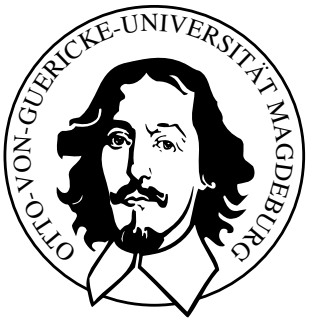
Bandbreitenkontrolle in drahtlosen Multihop-Netzwerken

– Diplomarbeit –

Martin Wewior



Otto-von-Guericke Universität Magdeburg
Fakultät für Informatik
Institut für Verteilte Systeme

Betreuer: Dipl.-Inform. André Herms

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACK ..... Acknowledgement

AODV ..... Ad-Hoc on Demand Distance Vector

AQOR ..... Ad hoc QoS On-demand Routing

ATM ..... Asynchronous Transfer Mode

CACP ..... Contention-aware Admission Control Protocol

CSMA ..... Carrier Sense Multiple Access

CSMA/CA ... Carrier Sense Multiple Access/Collision Avoidance

CSMA/CD ... Carrier Sense Multiple Access with Collision Detection

CTS ..... Clear to Send

DCF ..... Distributed Coordination Function

DIN ..... Deutsches Institut für Normung e. V.

DSR ..... Dynamic Source Routing

FIFO ..... First In First Out

FTP ..... File Transfer Protocol

GEA ..... Generic Event-based API

GSM ..... Global System for Mobile Communications

HCF ..... Hybrid Coordination Function

IEEE ..... Institute of Electrical and Electronics Engineers

IP ..... Internet Protocol

ISO ..... International Organization for Standardization

ISO/OSI..... International Organization for Standardization/Open Systems Interconnection

MAC..... Media Access Control

MANET..... Multihop Ad-Hoc Network

MMWN..... Multimedia support for Wireless Networks

NS2..... Network Simulator 2

OLSR..... Optimized Link State Routing

PCF..... Point Coordination Function

PDA..... Personal Digital Assistent

QoS..... Quality of Service

RTS..... Ready to Send

STL..... Standard Template Library

TCP..... Transmission Control Protocol

TDMA..... Time Division Multiple Access

UDP..... User Datagram Protocol

WLAN..... Wireless LAN (Local Area Network)

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Kontext

Die Mobilität von Computern, auch in Form anderer „Rechengeräte“, wie *PDAs* und *Smartphones*, hat in den letzten Jahren stark zugenommen. Die breite Verfügbarkeit drahtloser Netzwerktechnologien, höhere Leistung bei geringerer Größe und geringeren Kosten förderten zudem die weite Verbreitung und Verfügbarkeit.

Das Idealbild ist hierbei die Möglichkeit, auf beliebige Informationen zu jeder Zeit zugreifen zu können. Dafür sind entsprechende Geräte, Hardware, und der Zugriff auf Informationen über Datenkommunikation notwendig.

Im Bereich der drahtlosen Kommunikation von Computern haben drahtlose Netzwerke nach dem Standard IEEE 802.11 (u.a. [20, 12]) schnell eine dominierende Stellung eingenommen. Geräte nach diesem Standard erlauben Computern einen flexiblen Betrieb in spontan erstellten Ad-hoc-Netzen oder in Zusammenarbeit mit Basisstationen (*Access Points*). Solche Basisstationen werden von mobilen Knoten als Zugangspunkt zum drahtgebundenen Netz genutzt. Auf diese Weise können sämtliche Dienste eines kabelgebundenen Netzwerkes auch drahtlos zur Verfügung gestellt werden. Generell ermöglichen die Basisstationen die Kommunikation mit anderen Knoten, unabhängig davon, ob die Verbindung drahtlos im selben Empfangsbereich zustande kommt, oder über ein drahtgebundenes Netz, mit welchem der *Access Point* verbunden ist.

Der Ad-hoc-Modus ist im Bezug auf die Reichweite vergleichsweise eingeschränkt. Lediglich mit Knoten innerhalb der Reichweite der eigenen Sende-/Empfangseinheit kann kommuniziert werden. Je nach Situation und örtlichen Gegebenheiten können dies einige Meter bis selten knapp über 200 Meter sein. Mit der Möglichkeit zur Kommunikation können natürlich auch Dienste anderer Rechner genutzt werden.

Der große Vorteil eines Ad-hoc-Netzes ist die „sofortige“ Betriebsbereitschaft ohne zusätzliche Hardware oder Installationsarbeiten und der geringe Einrichtungsbedarf (im Vergleich zur Konfiguration vernetzter Basisstationen).

Drahtlose Netze gleichartiger Stationen, welche untereinander kommunizieren und als Router Nachrichten auch über die physikalischen Grenzen der Reichweiten der Funksendeempfänger weiterleiten, werden als *Mesh-Networks* (vermaschte Netze) oder auch *MANETs* (Multihop-Ad-hoc-Netzwerke) bezeichnet. Diese Netzwerke kombinieren die Vorteile von Ad-hoc- und Infrastrukturnetzwerken. Die einzelnen Knoten befinden sich im Ad-hoc-Modus, d.h. sie können mit jeder Station in Reich-

weite direkt kommunizieren. Durch Routingmechanismen können Pakete über die Reichweitengrenzen hinaus an Knoten versandt werden, indem Knoten das Paket entgegennehmen, welche sich näher am Ziel befinden als die Quelle. Im Beispiel in Abbildung 1.1 schickt Knoten 1 eine Nachricht an Knoten 3. Knoten 3 ist nicht in seiner Reichweite. Mit Knoten 2 kann Knoten 1 direkt kommunizieren und er ist außerdem näher an Knoten 3. Aus diesen Gründen schickt Knoten 1 seine Nachricht an 2. Dieser Knoten leitet sie weiter in Richtung 3. Da 3 in Reichweite von 2 liegt, kann 2 die Nachricht von 1 an das Ziel zustellen.

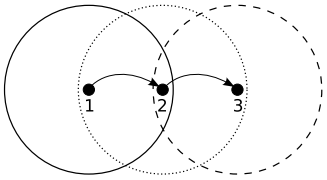


Abbildung 1.1: Drei Knoten in einem drahtlosen Netz. Ihre Sendereichweite ist durch Kreise (jeweils durchgezogen, gepunktet und gestrichelt) dargestellt

Auf diese Weise lassen sich in kurzer Zeit auch große Netze aufbauen, welche eine große Fläche abdecken können. Hilfreich ist dies in Fällen, wo ein kurzfristiger Einsatz zusätzlicher fester Infrastruktur nicht rentabel (kurzfristige Installationen) oder nicht möglich ist. Bauliche Gegebenheiten (etwa Denkmalschutzbestimmungen) können das Verlegen von Kabeln verbieten und bilden einen möglichen Einsatzzweck für Multi-Hop-Netzwerke. Weitere denkbare Einsatzmöglichkeiten liegen bei der Katastrophenhilfe oder wenn aus anderen Gründen die Infrastruktur gestört ist. Auch bei Großveranstaltungen können MANETs kurzfristig die Aufgabe einer „mobilen Infrastruktur“ übernehmen, d.h. flexibel und schnell zu installierende Backbone- und Routingfunktionen zur Kommunikation und Datenerfassung/-abfrage auch für andere Geräte übernehmen.

Die Aufgabe der Weiterleitung übernehmen Routingalgorithmen. Für drahtlose Multihop-Netze existieren mehrere: *AODV*, „Ad Hoc on Demand Distance Vector“ [21] und *DSR* Dynamic Source Routing [14] gehören zu den ersten Protokollen für Multi-Hop-Netze. Sie sind Weiterentwicklungen der ersten Routingalgorithmen: Distance-Vector bzw. Link-State-Routing für drahtlose Netzwerke.

Bei Anwendungen, wie EMail, FTP, usw., ist die fehlerfreie Übertragung der Daten das Hauptziel. Weitere Anforderungen an die Güte des Dienstes, etwa eine bestimmte Laufzeit (Latenz) oder eine minimale Anzahl übertragener Pakete pro Zeit können nicht zugesichert werden. Dies ist unkritisch bei den genannten Anwendungen. Für andere, wie beispielsweise Bild- oder Tonübertragungen („Voice over IP“, „VOIP“) sind dies kritische, nötige Eigenschaften. Die Protokolle, welche diese Daten

transportieren, müssen dafür Sorge tragen, dass eine bestimmte *Dienstgüte*, *Quality of Service* (*QoS*), garantiert werden kann. Typische Eigenschaften, welche gefordert werden können, sind z.B. maximale Latenz oder minimal verfügbare Bandbreite.

Derartige Anforderungen an die Güte von Diensten sollten auch in *MANETs* garantiert werden können. Die erwähnten Routingprotokolle *AODV* und *DSR* wurden nicht mit diesem Ziel entwickelt, unterstützen *QoS* daher auch nicht. Vielmehr unterstützen die zugrundeliegenden am häufigsten implementierten Teile des Standards IEEE 802.11 [12, 20], 802.11b/g und auch 802.11a, nur einen verteilten, „fairen“ Medienzugriffsmechanismus (*MAC*). Dabei kann jeder Knoten mit gleich hoher statistischer Wahrscheinlichkeit Daten versenden. Priorisierungen und Garantien für Bandbreiten oder Latenzzeiten sind nicht vorhanden. Ein bisher kaum implementierter Teil von 802.11, die Ergänzung IEEE 802.11e [11, 6] ermöglicht verschieden priorisierte Klassen, so genannte *Access Categories*. Zusammen mit Prioritäten wird auf diese Weise die Bevorzugung von Datenpaketen beim Versenden ermöglicht.

Allerdings sind keine Zuteilungs- oder Überwachungsfunktionen vorgesehen, so dass keine Garantien gemacht werden können, außer dass ein Paket vor einem anderen versandt wird. Wenn mehrere Knoten aber Pakete „bevorzugt“ versenden, so konkurrieren diese wieder untereinander. Ist die Auslastung des Mediums entsprechend hoch, können auch trotz der Priorisierung (zu) lange Paketlaufzeiten auftreten. IEEE 802.11e ist also alleine nicht ausreichend, kann aber helfen, um etwa die Menge der *Quality of Service*-Übertragungen generell bevorzugt gegenüber einem „Best effort“-Paket zu behandeln.

Außerdem beinhaltet keines der erwähnten 802.11-Protokolle einen Mechanismus um zu hohem Verkehr entgegenzuwirken. *Congestions*, also „Verstopfungen“ durch hohe Senderaten aller Stationen führen zu Verlusten von Paketen bereits vor dem Senden – „Drops“ in der Warteschlange. Daraus ergeben sich weitere Probleme, etwa die Wiederholung der ge*drop*ten Pakete, welche die höheren Schichten nach einer bestimmten Wartezeit auslösen können. Diese belasten das Medium und erschweren die Einschätzung der genutzten Bandbreite und anderer Parameter des Kommunikationsmediums.

1.2 Ziele

Drahtlose Ad-Hoc-Netzwerke ermöglichen die Kommunikation aller Teilnehmer untereinander über Routingmechanismen. Die zugrundeliegenden Protokolle stellen sicher, dass Nachrichten vollständig und fehlerfrei von einem Knoten zum nächsten Nachbarn in der Route gesendet werden. Verschiedene Lösungen existieren, um die Zusicherung von darüber hinaus gehenden Eigenschaften des Sendevorgangs zu gewährleisten, etwa die benötigte Bandbreite zur Zustellung eines Paketes. Keine bekannte Lösung erfüllt jedoch die Kriterien, etwa zur Kontrolle der Bandbreite, voll-

ständig. In dieser Arbeit soll daher ein System entwickelt werden, welches die Bandbreitenkontrolle in drahtlosen Multihop-Netzen ermöglicht.

Dazu ist es nötig, dass für jeden einzelnen „Sprung“ („Hop“) von Knoten zu Knoten die Bedingungen geprüft werden, welche an die Route gestellt wird. Ein Algorithmus assistiert dem eingesetzten Routingmechanismus, indem es auf Anfrage das Vorhandensein von Bandbreite prüft. Die Berechnung der Bandbreite, welche zur Verfügung steht, wird oft mit Abschätzungen und oberen Schranken durchgeführt. Zur genauen Feststellung der Bandbreite sind umfassende Kenntnisse der Umgebung eines Knotens und eine exakte Berechnung der Bandbreiten notwendig. Die physikalischen Eigenschaften des Mediums erfordern außerdem, dass die zu erforschende Nachbarschaft nicht durch die Verbindungen („Links“) zwischen Knoten definiert wird, sondern durch den erweiterten Einflussbereich des Trägersignals.

Aufgrund dieser Tatsache muss ein Algorithmus, welcher die Bandbreiten berechnen soll, exakte Informationen von Knoten in seiner Umgebung erhalten. Diese Knoten, eine verteiltes System, müssen einen Konsens zur Nutzung der Bandbreite finden. Dazu ist es nötig dass alle Knoten eine konsistente Sicht, die Nutzung der Bandbreite betreffend, haben.

In anderen Arbeiten beschränkt sich die Kontrolle der Bandbreite oft auf die passive Überwachung, *Monitoring*, führt jedoch keine aktive Beschränkung der Applikationen durch. Diese Beschränkung ist jedoch notwendig, wenn Garantien für bestimmte Dienstqualitäten gegeben werden sollen.

Durch die vorgesehene Nutzung von unmodifizierter „Standard“-Hardware ergeben sich Einschränkungen in Bezug auf die Art und Weise, wie auf das Medium zugegriffen wird. Der grundlegende Zugriff auf das Medium erfolgt durch die im Standard IEEE 802.11 ([12]) vorgesehene Funktion, welche eine statistische Gleichverteilung der gesendeten Pakete aller Stationen anstrebt. Zusätzlich wird die Mobilität der Knoten nicht betrachtet. Sie wird in Grenzen berücksichtigt durch die Betrachtung von Knotenausfällen. Ein Protokoll, welches aufgrund bestimmter Topologien Aktionen durchführt, muss davon ausgehen können, dass sich diese Topologien innerhalb bestimmter zeitlicher Grenzen nicht verändern. Annahmen über Nachbarn etc. wären sonst nicht möglich. Aus diesem Grunde wären dann Reservierungen nicht durchführbar, bzw. würden so schnell veralten, dass ihre Lebensdauer kürzer als die Einstellungszeit ist und sie damit hinfällig sind.

Zusammengefasst ist das Ziel dieser Diplomarbeit ein System für drahtlose Multihop-Netzwerken zu entwickeln, welches eine verteilte Bandbreitenkontrolle ermöglicht.

1.3 Ergebnisse

Im Rahmen dieser Arbeit wurden Protokolle entwickelt, welche in einem drahtlosen Multihop-Netzwerk Bandbreiten für Applikationen reservieren können. Auf diese Weise sollen Dienstgüten ermöglicht werden.

Bei der Analyse der vorhandenen Lösungsvorschläge und Konzepte wurde ersichtlich, dass in drahtlosen Netzwerken ohne Einschränkungen keine Kommunikation mit garantierten Dienstgüten möglich ist. Die Entwicklung begann deshalb mit der Annahme einer temporär-statischen Topologie. In dieser sind Änderungen der Konnektivitäten durch Mobilität und Knotenausfälle möglich, unter der Voraussetzung, dass die Häufigkeit bzw. die Abstände zwischen diesen Ereignissen hinreichend groß sind, um einen Algorithmus nutzen zu können, welcher diese Änderungen erfassen und auswerten kann.

Der erste Teil der Arbeit bestand aus der Entwicklung eines Protokolls zur Erkennung der Knoten der Nachbarschaft, welche durch die Übertragungen eines Knotens beeinflusst wird. Zur Feststellung der Größe dieser Nachbarschaft und zur Einordnung der Ergebnisse der durchgeführten Simulationen wurden Messungen durchgeführt. Diese ergaben u.a., dass die Nachbarschaft bis zu zwei Hops von Sendevorgängen eines Knotens beeinflusst wird. Aufbauend auf diesen Ergebnissen wurde ein Algorithmus zur Erkennung der n-Hop-Nachbarschaft mit $n = 2$ entwickelt.

Die Kenntnis der Nachbarschaft ermöglicht es dem entwickelten Zugriffskontrollprotokoll (*admission control protocol*) mit den Knoten zu kommunizieren, welche die Bandbreite untereinander aufteilen. Dadurch ist es möglich, dass das Protokoll Reservierungen auf diesen betroffenen Knoten durchführt und so Anwendungen Bandbreite zur Übertragung zusichern kann.

Diese zugesicherte Bandbreite wird durch den dritten Bestandteil der Arbeit überwacht. Ein Algorithmus, welcher die von den Applikationen versendete Datenmenge auf das zugesicherte Maximum beschränkt ist das Ergebnis. Dieser Algorithmus nutzt *Traffic-Shaping*-Verfahren, um zu verhindern, dass ein Knoten bzw. eine Applikation des Knotens mehr Ressourcen nutzt als ihr zugesagt werden. Auf diese Weise wird sichergestellt, dass alle Applikationen das vorher festgelegte Maximum an Daten auch versenden können.

Abschließend wurden Simulation mit durch die realen Messungen angepassten Parametern durchgeführt. Diese Simulationen belegten die korrekte Funktion der Nachbarschaftserkennung über mehrere *Hops* hinweg. Die Funktionsweise des Zugriffskontrollmechanismus zur Erkennung einer drohenden Übersättigung des Mediums wurde nachgewiesen und die Kontrolle der Applikationen zeigte die geplante Funktionalität unter allen Lastbedingungen.

1.4 Gliederung der Arbeit

Die Arbeit setzt sich wie folgt zusammen:

Abschnitt 2 führt in die Grundlagen der Thematik ein. Nach einer Übersicht der physikalischen Gegebenheiten und Besonderheiten des drahtlosen Mediums folgt ein Überblick über bekannte Konzepte und Lösungen. Andere Arbeiten zum Thema Dienstgütegarantien werden vorgestellt und analysiert. Verwendete Bezeichnungen und Begriffe werden erläutert.

Der folgende Abschnitt 3 verbindet die im Abschnitt 2 vermittelten Grundlagen und theoretischen Ansätze und entwickelt daraus ein Konzept, welches die Ziele der Arbeit erfüllen kann. Es werden Ansätze diskutiert, wie jedes der drei zentralen Probleme (Nachbarschaftserkennung, Bandbreitenbeschränkung und Berechnung der Bandbreite zum kontrollierten Zulassen von Applikationen) gelöst werden könnte und wie sie schließlich gelöst wurden.

Der Abschnitt 4 nutzt die Schlussfolgerungen des vorhergehenden Kapitels. Die konkrete Umsetzung der Protokolle wird beschrieben. Paketformate der Nachbarschaftserkennung und der Admission-Control werden betrachtet und erläutert. Außerdem wird die Integration der Programme in das umgebende Framework und die Art und Weise der Nutzung der dadurch vorgegebenen Funktionen beschrieben.

Ergebnisse aus Messungen und Simulationen werden im Abschnitt 5 vorgestellt. Dazu zählen die Messungen und Simulationen zur Feststellung der Reichweiten der Sender/Empfänger bzw. deren *Carrier-Sense-Reichweite* über den eigenen Sende-/Empfangsbereich hinaus. Außerdem werden in Simulationen die Ergebnisse der Arbeit überprüft und die daraus abgeleiteten Erkenntnisse zur Bestätigung von Annahmen und der Funktionsfähigkeit des Protokolls genutzt.

Abschnitt 6 fasst abschließend die Arbeit nochmals zusammen, diskutiert bestehende Probleme und gibt einen Ausblick auf mögliche weitere Entwicklungen bzw. Verbesserungen der Protokolle und Implementierungen, etwa im Hinblick auf die Behandlung von Nicht-Dienstgüte-abhängigem Verkehr und in Entwicklung befindliche Standards.

2 Grundlagen

Die folgenden Abschnitte beschäftigen sich mit den Grundlagen zur Durchführung drahtloser Kommunikation, Bandbreitenmanagement und -kontrolle. Es werden Voraussetzungen und Besonderheiten drahtloser Kommunikation beschrieben sowie Eigenschaften und Funktionalitäten der Protokolle erläutert, welche zur drahtlosen Kommunikation mit Dienstgütegarantien entwickelt wurden. Einige der Protokolle werden beispielhaft vorgestellt, wenn sie Lösungsansätze zu einer der drei Aufgaben Bandbreitenberechnung, Bandbreitenkontrolle oder Reservierung von Bandbreite beinhalten.

2.1 Drahtlose Netzwerke – IEEE 802.11

Die Entwicklung des mittlerweile weit verbreiteten Standards IEEE 802.11 [12] verfolgte das Ziel, bereits existierende Anwendungen und Protokolle höherer Schichten zu unterstützen. Zwischen drahtloser und drahtgebundener Kommunikation gibt es, wie in der Einleitung bereits angedeutet, Unterschiede. Die bekannten Protokolle und Vorgehensweisen der unteren Schichten (wie *Sicherungsschicht* bzw. *Data Link Layer* und *Physical Layer* bzw. *Bitübertragungsschicht* im ISO/OSI-Referenzmodell – (DIN) ISO 7498 [13, 3]) müssen deshalb modifiziert bzw. neu entwickelt werden und trotzdem gleiche Schnittstellen für höhere Schichten anbieten.

Die folgenden Abschnitte werden diese Unterschiede beleuchten und Lösungen der auftretetenden Probleme schildern, wie sie im Standard 802.11 der IEEE ([12, 20]) festgehalten sind.

2.1.1 Eigenschaften von WLAN im Vergleich mit Ethernet (IEEE 802.3)

Der Standard IEEE 802.11 [12] des *Institute of Electrical and Electronics Engineers* [9] spezifiziert eine Medienzugriffsschicht auf welcher u.a. TCP/IP und UDP aufsetzen können. Die angebotenen Schnittstellen und Eigenschaften sind für die höheren Schichten die gleichen, wie die des im Standard IEEE 802.3 [10] festgelegten *Ethernets*.

IEEE 802.11 spezifiziert, wie auch IEEE 802.3, einen gemeinsam genutzten Zugriff auf das Medium ohne zentrale Kontrolle oder koordinierte Steuerung des Zugriffs: Eine *Distributed Coordination Function*

Bei Anwesenheit eines zentralen Zugangspunktes (*Access Point*) sieht der Standard optional die *PCF – Point Coordination Function* vor. Der *Access Point* koordiniert den Medienzugriff der Stationen in seiner Reichweite. Ohne *Access Point* ist diese Option nicht verfügbar. In *Ad-Hoc-Netzen*, auf denen die *MANETs* basieren, ist dieser zentral koordinierte Medienzugriff deshalb nicht möglich.

Die *Distributed Coordination Function* implementiert grundsätzlich ein *Zeitmultiplexverfahren* (engl. *TDMA* – „Time Division Multiple Access“) – zur gleichen Zeit kann nur eine Station senden. Zur verteilten Festlegung, welcher Knoten in einem Netzwerk als nächster senden darf, wird ein *CSMA*-Verfahren genutzt („Carrier Sense Multiple Access“ – mehrfacher Zugriff durch Überwachung des Trägermediums).

Alle sendenden Knoten konkurriert um den Zugriff auf das Medium. Diese *Contention*-Phase bestimmt, welcher Knoten als nächster ein Datenpaket versenden darf. Das Prinzip dieser *Contention* ist bei IEEE 802.11 und IEEE 802.3 vergleichbar: Wenn ein Knoten ein freies Medium detektiert (*Carrier Sense*), beginnt er zu senden. Ist das Medium jedoch belegt wartet der Knoten und sendet nicht, bis es frei ist. Ist das Medium überbelegt, d.h. es sollen mehr Daten versandt werden, als möglich sind, werden Pakete verworfen. Das bedeutet, dass die internen Warteschlangen neue Pakete nicht mehr aufnehmen, sondern löschen. Diese Art Paketverluste werden *Drops* genannt.

Beim Senden von Paketen können Kollisionen auftreten, wenn mehrere Stationen gleichzeitig versuchen die Übertragung zu beginnen, bzw. eine Station während einer bereits laufenden Übertragung etwas sendet. Die Überlagerung der Signale verursacht eine Zerstörung der Daten, da diese nicht mehr aus dem „Mischsignal“ dekodiert werden können. Im Ethernet wird eine Kollision durch Überwachung des Zustands des Mediums (das Kabel), bereits während des Sendens festgestellt. Das Verfahren trägt deshalb den Zusatz „CD“ für *Collision Detection*, Kollisionserkennung, im Namen: *CSMA/CD*. Im Falle einer Kollision wird nach 802.3 die Übertragung abgebrochen und eine neue *contention*-Phase gestartet.

In drahtlosen Netzen nach 802.11 ist die Erkennung einer Kollision durch den Sender nicht möglich. Während im vorherigen Fall das gleiche Signal von allen Knoten (auch dem Sender) fast zeitgleich an allen Stellen des Mediums gleich stark messbar ist, schwächen sich elektromagnetische Wellen, Funkwellen, mit größerer Entfernung von Sender ab. Kollisionen können aus diesem Grund nicht erkannt werden. Selbst wenn eine zusätzliche Empfängereinheit das Medium überprüfen würde, kann nicht eingeschätzt werden, ob ein anderes Signal beim Empfänger stark genug ist, um die eigene Sendung zu stören. Vielmehr ist das eigene Signal so stark, dass Empfänger in der unmittelbaren Nähe des Senders gar keinen anderen Sender wahrnehmen können. Kurz zusammengefasst: Sendet ein anderer Knoten in Reichweite des Empfängers, kann dies von einem anderen Sender während der eigenen Übertragung nicht festgestellt werden. Die Folge dieser Situation wäre, dass das Medium so lange belegt bliebe, bis die Übertragung abgeschlossen ist, da die Sender von einer fehlerfreien

Übertragung ausgehen. Beim Empfänger treten aber Überlagerungen der Signale auf, so dass die gesendeten Pakete nicht mehr dekodiert werden können und so die Sendezeit verschwendet wird.

Das Ziel in drahtlosen Netzen muss deshalb die Kollisionsvermeidung (*collision avoidance*) sein. Das *CSMA/CA*-Verfahren („CA“ – „Collision Avoidance“) führt zu diesem Zweck zufällige Wartezeiten vor dem Senden ein (*Backoff*): Wird das Medium als „frei“ erkannt, kann nach einer zufälligen Wartezeit gesendet werden. Der Knoten, welcher die kürzeste Zeit gewartet hat, „gewinnt“ und beginnt zu senden. Alle anderen Knoten erkennen das belegte Medium und warten. Auf diese Weise ist es unwahrscheinlicher, dass mehrere Stationen gleichzeitig zu senden beginnen und so Kollisionen hervorrufen.

Bei großer, zunehmender Knotenzahl auf kleinem Raum steigt die Wahrscheinlichkeit für Kollisionen. Der Medienzugriffsmechanismus prüft periodisch, ob das Medium frei ist. Die Zahl dieser Prüfperioden ist das *Backoff*. Durch diese Periodizität bilden sich Zeitslots, in denen die Knoten das Medium prüfen, bevor sie senden.

Ist die Anzahl der Knoten sehr hoch, kann jeder der Slots, in welchem ein Knoten mit Senden beginnt, mehrfach belegt sein. Der Standard sieht für diese Fälle eine exponentielle Erhöhung des Wertebereichs der *Backoff*-Zeiten vor. Die spezifizierte, feste Obergrenze führt dennoch dazu, dass ab einer bestimmten Knotenzahl die *Backoff*-Funktion unwirksam wird. Als Folge nimmt die Zahl der Kollisionen in einem Maß zu, dass die Funktionsfähigkeit des Protokolls beeinträchtigt wird.

Protokolle, welche den IEEE 802.11-Standard nutzen, müssen diese Besonderheiten kennen und berücksichtigen, also auch das zu entwickelnde Zugangskontrollprotokoll.

2.1.2 Eigenschaften des drahtlosen Mediums

Die Substandards des 802.11er Standards definieren verschiedene Physical Layer (Bitübertragungsschichten) und darauf aufbauende Sicherungsschichten (Data-Link-Layer). Diese Schichten sind größtenteils in den Geräten integriert bzw. in den dazugehörigen Treibern. Die Substandards *b* und *g* arbeiten im „*ISM*“-Band („Industrial, Science, Medical“ ab 2,4 GHz). Dort stehen 22MHz „breite“ Kanäle mit einem Abstand von 5MHz zur Verfügung. Bei bis zu 14 Kanälen (von örtlichen Regelungen abhängig) bleiben real nur drei sich nicht störende Kanäle übrig (22MHz überlappen die 5MHz pro Kanal deutlich).

Neben der zur Verfügung stehenden Frequenzbreite hängt die *Bandbreite* vom genutzten Modulationsverfahren ab. Allgemein ist die (Daten)*Bandbreite* von Übertragungsmedien die Menge Daten, welche innerhalb einer bestimmten Zeit übertragen wird. Die Einheit ist typisch abgeleitet von Bit pro Sekunde. Von den theoretisch möglichen Übertragungsraten (Bandbreiten) der Standards entfällt ein großer Teil auf den Overhead des Protokolls und Mechanismen zur Sicherung der Datenübertragung.

Die oben bereits angesprochene Signalausbreitung vom Sender weg, führt zu einer weiteren Besonderheit des Funkmediums.

Eine Konstellation, wie in Abbildung 2.1 kann zu dem Auftreten des *Hidden Station*-Problems führen. Wenn sowohl A als auch C gleichzeitig senden, kann es zu Kollisionen bei B kommen. Da A und C sich nicht gegenseitig empfangen können, bemerken sie die Kollision nicht.

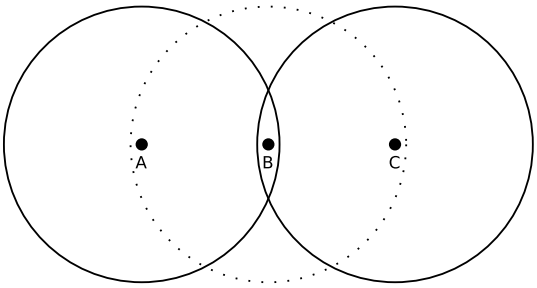


Abbildung 2.1: Beispiel für das *Hidden Station Problem*

IEEE 802.11 führt zur Vermeidung optional das RTS/CTS-Verfahren ein (siehe [23] und Abbildung 2.2). Dabei wird vor dem Senden eine Anfrage an den Empfänger geschickt (RTS – „Ready To Send“), welche die Dauer des geplanten Sendevorgangs enthält. Die Antwort (CTS – „Clear To Send“) enthält ebenfalls die ungefähre Sendedauer. Jeder Knoten, der das CTS empfängt, darf während dieser Zeit nicht senden. C empfängt das CTS und kann nicht gleichzeitig mit A senden. Die Pakete können nicht bei B kollidieren.

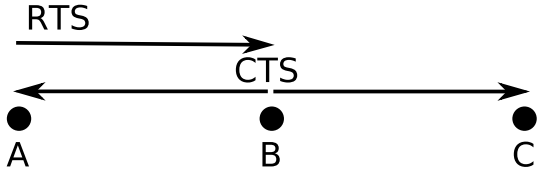


Abbildung 2.2: Lösung des *Hidden Station Problems* durch RTS/CTS

2.1.3 Bandbreite

Die Bandbreite ist ein entscheidendes Maß, für Dienstgütegarantien. Steht genügend Bandbreite zur Verfügung, ist beispielsweise die Latenz in erster Linie von Verzögerungen auf der Übertragungsstrecke und nicht von Wartezeiten in Warteschlangen abhängig.

Generell wird Bandbreite ρ wie in Gleichung 2.1 berechnet. s ist die Größe der versendeten Daten (die Anzahl der gesendeten Bytes), t ist die Zeit, welche diese Sendeoperation benötigt.

$$\rho = \frac{s}{t} \tag{2.1}$$

Ist die benötigte Zeit t kürzer, erhöht sich die Bandbreite bei gleichbleibender Paketgröße. Der Schluß ist naheliegend: Mit größerer Bandbreite kann man in der gleichen Zeit mehr Daten versenden. Gleichung 2.2 zeigt die umgestellte Berechnung.

$$s = \rho * t \tag{2.2}$$

Steht eine größere Bandbreite zur Verfügung, kann man mehr Daten übertragen, oder benötigt weniger Zeit. Die Frage, wieviel Zeit bei einer bestimmten Bandbreite benötigt wird, klärt eine weitere einfach Umstellung der Ausgangsgleichung (Gleichung 2.3). Auf diese Weise kann vorhergesagt werden, wann ein Sendevorgang bei bekannter Bandbreite und Datengröße abgeschlossen sein wird, bzw. wann frühestens der nächste gestartet werden kann.

$$t = \frac{s}{\rho} \tag{2.3}$$

2.1.4 Carrier-Sense-Bereich

Die für drahtlose Netzwerke nach dem Standard IEEE 802.11 [12] angegebenen Reichweiten von normalerweise 100 bis 300m beziehen sich auf den Bereich, in welchem eine Kommunikation zwischen zwei Stationen möglich ist.

