums bewirken [39–41]. Die WLAN-Technologie nutzt vorwiegend Kanäle in lizenzfreien *Industrial, Scientific and Medical (ISM)*-Bändern im Bereich von 2,4 und 5 GHz. Benachbarte Cluster auf überlappungsfreien Kanälen befinden sich ungeachtet ihrer räumlichen Nähe stets in getrennten Kollisionsdomänen. Dazu müssen die Cluster einen ausreichenden Frequenzabstand aufweisen, der abhängig von der gewählten Kanalbandbreite ist. Im Ergebnis kann der Medienzugriff in benachbarten Clustern ohne gegenseitige Beeinflussung und damit vollständig parallel erfolgen.

Ein kombinierter Ansatz zur Cluster-Bildung und Kanalwahl verspricht eine effizientere Nutzung der Kommunikationsressourcen durch verteilte Anwendungen, insbesondere wenn sich diese den entstehenden Clustern geeignet zuordnen lassen. In einer Vorarbeit des Autors wurde eine zentralisierte Monitoring-Lösung für 802.11s-Netzwerke entwickelt, bei der ein dedizierter Knoten die Statusinformationen aller Teilnehmer einholt [B 5, 18]. Mit steigender Netzwerkgröße und Länge der Mesh-Pfade werden die Kommunikationsvorgänge der Anwendung jedoch zunehmend ineffizient. Zudem stellt der zentrale Knoten einen Flaschenhals und Single Point of Failure (SPoF) dar. Praktische Untersuchungen haben gezeigt, dass ein dezentraler Ansatz notwendig ist, um die Skalierbarkeit und Ausfallsicherheit der Monitoring-Lösung zu gewährleisten [B 16] [C 21, 22]. So könnte die Aufgabe des Status-Monitorings aller Knoten dezentral durch mehrere Cluster Heads realisiert werden.

Das Beispiel in Abb. 5.1 stellt eine zentralisierte Datenabfrage (Teilabb. (a)) einem dezentralen Ansatz (Teilabb. (b)) gegenüber, der Cluster auf überlappungsfreien Kanälen nutzt. Hierbei fungiert der



(a) Zentralisiertes Monitoring, Single-Channel

(b) Dezentrales Monitoring, Multi-Channel

Abbildung 5.1: Verteiltes Status-Monitoring im Smart-City-Szenario

mittig positionierte Knoten als zweiter Cluster Head, der die Informationen für seinen Teil des Netzwerks (grüne Übertragungen) parallel zur Datenabfrage des administrativen Zugangsknotens (rote Übertragungen) einholt. Unter Annahme von mindestens zwei WLAN-Interfaces pro Mesh-Knoten, welche eine parallele Kommunikation auf separaten Kanälen innerhalb des Backbones ermöglichen, können die im Cluster gesammelten Daten auf einem unabhängigen Basiskanal synchronisiert werden. Gegenüber dem zentralisierten Fall verkürzen sich die Pfadlängen der Übertragungen bei gleichzeitig erhöhter Kommunikationsparallelität und Ausfallsicherheit. Bislang sind jedoch keine auf 802.11s zugeschnittenen Clustering-Lösungen bekannt, die dessen Standardmechanismen unverändert berücksichtigen und gezielt nutzen.

Dieses Kapitel beschäftigt sich daher mit dem Entwurf eines verteilten Clustering-Ansatzes, der Kompatibilität zur Standard-802.11s-Spezifikation bewahrt. Die entwickelte Lösung *Clustering Heuristic and Channel Assignment (CHaChA)* integriert die Link- und Pfadinformationen der 802.11s-Sicherungsschicht, um aus diesen Topologieinformationen abzuleiten und direkt für die Cluster-Bildung zu nutzen. Der Ansatz wird als Software-Prototyp umgesetzt und mittels der in Kap. 3 vorgestellten Testumgebung *Mini-Mesh* praktisch untersucht. Am Anwendungsbeispiel der Statusüberwachung des Mesh-Netzwerks werden die sich durch das Clustering ergebenden Performance-Vorteile aufgezeigt. Die vorgestellten Ergebnisse wurden international veröffentlicht [B 3]. Die Thematik war zudem Bestandteil mehrerer studentischer Arbeiten, die durch den Autor betreut wurden [C 6, 7, 11, 17, 18].

5.2 Stand der Technik

Frühere Arbeiten am Institut für angewandte Mikroelektronik und Datentechnik widmeten sich dem Clustering von ressourcenbeschränkten Wireless Sensor Networks (WSNs) zur Steigerung der Energieeffizienz und Verlängerung der Lebensdauer des Netzwerks. Ziel war hier, eine selektive Abschaltung von Knotengruppen zu ermöglichen, unter der Annahme, dass nur ein aktiver Sensor pro Region benötigt wird. Handy et al. [217] beschreiben eine Erweiterung des rundenbasierten, stochastischen Clustering-Verfahrens *LEACH*, welche das Energiebudget von Knoten bei der zeitlichen Rotation ihrer Zuständigkeit als Cluster Head (CH) berücksichtigt. You et al. [218] und Salzmann et al. [219] diskutieren die Unterteilung eines Sensorfelds in Zonen, deren Abdeckung nur durch jeweils einen Knoten für die Aufnahme und Weiterleitung von Sensorwerten sichergestellt werden muss. Die vorgestellten Verfahren benötigen jedoch Wissen über die Knotenverteilung anhand von Referenzknoten

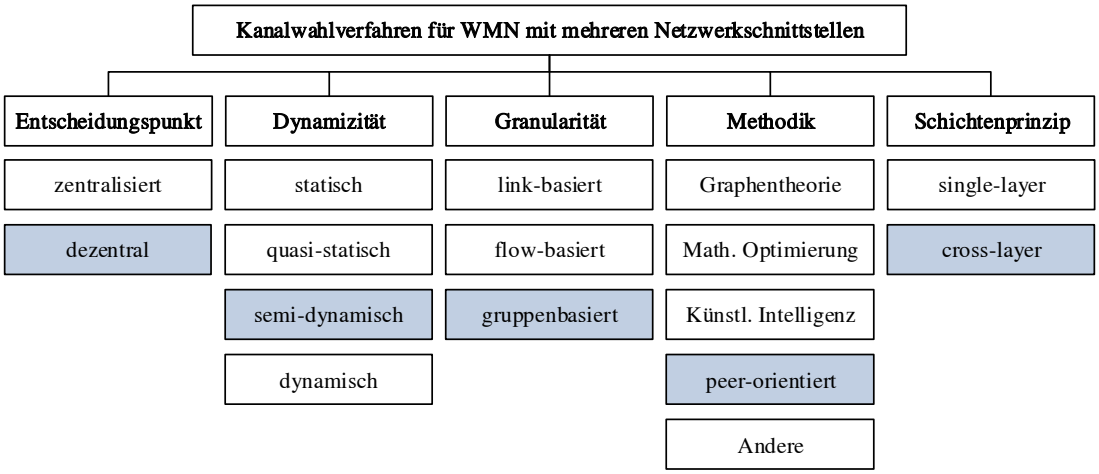


Abbildung 5.2: Klassifikation von Kanalselektionsverfahren für WMNs nach [39]
(Einordnung des eigenen Ansatzes CHaChA ist farblich hervorgehoben)

mit bekannter Position. In [220, 221] präsentieren Salzmann et al. einen broadcastbasierten, lokalisierungsfreien Ansatz. Beginnend mit vorausgewählten CHs wird eine Abstandsschätzung zwischen Knoten durch Senden von Nachrichten mit variierter Sendeleistung vorgenommen. Cluster entstehen durch Beitritt von Knoten zum dichtesten CH, wonach die Bildung weiterer CH an Cluster-Grenzen erfolgen kann. Dieser Ansatz erfordert die Fähigkeit zur homogenen Steuerung der Sendereichweite durch alle Knoten. Die genannten Arbeiten verfolgen mit der Suche einer minimalen Auswahl aktiver Sensoren und der Abschaltung redundanter Knoten in WSNs eine andere Zielstellung als die Verteilung und Performance-Steigerung von Kommunikationsanwendungen in WLAN-Mesh-Netzwerken. Aufgrund des begrenzten Energiebudgets ist nur eine Funkschnittstelle pro Knoten vorhanden und es wird keine Kanalwahl diskutiert. Alle Ansätze wurden zudem nur simulativ untersucht.

Im Bereich des Managements von Wireless Mesh Networks (WMNs) existieren verschiedene Forschungsarbeiten, die eine dezentrale Organisation des Netzwerks durch Partitionierung in logische Teilnetze (Cluster) vorschlagen, jedoch nicht den Aspekt der Kanalwahl für die entstehenden Knotengruppen behandeln. Die Motivation liegt vorrangig in der Verteilung von Management-Aufgaben zur Erhöhung ihrer Skalierbarkeit. Mit *ANMP* (Ad-Hoc Network Management Protocol) [222] und *Guerrilla* [223] werden graphentheoretische Verfahren für hierarchisches Clustering beschrieben mit dem Ziel, den durch Statusüberwachung der Knoten entstehenden Datenverkehr zu reduzieren. Als Monitoring-Protokoll wird jeweils das Simple Network Management Protocol (SNMP) angenommen. Eine weitere Lösung namens *ABCP* (Access-Based Clustering Protocol) [224] benötigt starke Anpassungen auf der WLAN-Sicherungsschicht. Mit *Mesh-Mon* [225] wurde ein broadcastbasiertes Clustering-Protokoll entwickelt, das ebenfalls der Verteilung einer SNMP-Monitoring-Lösung dient. *MeshMan* [226] definiert ein weiteres Protokoll auf Grundlage des Internet Control Message Protocol (ICMP), das eine hierarchische Adressierung und Gruppierung von Knoten vornimmt. Keine der genannten Arbeiten berücksichtigt die WLAN-Mesh-Spezifikation 802.11s. ANMP, Guerrilla und ABCP wurden zudem nur simulativ untersucht.

Im Forschungsfeld der Kanalselektion für WLAN-Mesh-Umgebungen existieren in Form von gruppenbasierten Ansätzen größere methodische Gemeinsamkeiten. Die Surveys [39–41] geben einen Überblick über die Vielzahl an Beiträgen in diesem Bereich. Zwar werden keine Ansätze diskutiert, die 802.11s berücksichtigen, das Survey [39] präsentiert jedoch eine hilfreiche Klassifikation von Kanalwahlstrategien für WMNs mit mehreren Netzwerkschnittstellen pro Gerät. Diese ist in Abb. 5.2

dargestellt, die Einordnung der eigenen Arbeit *CHaChA* ist farblich hervorgehoben. Nach [39] werden folgende Kriterien unterschieden:

Entscheidungspunkt: Eine grundlegende Eigenschaft ist der Entscheidungspunkt für die Kanalwahl. Es werden zentralisierte und dezentrale Verfahren unterschieden, abhängig davon, ob die Zuständigkeit bei einer dedizierten Instanz (meist mit globalem Netzwerkwissen) oder bei vielen bzw. allen Knoten liegt. Verteilte Ansätze besitzen in der Regel eine höhere Skalierbarkeit und Ausfallsicherheit. Zudem sind sie unabhängig von globalem Wissen und können lediglich unter Nutzung lokaler Informationen realisiert werden, wie z. B. Verbindungen zu Nachbarknoten oder Weiterleitungsregeln zu ausgewählten Endpunkten [40].

Dynamizität: Ein zweites Kriterium ist die Dynamizität, d.h. die Fähigkeit, auf Änderungen im Netzwerk reagieren zu können. Diese reicht von rein statischen und quasi-statischen Verfahren, die eine Konfiguration nur einmalig bei Inbetriebnahme des Netzwerks oder regelmäßig in großen Zeitabständen durchführen, über semi-dynamische Verfahren, die innerhalb kürzerer Perioden oder auf bestimmte Ereignisse hin neue Entscheidungen treffen können, bis hin zu vollständig dynamischen Strategien, die auch den Umgang mit mobilen Knoten erlauben.

Granularität: Die Granularität des Verfahrens definiert, ob die Wahl verschiedener Kanäle für einzelne Verbindungen (Links), Ende-zu-Ende-Übertragungen (Flows) oder Knotengruppen erfolgt. Zur letzten Kategorie zählen auch Clustering-Ansätze, die einen Kompromiss zwischen dem durch feingranulare Kanalseparation erreichbaren Performance-Gewinn und dem durch häufige Rekonfiguration verursachten Nachrichten-Overhead darstellen.

Methodik: Bestehende Verfahren unterscheiden sich insbesondere in der Art der Lösung des Kanalwahlproblems. Zentralisierte Ansätze basieren oft auf Graphentheorie (z. B. Graphfärbung), mathematischen Optimierungsverfahren (z. B. lineare Programmierung) oder künstlicher Intelligenz (z. B. evolutionäre Algorithmen). Sie benötigen meist globales Wissen über das Netzwerk und sind oft mit hohem Rechenaufwand verbunden. Andererseits arbeiten viele dezentrale Ansätze peer-orientiert und verwenden vorwiegend lokale Verbindungsinformationen der Knoten.

Schichtenprinzip: Ein weiteres Unterscheidungsmerkmal von Kanalwahlstrategien ist, ob sie nur auf einer einzigen Schicht des Netzwerkprotokollstapels angesiedelt sind (Single-Layer-Verfahren) oder schichtenübergreifend funktionieren (Cross-Layer-Verfahren). Letztere bieten durch die Nutzung von Informationen verschiedener Schichten oft ein höheres Optimierungspotential. Abhängig von der Verfügbarkeit praktischer Schnittstellen für den Informationsaustausch ergeben sich jedoch möglicherweise größere Herausforderungen hinsichtlich der Implementierung der Verfahren und ihrer Integration in bestehende Systeme.

Der Fokus der eigenen Arbeit CHaChA bestand in der Entwicklung einer *gruppenbasierten* Kanalselektionslösung in Form eines Clustering-Ansatzes, die gleichzeitig das Ziel der Anwendungsverteilung und der effizienteren Ausnutzung des Frequenzspektrums verfolgt. Der *dezentrale*, *semi-dynamische* und *peer-orientierte* Ansatz ist auf der Anwendungsschicht implementiert und integriert nach dem *Cross-Layer*-Prinzip lokale Verbindungsinformationen der 802.11s-Sicherungsschicht, ohne Veränderungen am Netzwerkprotokollstapel vorzunehmen. Eine weitere wichtige Zielstellung war die prototypische Implementierung zum Nachweis der Praxistauglichkeit.

Tab. 5.1 stellt CHaChA verwandten Forschungsbeiträgen gegenüber, die ebenfalls kombinierte Ansätze für Clustering und Kanalselektion darstellen. Ältere Lösungen ohne Integration von 802.11s, die bereits in den Surveys [39–41] diskutiert werden, sind ebenso aufgeführt wie neuere Beiträge, die 802.11s zumindest in vorläufiger oder modifizierter Form berücksichtigen. Die Ansätze sind gruppiert nach der Konstellation folgender Anforderungen: die Berücksichtigung von 802.11s, die Dezentralität des Lösungsansatzes sowie seine Untersuchung in einer realen Testumgebung.

Tabelle 5.1: Gegenüberstellung von *CHaChA* mit verwandten Forschungsarbeiten [B 3]
(*: vorläufiges oder modifiziertes 802.11s ohne ALM)

Arbeiten	**802.11s-Berücksichtigung**	**Verteilter Ansatz**	**Reales Testbed**
CCAS [227], ISC [228], MCI [229], MCCA [230]	—	—	—
CCA [231], CoMTaC [232], DCITCA [233], Max-Min-D [234]	—	✓	—
CGCA [36]	*	—	—
Kapse et al. [37], JRCAP [38]	*	✓	—
CHaChA [B 3]	✓	✓	✓

Der Großteil der früheren Arbeiten entstand vor der Spezifikation von 802.11s und nutzt daher nicht dessen Routing-Mechanismen und Verbindungsinformationen. Alle Arbeiten wurden zudem nur simulativ untersucht. Hervorzuheben sind dennoch die Beiträge *CCA* [231] und *ISC* [228], deren Konzepte bzgl. eines dedizierten Basiskanals (Common Channel) sowie der Metrik *Highest Connectivity* auch in CHaChA integriert wurden (siehe Kap. 5.3).

Lediglich die Arbeiten [36–38] betrachten 802.11s in ihren Untersuchungen. Das graphentheoretische Verfahren *CGCA* [36] nutzt zentralisiertes Clustering abhängig von einem Mesh Gateway, das globales Netzwerkwissen durch Kommunikation mit allen Knoten erhält. Der Ansatz benötigt zudem Positionsinformationen der Knoten und schließt jegliche Topologieänderungen aus. Der verteilte Ansatz von Kapse et al. [37] deklariert Mesh Gateways als CHs unabhängig von ihrer topologischen Eignung. Die Cluster-Zugehörigkeit von Knoten wird mittels Hop-Distanz bestimmt. Im Standardfall nutzt das 802.11s-Routing-Protokoll Hybrid Wireless Mesh Protocol (HWMP) jedoch die Metrik Airtime Link Metric (ALM), die nicht nur die Länge eines Pfades, sondern auch dessen mittlere Datenrate und Fehlerwahrscheinlichkeit berücksichtigt. Der Beitrag *JRCAP* [38] präsentiert eine Kombination aus dezentraler, clustering-basierter Kanalwahl und Routing-Protokoll. Das HWMP und die Metrik ALM werden ersetzt, sodass keine Interoperabilität mit Standard-802.11s besteht. Zur Realisierung des eigenen Routing-Protokolls wird ein Mesh Gateway vorausgesetzt. Da keine Grundkonnektivität über einen Basiskanal sichergestellt wird, sind zudem Relay-Knoten zwischen Clustern erforderlich. Im Rahmen von CHaChA werden jedoch Teilkonzepte von JRCAP zum Cluster-Roaming berücksichtigt (siehe Kap. 5.3). Die Ansätze [36–38] wurden nur simulativ oberhalb vorläufiger 802.11s-Modelle untersucht und integrieren keine Informationen des HWMP.

Im Gegensatz zu den bestehenden Arbeiten stellt CHaChA eine verteilte Lösung dar, die gleichzeitig Interoperabilität mit Standard-802.11s bewahrt und dessen lokale Verbindungsinformationen direkt über praktisch vorhandene Betriebssystemschnittstellen integriert. Darüber hinaus wurde CHaChA als Prototyp in einer realen Testumgebung untersucht.

5.3 Clustering-Verfahren *CHaChA*

5.3.1 Überblick und Terminologie

Der vorgestellte Ansatz *Clustering Heuristic and Channel Assignment (CHaChA)* umfasst einerseits Konzepte für die initiale Cluster-Bildung und Kanalzuweisung bei Inbetriebnahme eines 802.11s-Netzwerks. Andererseits werden Mechanismen für die automatische Cluster-Anpassung bei späteren Änderungen der Netzwerkstruktur vorgestellt. Im vorgestellten Konzept wird von Knoten mit

Tabelle 5.2: Knotenrollen in *CHaChA* [B 3]

Knotenrolle	**Abkürzung**	**Beschreibung**
Cluster-Free Node	CFN	Startrolle, Knoten ohne Cluster-Zugehörigkeit
Cluster Member	CM	Teilnehmer eines Clusters
Cluster Head	CH	Organisator eines Clusters
Proposed Cluster Head	PCH	vorübergehender CH-Kandidat im Phasenablauf
Master Cluster Head	MCH	Phasenkoordinator und zugleich erster gewählter CH

mehreren physischen WLAN-Interfaces ausgegangen, von denen mindestens zwei für die Nutzung innerhalb des Mesh-Netzwerks zur Verfügung stehen. Dies stellt eine für moderne WLAN-Geräte, wie sie insbesondere in komplexeren Backbone-Installationen zu erwarten sind, übliche Hardware-Ausstattung dar [39]. Von den Interfaces verbleibt eines als dediziertes Primär-Interface auf einem vorgegebenen Basiskanal für die cluster-übergreifende Kommunikation, während ein sekundäres Interface der cluster-internen Kommunikation vorbehalten ist. Dieses auch als *Common Channel* bekannte Prinzip wird in vielen Forschungsarbeiten genutzt, um unabhängig von der Cluster-Konfiguration jederzeit uneingeschränkte Konnektivität im Netzwerk zu gewährleisten [38, 228]. Da über den Basiskanal stets alle Knoten erreichbar sind, dient das Primär-Interface dem Austausch jeglicher CHaChA-Kontrollnachrichten. Zudem bilden seine 802.11s-Verbindungsinformationen die Grundlage, um Topologieinformationen und Metriken für die Cluster-Bildung abzuleiten. In diesem Zusammenhang gilt die Grundannahme, dass die physischen Parameter der Sekundär-Interfaces ähnlich zu denen des Primär-Interfaces sind und getroffene Clustering-Entscheidungen somit übergreifend anwendbar sind.

Die initiale Cluster-Bildung und Kanalzuweisung bei Inbetriebnahme des Mesh-Netzwerks werden in einer Sequenz von Phasen realisiert. Im Phasenablauf nehmen die Mesh-Knoten verschiedene Rollen ein, welche in Tab. 5.2 zusammenfasst sind. Abb. 5.3 zeigt den Überblick aller Schritte. Eine detaillierte algorithmische Beschreibung folgt in Kap. 5.3.3.

Die initiale Cluster-Bildung erfolgt in den Phasen 0–4, welche das Netzwerk unter Berücksichtigung der Mesh-Topologie in überlappungsfreie Regionen unterteilen. Von allen zu Beginn als *Cluster-Free Nodes (CFNs)* startenden Knoten bilden sich im Zuge der Phasen temporäre *Proposed Cluster Heads (PCHs)* heraus, die als Kandidaten um die finale Rolle als *CH* konkurrieren. Jedes Cluster besteht aus einem leitenden CH sowie mehreren *Cluster Members (CMs)* und bildet ein separates Mesh-Netzwerk auf dem Sekundär-Interface seiner Teilnehmer. Gegenüber dem Basisnetzwerk auf dem Primär-Interface stellen die einzelnen Cluster Broadcast-Domänen geringerer Größe dar. Dies birgt das Potential, die Kommunikationslast auf dem Basisnetzwerk durch Verschieben von Anwendungen in geeignete Cluster zu reduzieren.

Die Phasen 5 und 6 realisieren die Cluster-Kanalwahl und Konfiguration der Sekundär-Interfaces der Knoten. Die Trennung in überlappungsfreie Kanäle vermeidet Interferenzen zwischen benachbarten Clustern sowie gegenüber dem Basiskanal und begünstigt dadurch die Parallelität von Übertragungen im gesamten Netzwerk. Das aktuelle Konzept beschränkt sich auf die Einbindung von zwei WLAN-Interfaces pro Knoten. Die beschriebene Strategie lässt sich jedoch analog auf Szenarien erweitern, in denen weitere Interfaces für die Kommunikation im Mesh-Netzwerk zur Verfügung stehen. So könnten Anwendungen innerhalb eines Clusters gezielt verschiedenen Interfaces auf überlappungsfreien Kanälen zugeordnet werden.

In Phase 7 ist die Initialisierung des Netzwerks abgeschlossen und verteilte Anwendungen können von den entstandenen Clustern profitieren. Durch weiterführende Mechanismen kann die entstandene

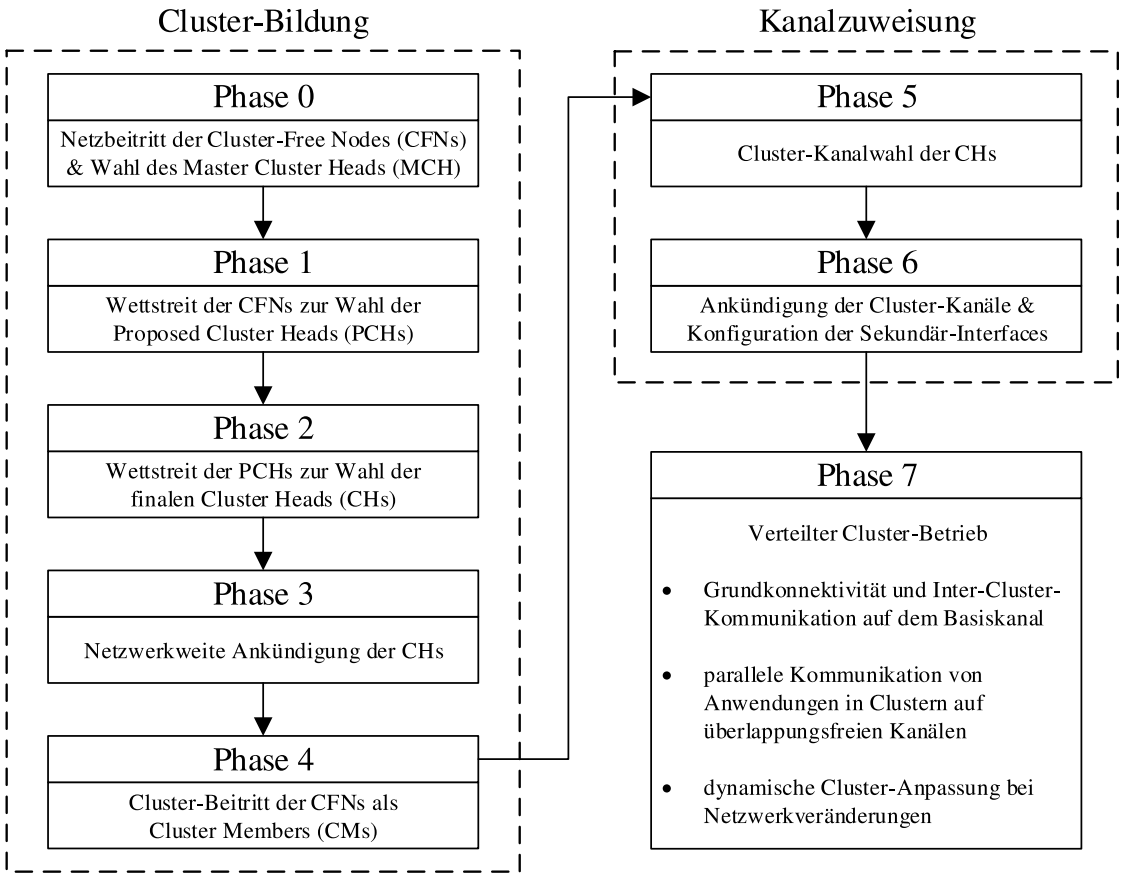


Abbildung 5.3: Phasenablauf der initialen Cluster-Bildung und Kanalzuweisung [B 3]

Ausgangskonstellation auf Verbindungsprobleme und moderate Topologieänderungen reagieren, wie z. B. die Nachinstallation, Umplatzierung oder Entfernung einzelner Knoten im Zuge von Wartungsmaßnahmen. Dabei werden erforderliche inkrementelle Cluster-Anpassungen automatisch durch die CHs bzw. CMs erkannt und durchgeführt, während neu hinzukommende Teilnehmer die bestehenden Cluster finden und ihnen beitreten können. Im Fall der Erkennung gravierender Änderungen der Netzwerkstruktur und Teilnehmerzahl kann jedoch die Einleitung einer erneuten Cluster-Bildung erforderlich sein. Die zugehörigen, über die initiale Phasensequenz hinausgehenden Konzepte werden in Kap. 5.3.4 diskutiert.

Die Koordination des Phasenablaufs übernimmt ein in Phase 0 durch alle Knoten gewählter *Master Cluster Head (MCH)*, der eine möglichst zentrale Position im Netzwerk und somit geringe mittlere Distanz zu den übrigen Knoten besitzt und die Phasenübergänge mittels entsprechender Kontrollnachrichten einleitet. Nach Erreichen von Phase 7 verfällt diese koordinierende Aufgabe des MCH und er verbleibt als herkömmlicher CH im Netzwerk. Die Design-Entscheidung für einen durch Kontrollnachrichten gesteuerten Phasenablauf bietet den Vorteil einer zeitlichen Synchronisation der Mesh-Teilnehmer, ohne dass diese eine präzise gemeinsame Zeitbasis benötigen. Eine solche wäre beispielsweise für Ansätze aus der Kategorie der Frequenzsprungverfahren erforderlich, in denen sich Knoten in festen Zeitabständen auf Rendezvous-Kanälen zur Aushandlung von Kanalsequenzen treffen müssen [39–41]. Die phasenbasierte Synchronisation erlaubt insbesondere die Stabilisierung der während der jeweiligen Phasendauer ermittelten Topologieinformationen und Metriken, auf deren Grundlage die Entscheidungen zur Cluster-Bildung und Kanalwahl erfolgen. Im Gegensatz zu vollständig asynchronen Verfahren kann der phasenbasierte Ansatz zudem in der Praxis gut nachvollzogen und analysiert werden.

Tabelle 5.3: Metriken in *CHaChA*
(c_a: Airtime-Kosten, O: 802.11-Protokoll-Overhead, B: Frame-Bitgröße,
r: 802.11-Sendedatenrate, e: Frame-Fehlerwahrscheinlichkeit)

Metrik	**Abkürzung**	**Berechnung**
Airtime Link Metric	ALM	pro Hop: $c_a = (O + \frac{B}{r}) \cdot \frac{1}{1-e}$ pro Pfad: $ALM = \sum_{i=1}^{\#Hops} c_{a_i}$
Centrality	CENT	$CENT = (\sum_{i=1}^{\#Pfade} ALM_i)^{-1}$
Network Size	N	$N = 1 + \#(\text{Pfade})$
Neighbor Count	NC	$NC = \#(\text{Links})$
PCH Neighbor Count	PCHNC	$PCHNC = \#(\text{Links zu PCHs})$
NC-to-PCHNC-Ratio	NPR	$NPR = \frac{NC}{(1+PCHNC)\cdot N}$
Weighted NPR	WNPR	$WNPR = NPR \cdot \frac{CENT}{CENT_{max}}$
Integer-Zahlenwert der MAC-Adresse	—	Bsp.: 00:00:00:00:fa:00 $\rightarrow$ 64000

5.3.2 Metriken, Kontrollnachrichten und Parameter

Eine Hauptmotivation von CHaChA ist seine herstellerunabhängige Anwendbarkeit oberhalb von Standard-Hardware und -Software. Als Grundlage dient der unmodifizierte Linux-802.11(s)-Protokollstapel, der Mesh-Interoperabilität bereits auf der Sicherungsschicht gewährleistet. Aufgrund des 802.11s-eigenen Routing-Protokolls HWMP, welches zu den Distanz-Vektor-Verfahren gehört, besitzt jeder Teilnehmer nur unvollständiges Wissen über die Netzwerktopologie (siehe Kap. 2.5.5). Diese Arbeit verfolgt daher einen heuristischen Ansatz, der sich auf die lokalen 802.11s-Verbindungsinformationen der Knoten stützt und aus diesen weitere Metriken ableitet. In CHaChA kommen verschiedene Metriken zum Einsatz, die zunächst unabhängig vom Phasenablauf erläutert werden. Tab. 5.3 gibt eine Übersicht aller Metriken und ihrer Berechnung.

ALM: Die ALM ist die durch 802.11s bereitgestellte Standardmetrik des Routing-Protokolls HWMP. Sie beschreibt die Zeitkosten einer Frame-Übertragung zu einem Zielknoten unter Berücksichtigung der WLAN-Sendedatenrate und Fehlerwahrscheinlichkeit (siehe Kap. 2.5.5). Durch ihre kumulative Berechnung entlang aller Hops eines Pfades repräsentiert sie sowohl dessen Länge als auch Gesamtqualität. In CHaChA dient die ALM als Distanzmaß, beispielsweise für die Bestimmung nahe gelegener CHs.

Centrality (CENT): Die CENT-Metrik ist eine Schätzung der Zentralität einer Knotenposition im Netzwerk. Der Knoten mit größtem CENT-Wert ($CENT_{max}$) erhält eingangs die Rolle des MCH-Knotens. In Mesh-Netzwerken bestehend aus Geräten mit ähnlichen WLAN-Hardware-Eigenschaften (Anzahl der Antennen und unterstützte Modulation and Coding Schemes (MCSs)) ergeben sich die ALM-Kosten eines Pfades maßgeblich aus seiner Hop-Länge, vorausgesetzt, dass sich individuelle Link-Konditionen (z. B. stark unterschiedliche Abstände oder Dämpfungseigenschaften) nicht in einer permanent hohen Fehlerrate und geringen Datenrate einzelner Hops niederschlagen. Folglich kann die Annahme getroffen werden, dass der im Zentrum einer beliebigen Mesh-Topologie positionierte Knoten den geringsten ALM-Durchschnitt aller Pfade zu anderen Knoten besitzt, da er diese über die kürzesten mittleren Pfadlängen erreicht. Nachdem ein Knoten Pfadinformationen zu allen anderen Teilnehmern bestimmt hat, beispielsweise durch kurzzeitige Aktivie-

rung des proaktiven Modus des 802.11s-Routing-Protokolls HWMP (siehe Kap. 2.5.5), kann er seine Zentralität nach der Formel $CENT = (\sum_{i=1}^{\#Pfade} ALM_i)^{-1}$ berechnen.

Network Size N: Als Ergebnis einer Pfadbestimmung im proaktiven HWMP-Modus erhält ein Knoten gleichzeitig Kenntnis über die Anzahl aller Teilnehmer im Netzwerk, nachfolgend bezeichnet mit N. Sie ergibt sich aus der Summe $N = 1 + \#(Pfade)$.

Neighbor Count (NC): Der NC eines Mesh-Knotens ergibt sich aus der Anzahl aktiver Links zu benachbarten Teilnehmern. Knoten mit maximalem NC in ihrer Nachbarschaft können den höchsten Anteil anderer Knoten direkt erreichen und laufen am wenigsten Gefahr, in ihrer Netzwerkregion isoliert zu werden. Aus diesen Gründen werden sie in CHaChA als geeignete Kandidaten für die CH-Rolle angesehen und bewerben sich um diese als vorübergehende Proposed Cluster Heads (PCHs) in einem Wettlaufverfahren. Die NC-Metrik wurde unter ähnlichen Bezeichnungen auch in anderen Forschungsarbeiten vorgeschlagen, wie z. B. *Highest Connectivity (HC)* in [228, 231].

Proposed Cluster Head Neighbor Count (PCHNC): Der PCHNC ist Bestandteil der Berechnung der NC-to-PCHNC-Ratio (NPR) und wird nur von PCH-Knoten ermittelt. Er bezeichnet die Anzahl der Mesh-Nachbarn, die sich momentan aufgrund gleichen Vernetzungsgrades (NC-Metrik) ebenfalls in der PCH-Rolle befinden und um den finalen Status als CH konkurrieren.

NC-to-PCHNC-Ratio (NPR): Die NPR ist Bestandteil der Weighted NC-to-PCHNC-Ratio (WNPR) und ergibt sich als Verhältnis von NC und PCHNC. Alle PCHs berechnen sie nach der Vorschrift $NPR = \frac{NC}{(1+PCHNC)\cdot N}$. Um einen Wertebereich zwischen 0 und 1 zu garantieren, wird die Metrik zusätzlich auf die Netzwerkgröße N normiert. Im Zuge ihres Wettstreits sollen PCHs mit höherer NPR Vorrang erhalten, da sie sich in Regionen mit geringerer Anzahl an CH-Kandidaten befinden. Würde jedoch allein die NPR als Kriterium genutzt werden, wären PCHs in direkter Nachbarschaft zu den äußersten Randknoten des Netzwerks besonders bevorzugt. Da die äußersten Knoten in der Regel einen geringeren Konnektivitätsgrad (NC-Metrik) als ihre weiter im Inneren des Netzwerks gelegenen Nachbarn besitzen, kandidieren sie selbst nicht als PCHs. Um eine unkontrollierte Verlagerung der finalen CHs in die direkte Nachbarschaft des Netzwerkrands zu kompensieren, wird die NPR in Form der WNPR um ein Zentralitätsgewicht ergänzt.

Weighted NC-to-PCHNC-Ratio (WNPR): Alle PCHs bewerben sich mit dieser Metrik um die CH-Rolle und berechnen sie nach der Vorschrift $WNPR = NPR \cdot \frac{CENT}{CENT_{max}}$. Sie ergibt sich somit als Produkt von NPR und Zentralität des PCH. Für die Vergleichbarkeit der WNPR verschiedener PCHs wird deren eigene Zentralität auf die des MCH-Knotens normiert, welche in der Startphase als $CENT_{max}$ ermittelt wurde. Durch den Wichtungsfaktor findet der Wettstreit einen Kompromiss zwischen der Bevorzugung von PCHs in unterbesetzten Regionen und deren Nähe zum MCH im Zentrum des Netzwerks. Dies kompensiert einerseits die aus der reinen NPR folgende Dominanz von PCHs in der Nachbarschaft des Netzwerkrands, welche mit Kandidaten inhärent dünn besetzt ist. Andererseits können verteilte Anwendungen von einer Nähe der CHs zum MCH profitieren, z. B. wenn dieser als administrativer Synchronisationspunkt und zentrale Datensenke auf dem Basiskanal fungiert (siehe Kap. 5.5.4).

Integer-Zahlenwert der MAC-Adresse: Im Fall einer Pattsituation, d.h. zwei miteinander verglichene Metrikwerte sind identisch, gewinnt derjenige Knoten mit dem größeren Integer-Zahlenwert seiner MAC-Adresse. Dies ermöglicht eine schnelle und klare Entscheidungsfindung („tie-breaking“) in den verschiedenen Wettbewerbsphasen von CHaChA. Eine Ausnahme bildet die NC-Metrik, bei der ein Gleichstand ausdrücklich zulässig ist. In der Folge entstehen zunächst mehrere CH-Kandidaten, die anschließend mittels WNPR ausgedünnt werden.

