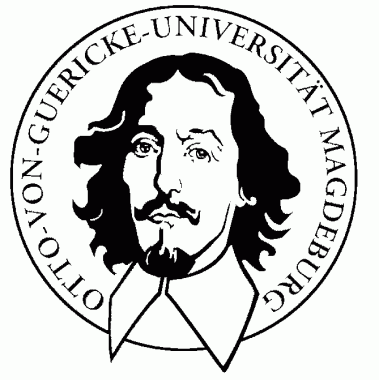
Otto-von-Guericke-Universität Magdeburg



Fakultät für Informatik
Institut für verteilte Systeme

Diplomarbeit

Vergleich von Routingmetriken in drahtlosen Maschennetzwerken

Jens Wienöbst
5. Oktober 2007

Betreuer
Dipl.-Inform. André Herms
Prof. Dr. Edgar Nett

Inhaltsverzeichnis

Abkürzungsverzeichnis

AARF adaptive auto rate fallback, 1
AODV Ad-hoc On-Demand Distance Vector Routing, 1
AP access point, 1
ARF auto rate fallback, Ein Mechanismus, der die Senderrate im physical layer automatisch anpasst, 1
ARF auto rate fallback, 1
ASK amplitude shift keying, 1

BER bit error ratio, 1
BFSK binary frequency shift keying, 1

CS channel select, 1
CS-ACK channel select acknowledgement, 1
CSMA/CD carrier sense multiple access/ collision detection, 1

DB Database, 51
DMBS Database Management System, 40

FSK frequency shift keying, 1

GEA generic event api, eine Schnittstelle zum ns2 und Realsystemen, 1

LPTT link predicted transmission time, 1

MANET mobile ad hoc network, 1
MPR multipoint relay, 1
MPRS multipoint relay selector, 1

NS2 Netzwerksimulator2, arbeitet mit diskreten Ereignissen, zur Erforschung von Netzwerken, siehe: http://www.isi.edu/nsnam/ns/, 1

OLSR	optimized link state routing, 1
Path-Loss	Path Loss bezeichnet den Soignalverlust auf der Strecke zwischen Sender und Empfänger, siehe auch Propagationsmodell, 29
PPTT	path predicted transmission time, 1
PSK	phase shift keying, 1
psql	Terminalbasiertes Frontend für PostgreSQL, 53
RBAR	receiver based auto rate, 1
RTT	round-trip tim, 1
SNR	signal to noise ratio, 1

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Maschennetzwerke bieten die Möglichkeit einer robusten und gleichzeitig kostengünstigen Netzwerkinfrastruktur in vielfältigsten Bereichen. Durch die drahtlose Kommunikation sind für die Installation eines solchen Netzwerkes keinerlei bauliche Veränderungen der Umgebung notwendig und zusätzlich ist Mobilität in einem gewissen Rahmen möglich. Diese Eigenschaften, durch die Maschennetzwerke sehr flexibel und leicht einzurichten sind, sorgen für eine immer größere Verbreitung und häufigeren Einsatz solcher Netze; ein ehrgeiziges Projekt und Beispiel dafür ist OLPC [olp] - „One Laptop Per Child“ - durch ein Maschennetzwerk zwischen diesen Computern soll auch - sozusagen for free - eine hochverfügbare Internetverbindung eingerichtet werden. Weitere Beispiele sind so genannte „Gemeinschaftsnetze“ [Gro] [fre] [MIT] [KSK04] und Maschennetzwerke in Produktionsanlagen.
Durch die Selbstorganisation eines Maschennetzwerkes sind diese Netze sehr robust gegenüber Störungen, wie zum Beispiel dem Ausfall eines Knotens, ebenso werden neue Knoten automatisch in das Netz integriert. Die Vermaschung des Netzwerkes ist zugleich Stärke wie Schwäche, eine große Anzahl von Links zwischen den Knoten verbessert die Erreichbarkeit eines Knotens, im Gegenzug wird aber auch die Routenfindung erschwert. Zum einen durch die schiere Anzahl möglicher Routen aus denen die Beste heraus gesucht werden muss, zum anderen auch durch die unterschiedliche Qualität der Links und die gegenseitige Verdrängung; in dem Broadcastmedium kann immer nur ein Sender aktiv sein, da gleichzeitiges Senden die Nachrichten verfälschen bzw. zerstören würde.
Die Datenrate und die Datenverlustrate zwischen verschiedenen Knoten differieren stark - sowohl von Knoten zu Knoten, als auch in der Zeit - und beeinflussen die Qualität der Kommunikation auf der gewählten Route entsprechend. In einem Routingprotokoll ist es die Metrik, die die Qualität der Links bewertet und so dass Protokoll in die Lage versetzt, Routen mit unterschiedlicher Qualität zu vergleichen. Eine Vielzahl von Routingmetriken hat sich dieser Probleme angenommen und bewertet die Links auf unterschiedliche Art und Weise, mit unterschiedlichem Erfolg.
Die Komplexität des Problems hat viele Metriken als mögliche Lösungen produziert, diese werden alle in unterschiedlichen Arbeiten vorgestellt bzw. entwickelt und nur unzureichend miteinander verglichen. Für den gezielten und erfolgreichen Einsatz einer Metrik ist ein aussagekräftiger Vergleich aber notwendig. Nur wenn die Metriken in identischen Szenarien eingesetzt und überprüft werden kann eine Aussage darüber getroffen werden,

welche Metrik in welchem Szenario von Vorteil ist und welche Eigenschaften der Knoten und der Netzwerke Einfluss auf die Funktion der Metrik haben.

1.2 Aufgabenstellung

Die Aufgabenstellung für diese Diplomarbeit umfasst die Zusammenstellung vorhandener Metriken für drahtlose Netzwerke, die Implementierung und den Vergleich ausgewählter Metriken in dafür entworfenen Testszenarien. Des Weiteren soll eine Metrik speziell für eine WLAN-Karte, die es ermöglicht die Übertragungsdauer für eine Transmission direkt auszuwerten[1], implementiert und mit den vorhandenen verglichen werden. Alle Metriken sollen mit Hilfe von GEA und dem NS2 implementiert, simuliert und getestet werden. Durch diese Implementierung soll es möglich sein, diese Metriken sowohl in Simulationen, als auch in realen Netzwerken einzusetzen, so dass ein aussagekräftiger Vergleich der Metriken stattfinden kann und metrikbeeinflussende Eigenschaften eines Netzwerkes und der Knoten identifiziert werden können. Neben dem Vergleich der Metriken sollen Eigenschaften ermittelt werden, die Metriken in Maschennetzwerken mitbringen müssen, um besonders gute Routen zu liefern. Der Netzwerksimulator muss außerdem so angepasst werden, dass Übertragungen mit verschiedenen Senderaten und entsprechenden Auto-Rate-Mechanismen simuliert werden können.

1.3 Ergebnis

Die Wichtigkeit von Routingmetriken ist allein schon durch die große Anzahl von verschiedenen Metriken und den dazugehörigen Arbeiten offensichtlich; jede Metrik versucht einen bestimmten Aspekt des Netzwerks, in dem sie eingesetzt wird, besser zu berücksichtigen als es bisherige Metriken schaffen. Das unterschiedliche Metriken Routen von verschiedener Qualität liefern und deshalb eine geeignete Metrik für das vorliegende Netzwerk gewählt werden muss, konnte auch in dieser Arbeit bestätigt werden. Weiter wurde gezeigt, dass die automatische Anpassung der Senderaten aktuelle Metriken stark negativ beeinflusst.
Durch den Vergleich von gleich 6 Metriken in einem Multirate-Maschennetzwerk konnten einige Eigenschaften von Metriken identifiziert werden, die für eine sinnvolle Bewertung von Links in Multirate-Netzwerken notwendig sind:

- Übertragungsdauer als Linkgewicht
- Direkte Messung[1]

[1] Normalerweise wird die Übertragungsdauer durch Testübertragungen ausgemessen, dies geschieht aber nicht so exakt, wie eine Hardware gestützte Messung (siehe Kapitel 2.6 Routingmetriken).

- Gleichverteilte Messungen

Insbesondere ist eine ungleich verteile Messung negativ aufgefallen, da durch die Ratenadaptionsmechanismen regelmäßige Übertragungsfehler auftreten, die zu fehlerhafter Linkbewertung führen.
Weiter wurde eine Möglichkeit vorgestellt, mit der Übertragungsfehler berücksichtigt und ungleich verteilte Messungen ausgeglichen werden können. Zum Ausgleich der Ungleichheit der Messung könnte eine Übertragungswahrscheinlichkeit für die eingesetzte Rate benutzt werden.
Diese und neue Routingmetriken müssen weiter gegeneinander verglichen werden. Metriken haben Parameter, die das Verhalten entsprechend beeinflussen: Dies konnte hier nicht erschöpfend betrachtet werden, da die Menge an notwendigen Simulationen den Rahmen sprengen würde. Außerdem scheint die Möglichkeit, mit der Übertragungsfehler berücksichtigt werden können gerade für Multirate-Netzwerke sehr interessant zu sein, so dass eine Implementierung und Messung einer entsprechenden Metrik sinnvoll ist.

2 Grundlagen

2.1 Drahtlose Netzwerke

Ein Wireless-Lokal-Area-Network (WLAN) ist ein drahtloses lokales Funknetz, ein Standard für solche Netze ist der IEEE 802.11 [IEE]. Die erste Version wurde 1997 verabschiedet. Der IEEE 802.11 spezifiziert den Mediumzugriff und die physikalische Schicht eines Funknetzwerks.
Funknetzwerke können im Allgemeinen in zwei verschiedenen Modi betrieben werden, im **Infrastrukturmodus** und im **Ad-hoc-Modus**.

Infrastrukturmodus Der Infrastrukturmodus ist ähnlich dem eines drahtgebundenen Netzwerks, es findet eine klare Trennung zwischen Routerknoten und Endknoten statt. Einige Knoten sind so genannte **Access-Points** und für den Anschluss neuer Knoten verantwortlich. Nur über die Access-Points können weitere Knoten als Clients in diese Netze eintreten. Netze im Infrastrukturmodus sind auf die Sendereichweite der Access-Points beschränkt. Diese sind im Allgemeinen miteinander drahtgebunden vernetzt, so dass ein Infrastrukturnetzwerk wenig Flexibilität bezüglich der Ausdehnung des Netzwerks bereitstellt.

Ad-hoc-Modus Im so genannten Ad-hoc-Modus übernimmt jeder Knoten im Netzwerk auch die Access-Point Funktionalität. Zwei Knoten im Ad-hoc-Modus können direkt miteinander ein Netz bilden und ermöglichen das Hinzufügen weiterer Knoten. Im Standard Ad-hoc-Modus ist keine Routingfunktionalität enthalten, Knoten, die nicht direkt verbunden sind, können nicht miteinander kommunizieren.

Meshed-Networks Maschennetzwerke (Meshed-Networks) sind sich selbst organisierende Netzwerke. Die Knoten werden im Ad-hoc-Modus betrieben und erkennen neue Teilnehmer selbstständig und integrieren diese in das bestehende Netz. Kommunikation ist über mehrere Knoten hinweg möglich, so dass Daten ausgetauscht und Dienste in Anspruch genommen werden können. Zum Beispiel kann ein Knoten eine Internetverbindung anbieten, die von allen anderen Knoten im Netzwerk genutzt werden kann, ohne direkt mit diesem Knoten verbunden zu sein.
Maschennetzwerke sind bezüglich ihrer Einrichtung sehr komfortabel, da sie keine manuelle Konfiguration benötigen; dafür müssen die Netzwerkknoten wesentlich mehr und

komplexere Netzwerkprotokolle beherrschen und bereitstellen. Dazu zählt zum Beispiel das eingesetzte Routingprotokoll (siehe Kapitel 2.5 Routingprotokolle).

MANet Mobile-Ad-hoc-Netze unterscheiden sich von den Maschennetzwerken dadurch, dass die am Netzwerk teilnehmenden Knoten außerdem noch mobil sind. Die Konnektivität in einem MANet verändert sich viel häufiger und schneller als in einem Meshed-Network. Dadurch werden die tolerablen Reaktionszeiten für die Routingprotokolle noch kürzer, zum Beispiel, da ein Knoten, der bei der letzten Momentaufnahme noch auf einer Route lag, mittlerweile gar nicht mehr erreichbar ist. Auch mit diesen Bedingungen sollen die Routingprotokolle möglichst gute Ergebnisse liefern.
Da die Mobilität der Knoten in MANets in so vielfältiger Weise auftreten und die Netzwerke schnell an die Grenze der Funktionsfähigkeit bringen kann, werden in dieser Arbeit nur Maschennetzwerke untersucht.

2.2 Physikalische Schicht

Die Physikalische Schicht eines WLAN wird im IEEE 802.11 Standard [IEE] definiert. Darin enthalten sind unter anderem die De-/Modulation des Signals, das Frequenzspektrum, das Zugriffsverfahren und die möglichen Datenraten. Außerdem wird ein CSMA/CD Verfahren festgelegt.
Die Eigenschaften der Übertragung haben zum Teil direkten Einfluss auf die Bit-Error-Ratio (ber), das heißt sie geben an, wie groß die Wahrscheinlichkeit eines Bitfehlers ist. Je höher die Senderate, desto wahrscheinlicher ist der Empfang eines fehlerhaften Bits. Ähnlich ist es mit der Signal-To-Noise-Ratio (snr), allerdings gilt hier, je höher der Wert, desto besser die Übertragung. Die Paket-Error-Rate (per) lässt sich aus der Länge L des Pakets in Bits und der ber berechnen:

$$per = 1 - (1 - ber)^L \tag{2.1}$$

Die Fehlerrate für ein Paket steigt also mit der Länge. Diese Eigenschaft ist für Routingmetriken wichtig, da sie in Ihren Messvorgängen die wirkliche Paketgröße berücksichtigen sollten. Nur so können die ermittelten Metrikwerte auch auf die Nutzdaten angewendet werden. Wenn die Größen der Testpakete und die der Nutzpakete zu sehr differieren, ist davon auszugehen, dass die Übertragungsdauer eine Testpaketes nicht mit der Übertragungsdauer eines Nutzpaketes korreliert. Die ermittelten Werte würden für die Nutzdaten keine optimale Routenfindung ermöglichen.

Mit der Distributed-Coordination-Function (DCF) koordinieren sich die Knoten bezüglich des Zugriffs auf das Medium, das heißt ein Knoten lauscht, ob das Medium frei ist. Ist das Medium frei, dann beginnt er seine Übertragung, wenn nicht, dann wartet er

für eine bestimmte Zeitspanne, die Backoff-Time, und versucht es erneut.

Auch dieses Zugriffsverhalten kann für die Metriken wichtig sein, da sie durch Einbeziehen dieser Information die wirklich benötigte Zeit für eine Kommunikation besser abschätzen können.

Die unterschiedlichen Senderaten, mit denen die Knoten untereinander kommunizieren, werden durch verschiedene Mechanismen bestimmt: Auto-Rate-Fallback (ARF) , Adaptive-ARF (AARF), Receiver-Based-Auto-Rate (RBAR) und Opportunistic-Auto-Rate (OAR), siehe [HVB01]. Diese Verfahren versuchen im Allgemeinen die optimale Datenrate zu bestimmen, in dem sie beobachten, ob eine Transmission mit einer bestimmten Datenrate erfolgreich war oder nicht und die Datenrate für die nächste Übertragung entsprechend anpassen.
Die Senderaten sind ebenfalls eine Eigenschaft, die von der Metrik berücksichtigt werden sollten; damit die Kommunikation auf dem schnellstmöglichen Weg stattfindet. Wenn eine Metrik die Datenrate nicht beachtet erscheinen alle Links gleich schnell und unterscheiden sich nur in der Fehlerrate. Dass das zu Fehleinschätzungen führen kann ist leicht einzusehen (siehe Expected-Transmit-Count 2.6.2).
Für jede Senderate werden die Nutzdaten (Payload) anders auf das Trägersignal aufmoduliert, grundlegende De-/Modulationstechniken werden im Folgenden kurz beschrieben. Eine bestimmte Senderate und das dazugehörige Modulationsverfahren sind äquivalent; wenn die Senderate gewechselt wird, wird das Modulationsverfahren gewechselt.

2.2.1 De-/Modulation

Die Modulation ermöglicht die Übertragung analoger und digitaler Daten über ein bestimmtes Frequenzband. Dabei wird das so genannte Trägersignal durch den Inhalt des Nutzsignals derart verändert (moduliert), dass die Nutzdaten über ein, für das Medium geeignetes, Frequenzband übertragen werden können. Bei der Modulation wird das Nutzsignal in einen anderen Frequenzbereich umgesetzt, dazu beeinflusst es Parameter wie Amplitude, Frequenz und/oder Phase des Trägersignals. Die Veränderung des Trägersignals geschieht nach einem bestimmten Mechanismus der durch das Nutzsignal entsprechend gesteuert wird. Überlagerung wäre ein einfaches Beispiel für einen solchen Mechanismus. Je nach Nutzsignal wird zwischen digitaler und analoger Modulation unterschieden. Im Weiteren wird nur die digitale Modulation betrachtet. Die Demodulation ist das Gegenstück zur Modulation und extrahiert aus den modulierten Signalen wieder die Nutzdaten.
Digitale Modulationsverfahren sind sowohl in der Zeit als auch in den zu übertragenden Daten diskret. Die einfachsten digitalen Modulationsverfahren sind Amplitudenmodulation (ASK) und Winkelmodulation (PSK), (FSK) .

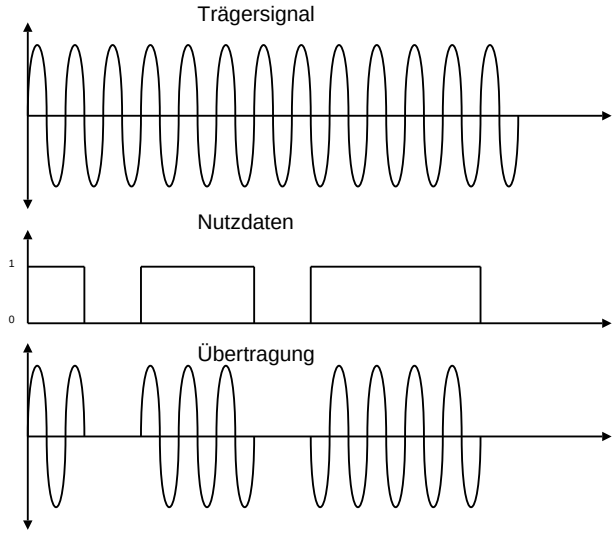


Abbildung 2.1: Amplitudenumtastung

Amplitudenumtastung Bei der Amplitudenumtastung (Amplitude-Shift-Keying, ASK) wird die Amplitude des Trägersignals verändert um die Nutzdaten zu codieren. Das einfachste Verfahren ist das so genannte On-Off-Keying (OOK), siehe Abbildung 2.1, dabei wird eine 1 durch An- und eine 0 durch Ausschalten des Trägersignals dargestellt. Durch den Einsatz von verschiedenen Amplituden können weitere Werte codiert werden, so dass pro Zeiteinheit mehr Daten übertragen werden können. Die Demodulierung des Signals wird bei einer solchen Codierung entsprechend aufwändiger und fehleranfälliger.

Frequenzumtastung Die Frequenzumtastung (Frequency-Shift-Keying, FSK) codiert die Nutzdaten in das Trägersignal, indem die Frequenz des Trägersignals verändert wird. Bei der digitalen Frequenzmodulation werden nur diskrete Frequenzwerte eingesetzt. Ein fließender Übergang ist nicht möglich und auch nicht notwendig, da digitale, also diskrete Werte übertragen werden sollen. Wenn genau zwei Frequenzen eingesetzt werden, spricht man von binärer FSK (BFSK) siehe Abbildung 2.2.

Phasenumtastung Phasenumtastung (Phase-Shift-Keying, PSK) ist ein Modulationsverfahren bei dem die Phase des Trägersignals modifiziert wird, um die Nutzdaten zu codieren. Die Phasenmodulation ist der Frequenzmodulation sehr ähnlich, beide zählen zu den Winkelmodulationsverfahren. Wenn die Phase immer um 180° verschoben wird, spricht man von binärer PSK (BPSK), siehe Abbildung 2.3.

Die hier vorgestellten Modulationsverfahren sind nur die Grundlage für die wesentlich komplexeren Verfahren, die in drahtlosen Netzwerken eingesetzt werden. Für ei-

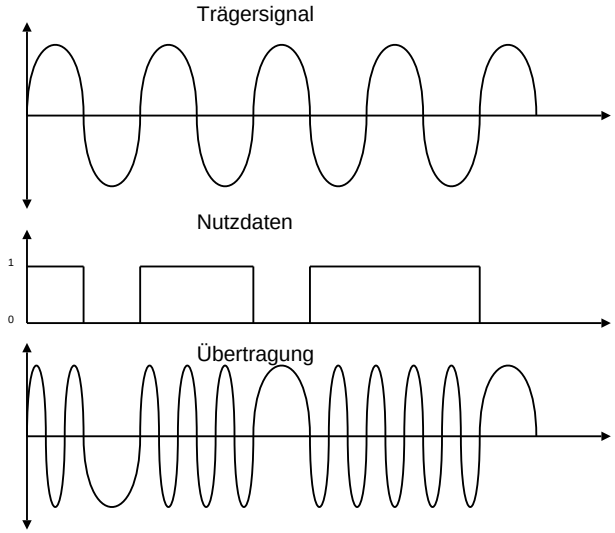


Abbildung 2.2: Frequenzumtastung

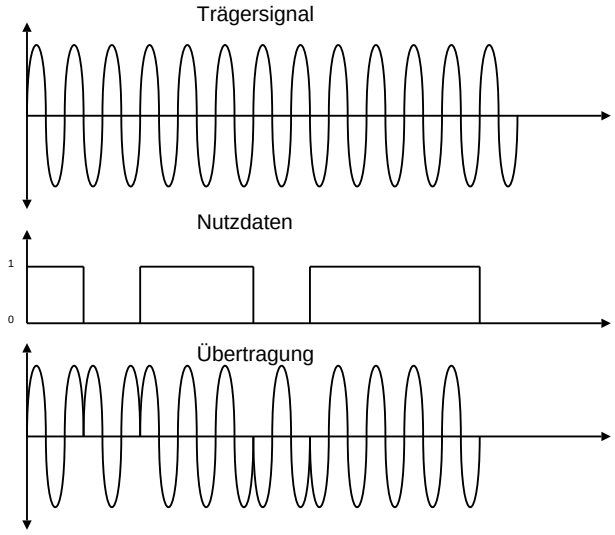


Abbildung 2.3: Phasenumtastung

Standard	Senderate	Modulationsverfahren
802.11b	1 Mbps	BPSK
802.11b	2 Mbps	QPSK
802.11b	5.5 Mbps	CCK
802.11b	11 Mbps	CCK
802.11g	6 Mbps	BPSK
802.11g	9 Mbps	BPSK
802.11g	12 Mbps	QPSK
802.11g	18 Mbps	QPSK
802.11g	24 Mbps	16-QAM
802.11g	36 Mbps	16-QAM
802.11g	48 Mbps	16-QAM
802.11g	54 Mbps	64-QAM

Tabelle 2.1: WLAN-Datenraten und das eingesetzte Modulationsverfahren

ne Datenrate von 11 MBit kann zum Beispiel die komplementäre Code-Umtastung (Complementary-Code-Keying, CCK) [RS03] eingesetzt werden. In Tabelle 2.1 ist eine Übersicht der Datenraten in WLAN-Netzen dargestellt.

2.2.2 Multi-Rate

Je nach eingesetztem Modulationsverfahren ist die Senderate unterschiedlich hoch. In Tabelle 2.1 sind die möglichen Senderaten in drahtlosen Netzwerken dargestellt. Modulationsverfahren mit einer niedrigen Senderate sind robuster gegenüber störenden Umwelteinflüssen. Es gilt, je größer die überbrückte Distanz, desto mehr Störungen. Ebenso gilt, je mehr Hindernisse in der Sichtlinie der Übertragung, desto größer ist die Wahrscheinlichkeit, dass Fehler auftreten.

Die Möglichkeiten einer hohen Senderate sollen in drahtlosen Netzen natürlich entsprechend genutzt werden, allerdings nicht komplett auf Kosten der Reichweite. Deshalb können WLAN-Karten mit verschiedenen Datenraten senden: Für große Entfernungen mit vielen Störungen wird eine niedrige Datenrate eingesetzt, für Verbindungen auf kurze Distanz eine entsprechend höhere. Niedrige Datenraten sind gegenüber den Umwelteinflüssen robuster als hohe Datenraten, so dass auch auf große Distanz eine fehlerfreie Kommunikation stattfinden kann.

Für die Auswahl der passenden Datenrate sind entsprechende Protokolle zuständig, als Beispiele seien hier Auto-Rate-Fallback (ARF), Adaptive-ARF (AARF) und Receiver-Based-Auto-Rate (RBAR) [HVB01] genannt.

In Infrastrukturnetzwerken, in denen sich die Clients mit Access-Points verbinden ist Multi-Rate, zufriedenstellend, denn die Clients finden die beste Datenrate zu ihrem Access-Point und können auch zu einem Access-Point mit einer besseren Verbindung wechseln. Das funktioniert deshalb, weil die Clients sowieso direkt zu dem Access-Point verbinden müssen und nicht die Wahl zwischen verschiedenen Routen zum AP haben. Die Probleme, die bei Multi-Rate und mehreren Routen - wie in MANets - auftreten, werden in Kapitel 2.6 Routingmetriken erläutert.

2.3 Routing

Eine Voraussetzung für die Kommunikation in Multi-Hop-Netzwerken ist - am Besten möglichst effizient - Routenfindung. In herkömmlichen (drahtgebundenen) Netzwerken gibt es zwei verschiedene Arten von Netzwerkknoten: Endknoten und Routerknoten. Endknoten sind nicht am Routingverfahren beteiligt; allerdings kann die Kommunikation zwischen zwei oder mehreren Endknoten nur mit Hilfe von Routern stattfinden.
Router tauschen untereinander Informationen über die Topologie des Netzwerks aus und ermöglichen so eine effizientere Paketweiterleitung als bloßes Fluten[1] der Pakete an alle Netzwerkknoten. Dazu verwalten sie entsprechende Tabellen, mit deren Hilfe sie die kürzeste Route zum Ziel eines Paketes bestimmen und dieses entsprechend weiterleiten können. Die Länge einer Route wird dabei anhand eines Gewichtes für jeden Link auf der Route berechnet. Die Art des Gewichtes ist von der eingesetzten Metrik abhängig.
Im Folgenden wird kurz der allgemeine Unterschied zwischen Routing in drahtgebundenen und Routing in drahtlosen Netzwerken dargestellt. Darauf folgt eine technischere Beschreibung von Routingprotokollen. Grundsätzlich lassen sich Routingprotokolle durch folgende Merkmale in Gruppen strukturieren (siehe [Fee99]):

Kommunikationsmodell Das Kommunikationsmodell, für dass das Routingprotokoll entworfen wurde, ist das grundlegendste Kriterium zum Vergleich von Routingprotokollen. Ein Protokoll, das mehrere Sendekanäle gleichzeitig effizient nutzen kann, ist anders implementiert als eines, dass sich nur auf einen einzigen Kanal beschränkt. Die hier betrachteten Protokolle gehen immer davon aus, dass die Kommunikation ausschließlich über nur einen gemeinsamen Kanal stattfindet. Allerdings kann der Kanal mit unterschiedlichen Senderaten, also mit unterschiedlichen Geschwindigkeiten genutzt werden. Die Protokolle sind sich der unterschiedlichen Senderaten im Allgemeinen nicht bewusst. Das Protokoll kann nicht erkennen, mit welcher Senderate ein Paket gesendet wird, das heißt für ein Protokoll ist im Wesentlichen die Existenz einer Verbindung von Interesse.

[1] Fluten bezeichnet den Vorgang, dass ein Paket an alle Knoten im Netzwerk gesendet wird. Es ist offensichtlich, dass Fluten mehr Bandbreite benötigt als ein gerichtetes Senden eines Pakets, bei dem es deutlich weniger wiederholt wird.

Dies ist eine Folge aus der Abstraktion der verschiedenen Schichten im Netzwerkstack. Dabei wären Informationen über die verwendeten Senderate in drahtlosen Netzwerken für die Routingprotokolle oft von Vorteil. So könnten Routingprotokolle zum Beispiel die Veränderung der Senderate beobachten und versuchen zukünftige Übertragungsqualitäten zu extrapolieren.

Struktur Die Struktur, mit der ein Routingprotokoll arbeitet, betrifft die Art der Knoten im Netzwerk. Im einfachen Fall haben alle Knoten die gleiche Funktionalität für das Routingprotokoll, diese Protokolle sind also nicht hierarchisch. Hierarchische Routingprotokolle weisen den Knoten unterschiedliche Aufgaben zu. Im Allgemeinen gruppieren sie die Knoten in so genannte Cluster, in denen ausgezeichnete Knoten dann bestimmte Aufgaben erfüllen. Die Cluster können dann weiter zu Clustern gruppiert werden. Spezielle Aufgaben übernehmen dann zum Beispiel die so genannten Clusterheads oder die Gateways. Die hierarchischen Routingprotokolle sind naturgemäß komplexer als nicht hierarchische. Genaueres findet sich zum Beispiel in [LG97].

Zustandsinformationen Routingprotokolle lassen sich anhand der Art, mit der sie Informationen über die Netzwerktopologie speichern, klassifizieren. Dabei unterscheidet man zwischen **Distance-Vector-Routing** und **Link-State-Routing**.

Distance-Vector-Routing Alle Distance-Vector-Routing-Verfahren zeichnen sich dadurch aus, dass die einzelnen Knoten keine komplette topologische Sicht auf das Netzwerk, sondern eine Tabelle mit Zielknoten, Kosten zu diesem Knoten und den nächsten Hop auf der Route zu diesem Knoten verwalten. Initial sind die Kosten zu den Knoten unendlich groß. Die Kosten zu den direkten Nachbarn werden über Hello-Beacons bestimmt, da diese sich in der 1-Hop-Nachbarschaft befinden und so die Hello-Beacons gegenseitig empfangen werden. Routen und Kosten zu anderen Knoten werden, auf diesen Nachbarschaftsinformationen aufbauend, über Austausch dieser Kostentabellen ermittelt. Eine ausführlichere Beschreibung und eine Beispielimplementierung findet sich in [LlP00].
Distance-Vector-Routing-Verfahren für drahtlose Netze sind zum Beispiel folgende: DSDV [PB94], DSR [SC05] und AODV [Per99].

Link-State-Routing Link-State-Routing-Verfahren zeichnen sich im Gegensatz zu Distance-Vector-Routing Verfahren dadurch aus, dass die Knoten untereinander Informationen solcher Art austauschen, dass sie eine topologische Gesamtsicht auf das Netzwerk - den Link-State - erhalten. Ein Knoten verwaltet Informationen über alle Knoten im Netzwerk und über die Verbindungen zwischen diesen. Zu jeder dieser Verbindungen werden außerdem Kosteninformationen gespeichert, aufgrund derer die Knoten mit entsprechenden Algorithmen die Routenfindung ausführen können (siehe [CGR94]).

LSR und OLSR [cit01] sind Link-State-Routing-Verfahren, die in drahtlosen Netzwerken eingesetzt werden.

Proaktivität In Maschennetzwerken und insbesondere in MANets ist die Änderungsrate der Konnektivität zwischen den Knoten wesentlich höher als in drahtgebundenen Netzwerken, deshalb wurden die Routingprotokolle entsprechend angepasst. Auf Veränderungen in der Netzwerktopologie muss schnell reagiert werden, da andernfalls beispielsweise neue Knoten überhaupt nicht erkannt werden oder andere Knoten, die das Netz längst verlassen haben, noch in eine Route mit einbezogen werden würden. Zum einen gibt es die Möglichkeit reaktiv (on demand) die aktuelle Netzwerktopologie in Erfahrung zu bringen, um dann eine Route zu bestimmen. Alternativ dazu kann die Information über die Topologie periodisch (proaktiv) auf dem aktuellen Stand gehalten werden, so dass bei einer Routinganfrage die Route sofort bestimmt werden kann, ohne vorher Topologieinformationen sammeln zu müssen. Man unterscheidet also zwischen reaktiven und proaktiven Routingprotokollen.
Durch die Notwendigkeit einer möglichst hohen Anpassungsrate an die Veränderungen in der Netzwerktopologie lässt sich relativ leicht ein zusätzliches Bewertungskriterium für Routingprotokolle einführen: Der Aufwand zur Erkennung von Topologieänderungen sollte im Verhältnis zu den Nutzdaten des Netzwerkes möglichst klein sein. Für Dynamic-Source-Routing beispielsweise beträgt dieser Aufwand ungefähr 1%, siehe [JM96]. Die Berechnung einer Metrik ist im Allgemeinen nicht ohne Sendevorgänge möglich, so dass auch durch diese Verwaltungsnachrichten die Bandbreite des Netzwerks beansprucht wird. Diese Belastung muss bzw. sollte offensichtlich so gering wie möglich sein.

2.4 WLAN-Routing

Drahtlose Netzwerke stellen ganz andere Anforderungen an das eingesetzte Routingverfahren als ein drahtgebundenes Netzwerk. Um in drahtgebundenen Netzwerken die beste Route zu finden, reicht es im Allgemeinen vollkommen aus, einfach die Anzahl der benötigten Hops zu vergleichen und die Route mit der geringsten Hop-Anzahl zu wählen. Dies funktioniert, da die Fehlerrate auf den einzelnen Links sehr gering ist. In WLAN-Netzen hingegen ist diese einfach Form des Vergleichs nicht unbedingt sinnvoll, denn Routenfindung über Hop-Count Vergleich kann zu nicht optimalen Routen führen. Ein leicht verständliches Beispiel ist folgendes: In einem Netzwerk mit den drei Knoten A,B,C existieren 3 Verbindungen mit folgenden Eigenschaften A $\rightarrow$ B Empfangswahrscheinlichkeit 10%, A $\rightarrow$ C 90%, C $\rightarrow$ B 90%. Routing mit Hop-Count führt so zu einer Route von A direkt nach B, sinnvoller wäre hingegen eine Route über C: $90\% * 90\% = 81\% > 10\%$. Siehe Abbildung 2.4
Die unterschiedlich gute Qualität der einzelnen Verbindungen ist aber nicht der einzige Faktor, der dazu führt, dass die Routingverfahren aufwändiger als in drahtgebundenen

Netzen sind. Folgende Liste enthält nur einige der Eigenschaften die für drahtlose Netzwerke typisch sind und in drahtgebundenen Netzen so normalerweise nicht auftreten:

- Unterschiedliche Datenraten zwischen den Knoten
- Unterschiedlich hoher Paketverlust auf den Links
- Broadcast-Medium
- Mobilität der Knoten
- Unterschiedliche Datenraten für verschiedene Pakettypen
- Unterschiedlich hohe Fehlerwahrscheinlichkeit für unterschiedliche Datenraten

Routingverfahren können diesen Eigenschaften der drahtlosen Netzwerke auf verschiedenen Ebenen begegnen (Forwarding [SPC03], Cross-Layer-Kommunikation, Metrik). Eine davon ist die Metrik, die den einzelnen Verbindungen bestimmte Kosten zuweist. Diese Kosten sollten für eine optimale Route minimal sein.

Die Dauer einer Übertragung auf einer direkten Verbindung zwischen zwei Knoten wird durch mehrere Faktoren bestimmt. Diese sind im Wesentlichen: Entfernung, Datenrate und Paketgröße. Die Entfernung spiegelt sich in der Verlustrate der Verbindung wieder: Je größer die Distanz zwischen den Knoten, desto länger dauert die Übertragung, da Pakete öfter wiederholt werden müssen. Die Entfernung der Knoten kann nur indirekt erfahren werden, da es keine Möglichkeit gibt diese mit den Netzwerkkarten auszumessen. Je höher die Datenrate, desto schneller können die Daten übertragen werden, allerdings funktioniert eine höhere Datenrate nur auf kurze Distanz mit genügend niedriger Verlustrate. Je geringer die Datenrate, desto weiter kann gesendet werden. Je größer das Paket ist, desto länger dauert die Übertragung, außerdem ist die Wahrscheinlichkeit, dass ein großes Datenpaket fehlerhaft am Ziel ankommt, größer als bei kleinen Paketen:

$$1 \geq P_p = {P_{bit}}^L \tag{2.2}$$

Die Wahrscheinlichkeit P_p für ein komplettes Paket der Länge L ergibt sich aus den Einzelwahrscheinlichkeiten für die übertragenen Bits P_{bit}.
Des Weiteren wird die Übertragungsqualität durch Hindernisse in der Sichtlinie der Knoten beeinflusst. Genauso wie die Entfernung zwischen den Knoten kann hierfür kein Wert angegeben werden, da diese beiden Parameter von den Netzwerkkarten nicht beobachtet werden können.