Außerhalb dieses Bereichs ist das Signal nicht mehr „lesbar“, d.h. das Signal ist nicht mehr vollständig vom Rauschen oder anderen Signalen zu unterscheiden. Die schematische Abbildung 2.3 zeigt einen Knoten, den durch ihn erzeugten Empfangsbereich und den „Störungsbereich“ (der *Interferenzbereich* oder auch *Carrier-Sense-Bereich*).

Das Signal eines sendenden Knotens ist in dem *Carrier-Sense-Bereich* immer noch messbar, aber nicht mehr dekodierbar. Durch die *CSMA/CA*-Mechanismen, dem Carrier-Sensing, erkennen Knoten dieses Signal. Auf diese Weise werden Kollisionen mit den möglicherweise störenden Restsignalen vermieden. Auf der anderen Seite bewirkt dies aber auch die Reduzierung der vorhandenen Bandbreite.

Das sich ergebende Problem ist, dass Bandbreite, nicht nur von anderen Knoten im Sende-/Empfangsbereich belegt wird, sondern auch von Knoten außerhalb dieses Bereiches. Diese Knoten sind aber für Protokolle höherer Schichten unsichtbar.

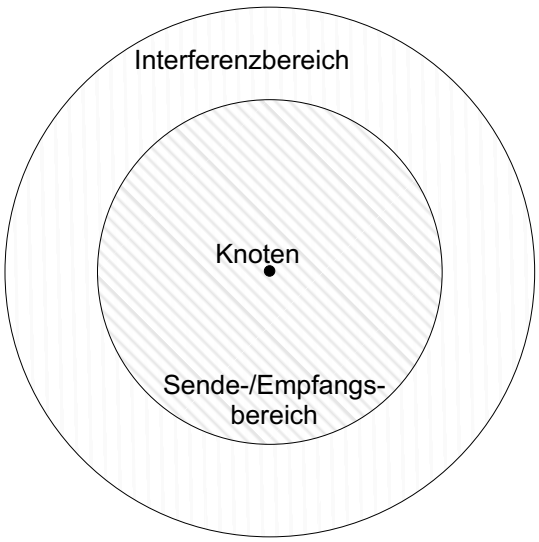


Abbildung 2.3: Empfangs- bzw Störungsbereiche um einen Knoten eines drahtlosen Netzes

2.1.5 Mobile Ad-Hoc-Netzwerke

Mobile Ad-Hoc-Netzwerke („MANETs“) sind Computer-Netzwerke, deren Knoten drahtlos verbunden sind und welche auf diese Weise mobil sein können. Das genutzte Medium ist ein geteiltes Medium. Alle Knoten können über Routingprotkolle ohne vorhandene Infrastruktur, wie etwa Accesspoints, Nachrichten im gesamten Netz verteilen, während der im Standard IEEE 802.11 ([12]) vorgesehene Ad-Hoc-Modus nur die Kommunikation mit Knoten in der direkten Reichweite ermöglicht.

Li et al [16] diskutieren ausführlich Besonderheiten der Kapazität in drahtlosen Multihop-Netzen. Eine der zentralen Aussagen betrifft die nutzbare Bandbreite von typischen „Knotenketten“, also Knoten, welche eine Nachricht von der Quelle zum Ziel weiterleiten. Die nutzbare Bandbreite auf einer derartigen Kette beträgt lediglich ein Viertel der Bandbreite eines Links. Warum, illustriert folgendes Beispiel: (Abbildung 2.4 aus [16])

Knoten A will eine Nachricht an Knoten F senden. Die Knoten dazwischen leiten das Paket weiter. Es ist offensichtlich, dass B keine neuen Nachrichten von A empfangen kann, während er die vorherige Nachricht an C weiterleitet. Das bedeutet, dass nur jeder zweite Knoten senden kann, weil die andere Hälfte empfängt.

Beachtet man, dass ein Knoten im drahtlosen Netz nicht gezielt in eine Richtung sendet, sondern in alle Richtungen, kann Knoten C nicht gleichzeitig mit Knoten A ein Paket versenden. Der Empfänger des Paketes von A, Knoten B würde durch

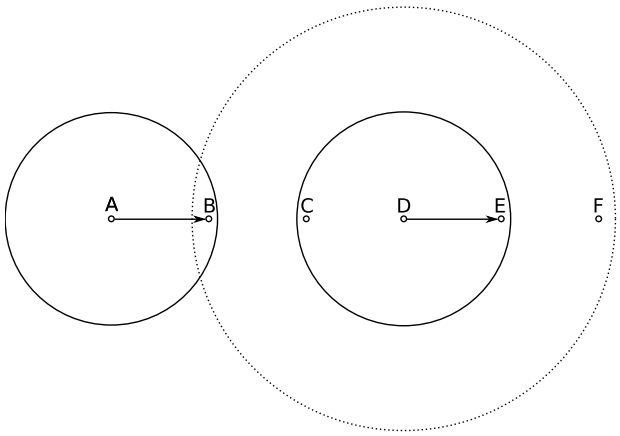


Abbildung 2.4: Einfluss des *Carrier-Sensings* auf die Kapazität eine Kette von Knoten

einen Sendevorgang gestört werden. Diese Situation entspricht dem „Hidden Station"-Problem, zu dessen Lösung das RTS/CTS-Schema eingesetzt wird (zur Erklärung siehe Abbildung 2.2 auf Seite 10). Es kann also nur jeder dritte Knoten in der Kette senden. Der Faktor der nutzbaren zur maximalen Bandbreite beträgt $\frac{1}{3}$.

Berücksichtigt man zusätzlich die Effekte im *Carrier-Sense-Bereich*, kann auch Knoten D nicht gleichzeitig mit Knoten A senden. Signale von D stören den Empfang der Übertragung von A nach B bei B. Wird der RTS/CTS-Mechanismus eingesetzt, erkennt bereits B das Trägersignal von D und sendet kein CTS an A zurück. Der Faktor der nutzbaren Bandbreite fällt auf $\frac{1}{4}$.

Von der maximalen Bandbreite bleibt nur ein Viertel nutzbar, obwohl lediglich eine einzige Übertragung über mehrere Hops stattfindet. Die Situation verschärft sich mit mehreren konkurrierenden Anwendungen und Knoten.

2.2 Thematisch verwandte Arbeiten

Die folgenden Abschnitte stellen Ansätze und Protokolle aus den Bereichen *Bandbreitenberechnung*, *Admission Control* und *Bandbreitenkontrolle* vor. Die grundlegenden Fragen, welche angesprochen werden sollten, sind Folgende:

- Auf welche Art und Weise wird die verfügbare Bandbreite ermittelt,
- wie wird die entschieden, ob genügend Bandbreite für eine weitere Übertragung zugelassen wird und

- wie wird sichergestellt, dass nicht mehr als die zugeteilte Bandbreite genutzt wird.

Der erste Absatz „*Quality of Service in drahtlosen Netzen*“ beschreibt zunächst allgemein die Arbeiten, welche Konzepte zur Garantie von Dienstgüten in drahtlosen Netzen erarbeitet haben.

Die darauf folgenden Abschnitte widmen sich der Fragestellung, auf welche Weise die drei Teilaufgaben *Bandbreitenberechnung*, *Admission Control* und *Bandbreitenkontrolle* in diesen Arbeiten gelöst wurden und wo Vor- und Nachteile der einzelnen Lösungen liegen. Dazu werden die eingangs kurz vorgestellten Arbeiten, wenn sie thematisch passen, nochmals aufgegriffen und detaillierter beschrieben.

2.2.1 Quality of Service in drahtlosen Netzen

Um Anwendungen bestimmte Dienstgüten (*Quality of Service*) zu garantieren, müssen die unteren Schichten eines Netzwerkstacks diese Garantien unterstützen. Verschiedene Arbeiten nutzen unterschiedliche Ansätze, um diese Unterstützung zu erreichen. Die folgenden Abschnitte beschreiben verschiedene Arbeiten, welche das Ziel „Dienstgütegarantien in drahtlosen Netzen“ haben und deren Annahmen und Lösungsvorschläge.

IEEE 802.11e

Die Erweiterung IEEE 802.11e [6, 11] führt Mechanismen für die Unterstützung von Dienstgüten als Erweiterung in den Standard IEEE 802.11 ein.

Neben der *DCF* und *PCF* in 802.11 (*Distributed Coordination Function* und *Point Coordination Function*) führt die Erweiterung die *HCF*, die *Hybrid Coordination Function* ein. Der Zugriff auf das Medium erfolgt dabei durch eine erweiterte *DCF*.

Die Dauer der Wartezeiten, *Backoffs* und *Inter Frame Spaces* werden dabei modifiziert. Ein längerer *Inter Frame Space* verringert die Wahrscheinlichkeit eine Nachricht zu senden, ein kürzerer *Backoff* erhöht diese Wahrscheinlichkeit. Durch Variation dieser Parameter entstehen verschiedene *Access Categories* und *Priorities*. Auf diese Weise können einzelne Pakete mit unterschiedlichen Prioritäten versendet werden.

Kurz gefasst: Nachrichten mit höherer Priorität müssen eine kürzere *Backoff*-Zeit warten und werden deshalb mit höherer Wahrscheinlichkeit versendet. Garantien, dass eine Nachricht zu einer bestimmten Zeit versendet oder empfangen wird, gibt es nicht.

Wu et al [15] nutzen die Erweiterung des Standards für ihr Protokoll.

AQOR

AQOR[25] ist ein reaktives Routingverfahren, welches Bandbreitenreservierungen

durchführt und Ende-zu-Ende-Verzögerungen berücksichtigt.

Die Zusicherung von Dienstgüten wird erreicht, indem bei der Erstellung von Routen geforderte Kriterien berücksichtigt bzw. festgestellt werden.

Verletzungen von Dienstgüteforderungen werden vom Ziel erkannt, welches daraufhin ein Neu-Routing startet („route update“). Eine QoS-Verletzung wird also wie eine Routenunterbrechung behandelt.

MACMAN

[17] nutzt einen Monitoring-Ansatz, die *Busy-Time*-Messung, zur Bandbreitenbestimmung. Der Ansatz verspricht hohe Exaktheit und geringe zusätzliche Last durch die passive Herangehensweise.

Das Zulassen von Strömen erfolgt abhängig von ausreichend verfügbarer Bandbreite. Die notwendige Bandbreite wird errechnet aus dem *Contention Count* und der Bandbreite des Stromes. Der *Contention Count* ist ein Multiplikator, abhängig von der Zahl der Sende-/Empfangsvorgänge an dem entsprechenden Knoten. Zu dessen Berechnung ist die Kenntnis der Knoten im Carrier-Sense-Bereich notwendig. Es wird vorgeschlagen, diese Informationen durch periodische „high-power transmissions“, also Übertragungen mit erhöhter Energie und Reichweite, durchzuführen. Der Einsatz von Spezialhardware ist in diesem Falle ebenso unvermeidbar, wie die Vergrößerung der Sende-/Empfangsreichweite *und* des *Carrier-Sense-Bereich.*

MMWN – Multimedia support for mobile wireless networks

[22] beschreibt ein System, welches für allgemeine drahtlose Netze entwickelt wurde. Die Knoten werden hierarchischen Clustern zugeordnet und verwalten Verbindungsinformationen zwischen den Clustern als „link states“. Die Eigenschaften (Verzögerung, Wahrscheinlichkeit des Paketverlustes) des Clusters werden von dedizierten Knoten, Managern, berechnet und verwaltet. Zu diesen Eigenschaften zählen auch die Dienstgüte-Eigenschaften der Verbindungen („virtual links“) zwischen den Gateway-Knoten („virtual gateways“). Optimale QoS-Eigenschaften für eine Route werden durch einen *shortest path*-Algorithmus berechnet. Jeder QoS-Manager der niedrigsten Clusterebene kann auf Informationen (die „link-states“) aller Cluster der Ebene über ihm nutzen. Auf diese Weise werden die nötigen Informationen für Einhaltung von Dienstgüten verbreitet.

CACP – contention-aware admission control protocol

In [26] wird das CACP (contention-aware admission control protocol) beschrieben, ein Routingprotokoll mit Zugangskontrollmechanismen (*admission control*). Basierend auf der Überwachung (*monitoring*) der subjektiv verfügbaren Bandbreite wird entschieden, ob Ströme zugelassen werden.

CACP ist in zwei Hauptbestandteile gegliedert: Routenfindung und Zugangskontrolle. Das Routing benutzt ein *on-demand*-Verfahren ähnlich Direct Source Routing (DSR) [14].

Die Entscheidung, ob ein Strom lokal an einem Knoten zugelassen wird, hängt von der erwarteten Bandbreitennutzung des Stromes und der verfügbaren Bandbreite des Knotens und aller seiner Nachbarn in *Carrier-Sense*-Reichweite ab.

Die Bandbreite, die der Strom nutzen wird, bestimmt im CACP-Protokoll der *contention-count*. Diese Zahl errechnet sich aus dem Pfad, welcher das Paket nehmen wird und aus Information über Knoten aus der *Carrier-Sense-Nachbarschaft*.

2.2.2 Bandbreitenberechnung

Die Ermittlung der Bandbreite in drahtlosen Netzen ist nach zwei grundsätzlichen Methoden möglich: Die erste ist die Abschätzung anhand der gemessenen freien Zeit (*idle time*) des Netzes. Der andere Ansatz ist die Protokollierung und Berechnung jeder Bandbreitennutzung.

Der erste Ansatz wird z.B. von [4] genutzt. Der Vorteil ist die Einfachheit des theoretischen Ansatzes. Die *Idle-Time* ist genau die Zeit, welche der *Carrier-Sense*-Mechanismus als „Medium frei“ erkennt. Das Problem ist die praktische Umsetzung. Das Carrier-Sense-Signal wird lediglich intern in der Hardware ausgewertet (ausgehend von weit verfügbarer „Standard“-Hardware). Weder über den Protokoll-Stack, noch über Treiber gelingt die Erfassung dieser Zeiten. Eine indirekte Methode wird von [17] vorgeschlagen. Wenn Pakete versendet werden, können Rückschlüsse auf die *Busy-Time* (die Zeit wenn das Medium belegt ist) gezogen werden, wenn man Zugriff auf Informationen der Ausgangswarteschlange des Netzwerk-Stacks hat. Präzise ist diese Art der Bestimmung aber nur, wenn ständig Pakete in der Ausgangswarteschlange sind und anhand dieser Sendezeiten die *Busy-Time* bestimmen kann.

Die zweite Methode ist präzise, wenn die Möglichkeit besteht, von allen Knoten in der Bandbreitennachbarschaft Informationen zu erhalten. Ein Zugriff auf die Hardware ist nicht notwendig. Die Schwächen dieses Ansatzes liegen in der mangelnden Möglichkeit unbekannte Faktoren, wie Störungen oder Knoten, welche nicht das Protokoll ausführen, zu beachten. Ein Beispiel für diesen Ansatz ist AQOR-Protokoll [25], welches der folgendende Unterabschnitt näher betrachtet.

AQOR

Das „Ad hoc QoS on-demand Routing“-Protokoll soll die vorhandene Bandbreite in der Umgebung der Knoten exakt berechnen. Zu diesem Zweck betrachtet [25] alle möglichen Übertragungen in der Sende-/Empfangsumgebung eines Knotens. Die drei Arten sind

- Kommunikation des Knotens mit einem Nachbarn

- Kommunikation von zwei Nachbarn untereinander
- Kommunikation eines Nachbarn mit einem Knoten außerhalb des Sende-/ Empfangsbereichs

Die „aggregierte Bandbreite“ des Knotens I ($B_{agg}(I)$) setzt sich aus der Kommunikation des Knotens I mit seinen Nachbarn ($B_{self}(I)$), der Kommunikation der Nachbarn untereinander ($B_{neighbourhood}(I)$) und der Übertragungen von Knoten über die Grenzen des Sende-/Empfangsbereichs hinaus ($B_{boundary}(I)$) zusammen (Gleichung 2.4).

$$B_{agg}(I) = B_{self}(I) + B_{neighbourhood}(I) + B_{boundary}(I) \tag{2.4}$$

Knoten I kann den Nachbarschafts- und Grenzverkehr nicht selbst bestimmen und benötigt diese Informationen von den anderen Knoten.

[25] schätzt die aggregierte Bandbreite $B_{agg}(I)$ mit der Summe des Eigenverkehrs aller Nachbarn ab (Gleichung 2.5).

$$B'_{agg} = \sum_{J \in N(I)} B_{self}(J) \tag{2.5}$$

Auf diese Weise wird mehr Bandbreite berechnet, als wirklich verbraucht wird. Wird der Nachbarschaftsverkehr zwischen den Knoten B und C erzeugt, gilt Gleichung 2.6

$$B_{self}(B) = B_{self}(C) = B_{neighbourhood} \tag{2.6}$$

Berechnet man die Gesamtbandbreite nach Gleichung 2.5, wird $2 * B_{neighbourhood}$ zum Eigenverkehr und Grenzverkehr addiert, d.h. $B'_{agg} = B_{agg} + B_{neighbourhood}$.

Die gesamte, reservierte Bandbreite eines Knotens ($B_{self}(I)$) wird periodisch mit einer „Hello“-Nachricht verschickt.

MMWN – Multimedia support for mobile wireless networks

Das MMWN-Protokoll [22] verbreitet Eigenschaften der Knoten in einem hierarchischen Clustersystem, d.h. benachbarte Knoten, welche bereits selbst Cluster sein können, werden zu Clustern gruppiert. Diese Gruppen werden wiederum ihrerseits zusammengefasst. Zu den Eigenschaften, welche jeder Cluster besitzt, zählen neben QoS-Eigenschaften auch Informationen über freie und reservierbare Übertragungskapazitäten.

In höheren Clustern wird die Bandbreite einfach als eine Eigenschaft des Clusters gespeichert. Bandbreitenbelegungen durch nicht sichtbare Knoten und andere eingangs erläuterte Effekte spielen auf der hohen Gruppierungsebene keine Rolle mehr. Auf der unterster Ebene muss dies aber betrachtet werden. Die genutzte Bandbreite wird an die unmittelbaren Nachbarknoten verbreitet, welche daraufhin ihre eigene freie und reservierbare Bandbreite reduzieren. Eine gesonderte Betrachtung der Randbereiche der Cluster ist nicht beschrieben.

2.2.3 Reservierung und Zugriffskontrolle

Verletzungen von Dienstgütegarantien können auftreten, wenn zwei Applikationen die selben Ressourcen zur gleichen Zeit nutzen wollen. Die Folge: Es stehen beiden nicht genügend Ressourcen zur Verfügung. Reservierungen beseitigen diesen Konflikt, indem sie genau festlegen, welche Ressource bzw. welcher Anteil an einer Ressource welcher Anwendung zur Verfügung steht. Auf diese Weise kann einer Anwendung eine bestimmte Ressource garantiert werden.

Die ersten Versuche, Dienstgüte (Quality of Service - QoS) in drahtlosen Netzen zu ermöglichen, benutzten Zeitschlitz-basierte TDMA-Verfahren (slotted Time Division Medium Access) [8, 26]. Bei derartigen Verfahren wird die zur Verfügung stehende Sendezeit in Abschnitte unterteilt, so genannte Zeitschlitze – Timeslots. Meist wird jedem Knoten im Netzwerk ein Zeitschlitz zugeteilt. Die Zeitschlitze wiederholen sich periodisch nacheinander in fester Reihenfolge. Ein Knoten darf nur während des ihm zugewiesenen Zeitabschnittes senden. Die zur Verfügung stehende Bandbreite pro Knoten (ρ_A für Knoten A) lässt sich aus der maximalen Übertragungsrate (ρ_{max}) und der Anzahl der Zeitschlitze (n) errechnen: $\rho_A = \frac{\rho_{max}}{n}$. Eine unmittelbare Konsequenz dieses Vorgehens ist, dass sich mit steigender Knotenzahl n im Netz die für jeden einzelnen Knoten verfügbare Bandbreite reduziert. *Räumliches Multiplexen* (*Spatial reuse* oder auch *SDMA*) ist ausgeschlossen. Das bedeutet in diesem Fall, dass Knoten sich beim Senden nicht gegenseitig beeinflussen, da sie so weit voneinander entfernt sind, aber trotzdem nicht senden dürfen.

Hsu et al [8] weichen von den eben dargestellten, grundsätzlichen Eigenschaften eines *slotted TDMA*-basierten Protokolls ab, um die negativen Eigenschaften zu minimieren: Das *slotted TDMA*-Protokoll wird nur innerhalb eines Teils, eines Clusters, des Netzes benutzt. Die Cluster nutzen, um sich nicht gegenseitig zu beeinflussen, verschiedene Frequenzen. Die Zeit wird in zwei Abschnitte aufgeteilt: Kontrollphase und Übertragungsphase. Während der Kontrollphase verbreitet jeder Knoten, welche Slots (Zeitschlitze) bei ihm noch frei sind. Diese Informationen leiten alle Knoten, welche diese Information empfangen haben, weiter. Beispielsweise teilt Knoten A mit, dass er die Zeitschlitze 1 und 3 frei hat. Knoten B empfängt dies und hat selbst die Schlitze 1 und 2 frei. Zusätzlich zu den eigenen freien Zeitschlitzen verbreitet B, dass der Zeitschlitz 1 genutzt werden kann, um A zu erreichen. Diese Menge der freien Zeitschlitze auf der Route zu einem Knoten ist die Schnittmenge der freien Slots von A und B. Eine Station kann nun über B mit Knoten A in dem ausgewiesenen Zeitschlitz kommunizieren, wenn eine entsprechende Reservierung durchgeführt wird.

Georgiadis et al [5] unterteilen die Zeit ebenfalls in Zeitschlitze mit der Länge der (maximalen) Sendedauer eines Paketes und nutzen eine getrennte Reservierungs- und eine Übertragungsphase. Sie beschreiben einen Algorithmus, welcher Reservierungen von Zeitschlitzen durchführt. Dabei wird der *Carrier-Sense-Bereich* beachtet, also

mehr als nur die direkten Links betrachtet. Es wird demonstriert, dass ein einfacher Scheduling-Algorithmus, welcher die Zuordnung der Ströme zu den Zeitschlitzen durchführt, die verfügbare Bandbreite nicht effektiv nutzen kann. Die Autoren entwickeln einen Scheduling-Algorithmus, welcher die Zuordnung deutlich effektiver machen soll. Diese Überlegungen werden in einen Routenreservierungsalgorithmus integriert und Überlegungen zur Integration in das OLSR-Protokoll [2] gemacht.

In [19, 18] wird allerdings gezeigt, dass die zur Nutzung von Zeitschlitz-basierten Verfahren notwendige Uhrensynchronisation in MANETs nur mit eingeschränkter Präzision bzw. modifizierter Soft- oder Hardware möglich ist.

Die Reservierung der Bandbreite hängt bei vielen routingbasierten Protokollen, wie etwa AQOR [25], eng mit der Routenfindung zusammen. Bei der Suche nach einer passenden Route werden auch die vorhandenden freien Bandbreiten beachtet und nur Routen in Erwägung gezogen, die genügend Ressourcen haben. Das Protokoll sieht ein Antwortpaket vom Ziel zur Quelle vor. Trifft dieses Paket mit den Routeninformationen ein, wird das erste Datenpaket versendet, welches die Route festlegt und auch die Bandbreiten reserviert.

In den ge*cluster*ten Netzen von MMWN (Multimedia support for mobile wireless networks) [22] erfolgt die Reservierung der Bandbreite nach einem ähnlichen Ansatz. Es nutzt jedoch die Vorteile der *Cluster*, um auf Änderungen einfacher reagieren zu können. Während der Erstellung der „virtual circuits“ werden neben den Routen auch Bandbreiten festgelegt. Die „virtual circuits“ sind aber keine statischen Routen, sondern werden über Cluster adressiert. Die einzelne Hops werden dynamisch in jedem Cluster neu berechnet.

3 Konzept

Dieses Kapitel erläutert die konzeptionelle Vorgehensweise zum Erreichen des eingangs erläuterten Ziels einer Bandbreitenzuteilung und -Überwachung in einem drahtlosen Multi-Hop-Ad-Hoc-Netzwerk.

Im ersten Abschnitt werden die Grundlagen und allgemeinen Ansätze aus anderen Arbeiten analysiert und deren Eignung für die Erfüllung der zu erreichenden Ziele bewertet.

Der Frage, wie die in einem drahtlosen Medium zur Verfügung stehende Bandbreite berechnet wird, widmet sich der zweite Abschnitt.

Der dritte Abschnitt beschreibt, wie die Bandbreitennachbarschaft, also die Menge Knoten, welche sich Bandbreite teilen, bestimmt werden kann.

Der letzte Abschnitt diskutiert die Möglichkeiten zum Erreichen einer Entscheidung über die Nutzung vorhandener Bandbreite und wie die Gewährleistung der Konsistenz dieser Operationen.

3.1 Eignung anderer Arbeiten

Erste Arbeiten zum Thema „Quality of Service in Multi-Hop-Ad-hoc-Netzen“ benutzen das TDMA-Verfahren (Time Division Medium Access) mit Zeitschlitzen (Timeslots) als Medienzugriffsmechanismus ([26]). Ähnlich dem Verfahren, welches in GSM-Netzen ([24]) genutzt wird, sind die Zeitschlitze auf allen beteiligten Knoten synchronisiert, d.h. jeder Knoten beginnt genau zur selben Zeit einen bestimmten Timeslot. Alle Sendevorgänge finden in genau einem Zeitschlitz statt. Welcher Knoten wann senden darf, ist entweder von Anfang an definiert (es gibt pro Periode genau einen Zeitschlitz pro Knoten) oder wird in einer Registrierungsphase vor der eigentlichen Sendephase bestimmt.

Jeder Knoten darf nur in dem ihm zugewiesenen Timeslot senden, empfangen in allen Übrigen. Sind alle Zeitschlitze aller Knoten einmal aktiv gewesen, beginnen sie in derselben Reihenfolge wieder von vorne. Auf diese Weise ist es jedem Sender und Empfänger möglich, genau zu berechnen, wann ein Paket spätestens gesendet, bzw. empfangen werden kann (Verzögerung – Delay). Die zur Verfügung stehende Bandbreite kann lokal anhand der Länge des Zeitschlitzes und der maximalen Bandbreite berechnet werden.

Der Vorteil der Berechenbarkeit der genutzten Bandbreite wird dadurch gewonnen, dass nur ein Knoten im gesamten Netz senden darf. Demgegenüber steht das

Problem, dass Bandbreite ungenutzt bleibt, wenn Knoten zwar einen Zeitschlitz reserviert haben, aber nichts versenden. In MANETs hat dieses Verfahren zusätzlich prinzipielle Probleme: Knoten in diesen Netzen sind nicht synchronisiert. Sollen die Uhren aller Knoten im Netz synchronisiert werden, entsteht ein erhöhter Aufwand und damit verbundener Overhead.

Noch höher ist die „Verschwendung“ von Bandbreite durch die Nicht-Ausnutzung des räumlichen Abstandes, auch als *spatial reuse* bezeichnet. Befindet sich ein anderer Knoten gar nicht im Carrier-Sense- oder Interferenzbereich des Anderen, darf er nicht gleichzeitig senden, obwohl keinerlei Störungen auftreten würden. Auf diese Weise wird die ohnehin knappe, nutzbare Bandbreite weiter eingeschränkt. Aus diesem Grund verwenden Protokolle, welche das Timeslot-basierte TDMA-Verfahren nutzen, abgewandelte Verfahren. So sollen die oben erwähnten Nachteile ausgeglichen werden.

Soll eine Implementierung mit derzeit weit verbreiteter und verfügbarer Hardware erfolgen, scheidet das TDMA-TimeSlot-Verfahren aus. Geräte, auf die dies derzeitig zutrifft, setzen den Standard IEEE802.11 [12] um, genauer die Teile b oder g. In die Medienzugriffschicht (MAC-Layer) ist der CSMA/CA-Mechanismus fest integriert.

Das zu entwickelnde Protokoll soll mit unmodifizierter „Standard-Hardware“ arbeiten. Der im Standard festgelegte und in der Hardware implementierte Zugriffsmechanismus muss also genutzt werden, da die Hardware (bzw. Firmware) der Karten nicht modifiziert werden kann. Naheliegend ist, dass und auch kein Verfahren implementiert werden kann, welches dem implementierten Mechanismus entgegen arbeitet.

Andere Arbeiten benutzen Scheduling-Ansätze, um das Ziel *QoS* oder eher „*better than best effort*“ zu erreichen. *Better than best effort* soll bedeuten, dass keine Hard-Realtime-Anforderungen erfüllt werden können. Gegenüber „normalem“ (*best effort*) Verkehr sollen die bevorzugten Ströme (streams) die Dienstgüteanforderungen von Applikationen gewährleisten können, wenn bestimmte Bedingungen erfüllt sind. (eingeschränkte Mobilität der Knoten, keine externen, nicht kontrollierbaren Störquellen). Hinzu kommen systembedingte Einschränkungen der genutzten Medienzugriffschicht (*MAC-Layer*). Das zur Kollisionsvermeidung genutzte *CSMA/CA*-Verfahren soll Kollisionen minimieren. Bei großer Knotenzahl führen die beschränkten, diskreten Wartezeiten aber mit hoher Wahrscheinlichkeit zu Kollisionen und damit zu einer nicht mehr funktionsfähigen Medienzugriffskontrolle.

Das weitere Kapitel ist unterteilt in die hauptsächlichen Problemstellungen. Die exakte Berechnung der Bandbreite ist eine Voraussetzung zur Entwicklung des Protokolls. Ein Lösungsvorschlag dazu wird zuerst beschrieben. Zur Bestimmung der Bandbreite ist die Kenntnis der Nachbarschaft nötig. Dies ist der erste Teil des zu entwickelnden Protokolls. Wie die Knoten in der Nachbarschaft innerhalb von zwei Hops ermittelt werden, wird im darauf folgenden Abschnitt beschrieben. Der dritte Abschnitt widmet sich der Fragestellung, wie kontrolliert werden kann, dass nicht mehr als die zugelassene Bandbreite genutzt wird. Der Traffic-Shaping-basierte Al-

gorithmus, der zweite Hauptteil des Protokolls wird beschrieben. Im letzten Absatz wird die Zugriffskontrolle diskutiert. Diese nutzt die von der Nachbarschaftserkennung ermittelten Knoten und erfragt deren genutzte Bandbreiten. Die verfügbare Bandbreite wird mit dem am Anfang ermittelten Algorithmus berechnet. Anhand dieser Bandbreite wird entschieden, ob bestimmte Applikationen zugelassen werden oder nicht. Die Entscheidung wird dem Traffic-Shaping mitgeteilt, welches diese Entscheidung durchsetzt.

3.2 Bandbreitenberechnung

Das Zulassen (*Admission*) eines Stromes muss aufgrund der exakten Kapazitäten (des betroffenen Teils) des Netzwerkes entschieden werden. Nur auf diese Weise kann sichergestellt werden, dass Ressourcen nicht ungenutzt bleiben oder gar nicht vorhandene belegt werden. Die Herleitung einer Formel bzw. eines Algorithmus, welcher die exakte Berechnung durchführt, ist Ziel dieses Abschnittes.

Um die zur Verfügung stehende Bandbreite zu berechnen, sind Informationen über Sende- und Empfangskapazitäten der einzelnen Nachbarknoten notwendig. Im Widerspruch zu der naheliegenden Annahme, dass ausschließlich die direkten Nachbarn in der eigenen Sende-/Empfangsreichweite (ein-Hop-Nachbarschaft) Bandbreite „verbrauchen“, stehen die Eigenschaften des Funkmediums (siehe 2.1.4) und der Medienzugriffsmechanismen der genutzten Hardware. Bandbreite wird auch von Knoten außerhalb des Sende-/Empfangsbereiches, in dem *Carrier-Sense-Bereich*, belegt.

Knoten in dieser *Carrier-Sense-Nachbarschaft* können Pakete nicht mehr oder nicht vollständig dekodieren, empfangen aber das Trägersignal, den Carrier. Ein Knoten in diesem Bereich könnte einen Sendevorgang stören, deshalb sieht IEEE 802.11 ([12]) vor, dass Knoten, welche einen belegten Carrier empfangen, nicht senden dürfen.

Durch diese Konstellation ergibt sich für jeden Knoten eine verfügbare Bandbreite, die in hohem Maße von den Nachbarn und deren Position abhängt. Eine geringe Veränderung der Position kann dazu führen, dass sich die verfügbare Bandbreite ändert. Dies macht noch einmal deutlich, dass Bandbreite nicht überall im Medium gleich ist, sondern an jeder Position für jeden Knoten eine individuelle, „subjektive“, Bandbreite existiert.

