Dissertation

Protokolle und Algorithmen für zuverlässige drahtlose Kommunikationssysteme

Vom Fachbereich Informatik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation von

Christopher Kramer

Datum der Einreichung:	07.02.2020
Datum der wissenschaftlichen Aussprache:	11.01.2021
Dekan:	Prof. Dr. Jens Schmitt
Promotionskommision:	Prof. Dr. Karsten Berns (Vorsitz), Prof. Dr. Reinhard Gotzhein, Prof. Dr. Jens Schmitt
Berichterstatter:	Prof. Dr. Reinhard Gotzhein, Prof. Dr. Jens Schmitt, Prof. Dr. Reinhard German

D 386

Danksagung

Während meines Studiums und meiner Promotion habe ich durch viele Menschen wertvolle Unterstützung erfahren, für die ich mich an dieser Stelle bedanken möchte. Zuerst möchte ich mich bei meinem Doktorvater Prof. Dr. Reinhard Gotzhein bedanken. In seiner Vorlesung *Algorithms in Ad-Hoc-Networks* schürte er mein Interesse für Kommunikationsprotokolle und bot mir an, bei ihm ein Projekt „Angeleitete Forschung“ zu absolvieren. Bereits in diesem Projekt habe ich von ihm hervorragende und intensive Betreuung erfahren. Die Entscheidung, an die Masterarbeit bei ihm auch die Promotion anzuschließen, fiel mir folglich dank der guten Erfahrungen leicht. Besonders dankbar bin ich für die unzähligen fachlichen Diskussionen, die mir immer wieder neue Denkanstöße gegeben haben.

Ein besonderer Dank gebührt außerdem den weiteren Gutachtern dieser Dissertation, Prof. Dr. Jens Schmitt und Prof. Dr. Reinhard German, für ihre Zeit und ihr Interesse an meiner Arbeit. Ebenso danke ich Prof. Dr. Karsten Berns, der als Vorsitzender der Promotionskommission die wissenschaftliche Aussprache geleitet und mit interessanten Fragen bereichert hat.

Meinen Kollegen in der Arbeitsgruppe *Vernetzte Systeme*, Dennis Christmann, Tobias Braun, Anuschka Igel, Markus Engel, Kiran Mathews, Christopher Kohlstruck und Paulo Aragao, möchte ich für die immer angenehme freundschaftliche Zusammenarbeit und die interessanten fachlichen wie privaten Gespräche bedanken. Bei der Abwicklung von Formalitäten konnte ich zudem stets auf die kompetente Unterstützung durch unsere Sekretärin zählen, zunächst Frau Barbara Erlewein und später Frau Anja Gerber, sei es bei Dienstreiseanträgen oder Beschaffungsverfahren.

Beim Fachbereich Informatik der Technischen Universität Kaiserslautern bedanke ich mich für das verliehene Stipendium, welches mir die Promotion zu Beginn finanziell ermöglicht hat.

Bei meiner Frau Michelle möchte ich mich von ganzem Herzen dafür bedanken, mich stets ermutigt und motiviert zu haben, aber auch dafür, dass sie sich, obwohl fachfremd, stets für meine Arbeit interessiert hat. Indem ich ihr meine Arbeit in immer ausgereifteren Analogien erklärte, betrachtete ich so manches Problem aus einer anderen Perspektive und gelangte so zu neuen Ideen. Bei meinen Eltern Hella und Helmut bedanke ich mich dafür, mich stets in meinen Plänen bestärkt und mir immer den Rücken gestärkt zu haben. Meinem Bruder Hendrik danke ich dafür, dass er für mich auch während meines Studiums ein stetiger Wegbegleiter war. All meinen Freunden danke ich für schöne gemeinsame Stunden, die mir Ablenkung und Ausgleich während Studium und Promotion waren.

Für Hella, Bärchen und Äffchen.

Zusammenfassung

Drahtlose Kommunikationssysteme dringen in immer mehr Anwendungsbereiche vor. Für einige Anwendungsszenarien, wie etwa die Prozessautomatisierung in Fabriken und Industrieanlagen, ist die Zuverlässigkeit vieler drahtloser Kommunikationssysteme wie IEEE 802.11 (WLAN) oder Bluetooth aber noch unzureichend. Daher wurden für diese Anwendungsbereiche spezielle Kommunikationssysteme wie WirelessHART oder ISA 100.11a entwickelt. Diese basieren meist auf Time Division Multiple Access (TDMA) und erreichen durch exklusive Reservierungen deterministische Zuverlässigkeit, falls kein anderes Kommunikationssystem die genutzten Kanäle stört.

Diese Arbeit behandelt geeignete Protokolle und Algorithmen, um die Zuverlässigkeit drahtloser Kommunikationssysteme zu verbessern. Im ersten Teil der Arbeit werden Verfahren für TDMA-basierte Kommunikationssysteme betrachtet. Basierend auf IEEE 802.15.4 werden mehrere Funktionalitäten für ProNet 4.0, einem an der Arbeitsgruppe Vernetzte Systeme der TU Kaiserslautern entwickelten Kommunikations-Stack für Industrie 4.0, entworfen und auf Imote 2 Sensorknoten implementiert. Zuverlässige Kommunikation bedarf Kenntnis von sowohl der Kommunikationstopologie, über die Knoten miteinander kommunizieren können, als auch der Interferenztopologie, die angibt, wie Knoten sich gegenseitig stören können. Dazu stellt die Arbeit mit dem Automatic Topology Discovery Protocol (ATDP) ein Verfahren zur automatischen Topologieerkennung vor. Anschließend wird QoS Multicast Routing betrachtet und mit dem QoS Multicast Routing Protocol (QMRP) ein Verfahren für partiell mobile Netzwerke entwickelt. Weiterhin wird mit ProMid eine Kommunikations-Middleware beschrieben, die ein hohes Abstraktionslevel aufweist und die darunter liegenden Schichten steuert. Die dienstorientierte Architektur nutzt eine verteilte Service Registry, wobei die Auswahl der Registry-Knoten anhand eines dafür entwickelten Clustering-Algorithmus erfolgt. Das Heterogeneous Network Clustering (HNC) genannte Verfahren berücksichtigt ein heterogenes Netzwerkmodell mit Knoten, die Clusterhead bzw. Gateway werden müssen, können bzw. nicht dürfen.

Der zweite Teil der Arbeit behandelt Protokolle und Algorithmen für zuverlässige wettbewerbsbasierte Kommunikationssysteme. Die in diesem Kapitel vorgestellten Verfahren sind in einem auf WLAN basierenden Kommunikations-Stack implementiert und evaluiert worden. Zunächst wird ein Verfahren für die Topologieerkennung in WLAN-Netzwerken vorgestellt. Anschließend wird ein auf dem Token Bucket-Mechanismus basierendes Verfahren zur Verkehrskontrolle entwickelt. Daraus wird mit der Unusable Wasted Bandwidth Ratio eine Metrik abgeleitet, die es erlaubt, die Auslastung des Mediums abzuschätzen. Aufbauend auf dem Verfahren zur Verkehrskontrolle wird eine kooperative faire Bandbreitenskalierung für WLAN vorgestellt. Das Verfahren verteilt die Bandbreite fair unter den internen Knoten unter Berücksichtigung der Quality of Service (QoS) Anforderungen. Dabei reagiert es dynamisch auf Änderungen des externen Verkehrs und verhindert so Überlastsituationen. Letztlich wird ein Clustering-Protokoll vorgestellt, welches durch das Anwendungsszenario der Überwachung von Güterzügen motiviert ist und Linientopologien bildet sowie dynamisch repariert. Das auf Bluetooth LE aufbauende Verfahren dient dazu, Energie einzusparen, und wurde in einer Kooperation mit der Bosch Engineering GmbH entwickelt.

Inhaltsverzeichnis

1. KAPITEL

Einführung

Drahtlose Netzwerke finden in immer mehr Anwendungsbereichen Verwendung und ersetzen oftmals drahtgebundene Netzwerke. Die Geschwindigkeit von drahtlosen Netzwerken wurde in den letzten Jahren enorm gesteigert, sodass drahtlose Netzwerke wie IEEE 802.11 (WLAN) mit den Erweiterungen IEEE 802.11ac und IEEE 802.11ad mittlerweile theoretische Datenraten von mehr als einem Gigabit pro Sekunde bieten. Bei drahtgebundenen Computernetzwerken sind Bandbreiten von einem Gigabit pro Sekunde immer noch weit verbreitet. Damit haben die drahtlosen Netzwerke mit den drahtgebundenen Netzwerken aufgeschlossen, was die theoretische Bandbreite angeht. Im Bereich der Unterhaltungselektronik werden daher mittlerweile die meisten Geräte mit drahtlosen Schnittstellen wie WLAN oder Bluetooth ausgestattet. Nicht nur Laptops und Smartphones, sondern auch smarte Fernsehgeräte, Spielekonsolen, Lautsprecher, Armbanduhren oder Brillen nutzen meist drahtlose Kommunikationssysteme für die Vernetzung untereinander oder mit dem Internet.

Für andere Anwendungsbereiche mangelt es vielen drahtlosen Kommunikationssystemen aber noch an der notwendigen Zuverlässigkeit. Ein solches Anwendungsszenario ist die Prozessautomatisierung in Fabriken und anderen Industrieanlagen. Für die automatische Regelung einer Produktionsanlage ist die Kommunikation mittels Kommunikationssystemen wie WLAN oder Bluetooth i.d.R. zu unzuverlässig. Daher wurden für derartige Anwendungsszenarien spezielle drahtlose Kommunikationssysteme entwickelt, wie beispielsweise WirelessHART oder ISA 100.11a. Diese erreichen eine höhere Zuverlässigkeit, indem sie den Zugriff auf das drahtlose Medium nicht wie bei WLAN durch offenen Wettbewerb organisieren, sondern exklusiv Zeitslots vergeben, also auf Time Division Multiple Access (TDMA) basieren. Der Vorteil dieser TDMA-basierten Kommunikationssysteme ist, dass deterministiche Kommunikation ermöglicht wird.

Im ersten Teil dieser Arbeit werden Protokolle und Algorithmen für TDMA-basierte Kommunikationssysteme vorgestellt. Dabei werden Verfahren zur Topologieerkennung, zum Multicast Routing und Clustering sowie Middleware-Schichten behandelt. Die entwickelten Verfahren flossen in ProNet 4.0 ein, einem Kommunikationssystem mit vollständigem Protokoll-Stack für die Prozessautomatisierung.

Genau wie WirelessHART und ISA 100.11a erreicht ProNet 4.0 durch TDMA deterministische Zuverlässigkeit. Bei TDMA werden Zeitslots exklusiv den Knoten (Geräten) des Netzwerks zugewiesen, sodass sichergestellt ist, dass maximal ein Knoten zeitgleich sendet und so keine Kollisionen auftreten können. Eine Schwäche dieses Ansatzes ist eine wichtige Voraussetzung: Das

exklusive Reservieren von Zeitslots kann Kollisionen nur dann zuverlässig verhindern, wenn keine anderen Systeme auf den verwendeten Frequenzen senden, die sich nicht an die Reservierungen halten. Allerdings verwenden die meisten drahtlosen Kommunikationssysteme das unlizenzierte ISM-Band um 2,4 GHz. Auch WirelessHART, ISA 100.11a und ProNet 4.0 basieren auf IEEE 802.15.4 und nutzen dessen Kanäle um 2,4 GHz. Da dieses Band auch von WLAN und Bluetooth verwendet wird, die sich nicht an die Reservierungen anderer Kommunikationssysteme halten, kann es leicht zu Interferenzen kommen. Somit ist die Kommunikation trotz TDMA nicht mehr deterministisch. Mit Channel Blacklisting lassen sich in Kommunikationssystemen wie WirelessHART jene Kanäle deaktivieren, die von anderen Kommunikationssystemen wie WLAN gestört werden. In Industrieanlagen konnte man so bisher jene Kanäle, die beispielsweise das WirelessHART-Netzwerk nutzt, und jene, die WLAN Access Points nutzen, überlappungsfrei konfigurieren, sodass sie sich nicht gegenseitig stören. Heute gibt es aber immer mehr tragbare Geräte mit WLAN und Bluetooth. Viele Menschen tragen ein Smartphone mit sich, das einen WLAN Access Point auf jedem beliebigen Kanal eröffnen kann. Mit lediglich vier Smartphones kann man alle 16 Kanäle stören, die IEEE 802.15.4 auf 2,4 GHz anbietet. TDMA-basierte Kommunikationssysteme unterstützten daher oft auch Channel Hopping, d.h. sie wechseln den benutzten Kanal häufig. Damit wird die Wahrscheinlichkeit von wiederholten Störungen verringert, was zwar die statistische Zuverlässigkeit verbessern kann, aber kein deterministisches Verhalten garantiert.

Im zweiten Teil der Arbeit wird daher eine zuverlässige Alternative zu TDMA-basierten Kommunikationssystemen betrachtet. Wenn man davon ausgeht, dass TDMA-basierte Kommunikationssysteme in der Praxis ohnehin oft nur statistische Zuverlässigkeit bieten können, so kann man auch versuchen, auf Basis von wettbewerbsbasierten Kommunikationssystemen eine vergleichbare statistische Zuverlässigkeit zu erreichen. Der zweiten Teil der Arbeit beschäftigt sich daher mit Protokollen und Algorithmen, welche zuverlässige Kommunikation basierend auf wettbewerbsbasierten Kommunikationssystemen wie WLAN oder Bluetooth erreichen sollen. Dabei wird ein Verfahren zur Topologieerkennung und ein auf dem Token Bucket-Mechanismus basierendes Verfahren zur Verkehrskontrolle vorgeschlagen. Damit die Verkehrskontrolle auf Änderungen des externen Verkehrs automatisch reagieren kann, wird sie um ein kooperatives Verfahren zur dynamischen Bandbreitenskalierung erweitert.

2. KAPITEL

Grundlagen

In diesem Kapitel werden einige Grundlagen erläutert, die für das Verständnis der Arbeit hilfreich sind. Die Arbeit ist eingeteilt in Verfahren für TDMA-basierte drahtlose Kommunikationssysteme in Kapitel 3 und wettbewerbsbasierte drahtlose Kommunikationssysteme in Kapitel 4. Dabei handelt es sich um zwei unterschiedliche Verfahren, den Zugriff auf ein drahtloses Medium zu regeln. Kapitel 2.1 geht daher kurz auf diese und weitere Zugriffsverfahren ein und erläutert die Unterschiede. Anschließend beschreibt Kapitel 2.2 ein Anwendungsszenario, das viele der in dieser Arbeit beschriebenen Verfahren motiviert hat.

2.1 MAC Zugriffsverfahren

Drahtlose Kommunikationssysteme übertragen Daten über elektromagnetische Wellen. Dabei wird das drahtlose Medium zwischen den Kommunikationspartnern geteilt. Wenn sich elektromagnetische Wellen überlagern, kommt es zur Interferenz zwischen den Wellen. Dies bedeutet, dass die von einem Kommunikationssystem gesendeten Rahmen mit Rahmen kollidieren können, die das gleiche oder ein anderes Kommunikationssystem sendet. Eine Rahmenkollision führt dazu, dass die zu übertragenden Rahmen verloren gehen oder verfälscht werden können. Die von zwei oder mehr Stationen (Geräten) gesendeten Rahmen kollidieren, wenn alle der folgenden Bedingungen erfüllt sind:

- Die Übertragung erfolgt zeitgleich bzw. *zeitlich* überlappend.
- Die Übertragung erfolgt *räumlich* nah genug beieinander. Genauer: Die Reichweite der Sender ist groß genug, dass ein Empfänger mehr als einen Sender und damit ein durch Interferenzen verändertes Signal empfängt.
- Die Übertragung erfolgt auf der gleichen *Frequenz* bzw. auf einem überlappenden Frequenzband.

Ein Zugriffsverfahren für das Medium, auch MAC-Verfahren (engl.: Medium Access Control) genannt, versucht Rahmenkollision zu vermeiden, indem es sicherstellt, dass mindestens eine der Bedingungen nicht erfüllt ist. Die wichtigsten existierenden Verfahren lassen sich in fünf Gruppen einteilen, die im folgenden genauer erklärt werden.

Time Division Multiple Access (TDMA) Bei TDMA wird der Zugriff zeitlich geregelt, d.h. Kollisionen werden verhindert, indem sichergestellt wird, dass nie zwei Stationen zeitgleich senden.

Dazu wird die Zeit i.d.R. in Zeitslots aufgeteilt. Oft ist die Aufteilung hierarchisch, beispielsweise in Superslots, Makroslots und Mikroslots aufgeteilt, wobei ein Superslot aus Makroslots und dieser wiederum aus Mikroslots besteht. Mikroslots werden Stationen zugeordnet, die damit das Recht erhalten, während dieser Zeit zu senden. Die Slots werden i.d.R. periodisch wiederholt, sodass Reservierungen periodisch wiederkehren. Um Kollisionen zuverlässig zu verhindern, müssen die Reservierungen exklusiv sein. In der Praxis können auf nicht lizenzierten Frequenzbereichen andere Kommunikationssysteme, die sich nicht an die Reservierungen halten, die Zuverlässigkeit erheblich einschränken. Außerdem muss eine weitere wichtige Voraussetzung erfüllt sein: Die sendenden Stationen müssen untereinander mit hinreichender Genauigkeit Zeit- oder Tick-synchronisiert sein. Da physikalische Uhren nie perfekt sind, treten ohne entsprechende Maßnahmen schnell Abweichungen zwischen den Uhren auf. TDMA-basierte Kommunikationssysteme erfordern daher immer ein Verfahren zur Tick- oder Zeitsynchronisation. Da die durch diese Verfahren erzielte Synchronisation nicht perfekt ist, wird zwischen den Übertragungen unterschiedlicher Sender i.d.R. ein Schutzintervall eingeplant.

Der größte Vorteil TDMA-basierter Kommunikationssysteme ist, dass der garantierte Zugriff auf das Medium zu definierten Zeitpunkten eine harte Echtzeitfähigkeit ermöglicht. Bei periodischen Nachrichten führt die Zeit-getriggerte Kommunikation darüber hinaus zu einer guten Bandbreitennutzung. Sporadische Nachrichten werden dagegen nicht gut unterstützt, da sie immer bis zum nächsten reservierten Zeitslot verzögert werden. Gerade bei Nachrichten mit kurzen Reaktionszeiten bedeutet dies, dass für den schlechtesten Fall sehr viele Zeitslots reserviert werden müssen, von denen die meisten ungenutzt bleiben. Daher ist die Bandbreitennutzung bei sporadischen Nachrichten schlecht. Außerdem entsteht oft Overhead zur Zeitsynchronisation und durch Schutzintervalle. In der Praxis können auf nicht lizenzierten Frequenzbereichen andere Kommunikationssysteme die Zuverlässigkeit von TDMA stark einschränken.

Space Division Multiple Access (SDMA) SDMA verteilt den Zugriff auf das Medium räumlich. Zwei Stationen können zeitgleich senden, ohne dass es zu einer Kollision kommt, wenn sie soweit auseinander liegen, dass keiner der Empfänger in Reichweite von beiden Sendern liegt. Zur Vereinfachung werden die Stationen oft räumlich in Zellen eingeteilt, wie etwa beim Mobilfunkstandard GSM. Wenn sich die Stationen räumlich bewegen können, wird das Verfahren aufwendiger und erfordert ggf. ein Handover-Verfahren, bei der eine Station von einer Zelle in eine andere wechselt. Die Verwendung von Richtfunktechnik ermöglicht eine verbesserte Kontrolle über die Reichweite der Sender und ist damit besonders gut für die Verwendung von SDMA geeignet. Oft wird SDMA auch mit anderen Zugriffsverfahren wie TDMA und FDMA kombiniert. So ist es ineffizient, in einem großen Netzwerk einen Zeitslot netzweit nur einer Station zuzuordnen. Die Kombination von TDMA mit SDMA erlaubt es, Stationen gleichzeitig senden zu lassen, wenn sie weit genug auseinander liegen. Auf diese Weise kann SDMA die Gesamtbandbreite eines TDMA-basierten Netzwerkes steigern.

Frequency Division Multiple Access (FDMA) Bei FDMA wird der Medienzugriff über unterschiedliche Frequenzen verteilt. Das vom Kommunikationssystem genutzte Frequenzband wird dabei meist in eine Menge Kanäle aufgeteilt. Auf nicht überlappenden Kanälen können zeitgleich Übertragungen stattfinden, ohne dass es zu Kollisionen kommt. Über lizenzierte Frequenzen kann FDMA besonders zuverlässige Kommunikation ermöglichen, da Störungen durch andere Kommunikationssysteme gesetzlich verboten sind. Allerdings sind die hohen Lizenzgebühren für lizenzierte Frequenzen nur für wenige Anwender akzeptabel. Oft wird FDMA auch

mit anderen Verfahren wie TDMA und SDMA kombiniert. So wechseln einige Kommunikationssysteme den Kanal nach einem Zeitplan (Channel Hopping) und verbinden so TDMA mit FDMA. Durch das Wechseln der Kanäle wird vermieden, dass die Kommunikation immer über einen Kanal stattfindet, der besonders stark gestört wird. So kann die Zuverlässigkeit zwar in vielen Fällen verbessert werden, eine deterministische Zuverlässigkeit ist dennoch nur gegeben, wenn alle benutzten Kanäle frei von externen Störungen sind.

Eine Weiterentwicklung von FDMA ist OFDMA (Orthogonal Frequency Division Multiple Access). Dabei werden jeder Station statt einem Frequenzbereich viele kleine Unterträger (engl: Subcarrier) zugeordnet. Durch eine orthogonale Anordnung der genutzten Frequenzen führen Überlagerungen mit den Signalen der anderen Stationen nicht zu zerstörenden Kollisionen. Das Verfahren kann damit das Frequenzspektrum deutlich effizienter nutzen als FDMA. Außerdem steigt die Zuverlässigkeit, da schmalbandige Störungen auf einzelnen Unterträgern meist durch Vorwärtsfehlerkorrektur behoben werden können [Sta16].

Code Division Multiple Access (CDMA) Bei CDMA-basierten Kommunikationssystemen können mehrere Stationen in Reichweite voneinander zeitgleich auf dem gleichen Frequenzbereich senden, ohne dass es zu zerstörenden Kollisionen kommt. Dabei wird ein breiterer Frequenzbereich genutzt und die Signale mit speziellen Spreizcodes codiert. Dadurch, dass die Codefolgen der zeitgleich sendenden Stationen (annähernd) orthogonal zueinander sind, wird das Trennen der Datenströme beim Empfänger ermöglicht. Mit längeren Codes können mehr Nutzer zeitgleich unterstützt werden, allerdings reduziert sich dadurch die Nutzdatenrate. Die Zuverlässigkeit von CDMA-basierten Kommunikationssystemen hängt von der Güte der verwendeten Codes ab, da nicht optimal orthogonale Codes zu Störungen führen, die sich ähnlich wie Rauschen äußern. Ist der störende Sender sehr nah und der gewünschte Sender sehr weit entfernt, kann es passieren, dass dieses Rauschen das Signal des gewünschten Senders zu stark überlagert und keine Kommunikation mehr möglich ist [Sta16]. CDMA macht den Austausch der verwendeten Codes notwendig, was Overhead beim Verbindungsaufbau bedeutet.

Carrier-sense Multiple Access (CSMA) Bei CSMA wird der Zugriff auf das drahtlose Medium wettbewerbsbasiert geregelt. Eine Station, die das Medium zum Senden benutzen möchte, hört bei CSMA zunächst das Medium ab, um festzustellen, ob es bereits durch eine andere Station belegt ist (Carrier Sensing). Gesendet werden darf nur, wenn das Medium als frei erkannt wurde. Wird das Medium als belegt erkannt, so muss gewartet werden, bis das Medium wieder frei ist. Damit nicht alle wartenden Stationen zeitgleich zu senden beginnen, sobald das Medium wieder frei ist, wird meist CSMA mit der Collision Avoidance-Erweiterung (CSMA/CA) eingesetzt. Dabei handelt es sich um ein randomisiertes Backoff-Verfahren, bei dem die Stationen eine zufällige Zeit warten, bevor sie beginnen zu senden. Die Station mit dem kleinsten Backoff beginnt als erste zu senden und gewinnt damit den Wettbewerb um das Medium, die anderen Stationen müssen erneut warten.

Mit CSMA/CA kann die Wahrscheinlichkeit, dass Kollisionen auftreten, reduziert werden. Allerdings kann das Verfahren Kollisionen nicht gänzlich verhindern. Einerseits haben viele Transceiver eine Umschaltzeit zwischen Empfangs- und Sendemodus, während der das Medium nicht abgehört werden kann. Beginnt eine andere Station während dieser Umschaltzeit zu senden, wird dies nicht erkannt und es kommt zur Kollision. Die randomisierten Backoff-Verfahren können darüber hinaus nicht ausschließen, dass zwei Stationen zufällig die gleiche Wartezeit ziehen und dann exakt zeitgleich zu senden beginnen.

Ein weiteres Problem entsteht, wenn die Reichweite von zwei Sendern v_{s1} und v_{s2} bis zu einem Empfänger v_r reichen, aber nicht bis zum jeweils anderen Sender. Eine bereits laufende Übertragung von v_{s1} zu v_r wird von v_{s2} nicht erkannt, da die Reichweite von v_{s1} nicht zu v_{s2} reicht und dessen Carrier Sense die Übertragung daher nicht erkennt. Somit beginnt v_{s2} ebenfalls zu senden und es kommt bei v_r zu einer Kollision. Dieses Problem ist als *Hidden Station-Problem* bekannt. Das Problem kann unter Einsatz des RTS/CTS-Mechanismus entschärft, aber nicht gelöst werden.

Bei drahtgebundenen Kommunikationssystemen kann eine auftretende Kollision einfach erkannt werden, indem das auf dem Medium anliegende Signal mit dem gesendeten verglichen wird. Wird eine Kollision erkannt, können die Übertragungen schnell abgebrochen werden und ein neuer Wettbewerb gestartet werden. Diese Collision Detection (CSMA/CD) genannte Erweiterung von CSMA kann bei drahtlosen Kommunikationssystemen aber meist nicht angewendet werden, da diese meist während des Sendens das Medium nicht abhören können[1]. Daher werden kollidierende Übertragungen vollständig durchgeführt und die Kollision erst durch ausbleibende Bestätigung (engl.: Acknowledgement) erkannt. Kollisionen reduzieren daher bei drahtlosen CSMA/CA Kommunikationssystemen die nutzbare Bandbreite und erhöhen Verzögerungen erheblich.

Der Ereignis-getriggerte Zugriff auf das Medium bei CSMA erlaubt prinzipiell einen kurzfristigeren Zugriff als Zeit-getriggerte Kommunikation wie bei TDMA. Dies ist für sporadische Nachrichten besser geeignet, da keine Bandbreite für nicht genutzte Reservierungen verschwendet wird. Allerdings ist die Verzögerung beim Mediumzugriff bei CSMA/CA-basierten Systemen unbegrenzt, da eine Station theoretisch immer wieder den zufallsbasierten Wettbewerb gegen die anderen Stationen verlieren kann.

Die Zuverlässigkeit von CSMA/CA-basierten Kommunikationssystemen hängt im wesentlichen davon ab, wie viele Stationen um das Medium konkurrieren. Umso mehr Stationen konkurrieren, umso wahrscheinlicher werden Kollisionen. Die zu Nachrichtenverlust führenden Kollisionen lösen meist Neuübertragungen aus, was die Auslastung des Mediums zusätzlich erhöht.

2.2 Anwendungsszenario: Drahtlose Kommunikation für die Prozessautomatisierung in Fabriken und Industrieanlagen

Um Produktionskosten senken zu können, sucht die Industrie stetig nach Möglichkeiten, mit denen die Produktionsabläufe optimiert werden können. Hierzu werden die Produktionsprozesse oft immer stärker automatisiert, da so Lohnkosten eingespart werden können. Allerdings erfordert ein hoch automatisierter Produktionsprozess auch eine umfassende Überwachung der Produktionsabläufe. Zuvor hatten Menschen in der Produktion gearbeitet und konnten Probleme oder Gefahren erkennen und ggf. eingreifen. Die automatisierten Produktionsabläufe müssen entweder weiterhin von Menschen überwacht werden, oder aber die Überwachung ebenfalls automatisiert werden. Hierzu werden *Sensoren* installiert, welche kontinuierlich Messwerte erfassen, wie beispielsweise Temperatur, Druck, Gaskonzentration oder Helligkeit.

Bei einer reinen *Überwachung* der Produktion werden diese Daten von den Sensorknoten beispielsweise an einen zentralen Leitstand übermittelt. Werden die dort eintreffenden Werte von

[1] Drahtlose Single Channel Full-Duplex Kommunikationssysteme ermöglichen mittlerweile CSMA/CD [Son+16].

Mitarbeitern nicht nur überwacht, sondern auch genutzt, um Parameter der Produktion anzupassen, so spricht man von *Steuerung* (engl.: open-loop control). Im Falle einer *Regelung* (engl.: closed-loop control) werden die gemessenen Sensorwerte dagegen direkt genutzt, um die Parameter der Produktion automatisch anzupassen. Der Produktionsprozess wird dabei basierend auf den gemessenen Werten kontinuierlich so geregelt, dass er sich wie gewünscht verhält. Dazu werden die Sensorwerte vom Sensorknoten an einen *Regler* übertragen, der daraus Steuerbefehle ableitet und diese an die *Aktuatoren* sendet, welche daraufhin in den Produktionsprozess eingreifen. Dies kann wiederum die von den Sensoren gemessenen Werte beeinflussen, womit der Regelkreis geschlossen ist. Die Sensorwerte und Steuerbefehle werden über ein Netzwerk bzw. einen Feldbus übertragen. Hierzu kamen zunächst kabelgebundene Lösungen wie der Highway Addressable Remote Transducer (HART) Feldbus zum Einsatz. Auf Grund der besonderen Bedingungen in Produktionsanlagen müssen Kabel oft sehr robust sein und sind damit teuer [AGB11; MTA05]. Außerdem sind Kabel und insbesondere Steckverbinder eine häufige Fehlerquelle [Ene02]. Drahtlose Kommunikationssysteme kommen dagegen ohne teure Kabel und fehleranfällige Steckverbindungen aus. Sie können flexibel umpositioniert werden und bieten damit die für sogenannte *wandlungsfähige Produktionssysteme* [Nyh10] erforderliche Flexibilität. Darüber hinaus können drahtlose Systeme einfacher auf beweglichen Geräten im Produktionsprozess platziert werden, wie etwa auf Robotern, oder gar auf dem Produkt selbst.

Den Vorteilen drahtloser Kommunikationssysteme stehen allerdings auch Nachteile gegenüber. Die in in der Prozessautomatisierung notwendige Zuverlässigkeit der Übertragungen kann von den ursprünglich für Unterhaltungssysteme entwickelten drahtlosen Kommunikationssystemen wie Bluetooth [Blu16] oder IEEE 802.11 (WLAN) [IEE16] meist nicht erreicht und vor allem nicht garantiert werden. Aus diesem Grund wurden für Anwendungen wie die Prozessautomatisierung spezialisierte Standards wie WirelessHART [Eur10] und ISA 100.11a [IEC12] entwickelt. Diese setzen TDMA-basierte Zugriffsverfahren ein, um Kollisionen der Übertragungen zu verhindern und damit die Zuverlässigkeit zu verbessern. Durch die exklusive Reservierung von Zeitslots lassen sich Kollisionen zwischen Übertragungen innerhalb des Netzwerkes zuverlässig verhindern. Senden andere Geräte wie beispielsweise WLAN Access Points, Smartphones oder DECT-Telefone auf einem überlappenden Frequenzband, führt dies allerdings zu Kollisionen. Sowohl WirelessHART als auch ISA 100.11a basieren auf dem Physical Layer von IEEE 802.15.4 [IEE11a] und nutzen dessen Kanäle um 2,45 GHz im ISM-Band. Da das gleiche Band auch z.B. von WLAN, Bluetooth und DECT verwendet wird, sind Kollisionen mit anderen Netzwerken möglich. In Produktionsstätten lassen sich andere Netzwerke wie beispielsweise die installierten WLAN Access Points gezielt auf Kanäle legen, die nicht mit den in WirelessHART bzw. ISA 100.11a genutzten Kanälen überlappen. Heutzutage können allerdings beispielsweise Smartphones, die Mitarbeiter oder Besucher mit in die Produktionsstätte bringen, auf jedem beliebigen Kanal einen Access Point eröffnen. Sowohl WirelessHART als auch ISA 100.11a bieten als Lösung das Blacklisting von jenen Kanälen an, die von anderen Netzwerken benutzt werden. Diese Lösung funktioniert allerdings nur, solange nicht zu viele Kanäle von anderen Netzwerken gestört werden. Um mit TDMA-basierten Netzwerken eine zuverlässige Übertragung garantieren zu können, ist es letztlich notwendig, sämtliche drahtlosen Netzwerke in Reichweite zu kontrollieren. Dies bedeutet, dass an der Pforte einer Produktionsanlage sämtliche Geräte abgegeben werden müssen, die auf den Frequenzen des Produktionsnetzwerks senden könnten.

3. KAPITEL

Protokolle und Algorithmen für zuverlässige drahtlose TDMA-basierte Kommunikationssysteme

Werden drahtlose Kommunikationssysteme in Anwendungen wie der Prozessautomatisierung eingesetzt (s. Kapitel 2.2), so bedeutet dies, dass eine hohe Zuverlässigkeit des Kommunikationssystems erforderlich ist. Wie in Kapitel 2.1 beschrieben, erlaubt es das TDMA Zugriffsverfahren, Zeitslots exklusiv zu reservieren. Dies verhindert Kollisionen mit anderen Übertragungen des Netzwerks und sorgt damit für eine deterministische Zuverlässigkeit, sofern der Kanal nicht durch externe Knoten gestört wird. Auf Grund der deterministischen Eigenschaften sind TDMA-basierte Kommunikationssysteme besonders geeignet, wenn eine hohe Zuverlässigkeit gefordert ist. In diesem Kapitel werden Protokolle und Algorithmen betrachtet, die speziell für TDMA-basierte drahtlose Kommunikationssysteme mit hohen Anforderungen an die Zuverlässigkeit entwickelt wurden.
Zunächst beschreibt Kapitel 3.1 *ProNet 4.0*, ein drahtloses Kommunikationssystem für die Prozessautomatisierung in Fabriken, für welches die in diesem Kapitel vorgestellten Algorithmen und Protokolle implementiert wurden, und geht auf die dabei eingesetzte Hardware ein. Anschließend werden Protokolle und Algorithmen für verschiedene Aufgaben vorgestellt. Als erstes Verfahren stellt Kapitel 3.2 das in [KCG15] publizierte Protokoll zur Topologieerkennung in drahtlosen TDMA-Netzwerken vor. Daraufhin wird in Kapitel 3.3 ein Verfahren zum Multicast Routing präsentiert, welches in [Geb+15] publiziert wurde. In Kapitel 3.4 wird auf Verfahren zum Clustern von drahtlosen Netzwerken eingegangen und das in [KCG16] publizierte Verfahren näher betrachtet, welches speziell für das Verteilen einer Service Registry einer dienstorientierten Middleware konzipiert wurde. Letztlich beschreibt Kapitel 3.5 eine Middleware für Kommunikationssysteme, die von den zuvor vorgestellten Verfahren profitiert.

3.1 ProNet 4.0

An der Arbeitsgruppe Vernetzte Systeme der TU Kaiserslautern entstand *ProNet 4.0* (Production Network 4.0), ein realzeitfähiges Kommunikationssystem mit vollständigem Protokoll-Stack für die Prozessautomatisierung in Fabriken. Es zeichnet sich durch einen großen Funktionsumfang, Realzeitfähigkeit, eine Service-orientierte Schnittstelle sowie ein hohes Maß an Flexibilität

aus und ist damit passend für die Anforderungen der *Industrie 4.0* [Got14]. Das Kommunikationssystem ist TDMA-basiert und erreicht deterministische Echtzeitfähigkeit durch exklusive Reservierungen bzw. deterministischen Wettbewerb. Es basiert auf dem an der gleichen Arbeitsgruppe entwickelten BiPS (Black-burst integrated Protocol Stack) Framework [Eng+19].

Dieses Kapitel gibt zunächst einen Überblick über die Architektur von ProNet 4.0 und zeigt auf, wie die in den nächsten Kapiteln vorgestellten Verfahren in ProNet 4.0 verwendet werden. Anschließend stellt Kapitel 3.1.2 die Hardware vor, für die ProNet 4.0 entwickelt wurde und die zum Testen der hier entwickelten Ansätze verwendet wurde.

3.1.1 Architektur

Die Architektur von ProNet 4.0 ist in Abb. 3.1 skizziert. Das Kommunikationssystem basiert auf dem Physical Layer von IEEE 802.15.4. Darüber bietet ProNet 4.0 einen vollständigen Protokoll-Stack, der für die Prozessautomatisierung in Fabriken ausgelegt ist und es so besonders einfach macht, Anwendungen für diesen Bereich zu entwickeln. Da ProNet 4.0 den Medienzugriff TDMA-basiert regelt, ist eine Zeit- bzw. Ticksynchronisation unabdingbar. Dazu ist das Modul *ProSync* zuständig, welches eine Implementierung des *Black Burst Synchronization*-Protokolls (BBS) [GK11] ist. Dieses deterministische Protokoll kann Multihop-Netzwerke netzwerkweit Tick-synchronisieren, wobei der entstehende Tick-Offset gering und nach oben beschränkt ist. Außerdem unterstützt es Zeitsynchronisation, beispielsweise um Messdaten mit Zeitstempeln zu versehen.

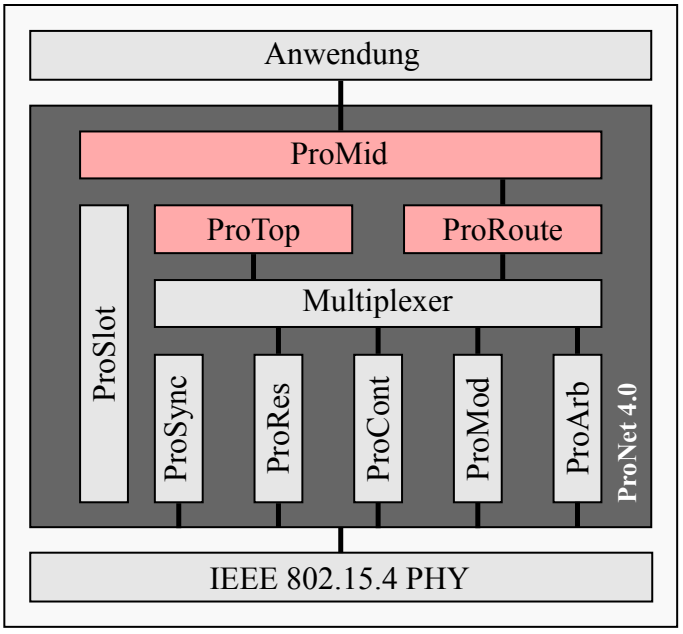


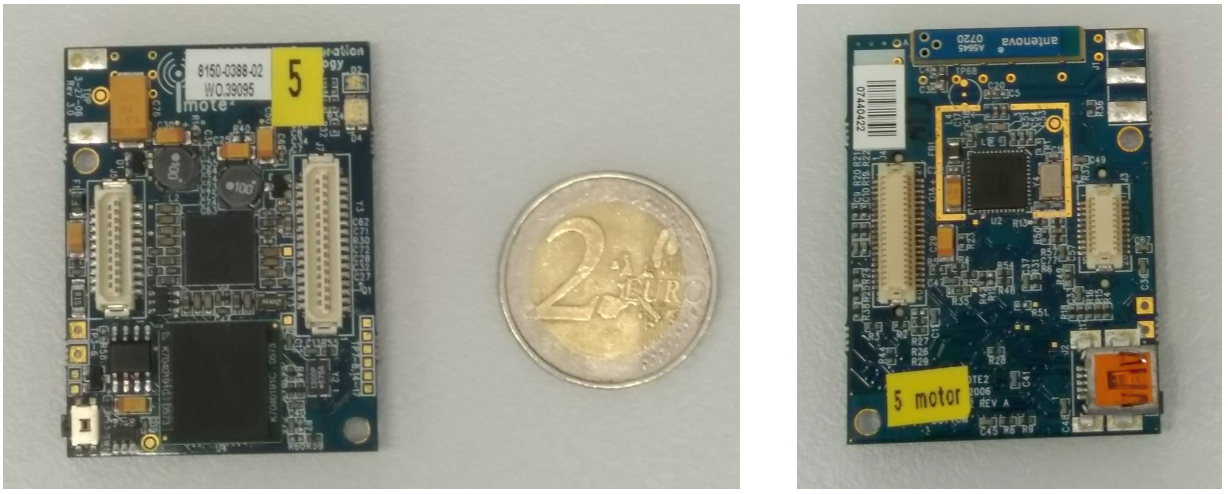
Abbildung 3.1: Architektur von ProNet 4.0

Weiterhin unterstützt ProNet unterschiedliche MAC Zugriffsverfahren, nämlich exklusive Reservierungen (*ProRes*), wettbewerbsbasierten Zugriff (*ProCont*) sowie prioritätsgesteuerten Zugriff (*ProMod*). Mit *ProArb* steht darüber hinaus eine Implementierung des *Arbitrating Value Transfer Protokolls* (AVTP) [CGR12] zur Verfügung, welches netzwerkweite deterministische Arbitrierung und Wertübermittlung bereitstellt. Welches der Zugriffsverfahren wann zum Einsatz kommt, regelt das Slotting-Verfahren *ProSlot* über einen Multiplexer.

Die in Abb. 3.1 rot hervorgehobenen Module beherbergen Implementierungen der in diesem Kapitel vorgestellten Algorithmen und Protokolle. Das in Kapitel 3.2 beschriebene Verfahren zur Topologieerkennung wurde für ProNet 4.0 unter dem Namen *ProTop* implementiert. Weiterhin wurde das in Kapitel 3.3 beschriebene Multicast Routing-Verfahren unter dem Namen *ProRoute* implementiert. *ProMid* stellt die Service-orientierte Middleware dar, welche die Schnittstelle der Anwendung zu ProNet 4.0 ist und in Kapitel 3.5 genauer erläutert wird. ProMid verwendet ein Clustering-Verfahren zur Verteilung der Service Registry, welches in Kapitel 3.4 vorgestellt wird.

3.1.2 Hardware

Die entwickelten Protokolle und Algorithmen sollen nicht nur durch Simulation evaluiert und getestet werden, sondern ihre Wirksamkeit in der Praxis unter Beweis stellen. Als Hardware-Plattform zum Implementieren der Protokolle und Algorithmen wurden Imote 2 [MEM] Sensorknoten ausgewählt. Diese sind auch im Einsatzszenario der Prozessautomatisierung eine denkbare Hardware-Plattform. Sie haben eine Größe von nur 48 mm Länge und 35 mm Breite und sind damit nicht viel größer als eine Zweieuromünze (s. Abb. 3.2). Das Board wird von einem PXA271-Prozessor angetrieben und beherbergt einen IEEE 802.15.4 [IEE03] kompatiblen CC2420 Transceiver [Chi07] inkl. Antenne zur drahtlosen Kommunikation. Es kann mit einem Sensorboard erweitert werden und verfügt dann über diverse Sensoren, wie beispielsweise einen Temperatursensor. Für die Entwicklung zuverlässiger Protokolle ist es besonders geeignet, da es die volle Kontrolle über die Hardware ermöglicht. Die in diesem Kapitel vorgestellten Protokolle sind für TDMA-basierte Kommunikationssysteme entworfen und erfordern daher eine Ticksynchronisation. Für die Imote 2-Plattform existierte bereits eine optimierte Implementierung des Black Burst Synchronization-Protokolls (BBS) [GK11] zur Ticksynchronisation sowie weiterer Black Burst-basierter Protokolle [Eng13]. Damit bietet sie die optimale Grundlage, die hier beschriebenen Protokolle zu implementieren und zu testen.



(a) Oberseite mit PXA271-Prozessor (unten) und einer Zweieuromünze als Größenvergleich

(b) Unterseite mit CC2420-Transceiver (Mitte oben) und Antenne (oben)

Abbildung 3.2: Ein Imote 2 Sensorknoten mit IEEE 802.15.4 kompatiblem CC2420 Transceiver

3.2 Topologieerkennung

Drahtlose Kommunikationssysteme sind stets in ihrer Reichweite beschränkt. Bei WLAN oder IEEE 802.15.4 sind das typischerweise nicht mehr als 30 m, wobei die tatsächliche Reichweite von vielen Faktoren abhängt, wie etwa der Sendestärke, der verwendeten Datenrate bzw. Modulation sowie Hindernissen zwischen Sender und Empfänger. Auf Grund der beschränkten Reichweite müssen Übertragungen über längere Distanzen von mehreren Zwischenknoten weitergeleitet werden. Welche Zwischenknoten dabei genutzt werden, bestimmt ein sog. *Routing-Protokoll* (wie das in Kapitel 3.3 beschriebene). Damit das Routing-Protokoll eine effiziente Route vom Start zum Zielknoten bestimmen kann, benötigt es Informationen über die existierenden (direkten) Verbindungen zwischen den Knoten. Eine solche Verbindung, über die ein Knoten direkt an einen anderen Knoten senden kann, wird *Kommunikationslink* genannt, die Menge aller Kommunikationslinks eines Netzwerks bezeichnet man als *Kommunikationstopologie*. Die meisten Routing-Protokolle bestimmen zur Berechnung einer Route einen Teil der Kommunikationstopologie und wählen basierend darauf eine Route aus. Routing-Protokolle enthalten somit meist Funktionalität zur Erkennung der Kommunikationstopologie. Die dynamische Erkennung ist notwendig, weil die Knoten eines drahtlosen Netzwerkes im Allgemeinen mobil sind und die Topologie sich daher mit der Zeit ändern kann. Selbst in Anwendungen wie der Prozessautomatisierung in Fabriken, bei denen die meisten Knoten fest in der Produktionsstraße montiert werden, ist die manuelle Konfiguration der Topologie sehr aufwendig und fehleranfällig. Eine automatische Erkennung der Topologie ist daher in jedem Fall wünschenswert.

Neben Routing-Protokollen benötigen auch andere Protokolle und ggf. Anwendungen in drahtlosen Netzwerken Topologieinformationen. Beispielsweise teilen *Clustering-Algorithmen* (wie der in Kapitel 3.4 beschriebene) das Netzwerk in Bereiche, sog. Cluster, auf. Da sie dazu die Kommunikationstopologie benötigen, enthalten diese Protokolle i.d.R. ebenfalls Funktionalität zur Topologieerkennung.

Damit Übertragungen in einem TDMA-basierten Netzwerk zuverlässig sind, ist es wichtig, dass die Reservierungen der Zeitslots so erfolgen, dass keine Kollisionen auftreten. Am einfachsten ist es, Zeitslots exklusiv nur einem einzelnen Knoten zuzuweisen. Indem stets maximal ein Knoten zeitgleich sendet, ist ausgeschlossen, dass zwei gleichzeitige Übertragungen kollidieren. In großen Netzwerken ist es auf Grund der begrenzten Reichweite der Knoten aber möglich, dass zeitgleich mehrere Knoten senden können, ohne dass ihre Rahmen kollidieren (SDMA). In einem TDMA-basierten Netzwerk kann man daher SDMA und TDMA kombinieren, indem man Reservierungen für TDMA-Zeitslots mehrfach vergibt, solange die Übertragungen auf Grund der räumlichen Entfernung nicht kollidieren können. Die Kombination erlaubt es, das drahtlose Medium sehr viel effizienter auszunutzen. Um entscheiden zu können, ob zwei gleichzeitige Übertragungen kollidieren, reicht allerdings die Kommunikationstopologie nicht aus. Dies liegt daran, dass die Reichweite, über die ein Knoten eine andere Übertragung stören kann, in der Regel größer ist, als die Reichweite, über die er kommunizieren kann. Nur mit Hilfe der sogenannten *Interferenztopologie* kann entschieden werden, ob ein Zeitslot für zwei gleichzeitige Übertragungen reserviert werden darf oder nicht. Im Gegensatz zur Erkennung der Kommunikationstopologie ist die Erkennung der Interferenztopologie allerdings nicht einfach. Daher liegen Reservierungs-Protokollen in der Regel heuristische Entscheidungsstrategien zu Grunde, die aus der Kommunikationstopologie auf die Interferenztopologie schließen. Derartige Interferenz-Modelle basieren allerdings auf Annahmen, die in der Praxis nicht immer gelten. Um eine mög-

lichst hohe Zuverlässigkeit zu erreichen, ist es daher besser, wenn die Topologieerkennung die Interferenztopologie direkt erkennen kann.

Sendet ein Knoten, können auch Knoten die von ihm ausgehende Energie auf dem Medium wahrnehmen, die so weit von ihm entfernt sind, dass sie weder mit dem Knoten kommunizieren, noch ihn durch Interferenz stören können. Diese reinen *Sensing-Links* sind in der Praxis oft nicht relevant, da sie weder Kommunikation erlauben noch stören. Es gibt allerdings einige Protokolle (z.B. [SK96] und [GK11]), die durch eine spezielle Black Burst-Kodierung die Kommunikation über Sensing-Links möglich machen. Kommen solche Protokolle zum Einsatz, ist es auch hilfreich, wenn die *Sensing-Topologie* bekannt ist.

3.2.1 Stand der Technik

Da Topologieinformation oft für mehrere Protokolle in einem drahtlosen Kommunikationssystem von zentraler Bedeutung ist, wäre es sinnvoll, ein Protokoll zur Topologieerkennung einzusetzen, welches die erkannte Topologie den anderen Protokollen zur Verfügung stellt. Allerdings wird Topologieerkennung oft nicht als eigenständiges Problem betrachtet, sondern als Teil von anderen Problemen wie Routing oder Clustering. Daraus ergibt sich, dass es fast keine reinen Protokolle zur Topologieerkennung gibt, sondern viele drahtlose Protokolle, wie etwa Routing- oder Clustering-Protokolle, die ihre eigene Topologieerkennung enthalten und die ermittelte Informationen i.d.R. nicht untereinander austauschen.

3.2.1.1 Topologieerkennung in Routing-Protokollen

Zunächst betrachten wir einige bekannte Routing-Verfahren in Bezug auf die in ihnen enthaltenen Funktionen zur Topologieerkennung. Verfahren, die auf Distanzvektoren basieren, wie Destination-Sequenced Distance Vector (DSDV) [PB94] und darauf aufbauende Verfahren wie Ad-hoc On-demand Distance Vector (AODV) [PR99] und Babel [Chr11] erkennen die Kommunikationstopologie, tauschen die Information über vorhandene Kommunikationslinks aber lediglich in aggregierter Form aus. Mit Distanzvektoren teilt ein Knoten seinen Nachbarknoten mit, wie viele Hops er zu anderen Knoten im Netzwerk entfernt ist. Welche Links dabei benutzt werden oder welche Zwischenknoten genutzt werden, wird nicht ausgetauscht. Dies verringert den Overhead, der durch das Austauschen der Topologieinformation entsteht, da die in Form von Distanzvektoren aggregierte Information deutlich geringer ist und die Übertragung daher weniger Bandbreite benötigt.
Allerdings ergeben sich durch die Aggregation auch neue Probleme, wie etwa Routing-Schleifen, wobei auf Grund von veralteter Information eine Route zu einer Schleife wird. DSDV, AODV und Babel lösen das Problem, indem sie Distanzvektoren zusätzlich mit Sequenznummern versehen, sodass veraltete Informationen erkannt und durch neuere überschrieben werden kann. Dies löst das Problem, erzeugt aber auch zusätzlichen Overhead, da die Sequenznummern stets mit übertragen werden müssen. Ein ähnliches Problem, was sich aus der Verwendung von Distanzvektoren ergibt, ist das Count-to-Infinity Problem. Hierbei zählen zwei Knoten bis ins Unendliche, da sie denken, sie könnten über den jeweils anderen Knoten noch einen nicht mehr verbundenen Knoten erreichen. Dieses Problem entsteht nur deshalb, weil die Knoten lediglich Distanzvektoren austauschen, und aus diesen nicht schlussfolgern können, dass der jeweils andere Knoten über sie selbst zum Zielknoten routet, und daher über ihn auch keine Route mehr zum Zielknoten führt. In drahtgebundenen Netzwerken wird dieses Problem durch das

Split-Horizon with Poisoned Reverse Verfahren begrenzt: Es verbietet einem Knoten, neue Informationen über eine Route auch an den Knoten zu senden, über den er diese Route gelernt hat. Dieser Ansatz ist in drahtlosen Netzwerken nicht ohne weiteres umsetzbar, da keine getrennten Ausgangsleitungen zu den einzelnen Nachbarn bestehen, sondern immer alle Nachbarn in Reichweite eine Nachricht erhalten. Allerdings kann das Problem auch durch Sequenznummern gelöst werden, da die neuere Information die höhere Sequenznummer trägt und somit veraltete Informationen des anderen Knoten überschreibt.
Alle aggregierenden Verfahren haben darüber hinaus noch den Nachteil, dass die aggregierte Information für andere Protokolle oder Anwendungen, die Topologieinformation benötigen, ggf. nicht ausreichend ist und diese daher ihre eigene Topologieerkennung benötigen, was wiederum Overhead erzeugt.

Link State Routing-Protokolle wie das 1980 für das ARPANET entwickelte Link State Protokoll [MRR80] und OLSR [CJ03] verteilen die erkannte Kommunikationstopologie im Netzwerk, ohne sie zu aggregieren. Dabei flutet jeweils ein Knoten das Netzwerk mit der Liste seiner Nachbarknoten. Dieser Ansatz verhindert Probleme wie Routing-Schleifen und das Count-to-Infinity Problem. Allerdings entsteht dadurch, dass das Netzwerk häufig mit vielen kleinen Paketen geflutet wird, viel Overhead durch den Austausch der Topologieinformation.
Global State Routing (GSR) [CG98] basiert auch auf dem Link State Verfahren und tauscht die erkannte Kommunikationstopologie netzwerkweit aus. Im Gegensatz zu klassischen Link State Protokollen flutet es aber nicht das Netzwerk mit vielen kleinen Paketen, um die Topologie auszutauschen. Stattdessen synchronisieren die Knoten ihr Topologiewissen miteinander, indem sie regelmäßig (Teile der) ihnen bekannten Topologie an ihre Nachbarn schicken. Da so nicht jede kleine Änderung einzeln durch das Netzwerk geflutet wird, sondern viele Änderungen mit einem Paket ausgetauscht werden können, beschränkt dieser Ansatz den Overhead. Insbesondere bei Ad-Hoc-Netzwerken, in denen sich Knoten bewegen und viele Links somit zeitgleich ändern können, verhindert dies, dass das Netzwerk durch einen Burst vieler kleiner Pakete blockiert wird. Im Gegensatz zu den meisten Routing-Protokollen bringt GSR keinen eigenen Algorithmus zur Berechnung der Route mit, sondern erkennt lediglich die Topologie und tauscht sie netzwerkweit aus. So kann jeder Knoten Routen leicht lokal, z.B. mit dem Dijkstra-Algorithmus [Dij59], bestimmen. Insofern ist GSR weniger ein vollständiges Routing-Protokoll, sondern eher ein Protokoll zur Topologieerkennung. Ein Problem von GSR ist, dass es nicht sicherstellt, dass alle Knoten eine konsistente Sicht der Topologie haben. Dies kann wiederum zu Routing-Schleifen führen, falls dies nicht anderweitig verhindert wird.

Auf Grund von Störungen auf dem Medium, Knotenbewegungen oder geringer Empfangsstärke über lange Distanzen, sind drahtlose Links oft nicht zuverlässig und instabil. Ein Problem vieler Protokolle, welche die Kommunikationstopologie erkennen, ist, dass sie die Zuverlässigkeit und Stabilität der erkannten Kommunikationslinks nicht ausreichend überprüfen. Darüber hinaus können drahtlose Links unidirektional sein, beispielsweise weil unterschiedliche Antennen zum Einsatz kommen oder die Batterien der Knoten unterschiedlich stark geladen sind. Auch dieser Aspekt wird von vielen Protokollen nicht ausreichend berücksichtigt. Bei AODV oder DSDV wird beispielsweise aus einer einzelnen erfolgreichen Übertragung auf einen bidirektionalen Link geschlossen. OLSR und Babel hingegen berücksichtigen bidirektionale Links. Die Zuverlässigkeit der Links wird bei OLSR nicht statistisch erfasst, sondern lediglich periodisch die Existenz der Links überprüft. Auch Babel führt keine Statistiken über die Zuverlässigkeit von Links, unterstützt es allerdings, Metriken zu verwenden, die dies tun. Ein Beispiel für eine solche Metrik ist ETX [De +05]. Diese Metrik ermittelt die Paket Delivery Rate (PDR) eines Links

in beide Richtungen und berechnet daraus die erwartete Anzahl notwendiger Übertragungen auf diesem Link. Durch Einsatz dieser Metrik werden stabile bidirektionale Links gegenüber instabilen und unidirektionalen Links bevorzugt verwendet.

3.2.1.2 Topologieerkennung in Clustering-Protokollen

Clustering-Protokolle gruppieren die Knoten eines Netzwerks in Gruppen, sogenannte *Cluster*. Dabei werden i.d.R. Knoten zu einem Cluster zusammengefasst, wenn sie räumlich nah beieinander liegen, also direkt oder über wenige Zwischenhops kommunizieren können. Beispielsweise bilden einige Clustering-Algorithmen *d-Hop Dominating Sets* [Ami+00], d.h. jedes Cluster ist eine Menge von Knoten $V' \subseteq V$, sodass für alle Knoten $v \in V'$ dieses Clusters gilt, dass sie in d-Hop Reichweite eines Knotens $v_h \in V'$ sind. Dieser Knoten v_h wird i.d.R. Clusterhead genannt. Um nun Cluster bilden zu können, benötigt ein Clustering-Verfahren Informationen über die Kommunikationstopologie des Netzwerks. Allerdings hängt die Menge an benötigter Information vom Verfahren ab. Viele Clustering-Verfahren erkennen nur eine partielle Topologie. So ist beispielsweise die Bildung eines d-Hop Dominating Sets möglich, wenn lediglich Information über die d-Hop-Nachbarschaft vorhanden ist. Indem nur die benötigte partielle Topologieinformation erkannt und verteilt wird, gewinnen Clustering-Verfahren an Effizienz. Genau wie aggregierende Routing-Verfahren bedeutet dies aber auch, dass die gewonnene Information oft nicht für andere Protokolle oder Anwendungen, die Topologieinformationen benötigen, ausreicht.

In [BBH13] und [AY07] werden bekannte Clustering-Algorithmen vorgestellt und gegenübergestellt. In Kapitel 3.4 wird ein eigenes Clustering-Verfahren beschrieben, welches allerdings keine eigene Topologieerkennung mitbringt, sondern ein Verfahren zur Topologieerkennung wie das in Kapitel 3.2.2 vorgestellte Automatic Topology Discovery Protocol (ATDP) voraussetzt. Wir betrachten hier exemplarisch das bekannte Clustering-Verfahren Max-Min-D [Ami+00] in Bezug auf die enthaltene Funktionalität zur Topologieerkennung. Das Verfahren ist für drahtlose Netzwerke konzipiert und hat zum Ziel, Konnektivität innerhalb der Cluster sowie zwischen den Clustern zu erzielen, sowie die Cluster-Größe zu balancieren, also möglichst gleich große Cluster zu bilden. Darüber hinaus versucht es, eine möglichst geringe Anzahl Cluster zu bilden. Das Verfahren bildet d-Hop Dominating Sets und ist entsprechend über den Parameter d parametrierbar. Die Cluster-Bildung läuft in mehreren Phasen ab: In der Floodmax-Phase werden die Clusterheads vorausgewählt, indem die Knoten die größte Knoten-ID in ihrer d-Hop-Nachbarschaft bestimmen. Anschließend balanciert das Verfahren die Cluster in der Floodmin-Phase, wobei weitere Informationen aus der $2d$-Hop-Nachbarschaft berücksichtigt werden.

Die Topologieerkennung in Max-Min-D weist ein paar Schwächen auf: So wird aus dem Empfang einer einzelnen Nachricht auf einen Kommunikationslink geschlossen. Dies hat zur Folge, dass auch extrem unzuverlässige Links Verwendung finden können. Darüber hinaus wird nicht überprüft, ob Links symmetrisch sind. Dadurch, dass das Verfahren versucht, eine kleine Cluster-Anzahl zu bilden, tendiert das Verfahren dazu, die Konnektivität über Links herzustellen, welche eine lange Distanz überbrücken. Die gleiche Tendenz haben Routing-Verfahren, welche Routen auswählen, die möglichst wenig Hops überqueren. Da bei drahtlosen Netzwerken Links über weite Distanzen i.d.R. unzuverlässiger sind als Links über kurze Distanzen, werden also auf Basis dieser Metriken besonders schlechte Links bevorzugt ausgewählt. Daraus lässt sich folgern, dass Max-Min-D dazu tendiert, Konnektivität über unzuverlässige Links herzustellen. Ähnliche Probleme lassen sich in vielen weiteren Clustering-Protokollen, wie etwa dem Algorithm for Cluster Establishment (ACE) [CP04] finden.

3.2.1.3 Erkennung der Interferenz- und Sensing-Topologie

Die genannten Routing- und Clustering-Algorithmen erkennen ausschließlich die Kommunikationstopologie, nicht aber die Interferenz- oder Sensing-Topologie. Zur Bestimmung der Interferenztopologie gibt es relativ wenige Ansätze, da die Erkennung von Interferenzlinks nicht einfach ist. Schickt ein Knoten einen Rahmen, dann empfangen Knoten in seiner Kommunikationsreichweite diesen und können auf einen Kommunikationslink schließen. Knoten in Interferenzreichweite empfangen den Rahmen allerdings nicht als Rahmen, sondern nehmen lediglich etwas mehr Energie auf dem Medium wahr oder empfangen einen korrupten Rahmen. Da sie den gesendeten Rahmen i.d.R. nicht auslesen können, wissen sie nicht, welcher Knoten den Rahmen gesendet hat. Daher kann der empfangende Knoten nicht bestimmen, von welchem Knoten der Interferenzlink ausgeht.
Um das Problem zu lösen, wurden Interferenz-Modelle entwickelt, welche die Interferenzreichweite unter Annahmen aus der Kommunikationstopologie herleiten. Beispielsweise beschreibt Sharma et al. [SMS06] ein einfaches solches Modell, bei dem ein Interferenzlink zwischen zwei Knoten immer dann besteht, wenn zwischen den Knoten eine Route über maximal k Hops (d.h. Kommunikationslinks) besteht. Es lässt sich aber zeigen, dass diese Heuristik mit $k = 2$ die tatsächliche Interferenz überschätzt, und dennoch nicht sicherstellen kann, dass alle Interferenzlinks berücksichtigt werden [Pad+05].

Padhye et al. [Pad+05] führten Experimente zur Bestimmung der Interferenztopologie mit IEEE 802.11 Knoten durch. Hierbei senden zunächst einzelne Knoten exklusiv, während alle anderen Knoten die Empfangsrate der gesendeten Rahmen messen. Daraufhin senden Paare von Knoten gleichzeitig, während alle anderen Knoten das Medium abhören und die Empfangsrate messen. Die gemessenen Empfangsraten eines Knotenpaars werden in Relation zu den Empfangsraten der einzeln sendenden Knoten gesetzt und so die sog. *Broadcast Interferenzrate* berechnet. Obwohl die Autoren von einer hohen Genauigkeit berichten, verlangt das Verfahren für n Knoten die Durchführung von $\mathcal{O}(n^2)$ aufwendigen Experimenten. Dies erklärt, warum die Autoren bei einem Testbett mit 22 Knoten von einer Laufzeit von 28 Stunden berichten. Für ein einmaliges Ausmessen eines fest installierten Netzwerkes ist dieser Ansatz durchaus interessant, auf Grund der hohen Komplexität und Laufzeit aber für viele Anwendungen ungeeignet.

Das Radio Interference Protocol (RID) [Zho+05] ist ein anderer Ansatz, die Interferenztopologie zu bestimmen. Es versendet zur Erkennung eines Interferenzlinks eine Sequenz aus zwei Rahmen. Der erste Rahmen wird mit erhöhter Sendestärke versendet, und dient lediglich dazu, die Identität des sendenden Knoten mitzuteilen. Der zweite Rahmen wird mit normaler Sendestärke gesendet und dient zur Messung des Links. Ein Knoten in Interferenzreichweite des Senders kann anhand des zweiten Rahmens den Link als Interferenzlink einstufen und ihn anhand des ersten Rahmens dem Sender zuordnen. Die Annahme dieses Ansatzes ist, dass die Kommunikationsreichweite mit hoher Sendestärke mindestens so groß ist wie die Interferenzreichweite mit normaler Sendestärke. In der Praxis schränkt diese Annahme die für normale Kommunikation nutzbare Sendestärke ein und führt zu einem erhöhten Energieverbrauch durch das Senden mit höherer Sendestärke. Auch wenn von den Autoren nicht erwähnt, lässt sich das Verfahren theoretisch auch nutzen, um die Sensing-Topologie zu erkennen. In diesem Fall müsste die Sendestärke des ersten Rahmens allerdings mindestens so hoch sein, dass die Kommunikationsreichweite mit hoher Sendestärke mindestens der Sensing-Reichweite mit normaler Sendestärke entspricht. Dies würde in der Praxis die für normale Kommunikation nutzbare Sendestärke und damit Reichweite wahrscheinlich zu stark einschränken.

3.2.2 Automatic Topology Discovery Protocol (ATDP)

Mit dem zuerst in [KCG15] publizierten Automatic Topology Disovery Protocol (ATDP) wurde ein Protokoll zur Topologieerkennung entwickelt mit dem Ziel, die in Kapitel 3.2.1 angesprochenen Einschränkungen vorhandener Ansätze zu überwinden. Sobald das Netzwerk eingeschaltet wird, beginnt ATDP mit der automatischen Erkennung der Kommunikations-, Interferenz- und Sensing-Topologie eines drahtlosen TDMA-basierten Netzwerks. Die erkannten Topologien werden dabei durch das Protokoll mit wenig Overhead netzweit verteilt, wobei das Protokoll sicherstellt, dass alle Knoten ein konsistentes Bild des Netzwerkes haben, sobald ATDP terminiert und der reguläre Betrieb des Netzwerkes startet. Das Protokoll wurde im Rahmen von ProNet 4.0 in Hinblick auf die Anwendung der Prozessautomatisierung mit Hilfe drahtloser Kommunikationssysteme (s. Kapitel 2.2) und den sich daraus ergebenden Anforderungen entwickelt. Eine wichtige Annahme ist dabei, dass die meisten Knoten des Netzwerkes fest installiert sind und sich nicht bewegen können. Da ATDP keine Topologieerkennung während des regulären Betriebs vorsieht, ist es nicht für Netzwerke geeignet, in denen die Knoten beweglich sind.

3.2.2.1 Zuordnung von TDMA-Zeitslots

ATDP ist für TDMA-basierte drahtlose Netzwerke konzipiert und erfordert Tick- bzw. Zeitsynchronisation mit beschränkten Offsets, sodass die Exklusivität der reservierten Übertragungen sichergestellt ist. Ein hierzu geeignetes Protokoll ist Black Burst Synchronization (BBS) [GK11]. Beim Einschalten des Netzwerkes beginnt ATDP mit der automatischen Topologieerkennung. Währenddessen ist die Zeit wie in Abb. 3.3 dargestellt strukturiert: Zunächst wird die Zeit in *Superslots* eingeteilt, welche sich periodisch wiederholen. Jeder Superslot beginnt mit einem Supertick. Neben dem Einleiten eines neuen Superframes dient der Supertick auch zur Resynchronisation. Superslots sind wiederum in *Macroslots* eingeteilt, die auch jeweils mit einem Tick beginnen. Macroslots dienen lediglich dazu, dass die Resynchronisation häufiger als einmal pro Supertick erfolgen kann. Zu Beginn eines Superslots folgt auf den Supertick die *TERM*-Phase, die dazu dient zu entscheiden, ob ATDP terminiert (s. Kapitel 3.2.2.5). Die restliche Zeit wird in Microslots aufgeteilt, welche die exklusiv reservierten TDMA-Zeitslots darstellen.

In ATDP wird jedem potenziell im Netzwerk befindlichen Knoten mindestens ein TDMA-Zeitslot (Microslot) exklusiv zugewiesen, der sich (schwach) periodisch wiederholt. Die Zuordnung der Zeitslots ist statisch und wird anhand der Knoten-Identifier bestimmt, sodass jeder Knoten

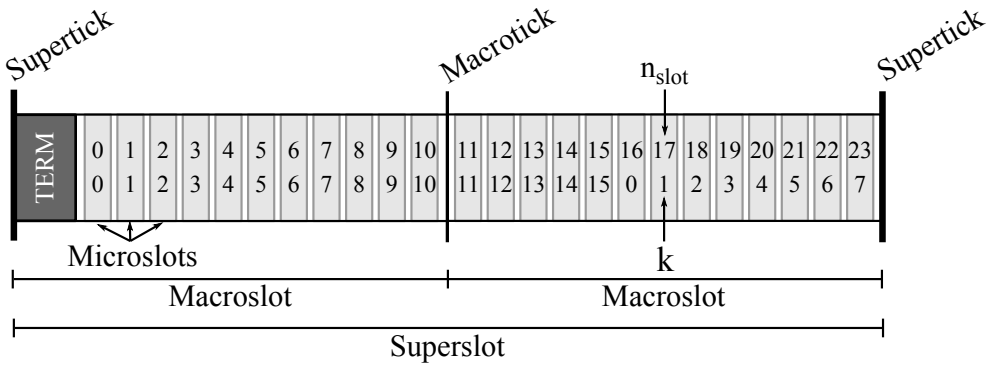


Abbildung 3.3: Beispiel für die Zuordnung der TDMA-Zeitslots mit zwei Macroslots pro Superslot und $N_{\text{nodesMax}} = 16$ Microslots sowie der resultierenden Zuordnung der Microslots zu Knoten.

die Zuordnung aller Zeitslots kennt. Es wird davon ausgegangen, dass jedem Knoten ein eindeutiger numerischer Knoten-Identifier k aus dem Intervall $[0, N_{nodesMax-1}]$ per Konfiguration zugeordnet ist. Der Parameter $N_{nodesMax}$ beschränkt damit die mögliche Anzahl Knoten im Netzwerk. Er ist allen Knoten durch Konfiguration bekannt. Damit jedem potenziell im Netzwerk vorhandenen Knoten ein Zeitslot zugeordnet werden kann, muss die Länge des Superslots mindestens so groß sein, dass darin $N_{nodesMax}$ Microslots passen. Die Länge des Superslots wird daher bei der Konfiguration abhängig von der Anzahl potenziell vorhandener Knoten festgelegt. Die Anzahl pro Superslot enthaltener Macroslots ergibt sich aus dem Intervall, in dem resynchronisiert werden muss, um die benötigte Synchronisationsgenauigkeit zu erreichen. Die Unterscheidung zwischen Superslots und Macroslots ist also nötig, um viele Knoten unterstützen zu können und gleichzeitig die benötigte Synchronisationsgenauigkeit sicherstellen zu können.

Die Microslots werden innerhalb eines Superframes durchnummeriert, d.h. die Nummerierung läuft über die Grenzen von Macroslots hinaus weiter. Der Knoten k, dem ein Slot mit der Nummer n_{slot} exklusiv zugeordnet wird, ergibt sich aus Formel (3.1):

$$k = n_{\text{slot}} \bmod N_{\text{nodesMax}} \tag{3.1}$$

Da $N_{nodesMax}$ per Konfiguration allen Knoten bekannt ist, kann jeder Knoten anhand dieser Formel die Zuordnung aller Zeitslots bestimmen. Insbesondere ist es nicht notwendig, die Zuordnung zwischen den Knoten auszutauschen, was sich günstig auf den Kommunikations-Overhead des Protokolls auswirkt. In Abb. 3.3 ist eine Beispiel-Konfiguration mit zwei Macroslots pro Superslot zu sehen, sowie die sich aus der Formel ergebende Zuordnung der Microslots für bis zu $N_{nodesMax} = 16$ Knoten.

3.2.2.2 Ausmessung der Links

Ein Knoten sendet in allen ihm zugeordneten Zeitslots eine *MEASURE*-Nachricht per Broadcast. In allen anderen Zeitslots hört der Knoten das Medium ab. Je nachdem, ob und wie stark er eine *MEASURE*-Nachricht eines anderen Knotens empfängt, kann er die Qualität des Links vom Absender der Nachricht zu ihm selbst einschätzen. Durch die exklusive Zuordnung von Slots kann selbst ein Knoten, der eine korrupte Nachricht empfängt oder lediglich erhöhte Energie auf dem drahtlosen Medium feststellt, wissen, welcher Knoten in diesem Slot sendet. Dies löst das Problem, dass ein Knoten in Interferenzreichweite eines anderen dessen Nachrichten im Allgemeinen nicht dekodieren kann, und daher nicht ohne zusätzliche Information bestimmen kann, welcher Knoten die Nachricht gesendet hat.

Hört ein Knoten k_2 in einem Zeitslot n_{slot} das Medium ab, der nach Formel (3.1) einem Knoten k_1 zugeordnet ist, dann bewertet er in diesem Slot den (potenziellen) Link $k_1 \rightarrow k_2$. Dabei können folgende Fälle auftreten:

1. Der Knoten k_2 empfängt einen korrekten Rahmen mit einer Signalstärke, die über dem Grenzwert $minRSS_{comm}$ liegt. In diesem Fall wird der Link $k_1 \rightarrow k_2$ als Kommunikationslink (T_{comm}) eingestuft.
2. Der Knoten k_2 empfängt einen korrupten Rahmen oder einen korrekten Rahmen mit einer geringen Signalstärke aus dem Intervall $[minRSS_{int}, minRSS_{comm}[$. In diesem Fall wird davon ausgegangen, dass keine zuverlässige Kommunikation auf dem Link möglich ist, aber k_1 den Empfang von k_2 stören kann. Entsprechend wird der Link als Interferenzlink (T_{int}) eingestuft. Die gleiche Einstufung erfolgt, falls kein Rahmen empfangen wird, aber für die

Dauer einer Rahmenübertragung auf dem Medium erhöhte Energie zwischen $minRSS_{int}$ und $minRSS_{comm}$ detektiert wird.

3. Der Knoten k_2 empfängt keinen Rahmen, aber auf dem Medium wird für die Dauer einer Rahmenübertragung erhöhte Energie detektiert mit einer Signalstärke aus dem Intervall $[minRSS_{sense}, minRSS_{int}[$. In diesem Fall wird der Link als Sensing-Link (T_{sense}) eingestuft.
4. Der Knoten k_2 empfängt weder einen Rahmen noch beobachtet er erhöhte Energie auf dem Medium, d.h. das Medium wird idle erkannt. Es kann davon ausgegangen werden, dass k_2 sich außerhalb der Reichweite von k_1 befindet oder k_1 gar nicht vorhanden ist. Der Link wird folglich als *Kein Link* (T_{no}) eingestuft.

Abb. 3.4 veranschaulicht die bei der Bestimmung des Linktyps benutzten Grenzwerte und nennt beispielhaft mögliche Werte. Zu beachten ist, dass neben der Signalstärke auch die Korrektheit des Empfangs berücksichtigt wird, wozu die in der Nachricht enthaltene Prüfsumme verglichen wird. Der Empfang einer verfälschten *MEASURE*-Nachricht mit einer Signalstärke von -80 dBm führt beispielsweise zu einer Einstufung des Links als Interferenzlink, auch wenn der Grenzwert $minRSS_{comm} = -82dBm$ überschritten wird.

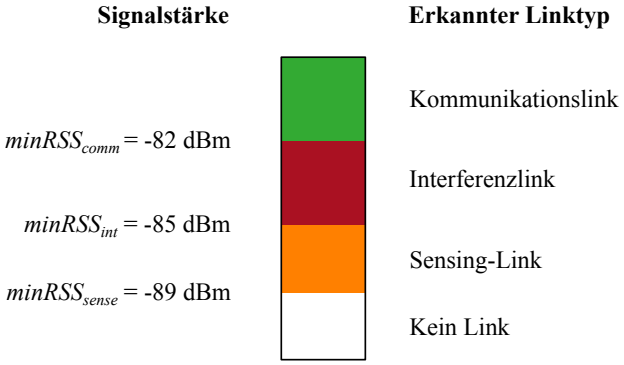
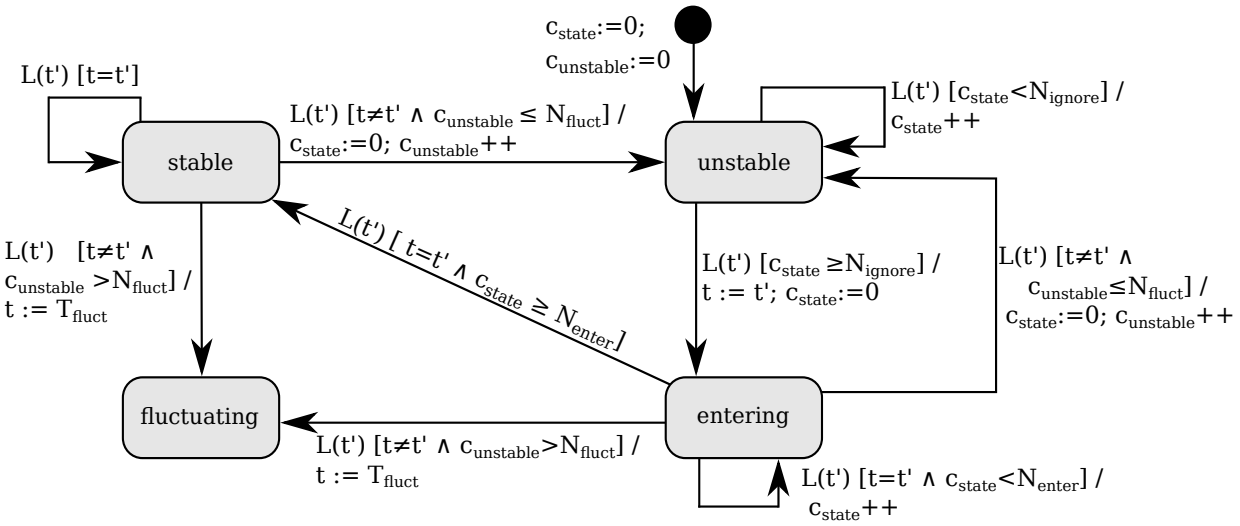


Abbildung 3.4: Beispielhafte Grenzwerte für die Bewertung eines Links

3.2.2.3 Bestimmung des Link-Zustands

Aus einer einzelnen Beobachtung kann nicht zuverlässig auf die langfristige Qualität eines Links geschlossen werden. Aus diesem Grund wiederholt ATDP die Bewertung der Links solange, bis alle Links entweder stabil einem Linktypen zugeordnet werden konnten, oder aber der Link als *fluctuating* eingestuft wird. Dies ist der Fall, wenn der Link sich mit der Zeit stetig verändert und daher nicht einem einzelnen Linktypen zugeordnet werden kann. Dies ist beispielsweise dann der Fall, wenn sich ein Knoten bewegt. Ein solcher Link sollte i.d.R. von anderen Protokollen wie Interferenzlinks behandelt werden, da der Link jederzeit zu Interferenzen führen kann. Protokolle, die über Sensing-Links arbeiten, dürfen im Gegensatz zu Interferenzlinks allerdings bei als *fluctuating* eingestuften Links nicht davon ausgehen, dass sie auch ein zuverlässiger Sensing-Link sind.

Um die Stabilität der Links zu bewerten, folgt ATDP dem in Abb. 3.5 veranschaulichten *Link-Zustandsgraphen*. Hier wird eine einzelne Einstufung eines Links L als einen Link vom Typ $t' \in \{T_{comm}, T_{int}, T_{sense}, T_{no}\}$ als Link-Event $L(t')$ bezeichnet. Zu Beginn der Topologieerkennung sind alle Links zunächst als *unstable* eingestuft, entsprechend beginnt jeder Link im gleichnamigen Startzustand (s. Abb. 3.5). In diesem Zustand wird zunächst unabhängig vom erkannten Linktyp t' für $N_{ignore} = 10$ Link-Events verharrt. Dies dient dazu, dem Link ein wenig Zeit zu geben, stabil zu werden. Dies ist beispielsweise hilfreich, wenn Knoten beim Start des Netzwerks



Ereignisse:

L(t')	Link-Event: Einstufung des Links L als Linktyp t'

Variablen:

t	aktuell angenommener Linktyp
$c_{unstable}$	Anzahl der Übergänge in Zustand unstable
c_{state}	Anzahl der Link-Events im aktuellen Zustand

Konstanten:

N_{fluct}	Maximale Anzahl Übergänge in den Zustand unstable
N_{ignore}	Anzahl im Zustand unstable zu ignorierende Link-Events
N_{enter}	Anzahl aufeinanderfolgende identische Link-Events, die für die Einstufung eines Links als stable benötigt wird

Abbildung 3.5: Zustandsgraph zur Bestimmung des Linktyps in ATDP

nicht alle zeitgleich eingeschaltet werden oder noch bewegt werden. Im Folgenden wird davon ausgegangen, dass die folgende Bewertung t' zutrifft. Da diese Bewertung zunächst nur auf einer einzelnen Beobachtung basiert, wird sie noch nicht als stabil eingestuft, sondern der Link in den Zustand *entering* versetzt, in welchem er darauf wartet, als stabil eingestuft zu werden. Damit dies geschieht, muss der Link für $N_{enter} = 30$ Link-Events gleich eingestuft werden, also der zuvor erkannte Typ t dem neu erkannten Typen t' entsprechen. Ist dies der Fall, gelangt der Link in den Zustand *stable* und verbleibt dort, solange die Einstufung des Links sich nicht ändert. Sollte der Link im Zustand *entering* oder *stable* anders bewertet werden als zuvor, so geht er zurück in den *unstable*-Zustand, wo er wieder für $N_{ignore} = 10$ Link-Events verharrt, um dem Link Zeit zu geben, stabil zu werden.

Manche Links verändern sich stetig, gelangen also immer wieder in den Zustand *unstable* zurück, beispielsweise weil einer der beteiligten Knoten sich bewegt. Jedes Mal, wenn ein Link in den *unstable*-Zustand überführt wird, erhöht sich daher der Zähler $c_{unstable}$ um eins. Überschreitet dieser Zähler die Schranke $N_{fluct} = 10$, so wird der Link als *fluctuating* eingestuft und in den gleichnamigen Zustand versetzt, in dem er fortan verharrt.

Ein Beispiel eines Kommunikationslinks von einem Knoten, der erst später eingeschaltet wird als die anderen und daher im Laufe der Topologieerkennung den Link-Zustand mehrfach verändert, ist in Abb. 3.6 illustriert. Der Link beginnt im Startzustand *unstable* und verbleibt dort für

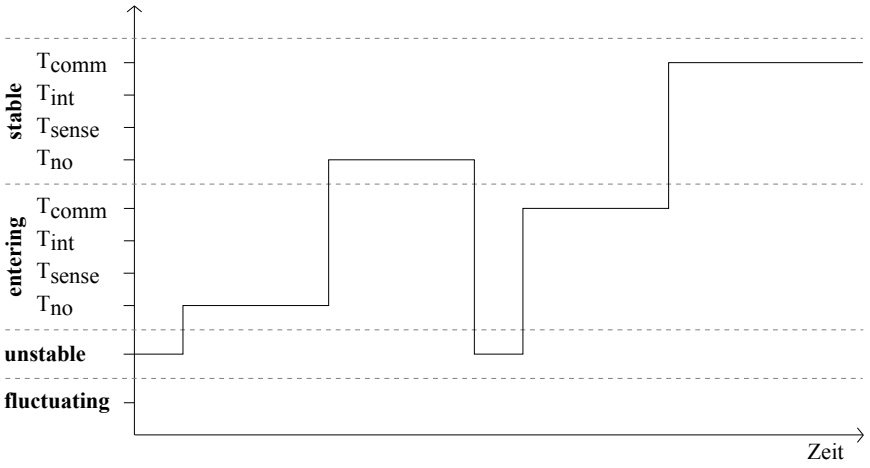


Abbildung 3.6: Beispiel eines Links, der seinen Link-Zustand mehrfach ändert

$N_{ignore} = 10$ Link-Events. Danach wechselt der Link in den Zustand *entering* mit der Einstufung als *kein Link* ($t = T_{no}$), da der Sender des Links noch nicht eingeschaltet ist und daher von ihm weder *MEASURE*-Nachrichten empfangen wurden noch Energie auf dem Medium gemessen wurde. Da der Link nun für die nächsten $N_{enter} = 30$ Link-Events gleich eingestuft wird, wechselt er in den Zustand *stable* mit unveränderter Einstufung ($t = T_{no}$). In diesem Zustand verbleibt er solange, bis der Sender des Links eingeschaltet wird und die erste *MEASURE*-Nachricht von ihm korrekt empfangen wurde. Der Empfang der Nachricht führt zur Einstufung als Kommunikationslink ($t' = T_{comm}$). Da diese Einstufung der bisherigen Einstufung ($t = T_{no}$) widerspricht, wechselt der Link in den Zustand *unstable*. Hier verbleibt er wieder für $N_{ignore} = 10$ Link-Events und wechselt dann in den Zustand *entering*, allerdings jetzt mit der neuen Einstufung T_{comm}. Nachdem diese Einstufung für $N_{enter} = 30$ Link-Events unverändert bleibt, wird der Link als stabiler Kommunikationslink eingestuft und wechselt in den Zustand *stable*, wo er verbleibt.

