Energieeffizientes Routing in Ad-Hoc-Netzen

Eine simulationsbasierte Analyse von OLSR und AODV

Marcel Ebbrecht

Energieeffizientes Routing in Ad-Hoc-Netzen

Eine simulationsbasierte Analyse von OLSR und AODV

Bachelorarbeit Informatik, April 2018 (v2.0)

Vorwort

Sie fragen sich jetzt vielleicht: „Ein Vorwort bei einer Bachelorarbeit?“. Ja, etwas ungewöhnlich. In der ursprünglichen Fassung war es auch nicht enthalten. Ich entschloss mich allerdings, mir ein Exemplar in Buchfassung zu drucken und wollte dem Ganzen einen professionellen Anstrich geben. Warum eigentlich? Ganz einfach: Ich halte diese Arbeit für wichtig. Im Moment des Verfassens dieser Zeilen befindet sich die Arbeit in Prüfung und ich hoffe sehr, dass es als positiv empfunden wird. Natürlich ist auch die Frage berechtigt, warum ich es überhaupt geschrieben habe. Nun, wie jeder Student muss ich eine Abschlussarbeit schreiben. So kam es zum Schreiben. Wie kam es aber zu dem Thema? Das ist, zumindest für mich, ganz einfach: Ich bin Netzwerk-Freak. Seit dem ich denken kann ... oder sagen wir, seit dem ich mit Computern umgehe, immerhin fast 25 Jahre, empfinde ich großen Spaß daran, diese kommunizieren zu lassen. Klar, am Anfang war das eher dadurch begründet, dass auf den LAN-Parties jemand benötigt wurde, der mehr als 2 Rechner vernetzen und solch ein Netz sauber betreiben kann, dennoch war es der Eintritt in meine Leidenschaft Netzwerk. Bis heute bin ich beruflich in diesem Bereich tätig, daher war ich sehr dankbar, dass ich in diesem Bereich meine Arbeit schreiben durfte.

Dies wäre allerdings nicht ohne die Arbeit und Unterstützung vieler Menschen möglich gewesen, daher möchte ich mich ganz herzlich bei den Menschen, die Omnet++ und die Frameworks geschaffen haben, den Teilnehmern der Mailingliste die bereitwillig Fragen beantworten, aber auch bei meinen Freunden und meiner Frau für jegliche Unterstützung bedanken. Und nun viel Spaß beim lesen.

Marcel Ebbrecht, 01.04.2018

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Hintergrund

„Ich denke, dass es einen Weltmarkt
für vielleicht fünf Computer gibt.“

— Thomas Watson, 1943

Ob Herr Watson zum Zeitpunkt seiner Aussage richtig lag, kann ich nicht bewerten. Eins ist allerdings klar: Die Gültigkeitsdauer war begrenzt. Heute sehen wir uns einer großen Anzahl an Systemen gegenüber, die, bedingt durch die rasante Entwicklung des Internets, in hohem Maße miteinander kommunizieren. Obschon sich Dezentralisierung in vielen Bereichen durchgesetzt hat, betreibt die Menschheit heute große Infrastrukturen, die dem verteilten Speichern oder Verarbeiten von Daten dienen. Die Übertragung dieser Daten erfolgt jedoch zu großen Teilen über zentralistische Vernetzung. Natürlich werden dabei Technologien zur Lastverteilung und Ausfallsicherheit eingesetzt, dennoch entstehen, bedingt durch die Topologie, immer wieder Engpässe. Dieses Problem wird, aufgrund stetig steigender Datenvolu-

mina weiter zunehmen. Meines Erachtens nach liegt die Zukunft, da wir mittlerweile über eine hohe Dichte an kommunikationsfähigen Geräten verfügen, in der Übermittlung durch *Vermaschte Netze (MESHs)*. Da sich in solchen Netzen besondere Anforderungen an die *Routingverfahren (RVs)* ergeben [CM99], wurden hierfür eben solche entwickelt, die den Betrieb sicherstellen sollen [BTA+11] [KAASYS13]. Die Aufmerksamkeit soll hier auf ein bestimmtes Thema gerichtet werden: Die Verteilung des Energieverbrauchs. Viele Endgeräte, wie *Smartphones* oder *Wireless Sensor and Actor Networks (WSANs)*, die sich für den Einsatz als *Router* anbieten, besitzen nur einen begrenzten Energievorrat. Bei den von mir untersuchten, gängigen RVs ist die Weglänge, also die Anzahl der *Netzwerkabschnitte (HOPs)*, die maßgebende *Metrik*, mit der die anzuwendende *Route* bestimmt wird. Wenn man an moderne Smartphones denkt, welche mit einer Akkuladung in der Regel 8 bis 12 Stunden ihren normalen Betrieb verrichten können, dann stellt sich dem Benutzer natürlich die Frage, ob er auch noch Batterieleistung für das Weiterleiten fremder Daten erübrigen möchte, zumal die Nutzung von drahtloser Übertragung einen großen Anteil des Energieverbrauchs dieser Systeme ausmacht [TET16][CH10][HQG+12]. Zudem würde eine gute Einschätzung seiner Verbindung durch *Ad-hoc On-demand Distance Vector (AODV)* oder *Optimized Link State Routing (OLSR)* dazu führen, dass der Akku noch schneller entladen wird. Es gibt viele Arbeiten, die sich mit der Optimierung des Energieverbrauchs befassen. Hier eine bescheidene Auswahl möglicher Anpassungen:

- Die Arbeiten befassen sich mit der allgemeinen Senkung des Verbrauchs durch Anpassung des Sendeverhaltens der Router [BTL13] [SWR98],
- Wegfindung auf Basis der Art der Energieversorgung [ALF04]
- oder anderen, grundlegenderen Änderungen, z. B. durch geschickte Cliquenbildung bei redundanten Wegen [LZL11].

Bei der Recherche zu diesem Thema waren jedoch keine Arbeiten auszumachen, die den Aspekt der *Fairness im Verbrauch* behandeln, also den Ladezustand der Teilnehmer in die Metrik einbeziehen. Wäre es nicht viel sinnvoller, wenn dies ebenfalls in die Bewertung der Route einfließt? Ein Teilnehmer mit hohem Ladestand wird bevorzugt als Router verwendet, einer mit weniger nur, wenn es keine Alternative gibt.

1.2 Zielsetzung

Die Umsetzung der genannten Punkte sollte bei mobilen Endgeräten wie Smartphones zu einer höheren Akzeptanz von *Wireless Mesh Network (WMN)*, bei stationären Einrichtungen, wie WSAN, zu einer längeren Betriebszeit bis zum nächsten Batteriewechsel einzelner Stationen führen. Im Rahmen dieser Arbeit wird gezeigt, dass es mit minimalen Anpassungen möglich ist, die Präferenzen bei der Wahl der Route nach dem Ladezustand auszurichten. Hierbei wird die Kompatibilität zu den nicht angepassten Systemen beibehalten. Um die getätigten Annahmen überprüfen zu können, wurde die *Simulationssoftware Omnet++*[1] ausgewählt, da diese zusammen mit den frei verfügbaren Frameworks *INET*[2] und *INETManet*[3] gute Möglichkeiten für die Umsetzung mittels AODV und OLSR bereitstellt und die umfassende Analyse der Ergebnisse ermöglicht. Der gesamte Code, der im Rahmen dieser Arbeit erstellt wurde, steht unter der GPL auf GitHub zur Verfügung.[4] Zudem steht dort die gesamte Arbeit als PDF und die darin verwendeten Abbildungen und Diagramme im Ordner *book* zur Verfügung.

[1] https://www.omnetpp.org
[2] https://inet.omnetpp.org
[3] https://github.com/aarizaq/inetmanet-3.x
[4] https://github.com/marcelebbrecht/powerrouting

1.3 Aufbau dieser Arbeit

Da sich diese Arbeit im Schwerpunkt mit WMNs beschäftigt, werden im folgenden Kapitel die beteiligten Technologien wie das *Internet-Protocol (IP)*, *Wireless LAN (WLAN)*, Routingverfahren, Metriken und ein paar andere Aspekte erläutert, um damit die Grundlage für das weitere Verständnis zu legen. In Kapitel 3 werden die untersuchten Routingverfahren näher erklärt, wobei sich die Beschreibung auf die in den entsprechenden *Requests for Comments (RFCs)* genannten Informationen und ein paar zusätzliche Arbeiten stützt. Anschließend werden die vorgenommenen Anpassungen und deren angedachte Wirkung beschrieben. Ferner wird das Thema Metrik eingehender behandelt und der Versuch unternommen, eine eigene Definition zur Bewertung der jeweiligen Konfiguration zu schaffen. Kapitel 4 widmet sich den durchgeführten Versuchen und behandelt die eingesetzte Software, den Aufbau und die Durchführung der Simulationen. Hierbei wird näher auf die Implementierung der betrachteten Verfahren AODV und OLSR in Omnet++ bzw. im INETManet Framework eingegangen. Kapitel 5 befasst sich umfangreich mit der Auswertung der im Versuch gesammelten Daten und stellt die Wirkung der Anpassungen ausführlich dar. Zudem wird auf die Stärken, aber auch auf die Schwächen eingegangen. Den Schluss bildet ein Resümee sowie ein Ausblick auf weitere Möglichkeiten und Aspekte der Verwendung von WMNs.

1.4 Anmerkung zu Fachbegriffen

Die vielfältig verfügbare Literatur ist größtenteils in englisch verfasst, die meisten Fachbegriffe sind nur in englisch definiert und vor allem den meisten potentiellen Lesern auch nur in dieser Sprache geläufig. Diese Arbeit ist in deutsch verfasst, enthält jedoch viele dieser englischen Begriffe, da es meist nicht möglich ist, diese zu übersetzen, ohne das

Informationen verloren gehen. Wenn es sich also anbot, dann wurden manche Begriffe übersetzt, vor allem dann, wenn sie so auch in Literatur auftauchten, ansonsten wurde das Original verwendet. Um Klarheit zu schaffen, sind viele Begriffe im Glossar erklärt. Aufgrund der hohen Anzahl an Fachbegriffen und deren Länge werden viele Abkürzungen eingeführt um den Text überschaubar zu halten. Die Akronyme und die Bedeutung einzelner Begriffe werden im Anhang dieser Arbeit aufgelistet und erläutert. Grundsätzlich werden die Fach- oder andere zentrale Begriffe, sofern sie erstmalig erwähnt werden, *kursiv* gesetzt, bei der Verwendung der Akronyme oder häufigen Wiederholungen wurde allerdings zu Gunsten der Optik darauf verzichtet. Das Glossar ist nicht Bestandteil der Leistung dieser Arbeit, es wird daher auf Verweise und Belege verzichtet.

Kapitel 2

Grundlagen

In diesem Kapitel werden die technologischen Grundlagen beschrieben. Die Darstellungen und Erläuterungen beschränken sich auf die für das Routing in Wireless Mesh Networks benötigten Teile.

2.1 Wireless Mesh Networks

Im Oxford Dictionary[1] wird ein *Mesh* folgendermaßen beschrieben:

> „A computer network in which each computer or processor is connected to a number of others, especially so as to form a multidimensional lattice."

Es handelt sich um ein multidimensionales Gitter aus Computern, die miteinander kommunizieren. Werden diesen Verbindungen mittels *Ad-Hoc-Modus* realisiert und existieren eigene Mechanismen für eine dezentrale Verwaltung, nennt man es *mobiles Ad-Hoc Netzwerk (MANET)*. Die grundlegenden Eigenschaften solcher Netze sind in RFC 2501 [CM99] ausführlich dokumentiert. Die Idee hinter MANETs lässt

[1] https://en.oxforddictionaries.com/definition/mesh

sich auf die Forschung im Bereich *Mobile Packet Radio Networking* der 70er und 80er Jahre zurückführen. Die Autoren der RFC sehen sie als Alternative oder Ergänzung für zellenbasierte Netze [CM99]. Die Anwendungsmöglichkeiten sind sehr vielfältig und die Autoren der RFC gehen weiter auf ein paar wichtige Eigenschaften ein [CM99]:

1. **Dynamische Topologien**: Die Teilnehmer bewegen sich, die Topologie kann sich kurzfristig ändern und Verbindungen müssen nicht zwingend bidirektional sein. Ein geeignetes RV muss also mit einer hohen Rate an Veränderungen zurechtkommen.

2. **Bandbreitenbeschränkung**: Die Bandbreite ist bei drahtloser Kommunikation spürbar eingeschränkt, es muss effizient damit umgegangen werden.

3. **Energiebeschränkung**: Es steht nur eine begrenzte Menge an Energie zur Verfügung, der Verbrauch muss optimiert werden. Zudem stellt sich die Frage nach der *Fairness der Verteilung der Arbeitslast* bei mehreren Teilnehmern, ein Aspekt den diese Arbeit näher beleuchten wird.

4. **Sicherheitsbeschränkung**: Funknetze sind vielfältigen Sicherheitsproblemen ausgesetzt. Im MANET stellt dies eine weitere Herausforderung dar.

Ferner nennen die Autoren folgende qualitativen Anforderungen, wie **verteilte Funktion**, **Schleifenfreiheit**, **anforderungsbasiertes** oder **Proaktives Routing**, **Sicherheit**, sowie die Unterstützung eines **Schlafmodus** und **unidirektionaler Verbindungen**. Leider gibt es derzeit kein natives Verfahren für *IEEE 802.11* auf der *Sicherungsschicht*, somit muss die Funktion des *Routings* in der *Netzwerkschicht* übernommen werden.

2.2 Schichtenmodelle

Die Grundlage der Kommunikation in modernen Netzen bildet das *OSI Schichtenmodell* von Hubert Zimmermann [Zim80] wie es, nebst der Implementation des TCP/IP Protocol-Stack (TCP/IP), in Abbildung 2.1 zu sehen ist. Die eigentliche Übertragung der Daten zwischen den Teilnehmern findet ausschließlich auf der *Bitübertragungsschicht* in Form von *Frames* statt. Darüber liegt die Sicherungsschicht, welche für den Zugriff auf das Medium verantwortlich ist. Beide OSI Schichten sind im TCP/IP Modell zur *Netzwerkzugriffsschicht* zusammengefasst und werden in der Regel durch eine Technologie, wie in unserem Fall durch IEEE 802.11 zur Verfügung gestellt. Die *Vermittlungsschicht* im OSI-Modell entspricht der *Internetschicht* bei TCP/IP. Sie übernimmt die Adressierung der *IP-Pakete* und Steuerfunktionen. Das IP ermöglicht durch seine Adressierung die *Segmentierung* von Netzen und eine Übermittlung von Paketen über die Grenzen von *Subnetzen* hinaus. Die *Transportschicht* wird beim Einsatz von IP durch *TCP/IP* realisiert. Es handelt sich hierbei um eine Sammlung verschiedenster Protokolle, die beiden meist eingesetzten sind das *Transmission Control Protocol (TCP)* und das *User Datagram Protocol (UDP)*. Die Aufgabe dieser Schicht besteht darin, die *Datagramme* den korrekten *Prozessen* auf den Hosts zuzuweisen, was bei TCP und UDP über *Ports* geschieht. Den Abschluss im TCP/IP Modell bildet die *Anwendungsschicht*, welche die verbleibenden Schichten aus dem OSI-Modell vereint. Hier werden dann Protokolle wie z. B. HTTP und FTP, aber auch OLSR und AODV realisiert.

Die Übertragung findet zwischen den Teilnehmern ausschließlich auf der untersten Schicht statt. Jede der o. g. Schichten verfügt über eigene *Header*. Möchte eine Anwendung auf Teilnehmer Alice eine Nachricht an die Anwendung bei Teilnehmer Bob schicken, dann muss

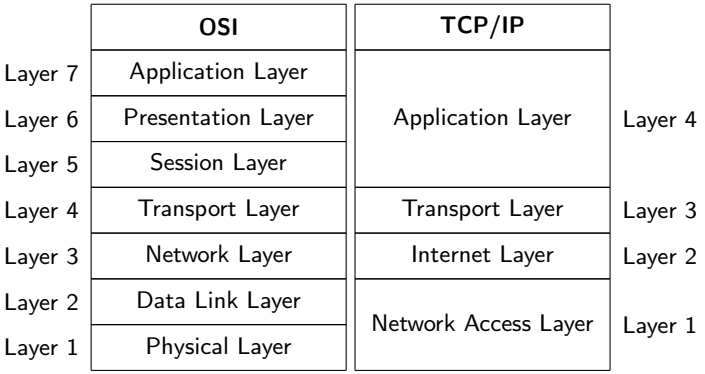

Abbildung 2.1: OSI [Zim80] & TCP/IP Modell [SK91]

diese Nachricht alle Schichten beider Teilnehmern passieren. Hierbei fügt jede Schicht auf dem System von Alice bei Weiterreichung an eine tiefere eigene Header hinzu, die nach der Übertragung zu Bob beim Weiterreichen an eine höhere wieder entfernt werden. Ein schöner Vergleich ist eine Zwiebel: Die Daten der eigenen Anwendung bilden den Kern, die Header der einzelnen Schichten die Schalen. Die Zwiebel ist das, was dann übertragen, der Kern das, was von den Anwendungen genutzt wird. Die Schalen sind Abfall, auch *Overhead* genannt.

2.3 Wireless-LAN

WLAN nach IEEE802.11 implementiert die *Netzwerkzugangsschicht*. Nach der Einführung des Standards wurde dieser sukkzessive um zahlreiche Subtypen erweitert, um den steigenden Anforderungen gerecht zu werden. Eine Übersicht bis 2013 findet sich im Artikel „On IEEE 802.11: Wireless Lan Technology“ [BSC13]. Man kann die Topologien in den **Infrastruktur-Modus** und den **Ad-Hoc-Modus** unterteilen [CWKS97]. Die Einheiten innerhalb eines Funktnetzes

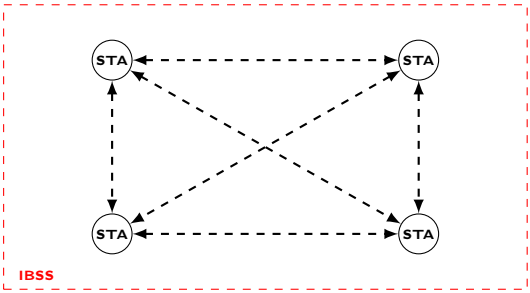

Abbildung 2.2: Ein IBSS im Ad-Hoc-Modus

werden als *Basic Service Sets (BSSs)* bezeichnet. Beim *Infrastruktur-Modus* werden mehrere Stationen zusammen mit einem AccessPoint (AP) zu einem BSS zusammengefasst. Sind mehrere APs über ein Distribution System (DS) miteinander verbunden, bilden diese ein *Extended Service Set (ESS)*. Hier werden dann die Frames auf der *Sicherungsschicht* zwischen den BSSs oder aber auch über ein *Portal* mit anderen *Netzwerksegmenten* ausgetauscht. Eine wichtige Eigenschaft des DS ist die Unabhängigkeit der Implementierung, so können zur Vermittlung zwischen den BSSs andere Technologien, wie *Ethernet*, *Token Ring* oder weitere drahtlose Netze eingesetzt werden [CWKS97].

