Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik
Institut für Verteilte Systeme

Diplomarbeit

Routing mit Qualitätsgarantien in mobilen Ad-Hoc-Netzen

Georg Lukas

17. Oktober 2005

Betreuer:
Dipl.-Inform. André Herms

Lukas, Georg:
Routing mit Qualitätsgarantien in mobilen Ad-Hoc-Netzen
Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, 2005.

Danksagung

Während der Entstehung dieser Arbeit haben mich zahlreiche Personen durch fachliche Hilfe, Ratschläge und Motivation unterstützt. An dieser Stelle möchte ich die Gelegenheit nutzen, mich bei ihnen zu bedanken.

Besonderer Dank gilt den Professoren Edgar Nett und Jörg Kaiser für ihre Tätigkeit als Gutachter dieser Diplomarbeit. Ich möchte mich auch bei meinem Betreuer André Herms bedanken, der stets den richtigen Weg aufzeigen konnte und mir in zahlreichen Gesprächen dabei half, diese Arbeit zu dem zu machen, was sie ist.

Mein Dank gilt ebenfalls den Mitarbeitern des Instituts für Verteilte Systeme, die mir Hilfe im Verlauf der Arbeit und eine sehr angenehme Atmosphäre geboten haben.

Nicht unerwähnt bleiben sollen auch meine Eltern, die mir das Studium und eine tiefgründige Beschäftigung mit der Materie ermöglichten, und mir immer Rückhalt gegeben haben.

Allerherzlichst bedanke ich mich bei Tina Pollee für ihre Hilfe beim Schreiben genauso wie für die moralische Unterstützung. Meine Anerkennung gilt auch Stefan Osterburg und Dennis Göbel, die mich mit zahlreichen Anregungen und konstruktiver Kritik in der letzten und wichtigsten Diplomphase begleitet haben.

Schließlich möchte ich mich bei allen Professoren, Mitarbeitern und Kommilitonen an der Fakultät für Informatik der Universität Magdeburg dafür bedanken, dass ich Gelegenheit hatte, von ihnen zu lernen, mit ihnen vieles zu unternehmen und dass mein Studium hier lehrreich, spannend und kurzweilig war.

Inhaltsverzeichnis

Abbildungsverzeichnis

x

Tabellenverzeichnis

Abkürzungsverzeichnis

ACK	Acknowledgement (Bestätigung)
AODV	Ad-Hoc On-Demand Distance Vector
API	Application Programming Interface
ASAP	Adaptive ReServation And Pre-allocation protocol
CEDAR	Core Extraction Distributed Ad hoc Routing protocol
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
CTS	Clear To Send
DSDV	Destination Sequenced Distance Vector protocol
FIFO	First In First Out
FSR	Fisheye State Routing protocol
GEA	Generic Event-based API
GPS	Global Positioning System
IARP	IntrAcluster Routing Protocol
IEEE	Institute of Electrical & Electronics Engineers
IERP	IntErcluster Routing Protocol
IETF	Internet Engineering Task Force
INSIGNIA	In-band Signaling Support for QOS in Mobile Ad Hoc Networks
ISM	Industrial, Scientific and Medical
IZR	Independent Zone Routing protocol
LAN	Local Area Network
LANMAR	Landmark Ad Hoc Routing Protocol
MAC	Media Access Control
MANET	Mobile Ad-Hoc Network
MDS	Minimum Dominating Set
MMWN	hierarchically-organized Multihop Mobile Wireless Networks for Quality of Service support
MPR	Multi-Point Relay
NACK	Negative Acknowledgement (Ablehnung)
NAM	Network Animator
NS	Network Simulator
OLSR	Optimized Link State Routing
POSIX	Portable Operating System Interface (for uniX)

QoS	Quality of Service
RFC	Request For Comment
RTS	Ready To Send
STL	Standard Template Library
TCL	Tool Command Language
TTL	Time To Live
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network
ZRP	Zone Routing Protocol

1 Einleitung

1.1 Motivation

Die drahtlose Übertragung von digitalen Daten ist zum festen Bestandteil des modernen Alltags geworden. Vom Mobilfunk über das WLAN-Surfen bis zur Radioübertragung von Staudaten an Automobil-Navigationssysteme findet man überall Systeme, die mit Hilfe der mobilen Kommunikation den Lebensstandard erhöhen.

Die meisten Systeme erfordern eine vorhandene Infrastruktur in Form von Sendemasten, Access-Points und Rundfunksendern, deren Bereitstellung teuer und zeitaufwändig ist und deren Ausfall zum Zusammenbruch der Kommunikation führt.

In Situationen, in denen man sich nicht auf die vorhandene Infrastruktur verlassen kann oder wenn diese ausgefallen ist, beispielsweise beim Einsatz in Katastrophen- und Krisengebieten, besteht aber genauso ein Kommunikationsbedarf wie in Bereichen, wo keine Kommunikationsleitungen vorhanden sind oder nicht aufgebaut werden können, z.B. in denkmalgeschützten Gebäuden.

Die steigende Verbreitung von mobilen Computern mit drahtloser Konnektivität und deren voranschreitende Miniaturisierung werden es in naher Zukunft erlauben, allein mit solchen Geräten automatisch Netzwerke aufzuspannen und den Benutzern die Kommunikation untereinander oder über Netzwerkgrenzen hinweg zu erlauben.

Gerade in Katastrophensituationen, in denen die Einsatzkräfte mit einheitlichen Geräten zu ihrer Unterstützung ausgestattet sind, ist der Aufbau solcher mobiler Ad-Hoc-Netze deutlich leichter realisierbar und würde die Koordination der Arbeiten signifikant erleichtern.

Für die Verwendung eines solchen Netzwerks für Sprach- oder Video-Übertragung, die eine intuitivere Kommunikation und bessere Übersicht erlauben würde, ist die reine Paketvermittlung allerdings nicht ausreichend. Dort müssen Bandbreiten für Datenströme garantiert werden können und die Garantien eingehalten bzw. das Scheitern schnellstmöglich signalisiert werden.

Um dies zu ermöglichen, müssen die Endgeräte der Helfer in der Lage sein, ein autonomes digitales Netzwerk untereinander aufzuspannen. Dieses Netzwerk wird den Teilnehmern als Grundlage für Datenkommunikation mit Gütegarantien (Quality Of Service) zur Verfügung gestellt. Die Geräte müssen Änderungen in der Konnektivität, die durch veränderte Teilnehmerposition oder Hindernisse bei der Signalausbreitung entstehen, selbständig erkennen und alternative Pfade finden, um die Kommunikation nicht zusammenbrechen zu lassen.

1.2 Aufgabenstellung

Drahtlose Ad-Hoc-Netzwerke erlauben es ihren Teilnehmern nur mit der direkten Nachbarschaft zu kommunizieren. Durch Verwendung geeigneter Routing-Protokolle lässt sich diese Einschränkung überwinden. Es existiert eine Vielzahl von Lösungen hierfür, wobei allerdings die Bereitstellung von Dienstgütegarantien nur ungenügend betrachtet wird.

Das Ziel dieser Diplomarbeit ist die Entwicklung eines Routing-Protokolls für mobile Ad-Hoc-Netzwerke. Dieses Routing-Protokoll soll es den an einem solchen Netz teilnehmenden Knoten ermöglichen, Daten an beliebige andere Teilnehmer zu versenden. Es sollen zwei Klassen von Daten berücksichtigt werden: Best-Effort und Quality of Service (QoS).

Das Best-Effort-Protokoll soll Daten über mehrere Zwischenstationen ausliefern können, wobei keinerlei Garantien bezüglich der Zuverlässigkeit oder der Auslieferungszeit gegeben werden. Das QoS-Routing dient dazu, die Bandbreiten- und Latenzanforderungen der Applikation zu erfüllen. Besteht dazu keine Möglichkeit, soll die Anwendung über das Scheitern informiert werden.

Als Grundlage für das Routing-Protokoll ist das an der Arbeitsgruppe „Echtzeitsysteme und Kommunikation" entwickelte Clustering-Schema zu verwenden. Dieses Verfahren sorgt für eine Gruppierung der Knoten mit gegenseitigem Funkkontakt und regelt den Medienzugriff innerhalb der von ihm gebildeten Cluster.

Das entwickelte Protokoll ist in C++ zu implementieren und mit geeigneten Mitteln zu evaluieren.

1.3 Ergebnisse

Im Rahmen dieser Diplomarbeit wurden Routing-Protokolle für Best-Effort- und QoS-Anforderungen entwickelt. Dazu wurden zunächst vorhandene Lösungen und Konzepte analysiert und miteinander verglichen.

Aus dieser Analyse wurde die Notwendigkeit einer Propagation der Netzwerktopologie ersichtlich. Deswegen wurde in der ersten Phase ein Verfahren entwickelt und implementiert, welches alle Knoten mit Informationen über die Netzwerkstruktur ausstattet und dazu ungenutzte Zeitschlitze des Clustering-Verfahrens verwendet.

Darauf aufbauend wurde ein Routing-Modul entwickelt, das Pakete von lokalen Anwendungen entgegennimmt, entsprechend den Topologiedaten Routen berechnet und die Datenpakete auf diesen Routen an seine Nachbarn verschickt, welche die Pakete weiterleiten.

Dieses Verfahren ohne Zustellungsgarantien wurde dann verwendet, um ein Reservierungsschema für Verbindungen zu entwickeln. Das sich daraus ergebende QoS-Protokoll bietet den Anwendungsprogrammen die Möglichkeit, Anforderungen an

die Bandbreite und die maximale Länge der Pfade zu stellen. Die einzuhaltenden Latenzzeiten können implizit über die Pfadlänge festgelegt werden.

Der Verbindungsbau erfolgt nach einem optimistischen reaktiven Suchverfahren zuerst über die kürzeste Verbindung zum Ziel. Kann auf dieser Strecke die geforderte Bandbreite nicht zugesichert werden, so werden mit einer verteilten Backtracking-Suche nach und nach die nächstlängeren Pfade ausprobiert, bis die Anforderungen erfüllt werden können.

Dieses Reservierungsschema wurde zusammen mit einer Überwachungskomponente für bestehende Verbindungen in Form eines QoS-Moduls implementiert, das Best-Effort-Nachrichten zur Signalisierung verwendet.

Abschließend wurde die Funktionsfähigkeit der Topologie-Propagation, des Best-Effort-Routings und der Verbindungsreservierung mit Garantien in Simulationsexperimenten belegt. Es wurde gezeigt, dass die Übertragung der Topologie-Daten selbst unter hoher Netzwerklast und bei Knotenmobilität funktioniert und dass die Zustellung von Datenpaketen sowohl ohne Garantien (Best-Effort) als auch mit zugesicherten Eigenschaften (QoS) in unterschiedlichen Szenarien gelingt.

1.4 Gliederung

Die vorliegende Diplomarbeit gliedert sich wie Folgt:

Im zweiten Kapitel dieser Arbeit werden die Grundlagen für die Entwicklung eines QoS-Routing-Protokolls vorgestellt. Dazu wird die Funktionsweise mobiler Ad-Hoc-Netzwerke analysiert, existierende Routing- und QoS-Protokolle vorgestellt und schließlich das zu verwendende Clustering-Schema erläutert.

Kapitel 3 stellt die Entwicklung eines Propagationsschemas für Topologiedaten und eines darauf aufbauenden Routing-Verfahrens vor. Das hier entwickelte Routing-Verfahren bietet keine Garantien für die Paketübertragung, wird jedoch als Grundlage für ein Reservierungsschema benötigt.

Im vierten Kapitel wird unter Verwendung des Weltmodells und des Best-Effort-Verfahrens aus Kapitel 3 ein QoS-Routing-Protokoll aufgebaut, welches die Reservierung von Verbindungen erlaubt und die Einhaltung der Garantien überwacht.

Kapitel 5 gibt die Evaluierung der entwickelten Protokolle in einer Simulationsumgebung wieder. Den Abschluss bilden eine Zusammenfassung der Ergebnisse und ein Ausblick in Kapitel 6.

2 Grundlagen und verwandte Arbeiten

Für die Entwicklung eines QoS-Routing-Protokolls für mobile Ad-Hoc-Netzwerke müssen zunächst die Grundlagen beleuchtet werden. Dazu dient dieses Kapitel.

Zuerst wird geklärt, was mobile Ad-Hoc-Netzwerke auf WLAN-Basis sind und wie sie sich von herkömmlichen Rechnernetzen unterscheiden. Danach werden Routing-Protokolle und QoS-Protokolle für mobile Netzwerke vorgestellt, um den Stand der Entwicklung in diesem Bereich aufzuzeigen.

In einem eigenen Abschnitt wird das Intracluster-Protokoll vorgestellt, das als Basis für diese Arbeit dient. Schließlich wird auf die bei der Entwicklung eingesetzte Simulationsumgebung eingegangen.

2.1 Mobile Ad-Hoc-Netze

Ein mobiles Ad-Hoc-Netzwerk (MANET) ist ein drahtloses Computer-Netzwerk, welches sich durch mehrere Eigenschaften von gewöhnlichen drahtlosen und drahtgebundenen Netzwerken abgrenzt.

Das wichtigste Kriterium von MANETs ist das Fehlen einer festen Infrastruktur, wie sie z.B. beim Mobilfunk als Verbindung der Basisstationen untereinander realisiert ist. MANETs bestehen ausschließlich aus sich selbst konfigurierenden mobilen Knoten und benötigen zu ihrem Betrieb keine weiteren Voraussetzungen. Des Weiteren sind die Teilnehmer eines MANETs meist nicht in der Lage, durch direkte Verbindungen alle anderen Knoten im Netzwerk zu erreichen und sind deshalb auf die Kooperation der anderen Stationen angewiesen, um mit weiter entfernten Knoten zu kommunizieren.

Hinzu kommt, dass die zur Kommunikation verwendete drahtlose Übertragung fehleranfällig ist und nur geringe Bandbreiten im Vergleich zu kabelbasierten Netzwerken bietet.

Durch die begrenzte Reichweite der Funkübertragung und die Mobilität der Knoten kommt es häufig dazu, dass Stationen keine Verbindung mehr zueinander haben oder neue Verbindungen aufbauen. Dadurch ändert sich die Topologie des Netzwerks auf unvorhersehbare Weise, was den Versand von Daten über mehrere Zwischenstationen (Multi-Hop-Betrieb) erschwert.

Wireless LAN

Das vom Institute of Electrical and Electronics Engineers (IEEE) als Industriestandard *IEEE 802.11* [IEE] herausgegebene Protokoll für drahtlose Netzwerkkommunikation hat sich durch die günstige Fertigung und weite Verfügbarkeit von Endgeräten für die Funkvernetzung in Haushalten und als Medium zum Aufbau mobiler Ad-Hoc-Netzwerke durchgesetzt.

Dieser Standard unterstützt neben dem Infrastruktur-Modus auf Grundlage von festen Basisstationen (Access-Points) auch eine Ad-Hoc-Betriebsart, in der sich Stationen in gegenseitiger Sichtweite dynamisch zu Funkzellen zusammenschließen.

Frequenzbänder

Zur Datenkommunikation spezifiziert der Standard unterschiedliche Modulationsverfahren, die auf zwei lizenzfrei benutzbaren Funkfrequenzbändern Brutto-Datenraten von bis zu 54 Mbit/s erlauben. Im ISM-Band (für *Industrial, Scientific and Medical*, 2400–2500MHz) sind dabei in Europa 13, in den USA 11 Kanäle mit einem Abstand von 5MHz definiert, die eingesetzt werden dürfen. Da die Übertragungen jedoch mit einer Funkbandbreite von 22MHz stattfinden, können effektiv nur drei Kanäle überlappungs- und störungsfrei parallel genutzt werden. Arbeiten zwei Netzwerke auf sich überlappenden Frequenzen, so funktioniert die Erkennung anderer Sender nicht mehr zuverlässig und die Wahrscheinlichkeit von Kollisionen steigt.

Das zweite Frequenzband, das bei 5100-5700MHz liegt, ist in Deutschland erst Ende 2002 freigegeben worden und darf nur unter bestimmten Auflagen außerhalb geschlossener Räume verwendet werden. Es ist nur der Einsatz im Infrastruktur-Modus mit Access-Points erlaubt, die eine automatische Absenkung der Sendeleistung und automatischen Frequenzwechsel durchführen, wenn sie eine Überlappung mit anderen Nutzern des Frequenzbandes wie militärischen Radaranlagen und Satellitenortungssystemen erkennen. Diese Vorschriften machen die Implementierung der Technik teurer und sorgten bisher für eine geringere Verbreitung. Aus diesem Grund ist das Spektrum aber auch nicht so überlastet, wie das ISM-Band, in dem neben den weit verbreiteten WLANs auch Mikrowellen-Öfen und Bluetooth-Geräte stören können [Tri02].

MAC-Schicht

Die MAC-Schicht (*Media Access Control*, Sicherungsschicht) des WLAN-Protokolls wurde dem in der PC-Vernetzung üblichen Ethernet nachempfunden, um den Umstieg zu erleichtern. Sie implementiert neben der Direktkommunikation von Knoten auch das Versenden von Broadcast-Nachrichten an alle Teilnehmer. Die Zugriffssteuerung erfolgt über *Carrier Sense Multiple Access / Collision Avoidance* (CSMA[TK75]/CA) – die Medienzugriff erwünschenden Teilnehmer können erkennen, ob das Medium gerade frei oder besetzt ist (*Carrier Sense*) und warten im

Zweifelsfall bis es für eine zufällig bestimmte Zeit frei ist. Wenn danach niemand anders eine neue Übertragung gestartet hat, können sie senden.

Da das Medium naturgemäß fehleranfälliger ist als Datenkabel, wurden zusätzliche Sicherungsmaßnahmen eingeführt. Dazu gehört die in Hardware durchgeführte Sicherung der Nachrichten-Zustellung bei Punkt-Zu-Punkt-Verbindungen: Jede Nachricht wird vom Empfänger mit einem kurzen Bestätigungspaket quittiert; bekommt der Sender die Bestätigung nicht, so wiederholt er die Aussendung des Datenpakets, was bei Bedarf mehrfach geschieht. Dieses Verfahren bietet zwar erhöhte Zuverlässigkeit bei der Zustellung der Pakete, kann letztere aber nicht garantieren. Außerdem lässt sich die Anzahl der Wiederholungen nur schwer durch eine Anwendung abfragen.

Durch die begrenzte Sende-Reichweite kommt noch das Hidden-Station-Problem [TK75] ins Spiel: Sind drei Knoten A, B und C so angeordnet, dass B, der Mittlere, die beiden anderen empfangen kann, diese sich aber nicht gegenseitig sehen, so können A und C gleichzeitig versuchen, Daten an B zu senden und erzeugen eine Kollision beim Empfänger, die sie aber nicht mitbekommen. Dazu wurde eine weitere Erweiterung eingebaut, bei der der Sender durch ein kurzes RTS-Paket (*Ready To Send*), das die erforderliche Sendezeit enthält, seine Sendebereitschaft signalisiert. Dieses Paket wird vom Empfänger durch ein CTS (*Clear To Send*) bestätigt, welches auch die angeforderte Zeit enthält und von allen Stationen in Reichweite des Empfängers empfangen werden kann. Letztere wissen dann, dass das Medium für diese Zeit belegt ist und starten währenddessen keine eigene Übertragung.

2.2 Routing-Protokolle

Um Daten an ein Gerät zu senden, das sich nicht in unmittelbarer Reichweite befindet, müssen Mittler gefunden werden, die die Informationen weiterleiten. Die Suche nach solchen Mittlern und ihre Verwendung werden unter dem Begriff *Routing* zusammengefasst. Verfahren, die diese Funktion bereitstellen, nennt man entsprechend *Routing-Protokolle*. Knoten, die Datenpakete für andere weiterleiten, werden *Gateway* oder *Router* genannt. In einem MANET können dabei alle Teilnehmer gleichzeitig auch Gateways für die anderen Knoten sein.

Um zu wissen, wohin die Daten geschickt werden müssen, brauchen die Stationen Kenntnis von der Netzwerkstruktur. In kabelgebundenen Netzwerken ändert sich die Struktur nur selten, so dass auf statische Verfahren wie z.B. manuelle Einrichtung zurückgegriffen werden kann. Mobile Ad-Hoc-Netzwerke weisen dagegen eine sehr dynamische Struktur auf, was eine automatische Rekonfiguration der Knoten notwendig macht.

Die Routing-Protokolle sorgen dabei für den Austausch von Routing-Informationen, also Daten darüber, welche Knoten über welche direkten Nachbarn erreichbar sind. Hat ein Knoten solche Informationen über ein bestimmtes Ziel erlangt, so sendet er das Datenpaket für den Zielknoten an denjenigen seiner Nachbarn, der aus

seiner Sicht dem Empfänger am nächsten ist. Dort wird es erneut analysiert und den vorliegenden Informationen entsprechend weitergereicht.

Dabei kann es vorkommen, dass zwei Knoten über unterschiedliche Informationen verfügen und ein Datenpaket immer an den jeweils anderen senden, weil sie glauben, dass dieser dem Ziel näher ist. Solche *Routing-Schleifen* zu vermeiden, die eine enorme Netzlast nach sich ziehen können, gehört zu den wichtigsten Aufgaben des Routing-Protokolls.

Es ist auch möglich, dass ein Knoten keinerlei Informationen darüber hat, wie er ein Ziel erreichen kann, obwohl eine Verbindung existiert. In diesem Fall kann das Paket überhaupt nicht zugestellt werden. Auch solche Situationen sollte das Routing-Protokoll umgehen.

Ein weiteres Problem, das Routing-Algorithmen zu vermeiden haben, ist das *count-to-infinity*-Phänomen[Sch98]. Es tritt bei Protokollen ohne vollständiges Weltmodell auf, wenn ein Knoten A, der von seinen Nachbarn B und C zunächst direkt, also mit einem Abstand von 1 Hop, gesehen wird, aus dem Netzwerk verschwindet. Daraufhin entfernen B und C jeweils die direkte Route, glauben aber, über den jeweils anderen Nachbarn noch A erreichen zu können. Dabei propagiert B, dass er A über 2 Hops erreichen kann, da er zuvor von C empfangen hatte, dass C von A nur 1 Hop entfernt ist. C empfängt diese Nachricht und aktualisiert seinen Abstand zu A auf 3 (über B, dessen Abstand angeblich 2 ist). B, der die neue Information von C kriegt, setzt den Abstand auf 4. Das gegenseitige Inkrementieren der Länge setzt sich fort, bis der Maximalwert erreicht ist und die Route gelöscht wird. Dieses Problem führt nicht nur zu sehr lange bestehenden ungültigen Routen, sondern erhöht auch die Netzwerkbelastung durch Routing-Updates und sollte daher vermieden werden.

Im Folgenden wird zunächst ein Klassifizierungsschema für Routing-Protokolle vorgestellt, danach werden ausgewählte Verfahren aufgezeigt, ihre Vor- und Nachteile diskutiert und die Protokolle miteinander verglichen.

2.2.1 Klassifizierung von Routing-Protokollen

Bei der Betrachtung von Routing-Protokollen ist es zunächst sinnvoll, diese nach unterschiedlichen Kriterien zu klassifizieren[Fee99]. Die Strukturierung erfolgt dabei nach *Kommunikationsmodell*, *Scheduling*, *Struktur* und *Zustandsinformationen*.

Kommunikationsmodell

Das verwendete Kommunikationsmodell ist ein sehr grundlegendes Kriterium – hierbei geht es darum, ob die Kommunikation auf unterschiedlichen Frequenzen/Kanälen bzw. in unterschiedlichen Ad-Hoc-Zellen stattfindet, oder ob sich alle Teilnehmer auf einem gemeinsamen logischen Kanal befinden.

Bei der Verwendung mehrerer Kanäle können die Stationen über mehrere physikalische Sende- und Empfangseinheiten verfügen, die ihnen die gleichzeitige Arbeit

mit mehreren Frequenzen erlauben, um damit Voll-Duplex-Betrieb, bessere Signalisierung oder höhere Bandbreiten zu realisieren. Dies macht sie aber technisch aufwändiger und teurer.

Alternativ dazu können mehrere Kanäle auch vom selben Sender/Empfänger nach einem bestimmten Zeitschema verwendet werden. Dieses Zeitschema muss dann unter den miteinander kommunizierenden Teilnehmern synchron gehalten werden.

Um die Komplexität von Hardware und Software zu beschränken, werden in dieser Arbeit nur Protokolle vorgestellt, bei denen sich alle Knoten in der selben Ad-Hoc-Zelle auf dem selben Funk-Kanal befinden. Dadurch ist sichergestellt, dass die Konnektivität der Teilnehmer einzig von geografischen Parametern wie Entfernung und Hindernissen sowie von potentiellen Störern abhängig ist.

Scheduling

Ein weiteres wichtiges Unterscheidungskriterium ist, ob ein Protokoll *proaktiv* ist, also alle möglichen Pfade bereits im Voraus berechnet und bei Bedarf sofort Pakete versenden kann, oder ob es *reaktiv* arbeitet, also erst dann den Pfad zu einem Knoten sucht, wenn eine Verbindung angefordert wird. Proaktives Vorgehen hat den Vorteil, dass beim Versenden von Daten keine hohe Verzögerung zum Ermitteln des Ziels notwendig ist und dass es keine Skalierungsprobleme bei einer hohen Anzahl an Verbindungen gibt. Dagegen werden permanent Routing-Informationen auf dem Medium ausgetauscht, selbst wenn keine eigentlichen Daten zu verschicken sind.

Reaktive Protokolle (auch als *on-demand* bezeichnet) belasten das Medium dagegen nur dann, wenn auch tatsächlich eine Verbindung aufgebaut werden soll. Dazu verschicken sie Nachrichten an ihre Nachbarschaft, in denen nach dem Pfad zum Zielknoten gefragt wird. Kennt einer der Nachbarn die Route, liefert er sie als Antwort zurück, ansonsten wird die Anfrage immer weiter geleitet. Um die damit verbundene hohe Netzwerklast bei vielen Anfragen zu reduzieren, setzen reaktive Protokolle auf aggressives Caching, also das Zwischenspeichern von Routen, die ihnen einmal bekannt geworden sind.

Es existieren auch Protokolle, die einen *hybriden* Ansatz implementieren, also einen, der proaktives und reaktives Routing für unterschiedliche Ziele kombiniert. Als Auswahlkriterium dient meist die Entfernung der Ziele vom eigenen Knoten – für die eigene Nachbarschaft im Umkreis von wenigen Hops werden vollständige Routing-Informationen bereitgehalten, weiter entfernte Knoten werden an den Grenzen der Nachbarschaft reaktiv gesucht.

Struktur

Die Struktur des Netzwerks kann zur Klassifizierung von Routing-Protokollen herangezogen werden. Haben alle Knoten das selbe Verhalten, spricht man von einer *einheitlichen* oder auch *flachen* Struktur. Alternativ dazu existieren Nachbar-Aus-

wahl-Protokolle, bei denen jeder Knoten nur einen Teil seiner Nachbarn für Routing benutzt sowie *hierarchische* Modelle, bei denen einzelne Knoten zusätzliche Steuerungsaufgaben gegenüber anderen Stationen wahrnehmen. Weit verbreitet sind dabei in *Cluster* aufgeteilte Netze, bei denen jeder Cluster-*Head* mehrere *Clients* beaufsichtigt und deren Kommunikation regelt. Diese Hierarchie-Strukturierung kann auch rekursiv vorgenommen werden, wobei Cluster einer Ebene die Stationen der nächsthöheren Ebene bilden.

Zustandsinformationen

Man kann Routing-Protokolle auch nach der Art der auf jedem Knoten gespeicherten Informationen unterteilen. *Topologie-basierte* Protokolle speichern dabei ein Weltmodell oder Teile davon auf allen Knoten und die Knoten verteilen Informationen über ihre eigenen Nachbarschaftsbeziehungen (*link-state*) an alle.

Dem gegenüber stehen Protokolle, die nur Informationen über die Richtung und den Abstand zu anderen Knoten (*distance-vector*) austauschen. Damit kann zwar die zu versendende Datenmenge reduziert werden, es treten aber andere Probleme auf, die das Routing erschweren, beispielsweise das count-to-infinity-Phänomen.

2.2.2 Ausgewählte MANET-Routing-Protokolle

Im Folgenden sollen einige Routing-Protokolle für mobile Netze vorgestellt werden. Es wird insbesondere auf DSDV, AODV, OLSR und FSR eingegangen, die von der IETF (*Internet Engineering Task Force*, Gremium zur Standardisierung des Internets) als *Internet Draft* akzeptiert wurden. Um das Feld auszuweiten, werden auch ZRP, IZR als hybride Verfahren, LANMAR als Verfahren mit Gruppenmobilität und GeoCast vorgestellt, welches zwar nicht für MANETs bestimmt ist, jedoch durch die Verwendung geographischer Informationen zu Routingzwecken interessant ist.

DSDV

Destination Sequenced Distance Vector (DSDV) [PB94, Fee99] ist ein flaches zielbasiertes proaktives Protokoll. Es trifft Vorkehrungen gegen Routing-Schleifen, indem Routendaten mit vom Ziel generierten Sequenznummern[1] versehen werden und reduziert Fluktuationen durch das Verzögern von Updates über instabile Pfade. Jeder Knoten versendet periodisch seine Routing-Tabelle, die die Abstände und Sequenznummern zu jedem Ziel enthält. Zusätzlich versieht er jede solche Nachricht mit einer eigenen Sequenznummer.

[1] Sequenznummern sind inkrementell erhöhte Zähler, die zum zeitlichen Einordnen der Informationen dienen. Empfängt ein Knoten zwei Pakete mit unterschiedlichen Sequenznummern vom selben Sender, verwendet er das mit der größeren Sequenznummer, da es aktuellere Daten enthält.

Die Empfänger von Routing-Nachrichten vergleichen Sequenznummern und Abstände zu allen Zielen und aktualisieren ihre Einträge, falls die Sequenznummer größer oder falls bei gleich großer Sequenznummer der Abstand kleiner ist. Dabei wird jeweils der Sender der Routing-Nachricht als nächstes Gateway zum Ziel eingetragen.

Verliert ein Knoten die Verbindung zu einem Nachbarn, so inkrementiert er dessen Sequenznummer und setzt den Abstand auf ∞. Dadurch wird diese Information von allen Empfängern übernommen (größere Sequenznummer) und der ausgefallene Knoten entsprechend als nicht erreichbar markiert. Baut letzterer die Verbindung zum Netz wieder auf, versendet er sein nächstes Update auch mit einer inkrementierten Sequenznummer, die aber der von seinem Nachbarn bereits propagierten entspricht. Da aber der Abstand nicht mehr ∞ ist, wird auch diese Nachricht in die Routing-Tabellen aufgenommen. Durch dieses „Fremd-Inkrementieren" wird das Count-To-Infinity-Problem umgangen.