3.2.2.4 Verteilung der Link-Information

MEASURE-Nachrichten sind normale Datenrahmen, die eine feste Länge haben und stets per Broadcast übertragen werden. Sie dienen im Gegensatz zu *HELLO*-Nachrichten oder Beacons in anderen Protokollen allerdings nicht nur dazu, die Linkqualität zu bestimmen. Die Payload der Nachrichten füllt ein Knoten stets mit den bisher erlangten Informationen über Links im Netzwerk auf. Dies beinhaltet sowohl von ihm selbst erkannte Links als auch die Links, die ihm durch andere Knoten mitgeteilt wurden. Auf diese Weise werden während der Ausmessung bereits die erkannten Topologien verteilt. Da zum Ausmessen der Links ohnehin Nachrichten gesendet werden müssen, erzeugt die Verteilung dieser Rahmen keinen zusätzlichen Kommunikationsaufwand. Falls die bekannte Topologieinformation nicht in eine Nachricht passt, wird fragmentiert. Neue Informationen werden stets bevorzugt übertragen, damit sich diese schnell verbreiten. Sollten weniger Links bekannt sein, als in eine Nachricht passen, so wird die Nachricht mit Fülldaten aufgefüllt. So wird sichergestellt, dass die Nachrichten immer eine feste Größe haben.

Für jeden in einer *MEASURE*-Nachricht enthaltenen Link werden folgende Informationen übertragen:

- *Knoten-Identifier von Sender und Empfänger:* Ein Link $k_1 \rightarrow k_2$ wird eindeutig über die Knoten-Identifier des Senders k_1 und des Empfängers k_2 identifiziert. Es wird davon ausge-

gangen, dass drahtlose Links asymmetrisch sein können. Daher wird der Link $k_2 \rightarrow k_1$ unabhängig vom Link $k_1 \rightarrow k_2$ betrachtet.

- *Erkannter Linktyp:* Hierbei werden folgende Typen unterschieden:
  - *Kommunikationslink*: Ein Link, über den k_1 an k_2 senden kann.
  - *Interferenzlink*: Ein Link, über den k_1 den Empfang bei k_2 stören kann, aber über den keine (zuverlässige) Kommunikation möglich ist.
  - *Sensing-Link*: Ein Link, bei dem k_2 erhöhte Energie auf dem Medium feststellen kann, wenn k_1 sendet.
  - *Kein Link*: Wenn k_1 sendet, empfängt k_2 nicht mehr als das übliche Rauschen.
  - *Fluctuating*: Ein Link, der sich mit der Zeit verändert und nicht klar eingestuft werden kann.
- *Sequenznummer:* Der Knoten k_2, der den Link $k_1 \rightarrow k_2$ bewertet, vergibt die Sequenznummer für diesen Link und erhöht sie jedes Mal, wenn sich die Einstufung ändert. Anhand der Sequenznummer kann ein Knoten die Aktualität der Information bewerten und seine gespeicherten Informationen entweder aktualisieren oder veraltete Informationen ignorieren.
- *Signalstärke:* Die durchschnittliche Signalstärke der auf diesem Link empfangenen Nachrichten, falls es sich um einen Kommunikationslink handelt. Andere Protokolle, wie z.B. Routing-Protokolle, können anhand dieser Information stärkere Links bevorzugt benutzen.

Um zu erkennen, ob die Nachricht fehlerfrei übertragen wurde, enthält jede *MEASURE*-Nachricht darüber hinaus eine Prüfsumme.

3.2.2.5 Terminierung

Ziel von ATDP ist es, mit einem netzweit einheitlichen Ergebnis zu terminieren. Das bedeutet, dass alle an der Topologieerkennung beteiligten Knoten die selben Informationen über die Topologie des Netzwerkes besitzen, nachdem ATDP terminiert hat. Dadurch, dass alle Knoten die gleiche Sicht auf das Netzwerk haben, lassen sich beispielsweise Routing-Schleifen ausschließen. Um dieses Ziel zu erreichen, darf ATDP nicht terminieren, solange irgendwo im Netz noch Links nicht abschließend eingestuft worden sind oder eine neue Information noch nicht netzweit ausgetauscht worden ist. ATDP terminiert also erst, sobald alle Knoten der Terminierung zustimmen. Diese Zustimmung gibt ein Knoten erst, sobald die folgenden Bedingungen erfüllt sind:

1. Alle erkannten Links sind entweder im Zustand *stable* oder *fluctuating*.
2. Die Link-Information, die von anderen Knoten über *MEASURE*-Nachrichten empfangen wurde, hat sich für $N_{requiredStable} = 3$ Superslots nicht verändert.

Die erste Bedingung stellt sicher, dass die Erkennung der Links von den Nachbarknoten zum Knoten selbst abgeschlossen ist. Aus dem Link-Zustandsgraphen (Abb. 3.5) ergibt sich, dass ein Link letztlich entweder als stabil erkannt wurde und sich im Zustand *stable* befindet, oder fortwährend unterschiedlich eingestuft wurde und daher in den Zustand *fluctuating* gewechselt hat (vgl. Kapitel 3.2.2.3). Es ist also sichergestellt, dass diese Bedingung irgendwann erfüllt sein wird.

Die zweite Bedingung stellt sicher, dass neue Informationen im ganzen Netz verteilt sind. Stuft ein Knoten einen Link neu ein, so wird er nach Bedingung 1 zunächst selbst die Terminierung verhindern. Da neue Informationen stets bevorzugt in *MEASURE*-Nachrichten übertragen werden (s. Kapitel 3.2.2.4), ist sichergestellt, dass (eine) neue Information mit der nächsten *MEASURE*-Nachricht übertragen wird. Sobald diese bei einem direkten Nachbarn ankommt, wird dieser die Terminierung nach Bedingung 2 verhindern. Für ihn ist diese Information neu und wird damit bevorzugt mit der nächsten *MEASURE*-Nachricht übertragen. Somit erreicht die Information Knoten, die zwei Hops von dem Knoten entfernt sind, der den Link ursprünglich neu eingestuft hatte. Auf diese Weise wird die Information im ganzen Netzwerk übertragen. Erst wenn die Information alle Knoten im Netzwerk erreicht hat, gibt es keinen Knoten mehr, der sie als neu einstuft. Somit stimmen erst alle Knoten der Terminierung zu, sobald jegliche neuen Informationen netzweit verteilt worden sind. Theoretisch ist die schwächere Bedingung ausreichend, dass nach dem Erhalt neuer Informationen mindestens eine *MEASURE*-Nachricht gesendet wurde. Für den Fall, dass diese Nachricht nicht korrekt übertragen wurde, wird mit dem Parameter $N_{requiredStable}$ zusätzlich Redundanz hinzugefügt.

Die Terminierungsentscheidung erfolgt also gemeinschaftlich verteilt über alle Knoten. Jeder einzelne Knoten hat das Recht, die Terminierung zu verhindern – nur wenn alle Knoten zustimmen, wird terminiert. Um dieses Verhalten effizient zu implementieren, wurde auf das AVTP-Protokoll zurückgegriffen [CGR12]. Dabei senden alle Knoten, die der Terminierung widersprechen, in der *TERM*-Phase gleichzeitig eine *NOTERM*-Nachricht. Obwohl die Übertragung zeitgleich erfolgt und es zu Interferenzen der Übertragungen kommen kann, stellt die kollisionsgeschützte Black Burst-Kodierung des AVTP-Protokolls sicher, dass eine *NOTERM*-Nachricht von allen Knoten empfangen wird, die nicht selbst eine gesendet haben. Mit Abschluss der *TERM*-Phase ist also allen Knoten bekannt, ob mindestens ein Knoten der Terminierung widersprochen hat und das Protokoll daher weiterhin ausgeführt wird. Das AVTP-Protokoll erreicht dieses Ziel auch über mehrere Hops, indem es in mehreren Runden *NOTERM*-Nachrichten übertragt. Wie dies AVTP im Detail umsetzt, beschreibt [CGR12].

3.2.3 Implementierung und Evaluierung

Im Rahmen von ProNet 4.0 wurde ATDP in C++ implementiert und auf Imote2-Knoten (s. Kapitel 3.1.2) getestet. In ProNet 4.0 existieren bereits effiziente auf IEEE 802.15.4 basierende Implementierungen für BBS zur Synchronisation und das für die Terminierung verwendete AVTP [Eng13]. Um die Implementierung des Protokolls zu testen und den Verlauf der Topologieerkennung verfolgen und protokollieren zu können, wurde die UART-Schnittstelle der Imote2-Knoten benutzt: Jegliche Link-Änderungen werden live über die Schnittstelle protokolliert. Darüber hinaus gibt die Implementierung zu Beginn jedes Superslots die aktuelle Link-Matrix aus, welche zu jedem Link den aktuell erkannten Linktypen enthält. Ein an die UART-Schnittstelle angeschlossener PC empfängt diese Ausgaben, zeigt sie in Echtzeit an und protokolliert sie. Auf diese Weise kann das Verhalten des Topologieerkennungs-Protokolls sowohl live beobachtet als auch statistisch ausgewertet werden.

Unter Berücksichtigung der minimalen Empfangsempfindlichkeit des eingesetzten CC2420 Transceivers von -90 dBm wurden die in Abb. 3.4 genannten Grenzwerte für die Einstufung der Links gewählt. Ziel dabei war es sowohl, ausschließlich zuverlässige Links als Kommunikationslink einzustufen, als auch vorhandene Sensing-Links zu erkennen. Mit der Anpassung der Grenzwerte kann das Protokoll auf die vorhandene Hardware und die gewünschten Ziele abgestimmt

werden. Senkt man beispielsweise den Grenzwert für die Bewertung als Kommunikationslink $minRSS_{comm}$ ab, so erhält man mehr Kommunikationslinks, da auch unzuverlässigere Links als Kommunikationslink eingestuft werden. Durch die zusätzlichen Kommunikationslinks werden kürzere Pfade durch das Netzwerk möglich. Benötigt man hingegen eine höhere Zuverlässigkeit, kann man den Grenzwert weiter erhöhen.

Um Interferenzlinks und Sensing-Links erkennen zu können, bei denen keine Nachricht empfangen wird, wurde der Clear Channel Assessment (CCA) Mechanismus des Transceivers auf den Grenzwert für Sensing-Links $minRSS_{sense} = -89dBm$ konfiguriert. Der CCA-Mechanismus löst einen Hardware Interrupt aus, sobald die auf dem Medium gemessene Energie über diesem Grenzwert liegt. Auf diesen Interrupt hin liest ATDP das Received Signal Strength Indicator (RSSI) Register zu Bestimmung der Signalstärke aus. Außerdem wird die Zeit gemessen, für den der CCA-Mechanismus ein belegtes Medium registriert. Nur wenn diese Zeit in etwa der Zeit entspricht, die die Übertragung einer MEASURE-Nachricht benötigt, wird die empfangene Energie berücksichtigt.

Zu Evaluierung von ATDP wurden unterschiedliche Experimente durchgeführt, wobei das Protokoll stets nach wenigen Minuten terminierte und die erkannte Topologie sinnvoll erschien. Um das Ergebnis der Topologieerkennung genauer zu überprüfen, wurde ein Testszenario detaillierter untersucht. Dazu wurden 5 Testknoten in mehreren Räumen eines Universitätsgebäudes verteilt (s. Abb. 3.7). In Reichweite dieser Räume befinden sich IEEE 802.11 Netzwerke auf 2.4 GHz Kanälen. Da IEEE 802.15.4 auch 2,4 GHz benutzt, wurde für die Experimente ein IEEE 802.15.4 Kanal ausgewählt, auf dem kein IEEE 802.11 Netzwerk sendet (auch nicht überlappend).

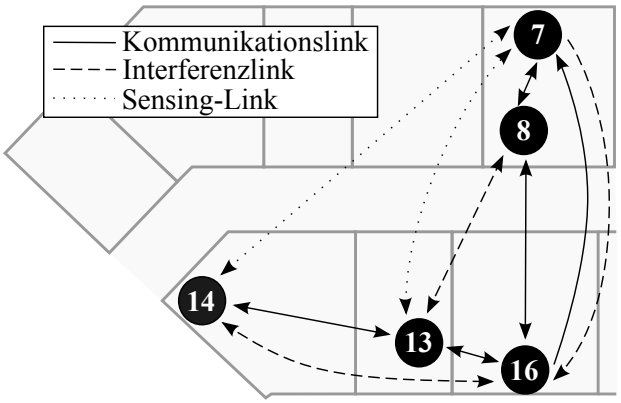


Abbildung 3.7: Experiment mit 5 Knoten in unterschiedlichen Räumen und den erkannten Links

Die Topologieerkennung wurde mit der selben Anordnung von Knoten mehrfach durchgeführt und ergab zuverlässig das in Abb. 3.7 veranschaulichte Ergebnis. Dieses Experiment ist deshalb sehr interessant, weil alle Typen von Links vorkommen: Neben drei symmetrischen Kommunikationslinks wurde zwischen Knoten 7 und Knoten 16 ein asymmetrischer Link erkannt, d.h. Knoten 7 kann Rahmen von Knoten 16 zuverlässig empfangen, in umgekehrter Richtung empfängt Knoten 16 aber Rahmen von Knoten 7 nicht zuverlässig. Dies entsteht möglicherweise dadurch, dass Knoten 7 direkt hinter einem PC-Gehäuse aus Metall platziert wurde, welches das von ihm gesendete Signal offenbar etwas abschirmt. Darüber hinaus wurden zwei symmetrische Interferenzlinks sowie zwei symmetrische Sensing-Links erkannt.

Um die erkannte Topologie zu validieren, wurden weitere Experimente durchgeführt. Zur Überprüfung der Kommunikationslinks wurde ein Experiment durchgeführt, bei dem je ein Knoten Rahmen sendet und der andere Knoten zählt, wie viele Rahmen er korrekt bzw. korrupt empfängt. Dieses Experiment lief für 5 Minuten und währenddessen wurden 48.100 Rahmen mit

einer Rahmengröße von 120 Bytes gesendet. Dadurch, dass die Anzahl gesendeter Rahmen bekannt ist, lässt sich die Anzahl verlorener Rahmen aus der Anzahl empfangener Rahmen bestimmen. Die Ergebnisse dieses Experiments sind in Tab. 3.1 zusammengefasst. Durchschnittlich gehen nur 0,012 % der Rahmen verloren oder werden verfälscht. Der Link $16 \rightarrow 13$ kommt mit 35 verlorenen oder verfälschten Rahmen (0,073%) auf die schlechtesten Werte, weißt allerdings damit immer noch eine hervorragende Zuverlässigkeit auf. Die Ergebnisse zeigen, dass ATDP mit dem gewählten Grenzwert $minRSS_{comm} = -82dBm$ bei der verwendeten Hardware tatsächlich nur sehr zuverlässige Links als Kommunikationslinks einstuft.

Tabelle 3.1: Validierung von Kommunikationslinks.

Experiment	Korrekt empfangen	Rahmenverlust	Verfälschung
$7 \rightarrow 8$	48.000		
$8 \rightarrow 7$	48.100		
$8 \rightarrow 16$	48.100		
$13 \rightarrow 14$	48.097	2	1
$13 \rightarrow 16$	48.100		
$14 \rightarrow 13$	48.089	4	7
$16 \rightarrow 7$	48.099		1
$16 \rightarrow 8$	48.100		
$16 \rightarrow 13$	48.065	7	28

Die Evaluierung der Interferenzlinks ist aufwendiger: Hier soll überprüft werden, ob durch diesen Link tatsächlich Störungen auftreten können. Zu betrachten sind dabei immer drei Knoten:

- *Sender*: Ein Knoten, der versucht, Rahmen über einen Kommunikationslink an den Empfänger zu senden.
- *Empfänger*: Ein Knoten, der über einen Kommunikationslink Rahmen des Senders empfangen soll.
- *Störer*: Ein Knoten, der einen Interferenzlink zum Empfänger hat.

Untersucht wird, ob es beim Empfänger zu Störungen kommt, wenn der Störer zeitgleich mit dem Sender einen Rahmen sendet. Für jede mögliche Kombination aus Sender, Empfänger und Störer wurde ein Experiment durchgeführt, welches untersuchen soll, ob der Störer in dieser Konstellation den Rahmenempfang über den Kommunikationslink stören kann. Um zu erreichen, dass Sender und Störer möglichst gleichzeitig senden, wurden diese durch BBS Synchronisation synchronisiert. Beide Knoten sendeten jeweils gleichzeitig für 5 Minuten 48.100 Rahmen mit je 120 Bytes Größe. Der Empfänger protokollierte die Anzahl korrupter Rahmen, die Anzahl korrekt vom Sender empfangener Rahmen sowie die Anzahl korrekt vom Störer empfangener Rahmen. Da die Anzahl gesendeter Rahmen bekannt ist, lässt sich die Anzahl verlorener Rahmen wieder leicht berechnen. Die Ergebnisse fasst Abb. 3.8 zusammen.

In den Ergebnissen fallen einige Links auf, bei denen fast alle Rahmen des Senders korrekt empfangen wurden und der Störer nicht in der Lage war, den Empfang zu stören. Beispielsweise wurden Übertragungen von Knoten 13 zu Knoten 16 kaum durch Knoten 7 gestört ($13 \rightarrow 16 \not\leftarrow 7$). Gleiches gilt für $7 \rightarrow 8 \not\leftarrow 13$ und $16 \rightarrow 13 \not\leftarrow 8$. Allerdings zeigen die Ergebnisse andere Kombinationen, in denen diese Interferenzlinks $7 \rightarrow 16$, $13 \rightarrow 8$ sowie $8 \rightarrow 13$ zu Störungen führen: Über

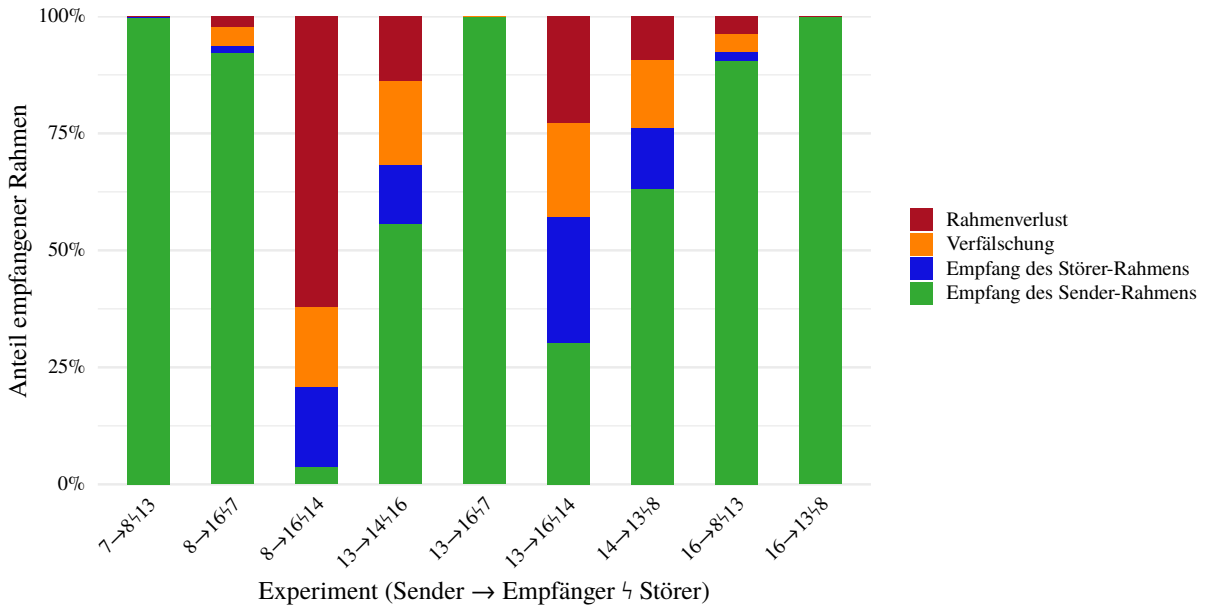


Abbildung 3.8: Ergebnisse der Experimente zur Validierung der erkannten Interferenzlinks

den Interferenzlink $7 \to 16$ stört Knoten 7 zwar Übertragungen von 13 an 16 kaum ($13 \to 16 \not\downarrow 7$), Übertragungen von Knoten 8 hingegen schon ($8 \to 16 \not\downarrow 7$), hier kommen noch ca. 92 % der Übertragungen korrekt an, in knapp 2 % der Fälle wird der Rahmen vom Störer (Knoten 7) statt der des eigentlichen Senders (Knoten 8) empfangen, knapp 4 % werden verfälscht empfangen und weitere 2 % gehen verloren. Der Interferenzlink $13 \to 8$ stört zwar Übertragungen von Knoten 7 kaum ($7 \to 8 \not\downarrow 13$), jene von Knoten 16 werden aber deutlich gestört ($16 \to 8 \not\downarrow 13$), sodass nur ca. 90 % der Nachrichten von Knoten 16 korrekt empfangen werden. Noch stärkere Einflüsse hat der Interferenzlink in der entgegengesetzten Richtung ($8 \to 13$): Er stört zwar Übertragungen von Knoten 16 kaum ($16 \to 13 \not\downarrow 8$) , jene von Knoten 14 kommen aber nur noch zu 63 % an ($14 \to 13 \not\downarrow 8$). Der Interferenzlink zwischen Knoten 16 und 14 führt in beiden Richtungen zu noch größeren Störungen: Übertragungen von Knoten 8 zu Knoten 16 kommen gar nur noch zu knapp 4 % an, wenn Knoten 14 zeitgleich sendet ($8 \to 16 \not\downarrow 14$), hier gehen fast 80 % der Nachrichten entweder verloren oder werden verfälscht.

Insgesamt zeigt sich, dass jeder der von ATDP gefundenen Interferenzlinks Störungen auf einem der erkannten Kommunikationslink verursacht. Offensichtlich hängt es neben dem Störer und Empfänger auch vom Sender ab, ob ein Interferenzlink Störungen bewirkt. Es wäre daher theoretisch wünschenswert, wenn ein Verfahren zur Topologieerkennung nicht nur wie ATDP Interferenzlinks als Paar aus Störer und Empfänger betrachtet, sondern erkennt, welche Tripel aus Störer, Empfänger und Sender zu Störungen führen. Anhand dieser Information könnte man eine noch effizientere Ausnutzung des Mediums per SDMA erreichen. Allerdings ist es kaum möglich, diese Information bei größeren Netzwerken effizient zu erkennen und auszutauschen. Das Problem ist, dass es bei n Knoten im Netzwerk $\mathcal{O}(n^2)$ mögliche Kombinationen aus Störer und Empfänger gibt, aber $\mathcal{O}(n^3)$ Kombinationen aus Störer, Empfänger und Sender. Dies bedeutet zum einen, dass der Aufwand zum Austausch der erkannten Interferenzlinks deutlich höher zu erwarten ist, da es nun deutlich mehr solche Links gibt. Darüber hinaus macht dies allerdings auch den Erkennungsprozess deutlich langsamer: Zur Erkennung der Interferenzlinks muss bei ATDP jeder Knoten eine gewisse Anzahl Rahmen senden, und alle anderen Knoten hören, was einem Aufwand von $\mathcal{O}(n)$ entspricht. Zur Erkennung der Tripel aus Störer, Empfänger

und Sender müsste man, ähnlich wie bei den hier durchgeführten Validierungs-Experimenten, Kombinationen aus Sender und Störer gleichzeitig senden lassen. Würde man dies mit allen Kombinationen tun, würde der Aufwand auf $\mathcal{O}(n^2)$ steigen. Einen solchen Ansatz verfolgt beispielsweise der in Kapitel 3.2.1 bereits erwähnte Ansatz von Padhye et al. [Pad+05]. Nicht nur der Aufwand dieses Ansatzes ist problematisch. Auch ist denkbar, dass kleine Änderungen, wie beispielsweise das Öffnen einer Tür oder eines Fensters, das Interferenzverhalten stark verändern: Dadurch, dass sich beispielsweise Reflektionen des Signals nun anders verhalten, kann der Mehrwegeempfang (engl.: Multipath propagation) derart verändert werden, dass nun statt der Rahmen des Senders die des Störers empfangen werden, oder gar kein Rahmen mehr fehlerfrei empfangen wird. Aus diesen Gründen wurde in ATDP darauf verzichtet, Interferenzlinks als Tripel aufzufassen.

Tabelle 3.2: Validierung von Sensing-Links

Experiment	Korrekt empfangen	Rahmenverlust	Verfälschung
$13 \rightarrow 14 \not\downarrow 7$	48.100		
$16 \rightarrow 13 \not\downarrow 7$	48.100		
$14 \rightarrow 13 \not\downarrow 7$	48.092	2	6
$8 \rightarrow 7 \not\downarrow 13$	48.100		
$16 \rightarrow 7 \not\downarrow 13$	48.063	12	25
$8 \rightarrow 7 \not\downarrow 14$	48.100		
$16 \rightarrow 7 \not\downarrow 14$	48.099	1	

Schließlich wurden ähnliche Experimente durchgeführt, um zu überprüfen, ob Sensing-Links tatsächlich keine Störungen von gleichzeitigen Übertragungen über Kommunikationslinks verursachen können. Dazu wurden die selben Experimente wie für Interferenzlinks mit den erkannten Sensing-Links durchgeführt, also für eine Dauer von 5 Minuten je 48.100 Rahmen mit einer Rahmengröße von 120 Bytes von je zwei Knoten gesendet und der Empfang protokolliert. Die Ergebnisse dieser Experimente fasst Tab. 3.2 zusammen. Hier zeigt sich deutlich, dass die als Sensing-Links erkannten Links nicht in der Lage sind, die Übertragungen auf den erkannten Kommunikationslinks zu stören: In allen Experimenten sind keine oder nur sehr wenige Rahmen verloren gegangen oder wurden verfälscht. In keinem Fall wurden Rahmen vom Störer empfangen.

3.2.4 Zusammenfassung

In diesem Kapitel wurde die Bedeutung von Topologieinformation für drahtlose Netzwerke erläutert und die Unterschiede zwischen Kommunikations-, Interferenz- und Sensing-Topologie aufgezeigt. Routing- und Clustering-Protokolle benötigen die Kommunikationstopologie zur Bildung von Routen bzw. Clustern. SDMA erlaubt es, das Medium effizienter zu nutzen, erfordert aber die Interferenztopologie. Schließlich existieren Protokolle, welche dank ihrer Black Burst-Kodierung auf Sensing-Links arbeiten können, und daher die Sensing-Topologie erfordern. Bei der Betrachtung der vorhandenen Verfahren stellte sich zunächst heraus, dass es kaum eigenständige Verfahren zur Topologieerkennung gibt, welche ihre Information anderen Protokollen zur Verfügung stellen. Vielmehr enthalten die allermeisten Routing- und Clustering-

Verfahren ihre eigenen Mechanismen zur Topologieerkennung. Wie sich gezeigt hat, haben diese aber oft Schwächen, insbesondere weil Kommunikationslinks auf Basis eines einzelnen Empfangs erkannt werden und damit auch unzuverlässige Links Berücksichtigung finden. Darüber hinaus werden Links oft als symmetrisch angenommen, was bei drahtlosen Netzwerken nicht der Fall sein muss, wie auch das Evaluations-Kapitel gezeigt hat. Existierende Verfahren zur Erkennung der Interferenztopologie sind entweder sehr aufwendig oder basieren auf Annahmen, die in der Praxis nur unter Einschränkungen zu erreichen sind. Existierende Protokolle zur Erkennung des Sensing-Topologie sind nicht bekannt.

Mit dem Automatic Topology Discovery Protocol (ATDP) wurde ein effizientes Protokoll zur Topologieerkennung vorgestellt, welches die Kommunikations-, Interferenz- und Sensing-Topologie eines drahtlosen TDMA-basierten Netzwerks ermittelt. Es berücksichtigt nicht nur einen einzelnen Empfang bei der Bewertung der Links und kann so die Zuverlässigkeit der Links unterscheiden. Die Evaluation zeigt, dass eine Implementierung von ATDP in der Praxis funktioniert. Anhand von Experimenten wurde nachgewiesen, dass ATDP zuverlässige Kommunikationslinks auswählt, die gefundenen Interferenzlinks tatsächlich Störungen verursachen können, Sensing-Links dagegen keine Störungen verursachen können. Das Protokoll verteilt darüber hinaus die erkannten Topologieen netzweit, ohne damit einen hohen Overhead zu erzeugen.

3.3 QoS Multicast Routing

In einem großflächigen Netzwerk wie etwa in einer Fabrik sind die Entfernungen zwischen einigen Knoten oft deutlich größer als die Reichweite des drahtlosen Kommunikationssystems. Da keine direkte Kommunikation aller Knoten möglich ist, müssen Nachrichten oft von Zwischenknoten weitergeleitet werden. Ein Routing-Verfahren wird benötigt, um in einem solchen Multihop-Netzwerk zu ermitteln, über welche Zwischenknoten Nachrichten weitergeleitet werden können, um den Zielknoten effizient zu erreichen. In Kapitel 3.2 wurde auf die Erkennung der Kommunikationstopologie eingegangen. Das dort beschriebene Automatic Topology Discovery Protocol (ATDP) erkennt und verteilt die Kommunikationstopologie. Damit ist allen Knoten die vollständige Kommunikationstopologie bekannt, sodass Routen einfach gefunden werden können, indem beispielsweise mit dem Dijkstra-Algorithmus [Dij59] der kürzeste Pfad berechnet wird.

Um in einem TDMA-basierten Netzwerk zuverlässige Übertragungen garantieren zu können, müssen aber auch Reservierungen für jeden Hop der Route vorhanden sein. Zeitslots müssen in TDMA-basierten Netzwerken allerdings nicht immer exklusiv reserviert werden, um eine zuverlässige Kommunikation zu garantieren. Falls zwei Übertragungen räumlich so weit auseinander liegen, dass es nicht zu Kollisionen durch Interferenz kommen kann, ist eine Mehrfachreservierung möglich. Diese Verknüpfung von TDMA mit SDMA ermöglicht eine effizientere Ausnutzung des Mediums. Die dafür benötigte Interferenztopologie wird ebenfalls von ATDP ermittelt. Hat die Anwendung weitere Quality of Service (QoS) Anforderungen, wie beispielsweise eine maximale Verzögerung, so müssen die genutzten Zeitslots darüber hinaus diesen Anforderungen genügen. Sinnvollerweise führt ein Routing-Verfahren für TDMA-Netzwerke daher auch Slotreservierungen für die gefundenen Routen aus und berücksichtigt dabei die Interferenztopologie sowie QoS-Anforderungen.

In einem Netzwerk zur Prozessautomatisierung, wie es in Kapitel 2.2 beschrieben wurde, kommunizieren Sensoren, Aktuatoren und Regler miteinander. Dabei kommt es häufig vor, dass

einzelne Dienste von mehreren Knoten genutzt werden. Beispielsweise könnte ein Regler redundant ausgelegt sein, sodass immer beide Knoten den gleichen Sensorwert benötigen. In solchen Fällen ist es effizienter, die Nachricht mit einer Multicast-Übertragung zu verteilen, als zu jedem Empfänger eine Unicast-Übertragung durchzuführen. In solchen Fällen ist es also wünschenswert, wenn das Routing-Verfahren Multicast-Übertragungen unterstützt.

Auch wenn in einer Fabrik die meisten Knoten stationär sind, kann es in solchen Anwendungsszenarien oft einzelne Knoten geben, die sich bewegen können. Beispielsweise können Knoten auf Robotern angebracht sein, die sich frei bewegen können. Ein Routing-Verfahren für dieses Anwendungsszenario sollte also optimalerweise einerseits ausnutzen können, dass die meisten Knoten im Netz stationär sind, andererseits aber auch mobile Knoten unterstützen.

In diesem Kapitel betrachten wir zunächst existierende Routing-Verfahren in Kapitel 3.3.1. Anschließend wird das zuerst in [Geb+15] publizierte Routing-Verfahren *QoS Multicast Routing Protocol (QMRP)* vorgestellt, welches speziell auf die eingangs genannten Anforderungen ausgelegt ist. Kapitel 3.3.2 beschreibt zunächst, wie das Verfahren in rein stationären Netzwerken arbeitet. Anschließend erweitert Kapitel 3.3.3 das Verfahren um eine Unterstützung für mobile Knoten. Die Implementierung und Evaluierungsergebnisse präsentiert Kapitel 3.3.4 und Kapitel 3.3.5 fasst das Kapitel zusammen.

3.3.1 Stand der Technik

In Kapitel 3.2.1.1 wurden bereits einige Routing-Protokolle in Hinblick auf Topologieerkennung diskutiert, darunter DSDV [PB94], AODV [PR99], Babel [Chr11] und OLSR [CJ03]. Diese Protokolle unterstützen Multicast-Übertragungen nicht oder nur unzureichend. Für manche der Protokolle wurden Erweiterungen für Multicast-Übertragungen vorgeschlagen. So erweitern Royer und Perkins in [RP99] AODV um Unterstützung für Multicast-Übertragungen und in [Lao+03] stellen Laouiti et al. mit MOLSR eine OLSR-Variante mit Multicast-Unterstützung vor. Außerdem gibt es eine Reihe weiterer Routing-Protokolle mit Multicast-Unterstützung [GM99; LSG02]. Allerdings sind diese Protokolle für das genannte Anwendungsszenario nicht geeignet, hauptsächlich, da sie keine QoS-Anforderungen unterstützen. Andere Routing-Protokolle [ZM05] konzentrieren sich auf QoS-Unterstützung, bieten aber keine Multicast-Übertragungen. Nur wenige Routing-Protokolle bieten sowohl QoS- als auch Multicast-Unterstützung. Beispiele sind das Hexagonal-Tree-Routing Protocol [CLL07] und PSLCB [ZL13]. Diese Protokolle nehmen allerdings an, dass die Interferenzreichweite nicht größer ist als die Kommunikationsreichweite [Geb14]. Diese Annahme ist in der Praxis nicht immer gegeben, sodass es zu Kollisionen kommen kann. Andere QoS Multicast Routing-Protokolle wie [CKL02] und [WJ07] erreichen zuverlässige Übertragungen, indem sie TDMA mit CDMA kombinieren. Allerdings berücksichtigen auch diese Protokolle nicht, dass die Interferenzreichweite größer als die Kommunikationsreichweite sein kann. Der Einsatz von CDMA erfordert darüber hinaus, dass Codes berechnet und ausgetauscht werden, was einen zusätzlichen Overhead erzeugt.

Keines der genannten Protokolle kann den von einem Protokoll zur Topologieerkennung wie ATDP bereitgestellten Topologiestatus verwenden, stattdessen bringen die genannten Protokolle alle ihre eigene Topologieerkennung mit. Wünschenswert für unseren Stack wäre dagegen ein Protokoll, was sich auf QoS- und Multicast-Unterstützung konzentriert und die Topologie von ATDP übernimmt. Dies hat auch den Vorteil, dass die von ATDP erkannte Topologie von anderen Verfahren, wie etwa zum Clustering (s. Kapitel 3.4), ebenfalls verwendet werden kann.

Die meisten Routing-Protokolle für drahtlose Netzwerke, wie z.B. AODV, bringen auch Unterstützung für mobile Knoten mit. Dabei behandeln sie aber i.d.R. alle Knoten gleich, d.h. erlauben allen Knoten, sich jederzeit zu bewegen. Dies erzeugt einen unnötigen Overhead für die unbeweglichen Knoten. Reservierungsprotokolle wie [Shi+06], die TDMA-Zeitslots reservieren können, gehen hingegen meist davon aus, dass die Topologie stabil ist, sich also keiner der Knoten bewegen darf. Dies erfordert dann aufwendige Korrekturen an den Reservierungen, was zu kurzzeitigen Ausfällen führt. Wünschenswert für unser Anwendung wäre dagegen ein Protokoll, das für die stationären Knoten exklusive Reservierungen vornehmen und ebenso mobile Knoten zuverlässig unterstützen kann.

3.3.2 QoS Multicast Routing in stationären Netzwerken

Das hier vorgestellte *QoS Multicast Routing Protocol (QMRP)* ist ein Routing-Protokoll, das speziell auf die eingangs erwähnten Anforderungen hin entwickelt wurde. Das bedeutet, es hat folgende Eigenschaften:

- Unterstützung für Multicast-Übertragungen
- Berücksichtigung von QoS-Anforderungen, beispielsweise bezogen auf die maximale Verzögerung und Durchsatz
- Reservierung von TDMA-Zeitslots unter Berücksichtigung der Interferenztopologie
- Dynamische Anpassung von Routen und Reservierungen zur Laufzeit, insbesondere Erweiterung von Multicast-Routen um weitere Empfänger
- Unterstützung für mobile Knoten

Wir betrachten in diesem Kapitel zuerst eine Variante von QMRP, welche nur stationäre Netzwerke unterstützt. Diese wird im Kapitel 3.3.3 dann um Unterstützung für mobile Knoten erweitert.

Beim Entwurf eines Protokolls wie QMRP ist ein wichtiger Aspekt die Entscheidung, ob das Protokoll zentralisiert oder dezentral entworfen werden soll. Eine zentralisierte Lösung verwendet einen Masterknoten, der Aufgaben wie Routing- und Slotreservierung zentral übernimmt. Bei einer dezentralen Lösung tauschen alle Knoten Informationen aus, um dezentral Routen- und Slotreservierungen zu ermitteln. Beide Varianten haben ihre Vor- und Nachteile. Zentralisierte Lösungen führen immer einen Single-Point-of-Failure ein, da das gesamte System ausfällt, wenn der zentrale Masterknoten ausfällt. Dieser Nachteil kann abgemildert werden, indem dieser Masterknoten besonders zuverlässig ausgelegt wird, beispielsweise mit einer unterbrechungsfreien Stromversorgung oder komplett redundant. Ein Vorteil einer zentralisierten Lösung ist, dass es einfacher ist, den Überblick über den Status des Netzwerks zu behalten und optimale Lösungen zu berechnen. Dezentrale Lösungen können immer nur auf Teilinformationen zugreifen und sind fehleranfällig, wenn Reservierungen benötigt werden. Konkurrierende Reservierungsanfragen von mehreren Knoten, die parallel berücksichtigt werden, können leicht zu Fehlern wie Mehrfachreservierungen führen. Das Problem ist noch komplexer, wenn man berücksichtigt, dass die Interferenzreichweite größer als die Kommunikationsreichweite ist. Dann müssen nämlich Reservierungen ggf. mit Knoten abgesprochen werden, die nicht in Kommunikationsreichweite liegen. Außerdem erzeugt die Absprache zwischen den Knoten zu einem weiteren Kommunikations-Overhead. Da wir in ProNet 4.0 im ProSync-Modul das Master-basierte BBS-Protokoll als Protokoll zur Zeitsynchronisation verwenden, haben wir ohnehin bereits einen Single-Point-of-Failure. Daher haben wir uns dafür entschieden, den Timing-Master auch als Master für Routing und Scheduling mit QMRP zu verwenden.

Im zentralisierten Entwurf müssen Multicast-Routen zunächst beim Masterknoten angefragt werden. Um diese Anfragen auch zuverlässig übertragen zu können, werden jedem Knoten exklusive Management-Slots für QMRP-Anfragen zugeordnet. Das Routing dieser Anfragen erfolgt auf dem kürzesten Pfad, indem jeder Knoten mit Hilfe des Dijkstra-Algorithmus den nächsten Knoten in Richtung des Masterknotens berechnet. Dabei kommt lediglich die von ATDP erkannte Kommunikationstopologie zum Einsatz. Auf diese Weise kommen die Anfragen streng serialisiert beim Masterknoten an, sodass es keine konkurrierenden zeitgleichen Anfragen gibt. Der Masterknoten erstellt oder erweitert den Multicast-Baum. Da das Problem der optimalen Routenfindung und Reservierung im Allgemeinen NP-vollständig und damit von hoher Komplexität ist, verwendet QMRP geeignete Heuristiken. Das berechnete Ergebnis gibt der Masterknoten an die beteiligten Knoten wieder über Management-Slots zurück.

Zunächst betrachten wir in Kapitel 3.3.2.1, wie QMRP Multicast-Bäume in stationären Netzwerken erstellt und erweitert. Die Reservierung von TDMA-Zeitslots beschreibt anschließend Kapitel 3.3.2.2.

3.3.2.1 Ermittlung von Multicast Routing-Bäumen

Routing-Bäume werden in QMRP dynamisch erzeugt und erweitert. Das bedeutet, dass nicht alle Empfänger einer Multicast-Route zu Beginn bekannt sein müssen. Vielmehr muss zum Erzeugen eines Baumes zunächst lediglich der Quellknoten s und der erste Zielknoten d_1 bekannt sein. Der von QMRP erzeugte Baum t besteht dann auch nur aus einem Pfad p_{s,d_1}, der einer Unicast-Route von s zu d_1 entspricht. Auch wenn es sich in diesem Fall zunächst nur um einen einfachen Pfad handelt, begreifen wir dies als Spezialfall eines Baumes und sprechen daher auch von einem Baum. Benötigt ein weiterer Empfänger d_2 die gleiche Übertragung von s, so wird der Baum t um einen weiteren Ast zu d_2 ergänzt.

Bei der Auswahl der genutzten Pfade kommt eine Reihe von Heuristiken zum Einsatz. Grundlegend basiert QMRP auf der Shortest-Path-Metrik, d.h. es versucht, möglichst kurze Routen zu finden. Kurze Routen benötigen weniger Übertragungen und nutzen das Medium daher effizienter. Nachteilig an der Shortest-Path-Metrik ist, dass sie dazu tendiert, Links auszuwählen, die eine weite Entfernung zwischen zwei Knoten überbrücken. Diese Links sind bei drahtloser Kommunikation oft unzuverlässiger als Links, die nur eine kurze Entfernung überbrücken. Daher tendiert die Shortest-Path-Metrik dazu, unzuverlässige Links zu bevorzugen. Wir gehen hier allerdings davon aus, dass wir mit einem geeigneten Topologieerkennungsverfahren, wie etwa ATDP, die Zuverlässigkeit der Links überprüft haben und überhaupt nur solche Links berücksichtigen, welche eine hohe Zuverlässigkeit aufweisen.

Beim Erzeugen eines neuen Baumes berechnet QMRP also den *kürzesten Pfad* von der Quelle s zum ersten Zielknoten d_1. Die dazu benötigte Kommunikationstopologie wurde zuvor durch ein Verfahren zur Topologieerkennung, wie etwa ATDP, erkannt und verteilt. Die Berechnung erfolgt auf dem Masterknoten mit Hilfe des Dijkstra-Algorithmus. Falls es mehrere kürzeste Pfade gibt, so wählt ATDP jenen Pfad aus, der die *größte Nachbarschaft* hat.

Mit der *Nachbarschaft eines Pfades p* bezeichnen wir die Menge Knoten, die direkte Kommunikationsnachbarn von Knoten des Pfades sind. Ein Beispiel dafür zeigt Abb. 3.9: Die Pfade p_1, p_2 und p_3 haben alle die gleiche Länge von 3 Hops, allerdings ist die Nachbarschaft von p_1 am größten, da sie alle Knoten enthält, die auch die Nachbarschaften von p_2 und p_3 enthalten, zusätzlich aber noch Knoten 6.

Formal: Seien V die Knoten des Netzwerks und $v \in V$ einer dieser Knoten. Dann bezeichnen wir mit $CN(v)$ die Menge der Kommunikationsnachbarn von v, d.h. alle Knoten $v' \in V$, die einen

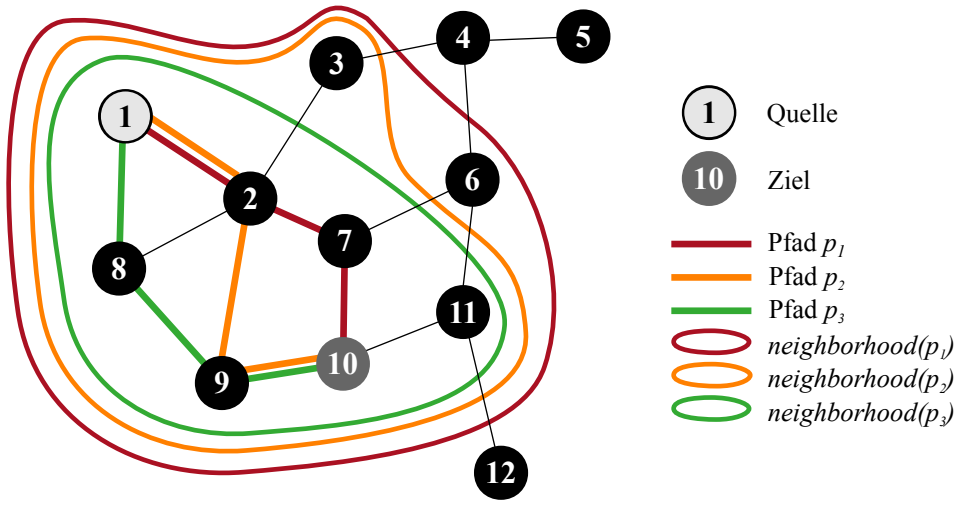


Abbildung 3.9: Drei Pfade vom Quellknoten 1 zum Zielknoten 10 mit der gleichen Länge von 3 Hops, aber unterschiedlich großen Nachbarschaften.

symmetrischen Kommunikationslink zu v haben. Der Pfad p sei definiert als Menge von Knoten, durch die er läuft. Dann definieren wir die Nachbarschaft von p wie folgt:

$$neighborhood(p) \stackrel{\text{def}}{=} \bigcup_{v \in p} CN(v) \tag{3.2}$$

Ein Pfad mit einer großen Nachbarschaft verläuft i.d.R. durch dichtere Teile des Netzwerks. Weniger dichte Bereiche, wie etwa am Rand des Netzwerks, haben dagegen i.d.R. eine kleinere Nachbarschaft. In Abb. 3.9 läuft beispielsweise Pfad p_3 am Rand des Netzwerks und p_1 durch das Zentrum.

Die Heuristik in QMRP bevorzugt Pfade mit einer möglichst großen Nachbarschaft, da dies die Wahrscheinlichkeit erhöht, dass künftige Erweiterungen des Baumes um weitere Zielknoten geringer ausfallen. Würde im Beispiel in Abb. 3.9 Knoten 6 hinzugefügt werden sollen, so ist dies bei Pfad p_1 mit einem zusätzlichen Link möglich, wohingegen bei den Pfaden p_2 und p_3 zwei Links hinzugefügt werden müssten. Ein möglicher Nachteil dieser Heuristik ist, dass in dichten Teilen des Netzwerks ggf. weniger Übertragungsslots frei sind. Diese Heuristik führt also u.U. in stark ausgelasteten Netzwerken dazu, dass gar kein Pfad gefunden wird, der die geforderten Anforderungen erfüllt, weil die erforderlichen Slots bereits belegt sind. In diesem Fall wäre ein denkbarer Ansatz, die Routensuche erneut durchzuführen, und dabei beispielsweise Pfade mit der kleinsten Nachbarschaft zu bevorzugen. Wir sehen davon jedoch ab, da das Netz in derartigen Fällen bereits so stark ausgelastet ist, dass die Chancen, einen anderen Pfad zu finden, der die geforderten Anforderungen erfüllt, eher gering sind.

Die Erweiterung eines existierenden Baums um einen weiteren Zielknoten folgt ähnlichen Heuristiken. Zum einen soll die Gesamtanzahl benutzter Links möglichst gering bleiben, da so die Anzahl benötigter Übertragungen und damit die benötigte Bandbreite minimiert wird. Darüber hinaus soll die Anzahl Links von der Quelle zum neuen Zielknoten gering gehalten werden, da dies i.d.R. die Übertragungsverzögerung minimiert. Die Verzögerung hängt allerdings letztendlich auch stark von den genutzten Zeitslots ab (s. Kapitel 3.3.2.2).

Beim Erweitern eines existierenden Baumes t um einen weiteren Zielknoten d' berechnet QMRP zunächst den *kürzesten Pfad* $p_{v,d'}$ für jeden Knoten v des Baums t zum Zielknoten d'. Falls es meh-

rere solcher Pfade gibt, berechnet QMRP die gesamte Pfadlänge des Pfades $p_{v,d'} = p_{s,v} \bullet p_{v,d'}$ bestehend aus dem Teil des ursprünglichen Baumes von der Quelle s zum Zwischenknoten v und den neuen Ast zum Zielknoten d'. Mit $\bullet$ bezeichnen wir hier die Konkatenation von Pfaden. Der kürzeste Pfad wird gewählt, um die Übertragungsverzögerung klein zu halten. Falls weiterhin mehrere Pfade möglich sind, wird der Pfad mit der größten Nachbarschaft ausgewählt. Der so ausgewählte Pfad $p_{v,d'}$ wird schließlich dem Baum t angefügt.

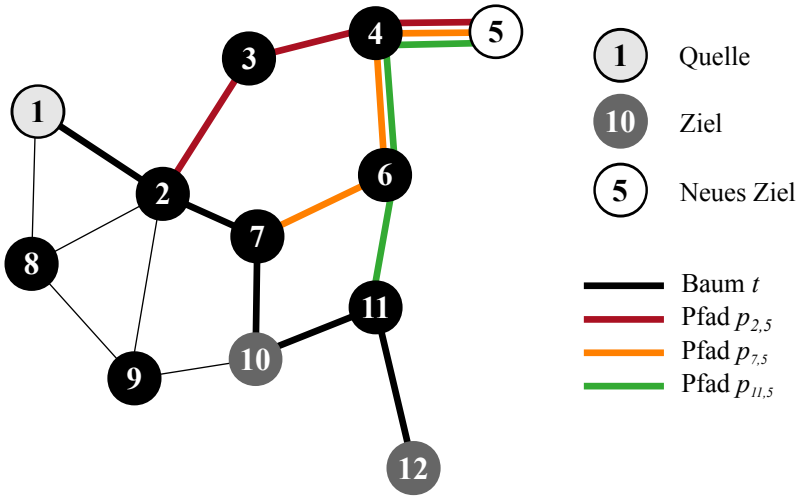


Abbildung 3.10: Erweiterung des existierenden Baums t von der Quelle 1 zu den Zielen 10 und 12 um das neue Ziel 5.

Die Erweiterung eines Baumes um ein neues Ziel illustriert Abb. 3.10 an einem Beispiel. Der existierende Baum t mit dem Quellknoten 1 und den beiden Zielen 10 und 12 soll um das neue Ziel 5 erweitert werden. Zunächst wird für alle Knoten des existierenden Baumes der kürzeste Pfad zum neuen Ziel 5 berechnet. Die kürzesten Pfade sind jene in Abb. 3.10 eingezeichneten, die an den Knoten 2, 7 und 11 abzweigen. Alle drei Pfade fügen drei zusätzliche Links zum Baum hinzu. Von der Quelle 1 zum neuen Ziel 5 ist die Variante über den beim Knoten 2 abzweigenden Pfad $p_{2,5}$ offensichtlich am kürzesten, sodass diese ausgewählt wird.

3.3.2.2 Reservierung von TDMA-Zeitslots

QMRP ist ausgelegt für TDMA-Netzwerke, die Zuverlässigkeit durch reservierte Zeitslots erreichen. Bisher wurde gezeigt, wie QMRP Routing-Bäume erzeugt und erweitert. Dabei wurde die Reservierung von TDMA-Zeitslots zunächst außer Acht gelassen. Nun soll das Verfahren um ein Scheduling-Verfahren erweitert werden, was Zeitslots für die Routing-Bäume reserviert.

Wir gehen davon aus, dass das TDMA-Kommunikationssystem Zeit in n gleichgroße Zeitslots $s_0, ..., s_{n-1}$ einteilt, die zusammen einen Superslot S bilden, der periodisch wiederholt wird. In einem Zeitslot s_i mit $i \in [0, n-1]$ kann genau ein Rahmen vom Sender-Knoten v_{si} zum Empfänger-Knoten v_{ri} sowie das dazugehörige Acknowledgement gesendet werden. Um die Zuverlässigkeit der Übertragungen zu gewährleisten, dürfen Übertragungen nicht durch Interferenzen zerstört oder verfälscht werden. Dies lässt sich am einfachsten realisieren, indem Zeitslots netzweit exklusiv vergeben werden, sodass zu jeder Zeit nur ein Knoten im Netzwerk senden darf.

Formal kann man dies wie folgt als *Reservierungskriterium für exklusive Slotreservierung* formulieren: Ein Slot s darf für eine Übertragung vom Sender v_s zum Empfänger v_r reserviert werden, wenn $E^s(v_s, v_r)$ erfüllt ist:

$$E^s(v_s, v_r) \stackrel{\text{def}}{=} \forall v' \in V : \neg TX^s(v') \tag{3.3}$$

Dabei sei $TX^s \subseteq V$ der *Sende-Reservierungsstatus*, d.h. die Menge der Knoten $v \in V$, denen im Zeitslot s ein Zeitslot zum Senden zugeordnet ist.

In größeren Netzwerken ist es aber möglich, dass Knoten, die weit genug voneinander entfernt sind, gleichzeitig senden, ohne dass es zu Kollisionen kommt. Durch diese Kombination von TDMA mit SDMA kann die Gesamtbandbreite des Netzwerks gesteigert werden und ggf. auch Verzögerungen verringert werden. QMRP nutzt dazu die von ATDP ermittelte Interferenztopologie. Da in einem Zeitslot s_i sowohl der Sender v_{si} die eigentliche Nachricht als auch der Empfänger v_{ri} das Acknowledgement sendet, sind beide Knoten im Zeitslot s_i sowohl Sender als auch Empfänger. Daher dürfen Knoten in Interferenzreichweite von beiden Knoten v_{si} und v_{ri} weder zeitgleich senden noch empfangen, um Kollisionen ausschließen zu können.

Wir definieren dieses *SDMA-Reservierungskriterium* nun formal. Dazu definieren wir zunächst analog zur Kommunikationsnachbarschaft $CN(v)$ die *Interferenznachbarschaft* $IN(v)$ des Knotens v als die Menge der Knoten, die in Interferenzreichweite von v liegen, d.h. für alle $v' \in IN(v)$ existiert ein Interferenzlink von v zu v'. Es wird vorausgesetzt, dass die Interferenzlinks durch ein Topologieerkennungsverfahren wie ATDP ermittelt wurden, sodass die Interferenznachbarschaft eines jeden Knoten daraus bestimmt werden kann. Weiterhin sei der *Sende-Reservierungsstatus* $TX^s \subseteq V$ wie zuvor die Menge der Knoten $v \in V$, denen im Zeitslot s ein Zeitslot zum Senden zugeordnet ist. Analog beschreibe der *Empfangs-Reservierungsstatus* $RX^s \subseteq V$ die Menge der Knoten $v \in V$, denen im Zeitslot s ein Zeitslot zum Empfang zugeordnet ist. Dann definieren wir das Reservierungskriterium $F^s(v_s, v_d)$, das angibt, ob Slot s für eine Übertragung vom Sender v_s zum Empfänger v_r reserviert werden darf, wie folgt:

$$F^s(v_s, v_r) \stackrel{\text{def}}{=} \forall v' \in IN(v_s) \cup IN(v_r) : \neg TX^s(v') \wedge \neg RX^s(v') \tag{3.4}$$

QMRP unterstützt zwei Scheduling-Strategien mit unterschiedlichen Zielen. Die erste Strategie minimiert die Übertragungsverzögerung. Die zweite Strategie optimiert die Ausnutzung der Slots. Indem bevorzugt Slots verwendet werden, die für wenige andere Routen in Frage kommen, können für mehr Routen Reservierungen vorgenommen werden, sodass das Medium effizienter genutzt wird.

Geringe Übertragungsverzögerung Wir betrachten zunächst die Strategie, welche die Übertragungsverzögerung minimiert. Die geringstmögliche Verzögerung wird erreicht, wenn aufeinanderfolgende Zeitslots reserviert werden. In diesem Fall ergibt sich die Verzögerung aus dem Produkt der Slotdauer und der Tiefe des Routing-Baumes. Im Fall eines Multicast-Baumes ist es allerdings oft nicht möglich, dieses Minimum zu erreichen, da dies erfordern würde, dass die Äste des Baumes parallel nebeneinander reserviert werden können. Dort, wo sich ein Baum in mehrere Äste aufteilt, erfolgen die auf die Abzweigung folgenden Übertragungen aber i.d.R. in Interferenzreichweite voneinander, sodass zeitgleiche Übertragungen zu Kollisionen führen könnten. Neben Interferenzen, die innerhalb eines Baumes durch die einzelnen Äste auftreten können, müssen weiterhin mögliche Interferenzen mit bereits vorhandenen Reservierungen für andere Bäume beachtet werden. Um zu überprüfen, ob eine Übertragung von einem Knoten v_s zu v_r im Slot s möglich ist, ohne dass Kollisionen auftreten können, haben wir bereits das Reservierungskriterium $F^s(v_s, v_r)$ definiert.

Um für einen gegebenen Pfad nun Reservierungen mit minimaler Verzögerung zu erhalten, beginnt man mit dem ersten Link des Pfades, $v_{s,0} \rightarrow v_{r,0}$. Falls es sich bei dem Pfad um eine Erweiterung eines bestehenden Baumes handelt, so startet man die Slotsuche einen Slot nach dem Slot, der für die vorangehende Übertragung im Baum reserviert wurde. Falls es sich um einen neuen Baum handelt, kann man die Suche theoretisch bei einem beliebigen Slot beginnen, also auch stets beim ersten Slot s_0. Dies hätte aber meist größere Übertragungsverzögerungen zur Folge, da versucht wird, alle Übertragungen am Beginn einzuplanen, was bald nicht mehr durch aufeinanderfolgende Slots möglich ist. Stattdessen beginnt QMRP die Suche in diesem Fall einen Slot nach jenem Slot, den es bei der letzten Suche zuletzt vergeben hat. Diese Heuristik soll dafür sorgen, dass öfter aufeinanderfolgende Slots vergeben und damit die Übertragungsverzögerungen minimiert werden. Für die Übertragung auf dem ersten Link sucht man nun vom Startslot s_i beginnend den nächsten Slot $s_{(i+k) \bmod n}$, für den das Reservierungskriterium $F^{s_{(i+k) \bmod n}}(v_{s,0}, v_{r,0})$ erfüllt ist. Den Zähler k erhöht man schrittweise von 0 bis $n-1$. Somit werden beginnend beim Startslot s_i alle n Slots geprüft, wobei beim ersten Slot s_0 fortgefahren wird, falls das Ende des Superslots erreicht wurde. Wurden alle Slots überpüft, d.h. $k = n-1$, und kein freier Slot gefunden, so bricht QMRP das Scheduling ab und beantwortet die Routensuche mit einer Fehlernachricht. Wird hingegen ein Slot $s_{(i+k) \bmod n}$ gefunden, so wird dieser gewählt und das Verfahren fährt mit dem nächsten Hop des Pfads fort. Um eine möglichst geringe Übertragungsverzögerung zu erhalten, wird die Suche dabei einen Slot nach dem zuvor ausgewählten Zeitslot begonnen, und dazu i auf $(k+1) \bmod n$ gesetzt. Dieses Vorgehen beschreibt Algorithmus 3.1 formal als Pseudocode.

```
1  FUNCTION Schedule(p, i): bool
2    // Iteriere durch die Links des Pfads p
3    for l ← 0 to |p| − 1 do
4      // Der l−te Link sei v_{s,l} → v_{r,l}
5      v_{s,l} ← getSender(p,l)
6      v_{r,l} ← getReceiver(p,l)
7      // Suche den nächsten möglichen Slot beginnend bei Slot i
8      k ← 0
9      while k < n − 1 ∧ ¬F^{s_{(i+k) mod n}}(v_{s,l}, v_{r,l})
10       k ← k + 1
11     if F^{s_{(i+k) mod n}}(v_{s,l}, v_{r,l}) then
12       // Es wurde ein Slot gefunden, wähle diesen
13       prereserve(s_{(i+k) mod n})
14       // Beginne die Suche für den nächsten Link beim Slot (k+1) mod n
15       i ← (k+1) mod n
16     else
17       // Es konnte kein Slot gefunden werden
18       return FALSE
19     end
20   end
21   return TRUE
22 end
```

Algorithmus 3.1: Scheduling Strategie zum Erreichen einer geringen Verzögerung

Wir betrachten diese Strategie nun anhand eines kleinen Beispiels. Damit in der kleinen Beispieltopologie überhaupt SDMA möglich ist, wird im Beispiel davon ausgegangen, dass die Interferenzreichweite mit der Kommunikationsreichweite übereinstimmt. QMRP trifft diese Annahme nicht, sondern nutzt die von der Topologieerkennung erkannte Interferenztopologie. In Tab. 3.3

Tabelle 3.3: Schedule für die Übertragung von Knoten 1 zu Knoten 10 in der Topologie aus Abb. 3.9 mit der Strategie zur Minimierung der Übertragungsverzögerung.

Slot:	s_0	s_1	s_2	s_3	s_4
Knoten 1:	TX_2		x		
Knoten 2:	RX_1		TX_7		x
Knoten 3:	x		x		
Knoten 4:	TX_6	x			
Knoten 5:	x				
Knoten 6:	RX_4	TX_{11}	x	x	x
Knoten 7:	x	x	RX_2		TX_{10}
Knoten 8:	x		x		
Knoten 9:	x		x		x
Knoten 10:		x	x	x	RX_7
Knoten 11:	x	RX_6		TX_{12}	x
Knoten 12:		x		RX_{11}	

Legende:

x	Durch eine Übertragung in Interferenzreichweite blockiert
RX_i	Empfang von Knoten i
TX_i	Übertragung an Knoten i
x / RX_i / TX_i	Zuvor vorhandene Reservierungen

sind bereits zuvor eingetragene Reservierungen und dadurch blockierte Slots in grau eingetragen. Nun soll für den Pfad p_1 in Abb. 3.9, d.h. von Knoten 1 über die Knoten 2 und 7 zu Knoten 10, Slots reserviert werden. Wir beginnen hier als Startslot mit dem ersten Slot s_0. Für den ersten Link von Knoten 1 zu Knoten 2 prüfen wir demnach das Reservierungskriterium $F^{s_0}(1,2)$. In Interferenzreichweite der Knoten 1 und 2 befinden sich die Knoten 3, 7, 8 und 9. Keiner dieser Knoten sendet oder empfängt in Slot s_0, sodass das Reservierungskriterium wahr ist und der Slot für diese Übertragung eingetragen werden kann. Für die Knoten 8 und 9 in Interferenzreichweite wird der Slot daraufhin blockiert, für die Knoten 3 und 7 ist dies bereits durch die bisherige Reservierung der Fall gewesen. Der nächste Link von Knoten 2 zu Knoten 7 kann in Slot s_1 nicht eingetragen werden, weil Knoten 7 durch eine andere Übertragung in Interferenzreichweite bereits blockiert ist. In Slot s_2 kann die Übertragung dagegen eingetragen werden, da der Slot noch unbenutzt ist. Dabei werden die Knoten 1, 3, 6, 8, 9 und 10, die in Interferenzreichweite von Knoten 2 und 7 sind, blockiert. Die letzte Übertragung von Knoten 7 zu Knoten 10 ist in Slot s_3 nicht möglich, da Knoten 10 bereits blockiert ist. In Slot s_4 kann die Übertragung dagegen vorgenommen werden, wobei die Knoten 2, 6, 9 und 11 blockiert werden. Die Übertragung für Pfad p_1 wird somit mit einer Übertragungsverzögerung von 5 Slots reserviert.

Slot-Ausnutzung Die zweite Strategie versucht, die Ausnutzung der Slots zu optimieren, sodass möglichst viele Reservierungen möglich sind. Dies wird erreicht, indem versucht wird, SDMA möglichst häufig auszunutzen. So wird das Medium effizienter genutzt und die Gesamtbandbreite des Netzwerkes kann gesteigert werden. Um die Ergebnisse dieser Strategie messbar zu machen, definieren wir zunächst ein Maß für die Ausnutzung eines Schedules.

Die *Slot-Ausnutzung* eines (nicht leeren) Schedules sei definiert als das Verhältnis zwischen der Anzahl zum Senden oder Empfangen belegter Slots durch die Gesamtanzahl belegter Slots. Formal:

$$slots^{TX} \stackrel{\text{def}}{=} \{(v,s) \in V \times S : TX^s(v)\} \quad (3.5)$$

$$slots^{RX} \stackrel{\text{def}}{=} \{(v,s) \in V \times S : RX^s(v)\} \quad (3.6)$$

$$slots^{free} \stackrel{\text{def}}{=} \{(v,s) \in V \times S : \neg TX^s(v) \wedge \neg RX^s(v) \wedge (\nexists\, v' \in IN(v) : RX^s(v') \vee TX^s(v'))\} \quad (3.7)$$

$$utilization \stackrel{\text{def}}{=} \frac{|slots^{TX}| + |slots^{RX}|}{|V| \cdot |S| - |slots^{free}|} \quad (3.8)$$

In einem Schedule wie in Tab. 3.3 beschreibt die Slot Auslastung also das Verhältnis von mit RX_i oder TX_i ausgefüllten Zellen zu allen ausgefüllten Zellen, d.h. einschließlich der mit x ausgefüllten Zellen. In Tab. 3.3 ergibt sich $utilization = \frac{6+6}{12 \cdot 5 - 26} \approx 0{,}35$. Das Optimum von 1 wird offensichtlich nur erreicht, wenn alle Slots für das Senden oder Empfangen genutzt werden, und keine Slots durch Übertragungen in Interferenzweite blockiert werden. Dieses Optimum kann allerdings höchstens in Spezialfällen erreicht werden, beispielsweise in einer Topologie mit nur zwei Knoten. In der Praxis ist die Interferenzreichweite oft größer als die Kommunikationsreichweite und damit blockiert eine einzelne Übertragung einen Slot bereits für eine Menge anderer Knoten.

Die Strategie zur Optimierung der Slot-Ausnutzung basiert auf drei *Slot Decision Policies*. In [Shi+06] haben Shih et al. bereits ähnliche Slot Decision Policies für dezentrale Verfahren definiert. Da QMRP einen zentralisierten Ansatz verfolgt, liegt dem Masterknoten stets das vollständige globale Wissen über alle Reservierungen vor. Wir haben daher Slot Decision Policies entworfen, die von diesem Wissen Gebrauch machen.

Die erste Policy ist die *Fewest Slots First* (FSF) Policy. Die Idee hinter dieser Strategie ist, dass es in einem Pfad einige Links gibt, die man in jedem beliebigen Slot einplanen kann, wohingegen andere Links nur in wenigen Slots überhaupt möglich sind. Vergibt man diese wenigen Slots an andere Links, so kann für diese Links kein passender Slot gefunden werden. Daher bearbeitet die FSF-Policy die Links eines Pfades nicht von vorne nach hinten, wie es die Strategie zum Erreichen einer geringen Übertragungsverzögerung tut. Stattdessen beginnt die FSF-Policy das Scheduling mit jenem Link, für den die wenigsten Slots möglich sind. Wurde für diesen Link ein Slot gefunden, fährt sie mit jenem verbleiben Link fort, für den es nun die wenigsten möglichen Slots gibt. Wir definieren diese Policy nun formal. Dazu definieren wir zunächst die Menge der für eine Übertragung von v_s zu v_r möglichen Slots wie folgt:

$$possibleSlots(v_s, v_r) \stackrel{\text{def}}{=} \{s \in S | F^s(v_s, v_r)\} \quad (3.9)$$

Nun definieren wir mit $FSF(p')$ die Menge der Links, welche nach FSF beim Scheduling des Pfades p als nächstes betrachtet werden sollen. Dabei sei p' die Menge der Links des Pfades p, für die noch ein Slot gefunden werden muss:

$$FSF(p') \stackrel{\text{def}}{=} \{(v_s, v_r) \in p' | \forall (v,v') \in p' : 0 < |possibleSlots(v_s, v_r)| \leq |possibleSlots(v,v')|\} \quad (3.10)$$

Falls es für den Restpfad p' keine Möglichkeit gibt, für alle Links passende Slots zu finden, d.h $\exists (v_s, v_r) \in p' : |possibleSlots(v_s, v_r)| = 0$, dann ist $FSF(p') = \emptyset$. In diesem Fall bricht QMRP das Scheduling ab und beantwortet die Routensuche mit einer Fehlernachricht. Falls mehr als

Tabelle 3.4: Schedule für die Übertragung von Knoten 1 zu Knoten 10 in der Topologie aus Abb. 3.9 mit der Strategie zur Optimierung der Slotnutzung.

Slot:	s_0	s_1	s_2	s_3	s_4
Knoten 1:	TX_2		RX_8	x	
Knoten 2:	RX_1		x	TX_7	
Knoten 3:	x			x	
Knoten 4:	TX_6	x			
Knoten 5:	x				
Knoten 6:	RX_4	TX_{11}	x	x	
Knoten 7:	x	x	TX_{10}	RX_2	
Knoten 8:	x		TX_1	x	
Knoten 9:	x		x	x	
Knoten 10:		x	RX_7	x	
Knoten 11:	x	RX_6	x	TX_{12}	
Knoten 12:		x		RX_{11}	

Legende:

x	Durch eine Übertragung in Interferenzreichweite blockiert
RX_i	Empfang von Knoten i
TX_i	Übertragung an Knoten i
x / RX_i / TX_i	Zuvor vorhandene Reservierungen

ein Link die minimale Anzahl möglicher Slots aufweist, d.h. $|FSF(p')| > 1$, so wird jener Link ausgewählt, der im Pfad zuerst auftritt.

Wir wenden die FSF-Policy nun in Tab. 3.4 auf ein ähnliches Beispiel an, wie zuvor die Strategie zur minimalen Verzögerung. Wieder sollen Slots für den Pfad p_1 in Abb. 3.9 von Knoten 1 über die Knoten 2 und 7 zu Knoten 10 gefunden werden. Die bereits belegten Slots unterscheiden sich leicht zum vorigen Beispiel. Für den Link von Knoten 1 zu Knoten 2 kommen alle Slots bis auf Slot s_2 in Frage, da hier Knoten 1 und 2 blockiert sind. Für den Link von Knoten 2 zu Knoten 7 kommen nur die Slots s_3 und s_4 in Frage, für den Link von Knoten 7 zu Knoten 10 die Slots s_2 und s_4. Folglich ist $FSF(p') = \{(2,7);(7,10)\}$. Von den zwei Links wählen wir jenen, der im Pfad p zuerst vorkommt, nämlich $(2,7)$.

Für den durch die FSF-Policy ausgewählten Link wird anschließend ein Slot anhand der *Least Conflict First* (LCF) Policy gewählt. Diese Strategie betrachtet für jeden möglichen Slot, wie viele der verbleibenden Links des Pfades diesen Slot ebenfalls nutzen könnten, und wählt jenen Slot, der für die wenigsten anderen Links in Frage kommt. Mit dieser Strategie sollen die Chancen, dass für die verbleibenden Links noch ein passender Slot gefunden werden kann, möglichst erhalten bleiben. Um dies formal zu definieren, beschreiben wir zunächst die Menge der für einen Slot s in Frage kommenden Links eines Restpfades p' wie folgt:

$$possibleLinks(s, p') \stackrel{\text{def}}{=} \{(v_s, v_r) \in p' | F^s(v_s, v_r)\} \tag{3.11}$$

Anschließend definieren wir für den Link (v_s, v_r) und den Restpfad p' die Menge $LCF(v_s, v_r, p')$ als jene möglichen Slots, für welche die wenigsten anderen Links des Restpfades in Frage kommen:

$$LCF(v_s, v_r, p') \stackrel{\text{def}}{=} \{s \in possibleSlots(v_s, v_r) | \\ \forall s' \in possibleSlots(v_s, v_r) : |possibleLinks(s, p')| \leq |possibleLinks(s', p')|\} \tag{3.12}$$

Im Beispiel kommen für den Link $(2,7)$ die Slots s_3 und s_4 in Frage. Slot s_3 würde ebenfalls für Link $(1,2)$ in Frage kommen, nicht aber für Link $(7,10)$, da Knoten 10 blockiert ist. Slot s_4 kommt für die beiden anderen Links in Frage, da er noch unbenutzt ist. Somit kommen für Slot s_3 die wenigsten anderen Links aus p' in Frage, sodass $LCF(v_s, v_r, p') = \{s_3\}$. Wenn die LCF-Policy nach der FSF-Policy angewandt wird, so ist durch die FSF-Policy bereits sichergestellt, dass für den durch FSF gewählten Link $possibleSlots(v_s, v_r) > 0$ gilt. Daraus folgt, dass $LCF(v_s, v_r, p') \neq \emptyset$.

Im Beispiel ergab die LCF-Policy nur den Slot s_3, sodass dieser gewählt wird. Anschließend wird der nächste zu betrachtende Slot wieder anhand der FSF-Policy ausgewählt. Für Link $(1,2)$ sind noch drei Slots möglich und für Link $(7,10)$ zwei Slots, sodass FSF den Link $(7,10)$ auswählt. Für diesen Link wählt die LCF-Policy Slot s_2, da dieser nicht für den verbleibenden Link $(1,2)$ in Frage kommt, wohingegen s_4 weiterhin frei ist und daher für alle Links in Frage kommt. Schließlich wird der letzte Link $(1,2)$ betrachtet. Für ihn sind die Slots s_0, s_1 und s_4 möglich. Die LCF-Policy schränkt diese Menge nicht weiter ein, da es keinen weiteren Link im Pfad gibt, auf den Rücksicht genommen werden könnte. In Fällen, in denen die LCF-Policy mehrere Slots ergibt, wird daher eine dritte Strategie angewendet.

Die *Most Reuse First* (MRF) Policy wählt jenen Slot aus, der für die meisten anderen Knoten in Interferenzreichweite bereits blockiert ist. Mit dieser Auswahl wird die Wiederverwendung von Slots maximiert und so wenig weitere Slots wie möglich für künftige Routensuchen blockiert. Wir definieren zunächst mit $possibleNodes(s, v_s, v_r)$ die Menge Knoten in Interferenzreichweite von v_s und v_r, welche den Slot s noch nutzen könnten, wie folgt formal:

$$possibleNodes(s, v_s, v_r) \stackrel{\text{def}}{=} \{v \in IN(v_s) \cup IN(v_r) | \exists v' \in CN(v) : F^s(v, v')\} \tag{3.13}$$

Zu beachten ist, dass diese Definition im Allgemeinen nicht den Knoten in Interferenzreichweite entspricht, für die in Slot s in einer Tabelle wie Tab. 3.4 kein Eintrag steht. Denn es kann Knoten in Interferenzreichweite geben, die in s nicht blockiert sind, aber alle Knoten, zu denen sie Kommunikationslinks besitzen, sind blockiert. So ist im Beispiel Knoten 12 in Slot s_0 nicht blockiert (s. Tab. 3.4), aber der einzige Knoten in seiner Kommunikationsreichweite ist Knoten 11 (s. Abb. 3.10), der in Slot s_0 aber bereits blockiert ist. Daher kann Knoten 12 den Slot s_0 nicht mehr nutzen. Allerdings suchen wir im Beispiel einen Slot für den Link (1,2) und Knoten 12 ist nicht in Interferenzreichweite von Knoten 1 oder Knoten 2, sodass er hier nicht betrachtet wird. In Interferenzreichweite befinden sich die Knoten 3, 7, 8 und 9, wobei Knoten 3 und 7 in Slot s_0 bereits blockiert sind, wohingegen Knoten 8 und 9 frei sind. Knoten 8 und 9 haben Kommunikationslinks untereinander und damit einen Kommunikationspartner, für die ein Slot gefunden werden könnte. Somit ist $possibleNodes(s_0, 1, 2) = \{8,9\}$. In Slot s_1 ist Knoten 7 bereits belegt, wohingegen die Knoten 3 , 8 und 9 frei sind. Knoten 3 könnte in s_1 noch mit Knoten 2 kommunizieren, die Knoten 8 und 9 könnten untereinander kommunizieren. Damit ist $possibleNodes(s_1, 1, 2) = \{3,8,9\}$. Slots s_2 und s_3 sind nicht möglich, weil die Knoten 1 und 2 beide in diesen Slots blockiert sind (diese Slots sind bereits durch die LCF-Policy weggefallen, da sie

nicht in $possibleSlots(1,2)$ sind). Slot s_4 ist unbenutzt, sodass alle 4 Nachbarn in Interferenzreichweite der Knoten 1 und 2 diesen noch nutzen können, d.h. $possibleNodes(s_1,1,2) = \{3,7,8,9\}$. Nun definieren wir als $MRF(v_s,v_r,S')$ die Untermenge der zuvor durch LCF gewählten Slots $S' = LCF(v_s,v_r,p')$, welche von den wenigsten anderen Knoten in Interferenzreichweite von v_s und v_r genutzt werden könnten:

$$MRF(v_s,v_r,S') \stackrel{\text{def}}{=} \{s \in S' | \forall s' \in S' : |possibleNodes(s,v_s,v_r)| \leq |possibleNodes(s',v_s,v_r)|\} \quad (3.14)$$

Im Beispiel wird damit der Slot s_0 ausgewählt, da dieser noch von 2 Knoten in Interferenzreichweite genutzt werden könnte, wohingegen die Slots s_1 und s_4 von 3 bzw. 4 Knoten genutzt werden könnten. Für den ganzen Pfad p_1 wurden nun die Slots s_0, s_3 und s_2 gewählt, sodass eine Gesamtverzögerung von 8 Slots entsteht. Schon in diesem kleinen Beispiel zeigt sich, dass diese Strategie zu einer höheren Verzögerung führt. Die Slot-Ausnutzung beträgt in diesem Beispiel $\frac{7+7}{12 \cdot 5 - 26} \approx 0{,}42$ und ist damit wie erwartet deutlich höher als mit der ersten Strategie.

Ausnutzen lokaler Multicasts Wenn ein neuer Empfänger zu einem Multicast-Baum hinzugefügt wird, so bedeutet dies, dass ein neuer Pfad an einen existierenden Baum angefügt wird. An der Stelle, an der der neue Ast abzweigt, muss eine zusätzliche Übertragung an den ersten Knoten des neuen Astes eingeplant werden. Da bereits eine Übertragung an den (oder die) nächsten Knoten im bestehenden Baum eingeplant ist, kann hier durch lokalen Multicast eine Übertragung eingespart werden. Im Beispiel in Abb. 3.10 wird der Pfad $p_{2,5}$ an den Baum t angefügt. Für Knoten 2 ist in t bereits eine Übertragung zu Knoten 7 eingeplant. Durch lokalen Multicast kann Knoten 3 des neuen Astes $p_{2,5}$ diese ohnehin stattfindende Übertragung mithören. Auf diese Weise kann eine Übertragung eingespart werden. Um sicherzustellen, dass Übertragungen zuverlässig erfolgen, muss aber ein Acknowledgement (ACK) von allen Empfängern eines lokalen Multicasts erfolgen. In QMRP ist die Slotgröße so bemessen, dass neben der eigentlichen Übertragung bis zu 3 ACKs in einen Zeitslot passen. Dabei werden die ACKs in der Reihenfolge der Knoten-IDs der Empfänger gesendet. Im Beispiel würde dies bedeuten, dass Knoten 2 vor Knoten 7 sein ACK sendet. Der Routing-Master informiert alle Knoten eines Baums über den Multicast-Baum und seinen Schedule, sodass alle Knoten an einer Abzweigung die Knoten-IDs der anderen Empfänger des lokalen Multicats kennen. Anhand der Reihenfolge der Knoten-IDs bestimmen sie, wann ihr ACK zu senden ist, ohne dass Kollisionen auftreten können. Falls an einer Stelle mehr als drei Äste abzweigen, muss eine zusätzliche Reservierung eingeplant werden. Dies kann auch dann nötig sein, wenn ein Empfänger hinzugefügt werden soll, der in dem zuvor gewählten Slot durch eine Übertragung in seiner Interferenzreichweite blockiert ist. Das weitere Vorgehen zum Reservieren von Zeitslots beim Anfügen eines neuen Asts an einen bestehenden Multicast-Baum erfolgt wie beschrieben abhängig von der Strategie.

3.3.3 QoS Multicast Routing in teilweise mobilen Netzwerken

Das bisher beschriebene QoS Multicast Routing-Verfahren basiert auf der von ATDP erkannten Topologie. Da ATDP zum Start des Netzwerks die Topologie einmalig erkennt und verteilt, kann es die Position von mobilen Knoten nicht während dem Betrieb aktualisieren. Aus diesem Grund ist ATDP nur für Netzwerke geeignet, deren Knoten stationär sind. In einem Anwendungsszenario wie der Prozessautomatisierung in Fabriken (vgl. Kapitel 2.2) sind zwar die meisten Knoten stationär. Manche Knoten, wie beispielsweise solche, die auf Robotern montiert

sind, können sich dagegen in einem bestimmten Bereich bewegen. Falls der Bewegungsbereich so groß ist, dass sich die Kommunikationstopologie oder Interferenztopologie durch die Bewegung des Knotens verändern, so sind die bisher beschriebenen Verfahren unzureichend. Routen, die mobile Knoten enthalten, würden durch Bewegung brechen oder unzuverlässig werden. Durch eine veränderte Interferenztopologie könnte es sein, dass der mobile Knoten nun in den Interferenzbereich von anderen Knoten gerät und deren Übertragungen daher stört. Auf Grund dieser Probleme ist es in den meisten Fällen notwendig, die Mobilität von Knoten zu berücksichtigen. Die meisten Routing-Protokolle, die mit Mobilität umgehen können, gehen davon aus, dass jeder Knoten sich jederzeit bewegen kann. Unter dieser Voraussetzung sind stabile Routen mit fest reservierten Zeitslots aber kaum möglich. QMRP verfolgt daher einen anderen Ansatz: Es unterscheidet zwischen stationären und mobilen Knoten. Welchem Typ ein Knoten angehört, wird durch Konfiguration festgelegt. Stationäre Knoten werden behandelt wie zuvor beschrieben, d.h. es wird eine statische Route auf der von ATDP erkannten Topologie berechnet und dafür feste Zeitslots reserviert. Mobile Knoten dagegen werden gesondert behandelt. Dabei wird vorausgesetzt, dass mobile Knoten sich immer in Reichweite eines *Access-Knotens* befinden. Im Folgenden bezeichnen wir die Menge der Access-Knoten als V_{access}. Dabei handelt es sich um eine Untermenge der stationären Knoten, welche die Aufgabe übernehmen, Zugangspunkte für die mobilen Knoten bereitzustellen. Die Access-Knoten sollten dabei so gewählt werden, dass die Voraussetzung immer erfüllt ist, d.h. dass die Access-Knoten die Bewegungsbereiche der mobilen Knoten vollständig abdecken. In diesem Kapitel werden die Details beschrieben, wobei Kapitel 3.3.3.1 die Routensuche und Kapitel 3.3.3.2 das Reservieren von Zeitslots erläutert.

3.3.3.1 Ermittlung von Multicast Routing-Bäumen

Zunächst erweitern wir die Routensuche von QMRP um eine Unterstützung für mobile Knoten. Mobile Knoten können Quelle oder Ziel einer Multicast-Route sein oder beides. Wir betrachten zunächst den einfacheren Fall, in dem ein mobiler Knoten (ein) Ziel einer Multicast-Route ist.

Mobiler Knoten als Ziel Wir gehen davon aus, dass ein mobiler Knoten sich stets in Reichweite eines der stationären Access-Knoten befindet. Um eine Nachricht an einen mobilen Knoten zu übertragen, kann man diese also zunächst an alle Access-Knoten senden. Dies lässt sich über einen Multicast-Baum erreichen, wobei die Access-Knoten die Ziele des Baums bilden. Die Berechnung eines derartigen Multicast-Baumes wurde in Kapitel 3.3.2.1 bereits beschrieben. Da der mobile Knoten sich immer in Reichweite eines der Access-Knoten befindet, reicht es, wenn derjenige Access-Knoten, mit dem der mobile Knoten gerade verbunden ist, die Nachricht an den mobilen Knoten überträgt. Die Access-Knoten erfahren anhand von Beacon-Rahmen, die in regelmäßigen Abständen zwischen dem mobilen Knoten und den Access-Knoten ausgetauscht werden, mit welchem Access-Knoten der mobile Knoten gerade verbunden ist. Die Übertragung dieser Beacon-Rahmen erfolgt zuverlässig in zuvor reservierten Management Zeitslots.

Die Route von einer stationären Quelle zu einem mobilen Knoten besteht also aus einem Multicast-Baum zu allen Access-Knoten sowie dem letzten Hop von einem der Access-Knoten zum mobilen Knoten. Dies wird anhand eines Beispiels in Abb. 3.11 veranschaulicht. Um von der Quelle, Knoten 1, zum mobilen Knoten M zu routen, wird zunächst ein Multicast-Baum von Knoten 1 zu den Access-Knoten 2, 4 und 10 aufgebaut. Zum Erstellen dieses Baumes wird zunächst eine Route von der Quelle zu jenem Access-Knoten erstellt, welcher der Quelle am nächsten ist. Im Beispiel ist dies offensichtlich Knoten 2, sodass der Pfad p_1 gewählt wird. Anschließend wird der jeweils am nächsten gelegene der restlichen Access-Knoten den Baum als

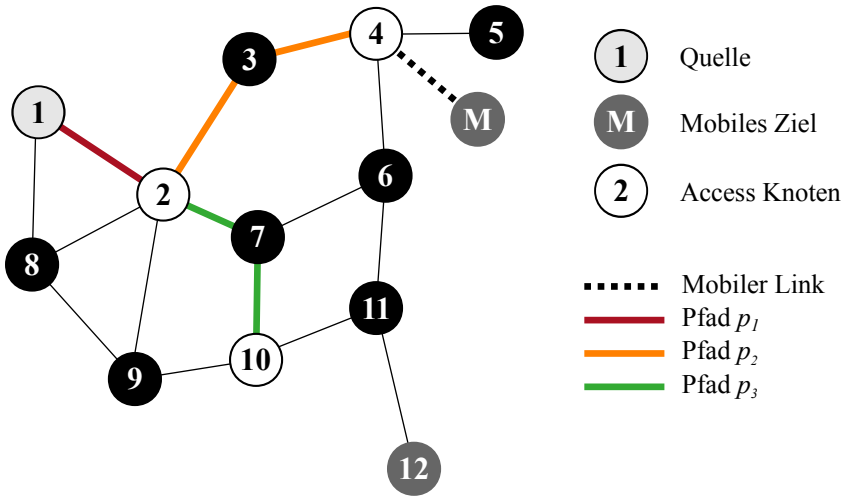


Abbildung 3.11: Multicast-Baum von Knoten 1 zu den Access-Knoten 2, 4 und 10 zur Anbindung des mobilen Ziels *M*.