Wie viel Einfluss die Metrik auf ein Routingprotokoll hat, wird zum Beispiel in [AHR03] deutlich: Alle eingesetzten Protokolle profitierten von der neuen Metrik. Die richtige Metrik und ihre Bewertung der Links bilden ein virtuelles Bild des Netzwerkes, das viel enger mit der Realität verknüpft ist, als es bisher in drahtgebundenen Netzen der Fall war. Je genauer die Metrik ist, desto besser ist auch das Routing insgesamt.

2.5 Routingprotokolle

Das Senden von Paketen geschieht in paketvermittelnden Netzwerken im Allgemeinen über mehrere Zwischenknoten, so dass es notwendig ist, Routen zu berechnen (routing) und die Pakete entlang dieser Route zu leiten (forwarding). Diese Aufgabe übernehmen die Routingprotokolle. Im Folgenden werden einige Protokolle speziell für drahtlose Netzwerke vorgestellt.

2.5.1 Dynamic-Source-Routing

Dynamic-Source-Routing (DSR) [SC05] und [JM96] ist ein nicht hierarchisches Routingverfahren, bei dem die Quelle die gesamte Route für ein Paket festlegt. Die einzelnen Knoten werden dabei in dem Paket mitgesendet. DSR ist ein reaktives Protokoll, die Routen werden bestimmt, sobald sie benötigt werden. Dadurch wird keine Netzwerkbandbreite durch periodisch versendete Topologiepakete verbraucht. Nur, wenn Routen bestimmt werden müssen, werden entsprechende Pakete durch das Netzwerk gesendet.
Zur Routenbestimmung wird ein Routerequest-Paket durch das Netzwerk geflutet. Jeder weiterleitende Knoten trägt sich in das Paket ein, der Zielknoten entscheidet dann anhand der Anzahl der eingetragenen Knoten, welche Route die beste ist und sendet diese an den Quellknoten zurück.
Die Standardmetrik dieses Verfahrens ist Hop-Count siehe 2.6.1, in [PK05] wird außerdem die EDR-Metrik in DSR eingebaut.

2.5.2 Destination-Sequenced-Distance-Vector-Routing

Destination-Sequenced-Distance-Vector-Routing (DSDV) [PB94] ist ein proaktives Routingverfahren, dass auf Distance-Vector-Daten basiert. Alle Knoten senden periodisch ihre Routingtabelle an die Umgebung, dabei nimmt kein Knoten eine Sonderstellung ein, das Verfahren ist also nicht hierarchisch. Um die Aktualität der Topologieinformationen zu gewährleisten, werden die Informationen zu einem Knoten immer mit einer Sequenznummer verbunden. Diese darf - bis auf eine Ausnahme - nur von dem Knoten selber erhöht werden. Außerdem enthält ein Datentupel neben Zielknoten und Sequenznummer die Entfernung in Hops. Anhand der Sequenznummern kann leicht festgestellt werden, ob die gerade erhaltenen Daten aktueller sind als die, die in der eigenen Routingtabelle vorhanden sind.
Ein wesentlicher Aspekt von DSDV ist die Häufigkeit und die Menge der ausgetauschten Topologieinformationen. Je häufiger Daten ausgetauscht werden, desto mehr Bandbreite des Netzwerks wird beansprucht; wie schon in Kapitel 2.3 Routing geschrieben sollte dieser Aufwand möglichst gering sein.
Standardmäßig wird Hop-Count als Metrik bei DSDV eingesetzt; der Einsatz von anderen Metriken scheint aber ohne größeren Aufwand machbar - wie zum Beispiel mit der

Expected-Transmit-Time-Metric in [CABM03]

2.5.3 Ad-hoc-On-Demand-Distance-Vector-Routing

In [Per99] wird Ad-hoc-On-Demand-Distance-Vector-Routing (AODV) entwickelt. Dieses Routingverfahren ist vollständig reaktiv und versendet keinerlei Statusinformationen über die Knotenkonnektivität. Jedes Mal, wenn eine Route berechnet werden muss, wird ein entsprechender Route-Request als Broadcastpaket versendet und von den anderen Knoten so lange wiederholt bis der Zielknoten einen Route-Reply zurücksendet. Das Verfahren ist nicht hierarchisch und sendet nahezu keine Verwaltungspakete, außer bei der Bestimmung einer Route.

2.5.4 Optimized-Link-State-Routing

Optimized-Link-State-Routing (OLSR) wird in [cit01] vorgestellt und ist ein Link-State-Routingverfahren, das in zweierlei Hinsicht optimiert wurde. Aufgrund der Proaktivität des Routingverfahrens ist es wichtig, dass so wenig Bandbreite wie möglich für den Austausch der Topologieinformationen genutzt wird. Dazu verringert OLSR die Menge an Informationen, die die Knoten untereinander austauschen und reduziert die Anzahl der Weiterleitungen dieser Informationspakete. Beides geschieht über die Einführung von so genannten Multipoint-Relays (MPR), nur MPR Knoten senden Topologieinformationen weiter, alle anderen Knoten können die Pakete empfangen und verarbeiten. Die Menge der Informationen wird dadurch reduziert, dass nicht die kompletten Topologieinformationen versendet, sondern nur die so genannten Multipoint-Relay-Selectors (MPRS) in die Tabellen eingetragen werden. Die MPRS sind sozusagen das Gegenstück zu den MPR; die Wahl der MPR wird nicht zentral gesteuert, ist aber ein wesentlicher Vorgang beim OLSR. Je weniger MPR, desto weniger Pakete werden durch das Netz geflutet; die Bandbreite wird für Nutzdaten freigehalten. Allerdings müssen noch genug MPR vorhanden sein, damit das ganze Netz über die Topologie informiert wird. Die Auswahl der MPR und der passende Algorithmus dazu sind deshalb nicht trivial.
Im Standardfall wird bei OLSR Hop-Count als Metrik genutzt.

2.5.5 Ad-hoc-Wireless-Distribution-Service

Der Ad-hoc-Wireless-Distribution-Service ist ein Routingverfahren, dass unterhalb der IP Ebene operiert. Dadurch, dass AWDS auf MAC Ebene arbeitet ist es zum Beispiel möglich DHCP auch dann noch einzusetzen, wenn der DHCP-Server und der anfragende Knoten nicht direkt miteinander verbunden sind. Außerdem ist es auf diese Weise möglich, dass ein Routing stattfindet, ohne dass die Knoten eine IP-Adresse haben. dies ist normalerweise Voraussetzung für Routingprotokolle, aber in Maschennetzwerken nicht immer ohne Weiteres möglich, da DHCP nicht funktioniert. AWDS ist eine Opensource-Entwicklung, der Code und weitere Informationen finden sich hier [Her].

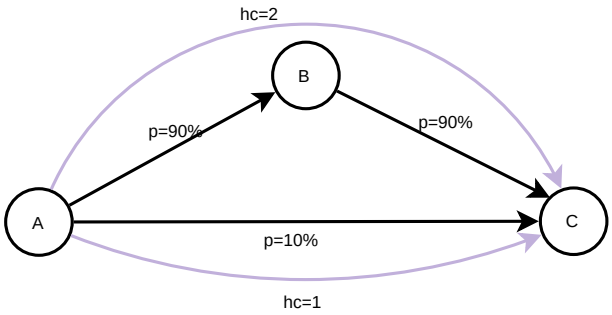


Abbildung 2.4: Hop-Count (Fall A)

2.6 Routingmetriken

2.6.1 Hop-Count

Hop-Count ist die einfachste Metrik, mit der Routen gegeneinander verglichen werden können. Dazu wird bei der Routenauswahl die Anzahl der Knoten auf der Route als Kriterium genutzt. Je weniger Hops, desto kürzer und besser die Route.
Diese Metrik funktioniert in drahtgebundenen Netzen sehr gut, in drahtlosen Netzwerken kann sie aber schnell zu einer nicht optimaler Routenauswahl führen:
In Abbildung 2.4 ist ein einfaches Beispiel dargestellt, bei dem Hop-Count die ungünstigere Route wählen würde. Die Metrik würde sich für die Route direkt von A nach C entscheiden, die Route über B wäre aber günstiger, da die Kommunikation zwischen A und C nur in 10% der gesendeten Pakete funktioniert, im Gegensatz zu 90%*90%=81% wenn die Route über B genommen wird. Gerade in infrastrukturellen Netzwerken entstehen so ungünstige Routen, da Hop-Count schon die Existenz einer Verbindung ausreicht, die Senderate oder der Paketverlust werden nicht beachtet. So werden auch Verbindungen genutzt, bei denen die Paketverlustrate sehr hoch ist und die Pakete oft wiederholt werden müssen, die Übertragung also sehr lange dauert. Außerdem unterscheidet diese Metrik nicht bezüglich der Senderichtung. Die Qualität einer Verbindung ist in beide Richtungen gleich. Vorteile dieser Metrik sind folgende:

- Die Routen sind stabil[2].
- Aufwändiges Ausmessen der Routen ist nicht notwendig.
- Hop-Count ist die „schnellste“ Metrik, da keine Messvorgänge notwendig sind.

Gerade in Multi-Rate-Netzwerken liefert Hop-Count keine optimale Routen. Einige Management-Pakete werden immer mit der niedrigsten Senderate und damit mit der höchsten

[2]Die Routen sind nur von der Topologie des Netzwerks abhängig, nicht von der Last

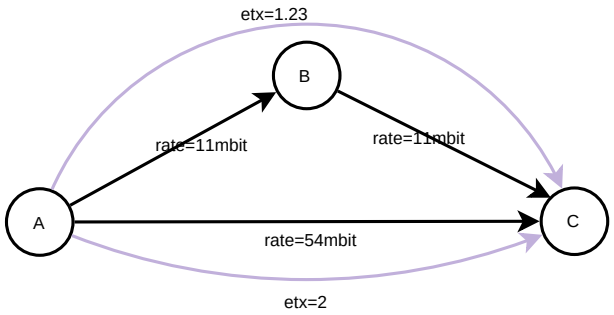


Abbildung 2.5: Expected-Transmit-Count (Fall B)

Reichweite versendet; das Routingverfahren benutzt diese Knoten dann wie einen Knoten in der Nähe mit besserer Verbindung - Hop-Count unterscheidet diesbezüglich nicht. Die schlechte Verbindung zu dem entfernten Knoten wird dadurch weiter verschärft, dass Broadcast-Pakete ohne Acknowledgement-Mechanismus versendet werden. Die Datenpakete hingegen werden mit dem vollen Acknowledgement-Zyklus[3] versendet; die Gefahr, dass eines davon nicht ankommt, steigt dann entsprechend der Anzahl der Pakete.
Die Existenz eines Knotens und seine direkte Erreichbarkeit sind durch Hop-Count nicht angemessen verknüpft.

2.6.2 Expected-Transmit-Count

Mit der Expected-Transmit-Count-Metrik (ETX) von [CABM03] können ein paar Nachteile von Hop-Count behoben werden. ETX benutzt zur Bestimmung einer Route nicht die nötigen Hops, sondern entscheidet anhand der Sendequalität. Dazu wird für einen Link ausgemessen wie oft ein Paket im Durchschnitt gesendet werden muss, damit es das Ziel erreicht. In Abbildung 2.4 ist leicht zu sehen, dass diese Metrik die Route über B wählen würde, weil die Empfangswahrscheinlichkeit auf diesem Weg offensichtlich deutlich besser ist. Außerdem ist es mit dieser Metrik möglich, die Senderichtung in die Routenfindung mit einfließen zu lassen. Ein Link zwischen zwei Knoten bekommt nicht nur eine Bewertung, sondern jeweils eine pro Richtung.
ETX beachtet jedoch nicht die Senderate, so dass auch mit dieser Metrik nicht immer die optimale Route gewählt wird. In Abbildung 2.5 ist zu erkennen, dass mit Hilfe von ETX der Weg über B gewählt werden würde, da auf diesem Weg die Übertragungswahrscheinlichkeit besser ist. Jedes Paket muss im Schnitt nur 1.23 mal versendet werden - im Gegensatz zu 2 Versuchen bei der direkten Route. Im dargestellten Fall ist es aber so, dass die direkte Verbindung die Daten trotz eines höheren ETX schneller übertragen würde, da die Senderate mehr als doppelt so hoch ist wie bei der Route über B. In einem

[3] Ein Zyklus besteht aus insgesamt 4 Paketen RTS/CTS/DATA/ACK

Netz mit unterschiedlichen Senderaten zwischen den Knoten werden so also nicht immer optimale Routen berechnet.
Diese Metrik hat also folgende Vorteile:

- Nicht symmetrische Verbindungen werden beachtet.
- Verbindungen mit geringem Paketverlust werden bevorzugt.

Probleme hingegen sind folgende:

- Die Senderate wird nicht berücksichtigt.
- Bestimmung des ETX Wertes ist nicht trivial.

2.6.3 Round-Trip-Time

Bei der Round-Trip-Time-Metrik (RTT) [ABP+04] wird die Zeit zwischen einem Channel-Select (CS) und einem CS-ACK als Maß genommen, um den Link zu bewerten. Dafür senden die Knoten regelmäßig CS-Nachrichten und wenn ein Knoten ein CS erhält, antwortet er mit einem CS-ACK. Die Zeit zwischen diesen beiden Paketen bezeichnet die RTT. Mit einem Faktor α kann dieser Wert noch gewichtet werden, so dass die Smoothed-RTT (SRTT) entsteht:

$$SRTT_{i+1} = \alpha * RTT_{new} + (1 - \alpha) * SRTT_i \tag{2.3}$$

Auf diese Weise wirken sich - möglicherweise nur kurzfristige - Änderungen der RTT nicht so stark auf die wirkliche Bewertung aus.
Das vermessen der RTT ist in der Praxis nicht so einfach zu bewältigen, wie die Theorie suggeriert. Ziel der Berechnung ist die wirkliche Sendezeit zwischen zwei Knoten. Allerdings lässt sich dieser Wert nicht so ohne Weiteres berechnen, da zum Beispiel nicht erzwungen werden kann, dass die Queue des Netzwerktreibers und die Hardwarequeue der Karte umgangen werden. Dies wäre aber für eine entsprechend genaue und zuverlässige Messung der Sendedauer notwendig. Die Queues, die normalerweise den Netzwerkbetrieb dadurch beschleunigen, dass durch sie weitere Pakete zum Versenden vorgehalten werden, behindern durch genau diese Funktionalität eine genaue Messung der Sendedauer. Da für einen Messvorgang zwei Sendevorgänge von zwei Knoten notwendig sind, verstärkt sich das Problem der Queues noch.
Zudem gibt es zwei weitere Faktoren, die die Wirksamkeit dieser Metrik wesentlich beeinflussen, dass sind das Intervall der Messungen und der verwendete α Wert für SRTT. Je kleiner das Intervall gewählt wird, desto genauer ist die Metrik und reagiert schneller auf Änderungen in der Netzwerktopologie, allerdings verbraucht sie auch mehr Netzwerkbandbreite, so dass sie ihrer eigentlichen Funktionalität, dem Finden einer guten Verbindung, entgegenwirkt. Durch α kann gesteuert werden, wie viel Einfluss der aktuell gemessene Wert auf die SRTT hat, je größer α, desto schneller stärker wird der

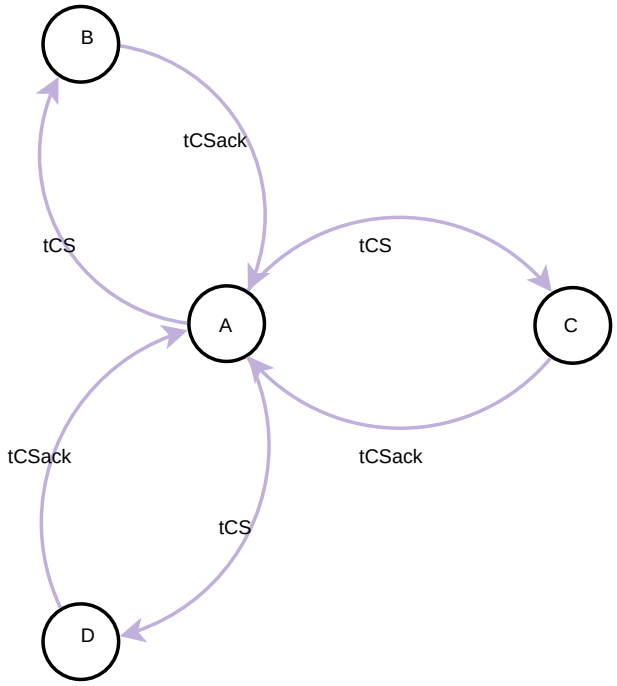


Abbildung 2.6: RTT Messung

neue Wert gewichtet. Gerade, wenn das Netzwerk, bezüglich der Konnektivität, starken Schwankungen unterliegt, kann dies zu nicht optimalen Ergebnissen führen, da eine Route, die im Mittel gut ist, genau im betrachteten Moment nicht beachtet wird. Durch zu kleine α Werte kann es dazu kommen, dass die Veränderungen von Links viel zu spät signifikant in die SRTT eingehen.

2.6.4 Packet-Pair

In [Kes91] wird die Metrik Packet-Pair (PktPair) entwickelt. PktPair funktioniert ähnlich wie RTT, die Links werden ausgemessen, indem Pakete über sie geschickt werden und die Zeit gemessen wird. Ein Problem der RTT war, dass Queueing-Effekte das Messergebnis verfälschen konnten und die Sendezeit größer eingeschätzt wurde, als sie eigentlich ist. Dieser Fehler wurde noch dadurch verstärkt, dass erst beim Initiator gemessen werden konnte, das Paket muss also durch zwei Queues, bevor überhaupt erst die Zeit gemessen werden kann.
Bei PktPair werden die Queueing-Effekte umgangen, dazu werden zwei Pakete direkt hintereinander gesendet, erst ein kleines um auf Empfängerseite den Messvorgang zu starten und direkt im Anschluss ein größeres[4]. So kann der Empfänger die reine Sendezeit des Datenpaketes berechnen und diese an den Sender zurückschicken. Allerdings kann es auch bei diesem Verfahren zu fehlerhaften Messungen kommen. Falls ein anderer Knoten Daten zwischen Start- und Datenpaket versendet, wäre die Messung verfälscht, in diesem Fall wächst der Fehler der Messung mit der Last im Netzwerk.
Des Weiteren besteht auch bei dieser Metrik die Schwierigkeit, dass ein passendes Intervall für die Vermessung gefunden werden sollte. Außerdem sollte nicht immer nur die letzte Messung den Link bewerten, sondern auch die Entwicklung der Messwerte mit einfließen. Beispielsweise über eine Historie, aus der dann der Minimalwert gewählt werden kann, oder durch Bildung eines Exponentialy-Weighted-Moving-Average (EWMA). Auch die Paketgröße kann nicht immer einfach gewählt werden, so dass es möglicherweise falsche Hinweise für die besondere Benutzung eines Links liefert. Die Paketgröße für die Messungen sollte der Größe der Pakete der Nutzdaten so nah wie möglich kommen, damit die Messung aussagekräftige Werte für die Metrik liefert. Dies wird dadurch erschwert, dass die Art der Kommunikation und die Größe der Pakete im Voraus im Allgemeinen nicht bekannt ist.

2.6.5 Path-Predicted-Transmission-Time

Die Path-Predicted-Transmission-Time-Metrik (PPTT) [YXZL06] berücksichtigt ganz speziell die gegenseitige Interferenz der Netzwerkknoten. Dabei wird zwischen so genannter Selbstinterferenz (self interference) und Nachbarinterferenz (neighbor interfe-

[4] Diese Größe kann entsprechend der beabsichtigten Kommunikation angepasst werden.

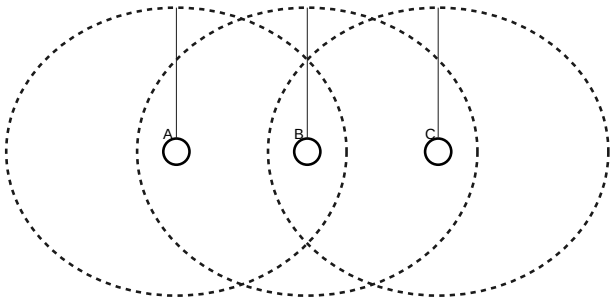


Abbildung 2.7: Selbstinterferenz

rence) unterschieden.

Selbstinterferenz bezeichnet dabei die Interferenz der Knoten auf der Route. Immer, wenn ein Knoten in einer Route sendet, kann der nächste Knoten nur empfangen und der übernächste darf ebenfalls nicht senden.

In Abbildung 2.7 ist leicht zu erkennen, dass sowohl die Kommunikation von Knoten A, als auch von C, den Empfang von B beeinflussen. Damit B Daten von A oder C erhalten kann darf der jeweils andere Knoten im entsprechenden Moment nicht senden - auch dann nicht, wenn die Daten gar nicht für B bestimmt sind.

Nachbarinterferenz ist die Interferenz, die durch Kommunikation von benachbarten Knoten erzeugt wird, die aber nicht zu der aktuellen Route gehören.

Zusätzlich zu den Interferenzinformationen arbeitet die PPTT die Senderate und den Paketverlust in die Bewertung der Links ein.
Auf das Routingverfahren wird in [YXZL06] nicht näher eingegangen, für den Austausch der Knotenkonnektivität wird das Mesh-Connectivity-Layer (MESH) [fMRR] eingesetzt. Dadurch ist es möglich, dass jeder Knoten auch über die Nachbarknoten seiner Nachbarknoten informiert ist und die Routen entsprechend bilden kann.
Die PPTT wird als Summe der einzelnen Link-Predicted-Transmission-Time (LPTT) entlang der Route berechnet.

$$[h]PPTT(\lambda) = \sum_{i=1}^{n} LPTT_i(\lambda_{cs_i} + CSF_i * \lambda, \lambda_{ht_i} + HTF_i * \lambda, \lambda) \tag{2.4}$$

mit

CSF	:	Carrier-Sensing-Factor
HTF	:	Hidden-Terminal-Factor
λ	:	traffic

Die Metrik wurde im NS2 siehe Kapitel 2.7 und in einer echten Umgebung getestet. Jeweils mit unterschiedlichen, aber festen Senderaten; die Knoten haben sich nicht bewegt. Im Vergleich mit ETX lieferte PPTT bessere Routen.

2.6.6 Expected-Data-Rate

In [PK05] wird die Expected-Data-Rate-Metrik vorgestellt. Anhand eines Interferenzmodells, und der Konkurrenzsituation auf einem Link, sowie der Quantifizierung des Medium-Access-Backoffs und durch Berücksichtigen gleichzeitiger Übertragungen auf unabhängigen Verbindungen, wird die zu erwartende Senderate abgeschätzt und so eine Bewertung der Links durchgeführt.
EDR soll als lastunabhängige Metrik funktionieren, so dass eine Änderung in der Menge der übertragenen Daten sich nicht auf die gewählte Route auswirkt. Dennoch ist es für die Berechnung von EDR notwendig, auch lastabhängige Werte zu bestimmen, das ist zum einen der ETX Wert und der Transmission-Contention-Degree (TCD). Der ETX ist derselbe wie in der Metrik ETX 2.6.2. Der TCD wird für die EDR neu eingeführt und über eine Messung der Ausgangsqueue des Knotens bestimmt. Aus der TCD wird dann der Total-Transmission-Contention-Degree $I(k)$ berechnet.

$$I(k) = \sum_{i=n_s}^{n_e} TCD(i) \tag{2.5}$$

$I(k)$	:	Total-Transmission-Contention-Degree auf Link k
$n_s, .., k, ..., n_e$	:	alle Links im Interferenzbereich von k
$TCD(i)$	:	Transmission-Contention-Degree auf Link i

Mit Hilfe dieses Wertes wird die EDR berechnet und schrittweise verfeinert.

$$EDR_{init}(k) = \frac{\Gamma}{E(k) * I(k)} \tag{2.6}$$

$EDR_{init}(k)$	:	initiale Expected-Data-Rate auf Link k
Γ	:	Senderate
$E(k)$	:	ETX wert auf Link k
$I(k)$	:	Total-Transmission-Contention-Degree auf Link k

Diese Formulierung der EDR erlaubt die Berechnung der EDR für jeden Link im Netzwerk. Jedoch ist für eine Route nur der Knoten mit der größten Fehlerrate ausschlaggebend, da dieser den Flaschenhals darstellt und so die EDR begrenzt. Auf diesem Link ist der ETX-Wert für die Links auf der Route maximal, da durch die hohe Fehlerrate die Pakete oft wiederholt werden müssen. Ebenso ist der Total-Transmission-Contention-Degree auf diesem Link maximal. Daraus ergibt sich Folgendes für die initiale EDR auf einer Route:

$$EDR_{init}(k) = \frac{\Gamma}{E_{max} * I} \tag{2.7}$$

$EDR_{init}(k)$	:	initiale Expected-Data-Rate auf Link k
Γ	:	Senderate
E_{max}	:	der ETX Wert am Link mit der größten Fehlerrate
I	:	Total-Transmission-Contention-Degree auf dem Link mit
	:	der größten Fehlerrate

Dieser Wert für die Expected-Data-Rate überschätzt die wirklich erreichte Senderate immer noch, so dass ein Reduction-Factor r eingeführt wird. Der Faktor soll den Overhead von RTS/CTS/DATA/ACK und auch des backoff berücksichtigen. r ist natürlich kleiner 1. In [JPS03] wird der Wert von r auf 0.55 beziffert.
Auch EDR_r überschätzt die reale Senderate, da die Contention-Window-Size nicht berücksichtigt wird. Dies geschieht in einem letzten Schritt, wodurch folgende Formel entsteht:

$$EDR_b = \frac{r\Gamma}{E_{max} * I_b} \tag{2.8}$$

r	:	Reduction-Factor
Γ	:	Senderate
E_{max}	:	der ETX Wert am Link mit der größten Fehlerrate
I_b	:	Total-Transmission-Contention-Degree auf dem Link mit der größten Fehlerrate inklusive Contention-Window-Size
EDR_b	:	EDR inklusive Contention-Window-Size und Reduction-Factor r

EDR wurde im Simulator NS2 siehe Kapitel 2.7 getestet und konnte die simulierten Datenraten sehr gut abschätzen. Unklar ist allerdings in welchen Intervallen der TCD gemessen wurde und wie die genauen Messparameter für ETX aussahen. Obwohl beschrieben wurde, dass es sinnvoll sei eine lastunabhängige Metrik zu implementieren, ist EDR ganz offensichtlich lastabhängig. Sowohl TCD, als auch ETX werden sich mit

zunehmendem Netzwerkverkehr erhöhen. Für den Fall eines statischen Netzes, in dem man von einem geringen Einfluss der Umgebung ausgeht, kann man die Werte in einem unbelasteten Zustand - mit entsprechendem Zeitaufwand - durchmessen und für die weitere Benutzung des Netzes und die Routenfindung heranziehen.

2.6.7 Expected-Transmit-Time

Die Expected-Transmit-Time-Metrik aus [DPZ04] weist jedem Link ein Gewicht - basierend auf zu erwartenden Sendedauer über diesen Link - zu. Dieses Gewicht kann von einer Funktion in Abhängigkeit von der Paketverlustrate und der Bandbreite der Verbindung bestimmt werden. Die ETT Metrik erweitert die ETX also um die Paketverlustrate. Dies lässt sich auch in der Berechnungsformel für die ETT erkennen:

$$ETT = ETX * \frac{S}{B} \tag{2.9}$$

ETT	:	berechnetes Gewicht
ETX	:	Expected-Transmit-Count, siehe 2.6.2
S	:	Paketgröße
B	:	Datensenderate

Dadurch, dass die ETX die Senderate bei der Linkbewertung nicht berücksichtigt, kann es, wie in Abbildung 2.5 zu erkennen ist, zu nicht optimaler Routenwahl kommen.

2.6.8 Medium-Time-Metric

In [AHR03] wird die Medium-Time-Metric (MTM) entwickelt. Diese Metrik bewertet die Verbindung zweier Knoten anhand der Zeit, die eine Übertragung über diesen Knoten dauern würde. Im Gegensatz zur ETT-Metrik wird dabei aber nicht auf ein Messverfahren gesetzt, sondern auf bestimmte Annahmen bzw. Voraussetzungen.
Eine dieser Voraussetzungen ist, dass die Metrik über die Senderate zwischen zwei Knoten informiert ist, ohne dass diese Senderate durch Versenden von Daten gemessen wird. Diese Funktion war nicht ohne weiteres real verfügbar, so dass in [AHR03] eine Umsetzung im NS2 siehe Kapitel 2.7 sowie eine theoretische Betrachtung der Metrik durchgeführt wurde.

Es wird ein theoretisches Modell für die Konnektivität und die gegenseitige Beeinflussung in einem drahtlosen Netz entwickelt, worauf basierend die MTM für eine bestimmte Senderate und Paketgröße bestimmt wird. In der nachfolgenden Tabelle stehen die MTM Gewichte für Pakete mit einer Größe von 1500 Bytes.

Link Rate	Inverse	1500 byte packet	
(Mbps)	Weights	μsec	MTM Weights
11.0	1.00	2542	1.00
5.5	2.00	3673	1.44
2.0	5.50	7634	3.00
1.0	11.00	13858	5.45

Tabelle 2.2: MTM-Senderaten und Linkgewichte

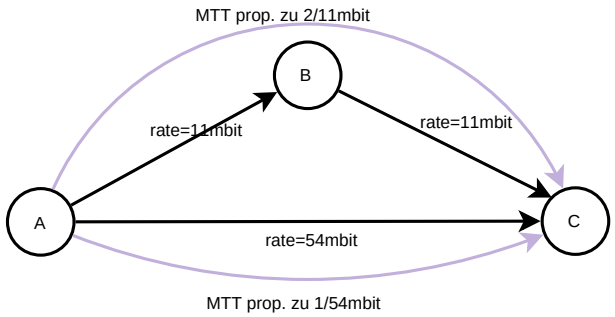


Abbildung 2.8: Medium-Transmit-Metric Fall A

Mit Hilfe dieser Gewichte kann die Route zu einem Knoten im Netzwerk bestimmt werden; die Summe der Einzelgewichte der Pfade auf der Route sollte minimal sein.

Eine wesentliche Eigenschaft/Voraussetzung der MTM ist die, dass die Datenrate und der Paketverlust nicht im Betrieb ausgemessen bzw. mitgezählt werden. Diese Metrik ist auf vorher berechnete Werte festgelegt, so dass Veränderungen in der Netzwerktopologie und der Konnektivität nicht berücksichtigt werden. Aus diesem Grund wird diese Metrik nicht implementiert werden.

2.6.9 Linkbewertung

Der Kern einer Routingmetrik ist, dass sie einer Verbindung zwischen zwei Knoten ein bestimmtes Gewicht zuordnet. Dieses Gewicht ist maßgeblich daran beteiligt, welche Route für eine Transmission gewählt wird. Es gibt unterschiedliche Verfahren diese Gewichte zu bestimmen, unterteilen lassen sich diese in zwei Gruppen. Zum einen in die, die auf direkt zugreifbaren Informationen basieren und jene, die die Verbindungen ausmessen.

Hop-Count und die Idee von MTM benötigen kein Messverfahren; wobei MTM allerdings auf spezielle Hardwareunterstützung zurückgreifen muss und Cross-Layer-Kommunikation benötigt, somit muss MTM also in höheren, eigentlich abstrahierten Schichten auf Daten aus den unteren Schichten zugreifen. Hop-Count funktioniert ganz alleine

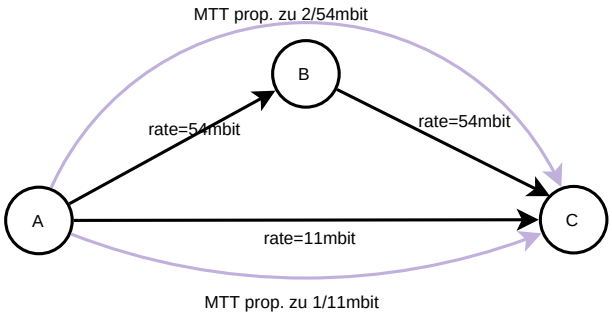


Abbildung 2.9: Medium-Transmit-Metric Fall B

mit der Topologie des Netzwerks.
ETX und ETT sind zwei Metriken die die Gewichte der Verbindungen durch Ausmessen der Knoten bestimmen. Diese Vorgänge sind nicht trivial; verschiedene Faktoren, wie die Datenmenge oder Dauer der Vermessung können das Ergebnis wesentlich beeinflussen. Außerdem ist zu beachten, dass es sich immer nur um eine Momentaufnahme handelt, die Qualität der Verbindung kann sich im nächsten Moment schon wieder deutlich verändert haben. Damit der Messvorgang nicht zu Lasten der Netzwerkbandbreite stattfindet, wird die normale Kommunikation für die Messung herangezogen. Auf diese Weise müssen nicht extra spezielle Daten versendet werden. Jedoch wird die Metrik so nur für aktive Verbindungen bestimmt. Es ist also notwendig weitere Datenpakete zu verschicken, um auch die Metrik für nicht aktive Verbindungen bestimmen zu können. Limitierend auf diesen Messvorgängen ist dann die offensichtliche Eigenschaft, dass durch die Messung Bandbreite konsumiert wird, die anderer Kommunikation vorenthalten wird. Diese Messvorgänge können weder beliebig lange, noch mit beliebig vielen Daten durchgeführt werden, ebenso sollte der Vorgang so selten wie möglich durchgeführt werden.
Wenn möglich sollte eine Technik der Bewertung eingesetzt werden, die ohne aufwändige Messverfahren auskommt und dennoch zufrieden stellende Ergebnisse liefert. Dabei helfen kann Cross-Layer-Kommunikation und spezielle Hardware, die Zugriff auf die Dauer einzelner Übertragungen ermöglicht.

2.6.10 Metrik-Alternativen zur Verbesserung der Routenfindung

In [SPC03] wird ein anderer Ansatz verfolgt, als die Metrik zu verändern, um die gewählte Route zur Kommunikation zu verbessern. Dazu wird ein weiteres Layer implementiert. Dieses arbeitet für alle anderen Schichten komplett transparent und entscheidet selbstständig, ob es bessere Routen gibt. Bevor ein Paket über eine auf herkömmlichem Weg[5] berechnete Route versendet werden kann, wird es von dem MAS-Layer untersucht.

[5]Herkömmlich bedeutet hier, wie es aktuell implementiert ist.

An dieser Stelle kann festgestellt werden, ob dem MAS-Layer eine bessere Route für das Ziel des Paketes bekannt ist und gegebenenfalls kann es umgeleitet werden. Dazu werden die Pakete in entsprechende MAS-Pakete verpackt und von den MAS-Layern wieder richtig zusammengesetzt.
Im MAS-Layer findet natürlich auch eine Bewertung der Links statt. Diese kann genau so stattfinden, wie sie die eine Metrik vornimmt, aus dieser Sicht bietet das MAS-Layer keinerlei Neuerung. Allerdings ist das MAS-Layer ein weiterer Beleg dafür, dass für eine gute Routenfindung in Maschennetzen eine Cross-Layer-Kommunikation sinnvoll ist. Routing findet normalerweise auf IP-Ebene statt, in diesem Layer sind aber keinerlei Informationen über die Beschaffenheit des Netzwerks - die für eine gute Routingmetrik notwendig sind - verfügbar.
Eine weitere Möglichkeit zur Routenbestimmung wäre, dass die Kommunikation immer über zwei Routen läuft und gemessen wird, welche Route besser ist. Dann würde die schlechtere verworfen und eine andere zweite Route aufgebaut werden. Ähnlich funktionieren Flooding-Verfahren wie in [SCK00] [CS95], bei denen alle möglichen Routen gleichzeitig vermessen werden und dann entsprechend des zu erfüllenden Kriteriums verworfen oder gewählt werden.
Beim gleichzeitigen Vermessen von Routen ist die Interferenz der Knoten untereinander zu beachten. So kann es in ungünstigen Fällen dazu kommen, dass keine Route in der Lage ist den Anforderungen zu genügen. Dadurch, dass beide Routen gleichzeitig gemessen werden, beeinflussen sie sich auch über die Interferenz und schränken die jeweilige Bandbreite weiter ein. Je nach Messverfahren und räumlichen Gegebenheiten hat die Interferenz unterschiedlichen Einfluss auf die Routensuche.
In Abbildung 2.10 ist gut zu erkennen, dass sich die Interferenzbereiche von B und D so überschneiden, dass sie sich gegenseitig beim Senden behindern. Wenn jetzt eine Route von A nach C gesucht wird und beide Routen gleichzeitig ausgemessen werden (mit welcher Datenrate gesendet werden kann), so wird das nicht die tatsächlich erreichbare sein, sondern deutlich geringer.