Alle im Rahmen von CHaChA genutzten Kontrollnachrichten sind in Tab. 5.4 aufgeführt. Diese umfassen Unicast-Nachrichten, die vorwiegend zwischen Nachbarknoten ausgetauscht werden, sowie Broadcast-Nachrichten zur Ausbringung von Informationen im gesamten Netzwerk. Im Gegensatz

Tabelle 5.4: Unicast (UC)- und Broadcast (BC)-Nachrichten in *CHaChA* [B 3]

Typ (Opcode)	**UC/BC**	**Beschreibung**
CENT	BC	dient CFNs in Phase 0 zur netzwerkweiten Verbreitung ihrer CENT-Metrik
NC	UC	dient CFNs in Phase 0 zur Ankündigung ihrer NC-Metrik an Nachbarn
PCH	UC	dient PCHs in Phase 1 zur Ankündigung ihrer Rolle an Nachbarn
WNPR	UC	dient benachbarten PCHs in Phase 2 zum Austausch ihrer WNPR-Metrik
CH	BC	dient CHs zur netzwerkweiten Ankündigung ihrer Rolle und Cluster-Informationen (Mesh-ID, Kanal, Liste der CM-MAC-Adressen) ab Phase 3
JOIN	UC	durch CFNs (CMs) gesendete Kontrollnachricht beim Cluster-Beitritt
LEAVE	UC	durch CMs gesendete Kontrollnachricht beim Verlassen eines Clusters
CHAN_SEL	UC	zwischen CHs ausgetauschte Kontrollnachricht zur Kanalwahl in Phase 5; enthält Wertepaare aus CH-MAC-Adresse und Cluster-Kanal
PHASE_X	BC	durch MCH gesendete Ankündigung für Übergang in Phase X (definiert für die Phasen 1–6)
NH2CH	UC	dient CMs ab Phase 7 zur Ankündigung ihrer Next-Hop-to-Cluster-Head (NH2CH)-Informationen an Nachbarn; werden beim Roaming zur Cluster-Balancierung berücksichtigt (siehe Kap. 5.3.4.4)
JOIN_REQ/RESP LEAVE_REQ/RESP	UC	zwischen CMs und CHs ausgetauschte Handshake-Nachrichten zur Koordination von Roaming-Vorgängen ab Phase 7 (siehe Kap. 5.3.4.3)

zu Unicast-Frames überträgt die WLAN-Sicherungsschicht Broadcast-Frames unbestätigt und somit ohne Sendewiederholungen im Fehlerfall. Zur Erhöhung der Zuverlässigkeit sieht das Konzept von CHaChA vor, Broadcast-Nachrichten auf der Anwendungsschicht stets mehrfach zu senden. Dies betrifft die CENT-Nachrichten zur Verbreitung der CENT-Metrik sowie die PHASE_X-Nachrichten des MCH zur Bekanntgabe von Phasenübergängen. Für die ohnehin regelmäßig erzeugten Cluster-Ankündigungsnachrichten der CH-Knoten ist der mehrfache Versand bereits inhärent gegeben. Verschiedene Zeitintervall- und Schwellenwertparameter erlauben darüber hinaus eine Anpassung hinsichtlich Zuverlässigkeit und Zeitverhalten. Tab. 5.5 gibt eine Übersicht aller CHaChA-Parameter.

5.3.3 Initiale Cluster-Bildung und Kanalzuweisung

Nachfolgend wird die initiale Cluster-Bildung ausführlich am Beispiel beschrieben und es erfolgt eine Einordnung aller Nachrichten und Parameter in den Phasenablauf. Eine Ausnahme sind die erst in Phase 7 für weiterführende Konzepte genutzten Nachrichtentypen NH2CH und JOIN_REQ/RESP bzw. LEAVE_REQ/RESP, die im späteren Kap. 5.3.4 erläutert werden.

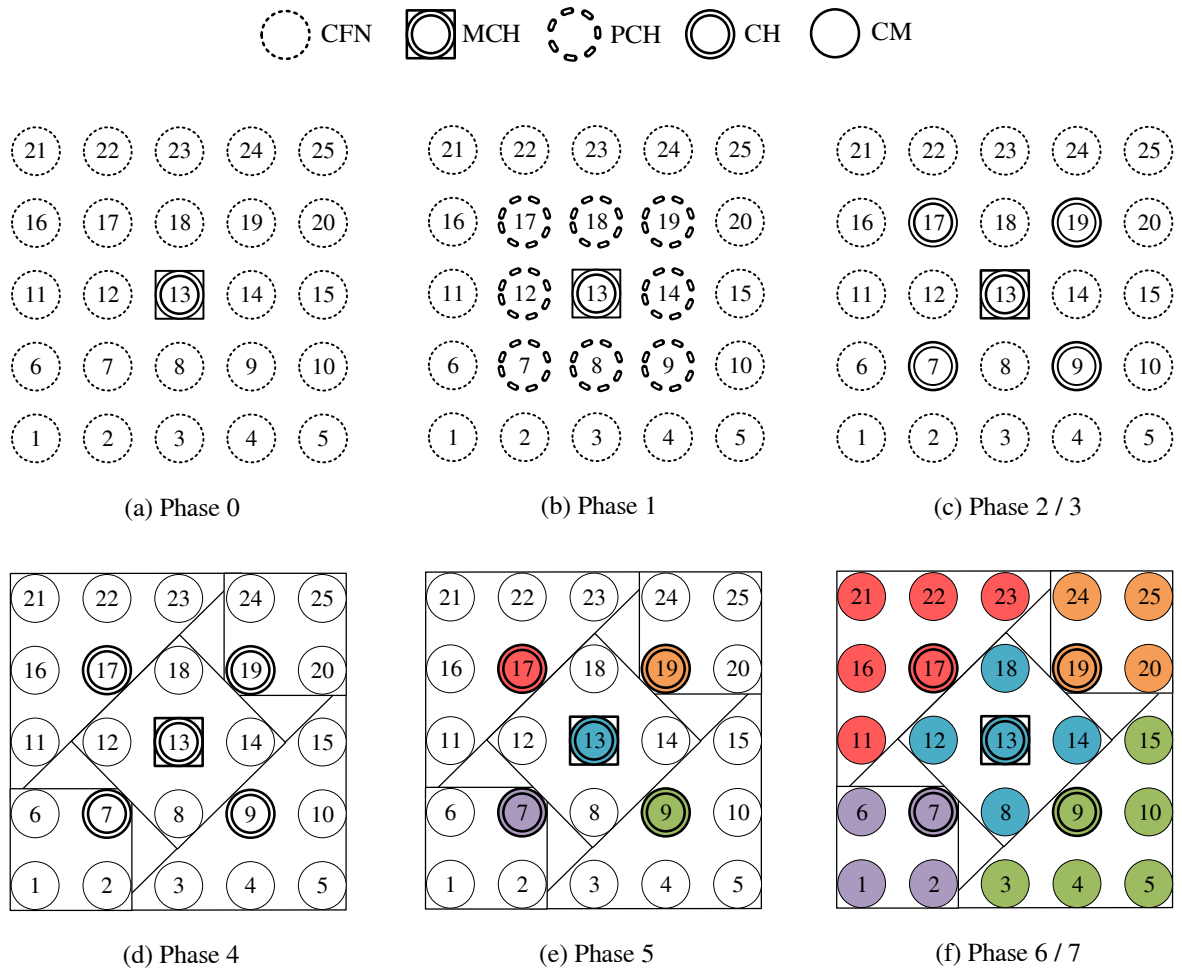
Phase 0 – Netzbeitritt & Wahl des MCH: In Phase 0 starten alle Knoten in der Rolle CFN und sind durch ihr Primär-Interface miteinander verbunden. Grundbedingung für die Ausführung des CHaChA-Algorithmus ist das Vorhandensein mindestens einer aktiven Verbindung zu einem Nachbarknoten auf dem Basiskanal. Nachfolgend wird von einem synchronen Ausführungsbeginn auf allen Geräten ausgegangen, sodass diese gemeinsam die initiale Phasensequenz durchlaufen. Ebenfalls in Phase 0 angesiedelte, weiterführende Mechanismen für den nachträglichen Beitritt in eine bestehende Cluster-Konstellation werden in Kap. 5.3.4.1 diskutiert.

Tabelle 5.5: Timing-Parameter und Schwellenwerte in *CHaChA* [B 3]

Parameter	**Einheit**	**Beschreibung**
CENT_PERIOD	ms	Sendeintervall für CENT-Nachrichten in Phase 0
CENT_THRESH	#	max. Anzahl gesendeter CENT-Nachrichten ohne Empfang selbiger; Überschreitung auf letztem Knoten im MCH-KO-Wettlauf in Phase 0
NC_PERIOD	ms	Sendeintervall für NC-Nachrichten in Phase 0
CH_PERIOD	ms	Sendeintervall für CH-Nachrichten ab Phase 3
CH_THRESH	#	max. Anzahl CH_PERIODs, die Knoten in Phase 0 auf den Empfang von CH-Nachrichten warten (Erkennung bestehender Cluster)
PHASE_X_DELAY	ms	Wartezeit in Phase X-1 bevor der MCH die Phase X ankündigt (definiert für die Phasen 2–5)
PHASE_X_PERIOD	ms	Sendeintervall für PHASE_X-Nachrichten (definiert für die Phasen 1–6)
PHASE_X_TRIES	#	Anzahl Sendewiederholungen für PHASE_X-Nachrichten (definiert für die Phasen 1–6)
PHASE_X_TIMEOUT	ms	Timeout für Übergang zur nächsten Phase (definiert für die Phasen 1–6); Überschreitung führt zu Rücksprung in Phase 0
NH2CH_PERIOD	ms	Sendeintervall für NH2CH-Nachrichten ab Phase 7
CONN_TIMEOUT	ms	Timeout für Verweilen in erkanntem Fehlerzustand (Verbindungsverlust) ab Phase 7; Überschreitung führt zu Rücksprung in Phase 0

Während Phase 0 sendet jeder CFN periodisch im Zeitabstand von NC_PERIOD NC-Unicast-Nachrichten an seine Nachbarknoten. Diese enthalten die aktuelle NC-Metrik, welche alle Knoten später in Phase 1 zur Bestimmung der PCHs vergleichen. Bis zum Erreichen der Phase 3 aktiviert zudem jeder CFN den proaktiven Modus des HWMP, um vorübergehend Mesh-Pfade zu allen Teilnehmern zu ermitteln. Im CHaChA-Konzept werden die Varianten des proaktiven Modus gezielt eingesetzt. An dieser Stelle wird der Effizienteste der drei Modi genutzt („proactive Path Request (PREQ) without Path Reply (PREP)“, siehe Kap. 2.5.5), wodurch sich ein Knoten im Netzwerk ankündigt, jedoch keine Antwort für einen bidirektionalen Pfadaufbau erwartet. Diese Variante ist ausreichend, da alle CFNs den Modus aktivieren und somit unidirektionale Pfadkosten zueinander erhalten. Im Vergleich zum herkömmlichen, reaktiven HWMP-Modus erzeugt dies zwar kurzzeitig mehr 802.11s-Kontrollnachrichten, führt aber zu einem geringeren Overhead als eine Netzwerkerkundung auf höherer Protokollschicht (z. B. mittels Ping Scan) [235, 236].

Im Ergebnis kennt jeder CFN die aktuelle Netzwerkgröße N sowie die ALM-Kosten zu allen anderen Knoten. Dies ermöglicht die Berechnung der CENT-Metrik, welche die Knoten daraufhin mittels CENT-Broadcast-Nachrichten periodisch im Zeitabstand CENT_PERIOD im Netzwerk bekannt geben und sich so um die Rolle des MCH bewerben. Beim Empfang von Nachrichten mit größerer Metrik (bzw. größerer MAC-Adresse bei Gleichstand) entscheiden sich Knoten, vom Wettstreit zurückzutreten und den Versand ihrer Nachrichten einzustellen. Sobald der CFN mit größtem CENT-Wert mindestens CENT_THRESH aufeinander folgende Nachrichten gesendet hat, ohne im Gegenzug welche zu empfangen, übernimmt er automatisch die Aufgabe des MCH. Dieser fungiert später als CH eines zentralen Clusters und koordiniert die verbleibende Phasensequenz. Seine zentrale Position im Netzwerk und somit geringste Durchschnittsdistanz zu allen anderen Knoten erzeugt beim Versand von Kontrollnachrichten den geringsten Frame-Weiterleitungs-Overhead und ermöglicht eine schnelle Ausbringung von Broadcasts. Abb. 5.4 (a) zeigt den Netzwerkzustand am Ende von Phase 0 im Beispiel einer 5x5-Knoten-Gittertopologie. Um allen Knoten den Übergang in Phase 1 anzukün-



(a) Phase 0 (b) Phase 1 (c) Phase 2 / 3

(d) Phase 4 (e) Phase 5 (f) Phase 6 / 7

Abbildung 5.4: Zwischenergebnisse der Phasen im Beispielnetzwerk
(Mesh-Links zwischen Gitternachbarn sind zur vereinfachten Darstellung nicht eingezeichnet;
Cluster-Kanäle der Sekundär-Interfaces werden durch verschiedenen Farben markiert)

digen, sendet der MCH (Knoten 13 im Beispiel) PHASE_1-Broadcast-Nachrichten im Zeitintervall von PHASE_1_PERIOD. Nach einer Anzahl von PHASE_1_TRIES Sendevorgängen wechselt er schließlich selbst in die nächste Phase. Dieses Vorgehen gilt analog für die Phasen 1–6.

Phase 1 – Wahl der PCHs: In Phase 1 erfolgt die Wahl der CH-Kandidaten, der sogenannten PCHs. Zu diesem Zweck vergleichen alle Knoten ihren NC mit den NC-Werten ihrer Nachbarn, die in Phase 0 ausgetauscht wurden. Die Knoten mit höchstem NC in ihrer Nachbarschaft wechseln von der CFN- in die PCH-Rolle, wobei hier ein Gleichstand der Metriken zulässig ist. Dadurch können vorübergehend mehrere benachbarte CH-Kandidaten mit gleichem Konnektivitätsgrad entstehen, welche in der nachfolgenden Phase weiter ausgedünnt werden. Alle PCHs informieren nun ihre Nachbarknoten durch den Versand von PCH-Unicast-Nachrichten über ihre Kandidatur. Abb. 5.4 (b) zeigt das Zwischenergebnis am Ende von Phase 1 im Beispielnetzwerk, in welchem acht PCHs entstehen. Nach dem Ablauf einer als PHASE_2_DELAY definierten Wartezeit (analoge Parametervorgaben für die Phasen 3–5) leitet der MCH mit PHASE_2-Nachrichten den Phasenübergang ein.

Phase 2 – Wahl der CHs: Alle PCHs berechnen zu Beginn dieser Phase ihre WNPR-Metrik und senden sie in Form von WNPR-Unicast-Nachrichten an PCHs in ihrer Nachbarschaft. Die PCHs mit größter Metrik, bzw. größter MAC-Adresse bei Metrik-Gleichstand, gewinnen das Auswahlverfahren und erhalten final den CH-Status. Die unterlegenen PCHs wechseln zurück in die Ausgangsrolle

CFN. Abb. 5.4 (c) zeigt das Zwischenergebnis am Ende von Phase 2 im Beispielnetzwerk mit vier zusätzlich zum MCH entstehenden CHs. Nach Ablauf der Wartezeit PHASE_3_DELAY leitet der MCH den Übergang in Phase 3 ein.

Phase 3 – Ankündigung der CHs: Alle CFNs beenden nun den proaktiven HWMP-Modus und nutzen ab hier das ressourcensparende reaktive Routing. Lediglich der MCH und die in Phase 2 gewählten CHs verbleiben im proaktiven Modus, um sicherzustellen, dass alle anderen Knoten jederzeit Pfadinformationen inklusive ALM-Distanz zu ihnen besitzen. Des Weiteren aktivieren MCH und CHs den periodischen Versand ihrer CH-Broadcast-Nachrichten, um ihre Rolle und Cluster-Informationen auf dem Basiskanal bekannt zu geben. Dazu gehören die eindeutige Mesh-ID sowie der WLAN-Kanal des Clusters, welche beitretende Knoten zur Konfiguration ihrer Sekundär-Interfaces benötigen. Da 802.11s das Mesh-Routing bereits auf der Sicherungsschicht realisiert, ist die Multi-Hop-Kommunikation von Anwendungen auch mit link-lokalen IPv6-Adressen möglich, welche direkt aus den MAC-Adressen der Netzwerk-Interfaces abgeleitet werden können [237]. Aus diesem Grund werden im Rahmen von CHaChA vorerst keine weiterführenden Konzepte für die IP-Konfiguration der Primär- und Sekundär-Interfaces der Knoten diskutiert. In Phase 3, d.h. noch vor dem Cluster-Beitritt der CFNs und der anschließenden Kanalwahl, enthalten die CH-Ankündigungsnachrichten nur die Mesh-ID des Clusters, welche der MAC-Adresse des jeweiligen CH entspricht. Nach der Wartezeit PHASE_4_DELAY leitet der MCH den Übergang in Phase 4 ein.

Phase 4 – Cluster-Beitritt der CFNs: Alle CFNs bestimmen in Phase 4 den hinsichtlich der ALM-Pfadkosten nächstgelegenen CH, dessen Cluster sie beitreten und in die Knotenrolle CM wechseln. Abb. 5.4 (d) zeigt das Zwischenergebnis am Ende von Phase 4 im Beispielnetzwerk. Nach Ablauf der Wartezeit PHASE_5_DELAY initiiert der MCH den Übergang in Phase 5. CFNs priorisieren bei der Wahl des Clusters CHs in ihrer direkten Nachbarschaft. Sollte sich zudem der MCH unter diesen befinden, wird er gegenüber anderen benachbarten CHs bevorzugt gewählt. Diese Strategie impliziert Vorteile für administrative Kommunikationsdienste wie das dezentrale Status-Monitoring aller Knoten. Hier kann der MCH als Synchronisationspunkt in zentraler Position dienen, zu dem eine regelmäßige Übertragung der Informationen der anderen CHs erfolgt und der so eine globale Netzwerksicht bereitstellt. Dabei entfällt der Austausch von Informationen derjenigen Knoten, die bereits zum Cluster des MCH gehören.

Um einem gewählten Cluster beizutreten, sendet ein CFN eine JOIN-Unicast-Nachricht an den entsprechenden CH, der ihn daraufhin in seine CM-Teilnehmerliste aufnimmt. Analog zur JOIN-Nachricht definiert CHaChA eine LEAVE-Nachricht sowie zugehörige Handshake-Nachrichten zur Aushandlung von Cluster-Bei- und -Austritten (JOIN/LEAVE Request bzw. Reply). In Kombination realisieren diese Nachrichten ein Abmelde-/Anmeldeverfahren für den koordinierten Wechsel (Roaming) von CMs zwischen Clustern, das als Teil weiterer Konzepte in Kap. 5.3.4 diskutiert wird.

Phase 5 – Kanalwahl der CHs: In dieser Phase wählen die CHs ihre Cluster-Kanäle für das Sekundär-Interface. Dabei erfolgt die sequentielle Entnahme von Elementen aus einem vordefinierten Pool von paarweise überlappungsfreien Kanälen im 2,4- und 5-GHz-Frequenzband. Für die kleinste Kanalbandbreite von 20 MHz ergeben sich, entsprechend den Regularien für Europa, bis zu drei überlappungsfreie Kanäle im 2,4-GHz-Band sowie weitere 19 Kanäle im 5-GHz-Band (siehe Kap. 2.4.4). Trotz höherer erreichbarer Link-Datenraten der mit 802.11n und 802.11ac eingeführten optionalen Kanalbandbreiten von 40, 80 und 160 MHz werden diese im Konzept von CHaChA bewusst vermieden. Die Nutzung einer Vielzahl schmaler anstatt weniger breiter Kanäle erleichtert insbesondere im Hinblick auf dicht besetzte Mesh-Topologien die Bildung von Clustern mit unabhängigen Kollisionsdomänen, deren Datenübertragungen parallel erfolgen können [39].

In der Praxis können technische und länderspezifische Einschränkungen die Zahl der nutzbaren Kanäle reduzieren. Dabei sind die Sendefilter realer WLAN-Transceiver generell nicht in der Lage, die erzeugte Signalleistung vollständig auf die jeweilige Kanalbandbreite (z. B. 20 MHz) zu

beschränken. Trotz der Einhaltung eines Mittenfrequenzabstands von einer Kanalbandbreite kann es daher, abhängig von der Intensität der eingestreuten Signalleistung, zur sogenannten *Adjacent Channel Interference (ACI)* zwischen benachbarten Zellen kommen [238, 239]. Um diese effektiv zu vermeiden, muss deren Frequenzabstand erhöht oder die Sendeleistung reduziert werden. Darüber hinaus werden Bereiche des 5-GHz-Bandes mit Wetter- und Flugradar gemeinsam genutzt, wobei für WLAN-Geräte eine Erkennungs- und Ausweichpflicht für Radarsignale besteht, die für bestimmte Anwendungsszenarien nicht tolerabel ist. Ebenso kann es zwischen verschiedenen WLAN-Geräten Unterschiede in der Unterstützung der Frequenzbänder geben, die bei der Wahl des Cluster-Kanals berücksichtigt werden müssen. Im Rahmen dieser Arbeit gilt für die Vorgabe des Kanal-Pools die vereinfachte Annahme eines homogenen Netzwerks aus Knoten mit gleichen Hardware-Fähigkeiten.

Die CHaChA-Kanalwahlsequenz wird durch den MCH begonnen, der den ersten Kanal aus dem Pool für sich beansprucht. Anschließend fügt er das Wertepaar aus eigener MAC-Adresse und gewähltem Kanal dem Inhalt einer CHAN_SEL-Unicast-Nachricht hinzu und sendet diese an den hinsichtlich ALM-Pfadkosten günstigsten CH. Dieser Vorgang wird vom jeweils nachfolgenden CH wiederholt bis alle CHs einen Kanal beansprucht haben. Sollte der Kanal-Pool bereits vor Erreichen des letzten CH der Kette aufgebraucht sein, werden Kanäle nach einem klassischen *Greedy*-Verfahren (engl. für *gierig*) mehrfach genutzt. Dazu wählt ein CH erneut den Kanal desjenigen der vorherigen CHs, der zu ihm hinsichtlich ALM am weitesten entfernt ist. Diese Strategie reduziert präventiv Interferenzen zwischen benachbarten Clustern. Der letzte CH der Sequenz sendet eine abschließende Nachricht mit allen MAC-Adress-/Kanal-Wertepaaren zurück an den MCH, der nun durch PHASE_6-Nachrichten den Übergang in Phase 6 auslöst. Abb. 5.4 (e) zeigt das Zwischenergebnis am Ende von Phase 5 im Beispielnetzwerk.

Im aktuellen Konzept erfolgt die Kanalwahl allein auf Basis des vordefinierten Kanal-Pools und es werden noch keine externen WLAN-Zellen berücksichtigt, da dies für die im Labor durchgeführte Leistungsbewertung nicht nötig war. In der Praxis könnten CHs zunächst eine aktive Suche nach umgebenden Netzen durchführen (mittels *Probe-Request*-Nachrichten, siehe Kap. 2.4.3), um bevorzugt Kanäle zu wählen, die auch zu diesen überlappungsfrei sind. Zudem konzentriert sich das Konzept bislang auf die Kanalwahl für Sekundär-Interfaces, die der Kommunikation innerhalb des Mesh-Netzwerks dienen. Die Zuordnung weiterer freier Interfaces als Access Points (APs) zur Anbindung herkömmlicher WLAN-Clients ohne Mesh-Fähigkeit könnte jedoch auf ähnliche Art und Weise erfolgen. Im einfachsten Fall könnten die gewählten Cluster-Kanäle auch durch ein virtuelles AP-Interface geteilt genutzt werden, das zusätzlich auf dem physischen Sekundär-Interface angelegt wird (siehe Kap. 2.5.7). Alternativ ließen sich auf Basis des Wissens über aktuelle Cluster-Kanäle und die Distanz zu anderen Clustern bevorzugt überlappungsfreie Kanäle für weitere physische AP-Interfaces wählen.

Mit dem Ende von Phase 5 enthalten die periodisch gesendeten CH-Broadcast-Nachrichten neben der Mesh-ID des Clusters auch den Cluster-Kanal sowie die Liste der dem Cluster zugehörigen CMs. Die Kanalinformation wird von CMs benötigt, um ihr Sekundär-Interface zu konfigurieren. Die Anzahl und Zusammensetzung der Teilnehmer wiederum sind mögliche Kriterien für eine spätere Cluster-Balancierung. So können CMs im Zuge von Roaming-Aktivitäten bevorzugt in unterbesetzte Nachbar-Cluster wechseln (siehe Kap. 5.3.4).

Phase 6 – Konfiguration der Sekundär-Interfaces: In Phase 6 konfigurieren alle CMs ihr sekundäres WLAN-Interface entsprechend den für ihr jeweiliges Cluster angekündigten Parametern, wodurch die Kommunikation auf dem Cluster-Kanal ermöglicht wird. Abb. 5.4 (f) zeigt das Endergebnis nach Phase 6 im Beispielnetzwerk. Nach der Konfiguration ihrer Sekundär-Interfaces wechseln alle Knoten direkt in Phase 7. Wie bereits auf dem Primär-Interface aktivieren die CHs den proaktiven HWMP-Modus nun auch auf dem Sekundär-Interface. Anders als auf dem Basiskanal nutzen sie hier jedoch die Modusvariante „proactive PREQ with PREP“ (siehe Kap. 2.5.5), die eine bidirektionale Pfaderstellung bewirkt. Dadurch werden zwischen CMs und ihrem CH Pfadinformationen zueinander

vorgehalten und Knotenausfälle können durch Überprüfung des Vorhandenseins von Pfadeinträgen beiderseitig erkannt werden.

Phase 7 – Verteilter Cluster-Betrieb: Mit dem Erreichen der Phase 7 ist das Clustering des Netzwerks abgeschlossen. Netzwerkdienste und -anwendungen, die sich verschiedenen Clustern geeignet zuordnen lassen, profitieren von einer höheren Leistungsfähigkeit, da sie parallel auf überlappungsfreien Kanälen sowie in, verglichen mit dem Basiskanal, kleineren Broadcast-Domänen kommunizieren. Dabei bleibt die uneingeschränkte Konnektivität zwischen den Clustern stets über das Primär-Interface gewährleistet. Durch das Auslagern von Anwendungen in unabhängige Cluster kann gleichzeitig der Basiskanal entlastet werden, sodass dort zusätzliche Kapazität für den clusterübergreifenden Datenaustausch entsteht.

Auf der Applikationsebene ist zu beachten, dass die Primär- und Sekundär-Interfaces der Mesh-Knoten in verschiedenen 802.11s-Netzwerken und somit auch in verschiedenen logischen Netzen mit eigenen IP-Adressbereichen agieren, welche bekannt sein müssen. Das aktuelle Konzept von CHaChA definiert keine Vorgaben, nach denen die Konfiguration dieser Adressbereiche und die Bindung von Anwendungen erfolgen müssen. Im Rahmen der Evaluation in Kap. 5.5 waren allen Mesh-Interfaces feste IP-Adressen zugewiesen und es wurde vereinfachend angenommen, dass Anwendungen über das benötigte Vorwissen verfügen. In der Praxis sind dagegen zahlreiche Strategien denkbar. So könnten automatisch konfigurierte link-lokale IPv6-Adressen der Primär- und Sekundär-Interfaces genutzt werden, welche CHaChA anderen Anwendungen als Teil des Wissens über die Cluster-Zugehörigkeit der Knoten bereitstellt. Alternativ könnten CHs erweiterte Dynamic Host Configuration Protocol (DHCP)- und Domain Name System (DNS)-Funktionen übernehmen, um ein geeignetes Adressierungs- und Namensschema für die verschiedenen Teilnetze zu realisieren.

Fehlerbehandlung im Phasenablauf: Mögliche Fehler im Phasenablauf können die verteilten Knoten selbständig erkennen. So besteht beim KO-Verfahren zur MCH-Wahl in Phase 0 theoretisch die Gefahr, dass infolge andauernder Verluste von CENT-Broadcast-Nachrichten mehrere MCHs entstehen, insbesondere wenn die Parameter CENT_PERIOD und CENT_THRESH zu klein gewählt werden. Bei geeigneter Parametrisierung ist dieses Risiko jedoch sehr gering. Anhand des Empfangs von MCH-Kontrollnachrichten verschiedener Absenderadressen lässt sich ein möglicher Fehlerfall leicht auf jedem Knoten erkennen. Spätere Probleme können anhand eines überschrittenen Timeouts (PHASE_X_TIMEOUT) innerhalb der Phasen festgestellt werden. Ursache hierfür wären z. B. der Ausfall des MCH oder ein Verbindungsabriss von Knoten, wodurch Phasenübergänge nicht rechtzeitig eingeleitet oder in Teilen des Netzwerks nicht empfangen werden. Ebenso könnte die Kanalwahlsequenz in Phase 5 infolge eines CH-Ausfalls unterbrochen werden, sodass ebenfalls kein Wechsel in Phase 6 erfolgt.

Zur Fehlerbehandlung springen Knoten als CFN zurück in Phase 0. Handelte es sich nur um einen temporären Verbindungsabriss zum MCH ohne Beeinträchtigung des Phasenablaufs im restlichen Netzwerk, können sich Knoten nachträglich in eines der entstandenen Cluster eingliedern. Hingegen wird beim Ausfall des MCH (oder einer permanenten Isolation von diesem) in den betroffenen Netzwerkteilen automatisch ein erneutes Clustering gestartet. Diese und weitere Mechanismen zur Behandlung von Fehlern und Netzwerkveränderungen werden nachfolgend in Kap. 5.3.4 diskutiert.

5.3.4 Cluster-Anpassung zur Laufzeit

Im Anschluss an die initiale Cluster-Bildung sind weiterführende Konzepte nötig, um auf Veränderungen der Netzwerkstruktur zu reagieren und Cluster-Anpassungen zur Laufzeit vorzunehmen. Mit Ausnahme von Funktionen zur nachträglichen bzw. erneuten Eingliederung in eine bestehende Cluster-Konstellation, welche bereits in Phase 0 angesiedelt sind, greifen alle Mechanismen erst nach Abschluss des Clustering-Phasenablaufs in Phase 7. Die Erkennung von Verbindungs-

problemen und Topologieänderungen stützt sich einerseits auf CHaChA-Protokollnachrichten der Anwendungsschicht, andererseits auf die Prüfung von Link- und Pfadinformationen der 802.11s-Sicherungsschicht. Die Abb. 5.5 zeigt eine Übersicht aller behandelten Fälle als Zustandsdiagramm.

5.3.4.1 Nachträglicher oder erneuter Cluster-Beitritt

Für die in Kap. 5.3.3 beschriebene initiale Cluster-Bildung bei Inbetriebnahme des Mesh-Netzwerks wurde ein synchroner Ausführungsbeginn von CHaChA auf allen Knoten angenommen. Im praktischen Betrieb eines städtischen Backbone-WMN ist jedoch auch mit einer Nachinstallation oder Umplatzierung von Geräten zu rechnen. Weiterführende Mechanismen dienen daher der nachträglichen Eingliederung von Knoten, ohne dass eine erneute initiale Cluster-Bildung durchgeführt werden muss. Hierbei handelt es sich um eine Erweiterung der Phase 0 des CHaChA-Phasenablaufs (siehe Einsprungpunkt des Zustandsdiagramms in Abb. 5.5).

Vorausgesetzt, dass mindestens ein aktiver Mesh-Link besteht, prüft ein CFN zu Beginn von Phase 0 zunächst über die Zeitdauer {CH_THRESH · CH_PERIOD}, ob auf dem Basiskanal CH-Broadcast-Nachrichten mit vollständigen Cluster-Informationen (Mesh-ID und Kanal für das Sekundär-Interface) gesendet werden. Diese zeigen an, dass eine initiale Cluster-Bildung bereits durchgeführt wurde und der CFN nachträglich hinzugekommen ist. Alternativ signalisiert der Empfang von MCH-Phasenankündigungen ein noch im Vorgang befindliches Clustering, auf dessen Abschluss der CFN warten muss. Für den Cluster-Beitritt sendet er, wie in Kap. 5.3.3 für Phase 4 beschrieben, eine JOIN-Nachricht an den CH in geringster ALM-Distanz. Auch hier werden CHs in direkter Nachbarschaft bevorzugt gewählt. Nach der Konfiguration seines Sekundär-Interfaces entsprechend der durch den CH angekündigten Cluster-Informationen (Mesh-ID und Kanal) wechselt der CFN als CM direkt in Phase 7.

Sind in Phase 0 weder CH-Nachrichten noch Phasenankündigungen zu verzeichnen, wird mit dem in Kap. 5.3.3 beschriebenen Phasenablauf zur initialen Cluster-Bildung fortgefahren. Kommt es darin zu Fehlern, erkannt anhand der Überschreitung eines Timeouts aufgrund ausbleibender MCH-Phasenankündigungen, springen Knoten zurück in Phase 0. Auch nach erfolgreichem Clustering auftretende Fehler und Verbindungsprobleme werden, wie nachfolgend im Detail beschrieben, durch einen Rücksprung in Phase 0 behandelt. Dort versuchen Knoten eine Wiedereingliederung in das Netzwerk oder starten alternativ einen erneuten Clustering-Phasenablauf.

5.3.4.2 Isolations- und Ausfallerkennung

Nach erfolgreichem Abschluss des Clustering-Phasenablaufs besitzen Knoten entweder die Rolle CM oder CH (siehe Zustandsdiagramm in Abb. 5.5). Anhand der Prüfung des Status ihrer Mesh-Interfaces können CMs und CHs die eigene Isolation sowie gegenseitige Verbindungsabrisse erkennen. Die möglichen Ursachen dafür sind vielfältig und umfassen z. B. dynamische Hindernisse, Wettereinflüsse, Umplatzierung von Geräten im Zuge der Wartung des Mesh-Backbones oder auch Geräteausfälle infolge von Hardware-Defekten oder ausbleibender Energieversorgung.

Darüber hinaus kann es sein, dass CMs ihren CH nach dem initialen Clustering nicht über das Sekundär-Interface erreichen können. Zwar erfolgt der Cluster-Beitritt in Phase 4 bereits mit der Strategie, den bzgl. ALM nächstgelegenen CH auszuwählen. Aufgrund der gleichzeitigen und individuellen Entscheidungsfindung aller Knoten ist jedoch nicht gewährleistet, dass diese stets zusammenhängende Gruppen bilden. Nach Konfiguration der Sekundär-Interfaces der Knoten (Setzen von Kanal und Mesh-ID in Phase 6) kann es daher zu Verbindungsproblemen innerhalb der Cluster kommen. Zwar garantiert das statisch auf den gemeinsamen Basiskanal konfigurierte Primär-Interface höchstmögliche Konnektivität, dennoch muss die Isolation auf dem Cluster-Kanal durch die nachfolgend beschriebenen Mechanismen behandelt werden.

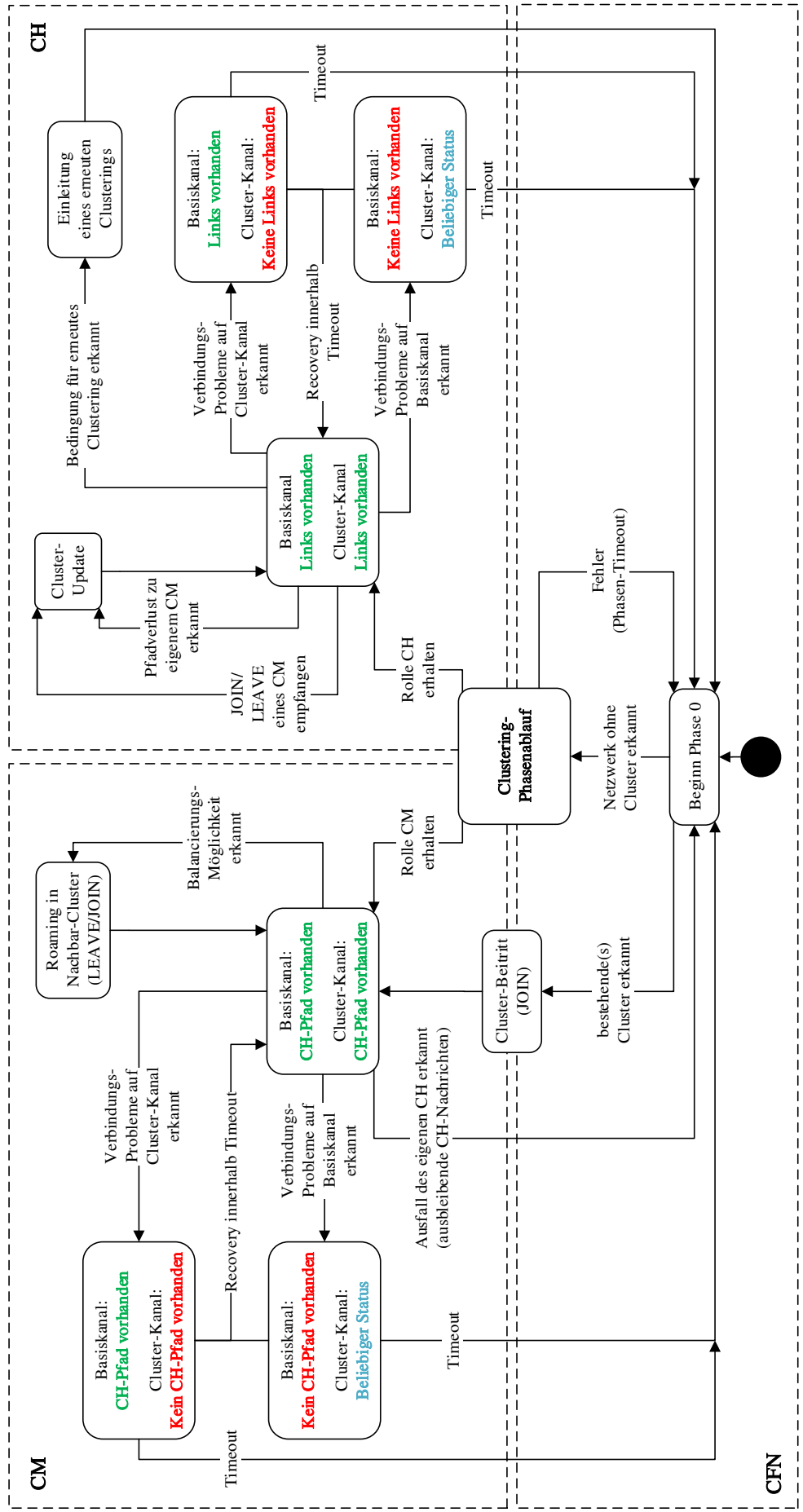


Abbildung 5.5: Zustandsdiagramm der Mechanismen zur dynamischen Cluster-Anpassung (CFN: Cluster-Free Node, CM: Cluster Member, CH: Cluster Head)

In allen genannten Situationen entsteht ein Verlust von Link- und Pfadinformationen auf der 802.11s-Sicherungsschicht. Hierbei existieren verschiedene Konstellationen für den Status des Primär-Interfaces auf dem Basiskanal und des Sekundär-Interfaces auf dem Cluster-Kanal, die abhängig von der Knotenrolle unterschiedlich behandelt werden. In Abb. 5.5 ist der Interface-Status in den entsprechenden Teilzuständen farblich hervorgehoben (grün – keine Verbindungsprobleme, rot – Verbindungsabriss, blau – beliebiger Status).