Ziel hinzugefügt. Im Beispiel führt dies dazu, dass die Pfade p_2 und p_3 zum Baum hinzugefügt werden. Der letzte Link von einem der Access-Knoten zum mobilen Knoten *M* erfolgt abhängig von der Position des mobilen Knotens. Im Beispiel ist *M* mit Access-Knoten 4 verbunden, sodass dieser Link genutzt würde.

Falls ein mobiler Knoten als zusätzliches Ziel zu einem bereits bestehenden Baum angefügt werden soll, so werden wie beschrieben alle Access-Knoten zum bestehenden Baum hinzugefügt.

Mobiler Knoten als Quelle Im Fall, dass ein mobiler Knoten die Quelle einer Multicast-Route ist, nutzen wir ebenfalls die Voraussetzung aus, dass mobile Knoten sich stets in Reichweite der Access-Knoten befinden. Dies bedeutet, dass eine Route mit mobilem Knoten als Quelle von allen Access-Knoten zu allen Zielen führen müsste. Um die Komplexität des Problems zu reduzieren, nutzt QMRP dazwischen einen *Distributor-Knoten*. Dabei handelt es sich um einen stationären Zwischenknoten, der die Nachrichten von den Access-Knoten erhält und an die Ziele weiterleitet. Der zweite Teil der Route vom Distributor-Knoten zu den Zielen ist ein normaler Multicast-Baum und kann, wie in Kapitel 3.3.2.1 erklärt, gefunden werden. Den erste Teil der Route von allen Access-Knoten zum Distributor-Knoten bezeichnen wir als *Concast-Baum*. Dabei handelt es sich um einen Baum, der von mehreren Quellen (den Access-Knoten) zu einem gemeinsamen Ziel (dem Distributor-Knoten) führt, wobei stets nur einer der Äste genutzt wird. Um diesen Baum zu berechnen, können wir den zuvor beschriebenen Ansatz für Multicast-Bäume anwenden, müssen dabei lediglich Quelle und Ziel vertauschen. Da QMRP ausschließlich bidirektionale Links verwendet, kann ein Multicast-Baum immer auch in die entgegengesetzte Richtung verwendet werden. Wir suchen also wie zuvor beschrieben einen Multicast-Baum vom Distributor-Knoten zu allen Access-Knoten und nutzen diesen in umgekehrter Richtung.

In Abb. 3.12 wird dies wieder anhand eines Beispiels veranschaulicht. Es soll eine Route vom mobilen Knoten *M* zu den Zielen 8 und 12 gefunden werden. Als Distributor-Knoten wurde Knoten 7 gewählt, als Access-Knoten dienen die Knoten 3, 4 und 10. Die Route beginnt mit dem ersten Link vom mobilen Knoten zum nächsten Access-Knoten, in diesem Fall Knoten 3. Von

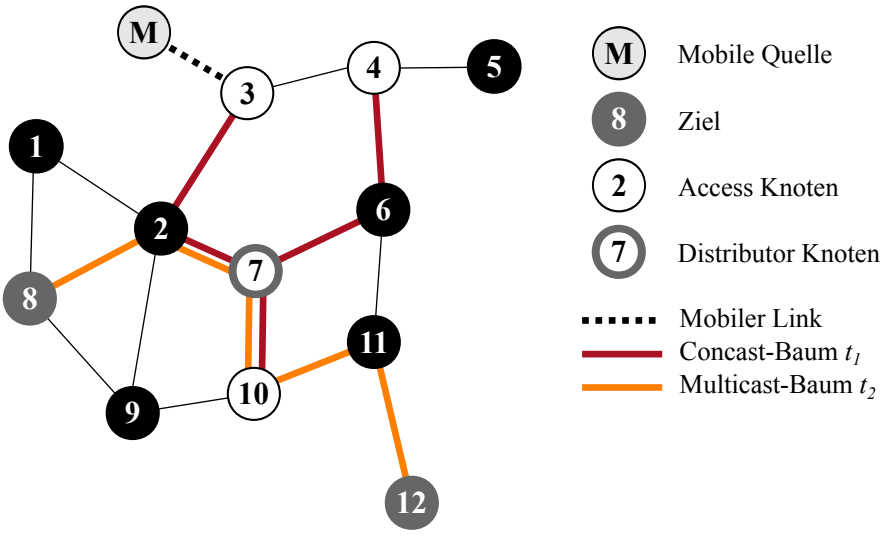


Abbildung 3.12: Concast-Baum von den Access-Knoten 3, 4 und 10 zum Distributor-Knoten 7 und Multicast-Baum vom Distributor-Knoten 7 zu den Zielen 8 und 12 zur Anbindung des mobilen Ziels M.

den Access-Knoten wird der Concast-Baum t_1 zum Distributor-Knoten 7 genutzt. Von dort wird der Multicast-Baum t_2 zu den Zielen 8 und 12 genutzt.

Im Beispiel kann man auch eine Optimierung beobachten, die QMRP beim Bilden vom Concast-Bäumen vornimmt. Bei Multicast-Bäumen wird versucht, neue Ziele möglichst kurz an den bestehenden Baum anzubinden, und so die Gesamtanzahl der Hops des Baumes zu minimieren. Bei Concast-Bäumen wird dagegen ausschließlich die Anzahl Hops des einzelnen Astes minimiert, d.h. der kürzeste Pfad von der Quelle zum Ziel gewählt. Im Beispiel führt dies dazu, dass Knoten 4 auf dem kürzesten Pfad über Knoten 6 mit dem Distributor-Knoten 7 verbunden wird. Bei der Bildung eines Multicast-Baumes wäre Knoten 4 über den Link zu Knoten 3 an den Baum angefügt worden. Da bei Concast-Bäumen immer nur ein Ast gleichzeitig genutzt wird, führt dies zu effizienteren Bäumen. So ist es im Beispiel nicht notwendig, dass die Übertragung von Knoten 4 zu Knoten 7 auch Knoten 3 erreicht.

Das Beispiel zeigt ebenfalls, dass es vorkommen kann, dass ein Link einmal zum Distributor-Knoten hin und anschließend vom Distributor-Knoten weg genutzt wird. Im Beispiel nutzt die Übertragung von M zum Ziel Knoten 8 den Link zwischen Knoten 2 und Knoten 7 zweimal. Dies ist offensichtlich ineffizient, aber notwendig, da die Position des mobilen Knotens nicht fest ist und dieser auch angebunden werden muss, wenn er beispielsweise mit Access-Knoten 10 verbunden wäre.

Die Wahl des Distributor-Knotens ist offensichtlich entscheidend, um kurze und effiziente Routen zu erzielen. Zur Auswahl sind unterschiedliche Ansätze denkbar. Ein möglicher Ansatz ist, einen der Access-Knoten als Distributor-Knoten zu wählen, beispielsweise jenen, über den der mobile Knoten die Route angefragt hat. Solange der mobile Knoten mit diesem Access-Knoten verbunden ist, entfällt damit die Übertragung vom Access-Knoten zum Distributor-Knoten. Bewegt der Knoten sich aber zu einem der anderen Access-Knoten, so könnten ineffiziente Routen die Folge sein, bei denen viele Links mehrfach benutzt werden. Ein anderer Ansatz ist die Wahl eines zentral gelegenen Knotens als Distributor-Knoten. Ein zentral gelegener Knoten kann unabhängig von der Position des mobilen Knotens von allen anderen Knoten gut erreicht werden. QMRP wählt diesen Ansatz und nutzt als Distributor-Knoten automatisch jenen Knoten, dessen

mittlere Entfernung zu den restlichen Knoten minimal ist. Unterschiedliche Auswahlkriterien für den Distributor-Knoten zu untersuchen und zu evaluieren ist ein interessantes Problem für künftige Forschung.

Mobile Knoten als Quelle und Ziel Den Fall, dass ein mobiler Knoten an einen anderen mobilen Knoten sendet, löst QMRP durch eine Kombination der vorherigen Fälle. Die mobile Quelle sendet an den verbundenen Access-Knoten. Von dort wird über einen Concast-Baum an einen Distributor-Knoten gesendet, der über einen Multicast-Baum die Nachricht an alle Access-Knoten weiterleitet. Der letzte Hop zum mobilen Ziel wird wieder abhängig von der Position des mobilen Ziels ausgewählt.

3.3.3.2 Reservierung von TDMA-Zeitslots

Auch die Übertragungen von und an mobile Knoten sollen zuverlässig erfolgen und die Übertragungen der stationären Knoten nicht stören. Daher ist es notwendig, auch für diese Übertragungen exklusive Zeitslots zu reservieren. Dies ist in QMRP möglich, da die Routen zur Kommunikation mit einem mobilen Knoten sich, abgesehen vom letzten bzw. ersten Hop, nicht in Abhängigkeit von der Position des mobilen Knotens ändern.

Bei Routen mit mobilem Ziel wurde ein Multicast-Baum zu allen Access-Knoten generiert. Für diesen Multicast-Baum erfolgt die Reservierung von Zeitslots wie in Kapitel 3.3.2.2 beschrieben. Der letzte Hop von einem der Access-Knoten zum mobilen Knoten muss dagegen anders behandelt werden. Wir wissen, dass der mobile Knoten immer nur mit einem Access-Knoten verbunden ist, d.h. es wird nur ein Zeitslot benötigt, nicht für jeden Zeitslot einer. Allerdings wissen wir nicht, wo sich der mobile Knoten befindet. Um zu garantieren, dass der mobile Knoten keine anderen Übertragungen stört, wenn er sein ACK sendet, darf in diesem Zeitslot keine Übertragung in Interferenzreichweite des mobilen Knotens stattfinden. Da wir diese Interferenzreichweite aber nicht kennen und sie sich permanent ändern kann, ist es erforderlich, diesen Zeitslot exklusiv zu reservieren, d.h. keine anderen Übertragungen im gleichen Slot zu erlauben. Da Mehrfachnutzung des Slots ohnehin ausgeschlossen ist, macht die Strategie zur Maximierung der Slot-Ausnutzung in diesem Fall keinen Sinn. Daher wird für den letzten Hop stets die Strategie zur Minimierung der Verzögerung angewendet, d.h. der nächste komplett freie Slot ausgewählt.

Im Falle einer Route mit mobiler Quelle besteht die Route wie beschrieben aus dem ersten Link vom mobilen Knoten zu einem der Access-Knoten, einem Concast-Baum von den Access-Knoten zum Distributor-Knoten, und einem Multicast-Baum vom Distributor-Knoten zu den Zielen. Den ersten Link behandeln wir genauso wie den letzten Link bei Routen mit mobilem Ziel, d.h. reservieren den nächsten freien Zeitslot exklusiv. Für den Concast-Baum von den Access-Knoten zum Distributor-Knoten kann man die Zeitslots genauso reservieren wie für Multicast-Bäume in stationären Netzwerken. Allerdings kann man hier die Eigenschaft von Concast-Bäumen ausnutzen, dass zu einer Zeit immer nur einer der Äste des Baumes genutzt wird. Das bedeutet, dass Zeitslots von unterschiedlichen Ästen des Baumes nicht miteinander in Konflikt stehen. Ein Zeitslot kann also mehreren Ästen des Baumes zugeordnet werden, selbst wenn diese in Interferenzreichweite voneinander sind. Abgesehen davon erfolgt die Reservierung der Zeitslots wie in Kapitel 3.3.2.2 beschrieben. Der letzte Teil der Route vom Distributor-Knoten zu den Zielen ist ein normaler Multicast-Baum zwischen stationären Knoten. Die Reservierung der Zeitslots für diesen Teil erfolgt genauso wie für andere Multicast-Bäume (s. Kapitel 3.3.2.2).

Im Fall von einer Route mit mobiler Quelle und mobilem Ziel ist wie zuvor beschrieben eine Kombination der anderen Fälle und wird auch bei der Reservierung von Zeitslots so behandelt.

3.3.4 Implementierung und Evaluierung

Eine Implementierung von QMRP wurde im ProRoute-Modul von ProNet 4.0 entwickelt (s. Kapitel 3.1.1) und auf der Imote 2 Plattform getestet. Um Multicast Routing in einem größeren Multihop-Netzwerk zu untersuchen, war das vorhandene Testbett aus Imote 2 Knoten aber nicht groß genug. Daher wurde QMRP zusätzlich auch in Simulationen evaluiert. Dabei lag der Fokus auf zwei Fragestellungen. Einerseits sollten die beiden Strategien zur Auswahl von Zeitslots gegeneinander evaluiert werden. Andererseits sollte untersucht werden, welche Vorteile die Nutzung von Access-Knoten und Concast-Bäumen bringt.

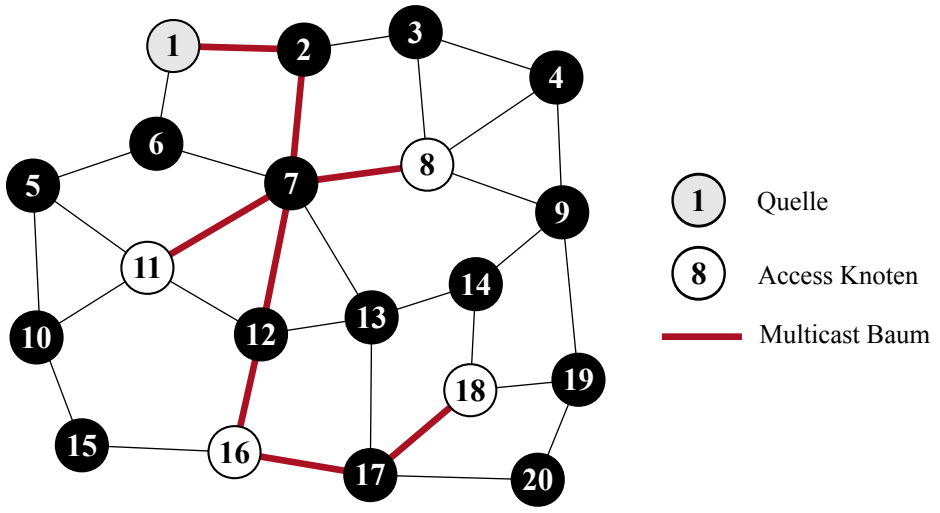


Abbildung 3.13: Topologie 1 mit einem Multicast-Baum von Knoten 1 zu den Access-Knoten 8, 11, 16 und 18.

Für die Simulationen wurden unterschiedliche Topologien genutzt. In diesem Kapitel betrachten wir die Ergebnisse von zwei dieser Topologien genauer. Topologie 1, abgebildet in Abb. 3.13, ist eine etwas dichtere Topologie mit ungefähr gleichmäßig verteilten Knoten. Topologie 2, zu sehen in Abb. 3.14, ist ein etwas in die Breite gezogenes Netzwerk, wie es beispielsweise entlang eines Fließbands in einem Netzwerk zur Prozessautomatisierung auftreten könnte. In der Simulation gehen wir davon aus, dass die Interferenzreichweite der 2-Hop-Kommunikationsnachbarschaft entspricht. Beispielsweise bedeutet dies in Topologie 1, dass die Interferenznachbarschaft von Knoten 1 die Knoten 2, 3, 5, 6 und 7 umfasst.

Zunächst untersuchen wir die beiden Strategien zur Reservierung von Zeitslots. Dazu wurden 50 Szenarien zufällig generiert, wobei jedes Szenario aus drei Multicast-Bäumen mit einer Quelle und drei Zielen besteht. Diese Szenarien wurden auf beide Topologien und mit beiden Strategien angewendet. Außerdem wurde die Länge des Superslots, d.h. die Anzahl Zeitslots, die periodisch wiederholt wird, zwischen 10 und 15 Slots variiert. Die Ergebnisse dieser Simulationen fassen die Tabellen Tab. 3.5 und Tab. 3.6 zusammen.

Ziel der minDelay-Strategie ist die Minimierung der Übertragungsverzögerung. Vergleicht man die Verzögerung mit der Strategie zur Maximierung der Slot-Ausnutzung (maxUtil), dann zeigt sich, dass die minDelay-Strategie bei ansonsten gleichen Parametern stets deutlich geringere

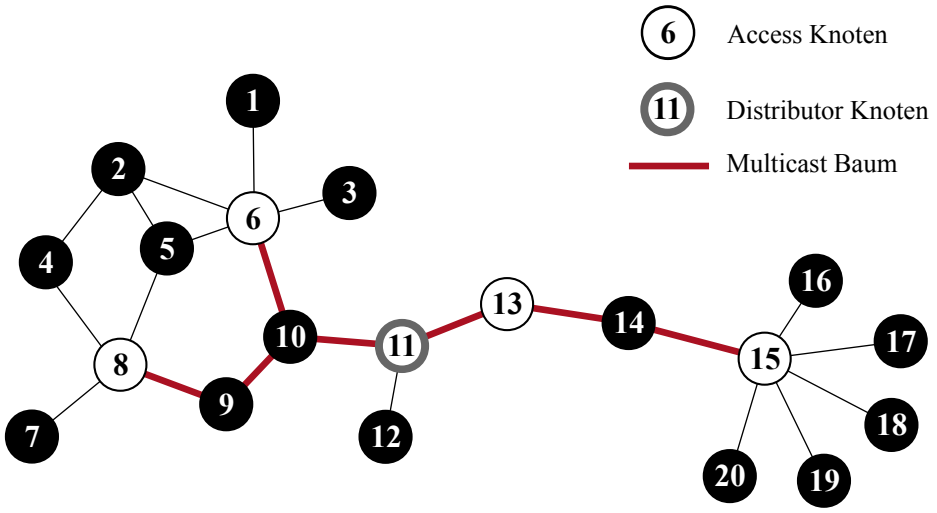


Abbildung 3.14: Topologie 2 mit einem Concast-Baum von den Access-Knoten 6, 8, 13 und 15 zum Distributor-Knoten 11.

Tabelle 3.5: Ergebnisse der Simulationen beider Strategien auf Topologie 1.

Strategie	Superslotlänge	Utilization	Tiefe des Baums	Verzögerung	Erfolgsrate
maxUtil	10	0.19 ± 0.02	2.59 ± 1.22	4.66 ± 4.10	54 %
minDelay	10	0.18 ± 0.02	2.52 ± 1.21	3.16 ± 2.07	40 %
maxUtil	12	0.18 ± 0.02	2.73 ± 1.23	5.35 ± 5.08	96 %
minDelay	12	0.17 ± 0.01	2.74 ± 1.24	3.61 ± 2.48	94 %
maxUtil	15	0.18 ± 0.02	2.74 ± 1.24	6.06 ± 6.39	100 %
minDelay	15	0.17 ± 0.01	2.75 ± 1.23	3.56 ± 2.18	100 %

Tabelle 3.6: Ergebnisse der Simulationen beider Strategien auf Topologie 2.

Strategie	Superslotlänge	Utilization	Tiefe des Baums	Verzögerung	Erfolgsrate
maxUtil	10	0.24 ± 0.01	3.54 ± 2.30	7.33 ± 7.28	16 %
minDelay	10	0.23 ± 0.01	3.33 ± 2.12	4.47 ± 3.63	10 %
maxUtil	12	0.24 ± 0.01	3.96 ± 2.22	9.08 ± 8.34	90 %
minDelay	12	0.23 ± 0.01	3.63 ± 2.06	5.36 ± 5.14	52 %
maxUtil	15	0.24 ± 0.01	3.99 ± 2.21	11.22 ± 11.15	100 %
minDelay	15	0.23 ± 0.01	3.99 ± 2.21	5.70 ± 4.34	100 %

Verzögerungen ergibt. Die Standardabweichung der Verzögerung ist darüber hinaus deutlich geringer als bei der maxUtil-Strategie, d.h. die Verzögerung schwankt bei der maxUtil-Strategie deutlich mehr. Die maxUtil-Strategie ergibt für den erste Multicast-Baum i.d.R. eine kurze Verzögerung, aber die folgenden Multicast-Bäume müssen auf immer mehr bereits vergebene Slots Rücksicht nehmen, wodurch die Verzögerung steigt.

An der durchschnittlichen Utilization zeigt sich, dass die maxUtil-Strategie auch wie zu erwarten eine höhere Slot-Ausnutzung erzielt, als die minDelay-Strategie. Allerdings sind die Unterschiede nur gering. Generell zeigt sich, dass die dichtere Topologie 1 eine geringere Utilization ermöglicht, als Topologie 2. Umso dichter eine Topologie ist, umso mehr Knoten werden durch eine einzelne Übertragung blockiert. Die hier betrachteten Topologien sind beide noch recht klein und dicht, sodass nicht all zu viel Slot-Wiederverwendung möglich ist. Vergleicht man die Erfolgsrate der beiden Strategien, stellt man allerdings einen signifikanten Unterschied fest. Wenn die Superslotlänge gering ist und die Auslastung damit hoch, so kann die maxUtil-Stratgie in deutlich mehr der Szenarien passende Slots finden. Dies zeigt, dass die drei Policies der maxUtil-Strategie die Zielsetzung erreichen.

Insgesamt lässt sich keine universelle Empfehlung aussprechen. Vielmehr sollte die gewählte Strategie abhängig gemacht werden von den Anforderungen. Liegen beispielsweise knappe QoS-Anforderungen an die Verzögerung vor, so ist in jedem Fall die minDelay-Strategie vorzuziehen. Ist dagegen die Auslastung des Netzwerkes hoch, ermöglicht die maxUtil-Strategie, für mehr Routen passende Zeitslots zu finden. Denkbar wäre auch eine Kombination beider Strategien. So könnte man zunächst stets die maxUtil-Strategie versuchen und deren Ergebnis nutzen, falls diese die geforderten Anforderungen an die Verzögerung erreichen. Werden die Anforderungen dagegen nicht erreicht, kann man anschließend die minDelay-Strategie versuchen. Wenig erfolgversprechend ist es dagegen, zunächst die minDelay-Strategie zu wählen und erst, sofern diese die keine Routen mehr findet, auf die maxUtil-Strategie zu wechseln. Denn die maxUtil-Strategie erreicht eine höhere Erfolgsquote, indem sie proaktiv Slots mehrfach benutzt, wann immer das möglich ist. Hat die minDelay-Strategie aber bereits viele Möglichkeiten für Slot-Wiederverwendung ungenutzt gelassen, so werden später beide Strategien keine passenden Slots mehr finden können.

Für die zweite Fragestellung nach dem Nutzen von Access-Knoten wurden ebenfalls Simulationen auf den beiden gezeigten Topologien ausgeführt. Würde man mobile Knoten ohne Access-Knoten unterstützen wollen, so müsste man stattdessen einen Broadcast an alle stationären Knoten durchführen, da nicht klar ist, über welchen Knoten der mobile Knoten gerade erreicht werden kann. In Abb. 3.13 ist ein Beispiel eines Multicast-Baums zu sehen, welches von der Quelle (Knoten 1) zu den vier Access-Knoten 8, 11, 16 und 18 führt. Sucht man für diesen Multicast-Baum mit der minDelay-Strategie passende Zeitslots, so ergibt sich eine Verzögerung von 6 Slots, wobei 69 Slots für Knoten in Interferenzreichweite blockiert werden. Berechnet man stattdessen einen Multicast-Baum an alle stationären Knoten, ergibt die minDelay-Strategie eine Verzögerung von 11 Slots und blockiert 128 Slots. In beiden Fällen wurde lokaler Multicast an bis zu drei Empfänger ausgenutzt. Dieses Beispiel macht deutlich, wie stark bereits in relativ kleinen Netzwerken die Auslastung des Netzwerkes durch den Einsatz von Access-Knoten gesenkt werden kann.

Die letzte Fragestellung betrifft die bei Routen mit mobiler Quelle eingesetzten Concast-Bäume. Hier nutzt QMRP aus, dass stets nur einer der Äste einen Concast-Baumes gleichzeitig genutzt werden kann, sodass Slots mehreren Ästen zugeordnet werden können. Anhand eines Beispiels untersuchen wir, wie viele Slots diese Optimierung einspart und wie sie sich auf die Verzöge-

rung auswirkt. Dazu betrachten wir den in Abb. 3.14 eingezeichneten Concast-Baum von den Access-Knoten 6, 8, 13 und 15 zum Distributor-Knoten 11. Ohne die Optimierung, Slots innerhalb des Baumes mehrfach zu verwenden, ergibt die minDelay-Strategie eine Verzögerung von 7 Slots, wohingegen mit der Optimierung nur 3 Slots benötigt werden. Bereits in diesem kleinen Beispiel erkennt man, dass die Optimierung nicht nur deutlich weniger Slots benötigt, sondern auch zu geringeren Verzögerungen führt.

3.3.5 Zusammenfassung

In größeren Netzwerken müssen Nachrichten oft über mehrere Hops weitergeleitet werden. Um eine geeignete Route durch das Netzwerk zu finden, werden Routing-Verfahren eingesetzt. In diesem Kapitel wurden Routing-Verfahren für TDMA-basierte Kommunikationssysteme behandelt. Bei TDMA-basierten Kommunikationssystemen müssen für die Übertragungen der Route passende Zeitslots reserviert werden, um zuverlässige Übertragungen sicherstellen zu können. Das Scheduling genannte Finden passender Zeitslots wurde daher in diesem Kapitel ebenfalls behandelt. Falls mehrere Empfänger die gleichen Daten anfordern, können Ressourcen eingespart werden, indem mehrere Empfänger durch eine Multicast-Route erreicht werden. Im Anwendungsszenario der Prozessautomatisierung in Fabriken ist dies beispielsweise der Fall, wenn zwei Regler den gleichen Sensorwert anfordern. Obwohl in Fabrikanlagen die meisten Knoten stationär montiert sind, kann es auch mobile Knoten, beispielsweise auf Robotern, geben. Auch die Kommunikation dieser Knoten muss zuverlässig erfolgen und darf die Kommunikation der stationären Knoten nicht stören. Bei der Betrachtung existierender Verfahren zeigte sich, dass diese die Anforderungen des Anwendungsszenarios nicht vollständig erfüllen. Daher wurde mit QMRP ein Routing-Verfahren entwickelt, welches alle Anforderungen des Anwendungsszenarios erfüllt.

QMRP ist ein Routing-Verfahren für TDMA-Netzwerke. Es kann Zeitslots entlang der Route reservieren und nutzt dabei SDMA aus, soweit dies möglich ist. Zur Auswahl der Zeitslots unterstützt QMRP zwei Strategien, wobei eine die Übertragungsverzögerung minimiert und die andere die Slot-Ausnutzung maximiert. Zu einer bestehenden Route können flexibel weitere Ziele hinzugefügt werden, sodass Multicast-Bäume entstehen. Sowohl Quelle als auch Ziel einer Route können mobil sein. QMRP bindet mobile Knoten effizient über Access-Knoten und einen Distributor-Knoten an das stationäre Netzwerk an. Für ProNet 4.0 wurde mit ProRoute eine Implementierung von QMRP entwickelt, die auf der Imote 2 Hardware ausgeführt werden kann. Größere Topologien wurde darüber hinaus anhand von Simulationen evaluiert. Dabei hat sich gezeigt, dass die beiden Scheduling-Strategien die jeweilige Zielsetzung erreichen und die Auswahl der Strategie abhängig von den Anforderungen erfolgen muss. Außerdem wurde gezeigt, dass der Einsatz von Access-Knoten und Concast-Routen Übertragungen einsparen und Verzögerungen verringern kann.

QMRP basiert auf Heuristiken, die sich in den Simulationen als zielführend erwiesen haben. Für weitere Forschung wäre es dennoch lohnenswert, Variationen der eingesetzten Heuristiken zu evaluieren. Beispielsweise wählt QMRP den kürzesten Pfad mit der größten Nachbarschaft, um künftige Erweiterungen der Route zu optimieren. Alternativ könnte man auch den Pfad mit der kleinsten Nachbarschaft wählen, was zu einer besseren Slot-Ausnutzung und damit einer höheren Erfolgsrate führen könnte. Weiterhin könnten unterschiedliche Strategien zur Auswahl des Distributor-Knotens untersucht werden. Die beiden Strategien zur Auswahl von Zeitslots könnten kombiniert oder die Auswahl der Strategie automatisiert werden.

3.4 Clustering

Mit Clustering-Verfahren werden die Knoten eines Netzwerks in Gruppen, sog. *Cluster*, eingeteilt. Die Einteilung erfolgt dabei in der Regel räumlich, d.h. die in einem Cluster zusammengefassten Knoten liegen räumlich nahe beieinander. Jedes Cluster hat einen Anführer, der meist als *Cluster Head (CH)* bezeichnet wird und besondere Aufgaben übernimmt [AY07]. Meist werden Cluster gebildet, um die Skalierbarkeit zu verbessern, denn durch das Gruppieren von Knoten zu Clustern können die Aufgaben im Netzwerk hierarchisch strukturiert und somit große Netzwerke einfacher betrieben werden. Beispielsweise kann das Routing in einem geclusterten Netzwerk hierarchisch erfolgen, d.h. eine Route von der Quelle v_s zum Empfänger v_r führt von v_s über den CH von v_s und den CH von v_r zum Knoten v_r. Das Routing wird damit deutlich einfacher, da sog. *Follower*, d.h. Knoten, die selbst kein CH sind, nur die Route zu ihrem CH benötigen. CHs wiederum müssen nur die Routen zwischen den CHs kennen, sowie welchem Cluster das Ziel v_r angehört.

Mit Clustering können über das Routing hinaus auch andere Aufgaben im Netzwerk hierarchisch organisiert werden. Für ProNet 4.0 wurde mit dem in Kapitel 3.5 näher beschriebenen ProMid eine dienstorientierte Middleware-Schicht entwickelt. Diese unterstützt das flexible Registrieren, Auffinden und Nutzen von Diensten im Netzwerk. Um das Auffinden von Diensten zu ermöglichen, werden die registrierten Dienste in einer *Service Registry* gespeichert. Diese Registry soll im Netzwerk verteilt werden, sodass jeder Knoten entweder selbst Registry-Knoten ist, oder aber einen solchen als direkten Nachbarn hat. Mit einem Clustering-Verfahren können die Knoten im Netzwerk in Cluster eingeteilt werden, wobei Cluster Heads die Rolle eines Registry-Knotens übernehmen und Follower auf die Registry über ihren CH zugreifen. Auf diese Weise wird der Zugriff auf die Service Registry hierarchisch strukturiert. Dadurch wird einerseits die Skalierbarkeit verbessert, andererseits aber auch eine bessere Redundanz und kürzere Zugriffszeiten auf die Registry erzielt als mit einer zentralisierten Registry. Die Forderung, dass jeder Knoten entweder selbst Registry-Knoten ist, oder einen Registry-Knoten als direkten Nachbarn hat, entspricht der Suche nach einem *1-Hop Dominating Set* [Ami+00], einem NP-vollständigen Problem [GJ79]. Da die Registry-Knoten Änderungen an der Registry miteinander austauschen müssen, macht es Sinn, die Inter-Cluster-Konnektivität durch die Auswahl von *Gateways* zu unterstützen. Dabei handelt es sich um Knoten, über die nicht direkt verbundene CHs Nachrichten austauschen.

Im Anwendungsszenario der Prozessautomatisierung in Fabriken sind, wie bereits in Kapitel 3.3 erwähnt, zwar die meisten Knoten stationär, einige Knoten können aber mobil sein, beispielsweise weil sie auf mobilen Robotern montiert sind. Die Rolle eines Registry-Knotens oder Gateways sollten diese Knoten nicht übernehmen, denn sie können ihre Position ändern, was instabile Cluster zur Folge hätte. Außerdem sind diese Knoten meist batteriebetrieben und ihre Energie sollte nicht für Aufgaben verwendet werden, die auch stationäre Knoten übernehmen können. Damit mobile Knoten genau wie stationäre Knoten immer über einen direkten Nachbarknoten Zugriff auf die Service Registry haben, kann man wie in Kapitel 3.3 *Access-Knoten* definieren. Da mobile Knoten immer direkte Nachbarn eines Access-Knotens sein müssen, genügt es, alle Access-Knoten als Registry-Knoten auszuwählen. Außerdem ist es denkbar, dass ein Knoten das drahtlose Netzwerk mit einem drahtgebundenen Netzwerk verbindet, sodass dieser Knoten auch auf jeden Fall die Rolle eines Registry-Knotens übernehmen sollte, um dem drahtgebundenen Netzwerk effizienten Zugriff auf die Registry zu ermöglichen. Das Clustering-Verfahren muss also ein heterogenes Netzwerkmodell unterstützen, in dem manche Knoten CH werden müssen, an-

dere Knoten weder CH noch Gateway werden dürfen und die übrigen Knoten nach Bedarf CH oder Gateway werden können.

In Kapitel 3.4.1 betrachten wir existierende Clustering-Verfahren und untersuchen diese auf die genannten Anforderungen. Kapitel 3.4.2 stellt *Heterogeneous Network Clustering (HNC)* vor, ein Verfahren, dass explizit für die Anforderungen in diesem Anwendungsszenario entwickelt wurde und zuerst in [KCG16] publiziert wurde. In Kapitel 3.4.3 wird die Implementierung von HNC für ProNet 4.0 vorgestellt und das Verfahren in einer Evaluierung mit anderen Verfahren verglichen. Kapitel 3.4.4 fasst das Kapitel zusammen.

3.4.1 Stand der Technik

Clustering-Verfahren sind ein gut erforschtes Forschungsfeld und in der Literatur wurde eine Vielzahl von Clustering-Verfahren mit unterschiedlichen Zielen vorgeschlagen. Insbesondere für Wireless Sensors Networks (WSNs) mit hunderten oder tausenden einfachen batteriebetriebenen Sensorknoten in einem Netzwerk bieten Clustering-Verfahren Vorteile in Bezug auf Skalierbarkeit, Energiebedarf und Komplexität. Eine gute Übersicht über existierende Verfahren für WSNs geben Abbasi und Younis [AY07] und Liu [Liu12]. Auch für ad-hoc Netzwerke wurde eine Vielzahl von Clustering-Verfahren vorgeschlagen, einen Überblick bieten Yu und Chong in [YC05] und Bentaleb et al. in [BBH13].

Für das Bilden eines 1-Hop Dominating Sets existieren diverse Verfahren, beispielsweise [DB97] und [WL99]. Diese unterstützen aber das geforderte heterogene Netzwerkmodell nicht, bei dem einige Knoten CH werden müssen, andere hingegen nicht als CH gewählt werden dürfen. Die meisten Verfahren lassen sich leicht anpassen, indem man CHs, die CH werden müssen, vor dem Start des Verfahrens in die Menge der gewählten CHs aufnimmt. Der Ausschluss von Knoten bei der Wahl als CH lässt sich dagegen nicht so einfach erzielen. Würde man die auszuschließenden Knoten aus dem Graphen entfernen, auf dem der Clustering-Algorithmus arbeitet, und hinterher wieder einfügen, so wäre das Ergebnis im Allgemeinen kein Dominating Set mehr. Die Ansätze [Jav+13] und [QZW06] unterstützen zwar heterogene Netzwerke, wählen aber keine Gateways für die Inter-Cluster-Konnektivität aus.

Wir betrachten nun zwei bekannte Clustering-Verfahren genauer und untersuchen dabei, in wie weit sie die geforderten Anforderungen erfüllen.

3.4.1.1 Linked Clustering Algorithm (LCA)

Der Linked Clustering Algorithm (LCA) ist ein frühes Clustering-Verfahren für drahtlose Netzwerke, das 1981 von Baker und Ephremides veröffentlicht wurde [BE81] und für die Kommunikation von Schiffen der US Navy entwickelt worden war. Das Verfahren erzeugt ein *3-Hop connected 1-Hop Dominating Set*, d.h. ein Dominating Set, bei dem alle CHs über ein Overlay-Netzwerk aus Gateways und CHs verbunden sind, wobei max. 3 Hops zwischen zwei CHs liegen. Somit unterstützt das Verfahren die Inter- und Intra-Cluster-Konnektivität. Weitere Entwurfsziele waren hohe Verfügbarkeit und Fehlertoleranz. Diese Ziele erreicht das Verfahren durch ein ebenfalls spezifiziertes MAC-Verfahren, welches auf TDMA basiert und Kanalwechsel nutzt.

Baker und Ephremides beschreiben eine zentralisierte und eine verteilte Version von LCA. Die zentralisierte Version geht davon aus, dass die gesamte Kommunikationstopologie verfügbar ist. Dies ließe sich durch ein Verfahren zur Topologieerkennung wie ATDP erreichen (s. Kapitel 3.2).

Die verteilte Version von LCA erzielt das selbe Clustering-Ergebnis wie die zentralisierte Verfahren, benötigt aber die Kommunikationstopologie nicht als Eingabe. Die Topologieerkennung von LCA basiert aber, anders als ATDP, auf einzelnen Beobachtungen, überprüft die Zuverlässigkeit von Links also nicht mehrfach. In diesem Kapitel legen wir den Fokus auf Clustering und betrachten daher die verteilte Version von LCA mit ihrer Topologieerkennung nicht genauer.

Das LCA-Verfahren geht davon aus, dass allen Knoten eindeutige Knoten-IDs aus der Menge V haben und auf $V \times V$ eine starke Totalordnung definiert ist. Das Verfahren wählt zunächst den Knoten mit der größten Knoten-ID als CH und seine direkten Nachbarn als seine Follower. Anschließend wird aus den verbleibenden Knoten der Knoten mit der größten Knoten-ID als CH gewählt und seine noch nicht klassifizierten direkten Nachbarn zu seinen Followern. Auf diese Weise fährt das Verfahren solange fort, bis alle Knoten entweder als CH oder Follower eingestuft wurden. Anschließend werden aus der Menge der Follower Knoten als Gateways ausgewählt, um die Konnektivität zwischen den Clustern zu unterstützen. Das Verfahren tendiert dazu, CHs mit großen Knoten-IDs größere Cluster zuzuweisen als CHs mit kleineren Knoten-IDs. LCA versucht nicht, die Cluster-Größe zu balancieren.

LCA liefert ein deterministisches Ergebnis, d.h. zu einer gegebenen Kommunikationstopologie als Eingabe werden stets die selben Knoten als CH, Gateway bzw. Follower ausgewählt. In Verbindung mit ATDP könnte man daher die zentralisierte Variante von LCA auf allen Knoten parallel ausführen und würde auf allen Knoten das selbe Ergebnis erhalten, da ATDP die Konsistenz der Kommunikationstopologie sicherstellt. Auf diese Weise wäre es möglich, die Cluster zu bilden, ohne dass (über die Topologieerkennung hinaus) Kommunikation stattfinden muss.

LCA unterstützt kein heterogenes Netzwerkmodell, d.h. es ist keine Möglichkeit vorgesehen, manche Knoten von der Wahl zum CH auszuschließen oder Knoten immer CH werden zu lassen (obligatorische Knoten). Allerdings lässt sich das Verfahren leicht entsprechend anpassen. Um LCA mit dem in Kapitel 3.4.2 vorgestellten HNC vergleichen zu können, definieren wir daher eine angepasste Version LCA', welche das geforderte heterogene Netzwerkmodell unterstützt. LCA' ist mit LCA identisch bis auf die folgenden Anpassungen:

- Zunächst werden alle obligatorischen Knoten in der absteigenden Reihenfolge ihrer Knoten-IDs als CH gewählt.
- Anschließend werden CHs, Follower und Gateways so wie bei LCA gewählt, wobei jene Knoten bei der Wahl als CH oder Gateway übersprungen werden, die ausgeschlossen werden sollen.

3.4.1.2 Max-Min-D

Max-Min-D [Ami+00] ist ein bekanntes Clustering-Verfahren für drahtlos Ad-Hoc-Netzwerke. Das Verfahren bildet ein d-Hop Dominating Set, wobei über den Parameter d die Cluster-Anzahl und Größe konfiguriert werden kann. Es wählt Gateways für die Inter-Cluster-Konnektivität aus und verbindet die CHs mit bis zu $2d+1$ hops, weshalb man von einem $(2d+1)$-Hop Connected d-Hop Dominating Set spricht. Mit $d = 1$ erzeugt Max-Min-D genau wie LCA ein 3-Hop connected 1-Hop Dominating Set. Genau wie LCA geht Max-Min-d davon aus, dass Knoten eindeutige Knoten-IDs zugewiesen sind und darauf eine starke Totalordnung definiert ist. Im Gegensatz zu LCA versucht Max-Min-D, die Cluster-Größe zu balancieren, also möglichst gleich große Cluster zu bilden. Dies ist hilfreich, um die Last zwischen den CHs möglichst gleich zu verteilen. Max-Min-D läuft verteilt und enthält ebenso wie die verteilte Version von LCA Funktionalität zur

Topologieerkennung. Auf diese wurde bereits in Kapitel 3.2.1.2 genauer eingegangen, weshalb hier Max-Min-D in Hinsicht auf das Clustering untersucht wird.

Das Verfahren ist eingeteilt in vier Phasen. Die Floodmax-Phase ist die erste Phase und führt eine Vorauswahl der CHs durch. Dazu wird in d Runden die jeweils größte Knoten-ID in der d-hop Nachbarschaft ermittelt. Anschließend balanciert die Floodmin-Phase die Cluster-Größen. Die dritte Phase wählt schließlich die CHs anhand von vier Regeln aus und weißt ihnen Follower zu. Die letzte Phase wählt Gateway-Knoten aus, um die Inter-Cluster-Konnektivität herzustellen.

Das Max-Min-D Verfahren ist deterministisch und lässt sich auch zentralisiert ausführen, sofern die Kommunikationstopologie vorliegt. Wie bei LCA sind heterogene Netzwerke in Max-Min-D nicht vorgesehen. Daher definieren wir eine angepasste Variante, die wir Max-Min-D' nennen. Diese unterscheidet sich zu Max-Min-D in folgenden Punkten:

- Obligatorische Knoten behalten in der Floodmax-Phase und Floodmin-Phase immer ihre eigene ID und stellen so sicher, dass sie in der dritten Phase als CH ausgewählt werden.
- Ausgeschlossene Knoten senden in der Floodmax-Phase ihre eigene Knoten-ID nicht an ihre Nachbarn, sodass sie keine CH-Kandiaten werden können.
- Ausgeschlossene Knoten werden bei der Auswahl von Gateway-Knoten übersprungen.

3.4.2 Heterogeneous Network Clustering (HNC)

Heterogeneous Network Clustering (HNC) ist ein Clustering-Verfahren, das entwickelt wurde, um Knoten auszuwählen, die zusammen eine verteilte Service Registry bilden. Das Verfahren unterstützt ein heterogenes Netzwerkmodell, welches vorsieht, dass Knoten entweder CH werden müssen, CH werden können, oder nicht CH werden dürfen. Auf diese Weise können Unterschiede der Knoten berücksichtigt werden, z.B. in Bezug auf spezielle Aufgaben oder die verfügbare Energie der Knoten. Neben der Berücksichtigung der Knoten-Klassifikation versucht HNC, möglichst wenige Cluster zu bilden, um den Synchronisierungsaufwand der verteilten Service Registry zu verringern. Weiterhin stellt HNC mit der Wahl von Gateways die Inter-Cluster-Konnektivität her und optimiert diese.

Das Verfahren enthält selbst keine Funktionalität zur Topologieerkennung. Es geht stattdessen davon aus, dass ein Topologieerkennungsverfahren wie ATDP die Kommunikationstopologie bestimmt und netzweit ausgetauscht hat. Basierend auf der vorliegenden Topologie wird HNC auf allen Knoten ausgeführt. Da das Verfahren deterministisch ist, ergibt es bei gleicher Eingabe stets die gleiche Ausgabe. ATDP stellt sicher, dass die Eingabe, die Kommunikationstopologie, netzweit konsistent vorliegt. Berechnen alle Knoten parallel das Clustering mit den gleichen Eingabedaten, so erhalten sie folglich das gleiche Ergebnis. Ein Nachrichtenaustausch ist nicht erforderlich, da alle Knoten durch lokale Berechnung zu einem konsistenten Ergebnis kommen. Damit ist HNC auch kein Protokoll, sondern vielmehr ein Algorithmus. Die Eingabe für diesen Algorithmus besteht aus der Kommunikationstopologie und der Klassifikation der Knoten als obligatorische, optionale oder ausgeschlossene CHs. Die Kommunikationstopologie wird von ATDP übernommen, wobei als bidirektional erkannte Kommunikationslinks Berücksichtigung finden. Formal definieren wir die Eingabe als 4-Tupel $(G, V_{obl}, V_{opt}, V_{excl})$. Dabei sei:

- $G = (V, E)$: Die Kommunikationstopologie als Graph mit
  - V: Die Menge aller Knoten, identifiziert über eindeutige numerische Knoten-IDs.
  - $E \subseteq V \times V$: Die Menge der bidirektionalen Kommunikationslinks.
- $V_{obl} \subseteq V$: Die Menge der obligatorischen Knoten, die CH werden müssen.

- $V_{excl} \subseteq V$: Die Menge der ausgeschlossenen Knoten, die weder CH noch Gateway werden dürfen.
- $V_{opt} \subseteq V$: Die Menge der optionalen Knoten, die CH oder Gateway werden dürfen, aber nicht müssen.

Die Eingabe hat folgende Eigenschaften:

- Da die Links in E bidirektional sind, gilt $\forall (v, v') \in E : (v', v) \in E$.
- Die Mengen V_{obl}, V_{excl} und V_{opt} sind disjunkt.
- Jeder Knoten ist entweder obligatorisch, ausgeschlossen oder optional, sodass gilt: $V_{obl} \cup V_{excl} \cup V_{opt} = V$.

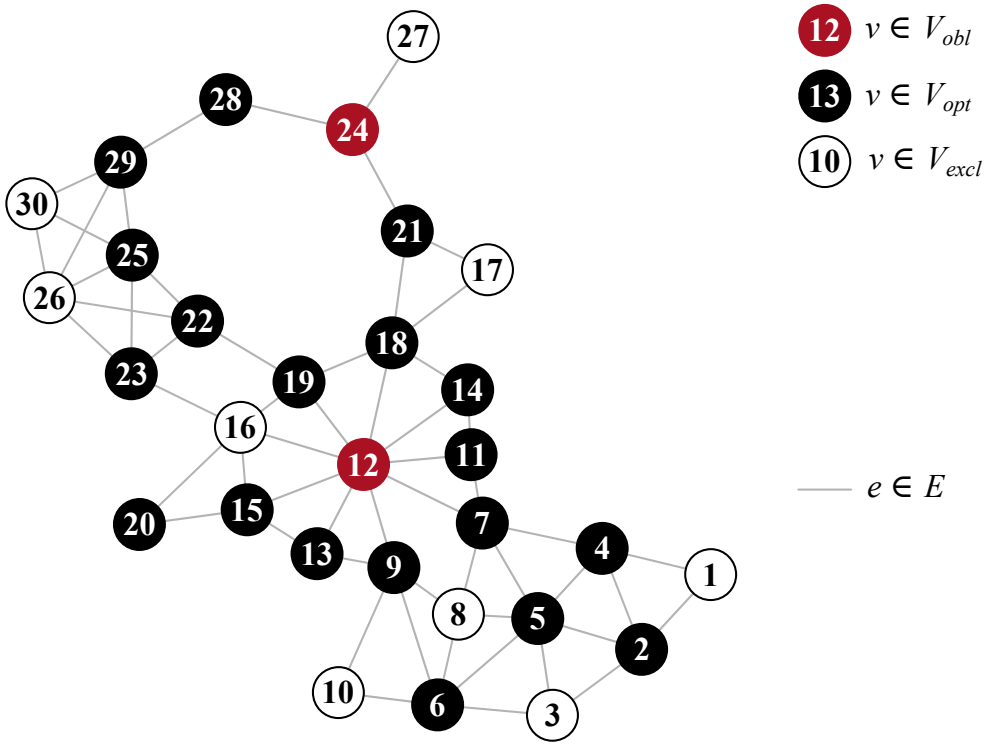


Abbildung 3.15: Beispiel eines Eingabe-Graphen mit obligatorischen Knoten (V_{obl}), ausgeschlossenen Knoten (V_{excl}) und optionalen Knoten (V_{opt}) sowie den bidirektionalen Links E.

In Abb. 3.15 ist eine beispielhafte Eingabe als Graph dargestellt.

Die Ausgabe von HNC umfasst neben den gewählten CHs und Gateways auch die Zuordnung der Follower zu ihrem Gateway sowie die resultierenden Overlay-Topologien. Formal definieren wir die Ausgabe als 5-Tupel $(V_{CH}, V_{GW}, E_{follow}, G_{router}, G_{out})$. Dabei sei:

- $V_{CH} \subseteq V_{obl} \cup V_{opt}$: Die Menge der als CH ausgewählten Knoten.
- $V_{GW} \subseteq V_{opt}$: Die Menge der als Gateway ausgewählten Knoten.
- $E_{follow} \subseteq E$: Die Menge der Links, welche die Follower mit ihren CHs verbinden.
- $G_{router} = (V_{router}, E_{router}$: Graph des Router-Netzwerks.
  - $V_{router} = V_{CH} \cup V_{GW}$: Die Menge der als Router, d.h. CH oder Gateway, ausgewählten Knoten.
  - E_{router}: Die Menge der Links, die zwei Router verbinden.
- $G_{out} = (V, E_{out}$: Graph des Router-Netzwerks sowie der Follower und den Verbindungen zu ihren CHs.

– $E_{out} = E_{router} \cup E_{follow}$: Die Menge der Links, die Router miteinander verbinden sowie Follower mit ihren CHs.

Die Ausgabe hat folgende weitere Eigenschaften:

- V_{CH} ist ein 1-Hop Dominating Set des Graphen G.
- Jeder obligatorische Knoten wird CH, d.h. $V_{obl} \subseteq V_{CH}$
- Genau wie E enthalten auch dessen Teilmengen E_{follow}, E_{router} und E_{out} nur bidirektionale Links.
- G_{router} und G_{out} sind zusammenhängende Teilgraphen von G.
- E_{router} enthält alle Links aus E, die Router verbinden, d.h. formal gilt: $\forall (v,v') \in E | v,v' \in V_{router} : (v,v') \in E_{router} \wedge (v',v) \in E_{router}$

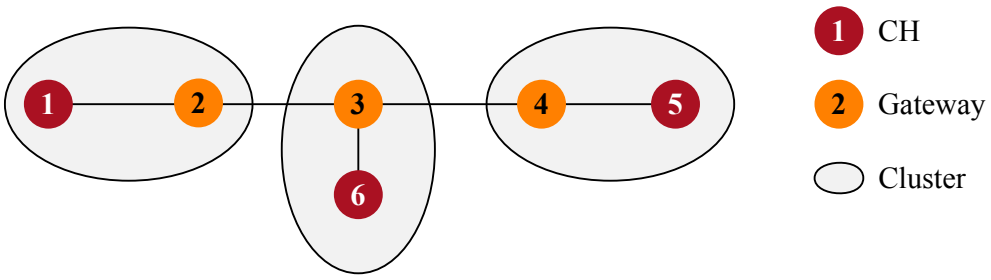


Abbildung 3.16: Dieses Beispiel verdeutlicht: Ein 1-Hop Dominating Set ist immmer 3-Hop-connected.

Aus der Tatsache, dass HNC ein 1-Hop Dominating Set bildet, folgt automatisch, dass dieses Dominating Set 3-Hop connected ist, sofern die dazu notwendigen Knoten als Gateway gewählt werden. Anders ausgedrückt können die CHs eines 1-Hop Dominating Sets immer miteinander verbunden werden, indem man max. zwei Zwischenknoten als Router auswählt. Dies lässt sich leicht an Abb. 3.16 nachvollziehen: Die CHs 1, 5 und 6 bilden ein 1-Hop Dominating Set. Sie können verbunden werden, indem zwischen 1 und 6 die beiden Knoten 2 und 3 als Router gewählt werden und zwischen Knoten 6 und 5 die Knoten 3 und 4. Somit sind die CHs über zwei Zwischenknoten bzw. 3 Hops miteinander verbunden, es handelt sich um ein 3-Hop connected 1-Hop Dominating Set. Ignorieren wir Knoten 6, so sind die CHs 1 und 5 über 4 Hops verbunden. Allerdings handelt es sich dann nicht mehr um ein 1-Hop Dominating Set, da Knoten 3 keinen CH als direkten Nachbarn mehr hat. Damit es sich um ein 1-Hop Dominating Set handelt, muss es also einen direkten Nachbarn von Knoten 3 geben, der CH ist. Ein solcher Knoten würde aber, genau wie Knoten 6, dazu führen, dass das Dominating Set 3-Hop connected ist.

Der Algorithmus von HNC besteht aus 6 Schritten, die nun beschrieben und anhand des Beispiels in Abb. 3.15 veranschaulicht werden. An einigen Stellen im Algorithmus können mehrere Knoten ein Auswahlkriterium erfüllen. Damit der Algorithmus deterministisch bleibt, muss die Reihenfolge, in der Knoten gewählt werden, aber eindeutig sein. Daher wird festgelegt, dass in diesen Fällen die Knoten in der aufsteigenden Reihenfolge ihrer Knoten-ID gewählt werden, d.h. der Knoten mit der kleinsten Knoten-ID zuerst.

3.4.2.1 Schritt 1: Obligatorische Knoten

Im ersten Schritt wählt der Algorithmus die obligatorischen Knoten aus V_{obl} als CH aus, sodass anschließend $V_{CH} = V_{obl}$ gilt. Da wir unter Routern neben Gateways auch CHs verstehen, werden die obligatorischen Knoten auch zur Menge der Router hinzugefügt, sodass $V_{router} = V_{obl}$ gilt. Immer, wenn ein Knoten als CH bestimmt und zur Menge V_{CH} hinzugefügt wird, werden

die ihn umgebenden Knoten, die noch keinem Cluster zugeordnet wurden, ihm als Follower zugeordnet. Dazu werden die Links, welche die Follower mit ihrem CH verbinden, zu E_{follow} und E_{out} hinzugefügt. Außerdem wird das Router-Netzwerk G_{router} immer aktualisiert, wenn ein Knoten als CH ausgewählt wird. Dazu wird der neue CH mit allen Nachbar-CHs verbunden, indem die entsprechenden Kanten zu E_{router} und E_{out} hinzugefügt werden.

Im Verlauf des Clusterings wählt HNC nach und nach CHs und Gateways aus, bis das Router-Netzwerk G_{router} ein verbundener Teilgraph des Ursprungsgraphen G ist. Um zu entscheiden, ob und wo noch Gateways benötigt werden, führt HNC über die gebildeten *Partitionen* Buch in einer Partitionstabelle. Als Partition wird ein größtmöglicher, verbundener Teilgraph des Router-Netzwerks G_{router} verstanden. Jeder Partition wird eine Partitionsnummer zugeordnet, die sich aus der kleinsten Knoten-ID der enthaltenen Router ergibt.

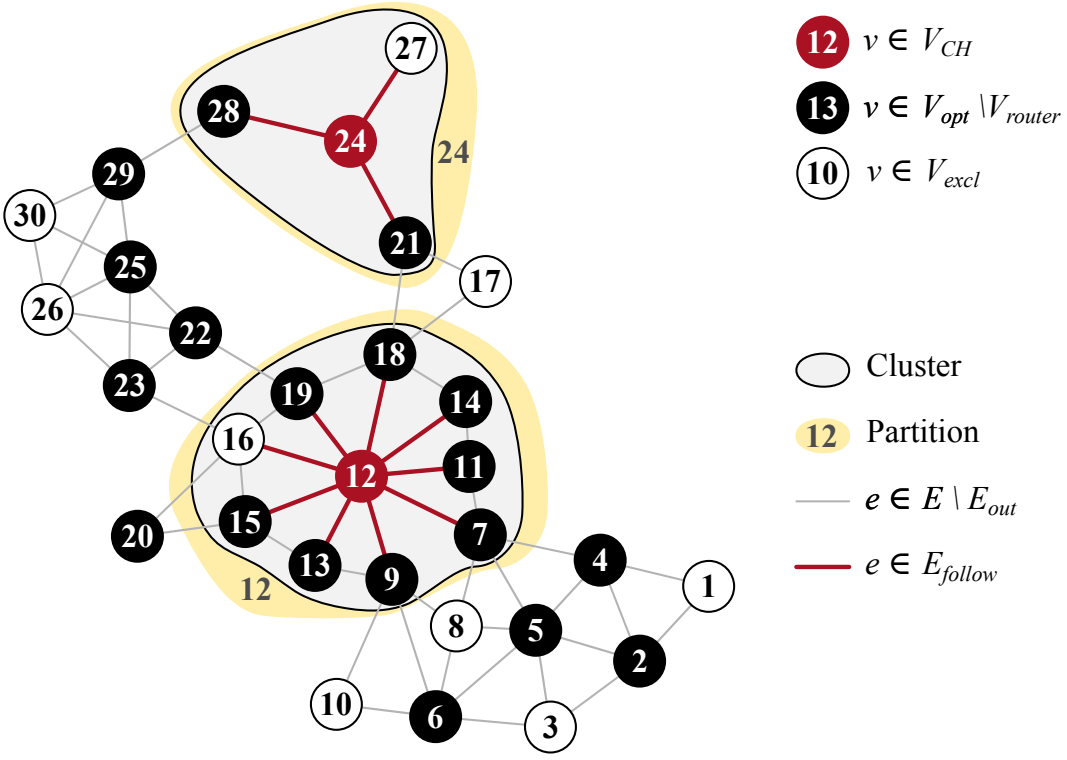


Abbildung 3.17: Ergebnis des ersten Schritts: Die obligatorischen Knoten 12 und 24 wurden als CH gewählt und bilden zwei nicht verbundene Cluster und damit zwei Partitionen.

Abb. 3.17 illustriert das Ergebnis des ersten Schritts, angewendet auf das Beispiel aus Abb. 3.15. Die obligatorischen Knoten 12 und 24 wurden als CH gewählt. Ihre Nachbarn wurden ihnen als Follower zugeordnet und die entsprechenden Links zu E_{follow} hinzugefügt. Da die beiden Cluster nicht verbunden sind, bilden sie zwei Partitionen, wobei die Partitionsnummern den Knoten-IDs der CHs entsprechen.

3.4.2.2 Schritt 2: Erzeugung eines Dominating Sets

Der zweite Schritt formt ein 1-Hop Dominating Set. Dazu wählt er solange optionale Knoten aus V_{opt} als CH aus, bis ein Dominating Set gebildet ist, d.h. bis jeder Knoten entweder CH ist oder einen solchen als direkten Nachbarn hat. Wie im vorigen Schritt werden nach dem Auswählen eines neuen CHs die umliegenden Knoten als Follower zugeordnet, falls sie nicht bereits einem

Cluster angehören. Außerdem werden wie zuvor die Ausgabegraphen G_{router} und G_{out} sowie die Partitionstabelle aktualisiert.

Die Reihenfolge der Knotenauswahl erfolgt anhand von zwei Heuristiken. Damit möglichst wenige Cluster gebildet werden, beginnt der Algorithmus mit jenen Knoten, die möglichst viele noch nicht abgedeckte Knoten abdecken können. Hierbei wird neben den Followern auch der CH selbst mitgezählt, sofern er bisher noch nicht abgedeckt war. Falls es mehrere Kandidaten gibt, die gleich viele Knoten abdecken können, so wird die zweite Heuristik angewendet. Sie bevorzugt Knoten, die bereits von einem anderen CH abgedeckt sind. Solche Knoten haben eine direkte Verbindung zu einem benachbarten CH, benötigen also keine weiteren Gateway-Knoten und bilden keine neue Partition. Dies trägt dazu bei, die Inter-Cluster-Konnektivität zu optimieren.

Da HNC nach und nach versucht alle optionalen Knoten als CH auszuwählen, findet HNC auf jeden Fall ein Dominating Set, falls es ein solches gibt. Falls alle Knoten aus V_{opt} zum CH gewählt wurden und immer noch nicht alle Knoten abgedeckt worden sind, so ist es nicht möglich, aus optionalen Knoten ein 1-Hop Dominating Set zu bilden. Dieser Fall kann beispielsweise auftreten, wenn ein ausgeschlossener Knoten aus V_{excl} nur Nachbarn hat, die ebenfalls ausgeschlossen sind, sodass kein Nachbarknoten sein CH werden kann. HNC terminiert in solchen Fällen mit einem Fehler. In der Praxis müsste man in einem solchen Fall weitere optionale oder obligatorische Knoten in der Nähe der ausgeschlossenen Knoten installieren.

Das Ergebnis des zweiten Schritts veranschaulicht Abb. 3.18. Man kann leicht erkennen, dass nun jeder Knoten einem Cluster zugeordnet ist. Da alle Knoten entweder CH sind oder einen solchen als direkten Nachbarn haben, handelt es sich um ein 1-Hop Dominating Set. Die CHs 2, 5 und 6 sind direkte Nachbarn und bilden daher zusammen eine Partition, welche die kleinste Knoten-ID der drei Knoten (2) als Partitionsnummer trägt. Genauso bilden die CHs 12, 15 und 18 eine Partition mit der Nummer 12. Es verbleiben im Beispiel nach diesem Schritt vier Partitionen, d.h. es müssen noch Gateways gewählt werden, um die Inter-Cluster-Konnektivität sicherzustellen.

3.4.2.3 Schritt 3: Verknüpfen der Partitionen (1)

Der dritte Schritt wählt Knoten aus V_{opt} als Gateways aus, die (mindestens) zwei Partitionen miteinander verknüpfen können. Dazu werden Knoten gesucht, die momentan Follower sind, aber einen Nachbar-CH haben, der zu einer anderen Partition gehört. Um die Zahl benötigter Gateways gering zu halten, wird mit Knoten begonnen, die möglichst viele Partitionen vereinigen können. Jedes Mal, wenn ein Knoten als Gateway ausgewählt wird, werden wie zuvor die Ausgabegraphen und die Partitionstabelle aktualisiert.

Abb. 3.19 veranschaulicht das Ergebnis des dritten Schritts grafisch. Zunächst wurde Knoten 7 ausgewählt, um die Partitionen 2 und 12 zu verbinden. Anschließend führt die Auswahl von Knoten 21 zum Gateway dazu, dass Partition 2 erneut erweitert wird, da sie mit Partition 24 vereint wird. Wie man in Abb. 3.19 sehen kann, wurden die meisten Partitionen durch diesen Schritt miteinander verbunden. Lediglich Partition 25 konnte noch nicht mit Partition 2 verknüpft werden.

3.4.2.4 Schritt 4: Verknüpfen der Partitionen (2)

Ziel des vierten Schritts ist es, alle Cluster über Gateways zu verbinden und dabei alle Partitionen zu verknüpfen. Dazu sucht dieser Schritt Paare $(v, v') \in V_{opt} \times V_{opt}$, die zwei Partitionen

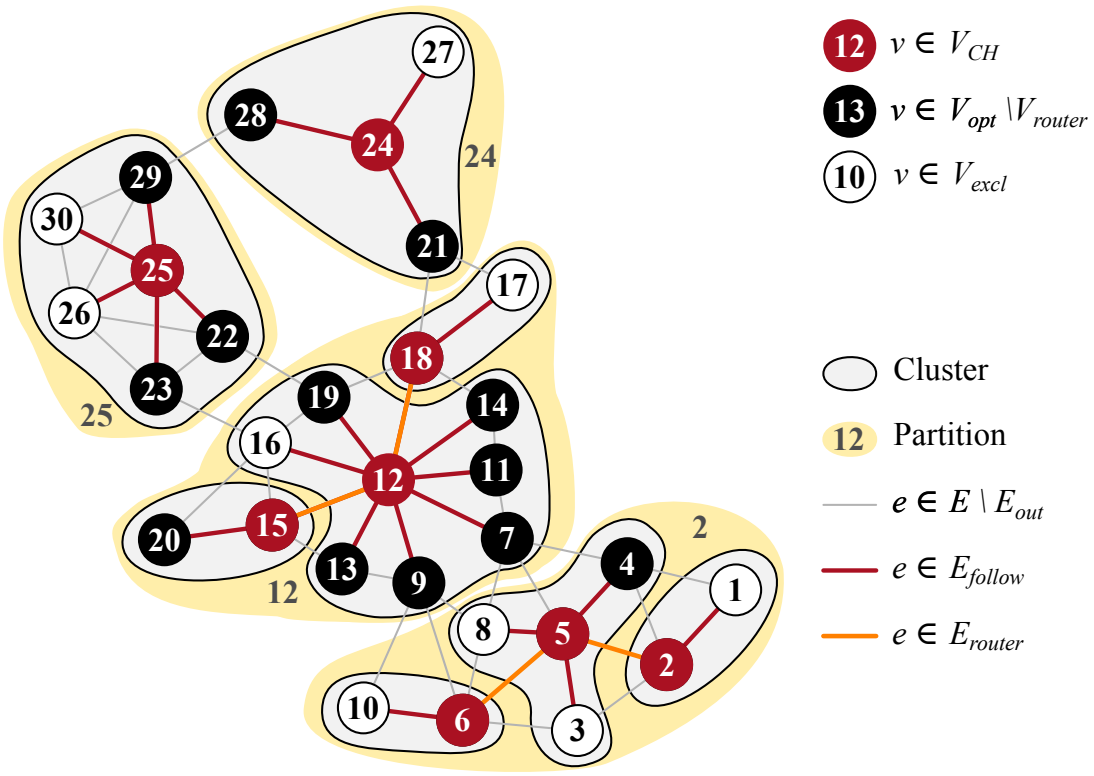


Abbildung 3.18: Ergebnis des zweiten Schritts: Ein 1-Hop Dominating Set wurde durch die Wahl von sechs weiteren CHs gebildet. Noch nicht alle Cluster sind miteinander verbunden, es bestehen noch vier Partitionen.

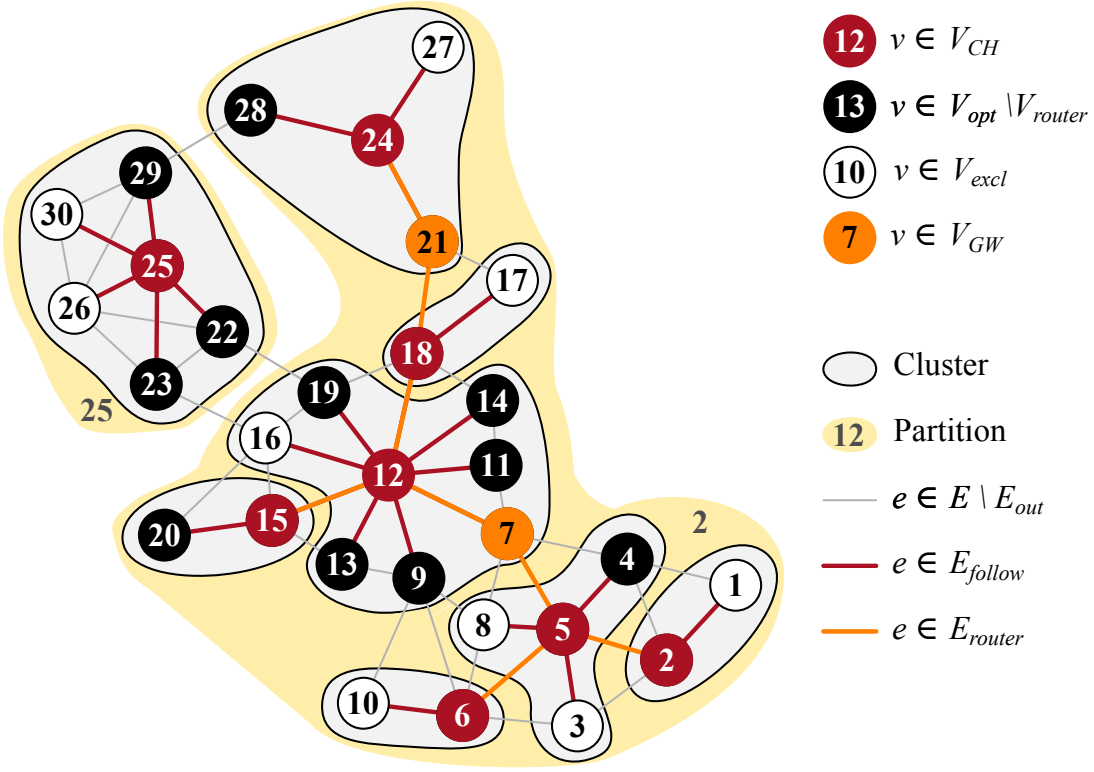


Abbildung 3.19: Ergebnis des dritten Schritts: Die Partitionen 2, 12 und 24 wurden durch Hinzufügen der Gateways 7 und 21 verknüpft.

verknüpfen können. Wie eingangs gezeigt wurde, lassen sich in einem 1-Hop Dominating Set alle Cluster über zwei Zwischenknoten bzw. 3 Hops verbinden, sodass ein 3-Hop connected 1-Hop Dominating Set entsteht. Da die Menge der CHs mit Abschluss von Schritt 2 ein 1-Hop Dominating Set ist, muss es daher möglich sein, alle Cluster mit maximal zwei Zwischenknoten zu verbinden. Jene Fälle, in denen dazu ein Zwischenknoten ausreicht, wurden im dritten Schritt abgedeckt. In diesem Schritt werden nun die restlichen Fälle abgedeckt, in denen Paare von Knoten notwendig sind, um zwei Partitionen zu verknüpfen. Da in diesem Schritt alle Paare von optionalen Knoten versucht werden, ist das Ergebnis diesen Schrittes ein 3-Hop connected 1-Hop Dominating Set, falls es möglich ist, dieses aus optionalen und obligatorischen Knoten zu bilden. Falls HNC alle Paare von optionalen Knoten als Gateway gewählt hat und es trotzdem noch mehr als eine Partition gibt, d.h. nicht alle Cluster verbunden wurden, bricht das Verfahren mit einem Fehler ab. Dieser Fall tritt auf, falls der Eingabegraph nicht verbunden ist oder aber Partitionen nur über ausgeschlossene Knoten verbunden sind. In der Praxis müssten in diesem Fall weitere optionale oder obligatorische Knoten installiert werden, damit die gewünschte Konnektivität erreicht werden kann.

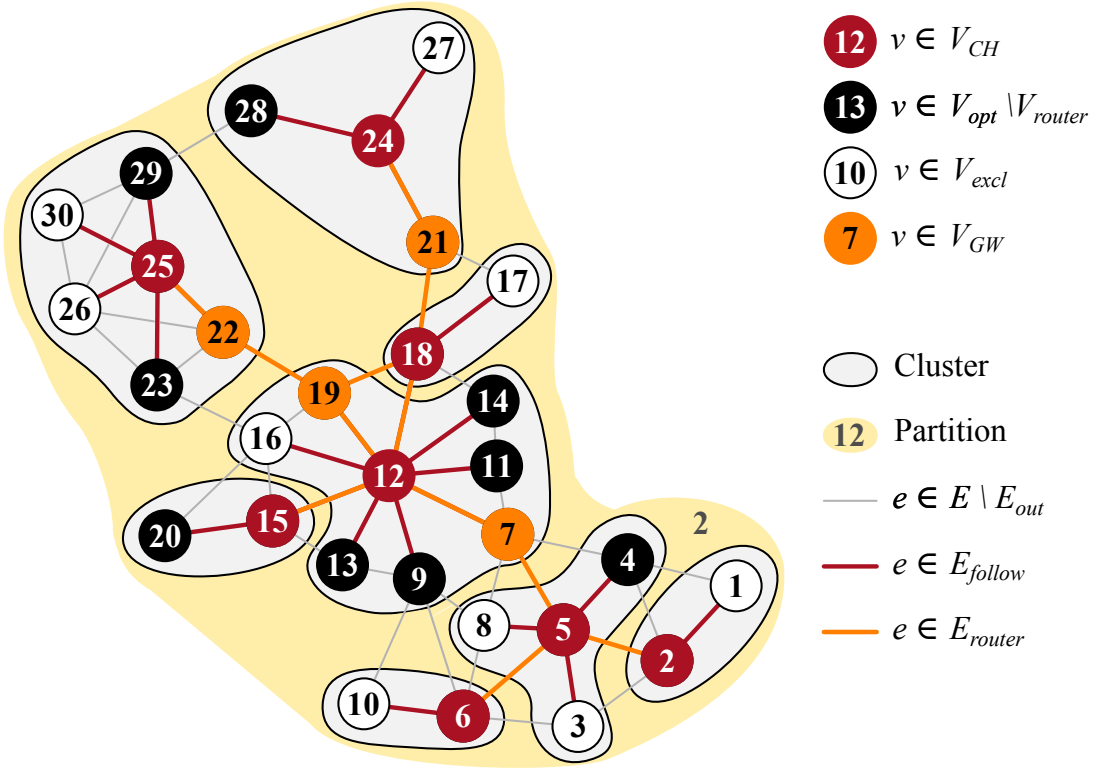


Abbildung 3.20: Ergebnis des vierten Schritts: Die Inter-Cluster-Konnektivität wurde durch die Wahl von Gateways hergestellt und ein 3-Hop connected 1-Hop Dominating Set gebildet.

Das Ergebnis des vierten Schritts illustriert Abb. 3.20. Das Knotenpaar (19,22) wurde hier als Gateways gewählt und damit die Partitionen 25 und 2 verknüpft, sodass alle Knoten nun zu Partition 2 gehören. Das Router-Netzwerk G_{router} ist damit ein verbundener Teilgraph von G und die Inter-Cluster-Konnektivität ist hergestellt.

3.4.2.5 Schritt 5: Inter-Cluster-Konnektivität optimieren (1)

In den letzten beiden Schritten soll die Inter-Cluster-Konnektivität optimiert werden. Dazu werden in diesem Schritt verbleibende optionale Knoten aus V_{opt} als Gateways ausgewählt, falls

dies dazu führt, dass der Pfad auf dem Router-Netzwerk G_{router} für mindestens ein Paar von CHs verkürzt wird.

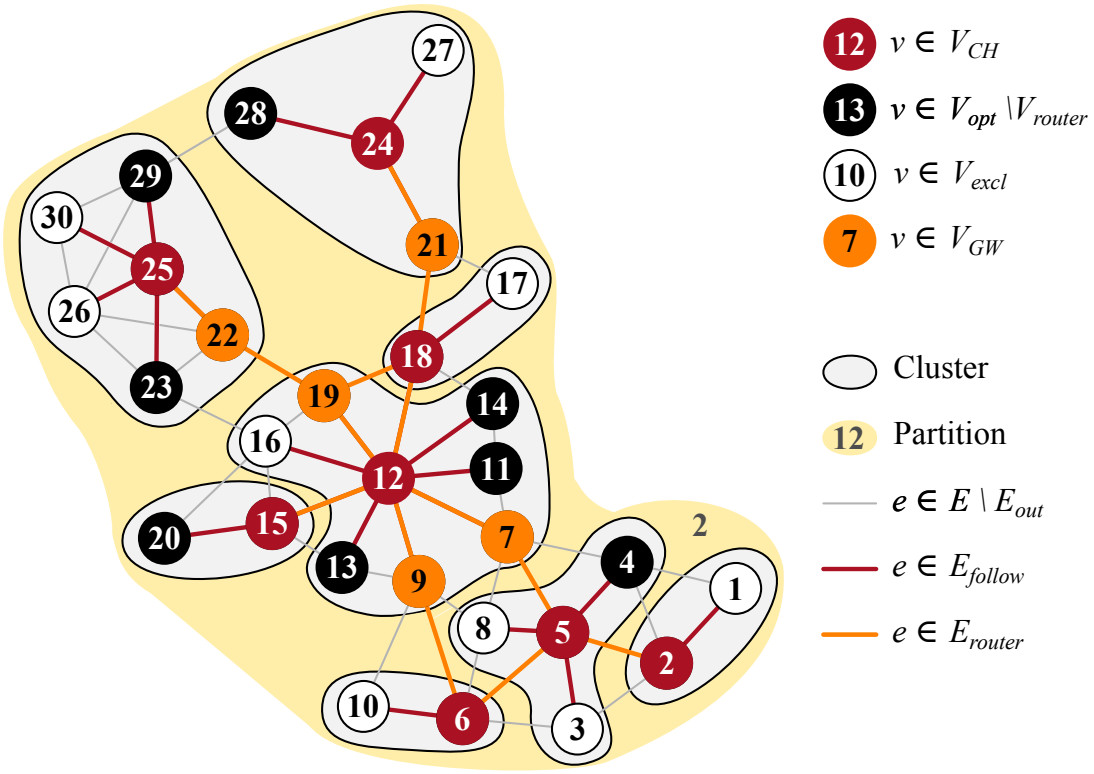


Abbildung 3.21: Ergebnis des fünften Schritts: Durch die Wahl von Knoten 9 als Gateway wurde der Pfad zwischen den CHs 6 und 12 verkürzt.

Das Ergebnis des Schritts im Beispiel zeigt Abb. 3.21. Hier wird Knoten 9 als Gateway ausgewählt und somit der Pfad zwischen den CHs 6 und 12 von drei auf zwei Hops verkürzt.

3.4.2.6 Schritt 6: Inter-Cluster-Konnektivität optimieren (2)

Der letzte Schritt verbessert die Inter-Cluster-Konnektivität weiter. Dazu werden nun Paare von Knoten aus V_{opt} als Gateways ausgewählt, falls dies die Pfade zwischen mind. zwei CHs verkürzt.

Das Ergebnis dieses Schrittes illustriert Abb. 3.22. Durch die Wahl der Knoten 28 und 29 als Gateways kann der Pfad zwischen den CHs 24 und 25 von fünf auf drei Hops verkürzt werden.

3.4.3 Implementierung und Evaluierung

In ProNet 4.0 findet eine Implementierung von HNC Verwendung in der dienstorientierten Middleware-Schicht ProMid, die in Kapitel 3.5 detaillierter betrachtet wird. Genau wie ProMid und ProNet 4.0 wurde die Implementierung von HNC in C++ geschrieben und kann auf der Imote 2 Hardware ausgeführt werden (s. Kapitel 3.1.2). Diese Sensorknoten können sowohl mit Batterien als auch mit einem festen Stromanschluss mit Energie versorgt werden. In einem Projekt mit Partnern aus der Wirtschaft wurde ProNet 4.0 in einer Demonstrator-Produktionsanlage zur Produktion von Seifenlauge eingesetzt. In diesem Szenario wurde ein batteriebetriebener Knoten auf einem Service-Roboter installiert, der sich frei im Raum bewegen kann. In ProMid

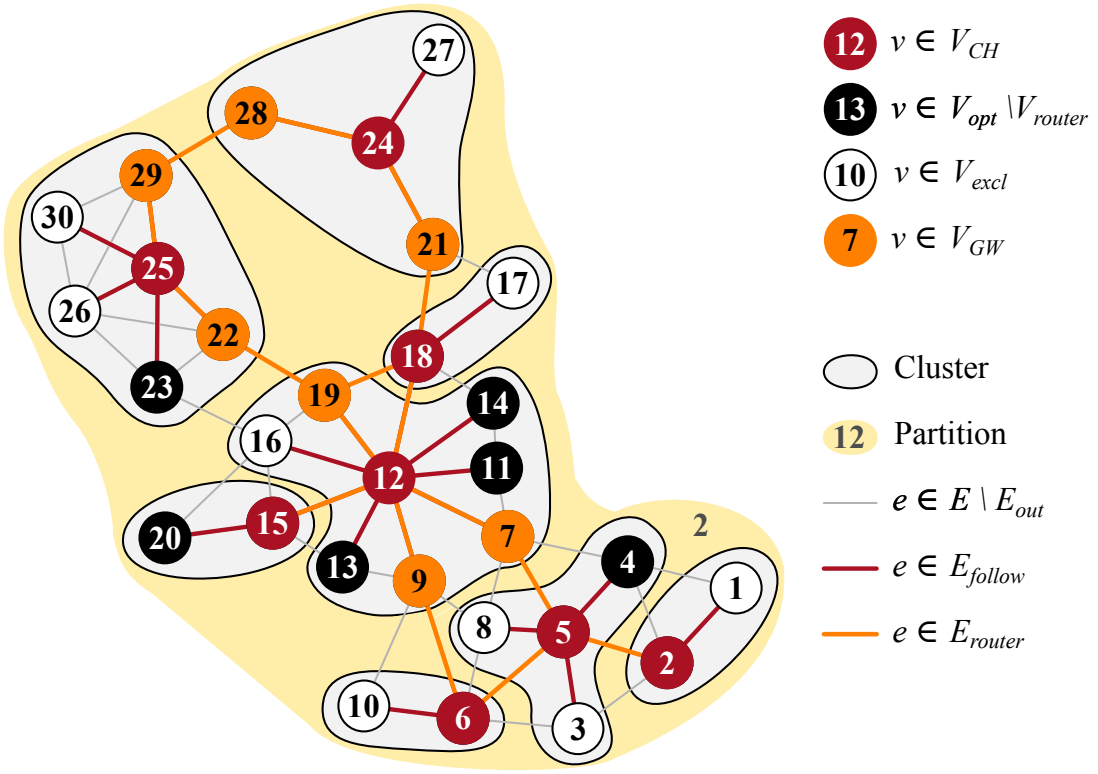


Abbildung 3.22: Ergebnis des sechsten Schritts: Die Inter-Cluster-Konnektivität wurde durch die Wahl von zusätzlichen Gateways optimiert und so die Pfade zwischen den CHs 24 und 25 verkürzt.

werden die batteriebetriebenen Knoten als ausgeschlossene Knoten (V_{excl}) behandelt, sodass zum Bereitstellen und Synchronisieren der verteilten Service Registry nicht die Energie der Batterien verwendet wird. Außerdem gibt es in diesem Szenario einen Knoten, der als Gateway zu einem drahtgebundenen Netzwerk agiert. Um einen schnellen und effizienten Zugriff auf die Service Registry zu erzielen, wurde dieser Knoten im Szenario als obligatorischer Knoten (V_{obl}) konfiguriert. Abb. 3.23 zeigt ein Foto der Produktionsanlage und der darin platzierten Knoten.

Neben den Tests unter realen Bedingungen in der Produktionsanlage wurde HNC auch mittels Simulationen evaluiert. Dies ermöglicht es, das Verhalten von HNC in vielen unterschiedlichen und größeren Topologien zu untersuchen. Um die Performance von HNC einschätzen zu können, wurde es mit den in Kapitel 3.4.1 vorgestellten Verfahren LCA' und Max-Min-D' verglichen. Dabei handelt es sich um auf das heterogene Netzwerkmodell angepasste Varianten der bekannten Verfahren LCA und Max-Min-D. Mit $\text{HNC}_{\text{reduced}}$ wurde außerdem eine reduzierte Variante von HNC evaluiert, welche die letzten beiden Schritte von HNC nicht enthält. So kann der Effekt dieser beiden Schritte untersucht werden.

Um die Verfahren zu vergleichen, wurden 1000 Topologien zufällig erzeugt. Dabei kam eine angepasste Version des Topologie-Generators aus [Nis08] zum Einsatz. Jede der Topologien besteht aus 100 Knoten, die auf einer Fläche von 100 m Breite und 100 m Länge zufällig platziert wurden. Knoten, die maximal 14 m voneinander entfernt sind, werden mit Links verbunden. Knoten werden mit einer Wahrscheinlichkeit von 10 % als ausgeschlossene Knoten und mit einer Wahrscheinlichkeit von 5 % als obligatorische Knoten kategorisiert. Nachdem eine Topologie generiert wurde, wird überprüft, ob die Topologie verbunden ist, d.h. zwischen jedem Paar von Knoten ein Pfad existiert. Ist dies nicht der Fall, wird die Topologie verworfen und stattdessen eine neue generiert.

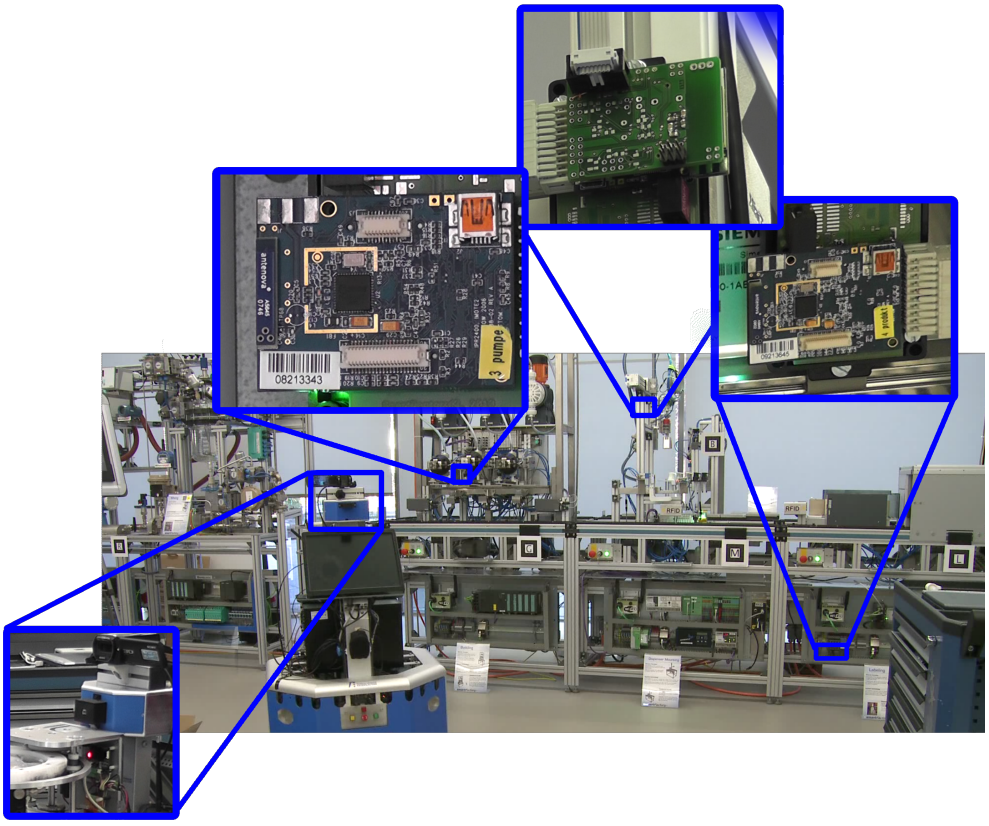


Abbildung 3.23: Produktionsanlage mit drei fest in der Anlage montierten Imote 2 Knoten und einem auf einem mobilen Roboter montierten Knoten.