2.6.11 Parameter der Messvorgänge

Jede Metrik, die in regelmäßigen Intervallen Messungen durchführt, um die Links zu bewerten, hat Probleme zu lösen, die für alle Metriken gleich sind:

- Messintervall
- Messwertakkumulierung
- Paketgrößen

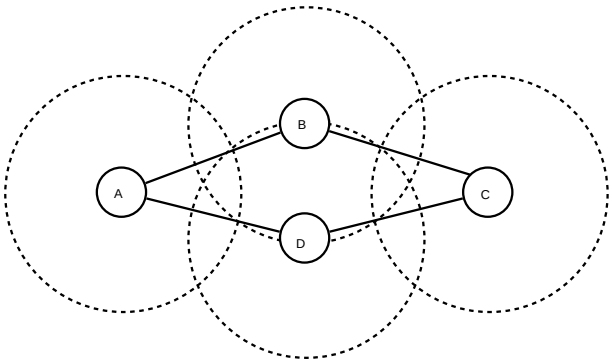


Abbildung 2.10: Interferenz bei Flooding

Das **Messintervall**, mit dem die Messungen jeweils durchgeführt werden, ist ein entscheidender Faktor für messende Metriken: Je kleiner das Intervall, desto jünger und aktueller sind die Werte und die Routenfindung ist höchstwahrscheinlich besser, als wenn auf veralteten Werten operiert werden muss. Je kleiner das Intervall, desto größer aber auch die Last des Netzes für Kontrolldaten, die verfügbare Bandbreite für Anwendungen nimmt ab, im Extremfall hilft dann auch die genaueste Routenberechnung nicht weiter. Es muss ein entsprechender Kompromiss gefunden werden. Das ist nicht einfach, da man verschiedenste Begebenheiten zu beachten hat. Veränderungen in der Netzwerktopologie sollten möglichst schnell erkannt werden, ebenso sollte bemerkt werden, dass sich die Verbindung zu einem Knoten verbessert bzw. verschlechtert.

Die **Messwertakkumulierung** bestimmt welchen Einfluss der aktuell gemessene Wert für die Bewertung des Links hat. Einfacherweise immer den letzten Wert zu verwenden ist offensichtlich nicht das Optimum. Kurzfristige Änderungen gehen sofort komplett in die Bewertung ein. Wenn das Messintervall relativ hoch ist, wird der Link für die lange Zeitspanne bis zur nächsten Messung zu gut oder zu schlecht bewertet. Auf der anderen Seite sollen Änderungen in der Linkqualität möglichst schnell widergespiegelt werden. Ein ewiges arithmetisches Mittel schließt sich damit ebenfalls aus. Bessere Alternativen sind ein Mittel über eine bestimmte Zeitspanne oder über eine bestimmte Menge an Messvorgängen. Die Bestimmung der Zeitspanne bzw. der Anzahl der eingehenden Messwerte ist aber ebenfalls nicht trivial. Weiter kann ein Exponentialy-Weighted-Moving-Average (EWMA) eingesetzt werden, bei dem alle Messwerte in die Bewertung eingehen, jedoch mit exponentiell abfallendem Gewicht. Das Gewicht für dieses Verfahren ist wieder entsprechend zu wählen.

Die **Paketgrößen** durch die ein Link bewertet wird, sind ebenso ein Faktor, der mit Rücksicht auf die Bandbreite des Netzwerkes gewählt werden muss. Kleine Pakete liefern

keine Informationen darüber, wie schnell und sicher große Datenpakete übertragen werden. Bei Paketgrößen, die mehr der Realität entsprechen, also Paketen in der Größe von durchschnittlichen Datenpaketen, wird wieder mehr Bandbreite des Netzes beansprucht.

Alle drei Faktoren sind für die jeweilige Metrik entsprechend zu wählen und vorsichtig zu variieren. Veränderungen in der einen Variable können im Allgemeinen nicht losgelöst von den Werten der anderen Variablen betrachtet werden, zum Beispiel ist die Messfrequenz eng mit der Größe der Pakete verknüpft.

2.7 Netzwerksimulator NS2

Der NS2 Netzwerksimulator ist aus dem REAL-Network-Simulator 1989 entstanden. Mit ihm können Kommunikationsnetzwerke zeitdiskret simuliert werden, um das Verhalten von Netzwerken mit verschiedenen Konfigurationen und Protokollen zu untersuchen. Programmiert wurde der NS2 mit einer Kombination aus C++ und OTcl, einer objektorientierten Version von Tcl. Die Kombination der beiden Sprachen wurde gewählt, damit mit C++ eine schnelle Abarbeitung der Daten und Ereignisse stattfindet und mit OTcl eine komfortable Möglichkeit zur Konfiguration und Steuerung des Simulator vorhanden ist. Der Simulatorkern ist in C++ geschrieben und bietet einen OTcl Interpreter als Frontend an. Dazu wird die C++ Klassenhierarchie von einer ähnlichen Klassenhierarchie in OTcl abgebildet und miteinander verknüpft. Die Methoden der C++ Klassen können vom Interpreter aus aufgerufen und genauso können die OTcl Methoden von C++ Methoden genutzt werden.
Der NS2 unterstützt die Simulation von (Routing-) Protokollen und Netzwerkanwendungen in Draht- und Funknetzwerken. Dadurch, dass der Quellcode des Simulators frei zugängig ist, ist es möglich auf jeder Netzwerkebene jede beliebige Funktionalität zu implementieren bzw. zu verändern. Der NS2 abstrahiert vollkommen von den realen Formaten der Datenpakete bei deren Übertragung. Stattdessen werden Links zwischen Netzwerkknoten eingerichtet, die die zu übertragenden Pakete erhalten. Erst auf der Zielseite wird dann entschieden, ob das Paket für den Knoten bestimmt war. Auf diese Weise können beliebige Headerstrukturen für die Pakete angelegt werden, da die Zieladresse für den Simulator zur Übermittlung der Pakete nicht notwendig ist.
Dadurch, dass der Simulator ein eigenes objektorientierte Interface anbietet, sind die dafür entwickelten Protokolle nicht ohne Weiteres in anderen Simulatoren oder Betriebssystemen lauffähig. Da die Schnittstellen nicht kompatibel sind, kann das Protokoll nicht für ein anderes System kompiliert werden, sondern muss neu implementiert werden. NS2 selber läuft unter verschiedenen Linux- und Unixsystemen, außerdem kann er in Kombination mit Cygwin auch unter Windows betrieben werden.
Die Programmierung innerhalb des NS2 funktioniert komplett ereignisgesteuert: Die ein-

zelnen Komponenten werden zum Beispiel über den Erhalt einer Nachricht durch den Aufruf einer entsprechenden Methode informiert. Der Methode wird dieses Paket übergeben, so dass direkt auf den Inhalt der Daten reagiert werden kann. Innerhalb der Methode können neue Ereignisse erzeugt werden und in die Ereignisliste des Simulators eingetragen werden. Solange der Simulator nicht alle Ereignisse aus der Liste abgearbeitet hat, werden diese weiter abgearbeitet. Auf diese Weise ist die Implementierung eines Timers bzw. der regelmäßige Aufruf einer Methode sehr einfach. Die Zeitpunkte müssen dazu in die Liste eingetragen werden, außerdem kann eine Methode sich selbst wieder eintragen, so dass sie selbst entscheiden kann, ob sie ein weiteres Mal ausgeführt werden soll. Sobald die Liste mit den Ereignissen leer ist endet die Simulation.

2.8 Propagationsmodelle

Propagationsmodelle sind ein entscheidender Teil bei der Simulation von drahtlosen Netzwerken mit dem NS2. Die Kommunikation zwischen Knoten in einem drahtlosen Netzwerk ist nicht so einfach wie in einem drahtgebundenen Netzwerk zu simulieren. Sie ist abhängig von verschiedenen Faktoren wie Entfernung und Objekten in der Sichtlinie (Bäume, Menschen, Mauern usw.). Ein gut simulierter Path-Loss ist sehr wichtig für eine gute Simulation. Wie in [KNG+04] untersucht wird, ist die Realitätsnähe der Simulation wesentlich von dem Propagationsmodell bestimmt. Natürlich gibt es verschiedene Propagationsmodelle mit unterschiedlichen Ansätzen und ebenso unterschiedlichen Ergebnissen.
In [KNG+04] wird der Einfluss von Propagationsmodellen auf die Entwicklung von Routingprotokollen untersucht: Mit dem Ergebnis, dass Routingprotokolle von ganz bestimmten Merkmalen des Propagationsmodells beeinflusst werden. An diesen Merkmalen muss sich auch das hier eingesetzte Modell messen, da insbesondere die untersuchte Routingmetriken ein realistisches Modell benötigen.

Folgende Modelle können mit dem NS2 eingesetzt werden:

1. Free-Space-Propagation-Model siehe 2.8.1
2. Two-Ray-Ground-Reflection-Model siehe 2.8.2
3. Shadowing-Model sieh 2.8.3
4. Ricean-Model siehe 2.8.4

Alle diese Modelle liefern eine Signalstärke, mit der ein Paket von einem Knoten empfangen wird. Je nach Implementierung der physikalischen Schicht wird diese anders

ausgewertet.

Im Folgenden wird die Funktionsweise dieser Modelle beschrieben, außerdem werden sie bezüglich der in [KNG+04] erarbeiteten Merkmalen untersucht.

Diese sind:

1. Die Umgebung ist flach.
2. Der Sendebereich eine WLAN-Knotens ist kreisförmig.
3. Alle Knoten haben die gleiche Sendereichweite.
4. Links sind immer bidirektional (symmetrisch).
5. Verbindungen sind perfekt.
6. Die empfangene Signalstärke hängt nur von der Entfernung ab.

Je mehr dieser Merkmale auf ein Propagtionsmodell zutreffen, desto mehr vereinfacht es die Realität und ist somit ungenauer in der Simulation.

2.8.1 Free-Space-Propagation-Model

Das Free-Space-Propagation-Model ist ein sehr einfaches Modell und geht in seiner einfachsten Form von einer Übertragung im Vakuum aus. Zur Berechnung der Signalstärke wird die Friis-Gleichung eingesetzt.

$$P_{rx} = P_{tx} \frac{G_{tx} G_{rx} \lambda^2}{(4\pi d)^2 L} \tag{2.10}$$

Friis-Gleichung [Fri45]

P_{rx} : empfangene Signalstärke
P_{tx} : Sendesignalstärke
G_{rx} : Verlust beim Empfänger ≤ 1
G_{tx} : Verlust beim Sender ≤ 1
λ : Wellenlänge
d : Entfernung zwischen den Knoten
L : Verlustrate des Systems ≥ 1

Die empfangene Signalstärke P_{rx} ist proportional zur Sendesignalstärke P_{tx}, die Verlustrate bei Sender und Empfänger G_{tx}/G_{rx} modellieren die unterschiedlichen Antennen. Je besser die Antennen das Signal abgeben, bzw. empfangen, desto größer ist P_{tx}. Die

Wellenlänge des Signals geht quadratisch in die Signalstärke ein, je länger, desto stärker. Im Gegenzug wirkt sich die Entfernung zwischen Sender und Empfänger negativ auf die Signalstärke aus, ebenfalls quadratisch, womit die kreisförmige Ausbreitung widergespiegelt wird. Die Verlustrate L ist eine Systemkonstante und modelliert den Verlust in unterschiedlichen Medien. Für Vakuum wird $L = 1$ gesetzt.

Auf der Empfangsseite wird die berechnete Signalstärke einfach gegen einen Grenzwert verglichen und dann entschieden, ob das Datenpaket empfangen wurde oder nicht.

Dieses Modell erfüllt jedes der oben genannten Merkmale.

2.8.2 Two-Ray-Ground-Reflection-Model

Die Idee des Two-Ray-Ground-Reflection-Model ist es, dass es zwischen zwei Knoten nicht nur eine direkte Verbindung wie im Free Space Propagation Model gibt, sondern auch noch reflektierte hinzukommen. Deshalb wird mit dem Two-Ray-Ground-Reflection-Model neben der direkten Verbindung auch eine am Boden reflektierte simuliert. In [Rap96] wird gezeigt, dass ein solches Modell bei großen Distanzen eine Signalstärke liefert, die der realen näher ist, als beim Free-Space-Propagation-Model.

$$P_{rx} = P_{tx} \frac{G_{tx} G_{rx} (h_t h_r)}{d^4 L} \tag{2.11}$$

Gleichung zum berechnen der Signalstärke im Two-Ray-Ground-Reflection-Model

$P_{rx}, P_{tx}, G_{tx}, G_{rx}, d, L$ Wie in der Friis Gleichung 2.10
h_t und h_r sind die Höhen der Antennen der Knoten

Dieses Modell berücksichtigt also auch noch die Höhe der Antennen, je höher, desto besser. Dadurch modelliert es die Realität ein wenig besser als das Free-Space-Propagation-Model, hat aber dennoch einige Schwächen. Bis auf Merkmal 3 und 6 erfüllt es alle schwachen Eigenschaften.

2.8.3 Shadowing-Model

Im Free-Space-Propagation-Model und im Two-Ray-Ground-Reflection-Model ist der Sendebereich eines Knotens immer ein Kreis, außerdem wird die Signalstärke durch eine deterministische Funktion des Abstands (und der Höhe der Antennen) berechnet. In der Realität ist die empfangene Signalstärke aber nicht nur vom Abstand (und der Höhe der Antennen) abhängig, sondern wird außerdem noch von vielen anderen - nicht immer konstanten - Faktoren beeinflusst. Das sind beispielsweise Objekte in der Line-of-Sight oder Überlagerungseffekte der Signale.

Das Shadowing-Model wird aus zwei Teilen berechnet, der erste ist das Path-Loss-Model, welches, wie in den beiden vorhergehenden Modellen, die durchschnittliche empfangene Signalstärke $\overline{P_r(d)}$ in Abhängigkeit von der Entfernung berechnet (siehe auch [ns2]).
Mit folgender Gleichung wird $\overline{P_r(d)}$ im Shadowing-Model berechnen:

$$\frac{P_r(d_0)}{\overline{P_r(d)}} = \left(\frac{d}{d_0}\right)^{\beta} \tag{2.12}$$

$P_r(d_0)$ bzw. d_0 sind dabei vor berechnete Konstanten, zum Beispiel mit Hilfe der Friis-Gleichung 2.10
d ist wieder der Abstand zwischen den beiden Knoten
β ist der so genannte Path-Loss exponent und wird normalerweise empirisch aus Feldversuchen bestimmt. Aus der Friis-Gleichung 2.10 kann man erkennen, dass $\beta = 2$ für das Free-Space-Propagation-Model 2.8.1 gilt.
Durch umschreiben der Gleichung nach dB erhält man Folgendes:

$$\left[\frac{\overline{P_r(d)}}{P_r(d_0)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) \tag{2.13}$$

Der zweite Teil des Shadowing-Models ist für die Variation der berechneten Signalstärke verantwortlich. Die Variation wird durch eine log-normal verteilte Zufallsvariable beschrieben, die sich gemessen in dB als Gauss-Verteilung darstellt:

$$\left[\frac{\overline{P_r(d)}}{P_r(d_0)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) + \chi_{dB} \tag{2.14}$$

Dabei ist χ_{dB} eine gaussverteilte Zufallsvariable mit Mittelwert 0 und einer Standardabweichung von σ_{dB}, genannt shadowing deviation. σ_{dB} wird ebenfalls durch Experimente bestimmt.

In dem Shadowing-Model sind die Sendebereiche zwar immer noch Kreise, aber durch die Zufallsvariable ist die Kommunikation nicht mehr deterministisch. Je nach Implementierung verändert sich die Konnektivität zweier Knoten für jedes Paket, nach jeder Bewegung der Knoten oder anderer Ereignisse, nach denen die Signalstärke für die Verbindung neu berechnet wird.

2.8.4 Ricean-Model

In [cit00] wird das Ricean-Fading-Model vorgestellt. Alle vorangegangenen Propagationsmodelle benutzen computergenerierte Zufallszahlen, um Fehler in der Übertragung zu simulieren, dadurch besteht keine zeitliche Abhängigkeit zwischen den Fehlern. Fehler mit zeitlicher Abhängigkeit, so genannte Burst-Fehler, also eine bestimmte Menge an

Fehlern direkt hintereinander, steigern die Qualität der Simulation. Das Ereignis, das einen Fehler in der Übertragung auslöst tritt, gerade in einem paketvermittelnden, im nächsten Moment gleich wieder ein. Diese Abhängigkeiten simuliert das Ricean-Model.

2.9 NS2-Erweiterung: ARF

Der Netzwerksimulator NS2 simuliert von Haus aus keine unterschiedlichen Senderaten. Am Institut National De Recherche En Informatique Et En Automatique (www.inria.fr) wurde für einen Snapshot des NS2 eine Multi-Rate Erweiterung entwickelt. Mit dieser Erweiterung wurden - neben drei neuen Modellen für die physikalische Schicht - Adaptive-Rate-Fallback (ARF) und Automatic-ARF (AARF) in den NS2 eingebaut.
Die Implementierung dieser Funktionen wurde allerdings nicht in den vorhandenen NS2-Sourcen vorgenommen; stattdessen wurden die betroffenen Schichten komplett neu implementiert. Speziell wurden sogar neue Nodes eingeführt, so dass aktuelle TCL-Scripts nicht ohne Weiteres mit dieser Erweiterung laufen.

2.10 GEA - Generic-Event-Api

Mit GEA [HM] ist es möglich, ereignisgesteuerte Netzwerkprotokolle sowohl für den Netzwerksimulator NS2 als auch für ein reales Linux Netzwerk zu entwickeln. Durch dieses Interface ist es nicht notwendig, die Sourcen für die unterschiedlichen Umgebungen anzupassen. Der Vergleich und der Test von Routingprotokollen wird auf diese Weise deutlich erleichtert. Die einzelnen Protokolle sind so genannte GEA-Module, die zur Laufzeit geladen und in den Netzwerkstack eingebaut werden.
Das AWDS-Routing 2.5.5 ist ein solches GEA-Modul, ebenso werden alle Metriken als GEA-Module implementiert werden.

3 Konzept

3.1 Problemanalyse

In den Grundlagen wurden 8 Metriken für Maschennetzwerke vorgestellt. Fast alle dieser Metriken wurden unabhängig voneinander entwickelt, implementiert und getestet. Die Tests der Metriken stellen sich naturgemäß als Vergleiche gegen andere Metriken dar; als Standardmetrik wurde in den meisten Fällen Hop-Count gewählt. Nur in wenigen Fällen wurden die Metriken auch gegeneinander verglichen.
Nicht alle Metriken wurden für echte Netzwerke implementiert, in diesen Fällen haben die Tests nur im Simulator stattgefunden. So konnte eine qualitative Einschätzung der Metrik vorgenommen werden, allerdings kann nicht ausgeschlossen werden, dass das Testszenario für die Metrik besonders gut auf die Metrik zugeschnitten war.
Die Implementierungen der einzelnen Metriken unterlagen keinem gemeinsamen System, so dass ein Austausch zum Vergleich nicht möglich ist, eine Neuimplementierung ist notwendig.
Einige der Metriken sind sich sehr ähnlich, so dass es ausreichend erscheint, nur eine Auswahl der vorgestellten Metriken zu untersuchen. Gerade in Hinblick auf die Menge der Parameter, die für die Messungen verändert werden können, ist es sinnvoll die Anzahl der untersuchten Metriken zu reduzieren. Der Suchraum kann auf diese Weise stark eingeschränkt werden, ohne dass das Ergebnis leidet. Des Weiteren sind einige Metriken keine reine Metrik, sondern implementieren eine komplette weitere Schicht im Netzwerkstack; mit deren Hilfe sie Daten - zum Beispiel über die Interferenz im Netzwerk - austauschen. Diese Metriken werden ebenfalls nicht weiter betrachtet.
Von besonderem Interesse ist die Anwendung der Metriken in Multi-Rate-Umgebungen, so dass es notwendig ist, den Simulator NS2 um eben diese Eigenschaft zu erweitern.
Für eine spezielle WLAN-Karte, die es ermöglicht, dass die Übertragungsdauer einer Transmission direkt ausgelesen werden kann, soll eine Metrik entwickelt werden. Diese kann nur in dem Simulator eingesetzt werden, wenn die Metrik Zugriff auf die Übertragungsdauer im Simulator bekommt. Dazu muss der Simulator bzw. die Implementierung der Knoten entsprechend angepasst werden.
Daraus ergeben sich folgende zu erfüllende Aufgaben und Anforderungen:

- Multi-Rate-Simulation ermöglichen.
- Anpassung des Simulators, so dass die Übertragungsdauer von einer Metrik ausgelesen werden kann.

- Neuimplementierung der Metriken.
- Simulatorfähige Implementierung.
- Echtsystemfähige Implementierung.
- Entwicklung von Testszenarien.

Durch die Neuimplementierung und dadurch, dass sie im Simulator und in einer realen Netzwerkumgebung funktionieren werden, können die unterschiedlichen Metriken in denselben Szenarios getestet werden. Nur so können die Messergebnisse entsprechende Aussagekraft erhalten, da der einzige Unterschied die eingesetzte Metrik ist.

3.2 Implementierungskonzept

Die Hauptanforderung, dass die Metriken sowohl im NS2, als auch in einem realen Netzwerk eingesetzt werden können, lässt sich elegant lösen, in dem die Metriken als GEA-Module entwickelt werden und GEA so die Einbindung in die Umgebung übernimmt. Daraus ergibt sich, dass als Routingprotokoll AWDS eingesetzt wird, da dieses schon als GEA-Modul implementiert ist und so die systemübergreifende Anforderung erfüllt.
Damit sind aber noch nicht alle Anforderungen an die Implementierung erfüllt. Das eingesetzte Routingprotokoll, sowie eine notwendige effiziente Implementierung und das GEA-System stellen weitere Bedingungen.

3.2.1 Anforderungen

Durch das eingesetzte **Routingprotokoll** AWDS-Routing werden die Datentypen der Metriken festgelegt. Für die Bewertung eines Links wird der Typ t_link_weight bereitgestellt. Des Weiteren wird der Zeitpunkt vorgegeben, an dem die Metrik über direkte Nachbarn des Knotens informiert wird. Jedesmal, wenn ein Topologiepaket versendet wird, wird die Metrik über jeden enthaltenen Knoten informiert. Über mittlerweile entfernte Knoten wird die Metrik nicht direkt unterrichtet, das Ausbleiben der Information über einen solchen Knoten muss somit als Nachricht ausreichen.

GEA ist die Verbindung zum Netzwerk bzw. zum Simulator und implementiert einen Mechanismus, durch den die Module zur Laufzeit geladen werden können. Aus diesem Grund ist es notwendig, dass die Metriken ihre Daten über die von GEA bereitgestellten Funktionen versenden. Ebenso müssen sie die Startroutine als Einsprungspunkt für GEA implementieren:

```
void gea_start(int argc, char **args)
```

3.3 Messkonzept

Jeder der implementierten Metriken soll in einer Reihe von Netzwerkkonfigurationen getestet werden. Für jeden dieser Messvorgänge wird dabei als Kriterium der Datendurchsatz zwischen zwei ausgezeichneten Knoten des Netzwerks herangezogen. Je höher der Durchsatz, desto besser wird die Metrik eingestuft. In jedem dieser Konfigurationen müssen eine Reihe von Parametern gesetzt werden, die die Konfiguration entsprechend beeinflussen. Dabei handelt es sich im Allgemeinen um folgende Parameter:

- Typ des Szenario
- Dichte[1] des Netzwerks
- Ausdehnung des Netzwerks
- Gemessene Datenmenge
- Paketgrößen
- Parameter der Metriken
- Last auf den nicht Messknoten

Des Weiteren können mehrere Datenströme gestartet werden, so dass der Vorgang der Routenfindung wiederholt getriggert wird. So kann untersucht werden, ob der selbst erzeugte Datenstrom Einfluss auf die Metriken hat und entsprechend reagiert wird. Dies ist möglicherweise von Vorteil, wenn durch die Daten eine genauere Vermessung der Links auf der Route stattgefunden hat, möglicherweise aber auch von Nachteil, wenn durch die Last eine eigentlich langsamere Route im Moment besser aussieht. So ergibt sich ein Suchraum in mindestens[2] 7 Dimensionen. Um die relevanten Bereiche dieses Raums entsprechend einzugrenzen, werden für die Parameter Werte verwendet, die zum einen erlauben, die Metrik möglichst gut "einzuschätzen" und zum anderen den realen Bedingungen so nahe wie möglich kommen. Das Einschätzen der Metriken soll in Szenarien ohne Last passieren, so dass nur die Metrik die Routenwahl bestimmt.

Im Folgenden werden zwei verschiedene Szenarien vorgestellt, die für die Untersuchung der Metriken herangezogen werden. Zu jedem werden die eingesetzten Werte für die Parameter ebenfalls dargestellt.

Neben der Implementierung der Metriken ist es außerdem noch notwendig eine Datenquelle und eine Datensenke zu erstellen, die es ermöglichen den Transfer einer Datenmenge zu vermessen. Diese werden ebenfalls als GEA-Module gestartet und lassen sich so komfortabel in ein NS2-Skript einbauen und steuern.

[1] Die Dichte ergibt sich aus der Anzahl der Knoten pro Sendereichweite.
[2] Abhängig von der eingesetzten Metrik und den damit verbundenen Parametern

3.3.1 Szenarien

In Abbildung 3.1 ist das erste Testszenario abgebildet. Gemessen werden soll die Datendurchsatzrate zwischen Start- und Zielknoten. Mit diesem Szenario soll gemessen werden, wie gut die Metrik in dem Fall funktioniert, dass Knoten auf der Route zum Ziel nicht unbedingt alle genutzt werden müssen. In diesem Szenario ist die Selbstinterferenz höchstwahrscheinlich eine bestimmende Komponente bei Routenfindung. Variiert werden hier folgende Variablen:

- Eingesetzte Metrik
- Anzahl der Knoten zwischen Start und Ziel
- Abstand der Knoten zwischeneinander (aequidistant)

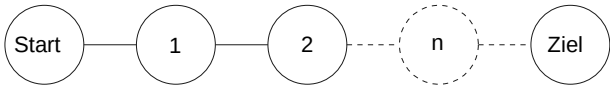


Abbildung 3.1: Szenario1

Abbildung 3.2 zeigt das zweite Testszenario. Bei dieser Netzwerkkonfiguration sind mehrere Routen zum Ziel möglich. Da die Knoten symmetrisch verteilt sind, sind die Routen höchstwahrscheinlich auch sehr ähnlich bewertet, so dass die Grundlast, die in dem Netzwerk herrscht möglicherweise ein Oszillieren der Routen erzeugt.

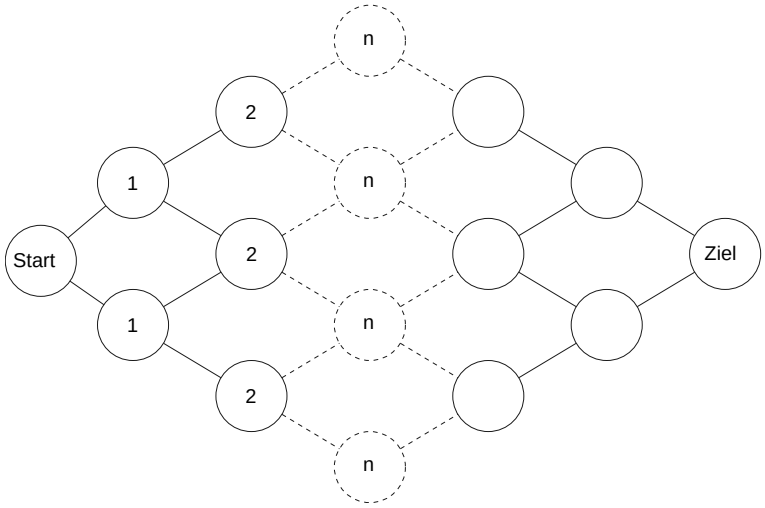


Abbildung 3.2: Szenario2

Folgende Variablen existieren in dieser Konfiguration und werden verändert:

- Eingesetzte Metrik
- Anzahl der Knotenschichten
- Dichte

3.3.2 Mögliche weitere Szenarien

Ein drittes Szenario könnte ein nicht künstlich aussehendes Netzwerk, sondern ein zufällig generiertes, sein. Nur der Ziel- und der Startknoten würden außerhalb rechts bzw. links vom Netzwerk platziert. Mit dieser Konfiguration könnte die Durchsatzrate eines Netzwerks getestet werden. In einem solchen Szenario könnten folgende Variablen variiert werden:

- Dichte
- Ausdehnung des Netzwerks
- Abstand von Start- und Zielknoten vom Netzwerk
- Last im Netzwerk
- Paketgröße der Messdaten

3.3.3 Voraussichtliche Schwächen der Metriken

Es ist davon auszugehen, dass die Metriken genau dann nicht optimale Routen liefern, wenn die Abstände zwischen den Knoten so groß sind, dass keine Übertragungsrate eindeutig bevorzugt werden kann. Bei diesen Abständen wechseln die Algorithmen für die automatische Ratenanpassung die eingesetzte Rate immer wieder, um festzustellen, welche Rate mit akzeptabler Fehlerrate funktioniert. Dadurch, dass es bei diesen Abständen genau um die Übergänge zwischen den Raten handelt, wird der Wechsel auf eine schnellere Rate fast immer mit einer fehlerhaften Übertragung enden, so dass die Übertragung wiederholt werden muss und die Dauer der Transmission ansteigt.
Höchstwahrscheinlich wird es so sein, dass die jeweils eingesetzte Metrik für die Verbindungen, die sich an einem Ratenübergang befinden dennoch relativ gute Ergebnisse berechnet, da die Metriken im Allgemeinen wiederholte Messungen ausführen und die Messergebnisse auf unterschiedliche Art akkumulieren. Dadurch werden fehlerhafte Übertragungen nicht notwendigerweise stark genug gewichtet, um diesen Link nicht in die Route einzubeziehen.

Ergebnisse an den Schwachstellen

Da die Metriken die Qualität der Links im oben geschilderten Fall besser bewerten, als sie ist, wird die Übertragungsdauer in Szenarien mit Knotenabständen, bei denen der Ratenübergang getroffen wird, deutlich ansteigen. Es gibt keinen Mechanismus, der bei einer bestimmten Übertragungsdauer eine neue Routenfindung anstößt, so dass es sogar dazu kommen kann, dass die Zeit für die Transmission sprunghaft ansteigt und für Anwendungen mit entsprechenden Anforderungen nicht mehr genügt.

Einfluss der Metriken

Eine Metrik hat normalerweise keine Möglichkeit die Rate der Übertragung zu beeinflussen, bzw. diese zu beobachten. Durch Beobachtung der Rate könnte eine Metrik feststellen, dass diese auf einem Link während einer Übertragung gewechselt wird. Wenn die Wechselfrequenz eine bestimmte Anzahl überschreitet, könnte die Metrik diesen Link schlechter bewerten. Wenn eine Metrik die eingesetzte Datenrate vorgeben könnte, könnte sie im einfachsten Fall erzwingen, dass immer eine Rate eingesetzt würde, die eine sicherere Übertragung gewährleisten würde, als die vorgegebene Rate anbietet. Dadurch würde zwar nie die maximale Übertragungsgeschwindigkeit genutzt werden können, aber im Mittel wäre die Übertragung wahrscheinlich schneller. Eine weitere Möglichkeit wäre, dass eine Metrik die Gesamtübertragungsdauer einer Übertragung beobachtet und ab einem bestimmten Schwellwert eine neue Routenfindung initiiert. Voraussetzung dafür wäre allerdings, dass eine Metrik erkennen kann, was eine Übertragung ist bzw. welche Pakete zusammengehören. Ebenso müsste die Metrik über Steuerungsmöglichkeiten bezüglich des Routingprotokolls verfügen. Beide Anforderungen widersprechen dem OSI-Schichtenmodell [Zim88], da schichtenübergreifende Kommunikation stattfinden müsste.

3.4 Evaluierungsumgebung und Simulation

Obwohl die einzelnen Simulationen zur Vermessung der Metriken wesentlich schneller abläuft, als dies in einem realen Experiment geschehen könnte, ist die Gesamtzeit aller Simulationen durch den großen Messraum so groß, dass es nicht praktikabel ist, diese auf nur einem Rechner durchzuführen. Deshalb ist es notwendig eine verteilte Simulationsumgebung zu erstellen, die folgende Eigenschaften bereitstellt:

- Einsatz möglichst vieler Rechner
- Komfortable Jobvergabe
- Flexibilität bezüglich der zu erledigenden Jobs

Jede Simulation läßt sich durch einen bestimmten Satz an Parametern bestimmen. Für die Vermessung der Metriken sind dies folgende: Szenario, Anzahl der Knoten, Abstand der Knoten und die eingesetzte Metrik. Statt diese Parameter in verschachtelten Schleifen nacheinander abarbeiten zu lassen wird jeweils ein Parameter-Tupel (S=Szenario, N=Knoten(Nodes), D=Abstand(Distance), M=Metrik) als ein zu erledigender Job interpretiert. Das Ergebnis einer Simulation ist hier die Dauer der Übertragung einer großen Datenmenge. Daraus ergibt sich folgende Abbildung f(S, N, D, M)=t. Die Logik bzw. die Komplexität des Programms (worker), das einen Job abarbeitet, soll möglichst einfach sein, um Fehler auf dieser Seite zu vermeiden. Außerdem ist die Benutzung weiterer Rechner deutlich einfacher, wenn das Programm keine besonderen Anforderungen an die Umgebung stellt. Notwendig ist dennoch die Möglichkeit, allen worker-Programmen Nachrichten zukommen zu lassen, zum Beispiel sich selbst zu beenden, um eine neue Version des workers zu starten. Ein worker hat wenigstens die untenstehenden Aufgaben:

- Selektion eines Jobs
- Markierung bzw. Locken eines Jobs
- Bearbeitung des Jobs
- Speicherung des Ergebnisses

Die Bearbeitung eine Auftrags ist dabei die unkomplizierteste Aufgabe, die ein worker zu erledigen hat. Selektion und Markierung eines Jobs hingegen müssen entsprechend synchronisiert werden, damit kein anderer worker zufällig den gleichen Job erledigt und die Simulation so unnötig länger dauert oder möglicherweise - zum Beispiel durch eine falsche Gewichtung eines Ergebnisses - sogar falsch wird. Bei der Speicherung können ebenfalls Synchronisierungseffekte auftreten: Beim Schreiben auf die gleiche Datei kann diese zum Beispiel korrumpiert werden, so dass die Ergebnisse nicht mehr gelesen werden können. Die Probleme beim Speichern der Ergebnisse lassen sich leicht dadurch lösen, dass jeder worker seine eigene Ausgabedatei erhält. Dadurch, dass alle worker in das gleiche Verzeichnis auf einem Netzlaufwerk schreiben können ist das Einsammeln der Dateien ohne weiteren Aufwand möglich.

```
cat <fileprefix>* > <outputfile>
```

erledigt diese Aufgabe. Durch das Schreiben der Ergebnisse in eine Datei kann ein weiterer Vorteil erzielt werden: Auf diese Weise können die Jobs so in die Datenbank geschrieben werden, dass der Aufruf des Jobs die Ergebnisdatei schon enthält, bzw. die Ausgabe des Ergebnisses in diese Datei. Damit ist es also nicht notwendig, dass die worker die Ergebnisse wieder in die Datenbank schreiben. Das macht die Simulationsumgebung bezüglich der zu erledigenden Aufgaben sehr flexibel. Weder der worker, noch

die Datenbank müssen für ein neues Problem angepasst werden.
Die Synchronisierung der Selektion und des Markierens eines Jobs wird dadurch gelöst, dass ein DBMS eingesetzt wird. Durch die automatische Serialisierung von Update-Statements ist es komfortabel möglich, einen Job für einen worker zu reservieren, ohne dass ein anderer worker diesen Job ebenfalls erhalten kann. Dazu wird das Update-Statement direkt mit einem Select-Statement verbunden. Nachdem der worker einen freien Job markiert hat, kann er diesen bearbeiten.
Jeder worker erzeugt eine für sich eindeutige ID; Dazu werden der Host und die Prozessid des workers genutzt. So kann die Identifizierung von abgestürzten Rechner bzw. worker erfolgen, ein eindeutiger Name für die Ausgabedatei erzeugt und ein Nachrichtensystem eingerichtet werden.
Für das Nachrichtensystem wird eine weitere Tabelle in der Datenbank angelegt, die die Nachrichten an die worker aufnimmt. Dazu enthält diese Tabelle Spalten zur Aufnahme der Nachricht und des Adressaten. 'any' steht dabei für eine Nachricht an alle laufenden worker. Für eine Nachricht an nur einen einzelnen worker muss entsprechend dessen ID als Adresse eingesetzt werden. Welche Nachrichten die worker verstehen, ist abhängig von der Implementierung des worker.