2.3.1 Ad-Hoc-Modus

Im Ad-Hoc-Modus existiert nur ein einziges BSS, das als *Independent Basic Service Set (IBSS)* bezeichnet wird. Die Teilnehmer kommunizieren nur innerhalb des IBSS, eine Weiterleitung von Nachrichten der Sicherungsschicht an Teilnehmer außerhalb des IBSS findet nicht statt [CWKS97][BCGS04]. Um zwischen verschiedenen IBSSs vermitteln zu können, wird ein RV benötigt. Diese Arbeit beschäftigt sich mit Verfahren, die auf der Vermittlungsschicht basieren und IP einsetzen. Dennoch gibt es Ansätze dies auch auf der Sicherungsschicht

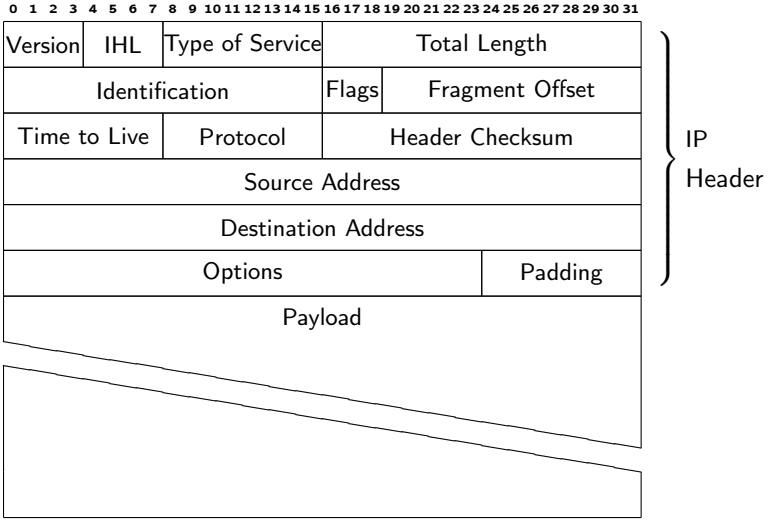

Abbildung 2.3: Ein IP-Paket nach RFC 791 [Pos81]

zu realisieren, wie z. B. *B.A.T.M.A.N* [JNA08] oder der experimentelle Standard *IEEE 802.11s*.

2.4 Das IPv4 Protokoll

Die im weiteren Verlauf dieser Arbeit behandelten Routingverfahren werden auf der Vermittlungsschicht mittels IP realisiert. Diese verwenden in erster Linie *Unicast-* und *Broadcastverkehr*. Es wird zwar auch *Multicastverkehr* von OLSR und AODV unterstützt, dies ist jedoch für diese Arbeit nicht relevant. Ferner sind beide Verfahren für den Betrieb mit *Internet-Protocol V4 (IPv4)* als auch *Internet-Protocol V6 (IPv6)* [PBRD00][CJ03] geeignet, dennoch beschränkt sich die weitere Betrachtung auf IPv4. Gemäß RFC 791 [Pos81] handelt es sich bei IP um ein verbindungsloses Protokoll und stellt die erste vom Übertragungsmedium unabhängige Schicht der Internetprotokollfamilie dar.

Mittels *IP-Adressen* und *Netzwerkmasken*, welche bei IPv4 je aus 32 Bit bestehen, wird die Adressierung von Nachrichten und die Segmentierung von Netzabschnitten sichergestellt. Diese Aufteilung in *Subnetze* liefert die Möglichkeit größere Strukturen hierachisch aufzubauen, macht aber Routing erforderlich, sollen *IP-Pakete* zwischen einzelnen *Segmenten* ausgetauscht werden. Innerhalb eines Subnetzes gibt es, neben den für die Teilnehmer verfügbaren Adressen (*Unicast Addresses (UCAs)*), immer eine Adresse, die das Subnetz an sich bezeichnet (*Network Addresses (NWAs)*) und eine weitere, die als Zieladresse für Broadcasts genutzt wird (*Broadcast Addresses (BCAs)*), damit die Teilnehmer diese identifizieren können. Der Aufbau kann der Abbildung 2.3 entnommen werden. Die genutzten RVs setzen in unserem Fall nur *Unicast* und *Broadcast*, die genauen Funktionen sind in RFC791, RFC1812 und RFC826 beschrieben. Bei Broadcast sei erwähnt, dass zwischen **Limited Broadcast** und **Directed Broadcast** unterschieden wird. Letzterer wird von Routern weitergeleitet, sofern die TTL größer 1 ist.

2.5 Der TCP/IP Protokollstapel

TCP/IP besteht aus einer Sammlung verschiedenster Protokolle, jedes für einen spezifischen Einsatzzweck mit individuellen Vor- und Nachteilen: TCP z. B. ist ein verbindungsorientiertes Protokoll, besitzt einen zusätzlichen Kontrollmechanismus für die Sicherstellung der Übertragung von Informationen, erfordert aber dafür eine Bestätigung des Empfangs, was zu zusätzlichem Overhead, Verzögerungen und mitunter zu einem Einbruch der Übertragungsraten führen kann. UDP hingegen, das verbindungslos arbeitet, stellt zwar die Übertragung nicht durch eigene Mechanismen sicher, kann aber aufgrund des des akzeptierten Paketverlustes höhere Raten und kürzere Verzögerungen erreichen.

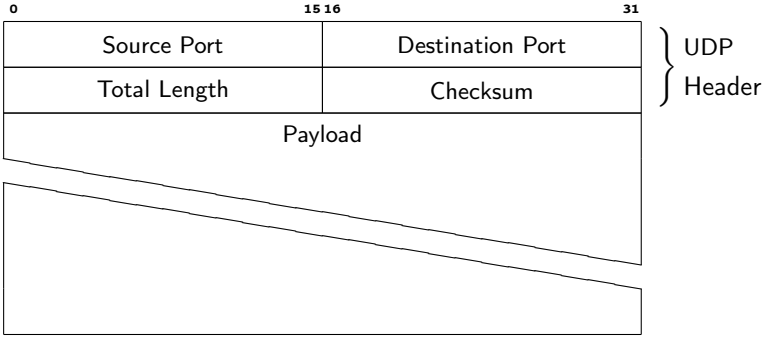

Abbildung 2.4: Ein UDP-Datagramm nach RFC 1122 [Bra89]

2.6 IP-Routing

Wie im Abschnitt 2.1 erwähnt, sollte ein geeignetes Routingverfahren auf der Vermittlungsschicht angesiedelt sein. Im Grunde bezeichnet *Router* ['raʊtə] einen Teilnehmer, der Pakete für andere Teilnehmer weiterleiten kann, sofern es über eine TTL größer als 1 verfügt und ihm eine mögliche *Route* bekannt ist. Hierzu nutzt er, falls vorhanden, die in seiner *Routingtabelle* vorliegenden Informationen über einen möglichen Weg zum Ziel. Das kann wieder ein Router sein oder das eigentliche Ziel, man spricht dann vom *nextHop*. Der Router verringert beim Weiterleiten des IP-Paketes die TTL um 1. Somit wird vermieden, dass es endlos zirkuliert. Bei der Art des Routings kann man grundsätzlich in 2 Klassen unterscheiden: statisch und dynamisch.

2.6.1 Statisches Routing

Beim *statischen Routing* werden die Einträge in der Routingtabelle auf jedem beteiligten Router manuell gepflegt. In sehr kleinen Netzwerken in denen es keine, oder nur sehr wenig Veränderungen gibt kann dies sinnvoll sein. Der Standardfall für eine solche Anwendung sind Heim-

netze, in denen in der Regel nur die Standardroute über den heimischen Router existiert.

2.6.2 Dynamisches Routing

Dynamisches Routing kommt immer dann zum Einsatz, wenn die manuelle Pflege der Routinginformationen nicht praktikabel erscheint. Dies kann im Aufwand begründet sein, aber auch, wie im Fall von MANET ein immanenter Bestandteil der Netzwerktopologie darstellen. Das Funktionsprinzip ist, unanhängig von der Klasse, meistens identisch: Auf den Routern läuft ein Prozess, der über verschiedene Mechanismen versucht, die Prozesse anderer Router zu finden, um mit diesen Informationen über die vorliegenden Routen auszutauschen. Bei den dynamischen Routingverfahren unterscheidet man in mehrere Klassen, eine umfassende Darstellung der Funktionsweise aller genannten Klassen und verschiedenster Implementationen wird in der Arbeit „Routing protocols in ad hoc networks: A survey“[BTA+11] vorgenommen, im Folgenden wird nur auf die Klassen der untersuchten Protokolle eingegangen.

Anforderungsbasiertes, reaktives Routing (Source-initiated)

In diese Klasse sind Verfahren einzuordnen, die nur dann eine Route ermitteln, wenn sie wirklich gebraucht wird. Dies kann bei erstmaliger Verwendung oder dem Ausfall einer Route zu einer hohen Latenz führen, da der gesamte Pfad erst ermittelt werden muss. Wenn die Menge der neu benötigten Routen überschaubar ist, bietet dieses Verfahren durch, im Vergleich zu proaktiven, *geringeren Protokolloverhead* und damit verbundenen Stromverbrauch Vorteile. Dieser Klasse ist auch das im weiteren Verlauf untersuchte AODV zuzuordnen [PBRD03]. Ein beliebtes Anwendungsfeld für solche Verfahren sind Netze, in denen der Verkehr in der Regel eine sehr begrenzte Anzahl an Zielen hat, z. B. bei

Streaminganwendungen oder bei Wireless Sensor and Actor Networks [BTL13].

Proaktives Routing (Table-driven)

Hierbei handelt es sich um Routingverfahren, die versuchen alle verfügbaren zu ermitteln und zu pflegen. Nach einer Vorlaufzeit sind jedem Router in einem solchen Netz die Routen zu beliebigen Zielen bekannt und Pakete können ohne Verzögerung weitergeleitet werden. Dies senkt spürbar die Latenz bei der erstmaligen Verwendung neuer oder ersetzter Router, geschieht allerdings auf Kosten des *höheren Overheads* und *benötigter Energie*. Selbst wenn keinerlei Daten in einem solchen Netz übertragen werden, wird dennoch Strom und Bandbreite verbraucht. Zudem kann es in größeren Netzen einen nicht unerheblichen *Bedarf an Speicher und Rechenzeit* beanspruchen. OLSR ist dieser Klasse zuzuordnen [CJ03] und findet auch, z. B. im Rahmen des Freifunk-Projektes[2] breite Anwendung.

Distanzvektorprotokolle

Eine weitere, wenn auch schwächere Unterscheidung kann man in *Distanzvektor*, *Linkstatus* und *Pfadvektor* basierte Protokolle vornehmen. Erstere zeichnen sich dadurch aus, dass sie in der Regel nur die Informationen verwenden, die sie von ihren direkten Nachbarn mitgeteilt bekommen. Sie haben in der Regel keine Übersicht über die gesamte Topologie. Zur Berechnung der Kosten einer Route kommt ein Distanzvektoralgorithmus zum Einsatz. Ein prominenter, wenn auch etwas älterer Vertreter dieser Klasse ist das *Routing Information Protocol*, ein *Interior Gateway Protocol* [Hed88] oder Ad-hoc On-demand Distance Vector [PBRD03]. Bei älteren Vertretern dieser Klasse kann es allerdings zur *Bildung von Routingschleifen* kommen.

[2] https://wiki.freifunk.net

Pfadvektorprotokolle

Es handelt sich hierbei um eine angepasste Version der Distanzvektorprotokolle. Hier wird der *Pfad*, den ein Kontrollpaket durch ein Netzwerk genommen hat, mitgeführt. Kommt ein System, dass ein Paket empfängt, in diesem Pfad bereits vor, muss es sich um eine *Schleife* handeln und das Paket wird verworfen. Ein Vertreter dieser Klasse ist das, im Internet hauptsächlich verwendete, *Border Gateway Protocol* [RL95], ein sog. *Exterior Gateway Protocol*.

Linkstatusprotokolle

Protokolle in dieser Klasse unterscheiden sich grundsätzlich in ihrem Vorgehen. Es werden umfangreiche Informationen über die gesamte Topologie gesammelt, um daraus den bestmöglichen Weg zu berechnen. Je nach Protokoll findet der *Austausch mit anderen Systemen* nur in einem *nahen Umfeld* oder im *gesamten Netz* statt. Vertreter dieser Klasse sind *Open Shortest Path First* [CFML08] sowie das Optimized Link State Routing Protokoll.

2.6.3 Metriken

Um die Güte der Verbindung zu bewerten und letzten Endes eine Routingentscheidung treffen zu können, werden *Metriken* verwendet. Dies wird vor allem immer dann interessant, wenn es mehrere mögliche Wege zu einem Ziel gibt. Die einfachste vorstellbare Metrik kann man am Beispiel des Routing Information Protocol zeigen: Der *HopCount*. Er gibt an, wie viele weitere Router bis zum Ziel zu passieren sind. Dies erscheint auf den ersten Blick als die sinnvollste Lösung, aber allein die Verfügbarkeit zahlreicher Routingverfahren mit anderen Bewertungskriterien zeigt, dass es immer auf den Einsatzzweck und die äußeren Umstände bei der Bewertung einer Verbindung ankommt. Allerdings lässt sich eine Einteilung in drei Berechnungsverfahren treffen [BHSW07]:

- **Additive Berechnung**: Hier wird der Wert pro Hop addiert. Dies erscheint z. B. bei der Verwendung des HopCounts als sinnvoll: Je geringer der Wert, desto besser die Verbindung. Für einen Pfad $p_{i,j} = (n_{i+1}, n_{i+2}, ..., n_{j-1}, n_j)$ vom Knoten n_i zum Knoten n_j berechnet sich die Metrik $d(p_{i,j})$ wie folgt:

$$d(p_{i,j}) = d(p_{i,i+1}) + d(p_{i+1,i+2}) + d(p_{i+2,i+3}) \\ +... + d(p_{j-2,j-1}) + d(p_{j-1,j})$$

- **Multiplikative Berechnung**: Hier wird der Wert pro Hop multipliziert. Dies erscheint z. B. bei der Verwendung des PacketLoss als sinnvoll. Berechnet man die Metrik aus der Differenz von PacketLoss und 1, dann ergibt der höchste Wert die beste Verbindung. Für einen Pfad $p_{i,j} = (n_{i+1}, n_{i+2}, ..., n_{j-1}, n_j)$ vom Knoten n_i zum Knoten n_j berechnet sich die Metrik $d(p_{i,j})$ wie folgt:

$$d(p_{i,j}) = d(p_{i,i+1}) \cdot d(p_{i+1,i+2}) \cdot d(p_{i+2,i+3}) \\ \cdot... \cdot d(p_{j-2,j-1}) \cdot d(p_{j-1,j})$$

- **Konkave Berechnung**: Hier wird der geringste Wert aller Hops gewählt. Dies wäre eine geeignete Methode, wenn die verfügbare Bandbreite als Bewertungskriterium verwendet werden soll, da hier der geringste Wert ausschlaggebend ist. Der höchste Wert ergibt wieder die beste Verbindung. Für einen Pfad $p_{i,j} = (n_{i+1}, n_{i+2}, ..., n_{j-1}, n_j)$ vom Knoten n_i zum Knoten n_j berechnet sich die Metrik $d(p_{i,j})$ wie folgt:

$$d(p_{i,j}) = \min((d(p_{i,i+1}), d(p_{i+1,i+2}), d(p_{i+2,i+3}) \\ , ..., d(p_{j-2,j-1}), d(p_{j-1,j})))$$

2.6.4 Bewertung von MANET Protokollen

Die folgende Liste quantitativer Metriken kann zur Bewertung von Verfahren herangezogen werden [CM99]:

- **Ende zu Ende Durchsatz und Latenz**: Hierdurch bekommt man eine Rückmeldung darüber, wie schnell und mit welcher Bandbreite Daten durch das Netz geleitet werden können.
- **Zeitaufwand für das Erstellen neuer Routen**: Hierbei handelt es sich, vor allem beim Vergleichen von reaktiven Verfahren, um einen interessanten Faktor, da dieser einen allgemeinen Nachteil darstellt.
- **Anteil abgelaufener Übermittlungen**: Einige Protokolle, wie TCP, erfordern eine zeitnahe Übermittlung der Pakete. Werden gewisse Zeitfenster überschritten, kann es zu eklatanten Leistungseinbußen kommen.
- **Effizienz des Verfahrens**: Müssen sich Kontrollinformationen und übertragene Daten die Verbindung teilen, kann es zu Leistungseinbußen und Fehlfunktionen kommen. Daher sollte das Verhältnis zwischen übertragenen Daten und benötigten Kontrollinformationen berücksichtigt werden.

Im Rahmen dieser Arbeit wird die oben genannte Liste um den Punkt *Lastverteilung* ergänzt, wobei der Fokus auf der *Verteilung des Energieverbrauchs* liegen wird.

Kapitel 3

AODV und OLSR

Für die Untersuchung im Rahmen dieser Arbeit wurden die o. g. Protokolle aus folgenden Gründen ausgewählt:

- Beide wurden für den Einsatz in vermaschten Netzen entwickelt.
- Weder AODV, noch OLSR berücksichtigen den Energieverbrauch oder den Ladezustand der Systeme, auf denen sie eingesetzt werden.
- Sie unterscheiden sich in ihrer Funktionsweise: AODV ist eine *reaktives Distanzvektor-Verfahren*, OLSR hingegen arbeitet *proaktiv* und setzt auf den *Linkstatus*.
- Beide Protokolle sind über das Framework INETManet für OMNET++ verfügbar und hinreichend implementiert.

3.1 AODV

Bei Ad-hoc On-demand Distance Vector gemäß RFC3561 [PBRD03] handelt es sich um eine Weiterentwicklung des *Destination-Sequenced*

Distance Vector (DSDV) Verfahrens mit dem Ziel, die Menge an *Broadcast-Nachrichten*, die für den Betrieb in das Netz geschickt werden, zu reduzieren [BTA+11].

3.1.1 Grundlegende Funktionsweise

Da es sich bei AODV um ein *anforderungsbasiertes Verfahren* handelt, führt ein Host erst Aktionen aus, wenn er ein IP-Paket versenden muss. Dies führt zu einem vergleichsweise geringen Overhead, wodurch es besonders für große Netze mit vielen Knoten interessant ist. Dem Problem der *Schleifenbildung*, welches bei dieser Klasse von Protokollen auftritt, wird durch den Einsatz von *Sequenznummern* entgegengewirkt. Für die Kommunikation zwischen den Hosts kommen *UDP-Pakete* auf *Port 654* zum Einsatz. Es gelten folgende Begriffe:

- **Host**: Ein *Teilnehmer* im Netz, der als Router fungiert.
- **Originator**: Der *Urheber* eines Route Request (RREQ).
- **Absender**: Der *Absender* eines IP Pakets, es muss nicht der Originator sein.
- **Route**: Eine Route besteht immer aus einer *Destination*, einer *Metrik* und dem *NextHop*.
- **Routingtabelle**: Hier sammelt ein Host *alle Routen*, die er kennt.
- **Destination**: Das *Ziel* einer Route.
- **NextHop**: Der *nächste Host* über den die Destination erreicht werden kann.
- **HopCount**: Die *Anzahl an Hosts* bis zur Destination.
- **Metrik**: Die *Güte* der Verbindung, bei AODV wird der HopCount dafür genutzt.

3.1.2 Sequenznummern

Ein Host verfügt immer über eine eigene und zudem speichert er für jede Verbindung eine *Sequenznummer*, womit er der Schleifenbildung entgegenwirkt. Grundsätzlich kann man sagen, dass Updates von Routinginformationen immer nur dann ausgeführt werden, wenn die vorliegenden Informationen mit höheren Sequenznummern versehen sind.

3.1.3 Route Request (RREQ)

Sofern ein Host ein IP-Paket zustellen muss, dann benötigt er dafür eine Route. Liegt diese nicht vor, dann wird ein RREQ erzeugt und versendet. Dieser ist wie Folgt aufgebaut:

- **Type**: Bei RREQ gleich 1.
- **Flags**: Bit 0 (Join) und 1 (Repair) sind für Multicast reserviert, Bit 2 (Gratuitous) markiert, ob es sich um einen *Gratuitous RREQ* handelt, Bit 3 (Destination only) gibt an, ob nur das *eigentliche Ziel* auf den RREQ antworten darf und Bit 4 (Unknown) zeigt, ob die Sequenznummer des Ziels *unbekannt* ist.
- **Reserved**: Ungenutzt und immer 0.
- **HopCount**: *Abstand in Hops* vom Originator zum Host, der den Request bearbeitet.
- **RREQ ID**: Die *eindeutige Kennung* des RREQ in Kombination mit dem Originator.
- **Destination IP Address**: Die *Adresse*, für die eine Route gesucht wird.
- **Destination Sequence Number**: Die *letzte bekannte Sequenznummer*, die der Originator für das Ziel erhalten hat.

- **Originator IP Address**: Die *Adresse des Originators*, des Erzeugers des RREQ.
- **Originator Sequence Number**: Die *aktuelle Sequenznummer* des Originators.