Die 1000 so generierten Topologien wurden gespeichert und anschließend die Verfahren HNC, $HNC_{reduced}$, LCA', Max-Min-D' darauf angewendet. Die sich daraus resultierten Clustering-Ergebnisse wurden anschließend statistisch ausgewertet. Die Ergebnisse fasst Tab. 3.7 zusammen.

Tabelle 3.7: Statistische Auswertung der Ergebnisse.

Verfahren	Cluster	Gateways	Router	Max. Cluster-Größe
HNC	22,4	38,9	61,3	10,8
$HNC_{reduced}$	22,4	18,1	40,5	10,8
LCA'	29,8	36,1	65,9	9,5
Max-Min-D'	32,7	52,5	85,2	8,4

Eines der Ziele von HNC war es, eine möglichst geringe Anzahl Cluster zu bilden, da dies den Synchronisationsaufwand der verteilten Registry reduziert. Die Ergebnisse zeigen, dass HNC mit durchschnittlich 22,4 deutlich weniger Cluster als LCA' (28,8) und Max-Min-D' (32,7) bildet. $HNC_{reduced}$ unterscheidet sich nur in der Auswahl von Gateways von HNC und bildet daher gleich viele Cluster. LCA' erzeugt hier etwas weniger Cluster als Max-Min-D'. Dies steht im Gegensatz zur Evaluation, welche die Autoren von Max-Min-D in [Ami+00] aufführen, bei der Max-Min-D weniger Cluster bildet als LCA. Der Effekt entsteht vermutlich durch die Anpassungen der Verfahren: Bei LCA' werden die obligatorischen CHs zuerst als CH bestimmt und im weiteren Verlauf berücksichtigt, sodass dadurch nicht unbedingt mehr Cluster als notwendig

entstehen. Bei Max-Min-D′ hingegen behalten die obligatorischen Knoten ihre eigene ID, um sicherzustellen, dass sie CHs werden. Dies kann oft dazu führen, dass das Verfahren zusätzlich weitere CHs wählt und somit mehr Cluster gebildet werden, als nötig. Bei durchschnittlich fünf obligatorischen Knoten kann dies durchschnittlich bis zu fünf zusätzliche Cluster erklären.

Bei der Betrachtung der Gateways unterscheiden sich HNC und $\text{HNC}_{\text{reduced}}$ stark: Die reduzierte Variante wählt mit durchschnittlich 18,1 die wenigsten Gateways von allen Verfahren. Darauf folgt LCA′ mit durchschnittlich 36,1 Gateways, dicht gefolgt von HNC (38,9). Da HNC weniger Cluster bildet als LCA, liegt es nahe, dass es mehr Gateways benötigt. Betrachtet man die Anzahl Router, d.h. CHs und Gateways zusammengenommen, so zeigt sich, dass HNC mit durchschnittlich 61,3 etwas weniger Router als LCA′ (65,9) wählt. Max-Min-D′ wählt mit 52,5 Gateways deutlich mehr Gateways als die anderen Verfahren. Dadurch ergibt sich auch, dass Max-Min-D′ die meisten Router auswählt. Bei durchschnittlich 10 ausgeschlossenen Knoten sind durchschnittlich maximal 90 Router möglich, mit 85,2 wählt Max-Min-D′ also fast jeden möglichen Knoten aus. Dies liegt daran, dass das Verfahren jeden Knoten als Gateway auswählt, der an der Grenze zu einem anderen Cluster liegt. Bei einem Clusterdurchmesser von $d = 1$ sind dies fast alle Knoten, lediglich am Rand gelegene Knoten werden nicht als Gateway ausgewählt.

Die maximale Cluster-Größe von HNC und $\text{HNC}_{\text{reduced}}$ ist mit 10,8 größer als bei LCA′ (9,5) und Max-Min-D′ (8,4). Dies liegt in erster Linie an der Heuristik, die Knoten als CH wählt, die möglichst viele Knoten abdecken. Das Ziel, möglichst wenige Cluster zu erzeugen, führt natürlich auch dazu, dass größere Cluster gebildet werden. Bei der Anwendung von HNC zur Auswahl von Knoten einer verteilten Service Registry sind große Cluster kein Problem, daher war das Balancieren der Cluster auch kein Ziel beim Entwurf des Algorithmus. Von den untersuchten Verfahren ist Max-Min-D′ das einzige, das balancierte Cluster als Zielsetzung hat. Für Anwendungen, in denen balancierte Cluster vorteilhaft sind, könnte man in HNC zwischen dem zweiten und dritten Schritt einen weiteren Schritt einfügen, der die Cluster balanciert.

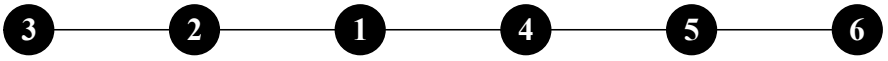


Abbildung 3.24: Beispieltopologie, bei der HNC kein minimales 1-Hop Dominating Set bildet.

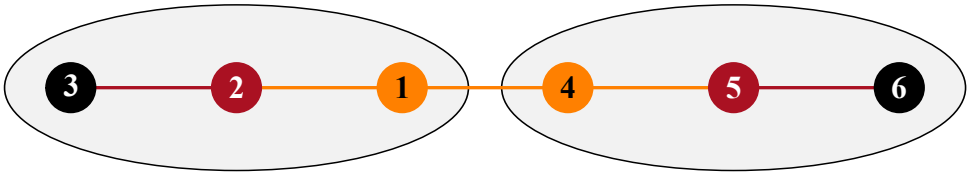


Abbildung 3.25: Minimales 1-Hop Dominating Set zur Topologie aus Abb. 3.24.

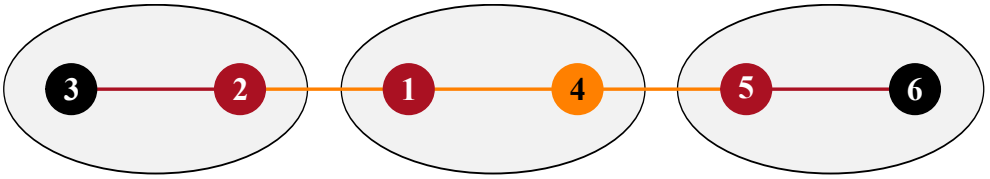


Abbildung 3.26: Clustering-Ergebnis von HNC für die Topologie aus Abb. 3.24.

Obwohl die Ergebnisse zeigen, dass es HNC gelingt, wenige Cluster zu bilden, so ist zu beachten, dass die gewählten Heuristiken nicht ausreichen, um in jedem Fall ein minimales 1-Hop Dominating Set zu bilden. Eine Eingabetopologie, für die HNC kein minimales Ergebnis liefert, zeigt Abb. 3.24. Das minimale 1-Hop Dominating Set für diese Topologie besteht, wie in Abb. 3.25 gezeigt, aus zwei Clustern. HNC bildet dagegen wie in Abb. 3.26 dargestellt drei Cluster. Wir betrachten den zweiten Schritt von HNC, in dem das Dominating Set gebildet wird, genauer. Das Verfahren wählt den Knoten als CH aus, der die meisten noch nicht abgedeckten Knoten abdecken kann. Im Beispiel können alle Knoten bis auf die äußeren Knoten 3 und 6 ihre beiden Nachbarn und sich selbst und damit drei Knoten abdecken. Die Knoten 3 und 6 können nur einen Nachbarn und sich selbst und damit weniger Knoten abdecken. Als Kandidaten verbleiben also die Knoten 1, 2, 4, und 5. Die zweite Heuristik wählt Knoten aus, die bereits von einem CH abgedeckt sind, was hier auf keinen Knoten zutrifft, da es noch keinen CH gibt. Das Verfahren wählt daher aus den Kandidaten den Knoten mit der kleinsten Knoten-ID aus, also Knoten 1. Mit dieser Wahl kann das optimale Ergebnis nicht mehr erreicht werden. Wären die Knoten-IDs anders vergeben, hätte HNC das optimale Ergebnis erreichen können. Dies zeigt, dass die Heuristiken keineswegs in allen Fällen zu einem minimalen Ergebnis führen.

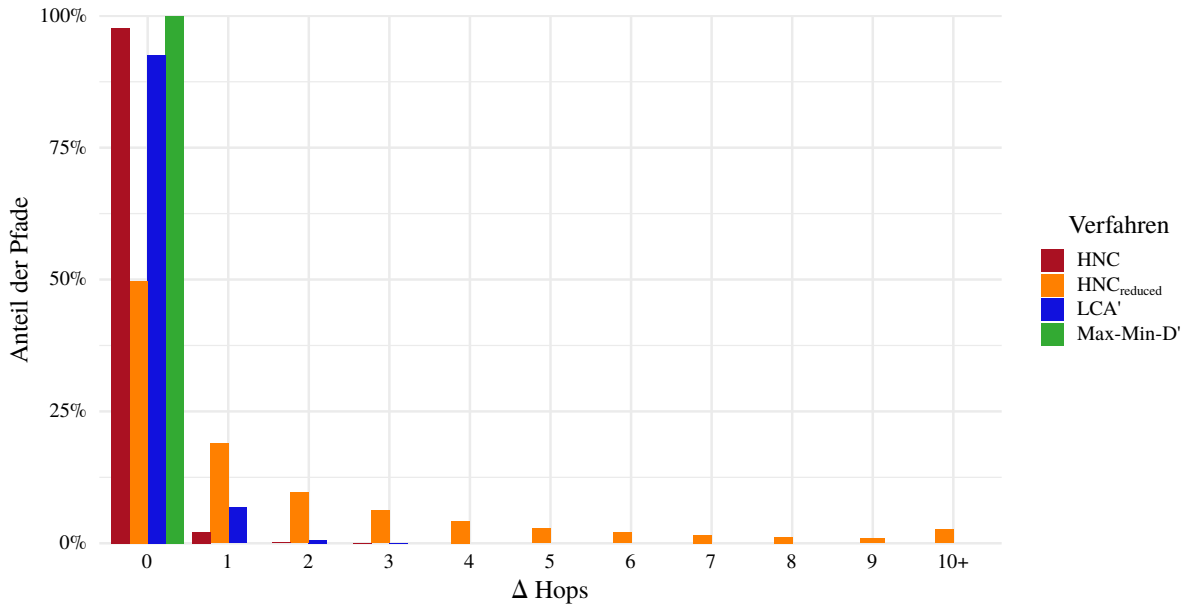


Abbildung 3.27: Anteil der Pfade zwischen Paaren von CHs, die Δ Hops länger sind als der kürzest mögliche Pfad.

Wir haben in Tab. 3.7 gesehen, dass HNC deutlich mehr Gateways auswählt als HNC$_{\text{reduced}}$. Ob und wie gut die zusätzlichen Gateways die Inter-Cluster-Konnektivität verbessern, soll nun betrachtet werden. Dazu wurde bei allen 1000 zufällig generierten Topologien die Länge der kürzesten Pfade auf dem Router-Netzwerk G_{router} zwischen allen Paaren von CHs berechnet. Diese wurde verglichen mit der minimal möglichen Pfadlänge, die sich ergibt, wenn alle optionalen Knoten als Router gewählt werden. Abb. 3.27 visualisiert die Ergebnisse für alle vier Verfahren. Aufgetragen ist der Anteil der Pfade zwischen Paaren von CHs, die Δ Hops länger sind als der kürzest mögliche Pfad. Für Max-Min-D' haben 100 % der Pfade die minimal mögliche Länge mit $\Delta = 0$. Dies ist nicht verwunderlich, da Max-Min-D' fast alle optionalen Knoten als Gateways auswählt. HNC verbindet mit 97,7 % fast genauso viele Pfade mit der optimalen Länge, benötigt dazu aber deutlich weniger Gateways (s. Tab. 3.7). Außerdem gab es kein einziges Paar

von CHs, dessen Pfad um mehr als $\Delta = 3$ Hops kürzer sein könnte. Die Ergebnisse von LCA' sind etwas schlechter als die von HNC, hier haben nur 92,6 % der Pfade die optimale Länge. Bei $\text{HNC}_{\text{reduced}}$ sind die Pfade dagegen deutlich länger: Nur knapp 50 % der Pfade sind mit der optimalen Länge verbunden, 2,6 % der Pfade sind sogar um 10 oder mehr Hops länger als der kürzest mögliche Pfad. Die Ergebnisse zeigen, dass die beiden letzten Schritt die Cluster-Konnektivität tatsächlich beträchtlich verbessern und dabei trotzdem die Anzahl Gateways gering halten.

3.4.4 Zusammenfassung

In diesem Kapitel haben wir uns mit Clustering-Verfahren beschäftigt. Diese kommen in drahtlosen Netzwerken meist zum Einsatz, um die Skalierbarkeit zu verbessern, indem das Netzwerk hierarchisch strukturiert wird. Oft wird diese hierarchische Struktur verwendet, um das Routing im Netzwerk zu vereinfachen. Aber auch andere Aufgaben können mit Hilfe von Clustering-Verfahren im Netzwerk hierarchisch strukturiert werden. In ProNet 4.0 gibt es eine verteilte Service Registry, in der die im Netzwerk verfügbaren Dienste registriert und nachgeschlagen werden können. Welche Knoten im Netzwerk die Rolle eines Service Registry-Knotens übernehmen, wird über ein Clustering-Verfahren bestimmt. Die vom Clustering-Verfahren als CHs ausgewählten Knoten übernehmen die Rolle eines Service Registry-Knotens, Gateways leiten Nachrichten weiter, die zur Synchronisation zwischen den Registrys regelmäßig ausgetauscht werden. Eine wichtige Anforderung in diesem Kontext ist die Heterogenität des Netzwerks: Einige Knoten sollen von der Wahl als Service Registry-Knoten ausgeschlossen werden, beispielsweise weil sie batteriebetrieben sind. Andere Knoten sollen auf jeden Fall Service Registry-Knoten werden, da sie den Zugang zum Service Registry für mobile Knoten oder ein drahtgebundenes Netzwerk bereitstellen. In Kapitel 3.4.1 haben wir existierende Clustering-Verfahren betrachtet. Keines der in der Literatur gefundenen Verfahren konnte alle geforderten Anforderungen erfüllen. Daher wurden mit LCA und Max-Min-D zwei bekannte Verfahren so angepasst, dass sie das heterogene Netzwerkmodell unterstützen.

Anschließend wurde mit Heterogeneous Network Clustering (HNC) in Kapitel 3.4.2 ein neues Clustering-Verfahren vorgestellt, welches speziell auf die gestellten Anforderungen entwickelt wurde. Das Verfahren bildet das geforderte 3-Hop connected 1-Hop Dominating Set und versucht dabei möglichst wenige Cluster zu bildet. Es berücksichtigt ein heterogenes Netzwerkmodell, indem manche Knoten von der Wahl als CH oder Gateway ausgeschlossen und andere als obligatorische CH festgelegt werden können. Durch die Wahl von Gateways bildet HNC ein Overlay-Netzwerk, dessen Konnektivität es mittels Heuristiken optimiert. HNC ist ein deterministischer Algorithmus, der bei gleicher Eingabetopologie stets die gleichen Cluster bildet. Daher kann HNC ohne jegliche Kommunikation zwischen den Knoten auskommen. Voraussetzung dafür ist, dass allen Knoten die Kommunikationstopologie bekannt ist und diese Information konsistent ist. In ProNet 4.0 wird dies durch das in Kapitel 3.2 vorgestellte Verfahren ATDP erreicht.

Die Implementierung von HNC in ProNet 4.0 stellte ihren Nutzen im Rahmen eines Demonstrators in der Praxis unter Beweis. Die simulative Evaluation von HNC hat darüber hinaus gezeigt, dass HNC weniger Cluster bildet als die Varianten von LCA und Max-Min-D, die das heterogene Netzwerkmodell unterstützen. Weiterhin hat die simulative Evaluation ergeben, dass die Schritte zur Optimierung der Inter-Cluster-Konnektivität diese erheblich verbessern und dabei nur eine geringe zusätzliche Anzahl Gateways hinzufügen.

3.5 Middleware

Als Middleware bezeichnet man eine Software-Schicht zwischen Betriebssystem und Anwendung, welche das Abstraktionslevel der Anwendung erhöht. Dies wird erreicht, indem gewisse Details, beispielsweise bezüglich Heterogenität oder Verteilung von Ressourcen, vor der Anwendung verborgen werden. Die Middleware übernimmt dabei einen Teil der Funktionalität, die sonst die Anwendung übernehmen müsste und vereinfacht damit die Anwendungsentwicklung. Der Einsatz einer Middleware kann viele Vorteile bieten, darunter höhere Flexibilität, Plattformunabhängigkeit, einfachere, schnellere und günstigere Anwendungsentwicklung sowie höhere Qualität der Software. Auch wenn der Begriff Middleware für viele unterschiedliche Abstraktionsschichten Verwendung findet, konzentrieren wir uns hier auf Kommunikations-Middleware, d.h. Middleware-Schichten, welche die Entwicklung verteilter Anwendungen vereinfachen, indem sie die Kommunikation für die Anwendung abstrahieren. Beispiele dafür sind Remote Procedure Call (RPC) Systeme wie Microsofts Distributed Component Object Model (DCOM) oder Java Remote Method Invocation (RMI). Diese Systeme erlauben es Anwendungen, Prozeduren bzw. Methoden auf einem entfernten Gerät auszuführen, ohne dass die Anwendung die dazu notwendige Kommunikation umsetzen muss. Moderne Middleware-Systeme haben oft eine dienstorientierte Architektur (engl.: Service Oriented Architecture - SOA) und basieren auf modernen offenen Standards. Ein Beispiel dafür sind die vom World Wide Web Consortium (W3C) spezifizierten Web Services, die auf HTTP und XML basieren. Im Bereich der Prozessautomatisierung wurde von der Open Platform Communications (OPC) Foundation mit OPC Unified Architecture (OPC UA) eine dienstorientierte Middleware-Schicht standardisiert. Im Gegensatz zum klassischem OPC, das in der Praxis auf DCOM aufsetzte, basiert OPC UA auf Web Services und kann die in der Prozessautomatisierung anfallenden Daten nicht nur transportieren, sondern auch semantisch beschreiben. Kommunikationstechnologien wie WirelessHART oder ZigBee enthalten eine Anwendungsschicht, die mit Einschränkungen auch als Middleware angesehen werden kann.

Dienstorientierte Architektur Bei einer dienstorientierten Architektur ist die Applikation in sogenannte Dienste (engl.: Service) unterteilt. Ein Dienst ist eine abgeschlossene Teilfunktionalität einer Applikation. Im Kontext der Prozessautomatisierung stellt beispielsweise das Auslesen eines Sensors oder das Steuern eines Aktuators eine solche abgeschlossene Teilfunktionalität dar. Dienste werden von Dienstanbietern (engl. Service Provider) angeboten und von Dienstnutzern (engl.: Service User) benutzt. In einer dienstorientierten Architektur gibt es i.d.R. eine Service Registry genannte Instanz, in der Dienste registriert und nachgeschlagen werden können.

Die Interaktion in einer dienstorientierten Architektur veranschaulicht Abb. 3.28. Zunächst machen alle Service Provider ihre Dienste bekannt, indem sie eine *Publish*-Nachricht an die Service Registry senden. Die dort registrierten Dienste können daraufhin von Service Usern per *Lookup* bei der Service Registry nachgeschlagen werden, woraufhin diese die gefundenen Dienste zurückliefert. Aus dem Ergebnis kann sich der Service User Dienste aussuchen, die er verwenden möchte. Je nach Dienst kann es unterschiedliche Möglichkeiten geben, wie der Service User mit dem Service Provider interagiert, um den Dienst zu nutzen. In Abb. 3.28 ist die Interaktion nach dem *Publish/Subscribe*-Muster dargestellt. In diesem Fall sendet der Service User eine *Subscribe*-Nachricht an den Service Provider, um den Dienst zu abonnieren. Daraufhin bekommt er den Dienst solange erbracht, bis er ihn wieder kündigt (nicht in Abb. 3.28 gezeigt). Zum Erbringen des Dienstes sendet der Service Provider dabei mehrere Nachrichten an den Service User. Im

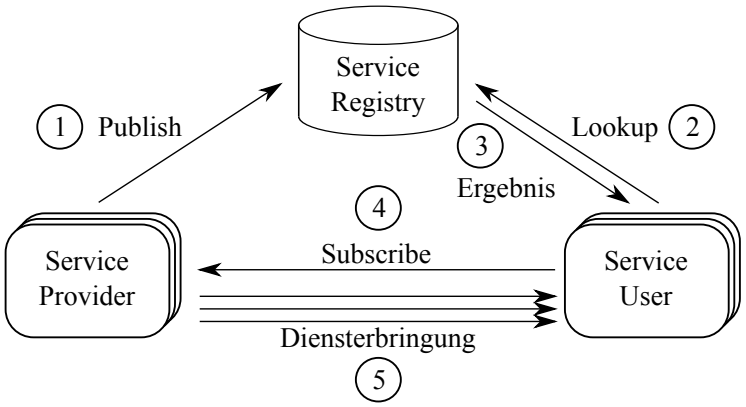


Abbildung 3.28: Interaktionen in einer dienstorientierten Architektur.

Kontext der Prozessautomatisierung könnte beispielsweise ein Regler bei einem Sensorknoten einen Dienst abonnieren, der ihm regelmäßig den vom Sensor gemessenen Wert übermittelt.

Message Exchange Patterns Die unterschiedlichen Muster, in denen Service User und Service Provider zur Nutzung eines Diensts Nachrichten austauschen, können in folgende Message Exchange Patterns (MEPs) eingeteilt werden:

- *Request/Response:* Der Service User ruft den Dienst auf (Request) und erhält eine einzelne Antwort (Response). Im Kontext der Prozessautomatisierung könnte dies beispielsweise genutzt werden, wenn ein Regler nur sporadisch einen Sensorwert abfragen möchte.
- *Fire and Forget*: Der Dienst wird vom Service User aufgerufen und vom Service Provider ausgeführt, es wird aber kein Ergebnis zurückgeliefert. Im Kontext der Prozessautomatisierung könnte dies beispielsweise genutzt werden, wenn ein Regler einen Befehl an einen Aktuator sendet, um einen Parameter zu verändern.
- *Publish/Subscribe:* Der Dienst wird vom Service Provider in der Service Registry bekannt gemacht (Publish). Anschließend können ein oder mehrere Service User diesen abonnieren (Subscribe), woraufhin der Dienst vom Service Provider erbracht wird, bis das Abonnement gekündigt wird. Der Service Provider sendet zur Erbringung des Dienstes i.d.R. mehrere Nachrichten an den Service User, ohne dass dieser einzeln anfordern muss. Die Nachrichten können dabei beim Service Provider auf zwei unterschiedliche Arten ausgelöst werden:
  - *Time-triggered:* Der Service Provider sendet Nachrichten zur Diensterbringung zu festgelegten Zeitpunkten, beispielsweise periodisch. Im Kontext der Prozessautomatisierung könnte beispielsweise ein Sensorwert von einem Regler in einem festen Intervall benötigt werden.
  - *Event-triggered:* Der Service Provider sendet Nachrichten zur Diensterbringung, wenn bestimmte Ereignisse eintreten. Im Kontext der Prozessautomatisierung könnte beispielsweise eine Lichtschranke immer dann ein Ereignis auslösen, wenn ein Produkt das Fließband an dieser Stelle passiert.

Im Folgenden werden in Kapitel 3.5.1 zunächst existierende Middleware-Lösungen diskutiert, die für den Einsatz im Kontext der Prozessautomatisierung denkbar sind. Anschließend wird in Kapitel 3.5.2 die für ProNet 4.0 entwickelte Middleware-Schicht ProMid vorgestellt. Zuletzt fasst Kapitel 3.5.3 die Inhalte des Kapitels zusammen.

3.5.1 Stand der Technik

In diesem Kapitel untersuchen wir mit WirelessHART [Eur10] und ZigBee [Zig] zunächst zwei drahtlose Kommunikationssysteme auf Middleware-Aspekte. Anschließend wird mit Web Services eine sehr bekannte Middleware-Technologie vorgestellt. Zuletzt wird das auf Web Services aufbauende OPC UA besprochen, ein verbreiteter Standard für die Kommunikation aus dem Automatisierungsumfeld.

WirelessHART Mit WirelessHART wurde 2007 der im industriellen Umfeld sehr verbreitete HART (Highway Addressable Remote Transducer) Standard um drahtlose Kommunikation erweitert. Der ursprüngliche HART-Standard war in den 1980er Jahren von der Firma Rosemount entwickelt worden und auf drahtgebundene Kommunikation beschränkt. Der WirelessHART-Standard ist auf Applikationsschicht mit dem HART-Standard kompatibel. Über einen Gateway-Knoten, der das drahtgebundene HART-Netzwerk mit dem drahtlose WirelessHART-Netzwerk verbindet, können so Applikationen auf HART-Geräten und auf WirelessHART-Geräten miteinander kommunizieren. Allerdings bedeutet dies auch, dass die Applikationsschicht keine moderne dienstorientierte Architektur umsetzt, sondern auf einfachen Kommandos basiert, die Geräte untereinander austauschen. Dabei können Geräte neben universellen Kommandos und Netzwerk-Management-Kommandos auch gerätespezifische Kommandos unterstützen. Da die Applikationsschicht das Versenden und Verarbeiten von Kommandos für die Applikation übernimmt und dabei prinzipiell Herstellerunabhängigkeit ermöglicht, kann sie im weitesten Sinn als Kommunikations-Middleware betrachtet werden. Allerdings kennt WirelessHART im Gegensatz zu einer dienstorientierten Middleware keine zentrale Registry, in der die im Netz verfügbaren Dienste nachgeschlagen werden können. Vielmehr muss per Konfiguration bekannt sein, welcher Knoten welche Kommandos unterstützt und damit welche Funktion erbringt. Da viele Kommandos auch dem Netzwerk-Management dienen, ist die Applikationsschicht nicht frei von Netzwerk-Management-Aufgaben. Die Applikationsschicht in WirelessHART schafft es daher nicht, die Verteilung im Netzwerk vor der Anwendung zu verbergen. Insgesamt erreicht die Applikationsschicht in WirelessHART kein hohes Abstraktionslevel und kann daher nur unter sehr weiter Auslegung des Begriffs als Middleware bezeichnet werden.

ZigBee ZigBee ist ein von der ZigBee Alliance standardisierter Kommunikations-Stack, der auf Automatisierungsaufgaben im Heim- und Industrie-Umfeld ausgerichtet ist. Er basiert auf IEEE 802.15.4 und fügt einen Netzwerk-Layer mit Funktionalität für Routing und Sicherheit sowie eine Anwendungsschicht hinzu. Durch den Einsatz von IEEE 802.15.4 im Physical und MAC Layer zeichnet sich ZigBee durch einen geringen Energiebedarf, aber auch durch geringe Übertragungsraten von maximal 250 kbit/s aus. Die Anwendungsschicht von ZigBee enthält das ZigBee Device Object (ZDO), was Management-Aufgaben übernimmt, und die Funktionalität der Anwendung in Anwendungsobjekte strukturiert. Ein Anwendungsobjekt kann dabei sowohl eine Funktionalität (auch über das Netzwerk) anbieten als auch aufrufen. Die Anwendungsunterstützungsschicht (APS) dient als Schnittstelle, über die Anwendungsobjekte auf das ZDO zugreifen können. Unter anderem unterstützt sie die Kopplung (engl.: Binding) zwischen Anwendungsobjekten auf verschiedenen Geräten. Ein ZigBee Gerät kann mehrere Anwendungsobjekte implementieren, die ähnlich Ports mittels Endpunkten adressiert werden können. Ein ZigBee Cluster ist eine Menge von Kommandos und Attributen für einen bestimmten Anwendungszweck. In der ZigBee Cluster Library sind die verfügbaren Cluster spezifiziert und in Gruppen wie Lichttechnik oder Sensoren eingeteilt. Darüber hinaus sind diverse Anwendungspro-

file definiert. Diese legen fest, welche Geräte in einem Anwendungsprofil vorkommen können und welche Cluster diese unterstützen müssen. Beispielsweise gibt es ein Anwendungsprofil Home Automation für die Steuerung von Licht, Heizung, Schließsystemen und ähnlichem im Heimbereich [KK14]. Weiterhin werden mittels Deskriptoren die Funktionen eines Geräts und die vorhandenen Endpunkte beschrieben. In einem Deskriptor können Endpunkte den Anwendungsprofilen und Clustern zugeordnet werden, die sie unterstützen. Die Deskriptoren werden aber nicht in einer zentralen Service Registry erfasst, sondern müssen vom Gerät abgerufen werden. Dennoch bietet die ZigBee Anwendungsschicht ein deutlich höheres Abstraktionslevel als WirelessHART und kann als Kommunikations-Middleware betrachtet werden.

Web Services Das World Wide Web Consortium (W3C) hat mit Web Services einen dienstorientierten Middleware-Standard für das Internet spezifiziert. Die Kommunikation erfolgt meist über HTTP, was wiederum TCP als Transportprotokoll nutzt. Nachrichten werden in der Extensible Markup Language (XML) oder der JavaScript Object Notation (JSON) ausgetauscht. Die Standards SOAP und XML-RPC beschreiben XML-basierte Protokolle zur Kommunikation mit Web Services. Welche Dienste von einem Anbieter angeboten werden und wie diese benutzt werden können, kann durch die Web Services Description Language (WSDL) beschrieben werden. Diese ebenfalls XML-basierte Sprache beschreibt die Schnittstelle eines Web Services, d.h. die verfügbaren Methoden und deren Parameter. Web Services können darüber hinaus in einem Universal Description, Discovery and Integration (UDDI) Verzeichnisdienst registriert und nachgeschlagen werden. Mit Web Services kann prinzipiell eine Middleware-Schicht für die Prozessautomatisierung über drahtlose Kommunikationssysteme umgesetzt werden. Allerdings erzeugen die XML-basierten Formate meist einen sehr hohen Overhead, der die zu übertragende Datenmenge leicht vervielfachen kann. Dies kann beispielsweise problematisch werden, wenn ein Sensorwert in einem sehr kurzen Intervall übertragen werden soll.

OPC UA Die Open Platform Communications (OPC) Foundation spezifizierte mit OPC UA (OPC Unified Architecture) seit 2003 einen Standard für den Datenaustausch in der Automatisierungstechnik, der das klassische OPC erweitern sollte und 2006 in erster Version veröffentlicht wurde [LIB13]. Das klassische OPC basierte auf dem von Microsoft entwickelten Distributed Component Object Model (DCOM), einem Standard für objekt-orientierte Remote Procedure Calls. DCOM brachte die Bindung an Microsoft Windows und einige weitere Nachteile mit sich. OPC UA basiert hingegen auf offenen Standards für Web Services und ist damit plattformunabhängig. Im Gegensatz zu OPC spezifiziert OPC UA nicht nur die Übertragung von Daten, sondern auch die maschinenlesbare semantische Beschreibung von Daten und Diensten. Der größte Nachteil des Einsatzes von Web Services ergibt sich aus der XML-Kodierung, die einen hohen Overhead erzeugt. Daher spezifiziert OPC UA neben der XML-Kodierung auch eine deutlich effizientere Binärkodierung. Außerdem wird neben SOAP über HTTP auch TCP direkt unterstützt, was den Overhead ebenfalls deutlich reduziert. In OPC UA werden Dienste in unterschiedliche Service Sets eingeteilt, darunter mit *Method* eine Dienstklasse für das Request/Response bzw. Fire and Forget MEP und mit *MonitoredItem* und *Subscription* für das Publish-Subscribe MEP.

Insgesamt ist OPC UA eine mächtige Middleware-Schicht mit hohem Abstraktionslevel. Sie ist allerdings in erster Linie für drahtgebundene Netzwerke konzipiert und erzeugt zu viel Overhead, um sie über drahtlose Netzwerke mit geringen Datenraten und kleinen Rahmengrößen wie IEEE 802.15.4 zu übertragen.

3.5.2 ProMid

Für ProNet 4.0 wurde mit ProMid eine Middleware-Schicht entwickelt, welche der Anwendung eine dienstorientierte Schnittstelle mit hohem Abstraktionslevel bietet. Die Middleware-Schicht wurde in Hinblick auf die spezifischen Anforderungen der Prozessautomatisierung für ProNet 4.0 konzipiert und entwickelt. ProMid setzt auf den zuvor beschriebenen Verfahren ATDP zur Topologieerkennung (s. Kapitel 3.2) und QMRP zum QoS Multicast Routing (Kapitel 3.3) auf. Außerdem nutzt es das in Kapitel 3.4 beschriebene Clustering-Verfahren HNC zur Auswahl von Service Registry-Knoten. Für eine Anwendung, die ProMid nutzt, ist die drahtlose Kommunikation transparent und sämtliche die Kommunikation betreffende Details sind verborgen. Das bedeutet, dass es für eine ProMid-Anwendung keinen Unterschied macht, ob ein Dienst lokal oder verteilt angeboten wird. Die Anwendung muss nicht wissen, auf welchem Knoten der gewünschte Dienst angeboten wird, muss keine Route berechnen oder Zeitslots dafür reservieren. Die Anwendung spezifiziert lediglich, was für einen Dienst sie abonnieren möchte. Die ProMid Middleware übernimmt daraufhin das Auffinden eines passenden Dienstes und richtet das Abonnement zwischen Service Provider und Service User ein, wobei die benötigte Route und das Scheduling automatisch eingerichtet werden. ProMid unterstützt QoS-Anforderungen der Anwendung und gibt diese automatisch an die Routensuche weiter.

In Kapitel 3.5.2.1 wird die dienstorientierte Architektur von ProMid genauer erläutert. Anschließend geht Kapitel 3.5.2.2 auf die verteilte und replizierte Service Registry von ProMid ein. Die Interaktionen der Anwendung mit ProMid werden in Kapitel 3.5.2.3 (Registrierung eines Dienstes), Kapitel 3.5.2.4 (Abonnieren eines Dienstes) und Kapitel 3.5.2.5 (Diensterbringung) beschrieben.

3.5.2.1 Dienstorientierte Architektur

Die Architektur von ProMid und ProMid-Anwendungen ist dienstorientiert. Die Funktionalität von ProMid-Anwendungen wird also in Dienste (engl.: Services) aufgeteilt und ProMid übernimmt die Verwaltung und den Zugriff auf diese Dienste. Ein Knoten, der eine Funktionalität zur Verfügung stellt, d.h. einen Dienst anbietet, wird *Service Provider* genannt. Knoten, die einen Dienst nutzen, werden *Service User* genannt. Beide Begriffe beschreiben Rollen, wobei ein Knoten prinzipiell mehrere Rollen gleichzeitig einnehmen kann. Beispielsweise kann ein Knoten einen Dienst anbieten und diesen selbst nutzen, also gleichzeitig Service Provider und Service User für ein und denselben Dienst sein. Ebenso kann ein Knoten beispielsweise mehrere Dienste nutzen und daraus einen aggregierten Wert berechnen und das Ergebnis als eigenen Dienst anbieten. Die dienstorientierte Architektur weist also einen sehr hohen Grad an Flexibilität auf. Da es für die Applikation keinen Unterschied macht, ob ein Dienst lokal oder über das drahtlose Netzwerk von einem anderen Knoten angeboten wird, führt die dienstorientierte Architektur darüber hinaus zu einem hohen Abstraktionsniveau.

Im Anwendungsszenario der Prozessautomatisierung werden Knoten i.d.R. in Sensoren, Aktuatoren und Regler eingeteilt. Sensorknoten stellen den Wert eines oder mehrerer Sensoren bereit, bieten sie als Dienst an und sind daher Service Provider. Ein Beispiel könnte ein Temperatursensor sein, der die Temperatur einer Flüssigkeit in einem Tank misst. Aktuatoren können in den Prozess eingreifen und Prozessparameter verändern. Beispielsweise könnte ein Heizelement, das die Flüssigkeit im Tank erhitzen kann, einen Aktuator darstellen. Auch Aktuatoren bieten einen Dienst an, nämlich das Verändern eines Prozessparameters, daher sind Aktuatoren auch Service Provider. Regler hingegen nutzen die Dienste von Sensorknoten, um den Prozess zu

überwachen, und die Dienste von Aktuatoren, um, falls notwendig, in den Prozess einzugreifen. Regler sind daher Service User. In der Praxis kann es Knoten geben, die mehrere Aufgaben übernehmen, beispielsweise Regler und Aktuator auf einem physikalischen Gerät kombinieren. Außerdem messen Sensorknoten häufig mehrere Werte und Aktuatoren können mehrere Prozessparameter beeinflussen. Auf Grund der hohen Flexibilität der dienstorientierten Architektur ist dies unproblematisch.

Service Provider und Service User werden durch die Middleware entkoppelt, d.h. der Service User muss nicht wissen, welcher Knoten den gewünschten Dienst erbringt. Stattdessen spezifiziert der Service User die Anforderungen an den benötigen Dienst in einer *Service Specification*. Diese übergibt er der Middleware, welche daraufhin, sofern vorhanden, einen auf die gegebene Spezifikation passenden Dienst findet und zur Nutzung einrichtet. Um passende Dienste finden zu können, müssen die im Netzwerk vorhandenen Dienste der Middleware bekannt gemacht werden. Die Service Provider müssen dazu die von ihnen angebotenen Dienste bei der Middleware registrieren. Dabei übergeben sie für jeden angebotenen Dienst eine *Service Description*, welche den angebotenen Dienst beschreibt. Die Middleware speichert die registrierten Dienste und ihre Service Descriptions in der *Service Registry*. Möchte ein Service User einen Dienst nutzen, gleicht die Middleware die Service Specification mit den in der Service Registry gespeicherten Service Descriptions ab, um einen passenden Dienst zu finden. Die Entkopplung von Service Provider und Service User ermöglicht es, Dienste zur Laufzeit von einem Knoten auf einen anderen zu verschieben oder im Falle eines Knotenausfalls dynamisch einen anderen Dienst zu finden, der ebenfalls die geforderte Service Specification erfüllt. Diese Flexibilität ist im Anwendungsszenario der Prozessautomatisierung äußerst wertvoll und ermöglicht es auch, die Ausfallsicherheit des Systems zu verbessern.

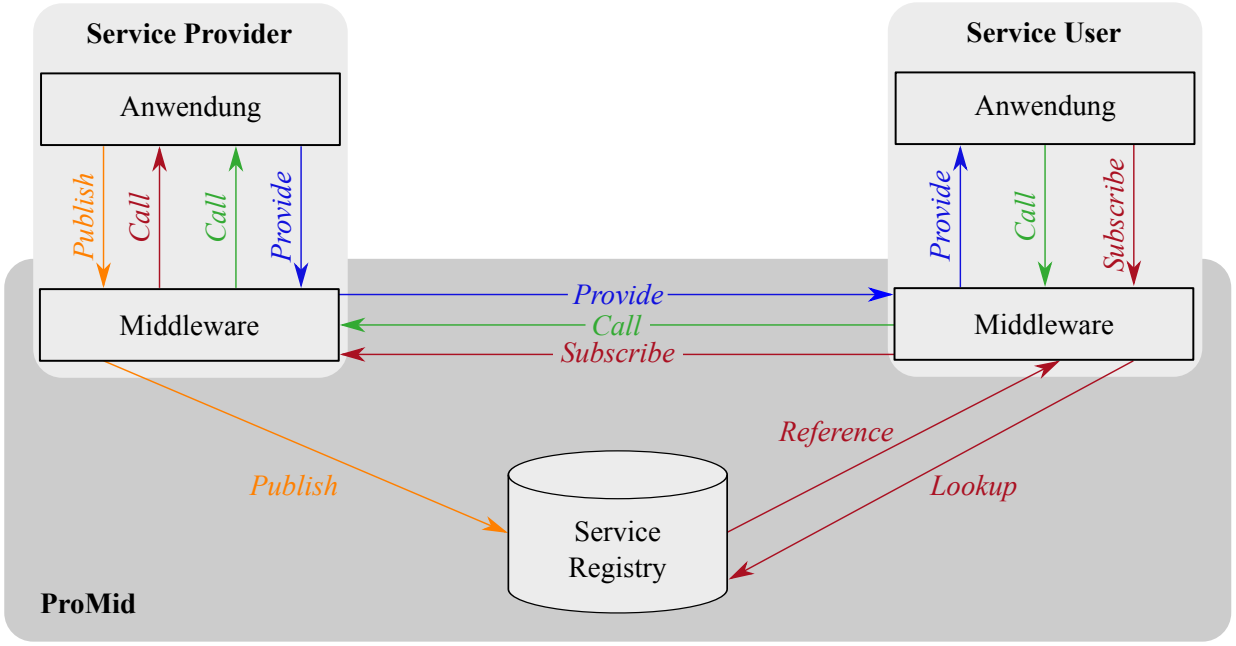


Abbildung 3.29: Interaktionen zwischen Service Provider, Service User und Service Registry in ProMid.

In Abb. 3.29 werden die Interaktionen in Promid zwischen Service Provider, Service User und Service Registry detaillierter dargestellt, als in der allgemeineren Abb. 3.28. Die Abbildung differenziert zwischen der Anwendung, die auf Service Provider und Service User ausgeführt wird, und der Middleware-Schicht von ProMid auf diesen Knoten.

Der Service Provider registriert seine Dienste bei ProMid, indem er die *Publish*-Schnittstelle seiner lokalen Middleware-Schicht aufruft. Daraufhin gibt die Middleware-Schicht das Publish an die Service Registry weiter, welche den neu registrierten Dienst speichert. Der Aufruf der Service Registry kann über das drahtlose Netzwerk erfolgen, die Service Registry kann aber auch lokal auf dem Knoten selbst verfügbar sein.

Der Service User nutzt einen Dienst, indem seine Anwendung die *Subscribe*-Schnittstelle der lokalen Middleware aufruft. Seine Middleware-Schicht führt daraufhin einen *Lookup* in der Service Registry aus, wobei diese im Erfolgsfall eine Referenz (*Reference*) auf den gefundenen Dienst zurückliefert. Diese Referenz wird nicht an die Anwendung zurückgeliefert, sondern von der Middleware-Schicht des Service Users genutzt, um den Dienst beim Service Provider zu abonnieren. Dazu sendet die Middleware des Service Users eine Subscribe-Nachricht an die Middleware des Service Providers. Die Adresse des Service Providers leitet die Middleware-Schicht aus der erhaltenen Service Referenz ab. Die Nachrichten zwischen Service User und Service Provider werden i.d.R. über das drahtlose Netzwerk übertragen, können aber auch lokal zugestellt werden, falls Service Provider und Service User auf dem gleichen Knoten ausgeführt werden. Zur Übertragung von Subscribe-Nachrichten über das Netzwerk sind Management-Slots reserviert, die eine zuverlässige Übertragung sicherstellen. Die Middleware-Schicht des Service Providers richtet daraufhin das Abonnement des Dienstes ein. Das beinhaltet, dass sie für die Diensterbringung eine Route zwischen Service User und Service Provider einrichtet und dafür Zeitslots reserviert. Außerdem verwaltet die Middleware-Schicht des Service Providers die Abonnements. Zur Diensterbringung eines zeitgetriggerten Dienstes ruft die Middleware daraufhin die Anwendung des Service Providers entsprechend des Zeitplans der Abonnements auf (Call), woraufhin diese den Dienst erbringt (Provide). Im Fall von ereignisgetriggerten Diensten erbringt die Anwendung des Service Providers von sich aus den Dienst, wann immer das entsprechende Ereignis eintritt. Provide-Nachrichten werden über die eingerichtete Route und die reservierten Zeitslots an die Middleware des Service Users übertragen, welche das Provide an die Anwendung des Service Users weiterreicht.

Dienste, die nicht das Publish / Subscribe Message Exchange Pattern nutzen, sondern Request / Response oder Fire and Forget, müssen trotzdem vom Service User über ein Subscribe (einmalig) abonniert werden. Im Anwendungsszenario der Prozessautomatisierung betrifft dies i.d.R. die von Aktuatoren angebotenen Dienste. Das Abonnement führt in diesem Fall nur dazu, dass der Dienst nachgeschlagen, die Route gesucht und Zeitslots reserviert werden. Das Aufrufen des Dienstes (*Call*) muss in diesem Fall von der Anwendung des Service Users initiiert werden. Dessen Middleware reicht dies an die Middleware-Schicht des Service Providers weiter, wobei die Kommunikation über die zuvor berechnete Route und reservierte Zeitslots erfolgt. Die Middleware des Service Providers leitet das Call an die Anwendung des Service Providers weiter, der daraufhin den Dienst erbringt. Im Falle von Request / Response antwortet der Service Provider mit einem Provide, welches genauso behandelt, wie bei Diensten nach dem Publish / Subscribe Muster. Beim Fire and Forget MEP löst das Aufrufen des Dienstes hingegen keine Provide-Nachricht aus.

3.5.2.2 Verteilte und replizierte Service Registry

Bei der Konzeption der Service Registry von ProMid stellte sich vor allem die Frage, ob diese zentralisiert oder verteilt umgesetzt werden soll, und ob die Informationen über die Dienste repliziert werden soll oder nicht. Zunächst sollen die Vor- und Nachteile dieser Optionen betrachtet werden.

Zentralisierung oder Verteilung Eine Service Registry kann zentralisiert umgesetzt werden, d.h. ein einzelner Knoten übernimmt die Rolle der Service Registry. Da alle Registry-Operationen dann auf einem einzelnen Knoten ausgeführt werden, vereinfacht dies den Betrieb der Registry. Dies bedeutet aber auch, dass zum Nachschlagen und Registrieren von Diensten i.d.R. Kommunikation notwendig ist, ggf. auch über mehrere Hops. Dies erzeugt Overhead und verlangsamt den Zugriff auf die Registry. Darüber hinaus kann die Registry zum Engpass werden. Beispielsweise beim Hochfahren des Netzwerks registrieren alle Service Provider zeitnah die von ihnen angebotenen Dienste, was viele Publish-Nachrichten an die Registry zur Folge hat. Da nur eine begrenzte Anzahl Management-Slots zur Verfügung stehen, müssen diese ggf. in Warteschlangen vorgehalten werden, bis ein freier Management-Slot die Übertragung zur Service Registry gestattet. Ein weiterer Nachteil der zentralisierten Lösung ist, dass dadurch ein Single-Point-of-Failure entsteht. Fällt der Knoten aus, der die zentralisierte Service Registry bereitstellt, so können im ganzen Netz keine Dienste mehr aufgefunden oder registriert werden. Insgesamt skaliert die zentralisierte Lösung nicht gut und ist vor allem für kleinere Netzwerke geeignet.

Bei einer verteilten Service Registry übernehmen mehrere Knoten die Rolle der Service Registry. Durch die Verteilung kann eine bessere Skalierbarkeit erreicht werden, die verteilte Registry ist weder Engpass noch Single-Point-of-Failure. Der Zugriff auf die Registry kann verkürzt werden, wenn die Service Registry-Knoten so im Netzwerk verteilt sind, dass jeder Knoten einen solchen gut erreichen kann. Zu beachten ist, dass die Rolle eines Service Registry-Knotens einem Knoten zusätzliche Ressourcen abverlangt. Daher sollte bei der Auswahl der Service Registry-Knoten darauf geachtet werden, dass diese Rolle nicht Knoten zugewiesen wird, die über geringere Ressourcen oder keine feste Energieversorgung verfügen.

In ProMid wurde eine verteilte Service Registry gewählt, um ein hohes Maß an Flexibilität zu erzielen und ein Engpass bzw. Single-Point-of-Failure zu vermeiden. Dabei legt ProMid einen großen Wert darauf, die Service Registry-Knoten vorteilhaft auszuwählen.

Replikation Wird eine Service Registry verteilt, so stellt sich die Frage, ob die Information über die gespeicherten Dienste repliziert werden soll oder nicht. Wird die Information nicht repliziert, so erfordert die Suche nach einem Dienst meist die Kommunikation zwischen allen Service Registry-Knoten. Dadurch entsteht Kommunikations-Overhead und die Abfrage wird verlangsamt. Erfolgt dagegen eine vollständige Replikation, kann jeder Service Registry-Knoten jeden Dienst lokal nachschlagen, was den Zugriff auf die Service Registry beschleunigt und Overhead bei der Suche vermeidet. Dafür erfordert die vollständige Replikation, dass Änderungen an der Registry stets mit allen Service Registry-Knoten ausgetauscht werden, sodass die Information stets synchron gehalten wird. Dadurch entsteht ebenfalls Management-Verkehr, allerdings ist dieser weniger zeitkritisch. Zu berücksichtigen ist allerdings, dass es zu kurzzeitigen Inkonsistenzen kommen kann.

Ein Kompromiss zwischen vollständiger Replikation und gar keiner Replikation stellt die partielle Replikation dar. Dies bedeutet, dass Dienste nur in einem parametrisierbaren Radius von n Hops um den Registry-Knoten repliziert werden, an dem der Service Provider ihn registriert hat. Mit $n = 0$ erfolgt keine Replikation, wählt man dagegen ein n, was mindestens so groß wie der Netzdurchmesser ist, so erfolgt eine vollständige Replikation. Mit Werten zwischen 1 und dem Netzdurchmesser wird eine lokal begrenzte partielle Replikation erreicht. Lokal verfügbare Dienste können so schnell gefunden werden. Weiter entfernt angebotene Dienste müssen dagegen durch eine Suche über alle Service Registry-Knoten gesucht werden.

In ProMid wurde die partielle verteilte Replikation umgesetzt. Da Änderungen an der Registry seltener zu erwarten sind als Lookups, entsteht durch Replikation weniger Overhead. Außerdem können Lookups so beschleunigt werden. Über den Parameter n kann der Replikations-Radius so gewählt werden, wie es im konkreten Einsatzszenario sinnvoll erscheint. Im Anwendungsszenario der Prozessautomatisierung ist es beispielsweise denkbar, dass Dienste hauptsächlich lokal genutzt werden, weil zusammengehörende Sensoren, Aktuatoren und Regler nicht weit voneinander platziert sind. In diesem Fall kann beispielsweise mit $n = 3$ sowohl der Overhead zur Synchronisation als auch zum Auffinden von Diensten i.d.R. stark begrenzt werden.

Das Protokoll, das die Replikation zwischen Service Registry-Knoten in ProMid durchführt, ist nicht sonderlich komplex und wird daher nicht detailliert beschrieben. Um die partielle Replikation zu erzielen, weist der Registry-Knoten, bei dem der Dienst registriert wird, einen Hop-Count von 0 zu. Bei der Replikation von Diensten werden diese zusammen mit ihrem Hop-Count übertragen. Erreicht der Hop-Count den Replikations-Parameter n, so wird der Dienst nicht weiter repliziert.

Auswahl der Service Registry-Knoten Die Service Registry in ProMid soll verteilt umgesetzt werden, aber nicht alle Knoten sollen die Rolle eines Service Registry-Knotens übernehmen. Da die Rolle des Service Registry-Knotens Ressourcen des Knotens verbraucht, sollen Knoten mit begrenzten Ressourcen nicht Teil der verteilten Service Registry werden. Dies betrifft insbesondere Knoten ohne feste Energieversorgung, deren begrenzte Batterielaufzeit nicht durch die zusätzliche Aufgabe verkürzt werden soll. Im Kontext der Prozessautomatisierung sind oft Sensor- und Aktuator-Knoten mit weniger Ressourcen ausgestattet als Regler, daher könnte es sinnvoll sein, nur Regler-Knoten bei der Auswahl von Service Registry-Knoten zu betrachten. Mobile Knoten sollten ebenfalls nicht Teil der Service Registry werden, da die wechselnden Pfade zum mobilen Knoten die Synchronisation zwischen den Service Registry-Knoten komplizierter und ineffizienter machen würden. Andere Knoten wiederum sollen auf jeden Fall Teil der Service Registry werden. Bei der Verwendung von QMRP (s. Kapitel 3.3) als Routing-Protokoll sollten die gewählten Access-Knoten auch als Service Registry-Knoten ausgewählt werden, um einen schnellen Zugriff der mobilen Knoten auf die Service Registry zu ermöglichen. Falls das drahtlose Netzwerk über einen Gateway-Knoten mit einem drahtgebundenen Netzwerk verbunden ist, so sollte der Gateway-Knoten als Service Registry-Knoten gewählt werden, um einen schnellen Zugriff des drahtgebundenen Netzwerks auf die Service Registry sicherzustellen. Ansonsten sollten einerseits möglichst wenig Knoten ausgewählt werden, da dies den Synchronisationsaufwand reduziert. Andererseits sollte möglichst jeder Knoten im Netzwerk einen schnellen Zugriff auf die Service Registry haben. Daraus folgt, dass die Service Registry-Knoten möglichst gleichmäßig im Netzwerk verteilt sein sollten. Wenn man fordert, dass jeder Knoten entweder selbst Service Registry-Knoten ist oder einen solchen als direkten Nachbarn hat, dann entspricht dies dem Bilden eines 1-Hop Dominating Sets. Folglich wurde das Problem der Knotenauswahl für die Service Registry als Clustering-Problem betrachtet. Speziell für die genannten Anforderungen wurde mit Heterogeneous Network Clustering (HNC) ein Clustering-Verfahren entwickelt, welches in ProMid für die Auswahl der Service Registry-Knoten verwendet wird. Das Verfahren selbst wurde bereits in Kapitel 3.4.2 detailliert beschrieben. Die von HNC gewählten Cluster Heads bekommen in ProMid die Rolle eines Service Registry-Knotens zugewiesen, die von HNC gewählten Router werden zur Synchronisation der Service Registry benutzt.

3.5.2.3 Registrierung eines Dienstes

Die Service Provider müssen die von ihnen bereitgestellten Dienste bei ProMid registrieren, damit sie von Service Usern genutzt werden können. Die dabei auftretenden Interaktionen illustriert Abb. 3.30. Die Anwendung des Service Providers generiert zunächst eine *Service Description*, welche den angebotenen Dienst beschreibt. Neben der Adresse des Service Providers enthält die Beschreibung einen hierarchisch strukturierten Service-Typ, anhand dessen Dienste von Service Usern nachgeschlagen werden können. Beispielsweise könnte ein Dienst, der einen Sensorwert eines Temperatursensors bereitstellt, dem Service-Typ *Sensor/Temperatur/Flüssigkeit* zugeordnet sein. Außerdem enthält die Beschreibung das vom Dienst unterstützte Message Exchange Pattern (beispielsweise *Publish/Subscribe*) und ob der Dienst periodisch oder ereignisgetriggert erbracht wird. Bei periodischen Diensten wird außerdem die minimale Periode angegeben, mit welcher der Dienst erbracht werden kann. Darüber hinaus kann eine Service Description Typ-spezifische Parameter enthalten. Beispielsweise könnten Temperatursensoren einen Parameter mit dem Standort angeben, an dem sie die Temperatur messen. Die Informationen in der Service Description werden beim Abonnieren von Diensten dazu genutzt, für den Service User einen Dienst zu finden, der seinen Anforderungen entspricht.

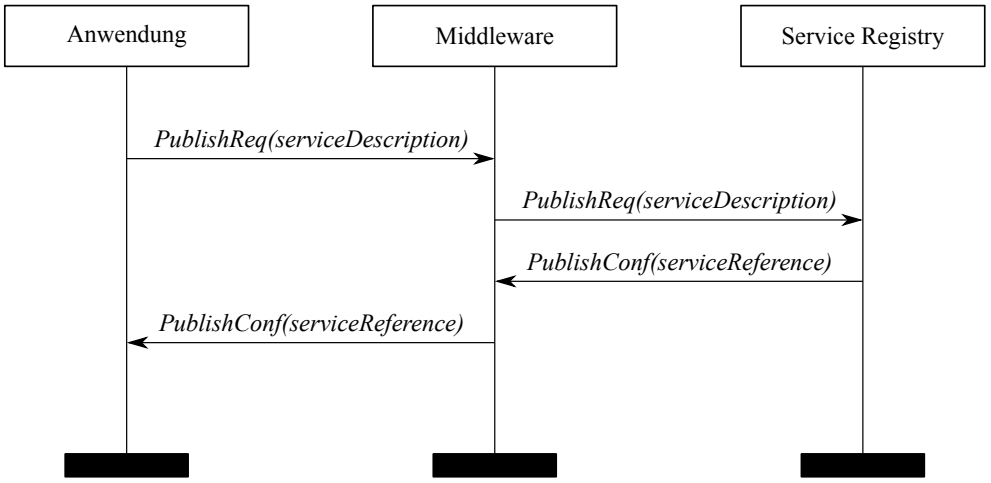


Abbildung 3.30: Interaktionen zwischen der Anwendung, der Middleware und der Service Registry beim Registrieren eines Dienstes.

Der Aufruf zwischen Anwendung und Middleware geschieht lokal auf dem Knoten des Service Providers. Die Middleware-Schicht leitet daraufhin das Publish an die Service Registry weiter. Dies erfolgt lokal, falls der Service Provider auch ein Service Registry-Knoten ist, und andernfalls über das drahtlose Netzwerk. Für die Übertragungen zwischen der Middleware und der Service Registy werden für die sporadisch auftretenden Publish-Nachrichten Management-Slots genutzt. Von der Service Registry wird eine Bestätigung mit einer Service Reference zurückgegeben. Diese wird an die Anwendung weitergeleitet und kann von dieser später genutzt werden, um den Dienst zu verändern oder zu deregistrieren. Die Service Registry repliziert die neue Dienstinformation anschließend auf die umliegenden Service Registry-Knoten.

3.5.2.4 Abonnieren eines Dienstes

Das Abonnieren eines Dienstes geht immer von der Anwendung des Service Users aus. Sie übergibt ihrer Middleware-Schicht eine Service Specification, welche ihre Anforderungen an

den Dienst beschreibt. Beispielsweise könnte ein Regler einen Dienst vom Typ *Sensor/Temperatur/Flüssigkeit* an einem definierten Standort suchen. Die sich ergebenden Interaktionen veranschaulicht Abb. 3.31. Die Middleware des Service Users schlägt den Dienst zunächst in der Service Registry nach, indem sie einen Lookup ausführt und dabei die Specification angibt. Die Service Registry gleicht die Specification mit den registrierten Diensten ab. Im Erfolgsfall gibt sie die Referenz auf einen Dienst zurück, dessen Service Description auf die geforderte Service Specification passt. Die Interaktionen zwischen Service User und Service Registry können wie beim Registrieren eines Dienstes lokal oder verteilt stattfinden. Im partiell verteilten Fall wird die Anfrage ggf. vom zuerst angefragten Service Registry-Knoten an die restlichen Service Registry-Knoten weitergereicht, falls der erste Service Registry-Knoten keinen auf die Specification passenden Dienst findet. Wurde ein passender Dienst gefunden, so startet die Middleware-Schicht des Service Users das Subscribe beim Service Provider, dessen Adresse sich aus der erhaltenen Service Reference ergibt. Die Middleware-Schicht des Service Providers richtet des Abonnement ein. Dabei richtet sie für das Abonnement auch eine Route ein und reserviert Zeitslots (in der Abbildung nicht gezeigt). Die erfolgreiche Einrichtung des Abonnements gibt sie an die Middleware-Schicht des Service Users zurück, welche dies an die Anwendung weiterreicht.

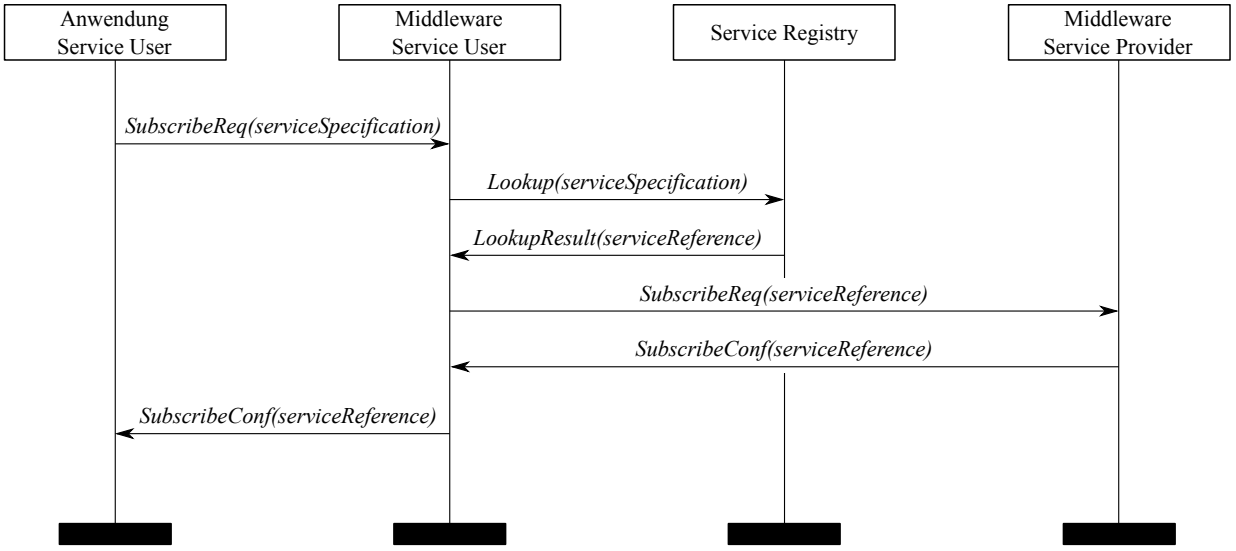


Abbildung 3.31: Interaktionen beim Abonnieren eines Dienstes.

3.5.2.5 Diensterbringung

Die Interaktionen zur Diensterbringung sind abhängig vom MEP, welches der Dienst unterstützt. In Abb. 3.32 sind die Interaktionen für den Fall gezeigt, dass das Request/Response MEP zum Einsatz kommt. In diesem Fall wird die Erbringung des Dienstes durch den Service User ausgelöst. Dazu ruft dieser die Call-Schnittstelle seiner Middleware-Schicht unter Angabe der Service Reference auf. Seine Middleware leitet das Call an die Middleware-Schicht des Service Providers weiter. Falls der selbe Knoten Service Provider und Service User ist, erfolgt der Aufruf lokal. Andernfalls erfolgt die Übertragung in exklusiv reservierten Zeitslots, welche beim Abonnieren des Dienstes dafür eingerichtet worden sind. Die Middleware-Schicht des Service Providers ruft schließlich die Anwendung des Service Providers auf. Anschließend wird das Ergebnis der Aktion mittels Provide-Nachrichten über die Middleware-Schichten von Service Provider und Service User zur Anwendung des Service Users weitergereicht. Bei Aktuatoren, die Dienste nach dem Fire and Forget Muster anbieten, entfallen die Provide-Nachrichten.

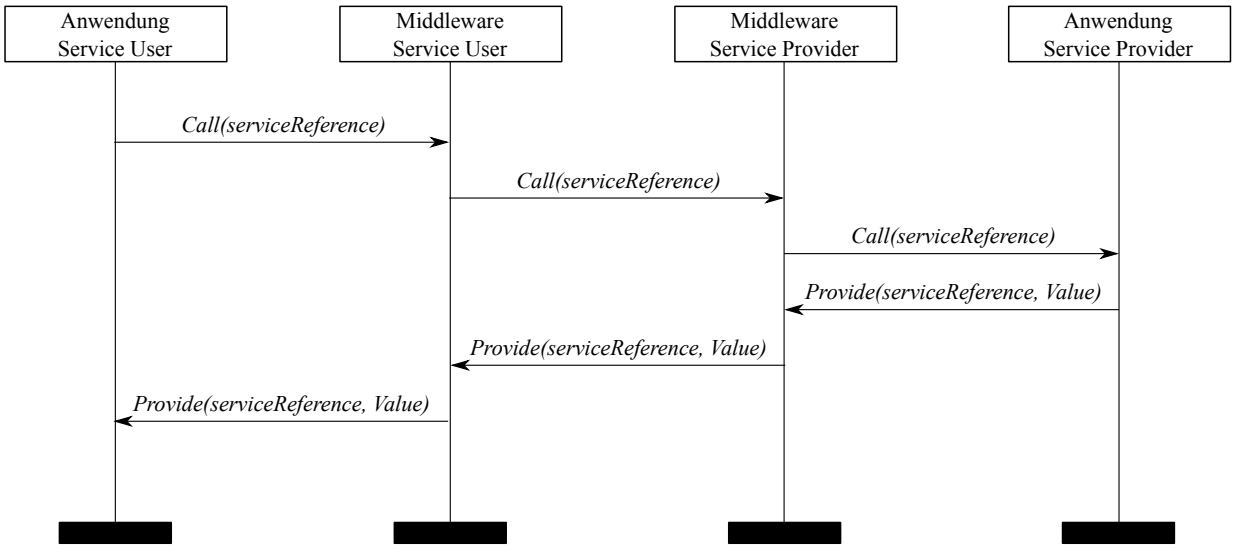


Abbildung 3.32: Interaktionen bei der Diensterbringung nach dem Request/Response MEP.

Im Fall eines periodischen Dienstes nach dem Publish/Subscribe Muster richtet die Middleware-Schicht des Service Providers beim Abonnement des Dienstes einen periodischen Timer ein. Immer wenn dieser Timer auslöst, löst die Middleware-Schicht einen Call des Dienstes aus. Anschließend wird das Provide genauso behandelt, wie in Abb. 3.32 illustriert.

Bei ereignisgetriggerten Diensten nach dem Publish/Subscribe Muster löst die Anwendung des Service Providers immer dann ein Provide aus, wenn das Ereignis auftritt. Da die Anwendung die vorhandenen Abonnements nicht kennt, weiß sie nicht, ob es für den Dienst überhaupt Nutzer gibt. Sie löst also in jedem Fall ein Provide aus, was aber von der Middleware-Schicht nur weitergeleitet wird, falls es Abonnements für den Dienst gibt.

3.5.3 Zusammenfassung

In diesem Kapitel haben wir uns mit Kommunikations-Middleware beschäftigt. Zunächst wurden die Vorteile von Middleware-Systemen bei Kommunikationssystemen aufgezeigt sowie die dienstorientierte Architektur von Kommunikations-Middleware und Message Exchange Patterns erläutert. Anschließend haben wir die drahtlosen Kommunikationssysteme WirelessHART und ZigBee auf ihre Middleware-Funktionalität untersucht sowie mit Web Services und OPC UA verbreitete Middleware-Konzepte betrachtet. Letztlich wurde mit ProMid die für ProNet 4.0 entwickelte Middlware-Schicht vorgestellt. ProMid wurde in Hinblick auf die spezifischen Anforderungen der Prozessautomatisierung entwickelt und baut auf den zuvor beschriebenen und in ProNet 4.0 implementierten Verfahren für Topologieerkenunng, Routing und Clustering auf. Die dienstorientierte Architektur von ProMid bietet ein hohes Abstraktionslevel und enthält eine verteilte und replizierte Service Registry. Auf diese Weise wird ein Single-Point-of-Failure vermieden, die Service Registry wird nicht zum Engpass und die Zugriffe auf die Service Registry werden beschleunigt. Durch die partielle Replikation wird der Overhead zur Synchronisation zwischen den Service Registry-Knoten begrenzt. Bei der Auswahl der Service Registry-Knoten können die Ressourcen der Knoten berücksichtigt werden und so beispielsweise batteriebetriebene Knoten von der Wahl als Service Registry-Knoten ausgeschlossen werden. ProMid unterstützt sowohl Dienste, die nach dem Request/Response Muster aufgerufen werden, als auch Dienste, die abonniert und daraufhin zeitgetriggert oder ereignisgetriggert erbracht wer-

den. Bei der Betrachtung der Interaktionen beim Registrieren, Abonnieren und Erbringen von Diensten wurde das hohe Abstraktionslevel deutlich.

4. KAPITEL

Protokolle und Algorithmen für zuverlässige drahtlose wettbewerbsbasierte Kommunikationssysteme

Nachdem im vorigen Kapitel Protokolle und Algorithmen für *TDMA-basierte* Kommunikationssysteme vorgestellt wurden, widmet sich dieses Kapitel Protokollen und Algorithmen für *wettbewerbsbasierte* Kommunikationssysteme. Gemeint sind damit Kommunikationssysteme wie IEEE 802.11 (WLAN), die keine exklusiven Reservierungen von Zeitslots vorsehen, sondern Knoten um das Medium konkurrieren lassen. Typischerweise kommt dabei CSMA (s. Kapitel 2.1) in Verbindung mit einem Random Backoff-Mechanismus zum Einsatz, um Kollisionen möglichst zu vermeiden. Dennoch kann es bei diesen Verfahren grundsätzlich zu Kollisionen kommen. Da diese Verfahren keine Kollisionen ausschließen und exklusive Reservierungen nicht unterstützen können, sind sie für Anwendungen mit hohen Anforderungen an die Zuverlässigkeit des Kommunikationssystems, wie die Prozessautomatisierung in Fabriken (s. Kapitel 2.2), grundsätzlich schlechter geeignet. Allerdings sind auch TDMA-basierte Kommunikationssysteme nur dann zuverlässig, wenn die Annahme erfüllt ist, dass keine anderen (externen) Geräte zeitgleich zu exklusiven Zeitslots im gleichen Frequenzbereich senden. Diese Annahme auf stark frequentierten Frequenzbereichen wie dem 2,4 GHz Band sicherstellen zu können wird zunehmend schwieriger. Beispielsweise kann heutzutage jedes moderne Smartphone einen IEEE 802.11 Hotspot auf einem vom Benutzer wählbaren 2,4 GHz Kanal bereitstellen. Möchte man im Umfeld einer Fabrik also sicherstellen, dass der 2,4 GHz Frequenzbereich, den ein TDMA-basiertes Kommunikationssystem benutzt, nicht von anderen Geräten benutzt wird, dürften keine Smartphones auf dem Fabrikgelände erlaubt sein. Eine andere Möglichkeit wäre, lizenzierte Frequenzbereiche zu nutzen, deren Nutzung allerdings mit teilweise hohen Kosten verbunden ist.

Wettbewerbsbasierte Kommunikationssysteme sind hingegen dafür ausgelegt, mit anderen Geräten um das Medium zu konkurrieren. Dies können sowohl Knoten des eigenen Netzwerkes, aber auch externe Knoten, die die gleichen Frequenzbereiche nutzen, sein. Gelingt es also, unter gewissen Voraussetzungen zuverlässige Kommunikation über prinzipiell unzuverlässige, wettbewerbsbasierte Kommunikationssysteme zu ermöglichen, würde dies eine Reihe von Vorteilen bringen. Vor allem wäre das Kommunikationssystem nicht so anfällig für Störungen durch andere Geräte, die die gleichen Frequenzbereiche nutzen. Selbstverständlich ist es nicht möglich, eine zuverlässige Kommunikation zu erreichen, wenn ein anderes (externes) Gerät ununterbro-

chen das drahtlose Medium benutzt. Eine zuverlässige Kommunikation kann also auch nur unter gewissen Voraussetzungen möglich sein. Diese Voraussetzungen könnten beispielsweise beinhalten, dass externe Knoten die Bandbreite des drahtlosen Mediums maximal zu einem gewissen Anteil nutzen, also nicht ununterbrochen senden dürfen. Da moderne wettbewerbsbasierte Kommunikationssysteme wie IEEE 802.11 sehr viel höhere Datenraten bieten, als viele TDMA-basierte Kommunikationssysteme wie die zuvor beschriebenen IEEE 802.15.4 basierten, ist es hinnehmbar, wenn ein Teil der verfügbaren Bandbreite von externen Knoten genutzt wird. Im Fall, dass andere Knoten kaum Bandbreite nutzen, bieten die hohen Datenraten von IEEE 802.11 weit mehr Anwendungsmöglichkeiten, als Kommunikationssysteme auf Basis von IEEE 802.15.4.

Neben der höheren Datenrate bietet IEEE 802.11 weitere Vorteile: Hardware für IEEE 802.11 ist extrem günstig und weit verbreitet. Dadurch, dass WLAN-Adapter in PCs weit verbreitet sind, können Protokolle und Algorithmen direkt auf dem PC, auf dem sie entwickelt werden, ausgeführt und getestet werden. Eine aufwendige Testumgebung, bei der das auf dem PC entwickelte Programm cross-kompiliert und auf das Gerät geflasht werden muss, wie sie in Kapitel 3 benutzt wurde, entfällt. Das Testen und Debuggen von IEEE 802.11 basierten Protokollen ist darüber hinaus sehr einfach, da Analyse-Tools wie Wireshark [Wir] verfügbar sind, mit denen der Verkehr auf dem drahtlosen Medium live beobachtet, analysiert oder mitgeschnitten werden kann.

In diesem Kapitel werden Algorithmen und Protokolle vorgestellt, die dem Ziel dienen, eine zuverlässige Kommunikation mit wettbewerbsbasierten Kommunikationssystemen zu ermöglichen. In Kapitel 4.1 wird zunächst das WiFiProtocolStack (WiPS) Framework vorgestellt, mit dem die hier beschriebenen Ansätze implementiert wurden. Dabei wird auch auf die Hardware eingegangen, die zum Testen der Protokolle und Algorithmen eingesetzt wurde. Anschließend beschreibt Kapitel 4.2 Verfahren zur Topologieerkennung für wettbewerbsbasierte Kommunikationssysteme. In Kapitel 4.3 wird ein in [MKG17] publiziertes Verfahren zur Verkehrskontrolle vorgestellt, das zum Ziel hat, die Zuverlässigkeit der Kommunikation zu erhöhen, indem es die von den (internen) Knoten genutzte Bandbreite über Reservierungen reguliert. Basierend darauf wird das Verfahren in Kapitel 4.4 dynamisch gemacht, sodass es sich selbstständig an die äußeren Gegebenheiten anpasst, wie beispielsweise Änderungen in der Nutzung des Mediums durch externe Knoten. Kapitel 4.5 betrachtet letztlich Clustering-Verfahren für wettbewerbsbasierte Netzwerke und stellt das in [Sef+18] publizierte Clustering-Verfahren für die dynamische Bildung und Aufrechterhaltung von Linientopologien vor.

4.1 WiFiProtocolStack

An der Arbeitsgruppe Vernetzte Systeme der TU Kaiserslautern wird ein Framework entwickelt, welches das Implementieren und Testen von Algorithmen und Protokollen für IEEE 802.11 vereinfacht. Das WiFiProtocolStack (WiPS) genannte Framework ist in C++ geschrieben und ist auf jeder Linux-fähigen Hardware ausführbar, die über einen WLAN-Adapter verfügt, der den Monitor-Modus unterstützt. Dieses Framework wurde genutzt, um die in diesem Kapitel beschriebenen Algorithmen und Protokolle zu implementieren und zu testen. In Kapitel 4.1.1 wird näher auf die flexible Architektur des Frameworks eingegangen und vorgestellt, wie die hier vorgestellten Ansätze in die Architektur passen. Kapitel 4.1.2 beschreibt anschließend die Hardware, die zum Testen der in diesem Kapitel vorgestellten Protokolle und Algorithmen zum Einsatz gekommen ist.

4.1.1 Architektur

Das WiPS Framework hat eine sehr flexible Architektur, die es ermöglicht, sehr verschiedene Protokolle zu entwickeln und, auch kombiniert miteinander, auszuführen. Die grundlegende Architektur folgt der bei Kommunikationssystemen üblichen Schichten-Architektur. Zuunterst liegt die IEEE 802.11 PHY-Schicht sowie die IEEE 802.11 MAC-Schicht (s. Abb. 4.1).

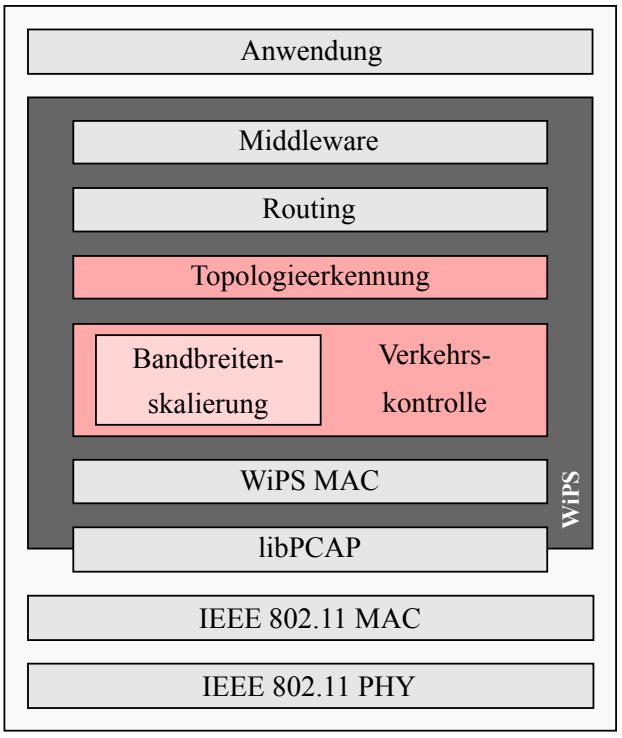


Abbildung 4.1: Beispielhafte Architektur aus verschiedenen Schichten im WiPS Framework

WiPS verlangt, dass der WLAN-Adpater bereits im Monitor-Modus ist. Dies lässt sich unter Linux mit Netzwerktools erledigen, etwa mittels `iw dev <devname> set type monitor` oder `airmon-ng start <devname>`. Außerdem lässt sich der Kanal, den der WLAN-Adapter im Monitor-Modus verwendet, über Netzwerktools einstellen, z.B. mittels `iw dev <devname> set channel <channel>`. Im Monitor-Modus lauscht der Adapter permanent auf den Kanal und gibt auf Wunsch alle Rahmen weiter, die er empfängt. Dies unterscheidet sich grundlegend von anderen Modi, wie etwa dem Managed-Modus, in dem ein WLAN-Adapter üblicherweise ist, wenn er mit einem WLAN-Accesspoint verbunden ist. Denn im Monitor-Modus werden auch Rahmen weitergegeben, die nicht an die MAC-Adresse des Adapters, sondern an andere Geräte adressiert sind; derartige Rahmen werden sonst üblicherweise ausgefiltert. Ein weiterer wichtiger Unterschied ist, dass der Adapter im Monitor-Modus die empfangenen IEEE 802.11 Rahmen direkt als solche an die oberen Schichten weitergibt, wohingegen er im Managed-Modus wie ein Ethernet-Adapter arbeitet, also an die darüber liegenden Schichten keine IEEE 802.11 Rahmen weitergibt, sondern die Tatsache, dass IEEE 802.11 zum Einsatz kommt, komplett abstrahiert und sich für darüberliegende Schichten genau wie ein Ethernet-Adapter für drahtgebundene Netzwerke verhält. Im Normalfall hat dies den Vorteil, dass sämtliche Protokolle höherer Schichten, d.h. meist UDP oder TCP/IP und darüber liegende Schichten, unverändert mit drahtlosen und drahtgebundenen Netzwerken funktionieren. Für die Entwicklung von Protokollen, die ausschließlich für IEEE 802.11 Netzwerke konzipiert sind, hat dies allerdings den Nachteil, dass Informationen über das drahtlose Medium ausschließlich dem Adapter (bzw. dessen Treiber)

und nicht den darüber liegenden Schichten zur Verfügung stehen. Beispielsweise lässt sich nur im Monitor-Modus auslesen, mit welcher Signalstärke Rahmen empfangen worden sind. Diese Meta-Informationen, die nicht Teil des IEEE 802.11-Rahmens sind, tauscht der WLAN-Adapter im Monitor-Modus über *radiotap*-Header [Rad] mit darüber liegenden Schichten aus. Da derartige Informationen für die hier entwickelten Protokolle von großem Nutzen sind, kommt im WiPS Framework der Monitor-Modus zum Einsatz.

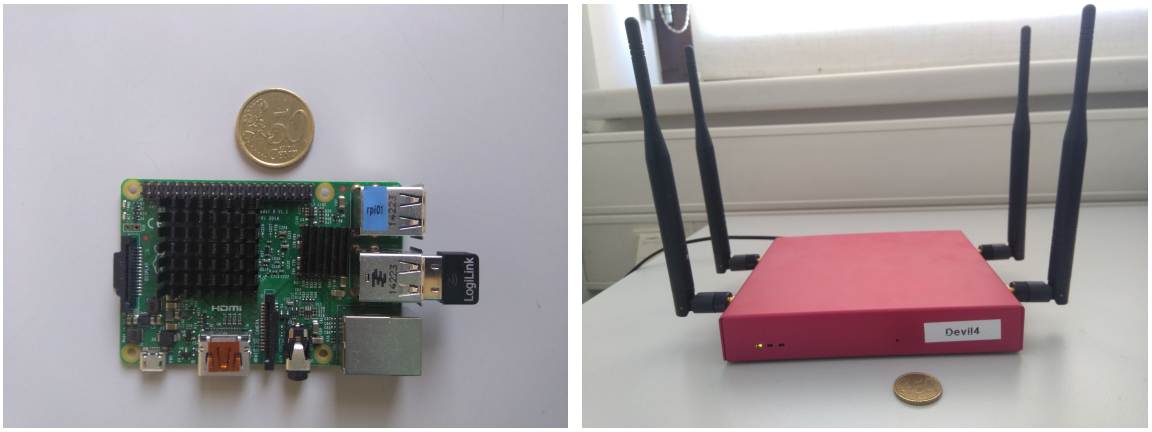
Auch beim Senden im Monitor-Modus werden mit dem WLAN-Adapter IEEE 802.11-Rahmen mit radiotap-Headern ausgetauscht. Das bedeutet, dass sämtliche IEEE 802.11 Header, beispielsweise Adress-Felder, selbst gesetzt werden können, oder auch Management-Rahmen gesendet werden können. Über die radiotap-Header können, sofern vom Adapter bzw. dessen Treiber unterstützt, außerdem Parameter wie Datenrate, Sendestärke oder genutzter Kanal für jeden Rahmen einzeln konfiguriert werden. Um möglichst hohe Kontrolle über die Aufgaben der MAC-Schicht zu erlangen, verwendet WiPS diese nicht für die Adressierung, Acknowledgements oder Neuübertragungen. Stattdessen enthält WiPS für diese Funktionalitäten die *WiPS MAC-Schicht* (s. Abb. 4.1), und übergibt der IEEE 802.11 MAC-Schicht stets IEEE 802.11 Broadcast-Rahmen, sodass diese keine Acknowledgements oder Neuübertragungen durchführt. Die WiPS MAC-Schicht übernimmt stattdessen die üblichen Funktionalitäten einer MAC-Schicht, nämlich Adressierung, Broadcasts, Acknowledgements, Neuübertragungen, Sequenznummern sowie das Ausfiltern von Duplikaten. Die Besonderheit ist, dass auf diese Funktionalitäten nun Einfluss genommen werden kann, was sonst nicht ohne weiteres möglich wäre. Die WiPS MAC-Schicht verwendet die Bibliothek *libPCAP* [JM09], um IEEE 802.11 Rahmen inklusive radiotap-Headern mit dem WLAN-Adapter auszutauschen.

Über der WiPS MAC-Schicht können im WiPS Framework weitere Protokolle zu einem Stack zusammengestellt werden. Die in Abb. 4.1 gezeigte Architektur zeigt ein Beispiel für einen solchen Stack. Für die rot markierten Schichten werden in den nachfolgenden Kapiteln Lösungen vorgestellt. Zu unterst ist eine Schicht zur Verkehrskontrolle, wie sie in Kapitel 4.3 beschrieben wird, eingezeichnet. Darüber befindet sich eine Schicht zur Topologieerkennung, wie sie in Kapitel 4.2 vorgestellt wird. In dieser Schicht ist ein Modul zur dynamischen Banbreitenregulierung eingezeichnet, wie es Kapitel 4.4 beschreibt. Darüber befinden sich Schichten für Routing und Middleware, für die hier keine Lösungen präsentiert werden. Über all diesen Protokollen steht letztlich die Anwendung. Alle Rahmen, die gesendet werden, gehen stets von der Schicht, die sie erzeugt, durch die darunter liegenden Schichten, bis zur IEEE 802.11-Schicht. Dabei kann jede Schicht ihre eigenen Header anfügen. Beim Empfang von Rahmen gelangen die Rahmen in umgekehrter Reihenfolge von der IEEE 802.11-Schicht über libPCAP in die WiPS MAC-Schicht. Handelt es sich um Rahmen, die diese Schicht selbst erzeugt hat, beispielsweise Acknowledgements, so werden diese von dieser Schicht verarbeitet und nicht nach oben weitergereicht. Andernfalls entfernt die WiPS MAC-Schicht ihre Header und gibt den Rahmen an die nächst höhere Schicht, die genauso weiter verfährt. Es gibt also Rahmen, die von einer Schicht erzeugt und beim Empfänger wieder von dieser verarbeitet werden. Außerdem kann es natürlich Rahmen geben, die von der Anwendung selbst erzeugt und am Empfänger auch wieder bis zur Anwendung weitergegeben werden.