4 Implementierung

Der erste Schritt der Implementierung war den NS2, mit der GEA-Erweiterung, mit der Multi-Rate-Erweiterung von INRIA zu verknüpfen. Ein Großteil der Implementierung konnte dadurch erledigt werden, dass die zusätzlichen Dateien und Verzeichnisse der Erweiterung in den NS2-Baum eingefügt wurden und das Makefile entsprechend angepasst wurde.
Daran anschließend mussten nur noch wenige Stellen in NS2-Dateien angepasst werden, hinzu kam allerdings, dass die Erweiterung einige Fehler enthielt, die gefunden und beseitigt werden mussten.
In paket.h im NS2-Baum mussten folgende Einträge ergänzt werden:
Pakettypen:

```
1   PT_MAC80211,
2   PT_MARP,
```

und ihre Bezeichnung:

```
1   name_[PT_MAC80211] = "Mac80211";
2   name_[PT_MARP] = "Marp";
```

4.1 Überarbeitung der Multi-Rate-Erweiterung

In der Multi-Rate Erweiterung waren einige Fehler zu finden, allesamt semantischer Natur. Bezüglich der Benutzbarkeit der Erweiterung läßt sich folgende Unterscheidung treffen:

- Fehlerhafte Simulation.
- Kann undefiniertes Verhalten erzeugen.

Fehler, die die Simulation zwar ablaufen lassen, aber die Ergebnisse beeinflussen, fallen unter die Kategorie „Fehlerhafte Simulation“. In die andere Kategorie fallen diejenigen Fehler, die das Programm nicht funktionieren lassen, es also mit einer Fehlermeldung abbricht.

4.1.1 Fehlerhafte Simulation

In einer Netzwerksimulation, in der Übertragungsfehler und Senderaten simuliert werden sollen, ist die Position der Knoten des Netzwerks offensichtlich von entscheidender Bedeutung. Allerdings war gerade hier ein Fehler, die Position eines Knotens wurde über die Methode

```
1 peekPosition (NodePosition *position)
```

des Interface einer Node bestimmt. In der Implementierung wurde dann fehlerhaft die Position des Interfaces ermittelt, obwohl das Setzen

```
1 setPosition(NodePosition *position)
```

einer Position den Zustand der Node verändert. Daraus ergab sich folgende Änderung: Statt der Position des Interface:

```
1 void
2 NetInterface80211::peekPosition (NodePosition *position)
3 {
4   *position = m_position;
5 }
```

wird auf die Position der Node zugegriffen:

```
1 void
2 NetInterface80211::peekPosition (NodePosition *position)
3 {
4   m_node->peekPosition(position);
5 }
```

Durch diesen Fehler hatten alle Knoten dieselbe Position und damit einen Abstand von 0. Deshalb konnte sich die Fehlerrate nicht mit der gesetzten Entfernung verändern und somit fand auch keine Senderatenanpassung statt.

Des Weiteren war der Empfang von Broadcastpaketen nicht richtig implementiert, diese wurden vom MAC-Layer gelöscht, ohne sie an die Node weiterzuleiten:

Folgender Code wurde hinzugefügt bzw. bearbeitet um Broadcastpakete entsprechend zu behandeln.
In mac-high-ap.cc:

```
if ((getFinalDestination(packet) == ((int)MAC_BROADCAST))
  && (getSource(packet) != interface()->getMacAddress()))
{
  MacStation *station =
    interface ()->stations ()->lookup (getSource (packet));
  TRACE ("%d associated: forwarding packet", getSource (packet));
  setDestination (packet, getFinalDestination (packet));
  setSource (packet, interface ()->getMacAddress ());
  interface()->ll()->sendUp(packet);
} else
{
  Packet::free (packet);
}
```

In ll-arp.cc:

```
  if (ipDest == IP_BROADCAST)
  {
    i = m_queue.erase(i);
    m_mac->enqueueFromLL(packet,MAC_BROADCAST);
    return;
  }
...
if (ipDest == (int32_t)IP_BROADCAST)
{
  m_mac->enqueueFromLL (packet, (int32_t)IP_BROADCAST);
} else {
```

4.1.2 Fehler mit undefiniertem Verhalten

Die Implementierung der MAC-Queue hatte einen Fehler, so dass ein Leeren dieser Queue zu undefiniertem Verhalten führen konnte. In der folgenden Methode „flush“ sollten alle Pakete der Reihe nach freigegeben werden, durch den Fehler wurde zum einen nur jedes zweite Paket freigegeben und zum anderen konnte es so passieren, dass auf nicht vorhandene Pakete zugegriffen wurde. Durch einen Zugriff auf nicht vorhandene Pakete wird im Normalfall undefiniertes Verhalten erzeugt, das Programm stürzt ab.

In folgender Methode wurde die Zählvariable zweimal pro Schleifendurchlauf erhöht:

```
mac-queue-80211e.cc

void
```

```
4  MacQueue80211e::flush (void)
5  {
6    PacketQueueI tmp;
7    for (tmp = m_queue.begin (); tmp != m_queue.end (); tmp++)
8    {
9      Packet::free ((*tmp).first);
10     tmp++;
11   }
12   m_queue.erase (m_queue.begin (), m_queue.end ());
13 }
```

Das zweite `tmp++` musste entfernt werden:

```
1  void
2  MacQueue80211e::flush (void)
3  {
4    PacketQueueI tmp;
5    for (tmp = m_queue.begin (); tmp != m_queue.end (); tmp++)
6    {
7      Packet::free ((*tmp).first);
8    }
9    m_queue.erase (m_queue.begin (), m_queue.end ());
10 }
```

4.2 Verbindung NS2-Erweiterung und GEA

Die GEA-Umgebung schaltet sich auf einem realen System zwischen Netzwerkkarte und Netzwerkstack des Betriebssystem, ähnlich funktioniert GEA auch im Simulator. Mit dem Unterschied, dass nicht nur eine Netzwerkschnittstelle umgeleitet werden muss, sondern für jeden simulierten Knoten. Aus diesem Grund ist es notwendig, dass GEA immer einen Verweis auf den aktuell bearbeiteten Knoten hat, um so die Pakete richtig weiterleiten zu können. Dieser Verweis war bisher ein Objekt vom Typ Node. In der Implementierung der Multi-Rate-Erweiterung wird der Type Node nicht mehr genutzt, sondern eine neue Klasse NetNode eingeführt. Eine Veränderung des GEA-Systems ist also notwendig. Diese soll aber kompatibel mit dem normalen NS2 ohne Erweiterung sein. Aus diesem Grund wurde statt eines Verweises auf eine Node oder eine NetNode ein Verweis auf ein TclObject genommen, diese Klasse ist gleichzeitig die Superklasse von Node und NetNode. Die aktuelle Node wurde in der bisherigen GEA-Implementierung nur zur Bestimmung der ID benutzt, dies funktionierte über eine Methode von Node. Diese Funktionalität kann auch für ein TclObject erzeugt werden:

```
::TclObject *gea::ShadowEventHandler::currentNode = 0;

int gea::ShadowEventHandler::getCurrentNodeID()
{
  Tcl &tcl = Tcl::instance();
  char buf[100];
  sprintf(buf,"%s id",currentNode->name());
  tcl.evalc(buf);
  return atoi(tcl.result())+1;
}
```

und in tcl-net-node.cc:

```
if ((argc == 2) && (strcmp(argv[1],"id") == 0))
{
  Tcl&        tcl     = Tcl::instance();
  char buf[100];
  sprintf(buf,"%i",id);
  tcl.result(buf);
  return TCL_OK;
}

```

Des Weiteren verbindet sich GEA mit einer Node, indem es einen Agent mit dieser verbindet. Dies geschieht normalerweise über einen Aufruf von attachAgent. Diese Methode bietet die NetNode nicht an, so dass sie nachimplementiert werden musste, in net-node.cc:

```

void NetNode::attachAgent(Agent *agent,int port)
{
  agent->port() = port;
  agent->addr() = getUid();
  agent->target(m_interfaceConnector);
  m_demux->install(port,agent);
}
```

und in tcl-net-node.cc:

```
if (strcmp(argv[1], "attach") == 0)
{
  Agent *agent = static_cast <Agent *> (lookup(argv[2]));
```

```
4     m_node->attachAgent(agent,atoi(argv[3]));
5     return TCL_OK;
6   }
7
8   if (strcmp(argv[1], "attach") == 0)
9   {
10    Agent *agent = static_cast <Agent *> (lookup(argv[2]));
11    m_node->attachAgent(agent);
12    return TCL_OK;
13  }
```

Außerdem wurden noch zwei weitere Kommandos implementiert. Diese Kommandos bieten, wenn sie vollständig implementiert werden, extra Funktionalitäten der Knoten, wie zum Beispiel Farbe für die Visualisierung oder eine zufällige Bewegung, ebenfalls in tcl-net-node.cc.

```
1
2   if (strcmp(argv[1], "color") == 0)
3   {
4     return TCL_OK;
5   }
6
7   if (strcmp(argv[1],"random-motion") == 0)
8   {
9     return TCL_OK;
10  }
```

4.3 Implementierung der Metriken

Im AWDS-Routing wird von der Topologieklasse RTopology auf die Metriken zugegriffen. Deshalb erhält diese Klasse einen Zeiger vom Typ Metric. Alle anderen Metriken sind von Metric abgeleitet, so dass die Benutzung für RTopology transparent ist.

Das Interface einer Metric sieht folgendermaßen aus:

```
1   namespace awds
2   {
3     class Routing;
4
5     class Metric
6     {
7     protected:
```

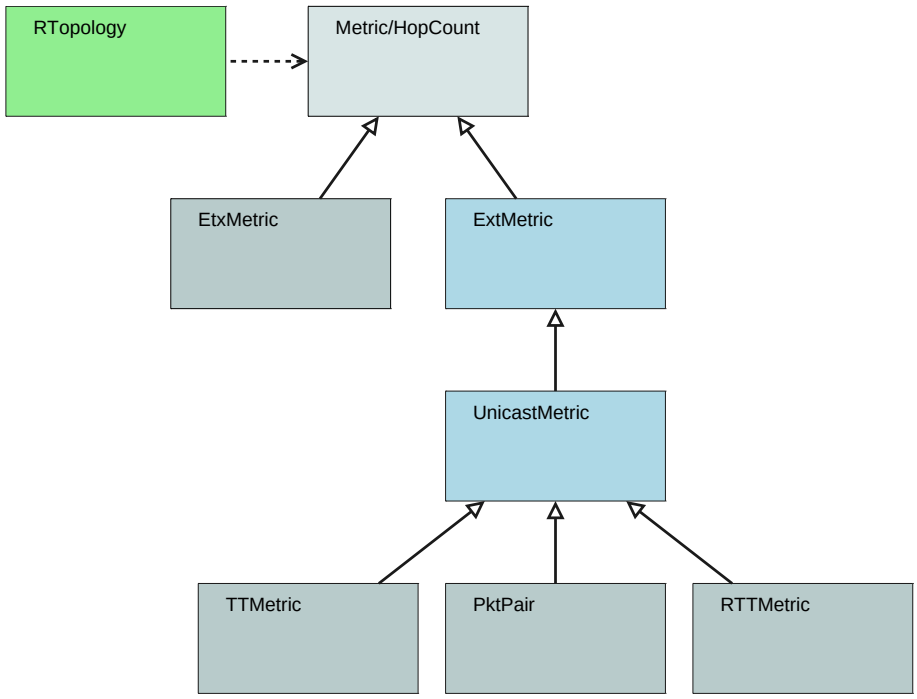


Abbildung 4.1: Klassendiagramm: Metriken

```
8       Routing *routing;
9
10      virtual RTopology::link_quality_t
11        my_get_quality(NodeDescr &ndescr);
12
13      virtual unsigned long
14        my_calculate(RTopology::link_quality_t forward,
15                                        RTopology::link_quality_t backward);
16    public:
17      Metric(Routing *r):routing(r) {}
18      virtual ~Metric() {}
19
20      virtual void addNode(NodeId &nodeId) {}
21      virtual void begin_update() {}
22      virtual void end_update() {}
23
24      virtual int update();
25
```

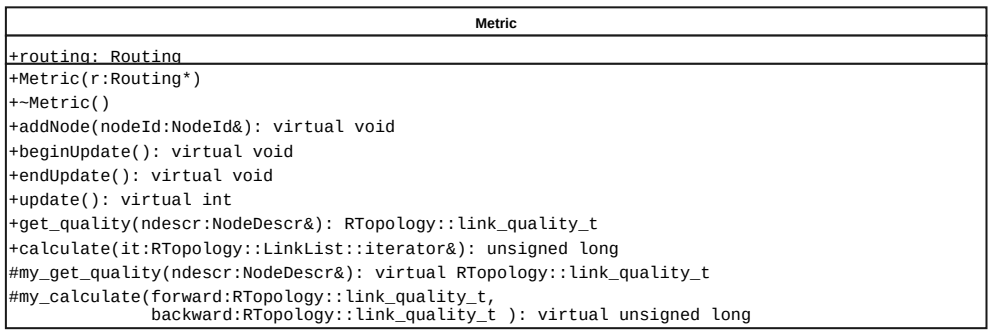


Abbildung 4.2: Superklasse: Metric

```
26      RTopology::link_quality_t get_quality(NodeDescr &ndescr);
27
28      unsigned long calculate(RTopology::LinkList::iterator &it);
29    };
30  }
```

Bei der Routenfindung entscheidende Methoden sind get_quality und calculate, diese werden an den entsprechenden Stellen von RTopology aufgerufen. get_quality liefert die Linkqualität zu dem im NodeDescr angegebenen Knoten, dabei handelt es sich um einen gerichteten Wert. Zwei solcher Werte werden von calculate verarbeitet und als Ergebnis gibt es eine ungerichtete Linkbewertung die zur Routenfindung genutzt werden kann. Die Implementierungen dieser beiden Methoden greifen jeweils auf my_get_quality bzw. my_calculate zurück, die von den Metriken implementiert werden müssen.
Jedesmal, wenn von RTopology ein Topologiepaket empfangen wird, wird das Paket geparst und die darin enthaltenen Knoten an die Metrik via addNode mitgeteilt. Bevor das Paket geparst wird, wird begin_update() aufgerufen, danach end_update(). Auf diese Weise haben die Metriken die Möglichkeit festzustellen, welche Knoten neu hinzugekommen sind und welche nicht mehr im Sende/Empfangsbereich liegen.

Diese Methoden werden von Metric virtuell deklariert, so dass ein Austausch der eingesetzten Metrik für RTolopogy transparent ist und jede Metrik ohne Änderungen an RTopology eingesetzt werden kann. Metric deklariert nicht nur das Interface der Metriken, sondern implementiert außerdem die erste und einfachste Metrik: Hop-Count. Diese Metrik ist deshalb auch die Standardmetrik und wird in RTopology direkt erzeugt, so dass eine lauffähige Topologieinstanz entsteht, ohne dass eine Metrik extra instanziiert werden muss.

In Abbildung 4.2 ist die Ableitungshierachie der Metriken dargestellt. wie zu sehen ist werden alle Klassen von Metric abgeleitet. Direkt von Metric kann EtxMetric, also Expected-Transmit-Count abgeleitet und implementiert werden. Dies ist möglich, da ETX keine Messvorgänge startet, d.h. keine Nachrichten versenden muss. Das Versen-

den von Nachrichten wird in der Klasse ExtMetric für Extended-Metric vorbereitet. Die UnicastMetric implementiert das Versenden von Paketen mit nur einem Empfänger, analog dazu könnte eine Klasse das Senden von Broadcastnachrichten implementieren, allerdings setzt keine der Metriken eine solche Implementierung voraus. Von der UnicastMetric können schließlich die Metriken abgeleitet werden, die ihre Bewertung der Links mit Hilfe von Messvorgängen, also dem Versenden von Nachrichten bestimmen. Diese sind TTMetric, PktPair und RTTMetric.

4.3.1 HopCount

Hop-Count ist die simpelste Metrik und entsprechend einfach zu implementieren, sobald dem Routing bzw. der Topologieklasse ein Knoten bekannt ist, kann Hop-Count angewendet werden. Für eine funktionierende Hop-Count-Metrik ist so auch nur die Implementierung einer Methode notwendig:

```
1 virtual unsigned long my_calculate(RTopology::link_quality_t forward,
2                                    RTopology::link_quality_t backward)
3 {
4   return 1;
5 }
```

Nur my_calculate muss implementiert werden, da alleine die Existenz eines Knotens ausreicht. Der von my_calculate zurückgegebene Wert ist ebenfalls beliebig, es muss nur immer der gleiche Wert sein, so dass das Routing zur Routenfindung eine Summe über die Gewichte der Links bilden kann.

4.3.2 EtxMetric

Die EtxMetric kann direkt von Metric abgeleitet und implementiert werden, da zur Berechnung des Expected-Transmit-Count die Beacons der Topologyklasse ausreichend sind. In RTopology werden regelmäßig Nachrichten versendet und die Sende-/ Empfangsquote für die letzten 32 Pakete mitgezählt. Diese kann über den übergebenen NodeDescr ausgelesen und für die Berechnung des ETX-Wertes genutzt werden.

```
1 RTopology::link_quality_t
2   awds::EtxMetric::my_get_quality(
3     NodeDescr &ndescr)
4 {
5   RTopology::link_quality_t q(32-ndescr.quality());
6   return q*(max_quality/32);
7 }
8
```

```
9  unsigned long
10   awds::EtxMetric::my_calculate(
11     RTopology::link_quality_t
12     forward,RTopology::link_quality_t backward)
13 {
14   forward /= (max_quality/32);
15   backward /= (max_quality/32);
16   double f(forward),b(backward);
17   double lra = f/32.0;  // loss rate
18   double lrb = b/32.0;
19   double etx = (1.0/((1.-lra)*(1.-lrb)))*scale;
20   return etx;
21 }
22
23 awds::EtxMetric::EtxMetric(Routing *r):Metric(r)
24 {
25   scale = std::numeric_limits<unsigned long>::max()/1024;
26 }
```

4.3.3 ExtMetric

Die ExtMetric-Klasse ist Superklasse für alle Metriken, die für ihre Vermessungen regelmäßige Nachrichten verschicken. Deshalb implementiert diese Klasse statische Methoden, mit denen auf den Empfang von Nachrichten reagiert werden kann. Diese rufen virtuelle Methoden auf, die von den abgeleiteten Klassen implementiert werden müssen, um die Übertragungszeiten zu bestimmen und/oder einen neuerlichen Versand zu starten. Die abgeleitete Klasse UCastMetric erweitert diese Klasse um die Möglichkeit, Unicast-Nachrichten zu verschicken.

ExtMetric.h:

```
1 static void recv_packet(BasePacket *p,gea::AbsTime t,void *data);
2 virtual void on_recv(BasePacket *p,gea::AbsTime t) = 0;
3
4 static void wait(gea::Handle *h,gea::AbsTime t,void *data);
5 virtual void on_wait(gea::Handle *h,gea::AbsTime t) = 0;
```

ExtMetric.cc:

```
1 void awds::ExtMetric::recv_packet(BasePacket *p,gea::AbsTime t,void *data)
2 {
3   ExtMetric *instance(static_cast<ExtMetric*>(data));
```

```
4   instance->on_recv(p,t);
5 }
6
7 void awds::ExtMetric::wait(gea::Handle *h,gea::AbsTime t,void *data)
8 {
9   ExtMetric *instance(static_cast<ExtMetric*>(data));
10  instance->on_wait(h,t);
11 }
```

4.3.4 UCastMetric

Mit dieser Klasse ist es möglich Unicastnachrichten zu verschicken, dazu registriert sie im AWDS-Routing einen neuen Packettyp: PACKET_TYPE_UC_METRIC. Des Weiteren implementiert sie zwei Methoden, mit denen es möglich ist die Nachrichten an einen Knoten zu verschicken.
Durch die sendvia-Methode ist es möglich, den Versand eines Pakets an einen Knoten zu erzwingen, so dass das Routingprotokoll an dieser Stelle nicht entscheiden kann, dass eine andere Route besser geeignet wäre.

UCastMetric.cc

```
1  awds::UCastMetric::UCastMetric(awds::Routing *r):ExtMetric(r)
2  {
3    routing->registerUnicastProtocol(PACKET_TYPE_UC_METRIC,
4      recv_packet,(void*)this);
5  }
6
7  void awds::UCastMetric::send(BasePacket *p,gea::AbsTime t,NodeId dest)
8  {
9    UnicastPacket uniP(*p);
10   uniP.setUcDest(dest);
11   p->setDest(dest);
12   routing->sendUnicast(p,t);
13 }
14
15 void awds::UCastMetric::sendvia(BasePacket *p,gea::AbsTime t,
16                                 NodeId dest,unsigned int size)
17 {
18   UnicastPacket uniP(*p);
19   uniP.setUcDest(dest);
20   p->setDest(dest);
21   uniP.packet.size = std::max(uniP.packet.size,size);
```

```
22   routing->sendUnicastVia(p,t,dest);
23 }
```

Zusätzlich zu den Methoden zum verschicken eines Unicastpaketes wird hier auch das entsprechende Paket definiert. Mit diesem Pakettyp sollen Übertragungszeiten gemessen werden, so dass entsprechende Felder definiert werden. Weiter werden noch Felder angelegt, mit denen eine Sequenznummer und die Herkunft des Paketes verwaltet werden kann. Für jedes dieser Felder werden die dazugehörigen get und set Methoden implementiert.

UCMetricPaket.h:

```
1 static const size_t OffsetSeq = UnicastPacket::UnicastPacketEnd;
2 static const size_t OffsetType = OffsetSeq+sizeof(unsigned int);
3 static const size_t OffsetOriginator = OffsetType+sizeof(Type);
4 static const size_t OffsetTime1 = OffsetOriginator+sizeof(NodeId);
5 static const size_t OffsetTime2 = OffsetTime1+sizeof(gea::AbsTime);
6 static const size_t OffsetDuration = OffsetTime2+sizeof(gea::AbsTime);
```

4.3.5 TTMetric

Die Transmit-Time-Metrik ist eine der Metriken, die regelmäßig Pakete an die Nachbarn des Knoten schickt. Obwohl die TTMetric die Übertragungsdauer direkt vom Knoten bzw. Treiber auslesen kann, ist es notwendig Nachrichten zu verschicken. Nur so können Gewichte für alle Nachbarknoten gefunden werden und nicht nur für diejenigen, die gerade an einer Transmission teilnehmen und so nebenbei vermessen würden.
Sobald die Metrik aktiviert wird startet diese einen Timer, um den regelmäßigen Testnachrichtenverkehr zu starten. Dabei ist die ExtMetric dafür verantwortlich dieses Ereignis zu empfangen und entsprechend an die richtige Methode weiterzuleiten. In der on_wait Methode der TTMetric wird eine Nachricht an einen Nachbarkonten gesendet, dabei wird der Knoten ausgewählt, der am längsten keine Testnachricht erhalten hat. Die Methode my_get_quality kann dann direkt auf den Knoten zugreifen und die dort gespeicherte Übertragungsdauer zurückliefern.

Alle von UCastMetric abgeleiteten Metriken haben wenigstens zwei Parameter, die ihnen beim Start übergeben werden können. Mit dem Parameter „Intervall“ wird die Zeit zwischen zwei Messvorgängen festgelegt; als Standard ist eine Sekunde vorgegeben. Des Weiteren kann die Paketgröße mit „Paketsize“ festgelegt werden; initial wird eine Größe von 800 Byte genutzt.

```
1 void
2 awds::TTMetric::start()
```

```
3  {
4    GEA.waitFor(&blocker,
5          gea::AbsTime::now()+interval,
6          &ExtMetric::wait,
7          (void*)this);
8  }
9
10 void
11 awds::TTMetric::on_wait(gea::Handle *h,gea::AbsTime t)
12 {
13   TTData::iterator oldest (ttData.begin());
14   TTData::iterator it(ttData.begin());
15   while (it != ttData.end())
16   {
17     if (oldest->second.lastsend > it->second.lastsend)
18     {
19       oldest = it;
20     }
21     ++it;
22   }
23   it = oldest;
24   if (it != ttData.end())
25   {
26     gea::AbsTime t(gea::AbsTime::now());
27     BasePacket *p = routing->newUnicastPacket(PACKET_TYPE_UC_METRIC);
28     UCMetricPacket mp(*p,awds::UCMetricPacket::req,2,1,t);
29     sendvia(p,t,it->first,packetSize);
30     it->second.lastsend = t;
31   }
32   GEA.waitFor(&blocker,
33         gea::AbsTime::now()+interval,
34         &ExtMetric::wait,
35         (void*)this);
36 }
37
38 awds::RTopology::link_quality_t
39 awds::TTMetric::my_get_quality(NodeDescr &ndescr)
40 {
41   RTopology::link_quality_t ret =
42     std::min(
43       (RTopology::link_quality_t)(g2m->getTT(ndescr.id)*max_quality),
```

```
44        (RTopology::link_quality_t)max_quality);
45    ttData[ndescr.id].tt = ret;
46    return ret;
47  }
48
```

4.3.6 PktPair

PktPair ist eine Metrik, die die Linkqualität wirklich ausmisst und die Übertragungsdauer durch zwei direkt hintereinander abgeschickte Pakete bestimmt. Wie die TTMetric findet dieser Vorgang regelmäßig statt, weshalb ebenfalls der Timermechanismus zum Einsatz kommt. Zusätzlich zur Behandlung des Timers muss eine Methode implementiert werden, die auf den Empfang des Paketpaares reagiert und die bestimmte Übertragungsdauer an den Sender zurückschickt bzw. erkennt, dass die Übertragungsdauer gesendet wurde:

```
1   void PktPair::on_recv(BasePacket *p,gea::AbsTime t)
2   {
3     UCMetricPacket mp(*p);
4     NodeId sender(mp.getSrc());
5     gea::Duration help_t(t-gea::AbsTime(0));
6     if (mp.getType() == awds::UCMetricPacket::first)
7     {
8       firstPackets[sender] = t;
9     } else
10    {
11      if (mp.getType() == awds::UCMetricPacket::second)
12      {
13        FirstPackets::iterator it(firstPackets.find(sender));
14        if (it != firstPackets.end())
15        {
16          gea::Duration duration(t-it->second);
17          BasePacket *p_response(
18            routing->newUnicastPacket(PACKET_TYPE_UC_METRIC));
19          UCMetricPacket response(
20            *p_response,awds::UCMetricPacket::resp,2,1,duration);
21          sendvia(p_response,gea::AbsTime::now(),sender);
22        }
23      } else
24      {
25        if (mp.getType() == awds::UCMetricPacket::resp)
```

```
26        {
27          Nodes::iterator it(nodes.find(sender));
28          if (it != nodes.end())
29          {
30            gea::Duration d(mp.getDuration());
31            it->second.setTime(d,alpha);
32          }
33        }
34      }
35    }
36  }
```

PktPair hat außerdem, zu den bisher bekannten, noch zwei weitere Parameter. Zum einen kann der Alpha-Wert angegeben werden, mit dem der neu gemessene Wert gewichtet wird (siehe 2.6.4), als Standard ist 1 vorgegeben. Zum anderen kann die PktPair Metrik auf eine Minimum Variante umgestellt werden, so dass sich der Wert für einen Link als Minimum aus einer bestimmten Anzahl gemessener Werte bestimmt. Die Größe der History kann ebenfalls angegeben werden.

4.3.7 RTTMetric

Das Messverfahren der RTTMetric ist ähnlich dem von PktPair. Hier findet die Auswertung nicht am Nachbarknoten statt, sondern am Initatorknoten, so dass die Empfangsroutine zwei Aufgaben hat. Zum einen muss ein CS-Paket sofort mit einem CS-Ack beantwortet werden und zum anderen aus der Dauer zwischen dem Versand und dem Empfang eines CS-Ack der Wert für die Metrik bestimmt werden. Deshalb ist es notwendig, dass der Zeitpunkt, zu dem ein CS-Paket versendet wird, gespeichert werden muss um die Differenz beim Empfang eines CS-Ack zu bestimmen.

```
1  void
2  awds::RTTMetric::on_recv(BasePacket *p,
3                           gea::AbsTime t) {
4    UCMetricPacket mp(*p);
5    if (mp.getType() == awds::UCMetricPacket::req)
6    {
7      BasePacket *p_response(
8        routing->newUnicastPacket(PACKET_TYPE_UC_METRIC));
9      UCMetricPacket response(
10       *p_response,awds::UCMetricPacket::resp,2,1,mp.getTime1());
11     sendvia(p_response,gea::AbsTime::now(),mp.getSrc());
12   } else
```

```
13    {
14      gea::AbsTime t1(mp.getTime1());
15      gea::Duration d(t-t1);
16      RTTData::iterator it(rttData.find(mp.getSrc()));
17      if (it != rttData.end())
18      {
19        it->second.time = alpha*d+(1-alpha)*(it->second.time);
20        it->second.lastrecv = t;
21        if (history)
22        {
23          (*history)[mp.getSrc()].push_back(d);
24          (*history)[mp.getSrc()].push_back(it->second.time);
25        }
26      }
27    }
28  }
29
30  void
31  awds::RTTMetric::on_wait(gea::Handle *h,
32                           gea::AbsTime t)
33  {
34    RTTData::iterator oldest (rttData.begin());
35    RTTData::iterator it(rttData.begin());
36    while (it != rttData.end())
37    {
38      if (oldest->second.lastsend > it->second.lastsend) {
39        oldest = it;
40      }
41      ++it;
42    }
43    it = oldest;
44    if (it != rttData.end())
45     {
46      gea::AbsTime t(gea::AbsTime::now());
47      BasePacket *p = routing->newUnicastPacket(PACKET_TYPE_UC_METRIC);
48      UCMetricPacket mp(*p,awds::UCMetricPacket::req,2,1,t);
49      sendvia(p,t,it->first,packetSize);
50      it->second.lastsend = t;
51    }
52    GEA.waitFor(&blocker,
53          gea::AbsTime::now()+interval+(rand()%10/10.0),
```

```
54          &ExtMetric::wait,
55          (void*)this);
56  }
```

Die RTTMetric hat ebenfalls den Parameter Alpha um diesen Wert vorzugeben.

4.4 Sender/Empfänger

Für die Vermessung der Metriken ist es weiterhin notwendig, dass eine bestimmte Menge an Daten zwischen einem Sender- und einem Empfängerknoten versendet werden kann. Diese Aufgabe übernimmt ein weiteres Modul, dass neben den offensichtlichen und parametrierbaren Eigenschaften „Paketanzahl“ und „Paketgröße“ vor allem die Eigenschaft implementiert, dass es auf eine gefunden Route zwischen Sender und Empfänger warten kann. Das Modul kann sowohl als Quelle, als auch als Ziel eingesetzt werden und der Datentransfer findet in einem Handshake-Verfahren statt. Jedes eingegangene Paket wird von der Senke mit einem minimalen Paket quittiert. Sobald das letzte gesendete Paket beantwortet wurde, gibt die Quelle die Dauer der Gesamtübertragung aus.

```
1   void awds::Traffic::send(int pCount,int pSize,NodeId d)
2   {
3     if (!packetCount)
4     {
5       packetCount = pCount;
6       packetSize = pSize;
7       dest = d;
8       lastcount = count = 0;
9       start = gea::AbsTime::now();
10    }
11    if (count > packetCount) {
12      return;
13    }
14    if (!routing->isReachable(dest))
15    {
16      GEA.waitFor(&blocker,
17      gea::AbsTime::now()+10,
18      &Traffic::wait,
19      (void*)this);
20      return;
21    }
22    if (count == 0)
23    {
```

```
24      start = gea::AbsTime::now();
25    }
26    BasePacket *p = routing->newUnicastPacket(PACKET_TYPE_TRAFFIC);
27    TrafficPacket tp(*p);
28    tp.setType(TrafficPacket::fromsrc);
29    tp.setSeq(count);
30    count++;
31    lastcount = count;
32
33    UnicastPacket uniP(*p);
34    uniP.setUcDest(dest);
35    uniP.packet.size = packetSize;
36    p->setDest(dest);
37    routing->sendUnicast(p,gea::AbsTime::now());
38    GEA.waitFor(&blocker,
39          gea::AbsTime::now()+10,
40          &Traffic::wait,
41          (void*)this);
42  }
43
44  void awds::Traffic::send_reply(NodeId dest)
45  {
46    BasePacket *rp = routing->newUnicastPacket(PACKET_TYPE_TRAFFIC);
47    TrafficPacket tp(*rp);
48    tp.setType(TrafficPacket::fromsink);
49    tp.setSeq(count);
50    UnicastPacket uniP(*rp);
51    uniP.setUcDest(dest);
52    rp->setDest(dest);
53    routing->sendUnicast(rp,gea::AbsTime::now());
54  }
55
```

4.5 Evaluierungsumgebung und Simulation

Zur Aufnahme der Jobs, die von den worker-Prozessen abgearbeitet werden sollen, wurde ein PostgreSQL-DBMS auf einem Rechner (crunchy) installiert, der von allen anderen Rechnern erreicht werden kann. In dem DBMS wurde eine „job_center“ genannte DB angelegt, diese enthält zwei Tabellen: jobs und worker_messages.

```
create table jobs (
  lfnr int,
  job varchar,
  type varchar,
  done bool,
  owner varchar(30),
  owned_at timestamp,
  done_at timestamp,
  primary key (lfnr)
);

create table worker_messages (
  lfnr int,
  message varchar,
  to_worker varchar,
  primary key (lfnr)
);
```

Mit folgendem Update-Statement kann ein worker einen Job selektieren und markieren:

```
update jobs set owner = <ownerid>,owned_at = now()
where lfnr = (select min(lfnr) from jobs where owner = '' and not done);
```

Durch den Select auf das Minimum der lfnr der freien Jobs wird genau ein Job markiert. Dieser kann im folgenden über ein Select aus der Tabelle entnommen und bearbeitet werden:

```
select job from jobs where ownder=<ownerid> and not done;
```

Nach der Abarbeitung muss das done flag für diese Job gesetzt werden, damit ein neuer Job ausgewählt werden kann.

In Kombination mit Shell-Script und entsprechen psql Aufrufen ergibt sich folgendes Script für die worker:

```
#! /bin/sh

if [ -f /usr/bin/psql ]
then
    #obtain unique id
    MYID=$$$HOSTNAME
    #say hello
```

```
8       echo $MYID
9       #export id so it can be accessed in the following statements
10      export MYID=$MYID
11
12      while (true)
13      do
14    #lookup messages
15    #1st messages to anybody, so to me
16    message_any=‘psql -h crunchy.euklab.ls.la job_center wienoebs -c
17      "select ’message ’
18      || message from worker_messages where to_worker = ’any’"
19      | grep message | tail -1 | (read a b ; echo $b)‘
20    #interpret message
21    if [ $message_any = "stop" ]
22    then
23        #say goodbye
24        echo "exiting due do stop message"
25        exit
26    fi
27    #2nd messages only for me
28    message_me=‘psql -h crunchy.euklab.ls.la job_center wienoebs -c
29      "select ’message ’
30      || message from worker_messages where to_worker = ’$MYID’"
31      | grep message | tail -1 | (read a b ; echo $b)‘
32    if [ $message_me = "stop" ]
33    then
34        #say goodbye
35        echo "exiting due to personal stop message"
36        exit
37    fi
38    #1st update (mark it as in use) my next job
39    psql -h crunchy.euklab.ls.la job_center -c "update jobs
40      set owner =’$MYID’, owned_at = now()
41      where lfnr = (select lfnr from jobs
42      where owner = ’’ and done = ’f’ limit 1)"
43    #2nd select the job and execute it
44    psql -h crunchy.euklab.ls.la job_center -c "select job
45      from jobs where owner = ’$MYID’
46      and done = ’f’" | grep Distance | sh
47    #3rd update job es done
48    psql -h crunchy.euklab.ls.la job_center -c "update jobs
```

```
49      set done = 't',  done_at = now()
50      where owner = '$MYID' and done = 'f'"
51      done
52  fi
```

Dieses worker-Script versteht zwei Nachrichten, beide beenden die Ausführung des Scripts, der Unterschied besteht einzig darin, dass diese Nachricht einmal an alle worker ist und einmal nur an einen bestimmten gerichtet ist.

5 Evaluierung

5.1 Überprüfung der Voraussetzungen

Für einen aussagekräftigen Vergleich der Metriken ist es notwendig, dass der Simulator die neu implementierte Multi-Rate-Eigenschaft in Zusammenarbeit mit GEA korrekt simuliert. Zur Überprüfung dieser Voraussetzung wurde die Übertragung zwischen zwei Knoten bei unterschiedlichen Entfernungen getestet. Ermittelt wurden in diesem Test die eingesetzte Senderate und die benötigte Dauer zur Übertragung von 80 MB Daten. Des Weiteren wurden in entsprechenden Einzeltests die Metriken bei der direkten Kommunikation zwischen zwei Knoten mit unterschiedlichen Entfernungen überprüft.