Das folgende Beispiel illustriert die möglichen Fälle für Übertragungen innerhalb der verschiedenen Empfangsbereiche des Knotens „O“. Abbildung 3.1 zeigt Knoten O mit den markierten Grenzen der Sende-/Empfangs- und Carrier-Sense-Bereiche. Alle Nachbarn (A bis L) verschicken untereinander Nachrichten (durch Pfeile markiert). Knoten O soll die Bandbreite in seiner Umgebung bestimmen. Er verfügt über Informationen von allen Knoten innerhalb des Carrier-Sense-Bereichs (alle Knoten außer I, L, K). Die Sende- bzw. Empfangsvorgänge werden in Tripeln dargestellt.

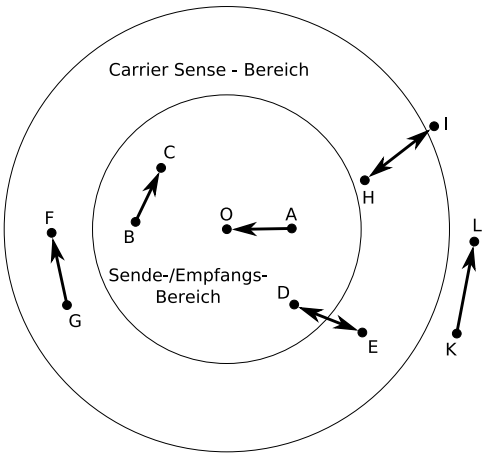


Abbildung 3.1: Beispieltopologie um Knoten O;Sende-/Empfangs- und Carrier-Sense-Bereich; Verbindungen zwischen den Knoten

Diese haben die Form $S(B, C, i)$ mit der Bedeutung „B sendet an C mit Bandbreite i“ bzw. $R(C, B, k)$ für „C empfängt von B mit Bandbreite k“.

Die Übertragungen in der Umgebung eines Knotens, welche alle möglichen Verbindungsarten enthalten, kann man in Gruppen fassen, wie in Tabelle 3.1 dargestellt.

Während der Durchführung des Zugriffskontrollalgorithmus (admission control) werden die Bandbreiteninformationen ermittelt. Die Wege über welche die Kommunikation geschieht, folgt dabei den durch die Nachbarschaftserkennung gefundenen Wege. O erhält so im Beispiel folgende Informationen: $S(A, O, a)$, $R(O, A, a)$, $S(B, C, b)$, $R(C, B, b)$, $S(D, E, c)$, $R(E, D, c)$, $S(E, D, d)$, $R(E, D, d)$, $S(G, F, e)$, $R(F, G, e)$, $S(H, I, f)$, $R(H, I, g)$

Informationen von Knoten, welche nicht an O weitergeleitet werden sind: $S(I, H, i)$, $R(I, H, h)$, $S(K, L, k)$, $R(L, K, k)$. Die Knoten, welche diese Informationen erzeugen, liegen nicht in der Bandbreitennachbarschaft. Deshalb beeinflussen die Übertragung von K zu L die Bandbreite der Umgebung von O nicht. Die Sende- bzw. Empfangsinformationen von Is Kommunikation mit H werden indirekt von H (durch das Tripel $R(H, I, g)$) weitergegeben.

Addiert man sämtliche Bandbreitenwerte, liegt der ermittelte Wert weit über der realen Bandbreite, weil Verbindungen doppelt gezählt werden. Sendet B mit Bandbreite b an Knoten C, so nutzt diese Übertragung die Bandbreite von b. C teilt mit, dass es von B eine Übertragung erhält. Summiert man die Werte dieser Nachrichten auf, ergibt sich ein Bandbreitenverbrauch von $2 * b$. Die Verbindung wird nicht als eine Einzelne erkannt, sondern als zwei unabhängige Sende-/Empfangsvorgänge gezählt. Durch die netzweit eindeutige Kennungen der Knoten, können die doppelt

Sender und Empfänger im Sende-/Empfangsbereich (1-Hop-Nachbarschaft)	$\Rightarrow S(B, C, a)$
Sender und Empfänger im Carrier-Sense-Bereich (außerhalb Sende-/Empfangsbereich, 2-Hop-Nachbarschaft)	$\Rightarrow S(G, F, b)$
Sender und Empfänger außerhalb des Carrier-Sense-Bereichs	$\Rightarrow S(K, L, c)$
Sender im Sende-/Empfangsbereich, Empfänger im Carrier-Sense-Bereich (außerhalb Sende-/Empfangsbereich)	$\Rightarrow S(D, E, d)$
Empfänger im Sende-/Empfangsbereich, Sender im Carrier-Sense-Bereich (außerhalb Sende-/Empfangsbereich)	$\Rightarrow S(E, D, e)$
Sender im Carrier-Sense-Bereich (außerhalb Sende-/Empfangsbereich), Empfänger außerhalb des Carrier-Sense-Bereichs	$\Rightarrow S(H, I, f)$
Sender außerhalb des Carrier-Sense-Bereichs, Empfänger im Carrier-Sense-Bereich (außerhalb des Sende-/Empfangsbereichs)	$\Rightarrow S(I, H, g)$

Tabelle 3.1: Verbindungsarten in der Umgebung eines Knotens

gemeldeten Verbindungen erkannt werden. Dies geschieht, indem den Sendetripeln die Empfangstripel zugeordnet werden. Tabelle 3.2 zeigt die Zuordnung der Sendetripel zu den Empfangstripeln, welche die gleiche Übertragung repräsentieren. Pro Zeile steht eine Kommunikation mit einer Bandbreitengröße. Diese Größen pro Zeile (dritte Spalte) addiert ergeben die genutzte Bandbreite in der Umgebung des Knotens O.

Es fällt auf, das bis auf die letzten zwei Zeilen, immer Sende- *und* Empfangstripel angegeben sind. Der Grund dafür ist, dass Sender *und* Empfänger befragt werden, wenn sie innerhalb des Carrier-Sense-Bereichs liegen. Knoten I, welcher außerhalb des Carrier-Sense-Bereichs liegt, wird seine Bandbreiteninformationen nicht verbreiten. H, welcher Daten an I sendet, nutzt einen Teil der Bandbreite von O.

Bleibt die Frage zu klären: Belegt I während des Sendens Bandbreite aus der Umgebung von O? Ist dies nicht der Fall, ist es ausreichend, wenn die Knoten Informationen über die eigenen gesendeten Ströme verbreiten.

Die Antwort hängt von der Einstellung der optionalen RTS/CTS-Funktion ab. Ohne das RTS/CTS sendet I und O empfängt keinerlei Signale, d.h. das Auswerten der gesendeten Ströme wäre ausreichend. Die Bestätigungssignale (ACK) von H nehmen allerdings (minimale) Bandbreite von O in Anspruch. Außerdem könnte das Signal von O den Empfang bei H stören, auch wenn es schwächer als das Signal von I ist.

Wird das RTS/CTS-Schema genutzt, beginnt I ein RTS zu senden und H antwor-

Sendetripel	Empfangstripel	resultierende Bandbreite der einzelnen Verbindungen
S(A,O,a)	R(O,A,a)	a
S(B,C,b)	R(C,B,b)	b
S(D,E,c)	R(E,D,c)	c
S(E,D,d)	R(E,D,d)	d
S(G,F,e)	R(F,G,e)	e
S(H,I,f)		f
	R(H,I,g)	g
		a+b+c+d+f+g

Tabelle 3.2: Zuordnung der Sendetripel zu Empfangstripeln, welche gleichen Verbindungen entsprechen

tet mit einem CTS. Alle Knoten, welche das CTS empfangen, verhalten sich während der Übertragungszeit von I so, als würde H senden. Es ist wahrscheinlich, dass O das CTS nicht dekodieren kann, weil H außerhalb des Sende-/Empfangsbereichs liegt. Während des Versendens des CTS selbst und aller Empfangsbestätigungen (ACKs), kann O das Carrier-Signal feststellen und wird nicht senden.

Zur Lösung kommen zwei Varianten in Frage:

1. Das Senden von I wird ignoriert, da es O nicht beeinflusst. Die Bestätigungsnachrichten von H belegen minimale Bandbreite. Die errechnete freie Bandbreite liegt also minimal über der Realen. Die Differenz, etwa durch Störungen in der Umgebung, muss über einen Puffer ausgeglichen werden.

2. Es wird so verfahren, als ob das Senden von I Bandbreite von O nutzt. Auf diese Weise wird weniger Bandbreite als in der Umgebung verfügbar, errechnet.

Für diese zwei Fälle sollen nun Formeln gefunden werden. Zu diesem Zweck wird statt der Tripel eine andere Schreibweise für die Bandbreiten bei Sende- bzw. Empfangsvorgängen genutzt: $b^S_{B,C}$ bedeutet „die Bandbreite b, welche der Sendevorgang von B an C benötigt“. Umgekehrt steht $b^R_{C,B}$ für „die Bandbreite b, welche der Empfang (Receive) bei C von B beansprucht“. Zusätzlich werden die Mengen N_C und $N_{S/R}$ für alle Knoten des *Carrier-Sense-Bereichs* bzw. des Sende-/Empfangsbereichs definiert. Es gilt: $N_{S/R} \subset N_C$. Die Summe aller Sende- und Empfangsvorgänge im Carrier-Sense-Bereich kann wie in Gleichung 3.1 berechnet werden.

$$\sum_{I,J \in N_C} b^R_{I,J} + b^S_{I,J} (I \neq J) \tag{3.1}$$

In dieser Gleichung werden die Bandbreiten für Knoten im Sende-/ Empfangsbereich (außer dem Knoten O, welcher die Bandbreite berechnet) doppelt gezählt. Dies wird gelöst, indem nur die Bandbreiten der Sendevorgänge gezählt werden (siehe Tabelle 3.2. Gleichung 3.2 erfüllt die Bedingungen der ersten Variante.

$$\sum_{I,J \in N_C} b^S_{I,J} (I \neq J) \tag{3.2}$$

Wird die zweite Variante gewählt, müssen noch die Bandbreiten der Knoten in der *Carrier-Sense-Nachbarschaft* hinzugefügt werden, welche Daten von Knoten außerhalb des *Carrier-Sense-Bereiches* empfangen. Gleichung 3.3 erfüllt diese Bedingung, indem zu allen Sendebandbreiten der Knoten im Carrier-Sense-Bereich die Empfangsbandbreiten von Übertragungen addiert werden, bei welchen sich der Sender außerhalb des Carrier-Sense-Bereichs befindet.

$$\sum_{I,J \in N_C, K \notin N_C} b^S_{I,J} + b^R_{I,K} (I \neq J) \tag{3.3}$$

Eine Umsetzung dieser Gleichungen in einen Algorithmus ist im folgenden Pseudocode (Tabelle 3.3) für die erste Variante beschrieben. Die Eingabe besteht aus zwei Listen, die eine enthält alle Sende- die andere alle Empfangstripel. Die Ausgabe, die genutzte Bandbreite, enthält den Bandbreitenzähler. Der zweite Algorithmus (Tabelle 3.4) bezieht die Sendeliste und Empfangsliste mit ein, um die zweite Variante (Knoten, welche von außerhalb des Carrier-Sense-Bereichs senden, werden mit einberechnet) einzubeziehen.

```
Bandbreitenzähler=0;
ersten Eintrag der Sende-Liste auswählen;
solange das Listenende nicht erreicht ist:
   lies Bandbreite aus dem Eintrag;
   addiere Bandbreite zum Bandbreitenzähler;
   nächsten Eintrag in der Liste auswählen;
wiederhole;
```

Tabelle 3.3: Variante 1 des Algorithmus zur Berechnung der Bandbreite

3.3 Bestimmung der Nachbarschaft - Multi-Hop-Beaconing

In diesem Abschnitt wird beschrieben, wie Informationen über die Nachbarschaft eines Knotens gewonnen werden können.

```
Bandbreitenzähler=0;
ersten Eintrag der Sendeliste auswählen;
solange das Ende der Empfangsliste nicht erreicht ist:
   Eintrag in Empfangsliste suchen und löschen;
   lies Bandbreite aus dem Sendelisten-Eintrag;
   addiere Bandbreite zum Bandbreitenzähler;
   nächsten Eintrag in der Sendeliste auswählen;
wiederhole;
ersten Eintrag der Empfangsliste auswählen;
solange das Ende der Sendeliste nicht erreicht ist:
   lies Bandbreite auswählen dem Empfangslisten-Eintrag;
   addiere Bandbreite zum Bandbreitenzähler;
   nächsten Eintrag in der Empfangsliste auswählen;
wiederhole;
```

Tabelle 3.4: Variante 2 des Algorithmus zur Berechnung der Bandbreite

Zur Durchführung kooperativer Aktionen ist die Kenntnis möglichst aller Knoten in der Umgebung notwendig. Ohne Kenntnis von Kommunikationspartnern und gezielte Kommunikation mit ihnen ist es nicht möglich, eine gemeinsame Entscheidung zu treffen.

Der Abschnitt 2.1 diskutiert die Besonderheiten bei der Ausbreitung von Funkwellen im WLAN und die daraus resultierenden Einflüsse auf die verfügbare Bandbreite in der Umgebung eines Knotens. Die wichtigste Erkenntnis: Auch die verfügbare Bandbreite von Knoten außerhalb des direkten Sende-/Empfangsbereiches wird beeinflusst. Die Kenntnis der Knoten innerhalb der eigenen Sende-/Empfangsreichweite ist also nicht ausreichend, wenn man die Menge der Knoten erstellen will, welche von einem Sendevorgang beeinflusst werden. Außer den eigenen Nachbarn ist es auch notwendig deren Nachbarn zu kennen, welche man selbst aber nicht mehr empfängt (*Zwei-Hop-Nachbarschaft*) oder sogar deren Nachbarn und mehr (*n-Hop-Nachbarschaft*).

Um Knoten zu finden, muss man mit ihnen kommunizieren. Zwei Knoten können gezielt Nachrichten versenden, wenn sie sich in gegenseitiger Reichweite befinden und die Adresse des anderen Knotens kennen. Soll allerdings die gesamte Nachbarschaft erkundet werden, also auch Knoten deren Existenz und damit deren Adresse unbekannt ist, müssen andere Möglichkeiten genutzt werden, z.B. *Broadcast*-Nachrichten – Nachrichten, adressiert „an alle“.

Es gibt zwei Vorgehensweisen wenn Nachbarschaftsinformationen benötigt wer-

den. Eine Möglichkeit ist eine reaktive Vorgehensweise. Erst wenn eine Applikation Informationen über die eigene Nachbarschaft des Knotens benötigt, sendet sie per *Broadcast* eine Anfrage aus. Alle Knoten, welche die Anfrage von diesem Knoten empfangen, reagieren mit einer Antwort. Aus den eingehenden Informationen liest die benötigten Adressen aus und erstellt so die *1-Hop-Nachbarschaft*. Das andere Vorgehen („proaktiv“) ist eine regelmäßig gesendete Nachricht (*Beacon* – Leuchtfeuer), so dass sich sofort alle Knoten in der unmittelbaren Umgebung zu erkennen geben. Der Nachteil hierbei ist die höhere verbrauchte Bandbreite.

Bei der Erstellung der Zwei-Hop-Nachbarschaft (oder allgemeiner die *n-Hop-Nachbarschaft*) reichen die Fähigkeiten eines einzelnen Tranceivers nicht mehr aus. Nachrichten müssen über mehrere Hops weitergeleitet werden. Das *Beacon* wird mit weiteren Informationen angereichert. Es enthält eine Liste der Nachbarschaft des sendenden Knotens. Diese Informationen werden in die eigenen Nachbarschaftsinformationen integriert. Soll eine n-hop-Nachbarschaft aufgebaut werden, so muss eine Nachbarschaftsinformation über $n - 1$ Hops verbreitet werden.

Die reaktive Methode erfragt zuerst die Nachbarn aller Knoten, welche im benötigten Bereich liegen. Diese Anfragen werden über $n - 1$ Hops verschickt (wenn eine n-Hop-Nachbarschaft erforderlich ist). Diese Weiterleitung erfolgt per Broadcast. Mechanismen müssen doppelte Pakete erkennen und unterdrücken. Dieses Vorgehen wird auch als „Fluten“ – *Flooding* bezeichnet. Je größer die Nachbarschaft, desto größer wird auch die Zahl der Weiterleitungen und damit die Dauer, bis die Anfrage beendet ist. Aufgrund von Paketverlust kann es zudem vorkommen, dass Knoten in der Umgebung nicht (oder nicht ohne Wiederholungen) über die Anfrage informiert werden und deshalb nicht antworten. Die Antworten müssen den selben Weg wie die Anfrage zurück zum Knoten nehmen.

Das proaktive Verfahren stellt sicher, dass innerhalb der Dauer von p Perioden alle Stationen mindestens ein Paket von allen Nachbarn empfangen haben. Dabei steht p für die Zahl der maximalen Paketverluste auf dem Medium plus eins. Ohne Paketverluste ist $p = 1$. Ein *Beacon*-Paket trägt die Adresse des Absenders. Auf diese Weise kann ein Knoten ohne eigene Aktivitäten ein vollständiges Bild der eigenen 1-Hop-Nachbarschaft erstellen. Treten Paketverluste auf, kann trotzdem aus den empfangenen Paketen eine Nachbarschaft erzeugt werden. Sollen Nachbarschaften über mehrere Hops aufgebaut werden, ist es notwendig, dass Informationen über mehrere Knoten weitergeleitet werden. Diese Weiterleitung kann auch mit unvollständigen Daten erfolgen, da die Nachbarschaftsinformationen in den nächsten Perioden ausgebaut bzw. vervollständigt werden.

Je mehr Hops die Nachbarschaft beinhaltet, desto länger dauert es, sie erstmals zu erstellen, minimal n Perioden bei einer n-Hop-Nachbarschaft. Bei der angestrebten Mehr-Hop-Nachbarschaft ist deshalb ein ständiges Aktualisieren der Nachbarschaft die bevorzugte Variante, da ein Erstellen einer kompletten Nachbarschaft bei Bedarf deutlich länger dauert und kurzzeitig eine hohe Last durch viele Anfragen erzeugt.

Darüber hinaus birgt die Weiterverbreitung von nicht-selbständig ermittelten Informationen das Problem, dass deren Aktualität nicht sicher ist. Die Information über die Nachbarschaft in n Hops Entfernung ist bereits n Perioden alt. Während dieser Zeit kann sich die Topologie verändert haben. Selbst ohne dieses Problem ist es notwendig zu unterscheiden, ob ein Knoten ausgefallen ist (bzw. sich wegbewegt hat) oder nur ein Paketverlust aufgetreten ist. Es wird ein Algorithmus benötigt, welcher das Alter von Informationen ermittelt und damit auch den Zeitpunkt festlegt, ab wann sie nicht mehr gültig sind.

Das Altern der Informationen wird mit dem Beaconing gekoppelt. Nach dem Senden eines Beacons werden alle Informationen überprüft und Qualitätsindizes aktualisiert. Bei der Unterschreitung eines festgelegten Wertes wird der Knoten aus der Liste entfernt.

Der Index berechnet sich aus der Anzahl der empfangenen Beacons während der letzten m Perioden. Der bestmögliche Wert ist m, d.h. während aller Perioden wurde ein Signal empfangen. 0 heißt, dass kein Signal eingegangen ist. Welcher Wert den Wegfall eines Knotens festlegt, bedarf näherer Betrachtung: Eine Verbindung zu einem Knoten kann über einen „schlechten Link“ durchaus vorhanden sein. Allerdings kann der Verlust von Paketen dafür sprechen, dass eine Verbindung über eine andere Route angebracht sein kann. Da dies vom verwendeten Routingalgorithmus abhängt, muss diesem auch entsprechende Informationen zur Verfügung gestellt werden. Zur Berechnung der Bandbreite sind alle Knoten in der Umgebung relevant, die freie Bandbreite belegen könnten. Um von der Bandbreitenberechnung ausgeschlossen zu werden, muss ein Knoten also den Index 0 erreichen.

Abschließend werden die beschriebenen Algorithmen im Pseudocode dargestellt. Der Algorithmus besteht dabei aus mehreren unabhängig voneinander aktiven Teilen. Ein Teil (Tabelle 3.5) ist für das Versenden des *Beacons* verantwortlich. Eingangsgrößen sind die Periodendauer und die Nachbarschaftsliste.

```
Listenaktualisierung starten;
Nachbarliste einlesen;
Liste in Sendeformat umwandeln;
Paket erstellten;
Beaconpaket per Broadcast versenden;
aktuelle Zeit einlesen, in Variable "jetzt" speichern;
Algorithmus zum Zeitpunkt
     "jetzt+Periodendauer" neu starten;
```

Tabelle 3.5: *Beacon*sendevorgang

Der in Tabelle 3.6 beschriebene Algorithmus wird immer dann gestartet, wenn ein *Beacon* eines anderen Knotens empfangen wird. Dieses *Beacon* enthält neben der Absenderadresse eine Liste der Nachbarn des Absenders.

```
Falls Absender in Nachbarnliste vorhanden:
   Zeitstempel aktualisieren;
sonst:
   Knotenadresse der Nachbarnliste hinzufügen;
Ende_Falls;
Empfangene Daten in Liste "A" speichern;
Solange "A" nicht leer ist:
   Ersten Eintrag aus "A" entnehmen;
   Falls Eintrag in der Nachbarnliste vorhanden ist:
      Falls neue Entfernung kleiner oder
      gleich alter Entfernung:
         Entfernung und Zeitstempel aktualisieren;
      Ende_Falls;
   sonst:
      Knoten der Nachbarnliste hinzufügen;
   Ende_Falls;
wiederhole;
```

Tabelle 3.6: Empfang eines *Beacon*s

Knoten, von denen über längere Zeit keine Signale mehr empfangen wurden, müssen aus der Liste entfernt werden. Zu diesem Zweck wird ein Qualitätsindex genutzt. Dazu wird während des Sendevorgangs eine Funktion zum Aktualisieren der Liste gestartet. Auf diese Weise werden periodisch alle Einträge überprüft und der Qualitätsindex aktualisiert. Es gibt einen einfacheren (Tabelle 3.7) und einen komplexeren (Tabelle 3.8) Algorithmus. Der einfache Algorithmus mit geringerer Komplexität sollte bei geringer Knotenmobilität eingesetzt werden. Verlassen Knoten häufig die Nachbarschaft und kehren zurück, kann der Algorithmus den Qualitätsindex nicht wieder erhöhen, dazu ist aber der zweite, komplexere Algorithmus fähig. Beide Algorithmen haben die Nachbarschaftsliste als Eingabe. Der erste Algorithmus nutzt einen einzelnen Wert als Qualitätsindex, der zweite Algorithmus benötigt zusätzlich eine Warteschlange mit Qualitätsindizes als Einträgen.

```
solange das Listenende nicht erreicht ist:
   falls der Qualitätsindex kleiner gleich null ist:
      Knoten aus Liste entfernen;
   sonst:
      falls der Zeitstempel des Eintrags
      älter als eine Periode ist:
         Qualitätsindex um eins reduzieren;
      ende_falls;
   ende_falls;
   zum nächsten Eintrag in der Liste gehen;
wiederholen;
```

Tabelle 3.7: 1. Aktualisierungsalgorithmus

3.4 Bandbreitenkontrolle - Traffic-Shaping

Der folgende Abschnitt beschäftigt sich mit mit den Möglichkeiten die verhandelte Bandbreite lokal zu überwachen und zu beschränken.

Die ausgehandelte maximale Bandbreite darf nicht durch fehlerhafte Applikationen überschritten werden. Bevor ein Paket gesendet wird, muss geprüft werden, ob die zu benutzende Bandbreite noch zur Verfügung steht.

Wenn innerhalb einer geforderten Zeit eine bestimmte Menge Daten übertragen werden müssen, wird eine bestimmte Bandbreite benötigt. Ob diese Daten in einem einzelnen „Burst“ mit maximaler zur Verfügung stehender Bandbreite oder mit kleinerer Übertragungsrate über eine längere Zeit übertragen werden, ist irrelevant. Wichtig ist nur, dass die Daten spätestens zum angegebenen Zeitpunkt versandt sind.

Das „Formen“ der Bandbreite wird durch *Traffic-Shaping* realisiert indem die Sendezeitpunkte in Abhängigkeit der Größe der vorherigen Pakete gewählt werden. Durch dieses Verfahren kann garantiert werden, dass zwischen zwei beliebigen Zeitpunkten des Sendens die vorgegebene Bandbreite nie überschritten wird.

Zwei Protokolle, welche Methoden zum *Traffic Shaping* beschreiben, sind das „Leaky Bucket“ und das „Token Bucket“-Protokoll, welche in den folgenden Abschnitten erläutert werden, bevor das im Rahmen der Diplomarbeit verwendete Verfahren erläutert wird.

3.4.1 Leaky Bucket Protokoll

Das namensgebende Modell hinter dem *Leaky Bucket Protokoll* ist ein leckender Eimer. Aus einem Loch an der Unterseite läuft ein gleichmäßiger Strom Wasser aus,

unabhängig davon, welche Menge Wasser oben eingefüllt wird. Auf diese Weise wird der Ausgangsstrom von Spitzen befreit und „geebnet“ ([23]).

Wie auch das *Token Bucket*-Protokoll ist der Leaky Bucket die Schnittstelle eines Knotens zum Netzwerk. Eine endliche Zahl ausgehender Pakete wartet in einer Warteschlange. Sie werden in Abhängigkeit von verschiedenen Parametern versandt. Ist diese Warteschlange, diese Queue, voll, werden keine weiteren Pakete akzeptiert und neue verworfen – ebenfalls eine Analogie zu dem Eimer, welcher überlaufen würde, wenn er voll ist und weiter Wasser hineingefüllt würde.

Systeme mit fester Paketgröße, wie etwa ATM-basierte Netzwerke, versenden pro Zeiteinheit ein Paket. Ist die Paketgröße variabel, wird zusätzlich ein Zähler („byte count“) genutzt. Dieser wird beim Beginn einer Periode auf einen festgelegten Wert initialisiert. Ist die Größe eines Pakets kleiner oder gleich diesem Wert, kann es versendet werden. Anschließend wird die Größe des Paketes von dem Zähler subtrahiert. Ist der Zähler größer als Null, kann ein neues Paket, welches das Kriterium „Paketgröße kleiner als Zähler“ erfüllt, versandt werden.

Ein Beispiel: In der Warteschlange befinden sich fünf Pakete (A bis E) verschiedener Größen (A: 20; B: 100; C: 40; D: 50; E: 30). Der Zähler wird periodisch jede Sekunde auf 100 gesetzt.

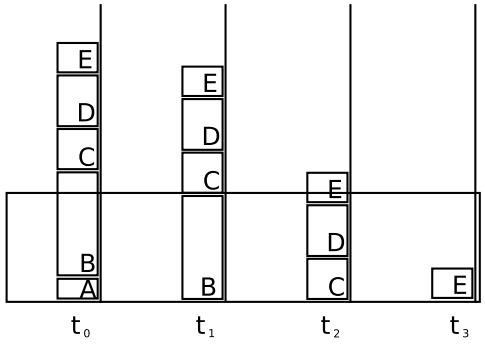


Abbildung 3.2: Das *Leaky Bucket*-Protokoll beschränkt das Versenden auf eine festgelegte Anzahl Bytes pro Periode

Abbildung 3.2 zeigt die schematische Darstellung der Warteschlange. Am Kopf (links) warten die Pakete, welche zuerst versandt werden sollen. Zum Zeitpunkt t_0 wird der Zähler auf 100 gesetzt. A hat eine Größe von 20 und wird versandt. Die Restgröße des Zählers ist 80 $(100 - 20)$. B hat eine Größe von 100. 100 ist größer als 80, also wird B nicht versandt. Die nächste Periode beginnt bei t_1. Der Zähler wird wieder auf 100 gesetzt. B kann versandt werden, der Zähler steht bei 0 ($S_{Zähler} - S_B = 100 - 100 = 0$) und es können keine weiteren Pakete versandt werden. Zum Zeitpunkt t_2 wird der Zähler wieder auf 100 gesetzt und C wird verschickt. Der

neue Zählerstand ist $100 - 40 = 60$. Die Größe von D ist kleiner, als der aktuelle Zählerstand ($S_D < S_{Zähler}$), D wird also ebenfalls noch versandt und der Zähler neu berechnet: $S_{Zähler} - S_D = 10$. Die Größe von E ist höher als der Zählerstand. E kann also erst in der nächsten Periode, ab t_3 versendet werden.

3.4.2 Token Bucket Protokoll

Das *Token Bucket*-Protokoll ist dem *Leaky Bucket*-Protokoll ähnlich. Zur Darstellung kann ebenfalls der „undichte Eimer“ benutzt werden. Der Unterschied ist, dass das *Token Bucket*-Protokoll nicht eine vollkommen gleichmässige Ausgabe von Paketen erreichen soll, sondern auch begrenzt *Bursts* zulässt.

Lässt das *Leaky Bucket*-Protokoll innerhalb eines bestimmten Zeitabschnittes eine bestimmte Anzahl Bytes zu, welche nach dem Ende dieses Zeitabschnittes verfallen, „spart“ das *Token Bucket*-Protokoll die nicht benutzte Bandbreite auf. Die Pakete werden ebenfalls in eine Queue eingefügt. Pro Tick wird ein Token erzeugt (und bei Nicht-Benutzung gespeichert). Soll ein Paket gesendet werden, muss es ein Token aquirieren (*capture*) und zerstören (*destroy*). Wiederum gilt dieses Prinzip für Netze mit fester Paketgröße. Bei variabler Paketgröße wird ein Counter benutzt, welcher die Menge der möglichen zu versendenden Bytes repräsentiert.

Parameter dieses Algorithmus sind die Übertragungsrate ρ, die Kapazität der Warteschlange C und die maximale Größe des Bucket B. Im Unterschied zu dem *Leaky Bucket* Algorithmus, kann ρ kurzzeitig überschritten werden, wenn vorher nur Daten mit niedrigerer Rate übertragen wurden und sich die „ungenutzten Kapazitäten“ im Bucket B gesammelt haben.

Der Algorithmus fügt pro Tick dem Token Bucket einen Token hinzu, wenn die Größe B nicht überschritten wird. Pakete werden wiederum in eine Warteschlange der Kapazität C eingefügt. Sind die Pakete von konstanter Größe, kann das erste Paket in der Warteschlange gesendet werden, wenn ein Token vorhanden ist. Das benutzte Token muss von dem Paket belegt („capture“) und zerstört („destroy“) werden. So lange Token vorhanden sind, können Pakete gesendet werden.

In Systemen mit variabler Paketgröße ist der Token Bucket ein Zähler. Dieser Zähler c enthält die Menge an Bytes, welche versandt werden können und wird bei jedem Tick um die Anzahl an Bytes erhöht, welche sich aus der Dauer eines Ticks und der Übertragungsrate ρ ergibt.

Das erste Paket der Warteschlange kann versendet werden, wenn dessen Größe s kleiner als der Zähler c ist. Die Größe des Pakets s wird dann von c abgezogen. So lange $s < c$ gilt, kann das jeweils erste Paket der Warteschlange versandt werden.

3.4.3 Eingesetztes Verfahren

Die beabsichtigte Bandbreitenkontrolle erfordert eine strenge Einhaltung der Grenzen. Das *Token Bucket*-Protokoll ermöglich im Gegensatz dazu gezielt das Überschreiten der maximalen Übertragungsrate ρ, wenn in der Vergangenheit weniger gesendet wurde. Im ungünstigsten Fall könnten sämtliche Knoten in der Nachbarschaft versuchen, ihre aufgespeicherten Bursts gleichzeitig zu senden. Damit wäre aber die maximal verfügbare Bandbreite in der Umgebung überschritten und gegebene Garantien könnten nicht eingehalten werden.

Zur Durchsetzung der Bandbreitengarantien wird deshalb eine Variante des *Leaky Bucket*-Protokolls benutzt, da dort die vorgegebene maximale Bandbreite nicht überschritten werden kann.

Das „klassische“ *Leaky Bucket*-Protokoll wird in abgewandelter Form verwendet. Der ursprüngliche Ablauf sieht einen Timer vor, welcher in konstanten Abständen einen Bytezähler schreibt, welcher aufgebraucht werden kann. Durch diese Vorgehen „verfallen“ am Ende u.U. immer einige Bytes, weil kein Paket klein genug war, um versandt zu werden.

Statt einen konstanten Timer zu benutzen, wird ein variabler eingeführt. Es sollen jetzt nicht mehr die Zeitpunkte markiert werden, zu denen die Verfügbarkeitscounter erhöht werden, sondern die Zeitpunkte zu denen der Sendevorgang des vorherigen Pakets mit der angegebenen Bandbreite abgeschlossen wäre.