Behandlung auf CM-Seite: Aus Sicht eines CM bestehen keine Verbindungsprobleme, wenn er seinen CH als Ankerpunkt des Clusters auf beiden Interfaces erreicht. Dies ist gewährleistet, solange 802.11s-Pfadinformationen in Richtung CH vorliegen, und setzt gleichzeitig voraus, dass mindestens ein Link pro Mesh-Interface existiert. Da CHs sowohl auf dem Basiskanal als auch in ihrem Cluster den proaktiven HWMP-Modus nutzen, werden Pfadinformationen zu ihnen periodisch auf allen Knoten aktualisiert. Im Zustand ohne Verbindungsprobleme besteht die Möglichkeit des koordinierten Wechsels (sog. *Roaming*) zwischen benachbarten Clustern mit dem Ziel der Cluster-Balancierung. Diese Mechanismen werden in Kap. 5.3.4.3 und 5.3.4.4 detailliert beschrieben.

Verliert ein CM auf dem Cluster- oder Basiskanal dauerhaft die Verbindung zu seinem CH, erkannt anhand der Überschreitung eines Timeouts (CONN_TIMEOUT) im Zustand ohne Pfadinformation zum CH, springt er als CFN zurück in Phase 0. Hier greifen die Mechanismen zur Wiedereingliederung in das Netzwerk bzw. erneuten Cluster-Bildung (siehe Kap. 5.3.4.1). Auch bei bestehender Mesh-Verbindung können CMs einen funktionalen Ausfall ihres CH anhand fehlender CH-Broadcast-Nachrichten auf der Anwendungsschicht erkennen. CHs senden diese Nachrichten als periodische Heartbeats und zur Ankündigung ihrer Cluster-Informationen. Ein Ausbleiben der Nachrichten signalisiert den Verlust der CH-Rolle, z. B. infolge von Software-Fehlern oberhalb der WLAN-Sicherungsschicht. Auch in diesem Fall wechseln die betroffenen CMs als CFNs zurück in Phase 0.

Behandlung auf CH-Seite: Ein CH trifft die Annahme des fehlerfreien Betriebs, solange beide seiner Mesh-Interfaces verbunden sind. In diesem Zustand existiert je mindestens ein Link zum Basis- und Cluster-Kanal und die Hauptaufgabe des CH besteht darin, auf mögliche Veränderungen seines Clusters zu reagieren. So erfordern Bei- und Austrittsgesuche von CMs, z. B. infolge einer Nachinstallation von Knoten oder im Zuge des Roamings, eine Anpassung der Cluster-Informationen, die der CH mittels CH-Nachrichten im Netzwerk ankündigt. Ebenso wird der Ausfall eines zugehörigen CM erkannt, wenn trotz proaktivem HWMP-Modus auf dem Cluster-Kanal keine Pfadinformationen zu diesem vorliegen.

Seine Isolation erkennt ein CH anhand Überschreitung eines Timeouts (CONN_TIMEOUT) im Zustand ohne Link-Informationen. Analog zu der CM-seitigen Isolationserkennung werden auch auf CH-Seite zwei mögliche Situationen unterschieden: Im ersten Fall besteht bereits auf dem für die CHaChA-Koordination maßgeblichen Primär-Interface keinerlei Verbindung, dabei ist der Zustand des Sekundär-Interfaces unerheblich. Der zweite Fall, der ausschließlich das Sekundär-Interface auf dem Cluster-Kanal betrifft, kann aus verschiedenen Gründen entstehen. Einerseits können CMs ausfallen, sodass ausschließlich Links zu Knoten anderer benachbarter Cluster auf dem Basiskanal existieren. Andererseits können CMs mangels eines verbindenden Zwischenknotens auf dem Cluster-Kanal abgetrennt werden, sodass auch sie nur über Knoten anderer Cluster auf dem Basiskanal erreichbar sind. Da der CH jeweils seine Eignung als koordinierender Referenzknoten verliert, muss die Situation entsprechend behandelt werden. Dazu springt der CH als CFN zurück in Phase 0. Hier erfolgt zunächst der in Kap. 5.3.4.1 beschriebene Versuch der Wiedereingliederung in das Netzwerk oder alternativ eine erneute Cluster-Bildung. Im Gegenzug wird der Verbindungsabriss auch von den CMs des Clusters behandelt. Diese erkennen ihrerseits den Verlust der Pfadinformationen zum CH und versuchen in Phase 0 ebenfalls den Wechsel in ein neues Cluster.

Unabhängig von der Isolations- und Ausfallbehandlung ist zudem die CH-seitige Einleitung einer erneuten Cluster-Bildung denkbar. In Kap. 5.3.4.5 werden Szenarien und Bedingungen diskutiert, die ein solches *Re-Clustering* motivieren könnten.

5.3.4.3 Roaming zwischen Clustern

Der koordinierte Wechsel eines CM von seinem bisherigen in ein benachbartes Cluster wird in CHaChA als *Roaming* bezeichnet. Nach Erreichen von Phase 7 dienen Roaming-Vorgänge der Cluster-Balancierung, die in Kap. 5.3.4.4 genauer erläutert wird. Das nachfolgende Konzept basiert auf einer Grundidee der Arbeit *JRCAP* [38], die balancierte Cluster ebenfalls durch Roaming realisiert. In JRCAP ist ein Wechsel der Cluster-Zugehörigkeit allein *Relay Nodes* vorbehalten. Dabei handelt es sich um Cluster-Randknoten, die benachbarte Cluster miteinander verbinden. Beim Cluster-Wechsel erfolgt zwischen Relay Node und neuem bzw. altem CH eine Aushandlung mittels ATTACH- und DETACH-Nachrichten, die von den CHs jeweils bestätigt werden müssen. Dadurch wird eine CH-seitige Kontrolle von Roaming-Vorgängen ermöglicht. Im Gegensatz zu CHaChA sieht JRCAP jedoch keinen dedizierten Basiskanal vor, über den CHs ihren Cluster-Status im gesamten Netzwerk ankündigen. So ist z. B. die Größe benachbarter Cluster nur Relay Nodes bekannt, welche diese im Zuge der Aushandlung an die CHs weitergeben müssen. Stattdessen wird in CHaChA eine Roaming-Variante vorgeschlagen, die sich auf die Informationen der CH-Broadcast-Nachrichten sowie lokale 802.11s-Verbindungsinformationen stützt.

Wie bei der verwandten Arbeit JRCAP [38] gilt als Voraussetzung für den Cluster-Wechsel eines CM, dass dieser ein Randknoten seines Clusters ist. Zudem muss Verbindung zu mindestens einem benachbarten Cluster existieren. CMs überprüfen dies im CHaChA-Konzept anhand ihrer 802.11s-Link- und Pfadlisten sowie der Cluster-Informationen, die von allen CHs periodisch als Broadcast-Nachrichten auf dem Basiskanal verschickt werden. Darin enthalten ist u.a. die Liste der MAC-Adressen aller CMs des jeweiligen Clusters. Existiert ein Mesh-Pfad zum CH eines anderen Clusters und befindet sich mindestens ein CM dieses Clusters unter den eigenen Mesh-Nachbarn auf dem Basiskanal, handelt es sich um ein Nachbar-Cluster, in das gewechselt werden kann. Existieren mehrere mögliche Roaming-Ziele, wird – analog zum herkömmlichen Cluster-Beitritt in CHaChA – der CH in kleinster ALM-Distanz gewählt.

Eine weitere Roaming-Voraussetzung ist, dass der wechselnde CM auf dem Cluster-Kanal nicht selbst als verbindender Zwischenknoten zwischen seinem CH und anderen CMs agiert. Dies stellt eine konservative Einschränkung dar, welche ignoriert, dass möglicherweise alternative Pfade über andere Knoten entstehen könnten. Dadurch wird jedoch vermieden, dass Cluster infolge von Roaming zerrissen werden und es zu unerwünschten Inselbildungen kommt. Um diesen vorzubeugen, tauschen benachbarte CMs ab Phase 7 sogenannte Next-Hop-to-Cluster-Head (NH2CH)-Nachrichten untereinander aus (siehe auch Tab. 5.4 in Kap. 5.3.2). Diese geben Auskunft darüber, welchen Nachbarknoten („Next Hop“) der Sender der Nachricht momentan auf seinem Pfad zum CH nutzt. Übernimmt ein CM für keinen seiner Nachbarknoten diese Funktion, ist für ihn ein Cluster-Wechsel möglich. Abb. 5.6 illustriert die Beschränkungen anhand eines Beispiels, in dem von vier Randknoten nur die beiden äußeren keine Next-Hop-Funktion übernehmen und daher für einen Wechsel infrage kommen.

Der Roaming-Vorgang eines CM erfolgt als Handover zwischen bisherigem und neuem CH entsprechend der Abb. 5.7. Da CMs den Entschluss zum Roaming unabhängig voneinander und ggf. gleichzeitig treffen, besteht die Gefahr von sprunghaften Cluster-Änderungen und Oszillationsverhalten. Im Zuge einer bidirektionalen Aushandlung (Handshake) entscheiden CHs daher über die Annahme oder Ablehnung von Roaming-Gesuchen. So lässt sich z. B. durch CH-seitige Vorgabe eines minimalen Zeitabstandes zwischen Roaming-Vorgängen eine Beschränkung der Häufigkeit von Cluster-Wechseln realisieren.

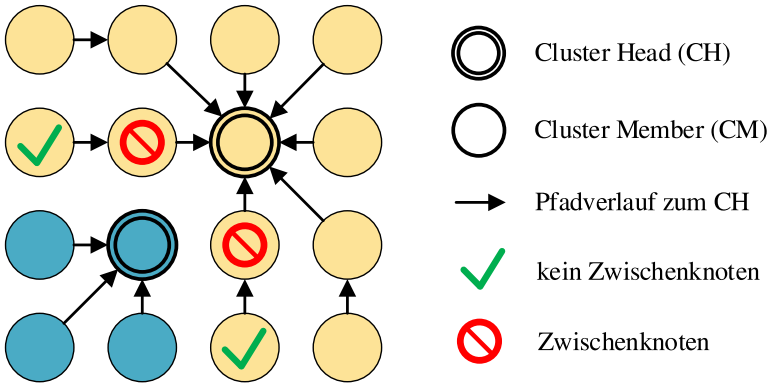


Abbildung 5.6: Roaming-Einschränkungen am Beispielnetzwerk

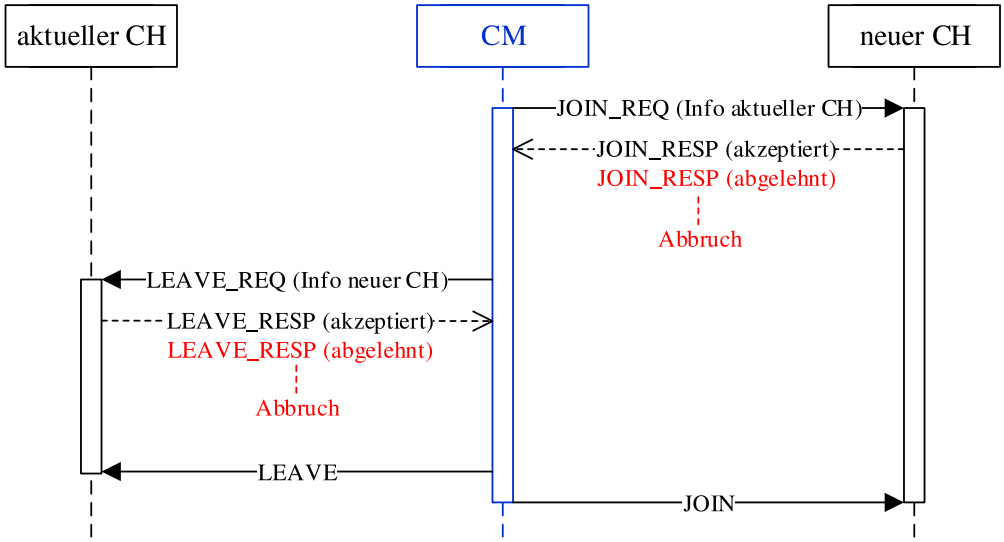


Abbildung 5.7: Sequenzdiagramm für das Roaming zwischen Clustern

Um in das neue Cluster zu wechseln, sendet der CM zunächst ein Beitrittsgesuch an dessen CH in Form einer JOIN_REQ-Unicast-Nachricht. Der CH des neuen Clusters antwortet mit einer JOIN_RESP-Nachricht, um die Anfrage zu bestätigen oder abzulehnen. Bei einem Ausbleiben der Antwort oder einer expliziten Ablehnung terminiert der CM den Roaming-Vorgang. Bei Annahme sendet der CM analog eine LEAVE_REQ-Nachricht an seinen aktuellen CH, der diese mit einer LEAVE_RESP-Nachricht quittiert. Mit einer abschließenden LEAVE-Nachricht verlässt der CM das bisherige Cluster und tritt mittels JOIN-Nachricht dem neuen Cluster bei. Gleichzeitig übernimmt er die Mesh-Konfiguration des neuen Clusters (Mesh-ID und Kanal) auf seinem Sekundär-Interface. Mögliche Verbindungsabrisse zwischen CM und CH nach erfolgter Roaming-Aushandlung könnten eine Nichtzustellung der finalen LEAVE-/JOIN-Nachrichten bewirken. Eine solche Situation würde jedoch durch die zuvor beschriebene Isolations-/Ausfallerkennung nachträglich behandelt werden.

JOIN_REQ- und LEAVE_REQ-Nachricht tragen die MAC-Adresse des Herkunfts- bzw. Ziel-CH, um die beidseitige Entscheidung über Annahme bzw. Ablehnung von CMs zu ermöglichen. Zwar kann die Cluster-Zugehörigkeit von CMs prinzipiell auch aus den Informationen der periodischen Ankündigungsnachrichten der CHs gewonnen werden. Diese werden auf dem Basiskanal jedoch als ungesicherte Broadcasts mittels User Datagram Protocol (UDP) versendet. Im Gegensatz dazu erfolgt die Roaming-Aushandlung über gesicherte Unicast-Nachrichten auf Basis von Transmission Control Protocol (TCP). Somit ist die Verfügbarkeit aktueller Informationen gewährleistet.

5.3.4.4 Balancierung von Clustern

Die Balancierung einer Cluster-Konstellation kann nach verschiedenen Gesichtspunkten erfolgen. Steht z. B. die Kanalauslastung im Fokus, könnten häufig kommunizierende Knoten in kleinere Cluster, selten kommunizierende Knoten dagegen in größere Cluster zusammengefasst werden. Dabei könnte auch die logische Verknüpfung von Geräten auf der Anwendungsschicht Berücksichtigung finden. Dies erfordert jedoch Vorwissen über Typ und Kommunikationsmuster der Anwendungen oder eine aktive Beobachtung der Kanalnutzung. Wird hingegen die räumliche Knotendichte zugrunde gelegt, so könnten in dicht besetzten Regionen viele kleine Cluster auf überlappungsfreien Kanälen wünschenswert sein, um *Co-Channel Interference (CCI)* zu reduzieren. In dünn besetzten Regionen wären wiederum größere Cluster zu bevorzugen, um redundante Pfade zu garantieren und Inselbildungen vorzubeugen. Derartige Strategien benötigen jedoch Kenntnis der räumlichen Verteilung der Knoten oder eine Messung bzw. Schätzung ihrer Positionen.

Ein intuitiver Ansatz, der mit wenigen Informationen auskommt, ist die Balancierung der Cluster-Teilnehmerzahl. Im Hinblick auf Netzwerk-Management-Szenarien verfolgt dies einerseits das Ziel, den Kommunikations- und Verwaltungsaufwand gleichmäßig auf die Cluster zu verteilen. Andererseits wird somit der Schaden durch einen CH-Ausfall begrenzt, infolge dessen sich betroffene CMs erneut in die Cluster-Konstellation eingliedern müssen. Der in Kap. 5.3.3 beschriebene Clustering-Phasenablauf garantiert keine Entstehung gleich großer Cluster. Aufgrund der individuellen und gleichzeitigen Entscheidungsfindung aller Knoten beim initialen Cluster-Beitritt kann ein Ungleichgewicht der Cluster-Größen erst nachträglich festgestellt werden. Das im Folgenden dargestellte Balancierungskonzept greift daher erst nach Abschluss des Clustering-Phasenablaufs in Phase 7. Wie in der Forschungsarbeit JRCAP [38] wird auch im Rahmen von CHaChA eine Balancierung *benachbarter* Cluster vorgeschlagen. Maßnahme zur Balancierung ist das zuvor in Kap. 5.3.4.3 beschriebene Roaming von CMs im Randbereich zwischen benachbarten Clustern.

Kommt ein Rand-CM prinzipiell für Roaming infrage, prüft er die Notwendigkeit für eine Cluster-Balancierung nach dem Erhalt neuer CH-Broadcast-Nachrichten für seine Nachbar-Cluster, welche die Liste der CMs des jeweiligen Clusters enthalten. Nach Ermittlung der aktuellen Cluster-Größen aus den Informationen der CH-Nachrichten vergleicht der CM die eigene Cluster-Größe mit der seiner Nachbar-Cluster. Existieren unter diesen solche, deren Teilnehmerzahl mindestens zwei Knoten unterhalb der eigenen Cluster-Größe liegt, erfolgt der Wechsel in das nach ALM dichteste Nachbar-Cluster. Somit wird eine Roaming-Richtung aus größeren in kleinere Cluster vorgegeben.

Abb. 5.8 zeigt die Balancierung am Beispiel einer 5x5-Knoten-Gittertopologie mit fünf Clustern. Im unbalancierten Ausgangszustand (Abb. 5.8 links) besitzen das rote und das grüne Cluster eine Teilnehmerzahl oberhalb der anvisierten Durchschnittsgröße von fünf Knoten. Die Randknoten 3 und 15 (grünes Cluster) sowie 11 und 23 (rotes Cluster) besitzen jeweils die Möglichkeit zum Roaming in das um zwei Knoten kleinere Nachbar-Cluster, wobei vereinfacht angenommen wird, dass die Knoten 11 und 15 diesen Umstand früher erkennen und das Roaming durchführen. Im Ergebnis entsteht eine ausgeglichene Cluster-Konstellation (Abb. 5.8 rechts).

Zur Koordination vom Roaming-Vorgängen führt jeder CH eine Liste der Nachbar-Cluster, in die kürzlich CMs abgewandert oder aus denen im Gegenzug CMs beigetreten sind. Nach jedem Roaming-Vorgang werden weitere Wechsel in dasselbe Ziel-Cluster bzw. Beitritte aus demselben Ursprungs-Cluster vorübergehend abgelehnt, um in Konflikt stehende Roaming-Vorgänge von CMs zu reduzieren, die eine Balancierungsmöglichkeit gleichzeitig erkannt haben. Diese Strategie bildet einen Kompromiss zwischen dem schnellen Ausgleich stark unterschiedlicher Cluster-Größen und der Prävention von Lawineneffekten mit der Gefahr oszillierender Cluster.

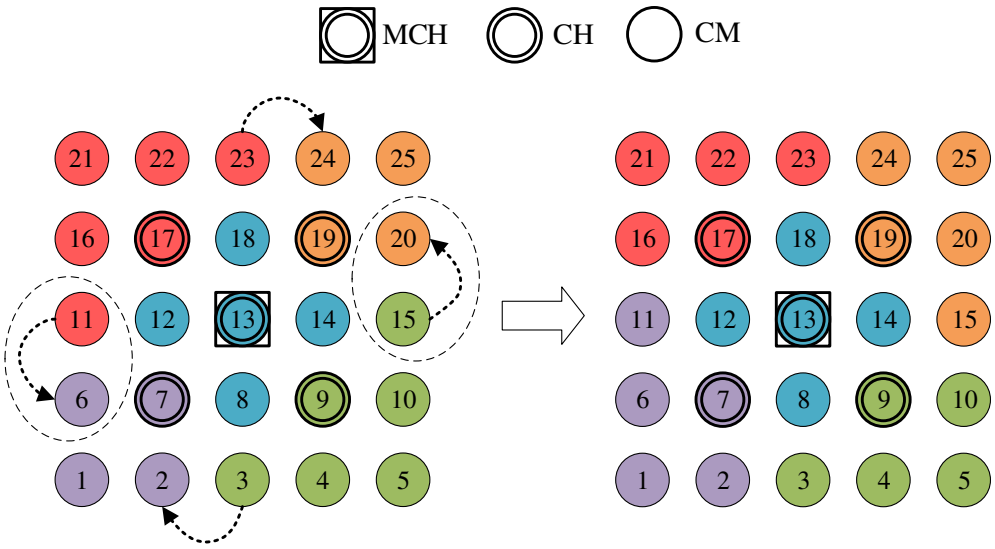


Abbildung 5.8: Balancierung am Beispielnetzwerk
(eingekreiste Roaming-Möglichkeiten werden im Beispiel früher erkannt und durchgeführt)

5.3.4.5 Erneute Cluster-Bildung

Basierend auf den Konzepten für den nachträglichen Cluster-Beitritt, die Isolations- und Ausfallerkennung sowie die Cluster-Balancierung ist es möglich, auch im Anschluss an den initialen Clustering-Phasenablauf auf Topologieänderungen und Verbindungsprobleme zu reagieren. So können sich nachinstallierte Geräte automatisch in ein bestehendes Netzwerk eingliedern, welches in der Lage ist, ein dadurch entstehendes Ungleichgewicht der Cluster-Größen selbständig auszugleichen. Ebenso können einzelne CH-Ausfälle kompensiert werden, indem die zugehörigen CMs in umliegende Cluster wechseln.

Darüber hinaus sind jedoch Szenarien denkbar, die eine Revision der Cluster-Konstellation erfordern. Ein möglicher Grund sind starke Topologieänderungen, ausgelöst z. B. durch Hinzukommen einer Vielzahl von Knoten, gehäufte CH-Ausfälle oder Isolation größerer Teilnetze. Auch könnte ein persistentes Ungleichgewicht der Cluster-Größen entstehen, das in Ermangelung von Cluster-Randknoten nicht durch Roaming behoben werden kann.

Die verteilte, asynchrone Problembehandlung durch einzelne Knoten ist möglicherweise nicht immer in der Lage, eine für den jeweiligen Anwendungsfall geeignete Cluster-Konstellation herzustellen. Eine zuverlässigere Alternative ist hier die Einleitung einer erneuten Cluster-Bildung („Re-Clustering“). Die Erkennung der Notwendigkeit für ein Re-Clustering sowie dessen koordinierte Ankündigung im Netzwerk sind in CHaChA konzeptuell den CHs vorbehalten. Mögliche Erkennungskriterien sind die Entwicklung der Gesamtknotenzahl des Netzwerks (Network Size N) oder des CM/CH-Verhältnisses seit der letzten Cluster-Bildung.

Im Rahmen dieser Arbeit wurden noch keine Re-Clustering-Mechanismen umgesetzt und erprobt, sie sind jedoch auf Grundlage der zuvor beschriebenen Konzepte realisierbar. Nach Erkennung einer Bedingung für das Re-Clustering übernehmen CHs dessen Einleitung auf dem Basiskanal (Primär-Interface). Ein intuitiver Ansatz ist das Versenden von PHASE_0-Broadcast-Nachrichten, vergleichbar mit der Steuerung der Phasenübergänge durch den Master Cluster Head (MCH) im Clustering-Phasenablauf. Dies bewirkt einen koordinierten Wechsel aller Knoten in Phase 0, woraufhin eine erneute Cluster-Bildung durchgeführt wird. Auch während des Re-Clusterings bleibt die Konnektivität im Netzwerk stets über das statisch konfigurierte Primär-Interface auf dem Basiskanal gewährleistet.

5.4 Prototypische Umsetzung

Die Konzepte des Clustering-Verfahrens *CHaChA* wurden als Java-Applikation realisiert. Abb. 5.9 zeigt die vereinfachte Architektur des entstandenen Software-Prototyps. Die Implementierung wird auf einem Linux-Betriebssystem mit 802.11s-Unterstützung ausgeführt (vgl. Kap. 2.5.7). Konfigurationsoptionen (z. B. Gerätenamen der WLAN-Interfaces) sowie jegliche Parameter des CHaChA-Algorithmus (Zeitkonstanten und Schwellenwerte, siehe Kap. 5.3.2) können individuell als Argumente beim Programmstart übergeben werden.

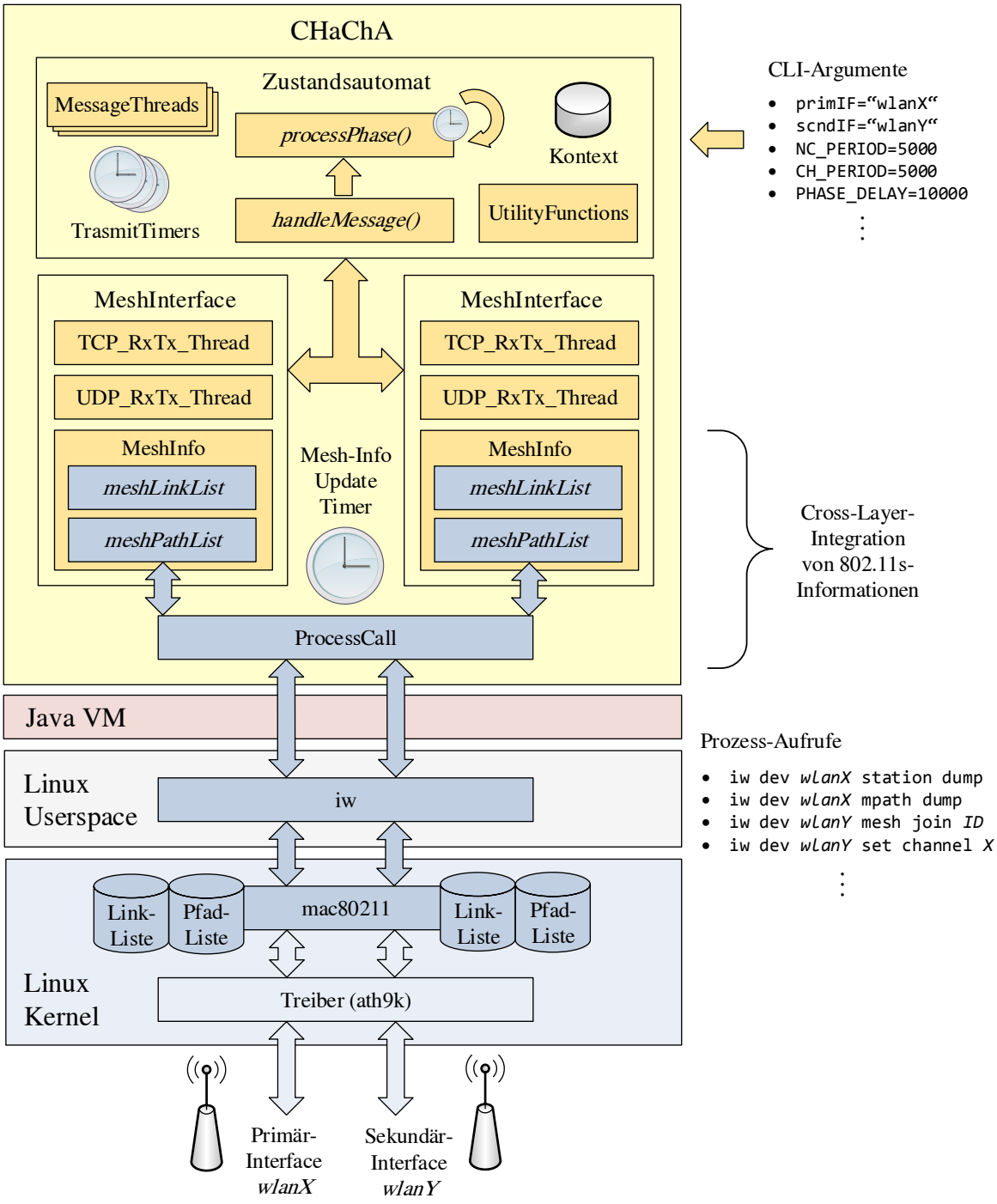


Abbildung 5.9: Software-Architektur der *CHaChA*-Implementierung

Der Phasenablauf zur initialen Cluster-Bildung samt der ab Phase 7 greifenden Mechanismen zur Online-Cluster-Anpassung wurden entsprechend den Konzeptbeschreibungen in Kap. 5.3.3 und 5.3.4 als Zustandsautomat umgesetzt. Je nach Phase und Knotenrolle sind Zustandsübergänge einerseits von Timeouts und andererseits von eingehenden CHaChA-Protokollnachrichten abhängig. Der zumeist periodische Versand verschiedener Nachrichtentypen, wie z. B. CENT-Nachrichten zur MCH-Wahl, Phasenankündigungen des MCH oder CH-Broadcasts, erfolgt in separaten Threads mit eigenen Sende-Timern, die abhängig von der Phase aktiv sind. Weitere Funktionen realisieren die Berechnung der verschiedenen CHaChA-Metriken (siehe Kap. 5.3.2) sowie Entscheidungen anhand des Verbindungsstatus auf dem Basis- und Cluster-Kanal.

Zur TCP-/UDP-Kommunikation sowie zur Verwaltung der Verbindungsinformationen besitzt das Programm Java-Objekte, welche die Mesh-Interfaces des Knotens abbilden. Ein dedizierter Update-Thread übernimmt den zyklischen Abruf der Link- und Pfadlisten aus dem Linux-Kernel und realisiert somit eine Cross-Layer-Integration der 802.11s-Informationen in die Anwendungsschicht. Analog zur OS-Anbindung des Mesh-Network-Aware BitTorrent (MeNTor)-Prototyps (siehe Kap. 4.5) erfolgt der Zugriff auf die 802.11s-Datenstrukturen des Linux-Kernelmoduls `mac80211` durch Prozessaufrufe des Kommandozeilenprogramms `iw`. Hierbei werden die aus `iw` eingelesenen Textausgaben der Link- und Pfadlisten in entsprechende Java-Datentypen (String-Listen) überführt und innerhalb der `MeshInterface`-Objekte gespeichert.

Neben diesen Verbindungsinformationen besitzt jedes Interface einen separaten Thread für das Senden und Empfangen von CHaChA-Protokollnachrichten mittels TCP bzw. UDP. Während TCP für zuverlässige Unicast-Übertragungen zu ausgewählten Knoten genutzt wird und somit auch für Nachrichten, deren Reichweite auf die eigenen Mesh-Nachbarn beschränkt werden soll, dient UDP in der aktuellen Implementierung ausschließlich dem Versand von Broadcast-Nachrichten für netzwerkweite Ankündigungen. Eine perspektivisch leichtgewichtigere Alternative für die Gruppenkommunikation zu Nachbarknoten ist die Nutzung zuverlässiger Multicast- und „Groupcast“-Mechanismen auf der WLAN-Sicherungsschicht. Diese sind Gegenstand jüngerer Standardisierungbestrebungen und in der Praxis noch mit verschiedenen Einschränkungen verbunden [198]. An dieser Stelle sei auf die ausführlichere Diskussion in Kap. 4.4.1 des MeNTor-Konzepts verwiesen.

Der Aufbau der CHaChA-Protokollnachrichten wird in Tab. 5.6 zusammengefasst, wobei für jeden Nachrichtentyp (Opcode) die Payload einer Beispielnachricht angegeben ist. Für die prototypische Implementierung wurde eine Klartextkodierung gewählt, die im Rahmen der Evaluation eine einfache Analyse der Paketmitschnitte erlaubte. Dazu werden alle Payload-Bestandteile, darunter MAC-Adressen, Metrik-Zahlenwerte und WLAN-Kanalnummern, beim Versand als Java-String-Repräsentationen serialisiert und auf Empfängerseite wieder in ihre entsprechenden Datentypen überführt. Beginnend mit dem Opcode der Nachricht werden die Felder der Payload durch ein reserviertes Trennzeichen (Delimiter-Character ’|’) abgegrenzt und können so beim Empfang unterschieden werden. Die einfachsten Nachrichtentypen PHASE_X, PCH und JOIN_REQ/LEAVE_REQ bestehen dabei nur aus ihrem Opcode. Die Antwortnachrichten JOIN_RESP/LEAVE_RESP enthalten zusätzlich eine 1 oder 0 als Hinweis auf Annahme bzw. Ablehnung von Roaming-Gesuchen. Analog führen die Metriknachrichten CENT, NC und WNPR den Zahlenwert ihrer Metrik, während die NH2CH-Nachricht die MAC-Adresse des Nachbarknotens enthält, über den der Mesh-Pfad zum eigenen CH verläuft. Die komplexeren CH-Ankündigungsnachrichten bestehen neben ihrem Opcode aus der Mesh-ID und Kanalnummer des Clusters sowie der Liste der CM-MAC-Adressen. Auch die CHAN_SEL-Nachricht kann abhängig von der CH-Anzahl eine gewisse Größe erreichen, da sie während der Kanalwahlsequenz in Phase 5 schrittweise um Paare aus CH-MAC-Adresse und Cluster-Kanal erweitert wird. In der Praxis erlauben kompaktere Kodierungen, wie z. B. die Nutzung von Binärdarstellungen für Opcodes und andere Zahlenwerte sowie die Definition fester Feldlängen ohne Trennzeichen, deutlich geringere absolute Nachrichtengrößen. Ebenso könnten Kompressionsverfahren verwendet werden, wie z. B. Lauflängenkodierung oder *DEFLATE*-basierte

Tabelle 5.6: Aufbau der Unicast (UC)- und Broadcast (BC)-Nachrichten

Typ (Opcode)	**UC/BC**	**Klartext-Payload einer Beispielnachricht**
CENT	BC	"`CENT|0.0021`"
NC	UC	"`NC|8`"
PCH	UC	"`PCH`"
WNPR	UC	"`WNPR|0.1875`"
CH	BC	"`CH|mesh_id|36|00:00:00:00:00:AA|...|00:00:00:00:00:AF`"
JOIN	UC	"`JOIN`"
LEAVE	UC	"`LEAVE`"
CHAN_SEL	UC	"`CHAN_SEL|00:00:00:00:00:A1|36|...|00:00:00:00:00:A5|153`"
PHASE_X	BC	"`PHASE_1`"
NH2CH	UC	"`NH2CH|00:00:00:00:00:AA`"
JOIN_REQ/RESP LEAVE_REQ/RESP	UC	"`JOIN_REQ`" → "`JOIN_RESP|1`" (ACK) oder "`JOIN_RESP|0`" (NACK) "`LEAVE_REQ`" → "`LEAVE_RESP|1`" (ACK) oder "`LEAVE_RESP|0`" (NACK)

Methoden [240, 241]. Im Rahmen der späteren Bewertung der Kommunikationskosten des CHaChA-Phasenablaufs in Kap. 5.5.2.4 lag der Fokus jedoch auf der Untersuchung des relativen Zusammenhangs zwischen der Netzwerkgröße und der gesendeten Datenmenge.

Der Java-Quellcode der CHaChA-Implementierung [5.1] sowie verschiedene zur Experimentautomatisierung entwickelte Shell-Skripte [5.2] sind frei verfügbar. Darüber hinaus wurden im Rahmen studentischer Arbeiten [C 6, 7] zwei Werkzeuge zur Offline- [5.3] bzw. Online-Visualisierung [5.4] der Cluster-Konstellation der Testumgebung entwickelt, die nachfolgend in Kap. 5.5 näher diskutiert werden.

5.5 Evaluation in der Testumgebung

Um die Praktikabilität des Clustering-Ansatzes CHaChA zu validieren, wurden umfangreiche Experimente in der Testumgebung *Mini-Mesh* (siehe Kap. 3) durchgeführt. Kap. 5.5.1 beschreibt die hier genutzte Gerätekonfiguration. Anschließend untersucht Kap. 5.5.2 den initialen Clustering-Phasenablauf in statischen Mesh-Topologien verschiedener Größe. Das Verfahren wird einerseits hinsichtlich der erzeugten Cluster-Konstellation, andererseits mit Blick auf seinen Zeitbedarf und Nachrichten-Overhead diskutiert. In Kap. 5.5.3 folgt die Erprobung weiterführender Konzepte zur dynamischen Cluster-Anpassung bei Netzwerkveränderungen. Anschließend werden in Kap. 5.5.4 die durch das Clustering entstehenden Performance-Vorteile anhand eines Netzwerk-Monitoring-Szenarios demonstriert. Dafür wird ein dezentraler Ansatz, der die durch CHaChA erzeugte Cluster-Konstellation nutzt, mit einem zentralisierten Ansatz ohne Cluster verglichen.

[5.1] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha`
[5.2] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha-scripts`
[5.3] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha-octave-toolkit`
[5.4] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha-monitor`

5.5.1 Gerätekonfiguration

Wie bereits die Untersuchung der Optimierungslösung *MeNTor* in Kap. 4 erfolgte auch die praktische Erprobung des CHaChA-Prototyps in der im Rahmen dieser Arbeit entwickelten miniaturisierten Testumgebung *Mini-Mesh* (siehe Kap. 3). Untersucht wurden reguläre Gitteranordnungen aufsteigender Teilnehmerzahl mit bis zu 25 der insgesamt 36 verfügbaren Intel-Galileo-Boards. Im Gegensatz zum 6x6-Knoten-Gitter besitzt der 5x5-Aufbau eine symmetrisch zentrale Knotenposition und wurde daher als größte Topologie gewählt. Sie erlaubt die Gegenüberstellung einer dezentral in Clustern durchgeführten Statusabfrage aller Knoten mit der Best- und Worst-Case-Platzierung eines dedizierten Management-Knotens in der Mitte bzw. Ecke des Gitters (siehe Kap. 5.5.4). Neben einer On-Board-Ethernet-Schnittstelle für die Integration in ein kabelgebundenes Netzwerk zur Experimentsteuerung besitzt jeder Knoten ein 802.11n-fähiges WLAN-Modul (Mini-PCI-Express (PCIe)-Karte) mit zwei Antennen. Analog zu den in Kap. 4.6.1 genutzten Einstellungen wurde die WLAN-Datenrate auf das 802.11n Modulation and Coding Scheme (MCS) 3 konfiguriert, wodurch sich für eine Kanalbreite von 20 MHz eine Bruttodatenrate von 26 Mbit/s ergab [109]. Daneben galten die Linux-Standardeinstellungen für TCP und die in Kap. 2.5 erläuterten 802.11s-Parameter.