In Abbildung 5.1 ist das Ergebnis des Multiratetests dargestellt, die Distanz zwischen den beiden Knoten wurde in Schritten von 1 Meter, beginnend von 1 Meter, bis zu einem Abstand von 800 Meter festgelegt. Übertragen wurde eine Datenmenge von 80 MB. In jeweils 100 Messungen pro Meter wurden durchschnittliche Datenrate und Übertragungsdauer ermittelt. Die jeweils 100 Messdaten wurden nicht weiter gemittelt. Es ist deutlich zu erkennen, dass die Übertragungsdauer mit zunehmender Entfernung der Knoten voneinander stufenweise zunimmt. Vor allem die stufenweise Zunahme bestätigt, dass der Simulator die Übertragung mit verschiedenen Senderaten berechnet. Ab einer Entfernung von 700 Metern steigt die Fehlerrate der übertragenen Daten deutlich an und die Übertragungsdauer nimmt entsprechend zu. Da bei dieser Entfernung schon mit der niedrigsten Rate gesendet wird, kann der Multi-Rate Mechanismus diese Fehler nicht mehr ausgleichen und die Dauer der Übertragung nimmt stetig und immer steiler zu. In den Übertragungszeiten bei geringeren Distanzen als 800 Metern ist zu erkennen, dass mit besserer Rate die Distanz, für die sie erfolgreich genutzt werden kann, kürzer wird. Bis ca. 140 m werden die beiden besten Senderaten genutzt und es wird die geringste Übertragungsdauer erzielt. Demgegenüber ist die folgende Stufe alleine schon ca. 160 m lang. Ab einer Entfernung von ca. 800 Metern ist die Kommunikation zwischen den Knoten praktisch nicht mehr akzeptabel.

Die durchgezogene Kurve in Abbildung 5.1 zeigt die Veränderung der Senderate in Abhängigkeit von der Entfernung. Diese zeigt die stufenweise Abnahme nicht so deutlich wie die Übertragungsdauer. Da der Multi-Rate-Mechanismus immer wieder versucht die Senderate zu erhöhen, ist diese Kurve deutlich glatter und zeigt auch längere Übergänge zur nächsten Senderate, als die Übertragungsdauer von einer Stufe zur nächsten. In

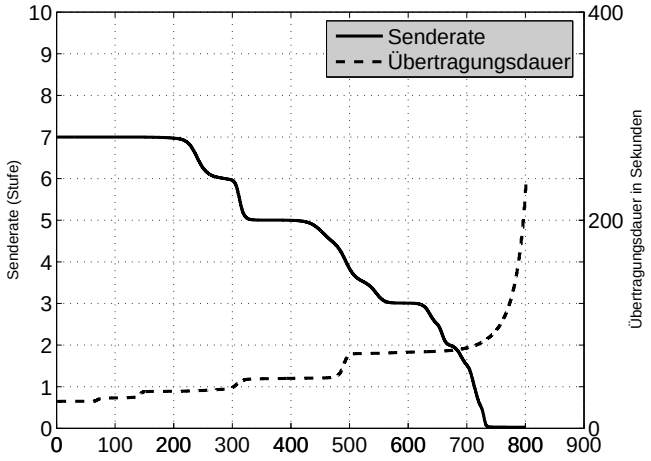


Abbildung 5.1: Senderate - Übertragungsdauer

diesem Übergangsphasen ist die Fehlerrate der Sendevorgänge genau so hoch, dass die 80 MB nicht mit einer Rate übertragen werden, sondern immer zwischen mindestens 2 gewechselt wird. Die Fehlerrate und die Ratenwechsel sind entsprechend so hoch bzw. häufig, dass die Übertragungsdauer in ihrer stufigen Ausprägung erscheint.

Beide Kurven zusammen belegen, dass der Simulator mit der Multi-Rate Erweiterung erwartungsgemäß funktioniert und bei kürzeren Entfernungen hohe Raten wählt, bei größerer Distanz entsprechend langsamere Rate, wobei die Differenzentfernung, für die eine Rate eingesetzt werden kann, wächst.

Eine Sendereichweite von 800 m entspricht natürlich nicht der Realität, dies ist für die Simulation der Netzwerke aber nicht weiter problematisch, da dies nur eine Abbildung im Simulator ist. Der Einfachheit halber wird der Koordinatenursprung nach (0,0) gelegt und 1≡1 Meter gesetzt. Durch einen entsprechenden Faktor kann dies auch realistischer abgebildet werden. Ausschlaggebend für eine funktionierende Simulation ist, dass die Senderate mit steigender Entfernung angepasst wird und abnimmt. Außerdem sind die Übergangsbereiche von besonderem Interesse, wie später noch zu sehen sein wird.

5.1.1 Überprüfung der Metriken

Im Folgenden werden die Ergebnisse dargestellt, die sich aus den Metrikü berprüfungen ergeben haben. Für diese Messung wurde ein Szenario eingesetzt, in dem nur zwei Kno-

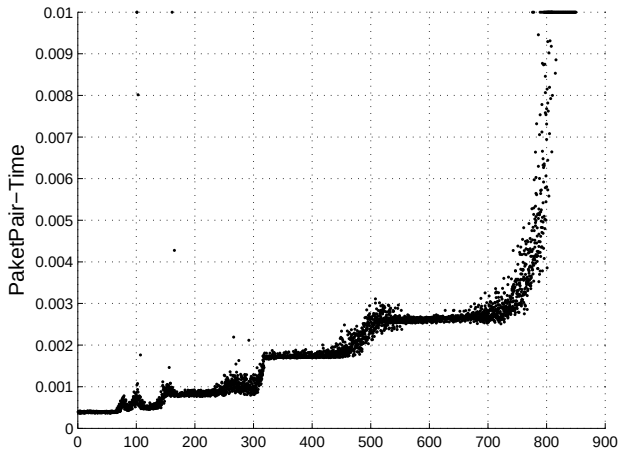


Abbildung 5.2: Messung - PktPair

ten eingesetzt werden. Von Interesse ist die Linkbewertung der Metriken, die über die Entfernung abgetragen wird. Jeder Versuch wurde 100 mal wiederholt, wobei für die Entfernung alle Werte zwischen 1 m und 850 m eingesetzt wurden.
Hop-Count und ETX wurden auf diese Weise nicht überprüft, da diese Metriken keine Messungen auf den Links vornehmen, sondern ausschließlich von Informationen des Routingprotokolls abhängen.

Packet-Pair

In Abbildung 5.2 und 5.3 sind die Werte dargestellt, die PktPair und PktPairMinimum dem Link zwischen zwei Knoten zuordnen. Beide Grafiken zeigen eine gestufte Kurve, wobei die Stufen jeweils zu den Entfernungen beginnen, an denen auch die Rate wechselt, (siehe Abbildung 5.1). Die PktPair-Variante mit Minimumauswertung zeigt einen sehr klaren und sauberen Verlauf, da im Gegensatz zur Variante ohne Minimumauswertung Ausreißer in der Messung natürlich nicht berücksichtigt werden. PktPair ohne Minimumselektion reagiert auf fehlerhafte Übertragungen sehr stark, die Gewichte für die Links gehen stark in die Höhe, so dass die Grafik nach oben begrenzt werden musste, da in der Darstellung sonst keine Stufen zu erkennen wären.
Beide Grafiken zeigen, dass die Metriken wie beabsichtigt funktionieren.

Round-Trip-Time

Die Round-Trip-Time-Metrik zeigt ebenfalls eine stufige Kurve, wobei die Ausreißer - wie in der Abbildung 5.4 zu sehen - deutlich geringer sind, als bei PktPair. Auch diese

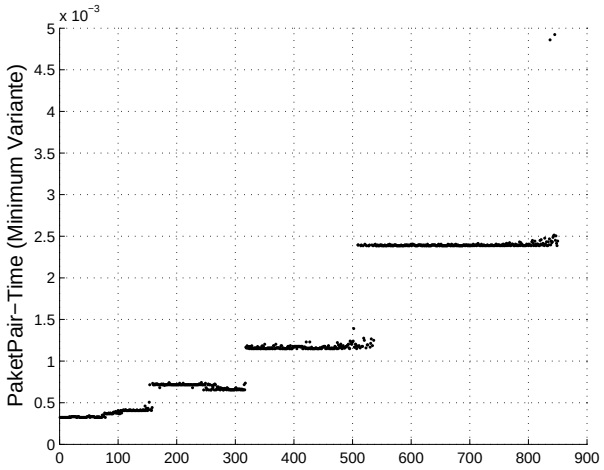


Abbildung 5.3: Messung - PktPair Minimumvariante

Metrik funktioniert ordnungsgemäß.

Transmit-Time

Durch die Skalierung der Darstellung in Abbildung 5.5 sind die Stufen in der Kurve nicht auf den ersten Blick zu erkennen. Aber auch mit der TTMetric steigt der Messwert für die Übertragungsdauer stufenweise. Diese Metrik funktioniert ebenfalls.

5.2 Evaluierung Szenario „Kette“

In diesem Teil findet die Auswertung für das Szenario statt, in dem die Übertragung einer großen Datenmenge entlang einer Reihe von Knoten gemessen wurde. Gemessen wurde die Dauer der Übertragung unter Einsatz der verschiedenen Metriken und unterschiedlichen äquidistanten Abständen der Knoten. Die Knotenanzahl wurde ebenfalls variiert. Jede Messung wurde 100 Mal wiederholt. Die Topologie dieses Netzwerks wurde schon in Kapitel 3.3.1 „Szenarien“ dargestellt.
Die Anzahl der Knoten bewegt sich von fünf bis 19, jeweils in zweier Schritten. Der Abstand zwischen den Knoten wurde auf alle Werte zwischen 5 m und 795 m mit 10er Schritten gesetzt. In den Diagrammen sind jeweils drei Linien zu erkennen, die gepunktete entspricht dem Mittelwert der Messungen, die durchgezogenen Linien grenzen das 95%-Perzentil-Intervall ab. Die Entfernung zwischen den Knoten ist horizontal aufgetragen, die Übertragungsdauer vertikal. Wenn die Übertragungsdauer mit 0 gemessen wurde, so konnte keine erfolgreiche Datenübertragung stattfinden.

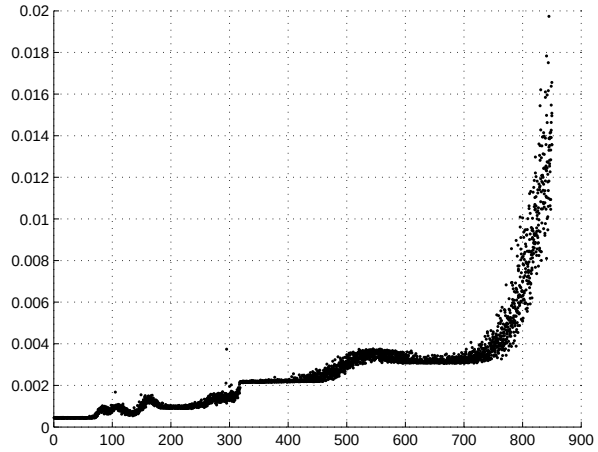


Abbildung 5.4: Messung - Round-Trip-Time

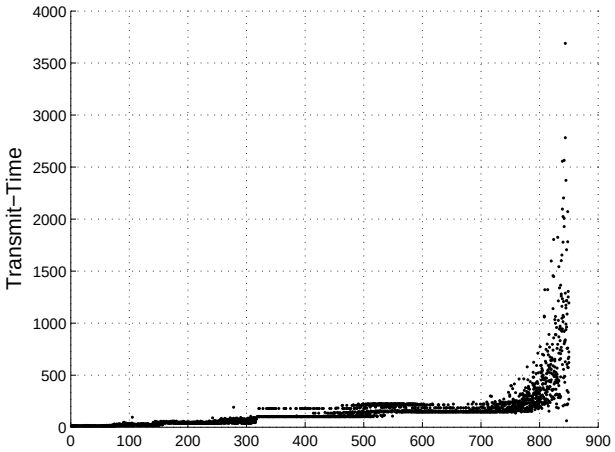


Abbildung 5.5: Messung - Transmit-Time

Allgemein ist zu erkennen, dass eine Auswertung bzw. Vergleich der Metriken ab einer Kettenlänge von 11 Knoten keinen Sinn ergibt. Sobald diese Kettenlänge erreicht wird funktionieren diese Netze nur noch, wenn die Abstände zwischen den Knoten sehr gering oder sehr hoch sind. In dem Fall von geringen Abständen kann der Zielknoten über ein oder zwei Hops erreicht werden. Wenn die Abstände sehr hoch sind, ist die Route auch eindeutig, es muss jeder Knoten genutzt werden. Außerdem nimmt die gegenseitige Beeinflussung ab.
Ab einem Abstand von ca. 450 m kann nur noch eine Route gewählt werden: Ein Knoten nach dem anderen. Einen Knoten zu überspringen ist nicht möglich, da die maximale Sendereichweite kleiner als 900 m ist. So ist es auch zu erklären, dass bei allen Metriken die Kurve ab 450 m sehr ähnlich aussieht. Ab dieser Entfernung zwischen den Knoten ist die Route klar und die Übertragungsdauer wird nur durch Übertragungsfehler verursacht. Die Fehler auslösenden Ereignisse sind dabei in allen Messungen die gleichen, wie zum Beispiel: Ratenfindung und Paketüberlagerung.

5.2.1 Hop-Count

In Abbildung 5.6 ist trotz weniger Knoten schon deutlich zu erkennen, dass Hop-Count unter der Ratenanpassung leidet. Hop-Count versucht immer eine möglichst große Strecke mit einem Hop zurückzulegen, auch dann, wenn die Pakete nur mit einer geringen Wahrscheinlichkeit ankommen. Sobald mindestens eine Übertragung stattfindet, die mit schlechter Qualität funktioniert, steigt die Sendedauer.
Je mehr Knoten auf der Strecke zwischen Start- und Zielknoten liegen, desto schneller werden kritische Gesamtentferungen erreicht. Bei fünf Knoten muss für einen Abstand von 10 m zwischen den Knoten insgesamt eine Entfernung von 50 m überbrückt werden, dies kann leicht mit einem Hop geschehen. Auch ein Abstand von 100 m kann bei fünf Knoten noch komplett mit einem Hop übverbrückt werden. Ist die Kette länger - beispielsweise 13 Knoten - so ergibt sich eine Gesamtdistanz von 1300 m. Diese wird mit einem Zwischenhop überbrückt, wobei vorher nicht festgestellt werden kann, ob es zwei Hops zu 600 m und 700 m oder zu 800 m und 500 m sind. Für den Fall von einer Gesamtdistanz von ca. 1600 m hingegen ist es wahrscheinlich, dass beide Hops ca. 800 m lang und damit sehr fehlerbehaftet sind. Diese Effekte sind deutlich daran zu erkennen, an welchem Abstand zwischen den Knoten die Übertragungsdauer übermäßig stark ansteigt. In Abbildung 5.6 geschieht dies erst bei ca. 190 m in Abbildung 5.7 schon bei ca. 140 m und in Abbildung 5.8 bei ca. 100 m.

Mit zunehmender Knotenanzahl werden die Übertragungen auf Links mit schlechter Qualität immer häufiger und die Übertragungsdauer schnellt geradezu in die Höhe. Siehe Abbildungen 5.11, 5.12 und 5.13.

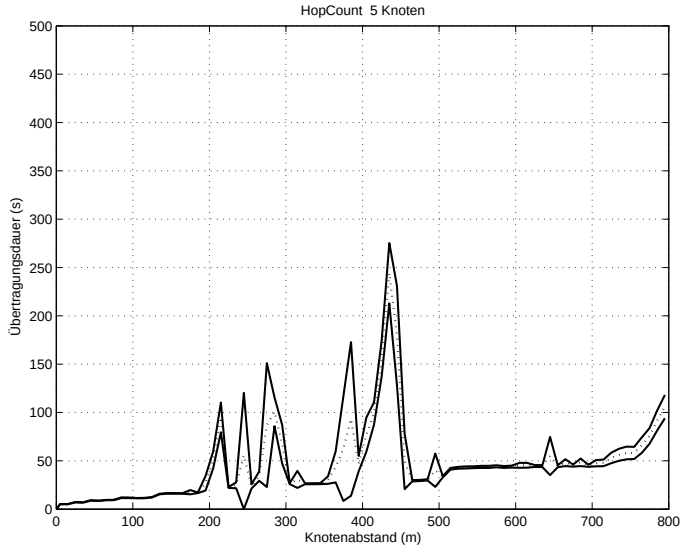


Abbildung 5.6: Hop-Count Szenario 1 mit 5 Knoten

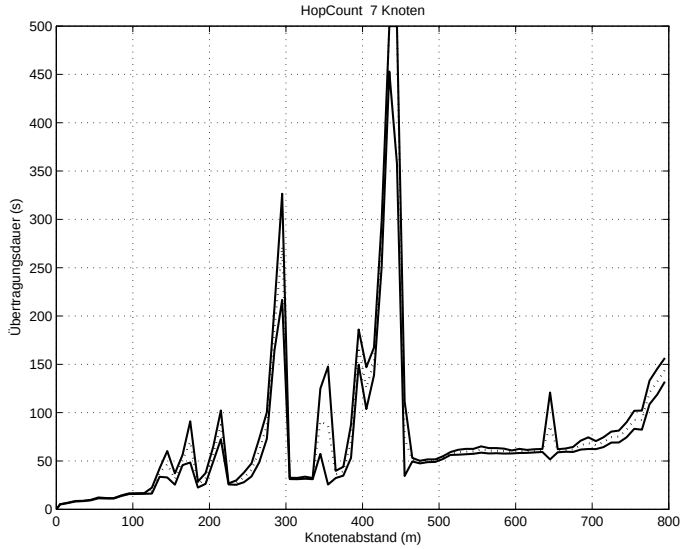


Abbildung 5.7: Hop-Count Szenario 1 mit 7 Knoten

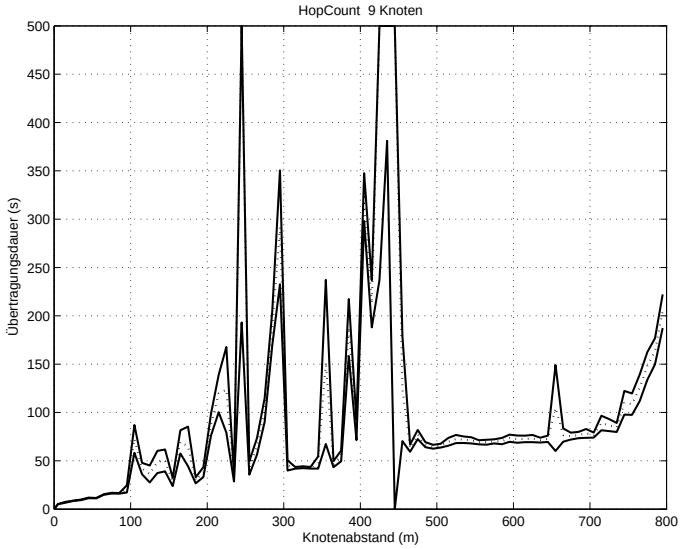


Abbildung 5.8: Hop-Count Szenario 1 mit 9 Knoten

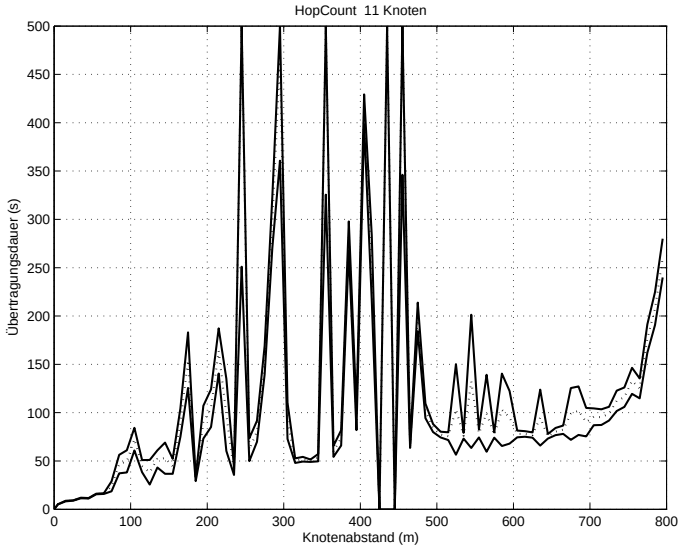


Abbildung 5.9: Hop-Count Szenario 1 mit 11 Knoten

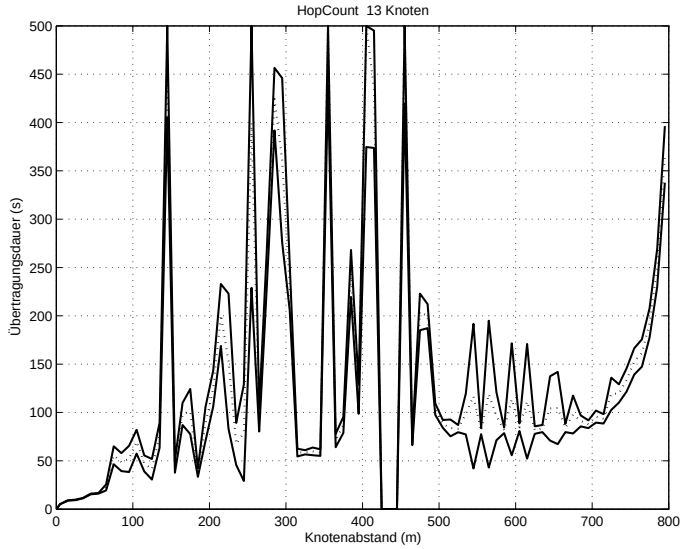


Abbildung 5.10: Hop-Count Szenario 1 mit 13 Knoten

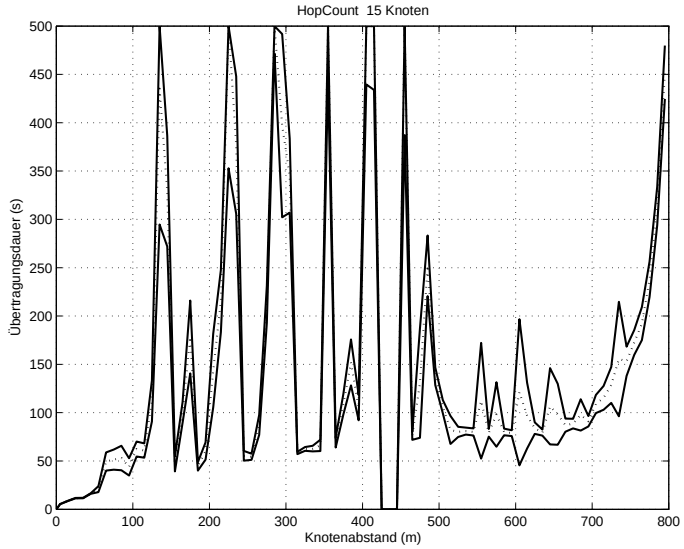


Abbildung 5.11: Hop-Count Szenario 1 mit 15 Knoten

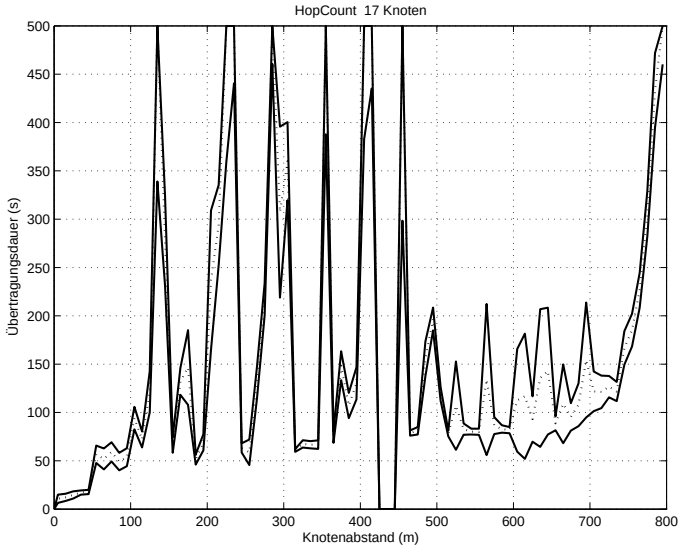


Abbildung 5.12: Hop-Count Szenario 1 mit 17 Knoten

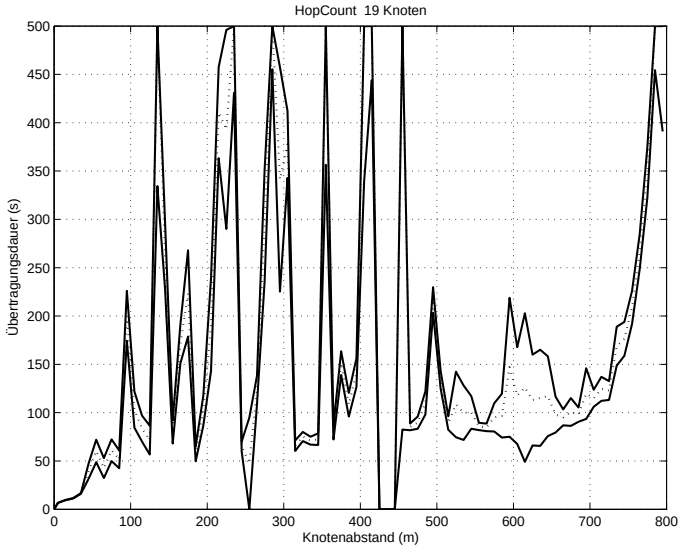


Abbildung 5.13: Hop-Count Szenario 1 mit 19 Knoten

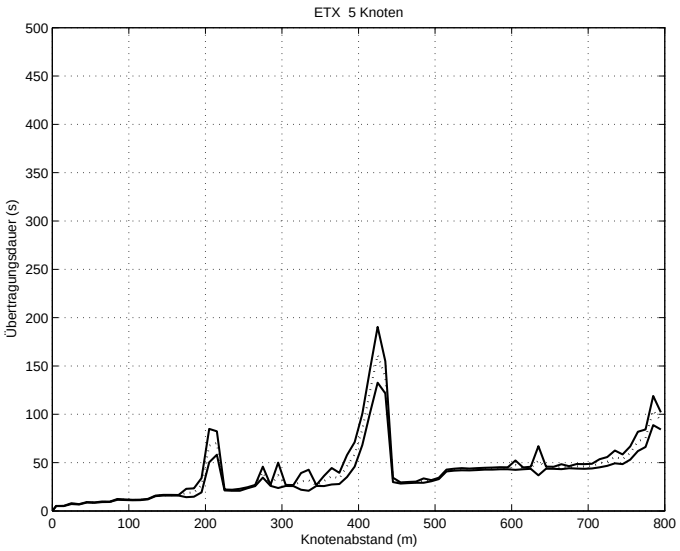


Abbildung 5.14: ETX Szenario 1 mit 5 Knoten

5.2.2 Expected-Transmit-Count

Auch bei ETX ist zu erkennen, dass sobald die kritische Sendereichweite erreicht wird, die Übertragungsdauer stärker steigt. In Abbildung 5.14 bei ca. 200 m, in Abbildung 5.15 schon bei ca. 140 m. Im Vergleich zu Hop-Count sind die Anstiege aber nicht so hoch und treten wesentlich seltener auf. ETX zieht den Sendeerfolg mit in Betracht und weicht deshalb auf andere Knoten aus, so dass bei dieser Metrik nicht versucht wird, eine möglichst lange Strecke zu überbrücken.

5.2.3 Packet-Pair

Packet-Pair (PktPair) ist im Gegensatz zu Hop-Count oder ETX eine Metrik, die die Sendedauer für einen Link in Sendereichweite durch Testpakete berechnet. Für Szenario 1 liefert PktPair gerade für eine kleine Anzahl von Knoten besonders gute Werte (siehe Abbildungen 5.22 und 5.23). Steigende Übertragungszeiten gehen schnell in die Metrik ein, so dass schnell auf alternative Routen mit geringerer Übertragungsdauer gewechselt werden kann. Wie insgesamt in den Diagrammen zu erkennen ist, steigt die Übertragungsdauer zu den kritischen Zeitpunkten immer sehr steil an; diese Eigenschaft ist gerade für Metriken, die periodisch messen von Vorteil, da so der Metrikwert stark beeinflusst wird. Für statische Netzwerkkonfigurationen ist dies von Vorteil. Sollen die Netze auch mobil sein ist diese Eigenschaft höchstwahrscheinlich von Nachteil. Kleine Veränderungen der Standorte der Knoten würden wahrscheinlich auch eine große Veränderung in der Übertragungsdauer erzeugen, dabei könnte der Messwert im nächsten Moment

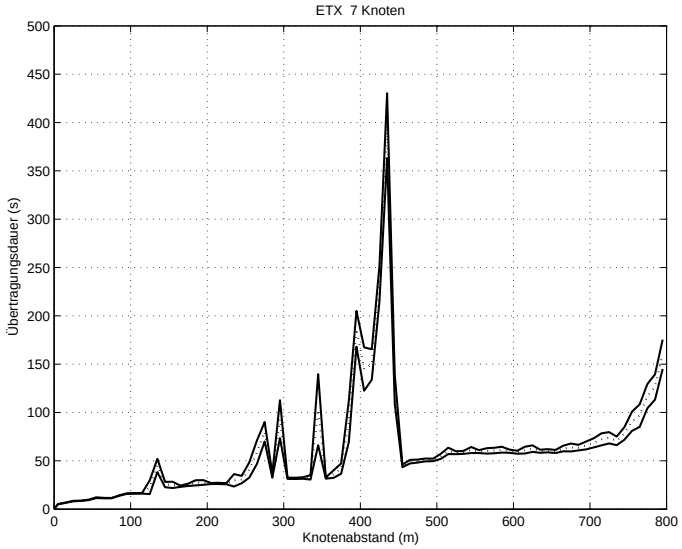


Abbildung 5.15: ETX Szenario 1 mit 5 Knoten

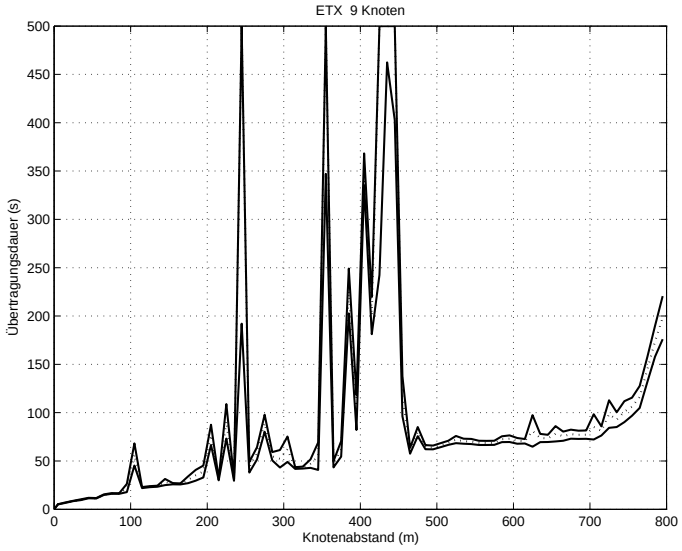


Abbildung 5.16: ETX Szenario 1 mit 9 Knoten

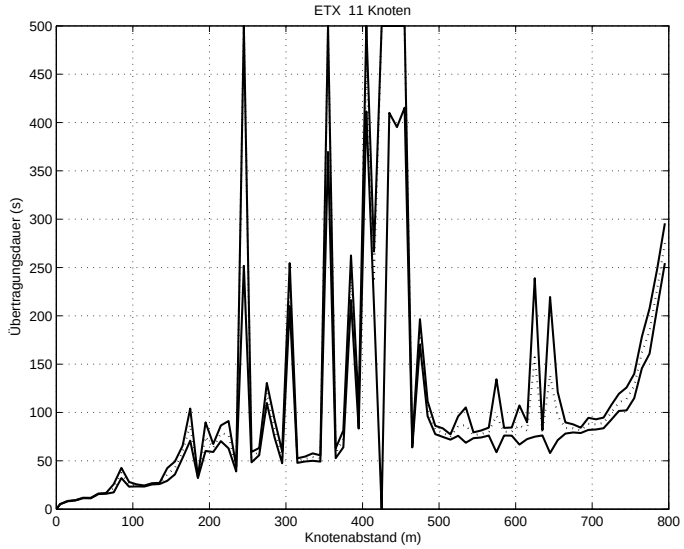


Abbildung 5.17: ETX Szenario 1 mit 11 Knoten

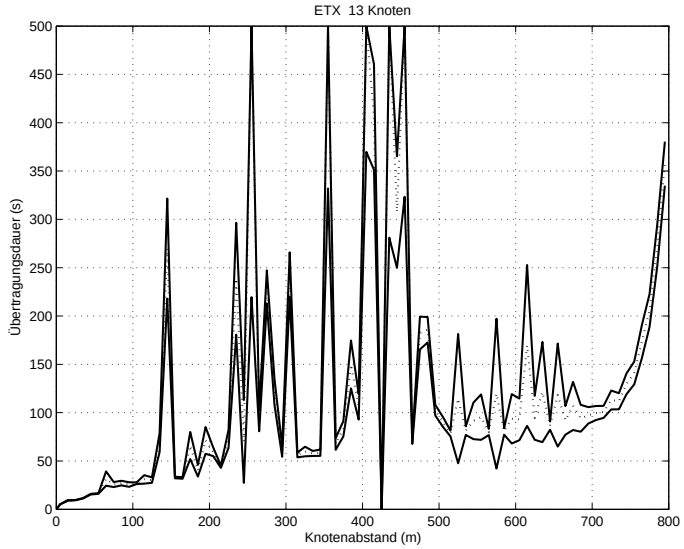


Abbildung 5.18: ETX Szenario 1 mit 13 Knoten

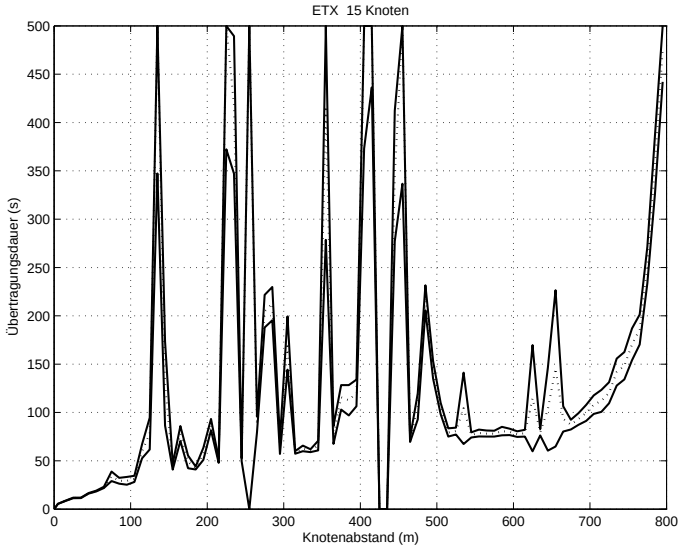


Abbildung 5.19: ETX Szenario 1 mit 15 Knoten

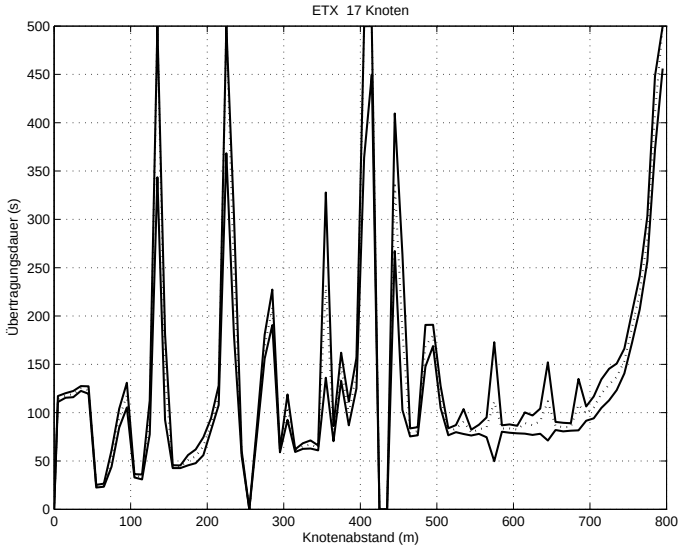


Abbildung 5.20: ETX Szenario 1 mit 17 Knoten

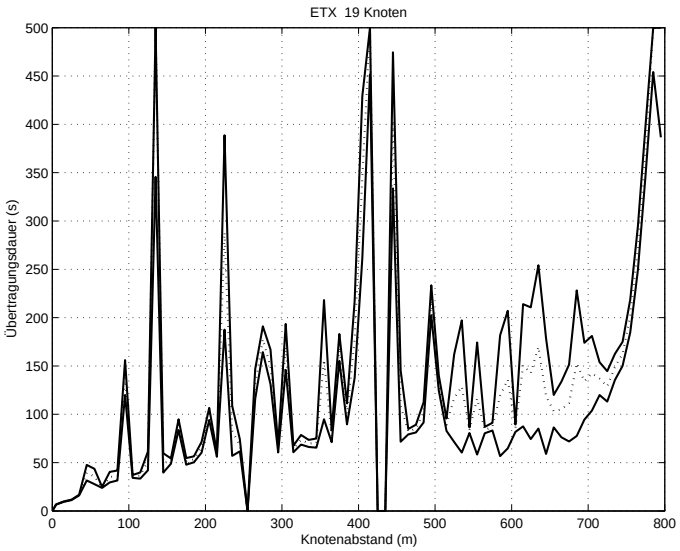


Abbildung 5.21: ETX Szenario 1 mit 19 Knoten

wieder viel besser sein. Die Frequenz, mit der die Kommunikation zu einem Nachbarknoten vermessen wird, ist also in statischen Netzwerken nicht so ausschlaggebend, wie in einem MANet.