AODV

Ad-Hoc On-Demand Distance Vector (AODV) [PRD99, PRD03] ist ein reaktives flaches zielbasiertes Protokoll. Bei diesem Protokoll flutet ein Knoten, der ein Ziel sucht, ein Anfrage-Paket mit der gesuchten Adresse. Das Fluten besteht darin, dass das Paket von allen Empfängern wiederholt wird, so dass auch deren Nachbarn es empfangen und wiederum weitersenden. Jede Zwischenstation vermerkt den zurückgelegten Pfad im Paket, um eine Rücksignalisierung zu ermöglichen. Sobald der Zielknoten oder ein Knoten erreicht wird, der die Route zum Ziel kennt, kann eine Antwort auf dem aufgezeichneten Pfad zurückgeschickt werden. Auch dabei findet eine Aufzeichnung der Route statt, so dass der Sender diese auswerten und zum Versenden der eigentlichen Daten benutzen kann.

Fällt eine Route aus, auf der aktiv Daten übertragen werden, antwortet der den Ausfall feststellende Knoten mit einem unangeforderten Routing-Antwortpaket, das ∞ als Ziel-Abstand enthält. Dieses Paket wird nach und nach bis zum Sender der Daten propagiert wird, der dann eine neue Route initiieren kann.

OLSR

Optimized Link State Routing (OLSR) [CeA+03] ist ein proaktives nichteinheitliches Nachbar-Auswahl-Protokoll, bei dem jeder Knoten seinen Link-Status über eine ausgewählte Menge seiner Nachbarn (Multi-point relay, MPR) weiterleitet. Gegenüber dem Fluten wird die Weitergabe der Informationen dabei so begrenzt, dass redundante Übertragungen nicht stattfinden. Das MPR berechnet sich deshalb als die Teilmenge der direkten Nachbarn, die man braucht, um alle Knoten in zwei Hops Entfernung zu erreichen.

Ein Knoten verschickt seine Routing-Daten per Broadcast, womit er alle unmittelbaren Nachbarn erreicht. Weitergeleitet werden die Informationen dann nur noch von

den Knoten in seinem MPR, was dabei hilft, die Netzlast zu reduzieren. Zusätzlich wird das Aktualisierungs-Intervall abhängig von der Stabilität des Netzabschnittes variiert, um unnötige Übertragungen einzusparen.

ZRP

Das Zone Routing Protocol (ZRP) [HPS02a, HPS02b] ist ein Nachbar-Auswahl-Protokoll, dessen Entwicklungsziel darin bestand, die Vorteile von proaktivem (schneller Routenaufbau) und reaktivem (geringe Netzbelastung) Routing zu kombinieren. ZRP besteht entsprechend aus zwei Unterprotokollen: Ein proaktives Intrazone Routing Protocol (IARP) sorgt dafür, dass jeder Knoten seine Nachbarschaft über mehrere Hops kennt. Das reaktive Interzone Routing Protocol (IERP) dagegen kümmert sich um die Suche von Knoten außerhalb der Nachbarschaft. Die Größe der Nachbarschaft (der Zone) wird dabei über den Zonen-Radius bestimmt, der die Anzahl der Hops angibt, über die das IARP ausgeführt wird. Die ideale Wahl der Zonen-Größe hängt vom Einsatzszenario ab, kleinere Zonen sind für dichte Netzwerke mit wenigen sehr mobilen Knoten besser, größere Zonen eignen sich dagegen eher für verstreute Netze, in denen sich viele Knoten langsam bewegen.

Als Optimierung für die Suche von Knoten außerhalb der Zone wird Bordercasting eingesetzt. Bei diesem Verfahren wird die Anzahl der Übertragungen minimiert, die für einen Broadcast einer Nachricht an der eigenen Zonengrenze notwendig sind.

An Stelle von IARP und IERP können auch andere proaktive bzw. reaktive Protokolle eingesetzt werden, um ein hybrides Zonenrouting zu erzielen. Das Prinzip von ZRP ist dabei übertragbar, wenn das reaktive Protokoll an die Bordercast-Möglichkeit angepasst wird.

IZR

Mit Independent Zone Routing [SPH04] existiert eine Optimierung von ZRP, bei der jeder Knoten über einen eigenen Zonenradius für die Unterscheidung zwischen proaktivem und reaktivem Protokoll verfügt. Dieser Radius wird dezentral in Abhängigkeit von der Mobilität der Nachbarschaft bestimmt. Auf diese Weise legt jeder Knoten seine eigene Zone fest. Kompliziert wird es allerdings dadurch, dass jeder Teilnehmer seine Informationen an alle Knoten verschicken muss, in deren Zone er sich befindet – daher müssen die Routing-Updates unterschiedliche Strecken zurücklegen, die der Knoten aus den ihm zur Verfügung stehenden Informationen bestimmen muss.

FSR

Fisheye State Routing (FSR) [PGC00] ist ein proaktives Routing-Protokoll mit flacher Struktur. Das Prinzip leitet sich vom Auge eines Fisches ab, das im Brennpunkt eine sehr hohe Detaildichte hat, die mit zunehmendem Abstand vom Brennpunkt abnimmt. Bei FSR wird das Prinzip so umgesetzt, dass ein Knoten sehr präzise Infor-

mationen über seine Nachbarschaft hat, die mit zunehmendem Abstand ungenauer und älter werden. Dazu werden die Link-Daten anderer Knoten in desto größeren Intervallen übertragen, je größer der Abstand zu diesen Knoten ist. Jeder Knoten hält dabei eine gesamte Karte der Topologie vor, die aktuelle Verbindungen aus seiner Nachbarschaft beschreibt, während weiter entfernte Verbindungen unpräzise dokumentiert sind. Dieser Nachteil wirkt sich in der Praxis allerdings wenig aus, da Pakete, die ihrem Ziel näher kommen, auf Knoten mit immer präziseren Routing-Informationen stoßen und damit präziser weitergeleitet werden können.

LANMAR

Interessant ist auch das Landmark Ad Hoc Routing Protocol (LANMAR) [PGH00]. Dieses hybride hierarchische Verfahren ist auf Gruppenmobilität (ursprünglich im militärischen Bereich) ausgerichtet und setzt dazu Landmarks (Orientierungspunkte) ein, wobei jede Gruppe über einen eigenen Landmark verfügt und jeder Teilnehmer die Pfade zu allen Mitgliedern seiner Gruppe und zu allen Landmarks proaktiv bezieht. Will ein Knoten mit einem Teilnehmer einer anderen Gruppe kommunizieren, so schickt er die Nachrichten an den Landmark der Zielgruppe. Sobald die Datenpakete diese Gruppe erreichen, werden sie mit Hilfe der lokalen Routing-Tabellen weitergeleitet. LANMAR reduziert die Größe der Routing-Tabellen und den Verwaltungsaufwand, funktioniert allerdings nur dann zufrieden stellend, wenn die Gruppen sich geschlossen bewegen – verteilen sich die Knoten nach anderen Mustern, oder entfernt sich ein Landmark von seiner Gruppe, sinkt die Effizienz rapide.

GeoCast

Ein Verfahren, das zusätzliche geografische Informationen aus dem GPS (*Global Positioning System* zur satellitengestützten Positionsbestimmung) für Routing verwendet ist GeoCast. Dabei werden die Ziele nicht über Knotenkennungen, sondern über den Aufenthaltsort der Knoten bestimmt: Die Adressierung der Datenpakete geschieht anhand von Punkten im Koordinatensystem, kreisförmigen oder durch Polygone beschriebenen Flächen. Die Auslieferung erfolgt an alle Stationen, die sich im Zielbereich befinden.

GeoCast-Router sind hierarchisch angeordnet und besitzen Zielbereiche, für die sie zuständig sind und deren Koordinaten sie kennen. Empfängt ein GeoRouter ein Datenpaket, das sich mit seinem Zuständigkeitsbereich überschneidet, sendet er es in den entsprechenden Bereich. Fallen Teile oder die gesamte Zielfläche nicht in den von ihm abgedeckten Raum, so schickt er die Nachricht zusätzlich an den nächsthöheren Router in der Hierarchie, der sie analog weiterverteilt. Allerdings ist GeoCast nicht auf Knotenmobilität ausgelegt – die Zielbereiche und Routing-Tabellen enthalten nur statische Koordinaten. Mit der Verfügbarkeit günstiger und stromsparender GPS-

Empfänger wird die Nutzung von Geo-Koordinaten zur Routing-Unterstützung in MANETs allerdings auch praxistauglich.

Zusammenfassung

Im vorhergehenden Abschnitt ist eine kleine Auswahl von unterschiedlichen Routing-Protokollen präsentiert worden (Tabelle 2.1). Eine vollständige Übersicht würde den Rahmen dieser Arbeit sprengen, jedoch sollte dieser Abschnitt für einen Überblick über die wichtigsten Protokolle und die bei solchen Protokollen möglichen Konzepte genügen.

Protokoll	*Scheduling*	*Struktur*	*Zustand*	*Anmerkungen*
DSDV	proaktiv	flach	distance-vector	frühes Protokoll
AODV	reaktiv	flach	distance-vector	
OLSR	proaktiv	Nachbar-Ausw.	link-state	praktischer Einsatz
ZRP	hybrid	Nachbar-Ausw.	hybrid	unterschiedliche proaktive / reaktive Protokolle möglich
IZR	hybrid	Nachbar-Ausw.	hybrid	ZRP mit individuellen dynamischen Zonen
FSR	proaktiv	flach	link-state	
LANMAR	hybrid	hierarchisch	link-state	Gruppenmobilität
GeoCast	proaktiv	hierarchisch	Ziel-Fläche	keine Mobilität

Tabelle 2.1: Routing-Protokolle in der Übersicht

DSDV und AODV, die ersten von der IETF als Internet Draft akzeptierten Protokolle, zeigen die Migration der bis dahin in drahtgebundenen Netzwerken üblichen proaktiven (DSDV) bzw. reaktiven (AODV) Vorgehensweise auf MANETs. Sie bieten die Grundlage für die Entwicklung zahlreicher neuerer Protokolle. OLSR und FSR versuchen, die Netzwerkbelastung durch Topologie-Daten zu verringern, indem sie die zu versendende Informationsmenge verringern. ZRP und IZR kombinieren reaktive und proaktive Arbeitsweisen für unterschiedliche Ziel-Entfernungen, um so die Vorteile der beiden Routing-Typen zu vereinen. LANMAR optimiert die Menge der Topologie-Daten ausgehend von der Annahme, dass bestimmte Knoten-Gruppen sich gemeinsam bewegen und man alle Knoten innerhalb ihrer Gruppe wiederfinden kann. GeoCast verwendet statt normaler Zieladressen geographische Koordinaten und ein hierarchisches Modell, um Datenpakete an Punkte oder Flächen auf der Erde auszuliefern, erlaubt aber keine Mobilität.

Es ist zu erkennen, dass die Entwickler der Protokolle unterschiedliche Ziele verfolgt haben und die Verfahren ihre Daseinsberechtigung in unterschiedlichen Sze-

narien besitzen. Proaktive Protokolle eignen sich besser für Netze, in denen viele unterschiedliche Knoten miteinander kommunizieren und die Mobilität nicht sehr hoch ist, reaktive sind dagegen besser geeignet, wenn sich viele Knoten bewegen und dafür nicht so häufig Verbindungen aufgebaut werden. Hybride Verfahren versuchen, den Vorteil der beiden Methoden zu kombinieren, indem sie diese jeweils für das passendere Szenario einsetzen.

Welches Routing-Protokoll eingesetzt wird, hängt im Endeffekt stark von den Anforderungen ab, die man an ein Netz stellt. Dadurch jedoch, dass einige Protokolle nicht vollständig implementiert sind, der Code häufig nicht öffentlich verfügbar ist, unterschiedliche Verfahren z.T. auf unterschiedliche MAC-Schnittstellen setzen und die meisten Evaluierungen nur als Simulationsergebnisse vorliegen, ist ein direkter Vergleich der Protokolle im eigenen Einsatzszenario fast unmöglich. Die Auswahl des eigenen Protokolls muss sich deswegen hauptsächlich auf theoretische Betrachtungen stützen.

2.3 Quality Of Service

Für einige Anwendungen ist es erforderlich, nicht nur Pakete zwischen entfernten Stationen übertragen zu können, sondern auch bestimmte Anforderungen an die Verbindung zu stellen. Während ein Dateitransfer je nach verfügbarer Bandbreite früher oder später abgeschlossen ist, erfordert eine digitale Sprachverbindung eine zugesicherte Mindestbandbreite. Kann diese Bandbreite nicht gewährleistet werden, lässt sich eine unnötige Belastung des Netzes dadurch vermeiden, dass die Verbindung gar nicht erst aufgebaut wird. Verändert sich die verfügbare Bandbreite im Betrieb, so sollte das der Anwendung signalisiert werden, damit diese sich daran anpassen kann – im Falle einer Sprachverbindung könnte ein Codec mit höherer Sprachqualität und höherer erforderlicher Bandbreite gewählt werden.

Ein QoS-Routing-Protokoll hat die Aufgabe, entsprechend den Anforderungen einer Anwendung an die Bandbreite und die maximale Latenz einen Pfad zum Ziel zu finden, der die Kriterien erfüllt, oder die Nichterfüllbarkeit zu signalisieren. Es muss dafür sorgen, dass auf den Gateway-Knoten Reservierungen für diese Verbindung vorgehalten werden und dass bei Ausfällen entweder alternative Pfade gefunden werden oder die Anwendung benachrichtigt wird.

Zu den auf diese Weise realisierbaren und überwachbaren Anforderungen gehört neben der Bandbreite der Verbindung auch die maximale Paket-Verlustrate. Dazu zählen auch die Latenz, also die Zeit zwischen dem Versand und dem Empfang der Pakete und der Jitter, also die Varianz der Paketlaufzeiten.

2.3.1 Strukturierung von QoS-Protokollen

Für das Quality-Of-Service-Management in Netzwerken existieren zwei grundsätzliche Herangehensweisen.

DiffServ (eine Abkürzung für *Differentiated Services*, [BBC+98]) ist eine QoS-Architektur für große Netze wie das Internet. Der Gedanke dahinter besteht in der Aufteilung des Datenverkehrs in unterschiedliche Klassen und die Zuordnung von QoS-Eigenschaften und Maximalwerten zu diesen Klassen.

Die genauen Parameter werden dabei in *Service Level Agreements* zwischen Service-Providern und ihren Kunden festgelegt, die Durchsetzung der Klasseneigenschaften erfolgt auf den Grenz-Routern der Service-Provider. Die Klassen werden als Richtlinie verwendet, welche Pakete bei einer Netzwerküberlast verzögert bzw. verworfen werden können und welche Daten mit Vorrang zu versenden sind.

Da DiffServ den Datenverkehr in Klassen ordnet, ist es praktisch unmöglich, Ende-zu-Ende-Qualitätsgarantien zu geben. Dies wird noch weiter erschwert, wenn der Datenverkehr über unterschiedliche Provider läuft und damit Teil unterschiedlicher QoS-Klassen werden kann. Eine Rückmeldung an die Anwendung findet auch nicht statt, sondern muss vom Empfänger über die Analyse der empfangenen Pakete realisiert werden.

IntServ (*Integrated Services* [BCS94]) ist im Gegensatz zu DiffServ feingranular ausgelegt. Die Regulierung findet auf Verbindungsebene statt und alle Anwendungen, die QoS-Garantien brauchen, müssen diese vor Aufbau der Verbindung anfordern. Dazu ist es erforderlich, dass jeder Router in dem Netzwerk das QoS-Verfahren implementiert und alle durch ihn führenden Verbindungen erfasst.

Mit IntServ ist ein deutlich höherer Netzwerk-Overhead durch die Reservierung und Erhaltung von QoS-Verbindungen verbunden, der aber im Gegenzug dafür sorgt, dass die Anforderungen der Anwendungen pro Verbindung durchgesetzt und gesichert werden können. Sind die Bedingungen für die QoS-Verbindung nicht gegeben, so findet eine entsprechende Benachrichtigung der Anwendung statt, welche ihr Verhalten auf die neue Situation einstellen kann.

Im Bezug auf MANETs stellt die Implementierung von IntServ eine besondere Herausforderung dar, weil dort nicht nur die Bandbreiten der bestehenden Pfade sondern auch Pfadänderungen und mobilitätsbedingte Unterbrechungen in Betracht gezogen werden müssen.

Bei den IntServ-Protokollen kann man weiterhin danach unterscheiden, ob die QoS-Signalisierung *in-band*, also als zusätzliches Feld in den übertragenen Datenpaketen, oder *out-of-band*, also in Form eigenständiger Nachrichten stattfindet. Erstere bietet einen geringeren Overhead, allerdings besteht keine Möglichkeit, QoS-Garantien für eine Verbindung von Anfang an zu gewährleisten – diese können erst bei bestehendem Datenaustausch verhandelt werden.

Es gibt QoS-Protokolle, die auf einem existierenden Routing-System aufsetzen und solche, die eine Kombination aus QoS und Routing anbieten – letztere haben den

Vorteil, dass die Pfad-Daten für beide Zwecke verwendet werden können, so dass alternative Routen unter Umgehung überlasteter Knoten gefunden werden können.

2.3.2 QoS-Protokolle für MANETs

Im Folgenden werden einige QoS-Protokolle für mobile Ad-Hoc-Netze vorgestellt. Dabei werden INSIGNIA und ASAP als reine QoS-Verfahren für bestehende Netze behandelt, es wird auf CEDAR als QoS-Routing-Protokoll und MMWN eingegangen, welches dem militärischen Bereich entstammt.

INSIGNIA

Ein IntServ-QoS-Verfahren, welches mit bestehenden Routing-Algorithmen kombiniert werden kann ist INSIGNIA[LAZC00]. Dieses Verfahren setzt auf IP-Netzwerken auf, indem zusätzliche Header für IP-Pakete eingeführt werden, die QoS-Parameter übertragen. Die Reservierung erfolgt *in-band*, also wenn die eigentliche Verbindung bereits aufgebaut ist. Dazu fügt der Sender einen INSIGNIA-Header in ein zu versendendes Paket ein, und gibt darin den Reservierungswunsch sowie die gewünschte Mindest- und Höchstbandbreite an. Durch dieses Verfahren ist der Mehraufwand für das Protokoll minimal.

Die Gateway-Knoten versuchen anhand dieser Angaben eine Reservierung der Bandbreite durchzuführen. Steht weniger als die Maximalbandbreite zur Verfügung, ändern sie die im Header vermerkten Anforderungen ab. Haben sie nicht genügend Bandbreite für die Mindestanforderung, so werden die Pakete als Best-Effort weitergeleitet. Der Empfänger sieht den Status der Reservierung anhand der an ihn übermittelten Header und kann diesen bei Antwortpaketen an den Sender zurückschicken, so dass dieser seine Bandbreite entsprechend anpasst. Daher richtet sich INSIGNIA an adaptive Anwendungen, die in der Lage sind, das Feedback auszuwerten und die verschickte Datenmenge auch zur Laufzeit an die Netzsituation anzupassen.

Die QoS-Header müssen regelmäßig in die Datenpakete eingebettet werden, da es keine Möglichkeit gibt, über Routen-Änderungen informiert zu werden, und Pakete möglicherweise über Gateways umgeleitet werden, die noch keine Reservierung für eine gegebene Verbindung haben. Aus diesem Grund erfolgt die Reservierung auch implizit – ein Gateway-Knoten führt sie durch, sobald er ein Paket mit einem QoS-Header sieht, und verwirft sie, wenn einige Zeit lang keine Pakete dieser Verbindung mehr über ihn verschickt wurden.

Als zusätzliche Maßnahme existieren zwei Paket-Typen: *base QoS* und *enhanced QoS*. Diese dienen der Priorisierung von Paketen einer Verbindung in Engpass-Situationen – Pakete mit dem enhanced-QoS-Bit enthalten Zusatzinformationen, die für die Kommunikation zwar relevant, aber nicht essenziell sind, und können somit auf dem Weg verworfen werden.

ASAP

Als Weiterentwicklung von INSIGNIA wurde *ASAP*[SXA04] (Adaptive ReServation and Pre-Allocation protocol) vorgestellt. Dieses Verfahren splittet die Bandbreitenreservierung in zwei Phasen auf. In der ersten – wenn ein QoS-Paket vom Sender zum Empfänger geschickt wird – tragen die Gateway-Knoten die erforderliche Bandbreite als *weiche Reservierung* in ihre Tabellen ein. Diese Bandbreite darf dann nicht für andere Anwendungen reserviert werden, ist aber immer noch für Best-Effort-Daten freigegeben. Wenn die Routen-Reservierung bis zum Empfänger vorgedrungen ist, wird mit dem Antwortpaket ein Signal verschickt, dass die Reservierung nun *hart* ist. Auf diese Weise ist die Netzbelastung durch unvollständige Routen geringer.

Beide Protokolle sind vom Routing vollständig abgekoppelt, was ihre Funktionsweise benachteiligt – so werden Routen-Änderungen dem QoS-Protokoll nicht explizit mitgeteilt und müssen eruiert werden. Außerdem besteht durch die Trennung keine Möglichkeit, alternative Pfade zu suchen, wenn die direkte Route zum Ziel nicht die erforderliche Bandbreite bietet – die Kommunikation wird lediglich auf Best-Effort-Niveau reduziert.

CEDAR

Ein Verfahren, welches Routing und QoS kombiniert ist CEDAR (Core Extraction Distributed Ad hoc Routing) [SSB99]. Das ist ein reaktives hierarchisches Protokoll, bei dem ein Teil der Knoten zu einem *Minimum Dominating Set* (MDS) gewählt wird. Das ist die kleinste Menge von Knoten, so dass jeder Teilnehmer höchstens einen Hop bis zum nächsten MDS-Knoten hat. Letztere bilden untereinander den Kern des Netzes und sind über „Tunnel" durch je höchstens zwei normale Knoten miteinander verbunden.

Kennt ein Knoten nicht den Pfad zu seinem Ziel, so schickt er eine Anfrage an seinen *dominator*, also den ihm benachbarten Kern-Knoten. Dieser leitet das Suchpaket dann an alle anderen Knoten im MDS und bestimmt so den Kern-Pfad zum Dominator des Zielknotens. Neben Routen werden im Kern auch QoS-Informationen verteilt, wobei stabile, langanhaltende Links am weitesten bekanntgegeben werden. Der Sender kann aus den Informationen, die er von seinem Dominator bekommt, eine mögliche Bandbreite für die Übertragung bestimmen, die er in Abstimmung mit dem Dominator in einen Pfad mit gesicherter Bandbreite umwandelt, um diesen für die Daten zu benutzen.

MMWN

Ein weiteres Beispiel für die Kombination von QoS und Routing ist MMWN (hierarchically-organized Multihop Mobile Wireless Networks for quality-of-service support). Dieses hierarchische link-state-basierte Verfahren ordnet mobile Knoten Zellen zu, wobei jede Zelle einen *cellhead* besitzt, der ein direkter Nachbar aller zur Zelle

gehörenden Knoten ist. Zellen können sich zu Clustern zusammenschließen, welche wiederum rekursiv noch größere Cluster bilden können. Die Knotenadressen bilden diese Hierarchie ab, indem sie aus den Cluster-Bezeichnern von der höchsten Schicht an zusammengesetzt werden. So ist die Adresse von Knoten X, der sich in Zelle U von Cluster A befindet *A.U.X*.

Neben den Link-Zuständen werden immer auch Dienstgüte-Informationen übertragen, dabei wird der Datenaustausch auf den unmittelbar davon betroffenen Bereich begrenzt. Bei der Routenberechnung werden die Dienstgüte-Anforderungen der Anwendung berücksichtigt, die neben der Pfadlänge eine entscheidende Rolle spielen. Zur Vermeidung von Routing-Schleifen durch veraltete Informationen wird auf eine Kombination aus Source-Routing und Destination-based-Routing gesetzt:

Der Sender setzt die Route soweit ihm bekannt in das Datenpaket – für den eigenen Cluster ist der Pfad dabei vollständig, über den Rest der Strecke enthält er nur Informationen, welche Hierarchie-Ebenen wie zu traversieren sind. Die detaillierten Informationen werden von den Gateways auf dem Weg vervollständigt.

Jeder Cluster besitzt einen *location manager*, der den Aufenthaltsort aller Knoten im Cluster kennt und Unterstützung bei der Pfadsuche anbietet. Dieser erlaubt es, auf Knotenbasis die Aktualisierungsintervalle entsprechend der Mobilität und Anfragehäufigkeit der Knoten anzupassen. Die Location Manager ermitteln auch die Ziel-Adressen für Anfragen aus dem Cluster, also die Anordnung der Ziele in der gesamten Hierarchie.

MMWN bietet zusätzliche Mechanismen zur Änderung bestehender Verbindungen an, um Dienstgüte trotz der Knotenmobilität zu gewährleisten. So können Teilstrecken einer Verbindung umgeleitet oder Pakete an mehrere mögliche Aufenthaltsorte des Zielknotens gesendet werden. Zusätzlich erlaubt das Verfahren die Überwachung und Änderung der Link-Eigenschaften – z.B. das Anheben der Sendesignalstärke, wenn die Verbindungsqualität unter einen gewissen Wert sinkt.

Neben Dienstgüte und Verbindungsreparaturen bietet MMWN auch Multicast-Unterstützung an. Dazu existiert in jedem Cluster ein *multicast manager*, der für die Verwaltung der Multicast-Gruppen und das Hinzufügen und Entfernen von Knoten zu diesen zuständig ist.

Dadurch, dass MMWN für militärische Zwecke entwickelt worden ist, ist eine Übertragung auf zivile Technik allerdings nur eingeschränkt möglich, da dort nicht nur andere Regulierungsvorschriften für Sendeleistung und weitere Betriebsparameter gelten, sondern auch begrenzte Anschaffungskosten die Möglichkeiten eingrenzen.

Zusammenfassung

Bei den QoS-Verfahren gilt um so mehr das, was schon für die Routing-Protokolle gesagt wurde: Jedes Verfahren wurde ausgehend von einem bestimmten Szenario entwickelt und funktioniert in anderen Situationen nicht optimal. Die Auswahl eines

Verfahrens oder die Kombination unterschiedlicher Methoden für eigene Zwecke muss daher sorgfältig abgewogen werden.

Protokoll	*Signalisierung*	*Routing*	*Anmerkungen*
INSIGNIA	in-band	verwendet IP	adaptive Bandbreitenreservierung
ASAP	in-band	verw. IPv6	Weiterentwicklung von INSIGNIA
CEDAR	out-of-band	reaktiv	hierarchisches Routing mit QoS
MMWN	out-of-band	proaktiv	hierarchisch; Militärtechnik

Tabelle 2.2: QoS-Protokolle in der Übersicht

In Tabelle 2.2 findet sich eine Übersicht über die QoS-Protokolle. Während INSIGNIA und ASAP die QoS-Signalisierung in die eigentlichen Datenpakete einbetten, weil sie das Routing der Pakete nicht beeinflussen können und bei scheiternden Reservierungen auf Best-Effort zurückschalten, kombinieren CEDAR und MMWN Routing und QoS. Das erlaubt ihnen, Pfade nicht nur nach ihrer Länge, sondern auch nach den Qualitätsanforderungen der Anwendung auszuwählen. Beide bauen eine Routing-Hierarchie auf, die bestimmte Knoten mit Zusatzaufgaben zur Pfad-Suche und zum Monitoring der Umgebung bestimmt, und die normale Clients beim Verbindungsaufbau unterstützt.

2.4 Intracluster-Protokoll

Um zuverlässig Daten zwischen entfernten Stationen in einem MANET auszutauschen, müssen zwei Voraussetzungen erfüllt sein. Zum einen müssen Knoten in der Lage sein, zuverlässig Pakete an ihre unmittelbaren Nachbarn zuzustellen und Fehler bei der Zustellung zu erkennen. Zum anderen brauchen sie eine konsistente Sichtweise auf ihre Nachbarschaft, die die tatsächliche Konnektivität möglichst gut abdeckt.

Um beide Ziele zu erreichen, wurde in einer vorangegangenen Arbeit der Arbeitsgruppe „Echtzeitsysteme und Kommunikation" das Intracluster-Protokoll entwickelt. Dieses Protokoll dient der selbständigen Anordnung der mobilen Knoten zu Clustern, der Synchronisierung des Paketversands innerhalb eines Clusters und der Kollisionsvermeidung von unterschiedlichen Clustern. Es implementiert ferner *reliable Broadcast*, also die zuverlässige Auslieferung von Datenpaketen an alle Cluster-Mitglieder. Durch die Regelung des Medienzugriffs tritt das Protokoll gegenüber seinen Anwendungen als erweiterte MAC-Schicht auf.

2.4.1 Clustering

In einem mobilen Ad-Hoc-Netzwerk müssen sich die Knoten selbst organisieren. Dazu verfügen sie über keine anderen Informationen als die Datenpakete, die sie von ihren unmittelbaren Nachbarn empfangen können.

Beim vorliegenden Clustering-Schema können Knoten einen von zwei Zuständen annehmen. Ein Cluster-*Head* ist eine Station, die einen Cluster leitet, ein *Client* ist einem oder mehreren Heads untergeordnet. Um die Existenz des Clusters anzuzeigen, versendet ein Head in regelmäßigen Zeitabständen Informationspakete (*Invite*), die die Cluster-Id und die Liste der Clients enthalten.

Cluster-Struktur

In Abbildung 2.1 sieht man eine typische Anordnung von Cluster-Heads und Clients. Cluster 1 (die Cluster-Id entspricht der Knoten-Kennung des Heads) ist dabei durch eine schraffierte Ellipse gekennzeichnet, zu ihm gehören neben dem Head (Knoten 1) die Clients 2, 3, 4 und 7.

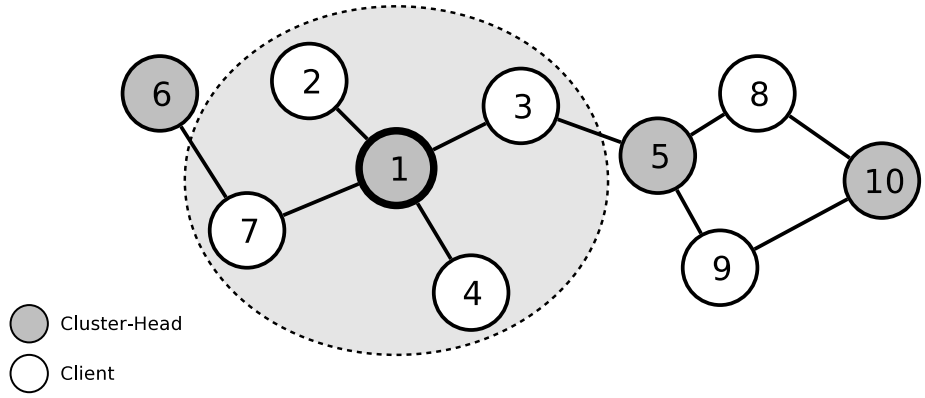

Abbildung 2.1: Cluster-Struktur (Cluster 1 hervorgehoben)

Die Cluster-Heads können dabei jeweils nur mit Clients verbunden sein, welche wiederum nur Verbindungen mit Heads eingehen können. Direktverbindungen zwischen Clients und Clients, bzw. zwischen Heads und Heads sind nicht zugelassen.