Der allgemeine Ablauf für die Anforderung von Routen ist in Abbildung 3.1 dargestellt, es gelten folgende Definitionen:

- ***RT*** entspricht der lokalen *Routingtabelle* des Hosts.
- ***i*** ist eine *eindeutige Kennung* der Anfrage.
- ***TTLS*** ist der *Startwert* für die TTL von RREQs, meist wird sie sehr niedrig gewählt.
- ***TTLI*** ist der Wert um den die TTL *schrittweise* erhöht wird.
- ***TTLM*** ist der *Maximalwert* auf den die TTL erhöht werden darf.
- ***RETRYM*** ist der *Maximalwert* für die Anzahl der *Versuche*, die für die Ermittlung einer Route unternommen werden darf.
- ***ORT*** ist eine *Liste* mit den RREQs, die der Host selbst erzeugt hat (*Own Requests Table*).

Im Grunde versucht der Host innerhalb einer gewissen Zeitspanne ein Route Reply (RREP) für die gesuchte Route zu bekommen. Klappt das nicht, wird eine *festgelegte Anzahl* weiterer Versuche unternommen und dabei die TTL *schrittweise* gesteigert. Dies führt zu einer *Ringsuche* und verhindert unnötiges Fluten des Netzes. Zudem gibt es weitere Werte, wie eine Beschränkung der Anzahl an RREQs, die ein Host pro Sekunde versenden darf, *Route lifetimes* die die Dauer und *Route validities* die den Status der Gültigkeit von Routen angeben, die jedoch nicht weiter betrachtet werden.

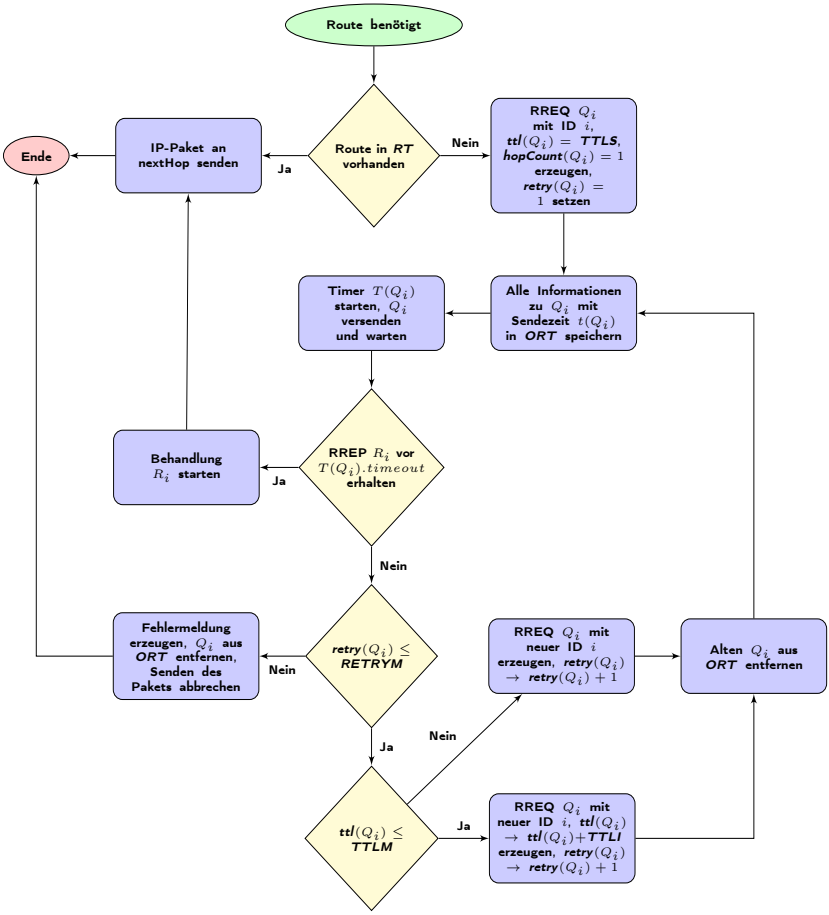

Abbildung 3.1: Vorgehen beim Versenden eines IP-Pakets nach RFC 3561 [PBRD03]

3.1.4 Route Reply (RREP)

Jeder Host, der einen RREQ empfängt verfährt damit, wie in Abbildung 3.2 dargestellt. Auch hierzu gelten folgende Definitionen:

- ***SRT*** ist eine *Liste* mit den RREQs, die der Host bearbeitet hat (*Seen Requests Table*).
- ***PDT*** ist der *zeitliche Abstand*, der zwischen der Bearbeitung von RREQs vom selben Originator für die selbe Destination liegen muss (*Path Discovery Time*).

Kann der Host die Anfrage positiv beantworten, sendet er ein RREP, das wie Folgt aufgebaut ist:

- **Type**: Bei RREP gleich 2.
- **Flags**: Bit 0 (Repair) ist für *Multicast* reserviert, Bit 1 (Acknowledgement) markiert, dass der *Empfang bestätigt* werden muss.
- **Reserved**: Ungenutzt und immer 0.
- **PrefixSize**: Wenn ungleich 0, gibt es die *Größe des Subnetzes* an, dass über den nextHop erreicht werden kann.
- **HopCount**: Der *Abstand in Hops* vom Originator zur Destination IP.
- **Destination IP Address**: Die *Route*, für die die Antwort geschickt wird.
- **Destination Sequence Number**: Die *Sequenznummer* für die Route.
- **Originator IP Address**: Die *Adresse des Originators*, des Erzeugers des RREQ.

Im Unterschied zu den RREQs wird die Antwort darauf per *Unicast* verschickt. Landet dieses RREP direkt beim Originator, dann werden die Informationen daraus entnommen. Empfängt ein anderer Host diese Antwort, dann handelt es sich um eine *Intermediate Node*. Die Behandlung unterscheidet sich zudem, abseits des in Abbildung 3.3 dargestellten Ablaufs in der Art, wie mit dem Aktualisieren und Anlegen von Routen im Zusammenhang mit den Sequenznummern umgegangen wird. Details hierzu können den entsprechenden Abschnitten in der RFC entnommen werden.

3.1.5 Routingtabelle

Die Routingtabelle, die durch AODV gepflegt wird, ist folgendermaßen aufgebaut:

- **Destination IP**: *Ziel* der Route.
- **Destination Sequence Number**: Die aktuelle *Sequenznummer* der Route.
- **Destination Sequence Number Validity**: Gibt die *Gültigkeit* der Sequenznummer an.
- **Otherstate Flag**: Gibt den *Zustand* der Route an, mögliche Werte sind *valid*, *invalid*, *repairable* und *being repaired*.
- **Interface**: Die *Schnittstelle* über die das Ziel erreicht werden kann.
- **HopCount**: Die *Anzahl* der Zwischenknoten bis zum Ziel.
- **NextHop**: Die IP Adresse des *nächsten Hosts*, über den das Ziel erreicht werden kann.
- **Precursors**: Eine *Liste* von Hosts, für die Verkehr über diese Route geleitet wurde.

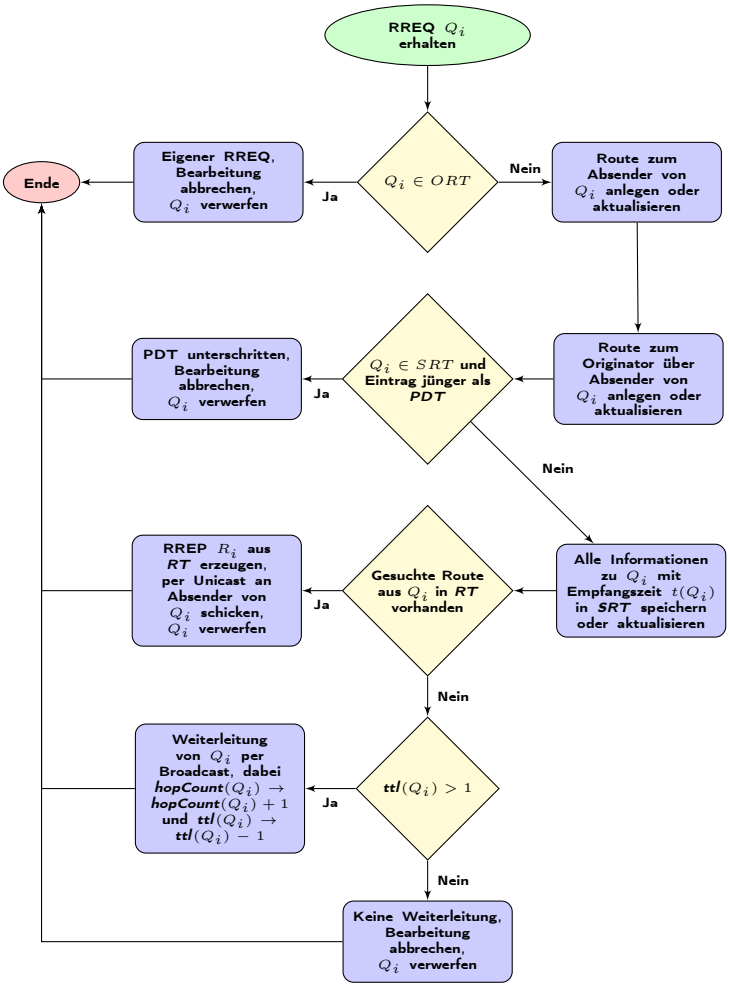

Abbildung 3.2: Behandlung eines RREQ nach RFC 3561 [PBRD03]

- **Lifetime**: Die *Gültigkeitsdauer* der Route. Wird bei jeder Benutzung aktualisiert, bleibt die Route bis zum Ablauf ungenutzt, wird sie als *invalid* markiert.

Die *Precursors-Liste* spielt im Zusammenhang mit *Route Errors (RERRs)* eine entscheidende Rolle. Eine *Intermediate Node* fügt jedes mal, wenn sie ein RREQ empfängt, den Absender der Liste des Eintrags für den nextHop zur entsprechenden Destination IP und diesen nextHop zur Liste des Eintrags für den Absender hinzu. Somit sind ihm zu jedem Eintrag immer genau die Hosts bekannt, die eine Route nutzen. Beim Weiterleiten von RREPs wird analog verfahren.

3.1.6 Route Error (RERR)

Ein Router wird einen RERR versenden, wenn er *während der Übertragung* über eine aktive Route einen *Ausfall* der Verbindung feststellt, wenn er ein Paket zur *Weiterleitung* für ein Ziel erhält, für das er *keine Route* kennt, oder er einen RERR von einem *Nachbarn* für eine aktive Route erhält. Auf die Betrachtung von Reparaturfunktionen wird bewusst verzichtet. In jedem der Fälle wird der Router zuerst alle *betroffen Routen* als ungültig markieren und die Auswirkungen auf andere Ziele in seiner Routingtabelle berechnen. Anschließend wird er, sofern Einträge in den entsprechenden Precursors-Listen vorhanden sind, die *betroffen Router* ebenfalls mittels eigener RERRs benachrichtigen. Um die Anzahl benötigter Nachrichten zu minimieren, können, wie dem Aufbau zu entnehmen ist, in einer Nachricht mehrere unerreichbare Ziele angegeben werden:

- **Type**: Bei RRER gleich 3.
- **Flags**: Bit 0 (NoDelete) gibt an, dass die Route nach Deaktivierung nicht gelöscht werden darf.
- **Reserved**: Ungenutzt und immer 0.

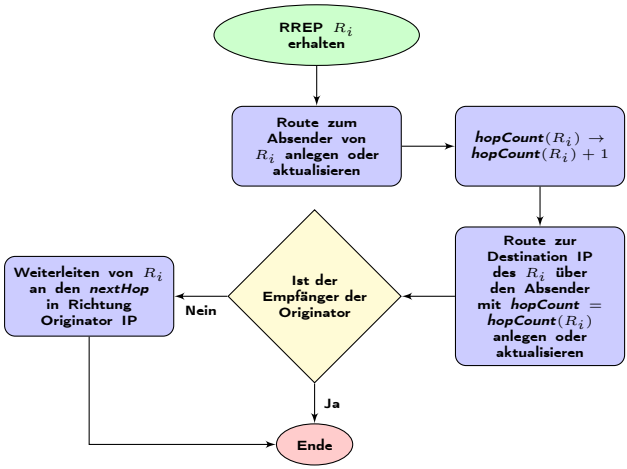

Abbildung 3.3: Behandlung eines RREP nach RFC 3561 [PBRD03]

- **DestinationCount**: Gibt die *Anzahl* unerreichbarer Ziele an, die im RERR enthalten sind größer 0.
- **Unreachable Destination IP Address**: Das erste *nicht erreichbares Ziel*, es können weitere folgen.
- **Unreachable Destination Sequence Number**: Die *Sequenznummer* des Ziels.
- **Additional Unreachable Destination IP Address**: Ein weiteres nicht erreichbares Ziel (darf mehrfach vorkommen).
- **Additional Unreachable Destination Sequence Number**: Die Sequenznummer des Ziels.

RERRs können in Anbetracht der Anzahl der Precursors sowohl per *Unicast* (einer), als auch per *Broadcast* verschickt werden (mehrere). Es ist jedoch auch möglich ausschließlich per Unicast zu versenden, wenn ein Broadcast unangemessen erscheint oder technisch nicht möglich

ist. Routen, die als inaktiv markiert sind, werden *nicht sofort* aus den Tabellen *entfernt*. Sie bleiben, da sie ggf. repariert werden können, einen *gewissen Zeitraum* in der Tabelle. Dieser wird jedes mal verlängert, wenn ein Paket für eine inaktive Route empfangen wird.

3.1.7 Zusammenfassung AODV

Es handelt sich bei AODV um ein vergleichsweise *simples* und *sparsames* Verfahren. Durch seine reaktiven Eigenschaften kommt es mit einem Miminum an Kontrollnachrichten aus und spart somit Strom und Bandbreite. Wenn bestimmte Routen nur sehr selten, aber dennoch regelmäßig benutzt werden, kann es durch ein Ablaufen der *Lifetime* zu neuen Routenanforderungen kommen. Dennoch bleibt der Overhead deutlich unter dem von reaktiven Verfahren, wie z. B. OLSR. Zudem kann die Lifetime durch Einstellungen beeinflusst werden, damit hier eine Optimierung möglich ist. Allerdings besitzt es dadurch auch eine gewisse Trägheit, die z. B. bei hoher *Mobilität* zu hohen *Verzögerungen* führen können. Ein beliebtes Anwendungsfeld für solche Verfahren sind Netze, in denen der Verkehr in der Regel eine sehr begrenzte Anzahl an Zielen hat, z. B. bei *Streaminganwendungen* oder bei *Wireless Sensor and Actor Networks* [BTL13]. In diesen Fällen kann man sehr *große Netze* aufbauen, da die Last durch die Kontrollnachrichten nicht durch die Anzahl der Teilnehmer, sondern der ausgeführten Anforderungen nach neuen Routen bedingt ist.

Zur Bewertung der *Güte der Verbindung* wird ausschließlich der *HopCount* genutzt. Daher muss eine Anpassung des Verfahrens, die den *Ladezustand* der Teilnehmer bei der Routenfindung berücksichtigt, den *HopCount* manipulieren. Dies hätte den Vorteil, dass Router, die die angepasste Version nutzen und Geräte, die mit normalem AODV arbeiten, *kompatibel* wären. Wie genau diese Anpassung aussehen kann, wird in den Kapiteln 4 und 5 beschrieben.

3.2 OLSR

Bei Optimized Link State Routing gemäß RFC3626 [CJ03] handelt es sich um eine Weiterentwicklung der klassischen Link-State Verfahren mit einigen speziell auf die Anforderungen von drahtlosen Netzen ausgerichteten Optimierungen. Als zentraler Entwicklungsschritt sind hier *Multipoint Relays (MPRs)* anzusehen, die jeder Teilnehmer für sich bestimmt und die die Organisation innerhalb des Netzes übernehmen [BTA+11]. Somit wird die Belastung der Infrastruktur durch Kontrollinformationen deutlich verringert. Die folgende Beschreibung basiert auf der RFC, daher werden keine weiteren Angaben zu Quellen gemacht.

3.2.1 Grundlegende Funktionsweise

Da es sich bei OLSR um ein *proaktives Verfahren* handelt, führt ein Host immer Aktionen aus, unabhängig davon, ob er ein IP-Paket versenden muss oder nicht. Dies stellt sicher, dass jeder Teilnehmer nach einem gewissen Zeitraum eine hinreichend aktuelle Übersicht über das Netz bekommt. Ausfälle oder Änderungen werden im Rahmen regelmäßiger Nachrichten im Netz propagiert. Der zeitliche Abstand dieser Nachrichten steht in konstantem Verhältnis zum verursachten *Overhead*: Je schneller das Netz auf Änderungen reagieren soll, desto höher die Belastung. Grundsätzlich gibt es folgende Intervalle, die Standardwerte sind der RFC entnommen:

- **HELLO_INTERVAL**: 2 Sekunden
- **REFRESH_INTERVAL**: 2 Sekunden
- **TC_INTERVAL**: 5 Sekunden
- **MID_INTERVAL**: TC_INTERVAL
- **HNA_INTERVAL**: TC_INTERVAL

Die entsprechenden *Holding Times* sollten nach RFC das Dreifache betragen, damit erst nach dreimaligem Ausbleiben eine Reaktion eintritt. Auf die einzelnen Nachrichten wird im weiteren Verlauf genauer eingegangen. Für die Kommunikation zwischen den Hosts kommen *UDP-Pakete* auf *Port 698* zum Einsatz. Im Gegensatz zu AODV gibt es jedoch nur eine Art von Paket, das dann als Nutzlast die *OLSR Messages* enthält. Es können mehrere Messages innerhalb eines Pakets verschickt werden.

3.2.2 OLSR Paket

Ein OLSR Paket ist folgendermaßen aufgebaut, wobei *Message Header* und *Message Data* paarweise wiederholt werden können (siehe Abbildung **??**):

- **Packet Length**: Die *Länge* des Pakets in Bytes.
- **Packet Sequence Number**: Die individuelle *Sequenznummer* des Pakets.
- **Message Type**: Die *Art* des Paketes, je nach enthaltener OLSR Message.
- **VTime**: Die *Gültigkeitsdauer* der enthaltenen OLSR Message, solange keine weitere gültige Nachricht erhalten wird.
- **Packet Length**: Die *Gesamtlänge* des Paketes, von *Message Type* bis zum Ende des Feldes *Message*.
- **Originator Address**: Die *IP Adresse* des Urhebers der OLSR Message.
- **Time To Live**: Die *TTL* der Nachricht, also die Anzahl noch erlaubter Weiterleitungen.

- **Hop Count**: Die *HopCount* der Nachricht, also die Anzahl bereits getätigter Weiterleitungen.
- **Message Sequence Number**: Die individuelle *Sequenznummer* der OLSR Message.
- **Message**: Die *eigentliche Information*, die transportiert wird in variabler Größe.

Es gibt je nach Erweiterung und Variante des Protokolls zahlreiche Arten von OLSR Messages, die von allen Teilnehmern weitergeleitet werden können (der entsprechende Algorithmus stellt alle Funktionen für das Weiterleiten unbekannter Arten von Nachrichten bereit). Für die Minimalfunktion müssen von einem Teilnehmer jedoch folgende Nachrichten verarbeitet werden:

- **MID**: Informationen über verfügbare Schnittstellen
- **HELLO**: Kontaktaufnahme durch Knoten
- **TC**: Informationen über die Topologie

3.2.3 Multiple Interface Declaration (MID)

OLSR unterstützt den Einsatz auf mehreren Schnittstellen. Somit kann ein Host mehrere Adressen besitzen, die *Interface Addresses*. Jeder Host wählt daraus eine Adresse als seine *Main Address* aus, die als Kennung im Netz dient. Sofern nur eine Schnittstelle und somit eine Adresse vorhanden ist, ist die Beziehung trivial. Wenn jedoch mehrere existieren, dann wird eine davon zufällig gewählt. Die Information über die vorhandenen Adressen werden dann mittels *Multiple interface declaration messages (MID messages)* verteilt. Als *TTL* sollte 255 gewählt werden (Maximum), um das gesamte Netz zu durchdringen. In der Nachricht sind alle Adressen, außer der *Main Address* enthalten. Die Zuweisung erfolgt über die *Originator Address* aus dem Header des

OLSR Pakets. Die MID messages werden periodisch von Hosts mit mehr als einem OLSR Interface verschickt, wobei sich der maximale zeitliche Abstand nach dem *MID_INTERVAL* richtet. Ein Host, der eine MID message verarbeitet, fügt sofern der Absender zu den symmetrischen One hop neighbours (1HNBs) gehört, die in der Nachricht enthaltenen OLSR Adressen in seine Datenbank ein. Die Weiterleitung erfolgt per *default forwarding algorithm (DFW)*.