5.2.4 Packet-Pair-Minimum

PktPair und PktPairMinimum unterscheiden sich in der Bewertung eines Links darin, dass bei der Metrik PktPair jede Messung über eine Gewichtung sofort in die Bewertung eingeht, während PktPairMinimum die kürzeste Übertragungsdauer aus den letzten x Messungen auswählt. Im vorliegenden Fall ist die Historie zehn Messungen groß. Mit dieser Variante vergeht natürlich mehr Zeit bis eine Verschlechterung der Linkqualität in die Bewertung eingeht. Eine Verbesserung der Linkqualität hingegen wird sofort berücksichtigt.
In einem statischen Szenario wie dem vorliegenden, hat dieser Unterschied der Linkbewertung keinen nennenswerten Einfluss. Die Kurven verhalten sich sehr ähnlich zu denen von PktPair.

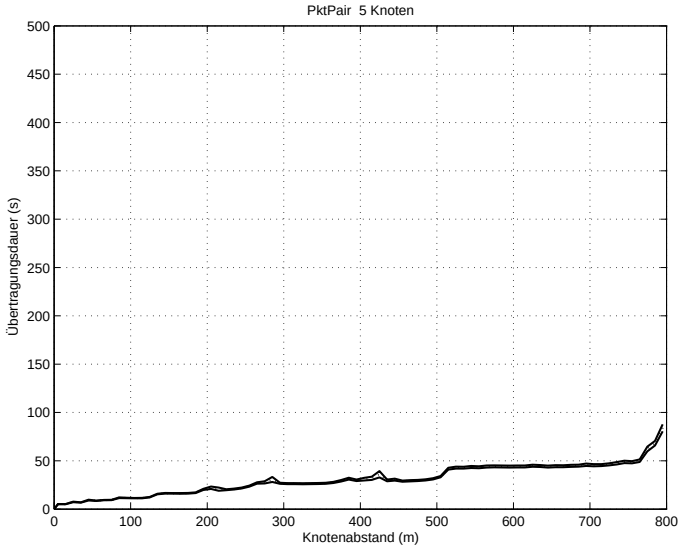


Abbildung 5.22: PktPair Szenario 1 mit 5 Knoten

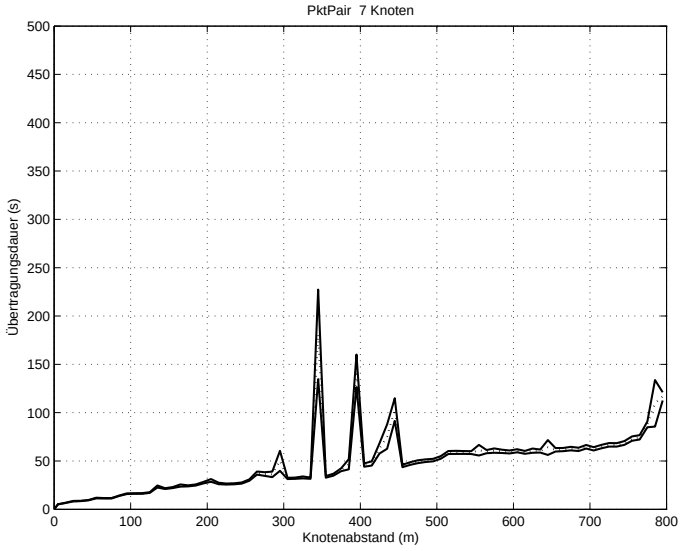


Abbildung 5.23: PktPair Szenario 1 mit 7 Knoten

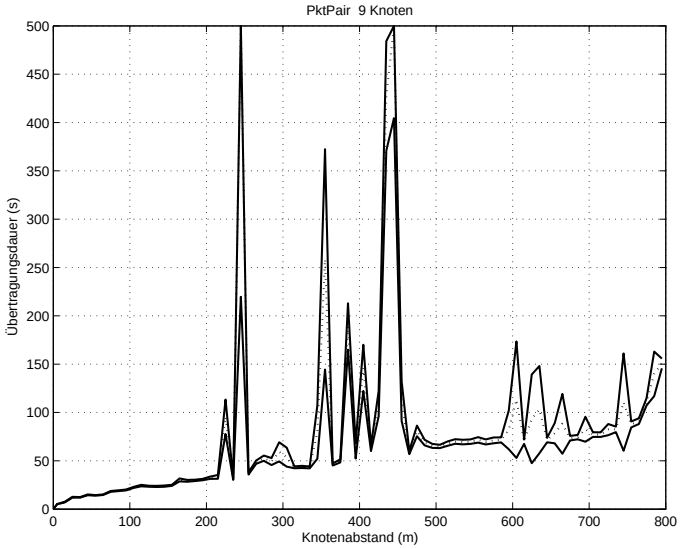


Abbildung 5.24: PktPair Szenario 1 mit 9 Knoten

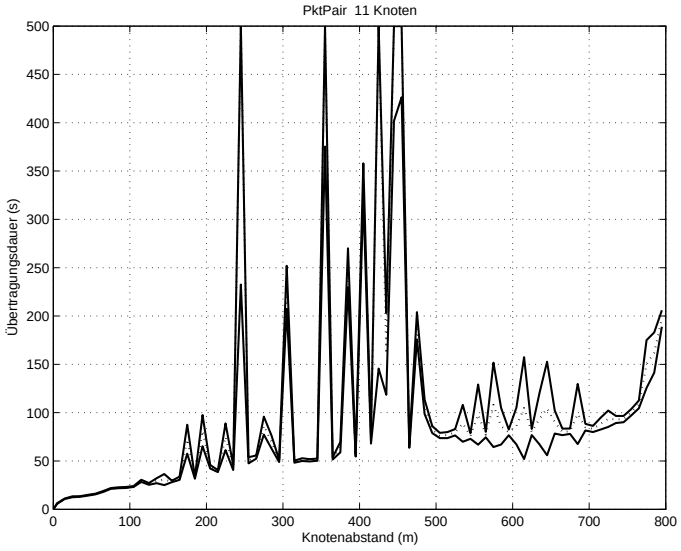


Abbildung 5.25: PktPair Szenario 1 mit 11 Knoten

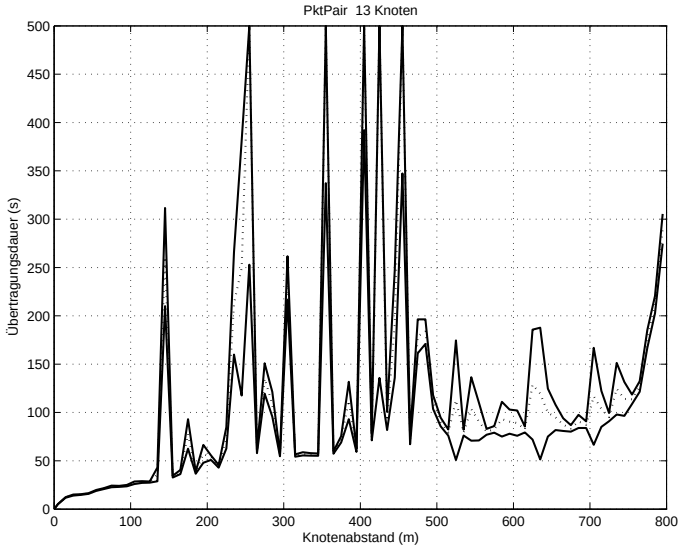


Abbildung 5.26: PktPair Szenario 1 mit 13 Knoten

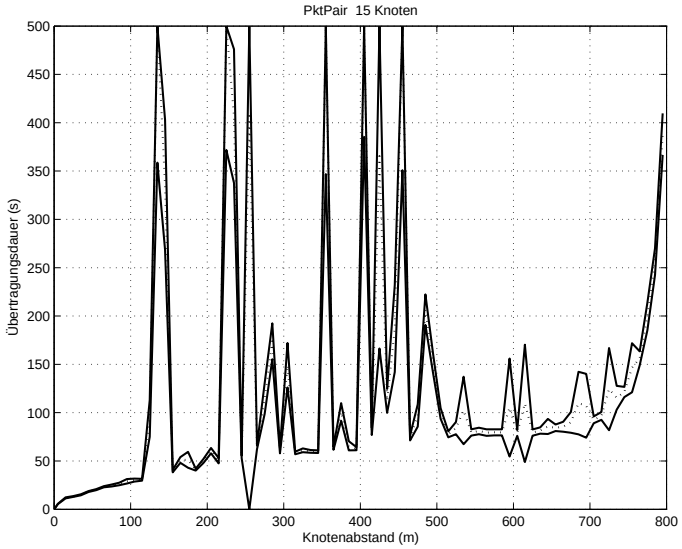


Abbildung 5.27: PktPair Szenario 1 mit 15 Knoten

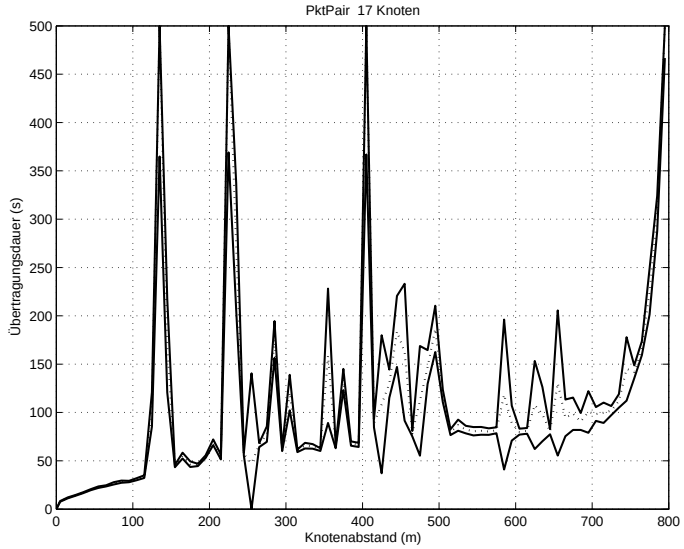


Abbildung 5.28: PktPair Szenario 1 mit 17 Knoten

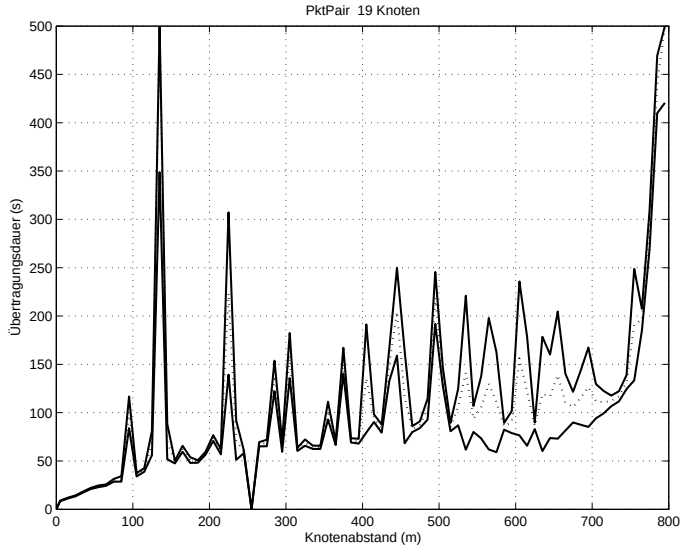


Abbildung 5.29: PktPair Szenario 1 mit 19 Knoten

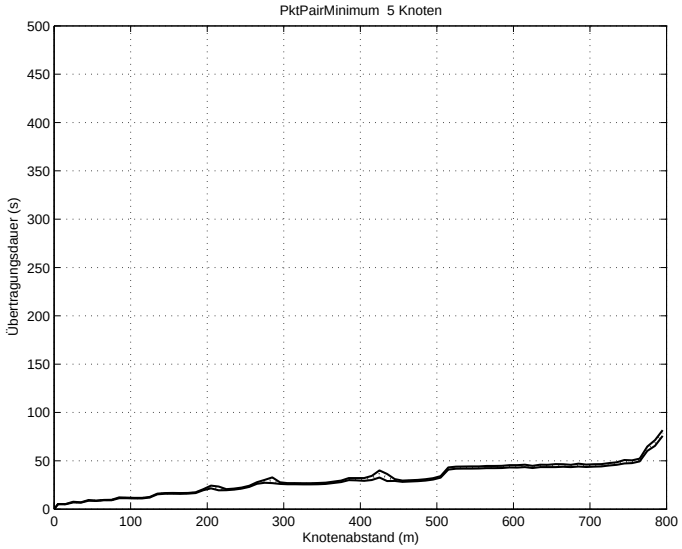


Abbildung 5.30: PktPairMinimum Szenario 1 mit 5 Knoten

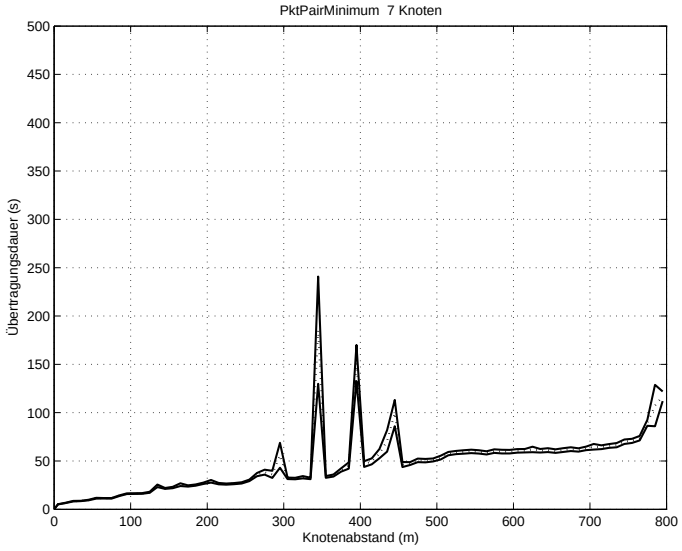


Abbildung 5.31: PktPairMinimum Szenario 1 mit 7 Knoten

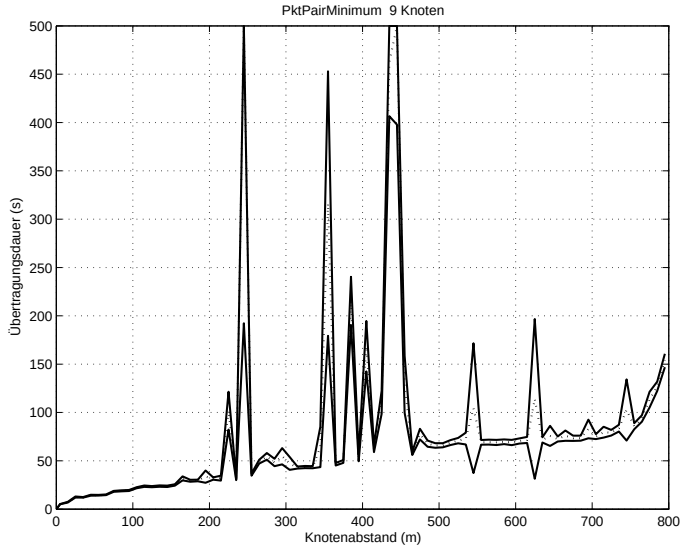


Abbildung 5.32: PktPairMinimum Szenario 1 mit 9 Knoten

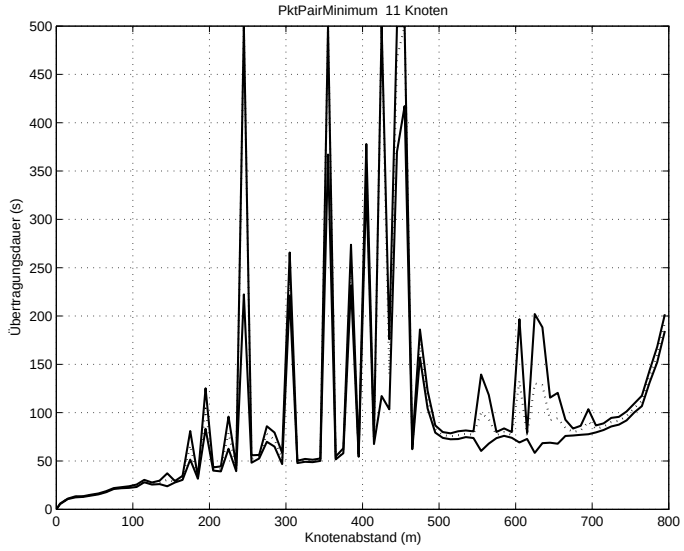


Abbildung 5.33: PktPairMinimum Szenario 1 mit 11 Knoten

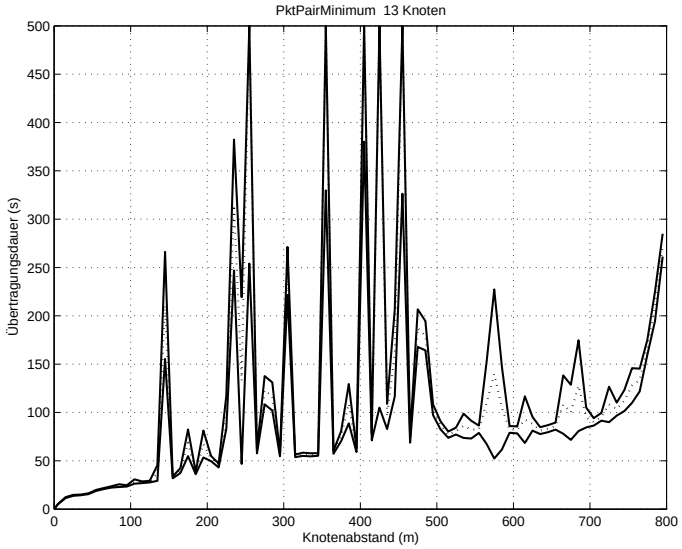


Abbildung 5.34: PktPairMinimum Szenario 1 mit 13 Knoten

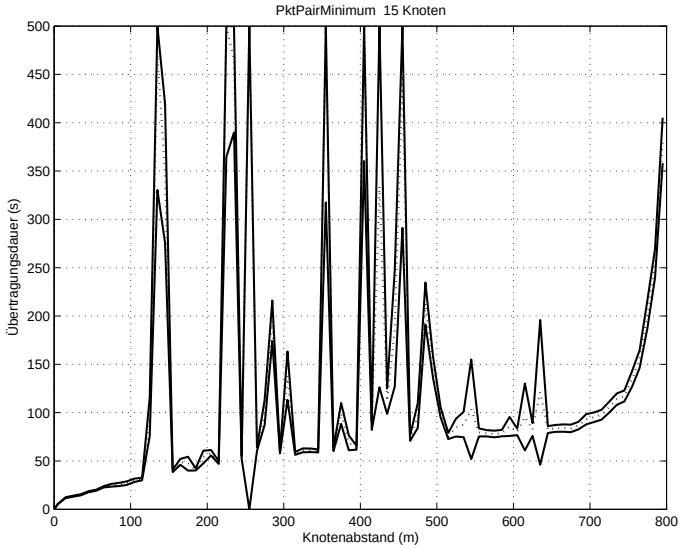


Abbildung 5.35: PktPairMinimum Szenario 1 mit 15 Knoten

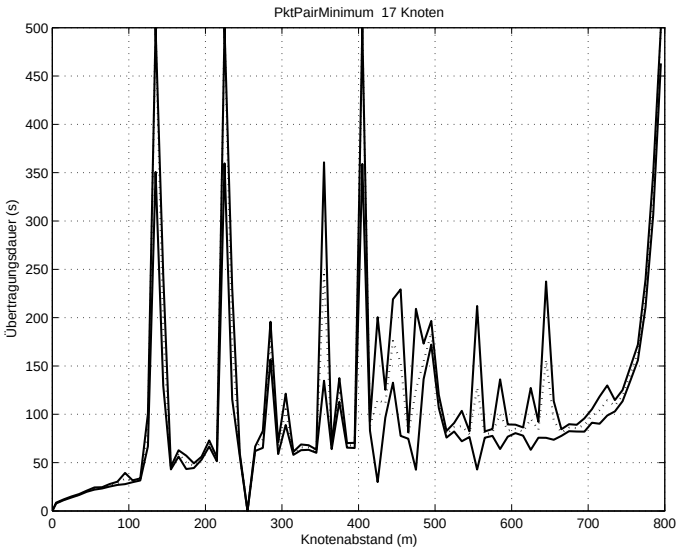


Abbildung 5.36: PktPairMinimum Szenario 1 mit 17 Knoten

5.2.5 Round-Trip-Time

Die Round-Trip-Time-Metrik liefert im Vergleich zu PktPair schlechtere Ergebnisse. Das ist dadurch zu erklären, dass die Messung der Übertragungsdauer in der RTT-Metrik nicht so gut funktioniert wie bei PktPair. Die RTT-Metrik benötigt zwei erfolgreiche Transmissionen von zwei verschiedenen Knoten. Einmal die CS-Nachricht und als Antwort das CS-Ack. Es kann nicht davon ausgegangen werden, dass die Antwortnachricht, das CS-Ack, sofort wieder gesendet wird, da der antwortende Knoten noch andere Pakete zu versenden haben kann. Außerdem kann es sein, dass das Netzwerk durch einen anderen Knoten belegt ist und so gar keine sofortige Sendemöglichkeit besteht. Durch diese Verzögerungen wird die Bewertung eines Links verfälscht und er bekommt schlechtere Werte.

5.2.6 Transmit-Time

Obwohl die Transmit-Time-Metrik die Möglichkeit hat, die Übertragungsdauer direkt von der Karte auszulesen, sind die Messergebnisse für diese Metrik schlechter als für die beiden PktPair-Varianten. Dies liegt daran, dass die TTMetric alle Übertragungen mit in die Bewertung der Links einbezieht und nicht nur bestimmte Messpakete die Gewichte der Links bestimmen. So wird auch jedes Paket, das mit der nächsthöheren Rate gesendet wird und deshalb nicht ankommt, in die Bewertung eingerechnet. Bei PktPair werden die Links nur in bestimmten Abständen ausgemessen. Daraus ergibt sich bei der TT-Metrik, dass die Links unterschiedlich stark ausgemessen werden. Auch

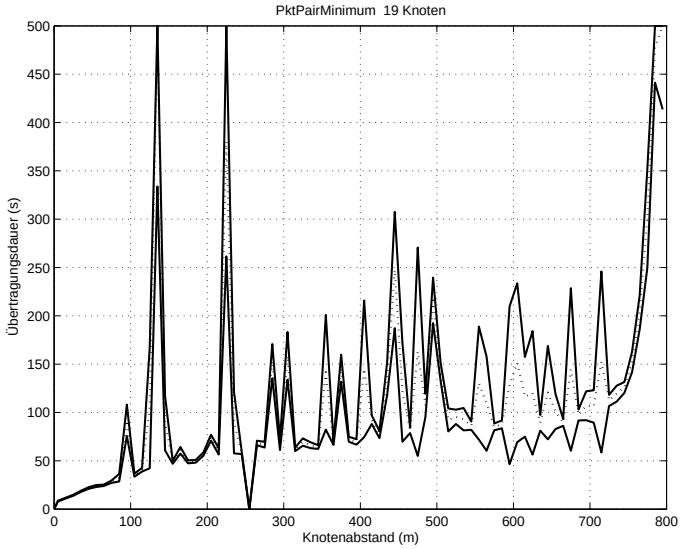


Abbildung 5.37: PktPairMinimum Szenario 1 mit 19 Knoten

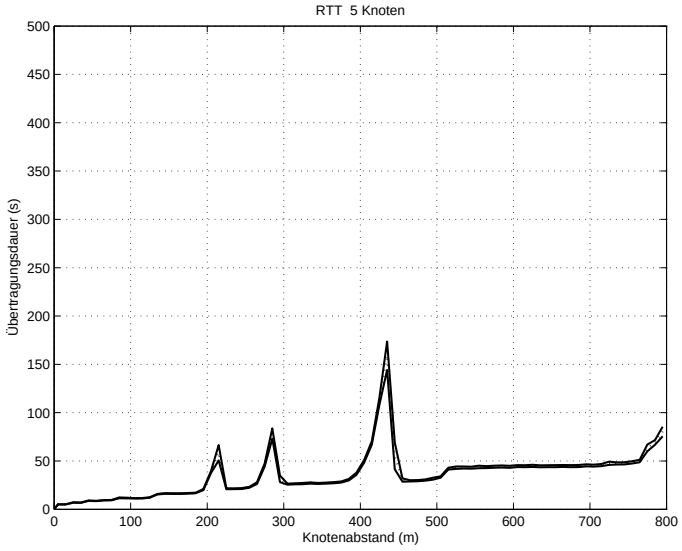


Abbildung 5.38: RTT Szenario 1 mit 5 Knoten

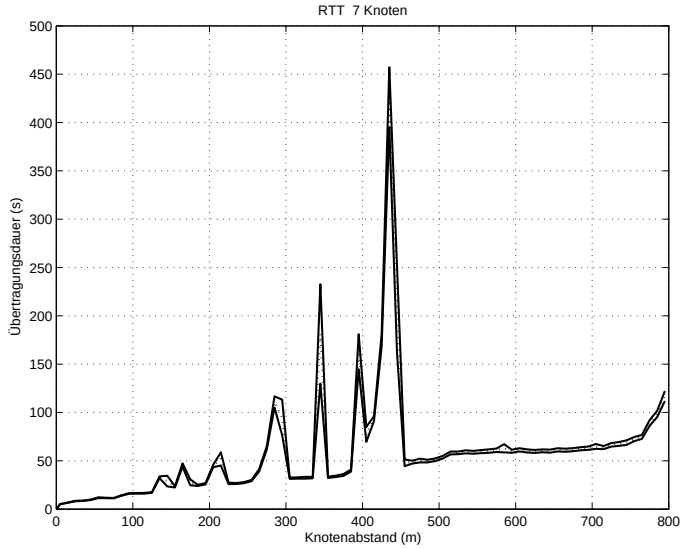


Abbildung 5.39: RTT Szenario 1 mit 7 Knoten

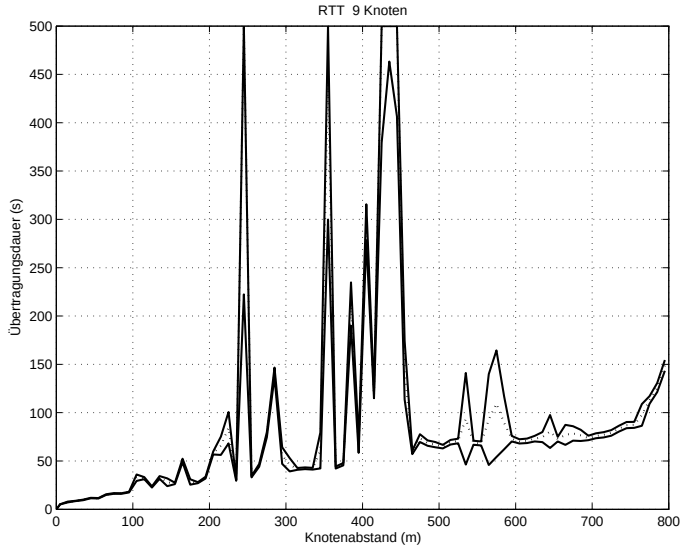


Abbildung 5.40: RTT Szenario 1 mit 9 Knoten

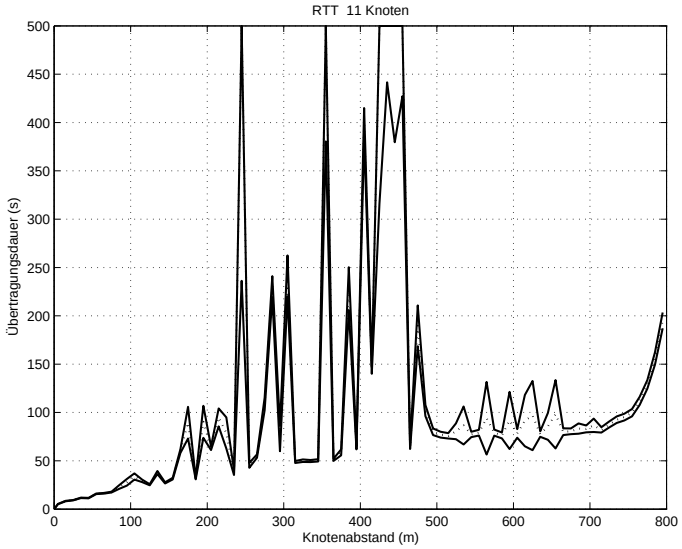


Abbildung 5.41: RTT Szenario 1 mit 11 Knoten

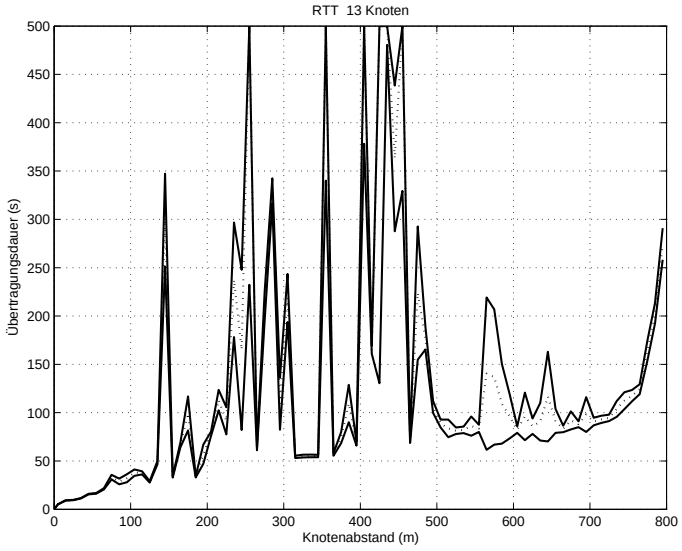


Abbildung 5.42: RTT Szenario 1 mit 13 Knoten

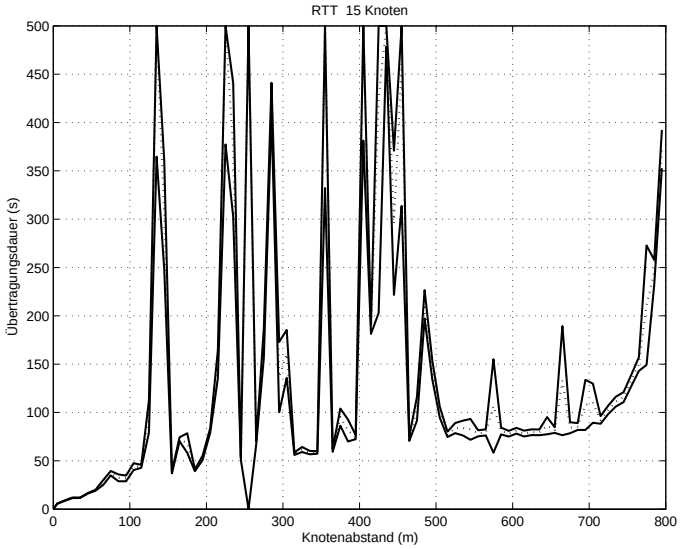


Abbildung 5.43: RTT Szenario 1 mit 15 Knoten

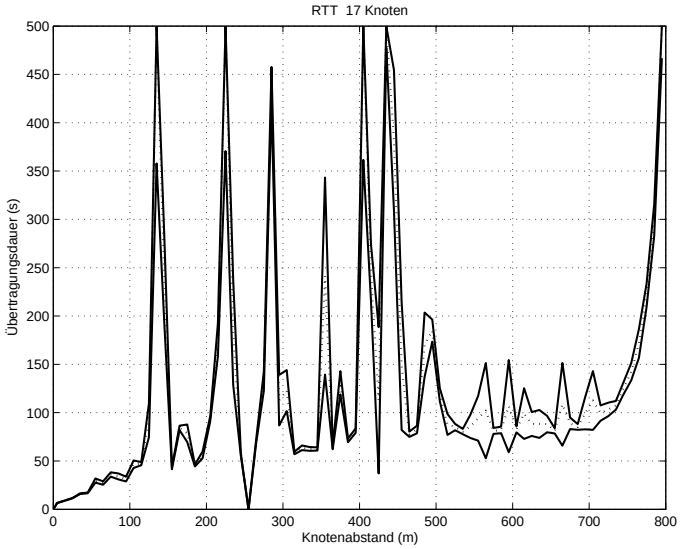


Abbildung 5.44: RTT Szenario 1 mit 17 Knoten

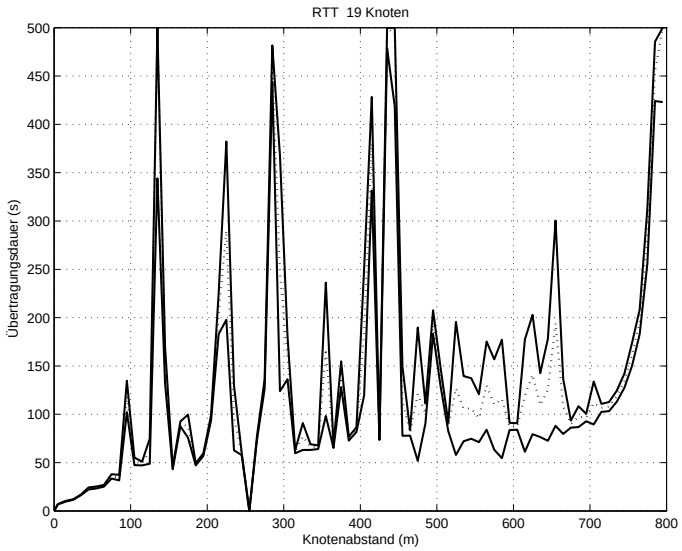


Abbildung 5.45: RTT Szenario 1 mit 19 Knoten

wenn die Fehler in der Übertragung gleichmäßig verteilt sind, ist es so wahrscheinlich, dass der vielbenutzte Link eine schlechtere Bewertung erhält. Mit folgender Rechnung wird dies schnell deutlich:
Angenommen ein Knoten habe sechs Nachbarn in Sendereichweite: Dann würde jeder Knoten alle 6 Sekunden einmal vermessen (Standardeinstellung). Das bedeutet, das ein Knoten in 60 Sekunden zehn Mal vermessen wird, alle 60 Sekunden werden die Topology- und Metrikinformationen ausgetauscht. Wenn die Fehlerrate kleiner als 10% ist, so ist es wahrscheinlich, dass dieser Knoten in 60 Sekunden ohne Fehler vermessen wird. Der Knoten, über den die Testübertragung läuft, wird hingegen deutlich häufiger als zehn Mal gemessen. Die Wahrscheinlichkeit, dass dieser Link deshalb einen schlechteren Wert bekommt, ist somit entsprechend höher. Nach dem Austausch der Metrikinformationen wird die Route gewechselt, obwohl sie ungünstiger ist. Nach weiteren 60 Sekunden wird wieder zurückgewechselt und der Zyklus beginnt von vorn.