Kommunikation zwischen Clustern ist damit über Gateway-Knoten, also solche Clients, die zu mehreren Heads verbunden sind, vorgesehen. Im Bild sind das die Stationen 3, 7, 8 und 9.

Bildung von Clustern

Wird ein Knoten zum ersten Mal aktiviert, startet er im Client-Betrieb und wartet zunächst auf Invite-Nachrichten. Hört er einen Cluster-Head in seiner Reichweite, so versucht er, dessen Cluster beizutreten. Dies geschieht durch ein *JoinRequest-*

Paket an den Head, welches unmittelbar nach einem Invite zu verschicken ist. Wird der Knoten aufgenommen, so wird er im nächsten Invite-Paket aufgeführt und vom Head regelmäßig kontaktiert.

Dabei kann ein Client auch mit mehreren Cluster-Heads verbunden sein – in dem Fall kann er als *Gateway-Knoten* für Multi-Hop-Übertragungen eingesetzt werden. Die Anzahl der Cluster, in denen ein Client gleichzeitig sein kann, ist dabei durch eine Konstante begrenzt, genauso wie die Clustergröße.

Hört ein Client keine Heads oder kann er zu keinem Cluster beitreten (z.B. weil dieser voll ist), so schaltet der Knoten nach einer bestimmten Wartezeit in den Head-Modus und verschickt eigene Invite-Pakete. Daraufhin versuchen alle Clients, die die Nachrichten empfangen, sich mit ihm zu verbinden.

Eine Maßnahme zur Reduktion der Clustermenge besteht in der Verschmelzung mehrerer Cluster zu einem – dabei versucht ein Head, der (z.B. durch eigene Bewegung) in Reichweite eines anderen Clusters kommt, zu dessen Client zu werden. Ist das nicht möglich, weil die beiden Heads zusammen mehr Clients haben, als einer verwalten kann, so findet keine Verschmelzung statt. Da die Entscheidung, welcher der beiden Heads zum Client werden soll, verteilt erfolgen muss, wurde dafür ein Kriterium herangezogen, welches beide Heads überprüfen können: Das Alter der Cluster, welches auch im Invite-Paket auftaucht. Um die Stabilität von lange bestehenden Verbindungen zu fördern, wird deshalb der jüngere Cluster aufgelöst.

Cluster-Zeitschema

In einem Cluster erfolgt die Kommunikation streng nach einem Zeitschema. Dieses besteht aus Runden, wobei jede Runde aus einer bestimmten Anzahl von Zeitschlitzen besteht. Das Zeitschema wird vom Head vorgegeben, indem er in jeder Runde Nachrichten verschickt. Dabei kann er zwei Arten von Paketen aussenden – einmal pro Runde generiert er ein Invite-Paket, welches von noch nicht verbundenen Stationen verwendet werden kann, um dem Cluster beizutreten. In allen weiteren Zeitschlitzen befragt er seine Clients der Reihe nach mit einem *Poll*-Paket, auf das sie mit einem *Client-Data*-Frame mit den zu versendenden Daten antworten.

Einen Zeitschlitz nutzt der Head, um seinen lokalen Anwendungen die Möglichkeit zum Datenversand zu geben. Dazu trägt er sich selbst als Pseudo-Client in die Liste ein und führt jede Runde einen lokalen Poll durch, den er selbst beantwortet. Die Anfrage wird dabei auf dem realen Medium verschickt, da sie auch für andere Clients relevante Daten enthält.

Damit ein Client, der in mehreren Clustern ist, nicht das Schema des einen Clusters stört, während er mit dem anderen kommuniziert, lässt jeder Head zwischen den tatsächlich verwendeten Zeitschlitzen jeweils einen festen Zeitabstand von $N-1$ Slots, in dem andere Cluster arbeiten können. Sieht ein Head, dass ein anderer Cluster in seinem Zeitschlitz kommuniziert, „springt" er einfach einen Zeitschlitz weiter und

entgeht auf diese Weise Kollisionen. Hierdurch kann ein Client gleichzeitig in bis zu N Clustern sein, die sich das Medium teilen.

Da die Cluster-Heads keine synchronisierten Uhren haben, kann es vorkommen, dass sich ihre Zeitschlitze nur teilweise überlagern. Um diesem Problem vorzubeugen, darf nur während der ersten Hälfte eines Zeitschlitzes kommuniziert werden. Die zweite Hälfte dient als Synchronisationspuffer und zum Erkennen von Zeitschlitzkollisionen.

2.4.2 Zuverlässiger Nachrichtenversand

Da die Clients sich nicht immer gegenseitig sehen, hat der Head die Aufgabe, alle Datenpakete, die er von einem Client empfängt, an die anderen weiterzuleiten. Dies geschieht, indem die Client-Daten im Poll-Paket des nächsten Slots eingebettet werden. Damit alle teilnehmenden Stationen die Nachricht empfangen können, wird der Poll als Broadcast-Nachricht versendet.

Ein weiterer Grund für die Verwendung der Broadcast-Übertragung ist die Tatsache, dass die MAC-Schicht der meisten WLAN-Karten eine eingebaute Sendewiederholung für Unicast-Pakete hat, die sich nicht abschalten lässt und die man nicht auswerten kann. Um garantierte Übertragungen bieten zu können, muss der Sender aber wissen, ob die Nachricht angekommen ist oder nicht, bzw. wie viele Sendeversuche benötigt wurden.

Diese Funktionalität wurde daher hardware- und firmwareunabhängig in der Cluster-API implementiert. Jeder Knoten führt hierzu eine Checkliste über die in der letzten Runde empfangenen Pakete. Diese Liste schickt er in Form eins Bit-Feldes mit jedem Paket an den Head mit. Stellt dieser dann fest, dass der Client ein Paket nicht empfangen hat, so fordert er dieses Paket in der nächsten Runde erneut beim Versender an und verteilt es wieder. Kann das Paket drei Mal nicht zugestellt werden, so geht der Head davon aus, dass die Verbindung zum Client zu schlecht ist und entfernt diesen aus dem Cluster.

Kommunikations-Beispiel

Im Beispiel (Abbildung 2.2) ist ein Abschnitt der Kommunikation der Cluster-Heads A und C sowie von Client B, der in beiden Clustern ist, dargestellt. Beide Cluster-Heads besitzen noch weitere Clients, die Kommunikation mit diesen wird durch gestrichelte Linien symbolisiert. Nachrichten zwischen A, B und C sind als Pfeile dargestellt.

Im ersten Zeitschlitz verschickt Head A eine Anfrage (Poll) an Client B, der diese mit einem Datenpaket beantwortet. Dabei wird sowohl die Anfrage als auch die Antwort als Broadcast verschickt, das Poll-Paket (mit den Daten des davor befragten Clients) erreicht alle anderen Stationen von Cluster A, die Antwort sehen alle Heads

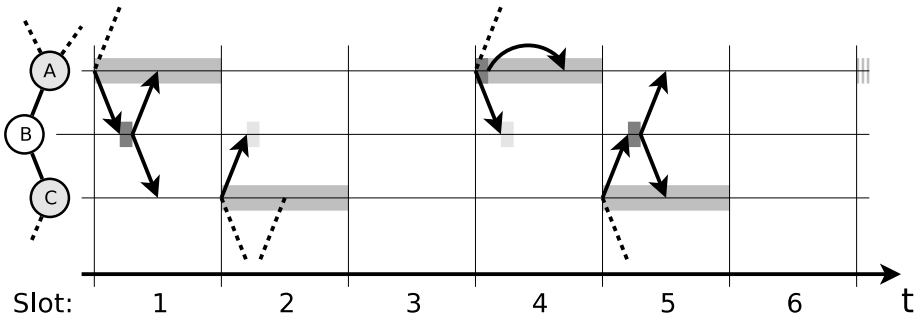

Abbildung 2.2: Zeitschema des Clustering-Protokolls

von Client B (also A und C). Auf diese Weise sieht Head C, dass der Zeitschlitz von einem anderen Cluster beansprucht wird, und verzögert u.u. die eigenen Pakete.

Zeitschlitz 2 wird von Head C benutzt, um einen anderen Client zu befragen. Das Poll-Paket erreicht dabei Client B, der die darin enthaltenen Daten auswertet, aber nicht darauf antwortet.

Der dritte Zeitschlitz wird weder von Cluster A noch von C benutzt und könnte von einem weiteren Cluster verwendet werden. Im vierten Zeitschlitz ist schließlich wieder Head A an der Reihe. Im vorliegenden Beispiel ist es für drei Cluster möglich, am selben Ort zu koexistieren und ein Client kann sich in bis zu drei Cluster einbuchen. Dieser Wert wurde aus Gründen der Anschaulichkeit gewählt, im praktischen Einsatz ist er höher.

Zeitschlitz 4 wird von Head A benutzt, um seine lokale Anwendung zu befragen. Dazu versendet er zunächst ein Poll-Paket, in dem er selbst als Empfänger eingetragen ist. Dieses Paket enthält die Daten, die der Head im Zeitschlitz 1 von Client B erhalten hat, und wird an alle Stationen im Cluster gesendet. Danach generiert die lokale Anwendung von A ein Datenpaket, welches aber nicht über das Medium übertragen wird, sondern nur in die Sende-Puffer des Heads kommt (gebogener Pfeil). Dieses Paket wird vom Head in seinem nächsten Zeitschlitz an alle versendet.

Im nächsten Zeitschlitz, Nr. 5, sendet wieder Head C. Diesmal schickt er ein Poll-Paket mit den zuvor (in Slot 2) empfangenen Daten und fordert gleichzeitig Client B auf, ein Datenpaket zu versenden. In dem Antwortpaket gibt B dann an, ob er das Datenpaket der Vorrunde empfangen hat, und versendet seine eigenen Daten zur weiteren Auslieferung über den Cluster-Head.

Der sechste Zeitschlitz bleibt von den beiden Clustern ungenutzt, danach beginnt wieder Head A.

2.4.3 Bandbreitenreservierung

Anfänglich steht jedem Client in seinem Cluster ein reservierter Zeitschlitz pro Poll-Runde zu. Hat der Head nicht die Maximalanzahl an Clients, werden nach Abfrage der zugesicherten Runden alle Clients im Round-Robin-Verfahren befragt.

Benötigt ein Client für eine Übertragung höhere Bandbreite, so muss er zunächst bestimmen, in welchen seiner Cluster diese Übertragung gehen soll. Danach kann er sich mehrfach in die Poll-Liste des Heads eintragen lassen und dadurch mehrere Zeitschlitze bekommen. Mit der Zusicherung der Zeitschlitze ist auch die Garantie verbunden, dass die so gesendeten Daten an alle Stationen im Cluster ausgeliefert werden. Der effektive Anteil an der Netzwerk-Bandbreite entspricht dabei der Anzahl der reservierten Slots dividiert durch die Anzahl der Slots pro Runde.

Zum Reservieren wartet ein Client das nächste Invite-Paket ab und verschickt eine Benachrichtigung an den Head, dass er eine höhere Slot-Anzahl benötigt. Nach einer Runde kann der Client dann im nächsten Invite-Paket sehen, ob sein Wunsch berücksichtigt wurde und er entsprechend oft eingetragen ist. Ist das der Fall, so wird er in den Runden darauf häufiger befragt und kann mehr Datenpakete versenden.

Die Anzahl der für einen Client verfügbaren Slots ist durch die maximale Anzahl der Clients pro Cluster minus der bereits eingebuchten Clients begrenzt. Können sich bis zu 16 Clients zu einem Head verbinden und sind schon 5 Clients eingebucht, so kann einer dieser Clients (oder der Head) bis zu 11 Slots nutzen (fünf sind für die Clients und einer für den Head reserviert, zusätzlich dazu kann jeder Teilnehmer die zehn freien Zeitschlitze reservieren).

Die Reservierung der Zeitschlitze erfolgt asynchron. Die Anwendung muss dem Clustering-Modul mitteilen, wie viele Slots sie in einem bestimmten Cluster benötigt. Daraufhin versucht der Knoten, die entsprechende Anzahl an Slots beim Head zu reservieren, und informiert die Anwendung über den Erfolg oder Misserfolg.

2.5 Simulationsumgebung

Die Entwicklung eines Kommunikationsprotokolls wird vom Testen der Implementierung begleitet. Die Überprüfung auf einem realen Netzwerk bringt dabei einen hohen Aufwand mit sich, da die aktuelle Softwareversion für alle Rechner zu übersetzen und auf sie zu verteilen ist, wonach die Ergebnisse zur Auswertung wieder eingesammelt und synchronisiert werden müssen. Außerdem ist es schwierig, bei jedem Testlauf identische Ausgangsbedingungen zu erzeugen, um Änderungen im Protokollverhalten besser analysieren zu können.

Um diese Probleme zu umgehen, kann statt einem realen mobilen Netzwerk auch auf eine Simulation eines solchen zurückgegriffen werden. Dabei hat sich der Network Simulator *ns-2* [NS2] als solides Werkzeug für die realitätsnahe Simulation drahtloser

Netze herausgestellt. Ns-2 ist ein diskreter ereignisbasierter Simulator, der in C++ geschrieben wurde und sehr flexibel gesteuert und erweitert werden kann.

Über TCL-Scripte erlaubt er neben der Konfiguration der Anzahl der Knoten auch Einstellungen der Sende-Reichweite, der Signalverluste und anderer Parameter. Außerdem ist es damit möglich, den Ablauf der Simulation genau zu beeinflussen und auf einzelnen Knoten zu bestimmten Zeitpunkten Ereignisse auszulösen.

Des Weiteren generiert ns-2 bei der Ausführung Ergebnisdateien (*trace files*), die sich im Anschluss an die Simulation mit *nam* (dem Network Animator[NAM], Abbildung 2.3) visualisieren lassen. Dabei ist es möglich, eigene Informationen in die Visualisierung einzubinden.

Allerdings besitzt ns-2 eine andere Programmierschnittstelle als die üblichen Netzwerkbetriebssysteme, so dass eine direkte Migration schwierig ist.

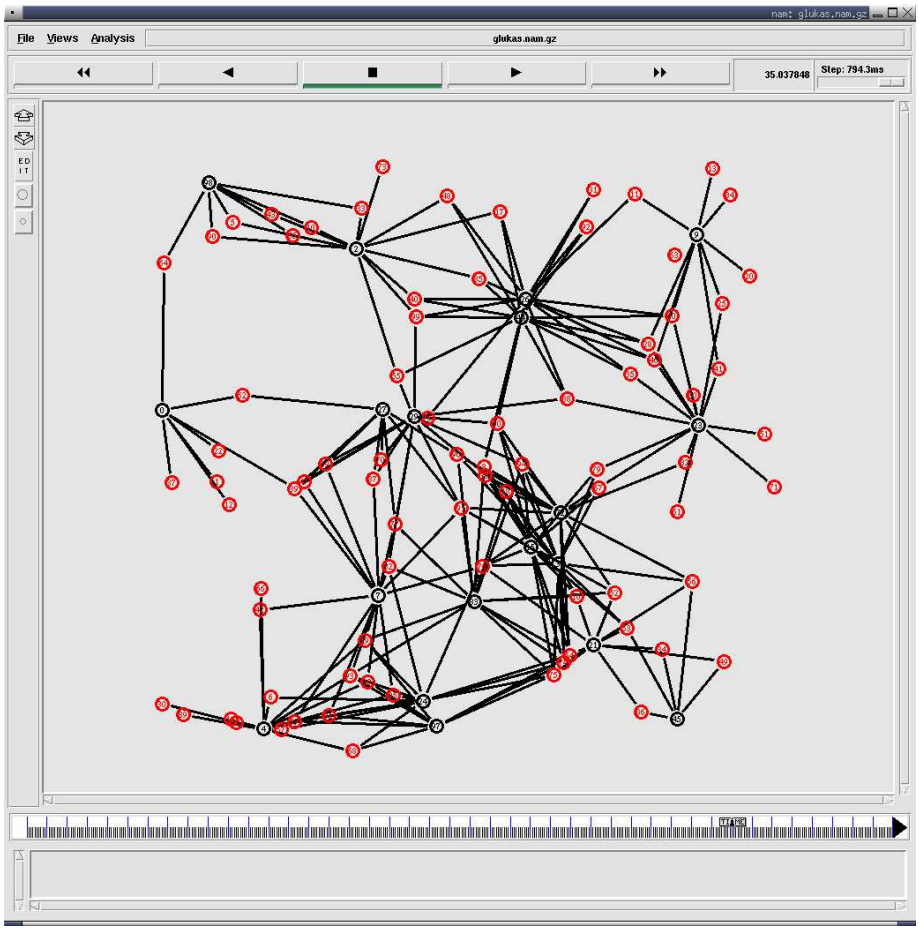

Abbildung 2.3: Network Animator: Simulierte Cluster-Struktur

2.6 Generic Event API

Um dennoch eine gemeinsame Basis für Simulation und Feldtest zu haben, kann auf die Generic Event-based API (GEA) [HM05] als Abstraktionsbibliothek zurückgegriffen werden. Diese Zwischenschicht wurde mit dem Ziel eingeführt, bei der Entwicklung von Netzwerkprotokollen die selbe Code-Basis sowohl in der Simulation als auch bei realen Experimenten einsetzen zu können. Dazu kapselt sie alle Netzwerkfunktionen gegenüber dem Protokoll und seinen Anwendungen ab und setzt diese auf die verwendete Systemschnittstelle um (Abb. 2.4).

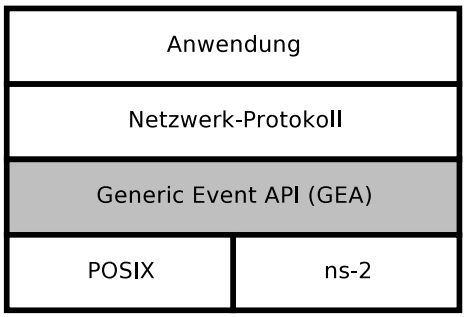

Abbildung 2.4: Systemarchitektur mit GEA

GEA-Anwendungen können ohne Quellcode-Änderung auf den von GEA unterstützten Plattformen eingesetzt werden. Aktuell sind das der Netzwerksimulator ns-2 und Betriebssysteme, die die POSIX[COR]-Netzwerkschnittstelle anbieten. GEA-Anwendungen für diese beiden Plattformen sind sogar binärkompatibel.

Das eigene Protokoll wird dabei in eine dynamisch ladbare Bibliothek gebunden, die zur Laufzeit von einem entsprechenden GEA-Backend geladen wird. Dabei ist es auch möglich, mehrere Protokollteile in unterschiedliche Bibliotheken zu binden, um diese zu unterschiedlichen Zeitpunkten oder nur auf einem Teil der Netzwerkknoten zu laden.

Analog wird auch mit der Anwendung verfahren, die das Protokoll verwendet. Diese muss lediglich nach dem Protokollmodul geladen werden.

GEA-Schnittstelle

GEA arbeitet ereignisorientiert: Es bietet eine Möglichkeit, zeitgesteuerte oder von externen Quellen (repräsentiert über Socket- oder Datei-Handles) ausgelöste Ereignisse zu bearbeiten, indem man dafür Rückruffunktionen definiert. Diese Rückruffunktionen werden von einer zentralen Behandlungsroutine in GEA aufgerufen, sobald das entsprechende Ereignis vorliegt oder wenn die für das jeweilige Ereignis gesetzte maximale Wartezeit überschritten ist.

Die API bietet einen für die Implementierung eines Netzwerkprotokolls erforderlichen Mindestsatz an Methoden. Dazu gehören neben dem Versenden und Empfangen

von Netzwerknachrichten auch Funktionen zum Bestimmen von Zeiten und Zeitdifferenzen, sowie die Möglichkeit, Rückruffunktionen für eigene Ereignisse und für bestimmte Zeitpunkte zu setzen.

Zusätzlich bietet GEA eine Möglichkeit, zwischen einzelnen Modulen Objekt-Referenzen auszutauschen. Dazu dient das *Object Repository*, eine Ablage, in der unterschiedliche Module ihre Objekte hinterlegen können. Diese Objekte können dann von anderen Modulen über ihren Namen und ihren Typen aus dem Object Repository bezogen werden.

Auf der POSIX-Schicht werden die Nachrichten als UDP-Pakete versendet. Obwohl die Verwendung von Paketen der MAC-Schicht für das Implementieren von Routing-Protokollen geeigneter wäre, erfordert die direkte Nutzung der Netzwerkkarte auf Unix-Systemen zusätzliche Benutzerrechte und ist stark plattformabhängig. UDP dagegen bietet auch die notwendigen Funktionen (Versand von Punkt-Zu-Punkt- und Broadcast-Nachrichten) und ist von den Socket-Funktionen der POSIX-API abgedeckt. Der Mehraufwand für die Verarbeitung der UDP-Pakete durch das Betriebssystem ist gleichzeitig so gering, dass er gegenüber dem eigentlichen Protokoll nicht ins Gewicht fällt.

Die Kombination aus ns-2 und GEA bietet damit ein flexibles Gerüst für die Entwicklung und Evaluierung eines Netzwerkprotokolls und erlaubt die einfache Migration von der Simulation auf echte Systeme.

3 Best-Effort Multi-Hop-Routing

Um für Verbindungen innerhalb des mobilen Netzwerks Garantien für Bandbreite und Zuverlässigkeit bieten zu können, müssen alle beteiligten Knoten, also die Gateways auf der Route, ihr Einverständnis dazu erteilen. Dazu müssen sie über die Anforderungen an die Verbindung informiert werden und diese bestätigen können. Dies erfordert einen Multi-Hop-Kommunikationskanal ohne Dienstgütegarantien.

Um einen solchen Kanal anzubieten, müssen die Knoten zum einen beim Versand von Nachrichten kooperieren, also nicht nur die eigenen Pakete versenden, sondern auch die Pakete anderer Stationen weiterleiten. Zum anderen benötigen sie ein Wissen über die Netzwerkstruktur, anhand dessen sie feststellen, ob sie für die Weiterleitung der Pakete überhaupt zuständig sind, und in welche Richtung (bzw. in welchen Cluster) sie die Pakete weiterzuleiten haben.

Dazu wurde im Rahmen dieser Arbeit ein Weltmodell-Propagationsverfahren und ein darauf aufbauendes Best-Effort-Routing-Protokoll entwickelt, welche in diesem Kapitel vorgestellt werden.

3.1 Diskussion möglicher Routing-Verfahren

Die Auswahl des Routing-Protokolls beruht auf mehreren Kriterien, die es möglichst gut zu erfüllen gilt. Zunächst erfordert die Aufgabenstellung, dass das Protokoll auf dem Reliable-Clustering-Modell mit festen Zeitschlitzen aufsetzt. Da keines der in Kapitel 2 vorgestellten Protokolle darauf ausgelegt ist, muss eines davon angepasst oder ein eigenes entwickelt werden. Beim Präzisieren der Auswahl hilft das Klassifizierungsschema für Routing-Protokolle aus Abschnitt 2.2.1.

Scheduling-Prinzip

Wesentliche Anforderungen an das Routing-Protokoll sind die schnelle Zustellung von Datenpaketen an alle möglichen Ziele im Netzwerk und die Möglichkeit, viele Verbindungen zwischen unterschiedlichen Knoten zu führen.

Aus diesen Gründen sind *reaktive* Protokolle, die vor einer Datenübertragung erst den Pfad suchen müssen und damit höhere Aufbaulatenzen und starke Netzwerklast bei vielen unterschiedlichen Verbindungen mit sich bringen, nicht sinnvoll.

Hybride Protokolle erlauben den schnellen Pfadaufbau zu Knoten innerhalb der eigenen Gruppe bzw. Zone, aber nicht zu allen Stationen im Netzwerk. Daher sind auch sie nicht geeignet.

Ein *proaktives* Routing-Protokoll dagegen führt auf jedem Knoten die nötigen Informationen, um schnell an beliebige andere Knoten Daten zu verschicken, so dass die Wahl darauf fällt. Die höhere Belastung des Netzwerks durch die Propagation der Routing-Daten fällt dagegen aus zwei Gründen nicht ins Gewicht: Zum einen finden, bedingt durch das Cluster-Schema, bereits permanent Datenübertragungen statt, die man ohne Beeinträchtigung des Betriebs für die Propagation nutzen kann. Zum anderen ist die Netzwerklast bei einem reaktiven Verfahren höher, wenn dieses immer viele Verbindungen durch das Netz aufzubauen hat.

Zustandsinformationen

Ein Routing-Protokoll, welches nur Ziel-Vektoren (*distance-vector*; Abstand und Gateway zu jedem Ziel-Knoten) austauscht, ist bandbreitenschonener als ein Verfahren, das auf vollständigen Nachbarschaftsbeziehungen (*link-state*) beruht.

Allerdings haben ziel-basierte Verfahren nicht nur mit dem Count-To-Infinity-Problem zu kämpfen, sondern erlauben es auch nicht, unterschiedliche Pfade zu einem bestimmten Ziel zu bestimmen.

Da die Verwendung alternativer Pfade für das später darauf aufsetzende QoS-Routing relevant ist, kommen also nur Link-State-Verfahren in Frage.

Struktur

Das Cluster-Schema gibt auf den ersten Blick bereits eine Struktur für das Routing-Protokoll vor. Allerdings sind alle Knoten gleichberechtigt, was ihre Sendemöglichkeiten angeht. Da alle Clients von der Clustering-Schicht nur über die ihnen benachbarten Heads, nicht aber über die anderen Clients in ihren Clustern, und alle Heads jeweils nur über ihre Clients informiert werden, kann hier auch eine flache Sichtweise auf die Netzwerktopologie angewandt werden. So muss lediglich beachtet werden, dass jeder zweite Knoten auf einem Pfad ein Gateway ist, während die anderen in ihrer Funktion als Cluster-Head nicht aktiv am Routing teilnehmen.

Auf diese Weise lässt sich mit einer flachen Weltmodell-Struktur ohne Beeinflussung der Arbeitsweise Datenverkehr einsparen (da weniger Struktur-Informationen zu übertragen sind) und das Routing-Konzept kann leichter auf andere, nicht clusterbasierte, MAC-Schichten portiert werden.

Auswahl des Verfahrens

Aus den in Kapitel 2 vorgestellten Protokollen sind nur OLSR und FSR proaktiv und tauschen Link-State-Daten aus. Da OLSR jedoch seine eigene Struktur verwendet, die in Kombination mit der Cluster-Struktur die Komplexität deutlich steigern

würde, und weil die Periodizität der Updates eigenen Mechanismen unterliegt und nicht so flexibel geregelt werden kann wie bei FSR, wurde Fisheye State Routing als Grundlage für die weitere Arbeit herangezogen.

Ein weiteres Argument dafür ist, dass FSR sowohl mit steigender Knotenanzahl als auch mit hoher Mobilität gut skaliert.

Da die bestehenden Implementierungen von FSR entweder auf einen anderen Simulator ausgelegt sind ([Bec]) oder Linux erfordern ([HOR]) und für die Kooperation mit GEA und dem Clustering-Schema stark abgewandelt werden müssten, wurde das Verfahren statt dessen ausgehend von der Entwicklungsumgebung und den aus dem Clustering resultierenden Anforderungen neu implementiert.

Die resultierende Implementierung sorgt dafür, dass die Informationen über die unmittelbare Nachbarschaft sehr präzise sind, während man von weiter entfernten Knoten eine ältere Topologie-Sicht hat, die zwar nicht die letzten Änderungen der Konnektivität berücksichtigt, dafür aber immer noch genügt, um zu bestimmen, welches Gateway für welches Ziel zuständig ist. Da die Gateways näher zum Ziel auch immer präzisere Informationen über die verbleibende Strecke haben, können sie das Paket zunehmend genauer zum Empfänger navigieren.

Besonderheiten

Abweichend vom originalen FSR-Entwurf wurde die Weltmodell-Propagation dem Zeitschema des Clustering-Protokolls angepasst: Da jeder Knoten regelmäßig von einem seiner Heads nach einem zu versendenden Datenpaket gefragt wird und auch eins versenden muss, aber nicht immer Daten dafür bereithält, werden diese ansonsten ungenutzten Zeitschlitze für den Versand von Topologiedaten eingesetzt.

Auf diese Weise wird das Medium optimal ausgelastet und die Aktualität des Weltmodells ist gut, ohne dass dafür normaler Datenverkehr zurückstehen muss. Zudem braucht die Routing-Schicht keine eigenen periodischen Updates mehr zu generieren, da ihr das vom MAC-Protokoll abgenommen wird.

3.2 Grundlegendes Konzept

Jeder Knoten ist zunächst nur in der Lage, seine unmittelbare Nachbarschaft zu sehen (Abb. 3.1, Schritt 1, unterlegte Fläche). Um mit weiter entfernten Knoten kommunizieren zu können, muss er Informationen über die Pfade zu diesen Knoten beziehen und entsprechend auch seine eigenen Informationen weitergeben, um erreichbar zu sein. All diese Informationen fasst ein Knoten dann in seinem lokalen Weltmodell – einer möglichst getreuen Abbildung der Netzwerkstruktur – zusammen und versendet Änderungen des Weltmodells an seine Nachbarn, die diese wiederum abspeichern und weitergeben. So erhält er erst ein Bild über die Umgebung seiner

unmittelbaren Nachbarn (Schritt 2), dann über die Drei-Hop-Nachbarn (Schritt 3) und irgendwann schließlich über das gesamte Netzwerk (Schritt 4).

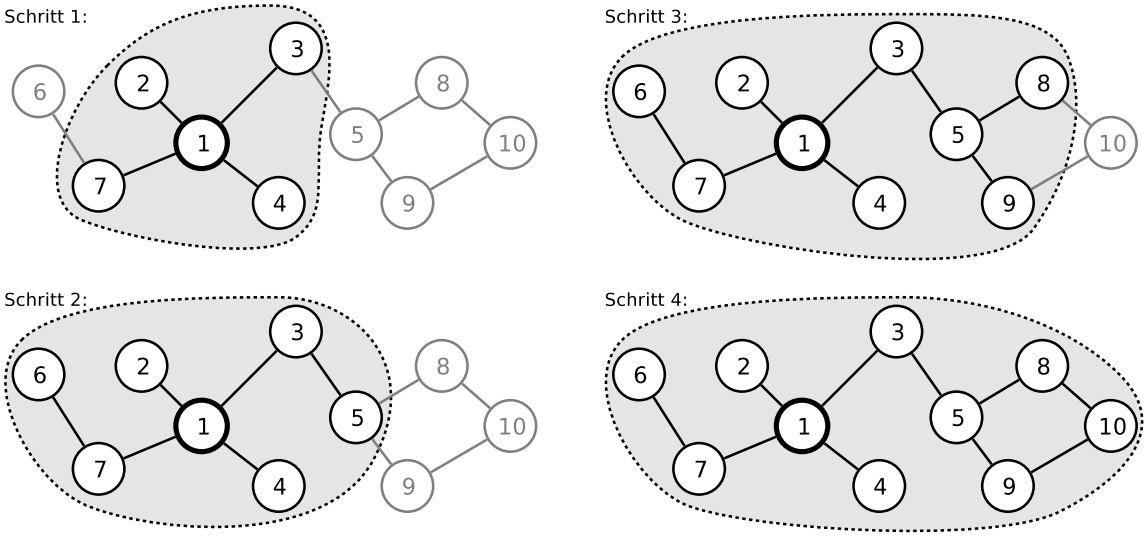

Abbildung 3.1: Sukzessive Sichtweise von Knoten 1 auf seine Nachbarschaft

Bleibt das Netzwerk stabil, erhalten alle Stationen nach einer bestimmten Zeit ein konsistentes Bild über die gesamte Topologie. Ändern sich die Verbindungen eines Knotens, so gibt er die neuen Informationen an seine direkten Nachbarn weiter, die diese nach und nach durch das gesamte Netzwerk propagieren, bis wieder alle ein konsistentes Abbild haben. Bewegen sich viele Knoten, so erreichen die aktuellen Informationen die unmittelbare Nachbarschaft. Die Topologiedaten werden nicht mehr zeitnah an weiter entfernte Knoten übertragen, was jedoch nach dem Fish-Eye-Prinzip unproblematisch ist.