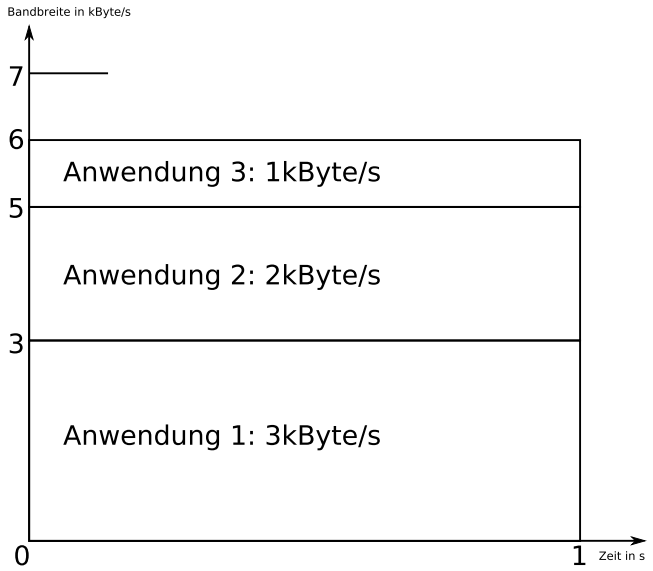


Abbildung 3.3: Drei Anwendungen senden mit unterschiedlichen Bandbreiten – schematische Darstellung

Abbildung 3.3 zeigt ein Beispiel. Drei Anwendungen übertragen mit unterschiedlichen Bandbreiten. Nach einer Sekunde soll die Übertragung abgeschlossen sein. Anwendung 1 will mit $3\frac{kByte}{s}$ übertragen. Nach einer Sekunde hätte sie also drei Kilobyte versandt. Die anderen Anwendungen wollen $2kByte$ (Anwendung 2) und $1kByte$ (Anwendung 3) übertragen. Bis zur Grenze der maximal verfügbaren Bandbreite von $7\frac{kByte}{s}$ ist noch $1kByte$ frei. Diese Darstellung entspricht der Vorstellung, dass alle Anwendungen eine bestimmte Bandbreite reserviert haben, die sie gleichzeitig nutzen können.

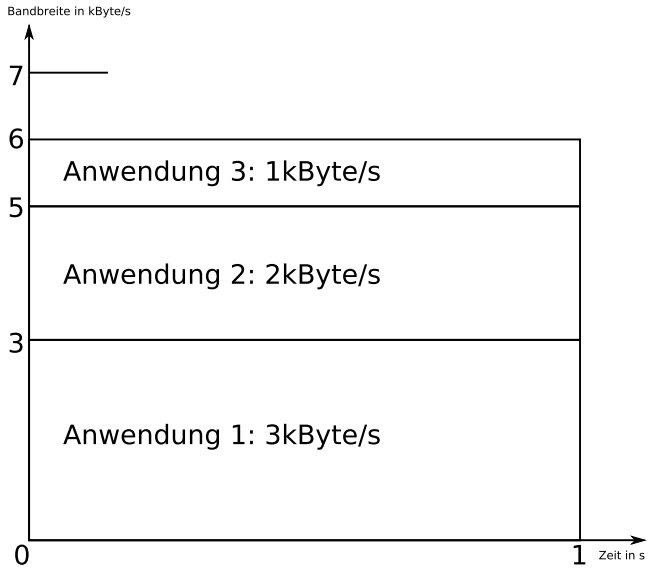


Abbildung 3.4: Drei Anwendungen senden mit unterschiedlichen Bandbreiten – realistischere Darstellung

Es ist offensichtlich, dass Daten real nicht gleichzeitig, jede mit ihrer eigenen Bandbreite in einem geteilten Medium übertragen werden können, wie es in Abbildung 3.3 dargestellt wird. Der Realität entspricht vielmehr Abbildung 3.4. Es ist zu erkennen, dass die Anwendungsdaten in kürzerer Zeit versandt werden. Die Anwendungen können aber genau die zugesicherte Menge Daten übetragen. Auch die Änderung der Reihenfolge der Pakete würde die Anforderung, dass die Daten in einer Sekunde versandt sein müssen, nicht verletzen.

Dieses Beispiel verdeutlicht die Schlussfolgerung, dass es unerheblich ist, wie viele Pakete in welcher Reihenfolge versandt werden. Solange die Übertragungen der Anwendungen in der Summe nicht die maximale Bandbreite überschreiten, findet die Übertragung statt, als würde ihnen ein exklusiver Kanal mit der benötigten Bandbreite zur Verfügung stehen.

3.5 Atomare Entscheidung

Der folgende Abschnitt beschreibt das Vorgehen zur Reservierung von Bandbreite in dem drahtlosen Multi-Hop-Netzwerk.

Zuerst wird die Notwendigkeit der Reservierung von Bandbreite gezeigt. Soll ein Knoten eine derartige Bandbreitenreservierung durchführen, so muss diese Operation auf allen Knoten der Nachbarschaft durchgeführt werden, ohne dass sie unterbrochen wird bzw. mit einer anderen Operation konkurriert. Abschnitt 3.5.1 erläutert die Gründe genauer am Beispiel.

Eine solche nicht unterbrechbare, unteilbare Operation wird auch als *atomare Operation* bezeichnet. Die folgenden Abschnitte 3.5.2 und 3.5.3 erläutern, wie mithilfe einer solchen Operation eine verteilte Bandbreitenreservierung erreicht wird.

3.5.1 Allgemein

Die Ausgangssituation ist Folgende: Ein Knoten A benötigt Bandbreite für einen Datenstrom. Dieser Strom belegt nicht nur Bandbreite auf Knoten A oder dem Zielknoten B, sondern in der gesamten *Bandbreitennachbarschaft* des Knotens A. Die *Bandbreitennachbarschaft* ist die Menge $N(A)$ der Knoten in der Umgebung von A.

Die Bandbreite, welche diesen Knoten zur Verfügung steht, wird durch die von A ausgehende Übertragung also verringert (siehe 2.1) Betroffen ist die *n-hop*-Nachbarschaft des Knotens $N(A)$, wobei die Größe von n von den physikalischen Eigenschaften des Mediums abhängt. Messungen (siehe 5.1) haben für n einen Wert von 2 Hops ergeben. Knoten bis zur doppelten Entfernung der Sende-/Empfangsreichweite werden so von Sendevorgängen beeinflusst.

Folgendes Beispiel soll verdeutlichen, warum dies von Bedeutung ist, wenn ein Knoten eine Übertragung mit bestimmten Garantien durchführen will: Ein Knoten A will einen Strom $\mathcal{S}$ etablieren. Eine Eigenschaft dieses Stromes ist die feststehende Bandbreite. Bei paketbasierten Protokollen (wie UDP) kann man diesen Wert aus der Größe eines einzelnen Paketes (p_{size}) und der Sendeperiode (t_{period}) errechnen: $\rho = \frac{p_{size}}{t_{period}}$ (siehe auch 2.1.3). Alle Knoten in der Nachbarschaft $N(A)$ müssen genügend freie Bandbreite zur Verfügung haben, damit A die vorgesehenen Übertragungen in der geforderten Zeit durchführen kann.

Soll dies über einen längeren Zeitraum garantiert werden, darf kein anderer Knoten in der Folgezeit so viel Bandbreite belegen, dass die Summe der verwendeten Bandbreite größer ist, als die maximal verfügbare Bandbreite. Gleichung 3.4 muss gelten, wobei ρ_i die vom i-ten Knoten genutzte Bandbreite ist.

$$\rho_{max} \geq \sum_{i} \rho_i; i \in \mathcal{N} \tag{3.4}$$

So müssen alle Knoten der Nachbarschaft der Benutzung der Bandbreite zustimmen und diese Zusicherungen einhalten. Kann ein Knoten die Bandbreite nicht reservieren, darf A die Übertragung nicht starten, da es zu Übersättigung des Mediums führt und damit bereits getätigte Zusicherungen an Applikationen nicht gewährleistet werden können.

In der vorliegenden Umgebung existiert kein zentraler Koordinator, welcher über die Bandbreitenverteilung bestimmt, sondern die Knoten müssen sich in einem verteilten Algorithmus einigen.

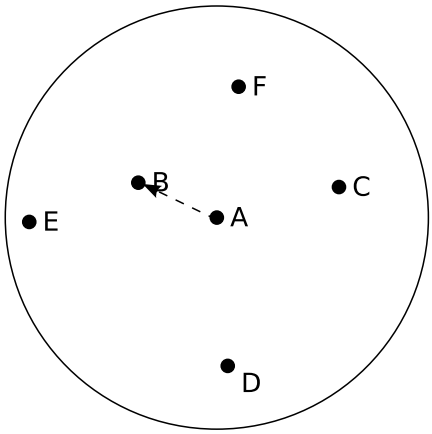


Abbildung 3.5: Knoten A, seine Nachbarn und die anstehende Transmission (der Kreis skizziert den Bereich, in welchem die Transmission Bandbreite belegt)

Im Beispiel will Knoten A einen Strom etablieren, welcher bestimmte Bandbreitengarantien benötigt. A muss deshalb alle Knoten in seinem Einflussbereich (siehe Abbildung 3.5 und 2.1.4) „befragen“, ob sie genug Bandbreite vorrätig haben. Diese Bandbreite muss er dann verbindlich belegen und dies auch allen beteiligten Knoten mitteilen.

Wie erwähnt, gehören alle Knoten im „Einflussbereich“ von A zu seiner *Bandbreitennachbarschaft*. Die Knoten der Menge $N(A) = \{B, C, D, E, F\}$ sind die Nachbarn von A. Abhängig von einem eindeutigen Bezeichner für jeden Knoten, wie etwa die IP-Adresse, können die Knoten nummeriert und sortiert werden, so dass $N_i(A)$ der $i - te$ Nachbarknoten von A ist.

Die erste Annäherung an dieses Problem ist folgender einfacher Algorithmus. A befragt in beliebiger Reihenfolge alle $N(A)$ nach ihrer verfügbaren Bandbreite. Anhand dieser Information entscheidet A. Ist die kleinste verfügbare Bandbreite größer als die benötigte Bandbreite, kann die Übertragung beginnen und der Strom etabliert

werden. Damit die anderen Knoten die jetzt genutzte Bandbreite aber in Zukunft nicht belegen, muss A vorher noch alle Nachbarn informieren, u.a. darüber wieviel Bandbreite von ihm genutzt wird.

Ein Problem bei diesem einfachen Algorithmus tritt sofort auf, wenn man mehrere parallele Reservierungsvorgänge betrachtet. Knoten E liegt nicht nur in der Nachbarschaft von A, sondern auch in der Nachbarschaft von B. Es gibt also eine Menge Knoten, die sowohl Nachbarn von A, als auch Nachbarn von B sind: $N(A, B)$

Ein Beispiel:

A befragt die Knoten in der Reihenfolge B,C,D,E,F. Angenommen, B teilt sich seine Bandbreite nur mit A, E und F und befragt diese auch in der Reihenfolge. (siehe Abbildung 3.6: gestrichelt die (Bandbreiten)Nachbarschaft von B und mit durchgezogener Linie skizziert die (Bandbreiten)Nachbarschaft von A)

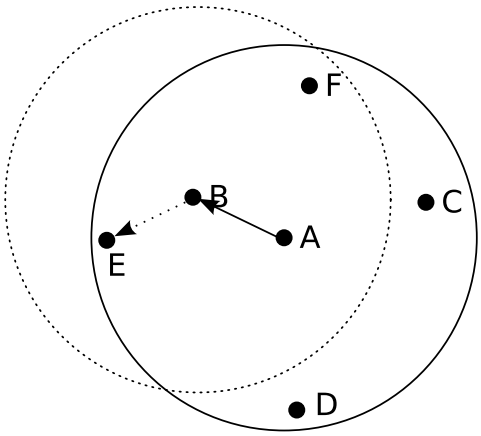


Abbildung 3.6: Die Zonen gemeinsamer Bandbreitennutzung sind für jeden Knoten anders.

Sowohl A als auch B befragen also die Knoten E und F. Die freie Bandbreite in der Umgebung der Knoten E und F soll mit gegenseitig unbeeinflussten $4\frac{kb}{s}$ angenommen werden. A benötigt $2\frac{kb}{s}$ und B benötigt $3\frac{kb}{s}$. Beide Knoten können nicht gleichzeitig senden, folgende Fälle können eintreten:

1. B kontaktiert E zuerst. B erhält die Information, dass genügend Bandbreite vorhanden ist und reserviert diese. In der Umgebung sind jetzt noch $1\frac{kb}{s}$ „übrig“. Anschließend nimmt Knoten A Verbindung zu E auf. Die von A benötigte Bandbreite ($2\frac{kb}{s}$) ist größer als die bei E vorhandene ($1\frac{kb}{s}$). A kann also nicht reservieren und bricht den Reservierungsvorgang ab.

   Gleichzeitig ist Knoten B bei F angelangt. Nach der Reservierung von A stellt F noch $2\frac{kb}{s}$ zur Verfügung – nicht genug für B. Der anstehende Strom wird

also abgelehnt und die schon reservierte Bandbreite auf allen Knoten wird freigegeben.

Würde A erst jetzt die Reservierung beginnen, verliefe die Reservierung anders: Sowohl auf dem Knoten E, als auch auf dem Knoten F stehen jeweils mindestens $2\frac{kb}{s}$ zur Verfügung. A hatte allerdings E schon vorher befragt und eine negative Auskunft erhalten, obwohl eine Reservierung möglich gewesen wäre.

Auf diese Weise kommt es zu einer Fehlentscheidung bei der Bandbreitenreservierung, die allerdings keine bereits bestehenden Bandbreitenzusagen gefährdet.

2. Erfragen A und B bei E die Bandbreite, bevor einer der beiden reservieren konnte, erhalten beide eine ausreichende Bandbreite angegeben ($4\frac{kb}{s}$). Zusammen benötigen A und B allerdings: $5\frac{kb}{s}$ ($2\frac{kb}{s} + 3\frac{kb}{s}$), also $1\frac{kb}{s}$ mehr, als vorhanden.

   Im günstigsten Falle wird dies in einem zusätzlichen Reservierungsschritt erkannt und verweigert (z.B. wird nur die erste Reservierungsanfrage zugelassen). Andernfalls kommt es zu einer *false admission* – zu einer Übersättigung des Mediums. Mit der Konsequenz, das die Knoten A und B mehr Bandbreite nutzen wollen, als tatsächlich vorhanden ist. Zugesagte Bandbreiten (auch von anderen Knoten) können daraufhin nicht mehr garantiert werden.

Es ist ersichtlich, dass die vorgeschlagene Vorgehensweise keine praktikable Lösung darstellt. Eine korrekte Entscheidung kann nur getroffen werden, wenn den reservierenden Knoten Informationen zur Verfügung stehen, die sich während der Operation nicht verändern. Dies bedeutet, dass keiner der beteiligten Knoten während der Entscheidungsfindung seinen Zustand ändern sollte. Während A also die Bandbreite seiner Nachbarn ($N(A)$) erfragt, um eine Entscheidung treffen zu können, darf kein Knoten der Menge $N(A)$ einem anderen Knoten Angaben über die eigene freie Bandbreite machen bzw. der andere anfragende Knoten muss warten.

Die einzige Möglichkeit, um diese Forderung zu erfüllen, ist die Durchführung der Bandbreitenreservierung als eine *nicht-unterbrechbare* Operation – eine *atomare* Operation.

3.5.2 Atomare Operation

Prinzip

Im vorhergehenden Abschnitt wurde die Notwendigkeit gezeigt, einen atomaren, d.h. nicht unterbrechbaren Prozess für die Reservierung von Bandbreite zu nutzen. Dabei müssen alle Knoten der *Bandbreitennachbarschaft*, $N(A)$), für andere Anfragen gesperrt (*lock*) werden, die Anfragen durchgeführt werden und nach der Entscheidung die Nachbarn wieder freigegeben (*unlock*) werden.

Zugriffe auf exklusiv nutzbare Ressourcen bergen immer die Gefahr von Verklemmungen (*Deadlocks*) durch wechselseitiges Warten/wechselseitigen Ausschluss (*mutual waiting* bzw. *mutual exclusion*). Abbildung 3.7 skizziert diese Situation.

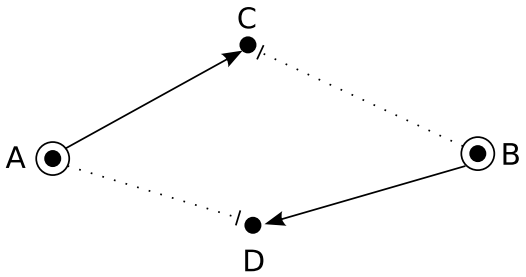


Abbildung 3.7: A und B sind in der *lock*-Phase und wollen beide C und D sperren.

A und B befinden sich in der *Lock*-Phase. A hat C ge*lock*t, B hat ein *lock* auf D. Will A nun D *lock*en, gelingt das nicht und A wartet. Andersherum wartet B, dass der *lock* von A auf C aufgehoben wird. Abbildung 3.8 zeigt die versandten Nachrichten.

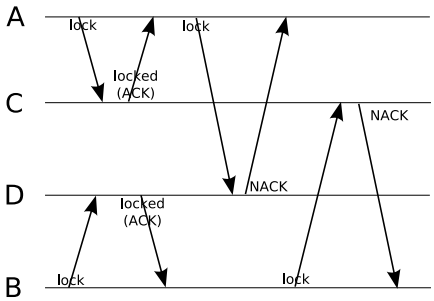


Abbildung 3.8: A und B sind in der *lock*-Phase und wollen beide C und D sperren.

Weder A noch B können ihre Operation beenden oder fortführen. Eine Lösung des *Locks* ist nur durch einen Verzicht eines der Knoten bzw. durch einen Timeout oder eine externe Instanz möglich.

Da die meisten Knoten schon durch die Definition der *Bandbreitennachbarschaft* zu mehreren Nachbarschaften gehören, ist ein *Deadlock* sehr wahrscheinlich und muss vermieden werden. Zu diesem Zweck werden normalerweise *Semaphoren* eingesetzt.

Alle Knoten, welche Bandbreite auf einer Menge gemeinsamer Knoten reservieren wollen, müssen die Reservierung dieser kritischen Menge bei genau einem Knoten beginnen. Dieser Knoten kann dann den wechselseitigen Ausschluß koordinieren. Um

diesen Knoten zu bestimmen, wird eine Reihenfolge festgelegt, in welcher Knoten ge*lock*t werden müssen. Diese Reihenfolge wird durch in dem gesamten Netzwerk eindeutige Eigenschaften, z.B. die IP-Adresse, festgelegt. Das Einhalten dieser Reihenfolge vermeidet Deadlocks, indem der gemeinsame Knoten mit der niedrigsten gemeinsamen Nummer die Funktion der Semaphore für den gesamten Bereich der gemeinsamen Knoten übernimmt und nur einem sperrenden Knoten „Zugang“ zu diesem Bereich gewährt.

Abbildung 3.9 zeigt ein Beispiel dazu. Wollen die Knoten A und B Knoten in ihrer (Bandbreiten)Nachbarschaft kontaktieren, müssen sie die Reihenfolge in Abhängigkeit der Nummern der Knoten einhalten.

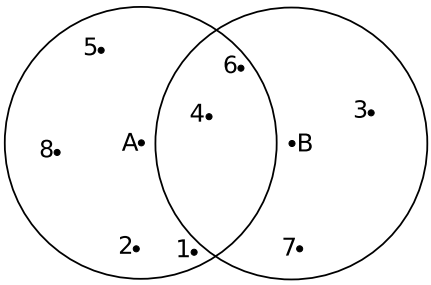


Abbildung 3.9: A und B wollen Reservierungen auf Knoten in ihrem Bandbreitenbereich durchführen ohne das *Deadlocks auftreten*

Beide, A und B, müssen die Knoten 4 und 6 *lock*en. Greift A zuerst auf 4 zu und B zuerst auf 6, kann es zu oben erläuterter *Deadlock*-Situation kommen. Durch die Sortierung nach der Knotennummer, laufen die Zugriffe allerdings so ab, wie in Abbildung 3.10 dargestellt.

A und B müssen in dieser Darstellung die Knotenzugriffe von oben nach unten abarbeiten (dies entspricht dem Zugriff nach einer sortierten Knotenliste). Die Pfeile in der Abbildung symbolisieren notwendige Zugriffe. Die drei Nachrichten, welche mit durchgezogenen Pfeilen dargestellt sind, können ohne gegenseitige Beeinflussung gesendet werden. Danach wollen beide Knoten den Nachbarn mit der Nummer 4 *lock*en (gestrichelte Linie). Nur ein Knoten (der schnellere) schafft dies und kann sein *Locking* fortsetzen. Der *lock* auf 4 übernimmt die Funktion einer Semaphore für den Bereich der gemeinsamen Knoten ($N(A, B)$).

Gewinnt A, fährt er mit Zugriffen auf 5, 6 und 8 fort (gepunktete Pfeile). B darf nicht fortfahren und etwa 6 reservieren, sondern muss auf die Freigabe von 4 warten. Hat A seine Operation beendet und alle Knoten wieder freigegeben, kann B den Knoten 4 reservieren und auch mit den anderen (6 und 7; Strich-Punkt-Pfeile) fortfahren. Die Möglichkeiten für einen *Deadlock* bestehen also nicht mehr, da die Zugriff zwingend zuerst auf einen einzelnen Knoten, statt beliebig in eine Menge

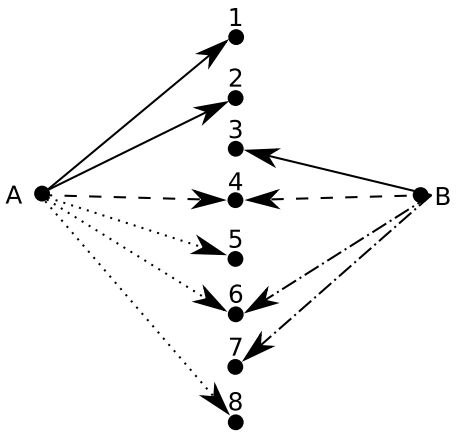


Abbildung 3.10: schematische Darstellung der Zugriffsreihenfolge zur Vermeidung von Deadlocks

von gemeinsamen Knoten, durchgeführt werden. Dies ist der Knoten, welcher sowohl in der Nachbarschaft von A, als auch in der Nachbarschaft von B liegt ($N(A, B)$) und die kleinste Nummer hat (auch bezeichnet als erster Knoten der gemeinsamen Nachbarschaft: $N_0(A, B)$).

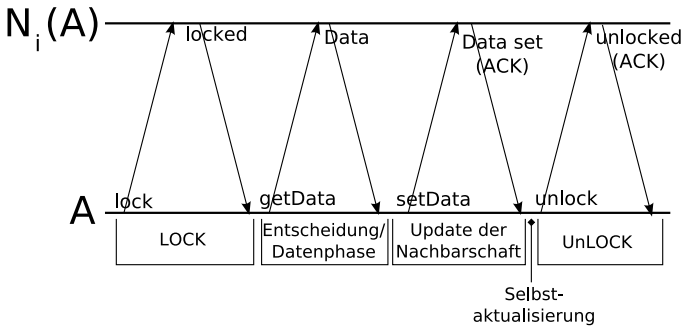


Abbildung 3.11: Die Aushandlung der verwendeten Bandbreite findet prinzipiell in vier Phasen statt

Die notwendigen Schritte und Phasen des vollständigen Protokolls zeigt im einzelnen Abbildung 3.11. Hierbei wird die Kommunikation von A mit einem einzelnen Nachbarn dargestellt ($N_i(A)$). Die erste und letzte Phase („lock“ und „unlock“) dienen der Einrichtung des exklusiven Zugriffs auf alle Nachbarknoten während der Operation. Erhält ein Knoten eine *lock*-Anfrage, beantwortet er diese Anfrage nur dann positiv, wenn er nicht bereits von einem anderen Knoten ge*lock*t ist. Kann

der Knoten ge*lock*t werden, so antwortet er mit einer entsprechenden Bestätigungsnachricht: *locked* oder *confirm lock*. Ab diesem Zeitpunkt können Anfragen aus den anderen Phasen empfangen und verarbeitet werden.

In der zweiten Phase fordert A von jedem Knoten der Nachbarschaft die aktuelle lokale Bandbreite ein (*getData*). Die Antwort folgt mit *Data*. Nun hat A alle Informationen, um zu entscheiden, ob in seiner Umgebung genügend Bandbreite für die geplante Übertragung zur Verfügung steht. Diese Entscheidung teilt er in der folgenden dritten Phase der Nachbarschaft mit („Update der Nachbarschaft“). Die Nachricht *setData* enthält Informationen darüber, wie viel Bandbreite reserviert werden soll (und ob überhaupt).

An dieser Stelle folgt die Selbstaktualisierung, welche prinzipiell auch vor der vorhergehenden dritten Phase stattfinden könnte. Der Knoten A, welcher die Operation initialisiert hat, weist den lokalen Mechanismen (Traffic Shaper etc.) die neuen Informationen zu, d.h. es wird die Instruktion an die anderen Teile der Admission-Control gegeben, lokal alles für die neue Transmission vorzubereiten.

In der vierten Phase wird, wie oben bereits angedeutet, der *Lock* wieder aufgehoben. Dazu sendet A eine *unlock*-Nachricht an den jeweiligen Nachbarknoten, welcher den Erhalt dieser Nachricht bestätigen muss und ein *unlocked* sendet – eine Acknowledgementnachricht auf die *unlock*-Nachricht.

Nach dem Erhalt der *unlock*-Nachricht darf jeder der Nachbarn wieder neue *lock*-Nachrichten akzeptieren, aber nicht mehr auf Nachrichten aus den Phasen zwei und drei reagieren.

Betrachtet man ein Schema mit zwei Nachbarknoten (wie in Abbildung 3.12), ist ersichtlich, dass sich die in den einzelnen Phasen dargestellte Kommunikation mit jedem Nachbarn wiederholt, bevor zur nächsten Phase übergegangen wird.

Zuerst werden alle Knoten in der *Bandbreitennachbarschaft* gesperrt (ge*lock*t). Damit ist sichergestellt, dass sie während der folgenden Operationen ihren Zustand nicht ohne Wissen des initiierenden Knotens (A) ändern. Anschließend folgt die Abfrage der lokal verfügbaren Bandbreite, wie vorher bereits erläutert. Die Entscheidung anhand der erhaltenen Bandbreiteninformationen wird verbreitet, lokal umgesetzt. Zum Schluß werden alle Knoten der Reihe nach entsperrt.

Ein besonderes Augenmerk ist hierbei darauf zu richten, dass eine festgelegte Reihenfolge für das Sperren (*lock*en) existiert. Durch diese Nummerierung anhand im ganzen Netz feststehender Eigenschaften, wird wie erwähnt erreicht, dass es nicht zu *Deadlocks* durch wechselseitiges Warten (*mutual waiting*) kommt.

Optimierungen

Nachdem der prinzipielle Ablauf des Protokolls erläutert wurde, sollen Optimierungen des Designs diskutiert werden.

Die Wahrscheinlichkeit einer Kollision steigt mit der Größe und der Häufigkeit des

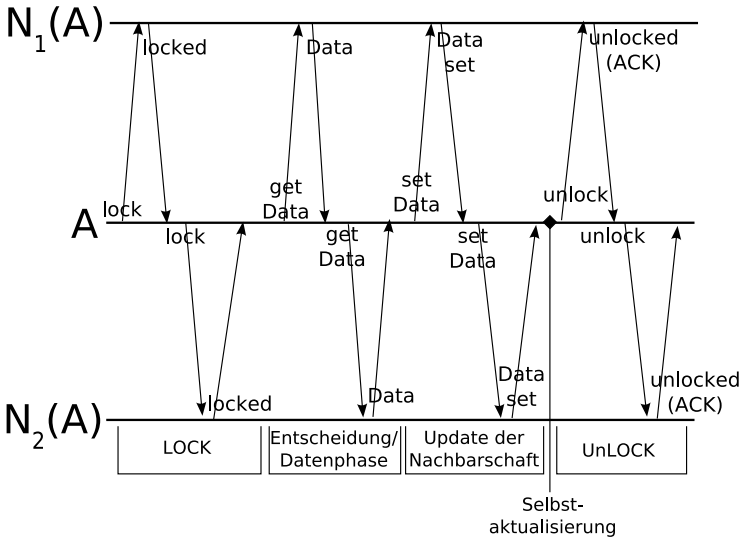


Abbildung 3.12: Die prinzipielle Aushandlung der verwendeten Bandbreite mit zwei Nachbarknoten

Versands von Paketen und dem Grad der Ausnutzung der vorhandenen Bandbreite. Darüber hinaus reduziert jedes Paket die freie Bandbreite. Die Anzahl und Größe der übertragenen Pakete sollte also minimiert werden. Nachrichten müssen möglichst mehr Informationen transportieren (*piggybacking*) oder erweiterte Bedeutung haben.

Die *Lock*-Phase und die Entscheidungs-/Datenphase können kombiniert werden. Eine *Lock*-Anfrage bedingt eine folgende *getData*-Anfrage. Die Semantik des *Lock* kann also so weit geändert werden, dass es ein *getData* impliziert. Das einfache *ACK*, welches ein Nachbarknoten auf ein *Lock* sendet, wird zu einem *piggybacked ACK* in einem Paket, welches die *Data*-Informationen enthält. Die zwei ersten Phasen werden auf diese Weise zu einer.

Die Ausführung der Selbstaktualisierung hängt mit der Datenphase zusammen. Wenn alle Nachbarn genügend Bandbreite zur Verfügung haben, kann auch lokal das Versenden vorbereitet werden. Falls die Selbstaktualisierung schon vorher stattfindet, müssten die Vorbereitungen rückgängig gemacht werden, wenn nicht genügend Bandbreite zur Verfügung steht.

Die *setData*-Phase wird zusätzlich mit der Bedeutung des *Unlock* versehen. Das *setData*-Paket enthält also ein implizites *Unlock* und die *Unlocked*-Nachricht ist sowohl eine Bestätigung für die erhaltenen Daten, als auch eine Bestätigung für die Freischaltung des Knotens.

Die neue Abfolge der Nachrichten wird in Abbildung 3.13 skizziert.

Knoten A im Beispiel Abbildung 3.5 auf Seite 38 will auf $N_0(A)$ (Knoten B), $N_1(A)$

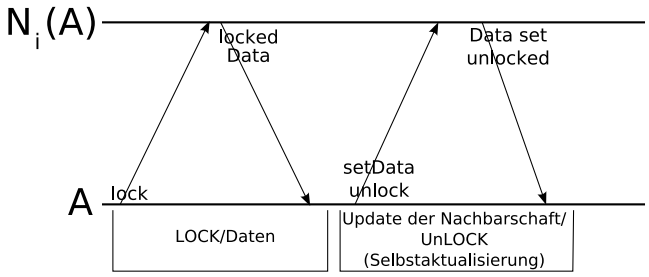


Abbildung 3.13: Aushandlung der Bandbreitenverwendung in zwei Schritten

(Knoten C), usw. bis einschließlich $N_4(A)$ (Knoten F) Bandbreite reservieren.

A beginnt mit der *LOCK/Daten*-Phase. Er sendet als erstes ein *LOCK*-Paket an B. Dieser antwortet positiv durch Übermittlung der lokal freien Bandbreite. Auf die selbe Weise verfährt A bei den Knoten C bis F. Sind alle Nachbarn ge*lock*t, kann A aus den empfangenen Daten die zur Verfügung stehende Bandbreite berechnen. Ist die verfügbare Bandbreite ausreichend für die geplante Übertragung, kann diese zugelassen werden.

Nach der Entscheidung über das Zulassen des Datenstromes beginnt die nächste Phase, die *Update/UnLOCK*-Phase. In dieser Phase informiert A alle Nachbarn mit einer *setData*-Nachricht über die Entscheidung. Diese Nachricht ist gleichzeitig eine *UnLOCK*-Aufforderung, welche den Knoten wieder „frei“ schaltet. Die Reihenfolge beginnt wieder bei dem ersten Knoten in der Nachbarschaft, B, und schreitet erst zum nächsten Knoten fort, wenn eine Bestätigung empfangen wurde (*Data set/ACK*). Hat der letzte Knoten (F) die Daten bestätigt, ist der Algorithmus beendet – alle haben eine gemeinsame Sicht der Situation. A beendet die Reservierungsoperation und geht in den Normalbetrieb über.

3.5.3 Fehler- und Ausnahmebetrachtungen

Die im vorhergehenden Abschnitten dargestellten Abläufe beziehen sich auf den fehlerfreien Betrieb ohne Nachrichtenverlust und ausfallende Knoten. Der Fall eines ge*lock*ten Knotens, welcher von einem anderen ge*lock*t werden soll, wurde nur am Rande erwähnt. Mit diesen Fällen beschäftigt sich der folgende Abschnitt.

Ausfälle

Bei der Kommunikation zwischen den Stationen gibt es vier verschiedene Nachrichten. Das Ausbleiben einer solchen Nachricht kann drei Gründe haben:

1. Es trat eine Kollision oder ein anders bedingter Verlust der Nachricht auf dem

Medium auf.

2. Der Nachbarknoten, welcher antworten sollte, ist ausgefallen.

3. Der initiierende Knoten, welcher die nächste Nachricht senden muss, ist ausgefallen.

Je nach Zustand müssen die Knoten anders reagieren. Wie sie reagieren sollten, wird im Folgenden diskutiert.