4.1.2 Hardware

Das WiPS Framework läuft auf jeder Linux-fähigen Hardware und benötigt einen WLAN-Adapter zur drahtlosen Kommunikation. Als Hardware zum Ausführen der auf Basis von WiPS ent-

wickelten Protokolle und Algorithmen sind sowohl kleine und recht schwache Einplatinencomputer wie der Raspberry Pi 2 Model B, zu sehen in Abb. 4.2 (a), als auch besser ausgestattete Entwicklerboards wie der PC Engines APU2 in Abb. 4.2 (b), aber auch handelsübliche Desktop-Computer möglich. Somit konnten die hier vorgestellten Algorithmen und Protokolle zunächst auf einem normalen Desktop-Computer entwickelt und kleinere Tests ausgeführt werden. Anschließend wurden die Protokolle auf einem Testbett aus Raspberry Pi oder APU2-Knoten ausgeführt, um das Verhalten mit mehreren, teils räumlich verteilten Knoten zu untersuchen.



(a) Ein Raspberry Pi 2 Model B mit einem Logilink WL0084B USB-Adapter.

(b) Ein PC Engines APU2 Knoten mit vier Antennen. Im Gehäuse sind zwei Compex WLE200NX mini PCIe-Karten verbaut und jeweils mit zwei Antennen verbunden.

Abbildung 4.2: Verschiedene Hardware für Experimente mit IEEE 802.11

Raspberry Pi Der Einplatinencomputer Raspberry Pi wird von der britischen Raspberry Pi Foundation [Rasb] entwickelt. Er basiert auf einem System-on-Chip (SoC) von Broadcom, das eine CPU mit ARM-Architektur enthält. Der Einplatinencomputer zeichnet sich durch seine geringe Größe (s. Abb. 4.2 (a)) und niedrige Kosten von ca. 30 Euro aus. Das erste Modell kam 2012 auf den Markt, seitdem wurden regelmäßig leistungsfähigere Modelle vorgestellt. Zum Evaluieren der in diesem Kapitel beschriebenen Protokolle wurden teilweise Raspberry Pi 2 Model B Rev. 1.1 und teilweise Raspberry Pi 3 Model B Rev 1.2 verwendet. Erstere basieren auf einem 900 MHz getakteten Broadcom BCM2836 SoC, wohingegen letztere einen schnelleren Broadcom BCM2837A0 SoC verwenden, der bis zu 1200 MHz getaktet werden kann [Rasa]. Beide Modelle haben vier Prozessorkerne und sind mit 1 GB Arbeitsspeicher ausgestattet. Als persistenter Datenspeicher kommen microSD Speicherkarten zum Einsatz. Neben Ethernet- und HDMI-Schnittstelle verfügen sie über 4 USB-Ports, sodass USB WLAN-Adapter verwendet werden können. Der Raspberry Pi 3 hat darüber hinaus einen integrierten WLAN-Adapter, der sich allerdings nicht ohne weiteres in den für WiPS benötigten Monitor-Modus bringen lässt, sodass er hier nicht verwendet wurde. Stattdessen wurden die Raspberry Pis hauptsächlich mit Logilink WL0084B und teilweise mit TP-Link TL-WN772N USB-Adaptern, beide zu sehen in Abb. 4.3, ausgestattet. Im Logilink WL0084B ist ein Ralink Chip verbaut, der unter Linux mit dem rt2800 Treiber läuft und den von WiPS benötigten Monitor-Modus unterstützt. Er zeichnet sich durch geringe Kosten um 6 Euro sowie kleine Abmessungen aus. Der TP-Link TL-WN772N v1.10 verwendet einen Atheros AR9002U Chipsatz, der den ath9k Linux-Treiber unterstützt. Dieser Treiber hat einen sehr hohen Funktionsumfang und unterstützt auch den von WiPS benö-

tigten Monitor-Mode. Er ist mit ca. 8 Euro ebenfalls kostengünstig und sticht durch seine große Antenne hervor, mit der er eine höhere Reichweite als der kleine Logilink WL0084B erreicht. Nachteilig ist, dass der Adapter zeitweise mehr Energie benötigt, als der Raspberry Pi über seinen USB-Port bereitstellt. Ein stabiler Betrieb ist am Raspberry Pi daher nur möglich, wenn die Energieversorgung des WLAN-Adapters über einen aktiven USB-Hub erfolgt. Ein weiterer Nachteil des Raspberry Pi ist, dass an seinen USB-Bus auch interne Controller wie etwa der Ethernet-Controller angebunden sind, was dazu führt, dass die USB-Performance oft schlecht ist. Dazu kommt, dass über USB angebundene WLAN-Adapter ohnehin oft langsamer sind, als über andere Schnittstellen wie etwa PCI express angebundene. Aus diesen Gründen eignet sich der Raspberry Pi nur für WLAN-Experimente, die nicht sehr zeitkritisch sind.

Abbildung 4.3: Verschiedene WLAN-Adapter: Eine Compex WLE200NX mini PCIe (fullsize) Karte (oben), ein Logilink WL0084B USB-Adapter (unten links) und ein TP-Link TL-WN722N USB-Adapter (unten rechts)

PC Engines APU2 Das Entwicklerboard APU2 von PC Engines, abgebildet in Abb. 4.2 (b), ist für die Entwicklung von Geräten wie Routern oder Firewalls konzipiert und deutlich größer als ein Raspberry Pi. Es basiert auf einer GX-412TC CPU von AMD mit x64-Architektur, 1 GHz Takt und vier Kernen [PCE]. Angeboten wird es in Varianten mit 2 oder 4 GB RAM, wobei hier die 4 GB RAM Variante verwendet wurde. Als persistenter Datenspeicher sind neben SD-Karten auch m-SATA SSDs möglich, die verglichen mit SD-Karten eine bessere Performance und höhere Lebensdauer aufweisen. Der größte Vorteil des APU2 Boards ist aber, dass es zwei mini PCIe-Karten aufnehmen kann. Dies ermöglicht es, den APU2 mit bis zu zwei miniPCIe WLAN-Adaptern wie dem Compex WLE200NX (s. Abb. 4.3) auszustatten. Die Anbindung des WLAN-Adapters per miniPCIe bietet mehrere Vorteile. Zum einen können die Karten deutlich schneller angesprochen werden als USB-Adapter. Zum anderen ist die Treiber-Unterstützung, insbesondere des ath9k Treibers, der für Atheros Chipsätze wie den im WLE200NX verbauten AR9280 verwendet wird, deutlich besser. Dies äußert sich beispielsweise daran, dass der Treiber mehr Daten über die Kanalqualität zugänglich macht. Außerdem unterstützt die verwendete WLE200NX zwei Antennen pro Karte und damit Multiple-Input Multiple-Output (MIMO). Der Einsatz von zwei WLAN-Karten erlaubt es darüber hinaus, mehrere drahtlose Knoten auf einem Gerät auszuführen, oder aber gleichzeitig unterschiedliche Kanäle nutzen zu können. Die

WLE200NX ist darüber hinaus dual-band fähig, unterstützt also neben dem 2,4 GHz Band auch Frequenzen um 5 GHz. Dies hat den Vorteil, dass sich auf 5 GHz noch weitgehend ungenutzte Frequenzen finden, was auf einfache Weise Experimente weitgehend ohne externe Einflüsse ermöglicht. Knoten auf APU2 Basis sind verglichen mit der Raspberry Pi Plattform relativ teuer; mit SSD, Gehäuse und Netzteil kostet ein Knoten ca. 150 Euro, dazu kommen die WLAN-Karten mit ca. 20 Euro das Stück.

4.2 Topologieerkennung

In Kapitel 3.2 wurden Verfahren zur Topologieerkennung für TDMA-basierte drahtlose Netzwerke betrachtet. Die Erkennung von Interferenz- und Sensing-Topologie sind bei wettbewerbsbasierten Kommunikationssystemen nicht notwendig, da die Information nicht sinnvoll genutzt werden kann. Interferenzen werden ohnehin ausschließlich durch CSMA verhindert bzw. reduziert. Ob Interferenzen mit einem anderen Knoten wahrscheinlich sind, wird dabei bei jedem Sendevorgang durch Carrier Sense ermittelt. Wäre die Interferenztopologie bekannt, könnte man sie nicht ausnutzen, um Interferenzen zu verhindern, da nicht bekannt ist, wann andere Knoten senden. Die Sensing-Topologie ist nur hilfreich, wenn Protokolle eingesetzt werden, die über Sensing-Links kommunizieren. Dies ist bei wettbewerbsbasierten Kommunikationssystemen schwierig, da die Kommunikation über Sensing-Links externe Knoten, die CSMA anwenden, nicht zuverlässig vom Senden abhalten würde.

Bei wettbewerbsbasierten Kommunikationssystemen wie IEEE 802.11 spielt dagegen die Erkennung der Kommunikationstopologie eine immer wichtigere Rolle. Bisher wird IEEE 802.11 hauptsächlich im Infrastruktur-Modus betrieben, in dem ein Knoten als Access Point agiert und alle anderen Knoten mit diesem Access Point als Clients verbunden sind. In diesem Fall wird also immer eine Stern-Topologie mit dem Access Point als Zentrum gebildet. Da jeder Client direkt mit dem Access Point kommunizieren kann, ist kein Multihop-Routing innerhalb des WLANs notwendig. Topologieerkennung muss daher auch nicht erfolgen, um Routen innerhalb des WLANs zu finden. Die Topologieerkennung beschränkt sich in diesem Fall darauf, dass Access Points in regelmäßigen Abständen Beacons per Broadcast senden, sodass Clients sich automatisch mit ihnen bekannten Access Points verbinden können, sobald sie deren Beacons empfangen. Durch die beschränkte Reichweite von IEEE 802.11 ist die räumliche Ausdehnung eines derartigen Netzwerks allerdings stark begrenzt. Um dieses Problem zu entschärfen, werden Repeater eingesetzt, welche von einem Client empfangene Rahmen wiederholen, um sie an den Access Point zu senden, und andersherum. Damit entsteht zwar ein Multihop-Netzwerk, allerdings werden in diesem Fall die Repeater per Konfiguration mit den Access Points verbunden, sodass keine automatische Topologieerkennung notwendig ist.

Ein Mesh-Netzwerk ist dagegen ein Multihop-Netzwerk, in dem alle Knoten für andere Knoten Rahmen weiterleiten können. Dies erlaubt es theoretisch, beliebig große Netzwerke aufzubauen, ohne dass dafür viel Infrastruktur aufgebaut werden muss. Allerdings wäre das manuelle Konfigurieren von Routen in einem solchen Netzwerk sehr aufwendig und fehleranfällig. Falls Knoten sich bewegen können, müssten die Routen darüber hinaus häufig aktualisiert werden. Um dies zu vermeiden sollte ein solches Ad-Hoc-Netzwerk selbst-organisierend und selbstkonfigurierend sein. Dazu ist Topologieerkennung notwendig, welche die Netzwerk-Topologie und darauf aufbauende Routen automatisch bestimmt.

Mit IEEE 802.11s hat die IEEE den IEEE 802.11 Standard um einen Mesh-Modus ergänzt. Dieser umfasst Routing auf MAC-Ebene und die dafür notwendige Topologieerkennung. In Kapi-

tel 4.2.1 wird er in Hinblick auf die enthaltene Topologieerkennung betrachtet. Anschließend stellt Kapitel 4.2.2 ein eigenes Verfahren zur Topologieerkennung für IEEE 802.11 Netzwerke vor. Dessen Implementierung und Evaluierung beschreibt Kapitel 4.2.3 und Kapitel 4.2.4 fasst das Kapitel zusammen.

4.2.1 Stand der Technik

In Kapitel 3.2.1 wurden bereits Routing- und Clustering-Protokolle besprochen, die Funktionalitäten zur Topologieerkennung enthalten. Die dort besprochenen Protokolle sind größtenteils sowohl für TDMA- als auch wettbewerbsbasierte Kommunikationssysteme geeignet und werden daher hier nicht erneut betrachtet. Stattdessen konzentrieren wir uns hier auf IEEE 802.11s [IEE11b], ein Amendment, das IEEE 802.11 um einen Mesh-Modus ergänzt. Um Daten in einem Mesh-Netzwerk zu übertragen, wird ein Routing-Protokoll benötigt, was wiederum Topologieerkennung voraussetzt. IEEE 802.11s beschreibt für das Routing das *Hybrid Wireless Mesh Protocol (HWMP)* als Default-Protokoll, welches durch jedes kompatible Gerät unterstützt werden muss [Hie+10]. Darüber hinaus sieht der Standard vor, dass Hersteller von WLAN-Hardware alternative Routing-Protokolle implementieren. HWMP kombiniert reaktives Routing mit einem proaktiven hierarchischen Ansatz.

Der reaktive Anteil von HWMP ist durch *Ad-hoc On-demand Distance Vector (AODV)* [PR99] inspiriert und diesem sehr ähnlich. Als reaktiver Ansatz sammelt das Verfahren keine Topologieinformation, bevor Routen benötigt werden. Erst sobald ein Knoten s eine Route zu einem Knoten d sucht, beginnt die Routensuche und damit die Topologieerkennung. Dazu sendet s eine PREQ (Path Request) Nachricht mit der Adresse von d und einer Sequenznummer als Broadcast [BAM12]. Per Flooding gelangt die Nachricht zu d, der sie mit einer PREP (Path Reply) Nachricht beantwortet, die per Unicast zu s gesendet wird. Zwischengelagerte Knoten nutzen die Nachrichten, um ihre eigenen Routen zu s bzw. d zu aktualisieren, und bewerten dabei die Aktualität der Information anhand der Sequenznummer. Nur wenn das Target Only-Flag in einer PREQ-Nachricht auf 0 gesetzt ist, darf die Suche auch durch einen Zwischenknoten, der bereits eine Route zu d kennt, mit einer PREP-Nachricht beantwortet werden.

Als Metrik zur Auswahl der besten Route aus mehreren möglichen Routen dient in HWMP nicht die Anzahl der Hops (Shortest-Path), sondern die sog. *Airtime* [Hie+10]. Diese beschreibt die Zeit, die ein Rahmen typischerweise benötigt, um auf dem gegebenen Pfad übertragen zu werden. Dabei wird der Overhead beim Mediumzugriff O_{medium}, der Protokolloverhead $O_{protocol}$, die Übertragungszeit eines Testrahmens mit $size = 8192$ Bits Größe [IEE11b] bei der verwendeten Datenrate $rate$ und die Fehlerrate $errorRate$ des Testrahmens berücksichtigt. Aus diesen Parametern berechnet sich die Airtime eines Links wie folgt [Hie+07]:

$$airTime = \left(O_{medium} + O_{protocol} + \frac{size}{rate} \right) \frac{1}{1 - errorRate} \tag{4.1}$$

Die Airtime eines Pfades ergibt sich als Summe über alle darin enthaltenen Links. Sie wird bestimmt, indem jeder Zwischenknoten die in einer Nachricht enthaltene Airtime um die Airtime des Links erhöht, über den er die Nachricht empfangen hat. Erhält der Zielknoten d eine weitere PREQ-Nachricht mit einer besseren Metrik, so aktualisiert er seinen Pfad zu s und sendet über den neuen Pfad eine neue PREP-Nachricht an s, um ihn über die bessere Route zu informieren. Bei s kommen entsprechend nur PREP-Nachrichten an, welche s über einen bidirektionalen Pfad

mit d verbinden. Über die in der Airtime-Metrik berücksichtigte Fehlerrate wird die Linkqualität berücksichtigt. Nimmt man die Fehlerrate und Datenrate für alle Links gleich an, so entspricht die Airtime-Metrik der Shortest-Path-Metrik. Die Schwäche der Shortest-Path-Metrik, lange und unzuverlässige Links zu bevorzugen, übernimmt die Airtime-Metrik damit, sofern Datenrate und Fehlerrate nicht in Abhängigkeit von der Linkqualität bestimmt werden. Allerdings lässt der Standard die Frage offen, wie genau die Fehlerrate *errorRate* bestimmt werden soll. Er beschreibt nur, dass damit die Wahrscheinlichkeit angegeben wird, dass ein Rahmen der Größe *size* bei einer Übertragung mit einer Datenrate *rate* auf diesem Link verloren geht oder verfälscht wird [IEE11b]. Damit ist ein wichtiger Baustein der Topologieerkennung von IEEE 802.11s nicht vollständig spezifiziert.
Der proaktive Anteil von HWMP ist hierarchisch, baut also eine Baumstruktur auf. Dazu muss per Konfiguration ein Knoten als Wurzel (engl: root) r festgelegt werden. Die Wurzel sendet periodisch eine PREQ-Nachricht mit einer speziellen Zieladresse, die im Netzwerk geflutet wird. Alle Knoten, die diese erhalten, aktualisieren ihren Pfad zur Wurzel anhand der erhaltenen PREQ-Nachricht und leiten sie weiter. Als Metrik dient wie im reaktiven Verfahren die Airtime des Pfades. Falls der Wurzel-Knoten das Proactive PREP-Flag in seinen PREQ-Nachrichten setzt, müssen die Knoten die Wurzel mit einer PREP-Nachricht über den gefundenen Pfad informieren. So entstehen bidirektionale Pfade zwischen der Wurzel und allen anderen Knoten. Anschließend kann eine Quelle s mit jedem Ziel d kommunizieren, indem es die Nachricht zunächst über den proaktiv ermittelten Pfad zur Wurzel r routet und von dort weiter über den ebenfalls proaktiv bestimmten Pfad von der Wurzel zum Ziel d. Ist das Proactive PREP-Flag nicht gesetzt, so steht es den Knoten frei, ob sie mit einer PREQ-Nachricht antworten und der Wurzel somit mitteilen, wie sie erreicht werden können.
Neben dem proaktiven PREQ-Mechanismus unterstützt HWMP auch den proaktiven RANN-Mechanismus. Hierbei sendet der Wurzel-Knoten r periodisch RANN (Root Announcement) Nachrichten. Diese werden im Netzwerk geflutet, sodass jeder Knoten weiß, wie er die Wurzel findet. Möchte ein Knoten mit der Wurzel kommunizieren, muss er aber zunächst eine PREQ-Nachricht per Unicast an die Wurzel schicken, welche mit einer PREQ-Nachricht antwortet und so den bidirektionalen Pfad zur Wurzel aufbaut. Weiterhin können die proaktiven und reaktiven Verfahren in HWMP auch nebeneinander verwendet werden. Beispielsweise kann ein Knoten s zunächst die proaktiv bestimmte Route über die Wurzel r zum Knoten d nutzen und anschließend über das reaktive Verfahren eine direkte Route von s zu d suchen, um weitere Übertragungen zu beschleunigen.
HWMP hat letztlich ähnliche Schwächen wie AODV, da Routensuche auf einzelnen Request- und Response-Nachrichten beruht. Zwar berücksichtigt HWMP die Linkqualität in der Airtime-Metrik über die Fehlerrate, aber spezifiziert nicht, wie die Fehlerrate ermittelt wird. Es ist somit denkbar, dass unzuverlässige Links bei der Routensuche Verwendung finden. Unidirektionale Links werden darüber hinaus nicht ausreichend berücksichtigt. So geht das Verfahren immer davon aus, dass über den Pfad, über den die PREQ-Nachricht gelaufen ist, auch die PREQ-Nachricht zurück gesendet werden kann. Ist einer der Links unidirektional, so ist dies nicht der Fall.

4.2.2 Automatic Topology Discovery Protocol for WiFi (ATDP4W)

Wir haben gesehen, dass Topologieerkennung in wettbewerbsbasierten Kommunikationssystemen wie IEEE 802.11 an Bedeutung gewinnt. In diesem Kapitel wird ein Verfahren zur Topologieerkennung mit IEEE 802.11 vorgestellt, welches, genau wie das in Kapitel 3.2.2 vorgestellte

TDMA-basierte Verfahren ATDP, in Hinblick auf Anwendungen wie die Prozessautomatisierung entwickelt wurde. In diesem Anwendungsszenario sind die Knoten größtenteils stationär, d.h. die meisten Knoten bewegen sich nicht oder nur sehr selten, beispielsweise wenn die Produktion auf ein neues Modell umgestellt wird. Nichtsdestotrotz ist es in einem solchen Szenario umständlich und fehleranfällig, die Topologie manuell zu konfigurieren, sodass eine automatische Topologieerkennung vorteilhaft ist. Das Verfahren soll, genau wie ATDP, die Topologie beim Hochfahren des Netzwerks erkennen und verteilen. Da wir davon ausgehen, dass sie sich nicht entscheidend ändert, kann die so erkannte und verteilte Topologie über einen längeren Zeitraum verwendet werden. Routing ist somit ohne Kommunikations-Overhead möglich, indem die Knoten auf der bekannten Topologie den jeweils nächsten Hop lokal berechnen, beispielsweise anhand des Dijkstra-Algorithmus [Dij59]. Außerdem kann die erkannte Topologie beispielsweise zum Clustering genutzt werden. Wichtig ist, dass verteilte Topologie über alle Knoten konsistent ist. Andernfalls wären beispielsweise Routing-Schleifen möglich. Weiterhin soll das Protokoll die Linkqualität bewerten, sodass unzuverlässige Links vermieden werden können. Asymmetrische Links sollen erkannt werden, sodass diese ausgeschlossen oder nur in eine Richtung verwendet werden. Die Interferenz- und Sensing-Topologie ist wie eingangs erwähnt für wettbewerbsbasierte Kommunikationssysteme kaum nützlich und soll daher nicht erkannt werden.

Das in diesem Kapitel vorgestellte Protokoll *Automatic Topology Discovery Protocol for WiFi (ATDP4W)* basiert auf ATDP [KCG15] und einem von Mathews in [Mat16] beschriebenen und für das WiPS Framework entwickelten Protokoll. Zunächst beschreiben wir in Kapitel 4.2.2.1 die grundlegende Funktionsweise des Protokolls und gehen auf Unterschiede zu ATDP ein. Anschließend beschreibt Kapitel 4.2.2.2 den Mechanismus zur Linkbewertung und Kapitel 4.2.2.3 die Bestimmung des Link-Zustands. Auf die Verteilung der Link-Informationen geht Kapitel 4.2.2.4 ein und Kapitel 4.2.2.5 beschreibt, wie das Verfahren terminiert.

4.2.2.1 Überblick

Zum Ausmessen der Linkqualität verwendet ATDP4W sogenannte *MEASURE*-Nachrichten und bewertet die Linkqualität anhand der beobachteten Verlustrate. Genau wie bei ATDP werden diese Nachrichten nicht nur zum Ausmessen der Linkqualität, sondern auch zum Verteilen der bisher erkannten Topologie genutzt. Auf diese Weise kann die erkannte Topologie effizient ausgetauscht werden. Anders als im TDMA-basierten ATDP werden *MEASURE*-Nachrichten nicht in durch exklusive Reservierungen vorgegebenen konstanten Intervallen, sondern in zufällig variierenden Abständen gesendet. Dies soll verhindern, dass die *MEASURE*-Nachrichten eines Knotens wiederholt mit denen eines Nachbarknotens um das Medium konkurrieren und ggf. wiederholt kollidieren. Nachrichtenverluste werden anhand von Sequenznummern erkannt oder sobald die Überschreitung des maximal möglichen Intervalls zwischen zwei *MEASURE*-Nachrichten beobachtet wird. Auf letztere Weise können auch gebrochene Links erkannt werden. Da das wettbewerbsbasierte DCF-Zugriffsverfahren in IEEE 802.11 Kollisionen nicht vollständig ausschließen kann, muss die Topologieerkennung mit vereinzeltem Paketverlust rechnen und kann Links nicht bereits auf Grund von wenigen Paketverlusten als unzuverlässig einstufen. In diesem Punkt unterscheidet sich ATDP4W stark von ATDP.

Die Terminierung des Protokolls unterscheidet sich ebenfalls stark von ATDP. Dort wurde mit Hilfe des AVTP-Protokolls [CGR12] eine gemeinschaftliche Entscheidung aller Knoten im Netzwerk über die Terminierung getroffen, die sicherstellt, dass alle Knoten eine konsistente Sicht auf das Netzwerk haben. Da das AVTP-Protokoll sich nicht ohne weiteres auf Basis von IEEE 802.11

umsetzen lässt, muss eine andere Lösung für dieses Problem gefunden werden. Die Terminierungsentscheidung trifft in ATDP4W daher ein zentraler Masterknoten, der zuvor per Konfiguration bestimmt wurde. Mit ihren *MEASURE*-Nachrichten tauschen die Knoten in ATDP4W Hash-Werte aus, welche über die bisher bekannte Topologie gebildet werden. Der Masterknoten sammelt diese Hash-Werte und vergleicht sie. Sobald die Topologie bei allen Knoten konsistent vorliegt, stimmen auch die Hash-Werte aller Knoten überein. Der Masterknoten sendet daraufhin eine *TERMINATE*-Nachricht, die den Hash-Wert enthält, den der Masterknoten übereinstimmend erhalten hat. Diese Nachricht wird per Flooding im Netz verteilt. Falls ein Knoten seine Topologie zwischenzeitlich aktualisiert hat, muss er auf die Version der Topologie zurück rollen, die der Hash-Wert aus der *Terminate*-Nachricht festlegt. So kann sichergestellt werden, dass alle Knoten mit einer konsistenten Netzwerk-Topologie terminieren.

4.2.2.2 Ausmessen der Links

Zum Ausmessen der Links sendet ATDP4W *MEASURE*-Nachrichten in zufälligen Intervallen per lokalem Broadcast. Das zufällige Intervall wird aus dem Bereich $delay_{min}$ und $delay_{max}$ gewählt. Da *MEASURE*-Nachrichten über IEEE 802.11 DCF als Broadcast-Rahmen übertragen werden, unterliegen sie dem normalen Wettbewerb um das Medium. Würden die Nachrichten in einem konstanten Intervall gesendet, so wäre es denkbar, dass zwei benachbarte Knoten wiederholt um das Medium konkurrieren und die Nachrichten ggf. durch Kollision wiederholt verloren gehen oder verfälscht werden. Dies würde dazu führen, dass das Protokoll einen Link, über den eigentlich zuverlässig kommuniziert werden kann, als unzuverlässig einstufen oder gar nicht erkennen würde. Durch die zufälligen Intervalle zwischen den Nachrichten ist es recht unwahrscheinlich, dass zwei benachbarte Knoten wiederholt um das Medium konkurrieren. Sollte das Netz sehr dicht und Kollisionen damit wahrscheinlicher sein, so ist es ratsam, das Intervall, aus dem die zufällige Verzögerung gezogen wird, groß zu konfigurieren und so die Kollisionswahrscheinlichkeit zu reduzieren.

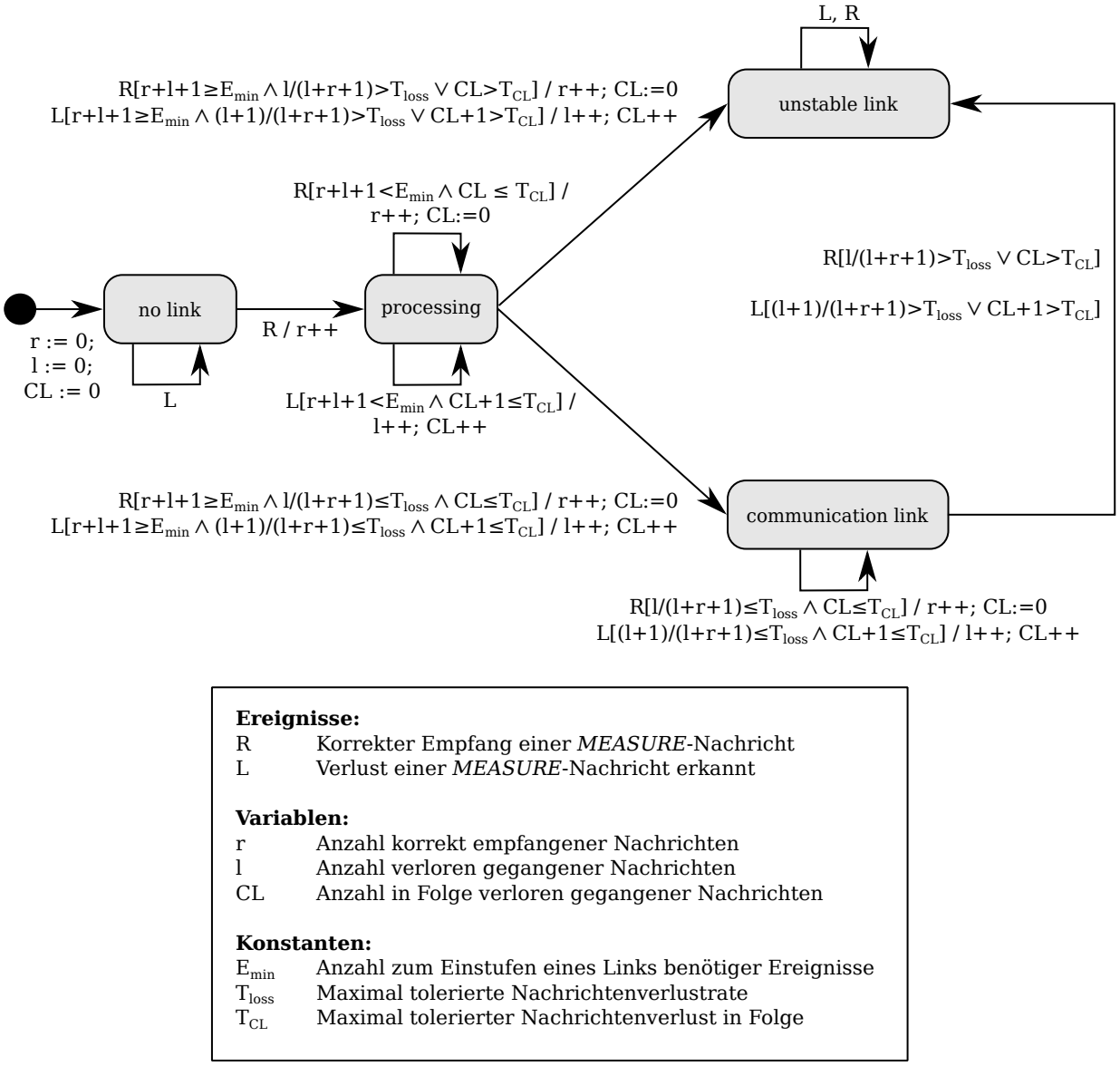
Empfängt ein Knoten die *MEASURE*-Nachricht eines Nachbarknoten, so beginnt er, den Link zu bewerten. Dazu bestimmt er die Verlustrate und wie viele Nachrichten in Folge verloren gegangen sind. Verluste werden anhand der in den *MEASURE*-Nachrichten enthaltenen Sequenznummern erkannt. Empfängt ein Knoten also eine Nachricht mit der Sequenznummer seq_1, nachdem er zuvor eine Nachricht mit Sequenznummer seq_0 erhalten hatte, so lässt sich der Nachrichtenverlust einfach berechnen:

$$messageLoss = seq_1 - seq_0 - 1 \tag{4.2}$$

Falls ein Link bricht, werden keine weiteren *MEASURE*-Nachrichten empfangen. Allerdings ist bekannt, dass zwei Nachrichten immer maximal in einem Abstand von $delay_{max}$ gesendet werden. Ist die letzte empfangene Nachricht älter als $delay_{max}$, kam es also zu mindestens einem Nachrichtenverlust. Falls der Link gebrochen ist, ist es unerheblich, wie viele Nachrichten genau verloren gegangen sind, da der Link auf jeden Fall als unzuverlässig eingestuft werden kann. Ist der Link nur kurzzeitig unterbrochen, so kann die exakte Anzahl verlorener Nachrichten bestimmt werden, sobald die erste *MEASURE*-Nachricht nach der Unterbrechung empfangen wird.

4.2.2.3 Bestimmung des Link-Zustands

Ähnlich wie bei ATDP durchläuft jeder einzelne Link in ATDP4W in Laufe der Einstufung unterschiedliche Zustände. In Abb. 4.4 sind die Zustände und ihre Übergänge in einem Zustandsgra-



Ereignisse:

R	Korrekter Empfang einer *MEASURE*-Nachricht
L	Verlust einer *MEASURE*-Nachricht erkannt

Variablen:

r	Anzahl korrekt empfangener Nachrichten
l	Anzahl verloren gegangener Nachrichten
CL	Anzahl in Folge verloren gegangener Nachrichten

Konstanten:

E_{min}	Anzahl zum Einstufen eines Links benötiger Ereignisse
T_{loss}	Maximal tolerierte Nachrichtenverlustrate
T_{CL}	Maximal tolerierter Nachrichtenverlust in Folge

Abbildung 4.4: Zustandsgraph zur Bestimmung des Linktyps in ATDP4W

phen veranschaulicht. Jeder Link beginnt im Zustand *no link* und verharrt dort, solange keine *MEASURE*-Nachricht empfangen wird. Mit dem Empfang der ersten Nachricht beginnt dann die Einstufung des Links im Zustand *processing*. In diesem Zustand werden korrekt empfangene Nachrichten und beobachteter Nachrichtenverlust gezählt, um den Link einzustufen. Jede korrekt empfangene Nachricht erhöht den Zähler r und jeder Nachrichtenverlust erhöht den Zähler l. Außerdem wird der Nachrichtenverlust in Folge über den Zähler CL erfasst. Immer, wenn eine Nachricht verloren geht, wird der Zähler CL erhöht. Wird eine Nachricht korrekt empfangen, so wird er zurückgesetzt. Überschreitet CL den Grenzwert für maximal tolerierten Nachrichtenverlust in Folge T_{CL}, so wird der Link als unzuverlässig eingestuft und geht in den Zustand *unstable link* über und verbleibt dort. Bleibt der Nachrichtenverlust in Folge dagegen im tolerierten Bereich, so verbleibt der Link für E_{min} Link-Ereignisse (Nachrichtenverlust oder Empfang) im Zustand *processing*. Anschließend erfolgt die Einstufung anhand der Verlustrate $\frac{l}{l+r}$. Diese darf T_{loss} nicht übersteigen, damit der Link als zuverlässiger Kommunikationslink eingestuft wird und in den Zustand *communication link* wechselt, andernfalls wechselt der Link

in den Zustand *unstable link*. Im Zustand *communication link* wird der Link weiter überwacht und wechselt ggf. noch in den Zustand *unstable link*, falls der Nachrichtenverlust oder der Nachrichtenverlust in Folge den jeweiligen Grenzwert übersteigt. Im Zustand *unstable link* verbleibt der Link, unabhängig davon ob Nachrichten korrekt empfangen werden oder es zu Nachrichtenverlust kommt.

4.2.2.4 Verteilung der Link-Information

Die erkannte Topologieinformation wird über *MEASURE*-Nachrichten mit den Nachbarn ausgetauscht. Diese aktualisieren anhand der erhaltenen Information ihre eigene Topologieinformation und verteilen diese in ihren eigenen *MEASURE*-Nachrichten weiter. Auf diese Weise wird die Topologie des gesamten Netzwerks netzweit verteilt. Da zum Ausmessen der Links ohnehin Nachrichten gesendet werden müssen, ist es effizient, die Topologieinformation über diese Nachrichten zu verteilen.

Für jeden Link $k_1 \rightarrow k_2$ werden in einer MEASURE-Nachricht folgende Felder übertragen[2]:

- *Knoten-Identifier von Sender und Empfänger:* Ein Link $k_1 \rightarrow k_2$ wird eindeutig über die Knoten-Identifier des Senders k_1 und des Empfängers k_2 identifiziert. Es wird davon ausgegangen, dass drahtlose Links asymmetrisch sein können. Daher wird der Link $k_2 \rightarrow k_1$ unabhängig vom Link $k_1 \rightarrow k_2$ betrachtet.
- *Erkannter Linktyp:* Hierbei werden folgende Typen unterschieden:
  - *Kommunikationslink*: Ein Link, über den k_1 an k_2 senden kann. Die Verlustrate auf diesem Link beträgt maximal T_{loss} und es kommt maximal T_{CL}-mal in Folge zu Nachrichtenverlust.
  - *Unstable*: Ein Link, über den nur unzuverlässig kommuniziert werden kann.
  - *Kein Link*: Wenn k_1 sendet, empfängt k_2 die Nachricht nicht.
- *Sequenznummer:* Der Knoten k_2, der den Link $k_1 \rightarrow k_2$ bewertet, vergibt die Sequenznummer für diesen Link und erhöht sie jedes Mal, wenn sich die Einstufung ändert. Anhand der Sequenznummer kann ein Knoten die Aktualität der Information bewerten und seine gespeicherten Informationen entweder aktualisieren oder veraltete Informationen ignorieren.
- *Signalstärke:* Die durchschnittliche Signalstärke der auf diesem Link empfangenen Nachrichten, falls es sich um einen Kommunikationslink handelt. Andere Protokolle, wie z.B. Routing-Protokolle, können anhand dieser Information stärkere Links bevorzugt benutzen.
- *Topologie-Hash des Senders:* Hash-Wert über die dem Sender-Knoten k_1 bekannte Topologie. Diese Information wird zur Terminierungsentscheidung (s. Kapitel 4.2.2.5) benötigt, um sicherzustellen, dass alle Knoten eine konsistente Sicht auf das Netzwerk haben.

Insgesamt werden für jeden Link 160 Bit übertragen. Die maximal mögliche Payload von IEEE 802.11 Rahmen ist mit ca. 2000 Bytes (je nach verwendeter Verschlüsselung) [IEE16] groß genug, um ca. 100 Links zu übertragen. In kleineren Netzen mit weniger als 100 Links kann also die komplette Topologie mit einer *MEASURE*-Nachricht übertragen werden. Falls mehr Links im Netzwerk existieren, so muss fragmentiert werden. Dabei werden neue Informationen priorisiert übertragen, sodass sie möglichst schnell verteilt werden.

[2] Dies ist sehr ähnlich zu ATDP, siehe Kapitel 3.2.2.4.

4.2.2.5 Terminierung

Die Terminierungsentscheidung erfolgt in ATDP4W durch einen per Konfiguration bestimmten Masterknoten. Dieser soll die Topologieerkennung terminieren, sobald alle Knoten die Erkennung der Topologie abgeschlossen haben, d.h. keine Links mehr im Zustand *processing* sind, und alle Link-Informationen ausgetauscht sind, sodass alle Knoten die selbe Topologieinformation haben.

Sobald ein Knoten bereit ist zu terminieren, d.h. keiner seiner Links ist noch im Zustand *processing*, beginnt er, über die ihm bekannte Topologie einen Hash-Wert zu berechnen. Diesen Hash-Wert schreibt er bei den von ihm erkannten Links in das Feld *Topologie-Hash des Senders*, wenn er eine *MEASURE*-Nachricht sendet. Zusammen mit der Link-Information wird der Hash im Netz verteilt und erreicht somit auch den Masterknoten. Dieser sammelt die Hash-Werte aller Knoten ein und vergleicht sie. Sobald alle Knoten den gleichen Hash-Wert übermitteln, d.h. die gleiche Topologieinformation haben, beginnt der Masterknoten den Terminierungsprozess. Dazu sendet er eine *Terminate*-Nachricht an alle seine direkten Nachbarn, zu denen er laut der erkannten Topologie einen bidirektionalen Kommunikationslink hat.

Die *Terminate*-Nachricht enthält den Hash-Wert, den der Master bei allen Knoten gesehen hat und mit dem folglich terminiert wird. Enthält ein Knoten eine *Terminate*-Nachricht, so vergleicht er den darin enthaltenen Hash mit dem aktuellen Hash der ihm bekannten Topologie. Stimmen diese nicht überein, so muss der Knoten seine Topologie auf jene Topologie zurückrollen, die dem Hash-Wert aus der *Terminate*-Nachricht entspricht. Dazu ist es erforderlich, dass der Knoten Änderungen an der ihm bekannten Topologie so protokolliert, dass er sie solange rückgängig machen kann, bis die resultierende Topologie den gesuchten Hash hat. Das Protokollieren ist allerdings erst nötig, sobald der Knoten zum ersten mal den Hash seiner Topologie bekannt gemacht hat. Zu diesem Zeitpunkt sollten ohnehin nicht mehr viele Änderungen auftreten, sodass das Protokoll i.d.R. nicht groß wird.

Terminate-Nachrichten werden per Unicast übertragen und per Acknowledgement (ACK) bestätigt. Wird kein ACK empfangen, wird die Nachricht neu übertragen. Die Nachbarknoten senden jeweils wieder *Terminate*-Nachrichten an ihre Nachbarn per Unicast und terminieren, sobald sie die ACKs von diesen erhalten haben. Auf diese Weise wird die *Terminate*-Nachricht und der darin enthaltene Hash im Netz verteilt. Terminieren darf ein Knoten erst, sobald er von allen seinen Nachbarn, zu denen er bidirektionale Kommunikationslinks hat, ein ACK zu seiner eigenen *TERMINATE*-Nachricht sowie von all diesen Nachbarn eine *Terminate*-Nachricht erhalten und diese per ACK bestätigt hat.

Das beschriebene Terminierungsverfahren funktioniert nur zuverlässig, wenn alle Knoten im Netz miteinander verbunden sind, d.h. für alle Paare von Knoten gibt es einen Pfad aus bidirektionalen Links, der sie verbindet. Problematisch sind Topologien, in denen ein Teil des Netzes nur über einen asymmetrischen Link $k_1 \rightarrow k_2$ mit dem restlichen Netz verbunden ist. In diesem Fall kann k_2 die Topologieinformation über seine Seite des Netzes nicht mit k_1 und damit der anderen Seite des Netzes teilen. Daher werden die Topologie-Hashs von k_1 und k_2 nicht übereinstimmen, sodass das Verfahren nicht terminiert. Da es in einem solchen Fall unmöglich ist, das Ziel zu erreichen, dass alle Knoten eine konsistente Topologie des gesamten Netzes kennen, lässt sich dieser Fall auch nicht sinnvoll lösen. Aus diesem Grund terminiert das Verfahren in solchen Fällen lediglich nach einen konfigurierbaren Maximallaufzeit mit einem Fehler.

4.2.3 Implementierung und Evaluierung

ATDP4W wurde als Schicht für das WiPS Framework implementiert. Die Schicht muss unterhalb von Schichten angeordnet sein, welche die erkannte Topologie verwenden. Schichten für Routing oder Clustering sind daher wie in Abb. 4.5 zu sehen oberhalb der Topologieerkennung anzuordnen. Unterhalb der Schicht zur Topologieerkennung benötigt die Implementierung von ATDP4W selbstverständlich die WiPS MAC-Schicht zum Senden von Nachrichten und die darunter liegenden Schichten.

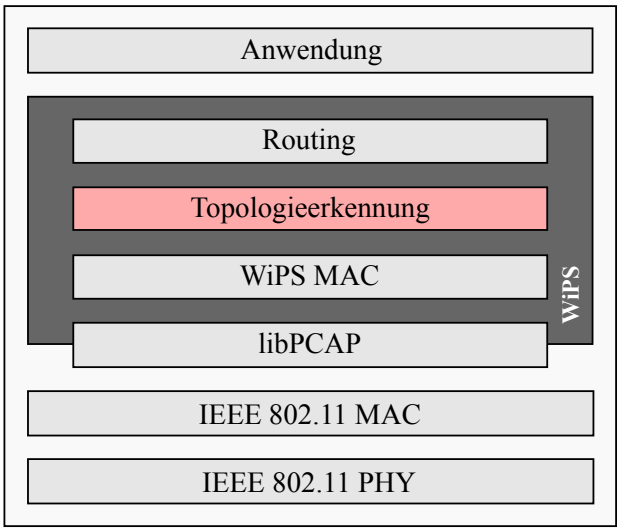


Abbildung 4.5: WiPS-Stack mit der Schicht zur Topologieerkennung

Die Implementierung wurde auf unterschiedlicher Hardware ausgeführt und terminierte stets mit konsistenten Topologien. Wie zu erwarten, entsprach die Zuverlässigkeit der ermittelten Links den konfigurierten Grenzwerten. Bei den Tests fiel auf, dass der Terminierungsprozess anfällig für Knotenausfälle ist. Fällt ein Knoten aus, nachdem der Terminierungsprozess bereits gestartet ist, so kann er die *TERMINATE*-Nachricht nicht bestätigen, sodass das Verfahren erst nach Erreichen der Maximallaufzeit mit einem Fehler terminiert. Dies ist aber auch ein Fall, in dem es nicht (mehr) möglich ist, mit einem der Knoten zu kommunizieren. Folglich ist es unmöglich, die Topologieinformation mit allen Knoten auszutauschen.

4.2.4 Zusammenfassung

Auch in wettbewerbsbasierten Kommunikationssystemen wie IEEE 802.11 wird Topologieinformation immer wichtiger. Obwohl die Interferenz- und Sensing-Topologie hier kaum nützlich sind, wird die Kommunikationstopologie in Mesh-Netzwerken benötigt, insbesondere zum Routing in Multihop-Netzwerken. In Kapitel 4.2.1 wurde daher die mit IEEE 802.11s standardisierte Topologieerkennung im *Hybrid Wireless Mesh Protocol (HWMP)* untersucht. Dabei zeigte sich, dass HWMP ähnliche Schwächen in Bezug auf die Topologieerkennung wie AODV aufweist. So wird die Linkqualität zwar berücksichtigt, aber kein Verfahren zu deren Ermittlung spezifiziert. Außerdem werden unidirektionale Links nicht ausreichend berücksichtigt. In Kapitel 4.2.2 wurde daraufhin mit *Automatic Topology Discovery Protocol for WiFi (ATDP4W)* ein Protokoll vorgestellt, welches sowohl die Linkqualität als auch unidirektionale Links berücksichtigt und die erkannte Topologie netzweit austauscht. Das Verfahren wird beim Hochfahren des Systems gestartet und misst daraufhin die Links des Netzwerks aus. Der netzweite Austausch der

erkannten Topologie erfolgt effizient. Das Verfahren stellt sicher, dass alle Knoten mit der selben Topologie terminieren. Die Implementierung von ATDP4W als Schicht in WiPS wurde in Kapitel 4.2.3 beschrieben und zeigte, dass das Verfahren unter realen Bedingungen funktioniert.

4.3 Verkehrskontrolle mittels Token Bucket

Viele Kommunikationssysteme erreichen ein gewisses Maß an Zuverlässigkeit durch schlichte Überprovisionierung, also indem sie beispielsweise deutlich mehr Bandbreite zur Verfügung stellen, als im Regelfall benötigt wird. Der traditionelle Nachteil dieses Ansatzes, dass Überprovisionierung zu höheren Kosten führt, spielt heute eine immer kleinere Rolle, da Bandbreite zunehmend günstiger wird. Die IEEE 802.11 basierten WLAN-Netzwerke sind ein gutes Beispiel für diesen Ansatz: Neue Versionen und Erweiterungen des Standards steigern meistens vor allem die Performance. So wurde die Übertragungsgeschwindigkeit von 1 MBit/s im ursprünglichen Standard von 1997 mit IEEE 802.11a bzw. IEEE 802.11g bereits wenig später auf 54 MBit/s erhöht. Seit IEEE 802.11n sind bis zu 600 MBit/s möglich, und IEEE 802.11ac und IEEE 802.11ad versprechen Datenraten von mehr als 1 GBit/s. Dadurch, dass neue Standards Daten mit einer höheren Übertragungsrate übertragen, erscheint das Netzwerk für den Nutzer oft nicht nur schneller, sondern auch zuverlässiger. Zwar gehen, beispielsweise durch Interferenzen, immer noch viele Rahmen bei der Übertragung verloren. Doch da die Neuübertragungen nun schneller sind, haben einzelne verlorene oder verfälschte Rahmen keinen so großen Einfluss mehr auf die vom Nutzer empfundene Qualität der Kommunikation (Quality of Experience, QoE). Indem mit einer höheren Datenrate übertragen wird, wird das drahtlose Medium darüber hinaus für die Übertragung der selben Datenmenge kürzer belegt, was die Wahrscheinlichkeit von Interferenzen reduziert. Die Kosten bei IEEE 802.11 Netzwerken sind dabei fast unbedeutend, da IEEE 802.11 Hardware verhältnismäßig günstig ist und für Übertragungen auf den genutzen ISM-Bändern keine Kosten anfallen (abgesehen von Kosten für Strom).

Durch Überprovisionierung kann allerdings keinerlei garantierte Zuverlässigkeit erreicht werden. Bei wettbewerbsbasierten drahtlosen Netzwerken wie IEEE 802.11 konkurrieren mehrere Knoten um die vorhandene Bandbreite. Dabei muss das drahtlose Medium oft nicht nur mit Knoten des eigenen Netzwerkes geteilt werden, sondern auch mit anderen Netzwerken in Reichweite, die den gleichen oder einen überlappenden Kanal nutzen. Insbesondere auf den von IEEE 802.11 genutzten 2,4 GHz Frequenzbereichen, welche in Deutschland nur 3 überlappungsfreie Kanäle bieten, teilen sich heutzutage häufig viele Netzwerke einen Kanal. Die mit IEEE 802.11g möglichen 54 MBit/s sind beispielsweise theoretisch ausreichend, um einen komprimierten HD Videostream mit 10-40 MBit/s zu streamen. Sehen aber mehrere Nutzer gleichzeitig einen solchen Videostream über konkurrierende IEEE 802.11 Netzwerke, so steht jedem Nutzer nur noch ein Bruchteil der Bandbreite zur Verfügung und der Videostream ist nicht (in der gleichen Qualität) möglich. Durch Überprovisionierung wird daher bestenfalls erreicht, dass die vorhandene Bandbreite *meistens* ausreichend ist, was keineswegs sicherstellt, dass dies *immer* der Fall ist. Darüber hinaus ist es oft schwierig zu bestimmen, unter welchen Bedingungen Überprovisionierung ausreichend ist.

Verkehrskontrolle versucht dagegen die Zuverlässigkeit des Netzwerkes zu erhöhen, indem sie die von den Knoten genutzte Bandbreite überwacht (Verkehrsüberwachung, engl.: Traffic Monitoring) und ggf. anpasst (Verkehrsformung, engl.: Traffic Shaping). So kann beispielsweise jedem Knoten ein gewisser Anteil der vorhandenen Bandbreite durch Reservierung zugeteilt werden. Im Folgenden wird überwacht, dass kein Knoten mehr Bandbreite nutzt, als für ihn

reserviert wurde. Versucht eine Anwendung auf einem Knoten, mehr Daten zu senden als vereinbart, so muss der Knoten entweder Pakete verzögern oder verwerfen, um seine vereinbarte Reservierung einzuhalten. Unter der Voraussetzung, dass alle Knoten in Reichweite sich an die Reservierungen halten und Überreservierung verhindert wird, d.h. insgesamt nicht mehr als die vorhandene Bandbreite reserviert wird, können Überlastsituationen vermieden werden. Letztlich lässt sich so die Zuverlässigkeit, auch wettbewerbsbasierter Netzwerke, steigern.

Der Token Bucket Algorithmus ist ein bekanntes Verfahren zur Verkehrsformung. In diesem Kapitel wird zunächst der Stand der Technik vorgestellt und dabei auch der Token Bucket Algorithmus erklärt. Anschließend wird ein für drahtlose Netzwerke optimiertes Verfahren zur Verkehrsformung auf Basis des Token Bucket Algorithmus vorgestellt. Weiterhin wird aufgezeigt, wie aus dem Token Bucket Algorithmus auch eine Metrik abgeleitet werden kann, welche Überlastsituationen erkennt und damit zur Verkehrsüberwachung eingesetzt werden kann. Letztlich werden die Implementierung der vorgestellten Verfahren und Evaluierungsergebnisse präsentiert.

4.3.1 Stand der Technik

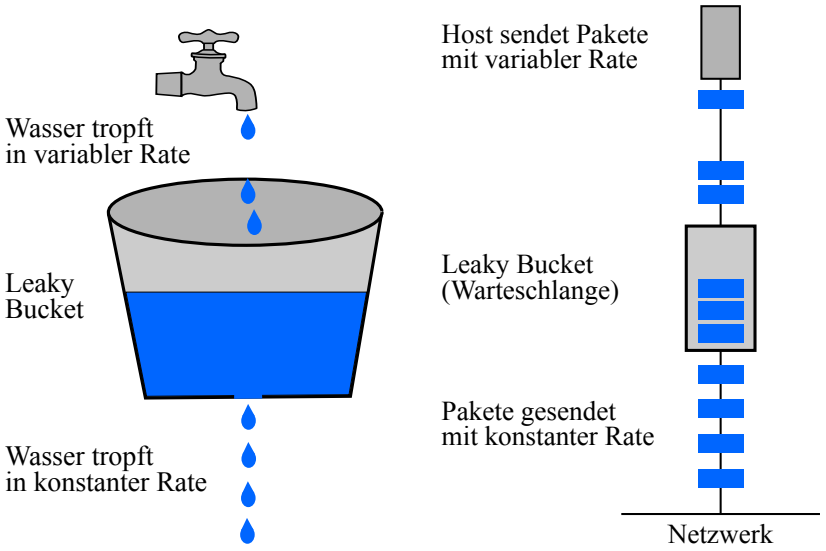
Wir betrachten zunächst existierende Verfahren zur Verkehrsformung, den Leaky Bucket und den Token Bucket Algorithmus sowie die für IEEE 802.11 Netzwerke besser geeignete Variante Time Token Bucket. Anschließend betrachten wir Verfahren zur Verkehrsüberwachung, die dazu geeignet sind, Überlastsituationen drahtloser Netzwerke zu erkennen.

4.3.1.1 Verkehrsformung

Verkehrsformung bezeichnet die Anpassung des Verkehrs auf Senderseite an gewisse Vorgaben. Meist wird die Bandbreite, mit der ein Host in dem Netzwerk senden kann, durch Verkehrsformung (künstlich) beschränkt. Sind n Knoten mit einem Netzwerk verbunden, kann man z.B. die Bandbreite jedes Knotens auf $1/n$ der Gesamtbandbreite beschränken, um eine faire Verteilung der Bandbreite zwischen den Knoten zu erzielen. Prinzipiell kann Verkehr geformt werden, indem Pakete entweder verzögert oder verworfen werden. Im Folgenden werden existierende Verfahren vorgestellt, die zur Verkehrsformung eingesetzt werden können.

Leaky Bucket Der Leaky Bucket Algorthmus [Tan03] ist ein sehr einfaches Verfahren zur Verkehrsformung. Man stelle sich einen Eimer vor, der im Boden ein kleines Loch hat, wie in Abb. 4.6 (a). Unabhängig davon, mit welcher Geschwindigkeit Wasser von oben in den Eimer fließt, tropft das Wasser unten in einer stets konstanten Rate aus dem Eimer. Nur wenn der Eimer leer ist, ändert sich die Rate, mit der das Wasser aus dem Eimer tropft, auf Null. Falls mehr Wasser von oben in den Eimer fließt, als dieser fasst, so fließt der Eimer über. Auch in diesem Fall bleibt die Rate, mit der das Wasser aus dem Loch im Eimerboden tropft, konstant.

Dieses Verfahren lässt sich auf Netzwerke übertragen: Der Eimer ist eine Warteschlange mit begrenzter Größe. Wenn der Hostcomputer ein Paket sendet, so wird dieses in der Warteschlange angefügt, analog zum Wasser, das von oben in den Eimer fließt. Ist die Warteschlange voll, wird das Paket verworfen, so wie der Eimer überläuft, wenn er voll ist. In einem konstanten Intervall wird je ein Paket aus der Warteschlange genommen, sofern vorhanden, und auf dem Netzwerk gesendet. Auf diese Weise erzielt man, dass der Host mit einer gleichmäßigen Datenrate auf dem Netzwerk sendet, unabhängig davon, mit welcher Datenrate er Pakete generiert.



(a) Ein tropfender Wassereimer

(b) Ein Netzwerkinterface mit Leaky Bucket Algorithmus

Abbildung 4.6: Der Leaky Bucket Algorithmus: Veranschaulichung als tropfender Wassereimer und als Netzwerkinterface.

Sind die Pakete nicht alle gleich groß, so wird das Verfahren leicht angepasst: Damit auch bei unterschiedlich großen Paketen mit einer gleichmäßigen Datenrate auf dem Netzwerk gesendet wird, reicht es nicht, die Pakete in einem konstanten Intervall zu senden. Stattdessen wird ein Zähler n eingeführt, der angibt, wie viele Bytes gesendet werden dürfen. Dieser Zähler wird in einem konstanten Intervall Δt [s] auf einen konstanten Wert von k [Bytes] zurückgesetzt. Mit jedem gesendeten Paket wird der Zähler um die Paketgröße p reduziert. Sobald der Zähler kleiner ist als die Paketgröße, d.h. $n < p$, so muss gewartet werden, bis der Zähler zurückgesetzt wurde. Solange werden weitere ankommende Pakete in der Warteschlange gespeichert. Falls mehr Pakete eintreffen, als in die Warteschlange passen, so werden diese verworfen. Letztlich erzielt man damit, dass maximal mit einer Senderate von $k/\Delta t$ [Bytes/s] gesendet wird.

Token Bucket Das Leaky Bucket-Verfahren begrenzt die nutzbare Bandbreite. Damit erlaubt es auch keinerlei Bursts, die über diese Bandbreite hinaus gehen. Dies bedeutet, dass letztlich der gesamte Verkehr verlangsamt wird, auch wenn nur sporadisch Verkehr auftritt und die genutzte Bandbreite im Schnitt auch ohne Verlangsamung im gewünschten Bereich wäre. Es kann also durchaus wünschenswert sein, kurzfristig die Nutzung der vollen Bandbreite zu gestatten, also kurzfristige Bursts in einem begrenztem Umfang zu erlauben. Das Token Bucket-Verfahren [Tan03] ist ein Ansatz zur Verkehrsformung, der eine durchschnittliche Bandbreite sicherstellt, dabei aber kurzfristige Bursts bis zu einer parametrisierbaren Grenze gestattet. Manchmal wird das Token Bucket-Verfahren auch mit dem Leaky Bucket-Verfahren gleichgesetzt. Beispielsweise beschreibt Turner 1986 [Tur86] ein Verfahren, welches dem hier als Token Bucket beschriebenen Verfahren gleichkommt, nennt es aber Leaky Bucket. Auch ATM (Asynchronous Transfer Mode) beschreibt unter dem Begriff Leaky Bucket das gleiche Verfahren wie Turner. Hier wird die von Tanenbaum [Tan03] verwendete Unterscheidung zwischen Token Bucket und Leaky Bucket benutzt.

Beim Token Bucket-Verfahren werden ankommende Pakete ebenfalls in einer Warteschlange gesammelt, wie Abb. 4.7 veranschaulicht. Um ein Paket auf dem Netzwerk zu senden, sind sog. To-

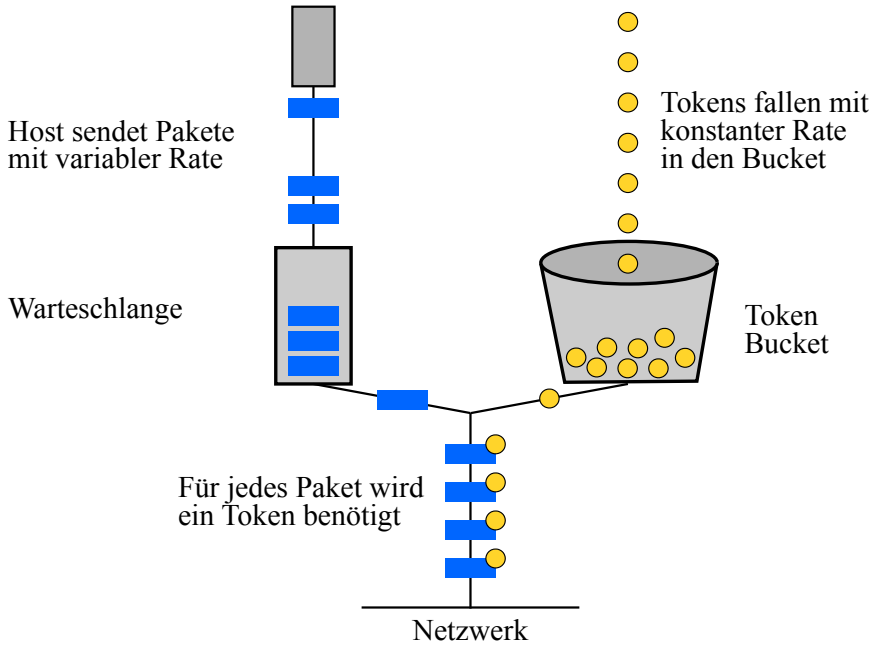


Abbildung 4.7: Das Token Bucket-Verfahren begrenzt die durchschnittliche Bandbreite, erlaubt aber Bursts bis zur Größe des Buckets.

kens nötig. Tokens kann man sich wie Münzen vorstellen, mit denen das Senden eines Rahmens bezahlt werden muss. Der Bucket stellt hier nicht die Paket-Warteschlange dar, sondern sammelt die Tokens, ähnlich einer Spardose. Tokens werden mit einem konstanten Auffüll-Intervall Δt [s] in den Bucket eingeworfen. Sie sammeln sich bis der Bucket voll ist, d.h. die Anzahl Tokens im Bucket die Bucket-Größe n erreicht hat. Weitere Tokens fließen über, verfallen also ungenutzt. Zum Senden eines Pakets auf dem Netzwerk ist im einfachsten Fall ein Token je Paket nötig. Wenn Tokens im Bucket gesammelt wurden, kann also kurzzeitig ein Burst von bis zu n Paketen gesendet werden. Im Durchschnitt wird die Senderate allerdings durch das Auffüll-Intervall auf $1/\Delta t$ [Pakete/s] begrenzt. Wenn die Pakete nicht alle gleich groß sind, ist es sinnvoller, die Anzahl Tokens, die zum Senden eines Pakets nötig ist, von der Paketgröße abhängig zu machen. Beispielsweise könnte ein Token das Senden eines Bytes erlauben. Bei einem Auffüll-Intervall von t können somit durchschnittlich $1/t$ [Bytes/s] gesendet werden, und der maximal mögliche Burst ist durch die Bucket-Größe n beschränkt auf n Bytes[3]. Das Token Bucket-Verfahren erlaubt es also, den Verkehr auf eine durchschnittliche Bandbreite zu begrenzen, und dennoch kurzzeitig Bursts zu gestatten. Die durchschnittliche Bandbreite kann über das Auffüll-Intervall und die Token-Größe parametrisiert werden, die maximal möglichen Bursts über die Bucket-Größe.

Time Token Bucket Die meisten Netzwerke senden Daten mit einer festen Datenrate. Bei IEEE 802.11 ist es dagegen möglich, Daten mit unterschiedlichen Modulationen und damit unterschiedlicher Bandbreite zu senden. Neben der Basis-Datenrate von 1 MBit/s wurden im Laufe der Zeit immer schnellere Datenraten hinzugefügt, so sind z.B. bis zu 600 MBit/s mit IEEE 802.11n möglich. Dies führt dazu, dass auf dem drahtlosen Medium Datenrahmen mit unterschiedlichen Datenraten miteinander konkurrieren. Beschränkt man nun über das Token Bucket-Verfahren, wie viele Bytes pro Sekunde die Knoten senden dürfen, so führt dies dazu, dass bei gleicher Konfiguration des Token Buckets ein langsam sendender Knoten das drahtlose Medium

[3]Dadurch, dass während des Sendens noch neue Tokens in den Bucket fallen können, ist der maximal mögliche Burst tatsächlich leicht größer.

länger nutzen darf als ein Knoten, der mit einer hohen Datenrate sendet. Es erscheint daher sinnvoller, nicht die Datenmenge zu beschränken, die ein Knoten in einer gewissen Zeit senden darf, sondern die Sendezeit. Diesen Ansatz nennen wir hier das Time Token Bucket-Verfahren. Das Verfahren ist identisch zum normalen Token Bucket-Verfahren mit dem einzigen Unterschied, dass ein Token nicht das Recht zum Senden einer gewissen Menge Bytes repräsentiert, sondern das Benutzen des Mediums für eine bestimmte Zeit. Zum Senden eines Rahmens auf dem drahtlosen Medium müssen daher so viele Tokens im Bucket sein, wie es der Sendezeit des Rahmens (inkl. Overhead wie Header und Interframe Spaces) entspricht. Dieser Ansatz wurde von Tan und Guttang in [TG04] unter dem Namen Time-based Regulator beschrieben.

4.3.1.2 Verkehrsüberwachung zur Erkennung der Auslastung eines drahtlosen Netzwerks

Neben der Verkehrsformung ist auch der Aspekt der Verkehrsüberwachung Teil der Verkehrskontrolle. Hier wollen wir uns auf einen speziellen Teilaspekt der Verkehrsüberwachung beschränken, der bei drahtlosen Netzwerken besonders relevant ist: Die Erkennung der Auslastung des drahtlosen Mediums. Damit Knoten eines drahtlosen Netzwerkes ihren Verkehr je nach Auslastung des Mediums anpassen können, müssen sie zunächst ein Maß für die Auslastung zur Verfügung haben. Wir gehen daher hier auf existierende Verfahren ein, die dazu geeignet sind, die Auslastung des drahtlosen Netzwerkes zu messen oder abzuschätzen, bzw. Überlastsituationen zu erkennen.

Channel Busy Fraction (CBF) Die Channel Busy Time (CBT) ist eine Metrik, die von manchen WLAN-Adaptern zur Verfügung gestellt wird. Der Adapter misst dazu, wie lange die Energie auf dem drahtlosen Medium über einem gewissen Grenzwert liegt, der Kanal also als belegt (busy) gilt. Die CBT kann man dann in Relation zur Messdauer setzen, und erhält dadurch die Channel Busy Fraction (CBF) [DKS10]. Nicht beachtet wird dabei, dass bei IEEE 802.11 zwischen zwei Rahmen immer eine gewisse Pufferzeit (Interframe Space) eingehalten werden muss, und auch während Wettbewerbsphasen das Medium nicht genutzt werden kann. Durch diesen Overhead ist es also selbst theoretisch nicht möglich, dass das Medium jederzeit belegt ist, also eine CBF von 100% erreicht wird. Je nachdem, mit welcher Rahmengröße und welcher Geschwindigkeit gesendet wird, ändert sich die maximal mögliche CBF. Die Interpretation der CBF ist daher schwierig. Dazu kommt, dass die Festlegung des Grenzwertes, ab dem das Medium als belegt gilt, nicht einfach ist. Möglichkeiten sind ein fest konfigurierter Grenzwert, das einmalige Bestimmen des Grenzwerts basierend auf dem gemessenen Rauschlevel sowie das dynamische Anpassen des Grenzwerts je nach aktuell gemessenem Rauschlevel. Dadurch, dass unterschiedliche Adapter einen unterschiedlichen Grenzwert verwenden, können mehrere Adapter in unmittelbarer Nähe voneinander zu deutlich unterschiedlichen CBF-Werten kommen. Zur Bestimmung von Überlastsituationen müsste ein weiterer Grenzwert festgelegt werden, ab dem das Medium als überlastet gilt, beispielsweise einer CBF von 80%. Auch hier ist nicht offensichtlich, wie dieser Grenzwert bestimmt werden soll.

Neben der Schwierigkeit, die CBF richtig zu interpretieren, kommt noch ein praktisches Problem hinzu: Die CBT wird bestimmt, indem die Channel Survey Details aus den Hardware-Registern des WLAN-Adapters ausgelesen werden. Allerdings erlauben die Treiber vieler Adapter dies nicht, sodass die CBF nicht ohne weiteres bestimmt werden kann. In [Ach+08; SAB08; Jar+05] wird die CBF zur Bestimmung der Auslastung des Mediums eingesetzt, allerdings wurde in diesen Arbeiten der Treiber des WLAN-Adapters (bzw. Kerneltreiber) angepasst, um die CBT auslesen zu können. Mittlerweile unterstützen einige wenige Adapter in Verbindung mit neuen

Treiber-Versionen ohne Anpassung das Auslesen des CBT. Beispielsweise gelang das Auslesen der Channel Survey Daten mit der Qualcomm Atheros AR928X Mini PCIe-Karte unter Verwendung des ath9k Treibers unter Linux 5.0.2. Allerdings unterstützt aktuell der Großteil der verfügbaren Karten das Auslesen der CBT nicht, sodass die CBF keine generell verfügbare Metrik ist.

Medium Utilization Fraction Ein weiterer Ansatz, die Auslastung des Mediums abzuschätzen, liegt darin, auf dem Medium jeden Rahmen mitzuschneiden und aus der Paketlänge und genutzten Datenrate die Zeit zu berechnen, die zur Übertragung des Rahmens benötigt wurde. Summiert man die Übertragungszeit aller mitgeschnittenen Rahmen auf, so erhält man die Medium Utilization Time. Setzt man diese in Relation zur Messdauer, so erhalten wir die Medium Utilization Fraction [Ach+08], die angibt, wie hoch der Zeitanteil ist, während dem Übertragungen stattfinden. Bei der Berechnung können Interframe Spaces zwischen den Übertragungen teilweise berücksichtigt werden und der Overhead für Wettbewerbsphasen abgeschätzt werden. In diesem Aspekt ist die Medium Utilization Fraction der Channel Busy Fraction überlegen. Außerdem ist hier kein Grenzwert nötig, da jeder vom Adapter empfangene Rahmen gezählt werden kann, egal wie schwach das Signal empfangen wurde. Finden auf dem gleichen Frequenzbereich Übertragungen eines anderen Protokolls, wie beispielsweise Bluetooth oder IEEE 802.15.4 statt, so führen diese Übertragungen nicht zu einem Empfang beim IEEE 802.11 Adapter. Dies bedeutet, dass die Medium Utilization Time nicht die Zeit enthält, während der andere Protokolle das Medium belegen. Dies ist ein Nachteil verglichen mit der Channel Busy Time, welche unabhängig vom Protokoll jegliche Kanalbelegung über dem Grenzwert berücksichtigt. Ein weiterer entscheidender Nachteil ist der Overhead, der dadurch entsteht, dass sämtliche Rahmen mitgeschnitten und verarbeitet werden müssen, um die Medium Utilization Time berechnen zu können. Schließlich muss der WLAN-Adapter zur Bestimmung der Medium Utilization Time alle Rahmen mitschneiden, also auch Managment-Rahmen wie Beacons. Dies ist im Monitoring-Modus möglich, welcher aber nicht von allen Karten bzw. Treibern unterstützt wird. Allerdings ist die Unterstützung für den Monitoring-Modus häufiger vorzufinden, als die Unterstützung für das Auslesen der Channel Busy Time. Beispielsweise unterstützt neben der Qualcomm Atheros AR928X Mini PCIe-Karte auch der Ralink RT5370 USB-Adapter den Monitoring-Modus, nicht aber das Auslesen der Channel Busy Time.

4.3.2 Bandbreitenreservierung und Verkehrsformung

Im Folgenden wird ein Konzept beschrieben, wie Bandbreite in einem IEEE 802.11 Netzwerk reserviert und die Einhaltung der Reservierungen durch Verkehrsformung sichergestellt werden kann. Dabei kommt das zuvor beschriebene Time Token Bucket-Verfahren zu Einsatz. Das hier beschriebene Konzept basiert auf dem in [MKG17] veröffentlichten Konzept.
Im Folgenden sei $G = (V, E)$ ein IEEE 802.11 Netzwerk, wobei V die Menge der Knoten bezeichnet und $E \subseteq (V \times V)$ die Knoten. Wir betrachten alle IEEE 802.11 Knoten auf dem gleichen Kanal als Bestandteil des Netzwerkes G, also auch Knoten, die nicht zu unserem Netzwerk gehören und daher nicht das hier beschriebene Verfahren zur Verkehrskontrolle einsetzen. Jene Knoten, die unser Verfahren zur Verkehrskontrolle nutzen, bezeichnen wir als interne Knoten V^{int}, andere Knoten als externe Knoten V^{ext}, sodass $V = V^{int} \cup V^{ext}$. Das auf die internen Knoten beschränkte Netzwerk wird mir $G^{int} = (V^{int}, E^{int})$ bezeichnet. Mit „Bandbreite" bezeichnen wir im Folgenden stets den Anteil an Sendezeit. Das bedeutet, dass ein Knoten, dem eine Bandbreite von 50% zugeordnet ist, durchschnittlich die Hälfte der Zeit senden darf.

4.3.2.1 Bandbreitenreservierung

Bandbreitenreservierungen sind eine Möglichkeit, um in einem drahtlosen Netzwerk mit mehreren Knoten sicherzustellen, dass es nicht zu Überlastsituationen auf dem drahtlosen Medium kommt. Dabei muss zum einen sichergestellt werden, dass sich jeder Knoten an seine Reservierung hält, also nicht mehr sendet, als vereinbart. Dies kann mit Verkehrsformung erreicht werden und wird in Kapitel 4.3.2.2 betrachtet. Außerdem muss sichergestellt werden, dass die vereinbarten Reservierungen in Summe nicht die insgesamt vorhandene Bandbreite überschreiten. Das bedeutet, dass Reservierungen andernfalls abgelehnt werden müssen. Ist außerdem auf dem gleichen Kanal noch mit anderen IEEE 802.11 Netzwerken oder Übertragungen anderer Protokolle wie IEEE 802.15.4 zu rechnen, so sollte außerdem der Bedarf für diese Knoten abgeschätzt und berücksichtigt werden.

Die Bandbreite, die interne Knoten des Netzwerks G in Summe maximal reservieren dürfen, bezeichnen wir mit $availableBw_{G^{int}} \in [0,1]$. Dieser Wert gibt also den Anteil der Zeit an, den Knoten des Netzwerks G^{int} maximal zum Senden verwenden dürfen. Wird viel Verkehr von anderen Netzwerken auf dem gleichen Kanal erwartet oder eine besonders hohe Zuverlässigkeit benötigt, so könnte dieser Wert beispielsweise konservativ bei 20% liegen. Ist dagegen nicht mit Verkehr von anderen Netzwerken zu rechnen, so wären auch 80% denkbar. Ein Verfahren, was die Grenze des Möglichen selbstständig dynamisch ermittelt, wird in Kapitel 4.4 beschrieben. Hier gehen wir davon aus, dass $availableBw_{G^{int}}$ per Konfiguration auf einen geeigneten Wert festgelegt wird.

Die von einem internen Knoten v reservierte und erfolgreich zugewiesene Bandbreite bezeichnen wir mit $assignedBw_v$. Nun soll gelten, dass die Summe der Bandbreiten, die allen Knoten $v \in V^{int}$ des Netzwerks G^{int} zugewiesenen worden sind, $availableBw_{G^{int}}$ nicht überschreiten darf:

$$\sum_{v \in V^{int}} assignedBw_v = assignedBw_{G^{int}} \leq availableBw_{G^{int}} \tag{4.3}$$

Die Bandbreitenreservierung erfolgt bei einem zentralen Knoten, der die Rolle des *Bandbreitenmanagers* übernimmt. Bevor ein Knoten senden darf, muss er an den Bandbreitenmanager eine Reservierungsanfrage senden. Diese enthält folgende Felder:

- $nodeID \in V^{int}$ gibt die Adresse des Knoten an, für den Bandbreite reserviert werden soll.
- $bandwidth \in [0,1]$ gibt die Menge an Bandbreite an, die reserviert werden soll.
- $action \in \{reserve, update, cancel, accept, deny\}$ beschreibt, ob die Bandbreite neu reserviert, aktualisiert oder storniert werden soll, bzw. ob die Anfrage vom Bandbreitenmanager bestätigt oder abgelehnt wird.

Solange ein Knoten noch keine Bandbreite zugewiesen bekommen hat, darf er eigentlich nicht senden. Um dennoch seine erste Reservierung beim Bandbreitenmanager anfragen zu können, darf ein Knoten, der noch keine Bandbreite zugewiesen bekommen hat, eine Reservierungsanfrage pro Sekunde an den Bandbreitenmanager senden. Da die Reservierungsanfragen außerdem sehr klein sind, entsteht durch sie nicht viel Bandbreitennutzung. Hat ein Knoten bereits Bandbreite zugewiesen bekommen, so muss er diese nutzen, um weitere Reservierungsanfragen zu senden, kann dann aber so viele Anfragen senden, wie ihm Bandbreite zur Verfügung steht.

Das Senden der Reservierungsanfrage erfolgt entweder in einem Singlehop-Netzwerk durch eine direkte Nachricht vom reservierenden Knoten zum Bandbreitenmanager oder aber im Falle eines Multihop-Netzwerks durch eine Nachricht, die per unicast Routing versendet wird. Es wird davon ausgegangen, dass hierzu eine geeignete Routing-Schicht vorhanden ist.

Erhält der Bandbreitenmanager eine Reservierungsanfrage vom Knoten $nodeID$, so prüft er, ob es möglich ist, $assignedBw_{nodeID}$ auf den gewünschten Wert *bandwidth* zu setzen, ohne dabei die Bedingung aus Formel (4.3) zu verletzen. Ist dies nicht möglich, lehnt er die Anfrage ab, indem er sie zum anfragenden Knoten zurücksendet und dabei das Feld *action* auf den Wert *deny* setzt. Andernfalls akzeptiert der Bandbreitenmanager die Anfrage, indem er die Nachricht zurücksendet und dabei das Feld *action* auf *accept* setzt. Der Bandbreitenmanager führt also eine Zugangskontrolle (engl.: admission control) durch, welche sicherstellt, dass die Ressourcen, in diesem Fall Bandbreite, ausreichend sind, bevor er die angeforderte Nutzung des Netzwerks gestattet. Benötigt ein Knoten mehr oder weniger Bandbreite als zuvor, kann er dem Bandbreitenmanager eine Nachricht senden, bei der das Feld *action* auf den Wert *update* gesetzt ist. Mit *action=cancel* kann er seine Reservierung schließlich auflösen. Die Antwort erfolgt in beiden Fällen genauso wie bei der Erstanfrage. Das Auflösen einer Reservierung ist selbstverständlich immer möglich, d.h. sie wird vom Bandbreitenmanager stets bestätigt, die Antwort dient hier lediglich als Quittung.

4.3.2.2 Verkehrsformung

Das Ziel der Verkehrsformung ist hier, sicherzustellen, dass die Knoten im Netzwerk maximal so viel Bandbreite nutzen, wie von ihnen reserviert wurde. Da der Zugriff auf das drahtlose Medium nicht ohne weiteres zentral gesteuert werden kann, wie etwa bei einem Router, der mehrere drahtgebundene Knoten verbindet und ihren Verkehr formen könnte, muss die Verkehrsformung auf den Knoten selbst durchgeführt werden. Das bedeutet, jeder (interne) Knoten v verwendet ein Verfahren zur Verkehrsformung, was sicherstellt, dass er nicht mehr als die ihm durch Bandbreitenreservierung zugewiesene Bandbreite $assignedBw_v$ nutzt. Zur Verkehrsformung kommt dabei das in Kapitel 4.3.1.1 beschriebene *Time Token Bucket*-Verfahren zum Einsatz. Der Time Token Bucket-Mechanismus des Knotens v habe folgende Parameter:

- $bucketSize_v \in \mathbb{N}_0$ [Tokens] gibt die Anzahl Tokens an, die in den Token Bucket des Knotens v passen, bis er überläuft.
- $tokenSize_v \in \mathbb{N}$ [µs] gibt an, wie lange der Knoten v das Medium für einen Token benutzen darf.
- $refillInt_v \in \mathbb{N}$ [µs] ist das Intervall, mit dem der Bucket befüllt wird, d.h. die Zeit zwischen zwei in den Bucket fallenden Tokens.

Wir setzen die Token-Größe zunächst fest auf einen geeigneten kleinen Wert, z.B.:

$$\forall v \in V \quad tokenSize_v = 10\,\mu\text{s} \tag{4.4}$$

Über die beiden anderen Parameter können nun zwei zentrale Größen gesteuert werden. Die durchschnittliche Bandbreite, mit der der Knoten sendet, die hier $assignedBw_v$ betragen soll, ergibt sich aus der Token-Größe $tokenSize_v$ und dem Refill-Intervall $refillInt_v$:

$$assignedBw_v = \frac{tokenSize_v}{refillInt_v} \tag{4.5}$$

Die zugewiesene Bandbreite $assignedBw_v$ wird durch die Bandbreitenreservierung festgelegt und ist daher ebenso wie die Token-Größe $tokenSize_v$ bekannt, somit lässt sich das Refill-Intervall $refillInt_v$ aus Formel (4.5) bestimmen.

Die zweite zentrale Größe, die das Token Bucket-Verfahren begrenzt, ist die Größe des maximal möglichen Bursts. Multipliziert mit der Token-Größe $tokenSize_v$ ergibt sich aus der Größe des

Buckets $bucketSize_v$ die maximale Belegungsdauer des Mediums, also die maximale Länge von Bursts $maxBurst_v$:

$$maxBurst_v = tokenSize_v \cdot bucketSize_v \tag{4.6}$$

Beide Parameter müssen mindestens so groß gewählt werden, dass der größtmögliche Rahmen mit $payload_{Max}$ [Bytes] bei der niedrigsten verwendeten Übertragungsrate $rate_{PayloadMin}$ [Bytes/s] gesendet werden kann. Die Übertragungszeit dafür beträgt:

$$txTime_{v,Max} = \frac{payload_{Max}}{rate_{PayloadMin}} + \frac{MLO}{rate_{Header}} + PLO \tag{4.7}$$

Hierbei wird der durchschnittliche Physical Layer Overhead PLO [s], der sich aus Zeiten für Interframe Spaces wie z.B. dem DCF Interframe Space (DIFS), dem Backoff-Intervall sowie der PHY-Präambel zusammensetzt, berücksichtigt. Außerdem muss der Overhead auf MAC-Ebene MLO [Bytes] , der den IEEE 802.11 Header enthält, berücksichtigt werden. Bei IEEE 802.11 ist es möglich, dass die Daten mit einer anderen (schnelleren) Datenrate als der Header gesendet werden, daher muss für den Header die ggf. abweichende Datenrate $rate_{Header}$ berücksichtigt werden. Damit alle Rahmen gesendet werden können, soll nun der maximal mögliche Burst $maxBurst_v$ mindestens $txTime_{v,Max}$ lang sein, also:

$$txTime_{v,Max} \leq maxBurst_v \tag{4.8}$$

Da die Token-Größe festgesetzt wurde, ergibt sich aus Formel (4.6) und Formel (4.8) die minimale Bucket-Größe $bucketSize_v$. Damit Bursts soweit wie möglich begrenzt werden, wird dieser minimale Wert für $bucketSize_v$ (aufgerundet) genutzt:

$$bucketSize_v = \left\lceil \frac{txTime_{v,Max}}{tokenSize_v} \right\rceil \tag{4.9}$$

Weiterhin ergibt sich die Zeit, um den Bucket vollständig zu füllen, falls er zunächst leer ist und während des Füllens keine Rahmen gesendet werden, wie folgt:

$$fillTime_v = bucketSize_v \cdot refillInt_v \tag{4.10}$$

Anstatt die Token-Größe $tokenSize_v$ festzusetzen und daraus das Refill-Intervall $refillInt_v$ zu bestimmen, wäre es auch denkbar, dass Refill-Intervall festzusetzen und daraus die Token-Größe zu berechnen. Dies hat allerdings den Nachteil, dass es dazu kommen kann, dass zwei oder mehr Knoten dadurch unbeabsichtigt synchronisiert auf das Medium zugreifen, und damit die Kollisionswahrscheinlichkeit steigt. Starten zwei Knoten zufälligerweise oder durch gemeinsames Einschalten exakt gleichzeitig, und nutzen sie das gleiche Refill-Intervall, so erhalten sie fortan immer zeitgleich neue Tokens. Warten beide auf das letzte Token, dass sie benötigen, um eine Nachricht zu senden, so erhalten sie dieses ebenfalls gleichzeitig und beginnen gleichzeitig den Sendevorgang. Obwohl der Random Backoff-Mechanismus von IEEE 802.11 versucht, diesen Wettbewerb aufzulösen, ist die Wahrscheinlichkeit für eine Kollision dennoch erhöht. Dies ist insbesondere dann der Fall, wenn die Knoten Hidden Stations sind. Darüber hinaus sieht IEEE 802.11 keinen Wettbewerb vor, falls das Medium zuvor als frei erkannt wurde, sodass eine Kollision in diesem Fall höchstwahrscheinlich ist. Da beide Knoten immer gleichzeitig neue Tokens erhalten, tritt diese Situation immer wieder auf und löst sich nicht von selbst auf, außer auf

Grund von Ungenauigkeiten der Uhren. Setzt man hingegen die Token-Größe für alle Knoten fest und bestimmt aus Formel (4.5) das Refill-Intervall, so haben Knoten mit unterschiedlicher zugeordneter Bandbreite unterschiedliche Refill-Intervalle. Dadurch, dass Knoten neue Tokens nicht im selben Intervall erhalten, ist es deutlich unwahrscheinlicher, dass die Sendeversuche der Knoten ungewollt synchronisiert werden.
Wir betrachten nun ein Beispiel und bestimmen die Parameter, die sich aus den Anforderungen der Anwendung ergeben. Sei die maximale Payload-Größe der Anwendung $payload_{Max} = 250B$ und wir verwenden IEEE 802.11b und die minimal mögliche Datenrate sei $rate_{PayloadMin} = 1MBit/s$. Wir berücksichtigen den MAC Layer Overhead mit $MLO = 34B$ für den IEEE 802.11 Header und die Checksumme und den Physical Layer Overhead mit $PLO = 274\mu s$, bestehend aus $50\mu s$ DIFS, $80\mu s$ (durchschnittlicher) Backoff-Zeit und $144\mu s$ für die (lange) PHY-Präambel. Der IEEE 802.11 Header werde mit $rate_{Header} = 1MBit/s$ übertragen. Wir berechnen zunächst nach Formel (4.7) die Übertragungsdauer inkl. Overhead für eine Nachricht mit der maximalen Payload von $250B$:

$$txTime_{v,Max} = \frac{250B}{1MBit/s} + \frac{34B}{1MBit/s} + 274\mu s = 2.546\mu s \tag{4.11}$$

Wie zuvor setzen wir die Token-Größe auf $tokenSize_v = 10\mu s$. Nach Formel (4.9) ergibt sich damit die Bucket-Größe wie folgt:

$$bucketSize_v = \left\lceil \frac{2.564\mu s}{10\mu s} \right\rceil = 256\ [Tokens] \tag{4.12}$$

Weiterhin sei die Anforderung der Anwendung, alle $appInt_v = 1500ms$ einen Rahmen mit bis zu $250B$ zu senden. Damit ergibt sich die maximal benötigte und daher zugewiesene Bandbreite wie folgt[4]:

$$assignedBw_v = \frac{txTime_{v,Max}}{appInt_v} = \frac{2.546\mu s}{1500ms} \approx 0{,}17\% \tag{4.13}$$

Dass Refill-Intervall beträgt nach Formel (4.5) damit:

$$refillInt_v = \frac{tokenSize_v}{assignedBw_v} = \frac{10\mu s}{0{,}17\%} \approx 5.882\mu s \tag{4.14}$$

Eine vollständige Füllung des Buckets benötigt nach Formel (4.10) dann:

$$fillTime_v = 256 \cdot 5.882\mu s \approx 1.500ms \tag{4.15}$$

Dies entspricht der Zeit $appInt_v$. Es ist also mit dieser Konfiguration wie von der Anwendung gefordert möglich, alle $appInt_v = 1500ms$ einen Rahmen mit $250B$ Payload zu versenden, denn die dazu benötigten 256 Token werden alle $fillTime_v = 1500ms$ aufgefüllt.
Der Time Token Bucket-Mechanismus wird wie in Kapitel 4.3.1.1 beschrieben verwendet. Zu beachten ist, dass sich bei IEEE 802.11 Rahmen zusätzlich im Transceiver (bzw. dessen Treiber) in einer Warteschlange anstauen können, während das Medium belegt ist. Dies würde dazu führen, dass Bursts entstehen können, die länger als $maxBurst_v$ sind. Um dies zu verhindern, dürfen nur Rahmen an den Transceiver übergeben werden, sobald der zuvor an den Transceiver gesendete Rahmen bereits gesendet (oder durch Timeout verworfen) wurde. Die Tokens zum Senden werden erst in dem Moment aus dem Bucket genommen, in dem der Rahmen an den Transceiver übergeben wird. Ist das Medium über längere Zeit belegt, bedeutet dies also, dass sich die Tokens im Bucket solange ansammeln und ggf. überlaufen. Dieses Verhalten nutzen wir im Folgenden aus, um Überlastung des Mediums anhand überlaufender Tokens zu erkennen.