3.2.4 Hello (HELLO)

Die *Hello messages (HELLO messages)* werden für das Erkennen von Links (*link sensing*) und *1HNBs* (*neighbourhood detection*) sowie die *Bekanntmachung* der selbst gewählten *MPRs* eingesetzt. In dieser werden Informationen über die eigenen 1HNBs per *Broadcast* an diese mit *TTL 1* gesendet, damit keine Weiterleitung stattfindet. Eine HELLO message, die als Nutzlast im OLSR Paket gesendet wird, kann mehrere Einträge, immer beginnend mit *Link Code*, enthalten und ist folgendermaßen aufgebaut:

- **Reserved**: Reservierter Bereich, besteht aus Nullen.
- **HTime**: Das *Intervall*, in dem HELLO messages gesendet werden, daraus kann erkannt werden wann die nächste HELLO message erfolgen muss.
- **Willingness**: Die *Breitschaft* des Absenders Daten weiterzuleiten.
- **Link Code**: Die *Beschreibung der Verbindung* zum 1HNB besteht aus *Link Type* und *Neighbour Type*. Für den Code stehen *Unbekannt (UNSPEC_LINK)*, *Asymmetrisch (ASYM_LINK)*, *Symmetrisch (SYM_LINK)* und *Unterbrochen (LOST_LINK)*, für den Type *Symmetrisch (SYM_NEIGH)*, *MPR (MPR_NEIGH)* und *Kein Nachbar (NOT_NEIGH)*, was

angibt, dass der 1HNB bisher weder ein MPR, noch ein symmetrischer 1HNB ist, zur Verfügung.

- **Reserved**: Weiterer reservierter Bereich, besteht aus Nullen.
- **Link Message Size**: Die *Größe* der Nachricht in Bytes, gemessen von *Link Code* bis zum nächsten *Link Code*.
- **Neighbour Interface Address**: Es folgen *alle bekannten Adressen* des 1HNB.

Die HELLO messages werden periodisch von Hosts verschickt, wobei sich der maximale zeitliche Abstand nach dem *HELLO_INTERVAL* richtet. Ein Host, der eine HELLO message verarbeitet, fügt die Adressen aus der Nachricht zusammen mit der Adresse der eigenen Schnittstelle über die empfangen wurde in seine Datenbank ein oder aktualisiert bereits vorhandene Einträge. Die Weiterleitung von HELLO messages ist verboten.

3.2.5 Topology Control (TC)

Durch die bisher bekannten HELLO messages besitzen alle Knoten im Netz die Informationen über die Adressen ihrer 1HNBs, durch die MID messages können sie die verfügbaren Netze hinter den *Main Addresses* ermitteln. Damit daraus brauchbare Routinginformationen generiert werden können, müssen weitere Informationen im Netz verteilt werden. Dies ist die Aufgabe der *Topology control messages (TC messages)*. Hiermit gibt ein Hosts dem gesamten Netz seine *Advertised Neighbours* bekannt (die Nachricht wird als Broadcast mit TTL 255 verschickt):

- **ANSN**: Die *Advertised Neighbour Sequence Number* dient als Indikator für die Aktualität der Informationen. Jedes mal, wenn sich etwas an den Verbindungen eines Hosts ändert wird diese inkrementiert. Auf Basis dieser Nummer können andere Teilnehmer entscheiden, ob Sie die Nachricht verarbeiten.

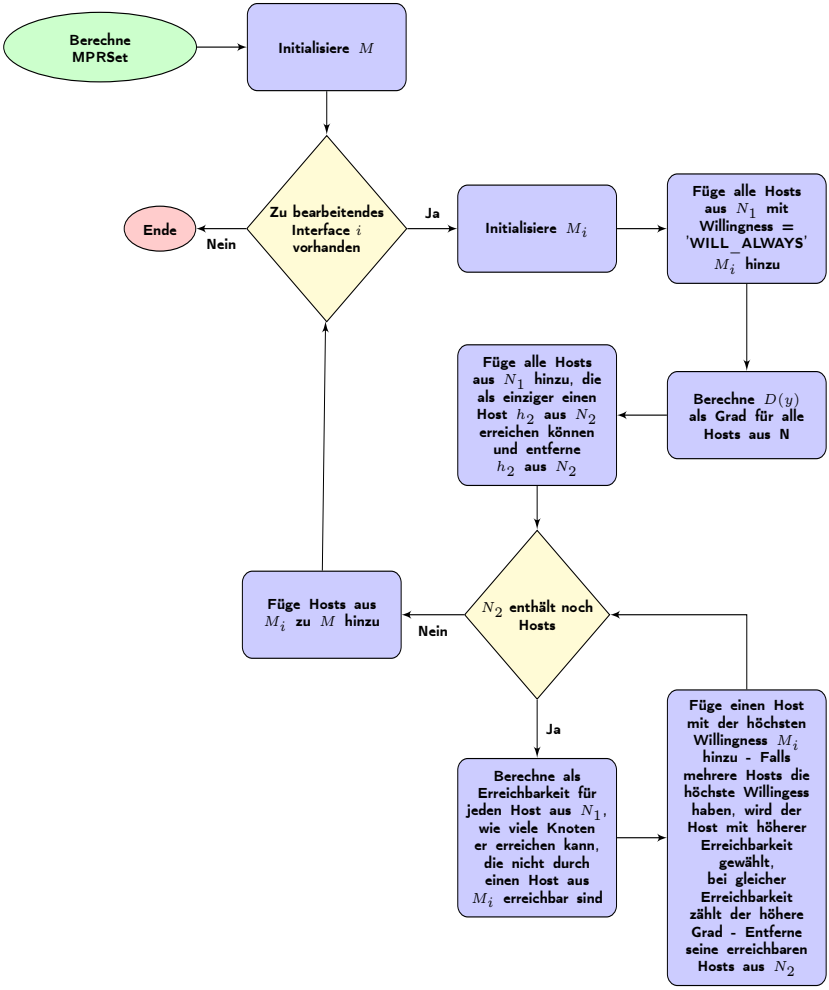

Abbildung 3.4: Bestimmung MPR Set nach RFC3626 [CJ03]

- **Reserved**: Ein reservierter Bereich.
- **Advertised Neighbour Main Address**: Die *Hauptadressen* seiner *Advertised Neighbours*. Hierbei handelt es sich mindestens um seine MPRs.

Die TC messages werden periodisch von Hosts verschickt, wobei sich der maximale zeitliche Abstand nach dem *TC_INTERVAL* richtet. Ein Host, der eine TC message verarbeitet, fügt, sofern der Absender zu den symmetrischen 1HNBs gehört, die in der Nachricht enthaltenen OLSR Adressen zusammen mit der Absenderadresse in seine Datenbank ein oder aktualisiert bereits vorhandene Einträge. Die Weiterleitung erfolgt per DFW.

3.2.6 Willingness

Die Willingness kann Werte zwischen 0 und 7 annehmen. Je höher der Wert, desto höher die Bereitschaft, wobei bei 0 gar kein Verkehr, bei 7 jedoch nach Möglichkeit immer weitergeleitet werden soll. Es gelten folgende Bezeichnungen:

- **WILL_NEVER**: 0
- **WILL_LOW**: 1
- **WILL_DEFAULT**: 3
- **WILL_HIGH**: 6
- **WILL_ALWAYS**: 7

3.2.7 Multipoint Relay (MPR)

Wie bereits erwähnt, erstellt jeder Host eine eigene Liste von MPRs aus seinen 1HNBs zu denen er eine symmetrische Verbindung besitzt. Im

Normalfall sind in dieser alle enthalten die ausreichen, um alle *Strict two hop neighbours (S2HNBs)* zu erreichen. Grundsätzlich werden Nachrichten eines OLSR Hosts ausschließlich von seinen MPRs weitergeleitet, um die Belastung des Netzes durch Kontrollinformationen gering zu halten. Die Erstellung erfolgt pro Schnittstelle, wie in Abbildung 3.4 dargestellt, hierbei entsprich M dem MPR-Set. Der Host teilt einem Nachbarn, ob er als MPR ausgewählt wurde, über das *Link Type* Feld der HELLO messages mit. Es gelten folgende Definitionen:

- ***I***: Die zu behandelnde Schnittstelle.
- **Neighbour of an Interface (NoI)**: Ein *1HNB*, zu dem der Host auf der Schnittstelle *I* einen Link zu einer beliebigen Schnittstelle auf diesem hat.
- **2-hop neighbours reachable from an interface (2HNBRI)**: Ein *Two hop neighbour (2HNB)*, zu dem die NoIs eine Verbindung haben.
- **MPR set of an interface (MPRSI)**: Eine Teilmenge der NoIs mit einer *Willingness* höher *WILL_NEVER*. Sie enthält genau die Menge an Hosts, über die alle S2HNBs erreicht werden können.
- ***N***: Menge aller 1HNBs, die über *I* erreichbar sind.
- ***N2***: Menge aller 2HNBs, die über *I* erreichbar sind, ohne die, die nur über Hosts aus *N* mit einer Willingness *WILL_NEVER* erreicht werden können, sich selbst und alle symmetrischen 1HNBs.
- ***D(y)***: Der Grad eines 1HNB y. Die Anzahl der symmetrischen Nachbarn ohne Elemente aus N und sich selbst.

Zudem führt er eine weitere Liste, in der alle Hosts enthalten sind, die ihn als MPR verwenden. Diese Information wird wie beschrieben aus den HELLO messages seiner Nachbarn generiert.

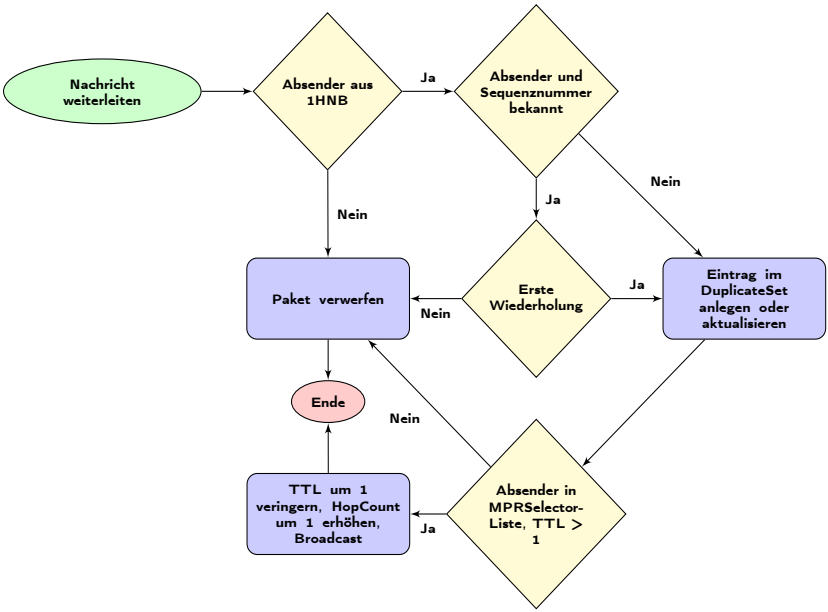

Abbildung 3.5: DFW nach RFC3626 [CJ03]

3.2.8 Default Forwarding Algorithm

MID messages und TC messages werden mittels des *DFW* weitergeleitet. Eine Beschreibung des Ablaufs ist der Abbildung 3.5 zu entnehmen. Grundsätzlich werden alle Nachrichten, sofern die TTL größer 1 ist von einem Host weitergeleitet, die von Hosts, die ihn als MPR gewählt haben und aus seiner symmetrischen 1HNB stammen, gesendet wurden.

3.2.9 Ablauf, Metrik, Routen

Alle Router senden kontinuierlich wie bereits dargestellt Ihre Nachrichten aus. Durch die HELLO messages wissen alle direkten Nachbarn von ihrer jeweiligen Existenz. Durch die MID messages und TC messages

werden im Netz per Broadcast alle weiteren notwendigen Informationen verteilt, die für das Erstellen der Routingtabelle notwendig sind, wobei nur die jeweiligen MPRs eines Hosts Nachrichten weiterleiten. Anschließend werden die Routen zu einem Ziel über einen *shortest path* Algorithmus berechnet, als maßgebende *Metrik* ist der jeweilige *HopCount* anzusehen.

3.2.10 Zusammenfassung OLSR

Es handelt sich bei OLSR um ein vergleichsweise *einfaches* Verfahren. Durch seine proaktiven Eigenschaften kommt es zu einer höheren Anzahl an Kontrollnachrichten und beansprucht somit mehr Strom und Bandbreite. Jedoch kann hierdurch nach einer gewissen Vorlaufzeit ein Minimum an Reaktionszeit bei der Wahl neuer Routen erreicht werden. Allerdings besitzt auch OLSR dadurch eine gewisse Trägheit, da sich neue Informationen erst im Netz verbreiten müssen. Ein mögliches Anwendungsfeld für dieses Verfahren sind Netze, in denen der Verkehr eine hohe Zahl oder ständig wechselnde Ziele hat. OLSR eignet sich am besten für dichte Netze mit vielen Systemen. Je größer jedoch das Netz und die Anzahl der Teilnehmer wird, desto höher wird auch die Belastung durch Kontrollnachrichten und die benötigte Zeit für die Ausbreitung der Routen-Informationen, was unter Umständen problematisch sein kann.

Zur Bewertung der *Güte der Verbindung* wird ausschließlich der *HopCount*, für die Wahl der MPRs jedoch auch die *Willingsness* berücksichtigt. Daher kann eine Anpassung des Verfahrens, die den *Ladezustand* der Teilnehmer bei der Routenfindung berücksichtigt, die *Willingness* von Hosts manipulieren. Dies hat den Vorteil, dass Router die die angepasste Version nutzen und Geräte, die mit normalem OLSR arbeiten, *kompatibel* wären. Auf die Veränderung des *HopCount* kann verzichtet werden, damit keine falschen Metriken erzeugt werden.

3.3 Zusammenfassung OLSR und AODV

Wie bereits beschrieben eignet sich OLSR sehr gut für dichte Netze, die auf eine geringe Verzögerung angewiesen sind und in denen häufig neue Routen benötigt werden. AODV hingegen ist vor allem dann vorteilhaft, wenn weniger Routen benötigt werden und die Verzögerung bei deren Anforderung eine untergeordnete Rolle spielt. Ein weiterer Unterschied liegt in der Zeit bis ein Netz betriebsbereit ist: Während bei AODV die benötigten Routen zielgerichtet und damit mit hoher Priorität ermittelt werden, müssen sich Routinginformation bei OLSR erst durch periodische Nachrichten verbreiten, was gerade bei größeren Netzen eine gewisse Zeit in Anspruch nimmt. Da bei OLSR periodisch Kontrollnachrichten verschickt werden, benötigt es, sofern keine Daten weitergeleitet werden, in der Regel mehr Strom als AODV, da hier kein Kontrollverkehr anfällt.

Kapitel 4

Versuchsbeschreibung

Im Rahmen dieser Arbeit wurden die genannten Protokolle dahingehend angepasst, dass bei mehreren verfügbaren Wegen der Verkehr aufgeteilt wird, damit Teilnehmer mit einem begrenzten Energievorrat möglichst ausgeglichene Ladestände aufweisen. Dieses Kapitel widmet sich der eingesetzten Technik und deren Anpassung.

4.1 Versuchsaufbau

4.1.1 Eingesetze Technik

Für die Simulation wurde folgende Software eingesetzt:

- Fedora Linux 26, 4.13.16-200.fc26.x86_64
- gcc-Version 7.2.1 20170915 (Red Hat 7.2.1-2) (GCC)
- Simulationssoftware OMNeT++, Version: 5.1
- Framework Inetmanet-3.5

Bei *OMNeT++* handelt es sich um ein Simulationssystem mit vielfältigen Möglichkeiten. Mittels dem Basis-Framework *INET* lassen sich alle erdenklichen Protokolle, Szenarios oder Technologien aus dem Bereich Netzwerk in beliebiger Tiefe simulieren. Da MANET Protokolle leider nicht Bestandteil dessen sind, wurde es jedoch durch einen Fork *Inetmanet* ersetzt. Letzteres stellt eine Erweiterung dar, in der diverse MESH Protokolle zur Verfügung stehen. Das Framework besteht aus einer großen Sammlung von C++ Klassen, die sich durch konsequente Vererbung auszeichnen. Als IDE kommt die bei Entwicklern sehr populäre Eclipse-GUI zum Einsatz (Abbildung 4.2) es kann jedoch auch mit einem Editor und dem C++ Compiler gearbeitet werden. Eine Simulation besteht in der Regel aus folgenden Elementen:

- Mindestens einer *Network-Description-Datei (NED)*. Diese Dateien haben zwei Aufgaben: Zum einen dienen Sie als Bindeglied zu den C++ Klassen, in dem Sie Schnittstellen bedienen und selbst zur Verfügung stellen. Zum anderen werden sie für die Definition des Aufbaus von Simulationen genutzt. Abbildung 4.1 zeigt ein einfaches Setup: Hier werden erst benötigte NEDs eingebunden, anschließend wird ein Netz mit dem Namen *AODVCoreNetwork* und einem *Objekt* vom Typ *AODVRouter* definiert.

- Mindestens einer *Konfigurationsdatei (INI)*. Hier werden alle wichtigen Einstellungen für die Elemente der Network-Description vorgenommen. In Abbildung 4.4 wird dem Objekt *sender1* eine *Ping-Anwendung* zugewiesen, die nach fünf Sekunden jede Sekunde eine Nachricht an das Objekt *receiver1* sendet. Es besteht die Möglichkeit, mehrere Konfigurationsdateien durch Vererbung zu kombinieren.

- Mindestens einer *C++ Klasse*, die das eigentliche Verhalten der Objekte definiert. In der Regel wird in einer Klasse nur das ganz

```
package powerrouting.simulations;
import powerrouting.node.aodv.AODVRouter;
import powerrouting.node.aodvpo.AODVPORouter;

network AODVCoreNetwork
{
    parameters:
        @display("bgb=1050,850");
    submodules:
        router32: AODVRouter {
            parameters:
                @display("i=device/pocketpc_s
                  ;r=,,#707070;p=400,250");
        }
    connections allowunconnected:
}
```

Abbildung 4.1: Auszug NED in OMNeT++

spezielle Verhalten des Objektes definiert, alle anderen Funktionen sind per Vererbung integriert.

Aus den benötigten Klassen wird mittels Compiler eine Bibliothek erzeugt, die für die Simulation genutzt wird. Hier gibt es wieder zwei Möglichkeiten:

- Eine grafische Oberfläche (Abbildung 4.3). Sie eignet sich besonders dann, wenn die Visualisierung der Prozesse im Vordergrund steht.

- Die Ausführung per Kommandozeile. Dies bietet sich vor allem dann an, wenn man keine Interaktion wünscht. Es besteht die Möglichkeit automatisiert verschiedene Variablenbelegungen in mehreren Durchläufen mit verschiedenen Zufallszahlen zu testen. Zudem ergibt sich gegenüber der GUI ein Geschwindigkeitsvorteil.