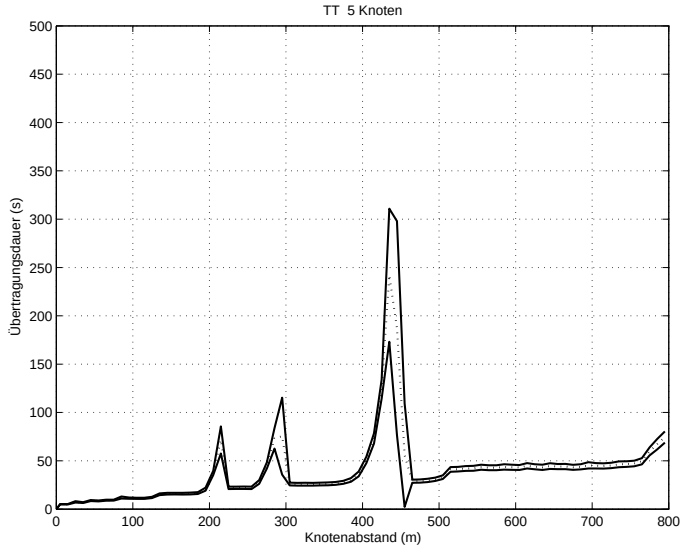


Abbildung 5.46: TT Szenario 1 mit 5 Knoten

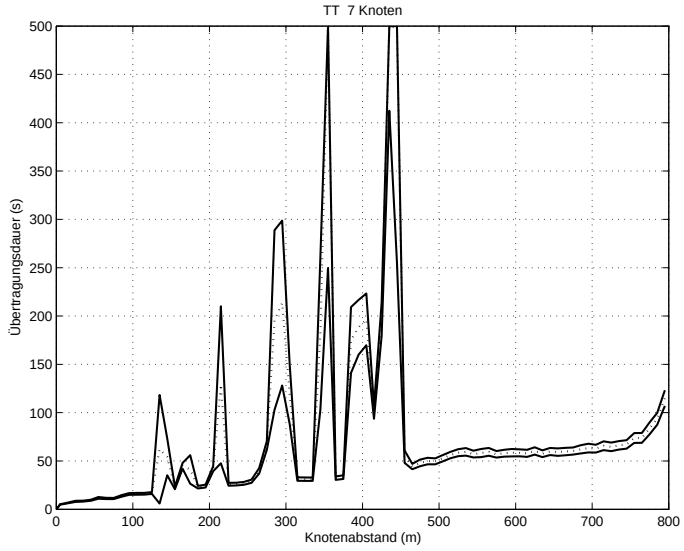


Abbildung 5.47: TT Szenario 1 mit 7 Knoten

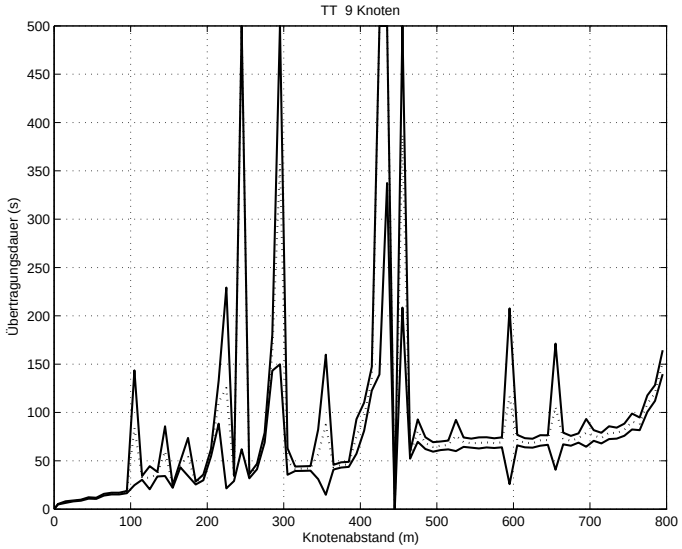


Abbildung 5.48: TT Szenario 1 mit 9 Knoten

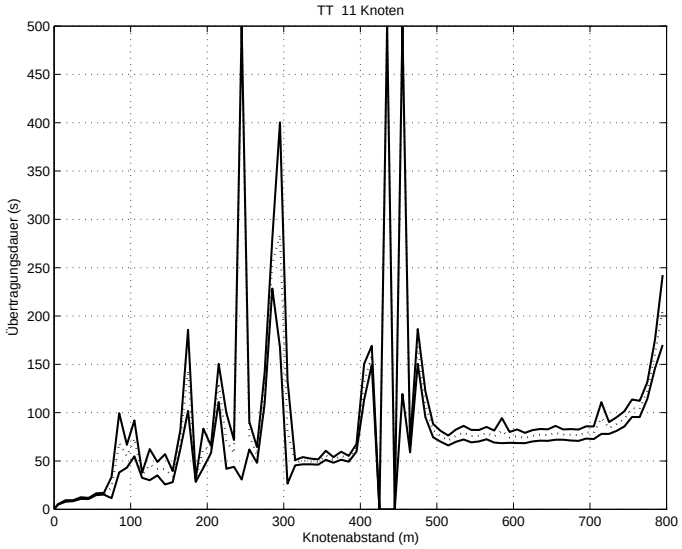


Abbildung 5.49: TT Szenario 1 mit 11 Knoten

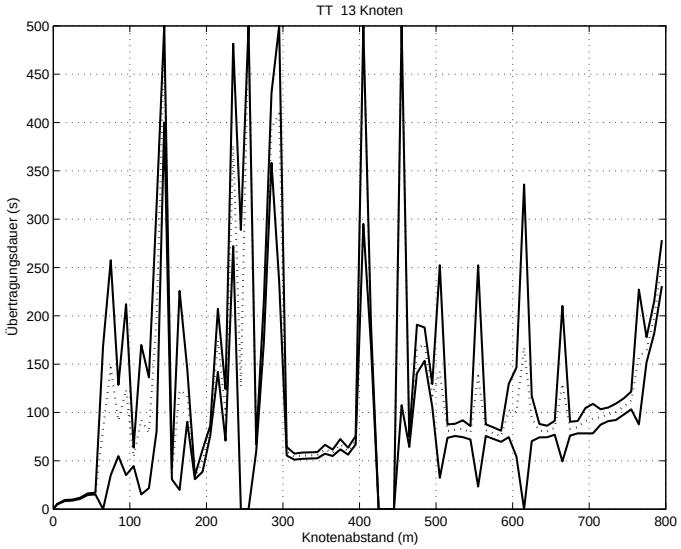


Abbildung 5.50: TT Szenario 1 mit 13 Knoten

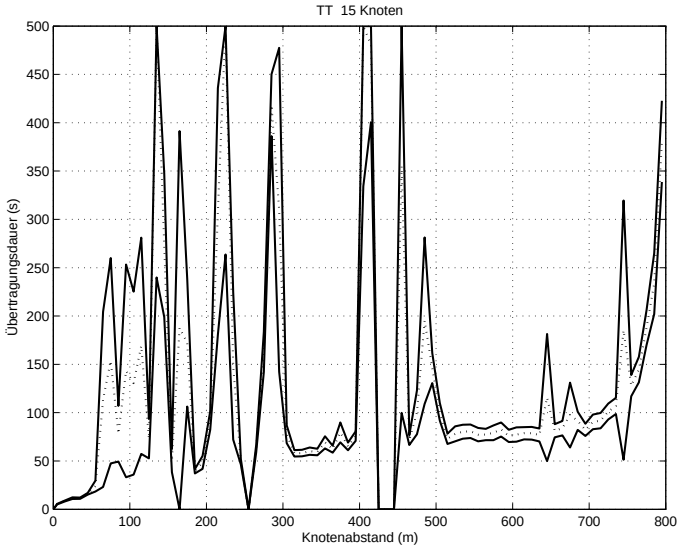


Abbildung 5.51: TT Szenario 1 mit 15 Knoten

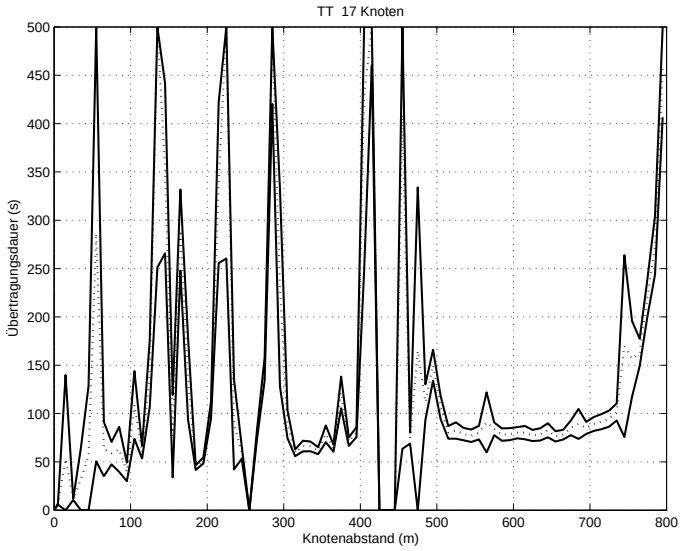


Abbildung 5.52: TT Szenario 1 mit 17 Knoten

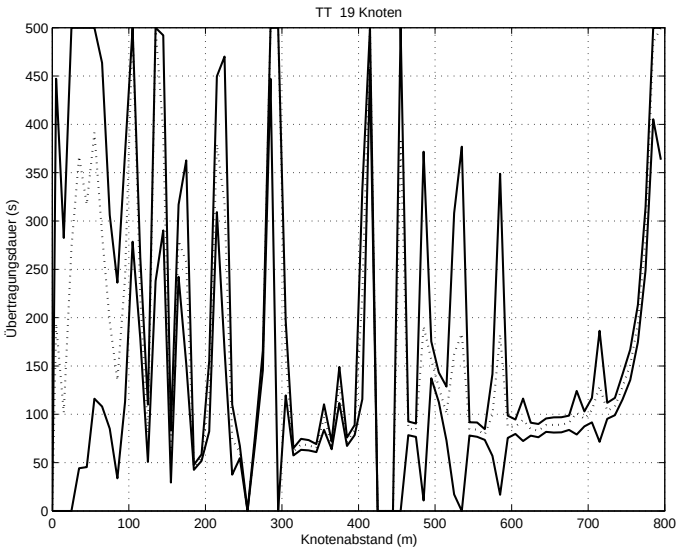


Abbildung 5.53: TT Szenario 1 mit 19 Knoten

5.3 Evaluierung Szenario „Diamant“

Die Auswertung des zweiten Szenarios ist ähnlich dem des ersten. Jedoch wurde hier nicht die Knotenanzahl direkt verändert, sondern die Anzahl der Schichten. Daraus ergibt sich, dass die Anzahl der Knoten quadratisch mit der Anzahl der Schichten wächst. Gemessen wurde die Übertragungszeit für drei bis acht Schichten, also neun bis 64 Knoten. Die übertragene Datenmenge wurde reduziert, da eine Datengröße wie bei Szenario 1 zu einer überlangen Simulationsdauer führen würde. Insgesamt sind die Ergebnisse denen aus Szenario 1 sehr ähnlich und lassen keine neuen Schlüsse zu, die Abbildungen finden sich deshalb der Vollständigkeit halber in Anhang A.
In allen Szenarien mit 49 und 64 Knoten ist zu erkennen, dass für Abstände unterhalb von ca. 290 m keine Kommunikation stattfindet. Dies ist höchstwahrscheinlich auf die gegenseitige Verdrängung zurückzuführen, die bei dieser Anzahl der Knoten auf so engem Raum dazu führt, dass die Übertragungen nicht fehlerfrei funktionieren. Die exakte Übereinstimmung dieses Ergebnisses über alle Metriken hinweg ist auf den Simulator zurückzuführen, in einem realen Netzwerk wäre dies wahrscheinlich nicht so zu beobachten.

5.4 Ergebnisse

Die untersuchten Metriken lassen sich anhand verschiedener Kriterien in Gruppen einteilen, wobei die Unterteilung, dargestellt in Tabelle 5.1, die Einheit der Linkgewichte

Metrik	Wahrscheinlichkeit	Dauer
Hop-Count	x	
Expected-Transmit-Count	x	
PktPair		x
PktPairMinimum		x
Round-Trip-Time		x
Transmit-Time		x

Tabelle 5.1: Einheiten der Linkgewichte

Metrik	Direkt	Messung
PktPair		x
PktPairMinimum		x
Round-Trip-Time		x
Transmit-Time	x	

Tabelle 5.2: Messverfahren

liefert. Hop-Count und ETX sind Metriken, die eine Übertragungswahrscheinlickeit liefern, außerdem basieren diese Metriken allein auf der Auswertung von Broadcastpaketen; dadurch wird die eingesetzt Senderate nicht berücksichtigt. In den Messungen ist zu erkennen, dass Metriken, die die Übertragungsdauer als Linkgewicht benutzen, bessere Ergebnisse liefern.

Die Metriken, die die Übertragungsdauer messen, lassen sich anhand des Messverfahrens weiter unterteilen, siehe Tabelle 5.2. Ein direktes[1] Bestimmen der Übertragungsdauer scheint sinnvoller zu sein, als eine Messung durch die Versendung von Testpaketen. Dennoch liefert nicht die Transmit-Time-Metrik die besten Ergebnisse, sondern PktPair bzw. PktPairMinimum. Dieses überraschende Ergebnis wird später genauer betrachtet. Das Messverfahren der RTT-Metrik ist anfälliger für Fehler in der Messung als das von PktPair und PktPairMinimum, dies wurde schon in 5.2.5 dargestellt. Das Verfahren von PktPair und PktPairMinimum ist robuster, da die Auswertung nicht am initiierenden Knoten stattfindet. So ist zu erklären, dass die Simulationen für PktPair und PktPairMinimum bessere Ergebnisse als die für RTT liefern.

PktPair und PktPairMinimum unterscheiden sich noch in einer weiteren Eigenschaft von der TT-Metrik: Den Messintervallen. Bei der TT-Metrik beeinflusst jedes übertragene Paket die Bewertung für den Link, auf dem die Übertragung stattfindet. Die beiden

[1] Zur Bestimmung der Übertragungsdauer ist keine Messverfahren durch Pakete notwendig, die Übertragungsdauer kann direkt von der WLAN-Karte ausgelesen werden.

PktPair-Varianten vermessen jeden Link mit dem gleichen Intervall; als Standard ist eine Messung pro Sekunde eingestellt, wobei die Nachbarknoten nacheinander abgearbeitet werden. Daraus ergibt sich bei x Knoten eine Messung alle x Sekunden. Die TT-Metrik muss ebenfalls regelmäßige Messungen durchführen, damit nicht benutzte Links ebenfalls aktuelle Gewichte erhalten. Wenn eine Übertragung stattfindet, ergibt sich so ein deutliches Ungleichgewicht in der Verteilung der Messungen. Eine kleine Rechnung zeigt, dass die Routen dadurch oft gewechselt werden und nicht dauerhaft die optimale Route gewählt wird (siehe 5.2.6). So ist zu erklären, dass die TT-Metrik trotz eines besseren Messverfahrens weniger gute Ergebnisse liefert als PktPair und PktPairMinimum.
Hiermit ergeben sich folgende Eigenschaften, die eine Metrik haben sollte, hier aber von keiner Metrik auf einmal erfüllt werden:

- Übertragungsdauer als Linkgewicht
- Direkte Messung
- Gleichverteilte Messungen

Die **Dauer** ist der Wahrscheinlichkeit einer Übertragung vorzuziehen, in Multi-Rate-Netzwerken ist die Wahrscheinlichkeit alleine keine verlässliches Maß. Weiter ist eine **direkte Messung** mit Hilfe der Netzwerkkarte genauer als über ein indirektes Messverfahren durch versendete Testpakete. Die **Gleichverteilung** der Messungen erlaubt einen sinnvollen Vergleich der berechneten Linkgewichte, ungleich verteilte Messungen erzeugen Werte, die bestimmte Links benachteiligen.

Weiterhin kann festgestellt werden, dass die Messung der Übertragungsdauer nicht nur von dem Messverfahren abhängt, sondern auch davon, wie Übertragungsfehler in den Messwert eingehen. Diese werden bisher als hoher Wert in das Gewicht eingerechnet. Hierdurch kann es zu einer nicht optimalen Routenfindung kommen. Diesem Problem könnte dadurch begegnet werden, dass die fehlerhaften Übertragungen nicht mehr als Dauer mit eingerechnet werden, sondern eine Wahrscheinlichkeit, ähnlich dem ETX-Wert, bestimmen. Dieser Wert wäre verschieden von dem der ETX-Metrik, da er nicht durch Broadcastpakete bestimmt wird, sondern durch Unicastpakete, die mit der entsprechenden Senderate gesendet werden. Außerdem sollte dieser Wert über einen längeren Zeitraum, als ein Topologieaustausch-Intervall bestimmt werden. Auf diese Weise könnten zwei unabhängige Werte - Übertragungsdauer und Übertragungswahrscheinlichkeit - ermittelt werden und als Produkt das Linkgewicht bilden.
Regelmäßige Übertragungsfehler, wie sie bei Senderate-Adaptions-Mechanismen auftreten, würden auf diese Weise nicht zu einem fehlerhaften Routenwechsel führen. Ebenso würde eine ungleich verteilte Messung ausgeglichen werden.

6 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden acht Routingmetriken für Maschennetzwerke vorgestellt. Diese Metriken wurden in anderen Arbeiten unabhängig voneinander entwickelt und nur teilweise gegeneinander verglichen. Aus diesen acht Metriken wurden fünf ausgewählt und mit einer weiteren neuen Metrik implementiert. Die Implementierung wurde mit Hilfe der Generic-Event-Api so durchgeführt, dass die Metriken sowohl im Simulator NS2, als auch in realen Netzwerken eingesetzt werden können.
Die implementierten Metriken wurden in zwei verschiedenen Netzwerkszenarien eingesetzt und mittels einer Simulation wurde die Dauer der Übertragung einer großen Datei gemessen. Verändert wurden in den einzelnen Szenarien die Abstände zwischen den Knoten, so dass untersucht werden konnte, wie unterschiedliche Senderaten die Metrik und damit die Routenfindung und der Gesamtdatendurchsatz von Start- zu Zielknoten beeinflusst werden. Auf diese Weise sollte festgestellt werden, welche Metrik in einem Multi-Rate-Maschennetzwerk die besten Routen liefert.
Neben unterschiedlichen Übertragungsdauern, die durch die Metriken erzeugt werden - alle anderen Parameter sind für jede Metrik dieselben - wurde in den Messergebnissen vor allem Folgendes festgestellt: Nahezu alle Metriken hatten Schwierigkeiten die Links gut zu bewerten, wenn sich die Abstände der Knoten in einem Ratenübergang befanden. Nur PktPair konnte für eine geringe Anzahl von Knoten auch bei diesen Abständen gute Ergebnisse liefern. Für die Simulationen wurde der Netzwerksimulator NS2 so angepasst, dass Multi-Rate-Maschennetzwerke simuliert werden können und die Metriken mittels der Generic-Event-Api geladen werden konnten.

Unterschiedliche Metriken erzeugen verschiedene Routen in gleichen Szenarien, das ist ein Ergebnis, das auch zu erwarten war. Die naivste Metrik Hop-Count liefert, ebenfalls wie erwartet, die schlechtesten Ergebnisse. Die besten Ergebnisse wurden von PktPair bzw. PktPairMinimum erzeugt, die neue Transmit-Time-Metrik war überraschenderweise nicht so effizient wie erwartet. Durch den Vergleich der Messungen konnten drei Eigenschaften für Routingmetriken in Multi-Rate-Maschennetzwerken identifiziert werden: Das Linkgewicht sollte die **Übertragungsdauer** sein, diese haben bessere Ergebnisse geliefert als Wahrscheinlichkeiten. Eine **direkte Messung** der Übertragungsdauer, also mit Unterstützung der WLAN-Karte ist genauer als ein Messverfahren das die Zeiten von ein- bzw. ausgehenden Paketen misst. Nur eine **gleichverteilte Messung** bezüglich aller Nachbarknoten liefert vergleichbare Werte.

Der automatische Senderatenwechsel ist für regelmäßige Übertragungsfehler verantwortlich, die bei ungleich verteilten Messungen zu zyklischen Wechsel der Route führen. Ungleich verteilte Messungen sind vor allem bei der TT-Metrik aufgetreten, wodurch zu erkennen war, dass fehlerhafte Übertragungen nicht als (lange) Übertragungsdauer in die Gewichte der Links eingehen sollten. Die ETX-Metrik benutzt eine Übertragungswahrscheinlichkeit, um die Links zu bewerten. Allerdings wird dieser Wert aus Broadcast-Paketen bestimmt, so dass dieser nicht mit der eingesetzten Senderate in Zusammenhang steht. Dennoch scheint dieser Ansatz für zukünftige Metriken in Multi-Rate-Netzwerken, in Kombination mit der Übertragungsdauer, viel versprechend zu sein.

Ausblick

Routingmetriken für drahtlose Netzwerke erzeugen in Multi-Rate-Maschennetzen im Allgemeinen nicht optimale Routen, dies ist darauf zurück zu führen, dass die Metriken nicht speziell für Multi-Rate entwickelt wurden. Durch diese Arbeit konnten Bedingungen ermittelt werden, die für eine Routingmetrik in Multi-Rate-Maschennetzwerken notwendig sind; keine der vorhandenen Metriken erfüllt derzeit alle. Weiter konnte abgeleitet werden, dass eine Kombination von Dauer und Wahrscheinlichkeit der Übertragung in Kombination mit einer direkten Messung der Übertragungsdauer wahrscheinlich die besten Ergebnisse liefern wird. Die TT-Metrik sollte diesbezüglich angepasst werden.

Außerdem sollten die Metriken weiter gegeneinander verglichen werden. Die Mehrheit der implementierten Metriken hat verschiedene Parameter, um ihr Verhalten zu beeinflussen, beispielsweise das Intervall der Link-Messungen. Eine erschöpfende Menge an Simulationen mit unterschiedlichen Parametern konnte in dieser Arbeit nicht durchgeführt werden, da die Menge und damit die Dauer der Simulationen den Rahmen dieser Arbeit gesprengt hätte.