Knoten, welche sich aus dem Carrier-Sense-Bereich bewegt haben, werden behandelt, als seien sie ausgefallen. Sollen Ausfälle von Knoten erkannt werden, wird auf *Timeouts* zurückgegriffen. Bei dem vorliegenden Protokoll sind sehr unregelmäßige Wartezeiten wahrscheinlich. Hervorgerufen wird dies u.a. durch das vor den Knoten verborgene Versenden von Nachrichten über mehrere Hops in die n-hop-Nachbarschaft. Für das Zugriffskontrollprotokoll ist es also sehr schwierig, eine Zeit festzulegen, nach der ein Knoten als ausgefallen eingeschätzt werden kann. Zur sicheren Erkennung eines Ausfalls wird deshalb der Beaconing-Mechanismus herangezogen. Dieser enthält Methoden und Funktionalitäten, um kurzfristige Empfangsstörungen von Komplettausfällen zu unterscheiden.

Ausfall der Lock-Nachricht: Bei einem einfachen Verlust der Nachricht erkennt der initiierende Knoten A den Ausfall anhand eines *Timeout*. Daraufhin wird eine *Retransmission*, ein erneutes Senden der Nachricht ausgelöst. (siehe Abbildung 3.14) Zur Vermeidung von Duplikaten und zur Erkennung der Retransmissions werden Sequenznummern in den Paketen eingesetzt.

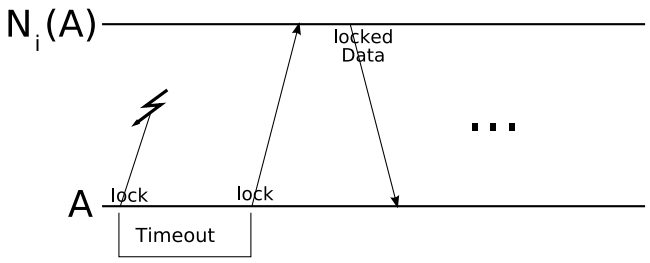


Abbildung 3.14: Eine nicht angekommene *Lock*-Nachricht wird nach einem Timeout erneut versandt

Wurde das Ausbleiben der Nachricht durch einen Ausfall von A hervorgerufen, ändert sich für das unmittelbar beteiligte Knotenpaar nichts. A ist nicht mehr aktiv und das Ziel der Kommunikation wusste noch nichts von dem Kommunikationsversuch. Alle Knoten, die vorher ge*lock*t wurden, erkennen über ihren Beaconing-

Mechanismus den Ausfall von A, brechen die Operation ab und schalten sich selbst frei.

Ausfall der locked/Data-Nachricht: Abbildung 3.15 zeigt den nächsten Fall: Den Ausfall der Bestätigungsnachricht, der *locked/Data*-Nachricht.

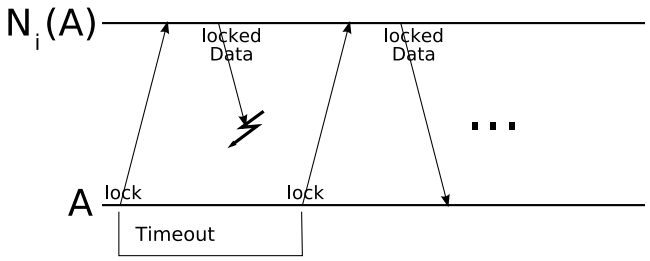


Abbildung 3.15: Eine nicht empfangene *Lock*-Bestätigung/*Data*-Nachricht führt zu einer Retransmission der *Lock*-Nachricht

Der dargestellte Fall kann eintreten, wenn der Zielknoten die *lock*-Nachricht empfangen und verarbeitet hat. Ist keiner der beiden Kommunikationspartner ausgefallen, d.h. die Nachricht ging auf dem Übertragungsweg verloren, erkennt A das über einen Timeout-Mechanismus. Dieser *Timeout* löst lediglich ein erneutes Senden der Nachricht aus, keinen Abbruch der Operation. Wie auch im vorherigen Fall ist der initiierende Knoten A für einen korrekten Versand der Nachrichten zuständig. Das bedeutet, wenn die Antwort auf seine Anfrage (das $LOCK$) ausbleibt, muss er nachfragen. Nach einem Timeout sendet A also $LOCK$ noch einmal. Dies wird so oft wiederholt bis eine Antwort eintrifft oder einer der beiden Knoten ausfällt bzw. die Reichweite verlässt. Die Wiederholungen werden mithilfe von Sequenznummern unterschieden.

Wenn ein Nachbarknoten ausfällt (bzw. die Reichweite verlässt) wird A vom *Beaconing*-Mechanismus informiert und entfernt seinen Nachbarn von der Liste. Fällt A aus, hebt der Nachbarknoten selbständig den *lock*-Zustand auf und kehrt in den Normalmodus zurück.

Ausfall der setData/unlock-Nachricht: Die *setData/unlock*-Nachricht ist eine Information von A, welche die durchzuführenden Änderungen an der Nutzung der Bandbreite und die Freischaltung des Knotens (*unlock*) enthält. Ein Ausfall dieser Nachricht wird ähnlich gehandhabt wie ein Ausfall der *lock*-Nachricht: Nachdem A eine bestimmte Zeit (*Timeout*) keine Antwort empfangen hat, sendet es die Nachricht erneut (siehe Abbildung 3.16), bis eine Antwort eingetroffen ist.

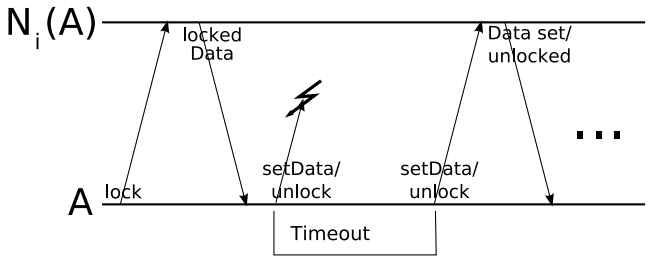


Abbildung 3.16: Eine verlorene *setData*-Nachricht wird nach einem Timeout erneut versandt

Ein Ausfall einer der beiden Stationen wird durch den *Beaconing*-Mechanismus der jeweils anderen erkannt:

- A entfernt den Nachbarknoten in diesem Falle von der Liste und fährt fort.
- Der Nachbarknoten erkennt, dass A nicht mehr in seiner Reichweite ist und hebt eigenständig den *lock* auf und kehrt in den Normalbetrieb zurück.

Ausfall der Data set/unlocked(ACK)-Nachricht: Der Ausfall der *Data set/ unlocked*-Nachricht, des *Acknowledgements* für die *setData/unlock*-Nachricht, wird ähnlich behandelt, wie in den vorhergehenden Fällen beschrieben (siehe Abbildung 3.17).

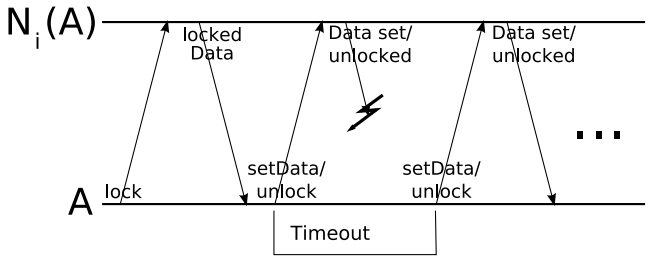


Abbildung 3.17: Eine verlorene*Data set/unlocked(ACK)*-Nachricht führt zu einem erneuten Senden der *setData/unlock*-Nachricht

Nachdem A eine bestimmte Zeit keine Antwort erhalten hat, sendet er die *setData/unlock*-Nachricht erneut, bis er eine Antwort in Form einer *Data set/ unlocked (ACK)*-Nachricht erhält oder durch die *Beaconing*-Schicht über den Ausfall des Knotens informiert wird.

Sollte A ausfallen, nachdem der Nachbarknoten das *unlock* bereits erhalten hat, ist das für den Ablauf des Protokolls nicht von Bedeutung, da es für den Nachbarn

ja bereits beendet ist. Die unnütze Reservierung wird durch das Beaconing erkannt (bzw. der Wegfall eines Senders) und dann aufgehoben.

Verhalten von gelockten Knoten bei Anfragen von anderen Knoten

Die im vorangegangenen Abschnitt beschriebenen Vorgehensweisen beim Verlust von Nachrichten müssen sich im Einklang mit den Aktionen bewegen, welche ausgeführt werden, wenn ein Knoten bereits von einem anderen Knoten ge*lock*t ist.

Wenn C beispielsweise bereits von A ge*lock*t ist und C nicht auf *lock*-Anfragen von B reagiert, darf B nicht annehmen, dass C ausgefallen ist. Es käme zu falschen Voraussetzungen (da C sehr wohl noch anwesend ist und Bandbreite benötigt). Es bedarf also einer genauen Unterscheidung der Fälle „Knoten ausgefallen“ und „Knoten bereits belegt“, wobei diese durch das Beaconing getroffen werden und im Protokoll selbst „nur“ optimiert werden sollten.

Abbildung 3.18 zeigt die resultierende Kommunikation, wenn Ausfälle ausschließlich von der Beacon-Schicht kontrolliert werden und keinerlei Rückmeldung vom kontaktierten Knoten erfolgt, wie es im Protokoll vorgesehen ist.

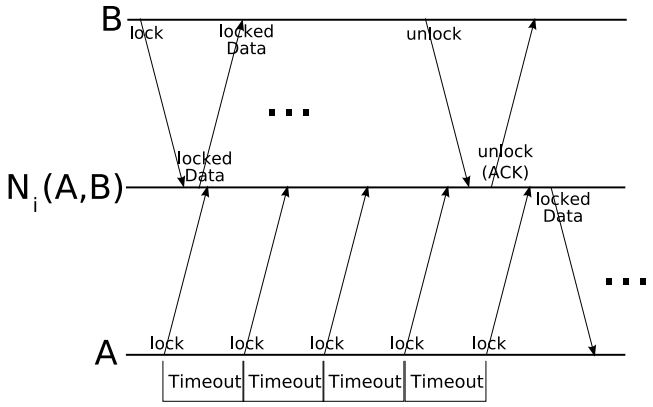


Abbildung 3.18: Wollen zwei Knoten einen gemeinsamen Bandbreiten-Nachbarn *lock*en, so muss einer warten, bis dieser frei ist.

Bei diesem Verfahren blockiert B den Knoten in der gemeinsamen Nachbarschaft mit A: $N_i(A, B)$, oder auch „C“. Dieser ignoriert fortan, bis zum *Unlock* alle Anfragen anderer Knoten als B. A erhält keine Antworten auf seine Anfragen, weiß aber durch den Beaconing-Mechanismus, dass C nicht ausgefallen (und immer noch in Reichweite) ist. Also wird das Senden der *Lock*-Anfragen nach einem Timeout wiederholt. Zur Reduzierung der Belastung des Mediums würde es sich anbieten, die *Lock*s auszusetzen oder zu reduzieren. Abbildung 3.19 zeigt ein Beispiel mit einem verlängerten Timeout.

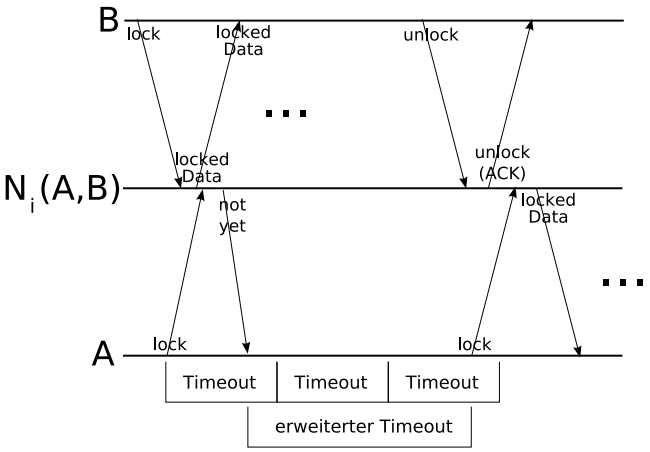


Abbildung 3.19: Die *notyet*-Nachricht signalisiert eine bereits aktive Operation. *A* reagiert mit einem erweiterten Timeout.

Wie im vorherigen Beispiel will *A* den bereits ge*lock*ten gemeinsamen Nachbarn belegen. Statt, wie vorher, keine Antwort zu geben, sendet der Knoten diesmal eine *not yet*-Nachricht. Er zeigt so an, dass er gerade belegt ist. *A* reagiert, indem er nicht den Standard-Timeout abwartet, sondern eine längere Wartezeit veranschlagt, bis er die nächste *Lock*-Anfrage sendet.

Auf diese Weise kann die Belastung des Netzes mit unnötigen Anfragen während der Wartezeit reduziert werden. Es besteht allerdings wegen der verlängerten Zeit zwischen den *lock*-Anfragen vermehrt die Gefahr, dass andere Knoten *A* zuvorkommen. Abbildung 3.20 skizziert diesen Fall.

Auch ohne verlängertes Timeout kann zu diesem „Wegschnappen“ der Chance auf ein *lock* kommen, eine erhöhte Wartezeit vergrößert aber gleichzeitig die Chance, dass dieser Fall eintritt. Soll diese „Contention“, dieser Wettbewerb, fairer gestaltet werden, müsste der kontaktierte Knoten eine Liste mit den Knoten führen, welche bei ihm angefragt haben. Ist eine Reservierungsphase beendet, würde er dem ersten Knoten in seiner Liste eine Nachricht zusenden und auf ein *lock* von Diesem Warten. Abbildung 3.21 illustriert diese Möglichkeit.

Schon auf dieser einfachen Skizze ist die Komplexität des Vorgehens erahnbar. Beachtet man, dass in der Zwischenzeit noch Knoten und Nachrichten ausfallen können, die Topologie sich ändern kann etc., ist dieses Schema auf den ersten Blick zwar fairer, aber zugleich unflexibler, aufwändiger bzw. fehleranfällig.

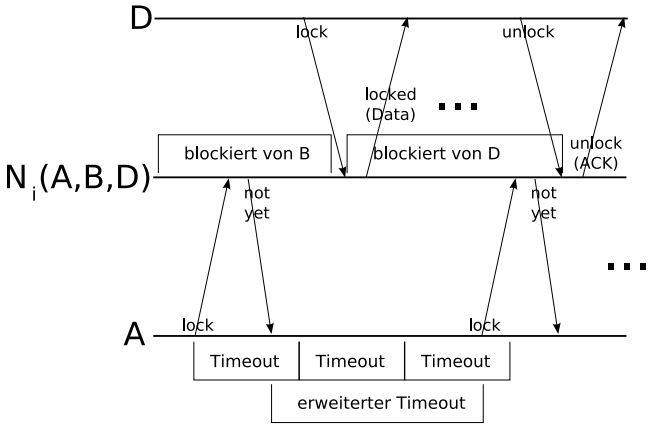


Abbildung 3.20: A verpasst seine Chance $C(N_i(A, B, D))$ zu *lock*en wegen der verlängerten Timeout-Zeit

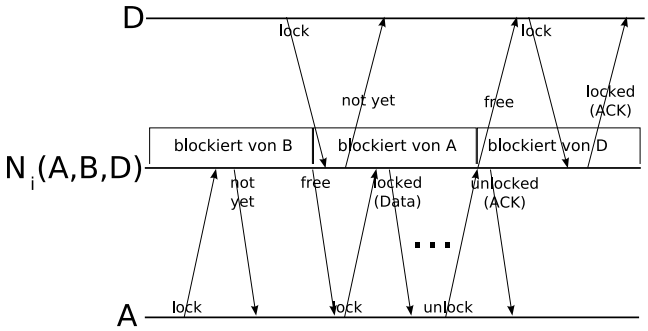


Abbildung 3.21: Durch *free*-Nachrichten kann explizit mitgeteilt werden, wann ein *Lock* beendet ist

```
solange das Listenende nicht erreicht ist:
   falls Qualitätsindex kleiner gleich null ist:
      Knoten aus der Liste entfernen;
   sonst:
      ersten Eintrag in Qualitätsindex-warteschlange
      löschen;
      falls der Zeitstempel des Eintrags älter als
      eine Periode ist:
         eine "Null" in die Qualitätsindex-warteschlange
         einfügen;
      sonst:
         eine "Eins" in die Qualitätsindex-warteschlange
         einfügen;
      ende_falls;
   ende_falls;
   Qualitätsindex=0;
   solange Einträge in der Qualitätsindex-warteschlange
   sind:
      falls Eintrag eine "Eins" ist:
         Qualitätsindex erhöhen;
      ende_falls;
      nächsten Eintrag auswählen;
   wiederholen;
nächsten Knoten aus der Liste holen;
wiederholen;
```

Tabelle 3.8: 2. Aktualisierungsalgorithmus

3.5.4 Algorithmus

Nach den bisherigen Vorüberlegungen soll dieser Abschnitt nun den konkreten Algorithmus für das Reservierungsproblem vorstellen.

Der Algorithmus beginnt im „Normalbetrieb“, d.h. es wird weder eine Reservierung durchgeführt noch ist der Knoten durch Anfragen anderer Knoten blockiert – es können mit dem Protokoll nicht in Zusammenhang stehende Anwendungsprogramme aktiv sein. Durch zwei Ereignisse kann dieser Modus verlassen werden (siehe Abbildung 3.22 S.54).

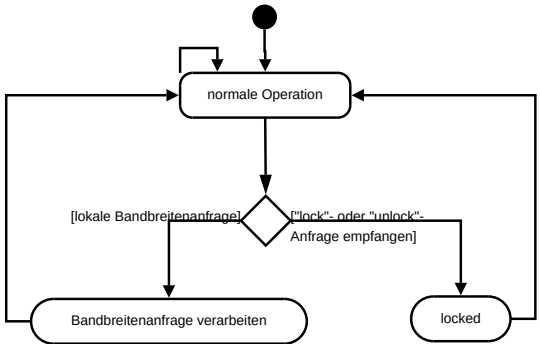


Abbildung 3.22: Die drei Hauptzustände bei der Abarbeitung des Bandbreitenreservierungprotokolls

Eine lokale Reservierungsanfrage einer Applikation aktiviert den „Bandbreitenanfrage verarbeiten“-Zustand. Sendet ein anderer Knoten eine *lock*-Nachricht, so wechselt der Zustand zu „locked“, um Anfragen eines anderen Knotens bearbeiten zu können.

Bandbreitenanfrage verarbeiten

Im „Bandbreitenanfrage verarbeiten“-Zustand (siehe Abbildung 3.23 S. 57) wird eine lokale Anforderung einer Reservierung bearbeitet.

Der Vorgang beginnt mit einer Initialisierung der Datenstrukturen und der Prüfung der lokal vorhandenen Bandbreite. Ist bereits lokal nicht genügend Bandbreite vorhanden, d.h. die Ströme über den aktuellen Knoten nutzen die Bandbreite schon so stark aus, dass keine Ressourcen mehr frei sind, wird die Anfrage abgelehnt.

Im nächsten Schritt wird die *Beaconing*-Komponente kontaktiert, welche Informationen über die Nachbarschaft bereitstellt. Die Nachbarschaftsinformationen werden in eine Arbeitsliste kopiert. Aus dieser Liste wird der erste Knoten geholt.

Drei Fälle sind möglich:

1. Ein Knoten wird aus der Liste geholt und keiner mit einer kleineren Kennung ist in der Liste.

2. Ein Knoten ist neu in der Liste, hat aber eine kleinere Nummer, als der aktuell betrachtete. In diesem Falle können die Knoten in der Nachbarschaft nicht in der korrekten Reihenfolge ge*lock*t werden und das Protokoll muss abgebrochen werden. (Details zum Rücksetzen der Knoten in Abschnitt 3.5.4 und in Abbildung 3.25)

3. Die Liste ist leer. In diesem Falle sind alle Knoten ge*lock*t und die *setData/ Unlock-Phase* (siehe Abschnitt 3.5.4 und Abbildung 3.24) beginnt, in deren Anschluss wieder in den „Normalmodus“ gewechselt wird.

Im ersten Fall wird an den Knoten aus der Liste eine *lock*-Nachricht gesendet. Trifft eine falsche Antwort ein oder wird eine bestimmte Zeit lang keine Nachricht empfangen (es kommt zu einem *Timeout*), erfolgt eine Prüfung. Diese stellt fest, ob sich der angesprochene Knoten noch in der Nachbarschaft befindet. Ist dies nicht mehr der Fall, wird er übergangen und mit dem nächsten Nachbarknoten fortgefahren. Ist er noch in der Nachbarschaft, so sind lediglich Verzögerungen bzw. Nachrichtenverluste aufgetreten. Das Senden der Nachricht wird wiederholt und erneut auf eine Antwort gewartet.

Wird die erwartete Antwort empfangen (ein *LOCKED* inklusive der lokalen Bandbreiteninformationen), wird mit dem nächsten Knoten in der Liste fortgefahren, wie oben beschrieben.

Knoten zurücksetzen

Kommen während des Ablaufs des Protokolls neue Knoten hinzu, muss unter Umständen die Verarbeitung abgebrochen werden. Der Ablauf wird in Abbildung 3.25 wiedergegeben.

Die bisher ge*lock*ten Knoten werden in eine Liste eingetragen. Aus dieser Liste wird der erste Knoten entnommen und eine *unlock*-Nachricht an diesen gesendet. Trifft eine Antwort ein, wird der nächste Knoten aus der Liste geholt. Trifft keine Antwort innerhalb einer bestimmten Zeit ein (*Timeout*), wird der Beaconing-Mechanismus genutzt, um festzustellen, ob der Knoten noch in der Nachbarschaft ist. Wenn der Knoten nicht mehr in der Nachbarschaft ist, wird der nächste Knoten aus der Liste geholt, andernfalls wird die Nachricht erneut versendet und gewartet.

setData/Unlock-Phase

Nach der Berechnung der Bandbreite wird das Ergebnis in der *Update/UnLOCK*-Phase den Knoten mitgeteilt und das *LOCK* aufgehoben.

Der Ablauf ist in Abbildung 3.24 dargestellt. Alle Knoten sind in einer Liste eingetragen. Aus dieser Liste wird der erste Knoten entnommen. Ist die Liste leer, sind alle Knoten freigegeben und die Funktion kann beendet werden. Ansonsten wird eine *UnLOCK*-Nachricht an den Knoten gesendet. Trifft die Antwort ein, wird der nächste Knoten aus der Liste geholt. Bei einem Timeout erfolgt auch hier die Überprüfung der Beacon-Liste und ein erneutes Senden, falls der Knoten nicht die Nachbarschaft verlassen hat.

Der Vorgang wird wiederholt, bis keine Einträge mehr in der Liste sind. Ist dies der Fall, ist der Zugriffskontrollmechanismus für einen Strom auf einem Knoten abgeschlossen.

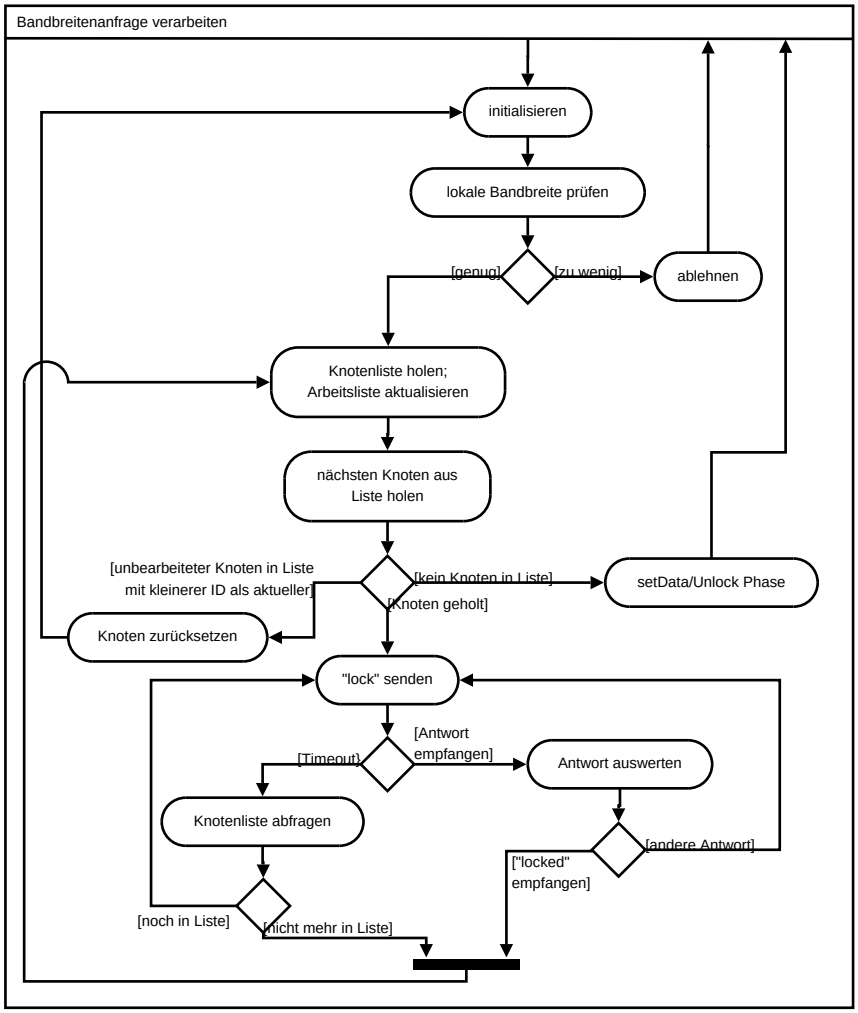


Abbildung 3.23: Diagramm des Ablaufs des Protokolls bei eingegangener Anfrage nach Bandbreite

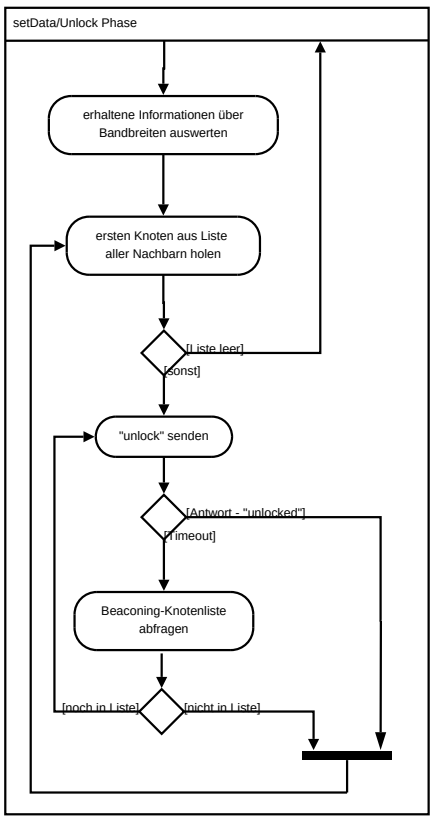


Abbildung 3.24: Das Diagramm zeigt die zweite Phase: das Mitteilen der Entscheidung und das *unlock*

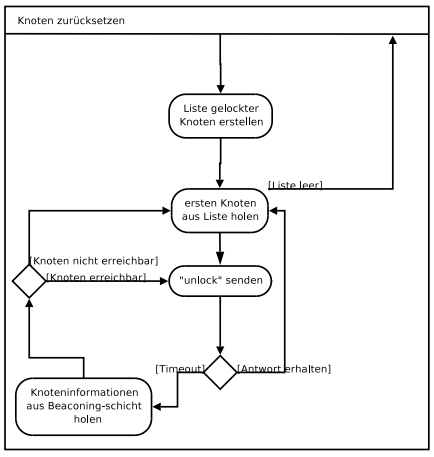


Abbildung 3.25: Die Freischaltung der Knoten aus dem *locked*-Zustand

4 Implementierung

Im Kapitel 3 wurde die konzeptionelle Lösung der Aufgabenstellung diskutiert und beschrieben. Das folgende Kapitel erläutert Details zur Implementierung. Dabei werden die drei Teilprobleme in eigenen Abschnitten erläutert, wobei der dritte Abschnitt die Zugriffskontrolle und damit das Zusammenspiel der drei Komponenten illustriert.

4.1 Bestimmung der Nachbarschaft - Multi-Hop-Beaconing

Dieser Abschnitt beschreibt, wie Informationen über die Nachbarschaft eines Knotens gewonnen werden. Die Kenntnis der Nachbarschaft ist unbedingt notwendig, um sich mit den Knoten in der Umgebung koordinieren und so überhaupt kooperative Aktionen durchführen zu können.

Die direkte „Eins zu Eins“-Kommunikation mit einem anderen Knoten ist möglich, wenn er sich erstens in der durch den Transceiver bestimmten Reichweite befindet und zweitens seine Adresse bekannt ist.

Die vorgesehene Funktionalität soll die gesamte Nachbarschaft erkunden, also auch Knoten deren Existenz und damit ihre Adresse unbekannt ist. Es wird also keine gezielte Übertragung zu einer bestimmten Zieladresse initiiert, sondern ein *Broadcast* genutzt. Ähnlich dem *Beacon* eines „Access-Points“ wird periodisch ein Paket gesendet. Dieser Pakettyp wird von allen Knoten in Reichweite empfangen und ausgewertet, aber nicht weitergeleitet. Die Adresse des Absenders ist in diesem Paket enthalten und kann so sicher der eigenen Nachbarschaft zugeordnet werden.

Im Abschnitt 2.1 werden die Besonderheiten bei der Ausbreitung von Funkwellen und die daraus resultierenden Einflüsse auf die Bandbreiten der Umgebung diskutiert. Die wichtigste Erkenntnis: Auch die verfügbare Bandbreite von Knoten außerhalb des Sende-/Empfangsbereiches wird beeinflusst. Die Kenntnis der Knoten innerhalb der eigenen Sende-/Empfangsreichweite reicht also nicht aus, wenn man die Menge der Knoten erstellen will, welche von einem Sendevorgang beeinflusst werden. Aus diesem Grund müssen zusätzlich zu den direkten Nachbarn in diese *Bandbreitennachbarschaft* auch deren unmittelbare Nachbarn aufgenommen werden (Zwei-Hop-Nachbarschaft).

Die Zwei-Hop-Nachbarschaft (oder allgemeiner die n-Hop-Nachbarschaft für $n > 1$) kann mit den Fähigkeiten eines einzelnen Tranceivers nicht erkundet werden.

Das *Beacon* wird deshalb mit weiteren Informationen angereichert. Es enthält eine Liste der Nachbarschaft des sendenden Knotens inklusive der Angabe der Distanz der anderen Knoten (in Hops). Tabelle 4.1 zeigt den Aufbau des Datenteils eines entsprechenden Paketes.

Feld	Datentyp	Beschreibung
Anzahl folgender Einträge	unsigned char	Das Byte am Anfang jeder Gruppe gibt die Anzahl der folgenden Einträge an.
Knoten ID	StationID (4 byte)	ein eindeutiger Bezeichner eines Knotens im Netzwerk
...		
Anzahl folgender Einträge	unsigned char	Das Byte am Anfang jeder Gruppe gibt die Anzahl der folgenden Einträge an.
Knoten ID	StationID (4 byte)	ein eindeutiger Bezeichner eines Knotens im Netzwerk
... usw.		
Endmarke	unsigned char(0xFF)	Das Byte mit dem Wert 0xFF markiert das Ende der Daten. Es steht anstelle des „Anzahl folgender Einträge“-Feldes

Tabelle 4.1: Beacon-Paket: Datenfelder, schematische Darstellung

Wegen der nicht vorhersagbaren Anzahl der Knoten in der Umgebung ist ein flexibles Paketformat notwendig. Der Datenteil besteht deshalb aus Gruppen, welche jeweils von einer Größenangabe eingeleitet werden („Anzahl folgender Einträge“). Die Position dieser Gruppen gibt implizit die Entfernung der darin aufgezählten Knoten wieder, d.h. die erste Gruppe beinhaltet alle Knoten der unmittelbaren 1-Hop-Nachbarschaft, die zweite Gruppe alle Knoten der 2-Hop-Nachbarschaft usw. Die Gruppen bilden also die verschiedenen Nachbarschaftsbereiche. Auf die Größenangabe folgt in jeder Gruppe eine Auflistung der Knotenkennungen. Diese bestehen in der vorliegenden Implementierung aus 4 Byte langen, von IP-Adressen abgeleiteten Datentypen.

Das Ende des Datenteils wird durch einen reservierten Wert in der Größenangabe dargestellt. Dieser Wert ist die größte Zahl des gültigen Wertebereiches, bei einem „unsigned char“ also $0xFF$. Nach diesem Eintrag in einem Größenangabefeld endet der Datenteil.