Mit Hilfe des so kommunizierten Weltmodells kann ein Knoten Pfade zu allen anderen Stationen im Netzwerk bestimmen. Auf diesen Pfaden können dann die eigenen Anwendungen Nachrichten versenden. Zusätzlich wirkt jeder Knoten auch als Gateway: Empfängt er Pakete von seinen Nachbarn, die weitergeleitet werden sollen, so muss er dies entsprechend den berechneten Pfaden tun.

3.3 Struktur des Multi-Hop-Routing-Systems

Um Routing zwischen Clustern zu ermöglichen, müssen mehrere Module miteinander interagieren (Abb. 3.2). Zum einen muss das Weltmodell aktuell gehalten werden. Dies geschieht, indem das Weltmodell-Modul Daten über die eigene Nachbarschaft von der MAC-Schicht bezieht und diese Informationen an die Nachbarn sendet, sowie solche Informationen von den Nachbarn empfängt und an andere Nachbarn weiterleitet.

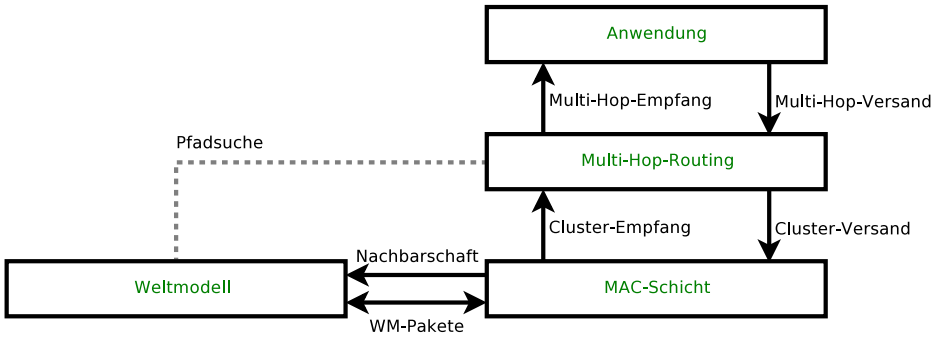

Abbildung 3.2: Konzept-Struktogramm Multi-Hop-Routing

Zum anderen muss der Knoten auch eine Aufgabe als Gateway wahrnehmen. Dazu muss er empfangene Pakete untersuchen, überprüfen ob er für die Weiterleitung zuständig ist und gegebenenfalls den weiteren Pfad berechnen und das Paket entsprechend verschicken. Analog muss er bei Paketen vorgehen, die von der lokal laufenden Anwendung verschickt werden oder an sie gerichtet sind. Die beiden Teilmodule „Weltmodell" und „Multi-Hop-Routing" sollen im Folgenden detailliert vorgestellt werden.

3.4 Weltmodell

Das Weltmodell ist in Unterkomponenten aufgeteilt (Abb. 3.3). Eine zentrale Stelle nimmt dabei der Topologiespeicher ein, in dem alle Informationen über die Netzwerkstruktur abgelegt sind. Diesen Speicher benutzt der Pfadsucher, wenn er von der Routing-Schicht damit beauftragt wird, das Gateway für einen bestimmten Knoten ausfindig zu machen.

Der Speicher wird zum einen dadurch gefüllt, dass die MAC-Schicht über eine Überwachungsschnittstelle Informationen über die eigenen Nachbarn liefert. Zum anderen wertet das Weltmodell ankommende Datenpakete mit den Nachbarschaftsbeziehungen anderer Knoten aus.

Wenn die Station von ihrem Head mit einem Poll-Paket abgefragt wird und die Anwendung keine Daten zu versenden hat, so füllt das Weltmodell aus seinem Speicher ein Datenpaket mit Topologiedaten, welches dann an die Nachbarn propagiert wird.

3.4.1 Aufbau des Weltmodells

Um Routen berechnen zu können, muss jeder Knoten die Topologie des gesamten Netzwerks kennen. Da ein einzelner Knoten jedoch keine fremden Verbindungen einsehen kann, muss er sich darauf verlassen, die Topologiedaten anderer Knoten über

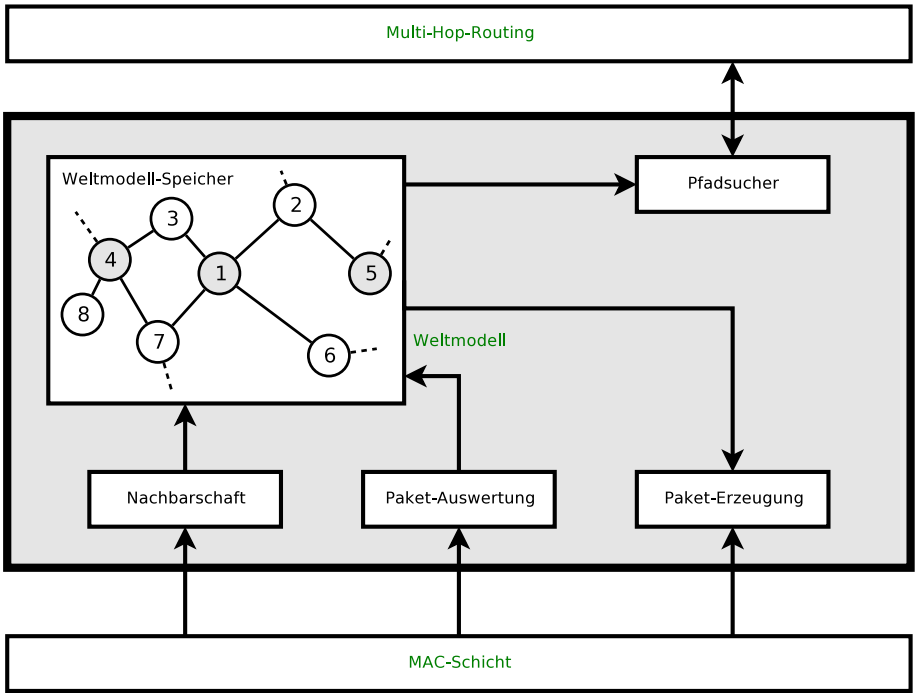

Abbildung 3.3: Konzept-Struktogramm Weltmodell

seine Nachbarn zu beziehen – und im Gegenzug dazu seine eigenen Informationen an die restlichen Teilnehmer weitergeben.

Da solche Daten auf unterschiedlich langen Wegen und durch die passive Ausbreitungsart mit starken Verzögerungen zu einem Knoten gelangen können, muss dieser entscheiden, ob er aktuellere Informationen erhalten hat, als er schon besitzt. Dies wird dadurch realisiert, dass jeder Knoten eine Sequenznummer für seine Nachbarschaftsbeziehungen führt und diese bei jeder Änderung seiner direkten Nachbarschaft erhöht. Diese Sequenznummer wird dann zusammen mit der aktuellen Liste seiner Nachbarn übertragen. Ein empfangender Knoten kann daran sehen, ob sich etwas geändert hat, und ob die Information, die er erhalten hat, aktuell ist.

Obwohl die Links zwischen zwei Knoten immer ungerichtet sind, werden sie im Weltmodell als zwei gerichtete Verbindungen abgebildet (Abb. 3.4, die Zahlen rechts unten in jedem Knoten sind die jeweiligen Topologie-Sequenznummern). Diese Herangehensweise erlaubt eine schnellere Pfadsuche im abgespeicherten Graphen (man erkennt bei jedem Knoten unmittelbar, zu wem dieser Verbindungen hat).

Des Weiteren umgeht man damit das Problem, dass eine Verbindung immer zwischen zwei Knoten besteht, die unterschiedliche Sequenznummern führen. Würden Verbindungen nur als ungerichtet übermittelt werden, so könnten Inkonsistenzen im Graphen auftreten, wenn Knoten A angibt, eine Verbindung zu Knoten B zu haben, während die letzte Information von Knoten B keine Verbindung zu A aufweist. Da

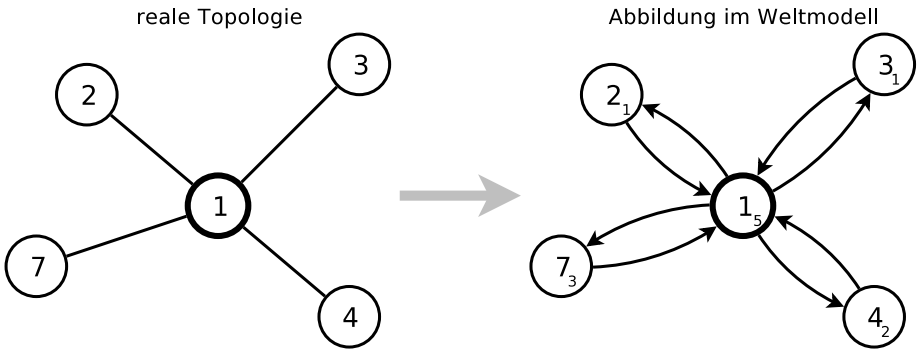

Abbildung 3.4: Reale Topologie und ihre Abbildung

die Sequenznummern von A und B nicht korrelieren, kann ein Außenstehender nicht beurteilen, welche der beiden Aussagen aktueller ist.

Wird eine Verbindung dagegen als ein Paar gerichteter Strecken abgelegt, so stellt die Information, dass A mit B verbunden ist, aber B nicht mit A, keine Verletzung des Modells dar. Obwohl dieser Zustand im realen Netz nicht existiert, wird er im Topologiespeicher so abgelegt, bis es von A bzw. B aktualisierte Informationen gibt und die Strecke wieder konsistent ist.

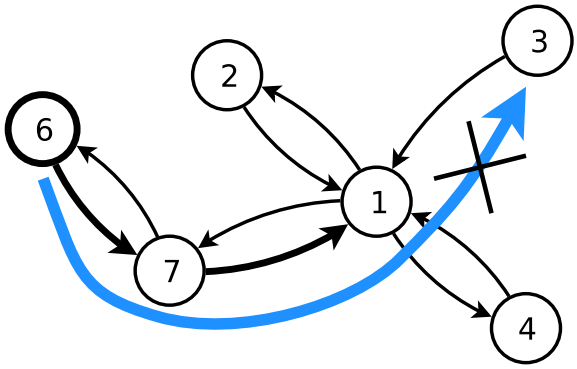

Abbildung 3.5: Pfadsuche nach dem Weltmodell von Knoten 6

Diese Herangehensweise erhöht zwar leicht den Netzwerk-Overhead, sie sorgt aber ansonsten nicht für eine Verschlechterung des Routing-Verhaltens – wird das Netz partitioniert, so stellen die Knoten auf der einen Seite fest, dass sie keine Verbindung zur anderen haben. Auch wenn in ihrem Weltmodell verzeichnet ist, dass die andere Seite eine Verbindung zu ihnen hat, ist das für das Routing nicht relevant (Abb. 3.5: Knoten 6 versucht erfolglos eine Verbindung zu Knoten 3 aufzubauen). Wenn die neuen Topologiedaten des isolierten Knotens dann das restliche Netz erreichen, verschwindet auch der veraltete Link in die Gegenrichtung.

3.4.2 Speicherung des Weltmodells

Das Weltmodell ist ein gerichteter Graph, dessen Knoten die Stationen sind und dessen Kanten direkte Verbindungen zwischen Stationen darstellen. Es wird in Form einer Adjazenzliste abgelegt, wobei es hilfreich ist, die Datenstruktur auf wahlfreien Zugriff über Knotennummern zu optimieren. Diese Zugriffsart wird für die Pfadsuche und für Aktualisierungen benötigt wird und ist daher performancekritisch.

Zu jedem Knoten wird im Speicher die zuletzt empfangene Liste seiner Nachbarn und die dazugehörige Sequenz-Nummer abgelegt – diese beiden Informationen bilden eine Einheit und werden immer zusammen ausgesendet oder aktualisiert.

Ist eine solche Struktur zur Speicherung der Knoten einmal vorhanden, kann sie auch für die Zwischenspeicherung der lokal berechneten Routen benutzt werden. Dies hat den Vorteil, dass Pfade nicht für jede Suche neu berechnet werden müssen, sondern erst dann, wenn sich das Weltmodell geändert hat.

Als Suchverfahren wird der Algorithmus von Dijkstra[Dij59] verwendet, da im Modell keine negativen Kantengewichte auftreten und die Dijkstra-Laufzeit bei n Knoten und m Kanten mit $\mathcal{O}(m + n \log n)$ gut ist.

Aus der Pfad-Berechnung ergeben sich zu jedem Knoten mehrere Informationen, die im Weltmodell abgespeichert werden. Zunächst einmal ist das der Abstand von der eigenen Station zum Zielknoten, gemessen in Hops. Des Weiteren gehört dazu der Next-Hop, also der eigene Nachbar, der dem Ziel am nächsten liegt. Zuletzt enthält die Liste noch den Vorgänger des Zielknotens. Dieser wird abgelegt, um ausgehend vom Ziel den Pfad rekonstruieren zu können.

Knoten	*Nachbarn*	*Seq-Nr.*	*Abstand*	*Next-Hop*	*Vorgänger*
6	[7]	1	0	-	-
7	[1, **6**]	4	1	7	**6**
1	[2, 3, 4, 7]	3	2	7	7
2	[1]	6	3	7	1
4	[1]	3	3	7	1
3	[1]	4	∞	-	-

Tabelle 3.1: Weltmodell und Routing-Cache von Knoten 6

In Tabelle 3.1 ist das Weltmodell von Knoten 6 aus Abb. 3.5 dargestellt. Der Knoten führt auch die Informationen über sich selbst im Weltmodell. Von diesen ausgehend berechnet er dann nach und nach den Routing-Cache. Die Berechnung ergibt, dass er alle anderen Knoten (bis auf 3, der gar nicht erreichbar ist) über seinen Nachbarn 7 als Gateway erreichen kann. Wird der komplette Pfad, z.B. zu Knoten 4 gesucht, so muss dieser zurückverfolgt werden: Der Vorgänger von 4 ist 1, der Vorgänger von 1 ist 7, und davor kommt 6, umgedreht ist der korrekte Pfad also: $6 \rightarrow 7 \rightarrow 1 \rightarrow 4$.

3.4.3 Propagation des Weltmodells

Da jeder Knoten durch seine Cluster-Zugehörigkeit seine direkten Nachbarn kennt, kann er diese Informationen als Grundlage für den Aufbau seines Weltmodells heranziehen. Auf Anforderung der MAC-Schicht generiert er dann ein Datenpaket, welches mit seiner lokalen Sicht gefüllt ist. Dabei werden bevorzugt neue Informationen über den eigenen bzw. über näher gelegene Knoten eingebettet, und zu einem Knoten seine Sequenznummer und die komplette Liste seiner Nachbarn abgelegt.

In einem Datenpaket befinden sich die Nachbarschaftsbeziehungen von so vielen Knoten, wie es der Speicherplatz zulässt. Da ab einer bestimmten Netzwerkgröße nicht mehr alle Knoten in einem Datenpaket untergebracht werden können, muss ausgewählt werden, welche Daten zu versenden sind und welche nicht.

Dazu werden zwei Warteschlangen geführt, eine für aktualisierte Nachbarschaftsdaten und eine für die, die seit ihrem letzten Versenden nicht verändert wurden. Jedes Mal, wenn der Knoten neue Daten von einer anderen Station empfängt, wird diese Station aus der zweiten in die erste Warteschlange übertragen, sofern sie nicht bereits darin steht.

Zum Versenden wird zunächst überprüft, ob sich die eigene Nachbarschaft geändert hat. Falls ja, wird die eigene Adjazenzliste als erstes in das Paket geschrieben.

Danach wird die Warteschlange mit aktualisierten Daten abgearbeitet und die entsprechenden Knoten-Nachbarn in das Paket geschrieben, bis entweder die Warteschlange leer oder das Paket voll ist. Dabei wandern alle Knoten, die im Paket erfasst wurden, an das Ende der zweiten Warteschlange.

Ist im Paket noch Platz, so wird daraufhin die zweite Warteschlange abgearbeitet, bis entweder das gesamte Weltmodell erfasst ist (was bei kleinen Netzen der Fall ist), oder bis das Datenpaket keine Knoten mehr aufnehmen kann.

Konnten keine Datenpakete aus der unprivilegierten Warteschlange aufgenommen werden, so wird das erste Element daraus an die „Geändert"-Warteschlange angehängt, um ein „Verhungern" der weniger wichtigen Daten zu verhindern.

3.5 Multi-Hop-Übertragung von Paketen

Die clusterübergreifende Übertragung von Daten erfordert die Kooperation der Teilnehmer im mobilen Netzwerk. Dazu muss jeder Knoten mehrere Komponenten bereitstellen, die im Folgenden vorgestellt werden und deren Zusammenspiel in Abbildung 3.6 dargestellt ist.

Will ein Knoten ein Datenpaket an einen beliebigen anderen Knoten im gesamten Netz übertragen, so muss er mit Hilfe des Weltmodells die Route zum Empfänger bestimmen. Für den Versand wird dabei der Gateway-Knoten aus den eigenen Clustern ermittelt, der dem Ziel am nächsten ist. Das Datenpaket wird in eine Warteschlange gesetzt, um bei der nächsten Anfrage des Ziel-Heads versendet zu werden.

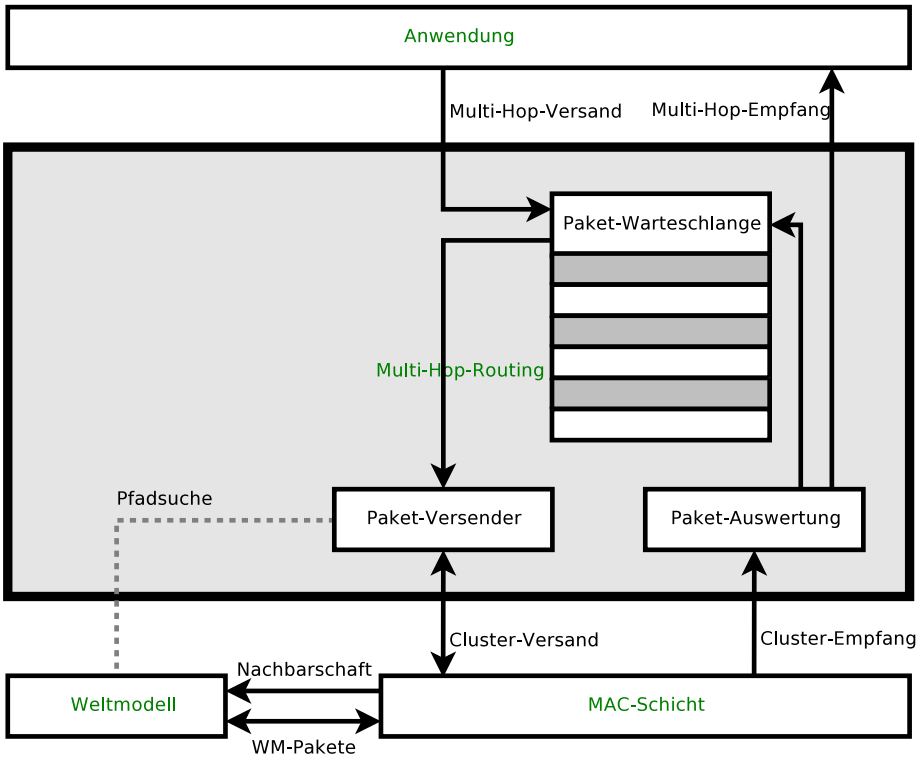

Abbildung 3.6: Konzept-Struktogramm Multi-Hop-Routing

3.5.1 Paket-Warteschlange

Um die Zwischenspeicherung zu realisieren ist es zunächst naheliegend, eine Paketwarteschlange pro Cluster einsetzen. Da sich jedoch die Topologie ändern kann, während ein Datenpaket in der Warteschlange verbleibt, ist es besser, die Route für alle wartenden Pakete anhand der Ziel-Adressen neu zu bestimmen, wenn diese tatsächlich benötigt wird, also wenn der Knoten aufgefordert wird, ein Paket zu versenden.

Deswegen wird nur eine zentrale Warteschlange eingesetzt, die bei jeder Paketanforderung untersucht wird. Dabei wird für jedes Paket, das sich dort befindet, der nächste Gateway-Knoten und der Cluster, in dem sich dieses Gateway befindet, bestimmt. Stimmt der ermittelte Cluster mit dem überein, aus dem die Paket-Anfrage stammt, so wird das Paket ausgeliefert und aus der Warteschlange gelöscht.

Zusätzlich muss aber auch dafür gesorgt werden, dass Datenpakete, für die keine Route mehr existiert, nicht ewig in der Warteschlange verbleiben. Da nicht vorhergesagt werden kann, wann der Pfad zu einem Knoten wieder verfügbar ist und ob man das Ziel überhaupt erreichen können wird, werden alle solchen Pakete gelöscht.

Die Warteschlange wird sowohl für von der Anwendung erzeugte Pakete als auch für die Datagramme verwendet, die der Knoten von seinen Nachbarn empfängt und

für deren Weiterleitung er zuständig ist. Die Pakete werden dort in Empfangsreihenfolge eingetragen und nach einem bei den ältesten Paketen beginnenden Auswahlverfahren wieder entfernt.

3.5.2 Paket-Auswertung

Innerhalb eines Clusters wird ein Datenpaket zuverlässig an alle Stationen ausgeliefert. Obwohl für das Multi-Hop-Routing nur die Auslieferung an das Gateway relevant ist, sieht das Clustering-Schema bisher keine Möglichkeit für die Einschränkung der Empfängerliste vor.

Da ein Datenpaket von allen Knoten im Cluster empfangen wird, müssen diese zunächst feststellen, ob sie für dessen Weiterleitung zuständig sind. Dazu muss im Paket-Header neben dem Empfänger auch das nächste Gateway vermerkt werden. Erhält ein Knoten ein Paket, in dem eine andere Station als Gateway eingetragen ist, so wird das Paket verworfen. Erkennt der Knoten, dass er zuständig ist, so überprüft er das Ziel-Feld im Paket. Ist das Paket an ihn gerichtet, reicht er es an die Anwendung weiter. Wird ein anderer Knoten adressiert, so wird das Paket an die Warteschlange angehängt.

3.5.3 Paket-Versand

Wird ein Knoten von einem seiner Cluster-Heads aufgefordert, Daten zu verschicken, so erhält das Multi-Hop-Modul eine Anfrage von der MAC-Schicht und überprüft seine Paket-Warteschlange.

Ist die Warteschlange leer, signalisiert die Multi-Hop-Schicht das entsprechend und das Paket wird stattdessen vom Weltmodell mit Topologie-Informationen gefüllt.

Abarbeitung der Warteschlange

Enthält die Warteschlange Datenpakete, so wird das Weltmodell zunächst aufgefordert, alle Pfade im Routing-Zwischenspeicher neu zu berechnen. Dies geschieht nur, wenn seit der letzten Berechnung Änderungen an der Topologie stattgefunden haben.

Liegen dann aktuelle Routen vor, so geht der Paket-Versender die Warteschlange von vorn durch und holt zu jedem Paket den Ziel-Cluster aus dem Routing-Speicher des Weltmodells. Entspricht der Ziel-Cluster für das Paket dem Cluster, an den das nächste Paket gehen soll, so wird das Gateway aus dem Routing-Cache in das Paket eingetragen, und es der MAC-Schicht zum Versand übergeben.

Nutzung des Weltmodells

Da das Weltmodell eine flache Struktur hat, unterscheidet es nicht zwischen Heads und Clients. Um also den Ziel-Cluster und das nächste Gateway zu bestimmen, muss der Zustand des eigenen Knotens in Betracht gezogen werden.

Ist der eigene Knoten ein Head, so werden alle Datenpakete in den eigenen Cluster verschickt. Der Next-Hop des Weltmodells ist dabei das Gateway, das ins Paket einzutragen ist, da ein Head immer nur mit Clients verbunden sein kann, und nur Clients eine Rolle als Gateway wahrnehmen können.

Ist der Knoten dagegen ein Client, so sind seine unmittelbaren Nachbarn Heads, und der Next-Hop aus dem Weltmodell entspricht dem Cluster, in den das Paket verschickt werden muss. Das Gateway im Ziel-Cluster ist dann aus Sicht des Weltmodells der übernächste Hop, auf den man aber nicht unmittelbaren Zugriff hat. Aus diesem Grund wurde der Routing-Speicher des Weltmodells um ein weiteres Feld für den übernächsten Hop erweitert, um zusätzlichen Berechnungsaufwand für den Knoten zu ersparen.

Entsprechend vergleicht der Client den Next-Hop für den Paket-Empfänger mit der Cluster-Id, aus der die Poll-Anfrage kommt. Stimmen diese überein, so wird der übernächste Hop aus dem Weltmodell als Gateway in das Paket eingetragen und das Paket an die MAC-Schicht gegeben. Auch hier ist durch die Cluster-Struktur sichergestellt, dass der übernächste Knoten ein Client ist und damit ein Gateway für das Paket sein kann (Client → Head → Client).

Routing-Schleifen

Da jeder Knoten sein eigenes Weltmodell hat, kann es vorkommen, dass zwei Gateways der Meinung sind, der jeweils andere würde für ein bestimmtes Ziel zuständig sein. Da Source-Routing, die klassische Lösung für das Routing-Schleifen-Problem, nicht mit dem Fish-Eye-Prinzip kombinierbar ist, muss ein anderes Mittel gewählt werden. Für Best-Effort-Pakete wurde dabei eine TTL, bzw. ein Hop-Zähler, implementiert, der vom Sender auf einen bestimmten Wert gesetzt und von jedem Gateway dekrementiert wird. Erreicht die TTL den Wert 0, so wird das Paket stillschweigend verworfen.

3.6 Zuverlässigkeit

Obwohl die Datenübertragung innerhalb jedes einzelnen Clusters zuverlässig abläuft (reliable broadcast an alle Stationen), können die Garantien nicht auf Verbindungen über Cluster-Grenzen hinweg ausgedehnt werden. Es gibt mehrere mögliche Ursachen dafür, dass ein Multi-Hop-Datenpaket sein Ziel nicht erreicht.

Zum einen ist es möglich, dass das Paket aufgrund von veralteten Topologiedaten verschickt wird und es keine reale Verbindung mehr zum Zielknoten gibt, weil dieser ausgefallen ist oder sich zu weit vom Kernnetz entfernt hat. In dem Fall wird das Paket auf der Station verworfen, deren Weltmodell keine Route mehr zum Empfänger hat.

Zum anderen kann es passieren, dass die TTL des Datenpakets vor Erreichen des Ziels auf 0 sinkt und das Paket deswegen verworfen wird. Die dafür möglichen Ursachen sind entweder eine Routing-Schleife oder ein Pfad, der länger ist als vom Sender berechnet. Die Wahl der TTL bildet dabei einen Kompromiss zwischen der Erreichbarkeit bei verlängerten Routen und der unnötigen Netzwerkbelastung bei Routing-Schleifen.

3.7 Implementierung

Im folgenden Abschnitt wird die Implementierung des Weltmodells und der Multi-Hop-Schicht detailliert vorgestellt und es wird auf ihre Komponenten und deren Zusammenarbeit miteinander, mit der Anwendung und mit der MAC-Schicht eingegangen.

3.7.1 Verwendung der MAC-API

Die Routing-Schicht arbeitet vollständig ereignisorientiert. Sie reagiert auf Ereignisse, die ihr von der unteren Schicht (MAC) oder von der höheren Schicht (Anwendung) signalisiert werden, und leitet diese Ereignisse bei Bedarf weiter.

Die MAC-Schicht stellt drei Ereignis-Typen bereit, für die sich die Routing-Schicht registrieren kann und die zum Empfang und Versand von Paketen dienen:

- `recvHooks` – Diese Rückruffunktion wird beim Empfang eines an den eigenen Knoten adressierten Pakets ausgeführt. Sie erhält dabei eine Referenz auf das Paket, das empfangen wurde.
- `promiscHooks` – Über diese Schnittstelle reicht die MAC-Schicht alle empfangenen Pakete unabhängig von ihrem Adressaten zur Anwendung. Sie entspricht dem *promiscuous mode* normaler Netzwerkkarten.
- `fillpacketHooks` – Hier signalisiert die MAC-Schicht, dass der Cluster-Head nach einem Paket gefragt hat. Dabei wird eine Referenz auf das mit Daten zu füllende Paket übergeben.

3.7.2 Rückruffunktionen und Hooks

Um Ereignisse zwischen unterschiedlichen Modulen signalisieren zu können, wird eine einheitliche Schnittstelle benötigt. Eine solche Schnittstelle kann mit Hilfe von Rückruffunktionen implementiert werden. Dazu übergibt ein Modul, welches über ein Ereignis benachrichtigt werden will, einen Funktionszeiger und einen Datenzeiger an das ereignisauslösende Modul. Wenn das Ereignis stattfindet, wird die Funktion mit dem Datenzeiger und Informationen über das Ereignis als Parametern aufgerufen

und kann darauf reagieren. Der Datenzeiger dient dabei der Rückruffunktion dazu, den Kontext der Registrierung zu rekonstruieren.

Da die Implementierung von Rückruffunktionen in Klassen bei den meisten C++-Compilern nicht sauber gelöst ist, werden statische Funktionen statt Methoden für Rückrufe eingesetzt und der Datenzeiger zum Abspeichern der Instanz verwendet, die den Rückruf registriert hat. Über diesen Zeiger können dann die Methoden des Objekts zur Ereignis-Behandlung aufgerufen werden.

Damit mehrere unterschiedliche Objekte auf ein Ereignis reagieren können, werden zur Ereignis-Mitteilung *Hooks* eingesetzt. Ein Hook ist ein Hilfsobjekt, das eine Liste der Rückruffunktionen zu einem Ereignis, sowie die den Funktionen zugehörigen Objektreferenzen führt. Wird das Ereignis ausgelöst, werden die Funktionen aus der Liste nacheinander ausgeführt, bis eine erfolgreiche Bearbeitung gemeldet wurde.

Es existieren zwei Hook-Typen – der erste ruft die Rückruffunktionen in der Reihenfolge auf, in der sie eingetragen wurden, der zweite richtet sich nach deren Priorität. Letzteres wird z.B. verwendet, um beim Versand von Paketen QoS vor BestEffort und BestEffort vor Topologiedaten zu stellen.