Das Konzept von CHaChA setzt das Vorhandensein von zwei WLAN-Interfaces pro Gerät voraus, um nach der initialen Cluster-Bildung gleichzeitig auf dem Basis- und Cluster-Kanal kommunizieren zu können. Zwar bestand die Möglichkeit, jedes Galileo-Board zusätzlich zu dessen Mini-PCIe-Karte mit einem USB-WLAN-Adapter auszustatten. Im Rahmen dieser Arbeit waren jedoch keine USB-Produkte verfügbar, die dem Chipsatz der Mini-PCIe-Karten entsprachen (Atheros AR9280) und gleichzeitig externe Antennen zur Anbringung von Dämpfungsgliedern unterstützten. Eine gemischte Hardware-Konfiguration aus WLAN-Interfaces mit verschiedenen Reichweiten- und Performance-Eigenschaften wurde bewusst vermieden und stattdessen jeweils nur die Mini-PCIe-Karte als Primär-Interface genutzt. Dieses wurde auf den festen WLAN-Kanal 149 (Mittenfrequenz $f_c = 5745$ MHz) konfiguriert, der somit als Basiskanal für CHaChA diente. Für die Untersuchung der initialen Cluster-Bildung (siehe Kap. 5.5.2) war ein zweites Interface zunächst nicht nötig, da einerseits alle CHaChA-Kontrollnachrichten auf dem Basiskanal versendet werden und andererseits die Möglichkeit bestand, den Cluster-Kanal nur virtuell zu wählen, ohne die Konfiguration des Sekundär-Interfaces tatsächlich durchzuführen. Bei der Erprobung der dynamischen Cluster-Anpassung (siehe Kap. 5.5.3) wurden dementsprechend nur Topologieänderungen auf dem Basiskanal untersucht. Auch die Bewertung der durch Clustering und Kanalseparation entstehenden Performance-Vorteile anhand eines Netzwerk-Monitoring-Anwendungsbeispiels erfolgte, wie in Kap. 5.5.4 detailliert beschrieben, lediglich unter Nutzung der Primär-Interfaces.

5.5.2 Clustering-Ergebnisse in statischen Topologien

Im Folgenden wird CHaChA in statischen Mesh-Topologien verschiedener Größe hinsichtlich Clustering-Ergebnis, Zeitbedarf und Nachrichten-Overhead untersucht. Zunächst beschreibt Kap. 5.5.2.1 die Abschätzung der Clustering-Dauer abhängig von den Parametern des Algorithmus. Nachdem Kap. 5.5.2.2 den allgemeinen Experimentablauf erläutert, diskutiert Kap. 5.5.2.3 das Clustering einer 5x5-Knoten-Gittertopologie für eine konservative Parametrisierung von CHaChA. Der Fokus liegt hierbei auf der Bestätigung der Reproduzierbarkeit der Ergebnisse sowie der zuvor abgeschätzten Clustering-Dauer. Anschließend werden in Kap. 5.5.2.4 für eine zweite Parametrisierung mehrere Gittertopologien ansteigender Knotenzahl untersucht, um den Einfluss der Netzwerkgröße auf Clustering-Dauer und Nachrichten-Overhead zu bestimmen.

5.5.2.1 Abschätzung der Clustering-Dauer

Aus der Entwurfsentscheidung, die initiale Cluster-Bildung als Phasenablauf zu realisieren, ergeben sich unter anderem Vorteile für die praktische Analysierbarkeit des Verfahrens. So kann im Vorfeld eine Abschätzung des Zeitbedarfs für den Phasenablauf durchgeführt werden, da diese sich maßgeblich aus der Parametrisierung des Algorithmus ergibt. Grundsätzlich erfolgt die Parameterwahl als Kompromiss zwischen geringem Overhead (Clustering-Dauer und Anzahl gesendeter Kontrollnachrichten) und erforderlicher Robustheit (hohe Zustellungswahrscheinlichkeit der für das Clustering benötigten Informationen). In der Praxis ist die Parametrisierung z. B. abhängig von der erwarteten Netzwerkgröße und Verbindungsqualität sowie der Verarbeitungsgeschwindigkeit der Mesh-Knoten. Für die ausführliche Beschreibung aller CHaChA-Parameter wird auf Kap. 5.3.2, Tab. 5.5 verwiesen.

Maßgeblich für die Clustering-Dauer sind feste Wartezeiten vor Phasenübergängen sowie periodische Sendewiederholungen von Broadcast-Nachrichten, die vorwiegend die Phasenankündigungen des Master Cluster Head (MCH) betreffen. Abb. 5.10 zeigt die allgemeine Abschätzung der Clustering-Dauer als Komposition der zeitlichen Beiträge aller Phasen. Hierbei wird angenommen, dass alle Knoten gleichzeitig mit der Ausführung des Phasenablaufs beginnen. Abb. 5.11 zeigt ergänzend eine Überschlagsrechnung unter Annahme der in Kap. 5.5.2.3 beschriebenen CHaChA-Parametrisierungsvariante P1 (siehe Tab. 5.7), die als Ausgangsparametrisierung für die praktische Erprobung des Prototypen diente. Dabei handelt es sich um eine großzügige Dimensionierung der Zeitkonstanten und Schwellenwerte zur Sicherstellung einer robusten Cluster-Bildung, mit der sich ein geschätzter Zeitbedarf von 130 s ergibt.

Der Zeitbedarf in Phase 0 (Netzbeitritt und MCH-Wahl) setzt sich aus vier Komponenten zusammen. Beim Programmstart des CHaChA-Prototyps wird zunächst eine feste Zeit *INIT_DELAY* gewartet, um Objekte und Datenstrukturen für die Netzwerk-Interfaces anzulegen und mit Informationen des Betriebssystems zu füllen. Dieser Parameter ist somit nicht Teil des Clustering-Algorithmus, sondern abhängig von der Implementierung und Hardware-/Software-Plattform. Abgestimmt auf die Testumgebung wurde für *INIT_DELAY* ein Wert von 2 s gewählt. Nach seiner Initialisierung prüft ein Knoten, ob bereits Cluster bestehen und der Phasenablauf übersprungen werden kann. Dazu wartet er für *CH_THRESH* Sendeperioden der Länge *CH_PERIOD* auf den Empfang von Cluster Head (CH)-Broadcast-Nachrichten. Treffen vor Überschreitung dieser Zeit neue Nachrichten zuvor unbekannter CHs ein, wird der Timer jeweils zurückgesetzt. Die Verzögerung, die durch das Kennenlernen neuer CHs entstehen kann, wird mit T_{NewCH} zusammengefasst. Diese ist nicht nur abhängig vom Zeitver-

	INIT_DELAY	*Phase 0*
+	T_{NewCH} + (CH_THRESH · CH_PERIOD)	
+	T_{KO} + (CENT_THRESH · CENT_PERIOD)	
+	(PHASE_TRIES · PHASE_PERIOD)	
+	PHASE_DELAY + (PHASE_TRIES · PHASE_PERIOD)	*Phase 1*
+	PHASE_DELAY + (PHASE_TRIES · PHASE_PERIOD)	*Phase 2*
+	CH_PERIOD + PHASE_DELAY + (PHASE_TRIES · PHASE_PERIOD)	*Phase 3*
+	PHASE_DELAY + (PHASE_TRIES · PHASE_PERIOD)	*Phase 4*
+	$T_{SelectChannel}$ + (PHASE_TRIES · PHASE_PERIOD)	*Phase 5*
+	$T_{ConfigIF2}$	*Phase 6*
=	erwartete Clustering-Dauer	

Abbildung 5.10: Zeitkomponenten zur Abschätzung der Clustering-Dauer

	2000 ms	*Phase 0*
+	(2 · 5000 ms)	
+	(4 · 500 ms) + (20 · 500 ms)	
+	(20 · 500 ms)	
+	10000 ms + (20 · 500 ms)	*Phase 1*
+	10000 ms + (20 · 500 ms)	*Phase 2*
+	5000 ms + 10000 ms + (20 · 500 ms)	*Phase 3*
+	10000 ms + (20 · 500 ms)	*Phase 4*
+	1000 ms + (20 · 500 ms)	*Phase 5*
=	130 s	

Abbildung 5.11: Geschätzte Clustering-Dauer für Parametrisierung P1 (siehe Tab. 5.7)

satz der CH-Broadcasts, sondern auch von deren Ausbreitungsdauer im Netzwerk. Die nachfolgend beschriebenen Clustering-Experimente wurden stets in einem neu initialisierten Netzwerk auf dem Basiskanal gestartet, sodass T_{NewCH} vernachlässigt werden kann.

Sind keine CHs vorhanden, kommt es zur MCH-Wahl. Alle Knoten geben dazu periodisch im Zeitabstand *CENT_PERIOD* ihre Centrality (CENT) per Broadcast im Netzwerk bekannt. Bei Empfang einer höheren CENT-Metrik scheiden Knoten gemäß KO-Prinzip aus. Der Knoten mit höchster Metrik gewinnt den Wettlauf nach *CENT_THRESH* konsekutiven Sendevorgängen ohne den Empfang einer Nachricht im Gegenzug. Die für das Ausscheiden aller Knoten mit niedrigerer Metrik benötigte Konvergenzzeit des KO-Verfahrens wird in Abb. 5.10 mit T_{KO} zusammengefasst. Auch sie ist abhängig von der Broadcast-Ausbreitungsdauer der jeweiligen Netzwerkgröße bzw. -topologie und beträgt ein Vielfaches der Sendeperiode *CENT_PERIOD*. Dabei sollte die Periode je nach Topologie groß genug gewählt werden, um die vollständige Ausbreitung der Broadcasts aller Knoten innerhalb einer Runde zu gewährleisten. Führen alle Knoten zuerst einen Sendevorgang aus, bevor sie eine Nachricht empfangen, werden mindestens zwei Broadcast-Runden für das KO-Verfahren benötigt. Zusätzliche Runden entstehen z. B. durch eine zu kleine Dimensionierung der *CENT_PERIOD* oder den Verlust von Broadcast-Nachrichten bzw. deren Weiterleitungen im Netzwerk. In Stichprobenversuchen mit der 5x5-Knoten-Gittertopologie und einer Sendeperiode von 500 ms wurde eine Konvergenz nach bis zu vier Broadcast-Runden beobachtet und T_{KO} nachfolgend vereinfacht mit einer festen Länge von (4 · *CENT_PERIOD*) abgeschätzt.

Im Anschluss an die Wahl des MCH leitet dieser den Phasenübergang mittels periodischer Broadcast-Nachrichten ein, was dem Zeitbedarf (*PHASE_TRIES* · *PHASE_PERIOD*) entspricht. Im Rahmen der Arbeit wurden für alle Phasenübergänge die gleiche Wiederholungsanzahl und Sendeperiode genutzt. Die jeweilige Dauer der darauffolgenden Phasen 1–4 setzte sich somit aus derselben Ankündigungszeit zuzüglich einer Verzögerung *PHASE_DELAY* zusammen. Diese Verzögerung berücksichtigt der MCH vor Einleitung des Phasenübergangs, um den Informationsaustausch aller Knoten pro Phase samt Stabilisierung etwaiger Metriken zu gewährleisten. In Phase 3 wird zudem die Zeit *CH_PERIOD* (Länge einer CH-Broadcast-Periode) abgewartet, um eine robuste Cluster-Ankündigung unabhängig vom Parameter *PHASE_DELAY* sicherzustellen.

Statt nach einer festen Wartezeit erfolgt die Ankündigung des Übergangs aus Phase 5 in Phase 6 direkt nach der Kanalwahl, an der die CHs inklusive MCH beteiligt sind. Der Zeitbedarf für die Kanalwahlsequenz und deren TCP-Unicast-Nachrichtenaustausch wird mit $T_{SelectChannel}$ zusammengefasst. Dieser ist neben der CH-Anzahl abhängig von der Netzwerktopologie (mittlerer Abstand der CHs zueinander) und Übertragungsgeschwindigkeit, sodass wie für T_{KO} eine generelle Abschät-

zung schwierig ist. In Stichprobenversuchen mit der 5x5-Knoten-Gittertopologie, dabei entstanden fünf CHs im Abstand von max. zwei Hops zueinander und es galt die WLAN-Datenrateneinstellung MCS 3 (Bruttodatenrate 26 Mbit/s), betrug $T_{SelectChannel}$ bis zu 1 s und wurde nachfolgend vereinfacht mit diesem oberen Wert angenommen. Die in Phase 6 theoretisch für die Konfiguration der Sekundär-Interfaces benötigte Zeit $T_{ConfigIF2}$ konnte bei den Experimenten ignoriert werden, da nur das Primär-Interface zum Einsatz kam. Praktisch ist $T_{ConfigIF2}$ zwar abhängig von der genutzten Geräteplattform, in der Regel benötigt die Konfiguration des WLAN-Kanals samt Netzbeitritt jedoch nur Bruchteile einer Sekunde [242]. Den anschließenden Übergang von Phase 6 in den verteilten Cluster-Betrieb („Phase 7“) führen alle Knoten stets selbständig aus, sodass keine zusätzliche Wartezeit entsteht.

5.5.2.2 Experimentablauf

Abhängig von der untersuchten Topologie waren nur die jeweiligen Knoten der Mini-Mesh-Testumgebung aktiv. In jeder Messung wurde der CHaChA-Prototyp gleichzeitig auf allen Geräten gestartet. Diese waren bereits zu Beginn der Messung auf dem Basiskanal miteinander verbunden (Single-Channel-Mesh-Netzwerk). Das Clustering galt als abgeschlossen, sobald der letzte Knoten Phase 7 des CHaChA-Phasenablaufs erreicht hatte. Da, wie in Kap. 5.5.1 beschrieben, auf die Installation von Sekundär-Interfaces verzichtet wurde, führten alle Knoten die Wahl und Anwendung des Cluster-Kanals in den Phasen 5 und 6 nur virtuell durch und speicherten das Ergebnis für die nachträgliche Analyse. Als exemplarischer Cluster-Kanal-Pool wurden fünf zum Basiskanal 149 ($f_c = 5745$ MHz) überlappungsfreie 5 GHz-Kanäle vorgegeben: 36, 40, 44, 48 und 158. Für die im Rahmen der Arbeit untersuchten Topologiegrößen mit bis zu fünf entstehenden Clustern war diese Vorgabe ausreichend. In der Praxis sind bei einer Kanalbandbreite von 20 MHz bis zu 19 überlappungsfreie Kanäle im 5-GHz-Band möglich [99]. Wie in Kap. 5.3.3 erläutert sieht das Konzept von CHaChA darüber hinaus die erneute Wahl bereits genutzter Kanäle mittels einer Distanzstrategie vor.

Neben einer Log-Datei zur Protokollierung seiner Aktionen im Phasenablauf erzeugte jeder Knoten einen Mitschnitt aller durch ihn versendeten TCP- und UDP-Nachrichten mithilfe des Kommandozeilen-Tools `tcpdump`. Am Ende jeder Messung produzierte die CHaChA-Implementierung zudem eine separate Statusdatei, die Informationen über die Zeitdifferenz zwischen Programmstart und Erreichen von Phase 7, die finale Rolle des Knotens sowie Informationen zu dessen Cluster (CH und Cluster-Kanal) enthielt. Jegliche auf den Knoten erzeugte Dateien wurden in einem RAM-Dateisystem (Linux `tmpfs`) gespeichert, sodass mögliche Einflüsse durch langsame Schreiboperationen auf den SD-Karten der Knoten vermieden wurden.

Die Initialisierung des Mesh-Netzwerks und der CHaChA-Instanzen sowie die Automatisierung der Experimente wurden durch Kommandozeilenskripte [5.5] realisiert. Diese wurden teils auf den Mesh-Knoten und teils auf einem Labor-PC ausgeführt, der über das separate Ethernet-Kontrollnetzwerk mit der Testumgebung verbunden war. Im Anschluss an eine Messreihe (Serie von Clustering-Durchläufen) wurden die Log-/Statusdateien und `tcpdump`-Paketmitschnitte aller Knoten gesammelt abgerufen und verarbeitet. Ein als Studienarbeit [C 7] unter Nutzung der freien MATLAB-Alternative *Octave* [243] entwickeltes Werkzeug [5.6] diente der Visualisierung aller gebildeten Cluster-Konstellationen und der Ermittlung ihrer jeweiligen Auftrittshäufigkeit in der Messreihe. Dabei waren die Gitterpositionen der Knoten in der Visualisierung statisch vorgegeben. Anhand der eingelesenen Statusinformationen über die Rolle und Cluster-Zugehörigkeit eines Knotens wurde dieser eingeblendet und farblich markiert. Aus den gesammelten Log-/Statusdateien und Paketmitschnitten berechnete es außerdem die mittlere Clustering-Dauer sowie den mittleren Kommunikations-Overhead, der durch CHaChA-Kontrollnachrichten erzeugt wurde (TCP-/UDP-Datenmenge und -Nachrichtenanzahl).

[5.5] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha-scripts`

[5.6] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha-octave-toolkit`

Tabelle 5.7: *CHaChA*-Parametrisierungsvariante P1

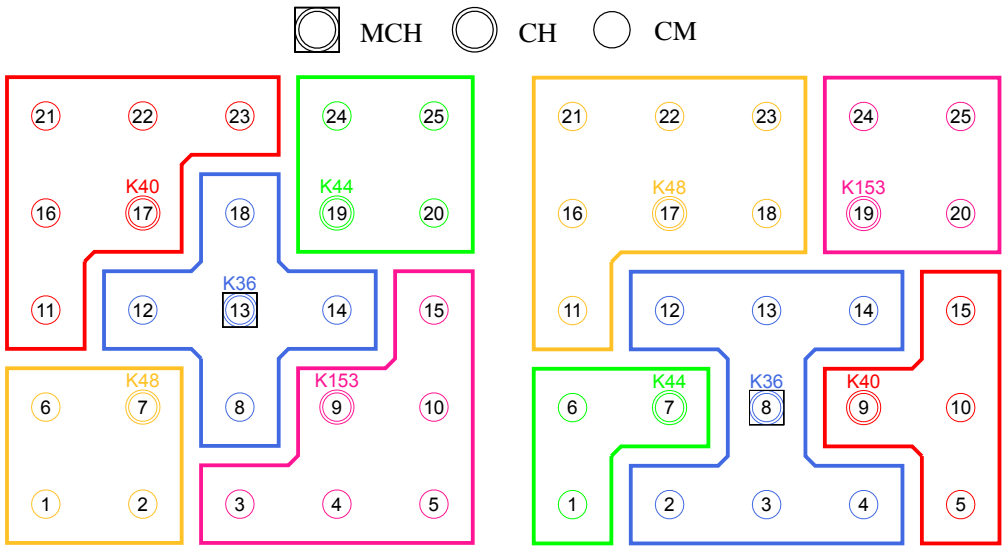
Parameter	**Einheit**	**Wert**
CENT_PERIOD	ms	500
CENT_THRESH	#	20
NC_PERIOD	ms	5000
CH_PERIOD	ms	5000
CH_THRESH	#	2
PHASE_DELAY	ms	10000
PHASE_PERIOD	ms	500
PHASE_TRIES	#	20

5.5.2.3 Erprobung und Reproduzierbarkeitsanalyse

Zunächst wurde CHaChA im Testaufbau mit 25 Geräten (5x5-Knoten-Gitter) hinsichtlich Clustering-Resultat, Zeitbedarf und Kommunikations-Overhead untersucht. Im Vordergrund standen die funktionale Erprobung des Clustering-Phasenablaufs sowie die Analyse der Reproduzierbarkeit seiner Ergebnisse unter kontrollierten Laborbedingungen. Um den Ergebnisspielraum zu maximieren, wurde mit der 5x5-Anordnung bewusst die größte der in dieser Arbeit betrachteten Mesh-Topologien gewählt. Tab. 5.7 zeigt die genutzte Parametrisierungsvariante P1 der Zeitkonstanten und Schwellenwerte des CHaChA-Algorithmus. Durch großzügige Dimensionierung der Wartezeiten und Broadcast-Sendewiederholungen sollte zunächst eine robuste Cluster-Bildung sichergestellt werden.

Die Messreihe umfasste 50 Clustering-Durchläufe, deren Ergebnisse (Zeitbedarf und gesendete Datenmenge) gemittelt wurden. Abb. 5.12 zeigt die in der Messreihe entstandenen Cluster-Konstellationen. Die zu einem Cluster gehörenden Knoten sind entsprechend ihrem in Phase 5 gewählten Cluster-Kanal in gleicher Farbe dargestellt. Die Knotenrollen MCH, CH, und Cluster Member (CM) sind wie in der Legende angegeben markiert. Wie für den statischen Testaufbau erwartet konnten reproduzierbare Clustering-Ergebnisse erzielt werden. Insgesamt entstanden in der Messreihe nur zwei verschiedene Konstellationen. Zudem bildete sich Konstellation 1 (Abb. 5.12 (a)) mit einer Auftrittsrate von 94 %, d.h. in 47 von 50 Durchläufen, besonders häufig heraus.

Bei der Bildung dieser Konstellation wurde im Wettlaufverfahren der Phase 0 des CHaChA-Phasenablaufs (siehe Konzept in Kap. 5.3.3) der mittig positionierte Knoten 13 aufgrund seiner CENT-Metrik zum MCH gewählt. In Phase 1 erhielten die Knoten des inneren 3x3-Gitters die Rolle Proposed Cluster Head (PCH), da sie mit jeweils acht Nachbarknoten eine maximale Neighbor Count (NC)-Metrik besaßen. Unter diesen neun PCHs gewannen erwartungsgemäß die vier Eckknoten 7, 9, 17 und 19 den Vergleich der Metrik Weighted NC-to-PCHNC-Ratio (WNPR) in Phase 2. Zusammen mit dem bereits in Phase 0 gewählten MCH erhielten somit fünf Knoten die CH-Rolle. Nach netzwerkweiter Ankündigung der CHs in Phase 3 führte der auf der ALM-Distanz zum CH basierende Beitritt der CMs in Phase 4 bereits zu weitgehend balancierten Clustern. Wie in Abb. 5.12 (a) dargestellt erreichte das zentrale Cluster des MCH eine Größe von 5 Knoten, was der anvisierten Durchschnittsgröße bei 25 Knoten und 5 Clustern entspricht. Die Größe der umliegenden Cluster wich um jeweils einen Knoten davon ab. Der in Kap. 5.3.4.3 beschriebene Roaming-Mechanismus würde nach Erreichen von Phase 7 zum Ausgleich der Cluster-Größen führen, indem jeweils ein Randknoten des roten bzw. violetten Clusters in das orange bzw. grüne Cluster wechselt. Die Kanalwahl in Phase 5 erfolgte, beginnend bei Knoten 13 (MCH), in Reihenfolge der jeweils kleinsten ALM-Distanz zu den verbleibenden CHs. Sequentiell entnahmen die Knoten 13, 17, 19, 7 und 9 die in Abb. 5.12 (a) annotierten Kanäle 36, 40, 44, 48 und 158 aus dem vordefinierten Pool überlappungsfreier WLAN-Kanäle



(a) Konstellation 1 (94%) (b) Konstellation 2 (6%)

Abbildung 5.12: Cluster-Konstellationen im 5x5-Knoten-Gitter für Parametrisierung P1 [B 3]
(Cluster entsprechend gewähltem Kanal („K#“) farblich markiert; Auftrittshäufigkeit in %)

im 5-GHz-Band. Nach rein virtueller Anwendung der Kanalkonfiguration in Phase 6 wechselten alle Knoten in Phase 7, womit die Cluster-Bildung abgeschlossen war.

Die zweite Cluster-Konstellation (siehe Abb. 5.12 (b)) entstand in nur 3 von 50 Durchläufen (Auftrittsrate von 6 %). Ihr Hauptunterschied gegenüber Konstellation 1 bestand darin, dass Knoten 8 anstatt Knoten 13 die Wahl zum MCH gewann. Ursache dafür waren Abweichungen der ALM-Distanzmetrik zwischen den Knoten während der MCH-Wahl in Phase 0. Die ALM ist wiederum Bestandteil der Berechnung der CENT-Metrik, auf der die MCH-Wahl beruht. Die Bildung anderer Cluster war ebenfalls auf die verschobene MCH-Position zurückzuführen, da CMs bevorzugt dem Cluster des MCH beitreten, wenn sich dieser in ihrer Nachbarschaft befindet. In Netzwerkszenarien mit höherer Knotenzahl, Ausdehnung und Dynamik sind geringe Abweichungen der MCH-Position vom Zentrum durchaus zu erwarten und als Ergebnis eines heuristischen Ansatzes akzeptabel.

Neben dem Clustering-Resultat wurden der mittlere Zeitbedarf sowie Nachrichten-Overhead des Phasenablaufs bestimmt. Für die großzügige Wahl der Zeitkonstanten und Schwellenwerte des CHaChA-Prototyps (siehe Parametrisierung P1 in Tab. 5.7) ergab sich unter Anwendung der Überschlagsrechnung in Kap. 5.5.2.1 eine geschätzte Clustering-Dauer von 130 s. Der in den Experimenten gemessene mittlere Zeitbedarf betrug 135,5 s (Standardabw. 0,9 s) und lag somit leicht oberhalb der Abschätzung. Während dieser Clustering-Dauer wurden durchschnittlich 9699 (Standardabw. 42) TCP-Nachrichten mit einer Gesamtdatenmenge von 1113,7 kB (Standardabw. 4,9 kB) gesendet, inklusive aller TCP-ACK- sowie -Handshake-Nachrichten für den Verbindungsauf- bzw. -abbau. Zusätzlich erzeugte CHaChA durchschnittlich 5248 (Standardabw. 152) UDP-Nachrichten mit einer Gesamtdatenmenge von 627,8 kB (Standardabw. 20,4 kB). Alle Angaben schließen den durch die Multi-Hop-Weiterleitung von Nachrichten entstandenen Datenverkehr sowie jegliche WLAN-Neuübertragungen mit ein.

Die über die Clustering-Dauer von 135,5 s erzeugte TCP-/UDP-Gesamtdatenmenge von rund 1740 kB kann vereinfacht als Overhead-Datenverkehr mit einer effektiven Datenrate von ca. 100 kbit/s betrachtet werden. Bezogen auf den in Kap. 3.3.3 für die WLAN-Datenrateneinstellung MCS 3 (Bruttodatenrate 26 Mbit/s) gemessenen maximalen Single-Hop-Nettodurchsatz von 22,9 Mbit/s (UDP) bzw. 17,8 Mbit/s (TCP) stellt dies nur etwa 0,5 % der verfügbaren Kommunikationsressourcen eines Mesh-Links dar. Dabei ist zu beachten, dass jeder Knoten im CHaChA-Phasenablauf nur einen

Bruchteil des gesamten Datenverkehrs erzeugt und sich dieser somit auf alle Links des Netzwerks verteilt. Die verglichen mit UDP höhere Nachrichtenanzahl und Datenmenge für TCP sind einerseits auf dessen inhärent höheren Protokoll-Overhead (längerer Header, bestätigte Kommunikation, 3-Wege-Handshake) zurückzuführen. Andererseits werden in der aktuellen Prototyp-Implementierung von CHaChA für jeden gesicherten Informationsaustausch individuelle TCP-Verbindungen auf- und abgebaut. Eine künftige Optimierung könnte darin bestehen, die Verbindungen zu Nachbarknoten durchgängig offen zu halten, da hauptsächlich mit diesen per TCP kommuniziert wird (NC-, PCH-, und WNPR-Nachrichten). Im Vergleich entstand durch UDP-basierte Broadcasts (CENT-, CH- und PHASE-Nachrichten) inklusive deren Weiterleitung im Netzwerk nur etwas mehr als die Hälfte der mit TCP gesendeten Nachrichtenanzahl. Dies korrespondiert ebenfalls mit der erzeugten UDP-Datenmenge. Die für die Verbreitung von Broadcast-Informationen benötigte Anzahl individueller Sendevorgänge ist jedoch stark abhängig von der Netzwerk-Topologie und Übertragungsfehlerwahrscheinlichkeit [244], sodass in größeren Netzwerken der durch UDP verursachte Kommunikationsanteil steigen könnte.

Bei der Konzeption von CHaChA (siehe Kap. 5.3.2) wurden verschiedene Designentscheidungen getroffen, um das Nachrichtenaufkommen auch in größeren Mesh-Topologien möglichst gering zu halten. Einerseits erfolgt der Austausch von Metriken mittels TCP-Unicast-Nachrichten nur zwischen benachbarten Knoten und ausschließlich während des Clustering-Phasenablaufs. Nach der initialen Cluster-Bildung entfallen diese Nachrichten und es werden beginnend mit Phase 7 nur noch Next-Hop-to-Cluster-Head (NH2CH)-Nachrichten zwischen Nachbarknoten ausgetauscht, die der Einschränkung von Roaming-Vorgängen dienen. Durch Roaming von Cluster-Randknoten kann es zum gelegentlichen Versand weiterer TCP-Nachrichten kommen (LEAVE_REQ/RESP bzw. JOIN_REQ/RESP). Die Erkennung von Verbindungsproblemen ist hingegen auf allen Knoten durch Prüfung lokaler Informationen der 802.11s-Sicherungsschicht möglich. Darüber hinaus werden netzwerkweite UDP-Broadcasts nur von CHs erzeugt, mit Ausnahme der CENT-Nachrichten zur MCH-Wahl in Phase 0. Nach dem initialen Phasenablauf verbleiben davon ausschließlich die CH-Broadcast-Nachrichten zur periodischen Ankündigung der Cluster-Informationen.

5.5.2.4 Ergebnisvergleich verschiedener Netzwerkgrößen

Ausgehend von den mit der konservativen Parametrisierung P1 im 5x5-Knoten-Gitter erzielten Ergebnissen wurden Gittertopologien verschiedener Größe untersucht. Im Vorfeld wurde die in Tab. 5.8 gegebene Parametrisierung P2 im 5x5-Knoten-Gitter experimentell bestimmt. Zielstellung dabei war die Reduktion der Clustering-Dauer unter Beibehaltung der Reproduzierbarkeit der zuvor erzielten Clustering-Ergebnisse. Zu diesem Zweck wurden für den Gesamtzeitbedarf maßgebliche Parameter schrittweise gesenkt und überprüft, dass keine negativen Auswirkungen auf das Clustering-Ergebnis entstanden. Zunächst wurde der Zeitbedarf in Phase 0 gesenkt. Dazu wurde einerseits die initiale Prüfung auf bereits existierende Cluster deaktiviert (CH_THRESH $= 0$), da in den Experimenten eine synchrone Inbetriebnahme aller Geräte gewährleistet war und diese feste Wartezeit somit im Gegensatz zu realen Szenarien entfallen konnte. Andererseits wurde der Schwellenwert für konsekutiv empfangene CENT-Nachrichten halbiert (CENT_THRESH $= 10$), was den Zeitbedarf des KO-Verfahrens zur MCH-Wahl reduzierte. Darüber hinaus wurde die Verzögerung vor Phasenübergängen (PHASE_X_DELAY) von 10 auf 2 s gesenkt. Ebenso wurde die Anzahl der Sendewiederholungen (PHASE_X_TRIES) der zugehörigen Ankündigungsnachrichten halbiert. Beide Parameter bestimmen maßgeblich die Dauer der einzelnen Phasen (siehe Abschätzung in Kap. 5.5.2.1). Aufgrund der kürzeren Phasendauer wurden zudem die Sendeintervalle für NC- und CH-Nachrichten gesenkt, damit diese rechtzeitig innerhalb ihrer Phasen zugestellt werden konnten.

Mit der so bestimmten Parametrisierung P2 wurde die durchschnittliche Clustering-Dauer auf knapp 40 % der mit Parametrisierung P1 benötigten Zeit reduziert (55,6 s vs. 135,5 s). Die zuvor am häufigsten erzeugte Cluster-Konstellation (siehe Abb. 5.12 (a)) konnte weiterhin in 9 von 10

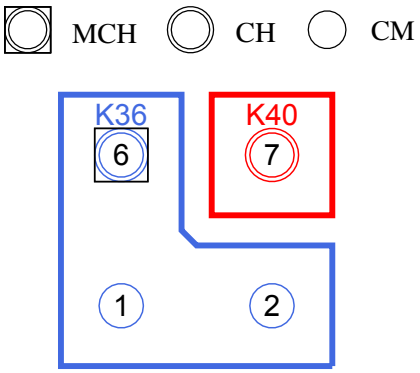
Tabelle 5.8: *CHaChA*-Parametrisierungsvariante P2
(Änderungen gegenüber Variante P1 sind in Spalte 3 fett hervorgehoben)

Parameter	**Einheit**	**Wert P2**	**Vgl. P1**
CENT_PERIOD	ms	500	500
CENT_THRESH	#	**10**	20
NC_PERIOD	ms	**2000**	5000
CH_PERIOD	ms	**2000**	5000
CH_THRESH	#	**0**	2
PHASE_DELAY	ms	**2000**	10000
PHASE_PERIOD	ms	500	500
PHASE_TRIES	#	**10**	20

Messungen reproduziert werden. Die in Tab. 5.8 aufgeführten Parameterwerte sind spezifisch für das Testbed *Mini-Mesh* und dessen Geräteplattform. Es ist davon auszugehen, dass mit leistungsfähigerer Hardware noch geringere Zeiten erreichbar sind.

Im Folgenden werden Clustering-Ergebnisse diskutiert, die mit Parametrisierung P2 für verschiedene Netzwerkgrößen erzielt wurden. Betrachtet wurden Gittertopologien mit 2x2 bis 5x5 Knoten. Das praktische Vorgehen entsprach unverändert dem in Kap. 5.5.2.2 beschriebenen Experimentablauf. Für die hier im Vordergrund stehende Trendanalyse wurden jedoch nur 10 Messungen pro Topologie herangezogen und deren Ergebnisse jeweils gemittelt. Zunächst werden die erzeugten Cluster-Konstellationen diskutiert und deren Entstehung im Kontext des Algorithmus erläutert. Anschließend folgt ein Vergleich der Szenarien hinsichtlich Clustering-Dauer und Kommunikations-Overhead.

2x2-Knoten-Gitter: In der kleinsten Topologie entstand in allen Messungen durchgängig die in Abb. 5.13 gezeigte Konstellation mit zwei Clustern. Obwohl im 2x2-Aufbau keine zentrale Knotenposition existierte, wurde in Phase 0 stets Knoten 6 aufgrund einer leicht höheren CENT-Metrik zum MCH gewählt. Diese berechnet sich aus den ALM-Distanzen aller Mesh-Pfade, deren geringe Unterschiede die Präferenz eines Knotens bewirkten.



Konstellation 1 (100%)

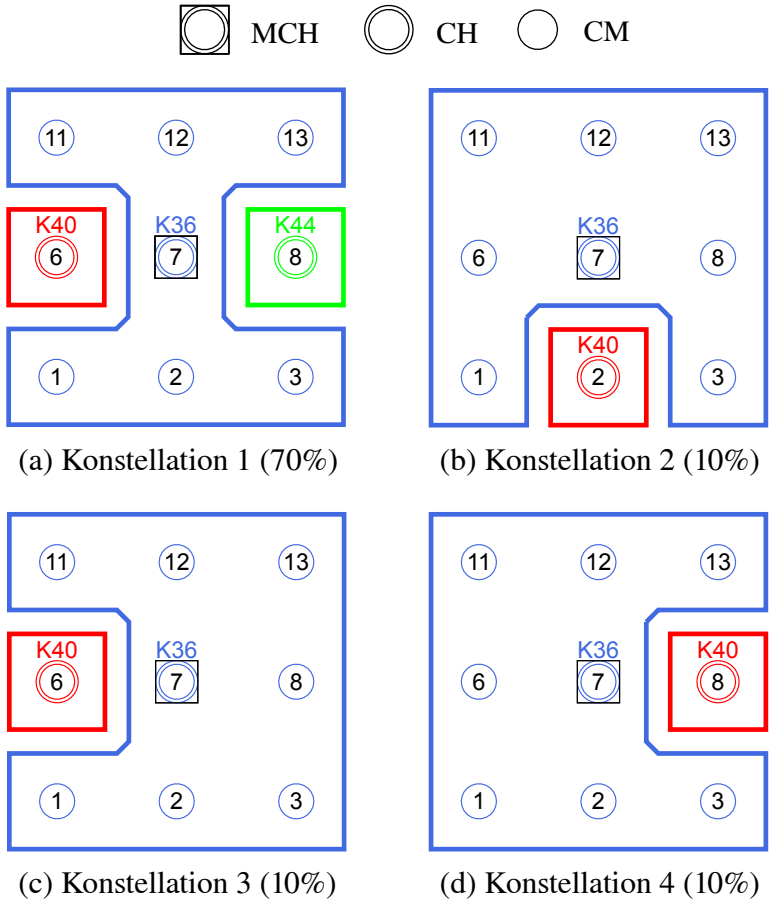
Abbildung 5.13: Cluster-Konstellationen im 2x2-Knoten-Gitter für Parametrisierung P2
(Cluster entsprechend gewähltem Kanal („K#“) farblich markiert; Auftrittshäufigkeit in %)

In Phase 1 bewarben sich die drei verbleibenden Knoten als PCHs, da sie dieselbe NC- und PCHNC-Metrik aufwiesen. Von diesen Kandidaten verblieb in Phase 2 reproduzierbar Knoten 7 als finaler CH. Auch hier war der Grund eine leicht höhere CENT-Metrik, die als Wichtungsfaktor in die Metrik WNPR eingeht. Das aktuelle Konzept von CHaChA erlaubt die Entstehung von CHs in direkter Nachbarschaft zum MCH, jedoch nicht zueinander. Motivation dafür sind Anwendungen, die dem MCH auch über die Cluster-Bildung hinaus eine administrative Sonderstellung einräumen. Darunter fällt z. B. das Abrufen von Statusinformationen (Monitoring) aller Knoten, welches in Kap. 5.5.4 als Fallbeispiel skizziert wird. Hier ist einerseits die Nähe zum MCH für die effiziente Datensynchronisierung wünschenswert, andererseits wird die Abdeckung individueller Netzwerkbereiche durch die CHs anvisiert. Nach Ankündigung der CHs in Phase 4 traten Knoten 1 und 2 stets dem Cluster des MCH bei, der gegenüber anderen benachbarten CHs bevorzugt wird. Auch diese Strategie begünstigt das Monitoring-Szenario, in dem die Synchronisierung der Statusinformationen von bereits durch den MCH verwalteten Knoten entfallen kann. In den Phasen 5 bzw. 6 erfolgte die virtuelle Kanalwahl der CHs, die erwartungsgemäß Kanal 36 und 40 aus dem vordefinierten Pool beanspruchten. Mit Erreichen von Phase 7 war die initiale Cluster-Bildung abgeschlossen.

Die momentan zulässige Entstehung von CHs in direkter Nachbarschaft zum MCH bewirkt, dass Cluster auch in Topologien gebildet werden, die aufgrund geringer Teilnehmerzahl und Ausdehnung keine weitere Unterteilung benötigen (siehe 2x2- und 3x3-Knoten-Gitter). Im Rahmen der Evaluation war dieses Verhalten jedoch gewünscht, um CHaChA im Laboraufbau untersuchen zu können. Für den praktischen Einsatz könnten CHs in direkter Nachbarschaft zum MCH unterbunden oder die Einhaltung einer minimalen ALM-Distanz zu diesem vorgesehen werden, die abhängig von der Teilnehmerzahl oder dem geschätzten Netzwerkdurchmesser ist. Ebenso könnte eine minimale Netzwerkgröße definiert werden, ab der eine Cluster-Bildung überhaupt erst durchgeführt wird.