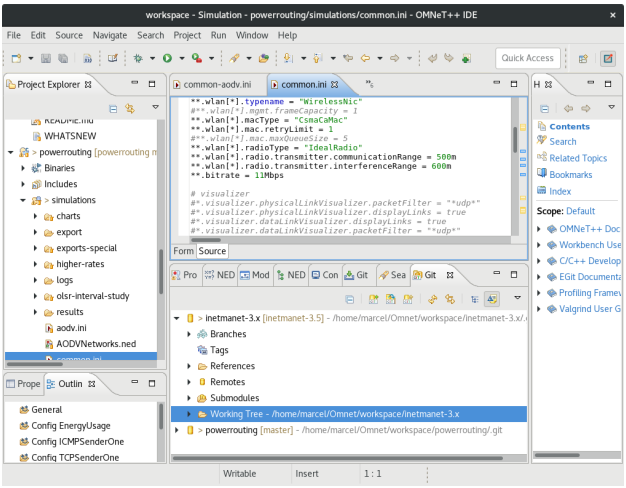

Abbildung 4.2: IDE-GUI von OMNeT++

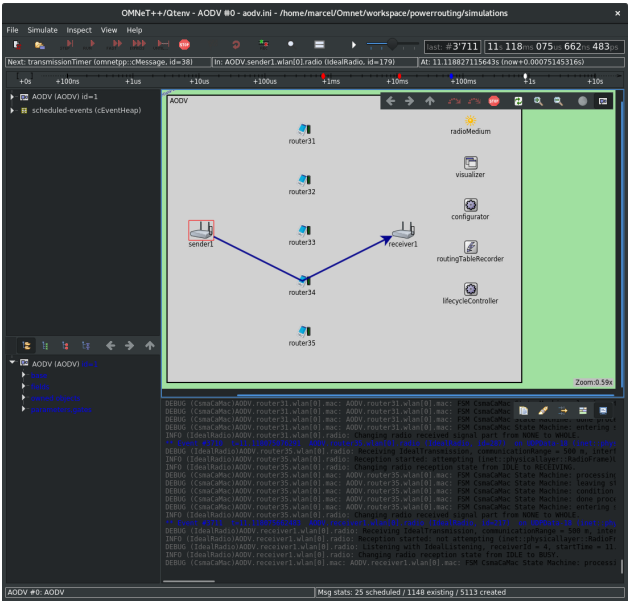

Abbildung 4.3: Simulations-GUI von OMNeT++

Die Simulationen erzeugen verschiedene Dateien, in denen alle Werte gemäß der Programmierung gespeichert sind. Des weiteren lassen sich die Nachrichten aller Objekte aufzeichnen. Eine tiefere Beschreibung der Möglichkeiten wird im nächsten Kapitel im Rahmen der Auswertung der Ergebnisse vorgenommen.

Da die Anpassung von OLSR per Vererbung eine Änderung am Framework erforderlich gemacht hätte, wurden die für die Protokolle OLSR und AODV benötigten Klassen komplett kopiert und in einen *eigenen Namespace* überführt. Dort wurden die angepassten Versionen dann per Vererbung realisiert. Der gesamte Code, der im Rahmen dieser Arbeit erstellt wurde, steht unter der GPL auf GitHub zur Verfügung.[1] Da bei OLSR nur die einfachste Version genutzt wird, sind beide Verfahren nah an den jeweiligen RFCs und vor allem für die angedachten Versuche hinreichend implementiert. Im Fall von AODV scheint es in speziellen Fällen Probleme mit der *loop avoidance* zu geben. Die Ursachen hierfür konnten aber nicht abschließend geklärt werden. Die Implementierung bestehen in der Regel aus folgenden Komponenten:

- Einer NED für die *Router*, in der die Verwendung des jeweiligen *Routingverfahrens* bestimmt und alle Eigenschaften und Funktionen eines *Wireless Hosts* erbt.
- Einer NED für das *Protokoll*. Hier wird die Verbindung zu den *C++ Klassen* hergestellt und es werden verschiedene *Einstellungen* definiert.
- Einer Sammlung aus *C++ Klassen*, welche die Kontrollnachrichten und das Verhalten definieren.

Leider sind die Protokolle nicht mit allen verfügbaren *Radio*, *MAC* und *Interface-Typen* kompatibel umgesetzt. Es konnte jedoch eine Schnittmenge gefunden werden, um die Vergleichbarkeit

[1] https://github.com/marcelebbrecht/powerrouting

```
# ping sender one
**.sender1.numPingApps = 1
**.sender1.pingApp[0].destAddr = "receiver1(ipv4)"
**.sender1.pingApp[0].sendInterval = 1s
**.sender1.pingApp[0].startTime = 5s
```

Abbildung 4.4: Auszug INI in OMNeT++

sicherzustellen. Da nur das grundsätzliche Verhalten der Protokolle untersucht werden soll, wurde das verwendete Funkmodell möglichst einfach gewählt. Das Setup für die Simulationen lautet wie folgt:

- Als Schnittstellen-Typ wird *WirelessNic* genutzt. Dieser stellt eine einfache WLAN Schnittstelle zur Verfügung.
- Auf der Sicherungsschicht kommt *CSMA/CA* zur Anwendung.
- Als Radio-Typ wird *IdealRadio* verwendet. Die Bitrate beträgt 11MBit/s, die Reichweite für die Übertragung 500m, die für Interferenzen 600m. Hierdurch können immer die jeweils direkten Nachbarn erreicht und gestört werden.
- Zur Realisierung der endlichen Energiequelle und des Verbrauchers wurden die Klassen *IdealEpEnergyStorage* und *StateBasedEpEnergyConsumer* gewählt. Um die Bewertung der Ergebnisse zu vereinfachen, wurde der Energieverbrauch für alle Vorgänge, außer Senden, auf Null gesetzt. Bei den normalen Simulationen starten die Router mit derselben Ladung. Sie Sender und Empfänger verfügen jedoch über einen unbegrenzten Energievorrat.
- Alle maßgeblichen Versuche wurden in *100 Wiederholungen* mit verschiedenen Zufallszahlen durchgeführt, um eine hinreichende Signifikanz zu erreichen. Die Seeds wurden in der Konfiguration hinterlegt.

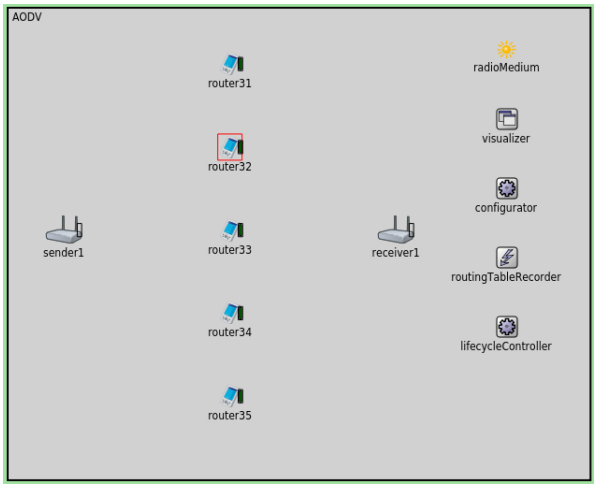

Abbildung 4.5: Single-Hop Testnetz

- Die Simulationszeit beträgt in der Regel 180 Sekunden. Da bei kürzerer Laufzeit die Ergebnisse abweichen, wurde eine zusätzliche Auswertung für 60 Sekunden erzeugt um die Unterschiede transparent zu machen.
- In den meisten Simulationen werden, nach einer *Vorlaufzeit von 10 Sekunden*, *UDP-Nachrichten* mit einem Abstand von circa *50ms* verschickt.

4.1.2 Simulationsbeschreibung

Für die verschiedenen Simulationen wurden folgende Topologien geschaffen:

- Ein Single-Hop Netz mit einem Sender, einem Empfänger und 5 Routern in der Mitte, die jeweils den Sender und Empfänger direkt erreichen können (Abbildung 4.5).

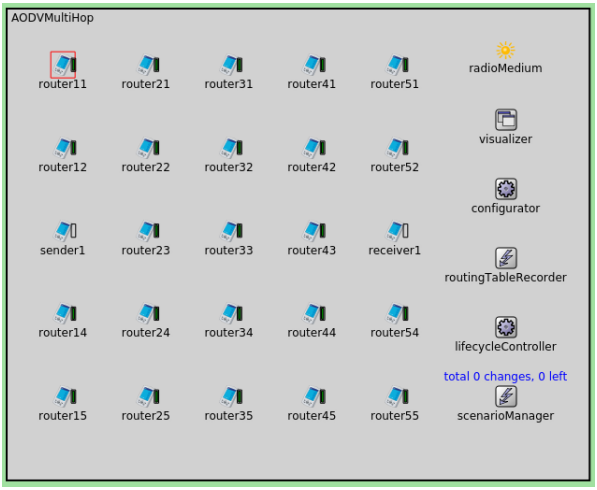

Abbildung 4.6: Multi-Hop Testnetz

- Ein Multi-Hop Netz mit einem Sender, einem Empfänger, mehr Routern und einer geringeren Reichweite. Die Pakete müssen mindestens über 3 Router geleitet werden, bis sie das Ziel erreichen (Abbildung 4.6).

Darüber hinaus gibt es weitere ergänzende Setups, auf welche in der Auswertung näher eingegangen wird:

- Abweichende Ladung beim Start
- Mischung aus angepasstem und normalem AODV/OLSR
- Eine ParameterStudy

4.1.3 Neue Metrik

Die RVs nutzen für die Bewertung der Routen den *HopCount*, wobei bei OLSR auch die *Willingness* in die Wahl der Gateways einbezogen wird. Um die Kompatibilität mit unangepassten Systemen beibehalten

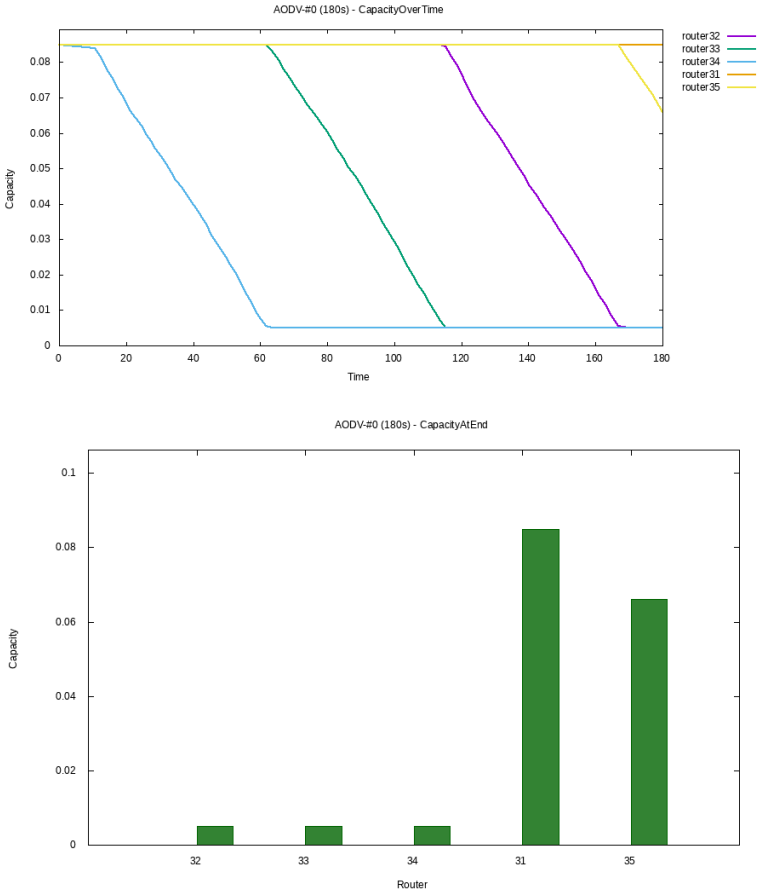

Abbildung 4.7: Energieverbrauch und -vorrat bei AODV nach 180 Sekunden

zu können, sollten diese Bewertungsmaßstäbe erhalten werden. Wie bereits in Kapitel 3 erwähnt, sollte also bei AODV der *HopCount*, bei OLSR die *Willingness* manipuliert werden. Im Normalfall wird, nachdem einmal eine gültige Route ermittelt wurde, ein Pfad solange genutzt, bis ein Router auf dem Weg nicht mehr zur Verfügung steht. Wenn man davon ausgeht, dass dies nur durch Abschaltung aufgrund von unzureichender Ladung der Energieversorgung geschehen kann, dann wird in solch einem Netz eine Energieversorgung nach der anderen verbraucht (siehe Abbildungen 4.7). Es ergibt sich also eine hohe Abweichung zwischen den Endständen. Da es das erklärte Ziel ist, die Weiterleitung von Paketen vom Stand der Energieversorgung der jeweiligen Router abhängig zu machen, muss eine neue Bewertungsmetrik definiert werden, die im weiteren als *Energy fairness* bezeichnet wird. Je geringer die Abweichung der Energiestände am Ende der Simulation, desto *fairer* ist die Verteilung. Die Metrik M entspricht somit der Standardabweichung der Ladestände. Sei n die Anzahl der Router, $C(i)$ die Restladung des jeweiligen Routers i und $\bar{x} = \sum_{i=1}^{n} \frac{C(i)}{n}$ das arithmetische Mittel der Restladungen, dann gilt:

$$M = \sqrt{\sum_{i=1}^{n} \frac{(C(i) - \bar{x})^2}{n}}$$

Je geringer dieser Wert ausfällt, desto ausgeglichener ist der Ladezustand und erfolgreicher die Anpassung. Allerdings muss auch berücksichtigt werden, dass Änderungen an den Routen zu Paketverlusten führen können, was bei häufigem Auftreten zu eklatanten Beeinträchtigungen führen kann. Um dies besser bewerten zu können, benötigen wird die *Performance* F. Sei L der durchschnittliche PacketLoss der Verbindungen im gesamten Netz und $c = 10^{-10}$, dann gilt

$$F = (1 - (L + c))/(M + c)$$

Der Grad in dem eine Neuberechnung der Verbindungen stattfinden muss, hängt an ein paar spezifischen Einstellungen, welche in den nach-

folgenden Abschnitten beschrieben und für die im Rahmen der Simulation mittels der *ParameterStudy* optimale Werte ermittelt wurden.

4.1.4 Anpassung AODV-PO (PowerOrientation)

Um den gewünschten Effekt zu erreichen, müssen die Router regelmäßig ihren Ladezustand überprüfen und dafür sorgen, dass die Routen angepasst werden. Durch das reaktive Design des Protokolls kommt es dabei aber zu kurzzeitigen Ausfällen, da erst eine neue Route ermittelt werden muss. Aus diesem Grund sollte die Anpassung nur nach einer signifikanten Änderung des Ladezustands eintreten. Diese Signifikanz wird im Weiteren als *Trigger* t bezeichnet und zwischen $0{,}1$ und $0{,}9$ gewählt. Sei $C(i)$ der relative Ladezustand des Hosts i zwischen 0 (leer) und 1 (voll). Wenn

$$(C(i) * 100) \mod (100t) = 0$$

gilt, wird eine Anpassung durchgeführt. So führt z. B. $t = 0{,}1$ zu einer Reaktion bei $100\%, 90\%, 80\%, ...$, also alle 10%. In diesem Fall wird ein RERR erzeugt und nach dem bekannten Schema verschickt, damit eine Neubestimmung der Routen beginnt. Ferner soll es möglich sein den Grad der Veränderung anzupassen, was als *Sensitivity* s bezeichnet wird und zwischen $0{,}1$ und 10 gewählt werden kann. Mit ihr kann bestimmt werden, wie stark der HopCount erhöht werden soll. Dieser Aufschlag wird als *Penalty* P bezeichnet. Im Normalfalls berechnet ein Router, der eine Verbindung per RREP weitergibt den neuen HopCount H_{neu} wie folgt aus dem ihm bekannten Wert H_{alt}:

$$H_{neu} = H_{alt} + 1$$

In der angepassten Version wird in die Berechnung des neuen HopCounts H'_{neu} die Penalty P einbezogen, sofern der Router über einen begrenzten Energievorrat verfügt. Ferner kann in so einem Fall ein fester

Aufschlag B definiert werden, damit immer mit einer höheren Penalty agiert wird. Es gilt:

$$H'_{neu} = H_{alt} + 1 + P$$

wobei die Penalty folgendermaßen berechnet wird:

$$P = \lceil \frac{s}{C_i} + B \rceil$$

Bei $C(i) = 0{,}8$, $s = 2$ und $B = 0$ ergibt sich also $P = \lceil \frac{2}{0{,}8} + 0 \rceil = \lceil 2{,}5 \rceil = 3$ und somit $H'_{neu} = H_{alt} + 1 + 3 > H_{neu} = H_{alt} + 1$. Daraus folgt $H'_{neu} - H_{neu} = 3$, somit ist der Aufschlag durch den Router um 3 Hops höher. Er stellt sich also erheblich schlechter dar als er ist, damit andere Router ggf. andere Wege wählen sofern diese zur Verfügung stehen. Allerdings sollte s mit Bedacht gewählt werden, da der Wert bei hohem s und niedrigem $C(i)$ schnell steigt (er beträgt z. B. für $s = 10, C(i) = 0.1$ bereits 100) und die Anzahl maximaler Hops *einer gesamten Route* auf 255 begrenzt ist. Daher sollte s immer in Abhängigkeit der Größe des Netzes gewählt werden.

Die Grundeinstellungen des Protokolls sind in OMNeT++ nach den Empfehlungen der RFC gesetzt und so belassen.

4.1.5 Anpassung OLSR-PO (PowerOrientation)

Bei OLSR erfolgt die Anpassung auf ähnliche Art. Der Unterschied besteht allerdings in der Anwendung: Statt die Routen wie bei AODV zu unterbrechen, wird die *Willingness* der Hosts heruntergesetzt. Dies führt zwar zu einer gewissen Verzögerung bis die Routen geändert werden, allerdings entsteht keine Unterbrechung der Verbindungen. Stattdessen findet ein fließender Übergang statt. Daher ist OLSR durch geringeren *PacketLoss* im Vorteil. Auch hier gibt es, aus den gleichen Gründen, einen Trigger t, der wie bei AODV definiert ist. Der Grad der Veränderung, was erneut als *Sensitivity* s bezeichnet wird und zwischen $0{,}01$

und $0{,}99$ gewählt werden kann, bestimmt, wie stark die Willingness W herabgesetzt wird. Auch hier erfolgt die Anpassung nur, wenn der Router über einen begrenzten Energievorrat verfügt. Ferner kann in so einem Fall ein fester Aufschlag B definiert werden, damit immer mit einer höheren Penalty agiert wird. Es gilt:

$$W_{neu} = \lfloor \max(1, (7 \cdot C(i) \cdot (1 - s) - B)) \rfloor$$

Bei $C(i) = 0{,}8$, $s = 0{,}2$ und $B = 0$ ergibt sich also $W_{neu} = \lfloor \max(1, (7 \cdot 0{,}8 \cdot (1 - 0{,}2) - 0)) \rfloor = \lfloor \max(1, 4{,}48) \rfloor = 4$. Somit wird die Willingness auf den Wert 4 gesetzt. Hierdurch ist es möglich, dass Mitglieder der 1HNBs, die den Host als MPR nutzen, auf einen anderen Host mit höherer Willingness als MPR umstellen und somit der Verkehr über einen anderen Router geleitet wird, sofern dieser zur Verfügung steht. Auch hier sollte s mit Bedacht gewählt werden, da bei hohem s die Willingness sehr schnell sinkt.

Die Grundeinstellungen des Protokolls wurden in OMNeT++ für die angepasste Version folgendermaßen gesetzt:

- **HELLO-Interval**: $2s$
- **TC-Interval**: $5s$
- **MID-Interval**: $5s$
- **OLSR_REFRESH-Interval**: $2s$

In der ursprünglichen Version wird statt dessen ein HELLO- und OLSR_REFRESH-Interval von 0.5 Sekunden genutzt. Dies erhöht zwar den Overhead, reduziert aber den Paketverlust durch ausgefallene Router drastisch.

Kapitel 5

Auswertung

5.1 Eingesetze Technik

Leider verfügt die in OMNeT++ enthaltene Software nicht über die Möglichkeiten, aufgezeichnete Daten in hinreichender Tiefe oder großer Menge effizient auszuwerten und erzeugt zudem regelmäßig Abstürze und falsche Ergebnisse. Somit musste ein anderer Weg gewählt werden: Die von OMNet++ erzeugten Messdaten wurden mittels *scavetool*, das zum Lieferumfang gehört, nach *CSV* exportiert. Nach einer weiteren Aufbereitung der Daten durch *PERL*, wie dem Entfernen von nicht benötigten Dezimalstellen und der Reduzierung der Datenmenge durch einen Dropout, wurden diese mittels *GnuPlot* visualisiert. Zwecks besserer Lesbarkeit wird in diesem Kapitel der Punkt als Dezimaltrennzeichen verwendet.