[4] Hier gehen wir davon aus, dass keine ACKs oder Neuübertragungen notwendig sind. In Kapitel 4.3.4 wird darauf eingegangen, wie ACKs und Neuübertragungen behandelt werden.

4.3.3 Verkehrsüberwachung

In Kapitel 4.3.1.2 wurden existierende Ansätze vorgestellt, die geeignet sind, um Verkehrsüberwachung zur Erkennung der Auslastung des drahtlosen Mediums durchzuführen. Diese Ansätze sind entweder Hardware-spezifisch und nicht mit jedem WLAN-Adapter bzw. Treiber möglich, oder erfordern einen hohen Aufwand. Im Folgenden wird ein erstmals in [MKG17] publiziertes Verfahren vorgestellt, welches den Token Bucket überwacht, und daraus Schlussfolgerungen über die Auslastung des Mediums gewinnt. Das Verfahren hat den Vorteil, dass es unabhängig vom WLAN-Adapter bzw. dessen Treiber ist und einen geringen Aufwand erzeugt.

Der Token Bucket fließt über, wenn die zugewiesene Bandbreite $assignedBw_v$ nicht vollständig genutzt wird. Dafür kann es zwei Gründe geben:

1. Der Knoten hat nicht so viele Rahmen zu senden, wie Bandbreite zugewiesen wurde, d.h. $usedBw_v < assignedBw_v$. Dies tritt beispielsweise auf, wenn der Knoten nur sporadisch Daten zu senden hat und Bandbreite für den ungünstigsten Fall reserviert wurde. Ebenso ist denkbar, dass die benötigte Bandbreite schlicht konservativ bemessen wurde. Da zugewiesene Bandbreite ungenutzt bleibt, aber genutzt hätte werden können, sprechen wir in diesem Fall von nutzbarer verschwendeter Bandbreite (engl: *usable wasted bandwidth*). Die dabei überfließenden Tokens bezeichnen wir als *usable wasted tokens*. Dieses Verhalten ist gewünscht und kann bestenfalls dadurch geändert werden, dass die Anwendung ihre Reservierungen anpasst.
2. Das drahtlose Medium ist überlastet. Die dem Knoten zugewiesene Bandbreite ist in diesem Fall gar nicht (vollständig) verfügbar, d.h. $assignedBw_v > availableBw_v$. Wenn ein Rahmen gesendet werden soll, das Medium aber belegt ist, verbleibt dieser Rahmen für einige Zeit in der Warteschlange des Transceivers (bzw. dessen Treibers). Solange dieser Rahmen nicht vom Transceiver gesendet oder verworfen wurde, darf kein weiterer Rahmen von der Token Bucket Warteschlange an den Transceiver übergeben werden. Weitere eintreffende Rahmen sammeln sich daher in der Warteschlange des Token Bucket-Mechanismus und der Token Bucket füllt sich mit Tokens. Ist das Medium für lange Zeit oder in kurzer Zeit sehr häufig belegt, so führt dies dazu, dass Tokens überfließen, obwohl der Knoten Rahmen senden möchte, die diese Tokens hätten verbrauchen können. Wir sprechen in diesem Fall von nicht nutzbarer verschwendeter Bandbreite (engl: *unusable wasted bandwidth*) und nicht nutzbaren verschwendeten Tokens (engl: *unusable wasted tokens*), da Überlast auf dem Medium dafür verantwortlich ist, dass sie nicht genutzt werden können.

Zur Verkehrsüberwachung machen wir uns nun das Verhalten des Token Buckets im zweiten Fall zu nutze. Indem wir zählen, wie viele Tokens auf Grund von Überlastung des drahtlosen Mediums überlaufen, können wir Überlast auf dem drahtlosen Medium erkennen. Wir führen daher zwei Zähler ein, welche die überlaufenden Tokens zählen.

1. $usableWastedTokens_v(t)$ beschreibt die Anzahl Tokens, die beim Knoten v bis zum Zeitpunkt t übergelaufen sind, aber nutzbar gewesen wären. Dieser Zähler wird also jedes Mal erhöht, wenn ein neues Token erzeugt wird und folgende Bedingungen alle erfüllt sind:
   - Der Token Bucket ist voll.
   - Es befindet sich kein Rahmen in der Warteschlange des Token Buckets.
   - Es befindet sich kein Rahmen im Transceiver.

2. $unusableWastedTokens_v(t)$ beschreibt die Anzahl Tokens, die beim Knoten v bis zum Zeitpunkt t nicht nutzbar übergelaufen sind, weil das Medium überlastet war. Dieser Zähler wird also jedes Mal erhöht, wenn ein neues Token erzeugt wird und folgende Bedingungen alle erfüllt sind:
   - Der Token Bucket ist voll.
   - Es befindet sich ein Rahmen in der Warteschlange des Token Buckets oder im Transceiver.

Basierend auf dem zweiten Zähler bestimmen wir nun die *unusable wasted bandwidth*, also die wegen Überlast nicht nutzbare verschwendete Bandbreite, indem wir die Anzahl übergelaufener Tokens mit der Token-Größe multiplizieren:

$$unusableWastedBw_v(t) = unusableWastedTokens_v(t) \cdot tokenSize_v \tag{4.16}$$

Ebenso kann die *usable wasted bandwidth*, also die vom Knoten unbenutzt gelassene Bandbreite, die aber nutzbar gewesen wäre, berechnet werden:

$$usableWastedBw_v(t) = usableWastedTokens_v(t) \cdot tokenSize_v \tag{4.17}$$

Nun setzen wir die jeweilige Bandbreite in Relation zu der seit dem Beginn des betrachteten Zeitraums t_{start} bis zum Zeitpunkt t zugewiesenen Bandbreite:

$$unusableWastedBwRatio_v(t) = \frac{unusableWastedBw_v(t)}{assignedBw_v \cdot (t - t_{start})} \tag{4.18}$$

$$usableWastedBwRatio_v(t) = \frac{usableWastedBw_v(t)}{assignedBw_v \cdot (t - t_{start})} \tag{4.19}$$

Damit erhalten wir je eine Metrik aus dem Intervall $[0;1]$, welche den Anteil an (nicht) nutzbarer verschwendeter Bandbreite in Relation zur zugewiesenen Bandbreite angibt. Ist beispielsweise $unusableWastedBwRatio_v(t) = 100\%$, so bedeutet dies, dass die komplette zugewiesene Bandbreite auf Grund von Überlast nicht nutzbar gewesen ist. Ist dagegen $usableWastedBwRatio_v(t) = 100\%$, so bedeutet dies, dass die zugewiesene Bandbreite komplett ungenutzt geblieben ist, weil der Knoten keine Rahmen zu senden hatte.

Es ist offensichtlich, dass Bandbreite entweder auf die eine oder andere Weise verschwendet sein kann, oder aber genutzt. Das bedeutet:

$$wastedBwRatio_v(t) = unusableWastedBwRatio_v(t) + usableWastedBwRatio_v(t) \tag{4.20}$$

$$usedBwRatio_v(t) = 1 - wastedBwRatio_v(t) \tag{4.21}$$

Wir können also über das Verfahren auch $usedBwRatio_v(t)$ einfach bestimmen, d.h. welchen Anteil der zugewiesenen Bandbreite der Knoten v bis zum Zeitpunkt t genutzt hat.

Anders ließe sich dies ebenso bestimmen, indem man zunächst alle vom Knoten v bis zum Zeitpunkt t gesendeten Rahmen $Frames_v(t)$ betrachtet, und deren Sendezeit aufsummiert:

$$txTime_v(t) = \sum_{\forall f \in Frames_v(t)} \left(\frac{payload_f}{rate_f} + \frac{MLO}{rate_{Header}} + PLO \right) \tag{4.22}$$

Die Berechnung der Sendezeit eines Rahmens f ist dabei analog zu Formel (4.7). Hierbei sei $payload_f$ die Rahmengröße des Rahmens f und $rate_f$ die Datenrate, mit der die Payload dieses

Rahmens gesendet wird. PLO und MLO beschreiben wie zuvor den Overhead auf Physical bzw. MAC Layer und $rate_{Header}$ die Datenrate, mit welcher der IEEE 802.11 Header gesendet wird.

Setzt man diese Zeit in Relation zur zugewiesenen Bandbreite, erhalten wir ebenso den Anteil verwendeter Bandbreite $usedBwRatio_v(t)$:

$$usedBwRatio_v(t) = \frac{txTime_v(t)}{assignedBw_v \cdot (t - t_{start})} \tag{4.23}$$

Um die vorgestellte Metrik zur Erkennung von Überlast auf dem Medium anschließend mit den in Kapitel 4.3.1.2 vorgestellten Alternativen, Channel Busy Fraction sowie Medium Utilization Fraction, zu vergleichen, formalisieren wir diese ebenso. Sei $channelBusyTime_v(t)$ die aus dem Treiber des WLAN-Adapters ausgelesene Channel Busy Time des Knoten v seit t_{start} bis zum Zeitpunkt t. Dann ist:

$$channelBusyFraction_v(t) = \frac{channelBusyTime_v(t)}{t - t_{start}} \tag{4.24}$$

Die vom Knoten v bestimmte Channel Busy Time $channelBusyTime_v(t)$ enthält die Zeit, während der andere Knoten das Medium durch Senden belegt haben, aber auch die Zeit, während der v selbst gesendet hat [Lin]. Wir definieren daher noch das alternative Maß *Medium Energy Sensed* (MES), welche nur die Zeit betrachtet, während der das Medium durch andere Knoten belegt war:

$$mediumEnergySensed_v(t) = channelBusyTime_v(t) - txTime_v(t) \tag{4.25}$$

Auch dies setzen wir in Relation zur vergangenen Zeit:

$$mediumEnergySensedFraction_v(t) = \frac{mediumEnergySensed_v(t)}{t - t_{start}} \tag{4.26}$$

Zur Bestimmung der Medium Utilization Fraction muss zunächst die Medium Utilization Time bestimmt werden. Diese ist analog zu Formel (4.22) die Summe der Übertragungszeiten aller vom Knoten v vom Start bis zum Zeitpunkt t empfangenen Rahmen $Received_v(t)$:

$$mediumUtilizationTime_v(t) = \sum_{\forall f \in Received_v(t)} \left(\frac{payload_f}{rate_f} + \frac{MLO}{rate_{Header}} + PLO \right) \tag{4.27}$$

Um die Medium Utilization Fraction zu erhalten, setzen wir diese Zeit schließlich wieder in Relation zur vergangenen Zeit:

$$mediumUtilizationFraction_v(t) = \frac{mediumUtilizationTime_v(t)}{t - t_{start}} \tag{4.28}$$

Wir haben nun mehrere Metriken definiert, die dazu geeignet erscheinen, Überlastsituationen des drahtlosen Mediums zu quantifizieren:

- Channel Busy Fraction
- Medium Energy Sensed Fraction
- Medium Utilization Fraction
- Unusable Wasted Bandwidth Ratio

Die ersten drei sind bereits bekannte Metriken. Diese haben den Nachteil, dass sie entweder auf nicht weit verbreitete Hardware-Unterstützung angewiesen sind oder aber mit dem hohen Aufwand verbunden sind, alle auf dem Medium von anderen Knoten gesendeten Rahmen zu verarbeiten. Das Unusable Wasted Bandwidth Ratio hingegen ist nicht auf Hardware-Unterstützung angewiesen und erfordert nicht, die Rahmen der anderen Knoten zu verarbeiten. Wird das Time Token Bucket-Verfahren ohnehin, wie in Kapitel 4.3.2.2 vorgeschlagen, zur Verkehrsformung eingesetzt, so erzeugt die Bestimmung des Unusable Wasted Bandwidth Ratio kaum zusätzlichen Aufwand, da lediglich die überlaufenden Tokens gezählt werden müssen und sich die Metrik damit direkt aus Formel (4.17) und Formel (4.18) ergibt. Im Folgenden werden die Metriken gegeneinander evaluiert und die dazu eingesetzte Implementierung kurz beschrieben.

4.3.4 Implementierung

Das beschriebene Verfahren zur Verkehrsformung und -überwachung basierend auf dem Time Token Bucket-Verfahren wurde mit dem WiPS Framework (s. Kapitel 4.1) als Schicht zur Verkehrskontrolle implementiert. Diese Schicht bekommt Rahmen von darüber liegenden Schichten bzw. der Anwendung übergeben und verzögert sie ggf., ehe sie die Rahmen an die darunter liegende Schicht weitergibt. Da die Experimente ausschließlich die hier vorgestellten Verfahren zur Verkehrskontrolle testen sollen, kam in WiPS neben der Schicht zur Verkehrskontrolle nur die WiPS MAC-Schicht zum Einsatz. Abb. 4.8 zeigt die sich daraus ergebende Architektur, in der anders als in Abb. 4.1 die Schicht zur Topologieerkennung, zum Routing und die Middleware nicht enthalten sind. Nichtsdestotrotz kann die implementierte Schicht zur Verkehrskontrolle in WiPS grundsätzlich problemlos mit beliebigen anderen Schichten kombiniert werden. Zu beachten ist selbstverständlich, dass die Schicht nur auf Rahmen wirkt, die durch sie hindurch laufen. Die Verkehrskontrolle-Schicht kann also nur Rahmen verzögern, die von Schichten erzeugt werden, die über ihr platziert sind. Aus diesem Grund sollte die Schicht zur Verkehrskontrolle in der Regel möglichst direkt über der WiPS MAC-Schicht platziert werden.

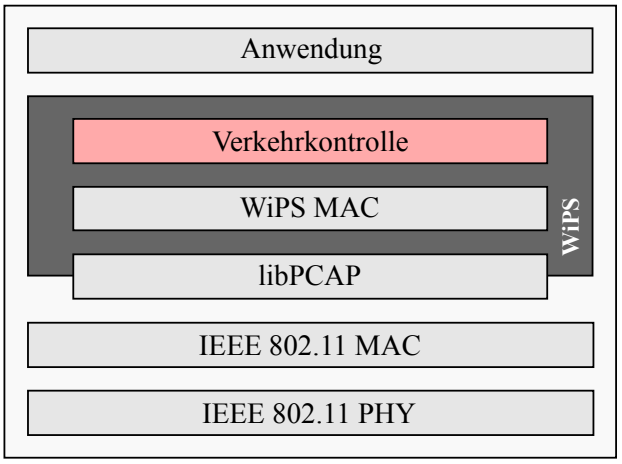


Abbildung 4.8: Architektur der Implementierung zum Testen der Vekehrskontrolle im WiPS Framework

Die WiPS MAC-Schicht stellt in diesem Zusammenhang eine Besonderheit dar. Sie erzeugt ausschließlich zwei Arten von Rahmen: Acknowledgements (ACKs) und Neuübertragungen. ACKs sind Quittungen für von anderen Knoten empfangene Rahmen. Hier wurde festgelegt, dass ein Knoten, der einen Rahmen mit ACK-Anforderung sendet, die zum Senden des ACKs benötigte Bandbreite bereits beim Senden des ursprünglichen Rahmens mit seinen eigenen Tokens be-

gleichen muss. Dieses Vorgehen hat mehrere Vorteile: Da die Bandbreite von ACKs bereits mit Tokens beglichen wurde, ist es nicht nötig, dass der Empfänger der ursprünglichen Nachricht ggf. erst Tokens sammeln muss, bevor er das ACK senden kann. Das bedeutet, dass ACKs immer sofort gesendet werden können und nicht verzögert werden müssen. Daher ist es auch kein Problem, dass ACKs nicht durch die Schicht zur Verkehrskontrolle laufen. Darüber hinaus kann der Empfänger nicht abschätzen, wie viele ACKs er zu Rahmen senden muss, die er von anderen Knoten gesendet bekommen hat, und daher den Bandbreitenbedarf dafür nicht abschätzen. Der ursprüngliche Sender der Nachricht kann hingegen wissen, wie viele Nachrichten er schickt, die ein ACK erfordern, und dies bei seiner Bandbreitenreservierung berücksichtigen.

Neuübertragungen werden von der WiPS MAC-Schicht dann ausgelöst, wenn das ACK zu einer gesendeten Nachricht ausbleibt, die Nachricht also potentiell verloren gegangen ist. Dass eine Neuübertragung erforderlich wird, lässt sich aber beim Senden der Nachricht noch nicht absehen, und auch nicht, wie viele Neuübertragungen nötig sein werden. Lediglich die maximale Anzahl möglicher Neuübertragungen ist bekannt. Daher ist es nicht möglich, dass, ähnlich wie bei ACKs, beim Senden der Nachricht die Bandbreite für die Neuübertragungen bereits mit Tokens beglichen wird. Es wurde allerdings entschieden, dass die Bandbreite für Neuübertragungen auch mit Tokens beglichen werden muss, und Neuübertragungen auch ggf. verzögert werden müssen, bis die erforderlichen Tokens verfügbar sind. Andernfalls würden im Falle von Neuübertragungen längere Bursts als vorgesehen entstehen. Außerdem sind Neuübertragungen i.d.R. dann notwendig, wenn das Medium ausgelastet ist, weil mehrere Knoten gleichzeitig senden. Würde man genau in diesem Fall erlauben, dass die Knoten mehr Bandbreite nutzen dürfen, als ihnen zugewiesen wurde, würde dies die Überlastsituation noch verstärken. Da Neuübertragungen aber in der WiPS MAC-Schicht ausgelöst werden, laufen sie normalerweise nicht durch die (darüber platzierte) Schicht der Verkehrskontrolle. Um Neuübertragungen dennoch verzögern zu können, wurde eine spezielle Schnittstelle in der WiPS MAC-Schicht geschaffen. Hier kann sich jede WiPS-Schicht registrieren, die über Neuübertragungen informiert werden und diese ggf. verhindern möchte. Die WiPS MAC-Schicht ruft bei jeder Neuübertragung nacheinander alle hier registrierten Schichten auf und bittet um die Erlaubnis, die ausstehende Neuübertragung zu senden. Hier registriert sich die Schicht zur Verkehrskontrolle und prüft in diesem Fall, ob für die Neuübertragung genügend Tokens vorhanden sind. Ist dies der Fall, entfernt sie die entsprechende Anzahl Tokens aus dem Bucket und gestattet der WiPS MAC-Schicht die Neuübertragung. Ist dies nicht der Fall, so wird die Neuübertragung in die Warteschlange der Verkehrskontrolle eingereiht und der WiPS MAC-Schicht die Neuübertragung untersagt. Damit ist die Schicht der Verkehrskontrolle dafür zuständig, die Neuübertragung ggf. wieder zur Übertragung an die WiPS MAC-Schicht zu übergeben. Dies tut sie, sobald genügend Tokens dafür im Bucket sind. In der Warteschlange werden Neuübertragungen so eingeordnet, dass sie vor normalen Übertragungen ausgeführt werden.

Der genannte Eingriff in die Neuübertragungen ist nur möglich, indem diese Funktionalität von der WiPS MAC-Schicht durchgeführt und nicht das Standard-Verhalten der IEEE 802.11 MAC-Schicht genutzt wird. Außerdem ist es wichtig, dass es in WiPS mit Hilfe der radiotap-Header möglich ist, für jeden Rahmen anzugeben, mit welcher Datenrate der Rahmen gesendet werden soll. Nur dadurch, dass die Datenrate bekannt ist, ist es möglich, die zum Senden eines Rahmens benötigte Bandbreite berechnen zu können, und so die benötige Anzahl Tokens zu bestimmen.

4.3.5 Evaluierung

Das vorgestellte Verfahren zur Verkehrsformung und Verkehrsüberwachung wurde in WiPS implementiert, um es in realen Experimenten ausführlich zu testen. Die wichtigsten Ergebnisse dieser Experimente werden in diesem Kapitel vorgestellt. Das Ziel war dabei in erster Linie zu untersuchen, ob die Metrik der Unusable Wasted Bandwidth Ratio tatsächlich geeignet ist, die Auslastung des drahtlosen Mediums abzuschätzen, und sie mit den anderen angesprochenen Metriken zu vergleichen.

Für die in der Evaluierung durchgeführten Experimente wurden zwei Testbetts aus Raspberry Pi Knoten genutzt (s. Kapitel 4.1.2). Das *stationäre Testbett* ist dabei in mehreren Räumen eines Universitätsgebäudes installiert. In diesem Umfeld sind eine Vielzahl von IEEE 802.11 Netzwerken aktiv und daher eine übliche Aktivität externer Knoten auf dem Medium zu erwarten. Das *mobile Testbett* wurde für die hier durchgeführten Experimente in das Kellergeschoss des Universitätsgebäudes gebracht, wo kein externer Verkehr durch IEEE 802.11 Knoten einen Einfluss auf die Experimente nimmt. Die Testbetts bestehen aus jeweils sechs Raspberry Pi Knoten, die mit Arch Linux und Linux Kernel 4.6.5-v7+ laufen. Als WLAN-Adapter kamen in diesen Experimenten pro Knoten ein Logilink WL0084B USB-Adapter zum Einsatz (s. Kapitel 4.1.2). Beide Testbetts sind Singlehop-Netzwerke, d.h. jeder Knoten kann direkt mit jedem anderen Knoten kommunizieren. Im Fall des mobilen Testbetts sind alle Knoten recht nah beieinander platziert, wohingegen im stationären Testbett die Knoten über mehrere Räume verteilt sind.

Zum Testen wurde eine Testanwendung geschrieben, welche nach Bedarf Broadcast-Rahmen erzeugt. In allen hier präsentierten Experimenten erzeugt diese Rahmen, die eine Payload von 480 Bytes transportieren. Die Anwendung erzeugt so viele dieser Rahmen, dass immer mindestens ein Rahmen in der Warteschlange des Token Bucket-Mechanismus vorhanden ist, der darauf wartet, dass genügend Tokens in den Bucket fallen, sodass er gesendet werden kann. Dies bedeutet, dass die Bandbreite, die der Knoten nutzen möchte, über der ihm zugewiesenen Bandbreite liegt, also $usedBw_v \geq assignedBw_v$ gilt. Falls das Medium das Senden der zugewiesenen Bandbreite zulässt, werden die in den Bucket fallenden Buckets stets genutzt. Dies bedeutet, dass in den Experimenten keine *usable wasted tokens* vorkommen können. Laufen dennoch Tokens über, so geschieht dies, weil das Medium überlastet ist, in diesem Fall handelt es sich um *unusable wasted tokens*. Die den Knoten zugewiesenen Bandbreiten $assignedBw_v$ wurden in den unterschiedlichen Experimenten unterschiedlich konfiguriert. Die für alle Experimente gültigen Parameter fasst Tab. 4.1 zusammen.

Experiment 1: Zunächst soll die Sensitivität der Metriken untersucht werden. Es ist als wünschenswert anzusehen, dass eine Metrik sensitiv genug ist, dass sie hohe Last auf dem Medium bereits erkennt, bevor das Medium überlastet ist und es dadurch zu Rahmenverlusten, Neuübertragungen und langen Verzögerungen kommt. Um dies untersuchen zu können, wurde den Knoten in diesem Experiment zunächst mit insgesamt 6 % nur wenig Bandbreite zugewiesen, sodass es nicht zu einer Überlastsituation kommt. Darüber hinaus wurde dieses Experiment auf dem mobilen Testbett ausgeführt, d.h. ohne Einflüsse durch externe Knoten. Es ist zu erwarten, dass unter der geringen Auslastung des Mediums keine oder nur sehr wenig Tokens überlaufen und die anderen Metriken ungefähr die Auslastung von 6 % messen. Die Parameter des Experiments fasst Tab. 4.2 zusammen.

Die Ergebnisse des Experiments fasst Tab. 4.3 zusammen. Wie erwartet traten keine *unusable wasted tokens* auf. Daraus ergibt sich, dass $unusableWastedBwRatio_v(t)$ auch 0 ist. Offensichtlich ist

Tabelle 4.1: Parameter aller Experimente

Parameter	Wert
Anzahl Knoten	6
Datenrate	1 MBit/s
Sendestärke	16 dBm
Frequenzbereich	2,4 GHz
Payload	480 Bytes
$txTime_{v,Max}(\forall v \in V)$	4545 μs
$tokenSize_v(\forall v \in V)$	1 μs
$bucketSize_v(\forall v \in V)$	4545 *Tokens*

Tabelle 4.2: Parameter zu Experiment 1

Parameter	Wert
Testbett	mobil (kein externer Verkehr)
Dauer	55 Minuten
$assignedBw_v(\forall v \in V)$	1 %
$assignedBw_G$	6 %
$refillInt_v(\forall v \in V)$	100 μs

die zugewiesene Bandbreite von insgesamt 6 % problemlos nutzbar, wenn kein externer Verkehr auf dem Medium auftritt. Das bedeutet, dass, wie zu erwarten, bei allen Knoten die verfügbare Bandbreite über der zugewiesenen Bandbreite liegt, also $\forall v \in V : availableBw_v \geq assignedBw_v$. Die $channelBusyFraction_v(t)$ und $mediumEnergySensedFraction_v(t)$ bewegen sich um die 6 %, die durch die zugewiesene Bandbreite zu erwarten wären. Wir sehen, dass die Metrik *Unusable Wasted Bandwidth Ratio* im Gegensatz zu den anderen Metriken nicht geeignet ist, um sehr geringe Auslastung des Mediums zu quantifizieren.

Tabelle 4.3: Ergebnisse von Experiment 1 (mobiles Testbett, $assignedBw_G = 6$ %)

Knoten	38	39	40	41	42	43
$unusableWastedTokens_v(t)$	0	0	0	0	0	0
$unusableWastedBwRatio_v(t)$	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %
$channelBusyFraction_v(t)$	5,70 %	7,80 %	8,14 %	5,80 %	5,74 %	6,36 %
$mediumEnergySensedFraction_v(t)$	4,79 %	6,88 %	7,21 %	4,87 %	4,81 %	5,44 %

Experiment 2: Im zweiten Experiment wird die gleiche Konfiguration wie beim ersten Experiment auf dem stationären Testbett, also unter Einfluss externen Verkehrs, durchgeführt. Der externe Verkehr sollte dazu führen, dass $channelBusyFraction_v(t)$ und *mediumEnergySensed-*

$Fraction_v(t)$ deutlich über 6 % liegen und ggf. eine geringe Menge an Tokens überlaufen. Die Parameter des Experiments fasst Tab. 4.4 zusammen.

Tabelle 4.4: Parameter zu Experiment 2

Parameter	Wert
Testbett	stationär (mit externem Verkehr)
Dauer	55 Minuten
$assignedBw_v(\forall v \in V)$	1 %
$assignedBw_G$	6 %
$refillInt_v(\forall v \in V)$	100 μs

Tabelle 4.5:]
Ergebnisse von Experiment 2 (stationäres Testbett, $assignedBw_G = 6$ %)

Knoten	7	8	22	23	28	29
$unusableWastedTokens_v(t)$	293.358	232.686	13.686	249.952	42.648	11.152
$unusableWastedBwRatio_v(t)$	0,89 %	0,71 %	0,04 %	0,76 %	0,13 %	0,03 %
$channelBusyFraction_v(t)$	24,80 %	19,87 %	16,90 %	24,57 %	17,90 %	14,32 %
$mediumEnergySensedFraction_v(t)$	23,83 %	18,89 %	15,91 %	23,59 %	16,92 %	13,32 %

Die Ergebnisse des Experiments fasst Tab. 4.5 zusammen. In diesem Fall treten *unusable wasted tokens* auf, bei Knoten 7 sind beispielsweise 293.358 Tokens übergelaufen. Bei der gewählten Token-Größe von 1 μs bedeutet dies, dass 293.358 μs der zugewiesenen Bandbreite nicht genutzt werden konnten ($unusableWastedTokens_v(t) \cdot tokenSize_v$). Das Experiment lief 55 Minuten lang, daher war jedem Knoten mit 1 % Bandbreite 1 % von 55 Minuten, also 33.000.000 μs, Sendezeit zugewiesen. Knoten 7 konnte von dieser zugewiesenen Sendezeit 293.358 μs nicht nutzen. Somit ergibt sich die $unusableWastedBwRatio_v(t) = \frac{293.358\ \mu s}{33.000.000\ \mu s} = 0,89$ % wie in Tab. 4.5 eingetragen.

Mit Werten um 20 % liegen die Werte der $channelBusyFraction_v(t)$ und $mediumEnergySensedFraction_v(t)$ deutlich über den 6 %, die den internen Knoten zugewiesen worden sind. Die Differenz entspricht ungefähr der von externen Knoten genutzten Bandbreite. Auffällig ist, dass die Knoten 7, 8 und 23 bei beiden Metriken eine deutlich höhere Auslastung des Mediums wahrnehmen als die Knoten 22, 28 und 29. Das gleiche lässt sich bei den $unusableWastedTokens_v(t)$ und der sich daraus resultierenden $unusableWastedBwRatio_v(t)$ beobachten. Offensichtlich gab es in der Umgebung dieser Knoten mehr externen Verkehr, wodurch die Knoten mit allen Metriken eine höhere Medienauslastung wahrgenommen haben als die anderen Knoten. Obwohl die Mediumauslastung insgesamt hier überall unter 25 % liegt und damit immer noch recht gering ist, zeigt sich, dass die Metrik der *Unusable Wasted Bandwidth Ratio* bereits anschlägt. Da es sich hier noch nicht um eine Überlastsituation handelt, ist die Metrik sensitiv genug.

Experiment 3: Wir wiederholen das vorige Experiment und reduzieren dabei die zugewiesene Bandbreite pro Knoten auf $assignedBw_v = 0,5$ %, sodass die zugewiesene Bandbreite mit den 6 Knoten des stationären Testbetts insgesamt $assignedBw_G = 3$ % beträgt. Untersucht werden soll

dabei, ob die Metrik der *Unusable Wasted Bandwidth Ratio* noch sensitiver ist, d.h. ob und bei welchen Knoten noch Tokens überlaufen. Die Parameter des Experiments fasst Tab. 4.6 zusammen.

Tabelle 4.6: Parameter zu Experiment 3

Parameter	Wert
Testbett	stationär (mit externem Verkehr)
Dauer	55 Minuten
$assignedBw_v(\forall v \in V)$	0,5 %
$assignedBw_G$	3 %
$refillInt_v(\forall v \in V)$	200 μs

Tabelle 4.7: Ergebnisse von Experiment 3 (stationäres Testbett, $assignedBw_G = 3$ %)

Knoten	7	8	22	23	28	29
$unusableWastedTokens_v(t)$	8.164	4.726	0	8.527	0	0
$unusableWastedBwRatio_v(t)$	0,02 %	0,01 %	0,00 %	0,03 %	0,00 %	0,00 %
$channelBusyFraction_v(t)$	14,80 %	10,25 %	11,86 %	13,73 %	12,74 %	8,39 %
$mediumEnergySensedFraction_v(t)$	14,31 %	9,76 %	11,37 %	13,24 %	12,24 %	7,89 %

Die Ergebnisse des Experiments fasst Tab. 4.7 zusammen. Die Knoten 7, 8 und 23, bei denen zuvor mehr Tokens übergelaufen sind als bei den anderen, sind nun die einzigen Knoten, bei denen Tokens überlaufen. Bei den Knoten 22, 28 und 29 laufen keinerlei Tokens über, d.h. sie können ihre zugewiesene Bandbreite vollständig nutzen. An der $channelBusyFraction_v(t)$ und $mediumEnergySensedFraction_v(t)$ lässt sich hingegen diesmal nicht eindeutig erkennen, welche Knoten stärker durch externen Verkehr gestört werden als andere: So ist die $channelBusyFraction_v(t)$ von Knoten 8 niedriger als von Knoten 22, obwohl bei Knoten 8 Tokens überlaufen und bei Knoten 22 nicht. Hier zeigt sich, dass die $unusableWastedBwRatio_v(t)$ eine klarere Metrik ist, wenn es darum geht, festzustellen, ob ein Knoten durch andere Knoten gestört wird. Schließlich bedeutet durch den Transceiver festgestellte Energie auf dem Medium nicht zwangsläufig, dass der Knoten beim Senden gestört wird. Hat hingegen die $unusableWastedBwRatio_v(t)$ kleine Werte über 0, so bedeutet dies bereits, dass Rahmen vermehrt verzögert gesendet werden, da das Medium zum Sendezeitpunkt belegt war. Auch wenn dies noch keine kritische Überlastsituation darstellt, ist es dennoch ein Punkt, bei dem man sinnvollerweise aufhören sollte, mehr Bandbreite zuzuweisen, da sonst schnell eine kritische Überlastsituation auftreten könnte. Ein Problem der $channelBusyFraction_v(t)$ bzw. $mediumEnergySensedFraction_v(t)$ ist auch, dass nicht klar ist, ab welchem Grenzwert ein Knoten gestört wird.

Vergleichen wir die $channelBusyFraction_v(t)$ in Tab. 4.7 und Tab. 4.5, so fällt auf, dass der Kanal im vorigen Experiment ca. 10 % mehr belegt war, obwohl nur 3 % mehr Verkehr den internen Knoten zugewiesen wurde. Offenbar beeinflusst der vermehrte interne Verkehr den externen Verkehr der anderen Access Points auf diesem Kanal. Ein Grund dafür könnte sein, dass der interne Verkehr zu Rahmenverlust und Neuübertragungen bei den externen Knoten und so zu einer höheren Auslastung des Mediums führt. Dies macht deutlich, dass es nicht sinnvoll ist, den internen Knoten einfach so viel zusätzliche Bandbreite zuzuweisen, wie aktuell ungenutzt

ist, da der zusätzliche interne Verkehr auch schwer vorherzusagenden zusätzlichen externen Verkehr erzeugt.

Experiment 4: In diesem Experiment soll nun herausgefunden werden, wie sich die Metrik der *Unusable Wasted Bandwidth Ratio* verhält, wenn sehr viel Bandbreite genutzt wird, bis hin zu einer Situation, in der den internen Knoten fast die komplette Bandbreite zugewiesen ist. Es wurden daher mehrere Teilexperimente gemacht, bei denen zwischen 30 % und 90 % der Bandbreite zugewiesen wurde, wobei die Bandbreite wie zuvor zwischen 6 Knoten gleichmäßig aufgeteilt wird. Einem einzelnen Knoten werden also zwischen 5 % und 15 % Bandbreite zugewiesen werden. Mit jeder Bandbreitenkonfiguration läuft das Experiment 20 Minuten. In diesem Experiment kommt das stationäre Testbett auf dem 2,4 GHz Kanal 1 zum Einsatz. Die Parameter des Experiments fasst Tab. 4.8 zusammen.

Tabelle 4.8: Parameter zu Experiment 4

Parameter	Wert
Testbett	stationär (mit externem Verkehr)
Dauer	6 mal 20 Minuten
Kanal	1
$assignedBw_v(\forall v \in V)$	$\in \{5\ \%; 9\ \%; 11\ \%; 13\ \%; 14\ \%; 15\ \%\}$
$assignedBw_G$	$\in \{30\ \%; 54\ \%; 66\ \%; 78\ \%; 84\ \%; 90\ \%\}$
$refillInt_v(\forall v \in V)$	$\in \{20{,}0; 11{,}1; 9{,}1; 7{,}7; 7{,}1; 6{,}7\}\ \mu s$

Die Ergebnisse des Experiments fasst Abb. 4.9 zusammen. Bis zu einer zugewiesenen Gesamtbandbreite von 54 % bewegt sich die nicht nutzbare Bandbreite im niedrigen einstelligen Prozentbereich. Bei 66 % Gesamtbandbreite kommt es bei Knoten 7 und 28 dazu, dass mehr als 10 % der zugewiesenen Bandbreite nicht nutzbar sind. Bei höheren Bandbreitenprofilen sind bei Knoten 7 sogar ca. 75 % der zugewiesenen Bandbreite nicht nutzbar, und bei Knoten 22 treten ebenfalls erhebliche Probleme auf. Es zeigt sich, dass die Knoten 7, 22 und 28 am meisten der ihnen zugewiesenen Bandbreite nicht nutzen können, wohingegen die Knoten 8, 23 und 29 die ihnen zugewiesene Bandbreite fast vollständig nutzen können. Das Experiment wurde auf dem stationären Testbett während der Arbeitszeiten ausgeführt, sodass der externe Verkehr vermutlich aus typischem Internet-Verkehr besteht. Beispielsweise steht der Access Point unserer Arbeitsgruppe in der Nähe von Knoten 7 und läuft ebenfalls auf Kanal 1. Es ist also nicht verwunderlich, dass Knoten 7 am stärksten durch externen Verkehr beeinflusst wird. Die Knoten 22 und 28 werden möglicherweise durch andere Access Points oder Clients, die mit den Acces Points kommunizieren, gestört. Das Experiment wurde mehrfach wiederholt und erzielte dabei vergleichbare Ergebnisse.

Das Experiment zeigt, dass es nicht ausreicht, nur zu betrachten, wie viel der zugewiesenen Bandbreite ein Knoten nicht nutzen kann. Denn es kann durchaus sein, dass einer seiner Nachbarknoten, mit denen der Knoten zu kommunizieren versucht, durch die hohe Mediumauslastung deutlich stärker gestört wird und daher nicht zuverlässig antworten kann. Um die nutzbare Bandbreite zu maximieren ohne dabei einige Knoten so stark zu stören, dass sie nicht mehr zuverlässig kommunizieren können, erscheint daher ein kooperativer Ansatz notwendig. Ein solcher wird in Kapitel 4.4 entwickelt.

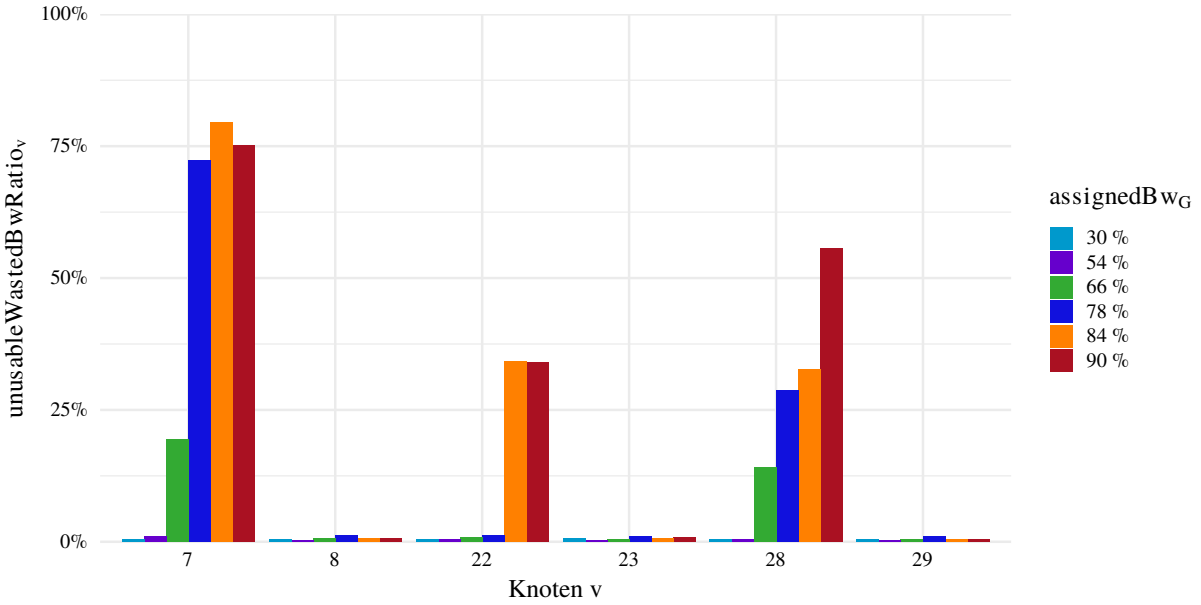


Abbildung 4.9: Ergebnisse zu Experiment 4: Unusable Wasted Bandwidth Ratio je Knoten v bei unterschiedlicher zugewiesener Gesamtbandbreite $assignedBw_G$ unter Einsatz des stationären Testbetts auf Kanal 1 mit externem Verkehr

Experiment 5: Im vorigen Experiment wurde gefolgert, dass einige Knoten durch externen Verkehr stärker beeinflusst werden als andere. Auf einem anderen (nicht überlappenden) Kanal müsste entsprechend anderer externer Verkehr vorherrschen und das Ergebnis anders ausfallen. Um dies zu überprüfen, wurde das gleiche Experiment auf Kanal 6 wiederholt. Die übrigen Parameter sind identisch mit dem vorigen Experiment und in Tab. 4.9 zusammengefasst.

Tabelle 4.9: Parameter zu Experiment 5

Parameter	Wert
Testbett	stationär (mit externem Verkehr)
Dauer	6 mal 20 Minuten
Kanal	6
$assignedBw_v(\forall v \in V)$	$\in \{5\ \%; 9\ \%; 11\ \%; 13\ \%; 14\ \%; 15\ \%\}$
$assignedBw_G$	$\in \{30\ \%; 54\ \%; 66\ \%; 78\ \%; 84\ \%; 90\ \%\}$
$refillInt_v(\forall v \in V)$	$\in \{20{,}0; 11{,}1; 9{,}1; 7{,}7; 7{,}1; 6{,}7\}\ \mu s$

Die Ergebnisse des Experiments fasst Abb. 4.10 zusammen. Diesmal sind die Knoten 8 und 22 am stärksten betroffen, die übrigen Knoten kaum bzw. erst bei hohen zugewiesenen Bandbreiten. Das Wechseln zu einem anderen Kanal führt hier dazu, dass bei Knoten 7 fast keine zugewiesene Bandbreite mehr verloren geht, wohingegen Knoten 8, der zuvor keine Probleme hatte, nun am stärksten betroffen ist. Wie zuvor zeigt sich also, dass manche Knoten stärker durch externen Verkehr beeinflusst werden als andere. Die Tatsache, dass dies sich auf unterschiedlichen Kanälen unterschiedlich verhält, könnte ein Kommunikationssystem nutzen, z.B. indem mehrere Kanäle parallel verwendet oder die Kanäle gewechselt werden.

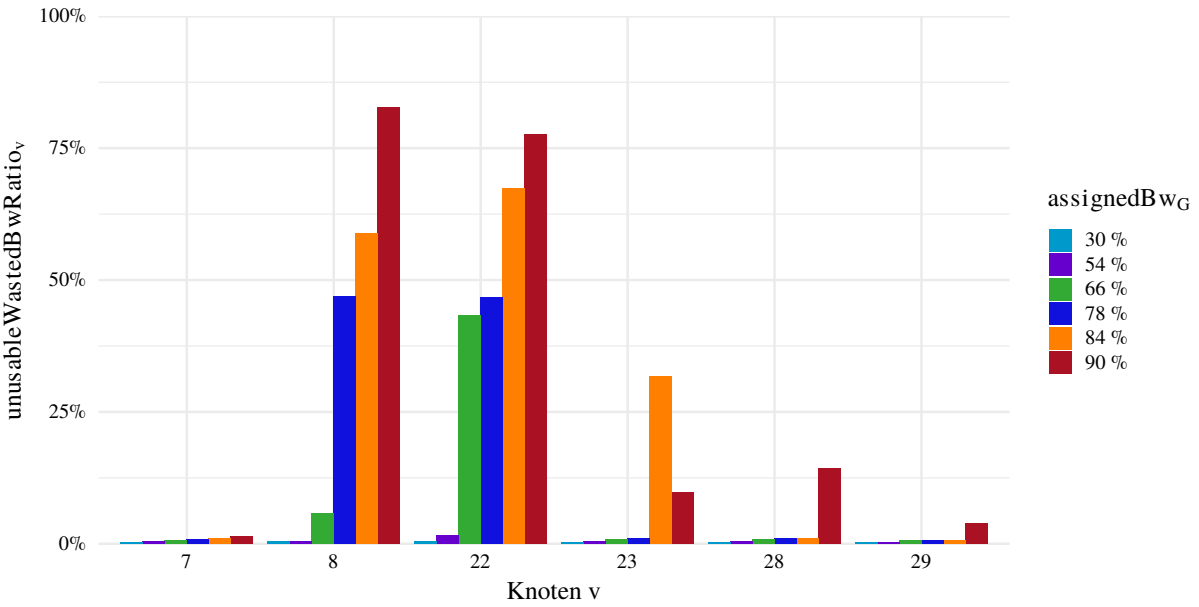


Abbildung 4.10: Ergebnisse zu Experiment 5: Unusable Wasted Bandwidth Ratio je Knoten v bei unterschiedlicher zugewiesener Gesamtbandbreite $assignedBw_G$ unter Einsatz des stationären Testbetts auf Kanal 6 mit externem Verkehr

Experiment 6: Bei den beiden vorigen Experimenten unter Einfluss von externem Verkehr zeigte sich, dass einige Knoten früher die ihnen zugewiesene Bandbreite nicht mehr nutzen können als andere. Dies wurde auf den Einfluss von externem Verkehr zurückgeführt. In diesem Experiment soll das vorige Experiment daher noch einmal auf dem mobilen Testbett, also ohne Einfluss externen Verkehrs, durchgeführt werden. Hierbei wird erwartet, dass die Knoten in etwa gleich betroffen sind. Die Parameter des Experiments fasst Tab. 4.10 zusammen.

Tabelle 4.10: Parameter zu Experiment 6

Parameter	Wert
Testbett	mobil (ohne externen Verkehr)
Dauer	6 mal 20 Minuten
Kanal	6
$assignedBw_v(\forall v \in V)$	$\in \{5\ \%; 9\ \%; 11\ \%; 13\ \%; 15\ \%; 16\ \%\}$
$assignedBw_G$	$\in \{30\ \%; 54\ \%; 66\ \%; 78\ \%; 90\ \%; 96\ \%\}$
$refillInt_v(\forall v \in V)$	$\in \{20{,}0; 11{,}1; 9{,}1; 7{,}7; 6{,}7; 6{,}3\}\ \mu s$

Die Ergebnisse des Experiments fasst Abb. 4.11 zusammen. Wie erwartet ist es nicht mehr eindeutig auszumachen, welche Knoten stärker betroffen sind. Die Verluste sind zwar auch nicht gleichverteilt, wirken allerdings zufälliger als zuvor. Beispielsweise zeigt Knoten 43 bei 66 % und 96 % zugewiesener Gesamtbandbreite hohe Verluste, nicht aber bei den Bandbreitenprofilen dazwischen. Diese eher zufällige Verteilung könnte durch den random backoff-Mechanismus der IEEE 802.11 MAC-Schicht entstehen.

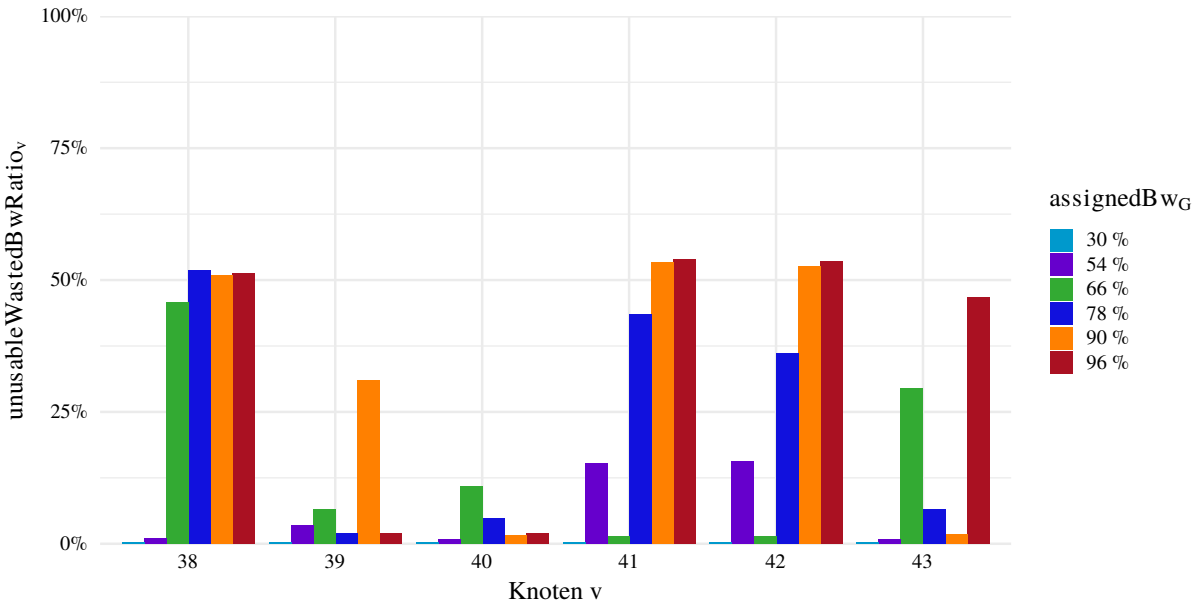


Abbildung 4.11: Ergebnisse zu Experiment 6: Das Unusable Wasted Bandwidth Ratio je Knoten v bei unterschiedlicher zugewiesener Gesamtbandbreite $assignedBw_G$ unter Einsatz des mobilen Testbetts ohne externen Verkehr

4.3.6 Zusammenfassung

Um die Zuverlässigkeit von wettbewerbsbasierten Kommunikationssystemen zu verbessern, ist es notwendig, den Verkehr stärker zu kontrollieren, als dies bei IEEE 802.11 standardmäßig der Fall ist. In diesem Kapitel wurden dazu geeignete Verfahren zur Verkehrsformung, insbesondere das Time Token Bucket-Verfahren, sowie zur Verkehrsüberwachung, insbesondere zur Bewertung der Mediumauslastung, betrachtet. Basierend auf dem Time Token Bucket-Verfahren wurde mit der *Unusable Wasted Bandwidth Ratio* eine Metrik entwickelt, um die Auslastung des Mediums abzuschätzen. Experimente haben gezeigt, dass diese Metrik zwar nicht geeignet ist, um geringe Mediumauslastungen zu quantifizieren, allerdings geeignet ist, um die Schwelle zu erkennen, an der das Medium droht, in eine Überlastsituation überzugehen. Zu diesem Zweck ist die Metrik klassischen Metriken wie der Channel Busy Fraction überlegen, da bei diesen unklar ist, ab welchem Grenzwert mit Überlast zu rechnen ist. Darüber hinaus ist die Metrik Hardware-unabhängig, d.h. sie erfordert im Gegensatz zur Channel Busy Fraction keinen Zugriff auf Hardware-Register, die oft nicht verfügbar sind und deren Interpretation darüber hinaus hardwareabhängig ist. Ebenso erzeugt die Metrik keinen nennenswerten Overhead, wenn das Time Token Bucket-Verfahren ohnehin schon zur Verkehrsformung zum Einsatz kommt. Die Experimente haben weiterhin gezeigt, dass es nicht ausreicht, die aktuell nicht nutzbare Bandbreite an einem Knoten zu betrachten, um zu bewerten, ob ihm problemlos weitere Bandbreite zugewiesen werden kann. Da auf Grund externer Einflüsse die Knoten unterschiedlich stark belastet werden, kann das Zuordnen zusätzlicher Bandbreite zu Problemen bei Nachbarknoten führen. Um die verfügbare Bandbreite optimal auszunutzen, ohne dass einzelne Knoten in Überlastsituationen geraten, in denen sie die ihnen zugewiesene Bandbreite nicht nutzen können, ist daher ein kooperatives Verfahren notwendig. Diesem Thema widmet sich das nächste Kapitel.

4.4 Kooperative faire Bandbreitenskalierung

Im vorherigen Kapitel wurde ein Konzept zur Verkehrskontrolle vorgestellt, welches auf dem Time Token Bucket-Verfahren basiert. Dabei wird sichergestellt, dass jeder Knoten im Netzwerk nur die ihm per Reservierung zugewiesene Bandbreite nutzt. Sofern insgesamt nicht zu viel Bandbreite an die Knoten zugewiesen wird, verbessert dies die Zuverlässigkeit des Netzwerkes, da Überlastsituationen vermieden werden. Wie viel Bandbreite den internen Knoten zugewiesen werden kann, ist allerdings schwierig zu entscheiden, da dies auch davon abhängt, wie viel externer Verkehr auf dem Medium stattfindet. Mit der *Unusable Wasted Bandwidth Ratio* wurde eine Metrik vorgestellt, welche es ermöglicht, die Auslastung des Mediums abzuschätzen und drohende Überlastsituationen zu erkennen. Neue Reservierungen für Bandbreite können somit abgelehnt werden, wenn das Medium damit wahrscheinlich in eine Überlastsituation geraten würde. Allerdings sieht der vorgestellte Reservierungsmechanismus nicht vor, bereits vergebene Reservierungen zurückzuziehen. Darüber hinaus haben die Experimente gezeigt, dass auf Grund der Topologie einige Knoten früher in Überlastsituationen geraten als andere. Wünschenswert wäre daher, dass die Knoten ihre Bandbreitennutzung automatisch dynamisch an die lokale Auslastung des Mediums anpassen und dabei auch auf Nachbarknoten Rücksicht nehmen. Dies sollte allerdings die Dienstgüteanforderungen der Anwendung berücksichtigen und die verfügbare Bandbreite sollte fair zwischen den Knoten verteilt werden. Das bedeutet, dass nach Möglichkeit die Mindestanforderungen jedes Knotens erfüllt werden und die darüber hinaus verfügbare Bandbreite fair verteilt wird.
In diesem Kapitel wird zunächst der Stand der Technik vorgestellt, wobei auch verschiedene Fairness-Definitionen betrachtet werden. Anschließend wird mit *QoS Fairness* eine in [KGM19] publizierte Fairness-Definition vorgestellt, welche die hier gewünschte Fairness unter Berücksichtigung von Dienstgüteanforderungen zum Ziel hat. Daraufhin wird ein Verfahren zur dynamischen Bandbreitenskalierung vorgestellt, welches auf dem Time Token Bucket-Verfahren basiert und in [KGM19] publiziert wurde. Das Verfahren ist kooperativ und erzielt die gewünschte QoS Fairness bei der Verteilung der verfügbaren Bandbreite. Schließlich wird die Implementierung des Verfahrens vorgestellt und Ergebnisse zur Evaluierung des Verfahrens mittels Experimenten präsentiert.

4.4.1 Stand der Technik

Zunächst betrachten wir in diesem Kapitel die in der Literatur am häufigsten verwendeten Definitionen von Fairness und eine Metrik, welche die Fairness einer Verteilung quantifizieren kann. Anschließend wird auf Algorithmen eingegangen, welche eine faire Bandbreitenverteilung zum Ziel haben.

4.4.1.1 Fairness-Definitionen und -Metriken

Bei der Verteilung von Bandbreite zwischen Knoten in einem drahtlosen Netzwerk gibt es verschiedene Fairness-Definitionen. Wir betrachten und vergleichen hier die unterschiedlichen Definitionen sowie Metriken, die es uns erlauben, die Fairness einer Verteilung zu bewerten.

Throughput Fairness Throughput Fairness hat zum Ziel, die Verteilung des Daten-Durchsatzes (engl.: Throughput) im Netzwerk fair, d.h. möglichst gleich, zu verteilen. Perfekte Throughput Fairness ist also dann gegeben, wenn über einen längeren Zeitraum betrachtet alle Knoten

im Netzwerk die gleiche Menge an Daten senden konnten. Bianchi untersuchte in [Bia00] unter Verwendung von Markow-Ketten die Sende-Wahrscheinlichkeiten und daraus resultierende Durchsatz-Verteilung des DCF-Mechanismus von IEEE 802.11 theoretisch. Er folgert, dass durch den DCF-Mechanismus Throughput Fairness erreicht wird, falls alle Knoten die gleiche Datenrate verwenden und gleich große Rahmen senden. Beide Annahmen sind allerdings in der Praxis fast nie erfüllt.

Time Fairness Bei der Time Fairness ist das Ziel nicht, den Daten-Durchsatz der Knoten gleich zu verteilen, sondern die Sendedauer. Das bedeutet, dass die Time Fairness dann perfekt ist, wenn über einen längeren Zeitraum betrachtet die Sendedauer aller Knoten im Netzwerk gleich ist. Wenn alle Knoten immer mit der gleichen Datenrate senden, dann ist offensichtlich, dass Time Fairness und Throughput Fairness identisch sind. Bei IEEE 802.11 sind mittlerweile allerdings viele unterschiedliche Datenraten spezifiziert. Knoten wählen die verwendete Datenrate dynamisch in Abhängigkeit der wahrgenommenen Linkqualität, welche z.B. durch die Entfernung beeinflusst wird. So ist es beispielsweise möglich, dass ein weit vom Access Point entfernter Knoten seine Daten mit langsamen 1 MBit/s überträgt, wohingegen ein nahe am Access Point positionierter Knoten eine Datenrate von 300 MBit/s verwendet. Eine unter Throughput Fairness optimale Verteilung lässt also das Medium 300 mal so lange durch den langsamen Knoten zum Senden belegen, wie durch den schnellen. Daraus folgt, dass Throughput Fairness in einem Szenario, in dem unterschiedliche Datenraten zum Einsatz kommen, den Gesamtdurchsatz des Netzwerkes drastisch reduziert. Dies haben Tan und Guttag in [TG04] experimentell gezeigt und schlagen als Lösung für dieses Problem Time Fairness vor.

Max-Min Fairness Bei der Max-Min Fairness wird Fairness als Optimierungsproblem betrachtet [Le+13]. Sie kann in der Flusskontrolle in Netzwerken angewendet werden [BG87], wir betrachten hier allerdings die Max-Min Fairness in Bezug auf die Verteilung von Ressourcen (Zeit bzw. Durchsatz) eines drahtlosen Singlehop-Netzwerks. Dabei ist sie geeignet in Fällen, in denen nicht alle Knoten gleich viele Ressourcen benötigen (bzw. anfordern). So mag es zwar fair sein, einem Knoten, der kaum Daten senden möchte, genauso viel Sendedauer zuzuteilen wie einem Knoten, der viel sendet. Dennoch ist es unsinnig, einem Knoten Ressourcen zuzuteilen, die er gar nicht nutzt, und dafür andere Knoten zu beschränken, die mehr Ressourcen nutzen könnten. Die Max-Min Fairness *maximiert* die Anteile der Knoten, die am wenigsten (die *minimalsten*) Ressourcen erhalten (da sie am wenigsten anfordern). Dazu verteilt sie die vorhandenen Ressourcen (d.h. Zeit oder Durchsatz) zunächst gleichmäßig auf alle Knoten. Falls Knoten damit mehr Ressourcen zugeteilt würden, als diese angefordert haben, so werden diese Ressourcen, die sonst ungenutzt bleiben würden, auf die verbleibenden Knoten gleichmäßig verteilt. Sollten damit nun wieder Knoten mehr Ressourcen zugeteilt bekommen haben, als sie angefordert haben, so werden diese auf die gleiche Weise weiter verteilt. Die Verteilung endet damit, dass entweder alle Ressourcen vergeben sind, d.h. nicht alle Anforderungen vollständig erfüllt werden konnten, oder aber alle Anforderungen erfüllt worden sind und übrige Ressourcen nicht benötigt werden. Diese Verteilung ist fair, da jeder Knoten entweder so viel Ressourcen zugeteilt bekommt, wie er angefordert hat, oder aber genauso viel Ressourcen wie alle anderen Knoten, die genauso viel oder mehr Ressourcen angefordert haben. Daher hat die resultierende Verteilung die Eigenschaft, dass ein Versuch, einem Knoten mehr Ressourcen zuzuordnen, stets dazu führt, dass ein anderer Knoten mit gleicher oder geringerer Zuordnung nun weniger Ressourcen erhält (da er weniger anfordert). Nehmen wir eine Ressourcenverteilung X, welche jedem

Knoten $v \in V$ die Ressourcen $X(v)$ zuordnet. Gesucht ist nun eine angepasste Ressourcenverteilung X', bei der ein Knoten v^+ mehr Ressourcen zugeteilt bekommen hat, d.h. $X'(v^+) > X(v^+)$, und die Max-Min fair ist. Die Summe der zugewiesenen Ressourcen soll unverändert bleiben, d.h. $\sum_{v \in V} X(v) = \sum_{v \in V} X'(v)$, was bedeutet, dass andere Knoten Ressourcen abgeben müssen. Die angepasste Verteilung X' ist nur dann Max-Min fair, wenn es einen Knoten v^- gibt, der zuvor genauso viel oder weniger Ressourcen zugeteilt bekommen hat wie v^+, d.h. $R(v^-) \leq R(v^+)$, und der jetzt weniger Ressourcen zugeteilt bekommt, d.h. $R'(v^-) < R(v^-)$. Dies ist nur möglich, wenn v^- weniger Ressourcen anfordert und daher weniger Ressourcen zugeteilt bekommt, sodass diese v^+ zugeteilt werden können.

Proportional Fairness Bei der Proportional Fairness [Kel97] wird die Verteilung der Ressourcen ebenfalls als Optimierungsproblem betrachtet. Hierbei werden die vorhandenen Ressourcen proportional zu einer Nutzenfunktion verteilt, die angibt, wie viel eine Einheit der Ressource dem Knoten Nutzen bringt. Kelly erklärt diesen Ansatz so, dass zunächst jeder Knoten angibt, wie viel er bereit ist, für eine Einheit der Ressourcen zu bezahlen [Kel97]. Die Ressourcenverteilung ist dann proportional fair, wenn die zugewiesenen Ressourcen im Gleichgewicht zu den gebotenen Gebühren stehen. Knoten, die bereit sind mehr zu bezahlen, bekommen also mehr Ressourcen zugewiesen. Es können also Knoten über die Nutzenfunktion priorisiert werden. Beispielsweise können Knoten mit einer höheren Datenrate priorisiert werden, da dies zu einem höheren Gesamtdurchsatz des Netzwerkes führt. Le et al. beweisen in [Le+13], dass Proportional Fairness (bezogen auf Zeit als Ressource) bei einem stabilen und gesättigten Netzwerk zu Time Fairness führt, wenn die Nutzenfunktion für alle Knoten den gleichen Wert ergibt, d.h. alle Knoten gleich viel zu zahlen bereit sind. Das Optimierungsproblem im Allgemeinen ist NP-schwer [LPY08], daher lassen sich optimale Lösungen im Allgemeinen nicht effizient berechnen. Ansätze, die Proportional Fairness für IEEE 802.11 zum Ziel haben, wurden beispielsweise von Banchs et al. [BSO07] und Li et al. [LPY08] vorgeschlagen.

Jains Fairness Index Von Raj Jain et al. [JCH84] wurde eine Metrik vorgeschlagen, welche die Fairness einer Verteilung x zwischen n Knoten auf einer Skala von 0 bis 1 (genauer: $\frac{1}{n}$ bis 1) bewertet. Die Metrik wird auch als *Jains Fairness Index* bezeichnet und ist wie folgt definiert [JCH84] :

$$fairnessIndex(x) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{4.29}$$

Dieser Fairness Index ist prinzipiell unabhängig von der Metrik, welche für die Verteilung x genutzt wird. Im einfachsten Fall gibt x_i den Durchsatz des Knoten i an, in diesem Fall kann der Fairness Index zur Bewertung der Throughput Fairness genutzt werden. Wenn x_i hingegen die dem Knoten i zugewiesene Sendezeit angibt, so lässt sich über den Index die Time Fairness bewerten. Durch Normalisierung kann der Index auch zur Bewertung der Max-Min oder Proportional Fairness genutzt werden. Beispielsweise schlagen Jain et al. in [JDB99] vor, den Durchsatz T_i des Knotens i durch das Optimum O_i zu teilen, welches durch einen Optimierungsansatz wie die Max-Min Fairness ermittelt wird:

$$x_i = \frac{T_i}{O_i} \tag{4.30}$$

Der Index hat einige positive Eigenschaften [JDB99]:

- Der Index ist unabhängig von der Einheit und Skalierung. Es ist also unerheblich, ob beispielsweise der Durchsatz in KBit/s oder MBit/s eingesetzt wird.
- Der Index ist auf einen Wertebereich zwischen 0 und 1 bzw. 0% und 100% begrenzt.
- Der Index lässt sich für jede beliebige Anzahl Knoten anwenden, d.h. insbesondere auch auf eine sehr kleine Anzahl Knoten.
- Es besteht eine direkte Beziehung zwischen dem Index und der Fairness, d.h. eine höhere Fairness wird durch einen höheren Index ausgedrückt.
- Der Wertebereich ist zusammenhängend.

Diese Eigenschaften treffen nicht alle auf Alternativen wie beispielsweise das arithmetische Mittel, die (empirische) Varianz oder die Standardabweichung zu.

4.4.1.2 Verfahren zur fairen Bandbreitenverteilung

In der Literatur wurden viele Verfahren vorgeschlagen, welche die Verteilung der Bandbreite bei IEEE 802.11 basierten Netzwerken fairer machen sollen. Oft wird dabei vorgeschlagen, den DCF-Mechanismus von IEEE 802.11 anzupassen. Beispielsweise passt der *Idle Sense* Algorithmus [Heu+05] die Größe des Contention Windows dynamisch basierend auf der Bitrate des Knotens sowie der Idle-Zeit, in der nicht auf dem Medium gesendet wurde, an. Le et al. [Le+13] schlagen ebenfalls ein Verfahren vor, bei dem die Größe des Contention Windows dynamisch berechnet wird. Sie verwenden dazu eine Kostenfunktion, um möglichst gute Time Fairness zu erreichen. Dieser Ansatz verlangt unter anderem auch, den Exponential Backoff-Mechanismus von IEEE 802.11 zu deaktivieren. Dieser Mechanismus verdoppelt die Größe des Contention Windows im Falle einer fehlgeschlagenen Übertragung. Dies soll die Wahrscheinlichkeit erhöhen, dass eine Neuübertragung nicht erneut fehlschlägt. Dieser Mechanismus benachteiligt natürlich Knoten, deren Übertragungen häufig fehlschlagen, und wirkt sich somit negativ auf die Fairness aus. Andererseits ist er notwendig, um sicherzustellen, dass Überlastsituationen aufgelöst werden können. Ansätze, welche die Größe des Contention Windows anpassen, haben außerdem meist das Problem, dass sie Legacy-Knoten, welche die vorgeschlagene Anpassung (noch) nicht umsetzen, nicht gut unterstützen. Wählt das Verfahren größere Contention Windows als die Legacy-Knoten, so führt dies dazu, dass die Legacy-Knoten im Vorteil sind und priorisiert werden. Wählt das Verfahren hingegen kleinere Contention Windows, so werden die Legacy-Knoten benachteiligt und u.U. stark eingeschränkt. Außerdem ist die Implementierung solcher Verfahren schwierig, da Änderungen der IEEE 802.11 MAC-Schicht meist Änderungen in der Hardware oder deren Treiber erfordern. Aus diesem Grund werden diese Verfahren in der Literatur i.d.R. nur simuliert und nicht auf echter Hardware implementiert.

Tan und Guttag schlagen in [TG04] einen Algorithmus vor, der auf dem Time Token Bucket-Verfahren basiert und den sie *Time-based Regulator* (TBR) nennen. Das Verfahren nimmt keine Änderungen an der MAC-Schicht von IEEE 802.11 vor, sondern formt auf den Access Points den Verkehr unter Verwendung des Time Token Bucket-Verfahrens, um damit Time Fairness zu erreichen. Damit erfordert es nur auf den Access Points Änderungen und hat damit eine bessere Unterstützung für Legacy-Knoten. Allerdings funktioniert das Verfahren damit auch nur in Access Point-basierten WLAN-Netzwerken und lässt sich daher nicht auf Mesh-Nertwerke mit Multihop-Topologien anwenden. Außerdem kann das Verfahren keine Dienstgüteanforderungen berücksichtigen. Choi et al. präsentieren in [CYK08] ein Scheduling-Verfahren, was ebenfalls auf dem Token Bucket-Verfahren basiert. Das Verfahren erlaubt pro Token das Senden eines Rahmens, unabhängig von der Rahmengröße und genutzten Datenrate. Dies würde normalerweise

bestenfalls zu Throughput Fairness führen. Um dennoch Time Fairness zu erreichen, passen die Autoren das Intervall, mit dem Tokens erzeugt werden, abhängig von der Datenrate des Knotens an. Weiterhin wird dieses Intervall dynamisch an die Anzahl Knoten angepasst, die (vermutlich) aktuell um das Medium konkurrieren. Auf diese Art soll sich das Verfahren dynamisch an stärkeren Wettbewerb um das Medium anpassen. Unterschiedliche Rahmengrößen werden nicht berücksichtigt, sodass Knoten, die größere Rahmen schicken, mehr Sendezeit erhalten als Knoten, die kleine Rahmen schicken. Dies kann sich positiv auf den Gesamtdurchsatz des Mediums auswirken, da große Rahmen weniger Overhead (wie Interframe Spaces) erzeugen. Allerdings wirkt sich dies schlecht auf die Time Fairness des Verfahrens aus.

Raniwala et al. [Ran+09] stellen mit *coordinated congestion control* (C3L) einen Algorithmus vor, der Max-Min Fairness zwischen konkurrierenden Datenflüssen erreichen will. Der Algorithmus schätzt die verfügbare Kanalkapzität ab, indem er die Warteschlangen der Knoten beobachtet. Er enthält ein Verfahren zur Topologieerkennung mittels HELLO-Nachrichten, welches die Linkqualität aus der Verlustrate von HELLO-Nachrichten ableitet. Das Verfahren geht, ähnlich wie das in Kapitel 3.2.1.3 betrachtete RID [Zho+05], davon aus, dass mit Knoten in Interferenzreichweite kommuniziert werden kann, indem die Sendestärke um 3 dB erhöht wird. Basierend auf der erkannten Topologie werden Cliquen gebildet, welche die Kollisionsdomains modellieren. Anschließend wird die Bandbreite innerhalb einer Clique fair verteilt. Trotz der zentralisierten Architektur ist das Verfahren recht komplex und berücksichtigt keine Dienstgüteanforderungen.

4.4.2 QoS Fairness

Die in Kapitel 4.4.1.1 vorgestellten Fairness-Definitionen verstehen Fairness meist so, dass alle Knoten gleich viele Ressourcen erhalten. Die Anforderungen der Knoten werden höchstens in sofern berücksichtigt, dass Knoten nicht mehr Ressourcen zugewiesen werden, als diese angefordert haben. Allerdings berücksichtigen diese Definitionen nicht, dass die Bandbreitenanforderung eines Knotens oft nicht nur ein einzelner Wert, sondern ein Bereich ist. Dienstgütefähige Anwendungen benötigen oft eine *Basisbandbreite*, die mindestens benötigt wird. Darüber hinaus können diese Anwendungen oft bis zu einer *präferierten Bandbreite* hoch skalieren. Jeder Knoten sollte also seine Basisbandbreite erhalten, und die verbleibende Bandbreite sollte darüber hinaus fair verteilt werden.

Um eine Fairness-Definition zu finden, welche die Dienstgüteanforderungen der Knoten berücksichtigt, formalisieren wir zunächst die Anforderungen der Knoten wie folgt:

- $requestedBwRatioBase_v \in [0;1]$ beschreibt die Basisbandbreite des Knotens v.
- $requestedBwRatioPref_v \in [0;1]$ beschreibt die präferierte Bandbreite des Knotens v.

Beide Angaben sind relativ, wobei $requestedBwRatioPref_v = 100\%$ bedeutet, dass der Knoten die komplette Bandbreite wünscht. Es wird davon ausgegangen, dass die Werte dieser Parameter durch eine Anwendung (oder Middleware) mit Dienstgüte-Unterstützung gegeben sind. Wir betrachten ein einfaches Beispiel, um zu verstehen, wie diese Werte berechnet werden können. Die Anwendung auf Knoten v sendet Pakete mit einer festen Größe $size = 400B$ in einem Intervall zwischen $interval_{base} = \frac{1\ frame}{1000\ ms}$ und $interval_{pref} = \frac{1\ frame}{500\ ms}$. Die Datenrate beträgt dabei $dataRate = 54MBit/s$. Daraus ergeben sich folgende Parameter:

$$requestedBwRatioBase_v = \frac{interval_{base} \cdot size}{dataRate} = \frac{\frac{1\ frame}{1000\ ms} \cdot 400\ B}{54\ MBit/s} = 5{,}9\ \% \tag{4.31}$$

$$requestedBwRatioPref_v = \frac{interval_{pref} \cdot size}{dataRate} = \frac{\frac{1\ frame}{500\ ms} \cdot 400\ B}{54\ MBit/s} = 11{,}9\ \% \tag{4.32}$$

Gibt es einen zentralen Bandbreitenmanager wie den in Kapitel 4.3.2.1 beschriebenen, so muss bei diesem die Basisbandbreite angefordert werden. So kann der Bandbreitenmanager sicherstellen, dass es nach der Reservierung weiterhin möglich ist, dass alle Knoten ihre Basisbandbreite nutzen können. Das bedeutet in einem Singlehop-Netzwerk, dass nicht mehr als die komplette verfügbare Bandbreite $availableBw_{G^{int}}$ zugewiesen werden darf:

$$\sum_{v \in V^{int}} requestedBwRatioBase_v \leq availableBw_{G^{int}} \leq 100\% \tag{4.33}$$

Die Knoten sollen dann die zugewiesene Bandbreite $assignedBw_v(t)$ dynamisch auf eine faire Weise abhängig von der Kanalauslastung zwischen den Grenzwerten $requestedBwRatioBase_v$ und $requestedBwRatioPref_v$ skalieren:

$$requestedBwRatioBase_v \leq assignedBw_v(t) \leq requestedBwRatioPref_v \tag{4.34}$$

Bei der Verteilung der Bandbreite, die über die Basisbandbreite hinaus vergeben werden kann, sollen die Knoten unter Berücksichtigung ihrer Dienstgüteanforderungen fair behandelt werden. Das bedeutet, dass die zusätzliche Bandbreite, die ein Knoten über seine Basisbandbreite hinaus zugewiesen bekommt, in ähnlicher Relation zu seiner präferierten Bandbreite stehen sollte wie bei anderen Knoten. Um dies zu formalisieren, definieren wir zunächst die zusätzliche Bandbreite $extraBw_v(t)$, die der Knoten v über seine Basisbandbreite hinaus zugewiesen bekommt:

$$extraBw_v(t) = assignedBw_v(t) - requestedBwRatioBase_v \tag{4.35}$$

Die zusätzliche Bandbreite ist dann maximal, wenn die präferierte Bandbreite zugewiesen wird:

$$extraBw_{v,Max} = requestedBwRatioPref_v - requestedBwRatioBase_v \tag{4.36}$$

Nun setzen wir die zusätzlich zugewiesene Bandbreite $extraBw_v(t)$ in Relation zur maximalen zusätzlichen Bandbreite $extraBw_{v,Max}$ und bezeichnen das Ergebnis als $extraBwRatio_v(t)$:

$$extraBwRatio_v(t) = \frac{extraBw_v(t)}{extraBw_{v,Max}} \tag{4.37}$$

Wir betrachten dabei ausschließlich Knoten, die Dienstgüteanforderungen haben, welche eine Skalierung erlauben. Das bedeutet, dass die präferierte über der Basisbandbreite liegen muss und damit $extraBw_{v,Max}$ nicht 0 sein kann.

Nach dieser Definition ist $extraBwRatio_v(t) = 0\%$, wenn Knoten v die Basisbandbreite zugeordnet ist und 100% wenn die präferierte Bandbreite zugeordnet ist.

Basierend auf dieser Definition definieren wir *QoS Fairness* als eine Bandbreitenverteilung, bei der das $extraBwRatio_v(t)$ fair verteilt, d.h. im optimalen Fall für alle Knoten gleich ist.

Um QoS Fairness messbar zu machen, definieren wir den *QoS Fairness Index* basierend auf Jains Fairness Index wie folgt:

$$QoSFairnessIndex(t) = \frac{(\sum_{v \in V^{int}} extraBwRatio_v(t))^2}{|V^{int}| \cdot \sum_{v \in V^{int}} extraBwRatio_v(t)^2} \tag{4.38}$$

Genau wie Jains Fairness Index hat der so definierte QoS Fairness Index einen Wertebereich zwischen $\frac{1}{|V^{int}|}$ und 1. Er ist 1, wenn alle Knoten die selbe $extraBwRatio_v(t)$ haben, also die von uns definierte QoS Fairness optimal ist. Dies ist beispielsweise dann der Fall, wenn allen Knoten ihre präferierte Bandbreite zugeordnet wurde, sodass $\forall v \in V^{int} : extraBwRatio_v(t) = 1$. Der Fairness Index sollte allerdings auch 1 sein, wenn allen Knoten die Basisbandbreite zugeordnet ist, d.h. $\forall v \in V^{int} : extraBwRatio_v(t) = 0$. In diesem Fall führt die obige Definition allerdings zu einer Division durch Null. Dies ist tatsächlich eine Schwäche von Jains Fairness Index: Auch wenn man den Fairness Index wie von Jain vorgeschlagen über den Durchsatz der Knoten berechnet, so kann es vorkommen, dass alle Knoten einen Durchsatz von 0 haben, beispielsweise weil die Verbindung eines Gateways gebrochen ist. In diesem Fall ist der Index nicht definiert. Eine Verteilung, bei der alle Knoten nichts zugewiesen bekommen, erscheint intuitiv jedoch als fair. Wir behandeln diesen Fall daher über eine Fallunterscheidung entsprechend:

$$QoSFairnessIndex(t) = \begin{cases} 1 & \sum_{v \in V^{int}} extraBwRatio_v(t)^2 = 0 \\ \frac{(\sum_{v \in V^{int}} extraBwRatio_v(t))^2}{|V^{int}| \cdot \sum_{v \in V^{int}} extraBwRatio_v(t)^2} & \text{sonst} \end{cases} \tag{4.39}$$

Genauso wie wir Knoten, die nicht skalieren können, aus der Berechnung des Fairness Index ausschließen, schließen wir auch Knoten aus, welche die ihnen zugewiesene Bandbreite nicht vollständig ausschöpfen. Die Berechnung des Fairness Index erfolgt also nur über jene Knoten, die noch hoch skalieren möchten.

4.4.3 Token Bucket basierte dynamische Bandbreitenskalierung

Jeder Knoten soll seine Bandbreite dynamisch an die Mediumauslastung anpassen und dabei $assignedBw_v$ zwischen seiner Bandbreite und seiner präferierten Bandbreite skalieren. Dazu teilen wir den Bereich zwischen der Basisbandbreite und der präferierten Bandbreite zunächst in $levels$ äquidistante Level auf. Die zugewiesene Bandbreite $assignedBw_v(t)$ berechnet ein Knoten dann basierend auf dem aktuellen Level $level_v(t)$ und der Anzahl Levels $levels$:

$$assignedBw_v(t) = requestedBwRatioBase_v + \frac{level_v(t)}{levels} extraBw_{v,Max} \tag{4.40}$$

Setzen wir Formel (4.40) und Formel (4.35) in Formel (4.37) ein und vereinfachen, so erhalten wir:

$$extraBwRatio_v(t) = \frac{level_v(t)}{levels} \tag{4.41}$$

Über $level_v(t)$ kann also die die $extraBwRatio_v(t)$ linear skaliert werden. Bei $level_v(t) = 0$ erhält der Knoten keine zusätzliche Bandbreite zugeordnet, d.h. $extraBwRatio_v(t) = 0$ und die zugewiesene Bandbreite $assignedBw_v(t)$ beträgt lediglich die Basisbandbreite $requestedBwRatioBase_v$. Je höher das Level steigt, desto mehr Bandbreite wird zugewiesen. Erreicht der Knoten das höchste Level, d.h. $level_v(t) = levels$, so beträgt $extraBwRatio_v(t) = 1$ und er bekommt seine präferierte Bandbreite $requestedBwRatioPref_v$ zugewiesen.

Betrachten wir unsere Definition von Qos Fairness und den daraus entwickelten QoS Fairness Index in Formel (4.39), so erkennen wir, dass die QoS Fairness optimal ist, wenn alle Knoten auf dem gleichen Level sind, d.h. es gibt ein gemeinsames Level $c \in [0; levels]$ für das gilt $\forall v \in V^{int} : level_v(t) = c$.

Um die Bandbreite über das Time Token Bucket-Verfahren dynamisch auf die zugewiesene Bandbreite zu begrenzen, ist es entweder möglich, das Intervall $refillInt_v$ dynamisch anzupassen und damit die Geschwindigkeit, mit der Tokens erzeugt werden. Alternativ wäre es möglich, die Token-Größe $tokenSize_v$ dynamisch anzupassen. Wir entscheiden uns für die erste Variante und bestimmen das Refill-Intervall über die Token-Größe $tokenSize_v$ aus der zugewiesenen Bandbreite $assignedBw_v(t)$ wie folgt:

$$refillInt_v(t) = \frac{tokenSize_v}{assignedBw_v(t)} \tag{4.42}$$

Dieser Ansatz hat einen entscheidenden Vorteil: Angenommen, wir skalieren über die Token-Größe statt über das Refill-Intervall. Dies bedeutet, dass das Refill-Intervall bei allen Knoten immer gleichgroß ist. Falls zwei Knoten zufällig exakt synchronisiert sind, erhalten diese bei gleichem Refill-Intervall stets zeitgleich neue Tokens und geraten damit bei jeder Übertragung in Konflikt. Auch wenn dies dazu führt, dass die Knoten das Medium als stark ausgelastet wahrnehmen und deshalb runter skalieren, löst dies den Konflikt nicht. Skalieren wir dagegen über das Refill-Intervall, so würde einer der Knoten früher als der andere runter skalieren und damit sein Intervall verändern, mit dem er neue Tokens erhält. Selbst wenn der andere Knoten dann ebenfalls runter skaliert und damit das selbe Refill-Intervall erhält, so ist es dennoch unwahrscheinlich, dass beide Knoten immer noch synchronisiert Tokens erhalten. Indem das Skalierungsverfahren das Refill-Intervall dynamisch anpasst, kann es also derartige Synchronisierungsprobleme auflösen.

Bisher wurde festgelegt, wie die zugewiesene Bandbreite in Levels skaliert und die Bandbreite über das Time Token Bucket-Verfahren dynamisch auf die zugewiesene Bandbreite beschränkt werden kann. Es bleibt noch zu definieren, wie ein Knoten sein Level verändert, d.h. wann er hoch skaliert, indem er sein Level erhöht, bzw. herunter skaliert, indem er sein Level dekrementiert.