3x3-Knoten-Gitter: In der nächstgrößeren Topologie mit neun Teilnehmern entstanden vier verschiedene Cluster-Konstellationen (siehe Abb. 5.14), wobei stets der zentrale Knoten 7 zuverlässig die MCH-Rolle erhielt. Generell kamen im 3x3-Gitter nur die Knoten 2, 6, 8 und 12 als weitere CH-Kandidaten infrage, da sie nach Knoten 7 die höchste Nachbarknotenzahl aufwiesen (NC = 5). In der häufigsten Konstellation (Abb. 5.14 (a), 7/10 Messungen) gewannen die Knoten 6 und 8 aufgrund dominierender CENT-Metrik jeweils den Wettstreit mit den dazwischen liegenden Knoten 2 und 12. Die restlichen drei Konstellationen (Abb. 5.14 (b)–(d), je 1/10 Messungen) besaßen dagegen nur zwei Cluster und unterschieden sich in der Position des zweiten CH. Aufgrund von Schwankungen der ALM-Distanzen zwischen den Knoten, aus denen sich der CENT-Wert als Wichtungsfaktor der WNPR-Metrik ergibt, ging in diesen Messungen jeweils ein anderes Paar diagonal benachbarter PCHs mit dominanter WNPR ins Stechen, sodass nur ein einziger CH neben dem MCH entstand. Wie schon im 2x2-Aufbau traten die übrigen Knoten in allen Konstellationen bevorzugt dem MCH-Cluster bei. Eine Balancierung nach dem Erreichen von Phase 7 würde zum Roaming von Randknoten in die noch unbesetzten Cluster führen.

4x4-Knoten-Gitter: Wie im 2x2-Gitter existierte auch in dieser Topologie keine symmetrisch zentrale Knotenposition. Entsprechend waren die Teilnehmer des inneren Quadrats (Knoten 7, 8, 12 und 13) aufgrund ihrer dominierenden CENT-Metrik und maximalen Nachbarknotenzahl (NC = 8) die erwarteten Kandidaten für die Rolle des MCH und eines weiteren CH. In den insgesamt zehn Messungen entstanden infolge von Variationen der ALM-Distanzen neun verschiedene Konstellationen (siehe Abb. 5.15), die sich im gewählten CH-Paar und / oder den nachfolgenden Beitrittsentscheidungen der CMs unterschieden. Während Knoten in direkter Nachbarschaft zum MCH bevorzugt dessen Cluster beitraten, trafen weiter entfernte Knoten ihre Entscheidung wiederum auf Grundlage der ALM-Distanzen zu den CHs. Im Ergebnis entstanden in den Konstellationen 4, 7 und 9 (Abb. 5.15 (d), (g), und (i)) bereits Cluster mit ausgeglichener Teilnehmerzahl. Auch in Konstellation 6 (Abb. 5.15 (f)) lag die Differenz bei nur einem Knoten. Ein anschließendes CM-Roaming würde, wie auch in den übrigen fünf Konstellationen, zur Balancierung der Cluster-Größen führen.



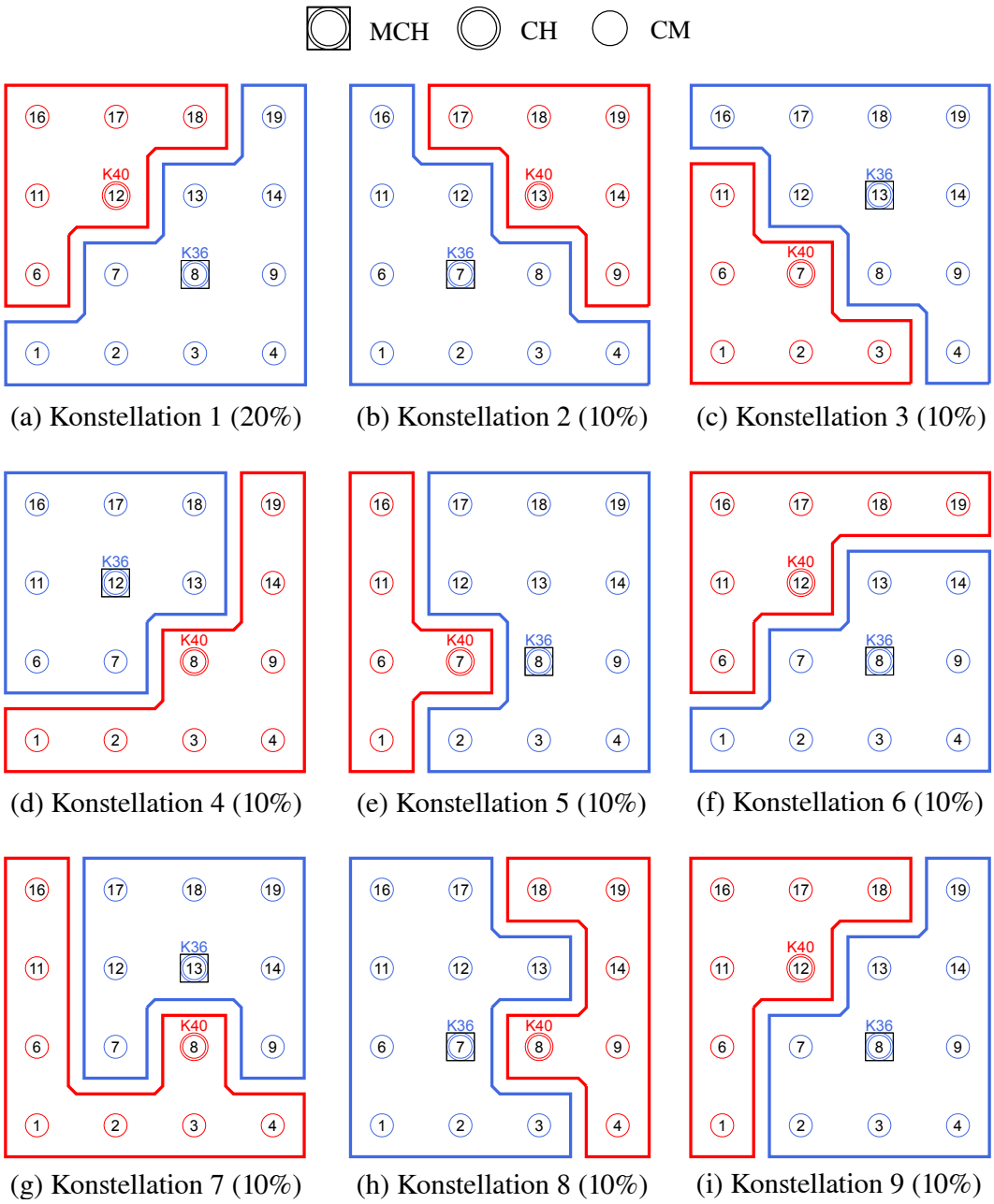
(a) Konstellation 1 (70%) (b) Konstellation 2 (10%)

(c) Konstellation 3 (10%) (d) Konstellation 4 (10%)

Abbildung 5.14: Cluster-Konstellationen im 3x3-Knoten-Gitter für Parametrisierung P2
(Cluster entsprechend gewähltem Kanal („K#“) farblich markiert; Auftrittshäufigkeit in %)

5x5-Knoten-Gitter: Wie bereits mit Parametrisierung P1 (siehe Kap. 5.5.2.3, Abb. 5.12) ergab sich auch für Parametrisierung P2 die in Abb. 5.16 (a) gezeigte gleiche häufigste Cluster-Konstellation in 9 von 10 Messungen. Unterschiede zum früheren Ergebnis bestanden nur in der CH-Reihenfolge bei der Kanalwahl in Phase 5, welche von der ALM-Distanz zwischen den CHs abhängt und daher leichten Metrikvariationen unterliegen kann. Die in Abb. 5.16 (b) gezeigte, in nur einer Messung entstandene Konstellation 2 beruht ebenfalls auf einer ALM-basierten Entscheidung. Hier wählte Knoten 11 in Phase 4 nicht CH 17 (rotes Cluster), sondern CH 7 (violettes Cluster) als den aus seiner Sicht nächstgelegenen CH. Aufgrund der im Verhältnis zu den vorherigen Topologien mehr als doppelten CH-Anzahl sowie der Mehrheit an Knoten außerhalb der Nachbarschaft des MCH entstanden im 5x5-Gitter bereits durch den Phasenablauf Cluster, deren Teilnehmerzahl sich nur um maximal zwei Knoten voneinander unterschied. Weitere Balancierungsmaßnahmen führen auch hier zu einem vollständigen Ausgleich, wie im nachfolgenden Kap. 5.5.3.1 für das 5x5-Gitter demonstriert wird.

Gegenüberstellung von Clustering-Dauer und Kommunikations-Overhead: Neben der funktionalen Untersuchung wurden Clustering-Zeitbedarf und Kommunikations-Overhead aller Szenarien verglichen. Tab. 5.9 stellt die mittlere Clustering-Dauer sowie TCP-/UDP-Datenmenge und -Paketanzahl der Topologien gegenüber. Die Ergebnisse des 5x5-Gitters für Parametrisierung P1 finden sich zum Vergleich in der letzten Tabellenzeile. Die Abbildungen 5.17 (a)–(c) zeigen die Datenpunkte für Parametrisierung P2 inkl. Standardabweichung. Für Parametrisierung P2 ergab sich mit der in Kap. 5.5.2.1 beschriebenen Überschlagsrechnung eine erwartete Clustering-Dauer von 50 s. Dabei ist hervorzuheben, dass die Abschätzung vereinfachte Annahmen für die eigentlich topologieabhängigen und zufallsbehafteten Zeitkomponenten T_{NewCH}, T_{KO}, und $T_{SelectChannel}$ trifft. Von diesen entfiel



(a) Konstellation 1 (20%) (b) Konstellation 2 (10%) (c) Konstellation 3 (10%)

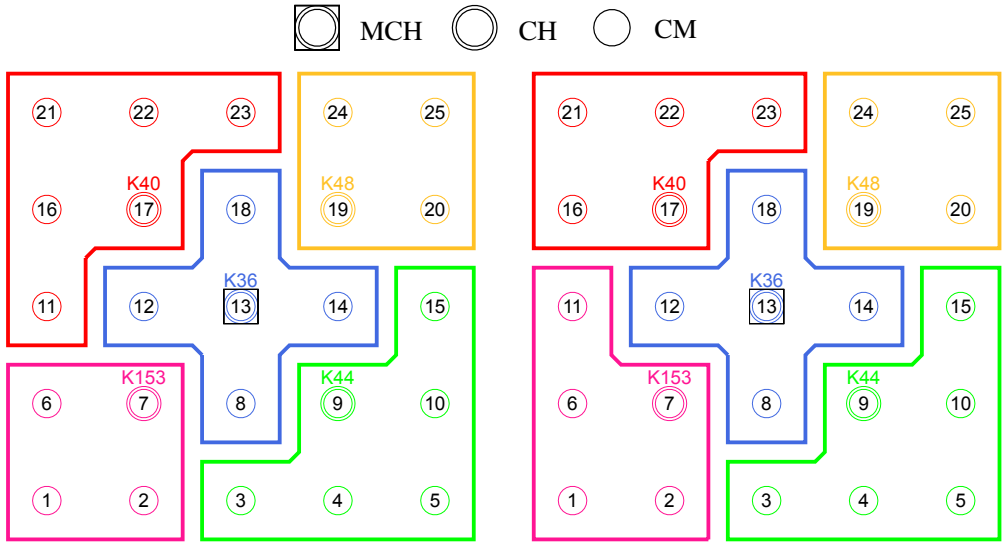
(d) Konstellation 4 (10%) (e) Konstellation 5 (10%) (f) Konstellation 6 (10%)

(g) Konstellation 7 (10%) (h) Konstellation 8 (10%) (i) Konstellation 9 (10%)

Abbildung 5.15: Cluster-Konstellationen im 4x4-Knoten-Gitter für Parametrisierung P2
(Cluster entsprechend gewähltem Kanal („K#“) farblich markiert; Auftrittshäufigkeit in %)

(wie schon mit Parametrisierung P1) die Zeit T_{NewCH}, da der Phasenablauf auf allen Knoten stets gleichzeitig gestartet wurde. Zudem war mit Parametrisierung P2 die initiale Prüfung auf bestehende Cluster in Phase 0 gänzlich deaktiviert. Dagegen waren T_{KO} (Konvergenzzeit der broadcast-basierten MCH-Wahl in Phase 0) sowie $T_{SelectChannel}$ (Zeitbedarf der CH-Kommunikation zur Kanalwahl in Phase 5) verantwortlich für eine mit der Netzwerkgröße und CH-Anzahl steigenden Clustering-Dauer. Darüber hinaus werden in der Abschätzung jegliche Ausführungsverzögerungen auf den Geräten vernachlässigt und somit auch der zeitliche Einfluss eines mit der Teilnehmerzahl steigenden Nachrichtenaufkommens, das durch die Knoten verarbeitet werden muss.

Im 2x2- bzw. 3x3-Knoten-Gitter waren Anteil und Variabilität der topologieabhängigen Zeitkomponenten noch gering und die gemessene, mittlere Clustering-Dauer mit 49,3 s bzw. 50,5 s entsprach



(a) Konstellation 1 (90%) (b) Konstellation 2 (10%)

Abbildung 5.16: Cluster-Konstellationen im 5x5-Knoten-Gitter für Parametrisierung P2
(Cluster entsprechend gewähltem Kanal („K#“) farblich markiert; Auftrittshäufigkeit in %)

nahezu dem überschlagenen Zeitbedarf von 50 s (siehe Abb. 5.17 (c)). Mit zunehmender Knotenzahl war ein langsames Ansteigen der Clustering-Dauer über die Schätzung hinaus zu beobachten. So wurden in der 4x4-Topologie bereits 52,9 s, im 5x5-Gitter 55,6 s für den Phasenablauf benötigt. Gleichzeitig stieg auch die Standardabweichung der Ergebnisse auf ca. 1 s (4x4-Gitter) bzw. 2 s (5x5-Gitter).

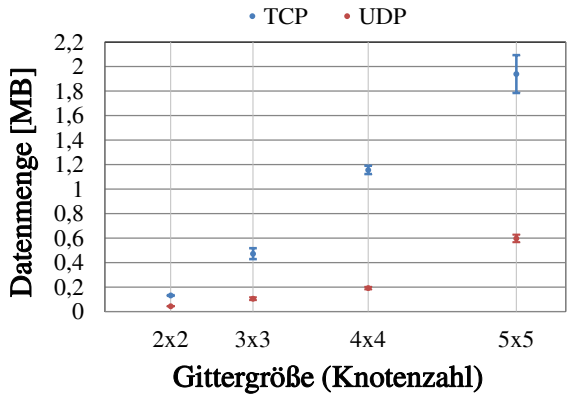
Im Hinblick auf den Kommunikations-Overhead ergaben sich die in Abb. 5.17 (a) dargestellten Trends für die gesendete TCP- und UDP-Datenmenge, wobei auch hier eine mit der Teilnehmerzahl steigende Standardabweichung der Ergebnisse zu beobachten war. Erwartungsgemäß zeigte der TCP-Trend eine nahezu lineare Abhängigkeit von der Teilnehmerzahl. Grund dafür ist, dass der zuverlässige Informationsaustausch mittels TCP bei CHaChA vorwiegend zwischen benachbarten Knoten erfolgt. Einzige Ausnahme sind JOIN-Nachrichten von CFNs an CHs in Phase 4 sowie CHAN_SEL-Nachrichten zwischen CHs in Phase 5, bei denen die Endpunkte auch mehrere Hops auseinander liegen können. Somit wird die TCP-Datenmenge kaum durch die Netzwerkausdehnung, sondern maßgeblich durch die Gesamtknotenmenge und Anzahl der Nachbarn pro Knoten bestimmt.

Beim UDP-Trend bestätigte sich die erwartete Abhängigkeit von der Gittergröße und CH-Anzahl. In der Prototyp-Implementierung von CHaChA dient UDP ausschließlich als Transportprotokoll für Broadcast-Nachrichten, die im ganzen Netzwerk verbreitet werden. Mit dem Netzwerkdurchmesser steigt auch die mittlere Pfadlänge zwischen den Knoten, sodass mehr Weiterleitungsschritte durch jeden Broadcast verursacht werden. Ab Phase 3 erfolgt zudem der durchgängige, periodische Versand von CH-Ankündigungsnachrichten, die somit maßgeblich zur UDP-Datenmenge beitragen. Mit steigendem Netzwerkdurchmesser kandidieren potentiell mehr PCHs, wodurch potentiell auch mehr CHs entstehen. So war zwischen dem 4x4- und 5x5-Gitter ein Sprung von zwei auf fünf CHs zu verzeichnen, der zu einem entsprechendem Anstieg der UDP-Datenmenge führte.

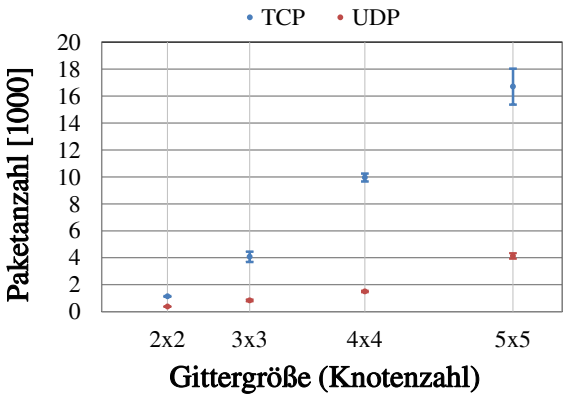
Sowohl für UDP als auch TCP korrespondiert der Verlauf der Datenmenge mit dem der Paketanzahl (siehe Abb. 5.17 (b)), wobei jegliche Multi-Hop-Weiterleitungen und durch TCP erzeugte Kontrollnachrichten enthalten sind. Darin begründet liegt auch der gegenüber UDP stärkere Anstieg der TCP-Datenmenge, in die zusätzlich Handshake- und ACK-Nachrichten eingehen. Im 5x5-Gitter betrugen die TCP-Datenmenge und -Paketanzahl gut das Drei- bis Vierfache der UDP-Ergebnisse. In den untersuchten Topologien hatte die Zunahme individueller Paketgrößen nur eine geringe Auswir-

Tabelle 5.9: Vergleich von Clustering-Dauer und Kommunikations-Overhead

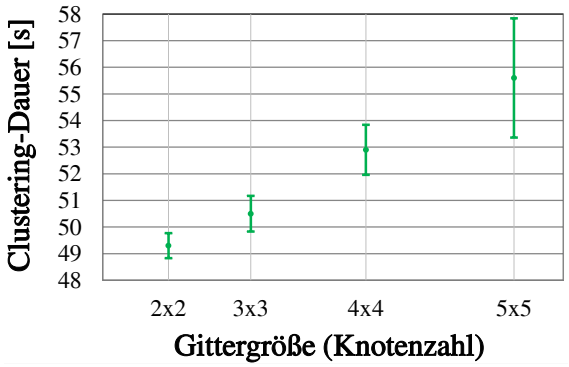
Topologie	Geschätzte Dauer [s]	Gemessene Dauer [s]	TCP-Datenmenge [kB]	TCP-Paketanzahl	UDP-Datenmenge [kB]	UDP-Paketanzahl
2x2-Gitter (P2)	50	49,3	131,3	1131	44,2	374
3x3-Gitter (P2)	50	50,5	472,8	4069	105,2	835
4x4-Gitter (P2)	50	52,9	1156,6	9952	191,5	1498
5x5-Gitter (P2)	50	55,6	1929,2	16695	597,2	4133
5x5-Gitter (P1)	130	135,5	1113,7	9699	627,8	5248



(a) TCP- und UDP-Datenmenge



(b) TCP- und UDP-Paketanzahl



(c) Clustering-Dauer

Abbildung 5.17: Kommunikations-Overhead und Clustering-Dauer abhängig von der Gittergröße/Knotenzahl (Parametrisierung P2; Datenpunkte mit Standardabweichung)

kung auf die Gesamtdatenmenge. Neben den CHAN_SEL-Nachrichten der Kanalwahlsequenz in Phase 5, deren Größe von der Anzahl der CHs abhängt, sind die ab Phase 4 gesendeten Broadcast-Ankündigungen der CHs die einzigen Nachrichten, deren Größe von der Netzwerktopologie abhängt, da sie die Liste der CMs des jeweiligen Clusters enthalten.

Die mit der Parametrisierung P2 im 5x5-Gitter über eine Clustering-Dauer von rund 55,6 s erzeugte TCP-/UDP-Gesamtdatenmenge von etwa 2526 kB kann erneut vereinfacht als Overhead-Datenverkehr mit einer effektiven Datenrate von ca. 360 kbit/s betrachtet werden. Bezogen auf den mit der WLAN-Datenrateneinstellung MCS 3 gemessenen Single-Hop-Nettodurchsatz (22,9 Mbit/s

UDP bzw. 17,8 Mbit/s TCP) stellt dies weiterhin nur etwa 1,5–2 % der verfügbaren Kommunikationsressourcen eines Mesh-Links in der Testumgebung dar. Grundsätzlich muss durch die CHaChA-Parametrisierung ein Kompromiss zwischen Clustering-Dauer und Kommunikationskosten gefunden werden. Für einen geringen Langzeit-Overhead, auch über den Phasenablauf hinaus, ist insbesondere die Wahl einer geeigneten Sendeperiode für die CH-Broadcast-Nachrichten erforderlich. Gleichzeitig sollte dabei die Responsivität der Mechanismen zur dynamischen Cluster-Anpassung gewahrt werden, deren Entscheidungen teils auf den Informationen der CH-Nachrichten basieren. Eine generelle Senkung der Nachrichtengrößen könnte künftig durch eine kompaktere Kodierung oder mittels Kompressionsverfahren erreicht werden. Weiterhin ist, je nach gewünschtem Grad der Zuverlässigkeit, die Nutzung von UDP auch für Unicast-Nachrichten zwischen Nachbarknoten denkbar.

5.5.3 Erprobung der dynamischen Cluster-Anpassung

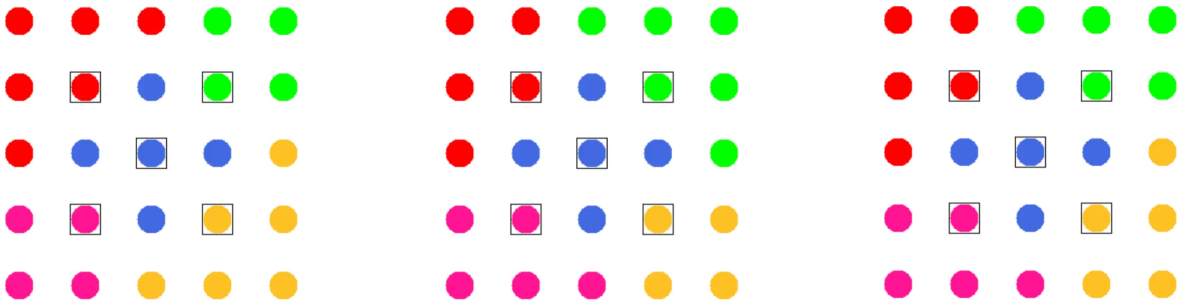
Nach der ausführlichen Analyse des CHaChA-Phasenablaufs zur initialen Cluster-Bildung wurden auch die in Kap. 5.3.4 beschriebenen Konzepte zur dynamischen Cluster-Anpassung im Langzeitbetrieb mithilfe einfacher Beispielszenarien funktional erprobt. Die zuvor genutzte Parametrisierung P2 (siehe Tab. 5.8) der Prototyp-Implementierung wurde beibehalten. Durch Vorgabe einer Cluster-Konstellation als Ausgangskonfiguration sprangen zudem alle Knoten beim Programmstart sofort in Phase 7, in der die Mechanismen zur Cluster-Anpassung griffen.

Zusätzlich zur Offline-Visualisierung der Cluster-Konstellation aus den Statusdateien der Knoten war es für die nachfolgenden Experimente hilfreich, aktuelle Cluster-Informationen zur Laufzeit zu erhalten und grafisch darzustellen. Eine zu diesem Zweck in Java entwickelte Anwendung [5.7] wurde auf einem separaten Labor-PC ausgeführt und erzeugte auf Grundlage von über Ethernet empfangenen Nachrichten ein Bild des aktuellen Cluster-Status. Dafür sendeten alle CHs ihre Cluster-Informationen (WLAN-Kanal und Liste der CMs) nicht nur periodisch als Broadcast im Mesh-Netzwerk, sondern gleichzeitig als Unicast über Ethernet an den PC. Somit war die Aktualität der Cluster-Informationen auf den Knoten und dem PC durch das Sendeintervall der CH-Nachrichten (CH_PERIOD) gegeben, wobei die Geräte in den Experimenten zeitgleich gestartet wurden. Wie bereits für die Offline-Visualisierung waren auch hier die Gitterpositionen der Knoten statisch vorgegeben. Anhand der empfangenen Informationen über die Rolle und Cluster-Zugehörigkeit eines Knotens wurde dieser eingeblendet und farblich markiert. Zur Diagnose der Vorgänge wurden Videos der Desktop-Anwendung während der Experimentdurchführung aufgezeichnet. Nachfolgend dargestellt sind Momentaufnahmen der Zeitpunkte, in denen Cluster-Änderungen in der Anwendung sichtbar wurden (Zeitauflösung 1 s).

5.5.3.1 Cluster-Balancierung durch Roaming

In einer ersten Experimentreihe wurde die nachträgliche Cluster-Balancierung durch autonomes Roaming von CMs untersucht. Wie im Konzept in Kap. 5.3.4.3 und 5.3.4.4 beschrieben treffen CMs ihre Roaming-Entscheidungen basierend auf den aktuellen Cluster-Informationen, die sie über die periodischen Broadcast-Nachrichten der CHs erhalten. Dabei kommen nur CMs für einen Wechsel infrage, die sich im Randbereich zu anderen Clustern befinden und für Mitglieder ihres Clusters keine Verbindungsknoten auf dem Pfad zum eigenen CH darstellen. Dazu tauschen alle CMs ab Phase 7 periodisch NH2CH-Unicast-Nachrichten in ihrer Nachbarschaft aus. Diese enthalten die Zusatzinformation, über welchen Nachbarknoten ihr Mesh-Pfad zum CH verläuft, sodass jeder CM seine Eignung für einen Cluster-Wechsel überprüfen kann. In allen Experimenten wurde das NH2CH-Sendeintervall identisch zur Broadcast-Periode der CH-Nachrichten gewählt (2 s für Parametrisierung P2). Da die Geräte der Testumgebung nur mit einem WLAN-Adapter ausgestattet waren, die

[5.7] `https://gitlab.amd.e-technik.uni-rostock.de/mesh/chacha-monitor`



Sekunde 0 Sekunde 3 Sekunde 6

Abbildung 5.18: Cluster-Balancierung im 5x5-Knoten-Gitter

Cluster-Gruppierung also nur logisch und ohne Konfiguration eines Sekundär-Interfaces erfolgte, wurden lediglich NH2CH-Informationen des Primär-Interfaces ausgetauscht. Dabei war es theoretisch möglich, dass der Pfad zum eigenen CH auch über Knoten eines Nachbar-Clusters verläuft. Im untersuchten Beispiel befanden sich jedoch alle CMs in direkter Nachbarschaft zu ihrem CH, sodass stets Pfade ohne Zwischenknoten entstanden. Somit konnten alle Knoten im Randbereich zu anderen Clustern Roaming-Vorgänge durchführen. Laut Balancierungskonzept (siehe Kap. 5.3.4.4) initiieren CMs im Randbereich zwischen Clustern nur dann ein Roaming-Gesuch, wenn die Größe ihres Clusters mindestens 2 Knoten über der eines Nachbar-Clusters liegt. Im Zuge des Handshakes mit altem und neuem CH werden zudem gleichzeitige Wechsel (innerhalb einer CH_PERIOD) mehrerer CMs aus demselben Ursprungs- in dasselbe Ziel-Cluster verhindert, um Lawineneffekten vorzubeugen.

Abb. 5.18 zeigt den Verlauf eines Experiments im 5x5-Knoten-Gitter, bei dem die mit den Parametrisierungen P1 und P2 am häufigsten entstandene Cluster-Konstellation als Startkonfiguration vorgegeben war. Die fünf CHs sind jeweils mit einem schwarzen Quadrat umrandet, wobei der MCH in der Gittermitte nicht gesondert hervorgehoben ist. Wie in den zuvor offline erzeugten Grafiken sind auch hier alle Teilnehmer eines Cluster in derselben Farbe dargestellt. Direkt nach Initialisierung der Konstellation in „Sekunde 0“ waren das rote und gelbe Cluster jeweils um 2 Knoten größer als das violette und grüne Cluster. Lediglich das zentrale blaue Cluster entsprach bereits der anvisierten Durchschnittsgröße mit einer Teilnehmerzahl von 5 Knoten. In Sekunde 3 des Experiments wurde ersichtlich, dass ein CM des roten in das grüne Cluster gewechselt war. Unabhängig davon waren zwei CMs des gelben Clusters zeitgleich in das grüne bzw. violette Cluster gewechselt. Dies war zulässig, da es sich um verschiedene Ziel-Cluster handelte, führte nun jedoch zu einem CM-Überhang im grünen Cluster. Nachfolgend sprang der zuvor in das grüne Cluster gewechselte CM zurück in das gelbe Cluster, wodurch die Balancierung in Sekunde 6 abgeschlossen war.

In diesem Beispiel lag die Ausgangskonstellation bereits nah am Balancierungsziel und es waren nur wenige Roaming-Schritte nötig, um die Cluster-Größen auszugleichen. Eine Konvergenz trat bereits nach zwei Roaming-Zyklen ein, wobei die Taktung der CM-Entscheidungen mit dem Broadcast-Intervall der CH-Nachrichten einhergeht. In größeren Topologien, die nach initialer Cluster-Bildung ein stärkeres Ungleichgewicht aufweisen, werden möglicherweise deutlich mehr Roaming-Schritte zur Balancierung benötigt. Das Zulassen paralleler Roaming-Vorgänge, wie die der CMs des gelben Clusters zwischen Sekunde 0 und 3, kann dabei die Konvergenzdauer einerseits positiv, andererseits auch negativ beeinflussen und bietet somit Spielraum für weitere Optimierungen.

5.5.3.2 Nachinstallation und Umplatzierung von Knoten

Weitere Experimente widmeten sich einem Wartungsszenario, in dem eine Kombination der Mechanismen zur dynamischen Cluster-Anpassung ausgelöst wird. Untersucht wurde die Nachinstallation

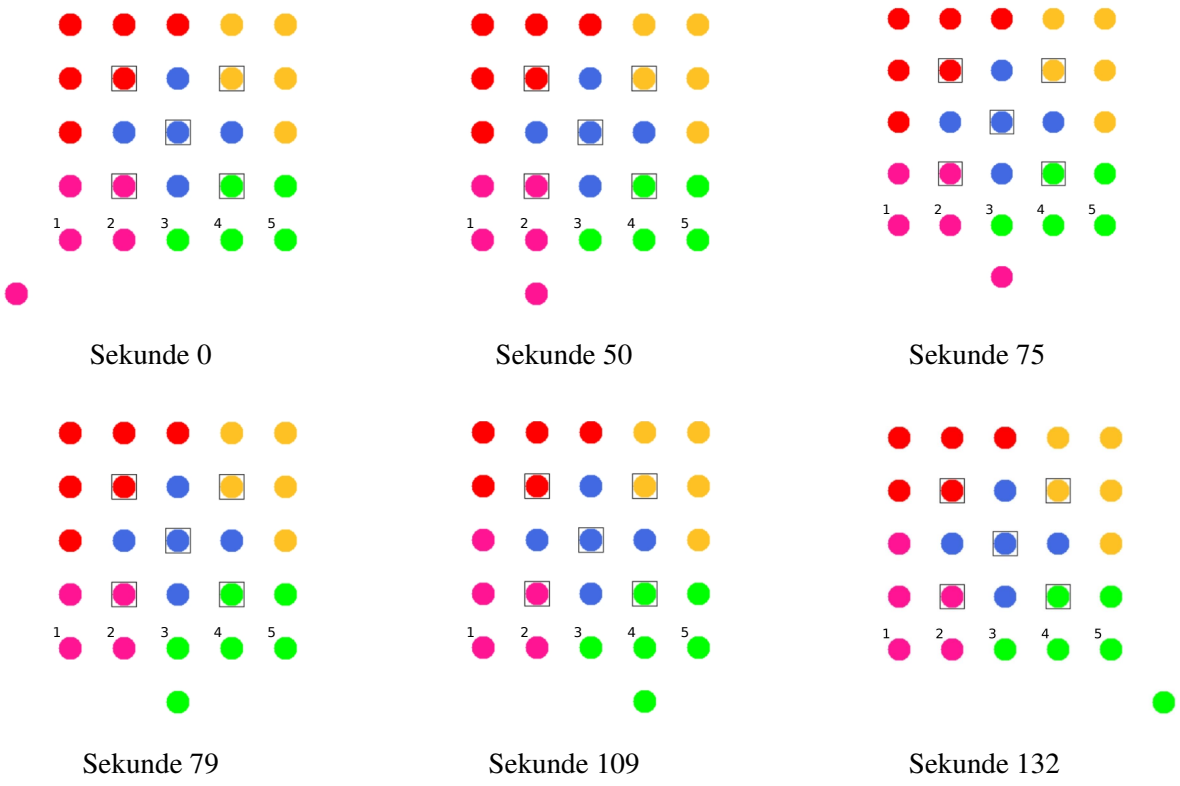


Abbildung 5.19: Virtuelle Nachinstallation und Umplatzierung eines Knotens

eines Knotens in eine bestehende Cluster-Konstellation. Dies erforderte zunächst den automatischen Cluster-Beitritt des Knotens entsprechend dem Konzept in Kap. 5.3.4.1. Anschließend erfolgte eine (virtuelle) Umplatzierung des Knotens, bis der Verbindungsverlust zum ursprünglichen CH eintrat. Die Isolationserkennung (Konzept in Kap. 5.3.4.2) auf dem Zusatzknoten führte daraufhin zum Wechsel in ein neues Cluster, was wiederum eine Cluster-Balancierung durch Roaming anderer Teilnehmer hervorrief (Konzept in Kap. 5.3.4.4). Im Kontext städtischer Mesh-Backbones könnten die Nachinstallation und Umplatzierung von Knoten z. B. einer erhöhten Abdeckung oder Ausfallsicherheit des Netzwerks dienen. Während des Experiments war der Zusatzknoten fest neben der Testumgebung platziert. Im Gegensatz zu den anderen Geräten waren seine Antennen nicht mit Dämpfungsgliedern versehen, wodurch zu jedem anderen Knoten eine Direktverbindung möglich war. Per WLAN-Interface-Konfiguration unter Linux wurde jedoch der Link-Aufbau zu allen bis auf einen der Knoten 1–5 in Abb. 5.19 blockiert. Beginnend bei Knoten 1 erfolgte die virtuelle Umplatzierung des Zusatzknotens entlang des unteren Netzwerkrands jeweils durch Blockieren der vorherigen und Zulassen der nächsten Verbindung. Analog zu den Broadcast-Nachrichten der CHs sendete der Zusatzknoten im 2 s-Abstand seine NH2CH-Nachrichten sowohl im Mesh-Netzwerk als auch per Ethernet an den Kontroll-PC. Basierend auf der Information, über welchen der Knoten 1–5 sein Pfad zum CH verläuft, wurde die Position des Zusatzknotens in der Visualisierungs-Software angepasst.

Abb. 5.19 zeigt die Ergebnisse einer Experimentdurchführung. Als vordefinierte Ausgangskonfiguration diente erneut die häufigste Cluster-Konstellation der 5x5-Gittertopologie. Abweichend von der ansonsten gültigen Parametrisierung P2 war auf dem Zusatzknoten die Prüfung auf bestehende Cluster in Phase 0 aktiviert. Nach Initialisierung des 5x5-Gitters war dem verzögerten Programmstart des Zusatzknotens zunächst ein Balancierungsvorgang zwischen grünem und gelbem Cluster vorausgegangen. Die ebenfalls anstehende Balancierung zwischen rotem und violettem Cluster war jedoch noch nicht erfolgt. Bei seinem Start besaß der Zusatzknoten bereits eine vorkonfigurierte Verbindung zum Netzwerk über Knoten 1. Nach Auswertung der eingehenden CH-Ankündigungen in Phase 0

wählte er den Beitritt zum violetten Cluster, dessen CH die geringste ALM-Distanz aufwies. Dadurch entstand eine vorerst balancierte Cluster-Konstellation.

Sekunde 0 in Abb. 5.19 zeigt den Zeitpunkt, in dem der Zusatzknoten dem Netzwerk erfolgreich beigetreten war und seine Position sowie Cluster-Zugehörigkeit in der Visualisierung als Ergebnis der empfangenen CH- und NH2CH-Nachrichten dargestellt wurden. Ab diesem Zeitpunkt wurde vom Kontroll-PC die schrittweise Umplatzierung des Knotens vorgenommen. Das Umschalten der Verbindung zum jeweils Nächsten der Knoten 1–5 erfolgte skriptbasiert im Abstand von 30 s.

In Sekunde 50, also 20 s nach der Link-Umschaltung zu Knoten 2, wurde die Positionsänderung in der Visualisierung sichtbar. Diese Zeitverzögerung ist einerseits bedingt durch die Parametrisierung des CHaChA-Protokolls und der 802.11s-Sicherungsschicht, andererseits durch die Zustellungswahrscheinlichkeit von 802.11s-Kontrollnachrichten. Entsprechend dem CHaChA-Konzept nutzen CHs auf dem Basis- sowie Cluster-Kanal den proaktiven Modus des 802.11s-Routing-Protokolls HWMP, sodass alle Knoten stets Pfadinformationen zu ihnen vorhalten. Das Sendeintervall der proaktiven *PREQ*-Frames zur Pfadaktualisierung ist über den 802.11s-Parameter `HWMProotInterval` konfigurierbar. Im Rahmen dieser Arbeit wurde dessen Standardwert von 5 s beibehalten (siehe Kap. 2.5.7 für eine Übersicht der 802.11s-Parameter). Zwar erkennen Knoten Link-Abbrüche in der Regel bereits nach einer *Beacon*-Periode (Standardwert 1 s), die Aktualisierung des Pfadeintrags zum CH, dessen Next-Hop-Information möglicherweise obsolet geworden ist, erfolgt jedoch erst nach Empfang des nächsten PREQ-Frames. Da PREQs stets als unbestätigte Broadcasts gesendet werden, besitzen sie im Vergleich zu Unicast-Frames eine geringere Zustellungswahrscheinlichkeit. Das Ausbleiben eines PREQ-Frames verzögert somit die Pfad-Aktualisierung um weitere 5 s.