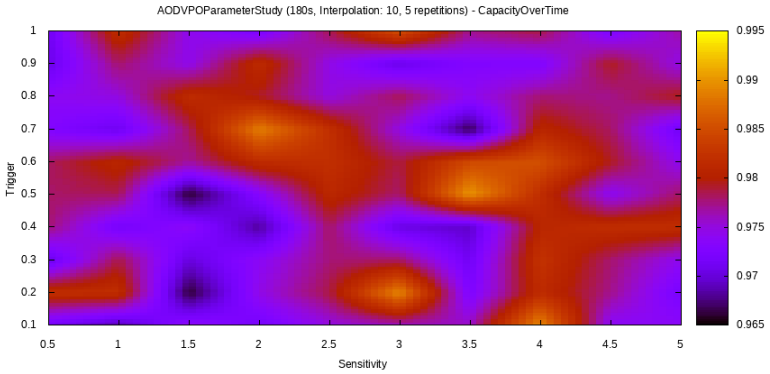

Abbildung 5.1: ParameterStudy AODV nach 180 Sekunden, Abweichung der Kapazität

5.2 Die Auswertung

5.2.1 Parameter Study

Wie zuvor beschrieben, gibt es bei beiden Protokollen die Möglichkeit die Stärke und Frequenz der Anpassung der Routingeigenschaften über Parameter zu steuern, namentlich Trigger t und Sensitivität s. Um hierfür geeignete Werte zu ermitteln, wurde eine *ParameterStudy* entworfen, die mögliche Kombinationen beider Werte in 5% Schritten des gültigen Intervalls in mehrfacher Wiederholung testet. Die Werte dazwischen wurden interpoliert. Grundsätzlich sehen Diagramme in 3D natürlich besser aus, dennoch sind sie für eine fundierte Analyse weniger hilfreich als *Heatmaps*. Daher werden im weiteren Verlauf nur Letztere gezeigt. Für alle im Rahmen der ParameterStudy gezeigten Diagramme gilt: Je höher der Wert, desto besser das Ergebnis.

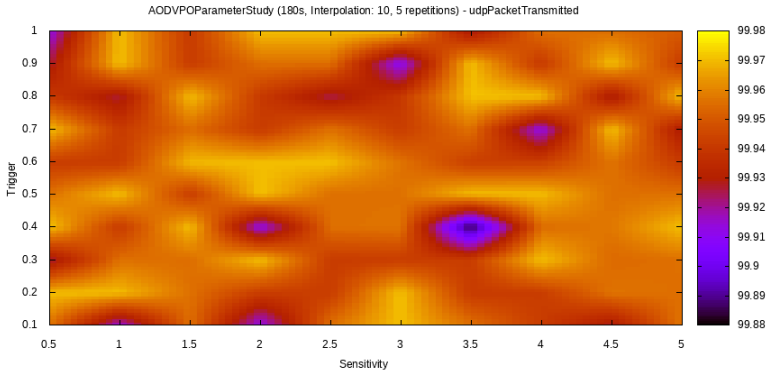

Abbildung 5.2: ParameterStudy AODV nach 180 Sekunden, PacketLoss

Parameter Study - AODV

In Abbildung 5.1 wird die Beziehung $1 - D(s,t)$ visualisiert, wobei D die *Standardabweichung der Kapazität* zum Ende der Simulation darstellt. Je heller der Wert, desto ähnlicher waren die Ladestände der Router. Es muss jedoch auch die Verlustrate betrachtet werden, denn was nützt ein ausgeglichener Verbrauch, wenn keine Daten mehr transportiert werden. Die Auswertung des *PacketLoss* L kann der Abbildung 5.2 entnommen werden, wobei hier wieder $1 - L(s,t)$ dargestellt wird. Wenn man beide Tests miteinander in Form der o. g. *Performance* kombiniert, dann ergibt sich daraus Abbildung 5.3. Aus den vorliegenden Daten können nun mehrere Parameterkombinationen ermittelt werden, die zu guten Ergebnissen führen müssten, also durch eine hohe Performance sowohl einen ausgeglichenen Ladezustand herstellen, als auch mit einem geringen PacketLoss auskommen. Für die weiteren Untersuchungen wurde das Tupel $(s,t) = (4, 0.3)$ gewählt. Die Auswertung des *Overhead* durch das Protokoll in Abbildung 5.4 zeigt, dass auch hier sehr gute Werte, die gewählte Kombination gehört zum besten

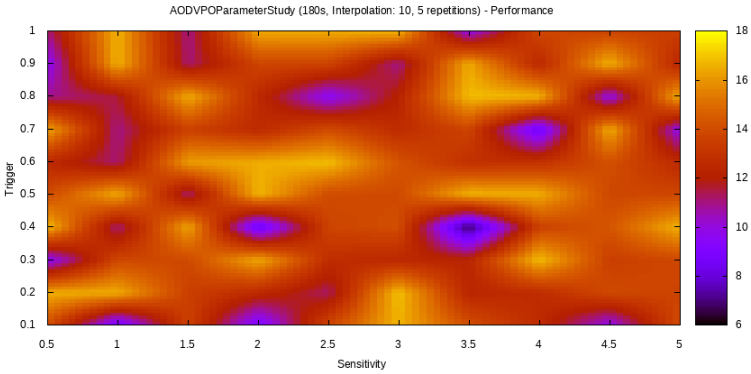

Abbildung 5.3: ParameterStudy AODV nach 180 Sekunden, Performance

Drittel, erreicht werden. Daher wurden die genannten Werte für fast alle weiteren Simulationen genutzt. Es liegen auch viele optimale Kombinationen in den Randbereichen, bei deren Verwendung kam es jedoch bei einigen Iterationen immer wieder zu ungewöhnlichen Abweichungen, die vermutlich auf ungünstige Zufallswerte zurückzuführen sind.

Parameter Study - OLSR

Hier wurden die selben Untersuchungen durchgeführt. Die Ergebnisse sind den Abbildungen 5.5, 5.6, 5.7 und 5.8 zu entnehmen. Es gelten die selben Aussagen wie bei AODV. Während jedoch bei AODV die Abweichung bis zum sechsfachen zwischen dem besten und schlechtesten Ergebnis beträgt, ist diese bei OLSR deutlich geringer, wie sich den Abbildungen entnehmen lässt. Es wurde das Tupel $(s, t) = (0.375, 0.3)$ gewählt, da es akzeptable Ergebnisse liefert.

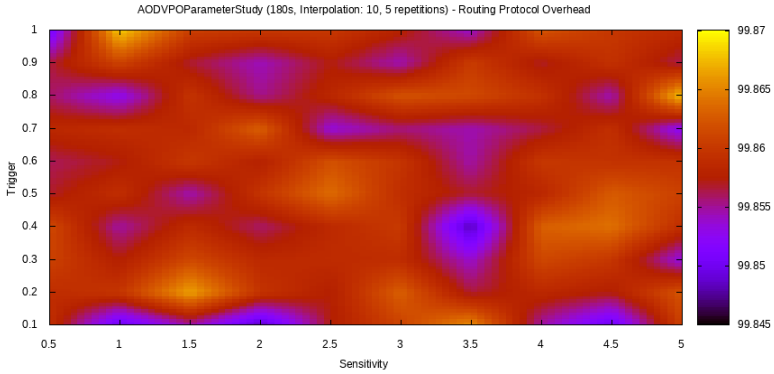

Abbildung 5.4: ParameterStudy AODV nach 180 Sekunden, Routing Overhead

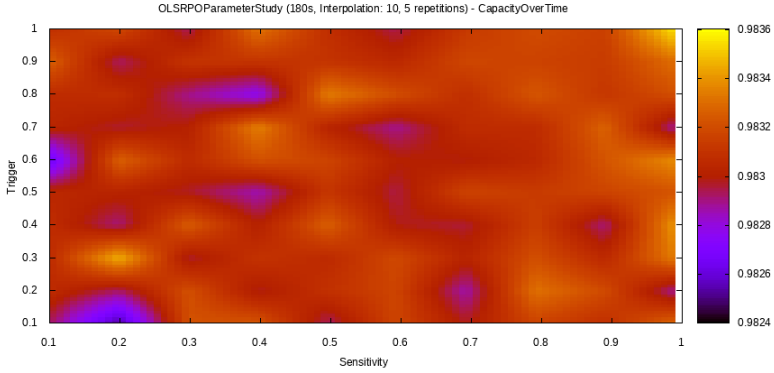

Abbildung 5.5: ParameterStudy OLSR nach 180 Sekunden, Abweichung der Kapazität

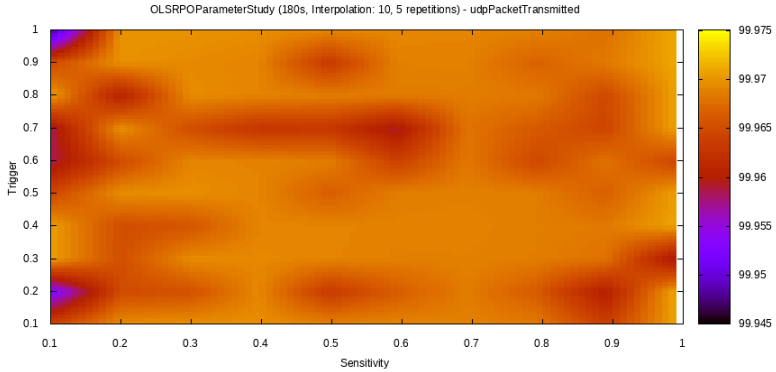

Abbildung 5.6: ParameterStudy OLSR nach 180 Sekunden, PacketLoss

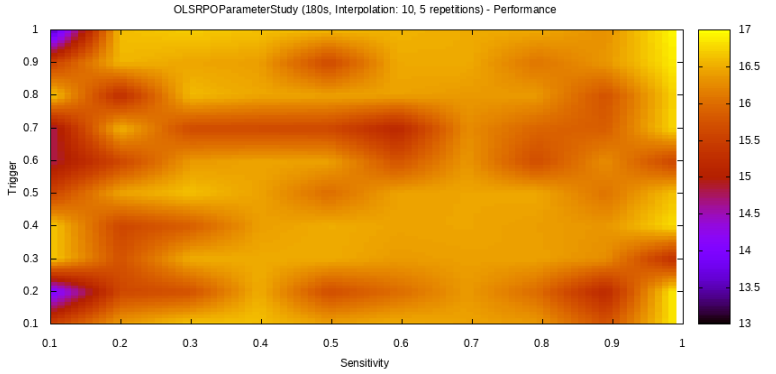

Abbildung 5.7: ParameterStudy OLSR nach 180 Sekunden, Performance

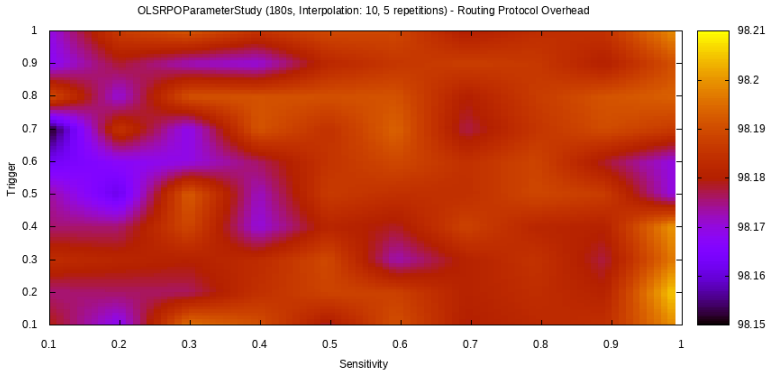

Abbildung 5.8: ParameterStudy OLSR nach 180 Sekunden, Routing Overhead

5.2.2 Auswertung AODV-PO

Zum Vergleich der Optimierung wurde zuerst ein Single-Hop Netz (vgl. Abbildung 4.5) getestet, das ausschließlich aus normalen AODV-Routern besteht. Alle Versuche wurden 100 mal mit verschiedenen Zufallszahlen wiederholt. Die Ladung der Energieversorgungen im Verlauf der Zeit ist in Abbildung 5.9 dargestellt. Wie man sieht, wird hier immer die komplette Energie eines Routers verbraucht. Anschließend schaltet der Router ab und der nächste Router wird genutzt. Nach der Anpassung stellt sich der Energieverbrauch wie in Abbildung 5.10 dar: Nachdem eine gewisse Menge Energie verbraucht wurde, wird ein RERR ausgelöst. Anschließend wird ein anderer Router genutzt, da er als günstigerer Pfad erkannt wird. So wird nach und nach immer wieder der Router gewechselt und der Energieverbrauch gleichmäßig verteilt.

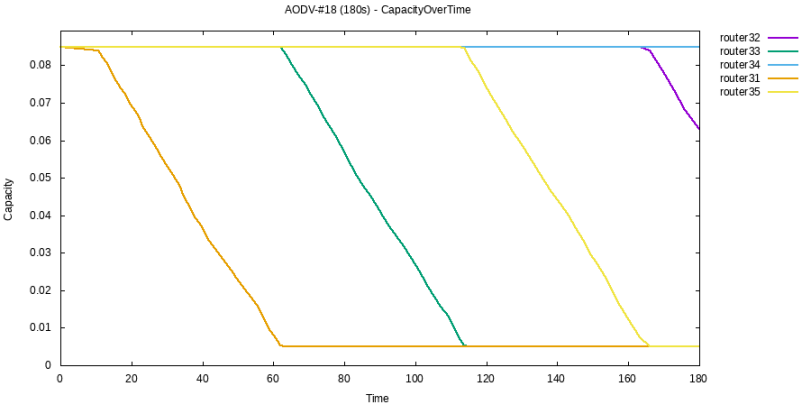

Abbildung 5.9: Verlauf Ladezustand bei AODV, 180 Sekunden

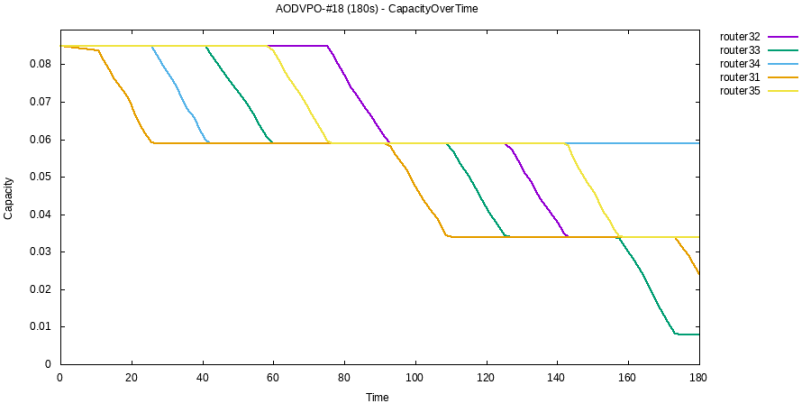

Abbildung 5.10: Verlauf Ladezustand bei AODV-PO, 180 Sekunden

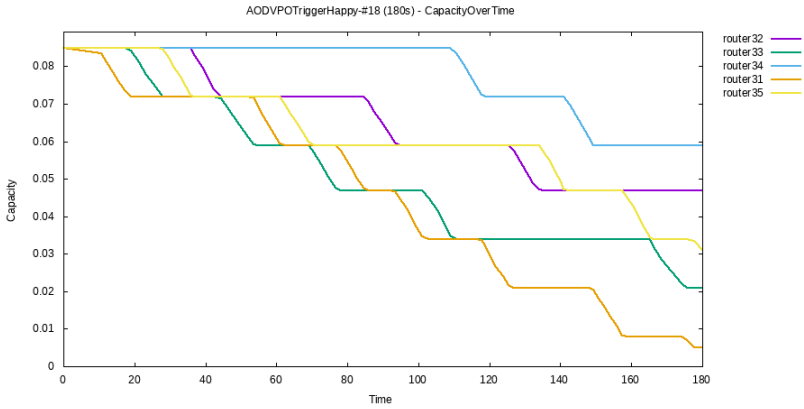

Abbildung 5.11: Niedrigerer Trigger $t = 0.2$ bei AODV-PO, 180 Sekunden

Um die Wirkung eines geänderten Triggers t zu testen, wurden zwei Versuche mit $t = 0.2$ (schnellere Reaktion) und $t = 0.4$ (langsamere Reaktion) durchgeführt. Wie man in den Abbildungen 5.11 und 5.12 erkennen kann, werden hier die Router öfter bzw. seltener gewechselt.

Zur Prüfung der Kompatibilität zwischen beiden Varianten des Protokolls wurde eine weitere Simulation durchgeführt. In dem Netz aus Abbildung 5.14 arbeiten die Router 22,23,24,42,43,44 mit normalem AODV, die anderen mit der angepassten Version. Es funktioniert grundsätzlich, jedoch wird der Verkehr nicht so gleichmäßig wie bei den anderen Untersuchungen verteilt (vgl. Abbildung 5.13). Auch ein Start mit unterschiedlichen Ladezuständen ist möglich. Der Verkehr wird so verteilt, dass sich diese angleichen.

Das Ziel, den Verkehr nach dem Energievorrat zu verteilen, kann als erreicht angesehen werden. Die erwünschte Kompatibilität zwischen neuem und altem Verfahren ist ebenfalls gegeben.

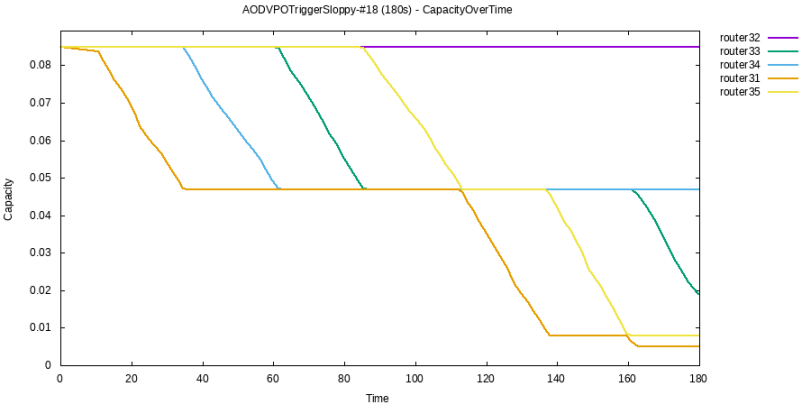

Abbildung 5.12: Höherer Trigger $t = 0.4$ bei AODV-PO, 180 Sekunden

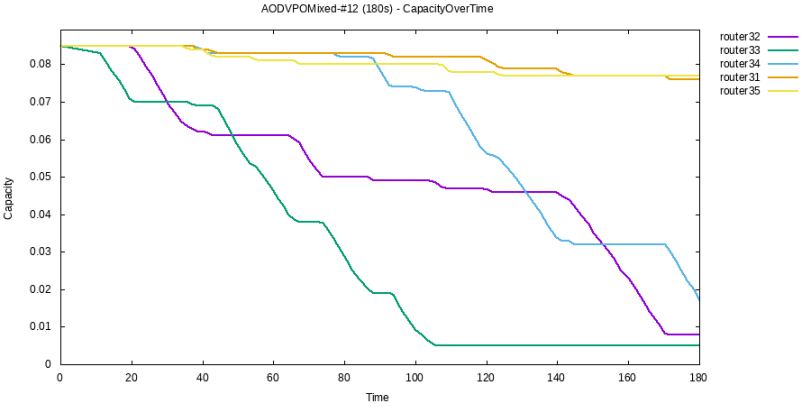

Abbildung 5.13: AODV und AODV-PO im Mischbetrieb, 180 Sekunden

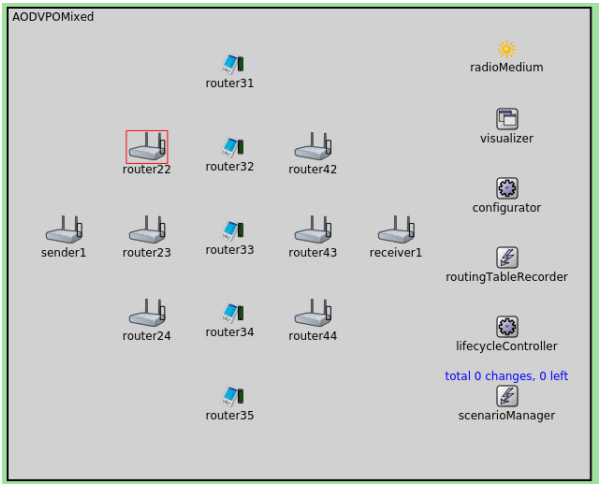

Abbildung 5.14: Netzwerk für den Mischbetrieb

5.2.3 Auswertung OLSR-PO

Für den Test der Anpassung bei OLSR wurden die selben Simulationen unter den selben Bedingungen durchgeführt. Die Ergebnisse zeigen das gleiche Verhalten wie bei AODV. Die grafischen Auswertungen sind hier zu Gunsten der Übersichtlichkeit nicht dargestellt, sondern dem Anhang A zu entnehmen. Das angestrebte Verhalten ist hier ebenfalls erreicht worden.