Zu Beginn sollen die Knoten mit ihrer Basisbandbreite senden. Wir gehen davon aus, dass dies möglich ist, da der zentrale Bandbreitenmanager sicherstellt, dass alle Knoten mit den von ihnen angeforderten Basisbandbreiten senden können. Danach wird regelmäßig überprüft, ob hoch- oder herunterskaliert werden soll. Dazu wird ein periodischer Timer verwendet, der z.B. alle $ControlInterval = 1000\ ms$ auslöst. Algorithmus 4.1 drückt dies in Pseudocode aus.

```
1 FUNCTION INIT()
2 begin
3   level_v ← 0
4   lastUpscale ← 0
5   ControlInterval ← 1000 ms
6   SetPeriodicTimer(ControlInterval, Scale())
7 end
```

Algorithmus 4.1: Initialisierung

```
1 FUNCTION Scale()
2 begin
3   if ScaleUpCondition() then
4     level_v ← level_v + 1
5     lastUpscale ← 0
6   else
7     lastUpscale ← lastUpscale + 1
```

```
8       if ScaleDownCondition() then
9         level_v ← level_v − 1
10        lastDownscale ← 0
11      else
12        lastDownscale ← lastDownscale + 1
13    neighborLevels ← ∅
14    neighborMaxUnusable ← 0
15  end
```

Algorithmus 4.2: Skalierung

Die über den Timer aufgerufene Funktion **Scale()** ist in Algorithmus 4.2 als Pseudocode aufgeführt. Sie prüft zunächst mit Hilfe der Funktion **ScaleUpCondition()**, ob die Bedingung zum Hochskalieren erfüllt ist. Ist dies der Fall, wird das Level $level_v$ inkrementiert und die Variable *lastUpscale* zurückgesetzt, die angibt, vor wie vielen Kontrollintervallen zuletzt hochskaliert wurde. Falls nicht hochskaliert wird, wird diese Variable inkrementiert und anschließend die Bedingungen zum Herunterskalieren mit Hilfe der Funktion **ScaleDownCondition** überprüft. Falls diese erfüllt sind, wird das Level dekrementiert und die Variable *lastDownscale* zurückgesetzt. Diese gibt, analog zu *lastUpscale*, an vor wie vielen Kontrollintervallen zuletzt herunterskaliert wurde. Entsprechend wird die Variable inkrementiert, falls nicht herunterskaliert wurde. Anschließend werden die Variablen *neighborLevels* und *neighborMaxUnusable* zurückgesetzt. Mit diesen merkt sich der Knoten den Zustand seiner Nachbarn:

- In *neighborLevels* merkt sich der Knoten die Levels seiner Nachbarn. Dies ist für das kooperative Verhalten notwendig, welches die gewünschte QoS Fairness sicherstellen soll.
- In *neighborMaxUnusable* speichert der Knoten die maximale Unusable Wasted Bandwidth Ratio der Nachbarknoten.

Diese Statusinformation wird mit den Nachbarn ausgetauscht, indem jedes gesendete Paket um die entsprechende Information ergänzt wird. In Algorithmus 4.3 ist der entsprechende Pseudocode aufgeführt. Die Funktion **Send()** wird aufgerufen, wenn die Anwendung oder eine höhere Schicht ein Paket senden möchte. Sie ergänzt das zu sendende Paket um das aktuelle Level $level_v$, sofern die zugewiesene Bandbreite auch ausgenutzt wird. Außerdem teilt sie die $unusabledWastedBwRatio_v$ mit den Nachbarknoten aus. Anschließend wird das Paket zum Versenden an die nächst tiefere Schicht übergeben, beispielsweise die MAC-Schicht.

```
1 FUNCTION Send(packet)
2 begin
3   if usabledWastedBwRatio_v = 0%
4     packet.level ← level_v
5   else
6     packet.level ← undefined
7   packet.unusabledWastedBwRatio ← unusabledWastedBwRatio_v
8   PassPacketToChildLayer(packet)
9 end
```

Algorithmus 4.3: Senden eines Pakets

```
1 FUNCTION ReceptionCallback(packet)
2 begin
3   if packet.level ≠ undefined
4     neighborLevels ← neighborLevels ∪ packet.level
5   if packet.unusabledWastedBwRatio > neighborMaxUnusable then
```

```
6       neighborMaxUnusable ← packet.unusabledWastedBwRatio
7   end
```

Algorithmus 4.4: Reception Callback

Beim Empfang einer Nachricht werden die beim Senden gesetzten Felder ausgelesen und in die Status-Variablen *neighborLevels* und *neighborMaxUnusable* aktualisiert. In Algorithmus 4.4 ist die Funktion **ReceptionCallback()** als Pseudocode aufgeführt, die dieses Verhalten umsetzt. Wichtig ist, dass diese Funktion auch von Unicast-Paketen durchlaufen werden muss, die nicht zwingend an diesen Knoten adressiert sind. Das bedeutet, dass die hier beschriebene Funktionalität der Bandbreitenskalierung unterhalb einer ggf. vorhandenen Routing-Schicht implementiert sein muss. Alternativ könnte man in einem regelmäßigen Intervall diese Information über Status-Nachrichten per Broadcast versenden.

```
1  FUNCTION ScaleUpCondition(): bool
2  begin
3     bool[6] condition
4     condition[1] = (level_v < levels)
5     condition[2] = (lastUpscale ≥ 10 + Random(0,9))
6     condition[3] = (usableWastedBwRatio_v = 0%)
7     condition[4] = (unusableWastedBwRatio_v ≤ 0.5%)
8     condition[5] = (∄ level_neighbor ∈ neighborLevels : level_neighbor < level_v)
9     condition[6] = (neighborMaxUnusable ≤ 0,5 %)
10    for i ← 1 TO 6 do
11       if condition[i] = FALSE then
12          return FALSE
13    end
14    return TRUE
15 end
```

Algorithmus 4.5: Bedingung zum Hochskalieren

In Algorithmus 4.5 ist der Pseudocode der Funktion **ScaleUpCondition()** aufgeführt, welcher die Bedingungen prüft, die zum Hochskalieren *alle* erfüllt sein müssen. Dies sind:

1. Der Knoten hat noch nicht das höchste Level erreicht, d.h. $level_v < levels$. Ein höheres Level würde dem Knoten mehr Bandbreite zuweisen, als er als präferierte Bandbreite angefordert hat. Die Knoten sollen aber nur zwischen der angeforderten Basisbandbreite und der präferierten Bandbreite skalieren.
2. Das letzte mal, dass der Knoten hochskaliert hat, liegt mindestens $10 + Random(0,9)$ Kontrollintervalle zurück. So soll verhindert werden, dass die Knoten zu schnell hoch skalieren. Der Zufallswert sorgt dafür, dass es unwahrscheinlicher ist, dass mehrere Knoten gleichzeitig hoch skalieren und dies dazu führt, dass die Bandbreite nicht ausreicht und alle Knoten wieder herunter skalieren müssen.
3. Der Knoten nutzt die ihm bisher zugewiesene Bandbreite vollständig aus, d.h. es gibt keine Usable Wasted Bandwidth. Knoten dürfen nicht hoch skalieren, wenn sie die ihnen zugewiesene Bandbreite nicht nutzen würden. Denn andernfalls wäre es möglich, dass sie plötzlich die zugewiesene Bandbreite vollständig ausnutzen und damit mehr Bandbreite nutzen, als verfügbar ist, was zu einer kurzfristigen Überlastsituation führen würde.
4. Der Knoten hatte ein Unusable Wasted Bandwidth Ratio von nicht mehr als 0,5 %. Es ist, insbesondere bei externem Verkehr, normal, dass ein wenig der zugewiesenen Bandbreite

nicht nutzbar ist. Aber wenn der Wert 0,5 % übersteigt, deutet dies daraufhin, dass nicht mehr Bandbreite verfügbar ist und hoch skalieren zu einer Überlastsituation führen würde.

5. Keiner der Nachbarknoten darf ein niedrigeres Level haben als der Knoten selbst. Dies stellt sicher, dass die Bandbreite entsprechend unserem Verständnis von QoS Fairness gerecht aufgeteilt wird. Andernfalls wäre es möglich, dass ein Knoten schneller hochskaliert als andere und damit mehr Bandbreite erhält. Durch diese Bedingung muss ein Knoten immer warten, bis alle seine Nachbarn das gleiche Level wie er selbst erreicht haben, bevor er weiter hoch skalieren darf. Zu beachten ist, dass Knoten, welche die ihnen zugewiesene Bandbreite nicht vollständig ausnutzen, dabei nicht berücksichtigt werden, da sie ihr Level gar nicht ihren Nachbarn mitteilen (s. **Send()**-Funktion). Andernfalls würde ein Knoten, der nicht hoch skaliert, weil er nicht viele Daten zu senden hat, seine Nachbarknoten unnötig am hoch skalieren hindern.
6. Kein Nachbar hat signalisiert, dass er bereits mehr als 0,5 % seiner zugewiesenen Bandbreite nicht nutzen kann. Diese Bedingung verhindert, dass Knoten auf Kosten ihrer Nachbarknoten hoch skalieren.

```
1  FUNCTION ScaleDownCondition() : bool
2  begin
3    if level_v > 0 AND lastDownscale ≥ 3 then
4      bool[4] condition
5      condition[1] = (usableWastedBwRatio_v > 5%)
6      condition[2] = (unusableWastedBwRatio_v > 5%)
7      condition[3] = (min(neighborLevels) < level − 1)
8      condition[4] = (neighborMaxUnusable > 5%)
9      for i ← 1 TO 4 do
10       if condition[i] = TRUE then
11         return TRUE
12     end
13   return FALSE
14 end
```

Algorithmus 4.6: Bedingung zum Herunterskalieren

Algorithmus 4.6 zeigt den Pseudocode der Funktion **ScaleDownCondition()** . Hierbei wird zunächst geprüft, ob das Level größer 0 ist, da herunter skalieren sonst dazu führen würde, dass weniger als die Basisbandbreite zugewiesen wird. Außerdem soll für mindestens 3 Kontrollintervalle nicht herunter skaliert werden. Da die verfügbare Bandbreite i.d.R. durch einige Knoten geteilt wird, wird bereits viel Bandbreite frei, wenn alle Knoten ein Level herunter skalieren. Daher ist es meistens nicht nötig, dass ein Knoten schnell mehrere Level herunter skaliert. Stattdessen wird den anderen Knoten etwas Zeit gegeben, um ebenfalls herunter zu skalieren. Anschließend werden folgende vier Bedingungen geprüft, wovon jede einzelne ausreicht, dass herunter skaliert wird:

1. Das Usable Wasted Bandwidth Ratio ist über 5 %, d.h. die zugewiesene Bandbreite wird zu 5 % nicht ausgenutzt, weil der Knoten nicht genügend Daten zum Senden hat. Dies kann vorkommen, wenn der Knoten vorher mehr Daten zu senden hatte, jetzt die zugewiesene Bandbreite aber nicht mehr ausschöpft. Er muss in diesem Fall herunter skalieren, denn andernfalls könnte er die zugewiesene Bandbreite plötzlich wieder nutzen, obwohl diese dann sehr wahrscheinlich nicht frei wäre, weil andere Knoten sie bereits durch hoch skalieren ausnutzen.

2. Das Unusable Wasted Bandwidth Ratio ist über 5 %, d.h. dass die zugewiesene Bandbreite zu 5 % nicht ausgenutzt wird, weil das Medium zu stark ausgelastet ist. Um die Last auf dem Medium zu reduzieren, soll der Knoten in diesem Fall herunter skalieren. Die Menge Bandbreite die er momentan zugewiesen hat, kann er ohnehin nicht ausnutzen.
3. Es gibt einen Nachbarknoten, der mehr als ein Level unter dem Knoten selbst ist. Dies kann insbesondere dann der Fall sein, wenn ein neuer Knoten auftaucht, der zunächst mit Level 0 startet. Um für ihn Bandbreite frei zu machen, müssen seine Nachbarn durch diese Bedingung ihr Level reduzieren. Auch hier werden wieder nur Knoten berücksichtigt, welche die ihnen zugewiesene Bandbreite auch ausnutzen.
4. Ein Nachbarknoten ist überlastet, d.h. kann mehr als 5 % seiner zugewiesenen Bandbreite nicht ausnutzen. Diese Bedingung sorgt dafür, dass Knoten Bandbreite für ihre Nachbarn frei machen, wenn diese überlastet sind.

Insgesamt erreichen die Bedingungen zur Skalierung folgende Ziele:

- Die zugewiesene Bandbreite kann von allen Knoten größtenteils (zu 95 %) genutzt werden.
- Die zugewiesene Bandbreite wird nach einer kurzen Stabilisierungsphase letztlich entsprechend unserer Definition von QoS Fairness fair verteilt.
- Knoten verhalten sich kooperativ gegenüber ihren Nachbarn und machen für diese Bandbreite frei, wenn diese überlastet sind oder sie nicht fair behandelt werden.
- Es wird so viel Bandbreite zugewiesen, wie unter Berücksichtigung der anderen Ziele möglich ist. Somit wird der Gesamtdurchsatz des Netzwerks soweit möglich optimiert.

4.4.4 Implementierung

Das beschriebene Verfahren zur kooperativen dynamischen Bandbreitenskalierung wurde im WiPS Framework (s. Kapitel 4.1) implementiert. Dabei wurde die Schicht zur Verkehrskontrolle, welche in Kapitel 4.3.4 beschrieben wurde, um ein Modul zur Bandbreitenregulierung erweitert. Die resultierende Architektur skizziert Abb. 4.12. Das Modul zu Bandbreitenskalierung ist dabei optional. Es konfiguriert den Time Token Bucket-Mechanismus der Verkehrskontrolle dynamisch, indem es dass Refill-Intervall des Token Bucket abhängig vom aktuellen Level anpasst. Die zuvor beschriebene Schicht zur Verkehrskontrolle hat Pakete ggf. verzögert oder verworfen, sie aber nicht verändert. Das Modul zur Bandbreitenskalierung fügt hingegen in von oben kommende Pakete einen eigenen Header mit Feldern ein, um Nachbarn das aktuelle Level sowie eine ggf. vorhandene Überlastsituation mitzuteilen. Bei empfangenen Paketen liest sie diese Felder aus und entfernt dem Header aus dem Paket. Die Änderungen sind also für andere Schichten transparent, da sie durch obere Schichten nicht sichtbar sind und untere Schichten sie als zur Payload gehörig ansehen.

Dennoch kann die Schicht zur Verkehrsformung nicht beliebig im Stack positioniert werden. Wie in Kapitel 4.3.4 beschrieben sollte die Schicht möglichst direkt über der WiPS MAC-Schicht platziert werden. Insbesondere muss eine ggf. vorhandener Routing-Schicht darüber platziert werden. Dies hat zwei Gründe: Zum einen müssen von der Routing-Schicht erzeugte Pakete evtl. auch verzögert werden. Zum anderen müssen auch Unicast-Pakete, welche nicht an den Knoten selbst adressiert sind, durch die Schicht laufen, damit die Bandbreitenskalierung ihre Felder auslesen kann. Ein Beispiel-Stack mit weiteren Schichten für Topologieerkennung, Routing und Middleware ist in Abb. 4.1 zu sehen.

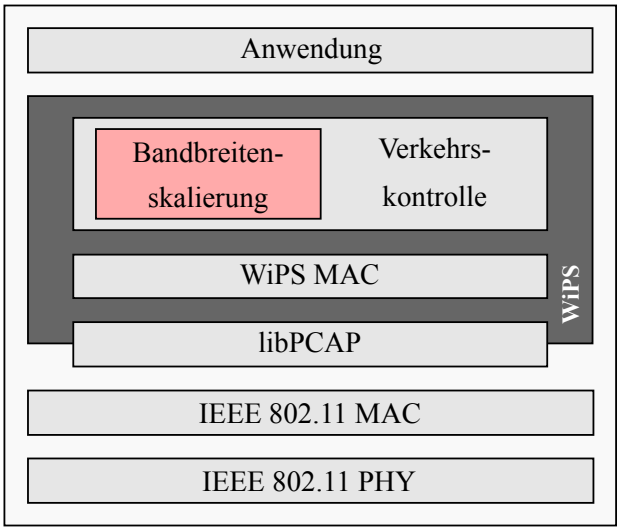


Abbildung 4.12: Architektur der Implementierung zum Testen der Bandbreitenskalierung im WiPS Framework

4.4.5 Evaluierung

In diesem Kapitel werden einige Ergebnisse von Experimenten vorgestellt, die das Verhalten der kooperativen Bandbreitenskalierung untersuchen. Zur Evaluierung wurde eine Testanwendung entwickelt, welche die implementierte WiPS-Schicht zur Verkehrskontrolle inkl. Modul zur Bandbreitenskalierung verwendet. Die Experimente wurden auf einem Testbett aus zehn APU2 Knoten ausgeführt (s. Kapitel 4.1.2). Auf diesen ist Arch Linux mit Kernel-Version 5.0.2 installiert. Als WLAN-Adapter wurde pro Knoten eine der zwei verbauten WLE200NX MiniPCI-E Karten verwendet. Diese unterstützen sowohl die 2,4 GHz als auch 5 GHz Kanäle. In den Experimenten wurde teilweise ein 5 GHz Kanal verwendet, auf dem so gut wie kein externer Verkehr herrscht, sodass das Verhalten ohne externe Einflüsse untersucht werden konnte. Die anderen Experimente wurden auf 2,4 GHz Kanälen ausgeführt, auf denen externe Knoten typischen Internetverkehr senden. Mit diesen Experimenten soll untersucht werden, wie sich das Verfahren in einer Umgebung mit externem Verkehr verhält. Die hier gezeigten Experimente wurden mehrfach wiederholt, wobei die Experimente auf dem 5 GHz Kanal sehr gut reproduzierbare Ergebnisse lieferten. Auf den 2,4 GHz Kanälen variieren die Ergebnisse durch unterschiedlichen externen Verkehr stärker. Allerdings sind diese Unterschiede nicht entscheidend und die hier gezeigten Ergebnisse repräsentieren das beobachtete Verhalten gut.

Experiment 1: In diesem Experiment untersuchen wir das Fairness-Verhalten des IEEE 802.11 Standards ohne unser Verfahren zur Bandbreitenskalierung. Dabei kommen vier Knoten in Singlehop-Reichweite zum Einsatz, die auf Kanal 40 (5,200 GHz) senden. Auf diesem Kanal herrscht so gut wie kein externer Verkehr. Die vier Knoten haben alle die selbe Basisbandbreite von 5 %, aber die präferierte Bandbreite variiert je nach Knoten zwischen 10 % und 40 %. Die einzelnen Werte sind zusammen mit den anderen Parametern des Experiments in Tab. 4.11 aufgeführt.

Die Ergebnisse des Experiments fasst Abb. 4.13 zusammen. Die Knoten 1, 2 und 3 können ihre präferierte Bandbreite fast vollständig nutzen. Lediglich Knoten 4 nutzt mit ca. 32% genutzter Bandbreite etwa 8 % weniger als er präferiert. Dies ist nach Max-Min Fairness in etwa die optimale Verteilung, aber nicht nach der von uns definierten QoS Fairness. Für die Knoten 1 bis 3 ist die über die Basisbandbreite hinausgehende zusätzliche Bandbreite $extraBw_v$ fast 1, weil sie

Tabelle 4.11: Parameter zu Experiment 1

Parameter	Wert
Anzahl Knoten	4
Bandbreitenskalierung	deaktiviert
Dauer	10 Minuten
Kanal (Frequenz)	40 (5,200 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_1$	10 %
$requestedBwRatioPref_2$	20 %
$requestedBwRatioPref_3$	30 %
$requestedBwRatioPref_4$	40 %

fast ihre präferierte Bandbreite nutzen. Knoten 4 kommt dagegen nur auf eine zusätzliche Bandbreite $extraBw_4$ von ca. 0,77 und wird damit gegenüber den anderen Knoten unfair behandelt. Daraus resultiert ein QoS Fairness Index von ca. 0,98, der ebenfalls in Abb. 4.13 eingezeichnet ist. Es wäre wünschenswert, dass der Fairness Index in einem solchen Fall sensitiver ist und stärker ausschlägt.

Experiment 2: Im zweiten Experiment untersuchen wir, wie sich das Verfahren zur kooperativen Bandbreitenskalierung verhält. Es kommen wieder vier Knoten zum Einsatz, welche die Bandbreitenanforderungen haben wie zuvor. Außerdem wird auch dieses Experiment auf Kanal 40 ausgeführt, auf dem so gut wie kein externer Verkehr herrscht. Die Parameter des Experiments fasst Tab. 4.12 zusammen.

Tabelle 4.12: Parameter zu Experiment 2

Parameter	Wert
Anzahl Knoten	4
Bandbreitenskalierung	aktiviert, $levels = 20$
Dauer	10 Minuten
Kanal (Frequenz)	40 (5,200 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_1$	10 %
$requestedBwRatioPref_2$	20 %
$requestedBwRatioPref_3$	30 %
$requestedBwRatioPref_4$	40 %

In Abb. 4.14 sind die Ergebnisse des Experiments veranschaulicht. Alle vier Knoten beginnen mit der angeforderten Basisbandbreite von 5 %. Daraufhin skalieren sie bis ca. Minute 4 hoch. Dabei skaliert Knoten 4 am schnellsten hoch, gefolgt von Knoten 3 und Knoten 2. Am langsamsten skaliert Knoten 1 hoch. Dies liegt an den unterschiedlichen präferierten Bandbreiten der

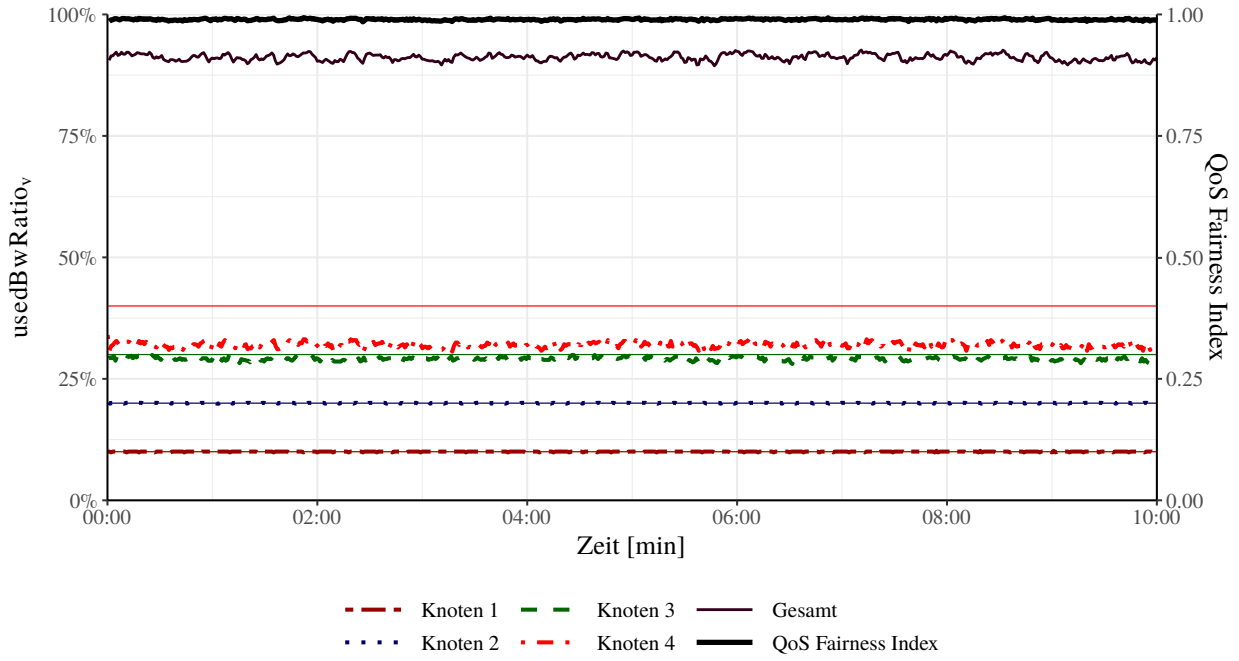


Abbildung 4.13: Genutze Bandbreite $usedBwRatio_v$ in Experiment 1 und der sich ergebende QoS Fairness Index

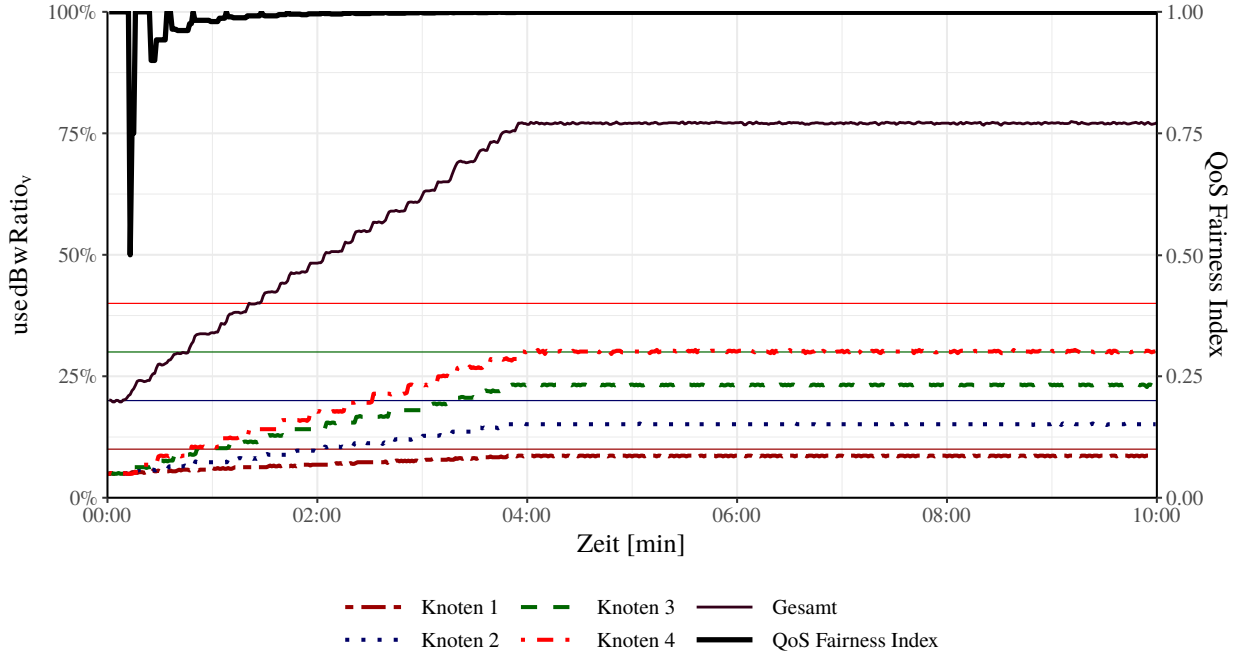


Abbildung 4.14: Genutzte Bandbreite $usedBwRatio_v$ in Experiment 2 und der sich ergebende QoS Fairness Index

Knoten. Knoten 4 hat eine präferierte Bandbreite von 40 % angefordert und skaliert daher bis zu einer maximalen zusätzlichen Bandbreite $extraBw_{4,Max} = 35$ % (s. Formel (4.36)). Diese Skalierungsbreite wird in $levels = 20$ Stufen eingeteilt. Für Knoten 4 entspricht ein Level daher 1,75 % Bandbreite. Knoten 1 hingegen hat eine präferierte Bandbreite von 10 % angefordert, sodass er nur maximal $extraBw_{1,Max} = 5$ % hoch skalieren kann. Ein Level entspricht für Knoten 1 daher

nur 0,25 % Bandbreite. Die Knoten erhöhen ihr Level mit der Zeit ungefähr gleich, aber Knoten 4 bekommt pro Level deutlich mehr Bandbreite zugewiesen als Knoten 1. Daher skaliert Knoten 4 deutlich schneller hoch als Knoten 1. Das Verfahren stellt sicher, dass alle Knoten immer maximal ein Level Unterschied haben. Dadurch erreicht die von uns definierte QoS Fairness (auch in Abb. 4.14 eingezeichnet) meistens einen optimalen Wert von 1,00. Lediglich am Anfang sinkt die QoS Fairness kurzzeitig auf niedrige Werte. Dies passiert, weil während der Phase des Hochskalierens immer ein Knoten kurzzeitig voraus geht und vor den anderen Knoten ein Level hoch skaliert. Insbesondere, wenn der erste Knoten von der Basisbandbreite ein Level hoch skaliert bedeutet dies, dass er der einzige Knoten ist, der mehr als die Basisbandbreite hat. Das führt dazu, dass die $extraBwRatio_v$, die für die Berechnung der QoS Fairness ausschlaggebend ist, kurzzeitig sehr unfair verteilt ist. Da die anderen Knoten aber schnell nachziehen, wird die QoS Fairness wieder 1,00. Im Laufe des weiteren Hochskalierens werden die Unterschiede, die ein Level ausmacht, immer unbedeutender, sodass die QoS Fairness dann weniger stark ausschlägt, wenn ein Knoten kurzzeitig ein Level über den anderen ist. Indem $levels$ größer gewählt wird, kann die Fairness also verbessert werden, da ein Level dann weniger Unterschied ausmacht. Durch eine höhere Anzahl Levels dauert der Prozess des Hochskalierens allerdings länger. Dies lässt sich ausgleichen, indem $ControlInterval$ kleiner gewählt wird, sodass schneller skaliert wird. Die hier gewählten Werte mit $levels = 20$ und $ControlInterval = 1000\ ms$ sorgen, wie auch das Experiment zeigt, dafür, dass eher langsam hoch skaliert wird. Dies wurde hier absichtlich so gewählt, damit das Verhalten des Verfahrens gut beobachtet werden kann. In einer Produktivumgebung würde es dagegen Sinn machen, ein deutlich kleineres Kontrollintervall und ggf. mehr Levels zu wählen.

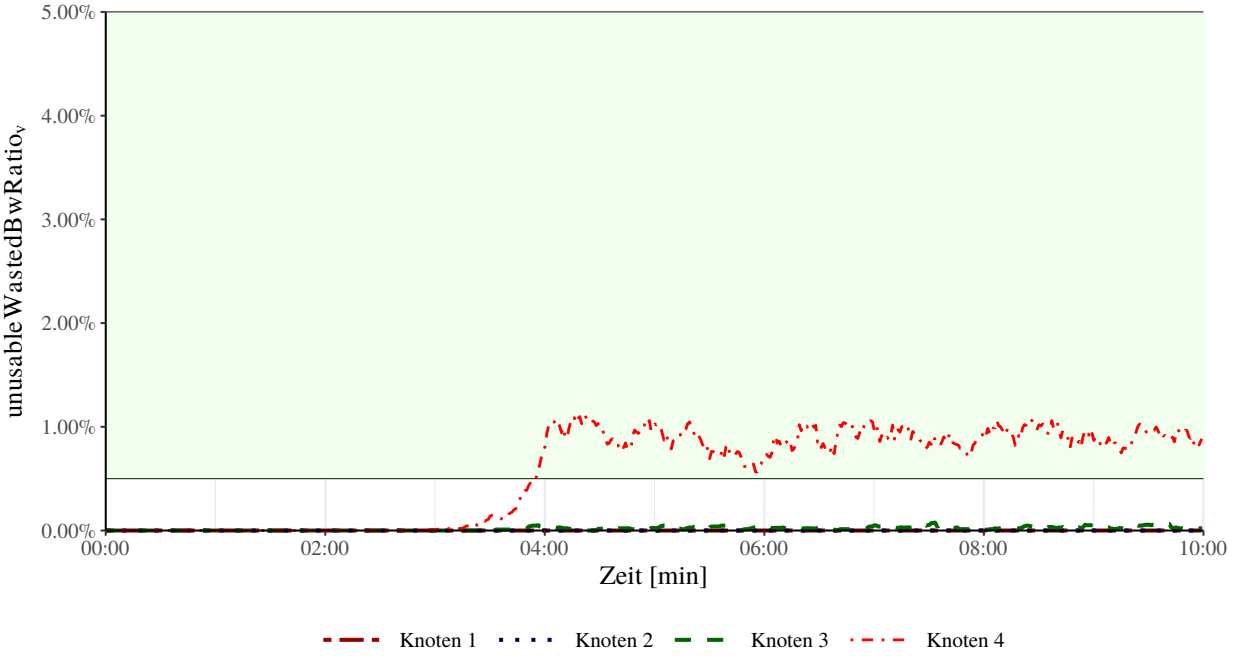


Abbildung 4.15: Unusable Wasted Bandwidth Ratio in Experiment 2

In Abb. 4.15 ist der Verlauf der Unusable Wasted Bandwidth Ratio der Knoten über die Zeit eingezeichnet. Hieran erkennt man den Grund, warum die Knoten nach 4 Minuten aufhören, weiter hoch zu skalieren. Zuvor war die Unusable Wasted Bandwidth Ratio für alle Knoten lange 0, d.h. sie konnten die zugewiesene Bandbreite vollständig nutzen. Erst nach 3 Minuten steigt die Unusable Wasted Bandwidth Ratio von Knoten 4 langsam an. Nach 4 Minuten steigt sie über

den Grenzwert von 0,5 %. Damit ist die vierte notwendige Bedingung zum Hochskalieren nicht mehr erfüllt (s. Algorithmus 4.5) und Knoten 4 skaliert nicht weiter hoch. Außerdem teilt Knoten 4 seine Unusable Wasted Bandwidth Ratio seinen Nachbarknoten mit. Dadurch ist bei diesen die sechste notwendige Bedingung zum Hochskalieren nicht mehr erfüllt (s. Algorithmus 4.5) und auch sie skalieren nicht weiter hoch.

Vergleichen wir die Ergebnisse dieses Experiments mit denen des vorigen Experiments, fällt auf, dass die Gesamtbandbreite aller vier Knoten um ca. 10 % unter der im ersten Experiment ohne Bandbreitenskalierung bleibt. Allerdings ist auch die von den Knoten genutzte Bandbreite in Experiment 2 nach der Phase des Hochskalierens deutlich stabiler als im ersten Experiment. Um ein stabiles und damit zuverlässigeres Verhalten zu erzielen, verzichten wir auf einen Teil der Bandbreite. Dies ist in anderen Ansätzen, die Zuverlässigkeit erzielen wollen, ähnlich: Beispielsweise sind für zuverlässige TDMA-Übertragungen, wie sie in Kapitel 3 benutzt wurden, Schutzintervalle (engl.: Guard Intervals) zwischen den Slots notwendig, die ebenfalls Bandbreite kosten. Die ersten beiden Experimente wurden auf einem Kanal durchgeführt, auf dem fast kein externer Verkehr herrscht. Auf einem solchen Kanal ist es allerdings verhältnismäßig einfach, stabile und zuverlässige Übertragungen zu erzielen. In späteren Experimenten werden wir das Verhalten unter dem Einfluss externer Knoten untersuchen und sehen, ob das Verfahren auch unter variierenden externen Einflüssen ein stabiles Verhalten erreichen kann.

Experiment 3: Zunächst analysieren wir im dritten Experiment das Verhalten des Hochskalierens genauer und betrachten den Fall, dass ein Knoten nach einiger Zeit hinzu kommt. Insgesamt kommen in diesem Experiment drei Knoten zum Einsatz, die diesmal alle die gleichen Bandbreitenanforderungen haben: Die Basisbandbreite beträgt weiterhin niedrige 5 %. Die präferierte Bandbreite ist mit 55 % nun so hoch, dass es unmöglich ist, dass zwei oder alle drei Knoten diese erreichen können, da dies über 100 % Gesamtbandbreite ergeben würde. Das Experiment wird wieder auf Kanal 40 (5,200 GHz) ausgeführt, auf dem fast kein externer Verkehr herrscht. Alle Parameter des Experiments sind in Tab. 4.13 zusammengefasst.

Tabelle 4.13: Parameter zu Experiment 3

Parameter	Wert
Anzahl Knoten	3
Bandbreitenskalierung	aktiviert, $levels = 20$
Dauer	10 Minuten
Kanal (Frequenz)	40 (5,200 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_v(\forall v \in V)$	55 %

Die resultierende Bandbreitenverteilung der drei Knoten während dieses Experiments illustriert Abb. 4.16 zusammen mit dem QoS Fairness Index, der sich aus dieser Verteilung ergibt. In Abb. 4.17 wird der Verlauf der Unusable Wasted Bandwidth Ratio gezeigt. Knoten 2 und Knoten 3 skalieren, ähnlich wie im vorigen Experiment, von ihrer Basisbandbreite von 5 % für ca. 3 Minuten in Schritten von 2,5 % hoch. Dabei unterscheiden sich beide Knoten immer um maximal ein Level (2,5 %), wobei Knoten 3 meist vor Knoten 2 auf das nächst höhere Level wechselt. Nach drei Minuten nutzt Knoten 2 ca. 34 % und Knoten 3 ca. 36 % Bandbreite, d.h. die genutzte

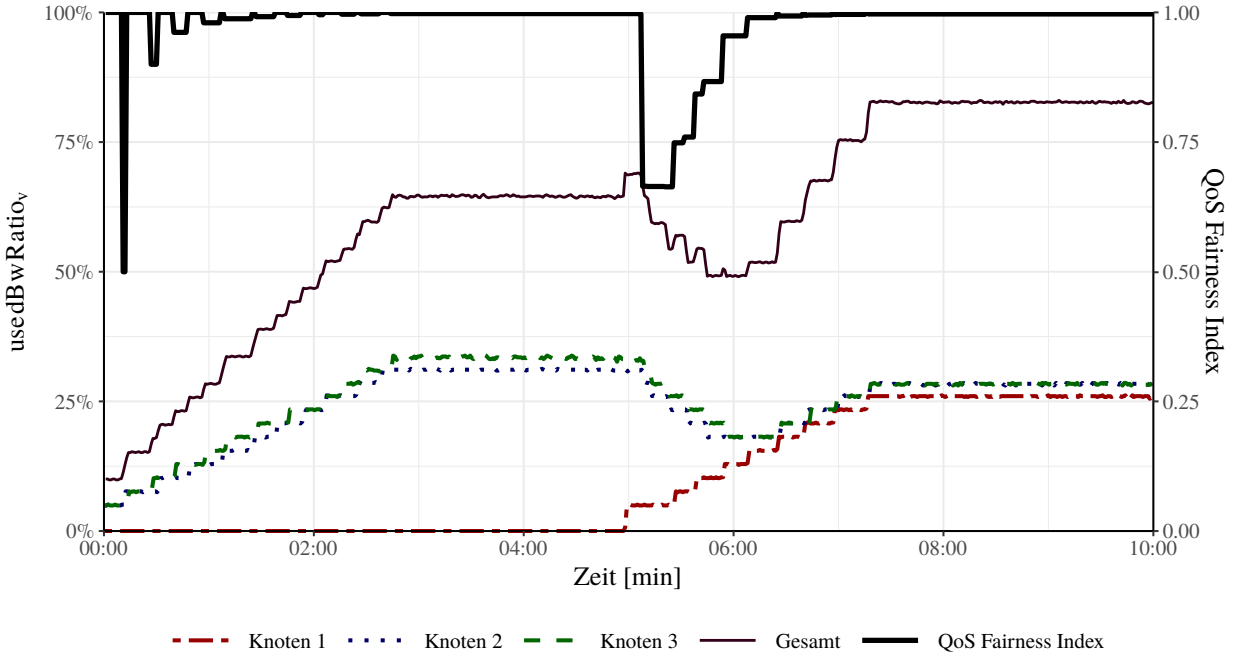


Abbildung 4.16: Genutzte Bandbreite $usedBwRatio_v$ in Experiment 3 und der sich ergebende QoS Fairness Index

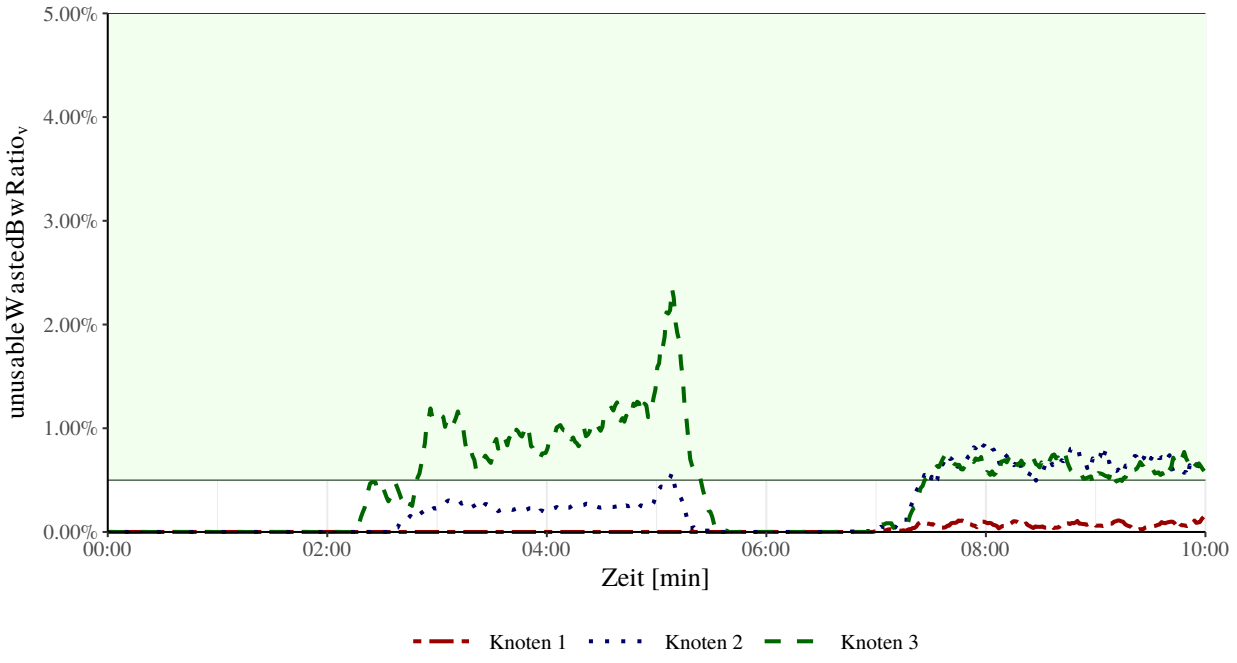


Abbildung 4.17: Unusable Wasted Bandwidth Ratio in Experiment 3

Gesamtbandbreite beträgt etwa 70 %. Die Unusable Wasted Bandwidth Ratio ist bei beiden Knoten über 0, aber bei Knoten 3 erreicht sie mehr als 0,5 %, was dazu führt, dass die Knoten nicht weiter hoch skalieren.

Knoten 1 sendet in den ersten 5 Minuten des Experiments nichts, daher ist die von ihm genutzte Bandbreite 0. Da er nichts sendet, nutzt er seine Basisbandbreite von 5 % nicht aus, sodass Usable Wasted Bandwidth entsteht. Die dritte notwendige Bedingung zum Hochskalieren (s. Algorith-

mus 4.5) ist damit nicht gegeben, sodass der Knoten auf seinem Basislevel verharrt. Außerdem teilt er aus diesem Grund den anderen Knoten sein eigenes Level nicht mit (s. Algorithmus 4.3), was dazu führt, dass diese hoch skalieren dürfen, obwohl einer ihrer Nachbarn auf einem niedrigeren Level ist. Nach 5 Minuten beginnt Knoten 1 nun, die ihm zugewiesene Bandbreite zu nutzen. Damit sind nun alle Bedingungen zum Hochskalieren erfüllt, und er skaliert schrittweise hoch. Der QoS Fairness Index sinkt nun stark auf 0,66, da nun ein Knoten dazu kommt, der auf einem deutlich niedrigen Level ist und daher unfair behandelt wird. Da Knoten 1 nun sein eigenes Level mit jeder Nachricht den anderen Knoten mitteilt, wissen diese aber, dass der Knoten unfair behandelt wird. Um die Fairness möglichst schnell wieder auszugleichen, kommen sie dem neuen Knoten entgegen und skalieren herunter. Verantwortlich ist dafür die dritte hinreichende Bedingung zum Herunterskalieren (s. Algorithmus 4.6). Dass die Knoten 2 und 3 bereits Bandbreite für Knoten 1 frei machen, bevor es zu einer Überlastsituation kommt, ist ein entscheidender Unterschied zu gierigen (engl.: greedy) Skalierungsverfahren. Der in Abb. 4.16 ebenfalls eingezeichnete QoS Fairness Index zeigt, dass dieser Ansatz die Fairness schnell wieder herstellen kann.

Nach ca. 6 Minuten unterscheiden sich die Knoten nur noch um max. ein Level, sodass sie erneut hoch skalieren können. Schließlich verharren Knoten 2 und 3 bei ca. 28 % und Knoten 1 ca. 26 % Bandbreite, sodass insgesamt ca. 82 % Bandbreite genutzt wird. Das Hochskalieren endet hier, da Knoten 2 und 3 den Grenzwert von 0,5 % Unusabled Wasted Bandwidth Ratio überschritten haben (s. Abb. 4.17).

Experiment 4: Nachdem wir das Verhalten des Verfahrens ohne Einfluss durch externen Verkehr untersucht haben, soll nun untersucht werden, wie sich das Verfahren bei externem Verkehr verhält. Dazu wurde das vorige Experiment auf dem 2,4 GHz Kanal 6 (2,437 GHz) wiederholt. Dieser Kanal wurde während des Experiments auch von einem Access Point in der Nähe genutzt, über den typischer Internetverkehr läuft. Internetverkehr ist typischerweise durch Bursts gekennzeichnet, die beim Laden einer Website oder E-Mail entstehen. In diesem Experiment muss das Verfahren daher zeigen, wie es mit externen, d.h. unkooperativen, Knoten klarkommt, die diskontinuierlichen Verkehr erzeugen. Tab. 4.14 fasst die Parameter des Experiments zusammen, die sich bis auf den Kanal nicht von denen im vorigen Experiment unterscheiden.

Tabelle 4.14: Parameter zu Experiment 4

Parameter	Wert
Anzahl Knoten	3
Bandbreitenskalierung	aktiviert, $levels = 20$
Dauer	10 Minuten
Kanal (Frequenz)	6 (2,437 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_v(\forall v \in V)$	55 %

Die im Verlauf des Experiments durch die Knoten genutzte Bandbreite ist in Abb. 4.18 zusammen mit dem QoS Fairness Index aufgetragen. Die genutzte Bandbreite der Knoten verläuft sehr ähnlich wie zuvor. Allerdings hören die Knoten früher auf, hoch zu skalieren, und kommen damit auf weniger als 25 % Bandbreite pro Knoten. Dies liegt offensichtlich daran, dass exter-

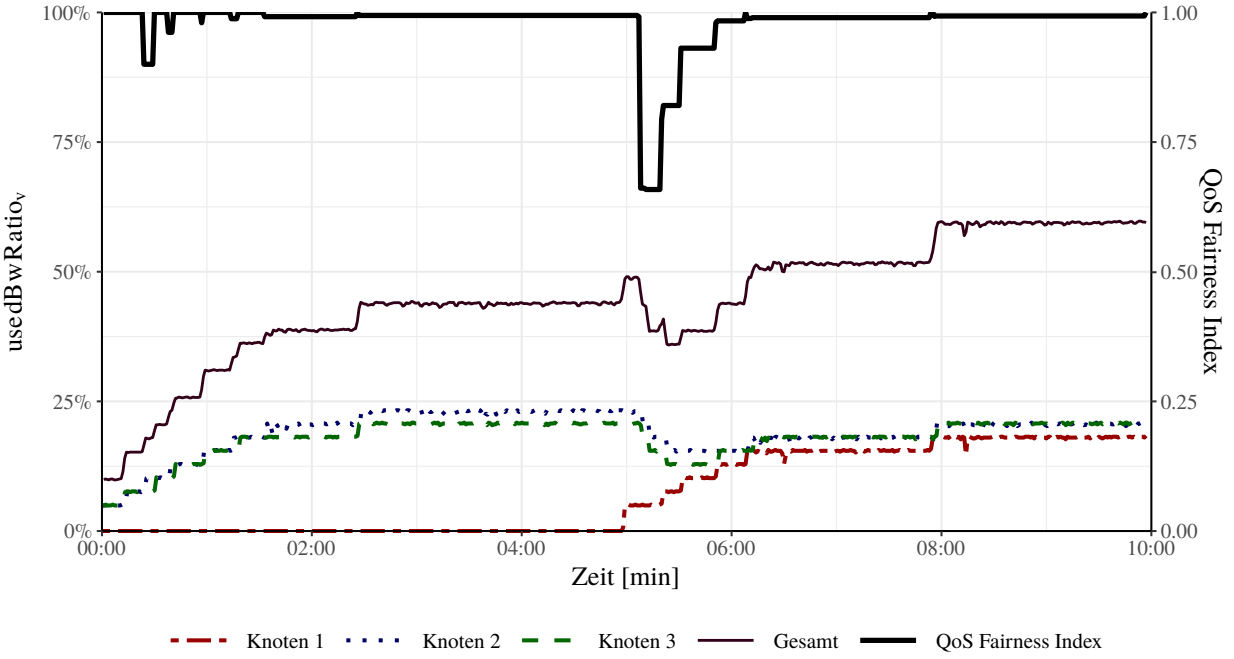


Abbildung 4.18: Genutzte Bandbreite $usedBwRatio_v$ in Experiment 4 und der sich ergebende QoS Fairness Index

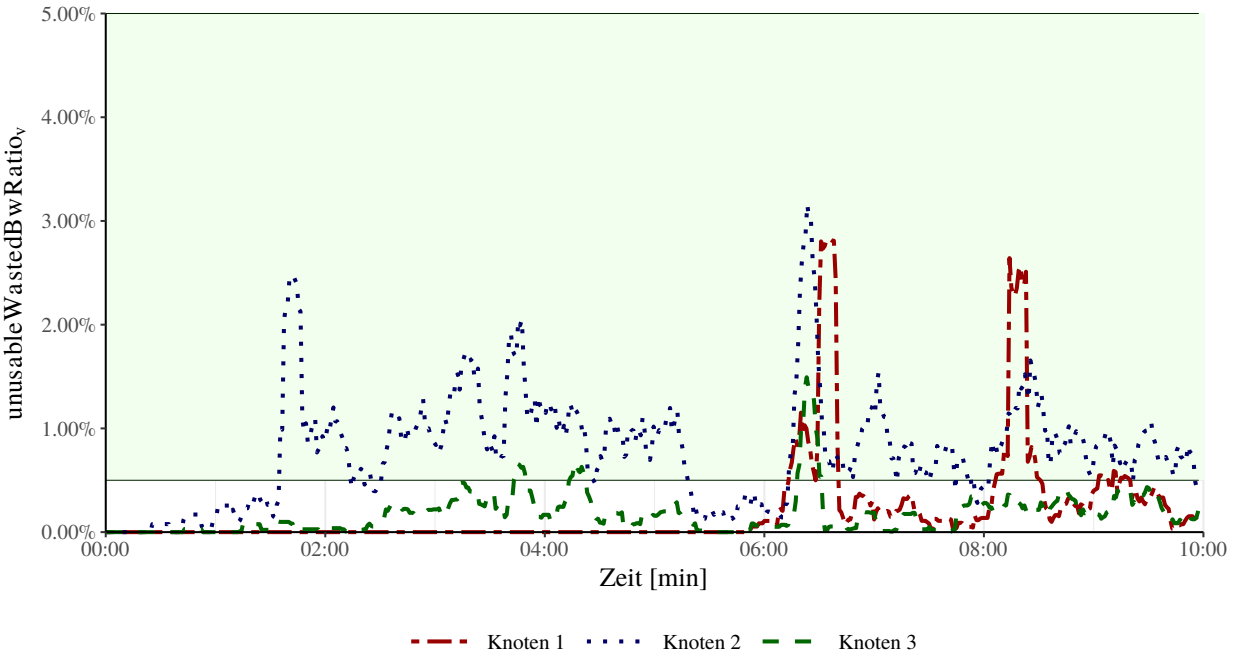


Abbildung 4.19: Unusable Wasted Bandwidth Ratio in Experiment 4

ne Knoten ebenfalls Bandbreite nutzen, und das Zuweisen weiterer Bandbreite an die internen Knoten nur dazu führen würde, dass diese nicht nutzbar ist. Dies zeigt sich bei Betrachtung des Unusable Wasted Bandwidth Ratio in Abb. 4.19. Knoten 2 erreicht bereits nach ca. 1:40 min den Grenzwert von 0,5 %, was dazu führt, dass die Knoten nicht weiter hoch skalieren. Auffällig ist, dass das Unusable Wasted Bandwidth Ratio deutlich stärker schwankt als im vorigen Experiment und auch zu Beginn des Experiments nicht 0 ist. Offensichtlich führt der stark schwanken-

de und von Bursts geprägte externe Verkehr dazu, dass auch die Unusable Wasted Bandwidth Ratio stärker schwankt. Das Verfahren schafft es aber dennoch, dass die Unusable Wasted Bandwidth unter 5 % bleibt und damit immer mindestens 95 % der zugewiesenen Bandbreite nutzbar sind. Die Fairness verhält sich ähnlich wie zuvor und wird auch hier schnell wiederhergestellt, nachdem Knoten 1 beginnt zu senden.

Experiment 5: Nachdem wir gesehen haben, wie unser Verfahren zur Bandbreitenskalierung mit externem Verkehr zurecht kommt, untersuchen wir in diesem Experiment, wie das Verhalten von IEEE 802.11 ohne unsere Bandbreitenskalierung ist. Dabei wird wie zuvor Kanal 6 (2,437 GHz) verwendet, der wieder auch durch externen Verkehr benutzt wird. Die drei internen Knoten haben wie zuvor jeweils eine präferierte Bandbreite von 55 %, was bedeutet, dass es unmöglich ist, dass mehr als ein Knoten seine präferierte Bandbreite nutzen kann. Knoten 1 sendet dabei wieder erst nach 5 Minuten, wohingegen die anderen beiden Knoten direkt mit Senden beginnen. Die Parameter des Experiments fasst Tab. 4.15 zusammen.

Tabelle 4.15: Parameter zu Experiment 5

Parameter	Wert
Anzahl Knoten	3
Bandbreitenskalierung	deaktiviert
Dauer	10 Minuten
Kanal (Frequenz)	6 (2,437 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_v(\forall v \in V)$	55 %

Die Ergebnisse des Experiments zeigt Abb. 4.20. Da keine Bandbreitenskalierung aktiviert ist, versuchen die Knoten 2 und 3 direkt zu Beginn, möglichst je 55 % der Bandbreite zu nutzen. Es gelingt ihnen dabei, ca. 25 % zu nutzen, was ein wenig mehr ist als im Experiment zuvor. Allerdings schwankt die genutzte Bandbreite verglichen mit dem letzten Experiment sehr stark, teilweise sogar um ca. 10 %. Der ebenfalls in Abb. 4.20 eingezeichnete QoS Fairness Index ist größtenteils hoch. Das liegt vor allem auch daran, dass alle Knoten die gleichen QoS-Anforderungen haben und sich in diesem Fall QoS Fairness nicht von Throughput Fairness unterscheidet. Die Fairness schwankt verglichen mit den vorigen Experiment häufiger und ohne erkennbaren Grund. Offensichtlich führt der variierende externe Verkehr nicht nur zu Schwankungen der genutzten Bandbreite, sondern auch der Fairness, da die genutzte Bandbreite zwischen den Knoten unterschiedlich schwankt. Wie im vorigen Experiment fällt der Fairness Index kurz, sobald Knoten 1 hinzukommt. Er erholt sich zwar schneller, sinkt aber kurz vor Minute 9 ohne erkennbaren Grund stark ab. Die Ergebnisse zeigen, dass ohne das Verfahren zur Bandbreitenskalierung die Knoten in einen Bereich gehen, in dem das Medium stark ausgelastet ist, sodass schwer vorherzusehende Schwankungen auftreten. Unser Verfahren schafft es hingegen, die genutzte Bandbreite der internen Knoten auch unter dem Einfluss externer Knoten mit variierendem Verkehr zu stabilisieren. Somit wird die Vorhersagbarkeit und Zuverlässigkeit des Netzwerks erhöht.

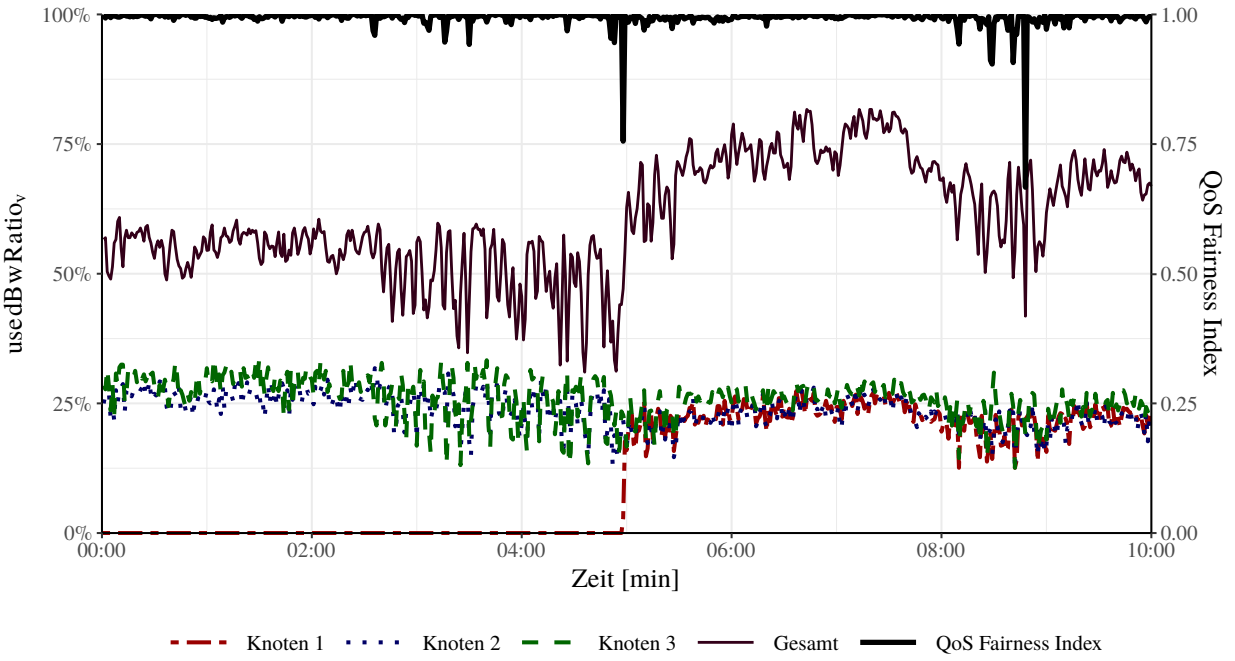


Abbildung 4.20: Genutzte Bandbreite $usedBwRatio_v$ in Experiment 5 und der sich ergebende QoS Fairness Index

Experiment 6: In diesem Experiment untersuchen wir das Verhalten der Bandbreitenskalierung unter dem Einfluss von externem Verkehr genauer, indem wir gezielt externen Verkehr erzeugen. Wir nutzen dazu Kanal 1 (2,412 GHz), auf dem normalerweise bereits externer Verkehr herrscht, allerdings weniger als auf Kanal 6. Nach ca. 4 Minuten starten wir dann den Download einer großen Datei über FTP und einen Videostream auf einem Laptop, der mit einem Access Point auf Kanal 1 verbunden ist. Um den Einfluss des externen Verkehrs besser zu verstehen, zeichnen wir diesmal außerdem den gesamten Verkehr auf dem Kanal während des Experiments mit Wireshark auf. Anschließend berechnen wir die durch externe Knoten genutzte Bandbreite, indem wir die Übertragungszeiten der externen Knoten anhand der Paketgrößen und Datenraten bestimmen und aufsummieren.

Tabelle 4.16: Parameter zu Experiment 6

Parameter	Wert
Anzahl Knoten	3
Bandbreitenskalierung	aktiviert, $levels = 20$
Dauer	10 Minuten
Kanal (Frequenz)	1 (2,412 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_v(\forall v \in V)$	55 %

In Abb. 4.21 ist neben der von den drei internen Knoten genutzten Bandbreite nun auch die von externen Knoten genutzte Bandbreite eingezeichnet. In der Grafik kann man erkennen, dass die internen Knoten teilweise herunter skalieren, wenn Spitzen im externen Verkehr auftreten. Bei Betrachtung des Unusable Wasted Bandwidth Ratios in Abb. 4.22 sieht man, dass hauptsächlich

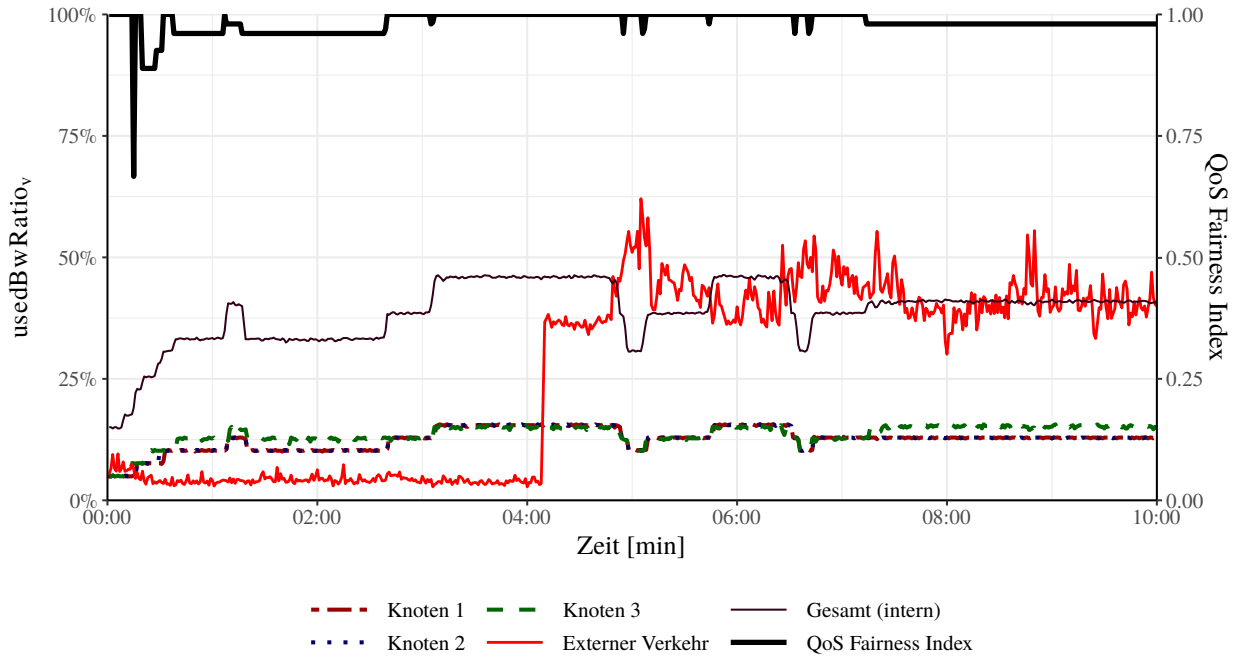


Abbildung 4.21: Genutzte Bandbreite $usedBwRatio_v$ in Experiment 6 und der sich ergebende QoS Fairness Index

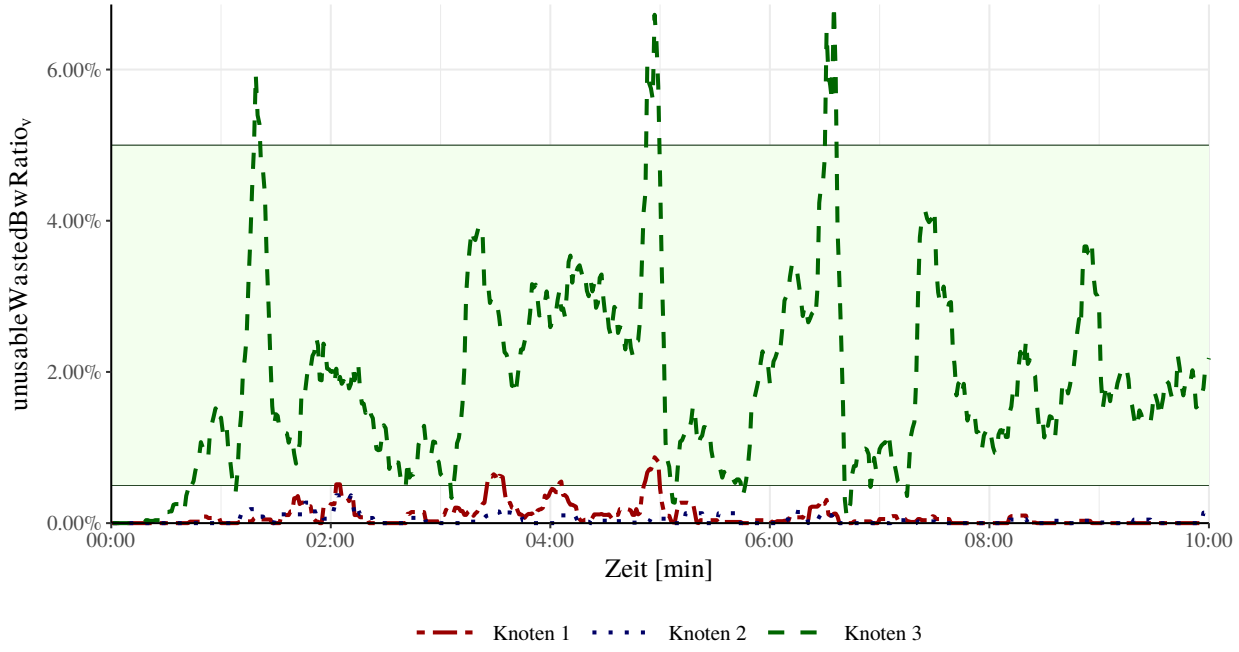


Abbildung 4.22: Unusable Wasted Bandwidth Ratio in Experiment 6

Knoten 3 die ihm zugeordnete Bandbreite nicht vollständig ausnutzen kann. Die Spitzen im externen Verkehr führen zu Spitzen in der Unusable Wasted Bandwidth von Knoten 3, die 5 % übersteigen. Dies sorgt dafür, dass die Knoten daraufhin herunter skalieren. Insgesamt zeigt sich, dass das Verfahren gut mit variierendem externen Verkehr umgehen kann und es schafft, die Unusable Wasted Bandwidth Ratio im Bereich zwischen 0,5 % und 5 % zu halten und die durch die internen Knoten nutzbare Bandbreite weitestgehend stabil zu halten.

Experiment 7: Im letzten Experiment soll das Verhalten der Bandbreitenskalierung in einem Multihop-Netzwerk mit zehn Knoten untersucht werden. Dabei sendet zu Beginn nur ein Knoten und alle drei Minuten kommt ein weiterer Knoten hinzu, bis schließlich alle zehn Knoten senden. Alle Knoten fordern in diesem Experiment eine Basisbandbreite von 5 % und eine präferierte Bandbreite von 40 %. Das Experiment wird auf Kanal 40 (5,200 GHz) ausgeführt, auf dem kaum externer Verkehr herrscht. Die Parameter des Experiments fasst Tab. 4.17 zusammen.

Tabelle 4.17: Parameter zu Experiment 7

Parameter	Wert
Anzahl Knoten	10
Bandbreitenskalierung	aktiviert, $levels = 20$
Dauer	30 Minuten
Kanal (Frequenz)	40 (5,200 GHz)
$requestedBwRatioBase_v(\forall v \in V)$	5 %
$requestedBwRatioPref_v(\forall v \in V)$	40 %

In Abb. 4.23 ist die genutzte Bandbreite der Knoten sowie der QoS Fairness Index aufgetragen. Zu Beginn sendet nur Knoten 1 und skaliert auf ca. 25 % Bandbreite hoch, bis Knoten 2 zu senden beginnt. Weil Knoten 1 nun einen Nachbarn auf einem niedrigeren Level hat, skaliert er herunter, um für diesen Bandbreite frei zu machen. Am QoS Fairness Index ist zu sehen, dass damit auch die Fairness wiederhergestellt wird. Wie zuvor wäre es mit einem kürzeren Kontrollintervall möglich, dass das Verfahren deutlich schneller reagiert. Schließlich skalieren beide Knoten wieder zusammen hoch, bis der nächste Knoten dazukommt. Sobald 6 Knoten senden, kann man beobachten, dass die Knoten nicht mehr so hoch skalieren wie zuvor. Dies liegt daran, dass die Bandbreite zwischen den Knoten aufgeteilt werden muss. Da es sich aber um ein Multihop-Szenario handelt, kommt es auch vor, dass Knoten soweit von einander entfernt platziert sind, dass sie zeitgleich senden können, ohne dass es zu Kollisionen kommt (Space Division Multiple Access). Durch diesen Effekt ist die Summe der genutzten Bandbreiten am Schluss des Experiments über 100 %, was in einem Singlehop-Experiment nicht möglich wäre.

Als Knoten 6 zu senden beginnt, kann man außerdem beobachten, dass Knoten 5 nicht so schnell und nicht so weit herunter skaliert wie die anderen Knoten. Dies lässt sich auch durch das Multihop-Szenario erklären: Knoten 6 und Knoten 5 sind soweit voneinander entfernt, dass sie nicht direkt miteinander kommunizieren können. Sobald Knoten 6 zu senden beginnt, bemerken dessen Nachbarn das und skalieren herunter. Nachdem sie zwei Levels herunter skaliert haben, sind sie zwei Levels tiefer als ihre zwei Hop von Knoten 6 entfernten Nachbarn. Das führt dazu, dass diese wiederum herunter skalieren. Es skalieren also nicht nur die direkten Nachbarn des neuen Knotens herunter, sondern auch die weiter entfernten Nachbarn, wobei sie umso weniger herunter skalieren, je weiter sie entfernt sind. Da die Interferenzreichweite bei drahtlosen Netzwerken i.d.R. größer ist als die Kommunikationsreichweite, muss der neue Knoten sich das Medium auch mit Knoten teilen, die weiter als einen Hop entfernt sind. Umso mehr Hops ein Knoten entfernt ist, umso unwahrscheinlicher wird es aber, dass es zu Interferenzen mit dem neuen Knoten kommen kann. Das Verfahren berücksichtigt dies, indem es pro Hop ein Level Unterschied gestattet.

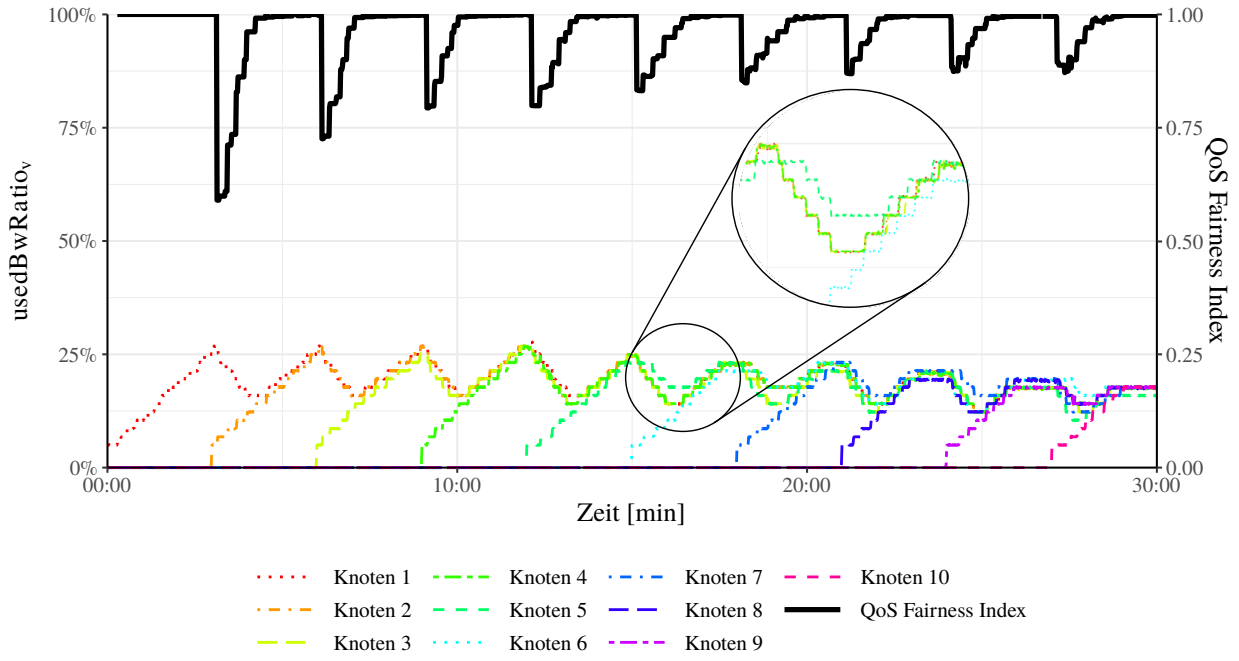


Abbildung 4.23: Genutzte Bandbreite $usedBwRatio_v$ in Experiment 7 und der sich ergebende QoS Fairness Index

4.4.6 Zusammenfassung

Wir haben uns in diesem Kapitel mit fairer Bandbreitenskalierung beschäftigt. Dabei haben wir zunächst unterschiedliche Fairness-Definitionen betrachtet und schließlich für unsere Bedürfnisse mit QoS Fairness eine eigene Fairness-Definition entwickelt. Um Fairness messbar zu machen, haben wir einen QoS Fairness Index definiert, der auf dem weit verbreiteten Fairness Index von Raj Jain beruht. Schließlich haben wir ein dynamisches, kooperatives Skalierungsverfahren entwickelt. Das Verfahren skaliert die Bandbreite der Knoten dynamisch zwischen der angeforderten Basisbandbreite und der präferierten Bandbreite. Wir haben Bedingungen für das Hoch- und Herunterskalieren definiert, welche dafür sorgen, dass das Verfahren QoS fair ist und die Knoten sich kooperativ verhalten. Die durch das Verfahren zugewiesene Bandbreite wird mit Hilfe des Time Token Bucket-Verfahrens kontrolliert, wobei das Refill-Intervall dynamisch angepasst wird. Außerdem wird das Time Token Bucket-Verfahren benutzt, um die Auslastung des drahtlosen Mediums zu bestimmen und Überlastsituationen zu vermeiden. Anhand einer Implementierung als WiPS-Schicht wurde das Verfahren in Experimenten ausführlich auf einem Testbett aus APU2 Knoten evaluiert. Dabei zeigt sich, dass das Verfahren Fairness nach einer kurzen Stabilisierungsphase erzielt und anschließend erhalten kann. Auch unter variierenden Einflüssen durch externe Knoten oder beim Hinzukommen neuer Knoten schafft es das Verfahren, die Fairness schnell wiederherzustellen. Verglichen mit Standard IEEE 802.11 erreicht das Verfahren eine höhere und stabilere Fairness. Außerdem schafft es das Verfahren auch unter variierenden externen Einflüssen, die genutzte Bandbreite zu stabilisieren. Durch den Einsatz des Verfahrens wird wettbewerbsbasierter Verkehr stabiler und damit vorhersagbarer und zuverlässiger.

4.5 Clustering

In Kapitel 3.4 wurden Clustering-Verfahren für TDMA-basierte Netzwerke besprochen und mit Heterogeneous Network Clustering (HNC) ein Clustering-Verfahren für das Bilden einer verteilten Service Registry vorgestellt. Für wettbewerbsbasierte Kommunikationssysteme bieten Clustering-Verfahren die gleichen Vorteile. Die theoretischen Grundlagen und Konzepte unterscheiden sich nicht und werden daher hier nicht wiederholt. Vielmehr wird in diesem Kapitel zunächst in Kapitel 4.5.1 ein anderes Anwendungsszenario vorgestellt, nämlich die Überwachung von Güterzügen. Anschließend betrachtet Kapitel 4.5.2 existierende Verfahren, die für dieses Anwendungsszenario in Frage kommen. In Kapitel 4.5.3 wird mit *Link Detection and Line Topology Establishment (LDLTE)* ein wettbewerbsbasiertes Verfahren vorgestellt, welches speziell für die Anforderungen dieses Anwendungsszenarios entwickelt wurde. Implementierung und Evaluierung des Verfahrens beschreibt Kapitel 4.5.4 und Kapitel 4.5.5 fasst das Kapitel zusammen.

4.5.1 Anwendungsszenario: Überwachung von Güterzügen

Die Bosch Engineering GmbH (BEG) entwickelt mit dem Projekt *Asset Monitoring for Rail Applications (AMRA)* eine Lösung zur Überwachung von Güterzügen. Das Projekt sieht vor, dass die Waggons von Güterzügen mit unterschiedlichen Sensoren ausgestattet werden, die drahtlos vernetzt und per GSM mit dem Internet verbunden sind. Mit den Sensoren soll der Status der Waggons und deren Ladung bestimmt werden, beispielsweise die Temperatur in Kühlwagen oder ob Türen geöffnet sind. Außerdem dienen GPS-Sensoren der Positionsbestimmung. Mittels Sensoren, die Stöße und Vibrationen messen, können Schäden an den Wagen frühzeitig erkannt und Wartungsarbeiten rechtzeitig geplant werden. Die ermittelten Daten müssen periodisch ermittelt und an ein zentrales Kontrollsystem weitergeleitet werden.

Die Weiterleitung der Daten soll einer hierarchischen Topologie folgen, die in Abb. 4.24 skizziert ist. Auf einem Waggon i werden die Sensoren $S_{i,j}$ platziert. Außerdem wird jeder Waggon mit einem *Local Data Collector* LDC_i ausgerüstet, der über Bluetooth LE oder Kabel mit den Sensoren in diesem Waggon verbunden ist und deren Daten sammelt und aggregiert. Falls verfügbar, leiten LDCs ihre Daten per Bluetooth LE an den *Global Data Collector (GDC)* weiter. Die Verbindung zwischen den LDCs und dem GDC erfolgt entlang einer Linientopologie von Waggon zu Waggon bis zum GDC. Beim GDC handelt es sich um einen Knoten, der beispielsweise auf der Lok montiert ist und die Daten des ganzen Zuges sammelt, aggregiert und per GSM an den Remote Data Collector (RDC) weiterleitet. Der RDC ist eine zentrale Leitstelle, bei der die Daten von allen Zügen zusammenlaufen und überwacht werden. Falls von einem LDC keine Verbindung zum GDC besteht, beispielsweise weil Waggons von der Lok abgekoppelt worden sind, können LDCs auch direkt per GSM mit dem RDC kommunizieren. Außerdem kann ein LDC die Rolle eines GDC für einen Zugabschnitt übernehmen.

Ein Hauptaugenmerk in diesem Anwendungsszenario ist der Energiebedarf der LDCs und Sensoren. Da auf den Waggons in der Regel keine externe Stromversorgung verfügbar ist, müssen diese mit Batterien betrieben werden. Da die Batterien eine lange Lebensdauer von mehreren Jahren haben sollen, muss mit der vorhandenen Energie extrem sparsam umgegangen werden. Damit die lokale Kommunikation möglichst wenig Energie benötigt, wurde Bluetooth Low Energy 4.0 [Blu10] als Kommunikationstechnologie ausgewählt. Um möglichst wenig Energie durch Idle Listening zu verschwenden, verfolgt das System einen Duty Cycling-Ansatz, der zwi-

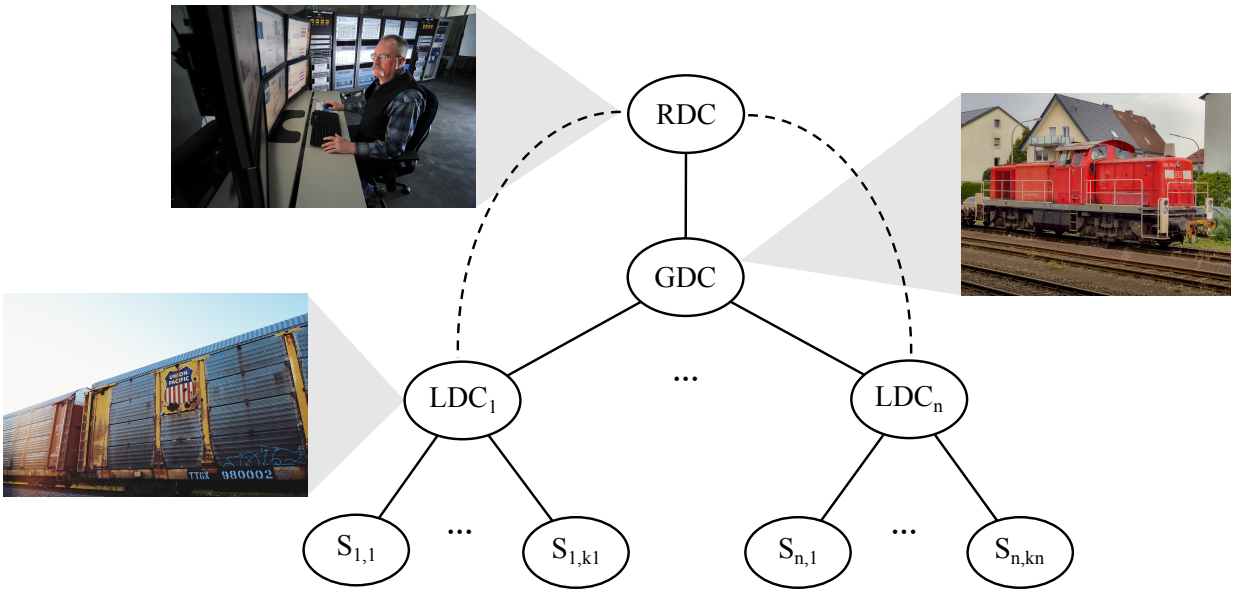


Abbildung 4.24: Hierarchische Topologie eines AMRA-Netzwerks

schen aktiven Phasen und inaktiven Phasen wechselt. Kommuniziert wird nur in den aktiven Phasen, wohingegen in den inaktiven Phasen der Transceiver zum Energiesparen abgeschaltet werden kann. Für die Positionsbestimmung sind die LDCs ohnehin mit GPS ausgestattet, sodass diese Technologie auch zur Zeitsynchronisation genutzt wird.

Aus der Anforderungsanalyse wurde das AMRA-System in folgende Funktionalitäten aufgeteilt:

- *Lokale Erfassung von Sensordaten:* Der LDC sammelt und aggregiert periodisch die Werte, welche die mit ihm verbundenen Sensoren ermittelt haben.
- *Weiterleiten von Daten:* Die vom LDC gesammelten und aggregierten Daten werden periodisch zum GDC weitergeleitet. Der GDC wiederum sammelt und aggregiert die Daten von mehreren LDCs und leitet sie an den RDC weiter.

Für das lokale Erfassen und Weiterleiten der Daten muss der LDC unterschiedliche Topologien nutzen. Zum einen muss der LDC die lokalen Sensoren erkennen und mit ihnen ein Bluetooth Piconet bilden, in dem der LDC als Master agiert. Darüber hinaus müssen die LDCs miteinander kooperieren, um eine Verbindung zum GDC herzustellen. Da die Knoten auf die Waggons eines Güterzuges montiert werden, liegt es nahe, eine Linientopologie zwischen den LDCs zum GDC aufzubauen. In manchen Fällen kann es vorkommen, dass es nicht möglich oder sinnvoll ist, alle LDCs eines Zuges mit dem GDC in der Lok zu verbinden, beispielsweise weil der Zug sehr lang ist oder einzelne Waggons nicht über einen LDC verfügen. Daher ist es auch möglich, dass ein LDC die Rolle eines GDC für einen Zugabschnitt übernimmt. Die Topologie des Zuges ist dann in mehrere *Linientopologieabschnitte* eingeteilt. Die Gruppierung mehrerer LDCs zu einem Linientopologieabschnitt kann als Clustering-Problem gesehen werden, wobei der Cluster Head (CH) die Rolle eines GDC übernimmt.

Zur Einteilung der LDCs in Cluster bzw. zum Bilden von Linientopologieabschnitten benötigt das AMRA-System zwei weitere Funktionalitäten:

- *Linkerkennung:* Die Existenz und Qualität von Kommunikationslinks muss bestimmt werden.

- *Topologiebildung:* Clustering und Verbinden der LDCs innerhalb eines Clusters in einem Linientopologieabschnitt unter Verwendung der als zuverlässig erkannten Kommunikationslinks.

Um Energie einzusparen, sollten aktive Phasen möglichst kurz sein. In den genannten Funktionalitäten wird immer nur eine Teilmenge der Knoten miteinbezogen. Beispielsweise kommunizieren während der lokalen Erfassung von Sensordaten die LDCs mit den Sensorknoten, wohingegen zum Weiterleiten der Daten nur die LDCs und GDCs beteiligt sind. Das Duty Cycling unterscheidet daher zwischen aktiven Phasen für die einzelnen Funktionalitäten.

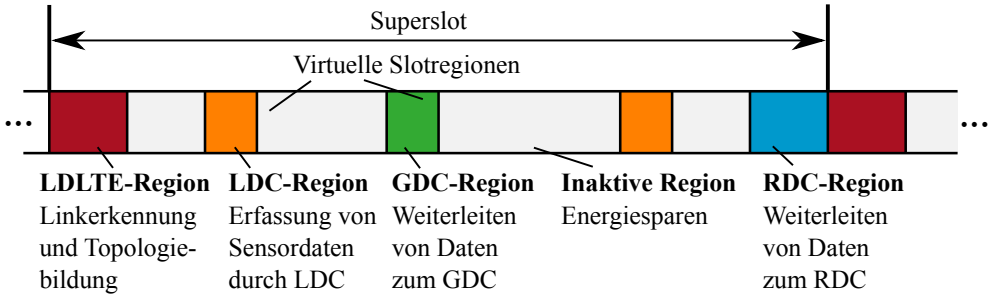


Abbildung 4.25: Duty Cycle mit virtuellen Slotregionen, die den einzelnen Funktionalitäten zugeordnet sind, und inaktiven Regionen.

Abb. 4.25 zeigt beispielhaft einen Duty Cycle mit den unterschiedlichen aktiven Phasen und inaktiven Phasen dazwischen. Zunächst wird die Zeit in eine Sequenz aus Superslots mit fester Länge eingeteilt. Innerhalb des Superslots werden virtuelle Slotregionen für die unterschiedlichen Funktionalitäten zugeordnet. Im Beispiel beginnt ein Superslot mit einer LDLTE-Region, welche den Funktionalitäten Linkerkennung und Topologiebildung zugeordnet ist. Außerdem sind zwei LDC-Regionen vorgesehen, in denen die LDCs die lokale Erfassung von Sensordaten durchführen. In der GDC-Region leiten LDCs ihre gesammelten Daten an ihren GDC weiter und in der RDC-Region werden sie von dort an das RDC weitergeleitet. Durch die Strukturierung der Zeit über virtuelle Slotregionen können die einzelnen Funktionalitäten flexibel eingeplant werden. Zwischen den aktiven Regionen werden inaktive Phasen eingeplant, in denen Energie eingespart wird, indem die Transceiver abgeschaltet oder in den Energiesparmodus versetzt werden können. Da in jeder virtuellen Slotregion nur die zugeordneten Funktionalitäten ausgeführt werden, können Knoten, die an dieser Funktionalität nicht beteiligt sind, während dieser Phase ebenfalls inaktiv bleiben und ihren Transceiver abschalten. Außerdem verhindert die zeitliche Trennung der Funktionalitäten, dass Nachrichten der unterschiedlichen Funktionalitäten miteinander kollidieren.