Eine entscheidende Frage betrifft die Wahl des Datentyps für die Größe des Nachbarschaftsbereiches. Durch die Beschränkung der Größenvariable auf ein Byte und

der weiteren Einengung des Wertebereichs durch die Nutzung von „0xFF“ als Endmarke liegt die Obergrenze mit dieser Implementierung bei 254 Knoten pro Nachbarschaftsbereich (Wertebereich eines *unsigned char* minus eins, weil ein Wert eine Sonderfunktion erfüllt). Ist diese Zahl ausreichend oder muss ein größerer Wert eingeplant werden? Die bestätigte Annahme ist, dass es eine Grenze für die Arbeitsfähigkeit des genutzten Protokolls nach IEEE802.11 gibt. Eine größere Anzahl Knoten würde die Funktionsfähigkeit des genutzten MAC-Layers an die Grenzen der Leistungsfähigkeit bringen würde (siehe Abschnitt 2.1.1). In einem durchschnittlichen Anwendungsszenario wird die Knotendichte von über 200 Knoten in einer Nachbarschaftsebene nicht erreicht. Nichtsdestotrotz ist eine Änderung des verwendeten Datentyps im Bedarfsfall recht problemlos möglich. Die Notwendigkeit erscheint jedoch momentan unwahrscheinlich: Eine Anpassung würde nur durch eine Änderung der Eigenschaften des MAC-Layers notwendig werden. In diesem Falle muss die gesamte Implementierung an die hypothetischen neuen Eigenschaften angepasst werden.

Die Verwaltung der Nachbarschaftsinformationen innerhalb eines Knotens findet nicht in der oben beschriebenen Form statt. Änderungen wären in einer solchen Datenstruktur nur mit ungenügend hoher Komplexität durchführbar, da jede Änderung ein Verschieben aller folgenden Daten nach sich ziehen würde. Die Nachbarschaftsinformationen werden stattdessen in einer Liste abgelegt, in welcher ein Eintrag einen Knoten repräsentiert. Die genutzte Standard-Template-Library-(STL)-Implementierung stellt einen Kompromiss zwischen schnellem Einfügen und Löschen und schnellen Such- und Sortierfunktionen dar. Ein Eintrag, welcher einen Nachbarn repräsentiert, enthält dabei Informationen, wie in Tabelle 4.2 dargestellt.

Timestamp	ein Zeitstempel in der *GEA*-Repräsentation eines Zeitpunktes enthält das Alter der Informationen, d.h. zu diesem Zeitpunkt war der Knoten das letzte mal in einem *Beacon* enthalten.
KnotenID	eine eindeutige Kennzeichnung des Knotens (ein von der IP-Adresse abgeleiteter Datentyp).
NextNeighbour	vom selben Typ wie die *KnotenID*, enthält dieses Feld den nächsten Knoten in der eigenen Nachbarschaft, über den die Station erreicht werden kann. Bei Routingprotokollen wird in ähnlichem Zusammenhang von „next Hop“ gesprochen.
distance	ist die Distanz des Nachbarn in *Hops*.
quality	ist ein Parameter, welcher die Qualität der Verbindung zu einem Knoten darstellt.
init	eine Variable, die einen neu hinzugekommenen Knoten anzeigt.

Tabelle 4.2: Die Felder eines Eintrages in der Nachbarschaftsliste

Die empfangenen Daten werden nach der Auswertung in eine temporäre Liste eingetragen. Anschließend werden die Knotenkennungen aus dieser Liste in der Nachbarschaftsliste gesucht. Sind sie nicht enthalten, werden sie im Initialisierungszustand hinzugefügt. Einträge, welche bereits in der Nachbarschaftsliste enthalten sind, werden aktualisiert – der Timestamp wird auf den aktuellen Zeitpunkt gesetzt.

Werden Informationen über denselben Knoten gleichzeitig von mehreren *Beacons* geliefert, wird der Eintrag mit der geringsten Distanz genutzt. Sind die Distanzen gleich, wird nur der erste Eintrag beibehalten und aktualisiert, andere werden ignoriert bis der erste Eintrag ungültig wird bzw. veraltet.

Knoten, die in der Nachbarschaftsliste, aber in keinem der *Beacons* enthalten sind, werden nicht sofort entfernt, um etwa bei Verlusten des *Beacons* keine falschen Nachbarschaftsinformationen zu erzeugen. Das Entfernen eines Knotens erfolgt ausschließlich anhand des *Quality*-Parameters.

Der *Quality*-Parameter dient der Einschätzung, wann ein Knoten nicht mehr in der Nachbarschaft enthalten ist bzw. wie stabil die Verbindung zu ihm ist. Der Qualitätsindex beschreibt, in wievielen der letzten n *Beacon*-Perioden der Knoten in einer der Nachbarschaften enthalten war. Zu diesem Zweck wird in der einfachsten Variante der Qualitätszähler zu Beginn mit dem Maximalwert initialisiert. In jeder Periode wird dieser Zähler erst um eins verringert und dann um eins erhöht, falls der Knoten in einem der *Beacons* enthalten ist.

Die andere Methode sollte eingesetzt werden, wenn eine hohe Mobilität zu erwarten ist, was dazu führt, dass ein Knoten die Nachbarschaft häufig verlässt und wieder zurückkehrt. Der oben erwähnte Qualitätsindex kann nämlich offensichtlich nur sinken, aber nicht steigen. Ein Knoten, der beispielsweise zu Anfang kaum in Reichweite lag, wird einen niedrigen Index behalten, auch wenn später eine gute Verbindung besteht. Die alternative Qualitätsindex-Berechnung nutzt eine Warteschlange der Größe n. In jeder Periode wird das älteste Element entfernt und ein Neues hinzugefügt. Jedes Element enthält nur einen boolschen Wert, welcher *true* ist, falls in dieser Periode der Knoten in der Nachbarschaft lag, *false* falls nicht. Im Anschluss werden die Elemente gezählt, welche *true* sind. Diese Anzahl ist der Qualitätsindex und wird in die entsprechende Variable geschrieben.

4.2 Bandbreitenkontrolle mittels Traffic-Shaping

Im folgenden Abschnitt wird erläutert, wie die vereinbarte Bandbreite lokal durchgesetzt werden kann.

Es soll ausgeschlossen werden, dass fehlerhafte Applikationen die ihnen zugewiesene Bandbreite überschreiten. Es wird bereits vor dem physischen Senden eines Paketes geprüft, ob die zugewiesene Bandbreite noch zur Verfügung steht.

Wie bereits in Kapitel 3 erwähnt, kann dazu ein Algorithmus zur „Formung“ des

Netzwerkverkehrs eingesetzt werden. Ein *Leaky Bucket* Algorithmus erlaubt pro Zeiteinheit nur eine bestimmte Menge an Daten zu übertragen. Alle überzähligen Pakete werden verworfen.

Die Implementierung nutzt zwei Stufen. Die unterste Stufe ist eine einzelne Warteschlange, welche die enthaltenen Pakete direkt an die darunterliegenden Schichten sendet. Darüber liegen mehrere Warteschlangen, welche von einzelnen Applikationen Pakete annehmen und in die darunterliegende Warteschlange einreihen. Die Details erläutern die folgenden Abschnitte.

4.2.1 Bandbreitenkontrolle pro Knoten

Algorithmen nach dem *Leaky Bucket*-Protokoll können einen Datenstrom formen. Alle Daten, welche von einem Knoten aus versandt werden müssen die Ausgangs-*Queue* des Traffic-Shapers durchlaufen. Diese Queue ist eine FIFO-Implementierung (eine *Warteschlange*) basierend auf einer Liste (*list*) der *Standard Template Library – STL*.

Der Vorteil der Listenimplementierung der *STL* ist die konstante Komplexität der Operationen (u.a.) „Hinzufügen am Listenende“ und „Entnehmen des Elements am Listenanfang“, welche bei einer ständigen Durchleitung von Paketen die bei weitem häufigsten Operationen sind.

Die *list* wurde in einer Klasse gekapselt, welche den Zugriff auf die Liste mit den für Warteschlangen typischen Befehlen „push(Element)“ und „pop(Element)“ erlaubt. Außerdem ist die Zahl der Einträge in die Liste begrenzt worden. Der *Leaky Bucket*-Algorithmus benötigt eine Warteschlange mit endlich vielen Einträgen, die Listenimplementierung sieht nur einen durch Systemressourcen begrenzten Maximalwert vor. Eine *push*-Operation fügt nur dann ein Element in die Liste ein, wenn die gegebene maximale Größe nicht überschritten ist, ansonsten werden die Elemente verworfen.

Die Elemente der FIFO müssen sie lediglich zur richtigen Zeit in der Reihenfolge wieder verlassen, deshalb enthalten die Elemente keine Zusatzinformationen, sondern bestehen nur direkt aus den Paketen, welche durch das Protokoll verarbeitet werden.

Wird ein Paket aus der Warteschlange genommen und versandt, berechnet der Algorithmus den Zeitpunkt, zu dem das nächste Paket versandt werden kann (siehe Abschnitt 3.4.3). Dieser Zeitpunkt hängt von der einstellbaren maximalen Bandbreite und der Größe des versandten Paketes ab. Vermerkt wird dieser Zeitpunkt in einer Variable („nextSend“). Befindet sich ein weiteres Paket in der Warteschlange wird die Sendeoperation mittels der entsprechenden GEA-Funktion [REFERENZ] zum Zeitpunkt „nextSend“ neu gestartet (siehe Abbildung 4.1).

Steht kein weiteres Paket zum Versand bereit, endet der Ablauf. Dieser Zustand wird in einer Variablen („nextScheduled“) vermerkt.

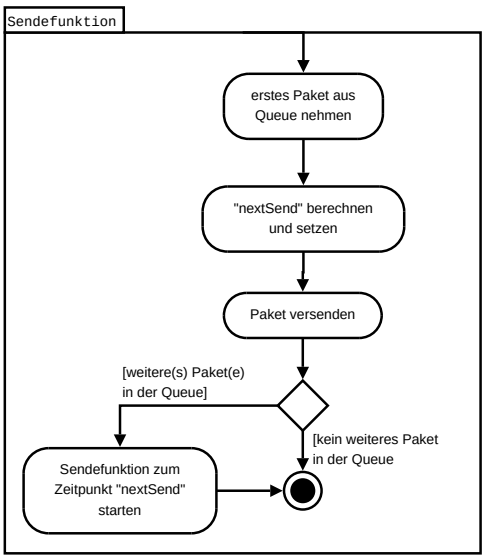


Abbildung 4.1: Die Sendefunktion versendet Pakete zu dem berechneten Zeitpunkt

Die Variable „nextScheduled“ wird von der Funktion „readyToSend“ genutzt. Diese wird aufgerufen, wenn neue Pakete in die Warteschlange gelegt werden. (siehe Abbildung 4.3)

Werden gerade Pakete versandt (die Variable „nextScheduled“ ist *true*) endet die Funktion, da das eingefügte Paket dann ebenfalls versandt wird. Ist dies nicht der Fall („nextScheduled“ ist *false*), wird der Zustand geändert („nextScheduled“ auf *true* gesetzt) und die Sendefunktion aktiviert. Wann sie aktiviert wird, hängt von der Variable „nextSend“ ab. Liegt ihr Wert in der Zukunft, bedeutet dies, dass die Bandbreite bereits benutzt wird. In diesem Falle wird das Versenden auf den Zeitpunkt „nextSend“ festgelegt, ansonsten wird die Versandfunktion sofort gestartet.

4.2.2 Bandbreitenkontrolle der Anwendungen

Im vorhergehenden Abschnitt wurde beschrieben, wie der ausgehende Verkehr kontrolliert wird, indem in eine Queue sämtliche Pakete eingefügt und zeitgesteuert versandt werden. Zur Kontrolle der den einzelnen Anwendungen zugeteilten Bandbreiten wird ähnlich verfahren. Alle Pakete, welche von einer Anwendung kommen, dürfen nicht die für diese Applikation reservierte Bandbreite überschreiten.

Pakete von Applikationen sind netzweit eindeutig gekennzeichnet. Die benutzte *ConnectionID* wird aus Quelle, Ziel und *ServiceID* gebildet. Die *ServiceID* hat in etwa die selbe Bedeutung wie eine TCP-Port-Nummer, d.h. sie dient der Unterschei-

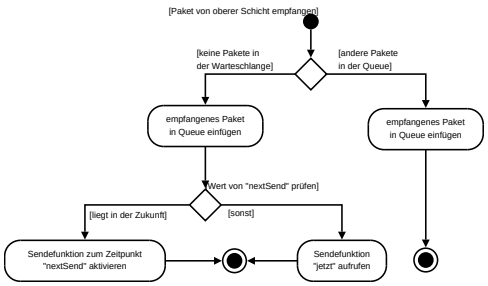


Abbildung 4.2: Einfügen eines neuen Pakets in die Sendequeue

dung verschiedener Services (Applikationen bzw. Funktionalitäten von Applikationen).

Die ConnectionID und die zugehörige Bandbreite übermittelt die Zugriffskontrollkomponente nach einer positiven atomaren Entscheidung der Bandbreitenkontrollkomponente. Mit diesen Daten wird eine Instanz der Unterklasse des TrafficShapers erzeugt.

Zur Verwaltung der Objekte dieser Unterklasse wird die STL-Implementierung einer *Map* genutzt. Diese Implementierung ist für diesen Fall besonders geeignet, da Einträge über die ConnectionID als Schlüssel referenziert werden können. Die Komplexität für Vergleichsoperationen ist konstant, Suchen erfolgt in logarithmischer Zeit.

Die in diese Map eingefügten Objekten enthalten jeweils folgende Attribute:

- eine FIFO wie im Abschnitt 4.2.1
- einen Zeiger auf die „OutFIFO“, eine FIFO, welche die Pakete von allen Anwendungen vor dem Senden sammelt (siehe Abschnitt 4.2.1)
- den Zeitpunkt des nächstmöglichen Sendens
- einen Zeiger auf eine Funktion, welche die „OutFIFO“ dazu bringt Pakete zu versenden

Empfängt der Traffic-Shaper ein Paket von der darüberliegenden Schicht, wird es anhand der ConnectionID identifiziert und in die *FIFO* eingefügt.

Das Entnehmen aus der Warteschlange erfolgt nach demselben Prinzip, wie oben beschrieben. Anhand der Größe des Paketes und der Bandbreite, welche diese Queue nutzen darf, wird der Zeitpunkt berechnet, zu dem das nächste Paket versandt werden darf. Ist mindestens ein Paket in der Warteschlange, wird die Sendefunktion zu

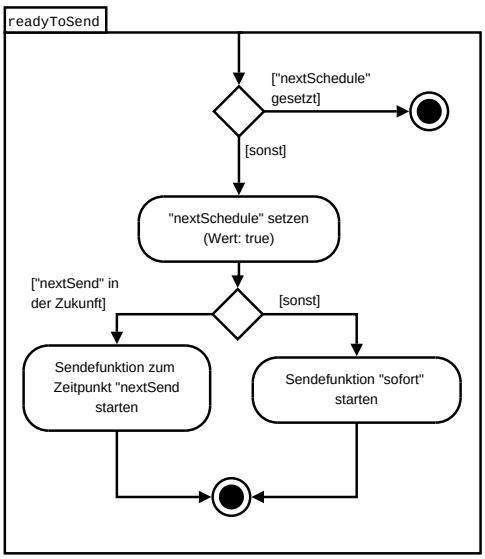


Abbildung 4.3: „readyToSend“ startet, wenn nötig, das Versenden von Paketen

diesem Zeitpunkt gestartet. Ist die Queue leer, endet die Sendefunktion und wird erst durch ein neu eintreffendes Paket reaktiviert.

4.3 Atomare Entscheidung

Der folgende Abschnitt beschreibt die Implementierung des Entscheider-Algorithmus, welcher über die Zulassung eines Datenstromes bestimmt. Die Implementierung benutzt die beiden vorher erläuterten Programme zur Gewinnung von Informationen (Beaconing) bzw. zur Durchsetzung der getroffenen Entscheidungen (Bandbreitenkontrolle).

Die Klasse implementiert sowohl den aktiven Teil, welcher die Bandbreiten seiner Nachbarn erfragt und anhand derer die Entscheidung über die lokale Zulassung des Stroms trifft, als auch die Antwortfunktionen. Beide Funktionen werden nicht gleichzeitig aktiv sein, so dass eine einfache Variable in der Klasse zur Unterscheidung ausreicht.

Im Normalmodus (siehe Abbildung 3.22 im Konzeptteil auf Seite 54) wird eine Schnittstelle für höhere Protokollschichten angeboten, über welche Anfragen gestellt werden können.

Die Primitive dieser Schnittstelle sind ein „Event“ und ein *Struct* „Reservierungsinformation“ welches die Details der Anforderung enthält (siehe Tabelle 4.3).

Die aufrufende Applikation sendet die Parameter der Reservierung in beschrie-

Name	Datentyp	Beschreibung
ID	ConnectionID	netzweit eindeutige Kennung des Datenstroms
bandwidth	unsigned long	Bandbreite in $\frac{byte}{s}$, wenn „0“ werden die folgenden Einträge ausgewertet
maxPacketSize	unsigned int	maximale Größe eines zu versendenden Pakets (byte)
period	unsigned int	(kleinste) Periode – Zeit zwischen zwei Paketen

Tabelle 4.3: Aufbau des Reservierungsinformation-Struct

benem *Struct* zusammen mit einem Zeiger auf eine eigene Funktion. Über diese *Callback*-Funktion wird nach Abschluss der Reservierungsoperation das Ergebnis mitgeteilt. Implementiert sind derzeit „ACCEPT“ für eine erfolgreich durchgeführte Reservierung, „REJECT“ für eine negativ verlaufene Reservierungsanfrage und „WAIT“, falls die Anfrage erst später bearbeitet werden kann.

Nach Eingang einer Anfrage wird eine „Working-List“, eine Arbeitskopie der aktuellen Nachbarschaftsliste, angelegt. Zusätzlich wird eine Liste angelegt, welche für *Timeouts* genutzt wird.

„Lock“-Phase:

Aus der Working-List wird der erste Eintrag entnommen. Die „Lock“-Nachricht wird an die KnotenID über den eingetragenen nächsten Nachbarn (*NextNeighbour*) gesandt. Der Knoten wird zur Timeoutliste hinzugefügt und der Aufruf der Timeout-Funktion ge*schedule*d.

Der reguläre Ablauf sieht das Eintreffen der „Locked/Data“-Nachricht von dem befragten Knoten vor. Daraufhin wird der Knoten in der Working-List als „bearbeitet“ markiert und die Liste mit der aktuellen Nachbarschaftsliste synchronisiert. Liegt in der sortierten Liste ein nicht bearbeiteter Knoten, hat sich die Nachbarschaft geändert, ein *Deadlock* ist möglich und das Protokoll wird abgebrochen. Ansonsten wird mit dem nächsten Knoten in der Working-List fortgefahren.

Beim Aufruf der Timeout-Funktion, liest diese zuerst den ersten Eintrag aus der Timeout-Liste. Sind dann noch weitere Einträge in der Liste, beendet sich die Funktion, da nie mehr als ein Timeout aktiv ist. Nur der letzte Listeneintrag enthält die Information des aktuellen Timeouts. Danach prüft das Programm, ob die „Lock“-Phase noch aktiv ist. Wenn ja, werden die Knotendaten in der Working-List gesucht. Ist der Knoten dort nicht als „bearbeitet“ markiert, ist ein Timeout aufgetreten. Das bedeutet, dass bisher keine Antwort auf die Anfrage eingetroffen ist. Die Sendefunktion wird erneut für diesen Knoten aufgerufen, d.h. der Listenzeiger der Working-List

wird zurückgesetzt und die Sendefunktion gestartet.

Bandbreitenberechnung:

Die angesprochenen Stationen empfangen die Anfrage aus den unterliegenden Schichten und werten sie aus. Eine Variable unterscheidet zwischen *locked* und *unlocked*-Zustand. Im *unlocked*-Zustand, sendet die Station mit einem „Locked/Data“-Paket die Antwort. Befindet sich der Knoten im emphlocked-Zustand, wird der Absender geprüft. Entspricht der Absender dem Knoten, welcher das ursprüngliche *lock* gesendet hat, darf der Knoten antworten. Bei Anfragen von allen anderen Knoten wird keine Antwort gesendet bzw. optional ein Paket verschickt, welches den „belegt“-Zustand mitteilt.

Das „Locked/Data“-Paket (Tabelle 4.4) beginnt mit einem kurzen Header mit Versionsinformationen. Der Rest des Paketes besteht aus mehreren Teilen. Die Teile beginnt mit der Angabe der Anzahl der folgenden Paare. Ein Paar besteht aus Ziel und Bandbreite, welche zu dem Ziel genutzt wird. Kurz: Es sind Tupel aus Bandbreite und Ziel der versandten Pakete.

Nach dem Ende dieses Teils folgt wieder eine Größenangabe, welche diesmal die Zahl der folgenden Quelle-Bandbreite-Paare angibt. Diese Paare geben jeweils Sender und für den Empfang von diesem Sender genutzte Bandbreite an. Wird das RTS-CTS-Verfahren nicht genutzt (siehe Abschnitt 3.2) kann der zweite Teil entfallen. Das Feld „Anzahl Receives“ gibt in diesem Fall die Länge „0“ für den zweiten Teil an.

Feld	Datentyp	Beschreibung
Version	unsigned char	verwendete Protokollversion
Anzahl „Sends“	unsigned char	Zahl der folgenden Sendetupel
Ziel	StationID	Eindeutiger Bezeichner des Ziels des Paketes
Bandbreite	unsigned long	zum Senden an das Ziel genutzte Bandbreite
weitere Ziel/Bandbreite-Paare		
Anzahl „Receives“	unsigned char	Zahl der folgenden Sendetupel
Quelle	StationID	Eindeutiger Bezeichner des Senders des Paketes
Bandbreite	unsigned long	zum Empfangen benötigte Bandbreite
weitere Quelle/Bandbreite-Paare		

Tabelle 4.4: Aufbau des *Locked/Data*-Paketes

Die empfangenen Pakete werden ausgewertet indem die Inhalte der Pakete in eine Sende- bzw. Empfangs-Queue eingetragen werden. Dabei werden aus den Daten und der Absenderadresse Tripel wie in Abschnitt 3.2 beschrieben gebildet. Die Einträge, welche die versendete Bandbreite betreffend, werden in die Sende-Queue eingetragen, in die Empfangs-Queue kommen Daten über die Empfangsbandbreiten.

Am Ende der *Lock*-Phase werden diese Queues sortiert. Ein Iterator durchläuft die Sende-Queue. Zu jedem Eintrag in der Sende-Queue wird der gleiche Eintrag in der Empfangs-Queue gesucht. Einträge sind gleich, wenn Sender, Empfänger und Bandbreite gleich sind. Diese Gleichheit bedeutet, dass es sich um die gleichen Übertragungsvorgänge handelt und nur einer davon gezählt werden darf. Bei Gleichheit wird der Eintrag aus der Empfangs-Queue gelöscht.

Durch die Sortierung sind gleiche Übertragungen in beiden Queues in derselben Reihenfolge angeordnet. Durch die zu erwartende hohe Zahl gleicher Pakete wird die Suche so auf die ersten Pakete der Queue beschränkt. Die ansonsten quadratische Komplexität der Operation (bezogen auf die Zahl der Übertragungen in der Nachbarschaft), liegt auf diese Weise unter der Komplexität einer binären Suche in der Empfangs-Queue.

Am Ende der Operation stehen in der Empfangs-Queue nur die Einträge, bei welchen der sendende Knoten außerhalb der Bandbreiten-Nachbarschaft ist. Die Bandbreiteneinträge beider Listen werden aufsummiert. Das Ergebnis ist die aktuell genutzte, „subjektive“ Bandbreite in der Umgebung des Knotens, welcher das *Lock*-Protokoll initiiert hat.

„Unlock“-Phase:

Nach der Ermittlung der Bandbreite wird entschieden, ob genug Bandbreite vorhanden ist. Wenn die angeforderte Bandbreite kleiner oder gleich der freien Bandbreite ist, wird die Anfrage zugelassen. Die Entscheidung wird zusammen mit der *Unlock*-Nachricht an alle Knoten der Nachbarschaft verteilt. Der Ablauf ähnelt der *Lock*-Phase. Nach dieser Phase liegt eine Liste aller ge*lock*ter Knoten vor.

Die Sendefunktion sendet an den ersten Knoten dieser Liste eine *Unlock*-Nachricht und *schedule*d die Timeout-Funktion. Die Daten des Knotens werden in die „Timeout“-Liste eingetragen.

Empfängt ein ge*lock*ter Knoten diese Nachricht von dem Knoten, der ihn ge*lock*t hat, ändert er seinen Zustand durch setzen der entsprechenden Variable in „frei“ und speichert die Kennung des Knotens, der ihn ge*lock*t hatte, ab. Auf diese Weise kann im Falle des Verlustes der nun gesendeten Bestätigungsnachricht (*Unlock/ACK*), entschieden werden, ob eine erneute Beantwortung einer *Unlock*-Nachricht korrekt ist.

Auf dem Knoten, welcher die *Unlock*-Nachricht gesandt hat, aktiviert eine eintreffende *Unlock/ACK*-Nachricht die Empfangsfunktion. Diese löscht den Knoten aus

der Liste ge*lock*ter Knoten und ruft die Sendefunktion erneut auf.

Trifft keine Nachricht ein, wird die Timeout-Funktion aktiviert. Diese Funktion arbeitet, wie oben beschrieben. Zuerst entnimmt sie der Timeout-Liste den ersten Eintrag. Anschließend wird geprüft, ob die *Unlock*-Phase noch aktiv ist. Wenn nicht, beendet sich die Funktion. Sind weitere Einträge in der Timeout-Liste, beendet sie sich ebenfalls. Treffen diese Punkte nicht zu, ist wirklich ein Timeout eingetreten und die Sendefunktion wird erneut gestartet.

Hat die Empfangsfunktion den letzten Eintrag aus der Knotenliste entfernt, beendet sie das *Lock*-Protokoll – die atomare Entscheidung ist getroffen.

5 Resultate

5.1 Experimentelle Bestimmung des Einflussbereiches eines WLAN-Knotens

In diesem Abschnitt soll die Frage geklärt werden, ob die theoretisch beschriebenen Effekte (Abschnitt 2.1.4) existieren. Wenn der Carrier-Sense-Bereich nachgewiesen werden kann, soll außerdem bestimmt werden, wie groß dieser ist.

Der erste Abschnitt erläutert die zugrundeliegenden Ideen, wie der Carrier-Sense-Bereich bestimmt werden kann. Im zweiten Abschnitt wird der Versuchsaufbau beschrieben, mit welchem die Messungen auf realer Hardware durchgeführt werden. Desweiteren werden die Simulationen, welche unter vergleichbaren Bedingungen im NS2 durchgeführt wurden, beschrieben.

5.1.1 Ansatz

Der *Carrier-Sense*-Bereich ist nicht direkt messbar, da keine auswertbaren Daten empfangen werden, also die höheren Schichten des Netzwerkstacks nicht erreicht werden. Die Messung muss deshalb mit einem indirekten Ansatz erfolgen, d.h. die zur Verfügung stehende Bandbreite wird ermittelt. Gibt es einen Carrier-Sense-bereich, wird in diesem die Bandbreite zwischen Stationen aufgeteilt.

Zwei Station werden dazu nahe beieinander (1-2 Meter) positioniert. Auf diese Weise sind ihre Sende-/Empfangs- und Carrier-Sense-Bereiche nahezu identisch. Eine Station sendet Testpakete, welche von der anderen empfangen werden. Paketgröße und Sendeperiode werden dabei so gewählt, dass das Medium ausgelastet (gesättigt) ist, aber noch keine Paketverluste (*Drops*) in der Sende-Queue auftreten.

Sendet nun mehr als eine Station in dem gemeinsam genutzten Bereich, kommt es zu *Contention*, zum Wettbewerb um den Medienzugriff. Versuchen zwei Stationen mit einer Rate zu senden, welche das Medium auslastet, so können sie nur die Hälfte ihrer Pakete senden. Pakete, die gesendet werden sollen, während der andere Sender den Medienzugriff erhalten hat, werden in einer Warteschlange gespeichert, bis der Zugriff auf das Medium wieder „gewonnen“ ist. Sind zu viele Pakete in der Warteschlange, werden weitere verworfen. Damit dieser Effekt schneller messbar ist, wird er durch eine kleine Sende-Queue provoziert. Auf diese Weise lassen sich genauere Aussagen über das Sendeverhalten treffen ohne eine zum Ende der Messzeit womöglich noch gefüllte Queue berechnen zu müssen.

Wird ein zweites Sender-Empfänger-Paar ebenfalls auf Auslastung des Mediums eingestellt, kann man den Abstand der beiden Paare variieren und aufgrund der unterschiedlichen Bandbreitennutzungen die verschiedenen Bereiche ermitteln.

5.1.2 Versuchsaufbau

Dieser Abschnitt beschreibt die Versuchsanordnung und die Vorgehensweise, welche angewandt wurde. Zuerst werden die Details beschrieben, welche sowohl die Simulation als auch die realen Messungen betreffen. Anschließend werden die Besonderheiten der Simulation und der realen Messung in eigenen Abschnitten betrachtete.

5.1.3 Allgemein

Jeweils zwei Knoten werden zu einem Paar zusammengefasst. Ein Sender überträgt Pakete mit maximal möglicher Bandbreite an den Empfänger. Zwei Sender-Empfänger-Paare werden in unterschiedlichen Abständen zueinander positioniert und die Übertragungen gestartet.

Es wird versucht, die Grenzen zwischen drei verschiedenen Bereichen zu finden. Im ersten Bereich sind die Knoten so nahe beieinander positioniert, dass alle vier Stationen untereinander Daten austauschen können. (Abbildung 5.1) Dabei werden sowohl eigene, als auch „fremde“, also Pakete des anderen Sender-Empfänger-Paares, empfangen.

Vergrößert man den Abstand zwischen den beiden Paaren, velassen sie irgendwann den Sende-/Empfangs-Bereich (Abbildung 5.2). Das Signal des anderen Paares ist in diesem *Carrier-Sense-Bereich* immer noch vorhanden und kann von dem *CSMA/CA*-Mechanismus erkannt werden.

Diese zwei Knotengruppen können also untereinander nicht kommunizieren, beeinflussen sich aber trotzdem. Die gewollte Auswirkung ist die Vermeidung von Kollisionen, welche durch Pakete von Stationen in diesem Grenzbereich hervorgerufen werden. Das Problem ist die Reduzierung der Bandbreite, ohne dass beide Stationspaare sich gegenseitig über ihre Anwesenheit informieren könnten.

Im *Carrier-Sense-Bereich* des jeweils anderen Paares werden keine „fremden“ Pakete mehr empfangen. Das Medium lässt sich aber auch nicht voll auslasten. Obwohl die Knotenpaare scheinbar „alleine“ sind, werden Pakete in der Sende-Warteschlange ge*drop*t. Die zur Verfügung stehende Bandbreite wird ähnlich wie im Sende-/-Empfangs-Bereich zwischen beiden Paaren aufgeteilt.

Entfernen sich die Paare weiter voneinander, verlassen sie den *Carrier-Sense-Bereich* des jeweils anderen Paares. Eine gegenseitige Beeinflussung findet nicht mehr statt und beide Paare können die Bandbreite allein für sich nutzen. Die Verluste durch *Drops* sinken wieder auf Null.

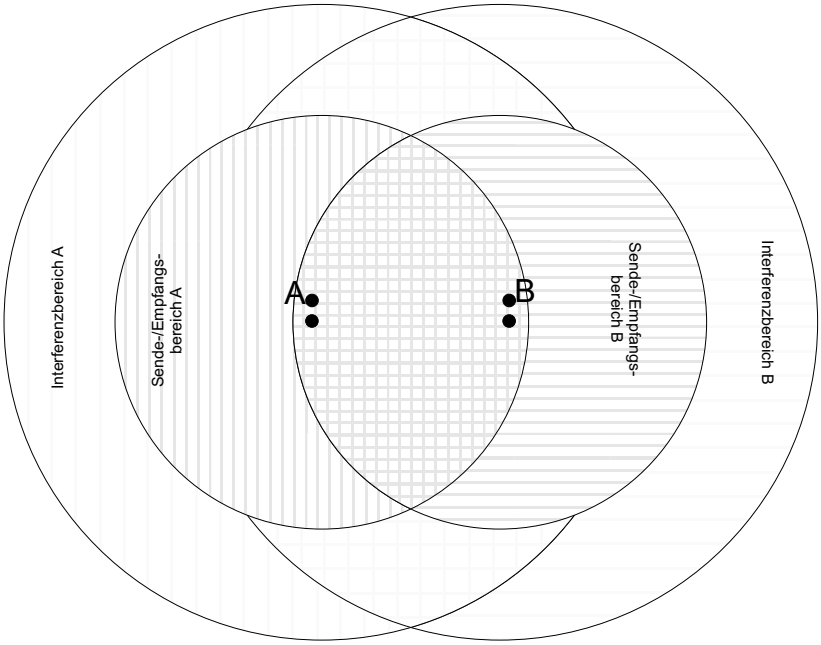


Abbildung 5.1: Zwei Knotengruppen in Empfangsreichweite.