Nach der Link-Umschaltung auf dem Zusatzknoten muss dieser das Pfad-Update nicht nur auf der Sicherungsschicht, sondern auch in der CHaChA-Anwendung feststellen, bevor er die aktuelle NH2CH-Information an den Kontroll-PC senden kann. Im CHaChA-Prototyp werden die 802.11s-Informationen des Linux-Kernels alle 2 s abgefragt. Je nach Versatz des PREQ-Intervalls (5 s) und Sampling-Zyklus wird eine Pfadänderung somit erst nach 6–8 s in der Anwendung des Knotens erkannt und nach weiteren 2 s entsprechend dem Sendeintervall der NH2CH-Nachrichten visualisiert. Hinzu kommen Verzögerungen durch nicht zugestellte PREQ-Nachrichten. Im hier diskutierten Experiment wurden Link-Wechsel nach 12–20 s in der Visualisierung sichtbar.

Sekunde 75 in Abb. 5.19 zeigt die erkannte Link-Umschaltung von Knoten 2 auf Knoten 3, die in Sekunde 60 durchgeführt worden war. Durch Überschreitung der Cluster-Grenze stellte die Isolationserkennung des Zusatzknotens den Verbindungsverlust zum CH des violetten Clusters fest, wodurch ein Wechsel in das grüne Cluster eingeleitet wurde. Da in allen Experimenten nur ein WLAN-Adapter als Primär-Interface genutzt wurde und alle Knoten auf dem Basiskanal verbunden waren, konnte der Pfad zum violetten CH auch weiterhin über den CM eines anderen Clusters aufgebaut werden. Somit war keine Isolationserkennung basierend auf Informationen des Sekundär-Interfaces möglich, die den Pfadverlust zum CH auf dem Cluster-Kanal signalisieren würde. Abweichend vom Konzept in Kap. 5.3.4.2 erfolgte die Erkennung stattdessen durch Überprüfung, ob der Pfad zum aktuellen CH über einen CM des gleichen Clusters verläuft. Die Teilnehmer aller Cluster werden in den CH-Broadcasts im Netzwerk angekündigt und waren somit auch dem Zusatzknoten bekannt. Den erfolgreichen Beitritt in das grüne Cluster zeigt Sekunde 79, nachdem aktuelle CH-Nachrichten aller Cluster an den Kontroll-PC gesendet worden waren. Die nun verringerte Teilnehmerzahl des violetten Clusters bewirkte anschließend das Roaming eines Randknotens aus dem roten Cluster, was zur erneuten Balancierung der Konstellation führte. Wie in Abb. 5.19 dargestellt, wurden die Link-Umschaltungen zu Knoten 4 und 5 nach 19 bzw. 12 s in der Visualisierung sichtbar.

Zusammenfassend konnten die Konzepte für den nachträglichen Netzwerkbeitritt sowie die Isolationserkennung und Cluster-Balancierung erfolgreich im Zusammenspiel erprobt werden. Während der Fokus dieser Experimente auf der funktionalen Untersuchung lag, kann die Wahl kürzerer Update-

Intervalle (CHaChA- und 802.11s-Parameter) eine für den praktischen Einsatz geforderte Responsivität der Erkennung von Netzwerkänderungen erhöhen. Je nach Anwendungsfall muss dabei ein Kompromiss zwischen Latenz und Nachrichtenaufkommen gefunden werden.

5.5.4 Performance-Vorteile am Anwendungsbeispiel

Ein potenzieller Anwendungsfall für CHaChA ist die Optimierung eines städtischen Mesh-Backbones oder Zugangsnetzes. In diesem Zusammenhang müssen administrative Aufgaben wie die Statusüberwachung des Netzwerks ausfallsicher und skalierbar gestaltet werden. Nachfolgend wird anhand einer Monitoring-Anwendung demonstriert, welche Performance-Vorteile aus der Cluster-Bildung resultieren. Gegenstand der Experimente war die zyklische Abfrage von Statusinformationen aller Knoten und deren Zusammenführung an einem Knoten. In der Praxis kann der Netzwerkbetreiber auf Basis einer so gewonnenen globalen Sicht Fehler erkennen und Wartungsaufgaben durchführen.

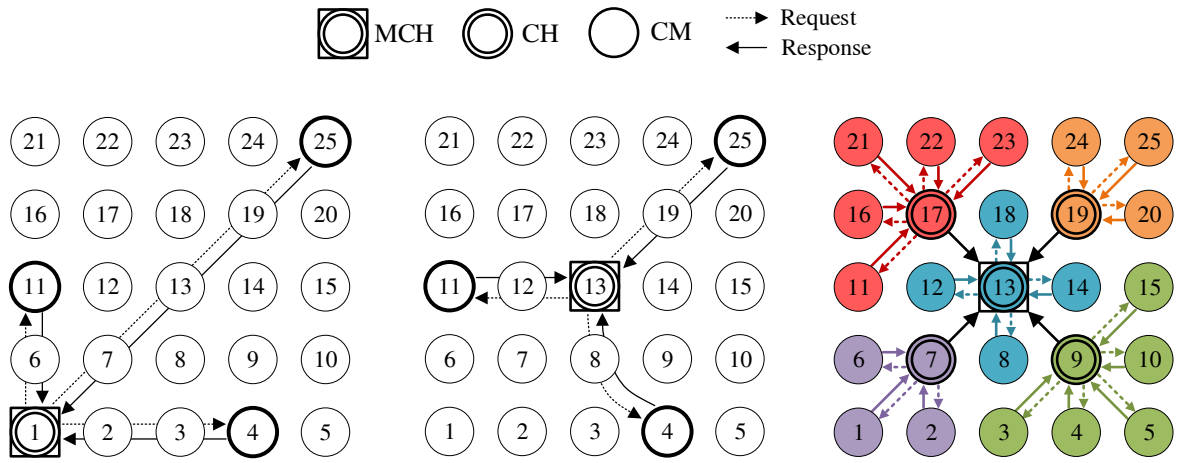
Als Beispielnetzwerk diente erneut die 5x5-Knoten-Gittertopologie. Untersucht wurde zunächst ein zentralisiertes Single-Channel-Szenario unter Berücksichtigung der Worst- und Best-Case-Platzierung eines dedizierten Monitoring-Knotens. Der zentralisierte Fall wurde anschließend einem dezentralen Szenario gegenübergestellt. In diesem erfolgte das Monitoring in vorgegebenen Clustern entweder auf demselben Kanal (Single-Channel-Szenario) oder auf überlappungsfreien Kanälen (Multi-Channel-Szenario). Es wurde die Cluster-Konstellation genutzt, die im 5x5-Knoten-Gitter am häufigsten durch CHaChA gebildet wurde. In diesen Experimenten übernehmen die CHs die Statusabfrage ihrer CMs und übertragen die gesammelten Daten anschließend zum MCH, der als Synchronisationspunkt dient. Für eine vereinfachte Ergebnisdiskussion werden die CHaChA-Knotenrollen auch zur Darstellung des zentralisierten Falls genutzt. Dabei wird das gesamte Netzwerk als ein großes Cluster aufgefasst, d.h. die abgefragten Knoten werden ebenfalls als „CMs“, der dedizierte Monitoring-Knoten als „MCH“ bezeichnet.

Als Performance-Metrik diente die *Zykluszeit* eines Monitoring-Laufs, welche die Dauer der Statusabfrage aller CMs samt deren Aggregation am MCH angibt. In allen Experimenten war eine feste 802.11-Bruttodatenrate konfiguriert (MCS 3: 26 Mbit/s). Die CMs lieferten jeweils Statusinformationen fester Größe (Beispieldatei mit 250 kB Zufallsdaten), die unter Nutzung des Kommandozeilenprogramms `ucspi-tcp tcpserver` [245] als Server-Instanz pro CM bereitgestellt wurden. Auf dem MCH (und den CHs im dezentralen Szenario) wurde `netcat` [246] als Client-Anwendung zur Abfrage der Daten verwendet. Für jeden abzufragenden CM wurde ein separater `netcat`-Prozess gestartet und alle Statusdaten wurden zu Beginn jedes Zyklus gleichzeitig über TCP angefordert.

5.5.4.1 Zentralisiertes Status-Monitoring

Im zentralisierten Referenzszenario wurden, wie in Abb. 5.20 (a) und (b) markiert, eine Worst- und Best-Case-Platzierung des dedizierten Monitoring-Knotens („MCH“) hinsichtlich der mittleren Hop-Distanz zu allen CMs unterschieden. Knoten 1 in der Netzwerkecke diente dabei als Worst-Case- und Knoten 13 in der Mitte als Best-Case-MCH.

Zur Abschätzung des Kommunikationsaufwands und der zu erwartenden relativen Zeitersparnis zwischen Worst- und Best Case wurde die Mindestanzahl an Single-Hop-Übertragungen betrachtet, die jeweils erforderlich ist, um die Statusdaten aller CMs zum MCH zu transportieren. Die Abschätzung berücksichtigt lediglich die Antwortrichtung von CM zu MCH. Es wird zudem vereinfacht angenommen, dass für jeden Datensatz nur ein Sendevorgang pro Hop notwendig ist. Dies ignoriert clientseitige Request-Nachrichten, deren Größe in Relation zu den angefragten Daten jedoch vernachlässigbar ist, sowie etwaige 802.11-Neuübertragungen und jegliche TCP-Mechanismen (Handshake, Fluss-/Überlastkontrolle). Mit Knoten 1 als MCH in der Ecke des Netzwerks ergibt sich so eine



(a) Zentralisiert (Worst Case) (b) Zentralisiert (Best Case) (c) Dezentral in Clustern

Abbildung 5.20: Verglichene Monitoring-Szenarien: (a) zentralisierter Worst Case mit Knoten 1 als „MCH“; (b) zentralisierter Best Case mit Knoten 13 als „MCH“; (c) dezentrale Abfrage durch CHs in Clustern mit anschließender CH-zu-MCH-Synchronisation (schwarze Pfeile); Teilabb. (a) & (b) zeigen nur eine Auswahl der Kommunikationsvorgänge zu CMs in unterschiedlicher Hop-Distanz

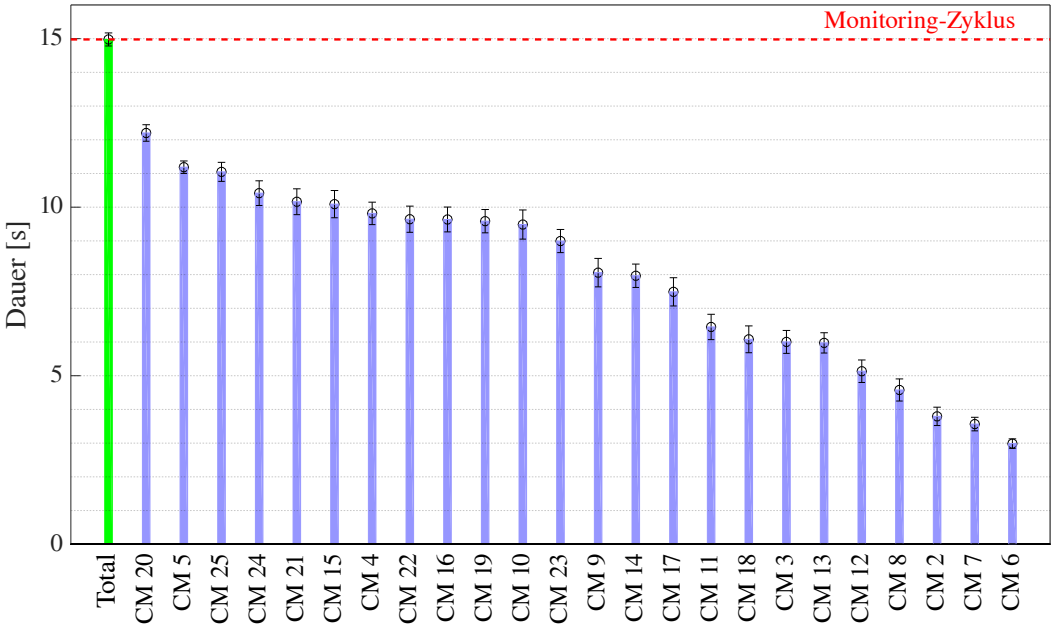
Summe der Übertragungen von 70. Mit Knoten 13 als MCH im Zentrum des Netzwerks beträgt die Summe dagegen 40, was einer Ersparnis von ca. 43 % entspricht.

Die individuelle Download-Dauer jeder `netcat`-Instanz auf dem MCH wurde über den Linux-Befehl `time` mit Millisekunden-Genauigkeit gemessen. Zudem wurden Zeitstempel vor Beginn und nach erfolgreicher Beendigung aller Downloads mithilfe des Befehls `date` genommen. Die Gesamtdauer eines Monitoring-Durchlaufs wurde als Zeitdifferenz zwischen der Rückkehr der letzten `netcat`-Instanz und dem Beginn des Experiments berechnet. Daher sind Scheduling-Latenzen für die Ausführung aller `netcat`-Prozesse mit eingeschlossen.

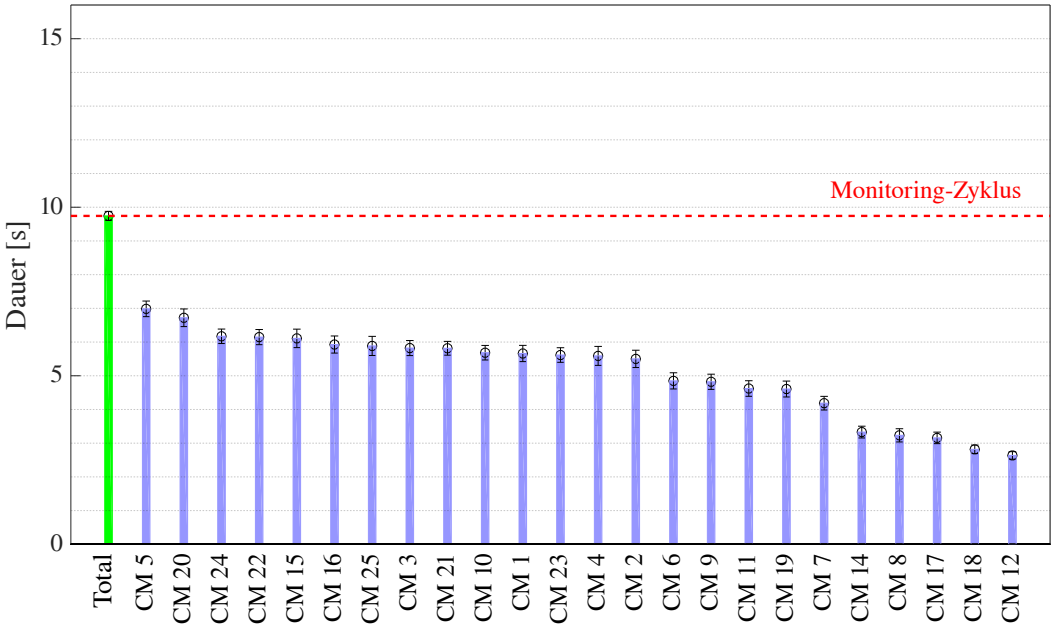
Abb. 5.21 zeigt die Zykluszeit sowie die individuelle Download-Dauer pro CM für den zentralisierten Worst- (a) und Best Case (b). Gezeigt werden die Mittelwerte aus 50 Monitoring-Durchläufen inkl. ihrer Standardabweichung. Die individuellen Download-Zeiten sind durch blaue Balken, die Gesamtdauer des Downloads der CM-Daten „CM DL“ durch einen grünen Balken dargestellt. Im zentralisierten Szenario entspricht dies direkt der Zykluszeit, die zusätzlich als rot gestrichelte horizontale Linie eingetragen ist. Im Worst Case mit Knoten 1 als MCH betrug die Zykluszeit ca. 15 s. Im Best Case mit Knoten 13 als MCH wurden knapp 10 s und somit eine Zykluszeitreduktion von 35 % beobachtet, was einer 8 %-Abweichung von der geschätzten Ersparnis (43 %) entspricht. Erwartungsgemäß waren in beiden Fällen die Download-Zeiten der jeweils näher am MCH positionierten CMs überwiegend kürzer, da weniger Weiterleitungen für die Nachrichten benötigt wurden. Die Best-Case-Zykluszeit von 10 s dient nachfolgend als Referenz für den verteilten Monitoring-Ansatz.

5.5.4.2 Verteiltes Status-Monitoring in Clustern

Für das dezentrale Monitoring-Szenario wurde die in Abb. 5.20 (c) dargestellte Cluster-Konstellation genutzt, welche aus einem MCH und vier CHs bestand. Diese entsprach dem häufigsten Ergebnis des CHaChA-Phasenablauf im 5x5-Aufbau ohne Anwendung einer nachträglichen Balancierung. Die Konstellation wurde in einem Single-Channel- und einem Multi-Channel-Setup evaluiert, wobei in Letzterem alle Cluster auf überlappungsfreie Kanäle konfiguriert wurden. Im Gegensatz zum zentralisierten Szenario setzte sich ein Monitoring-Zyklus aus zwei aufeinander folgenden Schritten zusammen. In Schritt 1 sammelten MCH und CHs gleichzeitig die CM-Statusdaten ihres Clusters,



(a) Worst Case: Datenabfrage durch Knoten 1 als „MCH“



(b) Best Case: Datenabfrage durch Knoten 13 als „MCH“

Abbildung 5.21: Monitoring-Dauer im zentralisierten Referenzszenario [B 3]
(Zykluszeit und individuelle Download-Zeit pro CM jeweils mit Standardabweichung)

wodurch sich analog zum zentralisierten Szenario eine Download-Zeit „CM DL“ pro Cluster ergab. Nachdem jeder CH seinen Download beendet hatte, übermittelte er sofort seine gesammelten Cluster-Informationen an den MCH. Dazu nutzten alle CHs eine Beispieldatei in der Größe der CMs-Daten ergänzt um ihre eigenen Statusdaten. Wie im zentralisierten Szenario galt ein Monitoring-Zyklus als abgeschlossen, sobald alle Daten beim MCH eingegangen waren.

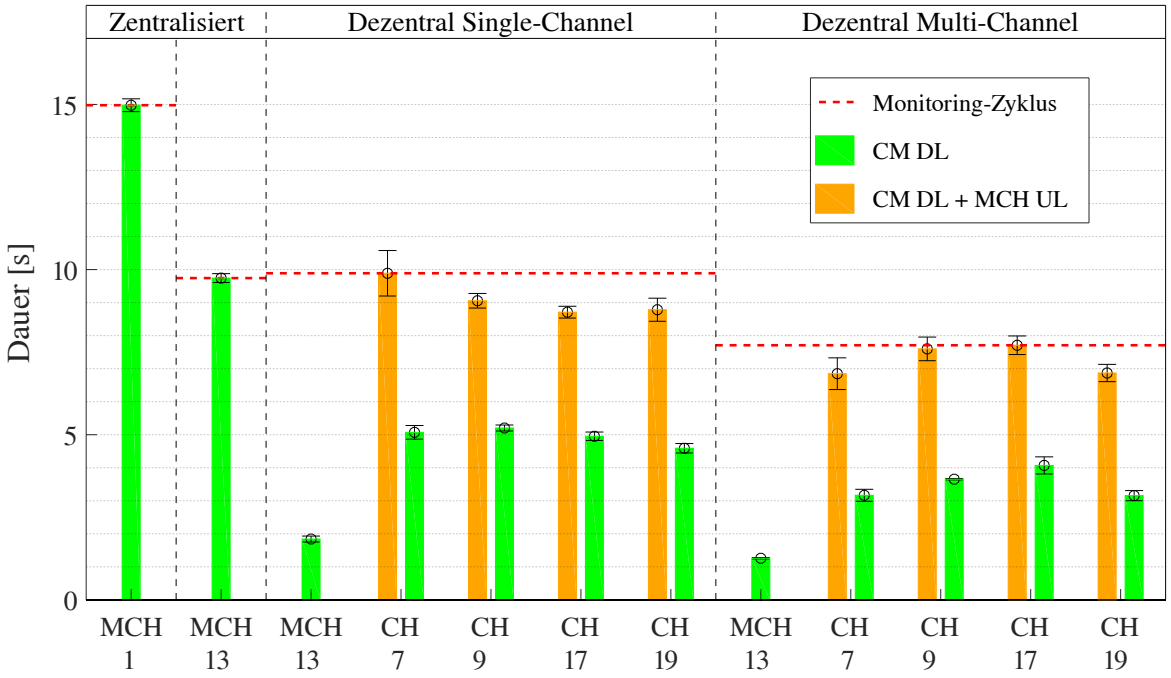


Abbildung 5.22: Vergleich von zentralisiertem und dezentralem Monitoring [B 3]
(Zykluszeit und individuelle Download-Zeiten pro Cluster jeweils mit Standardabweichung;
im dezentralen Szenario endet ein Zyklus nach dem letzten Upload zum MCH)

Wie in den vorherigen Experimenten wurde nur die Mini-PCIe-Karte der Knoten als Primär-Interface genutzt, um ein gemischtes Hardware-Setup zu vermeiden und die Vergleichbarkeit der Übertragungszeiten in den Download- und Upload-Schritten zu gewährleisten. Entsprechend dem CHaChA-Konzept würden zwei separate Interfaces eine gleichzeitige Verbindung über Basis- und Cluster-Kanal ermöglichen. In der betrachteten Topologie konnte jedoch sogar das Multi-Channel-Szenario mit nur einem Interface durchgeführt werden, da die Intra- und Inter-Cluster-Kommunikation nacheinander erfolgten und sich alle CHs in direkter Nachbarschaft zum MCH befanden. In der Single-Channel-Konfiguration befanden sich alle Interfaces auf dem gleichen Kanal und die logische Trennung der Cluster wurde nur durch die Verwendung unterschiedlicher Mesh-IDs erreicht. Um den Upload der Cluster-Daten in Schritt 2 zu ermöglichen, änderten die CHs vorübergehend ihre Mesh-ID auf die des MCH-Clusters. In der Multi-Channel-Konfiguration nutzten sie dagegen sowohl die Mesh-ID als auch den Kanal des MCH-Clusters während des Upload-Schritts.

Abb. 5.22 vergleicht die Zykluszeiten des verteilten Szenarios in Single- und Multi-Channel-Konfiguration mit denen des zentralisierten Single-Channel-Szenarios im Worst- und Best Case. Alle Ergebnisse sind Mittelwerte aus 50 Messungen und mit ihrer Standardabweichung dargestellt. Für die dezentralen Szenarien sind die Download-Zeit pro Cluster (CM DL, grün) sowie die Summe der Download- und Upload-Zeiten (CM DL + MCH UL, orange) angegeben. Die Monitoring-Zykluszeit ergibt sich in diesen Fällen aus der längsten DL+UL-Dauer unter allen CHs. Da der MCH nur Schritt 1 durchführt und in Schritt 2 die Cluster-Daten der anderen CHs erhält, zeigt Abb. 5.22 nur seine CM-DL-Zeit. Die relativen Zeitunterschiede zwischen den Clustern sind einerseits auf die verschiedenen Cluster-Größen und somit übertragenen Datenmengen, andererseits auf Unfairness-Effekte der TCP-Überlastkontrolle in WLAN-Umgebungen zurückzuführen So können sich der Verlust oder die Verzögerung von TCP-Daten- bzw. -ACK-Paketen infolge von WLAN-Übertragungsfehlern und Queueing unterschiedlich auf das effektive Sendefenster (TCP Congestion Window (CWND)) einzelner Flows auswirken [247–251].

Verglichen mit dem zentralisierten Worst Case zeigte das verteilte Monitoring in der Single-Channel-Konfiguration eine Reduktion der Zykluszeit von ca. 34 %. Diese Verbesserung ist vergleichbar mit der des zentralisierten Best Case und kann ebenfalls durch eine vereinfachte Abschätzung bestätigt werden. Fasst man alle 1-Hop-Übertragungen der Schritte 1 und 2 zusammen, waren wie im zentralisierten Best Case mindestens 40 Sendevorgänge erforderlich. Darüber hinaus waren parallele Übertragungen wie auch im zentralisierten Single-Channel-Szenario nur begrenzt möglich. Sind alle Cluster in Schritt 1 auf denselben Kanal eingestellt, kommt es beim Download der CM-Daten in den Randbereichen benachbarter Cluster zu gegenseitigen Störungen. Dies gilt ebenso für den Upload der Cluster-Daten zum MCH in Schritt 2. Somit war, trotz einer inhärenten Erhöhung der Ausfallsicherheit durch die Verteilung der Monitoring-Anwendung, noch keine Zeitersparnis gegenüber dem zentralisierten Best Case ersichtlich.

Im Gegensatz dazu zeigte das Multi-Channel-Setup durch Verwendung überlappungsfreier Cluster-Kanäle bereits im 5x5-Knoten-Gitter eine deutliche Reduzierung der Download-Zeiten. Für den Upload-Schritt war hingegen keine deutliche Verbesserung zu erwarten, da hier wie zuvor alle CHs auf dem Kanal des MCH-Clusters miteinander konkurrierten. So konnten lediglich die CHs der kleineren Cluster (CH 7 und 19) aufgrund ihrer etwas kürzeren Download-Schritte für ein kurzes Zeitfenster den Upload zum MCH mit geringerer Interferenz als im Single-Channel-Szenario durchführen. Allein die Zeitreduktion in Schritt 1 führte jedoch bereits zu einer Verringerung der Gesamtzykluszeit um 48 % bzw. 21 % im Vergleich zum zentralisierten Worst- bzw. Best Case. Durch Ausnutzung des vollen Potenzials von Clustering und Kanalzuweisung, wie es durch CHaChA bereitgestellt wird, erfolgte die Einholung der CM-Daten in getrennten Kollisionsdomänen und profitierte somit von einer erhöhten Parallelität.

In praktischen Netzwerkinstallationen und unter Nutzung getrennter Mesh-Interfaces für Basis- und Cluster-Kanal sind im Langzeitbetrieb noch größere Verbesserungen zu erwarten. So könnten aufeinander folgende Monitoring-Zyklen verschränkt werden, sodass CHs bereits einen neuen Download der CM-Informationen in ihrem Cluster starten, während sie noch die gesammelten Daten des vorherigen Zyklus auf dem Basiskanal an den MCH übermitteln. Je nach Anwendungsanforderungen kann zudem der Upload zum MCH in größeren Zeitabständen durchgeführt werden. Ebenso könnten die Cluster-Informationen auf den CHs vorverarbeitet oder komprimiert werden, bevor die Synchronisierung erfolgt. Darüber hinaus stellt die Aufgabe der CH-Synchronisation ein Beispiel für sinnvolle Kombinationsmöglichkeiten der in dieser Arbeit entwickelten Optimierungslösungen CHaChA und MeNTor dar (siehe auch Kap. 6.2). So ist es denkbar, dass alle CHs auf dem Basiskanal einen BT-Schwarm für den effizienten kollaborativen Austausch von Cluster-Informationen bilden und somit nicht nur der MCH eine globale Netzwerksicht erhält.

5.6 Zwischenfazit

In diesem Kapitel wurde mit *CHaChA* eine dezentrale Clustering- und Kanalselektionslösung für WLAN-Mesh-Netzwerke entworfen, welche die in IEEE 802.11s eingeführten Mesh-Funktionen gezielt berücksichtigt [B 3]. Wie bereits die in Kap. 4 beschriebene Optimierungslösung *MeNTor* nutzt auch CHaChA ausschließlich Standardmechanismen der 802.11s-Spezifikation und erfordert keinerlei Modifikation der WLAN-Sicherungsschicht.

Hauptbeitrag ist ein als Phasenablauf realisierter Algorithmus zur Cluster-Bildung in WLAN-Mesh-Backbones, deren Knoten heutzutage üblicherweise zwei oder mehr physische WLAN-Interfaces besitzen. Davon verbleibt eines auf einem dedizierten Basiskanal, der stets bestmögliche Konnektivität sicherstellt und der CHaChA-Koordination dient. Ein Sekundär-Interface erlaubt wiederum die cluster-interne Kommunikation auf überlappungsfreien Kanälen, wodurch der Basiskanal entlastet wird und Störungen reduziert werden. Weiterführende Konzepte umfassen die Balancierung der Cluster sowie deren automatische Anpassung an Netzwerkveränderungen.

Als Cross-Layer-Ansatz integriert CHaChA inhärent verfügbare Link- und Pfadinformationen der 802.11s-Sicherungsschicht in die Anwendungsschicht, um Topologieinformationen abzuleiten und für die Cluster-Bildung und Kanalselektion zu nutzen. Im Gegenzug erfolgt aus der Anwendungsschicht heraus die Optimierung der Netzwerkstruktur durch Konfiguration der Sekundär-Interfaces. Eine wichtige Rolle spielt die gezielte Instrumentierung von Standardmechanismen, wie die des 802.11s-Routing-Protokolls HWMP. So ermöglicht der selektive Einsatz verschiedener Varianten des proaktiven HWMP-Modus einerseits ein, verglichen mit Mechanismen höherer Protokollschichten, kostengünstiges Auffinden von Knoten und andererseits deren gegenseitige Ausfallerkennung.

Anhand der Untersuchung des CHaChA-Prototyps im realen Versuchsaufbau *Mini-Mesh* für Gittertopologien mit bis zu 25 Knoten wurde die Praxistauglichkeit des Clustering-Ansatzes nachgewiesen. Mit einer Auftrittsrate der häufigsten Cluster-Konstellation von über 90 % belegen die Experimente auch im größten untersuchten Szenario die Reproduzierbarkeit der Ergebnisse. Die zunächst konservativ gewählte Parametrisierung des CHaChA-Algorithmus wurde experimentell auf die Testumgebung abgestimmt, sodass sich im 5x5-Knoten-Gitter eine Clustering-Dauer unterhalb einer Minute bei einer gesendeten Datenmenge von ca. 2,5 MB ergab. Die Realisierung des Algorithmus als Phasenablauf erleichtert dessen praktische Analyse und ermöglicht Zeitabschätzungen, die gut mit der gemessenen Clustering-Dauer korrespondieren.

In verschiedenen Gittergrößen wurde die Abhängigkeit der gesendeten Datenmenge von der Teilnehmerzahl untersucht. Es ergab sich ein erwarteter, nahezu linearer Zusammenhang für TCP, das im Rahmen von CHaChA vorwiegend der Kommunikation zwischen benachbarten Knoten dient. Im Fall von UDP zeigte sich eine stärkere nichtlineare Topologieabhängigkeit, da es für die netzwerkweiten Broadcast-Nachrichten genutzt wird. Über die initiale Cluster-Bildung hinaus wird der Langzeit-Kommunikations-Overhead jedoch insbesondere durch periodische Ankündigungen der CHs dominiert, deren Häufigkeit je nach Anwendungsfall gewählt werden kann.

Nach einer Erprobung der Mechanismen zur Cluster-Balancierung und -Anpassung wurden abschließend die für Anwendungen entstehenden Performance-Vorteile anhand eines Netzwerk-Monitoring-Szenarios demonstriert. Ein dezentraler Ansatz, der die durch CHaChA erzeugte Cluster-Konstellation und Kanalseparation zur Parallelisierung der Datenabfrage nutzt, wurde dazu mit einem zentralisierten Single-Channel-Ansatz ohne Cluster verglichen. Hierbei wurden Worst- und Best-Case-Platzierung eines dedizierten Monitoring-Knotens unterschieden. Neben einer erhöhten Robustheit, die sich aus der Verteilung der Anwendung ergibt, reduzierte der Einsatz parallel kommunizierender Cluster den Zeitbedarf für die Datenabfrage um 48 % gegenüber dem zentralisierten Worst Case und 21 % gegenüber dem zentralisierten Best Case.

6 Zusammenfassung

6.1 Cross-Layer-Optimierungsansätze für WLAN-Mesh-Netzwerke

Mesh-Netzwerke auf Basis der Wireless Local Area Network (WLAN)-Technologie bieten durch ihre Fähigkeit zur Spontanvernetzung und Multi-Hop-Datenweiterleitung positive Eigenschaften in puncto Flexibilität und Ausfallsicherheit. Sie sind daher ideal für die kostengünstige und robuste Breitbandabdeckung großer Areale für verschiedene Anwendungsfälle, wie z. B. für die Bereitstellung von Zugangsnetzen und Informationsdiensten in künftigen Smart-City-Szenarien. Der großflächige Einsatz von WLAN-Mesh-Netzwerken als regionale und städtische Backbone-Infrastruktur benötigt dabei eine herstellerübergreifende Interoperabilität, um die Austauschbarkeit von Geräten im Langzeitbetrieb zu gewährleisten und das Zusammenspiel von Installationen zu ermöglichen, die in der Regel schrittweise über größere Zeiträume aufgebaut und erweitert werden. Aus diesem Grund ist die Nutzung von Standards unerlässlich. In diesem Zusammenhang definiert die WLAN-Standarderweiterung IEEE 802.11s grundlegende Mesh-Funktionalität direkt auf der 802.11-Sicherungsschicht. Oberhalb der standardisierten Grundfunktionalität existieren jedoch zahlreiche Anforderungen und Problemstellungen im Hinblick auf die skalierbare Orchestrierung und Administration des Netzwerks sowie die effiziente Kommunikation von Anwendungen im Wireless Mesh Network (WMN), die durch 802.11s nicht behandelt werden.

Die vorliegende Forschungsarbeit widmet sich dem Entwurf von Cross-Layer-Optimierungslösungen für 802.11s-konforme WLAN-Mesh-Netzwerke. Dabei wird untersucht, inwieweit sich die durch 802.11s bereitgestellten Standardmechanismen und Verbindungsinformationen über praktisch vorhandene Schnittstellen gezielt schichtenübergreifend integrieren und instrumentieren lassen, um die effiziente Nutzung der verfügbaren Kommunikationsressourcen auch mit zunehmender Netzwerkkomplexität zu garantieren und gleichzeitig Interoperabilität zur Mesh-Spezifikation zu bewahren. Die Entwurfsmethodik wird exemplarisch an zwei Ansätzen demonstriert: *Mesh-Network-Aware BitTorrent (MeNTor)* und *Clustering Heuristic and Channel Assignment (CHaChA)*. Sie erlauben einerseits die Berücksichtigung der Netzwerkstruktur und der dynamischen Verbindungseigenschaften innerhalb von Anwendungen und Diensten, um deren Kommunikationseffizienz zu verbessern. Andererseits wird die Verteilung von Anwendungen auf unabhängige Netzwerkregionen durch Partitionierung und Kanalselektion unterstützt, um die Kommunikationsparallelität zu erhöhen. Die entwickelten Lösungen sind vielseitig einsetzbar und realisieren in Kombination die effiziente Datenausbringung und Datenabfrage in WLAN-Mesh-Netzwerken.

Die prototypische Entwicklung und Evaluation der Ansätze erfolgte in der realen Testumgebung *MiniMesh*. Der im Rahmen der Arbeit entworfene Testaufbau erlaubt durch den Einsatz von Dämpfungsgliedern im Antennenweg und einer reduzierten Sendeleistung die Abbildung eines großflächigen Mesh-Netzwerks im Labormaßstab [B 6]. Die auf einer Laborfläche von $1\,m^2$ realisierte Gitteranordnung entspricht einer theoretischen Fläche von mehr als $300.000\,m^2$ ohne Reichweitenbeschränkung, vergleichbar mit einem Stadtviertel, Park, oder Campusgelände. Das Miniaturisierungskonzept wird mithilfe eines Dämpfungsmodells mathematisch beschrieben und praktisch validiert. Vergleichsmessungen des Datendurchsatzes auf der Transportschicht und der Routing-Metrik auf der Sicherungsschicht zeigen mit einer mittleren Abweichung von weniger als 7 % eine hohe Übereinstimmung der Ergebnisse im skalierten und unskalierten Aufbau und belegen somit die Eignung der Testumgebung.

Zunächst wurde der Cross-Layer-Optimierungsansatz *Mesh-Network-Aware BitTorrent (MeNTor)* vorgestellt. Dieser integriert die Verbindungsinformationen der WLAN-Sicherungsschicht in die Anwendungsschicht des Peer-to-Peer (P2P)-Protokolls BitTorrent (BT), um dieses mit Wissen über die Mesh-Topologie anzureichern und einen effizienten kollaborativen Datenaustausch in WLAN-Mesh-Netzwerken zu ermöglichen [B 4, 17]. Folglich erhält BT Informationen über die Kommunikationskosten zu logischen Teilnehmern, auf deren Basis eine Peer-Auswahl-Strategie umgesetzt wird,

die physisch nahe Peers als Upload-Ziele bevorzugt. Weitere Optimierungen umfassen die Priorisierung von Nachbarknoten, die Beschränkung paralleler Uploads zu unterschiedlichen Zielknoten sowie die vollständige Aufhebung des Belohnungsprinzips und der anteilig zufallsbehafteten Peer-Auswahl von Standard-BT. MeNTor ist zugeschnitten auf Standard-802.11s, benötigt keine Modifikation der WLAN-Sicherungsschicht und erzeugt aufgrund der Integration inhärent verfügbarer Verbindungsinformationen keinen zusätzlichen Datenverkehr. Die praktischen Ergebnisse im Testaufbau mit 25 Knoten belegen die Wirksamkeit der Methode und zeigen bereits für diese Netzwerkgröße, dass sich der mittlere Zeitbedarf für die Datenausbringung um mehr als 40 % reduzieren lässt.

Im Anschluss wurde der dezentrale Ansatz *Clustering Heuristic and Channel Assignment (CHaChA)* entwickelt, der unter Berücksichtigung der Mesh-Topologie eine logische Partitionierung des Netzwerks in getrennte Regionen (Cluster) vornimmt [B 3]. Die Lösung zielt darauf ab, das vorhandene Datendurchsatzpotential auch in komplexen 802.11s-Mesh-Netzwerken effizient zu nutzen. Jedes Cluster entsteht um einen automatisch gewählten *Cluster Head (CH)*, der für die Verwaltung der darin befindlichen Knoten zuständig ist. Zusätzlich wird das Clustering mit einer Kanalselektion kombiniert, sodass Teilnehmer benachbarter Cluster gleichzeitig auf überlappungsfreien Frequenzen kommunizieren können. Die cluster-übergreifende Konnektivität bleibt stets durch ein dediziertes Interface auf einem vordefinierten Basiskanal gewährleistet, während die cluster-interne Kommunikation über ein Sekundär-Interface erfolgt. Weiterführend wurden Mechanismen für die automatische Cluster-Anpassung auf Topologieänderungen vorgestellt. Darunter fallen der nachträgliche bzw. erneute Cluster-Beitritt, die gegenseitige Isolations- und Ausfallerkennung von Knoten sowie das Roaming zur Cluster-Balancierung. Auch dieser prototypisch auf der Anwendungsschicht realisierte Ansatz erfordert keine Modifikation der WLAN-Sicherungsschicht. Das integrierte 802.11s-Netzwerkwissen wird teils direkt, teils zur Bildung zusätzlicher Metriken genutzt, auf deren Basis Entscheidungen für die Cluster-Bildung getroffen werden. Die Orchestrierung des Netzwerks wird durch entsprechende Konfiguration der Sekundär-Interfaces der Mesh-Knoten vorgenommen. Dabei erfolgt eine gezielte Nutzung der 802.11s-Mechanismen, z. B. durch geeignete Kombination der reaktiven und proaktiven Modi des Routing-Protokolls Hybrid Wireless Mesh Protocol (HWMP) zur Netzwerkerkundung und gegenseitigen Ausfallerkennung von Knoten.