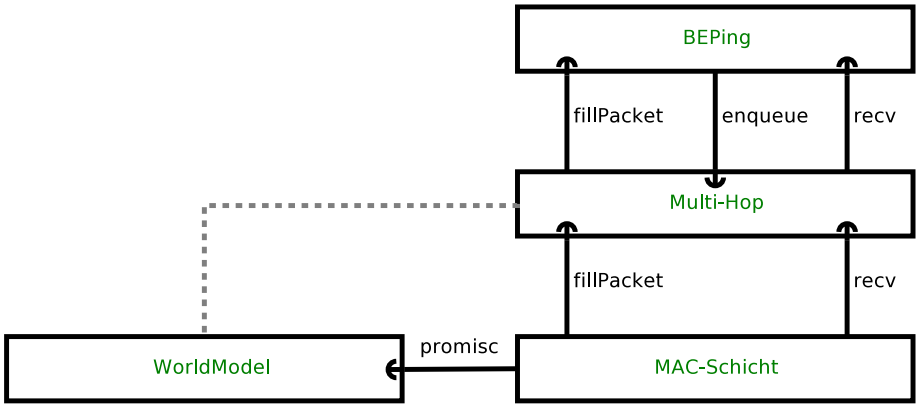

Abbildung 3.7: Interaktion zwischen den Modulen mittels Hooks

Abbildung 3.7 zeigt die Hooks, die zwischen der MAC-Schicht, der Multi-Hop-Middleware und deren Anwendung zum Tragen kommen. Sie werden zur Übergabe von Datenpaketen zur Auswertung und zum Befüllen mit Daten eingesetzt.

3.7.3 Knoten-Adressen

Jede Station im mobilen Ad-Hoc-Netz besitzt eine im Voraus festgelegte eindeutige Kennung, die in Datenfeldern des Typs `NodeId` abgespeichert wird. Der genaue Aufbau dieser Kennungen ist für das Routing-Protokoll nicht relevant, da sich aus ihnen wegen der Knotenmobilität keine Informationen zur Pfadsuche bestimmen lassen. In der Implementierung werden IP-Adressen als Knotennummern verwendet, die von GEA für den Versand und Empfang von UDP-Paketen eingesetzt werden.

Die Rechneradressen müssen dafür im Voraus vergeben werden. Denkbar wäre auch die Verwendung von MAC-Adressen, da diese bereits eindeutig sind, dazu müsste lediglich der Datentyp von `NodeId` angepasst werden.

3.7.4 Datenpakete

Damit Daten über das Netzwerk versendet werden können, müssen diese in Pakete verpackt werden. Dabei sind einige Besonderheiten zu beachten, wie z.B. die Byte-Reihenfolge von 16-Bit- und 32-Bit-Zahlen auf unterschiedlichen Plattformen. So speichern u.a. x86-Systeme das niederwertigste Byte eines größeren Datentyps zuerst (Little Endian), während Sparc- und PowerPC-Prozessoren mit dem höchstwertigen Byte beginnen (Big Endian). Im Netzwerk werden alle Datentypen als Big Endian übertragen, so dass CPU-abhängig Konvertierungen durchgeführt werden müssen. Manche Architekturen erlauben es außerdem nicht, 16- und 32-Bit-Felder an beliebigen Speicheradressen zu lesen und zu schreiben, sondern nur an Adressen, die durch die Feldgröße teilbar sind.

Damit die dafür nötige Kapselung möglichst transparent geschieht, wurden für unterschiedliche Pakettypen mehrere Klassen eingeführt. Dabei bildet `BcPacket` die Basisklasse, die von der MAC-Schicht verwendet und bei Ereignissen übergeben wird. Diese Klasse bildet alle für die clusterinterne Kommunikation nötigen Datenfelder auf ein Byte-Array ab. Dabei erfolgen Zugriffe auf die Felder über dafür vorgesehene Methoden, die eine Konvertierung in das Netzwerkformat automatisieren. Die BcPacket-Klasse gibt dem Nutzer die Möglichkeit, den Nutzdatenbereich direkt zu beschreiben und seine Größe festzulegen.

Der Inhalt des Nutzdatenbereichs hängt dabei vom Subtyp-Datenfeld ab (während das Typ-Datenfeld für die Cluster-Kommunikation reserviert ist).

Um weitere Variablen im Datenbereich ablegen zu können, wurde die darauf aufbauende (aber nicht davon abgeleitete) Klasse `MultiHopPacket` zur weiteren Kapselung eingeführt. Eine Instanz dieser Klasse wird auf einem BcPacket aufgesetzt und schreibt direkt in die Datenfelder, wobei hier wieder transparent die Konvertierung zwischen Rechner- und Netzwerk-Format stattfindet.

Ein `MultiHopPacket` sieht zusätzliche Datenfelder für die Cluster-übergreifende Kommunikation vor (Tabelle 3.2) und bietet der Anwendung wieder einen frei beschreibbaren Bereich, sowie Funktionen zum Ändern und Auslesen der Größe dieses Bereichs.

3.7.5 Monitoring-API

Um auf Topologieänderungen reagieren zu können, muss das Weltmodell darüber in Kenntnis gesetzt werden. Dazu sieht die MAC-Schicht eine Schnittstelle vor, die den Aufbau und Abbau von Links sowie Änderungen des eigenen Knoten-Zustands

Offset	*Feld*	*Datentyp*	*Beschreibung*
`0x00`	Src	NodeId	Absender des Pakets
`0x04`	Dest	NodeId	Empfänger
`0x08`	NextHop	NodeId	nächstes Gateway
`0x0c`	TTL	uint8_t	verbleibende Hops
`0x0d`	PayloadType	enum	EchoRequest, EchoReply, ...
`0x0e`	Payload	char[]	dem PayloadType entsprechende Daten

Tabelle 3.2: `MultiHopPacket`-Datenfelder

signalisiert. Diese Schnittstelle wird vom Weltmodell verwendet, um die Beziehungen des eigenen Knotens zu pflegen und an die Nachbarn weiterzusenden.

Realisiert wird diese Schnittstelle über die `IntraMon`-Klasse, von der man seine eigene Klasse ableiten kann. Instanziiert man die Klasse, so wird sie in einen Nachrichtenverteiler eingetragen, der Mitteilungen über die Änderung des Knotens von Head zu Client und zurück, sowie über hinzukommende und entfernte Heads bzw. Clients an entsprechende Klassenmethoden ausliefert.

Diese Schnittstelle wird vom Weltmodell verwendet, um die unmittelbare Nachbarschaft zu beobachten und Änderungen sofort zu übernehmen. Dabei wird die Nachbarliste komplett gelöscht und die Sequenz-Nummer inkrementiert, wenn der Knoten von Head zu Client oder umgekehrt wechselt. Bei hinzukommenden oder entfernten Nachbarn wird die interne Liste entsprechend angepasst und wiederum die Sequenznummer erhöht. Beides führt außerdem dazu, dass der eigene Eintrag eine erhöhte Sendepriorität bekommt, um bei der nächsten Weltmodell-Propagation an alle Nachbarn versendet zu werden.

Zusätzlich empfängt das Weltmodell über den `promiscHook` Pakete und wertet die darin evtl. vorhandenen Topologiedaten aus. Aus einem Topologie-Paket wird sequentiell jeder Knoten ausgelesen, seine Sequenz-Nummer mit der im eigenen Weltmodell verglichen, und falls die empfangende Nachbarschaftsliste aktueller ist, diese in das Weltmodell eingepflegt.

3.7.6 Weltmodell-Schnittstelle

Das Weltmodell hat die Aufgabe, die Verbindungen zwischen allen Knoten abzuspeichern und bei Bedarf Pfade von der eigenen Station zu bestimmten Zielen zu berechnen. Da die Suche dabei auf jedem Gateway neu durchgeführt wird, um veraltete Weltmodell-Informationen zu kompensieren, ist nur das nächste Gateway und sein Cluster für das Weitersenden eines Pakets relevant. Diese werden im Weltmodell nach einer Routenberechnung zwischengespeichert und über die Funktionen

`getNextHop()` und `getClusterHead()` bereitgestellt. Zuerst muss allerdings mit `isRoutable()` festgestellt werden, ob der Zielknoten überhaupt erreichbar ist.

Um den berechneten Pfad traversieren zu können, wurde des Weiteren `getPrevHop()` eingeführt. Diese Funktion liefert den vorletzten Knoten auf dem Pfad zu der übergebenen Id. Ruft man sie iterativ auf, so erhält man in umgekehrter Reihenfolge die Zwischenstationen zu einem beliebigen Ziel.

3.7.7 Weltmodell-Datenstrukturen

Das Weltmodell speichert neben einer Liste der bereits bekannten Knoten auch die Nachbarschaftsbeziehungen und Routing-Daten zu jedem dieser Knoten. Die Routing-Daten werden bei jedem neu ankommenden Topologiedaten-Paket als veraltet markiert und bei der nächsten Anforderung neu berechnet. Dadurch werden unnötige Berechnungen eingespart.

Knotenbezogene Informationen werden mit Hilfe der Klasse `WorldModelElement` geordnet. Eine Instanz davon speichert die von einem Knoten zuletzt gesehene Topologie-Sequenznummer, ein Feld seiner Nachbarschaft (`vector<NodeId>`) und die für diesen Knoten berechneten Routing-Daten.

Alle Elemente des Weltmodells werden in einem balancierten Baum abgelegt, der die Einträge nach Knotennummern ordnet (`map<NodeId,WorldModelElement>`). Auf diese Weise ist der wahlfreie Zugriff auf Einträge mit einem Aufwand von $\mathcal{O}(\log N)$ möglich. Dies ist besonders wichtig, da diese Operation sehr häufig benötigt wird.

3.7.8 Knoten-Neustart

Wird eine Station neu gestartet, so beginnt ihre Topologie-Sequenznummer wieder bei 1. Das Weltmodell der anderen Knoten enthält allerdings Informationen von der Station mit einer höheren Sequenznummer, so dass aktuellere Topologiedaten des neu gestarteten Knotens nicht übernommen werden.

Um dieses Problem einzudämmen, überprüft ein Knoten die von anderen verbreitete Sequenznummer für sein Weltmodell. Empfängt er einen Wert, der höher als sein eigener ist, geht er davon aus, dass er neu gestartet wurde. Um sein aktuelles Weltmodell an die Nachbarn propagieren zu können, „springt" er mit seiner Sequenznummer über den empfangenen Wert.

3.7.9 Weltmodell-Weitergabe

Wenn ein Knoten von einem seiner Cluster-Heads nach einem Paket gefragt wird, aber keine Daten zu versenden hat, generiert er ein Weltmodell-Paket (Format in Tabelle 3.3). Dabei werden bevorzugt neue bzw. aktualisierte Topologiedaten und danach die vorhandenen Bestandsdaten versendet.

Offset	*Datenfeld*	*Typ*	*Erläuterung*
0x00	Version	`byte`	konstanter Wert / Identifikation
0x01	Anzahl Knoten	`byte`	Anzahl im Paket geführter Knoten
0x02	Knoten-Id	`NodeId`	Knoten, dessen Nachbarschaft folgt
0x06	Sequenz-Nr	`unsigned short`	Sequenz-Nr. von diesem Knoten
0x07	Anzahl Nachbarn	`byte`	Länge der Nachbarschaftsliste
0x08	Nachbarn	`NodeId[]`	Feld mit allen Nachbarn
...	...	...	...

Tabelle 3.3: Aufbau Weltmodell-Paket

Ein solches Datenpaket, am Beispiel von Knoten 1 in Abbildung 3.8, ist in Tabelle 3.4 aufgeschlüsselt. Dabei hat der Knoten bereits Informationen von allen unmittelbaren Nachbarn erhalten, allerdings nicht von den weiter entfernten Knoten, und verschickt sein komplettes Weltbild, welches noch in einem einzelnen Daten-Paket Platz findet.

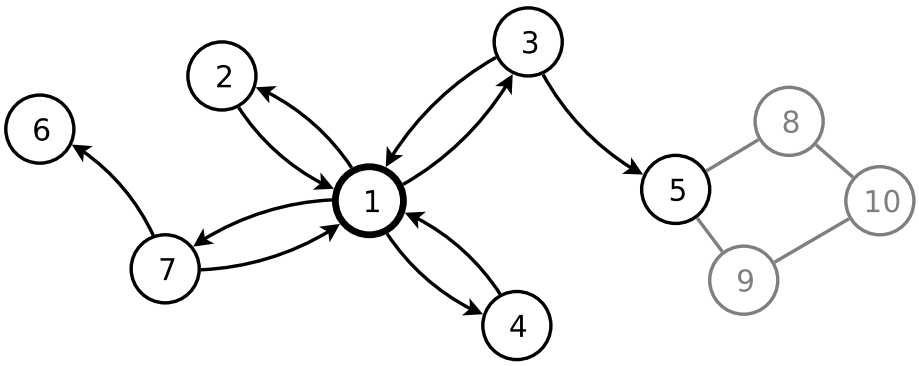

Abbildung 3.8: Weltmodell von Knoten 1 nach Schritt 2

3.7.10 Schnittstelle zur Anwendung

Gegenüber der Anwendung exportiert die Routing-Schicht drei Callbacks. Neben dem von unten getriggerten Empfang und Versand von Multi-Hop-Paketen (ein Promiscuous Mode ist zwar möglich, wurde aber nicht implementiert) existiert noch ein Hook, der von der Anwendung aufgerufen werden kann, um ein Paket in die Sendewarteschlange einzureihen. Dies ist notwendig, damit sowohl synchron als auch asynchron Pakete versendet werden können, und kann z.B. für die Beantwortung von Ping-Paketen eingesetzt werden.

Datenfeld	*Inhalt*	*Erläuterung*
Version	`0x5A`	konstanter Wert / Identifikation
Anzahl Knoten	`5`	Paket enthält Nachbarschaft von 5 Knoten
Knoten-Id	`1`	Es folgt die Nachbarnliste von Knoten 1
Sequenz-Nr	`1`	Sequenz-Nr. von Knoten 1
Anzahl Nachbarn	`4`	Anzahl der Nachbarn von Knoten 1
Knoten-Id[]	`[2, 3, 4, 7]`	Nachbarn von Knoten 1
Knoten-Id	`2`	Es folgt die Nachbarnliste von Knoten 2
Sequenz-Nr	`1`	Sequenz-Nr. von Knoten 2
Anzahl Nachbarn	`1`	Anzahl der Nachbarn von Knoten 2
Knoten-Id[]	`[1]`	Nachbarn von Knoten 2
Knoten-Id	`3`	Es folgt die Nachbarnliste von Knoten 3
Sequenz-Nr	`1`	Sequenz-Nr. von Knoten 3
Anzahl Nachbarn	`2`	Anzahl der Nachbarn von Knoten 3
Knoten-Id[]	`[1, 5]`	Nachbarn von Knoten 3
Knoten-Id	`4`	Es folgt die Nachbarnliste von Knoten 4
Sequenz-Nr	`1`	Sequenz-Nr. von Knoten 4
Anzahl Nachbarn	`1`	Anzahl der Nachbarn von Knoten 4
Knoten-Id[]	`[1]`	Nachbarn von Knoten 4
Knoten-Id	`7`	Es folgt die Nachbarnliste von Knoten 7
Sequenz-Nr	`1`	Sequenz-Nr. von Knoten 7
Anzahl Nachbarn	`2`	Anzahl der Nachbarn von Knoten 7
Knoten-Id[]	`1, 6`	Nachbarn von Knoten 7

Tabelle 3.4: Beispiel für ein Weltmodell-Paket

3.7.11 Test-Anwendung

Es wurde eine Test-Anwendung implementiert, die in regelmäßigen Zeitabständen Ping-Pakete an einen vorgegebenen anderen Knoten im Netz verschickt. Die Datenpakete enthalten dabei einen Zählerwert und die lokale Zeit, zu der sie versendet wurden.

Empfängt die Anwendung ein solches Ping-Paket, so erzeugt sie ein Antwortpaket, das Pong, welches eine Kopie der Daten aus der Anfrage enthält und an den Absender des Ping-Pakets gerichtet ist.

Wenn die Antwort beim Absender ankommt, so kann dieser aus dem Zeitstempel im Datenpaket und der aktuellen Uhrzeit die Laufzeit für beide Richtungen ermitteln und diese ausgeben.

3.7.12 Modularisierung der Komponenten

Die MAC-Schicht, das Best-Effort-Routing und die Test-Anwendung bilden eigene Module, die zur Laufzeit als dynamische Bibliotheken nachgeladen werden können. Dabei erzeugt jedes Modul beim Laden eine Instanz der eigenen Klasse. Damit die Module auch untereinander kommunizieren können, wird das Object Repository von GEA eingesetzt, das es erlaubt, unter einem bestimmten Namen ein Objekt und den dazugehörigen Typ abzuspeichern.

Das Object Repository wird für die in Tabelle 3.5 aufgezählten Objekte verwendet. Der Präfix in der Bezeichnung dient dabei der Unterscheidung zwischen den von der MAC-Schicht exportierten Objekten und denen aus der Multi-Hop-Routing-Schicht.

Bezeichnung	*Typ*	*Benötigt von*
`MAC/fillpacketHooks`	Hooks<BcPacket>	Multi-Hop
`MAC/recvHooks`	Hooks<BcPacket>	Multi-Hop
`MAC/slotReserve`	SlotReserve	QoS
`MAC/worldmodel`	WorldModel	Multi-Hop, QoS
`MultiHop/fillpacketHook`	Hooks<BcPacket,1>	Multi-Hop-Anwendungen
`MultiHop/recvHooks`	Hooks<BcPacket>	Multi-Hop-Anwendungen
`MultiHop/enqueueHook`	Hooks<BcPacket,1>	Multi-Hop-Anwendungen

Tabelle 3.5: Objekte im Object Repository

4 Routing mit Qualitätsgarantien

Die Hauptaufgabe einer Routing-Schicht für QoS besteht darin, gegenüber den Anwendungen Zusagen über die Zustellung von Paketen zu geben. Diese Garantien betreffen die bereitgestellte Bandbreite, die Laufzeit der Pakete und die Verlustraten. Die genauen Anforderungen muss die Anwendung über eine festgelegte Schnittstelle spezifizieren, woraufhin die QoS-Schicht prüft, ob eine Erfüllung möglich ist, und die Ressourcen reserviert. Danach muss sie stetig kontrollieren, ob die Verbindung nach wie vor hinreichend ist und auftretende Garantieverletzungen der Anwendung mitteilen.

Den zweiten wesentlichen Bestandteil der vorliegenden Arbeit bildete die Entwicklung und Implementierung eines Protokolls, welches diesen Anforderungen genügt. Im Folgenden wird der grundsätzliche Aufbau des Protokolls und seiner Komponenten vorgestellt und detailliert auf die einzelnen Bestandteile eingegangen. Daraufhin wird dargelegt, welche Garantien das Protokoll gewährleisten kann und worauf diese im Einzelnen beruhen. Schließlich wird die Implementierung mit ihren spezifischen Details vorgestellt.

4.1 Aufgaben und Lösungskonzepte

Um in einem Netzwerk für eine Verbindung Ressourcen garantieren zu können, müssen diese zunächst auf allen beteiligten Knoten reserviert werden. Wenn die Reservierung bestätigt ist und QoS-Daten übertragen werden, muss die Verbindung ständig überwacht werden, um Ausfälle schnell zu erkennen und die Anwendung zu benachrichtigen.

Reservierung

Das Reservieren einer QoS-Verbindung erfordert eine Signalisierung der Anforderungen vom Sender zum Empfänger und eine Rücksignalisierung der Bestätigung oder Ablehnung. Dazu gibt es zwei grundlegende Verfahren. Die Reservierung kann in-band (als Zusatzfeld in normalen Datenpaketen) oder out-of-band (mit im Voraus ausgetauschten Zusatzpaketen) erfolgen.

Obwohl der geringere Overhead für die In-Band-Arbeitsweise spricht, hat diese andere Nachteile. So sind die Anwendungsanforderungen nicht von vorn herein erfüllt, sondern werden erst nach dem Verbindungsaufbau signalisiert und bestätigt

bzw. abgelehnt. Daher können die QoS-Anforderungen auch nicht bei der Pfadsuche berücksichtigt werden, was zur Verwendung von Pfaden führen kann, welche die Garantien nicht anbieten können.

Dies macht eine der eigentlichen Verbindung vorangehende Out-Of-Band-Reservierung sinnvoller, die mit der Suche eines adäquaten Pfades verbunden ist. Dort greifen dann die Garantien für alle versendeten Datenpakete und es ist von vorn herein klar, dass die gefundene Route der Anwendung genügt.

Pfadsuche für QoS

Die Pfadsuche erfolgt analog zum Best-Effort-Routing nach dem Fish-Eye-Prinzip, indem das Paket immer ausgehend vom lokalen Weltmodell in Richtung des Empfängers weitergeleitet wird. Zusätzlich müssen auf dem Weg aber von jedem Gateway-Knoten Slots in seinem Ziel-Cluster reserviert werden, um die spätere Datenübertragung zu gewährleisten. Die Anzahl der Slots (und damit die Bandbreite der Verbindung) wird dabei von der Anwendung vorgegeben.

Für eine Pfadreservierung müssen Informationen darüber vorliegen, auf welchen Links noch genügend Bandbreite verfügbar ist, und wo die Verbindung nicht mehr entlanggeführt werden kann. Dieses kann proaktiv übermittelt werden, indem das Weltmodell mit den freien Bandbreiten einzelner Knoten bzw. Cluster angereichert wird. Ein solcher Ansatz führt aber dazu, dass nach jedem Routen-Auf- und Abbau sich die Bandbreiten-Daten der beteiligten Knoten ändern, was zusätzliche Netzbelastung durch Weltmodell-Updates nach sich zieht. Außerdem kann es immer noch passieren, dass durch veraltete Informationen eine Route durch einen Cluster gelegt wird, der die Bandbreite nicht mehr anbieten kann.

Aus diesen Gründen wurde für die Bandbreitenreservierung ein optimistisches reaktives Verfahren gewählt: Es wird davon ausgegangen, dass auf dem kürzesten Pfad hinreichend Bandbreite vorhanden ist, und wenn diese Annahme nicht erfüllt wird, so wird zur Laufzeit mit Hilfe eines verteilten Backtracking-Verfahrens um den Engpass herum geleitet.

Dazu überprüft jedes Gateway, der an einem QoS-Verbindungsaufbau teilnimmt, ob in dem Cluster, der zum Ziel führt, genug Bandbreite verfügbar ist. Ist dies nicht der Fall, schließt er diesen Cluster von der Pfadsuche aus und nimmt den nächsten mit dem kürzesten Weg. Findet er über keinen seiner Cluster einen Weg, so gibt er das Paket an seinen Vorgänger zurück, der weitere alternative Pfade ausprobiert. Die Liste der bereits überprüften Cluster wird dabei auch an den Vorgänger übertragen, so dass er diese nicht mehr in Betracht zieht.

Überwachung und Signalisierung

Für die Mitteilung von erfolgreichen und gescheiterten Verbindungsaufbauversuchen sowie zur Meldung von Topologieänderungen, die eine Route unbenutzbar machen,

werden weitere Kontrollnachrichten benötigt. Dabei müssen Topologieänderungen auf jedem Knoten überwacht und mit der Routing-Tabelle für QoS abgeglichen werden, um zerstörte Verbindungen zu erkennen.

Dies sollte zum einen periodisch erfolgen, um Timeouts zu erkennen, und zum anderen bei Änderungen der Nachbarschaft stattfinden, um auf zusammengebrochene Verbindungen sofort reagieren zu können. Zudem müssen die von anderen Knoten versendeten Kontrollnachrichten ausgewertet und weitergegeben werden.

4.2 Optimistische Reaktive Pfadsuche

Die Pfadsuche für eine QoS-Verbindung unterscheidet sich in zwei Aspekten vom Best-Effort-Routing: Die Menge der möglichen Pfade zum Ziel ist geringer, weil nicht alle Cluster auf dem Weg die für die Verbindung angeforderten Garantien anbieten können. Außerdem kann jedes Best-Effort-Paket einen anderen Pfad zum Ziel einschlagen, während QoS-Pakete durch die erforderliche Reservierung an die am Anfang gewählte Route gebunden sind.

Zunächst wird optimistisch davon ausgegangen, dass alle Cluster auf der vom Weltmodell bestimmten kürzesten Strecke zum Ziel die von der Anwendung geforderte Bandbreite bereitstellen können. Um das zu überprüfen, wird das Anfragepaket von Gateway zu Gateway auf dem Pfad zum Ziel weitergegeben und dabei jeweils Bandbreite in den Clustern dazwischen reserviert.

Scheitert eine der Reservierungen, so sucht das Gateway, welches den Fehler feststellt, einen alternativen Pfad. Wenn es keinen alternativen Pfad findet, signalisiert es seinem Vorgänger, dass dieser eine neue Route finden soll.

Zu diesem Zweck führt jedes Paket eine Liste der bereits besuchten Cluster, sowie der Cluster, von denen bekannt ist, dass sie zu wenig freie Bandbreite besitzen. Erstere dient der Vermeidung von Routing-Schleifen, in letztere werden alle die Cluster als *Bad Guys* eingetragen, die die Reservierungsanforderung abgelehnt haben. Beide Listen haben eine vom Initiator der Verbindung vorgegebene Höchstgröße: Wird diese erreicht, dann wird ein Scheitern des Verbindungsaufbaus zurücksignalisiert.

Bei der Pfadberechnung wird die Bad-Guy-Liste dem Weltmodell übergeben, Der Dijkstra-Suchalgorithmus wird dann erneut durchgeführt, allerdings werden keine durch die übergebenen Bad Guys führenden Verbindungen berücksichtigt. Durch die vorhandenen globalen Link-State-Daten ist das Weltmodell dann in der Lage, trotz der eingeschränkten Knoten-Auswahl alternative Pfade aufzufinden, sofern sie existieren.

Am Beispiel sieht das wie folgt aus (Abb. 4.1): Knoten 1 versucht eine QoS-Verbindung zu Knoten 10 herzustellen. Dazu bestimmt er den kürzesten Pfad, der über Cluster 3 und Gateway 5 führt. Er reserviert daraufhin die nötige Bandbreite beim Cluster-Head und versendet das Initialisierungspaket an Knoten 5 (Schritt 1). Knoten 5 bestimmt wiederum den kürzesten Pfad zu Knoten 10. Er stellt fest, dass das

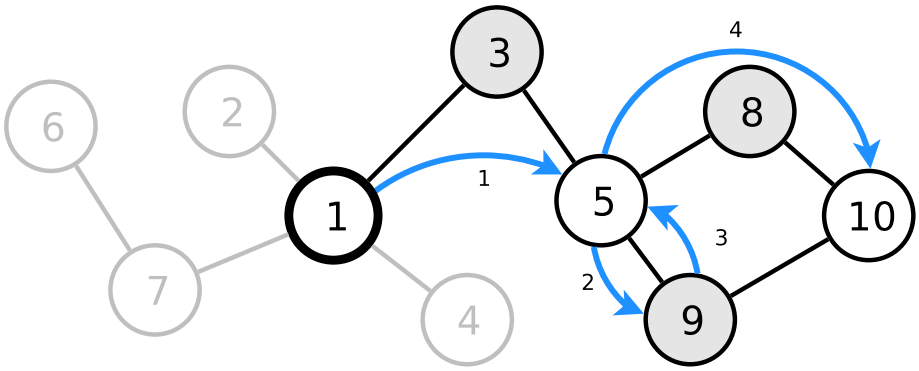

Abbildung 4.1: Optimistische Reaktive Pfadsuche: Beispiel

Ziel bereits in einem seiner Cluster ist und über den Head von Cluster 9 erreicht werden kann. Dazu versucht er eine Bandbreitenreservierung in Cluster 9 (Schritt 2). Diese wird jedoch vom Head abgelehnt (3), und daraufhin wird Cluster 9 im Paket als Bad Guy vermerkt. Das Weltmodell wird erneut nach einem Pfad befragt, berücksichtigt dabei die Nichterreichbarkeit über Knoten 9 und gibt Cluster 8 als Ziel aus. Daraufhin reserviert Knoten 5 die Bandbreite in Cluster 8 und versendet die Anforderung weiter (Schritt 4).

4.3 Struktur des QoS-Routing-Systems

Das QoS-Modul ist das Bindeglied zwischen der Clustering-MAC-Schicht, deren Weltmodell und der QoS-Anwendung (Abb. 4.2).

Nach oben hin kommuniziert das Modul mit der Anwendung, welcher es erlaubt, Verbindungen aufzubauen, Statusmitteilungen über bestehende Verbindungen zu erhalten und Pakete zu versenden und zu empfangen.

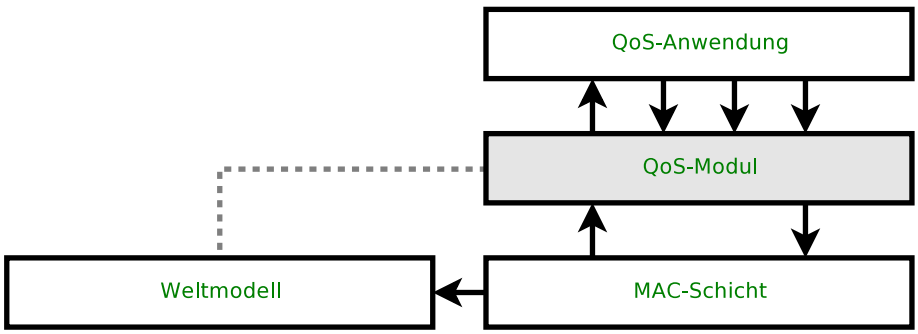

Abbildung 4.2: Einordnung des QoS-Moduls

Von der MAC-Schicht bezieht das QoS-Modul das Weltmodell und benutzt deren Slot-Reservierungs-Schnittstelle, sowie die Funktionen zum Paketversand und zur Auswertung empfangener Pakete.