Die zur Durchführung der Messung benutzte Software ist identisch mit der in der Simulation verwendeten, da sie die „GEA“-Schnittstelle [7] nutzt. Zur Vorbereitung der Messung mit zwei Sender-Empfänger-Paaren wurde eine Senderaten/ Paketgrößen-Kombination ermittelt, welche das Medium auslastet ohne das es zu *Drops* in der Sende-Queue kommt.

5.1.4 Simulation - NS2

Der Simulator NS2 unterstützt die Simulation verschiedener Umgebungsbedingungen, wie z.B. die Simulation zufälliger Paketverluste. Dadurch werden äußere Störungen, wie z.B. Maschinen, welche Störstrahlungen in dem benutzten Frequenzband aussenden, simuliert. Für diesen Versuch wurde diese Funktion allerdings deaktiviert, d.h. Verluste auf dem Übertragungsweg werden vernachlässigt, um ein unbeeinflusstes Bild der Paketverluste, welche wegen einer Überlastung des Mediums auftreten, zu erhalten.

Die Signalausbreitung wurde zuerst mit dem „*Two-Way-Ground*“-Modell simuliert. Dieses einfache, schnell berechenbare Modell simuliert die Signalausbreitung mit dem Modell von zwei Strahlen. Der erste verbindet die Quelle direkt mit dem Ziel, der zweite ist auf den Boden gerichtet, wird reflektiert und trifft dann auf das

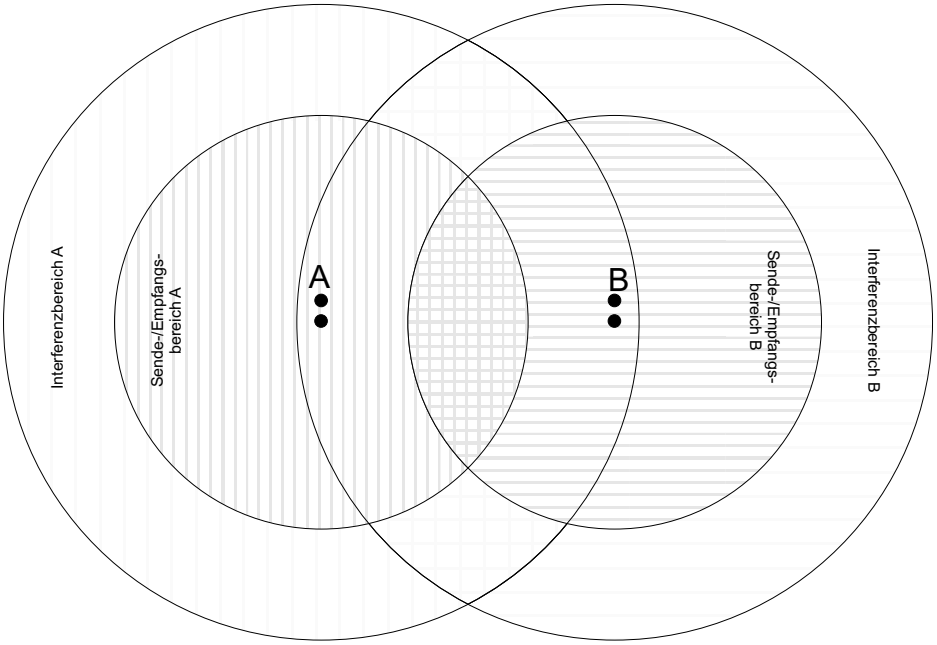


Abbildung 5.2: Zwei Gruppen mit je zwei Knoten befinden gegenseitig außerhalb der Empfangsreichweite, sind aber im gegenseitigen Carrier-Sense-Bereich

Ziel. Die als erstes durchgeführte Simulation wurde mit diesem Modell durchgeführt.

Das zweite Signalausbreitungsmodell wird „*Shadowing*"-Modell genannt. Es erlaubt die Einstellung zusätzlicher Parameter, etwa zur Simulation der exponentiellen Abnahme der Signalstärke. Dieses rechenintensivere Modell kommt den realen Eigenschaften einer WLAN-Übertragung näher. Nach den realen Messungen wurden die dort erhaltenen Parameter genutzt, um das Shadowing-Modell an die realen Karten anzupassen und damit die weiteren Messungen durchzuführen.

Die Pakete werden per Broadcast versandt. Der NS2 benutzt hierbei die in IEEE 802.11b spezifizierte maximale Übertragungsrate von $2\frac{Mbit}{s}$. Mit diesen Einstellungen wurde ermittelt, dass 833*byte* große Pakete (plus Header) mit einer Sendeperiode von 4ms das Medium auslasten. Dies entspricht etwa $208\frac{kByte}{s}$ oder rund $1,67\frac{Mbit}{s}$. Zwischen den beiden Sender-Empfänger-Paaren wurde der Abstand bei jeweils 100, 300 und 600 Metern bzw. um die bekannten Schwellenpunkte bei 250 und 550 Metern gewählt. Während der simulierten 200 Sekunden blieben die Knoten stationär.

5.1.5 Messung

Zur Durchführung des Versuchs wurden vier Knoten genutzt. Jeweils zwei dieser Knoten übernahmen die Funktion des Senders, die anderen zwei empfingen Daten. Zur Simulation wurde der Netzwerksimulator NS2 [1] genutzt. Die real genutzte Hardware bestand aus vier tragbaren Computern - „Dell Latitude D410“, Intel Pentium M Prozessor $1,6Ghz$, $1GB$ Arbeitsspeicher.

Für die Messungen wurden der eingebaute Chipsatz, ein Intel „PRO/Wireless 2915abg“ und zur Verifizierung eine Prism54-Karte (mit Prism54-Chipsatz) genutzt.

Im Vergleich zu den Simulationen mit dem *NS2* waren einige Modifikationen der Parameter der genutzten Betriebssystemumgebung (ein Standard Debian GNU Linux) notwendig, um vergleichbare Voraussetzungen wie in der Simulation zu schaffen. Die Messungen wurden im Freien durchgeführt, um Dämpfungen und Reflektion durch Wände in Gebäuden zu vermeiden und so zur Simulation vergleichbare Ergebnisse zu erhalten.

So senden die Intel „PRO/Wireless“-Karten im 802.11b/g-Protokoll-Setup im Ad-Hoc-Modus mit bis zu $24\frac{MBit}{s}$. Die Hardware musste auf den 802.11b-Modus festgelegt werden und darüber hinaus die Sende-Warteschlange verkleinert werden. Pakete werden dadurch wie gewünscht verworfen, wenn ein Versand nicht sofort möglich ist.

Die im Simulator ermittelten Parameter konnten nicht übernommen werden, da das reale Medium mit den Parametern der Simulation übersättigt war. Es wären so also mehr Pakete versandt worden, als auf dem Medium hätten übertragen werden können. Da dies die Ergebnisse schwerer zu deuten, wenn nicht gar unbrauchbar gemacht hätte, wurden die Parameter angepasst. Die Paketgröße wurde auf von 756 byte (plus Header) bei einer Sendeperiode von 4ms reduziert. Dies ergab die erwünschte Auslastung des Mediums.

Eine definitive Erklärung für den Unterschied von 77 byte pro 4 ms konnte nicht gefunden werden. Die Bandbreite ist um etwa $154\frac{kbit}{s}$ reduziert. Unterschiede in der Größe der Header sind sehr unwahrscheinlich, da diese dem Standard IEEE 802.11 folgen. Zusätzliche RTS/CTS-Nachrichten könnten der Grund für die fehlende Bandbreite sein, diese waren allerdings in beiden Fällen aktiv. Geringe Unterschiede können durch verschiedene Implementierungen des Standards hervorgerufen werden. Freiheiten bei der Umsetzung bzw. unterschiedliche Verarbeitungszeiten durch Hardwareunterschiede könnten die verfügbare Sendezeit und damit Bandbreite reduzieren. Die wahrscheinlichste Erklärung ist, dass auf dem Medium Übertragungsfehler aufgetreten sind. Diese werden durch *Retransmissions* korrigiert, welche Bandbreite belegen.

5.2 Ergebnisse der Messungen des Einflussbereiches

In den folgenden Abschnitten werden jeweils die Messergebnisse des Experiments präsentiert. Zuerst die Resultate der Simulation mittels des Netzwerksimulators NS2, dann zum Vergleich die Ergebnisse der realen Messung.

Der dritte Unterpunkt widmet sich der Auswertung der gewonnenen Erkenntnisse und sich daraus ergebender Schlussfolgerungen.

5.2.1 Simulation - NS2

Die Ergebnisse der Simulation im NS2 entsprechen den Erwartungen. Bis zu einer Entfernung von 250 Metern zwischen den Paaren empfangen sie die Pakete des anderen Sender-Empfänger-Paares. Der „faire“ *contention*-basierte Medienzugriffsalgorithmus führt dazu, dass beide nur jeweils die Hälfte ihrer Pakete senden können. Die andere Hälfte wird aus der Sendequeue entfernt, sobald diese gefüllt ist. Diese Pakete werden am Ende als verlorene Pakete gezählt. Alle Pakete, welche nicht von dem Partnerknoten stammen, werden als „Fremdpakete“ gezählt.

Die Zahl der Pakete, die gesendet werden sollen, liegt bei 50000. Eine Entfernung von 100m zwischen den Sender-Empfänger-Paaren ergab 49656 bzw. 49654 gesendete Pakete während 200 Sekunden. Davon wurde etwa die Hälfte (25561 und 25856 Pakete) erfolgreich übermittelt. „Fremde“ Pakete wurden dabei 24735 bzw. 25021 empfangen. Alle Distanzen zwischen den Knotenpaaren von einem Meter bis 250 Meter Distanz zwischen den Sender-Empfänger-Paaren wiesen etwa die selben Zahlen auf (mit Abweichungen unter einem Prozent).

Die Abbildung 5.3 spiegelt im ersten Drittel diese Messergebnisse wieder. Der obere Graph zeigt das (theoretische) Maximum. Bei der Hälfte des Maximums liegen etwa gleich auf die erfolgreichen, verlorenen und fremden Pakete.

Bei einer Entfernung von 250 Metern endet der Sende-/Empfangsbereich und die Knotenpaare befinden sich im Carrier-Sense-Bereich des anderen Paares. Hier ist keine Kommunikation zwischen den Sender-Empfänger-Paaren mehr möglich. Die Zahl der empfangenen Fremdpakete sinkt auf null. (das zweite Drittel der Abbildung 5.3)

Die Verluste und damit auch die erfolgreich übertragenen Pakete bleiben nahezu gleich denen im vorherigen Abschnitt. Die Werte bei einer Entfernung zwischen den Knotenpaaren von 300m ergeben 24927 von 49660 bzw. 25064 von 49642 erfolgreich übertragenen Pakete.

Diese Werte weisen darauf hin, dass die Belegung des Mediums in diesem Bereich unverändert hoch ist, das Medium also nach wie vor durch beide Sender-Empfänger-Paare genutzt wird. Da die Zahl der empfangenen „Fremdpakete“ null ist, werden keine dekodierbaren Pakete von dem anderen Sender-Empfänger-Paar mehr empfangen. Eine Verbindung, auf welcher nutzbare Daten übertragen werden können,

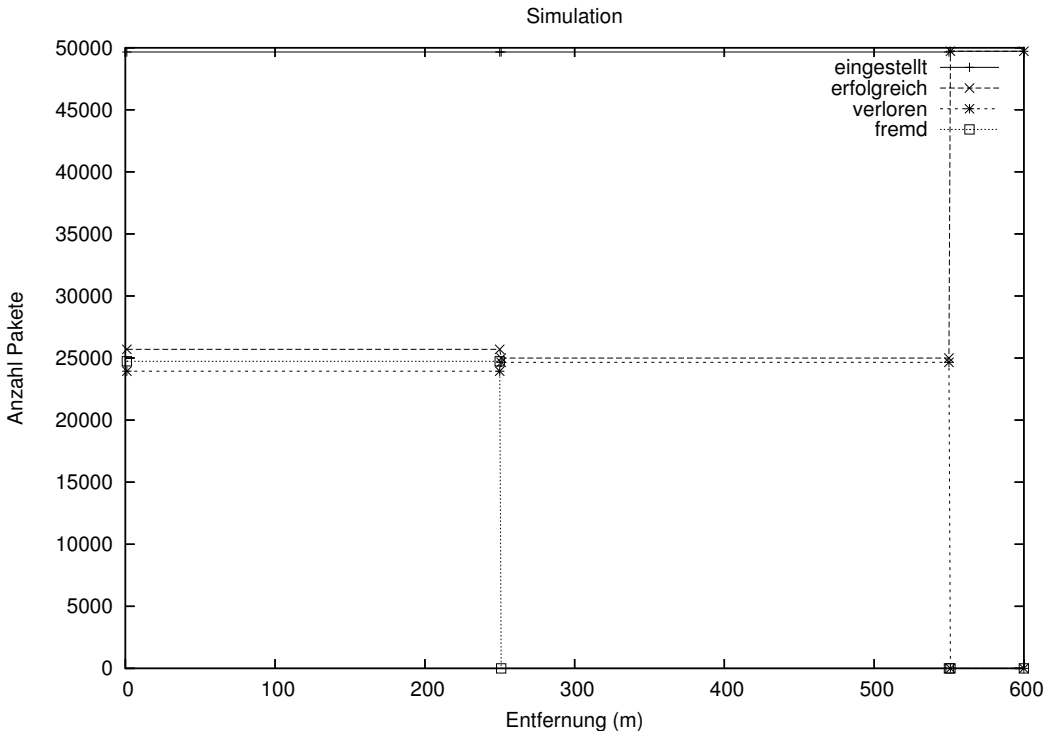


Abbildung 5.3: Ergebnisse des Interferenzexperiments im NS2

existiert also nicht.

Überschreitet man eine Entfernung von 550 Metern, erhöht sich die Zahl der empfangenen Pakete von dem Partnerknoten auf das theoretische Maximum. Von den 49720 Paketen, die beide Sender-Empfänger-Paare aussenden, werden 100% empfangen. Es treten keine Paketverluste auf (die Simulation der zufälliger Paketverluste aufgrund anderer Störungen im Medium war deaktiviert). In Abbildung 5.3 ist erkennbar, dass der Graph der verlorenen Pakete und empfangenen Fremdpakete auf Null liegt und der Graph der empfangenen Pakete sich mit dem der maximal gesendeten deckt.

Die Effekte sind durch Betrachtung der „subjektiven“ Bandbreite noch deutlicher zu erkennen. (Abbildung 5.4) „Subjektiv“ bedeutet in diesem Fall, dass die Summe aller empfangenen Pakete dargestellt wird. Dies entspricht der maximal verfügbaren Bandbreite, welche alle empfangenen Pakete belegen. Bandbreite, welche durch Knoten belegt wird, deren Pakete nicht dekodiert werden können, wird nicht berücksichtigt. Das bedeutet, dass dies der Bandbreite entspricht, die jeder Knoten „subjektiv“ als frei betrachtet.

Auf diese Weise kann, wie eingangs erwähnt, der Carrier-Sense-Bereich nachgewiesen werden. In dieser Abbildung (5.4) stellt sich der oben beschriebene Effekt noch viel stärker dar. Im ersten Drittel ist die subjektive Bandbreite hoch, auf dem Maximum. Sie setzt sich zur Hälfte aus den eigenen Paketen und zur anderen Hälfte aus den vom anderen Knotenpaar gesendeten Paketen zusammen. Beide Paare teilen sich die Bandbreite.

Im zweiten Drittel sinkt die „subjektive Bandbreite“ auf die Hälfte. Das liegt daran, dass beide Knotenpaare sich die Bandbreite teilen. Es kann also nur die Hälfte der geplanten Pakete gesendet werden. Auf der anderen Seite kann die Hälfte, welche von dem anderen Paar gesendet wird, nicht mehr empfangen werden. Der Knoten sieht, dass nur die Hälfte des Mediums genutzt werden kann.

Im letzten Drittel bietet sich ein ähnliches Bild wie im ersten Drittel. Die Bandbreite wird maximal ausgenutzt, diesmal nicht von zwei Knoten, die sich die Bandbreite teilen, sondern von einem einzigen Knoten.

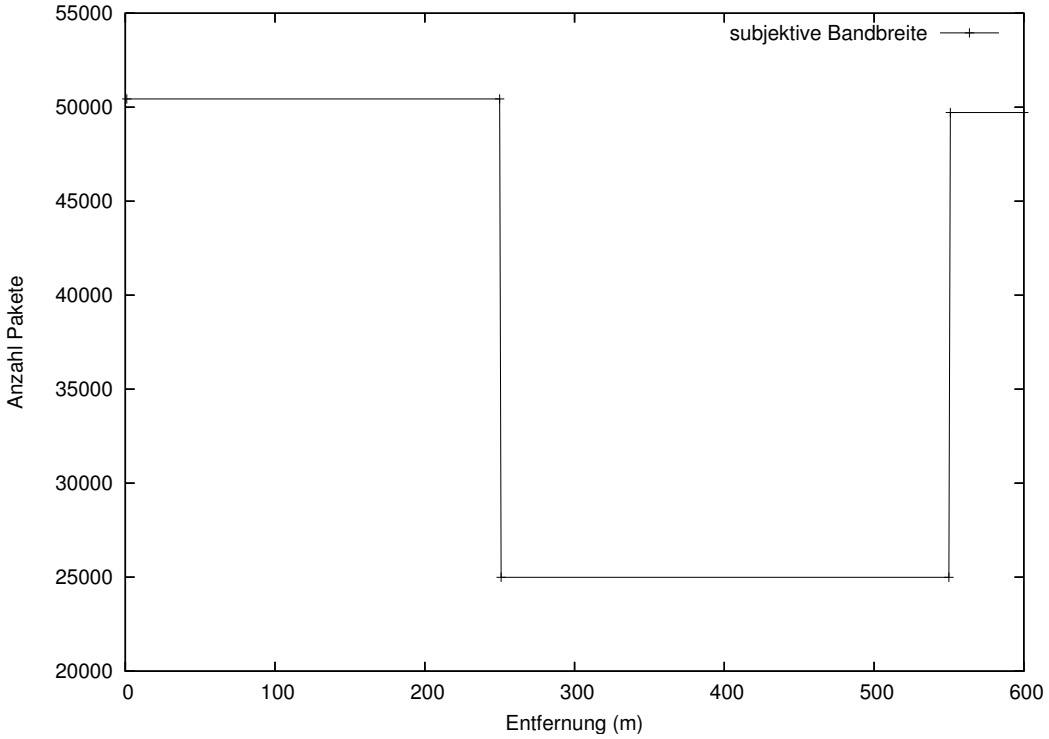


Abbildung 5.4: Summe aller empfangenen Pakete in der NS-2-Simulation

Die Messergebnisse und die daraus entstandenen Diagramme zeigen ungewöhnlich deutliche Grenzen. Der Grund hierfür liegt in dem verwendeten „Two-Way-

Ground“-Modell (siehe 5.1.4), welches eine sehr einfache Annahme nutzt. Gerade deshalb sind die Effekte, welche das weit empfangbare Carrier-Sense-Signal hervorruft, gut sichtbar. Die folgenden Messungen bestätigen diese Effekte auch in einer realen Umgebung, mit geminderten Auswirkungen und weniger deutlichen Grenzen. Für die weiteren Simulationen wurde deshalb das realistischere „Shadowing“-Modell genutzt.

5.2.2 Messung

Die zwei folgenden Abschnitte erläutern die Ergebnisse der vorher erläuterten Messungen.

5.2.3 Messungen im Freien

Abbildung 5.5 zeigt die gemittelten Werte beider Messpaare. Hier kann man bereits einen ähnlichen Verlauf wie in Abbildung 5.3 der Simulationsergebnisse erkennen. Die Messungen bestätigten also die Ergebnisse der Simulation, dass das Carrier-Sense-Signal außerhalb des Sende-/Empfangsbereiches messbar ist und ausgewertet wird.

Während im ersten Drittel die Anzahl der erfolgreich gesendeten Pakete beider Sender-Empfänger-Paare bei etwa der Hälfte des praktisch ermittelten Maximums liegt – die Bandbreite also aufgeteilt wird – erhöht sich im letzten Stück die Anzahl der Pakete auf das praktische Maximum. Innerhalb des Empfangsbereich des anderen Sender-Empfänger-Paares, also im ersten Drittel, werden alle Pakete des anderen Paares empfangen. Die Anzahl der „Fremdpakete“ ist also in etwa gleich der Zahl der eigenen empfangenen Pakete. Im mittleren Bereich sinkt jedoch die Anzahl der Fremdpakete nahe Null, während die empfangenen Pakete nicht das Maximum erreichen.

In der Grafik 5.6 ist die Summe der empfangenen Pakete, sowohl eigener als auch fremder abgebildet. Deutlich ist zu erkennen, dass im mittleren Bereich die ansonsten gerade Kurve „einbricht“. Die Bandbreite in diesem Entfernungsbereich scheint also niedriger zu sein als bei den übrigen Entfernungen. Die Annahme eines Carrier-Sense-Bereiches, wie oben beschrieben, erklärt die scheinbare Abnahme: Obwohl von dem anderen Sender-Empfänger-Paar keine verwertbaren Informationen empfangen werden, ist die Übertragungen immer noch wahrnehmbar. Bis zu einer Entfernung von etwa 190 Metern wird die Bandbreite also immer noch aufgeteilt. Tabelle 5.1 zeigt die während der Messungen ermittelten Zahlen zum direkten Vergleich.

Im Nahbereich, mit einem Abstand zwischen den Paaren von etwa 1,5 Metern, zeigt sich das aus dem Simulator erwartete Ergebnis: Beide Sender-Empfänger-Paare teilen sich die zur Verfügung stehende Bandbreite. Etwa die Hälfte der Pakete des jeweiligen Sendepartners gehen dabei „verloren“, weil es zum Überlaufen der Sende-

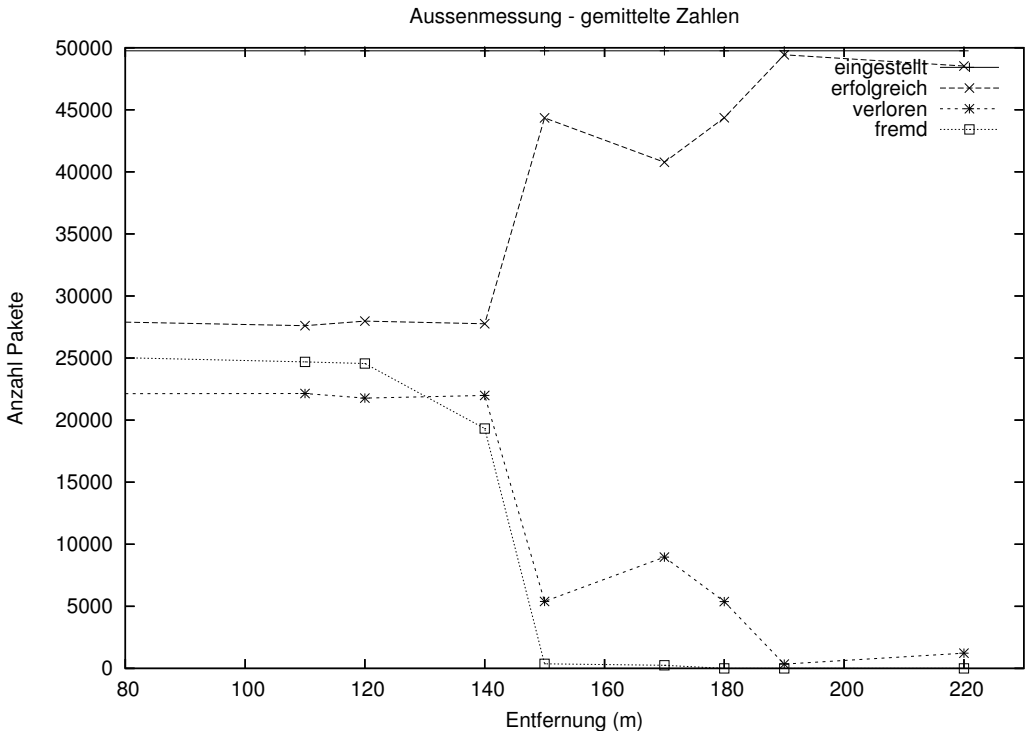


Abbildung 5.5: Gemittelte Messergebnisse beider Knotenpaare bei der Messung im Freien

queue kommt. Die angegebenen Verluste beziehen sich also zum Großteil auf *Drops* in der Queue des Senders wegen Überbelegung des Mediums.

Dazwischen, im Bereich von 150 und 180 Metern tritt der Carrier-Sense-Effekt auf: Die Zahl der von dem fremden Sender empfangenen Pakete geht stark zurück, die Zahl der erfolgreich übermittelten eigenen Pakete steigt aber nicht in gleichem Maße an. Das Medium wird also noch als belegt erkannt und erlaubt nur die zwischen den beiden Knotenpaaren geteilte Nutzung der Bandbreite.

Verlässt man bei etwa 190 Metern Distanz den Bereich des Carrier-Sensing, steigt auch die Rate der real gesendeten Pakete auf den Wert der vorher ermittelten Sättigung des Netzwerkes. Offensichtlich kann das Medium nun uneingeschränkt von einem einzelnen Sender-Empfänger-Paar genutzt werden und das andere Paar ist so weit entfernt, dass es keinen Einfluss mehr hat. Alle Pakete, welche versandt werden sollen, können auch versandt werden und werden empfangen. Die etwa $1 - 2\%$ Verluste sind auf andere Störungen zurückzuführen und sind reale Verluste auf dem Medium, also keine *Drops* in der Sende-Warteschlange.

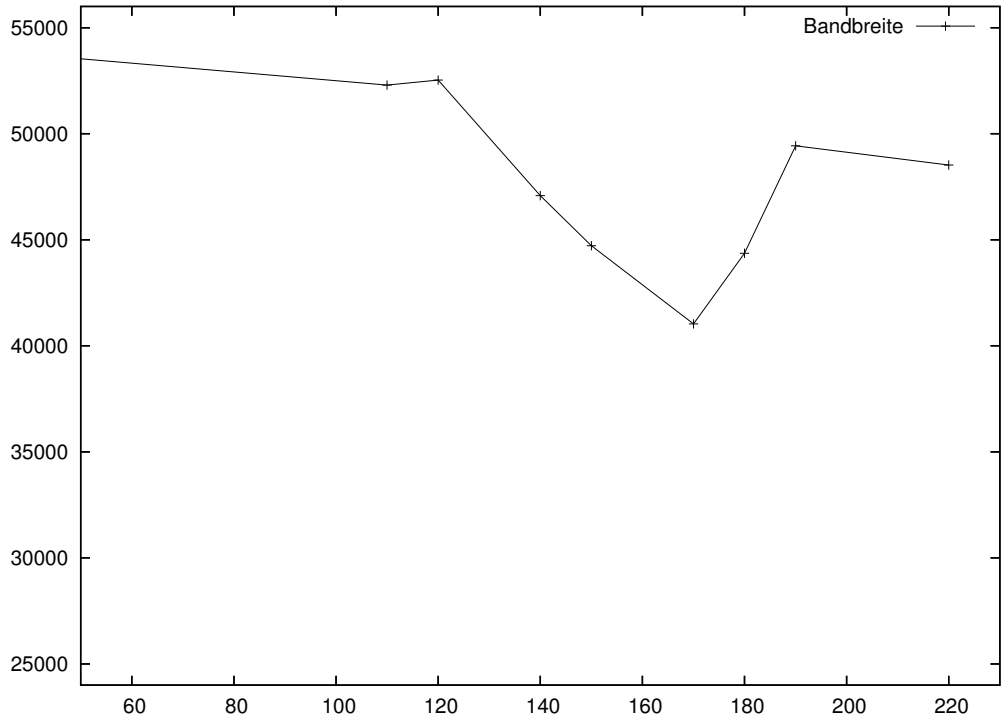


Abbildung 5.6: Benutzte Bandbreite anhand der empfangenen Pakete - subjektive Sicht der Knoten

5.2.4 Messungen im Gebäude

Die Ergebnisse der Messungen im Gebäude sind schwerer zu deuten, als die im Freien durchgeführten. Auch hier hat sich gezeigt, dass in bestimmten Bereichen das Medium belegt ist, obwohl keine bzw. weniger fremde Pakete empfangen werden.

Die verschiedenen Positionen der Sender-Empfänger-Paare sind auf Abbildung 5.7 zu erkennen.

Die Messungen wurden an verschiedenen Standorten durchgeführt. Es wurde versucht, einen Standort so zu wählen, dass eine freie Kommunikation möglich war und jeweils das zweite Sender-Empfänger-Paar zu bewegen. Dabei wurde versucht, den räumlichen Abstand so lange zu vergrößern, bis der Interferenzeffekt auftritt bzw. keine Kommunikation mehr möglich ist.

Das folgende Diagramm stellt die „subjektive“ Bandbreite jedes Knotens bei den Messungen dar. Die Abbildung 5.8 zeigt die festgestellte Bandbreite aller Messstationen, Tabelle 5.2 die korrespondierenden Positionen der Messstationen im Plan in

Knoten Eins

Entfernung	eingestellt	richtig empfangen	verloren	fremd
1,5	49737	29560	22177	25821
110	49747	27874	21873	24706
120	49747	28001	21746	24298
140	49747	26647	23097	21110
150	49747	40055	9692	730
170	49747	35492	14252	358
180	49747	39166	10581	0
190	49747	49411	396	0
220	49747	47919	1828	0

Knoten Zwei

Entfernung	eingestellt	richtig empfangen	verloren	fremd
1,5	49747	27739	22008	25964
110	49747	27342	22405	24669
120	49747	27947	21800	24688
140	49747	28865	20882	17536
150	49747	48643	1104	7
170	49747	46065	3682	145
180	49747	49559	188	0
190	49747	49451	296	0
220	49747	49123	624	0

Tabelle 5.1: detaillierte Ergebnisse der Messung im Freien

Abbildung 5.7.

Auffällig ist die Asymmetrie der Messungen. Die Werte an Position fünf und sechs im Diagramm 5.8 entstanden während derselben Messung, jedoch von unterschiedlichen Knotenpaaren. Verglichen mit der Simulation und auch den Ergebnissen der Messungen im Freien, wurden ähnliche Ergebnisse bei beiden Sender-Empfänger-Paaren erwartet, hier weichen sie allerdings deutlich voneinander ab. Die wahrscheinlichste Erklärung sind unterschiedliche Ausbreitungswege der Signale, Dämpfungen oder Reflexionen durch Wände, welche jedoch mit der verfügbaren „Standard“-Hardware nicht genauer zu bestimmen sind.

Ein deutlicher Abfall der Bandbreite ist im Diagramm 5.8 an den Stellen 5 und 17 zu erkennen. Diese Punkte stehen für Messpunkte an den Stellen 7 und 0 bzw. 4 und 1.

Im ersten Fall liegt das Knotenpaar scheinbar am Übergang zwischen Empfangs- und Carrier-Sense-Bereich, dafür sprechen die hohe Zahl empfangener Pakete des anderen Sender-Empfänger-Paares. Beim Versuch, den Abstand zu vergrößern (Mes-

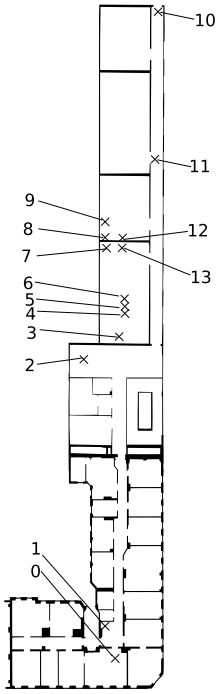


Abbildung 5.7: Positionen der Stationen während der verschiedenen Messungen

spunkte 8, 9 und 12) wurde allerdings die Dämpfung durch die dazwischen liegende Wand so hoch, dass keinerlei messbare Beeinflussung mehr festgestellt wurde. Ebenfalls eine Probemessung, welche die Messposition 0 verlagerte, führte durch zwischenliegende Wände zu ähnlichen Ergebnissen.

Bessere Ausgangsbedingungen schaffte eine Verlagerung von Position 0 zu Position 1. Zwar traten verstärkt asymmetrische Ergebnisse auf, allerdings ließ sich an den Positionen 4, 5 und 6 bei einem Knoten ein deutlicher Übergang zwischen Empfangs- und Carrier-Sense-Bereich feststellen (starke Abnahme der Fremdpakete bei fast konstanten Empfangsraten des eigenen Partners, dann langsames Zunehmen der erfolgreich übertragenen eigenen Pakete). Keinerlei messbare Beeinflussung war schließlich bei Punkt 13 erreicht.