Die Praxistauglichkeit des Clustering-Ansatzes wurde in Gittertopologien mit bis zu 25 Knoten nachgewiesen. Zunächst wurde experimentell eine für die Leistungsklasse der Geräte der Testumgebung geeignete Konfiguration des Algorithmus ermittelt. Die damit durchgeführten Messungen im 5x5-Knoten-Gitter belegen eine hohe Reproduzierbarkeit der Ergebnisse bei einer Clustering-Dauer von 55 s und einer gesendeten Datenmenge von ca. 2,5 MB. Nachfolgend wurde die Abhängigkeit der gesendeten Datenmenge von der Netzwerkgröße untersucht, wobei sich erwartungsgemäß eine stärkere Topologieabhängigkeit der als Broadcasts gesendeten CHaChA-Nachrichtentypen zeigte. Dies ist im Konzept bereits berücksichtigt, sodass Broadcast-Nachrichten im Langzeitbetrieb nur von Cluster Heads (CHs) ausgehen und deren Sendeperiode anwendungsabhängig gewählt werden kann.

Im Zuge der praktischen Auswertung wurden die Performance-Vorteile demonstriert, die für eine exemplarische Monitoring-Anwendung entstehen, bei der Statusinformationen aller Mesh-Knoten durch den Netzbetreiber eingeholt werden. Dazu wurde ein dezentraler Multi-Channel-Ansatz mit einem zentralisierten Single-Channel-Ansatz (Worst und Best Case-Platzierung eines dedizierten Monitoring-Knotens) verglichen. Durch die logische Verteilung der Datenabfrage auf mehrere CHs wurde im Vergleich zum zentralisierten Ansatz einerseits eine erhöhte Ausfallsicherheit der Anwendung erreicht, andererseits das Kommunikationsaufkommen durch Einsparung von Weiterleitungsschritten reduziert, wodurch der Monitoring-Zeitbedarf um 48 % gegenüber dem zentralisierten Worst Case bzw. 21 % gegenüber dem zentralisierten Best Case sank. Darüber hinaus wurden durch die Kanalseparation der Cluster Interferenzen zwischen diesen reduziert und die Kommunikationsparallelität erhöht, wovon neben der Monitoring-Anwendung potentiell auch andere Dienste im Mesh-Netzwerk profitieren.

6.2 Ausblick

MeNTor und CHaChA stellen orthogonale Optimierungslösungen dar, die unabhängig voneinander eingesetzt werden können. Darüber hinaus existieren Performance-Vorteile, die durch eine Kombination beider Verfahren entstehen. So könnte MeNTor in einem durch CHaChA orchestrierten Mesh-Netzwerk die effiziente cluster-übergreifende Kommunikation auf dem Basiskanal unterstützen. Dazu gehört beispielsweise die kollaborative Synchronisation von aggregierten Statusdaten zwischen den CHs, welche dazu einen eigenen BT-Schwarm bilden. Auch im Gegenzug könnte die kollaborative Ausbringung von Software-Updates in einer Staffelung zunächst auf dem Basiskanal an die CHs erfolgen und innerhalb der Cluster in separaten BT-Schwärmen fortgesetzt werden. Künftige Arbeiten könnten sich mit der Untersuchung dieser und möglicher weiterer Synergien beschäftigen. Ebenso könnte die Parametrisierung des CHaChA-Algorithmus für Geräteplattformen verschiedenster Leistungsklassen analysiert werden, um noch genauere Aussagen über die technischen Grenzen und die minimal erreichbare Clustering-Dauer treffen zu können. Auch die Integration einer kompakteren Nachrichtenkodierung sowie der Ausbau der Konzepte für ein mögliches Re-Clustering wären für den praktischen Einsatz interessant.

Ausgehend von den Ergebnissen dieser Arbeit, die für eine Netzwerkgröße von 25 Knoten ermittelt wurden, ist ein zunehmender Performance-Gewinn für Dimensionen städtischer WMNs mit steigender Ausdehnung und Knotenzahl zu erwarten. Zukünftige Arbeiten sollten sich daher der Untersuchung von MeNTor und CHaChA in noch größeren und dynamischeren Szenarien widmen. Dabei ist die reale Testumgebung Mini-Mesh momentan auf stationäre Topologien mit einer dominierenden Sichtverbindung (Line-of-Sight (LoS)) zwischen benachbarten Knoten beschränkt. Eine Miniaturisierung von Szenarien mit starker Non-Line-of-Sight (NLoS)-Charakteristik stellt eine große Herausforderung dar, da die Eigenschaften und Dimensionen realer Hindernisse nur schwer im Labor abgebildet werden können. Als Grundlage für weiterführende Untersuchungen wird derzeit am Institut für angewandte Mikroelektronik und Datentechnik in Zusammenarbeit mit Kollegen die Simulationsumgebung *ViPMesh* (Virtual Prototype Mesh) entwickelt [B 10, 11] [C 19, 20, 25]. Das in [B 11] vorgestellte Konzept basiert auf einem bestehenden Framework des Linux-Kernels und erlaubt die Untersuchung realer Anwendungen oberhalb des Linux-Netzwerkprotokollstapels und dessen 802.11s-Implementierung. Dazu wird das Kernelmodul `mac80211_hwsim`, das der Abstraktion von WLAN-Hardware dient und für funktionale Tests des Protokollstapels ausgelegt ist, um die Simulation eines Umgebungsmodells samt Medienzugriff, Kanaleffekten und Mobilität ergänzt. Die Ausführung der unveränderten Target-Software (Linux-Protokollstapel und Anwendungen) wird dabei geeignet mit der Simulation gekoppelt.

Eine weitere während der Bearbeitung der Dissertation identifizierte Forschungsfrage ist die Cross-Layer-Kooperation zwischen der 802.11-Sicherungsschicht und TCP in WLAN-Mesh-Netzwerken [B 15] [C 12–14, 23]. In ersten Untersuchungen wurden Probleme im Zusammenspiel der WLAN- und TCP-Fehlerbehandlungsmechanismen in einem Multi-Hop-Testaufbau nachgewiesen [B 15]. Ziel künftiger Strategien sollte es sein, die optimale Anzahl zulässiger WLAN-Neuübertragungen dynamisch zu bestimmen und einer redundanten Fehlerbehandlung auf den verschiedenen Schichten vorzubeugen. Im Ergebnis profitiert eine Vielzahl TCP-basierter Anwendungen von einer solchen Optimierung, wie z. B. das in dieser Arbeit genutzte P2P-Protokoll BT.

Zuletzt ergeben sich aus der durch CHaChA erzeugten Konfiguration mehrerer Mesh-Interfaces auf überlappungsfreien Kanälen Anknüpfungspunkte in verwandte Forschungsgebiete, die sich mit der Entwicklung von Multi-Interface-Nutzungsstrategien beschäftigen [252, 253]. Ein einfacher, aber unflexibler Ansatz ist die statische Bindung der Inter- bzw. Intra-Cluster-Kommunikation an das jeweilige Interface. Wünschenswert wäre jedoch eine für Anwendungen transparente, kombinierte Interface-Nutzung nach dem Prinzip der *Link Aggregation* (auch *Interface Bonding* bzw. *Trunking*) [254] oder basierend auf Transportprotokollen wie *Multipath TCP* [255, 256]. Über mehrere

Interfaces erreichbare Endpunkte könnten somit auf unabhängigen Pfaden parallel mit Daten versorgt werden, sodass sich der Durchsatz und die Ausfallsicherheit der Verbindung erhöhen. Im Gegensatz zu kabelgebundenen Szenarien mit vorwiegend stabilen Kanalbedingungen ergeben sich in WLAN-Mesh-Netzwerken jedoch besondere Herausforderungen hinsichtlich der dynamischen Lastbalancierung zwischen den Interfaces [257, 258].

Literaturverzeichnis

[1] Luigi Atzori, Antonio Iera und Giacomo Morabito. „The Internet of Things: A Survey“. In: *Computer Networks* 54.15 (2010), S. 2787–2805.

[2] S. S. Sabry, N. A. Qarabash und H. S. Obaid. „The Road to the Internet of Things: a Survey“. In: *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*. 2019, S. 290–296.

[3] Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Yasir Mehmood, Abdullah Gani, Salimah Mokhtar und Sghaier Guizani. „Enabling Communication Technologies for Smart Cities“. In: *IEEE Communications Magazine* 55.1 (Jan. 2017), S. 112–120. DOI: 10.1109/MCOM.2017.1600232CM.

[4] Pasquale Pace, Valeria Loscri, Zhengguo Sheng, Giuseppe Ruggeri und Athanasios V. Vasilakos. „Smart Wireless Access Networks and Systems for Smart Cities“. In: *Ad Hoc Networks* 43 (2016), S. 1–2.

[5] „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 802.11-2012 (Revision of IEEE Std 802.11-2007)* (2012).

[6] I. F. Akyildiz und Xudong Wang. „A survey on wireless mesh networks“. In: *IEEE Communications Magazine* 43.9 (2005), S23–S30.

[7] Marius Portmann und Asad Amir Pirzada. „Wireless Mesh Networks for Public Safety and Crisis Management Applications“. In: *IEEE Internet Computing* 12.1 (2008), S. 18–25.

[8] Antonio Cilfone, Luca Davoli, Laura Belli und Gianluigi Ferrari. „Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies“. In: *Future Internet* 11.4 (2019), S. 1–35. DOI: 10.3390/fi11040099.

[9] Peter Lars Dordal. *An Introduction to Computer Networks: Other LANs: WiFi: Mesh Networks*. Loyola University of Chicago, 2020. URL: http://intronetworks.cs.luc.edu/.

[10] ASSIA Inc. *Wi-Fi Mesh Extenders: Truth or Hype?* 2020. URL: https://www.assia-inc.com/wi-fi-mesh-extenders-truth-hype/ (besucht am 15. 08. 2020).

[11] Jim Geier. *Wi-Fi Mesh: What to know about enterprise mesh networks*. 2019. URL: https://www.networkworld.com/article/3331285/wi-fi-mesh-what-to-know-about-enterprise-mesh-networks.html (besucht am 15. 08. 2020).

[12] Guido Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann und Bernhard Walke. „IEEE 802.11s: The WLAN Mesh Standard“. In: *IEEE Wireless Communications* 17.1 (2010), S. 104–111.

[13] *Freifunk.net*. 2020. URL: https://freifunk.net/ (besucht am 15. 08. 2020).

[14] *Opennet Initiative e.V.* 2020. URL: https://wiki.opennet-initiative.de/wiki/Hauptseite (besucht am 15. 08. 2020).

[15] Fabrizio Granelli, DZMITRY Kliazovich und Nelson LS da Fonseca. „Performance limitations of IEEE 802.11 networks and potential enhancements“. In: *Wireless LANs and Bluetooth* (2005).

[16] Marcus Burton und GT Hill. *802.11 Arbitration*. Techn. Ber. Cisco Certified Wireless Network Professional, 2009. URL: https://www.cwnp.com/uploads/802-11_arbitration.pdf (besucht am 15. 08. 2020).

[17] K. Gierłowski. „Cross-layer mDNS/ARP Integration for IEEE 802.11s Wireless Mesh Network“. In: *2016 9th IFIP Wireless and Mobile Networking Conference (WMNC)*. 2016, S. 33–40.

[18] Sajid M. Sheikh, Riaan Wolhuter und Herman A. Engelbrecht. „A survey of cross-layer protocols for IEEE 802.11 wireless multi-hop mesh networks“. In: *International Journal of Communication Systems* 30.6 (2017), e3129. DOI: 10.1002/dac.3129.

[19] Djohara Benyamina, Abdelhakim Senhaji Hafid und Michel Gendreau. „Wireless Mesh Networks Design — A Survey“. In: *IEEE Communications Surveys & Tutorials* 14 (Jan. 2012), S. 1–12. DOI: 10.1109/SURV.2011.042711.00007.

[20] Md Ngadi, Saqib Ali, Hanan Abdullah und Rashid Hafeez Khokhar. „A taxonomy of cross layer routing metrics for wireless mesh networks“. In: *EURASIP Journal on Wireless Communications and Networking* 2012 (Dez. 2012). DOI: 10.1186/1687-1499-2012-177.

[21] Vinicius C. M. Borges, Marilia Curado und Edmundo Monteiro. „Cross-Layer Routing Metrics for Mesh Networks: Current Status and Research Directions“. In: *Computer Communications* 34.6 (Mai 2011), S. 681–703. DOI: 10.1016/j.comcom.2010.12.001.

[22] Kefeng Tan, Daniel Wu, An Jack Chan und Prasant Mohapatra. „Comparing simulation tools and experimental testbeds for wireless mesh networks“. In: *Pervasive and Mobile Computing* 7.4 (2011), S. 434–448.

[23] Piotr Owczarek und Piotr Zwierzykowski. „Review of simulators for wireless mesh networks“. In: *Journal of Telecommunications and Information Technology* 3 (2014), S. 82.

[24] *ns-3 Network Simulator: Mesh Design Documentation*. 2020. URL: https://www.nsnam.org/docs/models/html/mesh-design.html (besucht am 15. 08. 2020).

[25] Alfonso Quintana und Vincenzo Inzillo. „INETMANET Framework“. In: *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*. Mai 2019, S. 107–138. DOI: 10.1007/978-3-030-12842-5_3.

[26] Jairo L. Duarte, Diego Passos, Rafael L. Valle, Etienne Oliveira, Debora Muchaluat-Saade und Celio V. Albuquerque. „Management Issues on Wireless Mesh Networks“. In: *2007 Latin American Network Operations and Management Symposium*. IEEE, 2007, S. 8–19. ISBN: 1-4244-1182-3.

[27] Francoise Sailhan, Liam Fallon, Karl Quinn, Paddy Farrell, Sandra Collins, Daryl Parker, Samir Ghamri-Doudane und Yangcheng Huang. „Wireless mesh network monitoring: Design, implementation and experiments“. In: *Globecom Workshops, 2007 IEEE*. 2007.

[28] Nadir Shah, S.A. Abid, Depei Qian und Waqar Mehmood. „A Survey of P2P Content Sharing in MANETs“. In: *Computers & Electrical Engineering* 57.C (Jan. 2017), S. 55–68. DOI: 10.1016/j.compeleceng.2016.12.013.

[29] Nasreen Anjum, Dmytro Karamshuk, Mohammad Shikh-Bahaei und Nishanth Sastry. „Survey on Peer-Assisted Content Delivery Networks“. In: *Computer Networks* 116.C (Apr. 2017), S. 79–95. DOI: 10.1016/j.comnet.2017.02.008.

[30] Apostolos Malatras. „State-of-the-art survey on P2P overlay networks in pervasive computing environments“. In: *Journal of Network and Computer Applications* 55 (2015), S. 1–23. DOI: 10.1016/j.jnca.2015.04.014.

[31] BitTorrent. *Protocol Specification*. 2020. URL: http://www.bittorrent.org/beps/bep_0000.html (besucht am 15. 08. 2020).

[32] Sandvine Intelligent Broadband Networks. *Global Internet Phenomena Report 2019 Q3*. Sep. 2019.

[33] Kyriakos Manousakis, Sharanya Eswaran, David Shur, Gaurav Naik, Pavan Kantharaju, William Regli und Brian Adamson. „Torrent-Based Dissemination in Infrastructure-Less Wireless Networks“. In: *Journal of Cyber Security and Mobility* 4.1 (2015), S. 1–22.

[34] Burkhard Englert und Souroush Pourezza. „On Using Bittorrent for File Sharing in Mobile Ad-Hoc Networks“. In: *Internet and Distributed Computing Systems*. Springer Berlin Heidelberg, 2013, S. 78–91. ISBN: 978-3-642-41428-2.

[35] Mohamed Karim Sbai, Chadi Barakat, Jaeyoung Choi, Anwar Al Hamra und Thierry Turletti. „Adapting BitTorrent to Wireless Ad Hoc Networks“. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, S. 189–203.

[36] Anitha Manikandan und Yogesh Palanichamy. „Optimized Group Channel Assignment Using Computational Geometry over Wireless Mesh Networks“. In: *Mobile Information Systems* 2015 (2015), S. 1–18.

[37] Vinay Kapse und Urmila Shrawankar. „Interference-aware channel assignment for maximizing throughput in WMN“. In: *Int. Journal on AdHoc Networking Systems* 1.1 (2011).

[38] S. Ghannay und S. M. Gammar. „Joint routing and channel assignment protocol for multi-radio multi-channel IEEE 802.11s mesh networks“. In: *4th Joint IFIP Wireless and Mobile Networking Conference*. 2011, S. 1–8.

[39] A. B. M. Alim Al Islam, M. J. Islam, N. Nurain und V. Raghunathan. „Channel Assignment Techniques for Multi-Radio Wireless Mesh Networks: A Survey“. In: *IEEE Communications Surveys & Tutorials* 18.2 (2016), S. 988–1017.

[40] Ashaf Alzubir, Kamalrulnizam Abu, Adil Yousif und Albarraa Abuobieda. „State of the Art, Channel Assignment Multi-Radio Multi-Channel in Wireless Mesh Network“. In: *Int. Journal of Computer Applications* 37.4 (2012), S. 14–20.

[41] V. S. Kapse und U. N. Shrawanakar. „Survey of channel assignement schemes in wireless mesh network“. In: *3rd International Conference on Electronics Computer Technology*. Bd. 3. 2011, S. 103–107.

[42] Andrew S. Tanenbaum und David J. Wetherall. *Computer Networks*. 5th. USA: Prentice Hall Press, 2010. ISBN: 0132126958.

[43] James F. Kurose und Keith W. Ross. *Computer Networking: A Top-Down Approach*. 6th. Pearson, 2012. ISBN: 0132856204.

[44] Ian F. Akyildiz und Xudong Wang. *Wireless Mesh Networks*. Wiley Telecom, 2009. ISBN: 9780470032565.

[45] Jose A. Gutierrez, Edgar H. Callaway und Raymond Barrett. *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*. USA: IEEE Standards Office, 2003. ISBN: 0738135577.

[46] S. M. Sheikh, R. Wolhuter und G. J. van Rooyen. „A comparative analysis of MANET routing protocols for low cost rural telemetry Wireless Mesh Networks“. In: *2015 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*. 2015, S. 32–37.

[47] K S Nagegowda, H R Ranganath, C Puttamadappa und T G Basavaraju. „Performance Evaluation of Scalable Routing Protocols using Routing Metrics for Wireless Mesh Networks under different Network Scenarios“. In: *IRACST - Int. Journal of Computer Networks and Wireless Communications* 4.6 (2014).

[48] T. Clausen und P. Jacquet. *Optimized Link State Routing Protocol (OLSR)*. RFC 3626 (Experimental). Internet Engineering Task Force, Okt. 2003. URL: http://www.ietf.org/rfc/rfc3626.txt.

[49] Thomas H. Clausen, Christopher Dearlove, Philippe Jacquet und Ulrich Herberg. *The Optimized Link State Routing Protocol Version 2*. RFC 7181. Apr. 2014. DOI: 10.17487/RFC7181.

[50] D. Murray, M. Dixon und T. Koziniec. „An experimental comparison of routing protocols in multi hop ad hoc networks“. In: *Australasian Telecommunication Networks and Applications Conference*. 2010, S. 159–164.

[51] *olsr.org Wiki: OLSR Daemon*. 2020. URL: http://www.olsr.org/mediawiki/index.php/Olsr_Daemon (besucht am 15. 08. 2020).

[52] *olsr.org Wiki: OLSR.org Network Framework*. 2020. URL: http://www.olsr.org/mediawiki/index.php/OLSR.org_Network_Framework (besucht am 15. 08. 2020).

[53] Axel Neumann, Corinna Aichele, Marek Lindner und Simon Wunderlich. *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)* Internet-Draft draft-wunderlich-openmesh-manet-routing-00. Work in Progress. Internet Engineering Task Force, Apr. 2008. 24 S. URL: https://datatracker.ietf.org/doc/html/draft-wunderlich-openmesh-manet-routing-00.

[54] David Johnson, Ntsibane Ntlatlapa und Corinna Aichele. „A simple pragmatic approach to mesh routing using batman“. In: *In 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, Pretoria, South Africa*. 2008.

[55] *open-mesh.org Wiki: BATMAN V Protocol Specification*. 2020. URL: https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_V (besucht am 15. 08. 2020).

[56] *open-mesh.org Wiki: batmand*. 2020. URL: https://www.open-mesh.org/projects/batmand/wiki/ (besucht am 15. 08. 2020).

[57] *open-mesh.org Wiki: batman-adv*. 2020. URL: https://www.open-mesh.org/projects/batman-adv/wiki/ (besucht am 15. 08. 2020).

[58] Rosario G. Garroppo, Stefano Giordano und Luca Tavanti. „Experimental evaluation of two open source solutions for wireless mesh routing at layer two“. In: *IEEE 5th International Symposium on Wireless Pervasive Computing*. ISWPC'10. Mondena, Italy: IEEE, 2010. ISBN: 978-1-4244-6855-3.

[59] D. Johnson, Y. Hu und D. Maltz. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. RFC 4728 (Experimental). Internet Engineering Task Force, Feb. 2007. URL: http://www.ietf.org/rfc/rfc4728.txt.

[60] Axel Neumann, Ester López und Leandro Navarro. „An evaluation of BMX6 for community wireless networks“. In: Okt. 2012, S. 651–658. DOI: 10.1109/WiMOB.2012.6379145.

[61] *BMX6 Github Repository*. 2020. URL: https://github.com/bmx-routing/bmx6 (besucht am 15. 08. 2020).

[62] Juliusz Chroboczek. *The Babel Routing Protocol*. RFC 6126. Apr. 2011. DOI: 10.17487/RFC6126.

[63] *Babel – a loop-avoiding distance-vector routing protocol*. 2020. URL: https://www.irif.fr/~jch/software/babel/ (besucht am 15. 08. 2020).

[64] Charles E. Perkins und Pravin Bhagwat. „Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers“. In: *SIGCOMM Comput. Commun. Rev.* 24.4 (Okt. 1994), S. 234–244. DOI: 10.1145/190809.190336.

[65] C. Perkins, E. Belding-Royer und S. Das. *Ad hoc On-Demand Distance Vector Routing*. RFC 3561. Internet Engineering Task Force, Juli 2003. URL: http://www.ietf.org/rfc/rfc3561.txt.

[66] *MIT: The Grid Ad Hoc Networking Project*. 2020. URL: https://pdos.csail.mit.edu/archive/grid/index.html (besucht am 15. 08. 2020).

[67] *AODV-UU: AODV Linux Implementation by Uppsala University*. 2020. URL: https://sourceforge.net/projects/aodvuu/ (besucht am 15. 08. 2020).

[68] *Freifunk-Gluon Issue 1584: IBSS mesh and client mode doesn't work in parallel with ath10k-ct*. 2020. URL: https://github.com/freifunk-gluon/gluon/issues/1584 (besucht am 15. 08. 2020).

[69] *Freifunk-Gluon Issue 1747: Drop support for IBSS Mesh*. 2020. URL: https://github.com/freifunk-gluon/gluon/issues/1747 (besucht am 15. 08. 2020).

[70] *Freifunk-Berlin Firmware Pull Request 180: Add 802.11s support to the wizard*. 2020. URL: https://github.com/freifunk-berlin/firmware-packages/pull/180 (besucht am 15. 08. 2020).

[71] *Freifunk Frankfurt a.M.: Neue Stable Firmware 11s Migration*. 2020. URL: https://ffm.freifunk.net/2019/09/05/neue-stable-firmware-11s-migration/ (besucht am 15. 08. 2020).

[72] *Freifunk Wiesbaden: Umstellung von IBSS auf 802.11s*. 2020. URL: https://www.wiesbaden.freifunk.net/2017/05/23/umstellung-ibss-auf-80211s.html (besucht am 15. 08. 2020).

[73] *Freifunk Forum: IEEE 802.11s und das Hybrid Wireless Mesh Protocol*. 2020. URL: https://forum.freifunk.net/t/ieee-802-11s-und-das-hybrid-wireless-mesh-protocol/6907 (besucht am 15. 08. 2020).

[74] *AVM Fritz! Mesh*. 2020. URL: https://avm.de/mesh/ (besucht am 15. 08. 2020).

[75] *Ubiquiti AmpliFi*. 2020. URL: https://amplifi.com/ (besucht am 15. 08. 2020).

[76] *Amazon eero*. 2020. URL: https://eero.com/ (besucht am 15. 08. 2020).

[77] *TP-Link Deco*. 2020. URL: https://www.tp-link.com/de/home-networking/deco/ (besucht am 15. 08. 2020).

[78] *Netgear Orbi*. 2020. URL: https://www.netgear.com/orbi/ (besucht am 15. 08. 2020).

[79] *Asus Lyra*. 2020. URL: https://www.asus.com/Networking/Lyra/ (besucht am 15. 08. 2020).

[80] *Linksys Velop*. 2020. URL: https://www.linksys.com/us/velop/ (besucht am 15. 08. 2020).

[81] *Samsung SmartThings Wifi*. 2020. URL: https://www.samsung.com/us/hubs/ (besucht am 15. 08. 2020).

[82] *Google Store: Nest Wifi*. 2020. URL: https://store.google.com/product/nest_wifi (besucht am 15. 08. 2020).

[83] *Google Blog: Making a 'mesh' of your Wi-Fi*. 2020. URL: https://blog.google/products/google-wifi/making-mesh-your-wi-fi/ (besucht am 15. 08. 2020).

[84] *Tenda: Whole Home Mesh WiFi Coverage*. 2020. URL: https://www.tendacn.com/us/product/nova%5C%20mw6.html (besucht am 15. 08. 2020).

[85] *Stacey on IoT Blog: Wi-Fi EasyMesh: The good, the bad and the ugly*. 2020. URL: https://staceyoniot.com/wi-fi-easymesh-the-good-the-bad-and-the-ugly/ (besucht am 15. 08. 2020).

[86] *MeshPoint.One: Technical Specifications*. 2020. URL: https://www.meshpointone.com/technical-specifications/ (besucht am 15. 08. 2020).

[87] *FreeMesh: Fast, stable, secure WiFi*. 2020. URL: https://freemeshwireless.com/ (besucht am 15. 08. 2020).

[88] *Texas Instruments: WiLink technology solutions*. 2020. URL: https://www.ti.com/wireless-connectivity/simplelink-solutions/wi-fi/overview/wilink-combo-solutions.html (besucht am 15. 08. 2020).

[89] *Cisco/Meraki: Technologies: Mesh Routing*. 2020. URL: https://meraki.cisco.com/technologies/mesh-routing (besucht am 15. 08. 2020).

[90] *ABB: Wireless Networks*. 2020. URL: https://www.hitachiabb-powergrids.com/offering/product-and-system/communication-networks/wireless-overview (besucht am 15. 08. 2020).

[91] *Aruba Networks: ArubaOS User Guide – Secure Enterprise Mesh*. 2020. URL: https://www.arubanetworks.com/techdocs/ArubaOS_60/UserGuide/Mesh.php (besucht am 15. 08. 2020).

[92] *Firetide: Products*. 2020. URL: https://www.firetide.com/solutions/products/ (besucht am 15. 08. 2020).

[93] *Ruckus Wireless: Mesh Networking and SmartMesh*. 2020. URL: https://www.ruckuswireless.com/rucktionary/mesh-networking-and-smartmesh (besucht am 15. 08. 2020).

[94] *Juniper Networks TechLibrary: Understanding Wireless Mesh*. 2020. URL: https://www.juniper.net/documentation/en_US/junos-space-apps/network-director3.2/topics/concept/wireless-mesh-aps.html (besucht am 15. 08. 2020).

[95] *Open Mesh Is Now Datto Networking*. 2020. URL: https://www.openmesh.com (besucht am 15. 08. 2020).

[96] *MikroTik Wiki: HWMP+*. 2020. URL: https://wiki.mikrotik.com/wiki/Manual:Interface/HWMPplus (besucht am 15. 08. 2020).

[97] *Wi-Fi Alliance: Wi-Fi EasyMesh*. 2020. URL: https://www.wi-fi.org/discover-wi-fi/wi-fi-easymesh (besucht am 15. 08. 2020).

[98] *heise.de Meldung: EasyMesh: Wi-Fi Alliance zertifiziert jetzt Mesh-WLAN-Systeme*. 2020. URL: https://www.heise.de/newsticker/meldung/EasyMesh-Wi-Fi-Alliance-zertifiziert-jetzt-Mesh-WLAN-Systeme-4047177.html (besucht am 15. 08. 2020).

[99] Eldad Perahia und Robert Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. 2nd. USA: Cambridge University Press, 2013. ISBN: 1107016762.

[100] Matthew Gast. *802.11n: A Survival Guide*. O'Reilly Media, Inc., 2012. ISBN: 1449312047.

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

[102] Jim Geier. *Designing and deploying 802.11n wireless networks*. 1st. Pearson Education, 2010. ISBN: 1587058898.

[103] Pejman Roshan und Jonathan Leary. *802.11 Wireless LAN Fundamentals*. 2010. ISBN: 1587142244.

[104] Official IEEE 802.11 Working Group Project Timelines. 2020. URL: https://www.ieee802.org/11/Reports/802.11_Timelines.htm (besucht am 15. 08. 2020).

[105] Linux Wireless Wiki: List of Drivers. 2020. URL: https://wireless.wiki.kernel.org/en/users/drivers (besucht am 15. 08. 2020).

[106] S. Idwan, E. Fayyoumi, H. A. Muhareb, I. Matar und O. A. Rawashdeh. „Achieving extended displays prototype via Wi-Fi direct technology". In: *2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*. 2014, S. 109–114.

[107] Martijn Saelens, Jeroen Hoebeke, Adnan Shahid und Eli De Poorter. „Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs: an overview and future challenges". In: *EURASIP Journal on Wireless Communications and Networking* (2019), S. 1–32.

[108] Minstrel Rate Control. 2020. URL: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel (besucht am 15. 08. 2020).

[109] IEEE 802.11n MCS Rates. 2020. URL: https://wireless.wiki.kernel.org/en/developers/documentation/ieee80211/802.11n (besucht am 15. 08. 2020).

[110] Haider M Alsabbagh, CHEN Jianping und XU Youyun. „Influence of the limited retransmission on the performance of WLANs using error-prone channel“. In: *Int. Journal of Communications, Network and System Sciences* 1.01 (2008).

[111] Ying-Dar Lin, Jui-Hung Yeh, Tsung-Hsien Yang, Chia-Yu Ku, Shiao-Li Tsao und Yuan-Cheng Lai. „Efficient dynamic frame aggregation in IEEE 802.11s mesh networks“. In: *International Journal of Communication Systems* 22.10 (2009), S. 1319–1338. ISSN: 1099-1131.

[112] Vaduvur Bharghavan, Alan Demers, Scott Shenker und Lixia Zhang. „MACAW: A Media Access Protocol for Wireless LANs“. In: *SIGCOMM Comput. Commun. Rev.* 24.4 (Okt. 1994), S. 212–225. DOI: 10.1145/190809.190334.

[113] R. C. Carrano, L. C. S. Magalhães, D. C. M. Saade und C. V. N. Albuquerque. „IEEE 802.11s Multihop MAC: A Tutorial“. In: *IEEE Communications Surveys & Tutorials* 13.1 (2011), S. 52–67.

[114] Jerome Henry. *802.11s Mesh Networking*. Techn. Ber. Cisco Certified Wireless Network Professional, 2011. URL: https://www.cwnp.com/uploads/802-11s_mesh_networking_v1-0.pdf (besucht am 15. 08. 2020).

[115] D. Harkins. „Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks“. In: *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*. 2008, S. 839–844.

[116] SMS Bari, F Anwar und MH Masud. „Performance study of Hybrid Wireless Mesh Protocol (HWMP) for IEEE 802.11s WLAN mesh networks“. In: *2012 IEEE International Conference on Computation, Communication (ICCCE)*. IEEE, 2012, S. 712–716.

[117] Rosario G Garroppo, Stefano Giordano und Luca Tavanti. „A joint experimental and simulation study of the IEEE 802.11s HWMP protocol and airtime link metric“. In: *Int. Journal of Communication Systems* 25.2 (2011), S. 92–110.

[118] Linux Wireless Wiki: 802.11s. 2020. URL: https://wireless.wiki.kernel.org/en/developers/documentation/ieee80211/802.11s (besucht am 15. 08. 2020).

[119] open80211s Wiki: HowTo. 2020. URL: https://github.com/o11s/open80211s/wiki/HOWTO (besucht am 15. 08. 2020).

[120] Linux Wireless Wiki. 2020. URL: https://wireless.wiki.kernel.org/ (besucht am 15. 08. 2020).

[121] Rui Paulo. „Wireless mesh networks under FreeBSD“. In: *The FreeBSD Project, AsiaBSDCon* (2010).

[122] Linux Wireless Wiki: mac80211. 2020. URL: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211 (besucht am 15. 08. 2020).

[123] Linux Wireless Wiki: Atheros Drivers. 2020. URL: https://wireless.wiki.kernel.org/en/users/drivers/atheros (besucht am 15. 08. 2020).

[124] Pablo Neira-Ayuso, Rafael M. Gasca und Laurent Lefevre. „Communicating between the Kernel and User-Space in Linux Using Netlink Sockets“. In: *Software Practice and Experience* 40.9 (Aug. 2010), S. 797–810. ISSN: 0038-0644.

[125] Linux Wireless Wiki: nl80211. 2020. URL: https://wireless.wiki.kernel.org/en/developers/documentation/nl80211 (besucht am 15. 08. 2020).

[126] libnl Netlink Socket Library. 2020. URL: https://www.infradead.org/~tgr/libnl/ (besucht am 15. 08. 2020).

[127] open80211s Wiki: MeshParameters. 2020. URL: https://github.com/o11s/open80211s/wiki/MeshParameters (besucht am 15. 08. 2020).

[128] Linux Wireless Wiki: iw. 2020. URL: https://wireless.wiki.kernel.org/en/users/documentation/iw (besucht am 15. 08. 2020).

[129] R. Karmakar, S. Chattopadhyay und S. Chakraborty. „Impact of IEEE 802.11n/ac PHY/MAC High Throughput Enhancements on Transport and Application Protocols – A Survey“. In: *IEEE Communications Surveys & Tutorials* PP.99 (Feb. 2017), S. 1–39.

[130] *ns-3 Network Simulator*. 2020. URL: https://www.nsnam.org/ (besucht am 15. 08. 2020).

[131] *OMNet++ Network Simulator*. 2020. URL: https://omnetpp.org/ (besucht am 15. 08. 2020).

[132] W. Felter, A. Ferreira, R. Rajamony und J. Rubio. „An updated performance comparison of virtual machines and Linux containers“. In: *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on*. März 2015, S. 171–172.

[133] Thomas Werthmann, Matthias Kaschub, Mirja Kühlewind, Sebastian Scholz und David Wagner. „VMSimInt: A Network Simulation Tool Supporting Integration of Arbitrary Kernels and Applications“. In: *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics und Telecommunications Engineering). 2014, S. 56–65.

[134] Farooq Sultan, Alex Poylisher, Constantin Serban, Jeyull Lee, Ritu Chadha, C Jason Chiang, Keith Whittaker, Chris Scilla und Shady Ali. „Timesync: Virtual time for scalable, high-fidelity hybrid network emulation“. In: *IEEE MILCOM*. 2012.

[135] Y. Zheng und D. M. Nicol. „A Virtual Time System for OpenVZ-Based Network Emulations“. In: *Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on*. Juni 2011, S. 1–10. DOI: 10.1109/PADS.2011.5936745.

[136] Bastian Blywis, Mesut Guenes, Felix Juraschek und Jochen H. Schiller. „Trends, Advances, and Challenges in Testbed-based Wireless Mesh Network Research“. In: *Mobile Networks and Applications* 15.3 (Juni 2010), S. 315.

[137] Suleyman Uludag, Tom Imboden und Kemal Akkaya. „A taxonomy and evaluation for developing 802.11-based wireless mesh network testbeds“. In: *International Journal of Communication Systems* 25.8 (2012), S. 963–990.

[138] Kishan N Patel u. a. „A survey on emulation testbeds for mobile ad-hoc networks“. In: *Procedia Computer Science* 45 (2015), S. 581–591.

[139] Pablo Serrano, Pablo Salvador, Vincenzo Mancuso und Yan Grunenberger. „Experimenting with commodity 802.11 hardware: overview and future directions“. In: *IEEE Communications Surveys & Tutorials* 17.2 (2015), S. 671–699.

[140] Ying-Dar Lin, Shun-Lee Chang, Jui-Hung Yeh und Shau-Yu Cheng. „Indoor deployment of IEEE 802.11s mesh networks: Lessons and guidelines“. In: *Ad Hoc Networks* 9.8 (2011), S. 1404–1413.

[141] Tom Imboden, Kemal Akkaya und Zach Moore. „Performance evaluation of wireless mesh networks using IEEE 802.11s and IEEE 802.11n“. In: *2012 IEEE International Conference on Communications (ICC)*. IEEE. 2012, S. 5675–5679.

[142] Sandip Chakraborty und Sukumar Nandi. „Controlling unfairness due to physical layer capture and channel bonding in 802.11 n+s wireless mesh networks“. In: *2015 International Conference on Distributed Computing and Networking*. ACM. 2015, S. 21.

[143] Dawood Sajjadi, Maryam Tanha und Jianping Pan. „A comparative study of channel switching latency for conventional and SDN-based routing in multi-hop multi-radio Wireless Mesh Networks“. In: *2016 13th IEEE CCNC*. IEEE. 2016, S. 330–334.

[144] Silvia Krug, Alexander Brychcy und Jochen Seitz. „A mobile embedded-Linux-based testbed for outdoor ad hoc network evaluation“. In: *2016 IEEE WiSEE*. IEEE. 2016, S. 164–166.