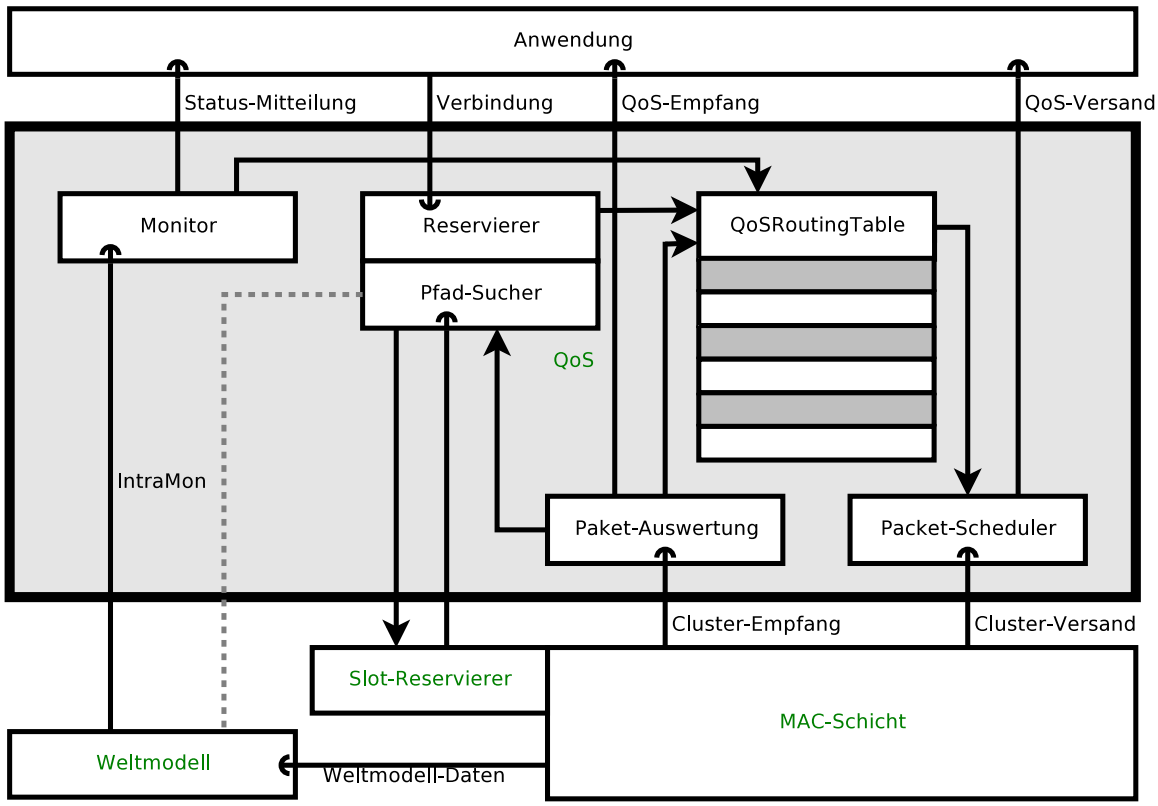

Abbildung 4.3: Komponenten des QoS-Moduls

Damit das QoS-Routing funktioniert, müssen zahlreiche Aufgaben erfüllt werden. Für eine bessere Modularität erfolgt die Abarbeitung der Teilaufgaben von einzelnen Komponenten, die miteinander interagieren, um die Gesamtanforderungen zu bewältigen. Eine Übersicht darüber findet sich in Abb. 4.3, das Zusammenspiel der Komponenten und ihre Funktionsweise wird im Folgenden erklärt. Ihre Interaktion lässt sich am besten am Ablauf einer Reservierung veranschaulichen.

4.3.1 Ablauf einer Reservierung

Eine Anwendung, die eine QoS-Verbindung aufbauen will, richtet ihre Anfrage zunächst an den *Reservierer/Pfadsucher*. Dieser überprüft mit Hilfe des Weltmodells, ob ein Pfad zum Zielknoten aufgebaut werden kann, und startet eine Slot-Reservierung für den Cluster, durch den diese Verbindung geht. Er fügt einen Eintrag in die *Routing-Tabelle* ein, markiert diesen als offene lokale Reservierung und fügt ein

Initialisierungspaket mit den Daten der Verbindung hinzu. Sobald die Reservierung erfolgt ist, wird dies von der MAC-Schicht an den Reservierer signalisiert, und dieser setzt den Status der Route auf Remote-Reservierung. Wenn der dazugehörige Cluster-Head das nächste Mal Daten anfordert, wird das Initialisierungspaket vom *Scheduler* an ihn ausgeliefert. Danach wartet der Knoten, bis eine Rückmeldung zurückkommt oder die Wartezeit überschritten wird. Letzteres überwacht der *Monitor* und meldet es an die Anwendung, die dann einen neuen Verbindungsaufbau versuchen kann.

Empfängt der nächste Knoten ein Initialisierungspaket, leitet sein *Auswerter* es an den Reservierer weiter, wo die QoS-Anforderung genau wie ein lokaler Verbindungsaufbau behandelt wird.

Erreicht die Aufbaunachricht schließlich den Empfänger, generiert dieser daraufhin eine Bestätigung, die den selben Pfad zurückgeht. Dabei setzen alle beteiligten Stationen die Route auf „reserviert". Wenn der Sender die Bestätigung erhält, ist die Route vollständig aufgebaut und die Anwendung kann Pakete mit der ihr nun zugesicherten Bandbreite versenden.

Wenn Pakete verschickt werden, wird der Timeout-Wert der zugehörigen Verbindung auf allen Gateways aktualisiert und die Route bleibt in deren Tabellen erhalten. Verschwindet der Sender plötzlich, so findet keine Erneuerung mehr statt und die Einträge werden als „veraltet" markiert und dann gelöscht.

4.3.2 Routing-Tabelle

Zu jeder Verbindung, die über einen Knoten läuft (unabhängig davon, ob der Knoten Sender, Empfänger oder Gateway ist), muss dieser bestimmte Informationen speichern (Tabelle 4.1).

Feld	**Inhalt**
Sender	Initiator der Verbindung
Route-Id	vom Sender vergebener eindeutiger Wert
Reservierung	Reservierungszustand der Route
Bandbreite	Anzahl reservierter Slots
Vorgänger	Knoten, von dem die Pakete kommen
Nachfolger	Knoten, zu dem die Pakete geschickt werden sollen
Timeout	Zeitpunkt, an dem der Eintrag ungültig wird

Tabelle 4.1: Inhalt der Routing-Tabelle pro Verbindung

Zunächst gehört dazu eine eindeutige Kennzeichnung der Verbindung, die aus der Sender-Knoten-Id und einer vom Sender vergebenen eindeutigen Routen-Id aufge-

baut ist. Diese beiden Werte zusammen kennzeichnen eine Verbindung eineindeutig und werden zur späteren Zuordnung von Paketen zu dieser Verbindung eingesetzt.

Der Zustand der Reservierung wird auch in der Tabelle abgelegt. Daran kann das Gateway erkennen, ob gerade eine lokale Reservierung bearbeitet wird und welche Aktionen mit ankommenden Paketen, die zu dieser Verbindung gehören, durchgeführt werden müssen.

Weiterhin muss bei jeder Verbindung vermerkt werden, welche Bandbreite dieser zugesichert wurde – entsprechend dem Cluster-Schema, auf dem das Routing basiert, wird diese Bandbreite in Slots gemessen. Daher befindet sich in der Routenbeschreibung die Anzahl der für die Verbindung reservierten Slots.

Um Datenpakete weiterzuleiten, muss der Nachfolger, also der Knoten, an den die Datenpakete weitergesendet werden, abgespeichert werden. Um Verbindungsunterbrechungen zu bemerken, wird außerdem der Vorgänger mit abgelegt. Zusätzlich wird die Verbindung über einen Timeout abgesichert – läuft dieser ab, ohne dass Pakete übertragen wurden, so kann die Verbindung stillschweigend abgebaut werden.

4.3.3 Pfadsucher

Da einzelne Knoten nicht über ein vollständiges und aktuelles Gesamtweltmodell verfügen und ihre Informationen über entfernte Knoten nicht auf dem neuesten Stand sind, ist die vollständige Pfadberechnung durch den Sender weder für Best-Effort noch für QoS zweckmäßig.

Daher wird das Weltmodell bei beiden Verbindungsklassen nur für die Bestimmung der Richtung verwendet, ausgehend von der Annahme, dass die Knoten sich im Netz nicht schnell bewegen, dass alte Informationen nicht vollkommen irreführend sind und dass die Weltmodelle der Knoten näher zum Ziel auch präziser sind.

Für QoS-Pakete muss die Route nur beim Verbindungsaufbau berechnet werden, danach ist der Pfad für Datenpakete vorgegeben und braucht nicht neu ermittelt werden. Die Pfadsuche erfolgt verteilt auf den Gateways, ausgehend von dem Initiator der Verbindung. Das grundsätzliche Verfahren arbeitet zunächst genauso wie bei Best-Effort-Paketen – aus dem eigenen Weltmodell wird bestimmt, welcher Nachbar dem Ziel am nächsten ist. Allerdings werden dabei zwei Einschränkungen in die Weltmodell-Sicht integriert:

Cluster, die im gerade weitergeleiteten Paket als *Bad Guys* markiert sind, also solche, die bereits angefragt wurden und die Bandbreitenkriterien nicht erfüllen, werden komplett übergangen. Dadurch vermeidet der Algorithmus bereits bekannte Engpässe im Netzwerk. Kann keine alternative Strecke um die Engpass-Cluster herum ermittelt werden, so wird der Vorgänger-Knoten darüber informiert, der daraufhin wenn möglich einen anderen Pfad probiert. Die Gesamtanzahl der Backtracking-Versuche ist dabei sowohl durch die Vorgabe eine maximale Suchtiefe im Paket als auch durch die Größe des mobilen Netzwerks begrenzt.

4.3.4 Reservierer

Wenn der Pfadsucher unter Berücksichtigung der Bad Guys einen Pfad zum Ziel gefunden hat, wird die Verbindung in die Routing-Tabelle eingetragen und der Reservierer fordert bei der MAC-Schicht eine Erhöhung der Slots für den eigenen Knoten im Zielcluster an. Bis diese abgeschlossen ist, verbleibt das Initialisierungspaket in einem Zwischenpuffer und die Routing-Tabelle enthält einen entsprechend markierten Eintrag.

Wenn die MAC-Schicht die erfolgreiche Reservierung mitteilt, wird das Init-Paket zum Versand freigegeben, bei einer Ablehnung wird der Ziel-Cluster als Bad Guy eingetragen und der Pfadsucher mit einer neuen Routen-Berechnung beauftragt.

4.3.5 Paket-Scheduler

Ein Knoten kann nur asynchron Pakete verschicken – der Sendezeitpunkt wird von der MAC-Schicht (bzw. vom Cluster-Head) vorgegeben. Eine Anwendung, die QoS-Datenpakete versenden möchte, kann daher warten, bis der Knoten vom Head zum Datenversand aufgefordert wird, und ihr Paket dann generieren. Weitergeleitete Pakete dagegen müssen in einem Puffer zwischengespeichert und auf Anfrage des entsprechenden Cluster-Heads gesendet werden.

Dazu wird der Paket-Zwischenpuffer bei einer Cluster-Head-Anfrage nach QoS-Datenpaketen durchsucht, die in diesen Cluster zu verschicken sind. Findet der Scheduler ein solches Paket, so liefert er es als Antwort an den Head aus. Liegen keine Datenpakete vor, so wird überprüft, ob QoS-Metapakete (Fehler oder Routenaufbaunachrichten, die keine reservierten Slots und damit eine geringere Priorität haben) in diesen Cluster verschickt werden müssen.

Liegen weder QoS-Daten- noch Meta-Pakete zum Versenden in den entsprechenden Cluster vor, so gibt der Scheduler die Kontrolle zurück und es können Best-Effort-Pakete oder Weltmodell-Informationen versendet werden.

4.3.6 Paket-Auswertung

Bei empfangenen Paketen ruft die MAC-Schicht über die entsprechende Callback-Routine den Paketauswerter auf. Dieser überprüft, ob es sich um QoS-Frames handelt und gibt ansonsten die Kontrolle sofort zurück. Pakete, die eine neue QoS-Verbindung initiieren, werden an den Reservierer zur weiteren Bearbeitung weitergegeben, weiterzuleitende QoS-Datenpakete in die entsprechenden Slots in der Routing-Tabelle (die gleichzeitig als Zwischenpuffer dient) eingetragen. Wird ein an diesen Knoten adressiertes QoS-Datenpaket empfangen, so wird die Anwendung benachrichtigt und kann den Inhalt auswerten.

Empfängt der Knoten Bestätigungspakete für eine über ihn führende Route, so wird der Reservierungsstatus in der Routing-Tabelle bestätigt und das ACK-Paket weiter an den Vorgänger in der Pfad-Kette gesendet.

Werden QoS-Ablehnungspakete empfangen, so gibt es einige Besonderheiten. Gehört die Ablehnung zu einer Route, die gerade aufgebaut wird, so muss ein neuer Pfad bestimmt werden. Verläuft der neue Pfad durch einen anderen Cluster als bisher, muss die Reservierung im vorherigen aufgehoben und im neuen angefordert werden. Gehört die Ablehnung zu einer bereits komplett aufgebauten Verbindung, oder hat das QoS-Init-Paket seine maximale Suchtiefe/Breite erreicht, so werden die reservierten Slots freigegeben und eine Ablehnung an den Vorgänger geschickt.

4.3.7 Monitor

Ein wesentlicher Bestandteil in einem QoS-System ist eine Überwachungsinstanz, die den Ist- und Soll-Zustand der reservierten Ressourcen miteinander vergleicht und Diskrepanzen an die Anwendung meldet.

Dazu müssen mehrere Mechanismen miteinander kombiniert werden. Zunächst einmal muss die Routing-Tabelle periodisch auf abgelaufene Timeouts überprüft und gegebenenfalls Reservierungen rückgängig gemacht und Einträge gelöscht werden.

Der zweite Mechanismus überprüft die von der MAC-Schicht gelieferten Nachbarschaftsinformationen. Verliert der Knoten die Zugehörigkeit zu einem Cluster, so müssen alle durch diesen Cluster führenden Routen entfernt werden. Weiterhin müssen die Nachbarschaftsbeziehungen der eigenen Cluster-Heads überprüft werden, um den Ausfall von Gateway-Knoten festzustellen.

Für jeden dieser Ausfälle muss ein Benachrichtigungspaket über den Routenausfall generiert werden. Dabei muss unterschieden werden, ob der Ausfall auf dem Pfad vor dem eigenen Knoten (also im Vorgängercluster) oder nach dem eigenen Knoten (im dem Ziel zugewandten Cluster) stattfand. Die Ausfallnachricht wird dabei in die jeweils andere Richtung gesendet. Da der Ausfall auf beiden Seiten erkannt wird, werden zwei Signalisierungspakete von der Bruchstelle aus zum Sender und zum Empfänger gesendet, so dass die Route schnell abgebaut wird.

Der letzte Monitoring-Mechanismus besteht darin, Ausfallnachrichten von anderen Knoten auszuwerten, betroffene Routen zu dereservieren und die Nachrichten weiterzuleiten. Die Weiterleitung erfolgt dabei an den in der Routing-Tabelle vermerkten Knoten und nicht auf dem direkten Weg zum Ziel – denn die Nachricht ist auch an alle Zwischenstationen gerichtet, die auf dem reservierten Pfad liegen.

Dadurch, dass die Unterbrechung eines Links von beiden Seiten erkannt wird und beide Gateway-Knoten eine Benachrichtigung versenden, werden die Reservierungen auf beiden Seiten schnell abgebaut und Sender und Empfänger zügig informiert.

Bekommt ein Gateway-Knoten ein Datenpaket für eine nicht in seiner Routing-Tabelle enthaltene Verbindung, so ist diese höchstwahrscheinlich zuvor durch einen Timeout gelöscht worden. In diesem Fall erzeugt das Gateway eine Ausfallnachricht

und schickt sie zurück, damit der Vorgänger die Route abbauen und der Sender eine neue aufbauen kann.

Da die Dereservierungsnachrichten als Best-Effort-Pakete verschickt werden, ist deren Verlust auf dem Weg nicht auszuschließen. Aus diesem Grund erfolgt die doppelte Absicherung über Nachrichten und Timeouts – geht eine Abbaunachricht verloren, so wird die Route trotzdem abgebaut, jedoch erst nach dem der Timeout erreicht wurde. Wird die Route danach wieder verwendet, so bemerkt das Gateway die Diskrepanz und schickt ein Dereservierungspaket. Dasselbe passiert auch, wenn eine Route nur teilweise abgebaut wurde und der Timeout noch nicht erreicht worden ist – es wird eine neue Abbaunachricht erzeugt und die restliche Route abgebaut.

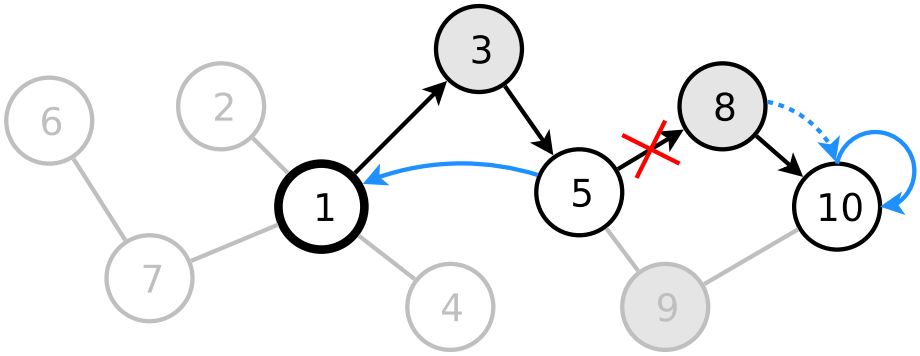

Abbildung 4.4: Signalisierung einer Routenunterbrechung

In Abbildung 4.4 wird eine zuvor aufgebaute QoS-Verbindung ($1 \rightarrow 5 \rightarrow 10$; Cluster-Heads 3 und 8) zwischen den Knoten 5 und 8 unterbrochen. Der QoS-Monitor auf Knoten 5 bekommt diese Information unmittelbar von seiner MAC-Schicht und erzeugt eine Abbaunachricht, die zu seinem Vorgänger, Knoten 1, geschickt wird. Die in Cluster 10 reservierte Bandbreite kann dabei nicht mehr explizit freigegeben werden, da keine Verbindung mehr zum Head besteht.

Knoten 10 beobachtet die Nachbarschaft seines Cluster-Heads 8 und stellt darüber (gepunktete Linie) fest, dass Knoten 5 nicht mehr im Cluster ist. Daher wird die lokale Anwendung über den Zusammenbruch informiert und die Route gelöscht.

Knoten 1 bekommt die Nachricht von Knoten 5 (Bogenpfeil $5 \rightarrow 1$) und informiert die Anwendung. Danach teilt er dem Head 3 mit, dass er weniger Slots benötigt und entfernt auch den Eintrag aus der Routing-Tabelle.

4.4 Implementierung

Dieser Abschnitt beschreibt wichtige Implementierungsdetails des QoS-Moduls. Dazu gehören neben einigen internen Strukturen das Paketformat und die Schnittstellen zur Anwendung und zur MAC-Schicht.

4.4.1 Einbindung ins System

Wird das QoS-Modul geladen, bindet es sich in die selben Rückruf-Schnittstellen ein wie das Best-Effort-Routing, allerdings mit einer höheren Priorität als dieses (Abb. 4.5). Zusätzlich verwendet es das MAC-Weltmodell für die Berechnung von Routen. Danach wird das QoS-Modul im Object Repository vermerkt, was es der Anwendung erlaubt, eine Referenz darauf zu bekommen.

Gegenüber der Anwendung werden Sende- und Empfangs-Callbacks für QoS-Pakete (`QOS/fillpacketHooks`, `QOS/recvHooks`) bereitgestellt. Dazu kommt eine Funktion zum Reservieren von Routen (`startReservation()`), die den Zielknoten, die erforderliche Bandbreite und eine Rückruffunktion für die Fehler- und Erfolgssignalisierung als Parameter erwartet.

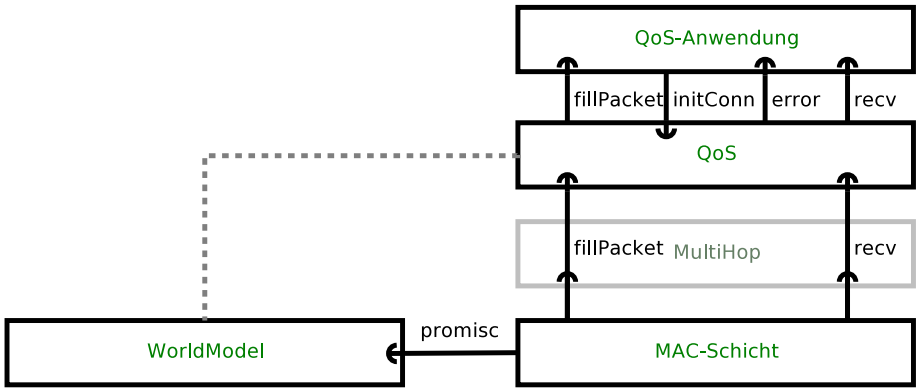

Abbildung 4.5: Verwendung von Hooks für die QoS-API

4.4.2 Paketaufbau für QoS

Für QoS-Verbindungen gelten etwas andere Regeln als für Best-Effort-Datenverkehr. Hier speichert jedes Gateway Zustandsinformationen, so dass diese nicht mehr im Paket übertragen werden müssen. Das macht den Netzwerk-Overhead geringer, erfordert aber mehr Speicher auf den Stationen als der Versand von Best-Effort-Paketen.

Entsprechend ist der Header von QoS-Paketen (Tabelle 4.2) schlanker als der von Best-Effort-Daten. Neben der eindeutigen Routen-Kennung, bestehend aus Absender- und Routen-Id, enthält er das nächste Gateway (das zur Weiterleitung des Pakets erforderlich ist) und den Pakettyp. Je nach Pakettyp weicht der weitere Aufbau des Pakets ab.

Initialisierungspakete führen weitere Datenfelder (Tabelle 4.3), die erforderlich sind, damit alle Zwischenstationen die Route reservieren können. So wird dort der Zielknoten und die Bandbreite abgelegt. Anhand des Ziels berechnet das Gateway die weitere Route, auf der das Paket versendet wird. Im Unterschied zu Best-Effort-

Offset	*Feld*	*Datentyp*	*Erläuterung*
0x00	NextHop	NodeId	nächstes Gateway
0x04	Src	NodeId	Absender des Pakets
0x08	RouteId	short	vom Sender vergebene eindeutige Kennung
0x0a	PktType	uint8_t	{ Init, ACK, NACK, BackNack, Payload }

Tabelle 4.2: `QoSPacket`-Datenfelder

Offset	*Feld*	*Datentyp*	*Beschreibung*
0x0b	Dest	NodeId	Ziel der Route
0x0f	Bandwidth	uint8_t	zu reservierende Bandbreite
0x10	MaxHopCount	uint8_t	maximale Anzahl Gateways
0x11	HopCount	uint8_t	bisherige Anzahl Gateways
0x12	MaxBadGuyCount	uint8_t	maximale Anzahl Backtracking-Schritte
0x13	BadGuyCount	uint8_t	bisherige Anzahl Backtracking-Schritte
0x14	Hops	NodeId[]	Liste der passierten Gateways
0x14+N	BadGuys	NodeId[]	Liste der erfolglos getesteten Cluster

Tabelle 4.3: `QoSPacket`: Datenfelder Routen-Initialisierung

Routing wird dabei aber die Liste der „Bad Guys" aus dem Paket ausgewertet – diese Cluster werden bei der Pfadsuche nicht als Möglichkeit betrachtet.

Für das Marshalling der Daten, also die Umwandlung aus den verwendeten Datentypen in das an die MAC-Schicht übergebene Byte-Feld wurde die Klasse `QoSPacket` eingeführt. Bei der Initialisierung muss ihr eine `BcPacket`-Instanz, also ein Paket der MAC-Schicht übergeben werden, danach werden alle Schreib- und Leseoperationen direkt in dem Datenfeld des MAC-Pakets durchgeführt.

`QoSPacket` stellt Get- und Set-Methoden für alle in den Tabellen 4.2 und 4.3 vorgestellten Datenfelder, überprüft aber vor dem Zugriff, ob der Inhalt der `PktType`-Variable diesen Zugriff erlaubt.

4.4.3 QoS-Routing-Tabelle

Die Routing-Tabelle ist ein zentrales Element des QoS-Moduls. Sie übt zwar keine eigenen Aktivitäten aus, wird aber von allen anderen Teilkomponenten verwendet. Sie dient auch als Ablage für Status- und Datenpakete, damit diese effizient über die Routen-Id gefunden werden können.

Zu jeder über den Knoten führenden Route enthält die Routing-Tabelle detaillierte Informationen (Tabelle 4.4). Das erste Feld enthält den Status der Reservierung, der

Variable	*Typ*	*Erläuterung*
`state`	enum	{LocalReserving, RemoteReserving, Reserved, Closing}
`lastnode`	bool	letzter Knoten auf dem Pfad?
`bandwidth`	int	reservierte Anzahl Slots
`prevHop`	NodeId	Vorgänger-Gateway
`nextHop`	NodeId	Nachfolger-Gateway
`prevHead`	NodeId	Vorgänger-Cluster
`nextHead`	NodeId	Nachfolger-Cluster
`timeout`	AbsTime	Zeitpunkt, wenn Eintrag ungültig wird
`data`	*BcPacket	Zeiger auf das zwischengespeicherte Paket

Tabelle 4.4: QoS-Routing-Tabelle: Elemente eines Eintrags

zum Anfang der Reservierung auf LocalReserving gesetzt wird, während die für die Verbindung erforderliche Bandbreite im lokalen Cluster angefordert wird. Danach wird der Status auf RemoteReserving gesetzt und das Initialisierungspaket weitergesendet. Nach Erhalt der Bestätigung vom Zielknoten gilt die Verbindung als reserviert (Reserved). Wird der Timeout überschritten oder die Verbindung abgebaut, so wird der Eintrag zunächst auf Closing gestellt und einige Zeit später komplett entfernt.

Das Feld `lastnode` ist nur auf dem Zielknoten `true`, es signalisiert, dass die Daten an die Anwendung und nicht an den `nextHop` weiterzuleiten sind.

In `bandwidth` ist die Anzahl der zu reservierenden bzw. der reservierten Slots abgelegt. Die Reservierung bezieht sich auf den in `nextHead` gespeicherten Cluster. Die anderen `NodeId`-Variablen enthalten den Pfad vor und nach dem Knoten (Abb. 4.6).

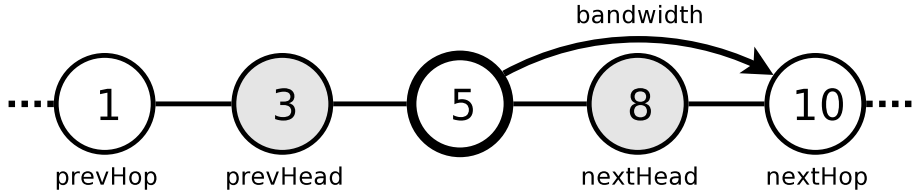

Abbildung 4.6: Sicht eines Gateways auf die QoS-Route

Die `timeout`-Variable enthält den Zeitpunkt, an dem der Eintrag nicht mehr gültig ist und `data` ist ein Zeiger auf ein Paket, welches in der Warteschlange des Gateways verbleibt. Dies wird durch das asynchrone Polling der Cluster-Heads erforderlich: Jedes Paket muss zwischengespeichert werden, bis der Knoten vom `nextHead` zum Weiterleiten aufgefordert wird. Je nach Zustand der Route kann es sich dabei um ein Statuspaket (Routenaufbau oder -abbau) oder um ein Datenpaket handeln. In dem

Fall, dass für die Route mehrere Slots reserviert wurden, kann es bei ungünstigem Polling-Verhalten dazu kommen, dass mehrere Datenpakete abgelegt werden müssen. Dann wird über den `data`-Zeiger eine verkettete Liste der abgelegten Datenpakete realisiert.

Ändert sich der Zustand der Reservierung, so werden eventuell zwischengespeicherte Pakete gelöscht – so wird verhindert, dass veraltete Informationen versendet werden bzw. Daten, für die keine Reservierung mehr besteht, das Netzwerk überfüllen.

Zugriff auf Routing-Tabellen-Einträge

Die Routing-Tabelle weist zwei unterschiedliche Zugriffsmuster auf: Wird ein QoS-Paket empfangen, muss anhand der Sender-Id und Routen-Id aus dem Paket der entsprechende Eintrag in der Tabelle ermittelt werden. Bei hohen Paketraten empfiehlt sich an dieser Stelle ein schneller Zugriffspfad, der mit einem balancierten Baum umgesetzt werden kann. Die Standard Template Library von C++ [Str00] bietet bereits eine entsprechende Datenstruktur (`stl::map`), die hier eingesetzt wird.

Ein weiteres Zugriffsmuster kommt beim Versenden von Paketen zum Tragen: Dort müssen Einträge gefunden werden, die einen bestimmten Cluster als `nextHead` haben und ein Paket zum Versand zwischenspeichern. Für dieses Muster wurde keine Optimierung implementiert; dies kann aber bei Bedarf nachgeholt werden.

Schließlich müssen regelmäßig alle Einträge auf abgelaufene Timeouts und unterbrochene Verbindungen kontrolliert werden. Dazu wird mit Hilfe eines Iterators der Inhalt der `stl::map` traversiert und jedes Element einzeln untersucht.

4.4.4 Reservierungsablauf

Will ein Knoten eine Route aufbauen, erzeugt er zunächst eine eindeutige Routen-Id. Dazu führt er einen Zähler, dessen Wert in die Id der neuen Verbindung übernommen wird und der danach inkrementiert wird.

Steht die Routen-Id fest, wird die Reservierung durchgeführt (Abb. 4.7). Dazu wird erst mit Hilfe des Weltmodells bestimmt, in welchen Cluster die Verbindung führt. Danach wird ein eine Slot-Reservierung für diesen Cluster gestartet. Ist die Reservierung erfolgreich, so wird beim nächsten Poll-Request ein Init-Paket in diesen Cluster geschickt.

Sind im Ziel-Cluster nicht mehr genügend Zeitschlitze frei, so gibt es zwei Möglichkeiten – entweder das Gateway versucht, einen Pfad über einen anderen Cluster zu finden, oder – falls keine alternativen Pfade existieren – wird eine Ablehnung an den vorangegangenen Knoten gesendet. Dabei entspricht die Ablehnung von ihrem Aufbau her dem Init-Paket und enthält entsprechend den Cluster mit unzureichender Bandbreite in der Bad-Guy-Liste.