5.2.5 Schlussfolgerungen aus den durchgeführten Messungen

Die Messungen mit realer Hardware haben bestätigt, dass der Carrier-Sense-Bereich existiert. In weiterführenden Betrachtungen zur Bandbreitenkontrolle und -steuerung

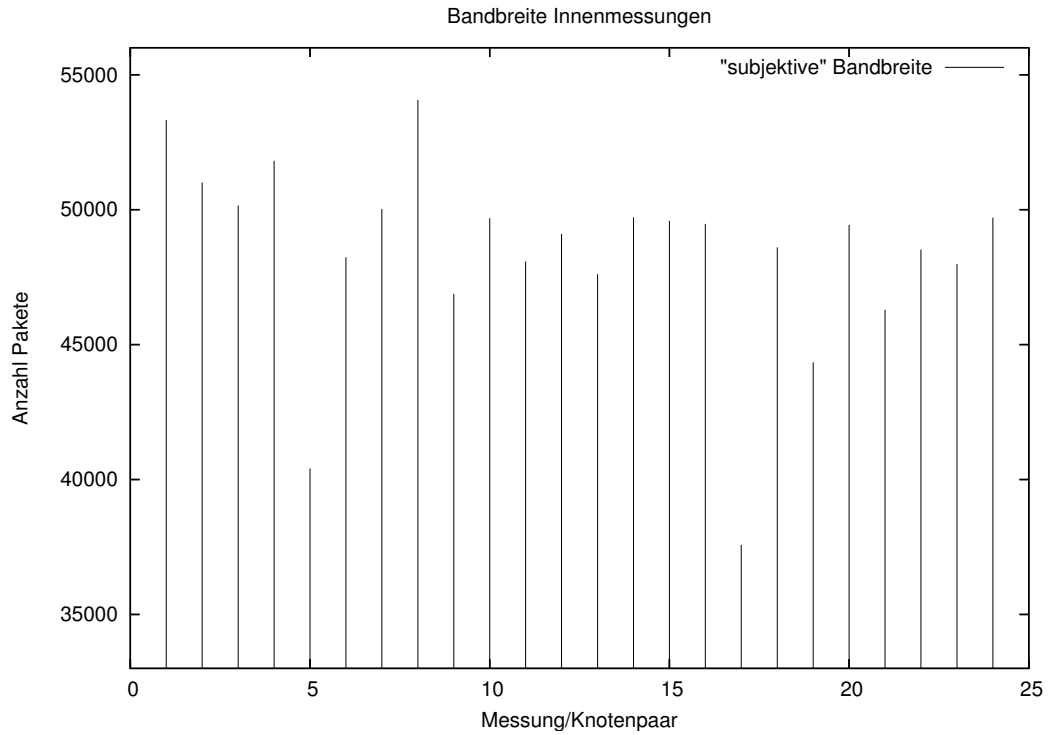


Abbildung 5.8: Benutzte Bandbreite anhand der empfangenen Pakete - „subjektive“ Sicht der Knoten - Legende in Tabelle 5.2

muss dieser Bereich deshalb unbedingt beachtet werden. Ohne ist es unmöglich, die verfügbare Bandbreite zu bestimmen und Zusagen diesbezüglich zu machen.

Die ermittelten Werte in Gebäuden sind, gerade in den Extrembereichen, oft asymmetrisch. Die Effekte des Carrier-Sense-Bereichs konnten trotzdem nachgewiesen werden.

Allerdings haben die Messungen auch gezeigt, dass der im Simulator angenommene Bereich deutlich größer als der real gemessene ist. Unterschiede bei verschiedenen Sendeleistungen von Hardware sind auch möglich. Sicher kann allerdings gesagt werden, dass der Carrier-Sense-Bereich innerhalb von bis zu zwei Hops um einen Knoten herum liegt.

X-Achse	1	2	3	4	5	6	7	8	9	10
Knotenrelation	2-0	0-2	3-0	0-3	7-0	0-7	13-0	0-13	12-0	0-12
X-Achse	11	12	13	14	15	16	17	18	19	20
Knotenrelation	8-0	0-8	9-0	0-9	11-0	0-11	4-1	1-4	5-1	1-5
X-Achse	21	22	23	24						
Knotenrelation	6-1	1-6	13-1	1-13						

Tabelle 5.2: Legende: Relation der Spalten des Diagramms in Abbildung 5.8 zu den Messstationen im Plan in Abbildung 5.7

Paramneter	*Wert*
Feldgröße	Höhe:10m, Breite:1000m
Datenrate des WLAN	$11\frac{MBit}{s}$ (Broadcasts: $2\frac{MBit}{s}$)
Propagationsmodell	Shadowing
Sende-/Empfangsbereich	bis 250m
Carrier-Sense-Bereich	bis 500m

Tabelle 5.3: Parameter des Simulators

5.3 Evaluierung

Die Messungen im folgenden Abschnitt dienen der Evaluierung der Implementierung. Die drei Module Nachbarschaftsexploration, Traffic-Shaping und Admission-Control werden nacheinander analysiert.

5.3.1 Nachbarschaftsexploration

Die Erforschung der Umgebung liefert entscheidende Informationen für andere Bestandteile des Protokolls, die 2-Hop-Nachbarschaft. Wegen der Größe des Carrier-Sense-Bereichs (siehe Abschnitt 2.1.4) müssen mehr als die Knoten in der direkten Nachbarschaft bekannt sein. Dieser Abschnitt beschreibt die Evaluierung der Funktionsfähigkeit durch Simulation.

Versuchsaufbau

Zur Beurteilung der Korrektheit der Nachbarschaftsexploration wurden im Simulator *ns-2* zwei Simulationen durchgeführt. Die Parameter des Simulators enthält Tabelle 5.3.

Da die Konnektivitäten in einem komplexen Netz schwer nachvollziehbar sind, wurde zu Anfang ein einfaches Beispiel herangezogen. Fünf Knoten wurden gleichmäßig auf einer 1000m langen Geraden aufgereiht, so dass der Abstand zwischen

zwei benachbarten Knoten 200m ist. Der übernächste Knoten befindet sich also in 400m Entfernung, außerhalb des Sende-/Empfangsbereichs, aber noch innerhalb des *Carrier-Sense-Bereichs* (siehe Abbildung 5.9). Die Knoten sind nicht mobil, d.h. sie behalten während der Simulation ihre Position bei.

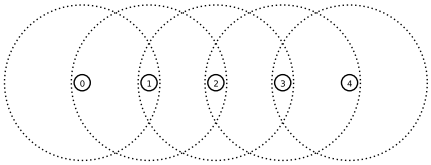


Abbildung 5.9: Fünf Knoten knapp jeweils gerade innerhalb des Sende- bzw. Empfangsbereichs ihres Nachbarn

Die Knoten geben nach dem Senden des periodischen *Beacons* auf die Standardausgabe eine Liste der aktuellen Nachbarschaft aus. Neben der Entfernung (Hops) wird die Kennung des Knotens ausgegeben (*StationID*). Die Form des Tripels ist Folgende: (<Nachbarschaftsebene>:<Kennung>;<nächster Nachbar>)

Auswertung

Nach dem Empfangen des ersten Broadcast-Pakets sind in der Nachbarschaftsliste jedes Knotens jeweils nur die direkten Nachbarn enthalten. Tabelle 5.4 zeigt die Ausgabe des Knotens Nr. 1. In der Nachbarschaftsebene „0“, also in 1-Hop-Entfernung, befinden sich die Knoten mit den Kennungen „0“ und „2“, welche direkt erreicht werden können (kein nächster Nachbar).

(0:00000000;–)
(0:00000002;–)

Tabelle 5.4: Nachbarschaftsliste des Knotens 1 nach dem ersten *Beacon*

1 und alle anderen Knoten können nach dem Empfang des ersten *Beacons* von allen Nachbarn nun diese Informationen ihrer eigenen 1-Hop-Nachbarschaft hinzufügen und diese Information über die periodischen *Beacons* verbreiten.

Nach dem Empfang von so angereicherten Paketen seiner Nachbarn sieht die Liste des Knotens 1 wie in Tabelle 5.5 aus. Zusätzlich zu den beiden Einträgen in der nullten Nachbarschaftsebene ist ein Eintrag in der Ebene „1“, also 2-Hop-Entfernung, hinzugekommen. Knoten 3 wurde von Knoten 2 gefunden, deshalb ist dieser als „nächster Nachbar“ eingetragen. Zur Kommunikation mit 3 muss 1 also über Knoten 2 kommunizieren.

(0:00000000;–)
(0:00000020;–)
(1:00000030;00000020)

Tabelle 5.5: Nachbarschaftsliste des Knotens 1 nach dem zweiten *Beacon*

Werden auch die 2-Hop-Nachbarn verbreitet, können Knoten auch drei Hops entfernte Nachbarn sehen (Tabelle 5.6). In der aktuellen Konfiguration muß die 2-Hop-Umgebung nicht weiter verbreitet werden, da maximal die eigene 2-Hop-Umgebung, also die 1-Hop-Umgebung der Nachbarn wichtig ist.

(0:00000000;–)
(0:00000020;–)
(1:00000030;00000020)
(2:00000040;00000020)

Tabelle 5.6: Nachbarschaftsliste des Knotens 1 nach weiteren *Beacons*

5.3.2 Bandbreitenbegrenzung

Das Traffic Shaping-Modul überwacht die Bandbreitennutzung der Anwendungen eines Knotens bzw. die gesamte Bandbreite, welche der Knoten nutzt. Dieser Abschnitt simuliert verschiedene Szenarien und zeigt dabei die Funktionsfähigkeit der Bandbreitenbegrenzung.

Die wichtigste Aufgabe der Bandbreitenbegrenzung ist die Einhaltung der durch die Zugangskontrolle gewährten Bandbreiten. Eine Anwendung darf maximal so viel Bandbreite nutzen, wie vorher vereinbart. Dieses Verhalten wird geprüft, indem vorsätzlich Applikationen mehr Bandbreite senden als sie zuvor reserviert haben. Das erwartete Verhalten ist, dass andere, korrekt arbeitende Applikationen dadurch nicht beeinflusst werden.

Versuchsaufbau

Der Versuch besteht aus zwei Teilen. Im ersten befinden sich zwei Knoten im Sende-/Empfangsbereich des jeweils anderen, um die lokale Bandbreitenbegrenzung zu evaluieren. Zwei Anwendungen senden auf einem Knoten. Anwendung 1 reserviert $10Byte$ alle $1,5ms$. Anwendung 2 reserviert für $750Byte$ große Pakete, welche alle $4ms$ übertragen werden sollen. Die zweite Anwendung sendet von Beginn an Daten. Anwendung 1 beginnt 40 Sekunden später zu senden. Die maximale genutzte Bandbreite der Datenpakete liegt bei rund $194\frac{kByte}{s}$, beide Ströme werden zugelassen. Anwendung 1 ignoriert allerdings die Reservierung und sendet $400Byte$ große Pakete. Die genutzte

Bandbreite wird gemessen und aufgezeichnet, um in einem Diagramm dargestellt zu werden.

Im zweiten Teil soll die globale Bandbreitenbegrenzung evaluiert werden. Dazu senden zwei Knoten mit jeweils einer Anwendung Daten an einen dritten Knoten. Die Einstellungen der Anwendungen entsprechen dem ersten Versuchsaufbau. Den Knoten wurden die Bandbreiten wie folgt zugeteilt: Der Knoten mit Anwendung 1 wird auf $80\frac{kBit}{s}$ beschränkt, der andere Knoten erhält die übrige freie Bandbreite. Anwendung 1 beginnt nach 40 Sekunden zu senden. Die jeweils genutzten Bandbreiten werden auf gezeichnet und dargestellt.

Auswertung

Ohne Bandbreitenkontrolle kann Anwendung 1 ungehindert große Pakete mit hoher Senderate übertragen. Die Auswirkungen in einer Umgebung ohne Traffic-Shaping zeigt Abbildung 5.10. Nach Aktivierung der ersten Anwendung kann Nr. 2 nicht annähernd die Übertragungsraten, welche voreingestellt sind, halten.

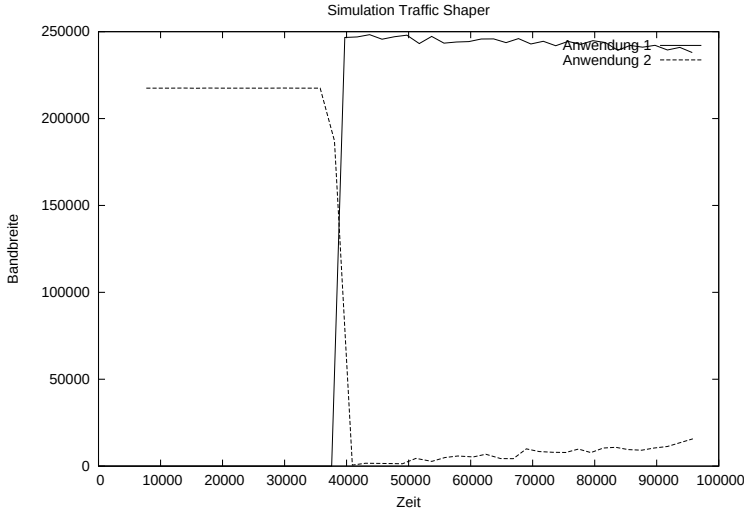


Abbildung 5.10: Applikationen senden ohne Bandbreitenkontrolle

Die Verlustraten sind sehr hoch. Die erste Anwendung kann 48.1% ihrer Pakete nicht übertragen. Die zweite, welche zusätzlich durch eine längere Sendeperiode benachteiligt wird, verliert 61.1% der Pakete.

Abbildung 5.11 zeigt die gleiche Situation mit aktivierter Bandbreitenkontrolle. Anwendung 1 kann nicht mehr senden, als der Kontrollmechanismus gestattet und verliert 82.9% der Pakete bereits in der Warteschlange der Bandbreitenkontrolle.

Die übrigen Pakete werden versandt und nutzen den vorher reservierten Anteil der Ressourcen. Wichtiger ist in diesem Beispiel der Einfluss auf die zweite Anwendung: Bereits an dem Graphen lässt sich ablesen, dass jederzeit die reservierte Bandbreite genutzt werden konnte, so treten auch keine Paketverluste durch *Drops* in den Queues auf.

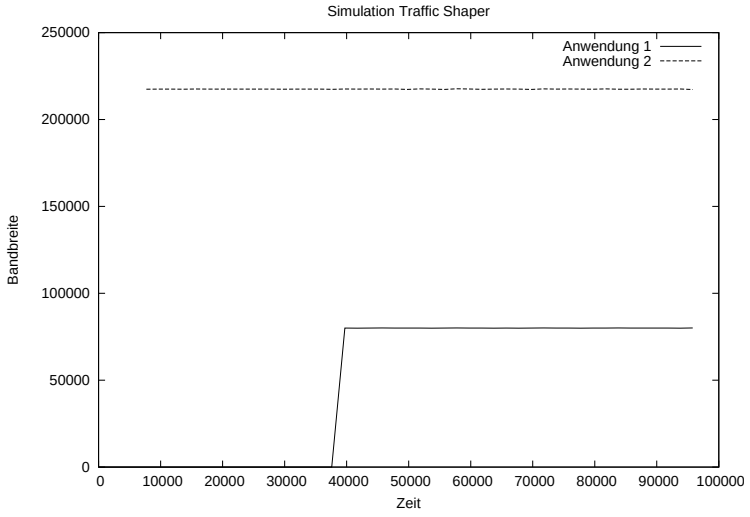


Abbildung 5.11: Falsch konfigurierte Applikationen können zugesicherte Bandbreiten anderer nicht gefährden

In diesem Versuch wurde gezeigt, dass die Bandbreitenkontrolle lokal auf einem Knoten die Anwendungen so kontrollieren kann, dass die zugewiesenen Bandbreiten nicht überschritten werden. Es soll nun weiter untersucht werden, ob auch global die Knoten die ihnen zustehende Bandbreite nicht überschreiten.

Abbildung 5.12 zeigt die Ergebnisse einer Simulation ohne Bandbreitenkontrolle. Anwendung 2 auf Knoten 2 beginnt wiederum zu senden. Nach 40 Sekunden beginnt die erste Anwendung auf Knoten 1 mit ihrer Übertragung. Die Folge ist ein starkes Absinken der Bandbreite, welche Anwendung 2 auf Knoten 2 nutzen kann. Außerdem können $26,1\%$ der Pakete nicht übertragen werden. Aktiviert man die Bandbreitenkontrolle, ergibt sich das Bild in Abbildung 5.13.

Hier ist deutlich zu erkennen, dass Anwendung 1 auf dem ersten Knoten nach 40 Sekunden ebenfalls zu senden beginnt, allerdings nicht die Bandbreite wie in der vorhergehenden Simulation belegen kann. Das vorgesehene Verhalten, welches sich in dieser Simulation nachweisen ließ, ist dass Knoten 1 die zugesicherte Bandbreite nicht überschreiten kann und Knoten 2 unbeeinflusst die zugesicherten Übertragungen durchführen kann. Die Verluste von Knoten 1 steigen zwar von etwa 72% auf $97,8\%$,

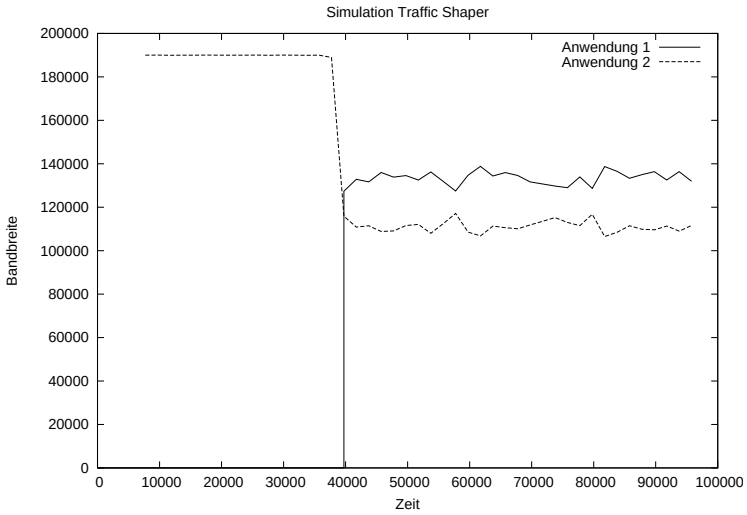


Abbildung 5.12: die Knoten senden ohne Bandbreitenkontrolle

allerdings hält dieser auch nicht die vorgegebenen Grenzen ein. Knoten 2 hat keine Paketverluste, kann also die zugesicherte Bandbreite ungestört nutzen.

5.3.3 Zugriffskontrolle

Die Zugriffskontrolle hat die Aufgabe, Bandbreite zu ermitteln, zu reservieren und Ströme zuzulassen oder abzulehnen. Die Entscheidung ist ein eindeutiges „Ja“ oder „Nein“.

Versuchsaufbau

Um die korrekte Funktionsweise zu evaluieren wurde ein Szenario gewählt, welches Abbildung 5.14 illustriert. Die äußeren Knoten senden Daten an die inneren Knoten. Die Positionen sind dabei so gewählt, dass der Sende-/Empfangsbereich der äußeren Knoten die Inneren überdeckt, die Äußeren untereinander keinen Kontakt haben.

$208\frac{kByte}{s}$ ist die eingestellte, maximale Bandbreite. Zwei der drei Knoten lasten das Medium nahezu aus indem sie Pakete mit $390byte$ alle $4ms$ senden. In der Summe entspricht dies $195\frac{kByte}{s}$.Knoten drei versucht mit verschiedenen Bandbreiten zu senden. Diese sind einmal mit $40Byte$ pro Paket und einmal $80Byte$ alle $4ms$ so gewählt, dass sie einmal zugelassen wird und einmal abgelehnt werden sollte.

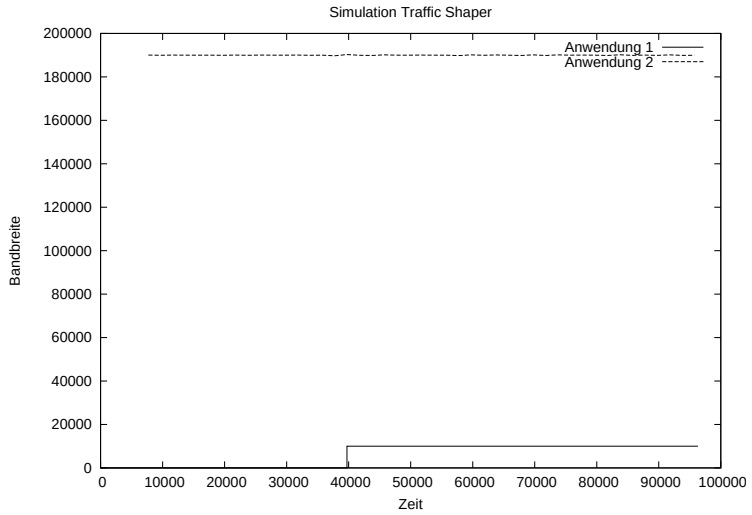


Abbildung 5.13: die Knoten senden mit kontrollierter Bandbreite

Auswertung

Die Simulationen lieferten als Ergebnis einmal zwei ohne Paketverluste übertragenen Datenströme und das zweite mal drei ohne Paketverluste übertragenen Datenströme. Die Überlastung des Mediums trat auf, wenn der dritte Strom mit $80 Byte$ pro $4ms$ senden sollte. Der Admission-Control-Algorithmus ließ in diesem Falle nur zwei Ströme zu.

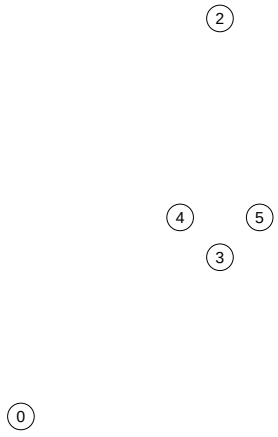


Abbildung 5.14: Topologie zur Evaluierung der Zugriffskontrolle

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Die Evaluierung des Systems hat gezeigt, dass es den Erwartungen entspricht. Das System erlaubt die exakte Berechnung, Zuteilung und Kontrolle von Bandbreiten in drahtlosen Multi-Hop-Netzwerken. Das System kann dabei mit weit verbreiteter „Standard“-Hardware nach dem Standard IEEE 802.11 genutzt werden und unterstützt Dienstgüteanforderungen in Form der Reservierung von Bandbreite.

Zur Erreichung dieses Zieles wurden zuerst thematisch verwandte Arbeiten analysiert, um ein Lösungskonzept zu entwickeln. Das Ergebnis ist ein aus drei Teilen bestehendes System. Der Erste ermöglicht die Kenntnis aller Knoten der Umgebung, welche sich die vorhandene Bandbreite teilen. Der zweite Teil sorgt dafür, dass keine Anwendung die gemachten Reservierungen überschreiten, d.h. die Bandbreite welche genutzt werden soll, wird kontrolliert und begrenzt. Der dritte Teil ist die Zulassungskontrolle (*Admission Control*), welche die Informationen der Nachbarschaftsexploration nutzt, um die Nachbarn zu kontaktieren und mit diesen Informationen eine exakte Berechnung der verfügbaren Bandbreite durchzuführen. Andere Arbeiten wählen „Monitoring“-Ansätze, welche jedoch mit „Standard“-Hardware nicht exakt durchführbar sind. Existierenden Lösungen zur Bandbreitenberechnung widersprechen sich teilweise, was den Einflussbereich eines sendenden Knotens betrifft. Zur Klärung dieser Frage wurden Messungen durchgeführt. Das Ergebnis ist, dass die Bandbreite von Knoten bis einschließlich zwei Hop Entfernung von einem sendenden Knoten verringert wird.

Der Algorithmus zur Erkundung der Nachbarschaft nutzt *Beacon* oder auch *Hello* genannte Nachrichten, um die Anwesenheit eines Knotens unter den direkten Nachbarn bekannt zu machen. Da eine Kenntnis des *Carrier-Sense-Bereichs* der zwei-Hop-Nachbarschaft notwendig ist, um alle Knoten zu kennen, welche die vorhandene Bandbreite beeinflussen, reicht diese Methode jedoch nicht aus. Die *Beacons* beinhalten zusätzlich die Informationen über ihre eigene Nachbarschaft. Ein Empfänger kann auf diese Weise seine eigene Nachbarschaft um die Informationen der zwei Hops entfernten Knoten ergänzen.

Die Bandbreitenkontrolle nutzt einen Traffic-Shaping-Algorithmus, welcher mit mehreren kaskadierten Warteschlangen implementiert ist. Die Parameter dieser Warteschlangen werden an die maximale Bandbreite einer Applikation bzw. eines Knotens angepasst und stellen so sicher, dass eine Nutzung von Bandbreite nur innerhalb

der zugesicherten Grenzen stattfindet.

Die Parameter für die Bandbreitenkontrolle ergeben sich aus der Berechnung der Bandbreite, welche eine Applikation benötigt. Die Zulassungskontrolle hat die Aufgabe, die vorhandene Bandbreite zu ermitteln und anhand derer zu entscheiden, ob eine Übertragung zugelassen werden darf. Ist die Entscheidung positiv, wird die Bandbreite auf allen Knoten in der Bandbreitennachbarschaft reserviert und der Bandbreitenkontrolle die entsprechenden Parameter übermittelt.

6.2 Ausblick

Im Laufe der Entwicklung des Bandbreitenkontrollsystems wurden verschiedene Annahmen gemacht. Eine davon betrifft beispielsweise die Festlegung auf Hardware nach dem Standard IEEE 802.11 [12] Dies führt zu Einschränkungen, welche die dadurch festgelegten Besonderheiten betreffen. Sollte sich die Situation ändern und die Verbreitung von Hardware zunehmen, welche den Standard IEEE 802.11e [11] implementiert, könnte dessen Prioritätssystem eventuell in die Bandbreitenkontrolle integriert werden.

Die Bandbreite hängt von der aktuellen Senderate ab. Der Standard sieht vor, dass diese in Abhängigkeit von den Empfangsbedingungen automatisch verändert wird, um die Störanfälligkeit der Übertragung zu minimieren. Es ist jedoch nicht vorgesehen, dass diese Raten gesetzt oder von höheren Protokollschichten ausgelesen werden können. Bei der Berechnung der Bandbreite ist dieser Faktor ein wichtiges, im Moment nicht lösbares Problem.

Im Laufe der Analyse der Aufgabenstellung haben sich viele Einzelprobleme ergeben, welche in der vorliegenden Arbeit gelöst wurden. Darüber hinaus zeigten sich noch Lösungswege auf, welche zusätzliche Funktionalitäten ermöglicht hätten, aber den Rahmen der Diplomarbeit überstiegen.

Die Art und Weise, wie mit *Best-Effort-Traffic* verfahren wird, gehört zu diesen Punkten. Im aktuellen System wird nicht zwischen *QoS-* und *Best-Effort-Traffic* unterschieden. Entweder wird nur *Qos*-Traffic zugelassen, d.h. alle Anwendungen müssen Bandbreite reservieren oder es wird eine Reservierung für alle *Best-Effort*-Anwendungen getätigt. Diese Bandbreite kann dann von allen *Best-Effort*-Anwendungen genutzt werden, d.h. alle Anwendungen, welche keine Bandbreite reserviert haben, werden automatisch in diese Kategorie eingeordnet. Eine weitere Möglichkeit betrifft Reservierungen ohne Zuordnung zu einer Anwendung. Ein Knoten könnte aus diesem Pool Bandbreite zuweisen, wenn eine Anwendung sie benötigt. Auf diese Weise würden langwierige Berechnungen der vorhandenen Bandbreiten nicht von einer Bandbreitenanfrage ausgelöst, sondern könnten zu anderen Zeitpunkten ausgehandelt werden.

Durch das genutzte Framework konnten die einzelnen Teile des Systems modular

programmiert werden. Das bedeutet, sie können einzeln eingesetzt, ausgetauscht oder anderweitig genutzt werden, wenn die Schnittstellen eingehalten werden. So ist es ohne weiteres möglich, dass das System zusammen mit jedem beliebigen Routingalgorithmus eingesetzt wird, der an das genutzte Framework angepasst wird. Diesem würde es konsistente Informationen der Nachbarschaft und Bandbreitenreservierung pro Link anbieten und so *QoS*-Routing ermöglichen.

Literaturverzeichnis

[1] : *The Network Simulator - ns-2.* – URL `http://www.isi.edu/nsnam/ns`

[2] Clausen, T. ; P.Jacquet: *RFC 3626 – Optimized Link State Routing.* October 2003

[3] DIN e.V.: *Deutsches Institut für Normung e.V..* – URL `http://www.din.de/`

[4] Ge, Ying ; Kunz, Thomas ; Lamont, Louise: Quality of Service Routing in Ad-Hoc Networks Using OLSR. In: *Proceedings of 36th Hawaii International Conference on System Sciences*, 2003

[5] Georgiadis, Leonidas ; Jacquet, Philippe ; Mans, Bernard: Bandwidth reservation in multihop wireless networks: complexity and mechanisms. In: *24th International Conference on Distributed Computing Systems Workshops. Proceedings*, 2004, S. 762–767

[6] Gu, Daqing ; Zhang, Jinyun: QoS Enhancement in IEEE802.11 Wireless Local Area Networks. In: *IEEE Communications Magazine* 41, June, Nr. 6, S. 120–124. – URL `http://www.merl.com/reports/docs/TR2003-67.pdf`

[7] Herms, Andre ; Mahrenholz, Daniel: GEA: A Method for Implementing and Testing of Event-driven Protocols. In: *MeshNets 2005.* Visegrad/Budapest, Hungary, July 2005

[8] Hsu, Yu-Ching ; Tsai, Tzu-Chieh ; Lin, Ying-Dar ; Gerla, Mario: Bandwidth routing in multi-hop packet radio environment. In: *Proceedings of the 3rd International Mobile Computing Workshop*, 1997

[9] IEEE: *Institute of Electrical and Electronics Engineers, Inc..* – URL `http://www.ieee.org`

[10] IEEE: *Website des IEEE 802 LAN/MAN Standards Committee.* – URL `http://www.ieee802.org/`

[11] IEEE: *IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–LAN/MAN Specific Requirements–Part*

11 Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications: Medium Access Control (MAC) Quality of Service (QoS) Enhancements. IEEE, 2005

[12] IEEE: *Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.* IEEE, 2005

[13] ISO: *International Organization for Standardization.* – URL `http://www.iso.org/`

[14] Johnson, D. ; Maltz, D.: *Dynamic source routing in ad hoc wireless networks.* 1996. – URL `citeseer.ifi.unizh.ch/johnson96dynamic.html`

[15] Kuo, Yu-Liang ; Wu, Eric H. ; Chen, Gen-Huey: *Admission Control for Differentiated Service in Wireless Mesh Networks.* July 2005

[16] Li, "Jinyang ; Blake, Charles ; De Couto, Douglas S. ; Lee, Hu I. ; Morris, Robert: Capacity of ad hoc wireless networks. In: *Proc. of Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, URL `citeseer.ifi.unizh.ch/li01capacity.html`, 2001, S. 61–69

[17] Lindgren, Anders ; Belding-Royer, Elizabeth M.: Multi-path Admission Control for Mobile Ad hoc Networks. In: *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2005, S. 407–417

[18] Mock, M. ; Frings, R. ; Nett, E. ; Trikaliotis, S.: Clock Synchronization for Wireless Local Area Networks. In: *12th Euromicro Conference on Real-Time Systems.* Stockholm, Germany, 2000

[19] Mock, M. ; Frings, R. ; Nett, E. ; Trikaliotis, S.: Continuous Clock Synchronization in Wireless Real-Time Applications. In: *19th IEEE Symposium on Reliable Distributed Systems (SRDS).* N rnberg, Germany, October 16-18 2000

[20] Nett, Edgar ; Mock, Michael ; Gergeleit, Martin: *Das drahtlose Ethernet – Der IEEE 802.11 Standard: Grundlagen und Anwendungen.* Addison-Wesley, Februar 2001 (Datacom)

[21] Perkins, C.: *Ad-hoc on-demand distance vector routing.* 1997. – URL `citeseer.ifi.unizh.ch/article/perkins02adhoc.html`

[22] Ramanathan, Ram ; Steenstrup, Martha: Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. In: *Mobile Networks and Applications* (1998), Nr. 3, S. 101–119

[23] Tanenbaum, Andrew S.: *Computer Networks*. 4th. Prentice Hall PTR, 2002

[24] Turletti, Thierry: A brief Overview of the GSM Radio Interface / Laboratory for Computer Science, Massachussets Institute of Technology. 1996. – Forschungsbericht

[25] Xue, Qi ; Ganz, Aura: Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks. In: *Journal of Parallel and Distributed Computing* 63 (2003), Nr. 2, S. 154–165. – ISSN 0743-7315

[26] Yang, Yaling ; Kravets, Robin: Contention-Aware Admission Control for Ad Hoc Networks. In: *IEEE Transactions on Mobile Computing* (2005)

Selbständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit „Bandbreitenkontrolle in drahtlosen Multihop-Netzwerken“ selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, sowie alle Zitate entsprechend kenntlich gemacht habe.

Magdeburg, 4.4.2006
Martin Wewior