5.2.4 Vergleich Protokolle

Während die Abweichung der Ladezustände bei beiden Protokollen in der Standardversion vergleichbar hoch ist (Abbildung 5.15), erkennt man wie diese bei den angepassten Varianten im Durchschnitt stark abnimmt, bei OLSR sogar deutlicher. Bemerkenswert ist auch, dass die trägere Version besser als die normale ist, was jedoch mit der Laufzeit erklärbar ist (durch Zufall haben alle Ladezustände den selben Wert).

Bei kürzerer oder längerer Laufzeit ergeben sich andere Werte. Der stärkere Grad der Verteilung bei dem proaktiven Verfahren fordert allerdings höheren Overhead (Abbildung 5.16).

Der Paketverlust liegt bei den angepassten Varianten deutlich niedriger. Hier liegt OLSR zudem erheblich vor AODV, was durch den nahtlosen Wechsel erklärt werden kann (vgl. Abbildung 5.17 und 5.18). Der Ausreißer bei unangepasstem OLSR ist darauf zurückzuführen, dass die Router unkontrolliert abschalten, wodurch es erst eine Weile dauert, bis wieder neue Routen gewählt werden und in der Zwischenzeit Pakete verloren gehen.

5.2.5 Weitere Erkenntnisse

Die Simulation wurden mit hundert Wiederholungen durchgeführt. Es zeigt sich, dass die Anpassungen beider Protokolle stabil Ihre Wirkung entfalten. Auch hierzu sind im Anhang C entsprechende Auswertungen beigefügt. Es fällt auf, dass die Ergebnisse bei *OLSR-PO* stärker streuen, dennoch sind diese konstant niedriger als bei OLSR. Darüber hinaus wurde das Verhalten in einem größeren Netz (Abbildung 4.6) überprüft. Die Funktion ist dort ebenfalls gegeben, auch wenn es zu stärkeren Abweichungen kommt. *AODV-PO* verhält sich zum Ende der Simulation instabil, die Ursache hierfür konnte nicht abschließend geklärt werden. Hierzu sind entsprechende Auswertungen im Anhang B zu finden. Bei der letzten Untersuchung galt es, den Vorteil von proaktiven Protokollen bei vielen verschiedenen Zielen zu untersuchen. Hierzu wurden die Pakete zufällig auf alle Hosts im MultiHop-Netz verteilt. Hier zeigt OLSR ein leicht geringere Verlustrate von Paketen, da die Routen sofort zur Verfügung stehen und nicht erst ermittelt werden müssen, was zu Zeitüberschreitungen und damit zum Verwerfen der Pakete führt (Abbildung 5.19). Es ist anzunehmen, dass dieser Vorteil mit wachsender Größe des Netzes zunehmen wird.

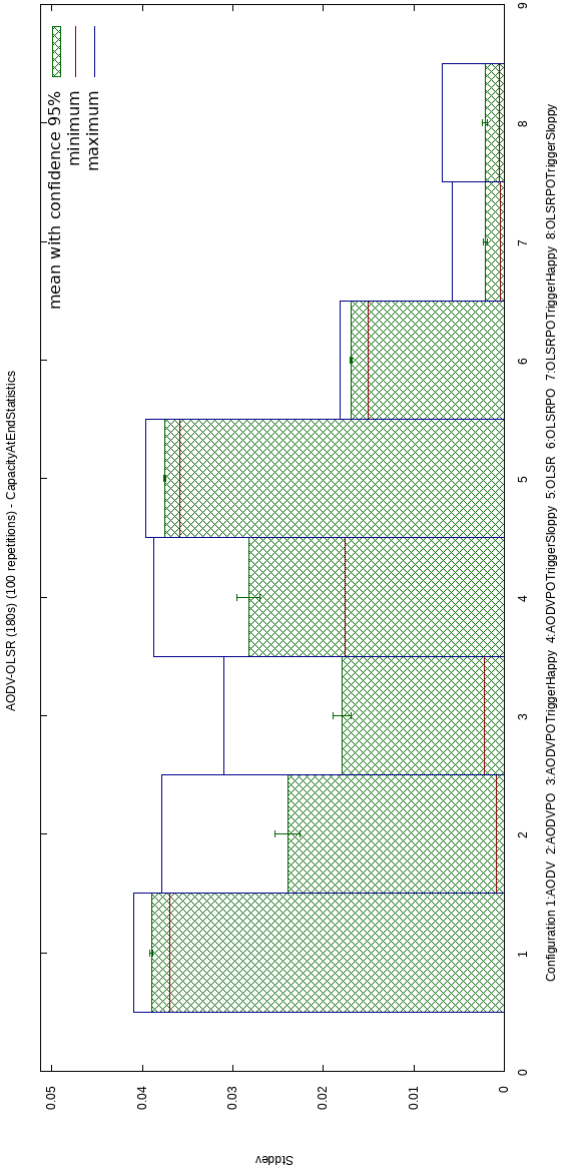

Abbildung 5.15: Vergleich Verlauf Ladezustand bei OLSR und AODV, 180 Sekunden

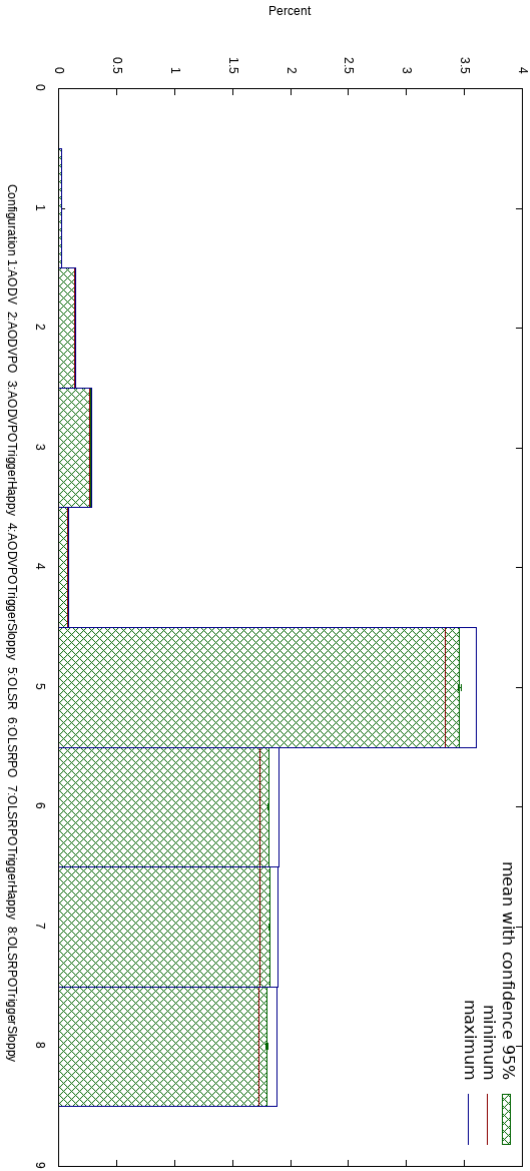

Abbildung 5.16: Vergleich Overhead bei OLSR und AODV, 180 Sekunden

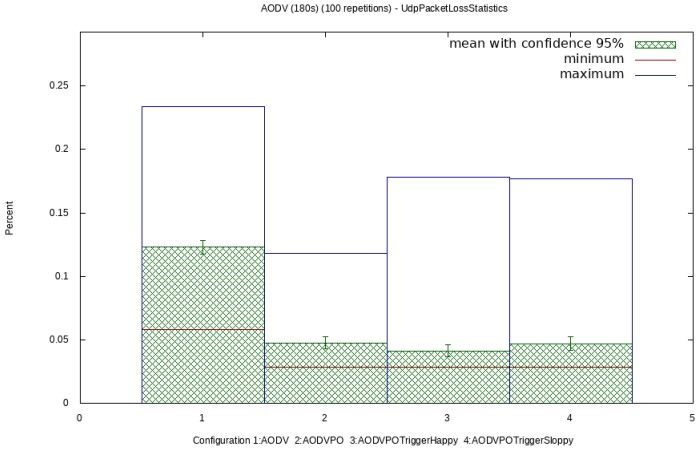

Abbildung 5.17: Vergleich PacketLoss bei AODV, 180 Sekunden

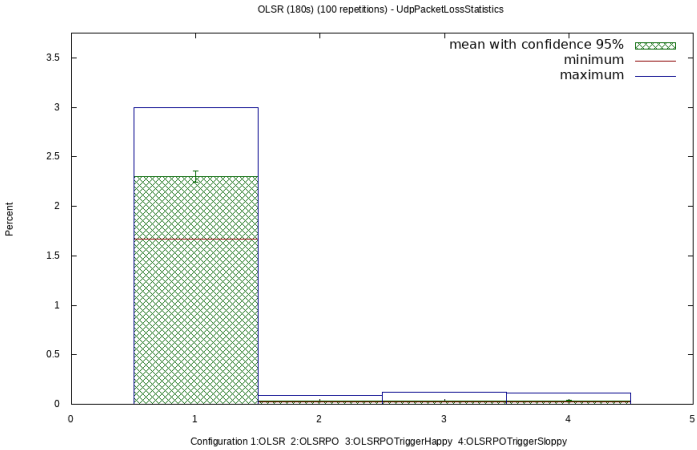

Abbildung 5.18: Vergleich PacketLoss bei OLSR, 180 Sekunden

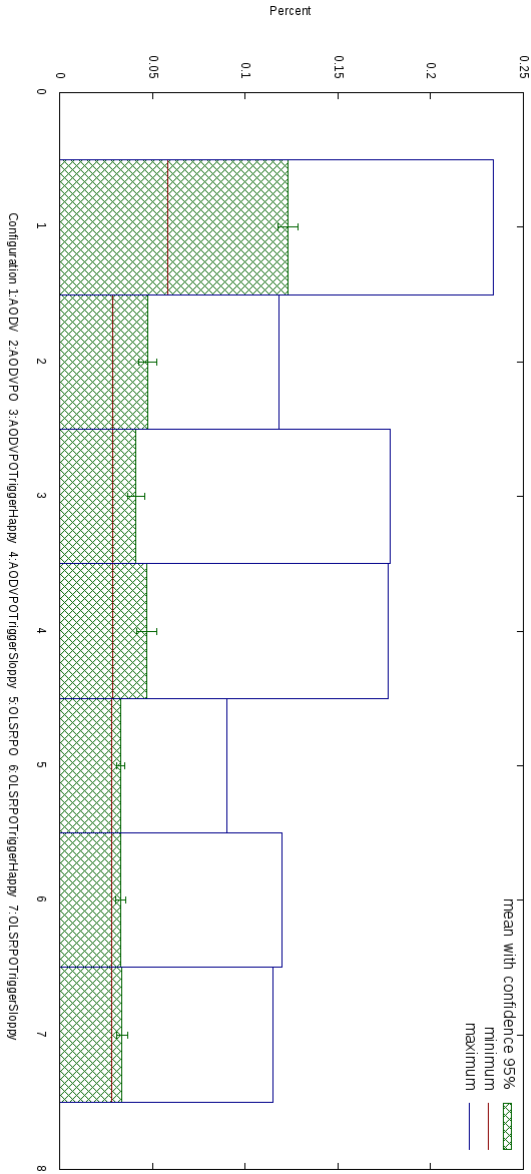

Abbildung 5.19: Vergleich Verlustrate Multi-Recipient bei OLSR und AODV, 180 Sekunden

Kapitel 6

Fazit

Es mag der Eindruck entstehen, dass die Auswertung ein wenig kurz geraten ist. Vielleicht liegt das aber auch daran, das es relativ gut funktioniert. Das gesteckte Ziel, eine faire Verteilung des Verkehrs zu erreichen, wurde, zumindest in den Kernexperimenten, erreicht. Die Funktion ist zudem in gemischten Netzen gegeben. Die Anpassung der Implementierung ist, sowohl im Simulator, als auch für echte Systeme relativ simpel gehalten, damit ein Test unter realen Bedingungen angebracht erscheint. Der Code beider Protokolle ist für den Linux-Kernel frei verfügbar [1] [2]. Es stellt sich allerdings die Frage ob es sinnig ist diese beiden sehr alten Protokolle weiter zu entwickeln. Stattdessen erscheint es als sinnvoll diese Anpassungen bei moderneren Verfahren wie *DANE* oder anderen auf den Stromverbrauch optimierten Protokollen zu testen. Wenn ein stromsparendes Verfahren zusätzlich noch den Verkehr fair verteilt, dann kann diese Technologie bei dem Aufbau moderner Ad-Hoc-Infrastrukturen durchaus zu höherer Akzeptanz für den Einsatz auf mobilen Geräten führen. Ich halte es persönlich für sehr realistisch,

[1] https://github.com/erimatnor/aodv-uu
[2] https://wiki.freifunk.net/Glossar#OLSR

dass MESH in kommenden Mobilfunkgenerationen eine erhebliche Rolle spielen wird. Nicht zuletzt weil es den Providern einen wirtschaftlichen Vorteil bringt.