In diesem Kapitel konzentrieren wir uns auf die Funktionalität der Topologiebildung. Bei der Topologiebildung werden die Knoten in Linientopologieabschnitten gruppiert. Dies entspricht einem Clustering-Problem. Wir betrachten daher zunächst existierende Clustering-Verfahren, die für dieses Anwendungsszenario in Frage kommen. Anschließend stellt Kapitel 4.5.3 das LDLTE-Protokoll vor, welches speziell für die Funktionalitäten der Linkerkennung und Topologiebildung im AMRA-Projekt entwickelt wurde.

4.5.2 Stand der Technik

In Kapitel 3.4.1 wurden bereits existierende Clustering-Verfahren besprochen, darunter der Linked Clustering Algorithm (LCA) und Max-Min-D. Außerdem wurde in Kapitel 3.2.1.2 die Funk-

tionalität der Topologieerkennung in existierenden Clustering-Protokollen am Beispiel von Max-Min-D diskutiert. Dabei stellte sich heraus, dass existierende Verfahren oft die Zuverlässigkeit der Links nur unzureichend bestimmen. Gerade beim Bilden einer Linientopologie ist es wichtig, nur zuverlässige Links auszuwählen, da die Reparatur der Linientopologie meist aufwendig ist. Ebenso wichtig ist, dass nur bidirektionale Links ausgewählt werden, da sonst keine Acknowledgements möglich sind. Auch hier haben viele existierende Verfahren Schwächen.

In der Literatur werden Linientopologien oft als Trivialfall behandelt. Auch wenn Knoten räumlich entlang einer Linie angeordnet sind, existieren allerdings meist mehr Links, als die reine Linientopologie. Daher müssen zum Bilden einer Linientopologie von den vorhandenen Links jene ausgewählt werden, die zusammen eine Linie bilden. Weiterhin muss die Overlay-Struktur repariert werden, wenn sich die gewählten Links verändern. Berücksichtigt man diese Punke, ist das Problem keineswegs trivial. Ein ähnliches Problem betrachten hierarchische Routing-Verfahren wie RPL [Hui+12]. Dabei wird statt einer Linie eine Baumstruktur als Overlay-Struktur gebildet. Letztlich ist eine Linie ein Spezialfall eines Baumes. Hierarchische Routing-Verfahren haben ähnliche Vorteile wie Verfahren, die Linientopologien bilden: Sobald die Overlay-Struktur gebildet ist, wird das Routing stark vereinfacht. Es reicht, dass jeder Knoten den nächsten Hop in Richtung der Wurzel kennt, damit alle Knoten zur Wurzel routen können. Dies ist besonders in Anwendungen hilfreich, bei denen alle Knoten in Richtung einer Datensenke senden. In einer Linientopologie ist das Routing zu einer Senke ebenso sehr einfach: Jeder Knoten muss lediglich den nächsten Hop in Richtung der Senke kennen. Nicht nur die Vorteile sind ähnlich, auch die Herausforderungen: Genau wie Verfahren, die Linientopologien bilden, müssen hierarchische Routing-Verfahren Funktionalität zum Aufbauen und Reparieren der Overlay-Struktur enthalten.

Es existieren wenige Clustering-Verfahren, die Linientopologien bilden, darunter Pegasis [LR02] und Track-sector clustering [GLP09]. Diese setzen allerdings globales Wissen über die Knotenpositionen voraus. Im Fall des beschriebenen Anwendungsszenarios zur Überwachung von Güterzügen ist dieses nicht vorhanden. Weiterhin berücksichtigen diese Verfahren die Zuverlässigkeit der Links nicht. Fällt ein gewählter Link weg, beispielsweise weil ein Knoten ausfällt, erfordert dies bei beiden Verfahren eine Neuorganisation, da keine Funktionalität zur Reparatur der Topologie vorgesehen ist.

4.5.3 Link Detection and Line Topology Establishment (LDLTE)

Mit Link Detection and Line Topology Establishment (LDLTE) wurde ein Protokoll zur Bildung und Reparatur von Linientopologien entwickelt, das zuerst in [Sef+18] veröffentlicht wurde. Das Verfahren wurde im Kontext des AMRA-Projekts für das Anwendungsszenario der Überwachung von Güterzügen entwickelt (s. Kapitel 4.5.1). Neben dieser Anwendung kann das Protokoll aber in allen anderen Anwendungen eingesetzt werden, in denen Linientopologien sinnvoll sind. So gibt es auch Straßenfahrzeuge, die sich in einer räumlichen Linie anordnen, weil sie in Konvois fahren. Neben militärischen Konvois sind auch Konvois aus LKWs, Bussen oder PKWs denkbar. Ähnlich wie bei der Schiene gibt die Straße auf natürliche Weise die räumliche Anordnung entlang einer Linie vor, sodass eine Linientopologie für die Kommunikation naheliegend ist. Ein weiteres Beispiel, bei dem dies der Fall sein kann, sind Produktionslinien. Da die Produktion entlang einer Linie organisiert ist, sind die kommunizierenden Maschinen räumlich entlang einer Linie angeordnet. In diesem Fall wird die Linie meist durch ein Fließband gebildet. In den genannten Anwendungsszenarien ist eine Linientopologie zur Kommunikation nicht

nur durch die räumliche Anordnung naheliegend, sondern auch auf Grund der Funktionalitäten, die diese Anwendungen erfordern. Das Aufsplitten oder Verbinden zweier Konvois oder Zügen wird durch Teilen bzw. Verbinden der Linientopologie unterstützt. Das Einteilen in Zug- oder Konvoiabschnitte wird durch das Clustern der Linie in Linienteilabschnitte ermöglicht. Die Kommunikation zwischen den Knoten entlang der Linie wird durch Routing entlang der Linie vereinfacht.

LDLTE wurde für den Bluetooth Low Energy (BLE) 4.0 Standard entwickelt, der keine Mesh-Funktionalität vorsieht. Eine Multihop-Topologie mit mehr als zwei Hops Netzdurchmesser in einem Bluetooth Piconetz zu verbinden ist nicht möglich. Stattdessen nutzt LDLTE die Advertising-Pakete des BLE-Standards zur Kommunikation und übernimmt die Vernetzung der Knoten selbst. Im klassischen Bluetooth erfolgt der Medienzugriff basierend aus einer Kombination aus Frequency Hopping (FDMA) und CSMA/CA für Datenübertragungen und TDMA für die Übertragung von Audiodaten. Das Versenden von Advertising-Paketen bei BLE erfolgt dagegen rein wettbewerbsbasiert. Deshalb ist LDLTE, im Gegensatz zum in Kapitel 3.4.2 beschriebenen HNC, ein Clustering-Verfahren für wettbewerbsbasierte Kommunikationssysteme.

4.5.3.1 Dynamische Linkerkennung

LDLTE setzt kein Protokoll zur Topologieerkennung voraus, sondern enthält selbst Funktionalität zur Erkennung und Bewertung der vorhandenen Kommunikationslinks. Das Protokoll ist grundsätzlich ähnlich dem in Kapitel 4.2.2 beschriebenen Verfahren ATDP4W. Es wird daher hier nur abstrakt beschrieben, Details finden sich in [Sef+18].

Durch die wettbewerbsbasierte Kommunikation über Advertising-Pakete kann es zu sporadischen Paketverlusten durch Kollisionen kommen. Daher können Links nicht basierend auf Einzelbeobachtungen bewertet werden. Stattdessen ist es notwendig, mehrere Beobachtungen statistisch auszuwerten. LDLTE sendet dazu periodisch *LinkDet*-Nachrichten per Broadcast, anhand deren Empfang die Links bewertet werden. Um die Wahrscheinlichkeit für wiederholte Kollisionen zu vermeiden, variiert das Intervall zwischen zwei *LinkDet*-Nachrichten um eine zufällige Verzögerung. Genau wie bei ATDP4W enthalten die Nachrichten eine Sequenznummer und es gibt ein maximales Intervall zwischen zwei Nachrichten, sodass verloren gegangene Nachrichten und gebrochene Links erkannt werden können. In LDLTE tauschen die Knoten die erkannten Links allerdings nicht wie bei ATDP und ATDP4W netzweit aus. Stattdessen ist in jeder *LinkDet*-Nachricht eine Liste der Nachbarknoten enthalten, von denen eine stabile Verbindung zu diesem Knoten besteht[5]. Anhand dieser Liste können die Nachbarn erkennen, welche der von ihnen erkannten stabilen Links auch in die Gegenrichtung stabil und damit bidirektional sind. Beispielsweise auf Grund von unterschiedlichem Ladestand der Batterien können Links asymmetrisch sein. Eine mit Acknowledgements bestätigte Kommunikation ist aber nur über bidirektionale Links möglich. Die als unidirektional erkannten Links werden von LDLTE daher bei der Topologiebildung nicht berücksichtigt.

Im Anwendungsszenario der Überwachung von Güterzügen sind die Knoten zwar fest auf den Waggons montiert. Waggons werden aber öfter von einem Zug an- oder abgekoppelt und zu neuen Zügen zusammengestellt. Außerdem können Knoten ausfallen, wenn deren Batterie leer wird oder eine mechanische Beschädigung auftritt. Darüber hinaus beeinflussen externe Einflüsse wie Wetter oder Tunnel die Qualität der Links. Die Links zwischen den Knoten können

[5] Auf Grund der geringen Rahmengröße von BLE Advertising-Paketen werden bei mehr als vier Nachbarknoten mit stabilem Link nur die vier Nachbarknoten aufgeführt, zu denen die stärksten Links bestehen. Zur Bildung einer Linientopologie sind die stärksten Links i.d.R. ausreichend.

sich also stetig verändern, daher muss die Linkerkennung dynamisch sein und Veränderungen erkennen. Dies unterscheidet die Topologieerkennung in LDLTE von ATDP4W. Die Topologieerkennung in ATDP4W terminiert mit einem netzweit konsistenten Ergebnis. LDLTE läuft dagegen kontinuierlich und tauscht die Topologie nicht netzweit aus, sodass es nicht notwendig ist, die Konsistenz sicherzustellen.

4.5.3.2 Bildung und Reparatur einer Linientopologie

Aus den als zuverlässig erkannten bidirektionalen Links bildet LDLTE ein Overlay-Netzwerk einer Linientopologie. Diese ist unterteilt in Cluster, die Abschnitte der vollständigen Linientopologie darstellen. Zu Beginn bildet jeder Knoten sein eigenes Cluster und ist somit Cluster Head (CH). Im Anwendungsszenario der Überwachung von Güterzügen bedeutet dies, dass jeder LDC direkt per GSM mit dem RDC kommuniziert. Dies muss möglich sein, da Waggons auch einzeln abgestellt werden können. Um Energie zu sparen, sollen die Waggons sich aber wenn möglich untereinander per BLE vernetzen und möglichst wenig per GSM kommunizieren. Sobald die Linkerkennung stabile bidirektionale Links zu einem Nachbarknoten erkennt, beginnt daher die Verknüpfung von mehreren Knoten zu einer Linientopologie. Zum Bilden einer Linientopologie muss sich ein Knoten mit einem anderen Knoten verbinden und so ein Cluster bzw. einen Linientopologieabschnitt aus zwei Knoten bilden. Um größere Linientopologieabschnitte zu bilden, können sich mehrere solcher Abschnitte miteinander verbinden. Andersherum können Linientopologieabschnitte auch aufgeteilt werden. Im Anwendungsszenario der Überwachung eines Güterzuges kann dies notwendig werden, wenn Wagen abgekoppelt werden. Außerdem kann es vorkommen, dass ein Knoten ausfällt, weil seine Batterie leer oder der Knoten beschädigt wird. Für solche Fälle unterstützt LDLTE das Auftrennen von Linientopologieabschnitten.
Die einzelnen Fälle beim Bilden und Reparieren von Linientopologien werden im Folgenden detailliert beschrieben. Dabei bezeichnen wir die Topologie, die alle Knoten eines Netzwerks (z.B. Zuges) miteinander verbindet, als *Linientopologie*. Teile davon bezeichnen wir als *Linientopologieabschnitt* (engl.: sub-line topology), diese entsprechen den Clustern. Ein Linientopologieabschnitt kann aus einem oder mehreren Knoten bestehen, die in einer Linie aus Links miteinander verbunden sind. Wir führen folgende formalen Definitionen ein:

- $Sub(v_k) = (v_1, v_2, ..., v_k, ..., v_i)$ ist eine geordnete Liste von Knoten-IDs, die zum Linientopologieabschnitt gehören, zu dem der Knoten v_k gehört. In diesem Linientopologieabschnitt hat der erste Knoten v_1 die Rolle des CH, v_i bildet das Ende des Abschnitts.
- $|Sub(v_k)| = i$ gibt die Anzahl Knoten an, die zum Linientopologieabschnitt des Knotens v_k gehören.
- $Pos(v_k)$ gibt die Position des Knotens v_k in seinem Linientopologieabschnitt an.

Da initial alle Knoten ihr eigenes Cluster bilden bestehen alle Linientopologieabschnitte aus nur einem Knoten, sodass für alle Knoten $v_k \in V$ gilt:

$$Sub(v_k) = (v_k) \tag{4.43}$$
$$|Sub(v_k)| = 1 \tag{4.44}$$
$$Pos(v_k) = 1 \tag{4.45}$$

Verknüpfen von Linientopologieabschnitten Zwei Linientopologieabschnitte können sich miteinander verknüpfen, um einen größeren Linientopologieabschnitt zu bilden. Erkennt die Link-

erkennung eines Knotens v_{req} einen neuen Nachbarknoten v_{rsp}, zu dem ein stabiler bidirektionaler Link besteht, so wird geprüft, ob sich die beiden Linientopologieabschnitte miteinander verbinden können. Die Verknüpfung der beiden Abschnitte wird durchgeführt, falls folgende Bedingungen erfüllt sind:

1. Der Knoten v_{req} ist entweder der erste oder letzte Knoten in seinem Linientopologieabschnitt:
   $Pos(v_{req}) = 1 \vee Pos(v_{req}) = |Sub(v_{req})|$
2. Der Knoten v_{rsp} ist entweder der erste oder letzte Knoten in seinem Linientopologieabschnitt:
   $Pos(v_{rsp}) = 1 \vee Pos(v_{rsp}) = |Sub(v_{rsp})|$
3. Der Knoten v_{rsp} ist noch nicht Teil des Linientopologieabschnitts von v_{req}:
   $v_{rsp} \notin Sub(v_{req})$
4. Der aus der Verknüpfung der beiden Linientopologieabschnitte resultierende Linientopologieabschnitt überschreitet die konfigurierbare Maximalgröße *maxSize* nicht:
   $|Sub(v_{req})| + |Sub(v_{rsp})| \leq maxSize$

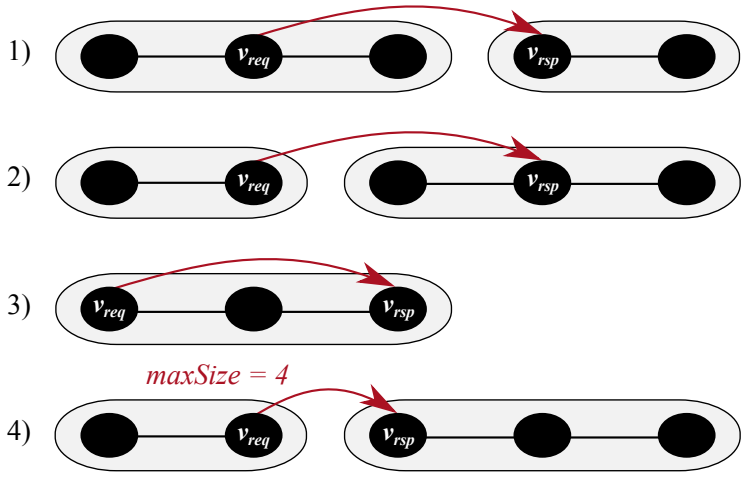


Abbildung 4.26: Gegenbeispiele, in denen die Bedingungen zum Verbinden zweier Linientopologieabschnitte nicht gegeben sind. In Beispiel n) ist die Bedingung n) nicht erfüllt.

Abb. 4.26 illustriert Gegenbeispiele, in denen je eine der Bedingungen verletzt ist. Die ersten drei Bedingungen stellen sicher, dass die resultierende Topologie weiterhin ein Linientopologieabschnitt ist. Sie verhindern, dass eine Baumstruktur gebildet wird oder Schleifen entstehen. Die vierte Bedingung begrenzt die Größe der Linientopologieabschnitte auf ein konfigurierbares Maximum. Sehr große Linientopologieabschnitte sind instabiler, da häufiger Änderungen an den verwendeten Links auftreten können, die eine Reparatur erforderlich machen. Außerdem steigt die Menge an weiterzuleitenden Daten mit der Anzahl Knoten. Über den Parameter *maxSize* kann ein für die Anwendung geeignetes Maximum definiert werden.

Der Knoten v_{req}, der die Verknüpfung initiiert, kennt seinen eigenen Linientopologieabschnitt $Sub(v_{req})$ und seine Position $Pos(v_{req})$. Damit kann er die erste und dritte Bedingung lokal prüfen. Zur Überprüfung der zweiten und vierten Bedingung fehlt ihm $|Sub(v_{rsp})|$ und $Pos(v_{rsp})$, diese sind nur v_{rsp} bekannt. Daher sendet v_{req} wie in Abb. 4.27 gezeigt eine *TopEstJoinRequest*-Nachricht an v_{rsp}. In dieser Nachricht ist $Sub(v_{req})$ enthalten. Damit kann v_{rsp} nicht nur die vierte Bedingung prüfen, sondern auch den aus der Verknüpfung resultierenden Linientopologieabschnitt ermitteln. Ist die zweite und vierte Bedingung erfüllt, so bestätigt v_{rsp} die Verknüpfung

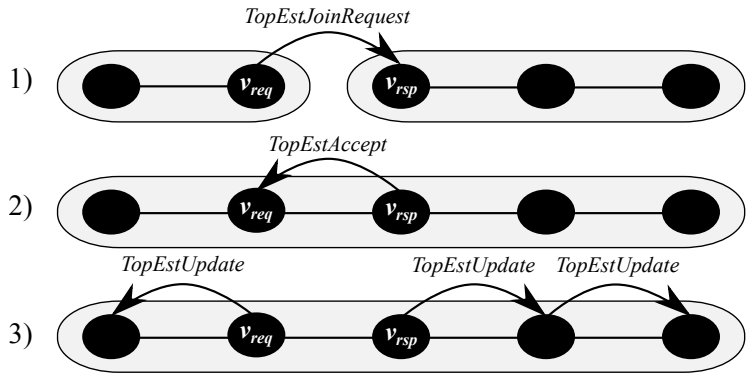


Abbildung 4.27: Ein erfolgreiches Verknüpfen zweier Linientopologieabschnitte und die dabei versendeten Nachrichten.

durch das Senden einer *TopEstAccept*-Nachricht. Diese enthält den resultierenden Linientopologieabschnitt, sodass dieser nun v_{rsp} und v_{req} bekannt ist. Beide Knoten informieren daraufhin wie in Abb. 4.27 gezeigt die anderen Knoten des Linientopologieabschnitts mittels *TopEstUpdate*-Nachrichten über die veränderte Topologie. Falls die zweite oder vierte Bedingung nicht erfüllt ist, lehnt v_{rsp} das Verknüpfen durch das Senden einer *TopEstReject*-Nachricht an v_{req} ab. In diesem Fall merken sich v_{req} und v_{rsp}, dass das Verknüpfen fehlgeschlagen ist und initiieren dies nicht erneut. Erst sobald sich der Linientopologieabschnitt einer der beiden Knoten verändert hat, kann das Verknüpfen erneut versucht werden.

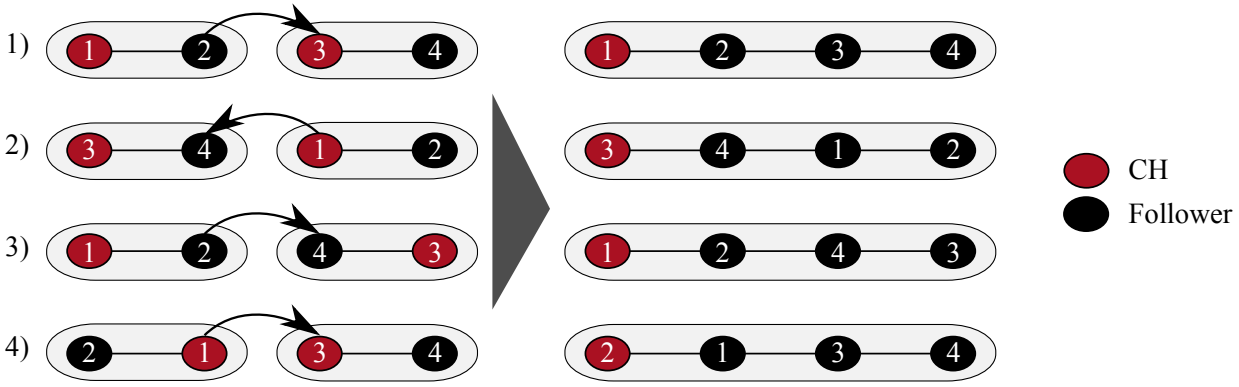


Abbildung 4.28: Es gibt vier verschiedene Fälle, wie zwei Linientopologieabschnitte miteinander verknüpft werden können.

Linientopologieabschnitte haben eine Knotenreihenfolge und der erste Knoten ist jeweils der CH. Zum Verknüpfen der Linientopologieabschnitte von v_{req} und v_{rsp} können die Knotenlisten $Sub(v_{req})$ und $Sub(v_{req})$ nicht in jedem Fall einfach konkateniert werden. Die vier möglichen Fälle sind in Abb. 4.28 illustriert und in Tab. 4.18 formal beschrieben, wobei $\bullet$ für die Konkatenation von Linientopologieabschnitten verwendet wird und $Reverse()$ einen Linientopologieabschnitt umdreht. In dem Fall, dass zwei Einzelknoten sich verknüpfen, sind die in Tab. 4.18 genannten Bedingungen von allen vier Fällen erfüllt. Es wurde definiert, dass in diesem Fall der erste Fall angewendet wird.

In Abb. 4.28 sind die CHs rot eingezeichnet. Dabei fällt auf, dass im ersten und dritten Fall der CH von $Sub(v_{req})$ zum CH des resultierenden Linientopologieabschnitts wird. Im zweiten Fall wird dies der CH von $Sub(v_{rsp})$. Im letzten Fall liegen die alten CHs nun in der Mitte des

Tabelle 4.18: Die vier Fälle beim Verknüpfen von Linientopologieabschnitten und der resultierende Linientopologieabschnitt.

Fall	$Pos(v_{req})$	$Pos(v_{rsp})$	$Join(Sub(v_{req}), Sub(v_{rsp}))$
1)	$\|Sub(v_{req})\|$	1	$Sub(v_{req}) \bullet Sub(v_{rsp})$
2)	1	$\|Sub(v_{rsp})\|$	$Sub(v_{rsp}) \bullet Sub(v_{req})$
3)	$\|Sub(v_{req})\|$	$\|Sub(v_{rsp})\|$	$Sub(v_{req}) \bullet Reverse(Sub(v_{rsp}))$
4)	1	1	$Reverse(Sub(v_{req})) \bullet Sub(v_{rsp})$

Linientopologieabschnitts. Da stets der erste Knoten CH sein soll, kommen sie als CH nicht mehr in Frage. Stattdessen wird der vormals letzte Knoten von $Sub(v_{req})$ zum CH.

Einfügen eines Einzelknotens in einen Linientopologieabschnitt Mit dem beschriebenen Mechanismus können sich zwei Linientopologieabschnitte zu einem längeren Abschnitt verknüpfen. Damit können Linientopologieabschnitte aber nur an den Enden erweitert werden. Manchmal kann es aber auch passieren, dass ein einzelner Knoten neu dazu kommt, der in einen bestehenden Linientopologieabschnitt integriert werden kann.

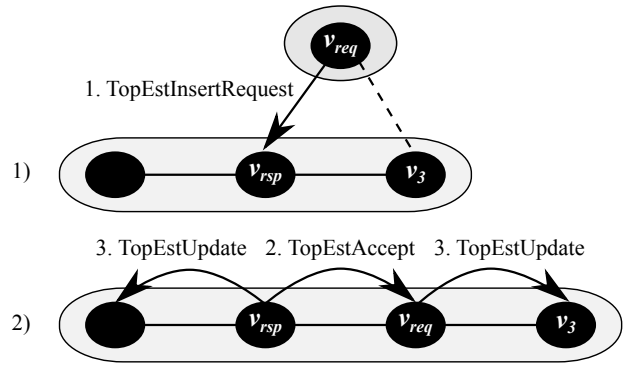


Abbildung 4.29: Beispiel eines Einzelknotens, der in einen Linientopologieabschnitt eingefügt wird.

Abb. 4.29 veranschaulicht diesen Prozess grafisch. Wir erlauben das Einfügen eines Knotens v_{req} in den bestehende Linientopologieabschnitt $Sub(v_{rsp})$ zwischen v_{rsp} und einem dritten Knoten v_3, falls folgende Bedingungen erfüllt sind:

1. Der Knoten v_{req} ist der einzige Knoten in seinem Linientopologieabschnitt:
   $Sub(v_{req}) = (v_{req})$
2. Der Knoten v_{req} hat eine stabile bidirektionale Verbdindung zu den Knoten v_{rsp} und einem dritten Knoten v_3.
3. Der Knoten v_3 ist Teil des Linientopologieabschnitts von v_{rsp} und darin entweder eine Position vor oder nach diesem eingeordnet:
   $v_3 \in Sub(v_{rsp}) \wedge Pos(v_3) \in \{Pos(v_{rsp}) - 1,\ Pos(v_{rsp}) + 1\}$
4. Der resultierende Linientopologieabschnitt überschreitet die konfigurierbare Maximalgröße $maxSize$ nicht:
   $|Sub(v_{rsp})| + 1 \leq maxSize$

Die ersten beiden Bedingungen kann v_{req} lokal prüfen. Die dritte und vierte Bedingung muss hingegen von v_{rsp} geprüft werden, da diesem $Sub(v_{rsp})$ bekannt ist. Wie in Abb. 4.29 illustriert

schickt v_{req} eine *TopEstInsertRequest*-Nachricht an v_{rsp}, in der die Knoten-ID von v_3 enthalten ist. Damit kann v_{rsp} die letzten beiden Bedingungen prüfen. Sofern sie erfüllt sind, wird das Einfügen mit einer *TopEstAccept*-Nachricht bestätigt. Abhängig von der Position von v_3 wird v_{req} vor oder nach v_{rsp} so in den Linientopologieabschnitt eingefügt, dass er zwischen v_{rsp} und v_3 liegt. Das Ergebnis des Einfügens wird den restlichen Knoten per *TopEstUpdate*-Nachrichten mitgeteilt. Falls die letzten beiden Bedingungen nicht erfüllt sind, lehnt v_{rsp} das Einfügen durch Senden einer *TopEstReject*-Nachricht ab.

Auftrennen von Linientopologieabschnitten Beim Bilden von Linientopologieabschnitten wird eine Overlay-Struktur gebildet, die mehrere Knoten in einer Linie miteinander verbindet. Falls einer der verwendeten Links bricht, muss der Linientopologieabschnitt aufgetrennt werden, da sonst nicht mehr alle Knoten den CH erreichen können. Im Anwendungsszenario der Güterzugüberwachung tritt dies beispielsweise auf, wenn Waggons von einem Zug abgekoppelt werden.

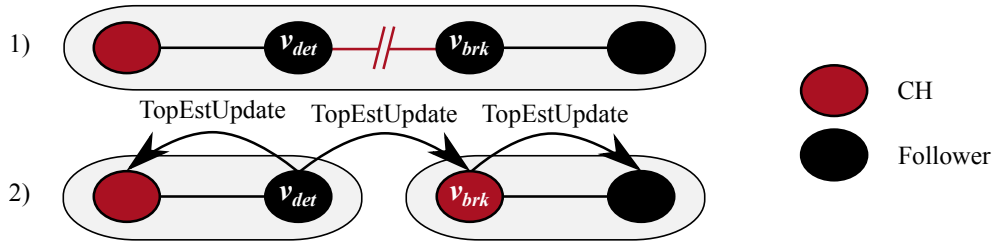


Abbildung 4.30: Erkennt ein Knoten, dass ein verwendeter Link gebrochen ist, so wird der Linientopologieabschnitt aufgetrennt.

Abb. 4.30 illustriert ein Beispiel, in dem der Knoten v_{det} erkannt hat, dass der Link, der ihn mit v_{brk} verbindet, gebrochen ist, d.h. nicht mehr stabil und bidirektional ist. Daraufhin startet v_{det} das Auftrennen. Er entfernt v_{brk} und alle über v_{brk} verbundenen Knoten aus seinem Linientopologieabschnitt. Über *TopEstUpdate*-Nachrichten teilt er die neue Topologie den anderen Knoten in seinem Abschnitt mit. Diese Nachricht sendet er auch an v_{brk}. Falls der Link zu diesem Knoten nur unzuverlässig oder asymmetrisch geworden ist, kann v_{brk} auf diese Weise schneller über die Topologieänderung informiert werden. Da der Link nicht mehr als stabil und bidirektional gilt, erwartet v_{det} allerdings kein Acknowledgement für die *TopEstUpdate*-Nachricht an v_{brk}. Falls v_{brk} die Nachricht noch erhält, entfernt er v_{det} und über ihn verbundene Knoten aus seinem Linientopologieabschnitt und teilt das Ergebnis mit den anderen Knoten in seinem Abschnitt. Erhält der Knoten v_{brk} die Nachricht nicht, so wird er wenig später selbst erkennen, dass der Link gebrochen ist und seinerseits das Aufsplitten der Linientopologie auslösen.

Falls der Knoten v_{brk} komplett ausgefallen ist, beispielsweise weil seine Batterie leer geworden ist, erkennen seine beiden Nachbarn einen gebrochenen Link und starten jeweils für ihre Seite den Prozess zum Aufsplitten. In diesem Fall können die beiden resultierenden Linientopologieabschnitte noch einen Prozess zum Verbinden der beiden Abschnitte ohne v_{brk} starten, falls der dazu benötigte Link zuverlässig und bidirektional ist.

Damit v_{det} und v_{brk} bei schwankender Linkqualität nicht wiederholt einen Prozess zum Verknüpfen und Auftrennen starten, merken sich beide Knoten (sofern noch in Betrieb) für einige Zeit, dass die Verbindung zum jeweils anderen Knoten getrennt wurde und starten solange keine neue Anfrage zum Verknüpfen.

Das Aufsplitten eines Linientopologieabschnitts kann auch noch nötig werden, wenn durch (nahezu) zeitgleiche Vorgänge zum Verknüpfen oder Einfügen die Bedingungen für das Verbinden

zweier Abschnitte verletzt werden. Insbesondere die Bedingung zur maximalen Länge eines Linientopologieabschnitts kann verletzt werden, wenn auf beiden Seiten des Abschnitts nahezu zeitgleich Abschnitte verknüpft werden. Falls Knoten räumlich in einem Ring angeordnet sind, könnte es auch auftreten, dass zwei Linientopologieabschnitte nahezu zeitgleich an beiden Seiten miteinander verknüpft werden und so eine Schleife entsteht. In diesen Fällen entstehen auf beiden Seiten des Abschnitts *TopEstUpdate*-Nachrichten. In der Mitte des Linientopologieabschnitts gibt es dann einen Knoten, der zuerst die Nachrichten von beiden Seiten erhalten hat. Dieser Knoten prüft die Bedingungen zum Verknüpfen von Linientopologieabschnitten. Falls sie nicht erfüllt sind, trennt er die Linientopologie an seiner Position wie beschrieben auf.

Optimierungen Beim Verbinden zweier Linientopologieabschnitte und beim Einfügen eines Einzelknotens können nicht alle notwendigen Bedingungen geprüft werden, ohne dass kommuniziert wird. Dies kann dazu führen, dass *TopEstJoinRequest*- oder *TopEstInsertRequest*-Nachrichten gesendet werden müssen, obwohl das Verbinden bzw. Einfügen nicht möglich ist. Um dies zu vermeiden, müsste der Knoten v_{req}, der das Verbinden bzw. Einfügen initiiert, Informationen über den Linientopologieabschnitt von v_{rsp} haben. Zum Verbinden wird $Pos(v_{rsp})$ und $|Sub(v_{rsp})|$ benötigt. Beim Einfügen wird darüber hinaus noch $Pos(v_3)$ benötigt und ob $v_3 \in Sub(v_{rsp})$. Diese Informationen könnten durch zusätzliche Status-Nachrichten ausgetauscht werden, was aber i.d.R. mehr Kommunikationsaufwand erzeugt als es einspart. Stattdessen haben wir die *LinkDet*-Nachrichten, die die Linkerkennung in LDLTE ohnehin regelmäßig sendet, um diese Information angereichert. Mit jeder *LinkDet*-Nachricht sendet ein Knoten nun auch seine Position in seinem Linientopologieabschnitt und dessen Länge. Außerdem ist die Knoten-ID des ersten Knoten im Linientopologieabschnitt, d.h. des CHs, enthalten. Indem v_{req} beim Einfügen die CHs von v_3 und v_{rsp} vergleicht, kann er prüfen, ob beide Knoten im gleichen Linientopologieabschnitt liegen. Trotz dieser Optimierung kann es noch vorkommen, dass Anfragen zum Verbinden oder Einfügen abgelehnt werden, da die über *LinkDet*-Nachrichten verteilte Information veraltet sein kann und die notwendigen Bedingungen nicht mehr erfüllt sind. Allerdings ist dies mit dieser Optimierung deutlich seltener der Fall.

4.5.4 Implementierung und Evaluierung

Das LDLTE-Protokoll umfasst die beiden beschriebenen Funktionalitäten zur dynamischen Erkennung von Links und zur Bildung und Reparatur von Linientopologien. Im Rahmen des AMRA-Projekts mit BEG wurde für das Anwendungsszenario der Güterzugüberwachung (s. Kapitel 4.5.1) eine Implementierung von LDLTE in C/C++geschrieben.

Abbildung 4.31: Die Platine der AMRA-Box, für die LDLTE implementiert wurde.

Als Hardware-Plattform kommt die im Rahmen des Projekts speziell entwickelte AMRA-Box zum Einsatz (s. Abb. 4.31). Sie enthält ein ublux SARA GSM-Modul, mit dem sich die Box über das Mobilfunknetz mit dem zentralen Server verbinden kann. Zur Konnektivität zwischen den Boxen und mit externen Sensoren ist ein Panasonic PAN1760 Bluetooth-Modul verbaut, das den BLE Standard unterstützt. Ein ublox 6 GPS-Receiver dient sowohl als Sensor zur Positionsbestimmung, als auch als externe Zeitquelle für die Zeitsynchronisation. Darüber hinaus enthält die Box weitere Sensoren, beispielsweise für Temperatur und Beschleunigung sowie einen Kompass. Als Stromversorgung enthält die Box zwei LSH 20 Lithium-Batterien mit einer Kapazität von je 13 Ah.

Die Implementierung von LDLTE basiert auf bereits vorhandenen Software-Komponenten, die im Rahmen des AMRA-Projektes für die Box entwickelt worden sind. Dies umfasst insbesondere Komponenten zur Kommunikation über BLE. In LDLTE werden *LinkDet*-Nachrichten zur Linkerkennung per Broadcast gesendet. Dies lässt sich in BLE über Advertising-Pakete realisieren. Die anderen Nachrichten in LDLTE sind konzeptionell betrachtet Unicast-Nachrichten. In BLE wird Unicast-Kommunikation aber nur in Piconetzen zwischen Master- und Slave-Knoten unterstützt. Der dazu nötige Verbindungsaufbau ist aufwendig und ermöglicht keine Kommunikation über mehrere Hops. Außerdem ist unklar, welche Knoten als Master und welche als Slave agieren sollten. Daher sendet die LDLTE-Implementierung auch Unicast-Nachrichten als BLE Advertising-Pakete. Der größte Nachteil dieses Ansatzes ist die beschränkte maximale Größe von Advertising-Paketen in BLE 4.0 von nur 28 Bytes Payload. Dieses Problem wurde in dem 2016 veröffentlichten Bluetooth-Standard BLE 5.0 [Blu16] entschärft, indem die maximale Größe von Advertising-Paketen verachtfacht wurde. Allerdings unterstützt das in der AMRA-Box verbaute Bluetooth-Modul diesen Standard nicht. Aus diesem Grund sind die in LDLTE spezifizierten Nachrichten möglichst klein gehalten, sodass sie mit 28 Bytes übertragen werden können. Immer wenn ein Linientopologieabschnitt übertragen wird, bedeutet dies bei größeren Abschnitten relativ viele Daten. LDLTE, wie es hier beschrieben wurde, überträgt ganze Linientopologieabschnitte in *TopEstJoinRequest*-, *TopEstAccept*- und *TopEstUpdate*-Nachrichten. Damit diese Nachrichten in 28 Bytes passen, sollten für die Knotenadressen nur kurze Identifier von zwei bis drei Bytes Länge genutzt werden. Dennoch muss bei der Wahl des Parameters $maxSize$ berücksichtigt werden, dass bei Verwendung von BLE 4.0 die maximale mögliche Länge eines Linientopologieabschnitts je nach Länge der Knotenadressen stark beschränkt ist. Als Lösung dieses Problems könnte man eine Hardware-Plattform einsetzen, die BLE 5.0 beherrscht. Alternativ kann man LDLTE anpassen, sodass statt kompletten Linientopologieabschnitten nur noch aggregierte Daten wie Hop-Counts, Knotenanzahl und Position sowie inkrementelle Updates übertragen werden. Im Rahmen des AMRA-Projekts wurde ebenfalls eine derartige Variante von LDLTE entwickelt.

Die Implementierung des LDLTE-Protokolls wurde im Rahmen des AMRA-Projekts ausgiebig getestet. Dabei wurde in unterschiedlichen Szenarien das Verbinden und Aufsplitten von Linientopologieabschnitten sowie das Einfügen einzelner Knoten getestet. Aus dem AMRA-Projekt entstand letztlich ein marktreifes Produkt. In einem Projekt mit der schweizerischen Bahngesellschaft SBB Cargo sind aktuell 150 Güterwaggons mit der smarten Überwachungstechnik ausgerüstet, 1500 weitere sollen folgen [Rob].

4.5.5 Zusammenfassung

In diesem Kapitel wurde das Thema Clustering für wettbewerbsbasierte drahtlose Kommunikationssysteme behandelt. Motiviert durch das Anwendungsszenario der Überwachung von

Güterzügen, wurden zunächst in Kapitel 4.5.2 Verfahren zur Bildung und Reparatur von Linientopologien untersucht. Dabei zeigte sich, dass das Bilden von Linientopologien relativ wenig Beachtung in der Literatur findet und existierende Verfahren erhebliche Nachteile aufweisen. Insbesondere die Reparatur der Topologie, beispielsweise bei einem Knotenausfall, wird von den betrachteten Verfahren nicht ausreichend unterstützt. Mit LDLTE wurde in Kapitel 4.5.3 anschließend ein Protokoll vorgestellt, das die dynamische Linkerkennung und das Bilden und Reparieren von Linientopologien unterstützt. Das Verfahren wurde im Rahmen des AMRA-Projektes mit BEG entwickelt und implementiert. Als Hardware-Plattform kam eine von BEG speziell für das Projekt entwickelte Box zum Einsatz. Ein Fokus bei der Entwicklung des Protokolls war insbesondere der Energieverbrauch, da die batteriebetriebene Box eine lange Lebensdauer haben soll. Um möglichst viel Energie einzusparen, können sich die Boxen mit dem LDLTE-Protokoll untereinander per BLE vernetzen und so Übertragungen per GSM einsparen, die deutlich mehr Energie kosten. Weitere Optimierungen in Hinblick auf den Energieverbrauch sind denkbar. In der beschriebenen Fassung von LDLTE ist der erste Knoten stets der CH eines Linientopologieabschnitts und damit für die Weitergabe der Daten per GSM an den zentralen Server zuständig. Um die Energie der Knoten gleichmäßiger zu verbrauchen, wäre es denkbar, die Rolle des CHs beispielsweise in Abhängigkeit der verbleibenden Batteriekapazität zu vergeben.

5. KAPITEL

Zusammenfassung und Ausblick

Drahtlose Kommunikationssysteme gewinnen immer mehr an Bedeutung. Dabei sollen diese verstärkt auch in Bereichen zum Einsatz kommen, in denen ein hohes Maß an Zuverlässigkeit verlangt wird. Ein Beispiel dafür sind drahtlose Kommunikationssysteme, die zur Prozessautomatisierung in Fabriken und Industrieanlagen zum Einsatz kommen. Etablierte Standards für drahtlose Kommunikationssysteme, wie IEEE 802.11 (WLAN), IEEE 802.15.4 oder Bluetooth, erfüllen die Anforderungen an Zuverlässigkeit nur unzureichend. In dieser Arbeit haben wir uns mit Protokollen und Algorithmen beschäftigt, die geeignet sind, die Zuverlässigkeit drahtloser Kommunikationssysteme zu verbessern. Der erste Teil der Arbeit behandelte Verfahren für TDMA-basierte Kommunikationssysteme, wohingegen der zweite Teil wettbewerbsbasierte Kommunikationssysteme betrachtete. In diesem Kapitel werden zunächst die wichtigsten Ergebnisse der beiden Teile zusammengefasst und anschließend ein Ausblick über offen gebliebene Fragen und Themen sowie mögliche Optimierungen gegeben.

5.1 Protokolle und Algorithmen für zuverlässige drahtlose TDMA-basierte Kommunikationssysteme

TDMA-basierte Kommunikationssysteme sind besonders geeignet, um hohe Zuverlässigkeit zu erreichen, da sie es durch exklusive Reservierungen ermöglichen, Kollisionen zwischen Rahmen des eigenen Netzwerks zuverlässig zu verhindern. Im Anwendungsszenario der Prozessautomatisierung sind die Anforderungen an die Zuverlässigkeit des Kommunikationssystems besonders wichtig. In der Arbeitsgruppe Vernetzte Systeme der TU Kaiserslautern wurde mit ProNet 4.0 (s. Kapitel 3.1) ein Protokoll-Stack entwickelt, der speziell für die Anforderungen an drahtlose Kommunikationssysteme für die Prozessautomatisierung ausgelegt ist. Er basiert auf TDMA und nutzt den IEEE 802.15.4 PHY Standard zur Kommunikation.

Da auch in TDMA-basierten Netzwerken mit exklusiven Reservierungen Kommunikation nur dann zuverlässig ist, wenn die benutzten Links zuverlässige Übertragungen ermöglichen, ist die Kenntnis der Kommunikationstopologie erforderlich. Um den Durchsatz im Netzwerk zu optimieren, kann TDMA darüber hinaus mit SDMA kombiniert werden, wozu allerdings die Interferenztopologie bekannt sein muss. Für Protokolle mit spezieller Black Burst-Kodierung ist darüber hinaus auch die Sensing-Topologie hilfreich. Da das manuelle Ausmessen und Konfigurieren dieser Topologien sehr aufwendig und umständlich ist, wird ein Protokoll zur Topo-

logieerkennung benötigt. In Kapitel 3.2 wurde daher das Problem der Topologieerkennung erläutert. Bei der Untersuchung bestehender Verfahren zeigten sich einige Schwächen. So berücksichtigen die Verfahren zur Topologieerkennung in den betrachteten Routing- und Clustering-Algorithmen asymmetrische Links oft nicht und überprüfen die Zuverlässigkeit von Kommunikationslinks unzureichend. Die betrachteten Verfahren zur Erkennung der Interferenztopologie sind entweder sehr aufwendig oder basieren auf Annahmen, die nur mit Einschränkungen in der Praxis sichergestellt werden können. Existierende Protokolle zur Erkennung der Sensing-Topologie sind nicht bekannt. Mit ATDP (Automatic Topology Discovery Protocol) wurde ein Protokoll vorgestellt, welches die drei genannten Topologien automatisch bestimmt und auf effiziente Weise netzweit verteilt. Die Schwierigkeit besteht dabei in der effizienten Bestimmung von Interferenz- und Sensing-Links, da über diese nicht zuverlässig kommuniziert werden kann. Die Evaluation von ATDP zeigte, dass die von ATDP erkannten Kommunikationslinks in der Praxis tatsächlich zuverlässig sind, die erkannten Interferenzlinks tatsächlich Kollisionen verursachen können und die erkannten Sensing-Links keine Störungen hervorrufen.

Auf Grund der oft geringen Reichweite von drahtlosen Übertragungen sind bei Anwendungen in Industrieanlagen oft keine direkten Übertragungen zwischen allen Knoten möglich. Daher wird ein Routing-Verfahren benötigt, welches die Weiterleitung von Rahmen über Zwischenknoten steuert. In Kapitel 3.3 wurde mit QMRP (QoS Multicast Routing Protocol) ein Protokoll vorgestellt, welches diese Aufgabe, zugeschnitten auf die Anforderungen der Anwendung, erfüllt. Das Protokoll unterstützt dabei partiell mobile Netzwerke, d.h. während die meisten Knoten im Netzwerk stationär sind, kann sich eine definierte Teilmenge der Knoten bewegen. Dies ist in Industrieanlagen dann der Fall, wenn es neben den fest montierten Knoten auch beispielsweise einige bewegliche Roboter gibt. Um zuverlässige Übertragungen zu ermöglichen, reserviert QMRP die TDMA-Slots, basierend auf der durch ATDP erkannten Topologie, exklusiv. Falls die selben Daten häufig an mehrere Knoten gesendet werden müssen, beispielsweise weil Knoten zur Steigerung der Zuverlässigkeit redundant ausgelegt sind, kann QMRP diese in Multicast-Bäumen effizient und zuverlässig bedienen. Eine Implementierung von QMRP wurde auf Imote 2 Hardware erfolgreich getestet und die in QMRP enthaltenen Heuristiken haben sich in Simulationen als zielführend erwiesen.

Um eine große Anzahl Knoten in einem drahtlosen Netzwerk effizient zu betreiben, ist es oft hilfreich, benachbarte Knoten zu Gruppen, sog. Clustern, zusammenzufassen. In Kapitel 3.4 haben wir uns daher mit Clustering-Verfahren beschäftigt. Eine Anforderung aus dem Anwendungsszenario war dabei, dass heterogene Netzwerke unterstützt werden, bei denen nicht alle Knoten die spezielle Aufgabe eines Clusterheads oder Gateways übernehmen sollen. Dies kann beispielsweise auf sich bewegende Knoten wie Roboter zutreffen, oder aber mit Batterie betriebene Knoten, deren begrenzte Energie nicht für die zusätzlichen Aufgaben verbraucht werden soll, die ein Clusterhead oder Gateway übernehmen muss. Die existierenden Verfahren unterstützten die gesuchten Anforderungen nur unzureichend. Daher wurden mit LCA und Max-Min-D zwei existierende Verfahren angepasst, sodass sie heterogene Netzwerke wie gewünscht unterstützen. Anschließend wurde mit HNC (Heterogeneous Network Clustering) ein Verfahren vorgestellt, welches speziell auf die Anforderungen zugeschnitten ist, die sich bei Anwendungen zur Prozessautomatisierung ergeben. Neben einer Implementierung für die Imote 2 Hardware wurde HNC auch simulativ evaluiert. Dabei zeigte sich, dass HNC die gesteckten Ziele besser erreicht als die angepassten Varianten von LCA und Max-Min-D.

Kapitel 3.5 beschäftigte sich schließlich mit Kommunikations-Middleware. Mit ProMid wurde eine Middleware-Schicht für ProNet 4.0 entwickelt, die einen dienstorientierten Ansatz verfolgt

und sich durch ein hohes Abstraktionslevel und Dienstgüte-Unterstützung auszeichnet. Zur Verwaltung der im Netzwerk verfügbaren Dienste implementiert ProMid eine replizierte und verteile Service Registry, die das Clustering-Verfahren HNC zur Auswahl der Service Registry-Knoten nutzt.

5.2 Protokolle und Algorithmen für zuverlässige drahtlose wettbewerbsbasierte Kommunikationssysteme

Im zweiten Teil haben wir uns mit Protokollen und Algorithmen für wettbewerbsbasierte Kommunikationssysteme wie IEEE 802.11 (WLAN) beschäftigt. TDMA-basierte Netzwerke sind nur dann zuverlässig, wenn sichergestellt werden kann, dass auf dem drahtlosen Medium nur Geräte senden, die sich an das TDMA-Verfahren halten. Da dies im häufig verwendeten ISM-Band heutzutage nicht mehr garantiert werden kann, besteht ein Bedarf für Kommunikationssysteme, welche ohne TDMA eine ausreichende Zuverlässigkeit erreichen. Zum Entwickeln und Testen der hierfür entwickelten Verfahren wurde das an der Arbeitsgruppe Vernetzte Systeme entwickelte WiPS Framework eingesetzt, welches auf IEEE 802.11 basiert und in Kapitel 4.1 beschrieben wurde. Da auch bei wettbewerbsbasierten Netzwerken die Kommunikationstopologie benötigt wird, um zuverlässige Kommunikation zu erreichen, haben wir uns in Kapitel 4.2 zunächst mit der Topologieerkennung in wettbewerbsbasierten Kommunikationssystemen auseinander gesetzt und mit ATDP4W (Automatic Topology Discovery Protocol for WiFi) eine Version von ATDP für WLAN vorgestellt.

Bei wettbewerbsbasierten Netzwerken können wenige Knoten leicht durch dauerhafte Übertragungen die anderen Knoten stark einschränken, sodass deren geforderte Zuverlässigkeit nicht gegeben ist. Um dennoch eine ausreichende Zuverlässigkeit zu erreichen, ist es daher notwendig, den Verkehr zu kontrollieren. Diesem Thema widmete sich Kapitel 4.3 und stellte ein auf dem Token Bucket-Verfahren basierendes Verfahren zur Verkehrsformung vor, welches sicherstellt, dass jeder Knoten nur die durch ein Reservierungsverfahren zugewiesene Bandbreite nutzt. Darüber hinaus wurde gezeigt, wie dieses Verfahren auch zur Verkehrsüberwachung genutzt werden kann, indem eine Metrik zur Bestimmung der Auslastung des drahtlosen Netzwerkes berechnet wird. Anschließend wurde das Verfahren in Kapitel 4.4 erweitert, sodass es die von den Knoten genutzte Bandbreite dynamisch, unter Berücksichtigung der Dienstgüteanforderungen, anpassen und so auf variierenden Verkehr externer Knoten automatisch reagieren kann.

Letztlich wurden in Kapitel 4.5 auch Clustering-Verfahren für wettbewerbsbasierte Kommunikationssysteme betrachtet. Dabei war diesmal das Anwendungsszenario der drahtlosen Überwachung von Eisenbahnwaggons der Ausgangspunkt. Aus der Anwendung auf Zügen ergibt sich eine natürliche Linientopologie. Das in Kooperation mit der Bosch Engineering GmbH entwickelte Verfahren LDLTE (Link Detection and Line Topology Establishment) bildet automatisch eine geclusterte Linientopologie und repariert diese, wenn sich die Topologie ändert, z.B. weil Waggons ab- oder angekoppelt werden. Neben Anforderungen an die Zuverlässigkeit der Übertragungen stehen bei dieser Anwendung auch hohe Anforderungen an die Energiesparsamkeit im Vordergrund, da die Knoten batteriebetrieben sind und die Batterie nur selten ausgetauscht werden kann.

5.3 Ausblick

In den einzelnen Kapiteln wurden Protokolle und Verfahren beschrieben, zu denen in der Regel noch Optimierungen denkbar sind. Die beiden Verfahren zur Topologieerkennung, ATDP und ATDP4W, sind bisher nur für Netzwerke ausgelegt, in denen die meisten Knoten stationär sind und sich die Topologie der stationären Knoten während des Betriebs nicht verändert. Falls ein neuer Knoten hinzukommt oder ein Knoten ausfällt, müssen beide Verfahren die Topologieerkennung neu starten, was einige Zeit dauert und ineffizient ist. Um diese Fälle besser zu unterstützen und Topologieänderungen während des Betriebs zu erkennen, müsste die Topologieerkennung entweder kontinuierlich im Hintergrund oder periodisch laufen. Denkbar ist, dass die Topologieerkennung die von der Anwendung ohnehin gesendeten Rahmen soweit möglich nutzt, um die Linkqualität während des Betriebs zu bewerten. Im Fall einer erkannten Topologieänderung sollten inkrementelle Updates ausgetauscht werden. Dabei ist das Problem zu berücksichtigen, dass das im Netz bekannte Topologiewissen inkonsistent sein kann, während ein inkrementelles Update ausgetauscht wird.

Das vorgestellte Multicast Routing-Verfahren QMRP basiert auf Heuristiken, die sich in Simulationen als zielführend erwiesen haben. Mögliche Variationen dieser Heuristiken wurden bereits erwähnt, aber nicht simulativ evaluiert. Des Weiteren umfasst das Verfahren zwei Strategien mit unterschiedlichen Optimierungszielen. Die Auswahl der Strategie könnte man automatisieren oder eine Hybrid-Strategie entwickeln, die beide Strategien kombiniert.

Für wettbewerbsbasierte Kommunikationssysteme wurde in dieser Arbeit noch kein Routing-Verfahren vorgestellt. Um das gewünschte Maß an Zuverlässigkeit zu erreichen, sollte ein Routing-Verfahren die von der Topologieerkennung bestimmte Zuverlässigkeit der Links berücksichtigen. Die Zuverlässigkeit einer Route kann als Produkt der Zuverlässigkeiten der einzelnen Links bestimmt werden. Ein QoS Routing-Verfahren könnte beispielsweise Routen suchen, die eine gewisse Mindestzuverlässigkeit aufweisen. In WiPS könnte ein Routing-Verfahren auch mit den Verfahren zur Verkehrskontrolle interagieren. Ähnlich wie QMRP Zeitslots für die gefundenen Routen einplant, könnte ein Routing-Verfahren in WiPS für die gefundenen Routen Bandbreite für die Knoten auf der Route reservieren. Außerdem könnte die aus der Unusable Wasted Token Ratio bestimmte Auslastung des Mediums in die Routing-Entscheidungen mit einbezogen werden und Routen vorzugsweise durch bisher wenig ausgelastete Bereiche des Netzwerks geleitet werden.

Das vorgestellte Verfahren zur kooperativen fairen Bandbreitenskalierung wurde bisher nur in einem kleinen Multihop-Experiment getestet, da kein größeres Testbett zur Verfügung stand. Zwar ist zu erwarten, dass das Verfahren in einem größerem Multihop-Szenario ohne Änderungen ebenso funktioniert, allerdings können die verwendeten Parameter und Ansätze sicherlich noch verbessert werden.

Für die TDMA-basierten Kommunikationssysteme wurde eine Middleware-Schicht entwickelt und vorgestellt. Eine ähnliche Middleware-Schicht wurde auch für das wettbewerbsbasierte WiPS entwickelt. Da sie sich konzeptionell nicht stark von ProMid unterscheidet, wurde sie in Kapitel 4 nicht weiter beschrieben. Bisher ist die für WiPS entwickelte Middleware unabhängig vom darunter liegenden Routing-Verfahren. Genauso wie ProMid in QMRP Zeitslots für die Diensterbringungen reservieren lässt, könnte die Middleware in WiPS die Bandbreitenreservierung über ein geeignetes Routing-Verfahren auslösen. Dazu fehlt bisher wie beschrieben im wesentlichen das Routing-Verfahren.

Ein Clustering-Verfahren wurde in WiPS bisher nicht implementiert, die Middlware-Schicht in WiPS implementiert auch bisher nicht wie ProMid eine verteilte und replizierte Registry, sondern eine einfache zentrale Registry. Hier könnte man ähnlich wie in ProMid eine verteilte und replizierte Registry umsetzen, bei der die Registry-Knoten wie in ProMid durch ein Clustering-Verfahren ausgewählt werden.

Das beschriebene Verfahren zur Bildung und Reparatur von Linientopologien könnte in Bezug auf den Energieverbrauch weiter optimiert werden. Momentan ist in einem Linientopologieabschnitt immer der erste Knoten Clusterhead (CH). Im Anwendungsszenario der Überwachung von Güterzügen bedeutet dies, dass der erste Knoten eines Linientopologieabschnitts die Daten per GSM mit dem RDC austauscht und seine Batterie damit stärker belastet wird als die der anderen Knoten. Um die Batterien gleichmäßiger aufzubrauchen, könnte man das Verfahren so modifizieren, dass regelmäßig jener Knoten zum CH bestimmt wird, der noch die meiste Energie zur Verfügung hat. Alternativ könnte man die Rolle des CH regelmäßig rotieren.

In dieser Arbeit wurde zwischen TDMA-basierten und wettbewerbsbasierten Kommunikationssystemen unterschieden. Betrachtet man beide Ansätze als Extreme, so stellt sich die Frage, ob Zwischenformen denkbar und sinnvoll sind. Auf der einen Seite steht TDMA, d.h. die vollständige Planung der Übertragungen anhand eines Zeitplans. Damit lassen sich Kollisionen verhindern und Übertragungen werden zu 100 % zuverlässig, sofern keine anderen Kommunikationssysteme auf dem genutzten Frequenzband senden. Da diese Annahme in der Praxis in vielen Frequenzbereichen nicht mehr gegeben ist, verliert TDMA in der Praxis oft seine deterministischen Eigenschaften. Ansätze wie Channel Hopping verbessern die Zuverlässigkeit lediglich statistisch.

Auf der anderen Seite steht wettbewerbsbasierte Kommunikation, die den offenen Wettbewerb um das Medium gestattet und damit keine Garantien ermöglicht und somit oft unzuverlässig ist. Wir können die Zuverlässigkeit eines Links, wie im beschriebenen Verfahren ATDP4W, statistisch analysieren und beispielsweise als Durchschnittswert angeben. Bei TDMA wird die Zuverlässigkeit erhöht, indem die Wahrscheinlichkeit, dass ein anderer Knoten des gleichen Netzwerkes zeitgleich sendet, durch entsprechende Planung auf 0 % sinkt. Die Zuverlässigkeit kann aber auch erhöht werden, indem die Wahrscheinlichkeit, dass ein anderer Knoten des gleichen Netzwerkes zeitgleich sendet, lediglich reduziert wird, aber nicht garantiert 0 % beträgt. Ähnlich wie bei TDMA könnte man die Zeit in Slots einteilen und diese einzelnen Knoten zuordnen. Allerdings würde man erlauben, dass sich Slots überlappen dürfen und deshalb trotzdem wettbewerbsbasiert um das Medium konkurriert wird. Dieser wettbewerbsbasierte Zugriff ist ohnehin notwendig, um Kollisionen mit externem Verkehr zu vermeiden. Die Anzahl interner Knoten, die maximal zeitgleich um das Medium konkurrieren, könnte durch entsprechende Planung begrenzt werden. Dabei müsste man in einem Multihop-Szenario auch Hidden Stations beachten. Da die Slots sich ohnehin überlappen dürfen und der Zugriff wettbewerbsbasiert geschieht, ist keine so genaue Synchronisation zwischen den Knoten notwendig wie bei TDMA. Als Ergebnis erhält man nicht 100 % Zuverlässigkeit in einem kurzen Zeitslot wie bei TDMA, sondern eine gesteigerte Zuverlässigkeit in einem längeren Slot. Die Wahrscheinlichkeit, ob ein Knoten sendet oder nicht, kann man über Mechanismen der Verkehrskontrolle wie den Token Bucket-Mechanismus beeinflussen. So könnte man die einem Knoten zugewiesene Bandbreite nach einem Zeitplan steuern. Der Knoten würde also in ihm zugeordneten Zeitslots mehr Bandbreite erhalten als während der restlichen Zeit, sodass er mit einer höheren Wahrscheinlichkeit in den ihm zugewiesenen Slots sendet. Dennoch kann er auch außerhalb der ihm zugewiesenen

Slots senden, dies ist allerdings deutlich unwahrscheinlicher. Ein solches Verfahren ließe sich gut in WiPS auf Grundlagen der beschriebenen Verkehrskontrolle mittels Token Bucket realisieren.

Abkürzungsverzeichnis

ACE	Algorithm for Cluster Establishment
ACK	Acknowledgement
AMRA	Asset Monitoring for Rail Applications
AODV	Ad-hoc On-demand Distance Vector
ATDP	Automatic Topology Discovery Protocol
ATDP4W	Automatic Topology Discovery Protocol for WiFi
AVTP	Arbitrating Value Transfer Protokoll
BBS	Black Burst Synchronization
BEG	Bosch Engineering GmbH
BiPS	Black-burst integrated Protocol Stack
BLE	Bluetooth Low Energy
CBF	Channel Busy Fraction
CBT	Channel Busy Time
CCA	Clear Channel Assessment
CDMA	Code Division Multiple Access
CH	Clusterhead
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
DCF	Distributed coordination function
DCOM	Distributed Component Object Model
DIFS	DCF Interframe Space
DSDV	Destination-Sequenced Distance Vector
FDMA	Frequency Division Multiple Access
FSF	Fewest Slots First
GDC	Global Data Collector
GSR	Global State Routing
HART	Highway Addressable Remote Transducer
HNC	Heterogeneous Network Clustering
HWMP	Hybrid Wireless Mesh Protocol
JSON	JavaScript Object Notation

LAN Local Area Network
LCA Linked Clustering Algorithm
LCF Least Conflict First
LDC Local Data Collector
LDLTE Link Detection and Line Topology Establishment
LLC Logical Link Control
MAC Medium Access Control
MEP Message Exchange Pattern
MES Medium Energy Sensed
MIMO Multiple-Input Multiple-Output
MLO MAC Layer Overhead
MRF Most Reuse First
OFDMA Orthogonal Frequency Division Multiple Access
OLSR Optimized Link State Routing Protocol
OPC Open Platform Communications
OPC UA OPC Unified Architecture
PDR Paket Delivery Rate
PHY Physical (Layer)
PLO Physical Layer Overhead
PREP Path Reply
PREQ Path Request
QMRP QoS Multicast Routing Protocol
RANN Root Announcement
RDC Remote Data Collector
RID Radio Interference Protocol
RMI Remote Method Invocation
RPC Remote Procedure Call
RSS Received Signal Strength
RSSI Received Signal Strength Indicator
RTS Request to send
SDMA Space Division Multiple Access
SNAP SubNetwork Access Protocol
SOA Service Oriented Architecture
TBR Time-based Regulator
TDMA Time Division Multiple Access
UDDI Universal Description, Discovery and Integration
UWBR Unusable Wasted Bandwidth Ratio
WiPS WiFi Protocol Stack
WLAN Wireless LAN
WSDL Web Services Description Language
WSN Wireless Sensors Network
XML Extensible Markup Language
ZDO ZigBee Device Object

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

Publikationsliste

[Eng+19] Markus Engel, Christopher Kramer, Tobias Braun, Dennis Christmann und Reinhard Gotzhein. "BiPS–A Real-Time-Capable Protocol Framework for Wireless Networked Control Systems and its Application". In: *International Conference on E-Business and Telecommunications*. Springer. 2019, S. 313–336. ISBN: 978-3-030-11039-0.

[Geb+15] Johann Gebhardt, Reinhard Gotzhein, Anuschka Igel und Christopher Kramer. "QoS Multicast Routing in Partially Mobile Wireless TDMA Networks". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE. Dez. 2015, S. 1–7. DOI: 10.1109/GLOCOM.2015.7417382.

[KCG15] Christopher Kramer, Dennis Christmann und Reinhard Gotzhein. "Automatic Topology Discovery in TDMA-based Ad Hoc Networks". In: *Wireless Communications & Mobile Computing (IWCMC), 11th International Conference on*. Aug. 2015, S. 634–639. DOI: 10.1109/IWCMC.2015.7289157.

[KCG16] Christopher Kramer, Dennis Christmann und Reinhard Gotzhein. "A Clustering Algorithm for Distributed Service Registries in Heterogeneous Wireless Networks". In: *2016 Wireless Days (WD)*. IEEE. März 2016, S. 1–7. DOI: 10.1109/WD.2016.7461485.

[KGM19] Christopher Kramer, Reinhard Gotzhein und Kiran Mathews. "Cooperative Fair Bandwidth Scaling in Contention-based Wireless Networks using Time Token Bucket". In: *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE. Okt. 2019, S. 1–9. DOI: 10.1109/IPCCC47392.2019.8958764.

[KHH12] Christopher Kramer, Volker Höfner und Theo Härder. "Lastprognose für energieeffiziente verteilte DBMS". In: *INFORMATIK 2012*. Gesellschaft für Informatik e.V., 2012, S. 397–411. ISBN: 978-3-88579-602-2.

[MKG17] Kiran Mathews, Christopher Kramer und Reinhard Gotzhein. "Token Bucket Based Traffic Shaping and Monitoring for WLAN-based Control Systems". In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE. Okt. 2017, S. 1–7. DOI: 10.1109/PIMRC.2017.8292201.

[Sef+18] Hamed Sefati, Reinhard Gotzhein, Christopher Kramer, Stephan Schloesser und Martin Weiss. "Dynamic Overlay Line Topology Establishment and Repair in Wireless Networks". In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. Apr. 2018, S. 1–6. DOI: 10.1109/WCNC.2018.8377079.

Literaturverzeichnis

[Ach+08] Prashanth Aravinda Kumar Acharya, Ashish Sharma, Elizabeth M Belding, Kevin C Almeroth und Konstantina Papagiannaki. "Congestion-aware rate adaptation in wireless networks: A measurement-driven approach". In: *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*. IEEE. 2008, S. 1–9.

[AGB11] Johan Akerberg, Mikael Gidlund und Mats Bjorkman. "Future research challenges in wireless sensor and actuator networks targeting industrial automation". In: *9th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE. 2011, S. 410–415.

[Ami+00] A.D. Amis, R. Prakash, T.H.P. Vuong und D.T. Huynh. "Max-min d-cluster formation in wireless ad hoc networks". In: *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Bd. 1. 2000, 32–41 vol.1. DOI: 10.1109/INFCOM.2000.832171.

[AY07] Ameer Ahmed Abbasi und Mohamed Younis. "A survey on clustering algorithms for wireless sensor networks". In: *Computer communications* 30.14 (2007), S. 2826–2841.

[BAM12] SMS Bari, Farhat Anwar und MH Masud. "Performance study of hybrid Wireless Mesh Protocol (HWMP) for IEEE 802.11 s WLAN mesh networks". In: *2012 international conference on computer and communication engineering (ICCCE)*. IEEE. 2012, S. 712–716.

[BBH13] Abdelhak Bentaleb, Abdelhak Boubetra und Saad Harous. "Survey of Clustering Schemes in Mobile Ad hoc Networks". In: *Communications and Network* 5 (2013), S. 8.

[BE81] D.J. Baker und Anthony Ephremides. "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm". In: *Communications, IEEE Transactions on* 29.11 (Nov. 1981), S. 1694–1701. ISSN: 0090-6778. DOI: 10.1109/TCOM.1981.1094909.

[BG87] Dimitri P Bertsekas und Robert G Gallager. *Data networks*. Prentice-Hall International New Jersey, 1987.

[Bia00] Giuseppe Bianchi. "Performance analysis of the IEEE 802.11 distributed coordination function". In: *IEEE Journal on selected areas in communications* 18.3 (2000), S. 535–547.

[Blu10] Bluetooth, SIG. "Bluetooth core specification version 4.0". In: *Specification of the Bluetooth System* (2010).

[Blu16] SIG Bluetooth. "Bluetooth core specification version 5.0". In: *Specification of the Bluetooth System* (2016).

[BSO07] Albert Banchs, Pablo Serrano und Huw Oliver. "Proportional fair throughput allocation in multirate IEEE 802.11 e wireless LANs". In: *Wireless Networks* 13.5 (2007), S. 649–662.

[CG98] Tsu-Wei Chen und Mario Gerla. "Global state routing: A new routing scheme for ad-hoc wireless networks". In: *Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on*. Bd. 1. IEEE. 1998, S. 171–175.

[CGR12] D. Christmann, R. Gotzhein und S. Rohr. "The Arbitrating Value Transfer Protocol (AVTP) - Deterministic Binary Countdown in Wireless Multi-Hop Networks". In: *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*. Aug. 2012, S. 1–9. DOI: 10.1109/ICCCN.2012.6289227.

[Chi07] Chipcon / Texas Instruments. *CC2420 datasheet*. http://focus.ti.com/lit/ds/symlink/cc2420.pdf. Revision SWRS041b. 2007.

[Chr11] J. Chroboczek. *The Babel Routing Protocol*. RFC 6126 (Experimental). Updated by RFCs 7298, 7557. Internet Engineering Task Force, 2011. URL: http://www.ietf.org/rfc/rfc6126.txt.

[CJ03] T. Clausen und P. Jacquet. *Optimized Link State Routing Protocol (OLSR)*. RFC 3626 (Experimental). Internet Engineering Task Force, 2003. URL: http://www.ietf.org/rfc/rfc3626.txt.

[CKL02] Yuh-Shyan Chen, Yun-Wen Ko und Ting-Lung Lin. "A lantern-tree-based QoS multicast protocol for wireless ad-hoc networks". In: *Proceedings. Eleventh International Conference on Computer Communications and Networks*. IEEE. 2002, S. 242–247.

[CLL07] Yuh-Shyan Chen, Tsung-Hung Lin und Yun-Wei Lin. "A hexagonal-tree TDMA-based QoS multicasting protocol for wireless mobile ad hoc networks". In: *Telecommunication Systems* 35.1-2 (2007), S. 1–20.

[CP04] Haowen Chan und Adrian Perrig. "ACE: An emergent algorithm for highly uniform cluster formation". In: *European workshop on wireless sensor networks*. Springer. 2004, S. 154–171.

[CYK08] Jaehyuk Choi, Joon Yoo und Chong-Kwon Kim. "A distributed fair scheduling scheme with a new analysis model in IEEE 802.11 wireless LANs". In: *IEEE Transactions on Vehicular Technology* 57.5 (2008), S. 3083–3093.

[DB97] Bevan Das und Vaduvur Bharghavan. "Routing in ad-hoc networks using minimum connected dominating sets". In: *Communications, 1997. ICC'97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on*. Bd. 1. IEEE. 1997, S. 376–380.

[De +05] Douglas SJ De Couto, Daniel Aguayo, John Bicket und Robert Morris. "A high-throughput path metric for multi-hop wireless routing". In: *Wireless Networks* 11.4 (2005), S. 419–434.

[Dij59] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (1959), S. 269–271.

[DKS10] Peter Dely, Andreas J Kassler und Dmitry Sivchenko. "Theoretical and experimental analysis of the Channel Busy Fraction in IEEE 802.11". In: *Future Network and Mobile Summit, 2010*. IEEE. 2010, S. 1–9.

[Ene02] Energetics Inc. (in collaboration with US Dept. of Energy, Office of Energy Efficiency and Renewable Energy). "Industrial wireless technology for the 21st century". In: *Proc. Industrial Wireless Workshop* (Dezember 2002). URL: http://www.energetics.com/resourcecenter/products/roadmaps/Documents/wireless.pdf.

[Eng13] Markus Engel. "Optimierung und Evaluation Black Burst-basierter Protkolle unter Verwendung der Imote 2-Plattform". Masterarbeit. TU Kaiserslautern, 2013.

[Eur10] European Committee for Electrotechnical Standardization (CENELEC). *Industrial communication networks - Wireless communication network and communication profiles - WirelessHART™ (IEC 62591:2010)*. Juni 2010.

[Geb14] Johann Gebhardt. "TDMA-based Multicast-QoS-Routing-Approaches for Mobile Ad Hoc Networks". In: *Seminar thesis, TU Kaiserslautern*. 2014.

[GJ79] MR Garey und DS Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1979.

[GK11] Reinhard Gotzhein und Thomas Kuhn. "Black Burst Synchronization (BBS) – A Protocol for Deterministic Tick and Time Synchronization in Wireless Networks". In: *Computer Networks* 55.13 (2011), S. 3015–3031.

[GLP09] Navin Gautam, Won-Il Lee und Jae-Young Pyun. "Track-sector clustering for energy efficient routing in wireless sensor networks". In: *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*. Bd. 2. IEEE. 2009, S. 116–121.

[GM99] Jose Joaquin Garcia-Luna-Aceves und Ewerton L Madruga. "The core-assisted mesh protocol". In: *IEEE Journal on selected Areas in Communications* 17.8 (1999), S. 1380–1394.

[Got14] Reinhard Gotzhein. *ProNet4.0 – A Wireless real-time Communication System for Industry 4.0*. Techn. Ber. 2014. URL: `https://vs.cs.uni-kl.de/publications/2014/Go14/`.

[Heu+05] Martin Heusse, Franck Rousseau, Romaric Guillier und Andrzej Duda. "Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless LANs". In: *ACM SIGCOMM Computer Communication Review*. Bd. 35. 4. ACM. 2005, S. 121–132.

[Hie+07] Guido R Hiertz, Sebastian Max, Rui Zhao, Dee Denteneer und Lars Berlemann. "Principles of IEEE 802.11 s". In: *2007 16th International Conference on Computer Communications and Networks*. IEEE. 2007, S. 1002–1007.

[Hie+10] Guido R Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann und Bernhard Walke. "IEEE 802.11 s: the WLAN mesh standard". In: *IEEE Wireless Communications* 17.1 (2010), S. 104–111.

[Hui+12] J. Hui, JP. Vasseur, D. Culler und V. Manral. *An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)*. Internet Requests for Comments. RFC. Internet Engineering Task Force, März 2012.

[IEC12] IEC. *IEC/PAS 62734 ed1.0*. März 2012. URL: `http://webstore.iec.ch/Webstore/webstore.nsf/ArtNum_PK/46254?OpenDocument`.

[IEE03] IEEE Std. 802.15.4. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. New York, NY, USA: IEEE Computer Society, Okt. 2003.

[IEE11a] IEEE. *IEEE 802.15.4™-2011*. `http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf`. Sep. 2011.

[IEE11b] IEEE Std. 802.11s. *IEEE Std. 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 10: Mesh Networking*. IEEE Computer Society, 2011.

[IEE16] IEEE. *IEEE 802.11™-2016*. Dezember 2016. URL: https://standards.ieee.org/standard/802_11-2016.html.

[Jar+05] Amit P Jardosh, Krishna N Ramachandran, Kevin C Almeroth und Elizabeth M Belding-Royer. "Understanding congestion in IEEE 802.11 b Wireless Networks". In: *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. USENIX Association. 2005, S. 25–25.

[Jav+13] Nadeem Javaid, TN Qureshi, AH Khan, Adeel Iqbal, E Akhtar und M Ishfaq. "EDDEEC: Enhanced developed distributed energy-efficient clustering for heterogeneous wireless sensor networks". In: *Procedia Computer Science* 19 (2013), S. 914–919.

[JCH84] Rajendra K Jain, Dah-Ming W Chiu und William R Hawe. "A quantitative measure of fairness and discrimination". In: *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* (1984).

[JDB99] Raj Jain, Arjan Durresi und Gojko Babic. "Throughput fairness index: An explanation". In: *ATM Forum contribution*. Bd. 99. 45. 1999.

[JM09] Van Jacobson und S McCanne. "libpcap: Packet capture library". In: *Lawrence Berkeley Laboratory, Berkeley, CA* (2009).

[Kel97] Frank Kelly. "Charging and rate control for elastic traffic". In: *European transactions on Telecommunications* 8.1 (1997), S. 33–37.

[KK14] Markus Kraußе und Rainer Konrad. *Drahtlose ZigBee-Netzwerke*. Springer, 2014. ISBN: 365805820X.

[Lao+03] Anis Laouiti, Philippe Jacquet, Pascale Minet, Laurent Viennot, Thomas Clausen und Cedric Adjih. *Multicast Optimized Link State Routing*. Research Report RR-4721. INRIA, 2003. URL: https://hal.inria.fr/inria-00071865.

[Le+13] Yuan Le, Liran Ma, Wei Cheng, Xiuzhen Cheng und Biao Chen. "A time fairness-based MAC algorithm for throughput maximization in 802.11 networks". In: *IEEE Transactions on Computers* 64.1 (2013), S. 19–31.

[LIB13] Jürgen Lange, Frank Iwanitz und Thomas J Burke. *OPC: Von Data Access bis Unified Architecture*. VDE, 2013. ISBN: 3800735067.

[Lin] Linux kernel. *802.11 netlink interface public header*. https://elixir.bootlin.com/linux/v4.6.5/source/include/uapi/linux/nl80211.h. Zeilen 2951-2974, Abgerufen am: 27.01.2020.

[Liu12] Xuxun Liu. "A survey on clustering routing protocols in wireless sensor networks". In: *Sensors* 12.8 (2012), S. 11113–11153.

[LPY08] Li Li, Martin Pal und Yang Richard Yang. "Proportional fairness in multi-rate wireless LANs". In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, S. 1004–1012.

[LR02] Stephanie Lindsey und Cauligi S Raghavendra. "PEGASIS: Power-efficient gathering in sensor information systems". In: *Aerospace conference proceedings, 2002. IEEE*. Bd. 3. IEEE. 2002, S. 3–3.

[LSG02] Sung-Ju Lee, William Su und Mario Gerla. "On-demand multicast routing protocol in multihop wireless mobile networks". In: *Mobile networks and applications* 7.6 (2002), S. 441–453.

[Mat16] Kiran Mathews. *Automatic Topology Discovery in WiFi (IEEE 802.11) Multi-hop Ad Hoc Networks*. Techn. Ber. Technische Universität Kaiserslautern, 2016.

[MEM] MEMSIC Inc. *Imote2 Datasheet*. http://vs.cs.uni-kl.de/downloads/Imote2NET_ED_Datasheet.pdf. 2021.

[MRR80] John M McQuillan, Ira Richer und Eric Rosen. "The new routing algorithm for the ARPANET". In: *Communications, IEEE Transactions on* 28.5 (1980), S. 711–719.

[MTA05] Mogens Mathiesen, Gilles Thonet und Niels Aakwaag. "Wireless ad-hoc networks for industrial automation: current trends and future prospects". In: *Proceedings of the IFAC World Congress, Prague, Czech Republic*. 2005, S. 89–100.

[Nis08] Mattias Nissler. "Modeling and Analysis of Optimal Slot Allocations in Wireless Ad-Hoc Networks". Diplomarbeit. Technische Universität Kaiserslautern, 2008.

[Nyh10] Peter Nyhuis. *Wandlungsfähige Produktionssysteme*. GITO mbH Verlag, 2010.

[Pad+05] Jitendra Padhye, Sharad Agarwal, Venkata N. Padmanabhan, Lili Qiu, Ananth Rao und Brian Zill. "Estimation of Link Interference in Static Multi-hop Wireless Networks". In: *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*. IMC '05. Berkeley, CA: USENIX Association, 2005, S. 305–310. URL: http://dl.acm.org/citation.cfm?id=1251086.1251114.

[PB94] C. E. Perkins und P. Bhagwat. "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers". In: *Proceedings of the ACM SIGCOMM '94*. 1994, S. 234–244.

[PCE] PC Engines GmbH. *PC Engines apu2 system boards*. https://www.pcengines.ch/apu2.htm. Abgerufen am: 27.01.2020.

[PR99] C. E. Perkins und E. M. Royer. "Ad-hoc On-Demand Distance Vector Routing". In: *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applicatons*. New Orleans, LA, Feb. 1999, S. 90–100.

[QZW06] Li Qing, Qingxin Zhu und Mingwen Wang. "Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks". In: *Computer communications* 29.12 (2006), S. 2230–2237.

[Rad] Radiotap. *Introduction*. http://www.radiotap.org. Abgerufen am: 27.01.2020.

[Ran+09] Ashish Raniwala, Pradipta De, Srikant Sharma, Rupa Krishnan und Tzi-cker Chiueh. "Globally fair radio resource allocation for wireless mesh networks". In: *2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*. IEEE. 2009, S. 1–10.

[Rasa] Raspberry Pi Foundation. *FAQs - Raspberry Pi Documentation / The computer hardware*. https://www.raspberrypi.org/documentation/faqs/#hardware. Abgerufen am: 27.01.2020.

[Rasb] Raspberry Pi Foundation. *Teach, Learn, and Make with Raspberry Pi – Raspberry Pi*. https://www.raspberrypi.org. Abgerufen am: 27.01.2020.

[Rob] Robert Bosch GmbH. *Smarte Schienenkolosse*. https://www.bosch.com/de/stories/smarte-schienenkolosse/. Abgerufen am: 27.01.2020.

[RP99] Elizabeth M Royer und Charles E Perkins. "Multicast operation of the ad-hoc on-demand distance vector routing protocol". In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM. 1999, S. 207–218.

[SAB08] Irfan Sheriff, Prashanth Aravinda Kumar Acharya und Elizabeth M Belding. "Measurement driven Admission Control on Wireless Backhaul Networks". In: *Computer Communications* 31.7 (2008), S. 1354–1371.

[Shi+06] Kuei-Ping Shih, Chih-Yung Chang, Yen-Da Chen und Tsung-Han Chuang. "Dynamic bandwidth allocation for QoS routing on TDMA-based mobile ad hoc networks". In: *Computer Communications* 29.9 (2006), S. 1316–1329.

[SK96] J.L. Sobrinho und A.S. Krishnakumar. "Real-Time Traffic over the IEEE 802.11 Medium Access Layer". In: *Bell Labs Technical Journal* (autumn 1996), S. 172–187.

[SMS06] Gaurav Sharma, Ravi R. Mazumdar und Ness B. Shroff. "On the Complexity of Scheduling in Wireless Networks". In: *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. MobiCom '06. Los Angeles, CA, USA: ACM, 2006, S. 227–238. ISBN: 1-59593-286-0. DOI: 10.1145/1161089.1161116. URL: http://doi.acm.org/10.1145/1161089.1161116.

[Son+16] Liwei Song, Yun Liao, Kaigui Bian, Lingyang Song und Zhu Han. "Cross-layer protocol design for CSMA/CD in full-duplex WiFi networks". In: *IEEE Communications Letters* 20.4 (2016), S. 792–795.

[Sta16] William Stallings. *Wireless communication networks and systems*. Boston u.a.: Pearson, 2016. ISBN: 1-292-10871-1.

[Tan03] Andrew S. Tanenbaum. *Computer Networks*. 4. Prentice Hall International, Inc., 2003. ISBN: 0-13-038488-7.

[TG04] Godfrey Tan und John V Guttag. "Time-based Fairness Improves Performance in Multi-Rate WLANs." In: *USENIX annual technical conference, general track*. 2004, S. 269–282.

[Tur86] Jonathan Turner. "New directions in communications (or which way to the information age?)" In: *IEEE communications Magazine* 24.10 (1986), S. 8–15.

[Wir] Wireshark Foundation. *Wireshark · Go Deep.* https://www.wireshark.org. Abgerufen am: 27.01.2020.

[WJ07] Huayi Wu und Xiaohua Jia. "QoS multicast routing by using multiple paths/trees in wireless ad hoc networks". In: *Ad Hoc Networks* 5.5 (2007), S. 600–612.

[WL99] Jie Wu und Hailan Li. "On calculating connected dominating set for efficient routing in ad hoc wireless networks". In: *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. ACM. 1999, S. 7–14.

[YC05] Jane Yang Yu und Peter Han Joo Chong. "A survey of clustering schemes for mobile ad hoc networks". In: *IEEE Communications Surveys & Tutorials* 7.1 (2005), S. 32–48.

[Zho+05] Gang Zhou, Tian He, John A Stankovic und Tarek Abdelzaher. "RID: Radio Interference Detection in Wireless Sensor Networks". In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Bd. 2. IEEE. 2005, S. 891–901.

[Zig] ZigBee Alliance. *Zigbee - The full-stack solution interlacing all your smart devices*. https://zigbeealliance.org/solution/zigbee/. Abgerufen am: 28.01.2020.

[ZL13] Xu Zhen und Zhou Long. "Bandwidth constrained multicast routing for TDMA-based mobile ad hoc networks". In: *Journal of Communications* 8.3 (2013), S. 161–167.

[ZM05] Baoxian Zhang und Hussein T Mouftah. “QoS routing for wireless ad hoc networks: problems, algorithms, and protocols”. In: *IEEE Communications Magazine* 43.10 (2005), S. 110–117.

Lebenslauf

Ausbildung

03/2008	**Abitur**, *Lina Hilger Gymnasium Bad Kreuznach*
10/2008 - 03/2012	**Bachelor of Science (B. Sc.)**, *Technische Universität Kaiserslautern*, Studiengang Informatik mit Nebenfach Wirtschaftswissenschaften Thema der Bachelorarbeit: „Workload Prediction for Energy-Efficient Distributed Database Management Systems“
04/2012 - 06/2014	**Master of Science (M.Sc.)**, *Technische Universität Kaiserslautern*, Studiengang Informatik mit Nebenfach Wirtschaftswissenschaften Thema der Masterarbeit: „Drahtlose Kommunikationssysteme für den Produktionsbereich“
07/2014 - 01/2020	**Doktorand**, *Technische Universität Kaiserslautern*

Beruflicher Werdegang

02/2007 - 01/2020	**Softwareentwickler**, *Mittelstandsoptimierer. Vertumno GmbH*
02/2016 - 01/2020	**Wissenschaftlicher Mitarbeiter**, *Technische Universität Kaiserslautern*, Arbeitsgruppe Vernetzte Systeme am Fachbereich Informatik