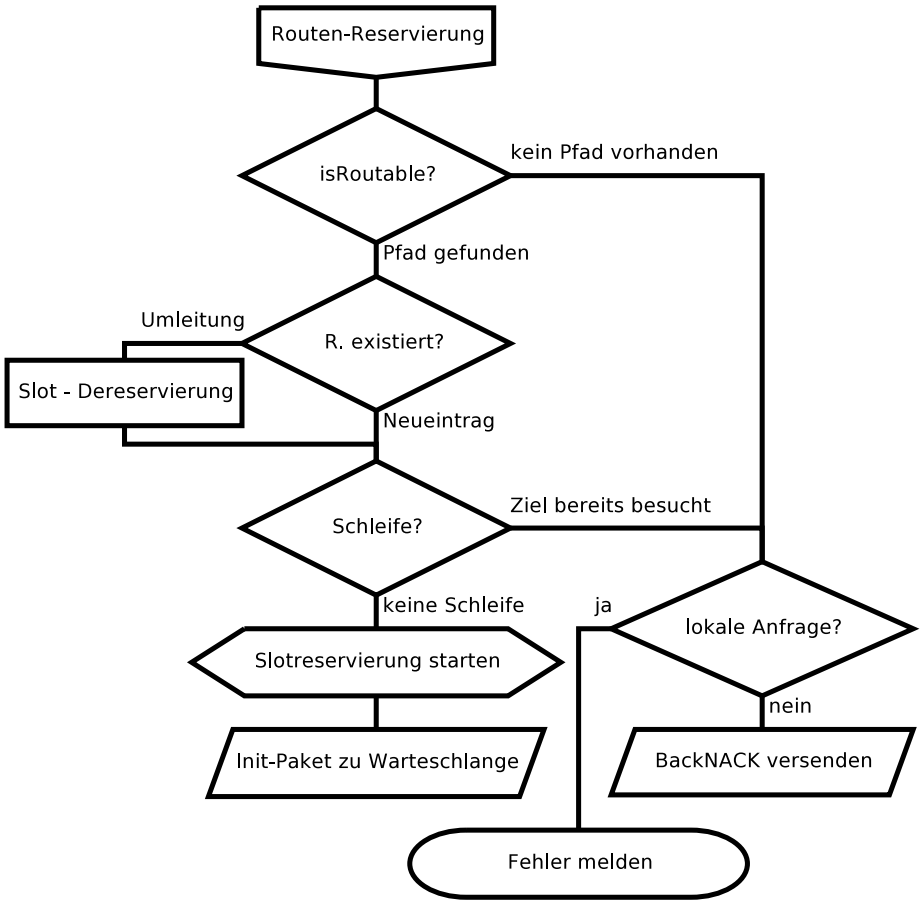

Abbildung 4.7: Ablauf der QoS-Reservierung

Diese Traversierung wird solange durchgeführt, bis das Ziel erreicht ist, oder die Bad-Guy-Liste keine Cluster mehr aufnehmen kann. Im letzten Fall wird ein NACK-Paket an den Sender zurückgeschickt.

Die Beschränkung der Hop- und Traversierungstiefe erlaubt es, die maximale Anzahl der zu durchlaufenden Knoten und damit auch die maximale Laufzeit des Pakets im Voraus festzulegen.

4.5 Garantien

Im Folgenden wird betrachtet, wie schnell QoS-Datenpakete innerhalb des Netzwerks zugestellt werden können. Diese Betrachtung gilt eingeschränkt auch für QoS-Statusnachrichten und Multi-Hop-Daten, nämlich dann, wenn keine Pakete mit einer höheren Priorität im Netzwerk unterwegs sind. Die höchste Priorität besitzen QoS-Daten, danach folgen QoS-Verbindungs- und Fehler-Pakete, gefolgt von regulären Best-Effort-Paketen. Die geringste Priorität haben Pakete mit Weltmodell-Daten.

Entsprechend kann ein Best-Effort-Paket in der selben Zeit wie ein QoS-Paket zugestellt werden, sofern kein Netzwerkengpass vorliegt und die Warteschlangen aller beteiligten Knoten leer sind. Allerdings kann der Sender diese zeitlichen Anforderungen nicht vorhersagen, da er den Pfad zum Ziel im Gegensatz zu einer aufgebauten QoS-Verbindung nicht kennt.

Sind die Warteschlangen der Gateways dagegen nicht leer, so ist die Verweildauer des Pakets auf den Zwischenstationen entsprechend länger – die Auslieferung gleich priorisierter Daten erfolgt nach dem FIFO-Prinzip.

Die MAC-Schicht erlaubt es, zuverlässig Pakete an andere Knoten im selben Cluster zuzustellen. Sie geht dabei von der Annahme aus, dass ein Paket spätestens nach drei Zustellungsversuchen garantiert ankommt. Um diese nicht auszuschöpfen, wird allerdings bei jeder Runde überprüft, ob das Paket in der vorherigen Runde erfolgreich ausgeliefert werden konnte. Im positiven Fall kann der Zeitschlitz dann für andere Informationen verwendet werden, ansonsten wird das Paket der Vorrunde wiederholt.

Die Rundenzeit berechnet sich dabei aus mehreren Faktoren: Eine Runde dauert so lange, bis alle N Cluster-Heads, die überlappungsfrei im selben Sendebereich arbeiten können, ihre M Clients befragt und eine Antwort erhalten haben. Die Zeit für eine Frage (Poll-Request) und die Antwort des Clients hängt dabei hauptsächlich von der Übertragungszeit auf dem Medium ab. Dazu kommt noch die Verarbeitungszeit auf dem Client (Empfang des Pakets, Bearbeitung und Versenden der Antwort), die aber nicht so signifikant ist.

Das Versenden eines üblichen Pakets auf der MAC-Schicht dauert bei einer Brutto-Datenrate von $54MBit/s$ ca. $0.3ms$. Ausgehend von einem Netz, in dem 6 Cluster mit je 16 Clients koexistieren können, ergibt das eine Rundenzeit von $16 \cdot 6 \cdot 2 \cdot 0.3ms = 57.6ms$, wenn die Verarbeitungszeit vernachlässigt wird.

Da jedes Datenpaket, das der Head von einem Client empfängt, sofort wiederholt wird, um alle Stationen im Cluster erreichen zu können (es wird nicht davon ausgegangen, dass die Clients untereinander eine Sichtverbindung haben müssen), erreicht es im Idealfall nach $t_{intra} = 0.6ms$ sein Ziel. Scheitert der erste Versand, wird es nach einer Runde wiederholt, kommt also nach $58.2ms$ an. Im schlechtesten Fall, also wenn die Auslieferung drei Runden benötigt, dauert die Zustellung des Pakets $115.5ms$.

Schwieriger ist die Vorhersage der Zustellung über Cluster-Grenzen hinweg. Dafür ist nicht nur die Zustellung innerhalb der Cluster, sondern auch die Wartezeit auf den Gateway-Knoten relevant, bis das Paket vom Head des Zielclusters angefordert wird. Diese Zeit t_{inter} variiert zwischen 0 und $57.0ms$ (eine Rundendauer abzüglich eines Zeitschlitzes), da jeder Knoten mindestens einmal pro Runde gerufen werden muss und die Aufrufe aus unterschiedlichen Clustern nicht synchronisiert werden können. Da Knoten häufig auch mehr als einmal pro Runde abgefragt werden, ist diese Latenz zudem nicht vorhersagbar, da die Polling-Reihenfolge sich ständig ändert.

Ein Paket, welches von einem Knoten in einen anderen Cluster versendet wird, benötigt für die Zustellung an das Gateway in seinem Cluster t_{intra}, für jedes weitere Gateway und für das Ziel jeweils $t_{inter} + t_{intra}$. Dabei ist zu beachten, dass zwischen zwei Gateways immer ein Cluster-Head ist, den diese Berechnung berücksichtigt.

Das ergibt eine maximale Laufzeit von $115.5ms + N \cdot 172.5ms$ für eine Strecke über N Gateways. Geht man von einem Medium ohne Übertragungsfehler aus, so sinkt die Zeit auf $0.6ms + N \cdot 57.6ms$ bei ungünstiger Cluster-Kopplung bzw. $0.6ms + N \cdot 0.6ms$ im idealen Übertragungs-Fall, dies ergibt einen Jitter von $N \cdot 57ms$.

Wenn die Auslieferung an einen Cluster-Head erfolgen soll, bzw. wenn ein Cluster-Head der Sender ist, gelten die selben Betrachtungen, da der Head bezogen auf die Aussendung und den Empfang von Paketen einen internen Client emuliert.

Routen-Aufbau

Der Aufbau einer QoS-Verbindung muss als Best-Effort-Anfrage erfolgen, da zu diesem Zeitpunkt noch keine Bandbreite zwischen Start und Ziel reserviert ist. Entsprechend ist eine Vorhersage der Höchstzeitdauer bis zum Empfang der Antwort (ob positiv oder negativ) nur unter der Voraussetzung möglich, dass zeitgleich keine weiteren Verbindungen über die selben Knoten aufgebaut werden.

Das Weltmodell liefert dabei nur einen Richtwert für die Anzahl der Zwischenstationen, der reale Wert muss zur Laufzeit durch die verteilte Pfadsuche bestimmt werden und ist durch den `MaxHopCount`-Wert nach oben begrenzt. Dazu kommen die möglichen Umleitungen aufgrund unpräziser Weltmodelldaten oder fehlender Bandbreite, deren Begrenzung als `MaxBadGuyCount` im Paket gelistet ist.

Im Gegensatz zu normalen Best-Effort- und QoS-Datenpaketen kommt beim Verbindungsaufbau aber eine weitere Verzögerung hinzu: Die Zeitschlitze für eine Verbindung muss der Client beim Head anfordern und auf die Bestätigung warten. Die Anforderung kann dabei erst nach einem vom Cluster einmal pro Runde verschickten *JoinRequest* erfolgen und das Ergebnis wird erst im nächsten *JoinRequest* bekanntgegeben. Erst danach kann die Verbindungsaufbaunachricht (auf einen Poll des Heads hin) verschickt werden.

Zusammengefasst bedeutet das Folgendes: Ein Gateway empfängt die Nachricht aus einem anderen Cluster (bzw. der Sender von der Anwendung) und startet sofort eine Slot-Reservierung. Diese wird nach dem Ende der laufenden Runde (0 - $57ms$) an den Head gesendet und eine Runde später ($+57.6ms$) kommt die Antwort beim Client an. In der folgenden Runde wird der Client von dem Head kontaktiert und versendet das Paket ($+0$ bis $57ms$).

5 Evaluierung

In diesem Kapitel wird die Funktionsfähigkeit der Weltmodell-Propagation, des Best-Effort-Routing-Protokolls und der QoS-Schicht im Simulator getestet und die Messergebnisse vorgestellt und ausgewertet.

5.1 Propagation des Weltmodells

Die Aktualität des Weltmodells ist eine wichtige Grundlage für die Funktionsweise des Routings und der QoS-Übertragung. Die Propagation der Weltmodell-Daten wird allerdings durch reguläre Datenübertragungen verzögert.

Um zu überprüfen, wie schnell sich das Weltmodell in Abhängigkeit von der Netzwerk-Belastung mit anderen Daten ausbreitet, wurde die Geschwindigkeit der Weltmodell-Propagation bei unterschiedlichen Lasten gemessen.

5.1.1 Weltmodell-Korrektheit

Um den Grad der Propagation zu einem bestimmten Zeitpunkt zu bestimmen, wurde ein Maß für die lokale und globale Weltmodell-Korrektheit k bzw. $\mathcal{K}$ eingeführt.

Die lokale Weltmodell-Korrektheit eines Knotens beschreibt die Abweichung seines Weltmodells von der realen Topologie. Sie bestimmt sich aus der Anzahl korrekter (also mit der realen Topologie übereinstimmender) Links abzüglich der inkorrekt im lokalen Weltmodell abgelegten Verknüpfungen (von Links, die nicht mehr tatsächlich existieren).

Um den Wert zu unterschiedlichen Zeitpunkten vergleichen zu können, wird er normalisiert und die auf diese Weise erhaltene relative Angabe verwendet.

Ist ein vorhandener Link dem Weltmodell nicht bekannt, so trägt das zu einem kleineren Wert bei, da die Anzahl der realen Links größer ist als die im Weltmodell. Damit sind sowohl positive als auch negative Abweichungen berücksichtigt.

Der Wert für die lokale Korrektheit kann höchstens 1 sein, wenn die Abbildung exakt mit der realen Topologie übereinstimmt, er ist 0, wenn der Knoten keinerlei Weltmodell-Daten hat, und kann negativ werden, wenn die Anzahl falscher Informationen die richtigen überwiegt.

Die globale Weltmodell-Korrektheit ergibt sich als der Durchschnitt der lokalen Korrektheiten aller Knoten. Auch hier gilt, die Werte sind 1 bei perfekter Propagation im gesamten Netz und werden bei zunehmenden Abweichungen kleiner.

5.1.2 Versuchsaufbau

Um die Weltmodell-Korrektheit zu testen, wurde die zeitliche Qualität der Propagation in unterschiedlichen Belastungsszenarien gemessen. Dazu wurde im Simulator ns-2 ein $1000 \times 1000m^2$ großes Feld ohne Hindernisse mit 100 zufällig verteilten Knoten erzeugt (weitere Einstellungen in Tabelle 5.1).

Parameter	*Wert*
Feldgröße	$1000 \times 1000m^2$
Anzahl Stationen	100
Verteilung Stationen	zufällig
Senderadius	$250m$
Interferenzradius[1]	$500m$
WLAN-Datenrate	$54MBit/s$
Verlustrate	5%
max. Clients pro Cluster	16
max. Heads pro Client	7
Slot-Länge	$4ms$

Tabelle 5.1: Konfiguration des Simulators

Die Messung wurde dabei im 1-Sekunden-Takt ab dem Moment durchgeführt, wenn die ersten Verbindungen zwischen Knoten geschlossen wurden. Dazu gab jeder simulierte Knoten seine unmittelbare tatsächliche Nachbarschaft, sowie sein komplettes Weltmodell zu jedem Messzeitpunkt aus.

Die prozentuale Belastung des Mediums wurde dadurch erzeugt, dass jeder Knoten mit der vorgegebenen Wahrscheinlichkeit 1-Hop-Datenpakete an die MAC-Schicht übertragen hat, wenn er zum Versand aufgefordert wurde. Die verbleibenden Zeitschlitze wurden für die Propagation des Weltmodells eingesetzt.

Der Versuch wurde zunächst ohne Knotenmobilität ausgeführt, was zu einer geringen Fluktuationsrate der Verbindungen führte. Damit wurde untersucht, wie schnell sich das Weltmodell über alle Knoten ausbreitet. Die Netzwerkbelastung wurde dabei von 0 an in 10%-Schritten erhöht. Der Bereich zwischen 90% und 100% wurde zusätzlich in 1%-Schritten gemessen, um das Verhalten unter extrem hoher Last zu studieren.

Danach wurde der Versuch mit Knotenmobilität wiederholt. Das Mobilitätsmodell besteht dabei darin, dass alle Knoten abwechselnd für eine zufällige Zeit zwischen 10 und 1000 Sekunden auf ihrer Position verharren und sich dann mit einer festen

[1] Der Interferenzradius ist der Abstand, in dem das Signal von anderen Stationen zwar als Belegung des Mediums wahrgenommen wird, aber kein Empfang der Daten mehr möglich ist.

Geschwindigkeit zwischen $3 - 5\frac{m}{s}$ zu einem zufällig bestimmten Zielpunkt innerhalb des Terrains bewegen.

5.1.3 Auswertung

Unter normalen Bedingungen (bei einer Netzbelastung bis 90%, Abb. 5.1) sieht man, dass das Weltmodell sich innerhalb weniger Sekunden über das gesamte Netz ausbreitet, und dass die Ausbreitungsgeschwindigkeit von der Netzwerkbelastung abhängt. Interessant ist, dass der Einfluss der Netzwerkbelastung bis zu 80% nur gering ist – ohne Last ist die Weltmodell-Korrektheit nach 3 Sekunden bei 70%, mit 80% Last nach 10 Sekunden. Nimmt die Netzlast um ein weiteres Zehntel zu, dauert es aber 6 Sekunden länger, bis die Korrektheit 70% erreicht.

Dadurch, dass im Verlauf der Simulation einige Links auf- und abgebaut werden, erreicht die Weltmodell-Korrektheit selbst bei geringer Belastung nicht den Idealwert von 100%.

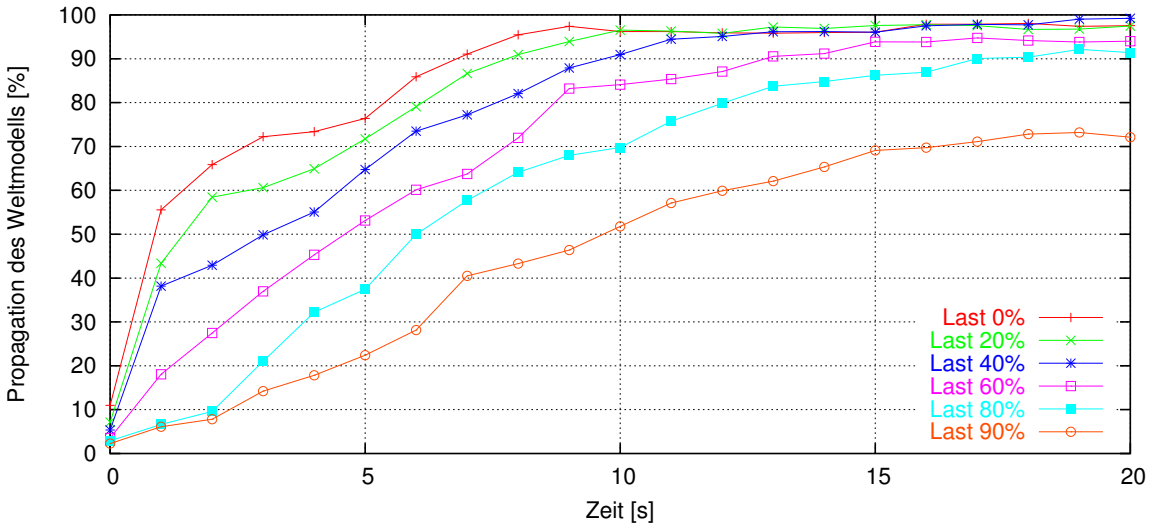

Abbildung 5.1: Propagation des Weltmodells unter geringer Last

Betrachtet man höhere Netzlasten (Abb. 5.2), muss zunächst der Zeitrahmen der Untersuchung ausgedehnt werden, hier auf 90 Sekunden. Dabei kann man feststellen, dass die Qualität des Weltmodells selbst bei einer Netzwerkauslastung von 99% noch stetig besser wird. Bei einer vollständigen Auslastung des Netzwerks (100%, alle Slots werden für höher priorisierte Daten eingesetzt) jedoch kann das Weltmodell nicht mehr propagiert werden, jeder Knoten hat nur seine eigene Nachbarschaft im Weltmodell und kommt damit bei insgesamt 100 Knoten zu einer Kenntnis von 1% der Topologie.

Abbildung 5.3 zeigt beispielhaft die Netzwerktopologie nach 90 Sekunden in einem Szenario ohne Knotenmobilität.

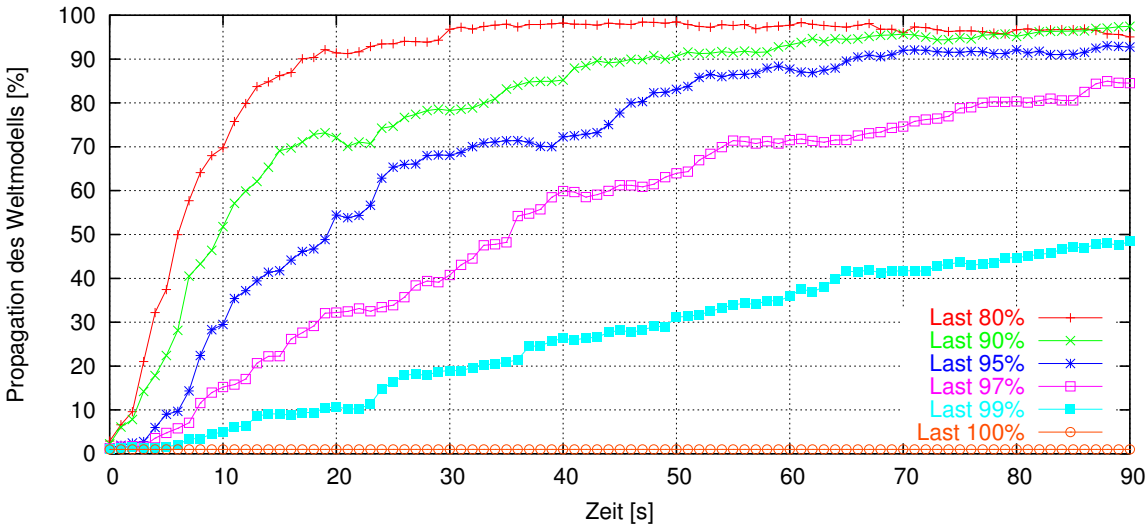

Abbildung 5.2: Propagation des Weltmodells unter hoher Last

Knotenmobilität

Betrachtet man die Ausbreitung der Topologiedaten in einem Netz mit sich bewegenden Knoten (Abb. 5.4), sind die Unterschiede zu still stehenden Teilnehmern bei einer Belastung bis 90% nur marginal. Bei 95% Last erkennt man dagegen, dass die Korrektheit sich zwischen 70 und 80% einstellt. Hier halten sich die änderungsbedingten Topologiedaten und die generierte Netzlast die Waage. Wird die Netzlast erhöht, verschiebt sich dieses Gleichgewicht auf eine kleinere Weltmodell-Korrektheit (60% bei 97% Auslastung). Bei 99% Last wird der Gleichgewichtspunkt im Verlauf der Simulation nicht erreicht. Bei kompletter Auslastung findet genau wie bei der vorangegangenen Messreihe ohne Mobilität keine Ausbreitung der Weltmodelldaten statt.

5.2 Best-Effort-Routing

Verfügen die Knoten über Topologie-Informationen, können sie Datenpakete über das Netz verschicken. Dabei hängt die Qualität der Übertragung von dem Anteil der erfolgreich übertragenen Pakete an den ausgesendeten Paketen und von der Laufzeit vom Sender zum Empfänger ab.

5.2.1 Versuchsaufbau

Um die Übertragungsqualität zu testen, wurden in einem Simulationsexperiment mit den selben Parametern wie zuvor (Tabelle 5.1) zehn Knoten bestimmt, die jeweils zu einem drei Links entfernten anderen Knoten Datenpakete gesendet haben, die

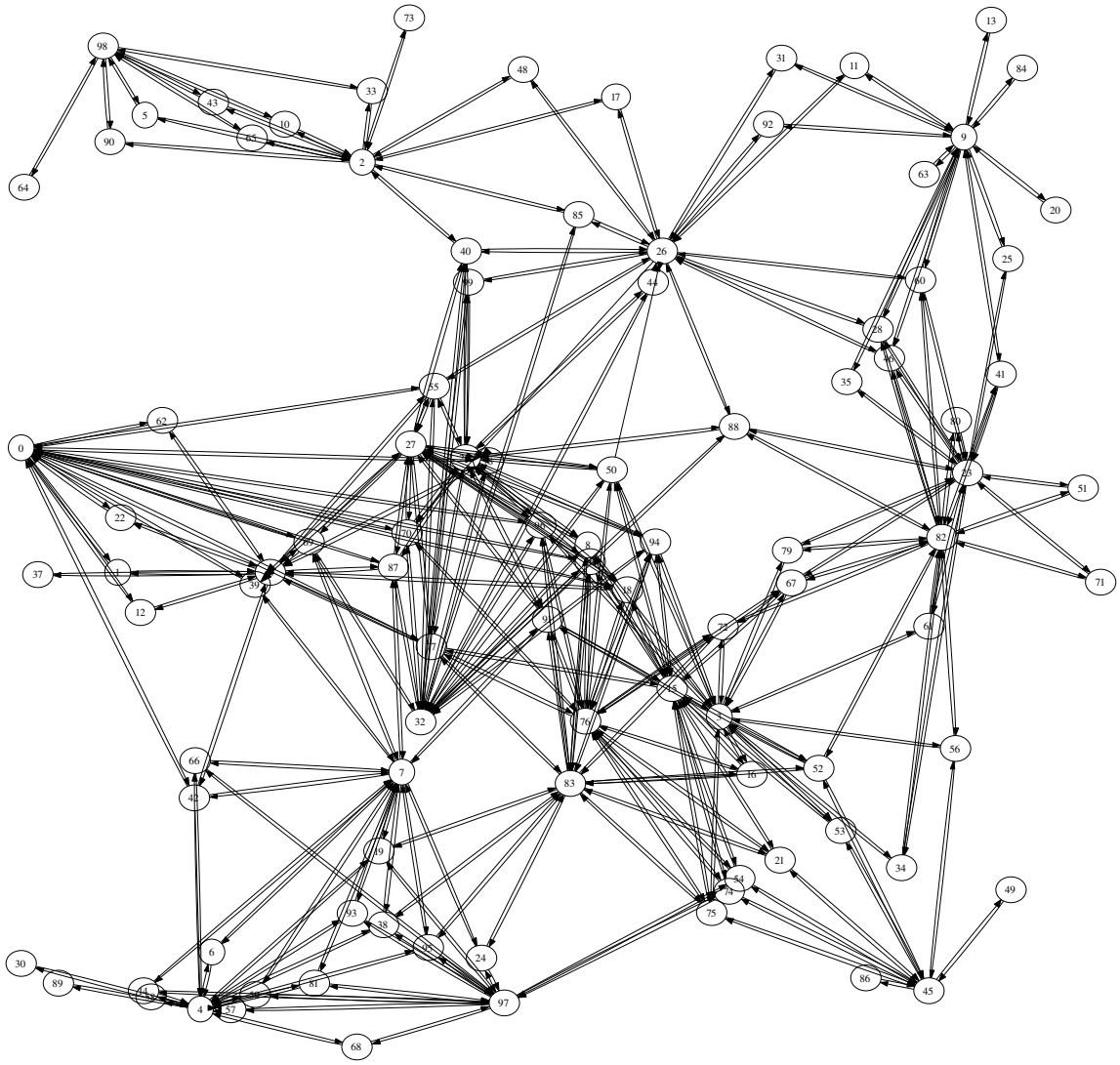

Abbildung 5.3: Netzwerktopologie ohne Mobilität nach 90 Sekunden

vom Zielknoten zu beantworten waren (Ping-Versand mit der Test-Anwendung aus Abschnitt 3.7.11). Durch den Abstand wurde sichergestellt, dass zwischen den kommunizierenden Stationen jeweils ein Gateway und ein Cluster-Head platziert waren. Auf dem Sender wurde im Versuch die summierte Laufzeit für die Übertragung zum Empfänger und zurück gemessen.

Außer den Ping-Paketen wurden auf dem Medium nur Weltmodell-Datenpakete mit einer geringeren Priorität versendet.

Die Gesamtzeit der Simulationsläufe wurde auf 180 Sekunden erhöht. Um Einflüsse eines noch nicht aufgebauten Weltmodells zu minimieren wurde der Versand der Datenpakete erst 20 Sekunden nach Start der Simulation begonnen. Nach 60 Sekun-

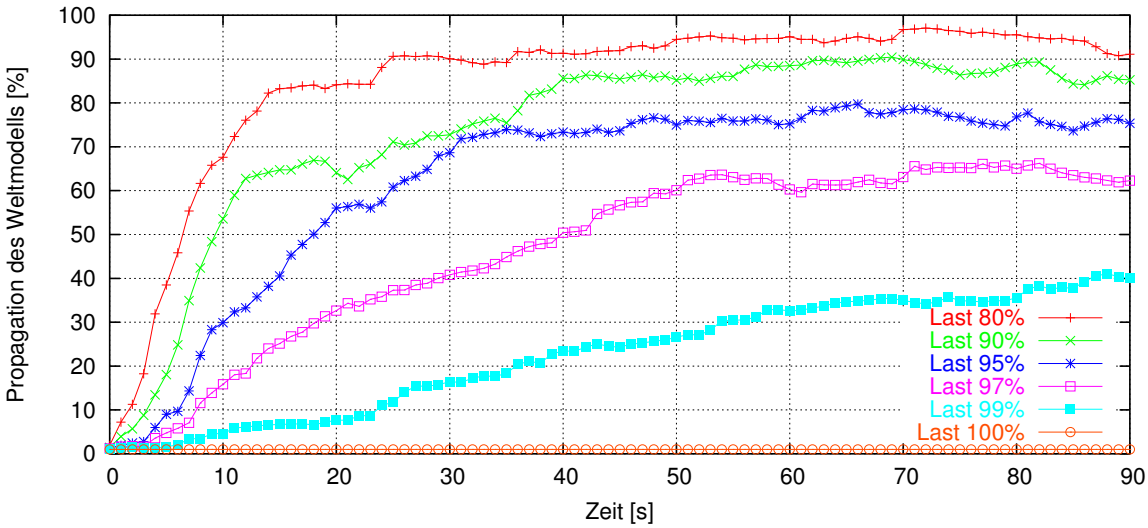

Abbildung 5.4: Propagation des Weltmodells bei Knotenmobilität

den Sendezeit (bei 80s Simulationszeit) wurde der Versand dann wieder eingestellt, um allen Knoten anschließend Zeit zu geben, ihre Sendewarteschlangen zu leeren und alle sich noch auf dem Weg befindenden Pakete auszuliefern.

In mehreren Versuchen wurde dabei nach und nach die Netzwerklast erhöht, indem die Sendewahrscheinlichkeit der 10 Knoten kontinuierlich von 0% auf 100% erhöht wurde. Dieser Wert gibt an, mit welcher Wahrscheinlichkeit ein Knoten ein Ping-Paket versendet, wenn er vom zum Ziel führenden Cluster-Head kontaktiert wird.

5.2.2 Auswertung

In Abbildung 5.5 ist die Anzahl der versendeten Pakete in Abhängigkeit von der Sendewahrscheinlichkeit abgebildet. Man kann sehen, dass auch bei hoher Netzlast fast alle Pakete zugestellt und beantwortet werden. Die Nichtzustellung ist dabei auf inkorrekte Weltmodell-Daten zurückzuführen, deren Präzision mit steigender Belastung des Netzwerks abnimmt.

Abb. 5.6 zeigt die durchschnittlichen Paketlaufzeiten in Abhängigkeit von der Last bei diesem Versuch. Das Diagramm bildet die Mindest- und Höchstwerte sowie den Durchschnitt und die Standardabweichung (senkrechte Striche über dem Durchschnitt) in Abhängigkeit von der Senderate ab. Dabei kann man sehen, dass der Jitter (Differenz zwischen Minimum und Maximum) bei der Best-Effort-Übertragung anfangs moderat ausfällt, aber bei zunehmender Netzwerkbelastung (ab 30%) deutlich ansteigt.