[145] Nikos Makris, Thanasis Korakis, Vasileios Maglogiannis, Dries Naudts, Navid Nikaein u. a. „FLEX Testbed: a platform for 4G/5G wireless networking research“. In: ICT Fire Book November 2016, 2016, S. 1–25.

[146] Jakob Pojda, Andreas Wolff, Mohamad Sbeiti und Christian Wietfeld. „Performance analysis of mesh routing protocols for UAV swarming applications“. In: *2011 8th ISWCS*. IEEE. 2011, S. 317–321.

[147] Samira Hayat, Evşen Yanmaz und Christian Bettstetter. „Experimental analysis of multipoint-to-point UAV communications with IEEE 802.11n and 802.11ac“. In: *2015 26th IEEE PIMRC*. IEEE. 2015, S. 1991–1996.

[148] Sagar Sanghani, Timothy X Brown, Shweta Bhandare und Sheetalkumar Doshi. „EWANT: The emulated wireless ad hoc network testbed“. In: *2003 IEEE WCNC*. Bd. 3. IEEE. 2003, S. 1844–1849.

[149] Pradipta De, Ashish Raniwala, Srikant Sharma und Tzi-cker Chiueh. „MiNT: A miniaturized network testbed for mobile wireless research“. In: *2005 24th IEEE INFOCOM*. Bd. 4. IEEE. 2005, S. 2731–2742.

[150] Vinayak Naik, Emre Ertin, Hongwei Zhang und Anish Arora. „Wireless Testbed Bonsai“. In: *2006 4th IEEE WiOpt*. IEEE. 2006, S. 1–9.

[151] Albert A Lysko und David L Johnson. „A study of propagation effects in a wireless test bed“. In: *WSEAS Transactions on Communications* 7.8 (2008), S. 857–871.

[152] Yang Su und Thomas Gross. „Validation of a miniaturized wireless network testbed“. In: *2008 3rd ACM WiNTECH*. ACM. 2008, S. 25–32.

[153] Sherif M. ElRakabawy, Simon Frohn und Christoph Lindemann. „A scalable dual-radio wireless testbed for emulating mesh networks“. In: *Wireless Networks* 16.8 (2010), S. 2191–2207.

[154] Konstanty Bialkowski und Marius Portmann. „Design of testbed for wireless mesh networks“. In: *2010 IEEE APSURSI*. IEEE. 2010, S. 1–4.

[155] Pablo Serrano, Carlos J Bernardos, Antonio de La Oliva, Albert Banchs, Ignacio Soto und Michael Zink. „FloorNet: deployment and evaluation of a multihop wireless 802.11 testbed“. In: *EURASIP Journal on Wireless Communications and Networking* 2010 (2010), S. 8.

[156] T Charles Clancy und Brenton D Walker. „MeshTest: Laboratory-based wireless testbed for large topologies“. In: *2007 3rd TridentCom*. IEEE. 2007, S. 1–6.

[157] Patch ath9k: add noise floor override option. 2017. URL: https://patchwork.kernel.org/patch/9641107/ (besucht am 15. 08. 2020).

[158] Raspberry Pi. 2020. URL: http://www.raspberrypi.org/ (besucht am 15. 08. 2020).

[159] PCEngines. *APU2 System Boards*. 2020. URL: https://pcengines.ch/apu2.htm (besucht am 15. 08. 2020).

[160] Solid-Run. *Hummingboard*. 2020. URL: https://www.solid-run.com/nxp-family/hummingboard/ (besucht am 15. 08. 2020).

[161] Intel Galileo Gen. 1. 2020. URL: https://www.arduino.cc/en/ArduinoCertified/IntelGalileo (besucht am 15. 08. 2020).

[162] Compex. *WLE200NX*. 2020. URL: https://compex.com.sg/shop/wifi-module/802-11n/wle200nx/ (besucht am 15. 08. 2020).

[163] Mini-Circuits. *VAT-30+ Signal Attenuators*. 2020. URL: https://www.minicircuits.com/pdfs/VAT-30.pdf (besucht am 15. 08. 2020).

[164] Jim Martin, Jack Burbank, William Kasch und Professor David L. Mills. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 5905. Juni 2010. DOI: 10.17487/RFC5905.

[165] Vinay Shankarkumar, Laurent Montini, Tim Frost und Greg Dowd. *Precision Time Protocol Version 2 (PTPv2) Management Information Base*. RFC 8173. Juni 2017. DOI: 10.17487/RFC8173.

[166] T. Kovácsházy und B. Ferencz. „Performance evaluation of PTPd, a IEEE 1588 implementation, on the x86 Linux platform for typical application scenarios“. In: *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*. 2012, S. 2548–2552.

[167] Zhinong Ying, Chi-Yuk Chiu, Kun Zhao, Shuai Zhang und Sailing He. „Antenna Design for Diversity and MIMO Application“. In: *Handbook of Antenna Technologies*. Juli 2015, S. 1–43. DOI: 10.1007/978-981-4560-75-7_53-1.

[168] *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*. 2020. URL: https://iperf.fr/ (besucht am 15. 08. 2020).

[169] Sangtae Ha, Injong Rhee und Lisong Xu. „CUBIC: a new TCP-friendly high-speed TCP variant“. In: *ACM SIGOPS OS Review* 42.5 (2008).

[170] GT Mailing List. *TCP reaching to maximum throughput after a long time*. 2020. URL: https://lists.gt.net/linux/kernel/2414672 (besucht am 15. 08. 2020).

[171] P Series. „Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 900 MHz to 100 GHz“. In: *Recommendation ITU-R* (2012), S. 1238–7.

[172] United States Patent 7349503: Adaptive Interference Immunity Control. 2020. URL: http://www.freepatentsonline.com/7349503.html (besucht am 15. 08. 2020).

[173] Jemish V Maisuria und Reena M Patel. „Overview of Techniques for Improving QoS of TCP over Wireless Links“. In: *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*. IEEE. 2012.

[174] Mario Bisignano, Giuseppe Di Modica, Orazio Tomarchio und Lorenzo Vita. „P2P over MANET: A Comparison of Cross-layer Approaches“. In: *18th International Workshop on Database and Expert Systems Applications*. IEEE, 2007, S. 814–818.

[175] Bartosz Biskupski, Jim Dowling und Jan Sacha. „Properties and mechanisms of self-organizing MANET and P2P systems“. In: *ACM Transactions on Autonomous and Adaptive Systems* 2.1 (2007), S. 1.

[176] Jörg Eberspächer und Rüdiger Schollmeier. „First and Second Generation of Peer-to-Peer Systems“. In: *Peer-to-Peer Systems and Applications*. Springer Berlin Heidelberg, 2005, S. 35–56.

[177] Ralf Steinmetz und Klaus Wehrle. *P2P Systems and Applications, Springer Lecture Notes in Computer Science*. Springer-Verlag Berlin, 2005.

[178] Bram Cohen. „Incentives Build Robustness in BitTorrent“. In: *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*. 2003.

[179] Gang Ding, John Vicente, Sanjay Rungta, Dilip Krishnaswamy, Winson Chan und Kai Miao. „Overlays on wireless mesh networks: implementation and cross-layer searching“. In: *Proceedings of the 9th IFIP/IEEE international conference on Management of Multimedia and Mobile Networks and Services*. MMNS'06. Dublin, Ireland: Springer, 2006, S. 171–182. ISBN: 978-3-540-47654-2.

[180] Sherif ElRakabawy und Christoph Lindemann. „P2P file transfer in wireless mesh networks“. In: *Fourth Annual Conference on Wireless on Demand Network Systems and Services*. IEEE, 2007, S. 114–121.

[181] Antonio Marques, Fernando Mira da Silva und Rui Rocha. „P2P over Mobile Ad-hoc Networks“. In: *6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*. IEEE, 2009, S. 1–3.

[182] Nguyen Chan Hung, VT Vinh u. a. „Challenges in the development of mobile P2P applications and services“. In: *Journal of Infomation & Communications, Vietnam Ministry of Information and Communications* (2009).

[183] Vuze BitTorrent Client. 2020. URL: https://dev.vuze.com/ (besucht am 15. 08. 2020).

[184] Arnaud Legout, Guillaume Urvoy-Keller und Pietro Michiardi. „Rarest first and choke algorithms are enough“. In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006, S. 203–216.

[185] Paul Gillard, Padmini Vellore und Ramachandran Venkatesan. „BEAN - BitTorrent enabled ad hoc networks“. In: *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*. Bd. 3. IEEE, 2005, S. 186–191.

[186] Sundaram Rajagopalan und Chien-Chung Shen. „A Cross-layer Decentralized BitTorrent for Mobile Ad hoc Networks“. In: *Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*. IEEE, 2006, S. 1–10.

[187] Gábor Balázsfalvi und János Sztrik. „BitTorrent File Sharing in Mobile Ad Hoc Networks“. In: *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae*. 2006, S. 159–170.

[188] Pietro Michiardi und Guillaume Urvoy-Keller. „Performance analysis of cooperative content distribution in wireless ad hoc networks“. In: *Fourth Annual Conference on Wireless on Demand Network Systems and Services*. IEEE, 2007, S. 22–29.

[189] Amir Krifa, Mohamed Karim Sbai, Chadi Barakat und Thierry Turletti. „BitHoc: A content sharing application for Wireless Ad hoc Networks“. In: *IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2009, S. 1–3.

[190] Mohamed Karim Sbai, Emna Salhi und Chadi Barakat. „A Membership Management Protocol for Mobile P2P Networks“. In: *6th International Conference on Mobile Technology, Application & Systems*. ACM, 2009, S. 1–8.

[191] Nivia Cruz Quental und Paulo Andre da S. Goncalves. „Exploiting application-layer strategies for improving BitTorrent performance over MANETs“. In: *IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2011, S. 691–692.

[192] Francesco Paolo D'Elia, Giovanni Di Stasi, Stefano Avallone und Roberto Canonico. „BitTorrent traffic optimization in Wireless Mesh Networks with ALTO service“. In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, 2011, S. 1–6.

[193] Giovanni Di Stasi, Roberto Bifulco, Francesco Paolo D'Elia, Stefano Avallone, Roberto Canonico, Apostolos Apostolaras, Nikolaos Giallelis, Thanasis Korakis und Leandros Tassiulas. „Experimenting with P2P traffic optimization for wireless mesh networks in a federated OMF-PlanetLab environment“. In: *IEEE Wireless Communications and Networking Conference*. IEEE, 2011, S. 719–724.

[194] Afzal Mawji und Hossam Hassanein. „Efficient content distribution for peer-to-peer overlays on mobile ad hoc networks“. In: *Journal of Advanced Research* 2.3 (2011), S. 265–279.

[195] Marcel Castro, Andreas Kassler und Stefano Avallone. „BestPeer - a load-aware multi-path peer selection for Wireless Mesh Networks“. In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, 2012, S. 1–6.

[196] Enrico Casini, Giacomo Benincasa, Alessandro Morelli, Niranjan Suri und Maggie Breedy. „An experimental evaluation of data distribution applications in tactical networks“. In: *IEEE Military Communications Conference*. IEEE, 2016, S. 1267–1272.

[197] Emna Salhi, Mohamed Karim Sbai, Chadi Barakat u. a. „Neighborhood selection in mobile P2P networks“. In: *Algotel*. Carry-Le-Rouet, France, 2009.

[198] IETF. *Internet Draft: Multicast Considerations over IEEE 802 Wireless Media*. 2020. URL: https://datatracker.ietf.org/doc/draft-ietf-mboned-ieee802-mcast-problems/ (besucht am 15. 08. 2020).

[199] B. T. Vijay und B. Malarkodi. „A study of IEEE 802.11aa“. In: *2017 8th Int. Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2017, S. 1–6.

[200] Pablo Salvador, Luca Cominardi, Francesco Gringoli und Pablo Serrano. „A First Implementation and Evaluation of the IEEE 802.11aa Group Addressed Transmission Service“. In: *SIGCOMM Computer Communication Review* 44.1 (2016), S. 35–41. ISSN: 0146-4833.

[201] Yousri Daldoul, Djamal-Eddine Meddour, Toufik Ahmed und Raouf Boutaba. „Performance and scalability evaluation of IEEE 802.11v/aa multicast transport“. In: *Wireless Communications and Mobile Computing* 16.14 (2016), S. 1987–2000. ISSN: 1530-8677.

[202] E. Coronado, R. Riggio, J. Villalón und A. Garrido. „Joint Mobility Management and Multicast Rate Adaptation in Software–Defined Enterprise WLANs“. In: *IEEE Transactions on Network and Service Management* 15.2 (2018), S. 625–637.

[203] Krishna Ramachandran, Irfan Sheriff, Elizabeth Belding und Kevin Almeroth. „Routing Stability in Static Wireless Mesh Networks“. In: *Proceedings of the 8th International Conference on Passive and Active Network Measurement*. PAM'07. Louvain-la-Neuve, Belgium: Springer-Verlag, 2007, S. 73–83. ISBN: 9783540716167.

[204] Saumitra M. Das, Himabindu Pucha, Konstantina Papagiannaki und Y. Charlie Hu. „Studying Wireless Routing Link Metric Dynamics“. In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. IMC '07. San Diego, California, USA: Association for Computing Machinery, 2007, S. 327–332. DOI: 10.1145/1298306.1298352.

[205] Y. Huang und S. Bhatti. „Fast-Converging Distance Vector Routing for Wireless Mesh Networks“. In: *2008 The 28th International Conference on Distributed Computing Systems Workshops*. 2008, S. 279–284.

[206] BASS Plugin. 2020. URL: https://code.google.com/archive/p/bass-plugin/ (besucht am 15. 08. 2020).

[207] Oracle. *Java Docs: Class ProcessBuilder*. 2020. URL: https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html (besucht am 15. 08. 2020).

[208] Lucia D'Acunto, Nitin Chiluka, TamáS Vinkó und Henk Sips. „BitTorrent-like P2P approaches for VoD: A comparative study“. In: *Computer Networks* 57.5 (2013), S. 1253–1276. ISSN: 1389-1286.

[209] *Bencode Specification*. 2020. URL: https://wiki.theory.org/index.php/BitTorrentSpecification#Bencoding (besucht am 15. 08. 2020).

[210] BitTorrent. *Multitracker Metadata Extension*. 2020. URL: http://www.bittorrent.org/beps/bep_0012.html (besucht am 15. 08. 2020).

[211] BitTorrent. *DHT Protocol*. 2020. URL: http://www.bittorrent.org/beps/bep_0005.html (besucht am 15. 08. 2020).

[212] BitTorrent. *Extension for Peers to Send Metadata Files*. 2020. URL: http://www.bittorrent.org/beps/bep_0009.html (besucht am 15. 08. 2020).

[213] BitTorrent. *Peer Exchange (PeX)*. 2020. URL: http://www.bittorrent.org/beps/bep_0011.html (besucht am 15. 08. 2020).

[214] BitTorrent. *LAN Peer Discovery (Local Service Discovery)*. 2020. URL: http://www.bittorrent.org/beps/bep_0014.html (besucht am 15. 08. 2020).

[215] Vuze Plugin: LAN Peer Finder. 2020. URL: https://wiki.vuze.com/w/UG_Plugins#LAN_Peer_Finder (besucht am 15. 08. 2020).

[216] Cleitianne Silva, Yuri Oliveira, Clayson Celes, Reinaldo Braga und Carina Oliveira. „Performance Evaluation of Wireless Mesh Networks in Smart Cities Scenarios“. In: *Proceedings of the Euro American Conference on Telematics and Information Systems*. EATIS '18. Fortaleza, Brazil: Association for Computing Machinery, 2018. DOI: 10.1145/3293614.3293615.

[217] Matthias Handy, Marc Haase und Dirk Timmermann. „Low-Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection“. In: *4th IEEE International Conference on Mobile and Wireless Communications Networks*. IEEE. Sep. 2002, S. 368–372. ISBN: 0-7803-7606-4.

[218] Jiaxi You, Dominik Lieckfeldt, Matthias Handy und Dirk Timmermann. „Budget-Based Clustering with Context-awareness for Sensor Networks“. In: *4th IEEE International Workshop on Sensor Networks and Systems for Pervasive Computing*. IEEE. März 2008, S. 306–311. ISBN: 978-0-7695-3113-7.

[219] Jakob Salzmann, Ralf Behnke und Dirk Timmermann. „Hex-MASCLE - Hexagon based Clustering with Self Healing Abilities“. In: *IEEE Wireless Communications and Networking Conference*. IEEE-SOCNE. März 2011, S. 1339–1344. ISBN: 978-1-61284-253-0.

[220] Jakob Salzmann, Ralf Behnke und Dirk Timmermann. „A Localization-Free Wireless Sensor Network Clustering Approach with Convergence to Uniformity“. In: *International Forum Life Science Automation*. LSA. Sep. 2008, S. 81. ISBN: 978-3-938042-17-5.

[221] Jakob Salzmann, Ralf Behnke, Jiaxi You und Dirk Timmermann. „Free-CLASH – Improved Localization Free Clustering in Large Wireless Sensor Networks“. In: *The International Workshop on Scalable Ad Hoc and Sensor Networks*. Okt. 2009. ISBN: 978-1-4244-3941-6.

[222] Wenli Chen, Nitin Jain und Suresh Singh. „ANMP: Ad hoc network management protocol“. In: *Selected Areas in Communications, IEEE Journal on* 17.8 (1999), S. 1506–1531.

[223] Chien-Chung Shen, Chaiporn Jaikaeo, Chavalit Srisathapornphat und Zhuochuan Huang. „The Guerrilla management architecture for ad hoc networks“. In: *MILCOM 2002. Proceedings*. Bd. 1. IEEE. 2002, S. 467–472.

[224] Ting-Chao Hou und Tzu-Jane Tsai. „A access-based clustering protocol for multihop wireless ad hoc networks“. In: *IEEE Journal on Selected Areas in Communications* 19.7 (2001), S. 1201–1210.

[225] Soumendra Nanda und David Kotz. „Mesh-Mon: A multi-radio mesh monitoring and management system“. In: *Computer Communications* (2008).

[226] Vivek Aseeja und Rong Zheng. „Meshman: A management framework for wireless mesh networks“. In: *Integrated Network Management, 2009. IFIP/IEEE International Symposium on*. IEEE. 2009, S. 226–233.

[227] A. Naveed und S. S. Kanhere. „Cluster-based channel assignment in multi-radio multi-channel wireless mesh networks“. In: *IEEE 34th Conference on Local Computer Networks*. 2009, S. 53–60.

[228] N. Letor, C. Blondia, S. Bouckaert, I. Moerman und P. Demeester. „A cluster driven channel assignment mechanism for wireless mesh networks“. In: *5th IEEE Int. Conference on Mobile Ad Hoc and Sensor Systems*. 2008, S. 659–665.

[229] A. Brzezinski, G. Zussman und E. Modiano. „Distributed Throughput Maximization in Wireless Mesh Networks via Pre-Partitioning“. In: *IEEE/ACM Transactions on Networking* 16.6 (Dez. 2008), S. 1406–1419. ISSN: 1063-6692.

[230] S. Avallone und I. F. Akyildiz. „A Channel Assignment Algorithm for Multi-Radio Wireless Mesh Networks“. In: *16th Int. Conference on Computer Communications and Networks*. Aug. 2007, S. 1034–1039.

[231] S. A. Makram und M. Gunes. „Distributed channel assignment for multi-radio wireless mesh networks“. In: *IEEE Symposium on Computers and Communications*. Juli 2008, S. 272–277.

[232] A. Naveed, S. S. Kanhere und S. K. Jha. „Topology Control and Channel Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks“. In: *IEEE Int. Conference on Mobile Adhoc and Sensor Systems*. 2007, S. 1–9.

[233] B. Raju, K. Athota und A. Negi. „A Distributed Cluster based Interference-Traffic aware CA for MRMC WMN“. In: *5th Int. Conference on Wireless Communication and Sensor Networks (WCSN)*. 2009, S. 1–6.

[234] Chao Liu, Zhongyi Liu, Yongqiang Liu, Huizhou Zhao, Tong Zhao und Wei Yan. „A Clustering-based Channel Assignment Algorithm and Routing Metric for Multi-channel Wireless Mesh Networks“. In: *5th Int. Conference on Parallel and Distributed Processing and Applications*. ISPA'07. Springer, 2007, S. 832–843.

[235] A. Barrat, T. Erlebach, M. Mihalák und A. Vespignani. „A (Short) Survey on Network Discovery“. In: *The European integrated project "Dynamically Evolving, Large Scale Information Systems (DELIS)" : Proceedings of the Final Workshop, Barcelona, February 27-28, 2008*. Bd. 222. HNI-Verlagsschriftenreihe. Paderborn: Heinz Nixdorf Inst., 2008, S. 63–80. ISBN: 978-3-939350-41-5.

[236] Mohamed Guesmia, Mustapha Guezouri und Nader Mbarek. „Performance evaluation of the HWMP proactive tree mode for IEEE 802.11s based Wireless Mesh Networks“. In: *Communications and Networking (ComNet), 2012 Third Int. Conference on*. IEEE. 2012, S. 1–7.

[237] Dr. Thomas Narten, Tatsuya Jinmei und Dr. Susan Thomson. *IPv6 Stateless Address Autoconfiguration*. RFC 4862. Sep. 2007. DOI: 10.17487/RFC4862.

[238] V. Angelakis, S. Papadakis, V. A. Siris und A. Traganitis. „Adjacent channel interference in 802.11a is harmful: Testbed validation of a simple quantification model“. In: *IEEE Communications Magazine* 49.3 (März 2011), S. 160–166. ISSN: 0163-6804.

[239] W. L. Tan, K. Bialkowski und M. Portmann. „Evaluating Adjacent Channel Interference in IEEE 802.11 Networks“. In: *IEEE 71st Vehicular Technology Conference*. 2010, S. 1–5.

[240] Uthayakumar .J, T. Vengattaraman und P. Dhavachelvan. „A Survey on Data Compression Techniques: From the Perspective of Data Quality, Coding Schemes, Data Type and Applica-

tions“. In: *Journal of King Saud University - Computer and Information Sciences* (Mai 2018). DOI: 10.1016/j.jksuci.2018.05.006.

[241] L.A. Fitriya, Tito Purboyo und Anggunmeka Prasasti. „A review of data compression techniques“. In: *Int. Journal of Applied Engineering Research* 12 (Jan. 2017), S. 8956–8963.

[242] Paramvir Bahl, Ranveer Chandra und John Dunagan. „SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-hoc Wireless Networks“. In: *10th Annual Int. Conference on Mobile Computing and Networking*. MobiCom '04. ACM, 2004, S. 216–230. ISBN: 1-58113-868-7.

[243] *GNU Octave*. 2020. URL: https://www.gnu.org/software/octave/ (besucht am 15. 08. 2020).

[244] Y. Saadi, B. Nassereddine, S. Bennani und A. Maach. „An adaptive approach to control broadcast traffic in wireless mesh networks based IEEE 802.11s“. In: *2012 IEEE International Conference on Complex Systems (ICCS)*. 2012, S. 1–7.

[245] *ucspi-tcp*. 2020. URL: https://cr.yp.to/ucspi-tcp.html (besucht am 15. 08. 2020).

[246] *GNU netcat*. 2020. URL: http://netcat.sourceforge.net/ (besucht am 15. 08. 2020).

[247] S. Pilosof, Ramachandran Ramjee, D. Raz, Y. Shavitt und Prasun Sinha. „Understanding TCP fairness over wireless LAN“. In: *IEEE INFOCOM 2003. 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*. Bd. 2. 2003, S. 863–872.

[248] D. J. Leith, P. Clifford, D. Malone und A. Ng. „TCP fairness in 802.11e WLANs“. In: *IEEE Communications Letters* 9.11 (2005), S. 964–966.

[249] Mojtaba Seyedzadegan, Mohamed Othman, Shamala Subramaniam und Zuriati Zukarnain. „The TCP fairness in WLAN: A review“. In: *2007 IEEE Int. Conference on Telecommunications and Malaysia Int. Conference on Communications*. 2007, S. 644–648.

[250] J. Yoo und J. Kim. „Impact of TCP ACK Losses on TCP Fairness in Wireless Mesh Networks“. In: *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*. 2009, S. 1–6.

[251] S. R. Pokhrel, M. Panda, H. L. Vu und M. Mandjes. „TCP Performance over Wi-Fi: Joint Impact of Buffer and Channel Losses“. In: *IEEE Transactions on Mobile Computing* 15.5 (2016), S. 1279–1291.

[252] Andrey Krendzel. *Wireless Mesh Networks - Efficient Link Scheduling, Channel Assignment and Network Planning Strategies*. Aug. 2012. ISBN: ISBN 978-953-51-0672-2.

[253] P. Kyasanur und N. H. Vaidya. „Routing and interface assignment in multi-channel multi-interface wireless networks“. In: *IEEE Wireless Communications and Networking Conference, 2005*. Bd. 4. 2005, 2051–2056 Vol. 4.

[254] „IEEE Standard for Local and metropolitan area networks – Link Aggregation“. In: *IEEE Std 802.1AX-2014 (Revision of IEEE Std 802.1AX-2008)* (2014), S. 1–344.

[255] Alan Ford, Costin Raiciu, Mark J. Handley und Olivier Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824. Jan. 2013. DOI: 10.17487/RFC6824.

[256] Olivier Bonaventure, Christoph Paasch und Gregory Detal. *Use Cases and Operational Experience with Multipath TCP*. RFC 8041. Jan. 2017. DOI: 10.17487/RFC8041.

[257] David Gómez Fernández, Carlos Rabadán, Pablo Garrido und Ramon Aguero. „Multipath Algorithms and Strategies to Improve TCP Performance over Wireless Mesh Networks“. In: *Mobile Networks and Management. MONAMI 2013. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Bd. 125. Springer. Sep. 2013, S. 15–28. DOI: 10.1007/978-3-319-04277-0_2.

[258] S. Gheorghiu, A. L. Toledo und P. Rodriguez. „Multipath TCP with Network Coding for Wireless Mesh Networks“. In: *2010 IEEE Int. Conference on Communications*. 2010, S. 1–5.

A Liste der Veröffentlichungen und Fachvorträge auf Tagungen

[1] Christoph Niemann, Christian Ewert, Henning Puttnies, Michael Rethfeldt, Dirk Timmermann und Peter Danielis. „Modeling Energy Consumption for Task-Offloading Decisions on Mobile and Embedded Devices“. In: *IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech)*. Kyoto, Japan: IEEE, März 2020, S. 400–404. DOI: 10.1109/LifeTech48969.2020.1570618809.

[2] Arne Wall, Björn Butzin, Frank Golatowski, Michael Rethfeldt und Dirk Timmermann. „Software-Defined Security Architecture for Smart Buildings Using the Building Information Model“. In: *IEEE 3rd Global Conference on Internet of Things (GCIoT)*. Dubai, VAE: IEEE, Dez. 2019, S. 1–5. DOI: 10.1109/GCIoT47977.2019.9058404.

[3] Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Tim Brockmann, Christian Haubelt und Dirk Timmermann. „CHaChA: Clustering Heuristic and Channel Assignment for IEEE 802.11s Mesh Networks“. In: *IEEE 9th Annual Information Technology, Electronics & Mobile Communication Conference (IEMCON)*. Vancouver, Kanada: IEEE, Nov. 2018, S. 1–7. DOI: 10.1109/IEMCON.2018.8615043.

[4] Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Felix Uster, Christian Haubelt und Dirk Timmermann. „MeNTor: A Wireless-Mesh-Network-Aware Data Dissemination Overlay based on BitTorrent“. In: *Elsevier Ad Hoc Networks* 79.1 (Okt. 2018), S. 146–159. DOI: 10.1016/j.adhoc.2018.06.013.

[5] Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Christian Haubelt und Dirk Timmermann. „Enabling the Management of IEEE 802.11s Wireless Mesh Networks“. In: *GI/ITG 17. KuVS (Kommunikation und Verteilte Systeme) FGSN (Fachgespräch Sensornetze)*. Braunschweig, Deutschland: GI/ITG, Sep. 2018, S. 1–4. DOI: DOI:10.24355/dbbs.084-201809121401-1.

[6] Michael Rethfeldt, Benjamin Beichler, Hannes Raddatz, Felix Uster, Peter Danielis, Christian Haubelt und Dirk Timmermann. „Mini-Mesh: Practical Assessment of a Miniaturized IEEE 802.11n/s Mesh Testbed“. In: *IEEE 16th Wireless Communications and Networking Conference (WCNC)*. Barcelona, Spanien: IEEE, Apr. 2018, S. 1–6. DOI: 10.1109/WCNC.2018.8377247.

[7] Arne Wall, Hannes Raddatz, Michael Rethfeldt, Peter Danielis und Dirk Timmermann. „Performance Evaluation of MAC-Layer Trust Zones over Virtual Network Interfaces“. In: *IEEE 4th Conference on Mobile and Secure Services (MobiSecServ)*. Miami Beach, Florida, USA: IEEE, Feb. 2018, S. 1–6. DOI: 10.1109/MOBISECSERV.2018.8311442.

[8] Arne Wall, Hannes Raddatz, Michael Rethfeldt, Peter Danielis und Dirk Timmermann. „ANTs: Application-Driven Network Trust Zones on MAC-Layer in Smart Buildings“. In: *IEEE 15th Consumer Communications & Networking Conference (CCNC)*. Las Vegas, NV, USA: IEEE, Jan. 2018, S. 1–2. DOI: 10.1109/CCNC.2018.8319304.

[9] Martin Kasparick, Benjamin Beichler, Björn Konieczek, Andreas Besting, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „Measuring Latencies of IEEE 11073 Compliant Service-Oriented Medical Device Stacks“. In: *IEEE 11th Workshop on Service-Oriented*

Cyber-Physical Systems in Converging Networked Environments (SOCNE). Peking, China: IEEE, Okt. 2017, S. 8640–8647. DOI: 10.1109/IECON.2017.8217518.

[10] Benjamin Beichler, Michael Rethfeldt, Hannes Raddatz, Björn Konieczek, Peter Danielis, Christian Haubelt und Dirk Timmermann. „Optimization of a novel WLAN Simulation Framework for Prototyping Network Applications and Protocols“. In: *20th GI/ITG/GMM Workshop - Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV)*. Bremen, Deutschland: GI/ITG/GMM, Feb. 2017.

[11] Michael Rethfeldt, Hannes Raddatz, Benjamin Beichler, Björn Konieczek, Dirk Timmermann, Christian Haubelt und Peter Danielis. „ViPMesh: A Virtual Prototyping Framework for IEEE 802.11s Wireless Mesh Networks“. In: *IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. New York, USA: IEEE, Okt. 2016, S. 1–7. DOI: 10.1109/WiMOB.2016.7763263.

[12] Björn Konieczek, Jan Skodzik, Peter Danielis, Vlado Altmann, Michael Rethfeldt und Dirk Timmermann. „HaRTKad: A P2P-based Concept for Deterministic Communication and Its Limitiations“. In: *IEEE 21st International Symposium on Computers and Communications (ISCC)*. Messina, Italien: IEEE, Juni 2016, S. 1198–1203. DOI: 10.1109/ISCC.2016.7543893.

[13] Björn Konieczek, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „A Distributed Time Server for the Real-Time Extension of CoAP“. In: *IEEE 19th International Symposium on Real-Time Computing (ISORC)*. York, England: IEEE, Mai 2016, S. 84–91. DOI: 10.1109/ISORC.2016.21.

[14] Björn Konieczek, Martin Kasparick, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „Towards a TDMA-based Real-Time Extension for the Constrained Application Protocol“. In: *IEEE 12th World Conference on Factory Communication Systems (WFCS)*. Aveiro, Portugal: IEEE, Mai 2016, S. 1–4. DOI: 10.1109/WFCS.2016.7496529.

[15] Michael Rethfeldt, Peter Danielis, Benjamin Beichler, Björn Konieczek, Felix Uster und Dirk Timmermann. „Evaluating Cross-Layer Cooperation of Congestion and Flow Control in IEEE 802.11s Networks“. In: *IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. Crans-Montana, Schweiz: IEEE, März 2016, S. 181–188. DOI: 10.1109/AINA.2016.12.

[16] Michael Rethfeldt, Arne Wall, Peter Danielis, Björn Konieczek und Dirk Timmermann. „AKadeMesh: Software-Defined Overlay Adaptation for the Management of IEEE 802.11s Networks“. In: *IEEE 13th Consumer Communications & Networking Conference (CCNC)*. Las Vegas, NV, USA: IEEE, Jan. 2016, S. 477–482. DOI: 10.1109/CCNC.2016.7444826.

[17] Michael Rethfeldt, Peter Danielis, Björn Konieczek, Felix Uster und Dirk Timmermann. „Integration of QoS Parameters From IEEE 802.11s WLAN Mesh Networks Into Logical P2P Overlays“. In: *IEEE International Conference on Ubiquitous Computing and Communications (IUCC)*. Liverpool, England: IEEE, Okt. 2015, S. 1170–1177. DOI: 10.1109/CIT/IUCC/DASC/PICOM.2015.175.

[18] Michael Rethfeldt, Peter Danielis, Guido Moritz, Björn Konieczek und Dirk Timmermann. „Design and Development of a Management Solution for Wireless Mesh Networks based on IEEE 802.11s“. In: *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ottawa, Kanada: IFIP/IEEE, Mai 2015, S. 902–905. DOI: 10.1109/INM.2015.7140405.

[19] Björn Konieczek, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „Real-Time Communication for the Internet of Things using jCoAP“. In: *IEEE 18th International Symposium on Real-Time Computing (ISORC)*. Auckland, Neuseeland: IEEE, Apr. 2015, S. 134–141. DOI: 10.1109/ISORC.2015.35.

[20] Michael Rethfeldt. „Dynamic Optimization of Communication Resources and Communication Strategies in Wireless Mesh Networks“. In: *8th Joint Workshop of the German Research Training Groups in Computer Science*. Dagstuhl, Deutschland, Juni 2014. ISBN: 978-3-86386-719-5.

B Liste der betreuten studentischen Arbeiten

[1] Farooq Ahmad. „RSSI-based Indoor Localization of Wireless Sensor Nodes“. Masterarbeit. Universität Rostock, 2020.

[2] Lisa-Marie Odebrecht. „Development of a Mobile Particulate Matter Sensor for Smart City Applications“. Bachelorarbeit. Universität Rostock, 2020.

[3] Lukas Steffen. „Design and Development of a SystemC-based Simulation of the IEEE 802.11 Channel Access Mechanism EDCA“. Bachelorarbeit. Universität Rostock, 2020.

[4] Benjamin Bartels. „Implementation of an Anchor-Free Distributed Localization System in Wireless Mesh Networks“. Bachelorarbeit. Universität Rostock, 2020.

[5] Preetham Rengarajan Kannan. „Simulation of Building Automation Networks using unmodified Applications“. Spezialisierungsmodul EE. Universität Rostock, 2019.

[6] Tim Brockmann. „Weiterentwicklung des CHaChA-Verfahrens für die robuste Cluster-Anpassung bei Topologieänderungen in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Masterarbeit. Universität Rostock, 2018.

[7] Tim Brockmann. „Praktische Untersuchung eines verteilten Ansatzes für räumliches und spektrales Clustering in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Masterprojekt. Universität Rostock, 2018.

[8] Jakob Heller. „Entwicklung und Bewertung einer verteilten Content-Caching-Lösung für IEEE-802.11s-WLAN-Mesh-Netzwerke“. Masterarbeit. Universität Rostock, 2018.

[9] Jakob Heller. „Berücksichtigung von physikalischer Netzwerk-Nähe im DHT-Protokoll Kademlia“. Masterprojekt. Universität Rostock, 2017.

[10] Amir Khatib. „Designablauf- und Regelwerkerstellung zur Implementierung einer Onboard-Controller-DDR4RAM-Interface-Leiterplattenimplementierung, einschließlich deren Simulation und Verifikation“. Bachelorarbeit. Universität Rostock, 2017.

[11] Iven Silvio Kolterjahn. „Umsetzung und Bewertung einer verteilten Kanalselektionsstrategie für IEEE 802.11s - WLAN-Mesh-Netzwerke“. Masterarbeit. Universität Rostock, 2017.

[12] Felix Uster. „Umsetzung und Bewertung einer Cross-Layer-Optimierung für TCP und ARQ in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Masterarbeit. Universität Rostock, 2016.

[13] Felix Uster. „Dynamische TCP-orientierte ARQ-Optimierung in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Masterprojekt. Universität Rostock, 2016.

[14] Felix Uster. „Evaluierung von Cross-Layer-Verfahren für Überlast- und Flusskontrolle in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Bachelorarbeit. Universität Rostock, 2015.

[15] Sascha Rohde. „Implementierung und Evaluierung einer energieautarken Bluetooth-LE-Lokalisierungslösung“. Masterprojekt. Universität Rostock, 2015.

[16] Sascha Rohde. „Implementierung und Evaluierung einer Bluetooth-Low-Energy-Lösung zur energieeffizienten Ortung von Objekten“. Bachelorarbeit. Universität Rostock, 2015.

[17] Daniel Roisch. „Implementierung und Bewertung einer Lösung zur dynamischen Kanalselektion in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Bachelorarbeit. Universität Rostock, 2015.

[18] Daniel Roisch. „Untersuchung und Bewertung von dynamischen Kanalselektionsverfahren für WLAN-Mesh-Netzwerke“. Literaturarbeit. Universität Rostock, 2015.

[19] Hannes Raddatz. „Entwurf und Bewertung einer Emulations-Simulationsumgebung für IEEE 802.11s - MIMO-WLAN-Mesh-Netzwerke im Mehrkanalbetrieb“. Masterarbeit. Universität Rostock, 2015.

[20] Hannes Raddatz. „Aufbau und Bewertung einer Simulations-Emulationsumgebung für IEEE 802.11s - WLAN-Mesh-Netzwerke“. Masterprojekt. Universität Rostock, 2015.

[21] Arne Wall. „P2P-basierte Verteilung von Ressourcen und Managementfunktionen in IEEE 802.11s - WLAN-Mesh-Netzwerken“. Masterarbeit. Universität Rostock, 2015.

[22] Arne Wall. „Entwurf und Evaluation eines Kad-basierten P2P-Systems für Android-Plattformen“. Masterprojekt. Universität Rostock, 2014.

[23] Felix Uster. „Gegenüberstellung der Überlast- und Flusskontrollverfahren auf Sicherungs-, Vermittlungs- und Transportschicht“. Literaturarbeit. Universität Rostock, 2014.

[24] Sebastian Stolz. „Integration von Dienstgüteparametern aus physikalischen WLAN-Mesh-Netzwerken in logische P2P-Netzwerke“. Masterarbeit. Universität Rostock, 2014.

[25] Hannes Raddatz. „Modellierung, Simulation und Analyse von MIMO-Kommunikationssystemen“. Bachelorarbeit. Universität Rostock, 2013.