A Messergebnisse Szenario 2

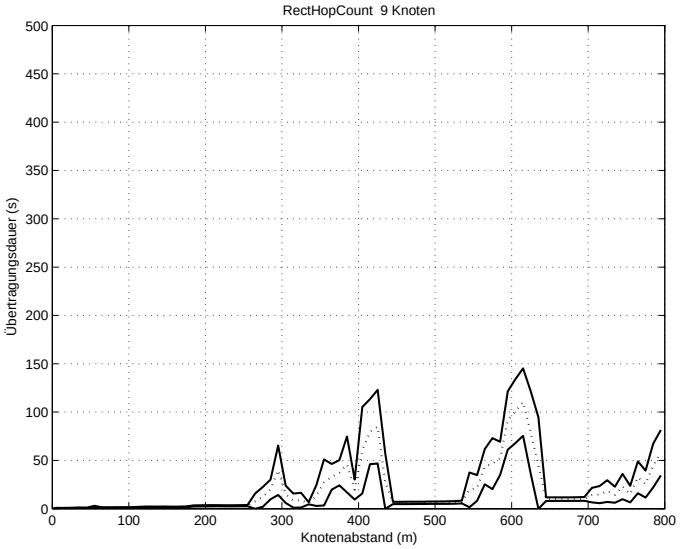


Abbildung A.1: Hop-Count Szenario 2 mit 9 Knoten

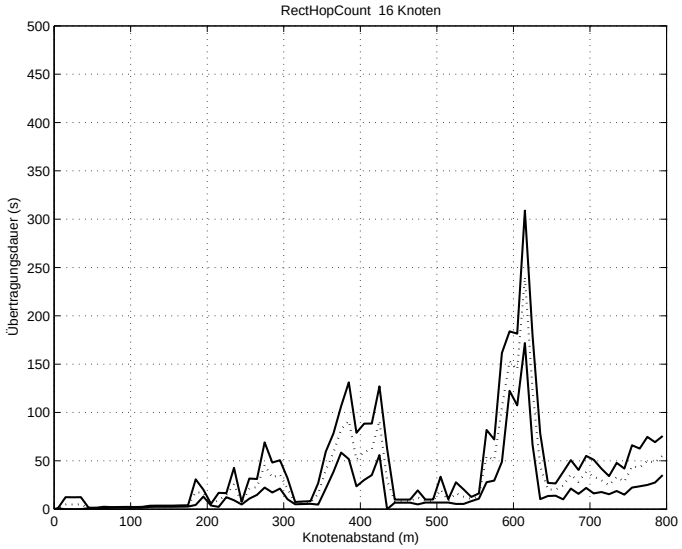


Abbildung A.2: Hop-Count Szenario 2 mit 16 Knoten

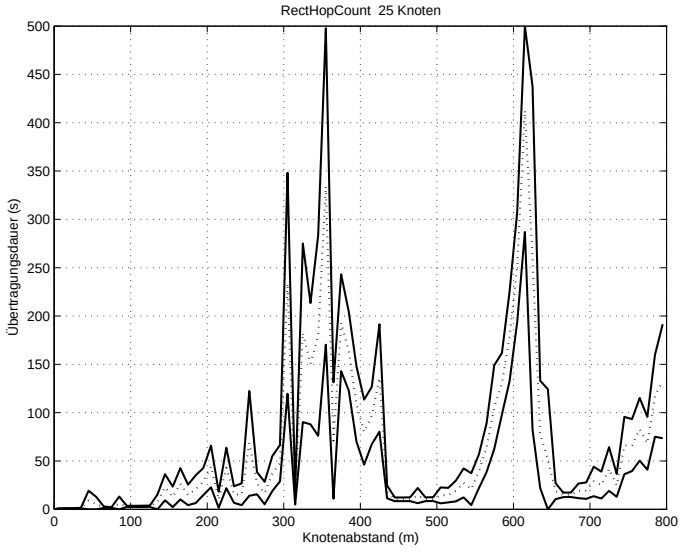


Abbildung A.3: Hop-Count Szenario 2 mit 25 Knoten

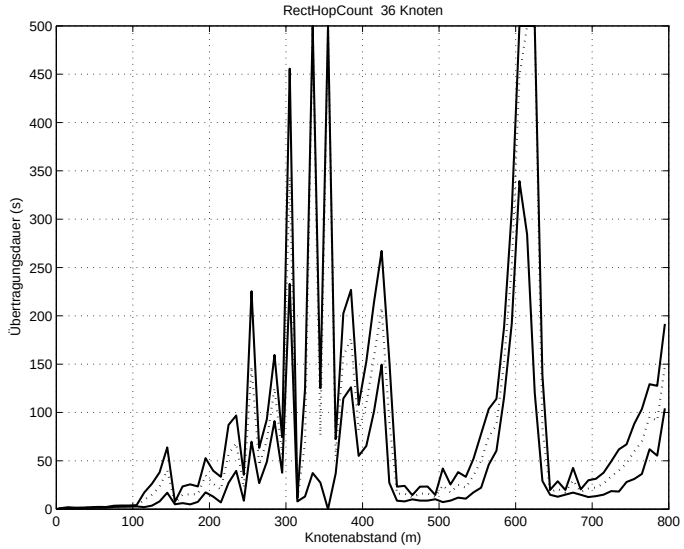


Abbildung A.4: Hop-Count Szenario 2 mit 36 Knoten

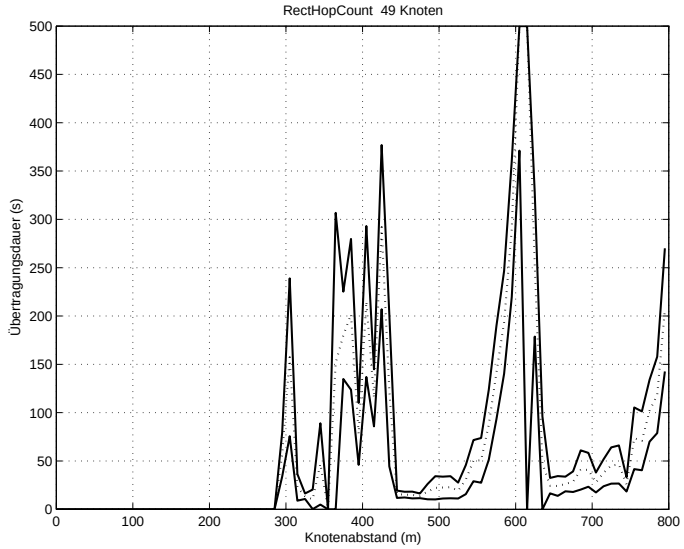


Abbildung A.5: Hop-Count Szenario 2 mit 49 Knoten

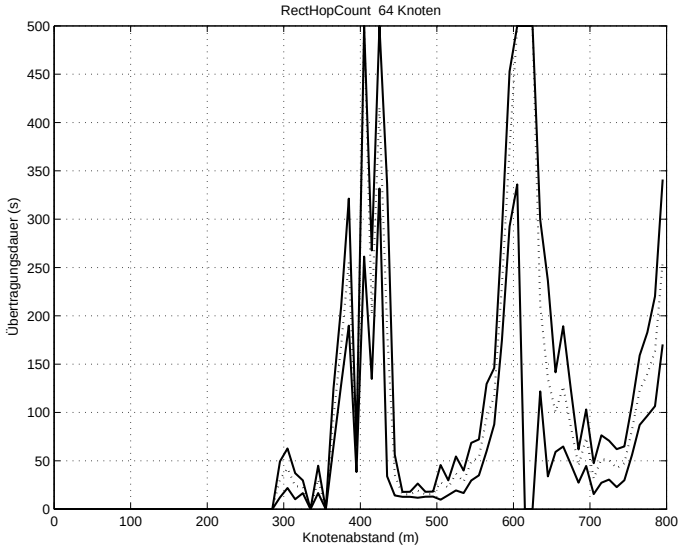


Abbildung A.6: Hop-Count Szenario 2 mit 64 Knoten

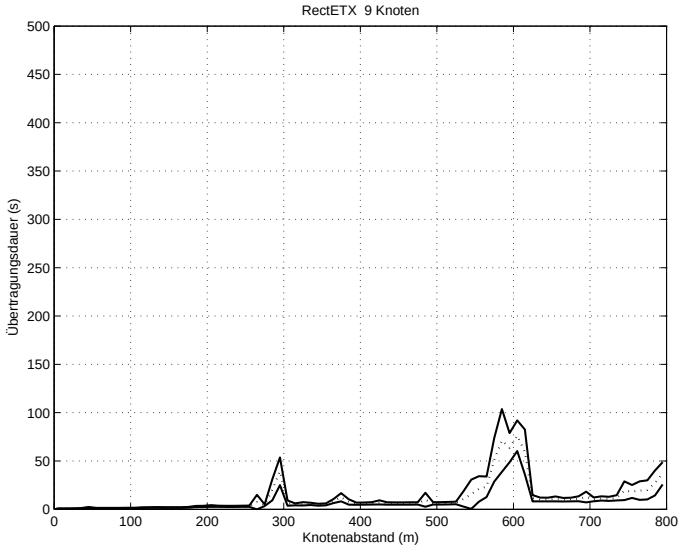


Abbildung A.7: ETX Szenario 2 mit 9 Knoten

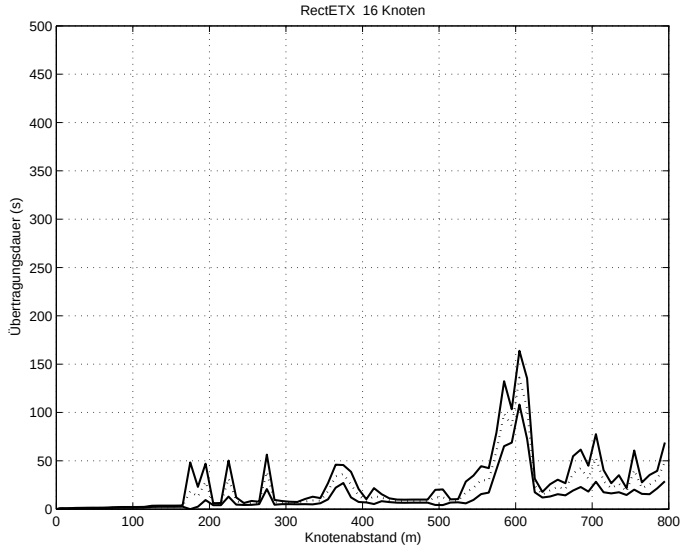


Abbildung A.8: ETX Szenario 2 mit 16 Knoten

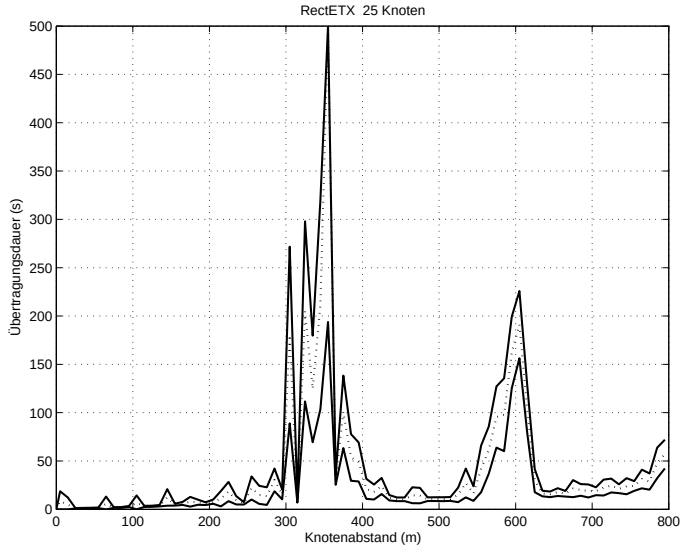


Abbildung A.9: ETX Szenario 2 mit 25 Knoten

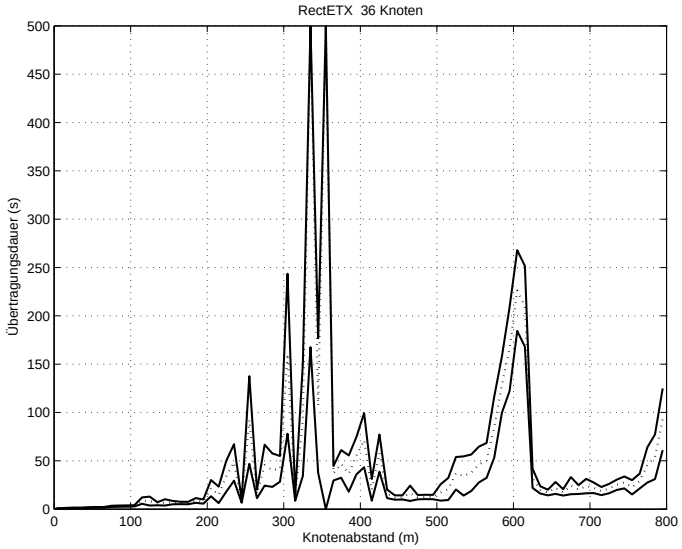


Abbildung A.10: ETX Szenario 2 mit 36 Knoten

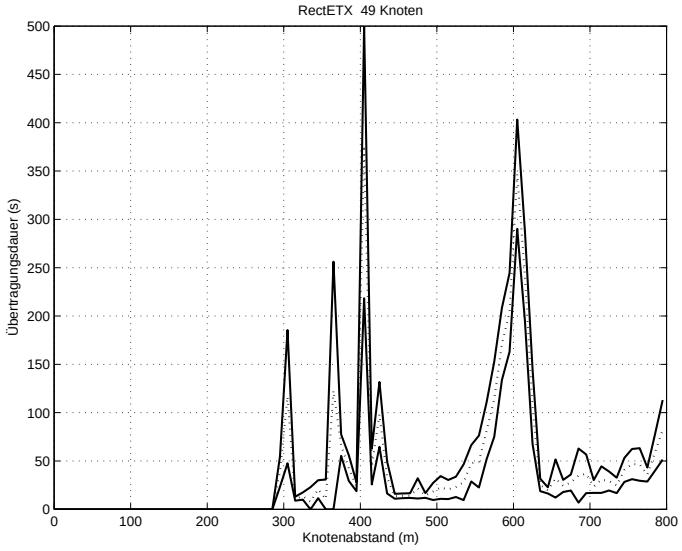


Abbildung A.11: ETX Szenario 2 mit 49 Knoten

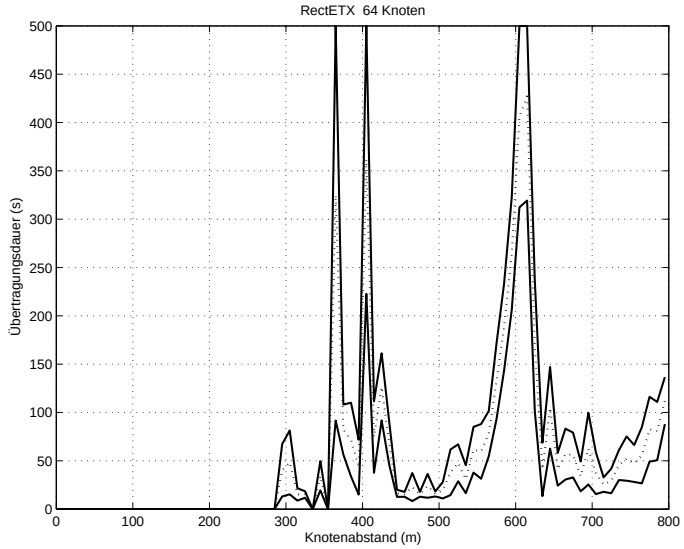


Abbildung A.12: ETX Szenario 2 mit 64 Knoten

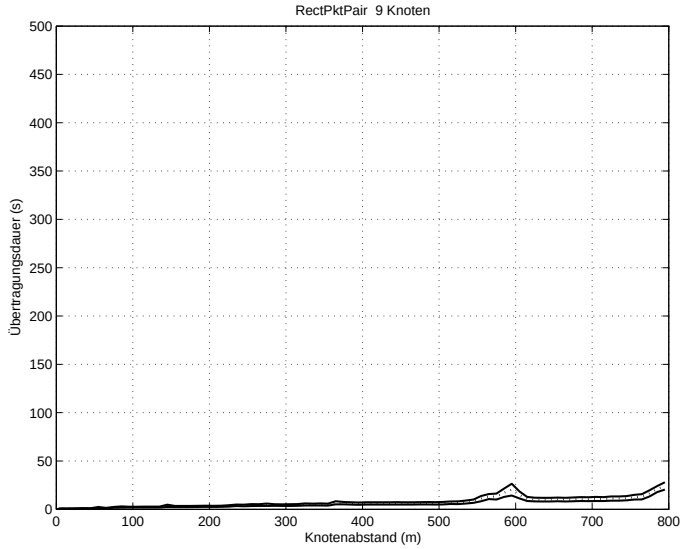


Abbildung A.13: PktPair Szenario 2 mit 9 Knoten

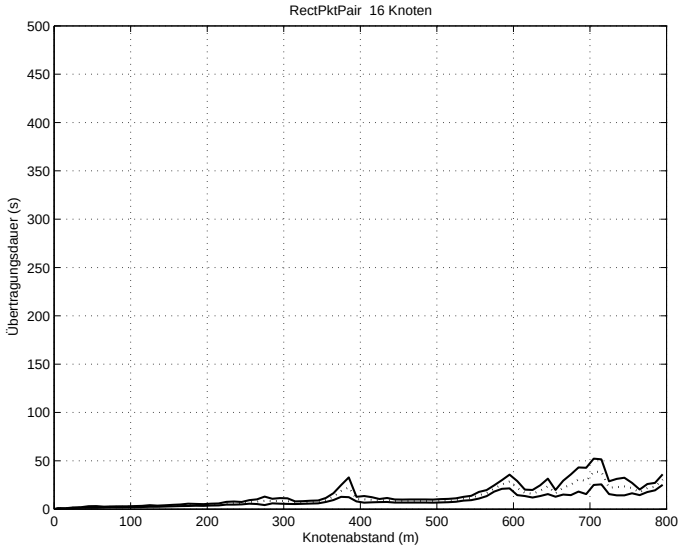


Abbildung A.14: PktPair Szenario 2 mit 16 Knoten

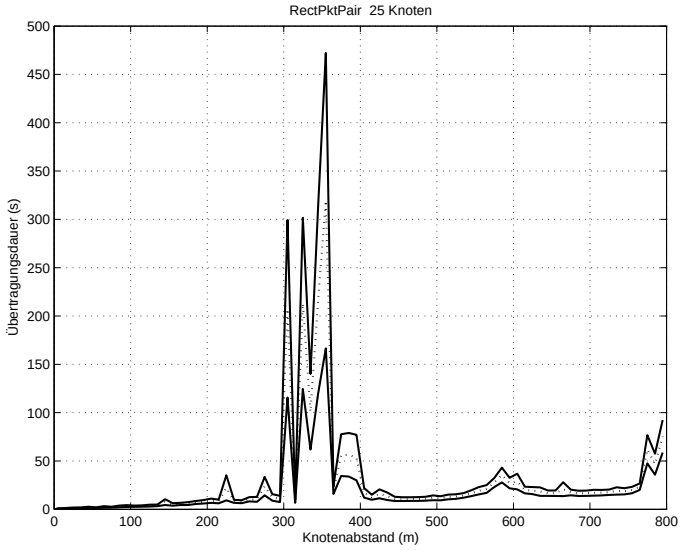


Abbildung A.15: PktPair Szenario 2 mit 25 Knoten

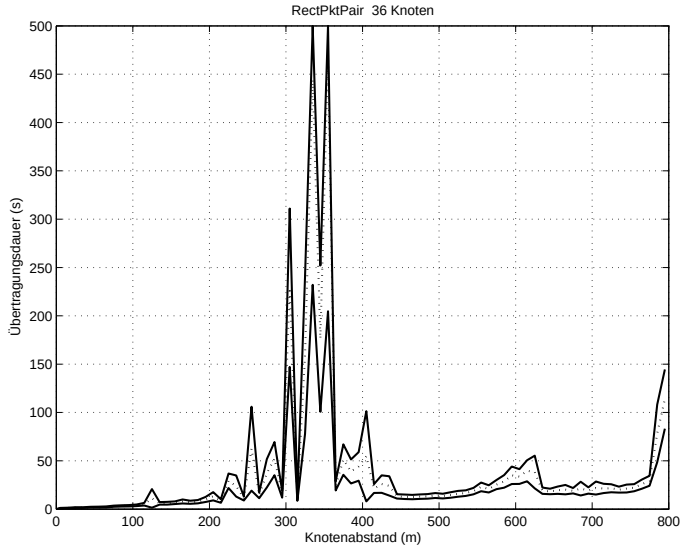


Abbildung A.16: PktPair Szenario 2 mit 36 Knoten

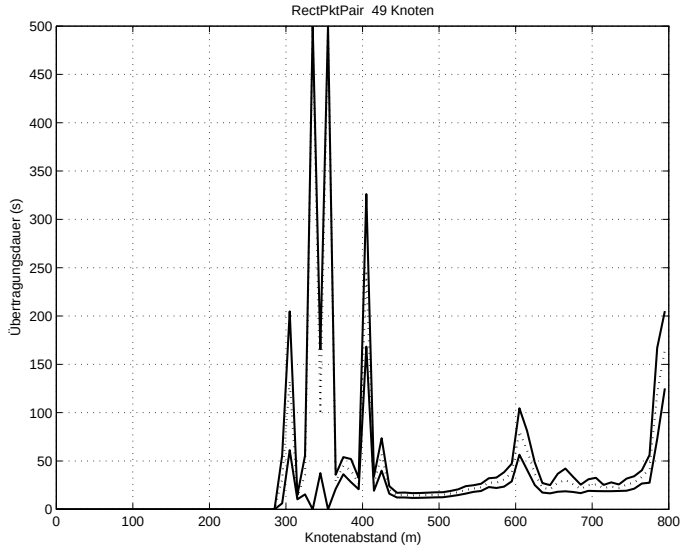


Abbildung A.17: PktPair Szenario 2 mit 49 Knoten

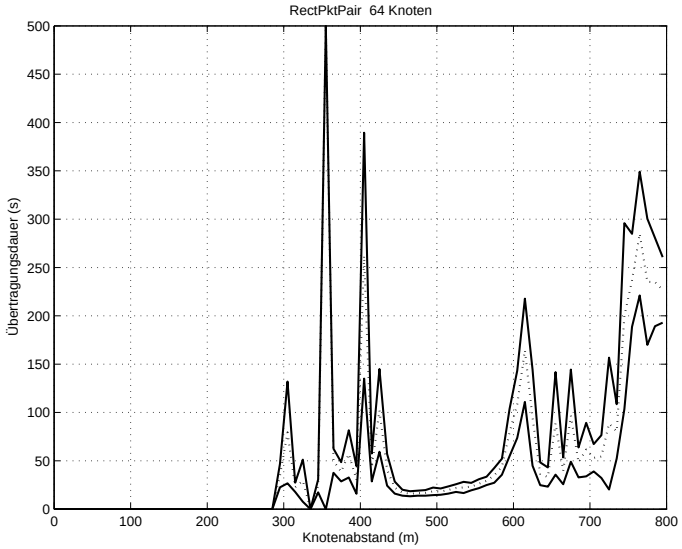


Abbildung A.18: PktPair Szenario 2 mit 64 Knoten

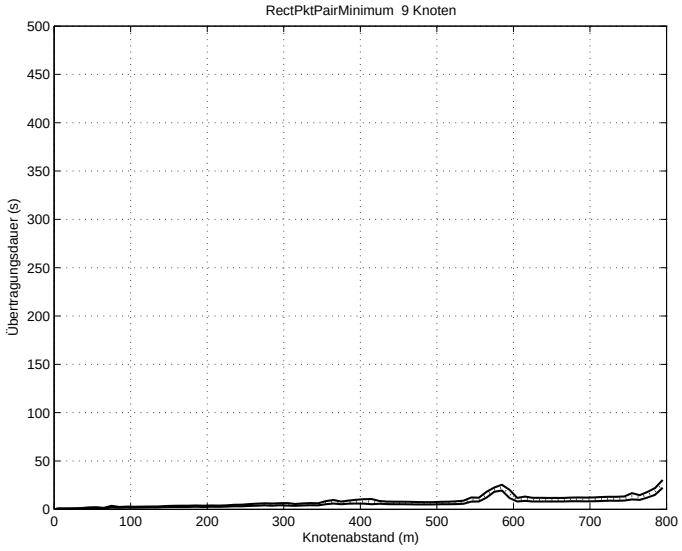


Abbildung A.19: PktPairMinimum Szenario 2 mit 9 Knoten

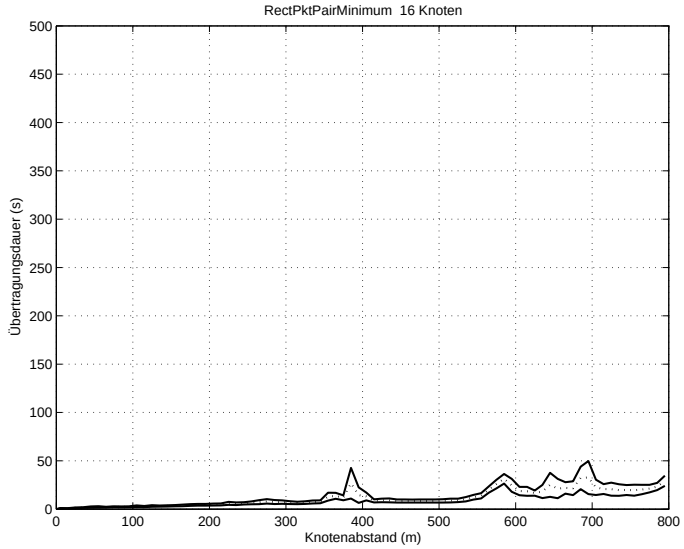


Abbildung A.20: PktPairMinimum Szenario 2 mit 16 Knoten

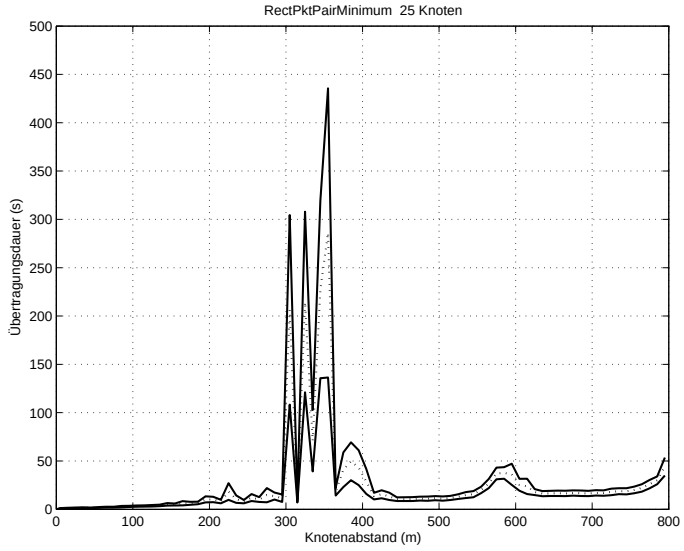


Abbildung A.21: PktPairMinimum Szenario 2 mit 25 Knoten

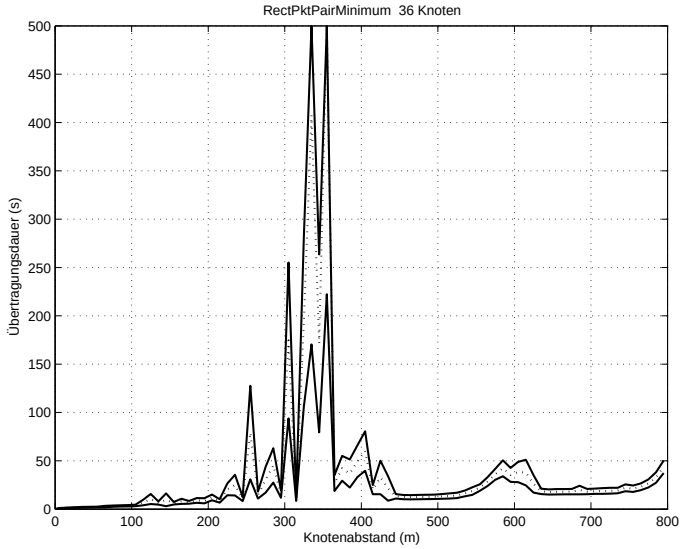


Abbildung A.22: PktPairMinimum Szenario 2 mit 36 Knoten

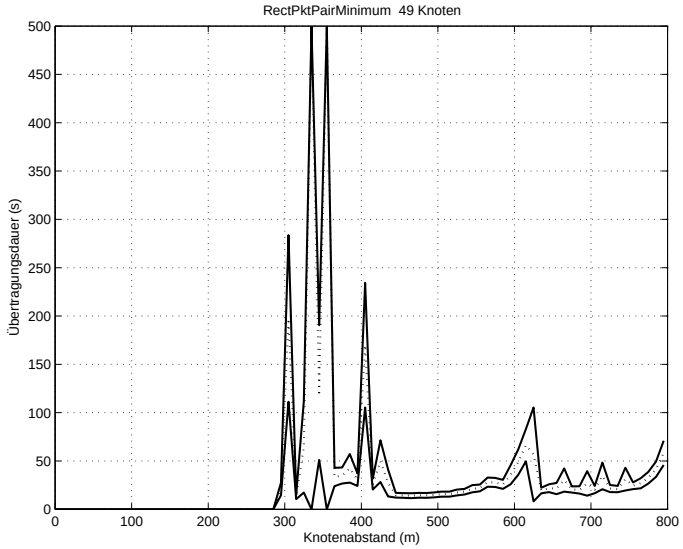


Abbildung A.23: PktPairMinimum Szenario 2 mit 49 Knoten

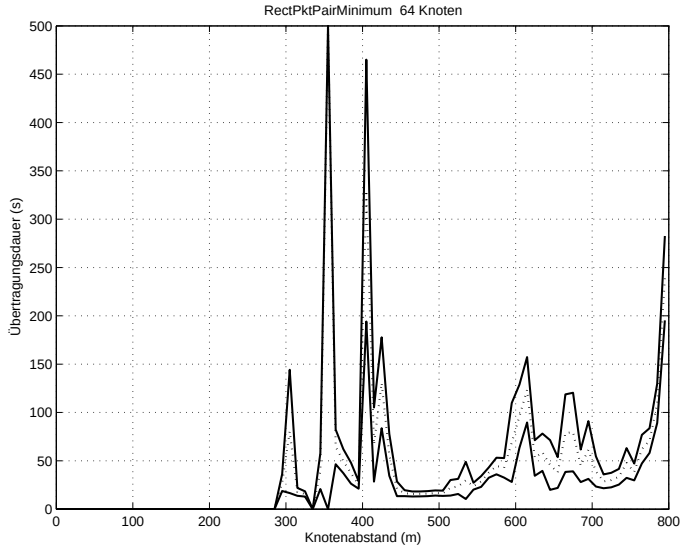


Abbildung A.24: PktPairMinimum Szenario 2 mit 64 Knoten

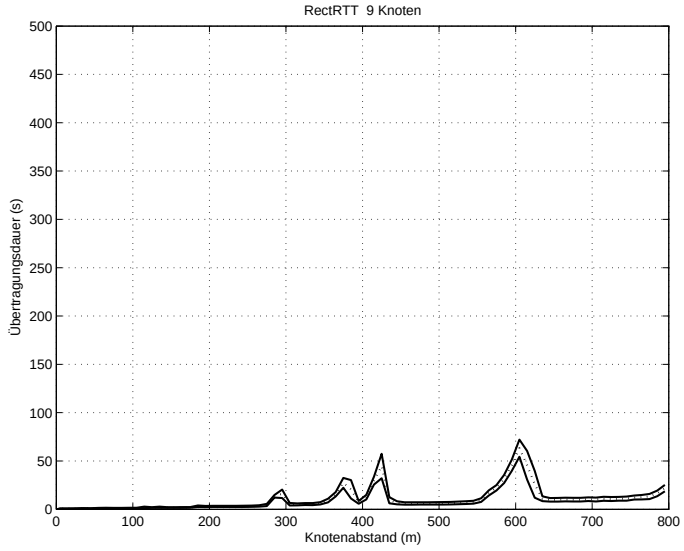


Abbildung A.25: RTT Szenario 2 mit 9 Knoten

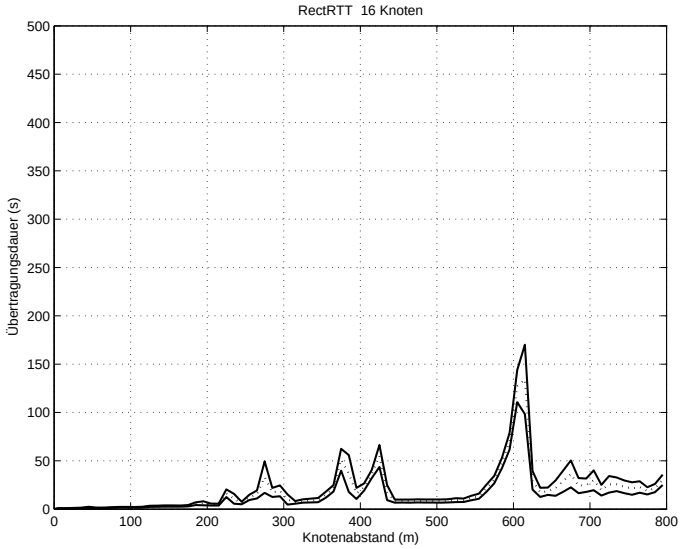


Abbildung A.26: RTT Szenario 2 mit 16 Knoten

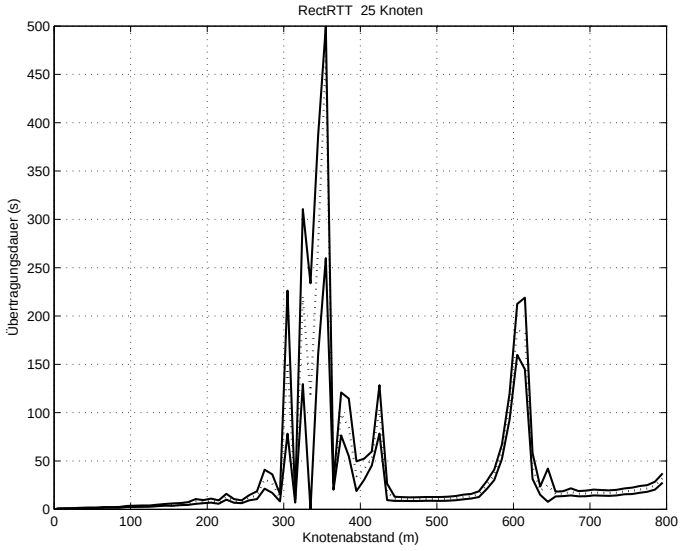


Abbildung A.27: RTT Szenario 2 mit 25 Knoten

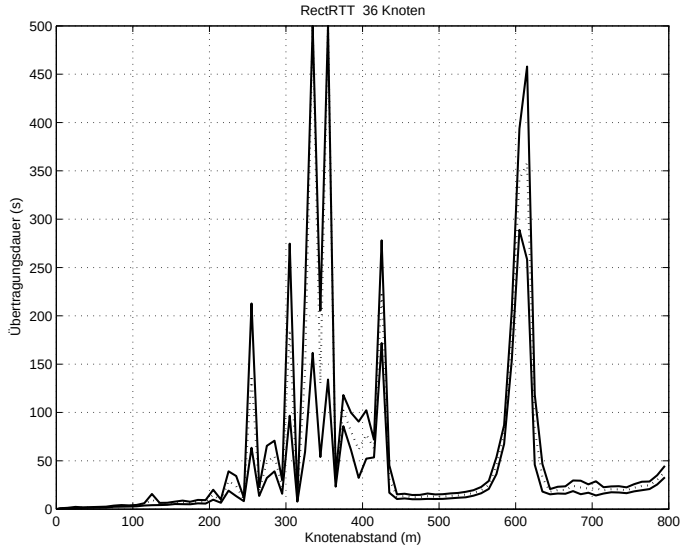


Abbildung A.28: RTT Szenario 2 mit 36 Knoten

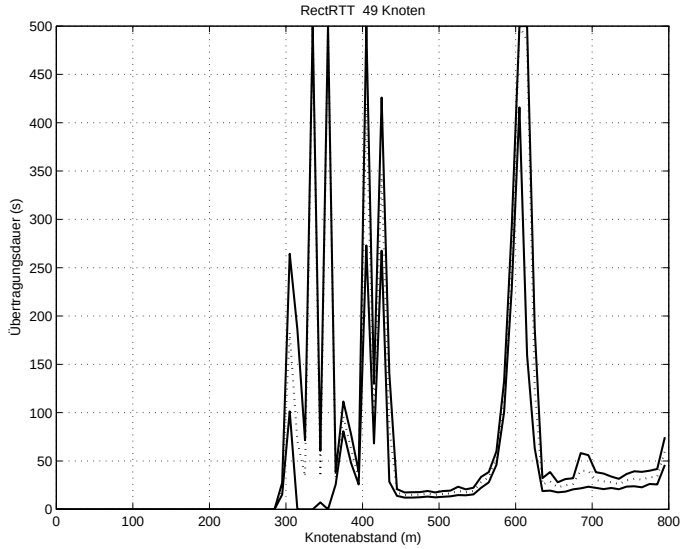


Abbildung A.29: RTT Szenario 2 mit 49 Knoten

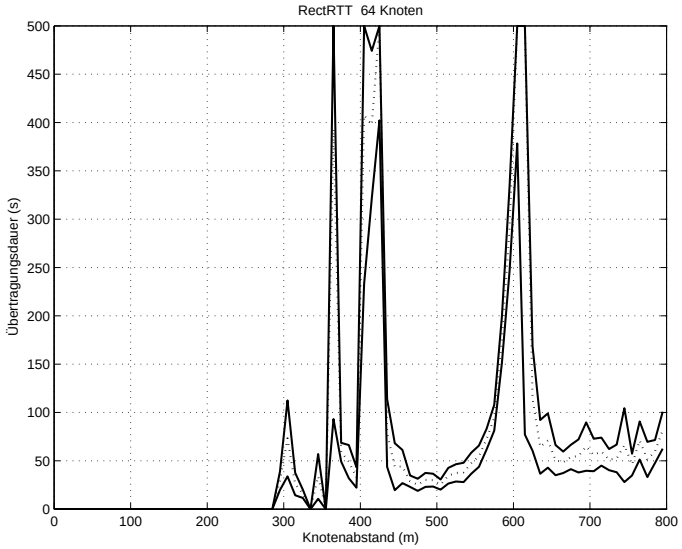


Abbildung A.30: RTT Szenario 2 mit 64 Knoten

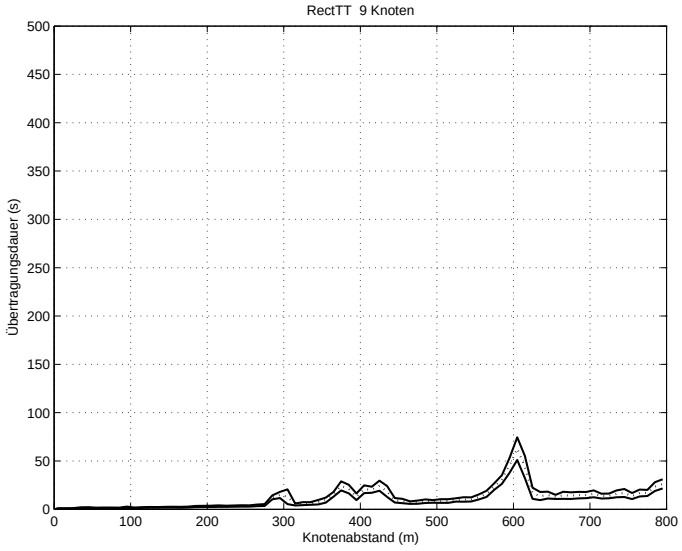


Abbildung A.31: TT Szenario 2 mit 9 Knoten

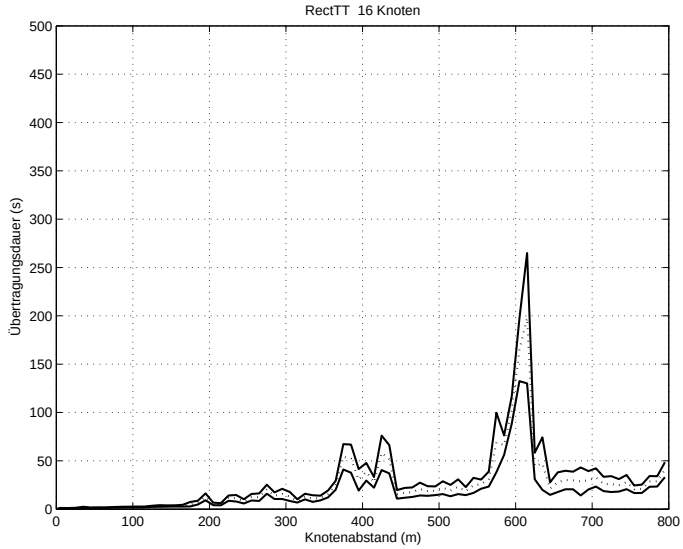


Abbildung A.32: TT Szenario 2 mit 16 Knoten

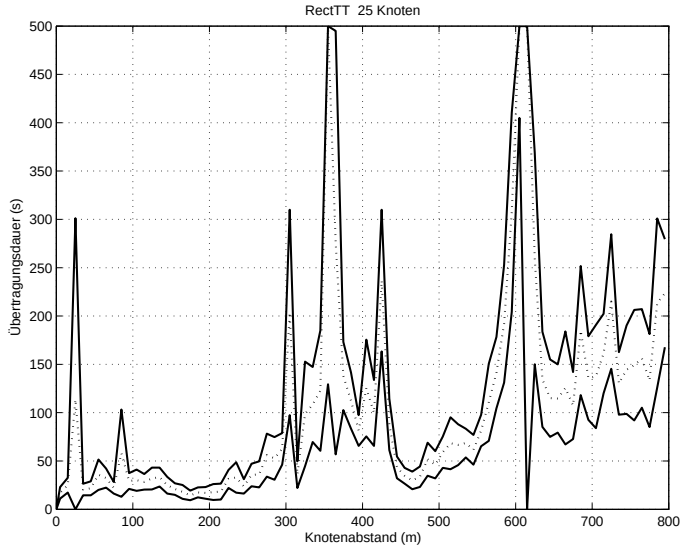


Abbildung A.33: TT Szenario 2 mit 25 Knoten

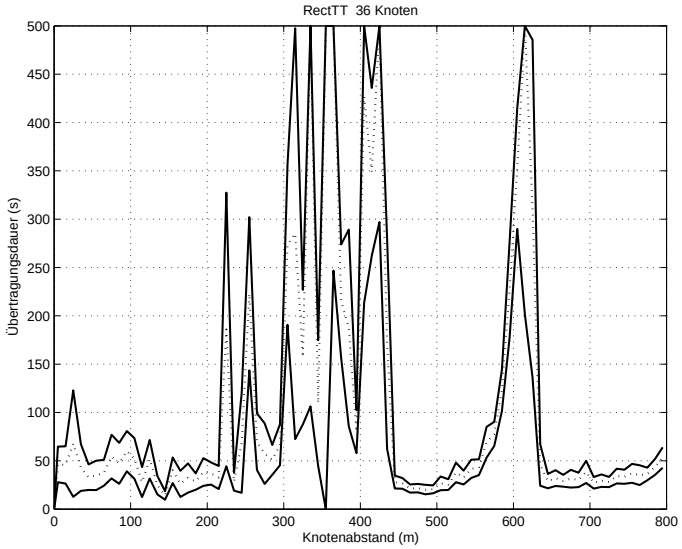


Abbildung A.34: TT Szenario 2 mit 36 Knoten

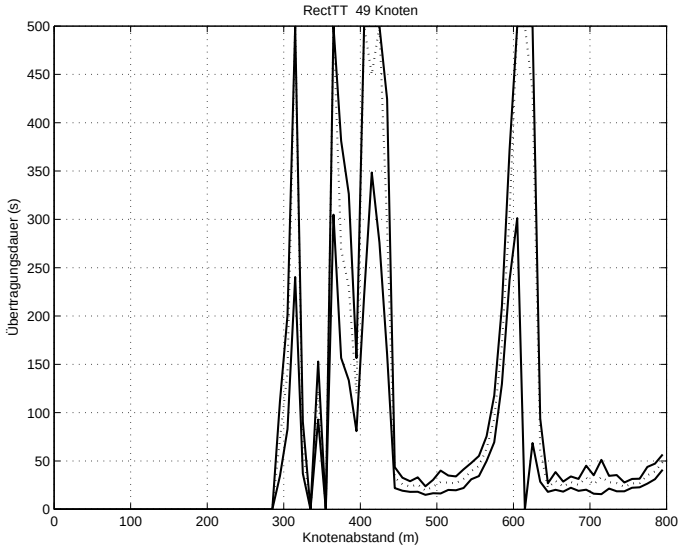


Abbildung A.35: TT Szenario 2 mit 49 Knoten

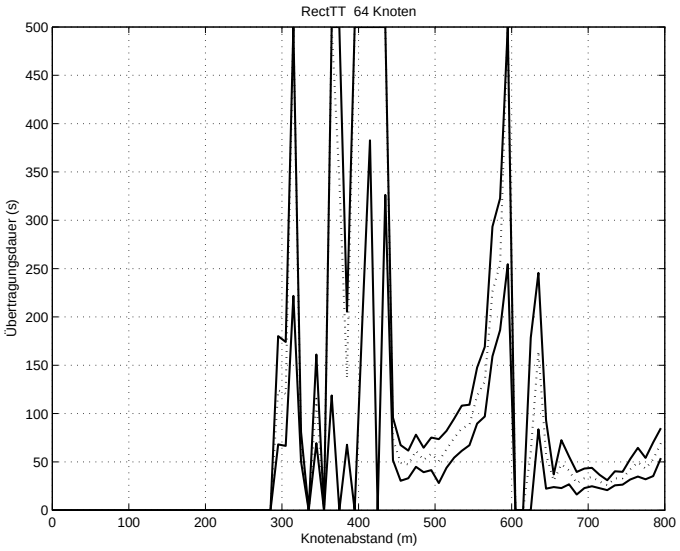


Abbildung A.36: TT Szenario 2 mit 64 Knoten

Literaturverzeichnis

[ABP+04] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. A multi-radio unification protocol for ieee 802.11 wireless networks. *broadnets*, 00:344–354, 2004.

[AHR] Baruch Awerbuch, David Holmer, and Herbert Rubens. Effects of multi-rate in ad hoc wireless networks. `citeseer.ist.psu.edu/738144.html`.

[AHR03] B. Awerbuch, D. Holmer, and H. Rubens. High throughput route selection in multi-rate ad hoc wireless networks. `citeseer.ist.psu.edu/awerbuch04high.html`, 2003.

[CABM03] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. 2003.

[CGR94] Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1994.

[cit00] *Efficient simulation of Ricean fading within a packet simulator*, volume 2, 2000.

[cit01] *Optimized link state routing protocol for ad hoc networks*, 2001.

[CS95] Chih-Che Chou and Kang G. Shin. A distributed table-driven route selection scheme for establishing real-time video channels. In *International Conference on Distributed Computing Systems*, pages 52–59, 1995.

[DPZ04] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 114–128, New York, NY, USA, 2004. ACM Press.

[EABP06] Jakob Eriksson, Sharad Agarwal, Paramvir Bahl, and Jitendra Padhye. Feasibility study of mesh networks for all-wireless offices. In *MobiSys 2006: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 69–82, New York, NY, USA, 2006. ACM Press.

[Fee99] Laura Marie Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report T99–07, 1, 1999.

[fMRR] Researchers from Microsoft Research Redmond. Microsoft mesh connectivity layer (mcl) software. `http://research.microsoft.com/mesh`.

[fre] freifunk.net. freifunk.net. `http://freifunk.net/`.

[Fri45] H. T. Friis. Noise figures af radio receivers. *Proceedings of the IRE*, 33:125–127, 1945.

[Gro] Bay Area Wireless Users Group. Bay area wireless users group. `http://www.bawug.org/`.

[Her] Andre Herms. Ad-hoc wireless distribution service. `http://awds.berlios.de`.

[HM] Andre Herms and Daniel Mahrenholz. Unified development and deployment of network protocols. `citeseer.ist.psu.edu/herms05unified.html`.

[HVB01] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A rate-adaptive mac protocol for multi-hop wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 236–251, New York, NY, USA, 2001. ACM Press.

[IEE] IEEE. Ieee standard for local and metropolitan area networks: Overview and architecture. `http://standards.ieee.org/getieee802/index.html`.

[JM96] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[JPS03] Jangeun Jun, Pushkin Peddabachagari, and Mihail Sichitiu. Theoretical maximum throughput of ieee 802.11 and its applications. In *NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, page 249, Washington, DC, USA, 2003. IEEE Computer Society.

[Kes91] Srinivasan Keshav. A control-theoretic approach to flow control. In *SIGCOMM '91: Proceedings of the conference on Communications architecture & protocols*, pages 3–15, New York, NY, USA, 1991. ACM Press.

[KNG+04] David Kotz, Clvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliot. An effective heuristic algorithm for the travelling-salesman problem. *Experimental Evaluation of Wireless Simulation Assumptions*, 2004.

[KSK04] R. Karrer, A. Sabharwal, and E. Knightly. Enabling large-scale wireless broadband: the case for taps. *SIGCOMM Comput. Commun. Rev.*, 34(1):27–32, 2004.

[LG97] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

[LK73] Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 21:498–516, 1973.

[LlP00] Bruce S. Davie. Larry l. Peterson. *Computernetze*. dpunkt.verlag, 2000.

[MCE01] S. Mangold, S. Choi, and N. Esseling. An error model for radio transmissions of wireless lans at 5ghz. volume 0, pages 209–214, Aachen, Germany, Sep 2001.

[MIT] MIT. MIT roofnet. `http://www.pdos.lcs.mit.edu/roofnet/`.

[ns2] Radio propagation models. `http://www.isi.edu/nsnam/ns/`.

[olp] One laptop per child. `http://www.laptop.org/`.

[PB94] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[Per99] *Ad-hoc on-demand distance vector routing*, 1999.

[PK05] Jun Cheol Park and Sneha Kumar Kasera. Expected data rate: an accurate high-throughput path metric for multi-hop wireless routing. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*. 2005.

[Rap96] T. S. Rappaport. *Wireless communications, principles and practice.* Prentice Hall, 1996.

[RS03] Boris Ribov and Grisha Spasov. Complementary code keying with pic based microcontrollers for the wireless radio communications. In *CompSysTech '03: Proceedings of the 4th international conference conference on Computer systems and technologies*, pages 66–70, New York, NY, USA, 2003. ACM Press.

[SC05] P. Savola and T. Chown. A survey of ipv6 site multihoming proposals. `citeseer.ist.psu.edu/savola05survey.html`, 2005.

[SCK00] K. G. Shin, C.-C. Chou, and S.-K. Kweon. Distributed route selection for establishing real-time channels. *IEEE Transactions on Parallel and Distributed Systems*, 11(3):318–??, 2000.

[SPC03] Y. Seok, J. Park, and Y. Choi. Multirate aware routing protocol for mobile ad hoc networks. 2003.

[Wir] Seattle Wireless. Seattle wireless. `http://www.seattlewireless.net/`.

[YXZL06] Shouyi Yin, Yongqiang Xiong, Qian Zhang, and Xiaokang Lin. Prediction-based routing for real time communications in wireless multi-hop networks. In *QShine '06: Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks*, page 28, New York, NY, USA, 2006. ACM Press.

[Zim88] H. Zimmermann. Osi reference model-the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on [legacy, pre - 1988]*, pages 2–9, 1988.

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit „Vergleich von Routingmetriken in drahtlosen Mesh-Networks“ selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, sowie alle Zitate entsprechend kenntlich gemacht habe.

Magdeburg, 5.10.2007
Jens Wienöbst