Die Ursache dafür sind zum einen Paketverluste auf dem Medium, die eine Wiederholung des Pakets in der nächsten Runde erforderlich machen, zum anderen aber volllaufende Paketwarteschlangen auf den Gateways, auf denen die Datenpakete nach

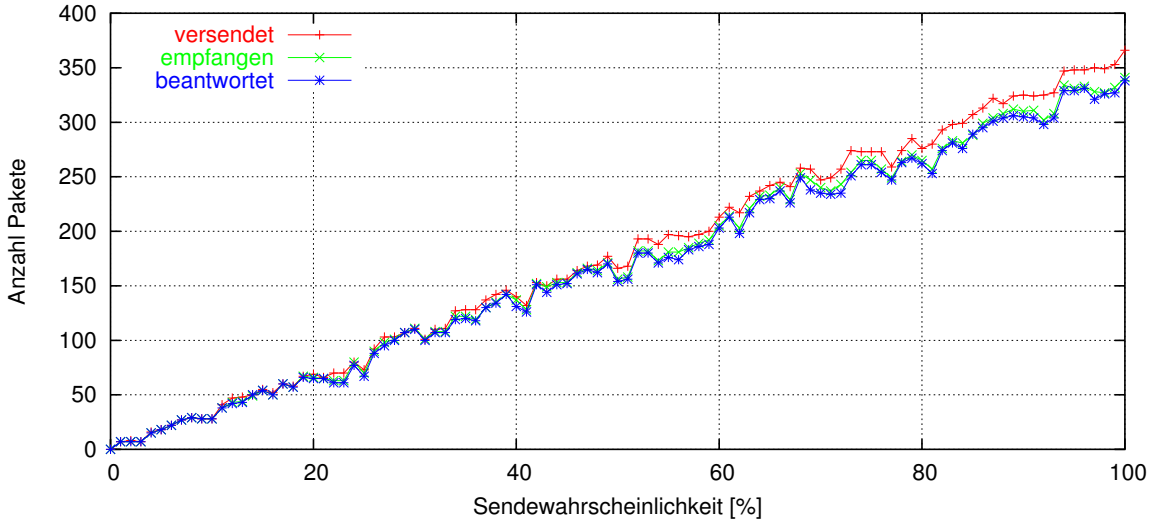

Abbildung 5.5: Multi-Hop-Paketzustellung

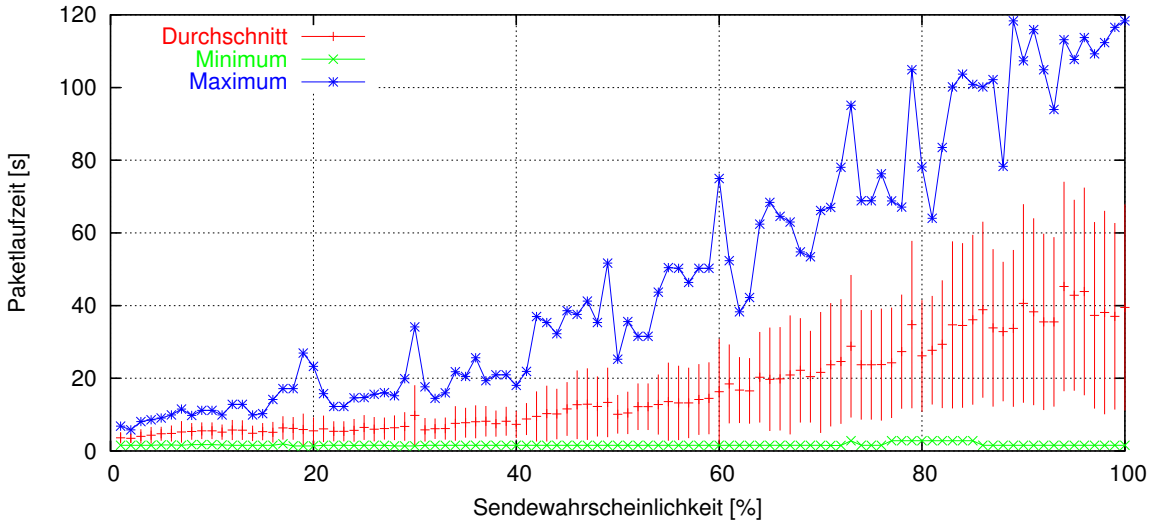

Abbildung 5.6: Multi-Hop-Laufzeiten über 3 Hops

dem FIFO-Prinzip abgearbeitet werden. Die Mindestlaufzeiten bleiben jedoch gering, da zum Anfang und zum Ende der Versuche die Warteschlangen leer sind und die Pakete damit zügig zugestellt werden können. Ist das Netzwerk dagegen ausgelastet, verzögert sich die Auslieferung deutlich.

5.2.3 Knotenmobilität

Eine Wiederholung der Versuchsreihe mit mobilen Knoten (und dem Mobilitätsmodell aus dem Weltmodell-Versuch) hat ähnliche Ergebnisse bezüglich der Zustellraten

(Abb. 5.7) und der Laufzeiten (Abb. 5.8) ergeben. Man kann allerdings sehen, dass bei einer Auslastung von mehr als 90% die Anzahl der korrekt beim Empfänger zugestellten Pakete nicht mit der Senderate zunimmt.

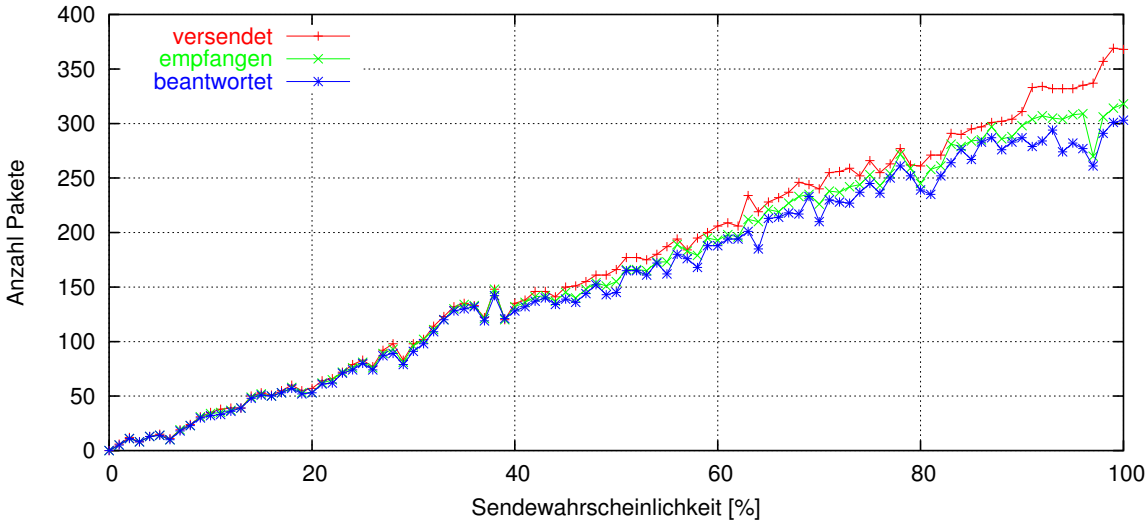

Abbildung 5.7: Multi-Hop-Paketzustellung bei Mobilität

Die Paketlaufzeiten verschlechtern sich dagegen nicht, wenn die Knoten sich bewegen. Auch das ist ein Hinweis darauf, dass die langen Laufzeiten in erster Linie von den vollen Warteschlangen und nicht von anderen Faktoren abhängen. Die maximalen Zustellzeiten sind bei mobilen Knoten z.T. höher, da sich die kommunizierenden Stationen voneinander entfernen können und die Pakete deshalb mehr Zwischenstationen passieren müssen.

Die unkalkulierbaren und z.T. sehr langen Zustellzeiten lassen sich mit den Best-Effort-Mitteln jedoch nicht umgehen, da bei einer Verkleinerung der Warteschlangen Pakete verworfen werden und bei großen Warteschlangen Pakete lange unterwegs sind.

5.3 Quality of Service

Um die Latenzzeiten der Datenübertragungen auch ohne das Verwerfen von Datenpaketen gering zu halten, muss ein Prioritätsschema eingesetzt werden, welches Überlastungen an einzelnen Knoten vermeidet. Dieses wird vom QoS-Routing-Modul bereitgestellt, das Verbindungen um Engpässe herum aufbaut und damit einem Volllaufen der Warteschlangen entgegenwirkt. In diesem Abschnitt soll die Qualität dieses Verfahrens überprüft und mit dem Datenversand ohne Garantien in Relation gesetzt werden.

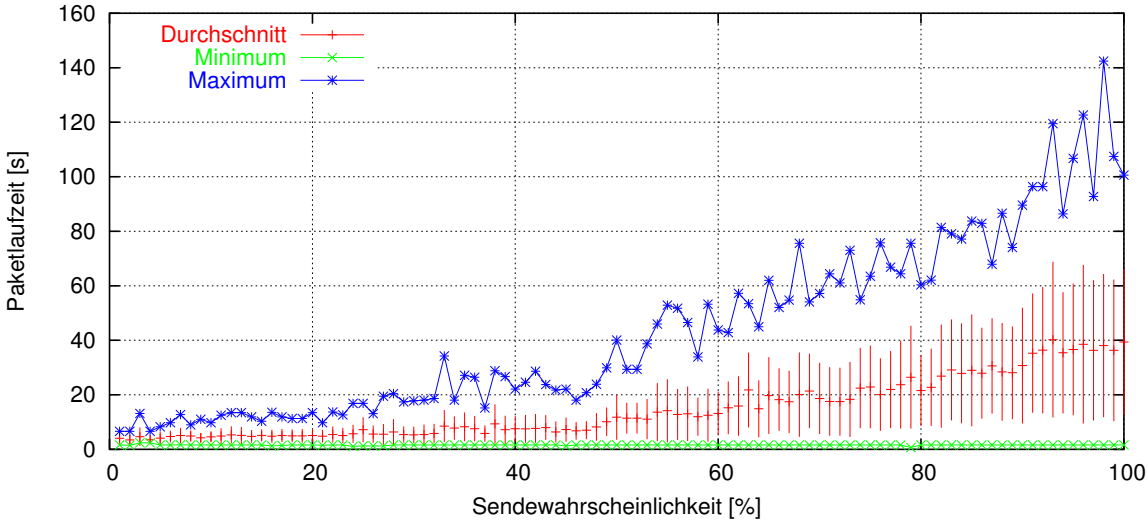

Abbildung 5.8: Multi-Hop-Laufzeiten bei Mobilität

5.3.1 Versuchsaufbau

Zum Testen der Skalierbarkeit wurde in aufeinander folgenden Versuchen die Anzahl der aufzubauenden QoS-Verbindungen von 0 auf die Anzahl der Knoten, also 100, erhöht. Die Sender- und Empfängerknoten (und damit die Pfadlängen) wurden dabei zufällig ausgesucht.

Da QoS-Verbindungen nur in eine Richtung bestehen, wurde statt einem bidirektionalen Paketfluss nur die Laufzeit vom Sender zum Empfänger gemessen. Um die Messwerte mit dem Best-Effort-Routing vergleichen zu können, wurde mit letzterem die selbe Versuchsreihe durchgeführt, bei der Daten direkt (und damit ohne Garantien) versendet wurden. Die Senderate entsprach dabei der Datenrate der QoS-Strecken und die Anzahl der sendenden Knoten wurde nach und nach erhöht.

Um die beiden Verfahren zu vergleichen, wurden die maximalen und durchschnittlichen Paketlaufzeiten gemessen und die Anzahl der ausgelieferten Pakete vermerkt. Beim QoS-Routing wurden zusätzlich die für den Verbindungsaufbau nötigen Zeiten erfasst.

Die Übertragung bzw. Routenreservierung begann 20 Sekunden nach Simulationsstart und wurde 60 Sekunden später eingestellt. Die Gesamtdauer jedes Experiments war 180 Sekunden. Die Versuchsreihen wurden sowohl mit statisch positionierten Knoten als auch mit dem zuvor genutzten Mobilitätsmodell durchgeführt. Da die Ergebnisse sehr ähnlich ausgefallen sind, wird im Folgenden nur auf die Experimente mit beweglichen Knoten eingegangen.

5.3.2 Auswertung

Wie man in Abbildung 5.9 sehen kann, steigen die maximalen und durchschnittlichen Übertragungszeiten bei Best-Effort-Transfers mit der Netzwerkauslastung an. Ihr Anstieg ist in etwa linear, zur besseren Veranschaulichung der stark gestreuten Best-Effort-Maximalzeiten wurde eine Annäherungsgerade eingefügt. Die QoS-Übertragungen bieten dagegen selbst bei hoher Netzwerkbelastung und weiten Verbindungen stabile maximale Latenzzeiten.

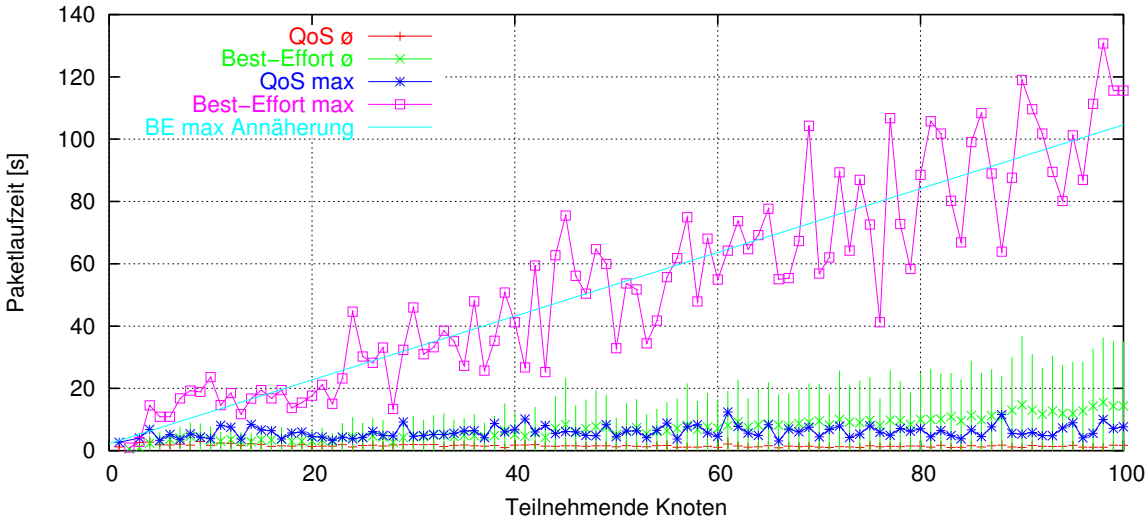

Abbildung 5.9: Paketlaufzeiten QoS und Best-Effort

Obwohl die durchschnittlichen Laufzeiten der Best-Effort-Pakete in der selben Größenordnung liegen, kann für deren Zustellung keine Garantie gegeben werden und einzelne Pakete benötigen deutlich länger. Im Gegenzug dazu wurden alle QoS-Pakete in weniger als 13s zugestellt.

Dieser Wert erscheint zwar für normale Netzwerke recht hoch, ergibt sich aber in erster Linie aus dem Zeitschlitz-Verfahren der teilnehmenden Cluster. Durch das nicht vom QoS-Modul beeinflussbare Polling-Schema verbleibt jedes Datenpaket einige Zeit auf jedem Gateway des Pfades. Bei Übertragungsfehlern wird die Wartezeit bis zur Wiederholung nochmal deutlich erhöht.

Einen deutlich geringeren Einfluss auf die Latenz haben der Versand der Pakete auf dem Medium und die Verarbeitung auf den Zwischenstationen. Da diese Verzögerungen alle Pakete unabhängig von der Cluster-Zusammenarbeit betreffen, gelten sie auch für die schnellsten Pakete, deren Laufzeiten deutlich unter einer Sekunde bleiben (Abb. 5.10). Lediglich bei weniger als 10 Knoten ist die Anzahl der versendeten Pakete so klein, dass nur ungünstige Cluster-Konstellationen auftreten.

Betrachtet man die Menge der erfolgreich zugestellten Datenpakete (Abb. 5.11), ergibt sich ein gänzlich anderes Bild: Während die Paketmenge bei Best-Effort-Übertragungen mit der Anzahl der Teilnehmer steigt, stagniert sie bei QoS-Verbindungen

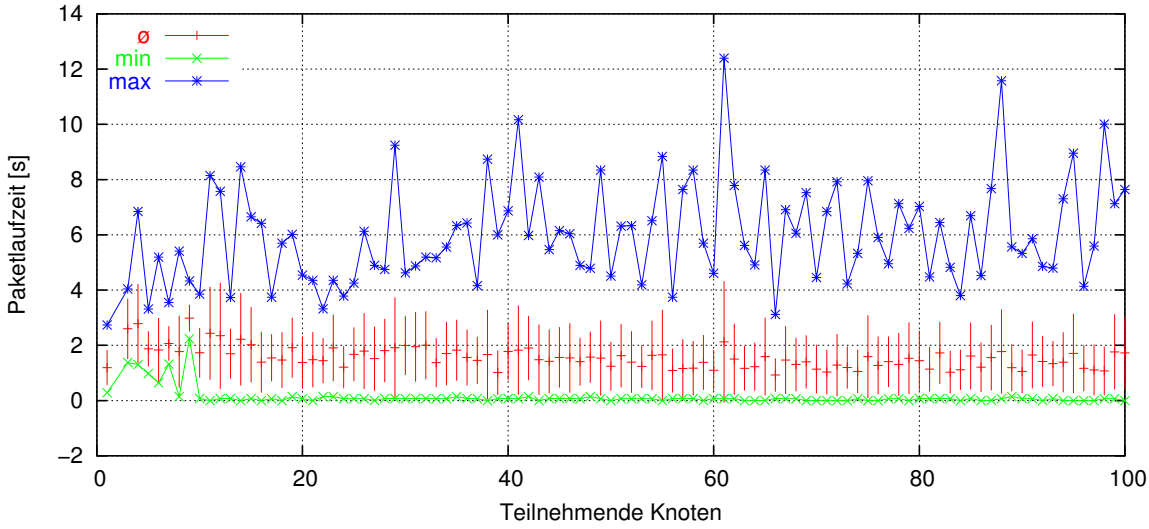

Abbildung 5.10: Paketlaufzeiten von QoS-Verbindungen

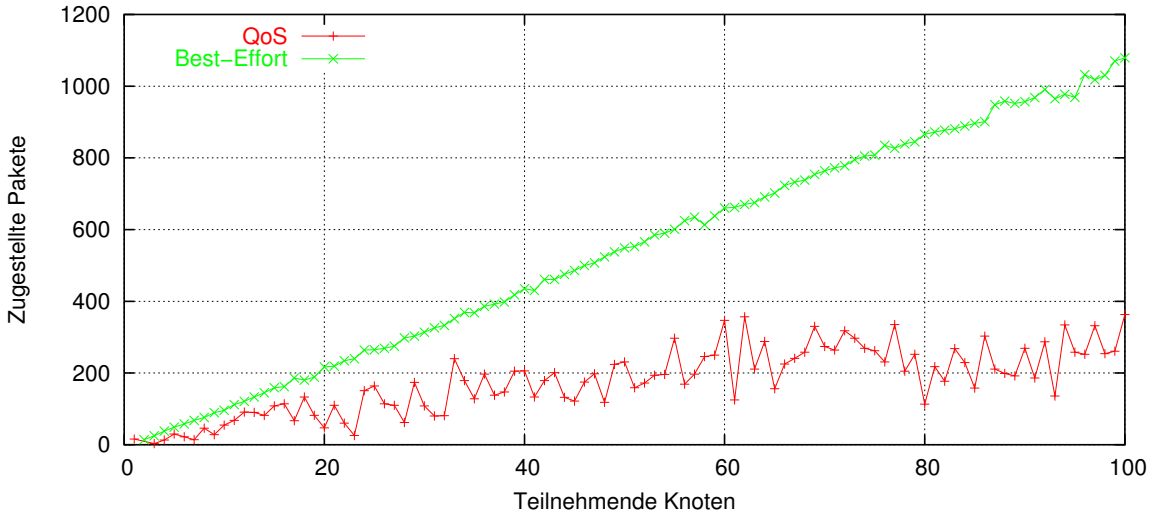

Abbildung 5.11: Zugestellte Pakete QoS und Best-Effort

bei mehr als 50 aktiven Stationen. Dies beruht darauf, dass nur dann Datenpakete versendet werden können, wenn eine QoS-Verbindung aufgebaut ist. Versuchen mehr als 50 Knoten, an der Kommunikation teilzunehmen, ist das Netzwerk ausgelastet und weitere Verbindungsversuche werden abgelehnt.

5.4 Ergebnisse

Die Evaluierung hat gezeigt, dass die Teilmodule gut funktionieren und auch ihre Zusammenarbeit problemlos erledigen. Die Propagation der Topologiedaten klappt

auch bei hoher Knotenmobilität und starker Auslastung des Netzwerks; lediglich wenn alle Teilnehmer permanent Daten senden, können keine lokalen Weltmodelle mehr aufgebaut werden.

Das Weiterleiten von Datenpaketen ohne Garantien funktioniert auch, allerdings tritt bei steigender Belastung des Mediums eine Warteschlangensättigung auf den Gateway-Knoten ein: Es werden mehr Datenpakete eingeliefert als versendet werden können. Dieser Effekt führt dazu, dass die Zustellung der eigenen Datenpakete fast beliebig verzögert werden kann, oder dass die Pakete durch den Überlauf einer Warteschlange gar nicht ankommen.

Versendet man seine Daten über eine QoS-Verbindung, so sind die Latenzzeiten dagegen genau so kalkulierbar wie die erreichbare Datenrate. Allerdings ist die Anzahl der QoS-Verbindungen, die durch ein Netzwerk führen können, stark limitiert und der Aufbau solcher Verbindungen ist mit einem erhöhten Aufwand verbunden.

Aus diesem Grund sollte beim Design von Anwendungen besonders sorgfältig abgewogen werden, ob eine QoS-Verbindung für bestimmte Datentransfers erforderlich ist, oder ob die Informationen auf dem Best-Effort-Kanal versendet werden können.

6 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde in mehreren Schritten die Basis für ein Quality-Of-Service-Routing-Protokoll in einer Clustering-Umgebung erarbeitet und ein solches Protokoll implementiert.

In Kapitel 2 wurden zunächst die Grundlagen für diese Arbeit erläutert und bereits existierende Routing- und QoS-Protokolle vorgestellt. Anschließend wurde das Intracluster-Protokoll, auf dem die Arbeit aufbaut und die Umgebung, in der die Simulation der Protokolle stattfindet, erläutert.

Im dritten Kapitel wurde ein Propagationsverfahren für Topologiedaten entwickelt, das allen am mobilen Netzwerk teilnehmenden Knoten eine aktuelle und möglichst komplette Sicht auf die Netzwerkstruktur bietet und dafür die ansonsten ungenutzten Zeitschlitze des Clustering-Protokolls verwendet.

Dieses Verfahren funktioniert selbst unter ungünstigen Bedingungen wie Knotenmobilität und Netzbelastungen über 90% und erlaubt den Knoten dabei, präzises Wissen über mehr als zwei Drittel des Netzes zu haben, während die Informationen über den Rest zwar veraltet, aber nicht komplett irreführend sind.

Aufbauend auf der Weltmodell-Propagation wurde ein Routingverfahren entwickelt, um Datenpakete zwischen beliebigen Knoten auszutauschen. Die Topologiedaten des Weltmodells werden dabei auf jedem teilnehmenden Knoten verwendet, um die kürzeste Route für ein Datenpaket zu bestimmen und das Paket auf dieser Route weiterzuleiten. Das Zusammenspiel der Stationen erlaubt es somit, Datenpakete über Entfernungen zu versenden, die deutlich die Sende-Reichweite eines einzelnen Knotens übersteigen.

Obwohl dieses Verfahren die meisten versendeten Pakete zustellt, gibt es weder Bandbreitengarantien, noch kann mit einer Obergrenze für die Paket-Laufzeit gerechnet werden, da diese von der Auslastung des Netzwerks abhängt.

Aus diesen Gründen wurde im vierten Kapitel ein Verfahren ausgearbeitet, bei dem mit Hilfe von Reservierungen Bandbreite und Latenzzeit für Datenverbindungen garantiert werden können. Dieses QoS-Protokoll sucht verteilt die kürzeste Strecke, die die Bandbreitenanforderung erfüllt und dabei weniger als die vorgegebene Anzahl an Zwischenstationen hat. Existiert eine solche Strecke, wird die Bandbreite dort gleich registriert und die Anwendung in Kenntnis gesetzt. Gibt es keine passende Verbindung oder bricht diese nach ihrem Aufbau zusammen, wird die Anwendung wiederum benachrichtigt.

Schließlich wurde in Kapitel 5 die Funktionsfähigkeit der zuvor entwickelten Protokolle in unterschiedlichen Testszenarien im Simulator unter Beweis gestellt. Dazu

wurden die drei Teilmodule in aufeinander folgenden Versuchsreihen mit unterschiedlicher Netzwerkbelastung und mit bzw. ohne Knotenmobilität untersucht und die Ergebnisse dargestellt und ausgewertet.

Die Evaluierung hat gezeigt, dass sowohl die Propagation der Topologiedaten als auch die beiden Routing-Module für Best-Effort-Daten und QoS-Verbindungen erwartungsgemäß funktionieren und auch bei sich bewegenden Stationen ihre Arbeit erfüllen.

Ausblick

Im Lauf der Entwicklung des QoS-Protokolls wurden einige Einschränkungen festgestellt, deren Behebung über den Rahmen dieser Arbeit hinausgeht. Außerdem haben sich einige Weiterentwicklungsmöglichkeiten für nachfolgende Projekte ergeben.

Die interessanteste Perspektive ist eine Erweiterung des QoS-Protokolls um Mehrpfadausbreitung. Dies würde nicht nur die Möglichkeit bieten, Multicast-Datenströme zu implementieren, sondern auch durch parallelen Versand zum selben Ziel die Ausfallsicherheit steigern. Darauf aufbauend ließe sich auch eine dynamische Link-Reparatur ohne Paketverluste realisieren – fällt eine redundante Teilstrecke aus, so kann das erkannt und ein alternativer Pfad dafür gesucht werden, während die Daten auf der noch bestehenden Verbindung übertragen werden.

Es wurde festgestellt, dass der Jitter bei der Datenübertragung durch die Asynchronität zwischen den Cluster-Heads recht hoch ist. Um dieses Problem zu beheben, ist es denkbar, ein Signalisierungsverfahren für die Polling-Reihenfolge zwischen Client und Head zu implementieren. Damit könnte ein Gateway, über das eine reservierte Route führt, das Abfrage-Schema seiner Heads so anpassen, dass die Datenpakete der garantierten Verbindung nach ihrer Einlieferung sofort vom Ziel-Cluster abgefragt werden. Dies würde die maximale Paketlaufzeit zwar nur dann signifikant verbessern, wenn keine Medienfehler auftreten, allerdings ließe sich der Jitter dann bei bekannten Eigenschaften des Mediums deutlich besser berechnen.

Denkbar wäre auch die Benutzung des QoS-Protokolls auf einer anderen MAC-Schicht als dem Clustering-Schema. Da das Weltmodell mit einer flachen Hierarchie arbeitet, wären nur geringe Anpassungen am Routing notwendig, um ein anderes Protokoll als Basis einzusetzen. Ein solches Protokoll müsste allerdings konsistente Nachbarschaftsinformationen liefern und Bandbreitenreservierungen für lokale Links durchführen können.

Literaturverzeichnis

[BBC+98] Blake, S., D. Black, M. Calson, E. Davies, Z. Wang und W. Weiss: *An Architecture for Differentiated Services*. RFC 2475, Dezember 1998. Network Working Group.

[BCS94] Braden, R., D. Clark und S. Shenker: *Integrated Services in the Internet Architecture: an Overview*. RFC 1633, Juli 1994. Network Working Group.

[Bec] Bechler, Marc: *implementation of the internet drafts „draft-ietf-manet-fsr-01.txt" and „..-02.txt"*. TU Braunschweig, Institute of Operating Systems and Computer Networks (IBR). http://whale.hit.bme.hu/cgi-bin/contrib.pl?dir=models&txt=FSR-20021124.

[CeA+03] Clausen, T., P. Jacquet (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum und L.Viennot: *Optimized Link State Routing Protocol (OLSR)*. RFC 3626, Oktober 2003. Network Working Group.

[COR] CORPORATE IEEE, Institute of Electrical and Electronics Engineers, Inc. Staff: *IEEE Standard for Information Technology - Portable Operating System Interface (POSIX): System Application Program Interface (API), IEEE Std 1003.1*. IEEE Standards Office, New York, NY, USA.

[Dij59] Dijkstra, E. W.: *A note on two problems in connection with graphs*. Numerische Mathematik, 1:269–271, 1959.

[Fee99] Feeney, Laura Marie: *A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks*. Technischer Bericht T1999:07, Swedish Institute of Computer Science, Oktober 1999.

[HM05] Herms, Andre und Daniel Mahrenholz: *Unified Development and Deployment of Network Protocols*. In: *Meshnets '05*, Budapest, Hungary, July 2005.

[HOR] HORISAWA, Shingo: *GSR/FSR Implementation for Linux*. ATR Adaptive Communications Research Laboratories. http://www.acr.atr.jp/ masayama/download/gsrfsr.html.

[HPS02a] Haas, Zygmunt J., Marc R. Pearlman und Prince Samar: *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. Juli 2002.

[HPS02b] Haas, Zygmunt J., Marc R. Pearlman und Prince Samar: *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. IETF MANET Internet Draft, Juli 2002. http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt.

[IEE] IEEE: *IEEE Std. 802.11b, Higher-Speed Physical Layer Externsion in the 2.4 GHz Band* . New York.

[LAZC00] Lee, Seoung-Bum, Gahng-Seop Ahn, Xiaowei Zhang und Andrew T. Campbell: *INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks*. Journal of Parallel and Distributed Computing, 60:374–406, 2000.

[NAM] *Nam: Network Animator*. http://www.isi.edu/nsnam/nam/.

[NS2] *The Network Simulator ns-2*. http://www.isi.edu/nsnam/ns/.

[PB94] Perkins, Charles und Pravin Bhagwat: *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*. In: *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, Seiten 234–244, 1994.

[PGC00] Pei, Guangyu, Mario Gerla und Tsu-Wei Chen: *Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks*. In: *ICC (1)*, Seiten 70–74, 2000.

[PGH00] Pei, G., M. Gerla und X. Hong: *LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility*. In: *IEEE/ACM MobiHOC 2000*, Seiten 11–18, August 2000.

[PRD99] Perkins, Charles E., Elizabeth M. Royer und Samir R. Das: *Ad hoc on demand distance vector routing*. In: *2nd IEEE Workshop on Mobile Computing Systems and Applications*, Seiten 99–100, 1999.

[PRD03] Perkins, Charles E., Elizabeth M. Royer und Samir R. Das: *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF MANET Internet Draft, Februar 2003. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt.

[Sch98] Schmid, Andreas: *Solution for the counting to infinity problem of distance vector routing.* Fachberichte Informatik 7–98, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 1998.

[SPH04] Samar, Prince, Marc R. Pearlman, and Zygmunt J. Haas: *Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks.* IEEE/ACM Trans. Netw., 12(4):595–608, 2004.

[SSB99] Sinha, Prasun, Raghupathy Sivakumar, and Vaduvur Bharghavan: *CEDAR: a core-extraction distributed ad hoc routing algorithm.* In *INFOCOM (1)*, pages 202–209, 1999.

[Str00] Stroustrup, Bjarne: *Die C++ Programmiersprache.* Addison-Wesley Verlag, München, 4. edition, 2000.

[SXA04] Stuedi, Patrick, Jianbo Xue, and Gustavo Alonso: *ASAP - Adaptive QoS Support with Reduced Reservation Overhead in MANETs.* Technical report, Swiss Federal Institute of Technology (ETHZ), 2004.

[TK75] Tobagi, F. and L. Kleinrock: *Packet Switching in Radio Channels: Part II–The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution.* IEEE Transactions on Communications, 23(12):1417–1433, December 1975.

[Tri02] Trikaliotis, Spiro: *Performanz eines fehlertoleranten WLAN-Gruppenkommunikationsprotokolls bei Störungen durch Bluetooth.* In *ARCS*, Karlsruhe, Germany, April 2002.

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit „Routing mit Qualitätsgarantien in mobilen Ad-Hoc-Netzwerken" selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, sowie alle Zitate entsprechend kenntlich gemacht habe.

Magdeburg, 17.10.2005
Georg Lukas