Anhang A

Auswertungen OLSR

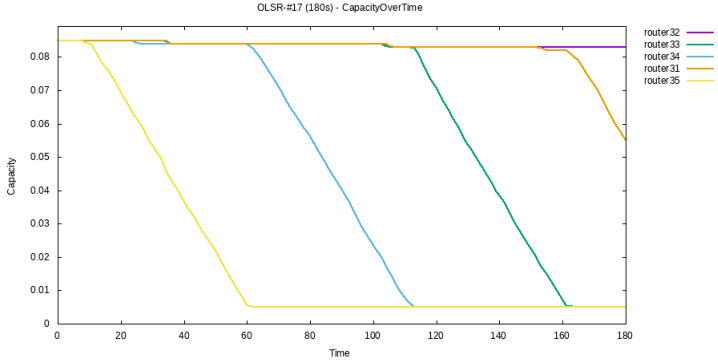

Abbildung A.1: Verlauf Ladezustand bei OLSR, 180 Sekunden

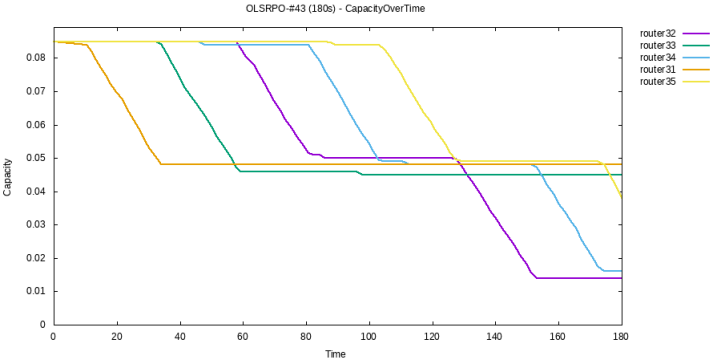

Abbildung A.2: Verlauf Ladezustand bei OLSR-PO, 180 Sekunden

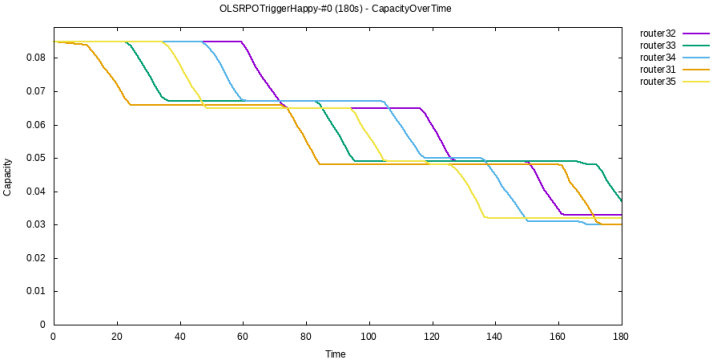

Abbildung A.3: Niedrigerer Trigger $t = 0.2$ bei OLSR-PO, 180 Sekunden

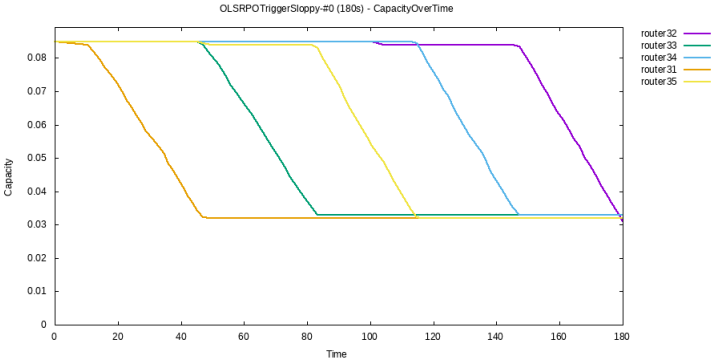

Abbildung A.4: Höherer Trigger $t = 0.4$ bei OLSR-PO, 180 Sekunden

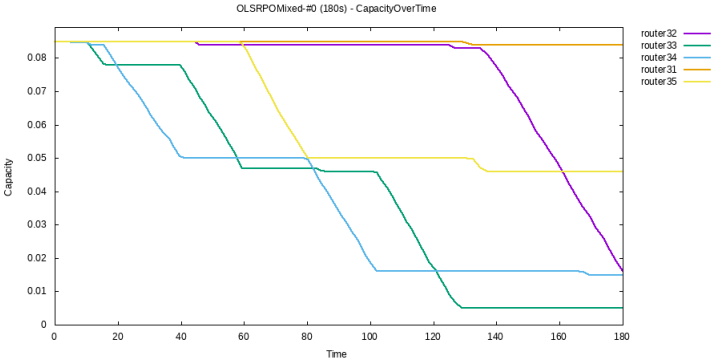

Abbildung A.5: OLSR und OLSR-PO im Mischbetrieb, 180 Sekunden

Anhang B

Multi Hop

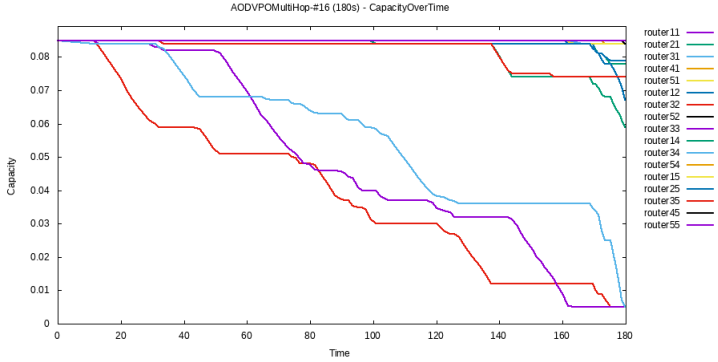

Abbildung B.1: Verlauf Ladezustand bei AODV-PO im MutiHop Netz, 180 Sekunden

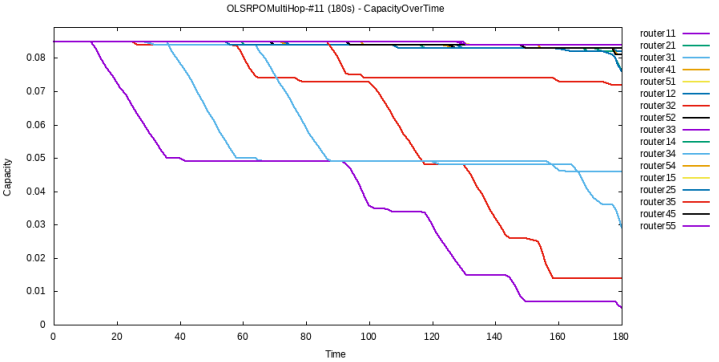

Abbildung B.2: Verlauf Ladezustand bei OLSR-PO im MutiHop Netz, 180 Sekunden

Anhang C

Wiederholungen

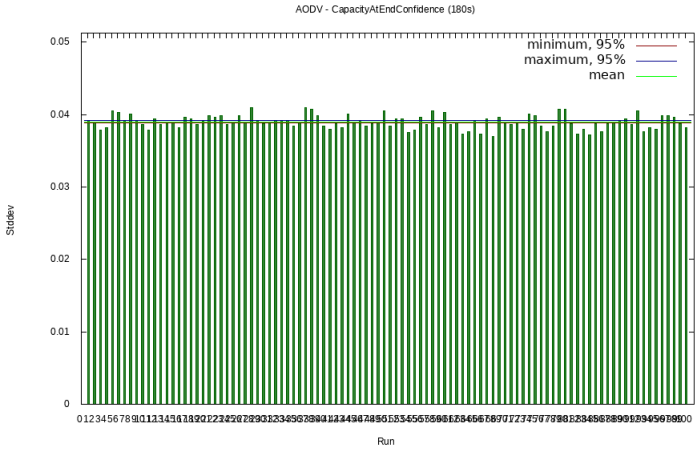

Abbildung C.1: Analyse Durchläufe AODV, 180 Sekunden

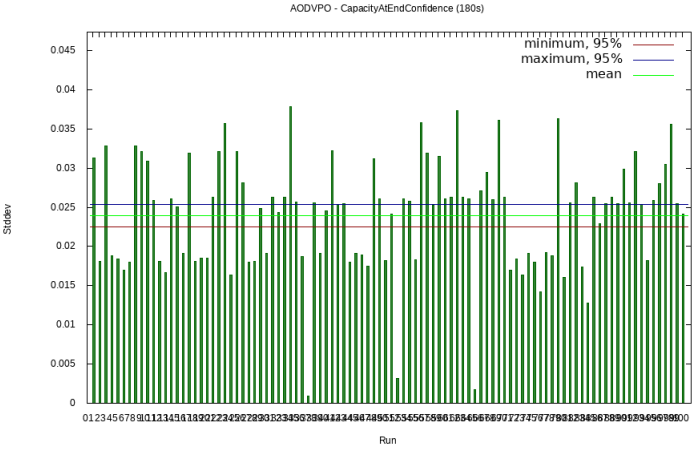

Abbildung C.2: Analyse Durchläufe AODV-PO, 180 Sekunden

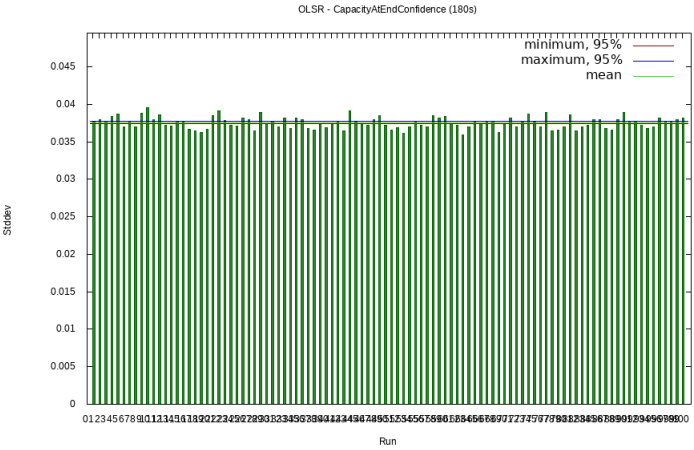

Abbildung C.3: Analyse Durchläufe OLSR, 180 Sekunden

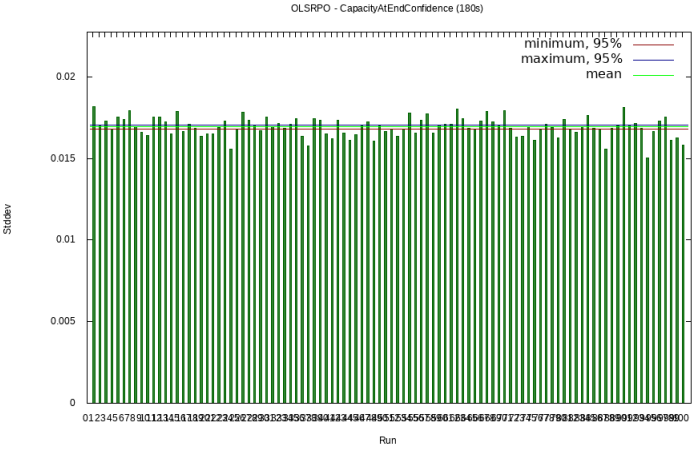

Abbildung C.4: Analyse Durchläufe OLSR-PO, 180 Sekunden

Anhang D

3D Diagramme

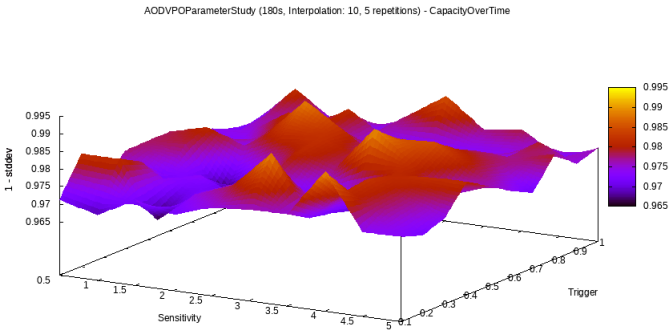

Abbildung D.1: 3D Darstellung Ladezustand ParameterStudy bei AODV, 180 Sekunden

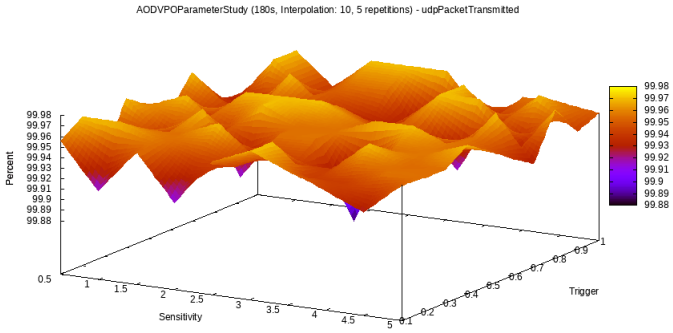

Abbildung D.2: 3D Darstellung PacketLoss ParameterStudy bei AODV, 180 Sekunden

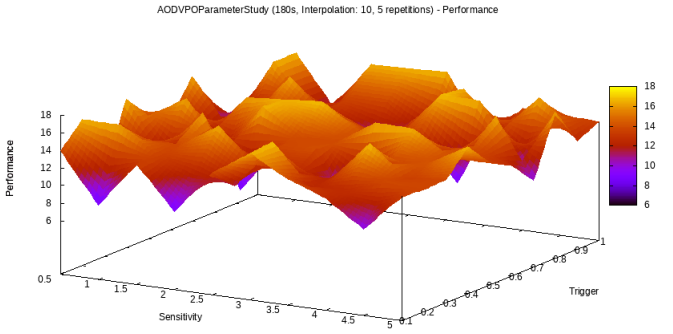

Abbildung D.3: 3D Darstellung Performance ParameterStudy bei AODV, 180 Sekunden

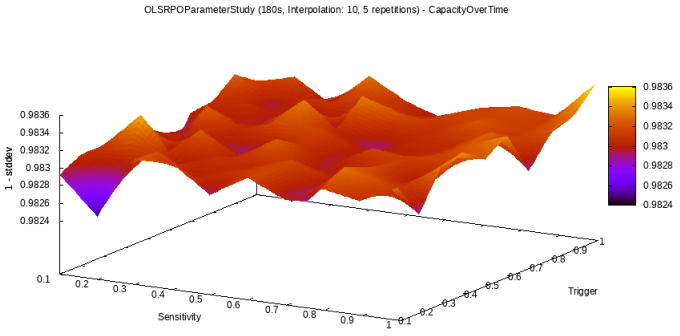

Abbildung D.4: 3D Darstellung Ladezustand ParameterStudy bei OLSR, 180 Sekunden

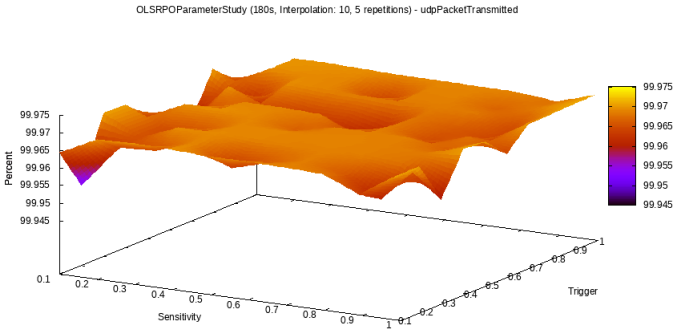

Abbildung D.5: 3D Darstellung PacketLoss ParameterStudy bei OLSR, 180 Sekunden

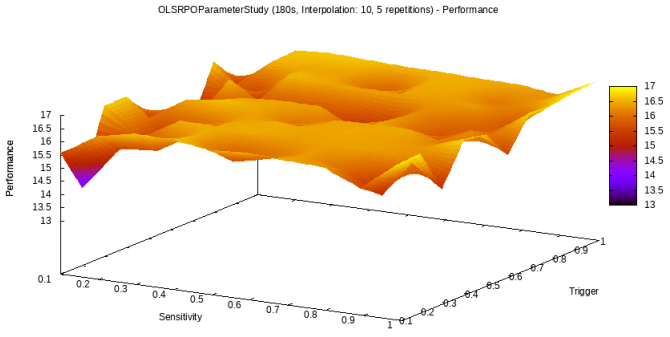

Abbildung D.6: 3D Darstellung Performance ParameterStudy bei OLSR, 180 Sekunden

Akronyme

1HNB One hop neighbour. 36, 37, 39–41, 57

2HNB Two hop neighbour. 40

AODV Ad-hoc On-demand Distance Vector. 2–4, 9, 12, 15, 16, 21, 22, 27, 32, 34, 43, 49, 52, 54, 56, 62, 67, 69, 70

AP AccessPoint. 11

BCA Broadcast Address. 13

BSS Basic Service Set. 11

DFW default forwarding algorithm. 36, 39, 41

DS Distribution System. 11

DSDV Destination-Sequenced Distance Vector. 21

ESS Extended Service Set. 11

HELLO message Hello message. 36, 37, 40, 41

HOP Netzwerkabschnitt. 2

IBSS Independent Basic Service Set. 11

IP Internet-Protocol. 4, 9, 11, 12

IPv4 Internet-Protocol V4. 12, 13

IPv6 Internet-Protocol V6. 12

MANET mobiles Ad-Hoc Netzwerk. 7, 8, 15, 46

MESH Vermaschtes Netz. 2, 46, 76

MID message Multiple interface declaration message. 35–37, 41

MPR Multipoint Relay. 33, 36, 37, 39–42, 57

NWA Network Address. 13

OLSR Optimized Link State Routing. 2–4, 9, 12, 16, 17, 21, 32, 33, 35, 36, 42, 43, 49, 52, 54, 56, 62, 69, 70

RERR Route Error. 29, 30, 55, 65

RFC Request for Comments. 4, 49

RREP Route Reply. 24, 26, 27, 29, 55

RREQ Route Request. 22–24, 26, 27, 29

RV Routingverfahren. 2, 4, 8, 11–14, 16, 17, 52

S2HNB Strict two hop neighbour. 40

TC message Topology control message. 37, 39, 41

TCP/IP TCP/IP Protocol-Stack. 9, 13

TCP Transmission Control Protocol. 9, 13

UCA Unicast Address. 13

UDP User Datagram Protocol. 9, 13, 22, 34

WLAN Wireless LAN. 4, 10, 50

WMN Wireless Mesh Network. 3, 4, 7

WSAN Wireless Sensor and Actor Network. 2, 3, 16, 32

Glossar

AccessPoint Gemeinsamer Zugangspunkt für die Teilnehmer innerhalb eines BSS im Infrastruktur-Modus. 11, 91

Ad-hoc On-demand Distance Vector Ein Verfahren zum Weiterleiten von Daten durch ein mobiles Ad-hoc-Netz. Das Protokoll gehört zu den topologiebasierten reaktiven Routingverfahren. Routen zu bestimmten Zielen werden erst bei Bedarf ermittelt. Das Protokoll wird in RFC 3561 beschrieben. 3, 4, 9, 12, 15, 21, 22, 27, 32, 34, 43, 49, 52, 54, 56, 62, 67, 69, 70, 91

Ad-Hoc-Modus Ein Modus für ein Funknetzwerk, in dem die Teilnehmer direkt miteinander kommunizieren. 10, 11

Basic Service Set Fundamentale Einheit eines Funknetzes bei WLAN. Es definiert eine Gruppe von Teilnehmern, die eine gemeinsame Koordinationsfunktion nutzen. 11, 91

Border Gateway Protocol Das im Internet eingesetzte Routingprotokoll. 17

Broadcast Address Eine IP-Adresse, die den Broadcast innerhalb eines Subnetzes bezeichnet. 91

default forwarding algorithm Der Standard-Algorithmus für die Weiterleitung von Nachrichten bei OLSR. 39, 41, 91

Destination-Sequenced Distance Vector Ein einfaches, auf dem Distanzvektoralgorithmus basierendes Routingverfahren. 91

Distribution System Ein implementationsunabhängiges System, dass mehrere BSS im WLAN Infrastruktur-Modus miteinander verbindet. 11, 91

Extended Service Set Ein Verbund mehrerer BSS über ein DS im WLAN Infrastruktur-Modus. Ein ESS kann über ein Portal an andere Netzwerke angeschlossen sein. 91

Exterior Gateway Protocol Routingprotokolle, die grundsätzlich für Verbindung geschlossener Netze, sogenannte Autonome Systeme, gedacht sind. 17

GPL Die GNU General Public License (kurz GNU GPL oder GPL) ist die am weitesten verbreitete Softwarelizenz, die einem gewährt die Software auszuführen, zu studieren, zu ändern und zu verbreiten (kopieren). 3, 49

Hello message Informationen über die Interface Adressen eines OLSR Hosts. 36, 37, 40, 41, 91

Independent Basic Service Set Ein BSS, dass von den Teilnehmern innerhalb eines WLAN Ad-Hoc Netzes aufgespannt wird. 11, 91

Infrastruktur-Modus Ein Modus für ein Funknetzwerk, in dem die Teilnehmer über einen zentralen Accesspoint kommunizieren. 10, 11

Interior Gateway Protocol Routingprotokolle, die grundsätzlich für den Einsatz innerhalb geschlossener Netze, sogenannte Autonome Systeme, gedacht sind. 16

Internet-Protocol Das Internet Protocol ist ein in Computernetzen weit verbreitetes Netzwerkprotokoll und stellt die Grundlage des Internets dar. Es ist die Implementierung der Internetschicht des TCP/IP-Modells bzw. der Vermittlungsschicht des OSI-Modells. IP ist ein verbindungsloses Protokoll, das bedeutet bei den Kommunikationspartnern wird kein Zustand etabliert. 9, 11, 12, 92

Internet-Protocol V4 Die derzeit dominante Version des Internet Protocols mit TCP in der Version 4. Es kommen Netzwerkadressen mit einer Länge von 4 mal 8 Bit zum Einsatz. 12, 13, 92

Internet-Protocol V6 Eine neue Version des Internet Protocols, die derzeit aufgrund der Knappheit verfügbarer IPv4-Adressen eingeführt wird. Es kommen Adressen mit einer Länge von 8 mal 16 Bit zum Einsatz. 92

Metrik Im Netzwerkbereich definiert die Metrik ein numerisches Maß für die Güte einer Verbindung bei Verwendung einer bestimmten Route. 2–4, 17

mobiles Ad-Hoc Netzwerk MESHs, die sich selbständig aufbauen und konfigurieren, nennt man auch mobile Ad-hoc-Netze oder MANET. 7, 8, 15, 46, 92

Multiple interface declaration message Informationen über die Adressen verschiedener Schnittstellen eines OLSR Hosts. 36, 37, 41, 92

Multipoint Relay Ein Host in einem OLSR Netz, der die Verteilung von Routinginformationen übernimmt. Jeder Teilnehmer bestimmt die Liste seiner MPRs selbst. 36, 37, 39–42, 57, 92

Network Address Eine IP-Adresse, die ein Subnetz bezeichnet. 92

Netzwerkabschnitt Ein Hop ist ein Netzwerkabschnitt und wird als Zähleinheit benutzt. Die Anzahl an Hops sagt, über wie viele Netzwerk-Abschnitte die Datenpakete übertragen wird. Solche Netzwerk-Abschnitte können durch Router oder andere Knotenpunkte definiert sein. 91

OLSR Message Die Kontrollinformationen bei OLSR, die als Nutzlast der OLSR Pakete verschickt werden. 34, 35

One hop neighbour Ein direkter Nachbar eines OLSR Hosts. 36, 37, 39–41, 57, 91

Open Shortest Path First Ein von der IETF entwickeltes Link-State-Routing-Protokoll. 17

Optimized Link State Routing Ein Routingprotokoll für mobile Adhoc-Netze, das eine an die Anforderungen eines mobilen drahtlosen LANs angepasste Version des Link State Routing darstellt. Das Protokoll wird in dem RFC 3626 beschrieben. 3, 4, 9, 12, 16, 21, 32, 33, 35, 36, 42, 43, 49, 52, 54, 56, 62, 69, 70, 92

Request for Comments Request for Comments - eine Reihe technischer und organisatorischer Dokumente des RFC-Editors zum Internet (ursprünglich Arpanet), die am 7. April 1969 begonnen wurden. 49, 92

Route Der gesamte Weg, den ein Paket durch ein Netzwerk wählt. 2, 3

Route Error Die Meldung über die Nichtverfügbarkeit einer Route durch einen AODV Router. 29, 30, 55, 65, 92

Route Reply Eine Antwort auf die Routenanforderung eines AODV Routers. 26, 27, 29, 55, 92

Route Request Eine Routenanforderung eines AODV Routers. 23, 24, 26, 27, 29, 92

Router Netzwerkteilnehmer, der Pakete für andere Teilnehmer weiterleitet. 2

Routing Information Protocol Ein Routing-Protokoll auf Basis des Distanzvektoralgorithmus, das innerhalb eines autonomen Systems (z. B. LAN) eingesetzt wird, um die Routingtabellen von Routern automatisch zu erstellen. 16, 17

Routingverfahren Ein Verfahren nach dem bestimmt wird, wie eine Nachricht zum Ziel geleitet wird und wie dieser Weg ermittelt wird. 2, 8, 11, 13, 52, 92

Sicherungsschicht Die Sicherungsschicht soll die korrekte Übertragung von Frames auf Schicht 2 des OSI Modells zwischen zwei miteinander verbundenen Systemen bewerkstelligen. 11, 50

Smartphone Ein Smartphone ist ein Mobiltelefon (umgangssprachlich Handy), das erheblich umfangreichere Computer-Funktionalitäten und -konnektivität als ein herkömmliches Mobiltelefon zur Verfügung stellt. 2, 3

Strict two hop neighbour Ein direkter Nachbar des Nachbarn eines OLSR Hosts, ohne den Host selbst. 40, 92

TCP/IP Protocol-Stack Eine Sammlung diverser Protokolle wie UDP, TCP uvm. für den Einsatz mit dem Internet-Protocol. 9, 13, 92

Topology control message Informationen über die erreichbaren Nachbarn eines OLSR Hosts. 39, 41, 92

Transmission Control Protocol Eine Familie von Netzwerkprotokollen. Sie wird wegen ihrer großen Bedeutung für das Internet auch als Internetprotokollfamilie bezeichnet. Die Identifizierung der am Netzwerk teilnehmenden Rechner geschieht über IP-Adressen. 9, 13, 92

Two hop neighbour Ein direkter Nachbar des Nachbarn eines OLSR Hosts. 40, 91

Unicast Address Eine IP-Adresse, die einen Teilnehmer bezeichnet. 92

User Datagram Protocol Ein minimales verbindungsloses Netzwerkprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. 9, 13, 22, 34, 93

Vermaschtes Netz Ein Netz, das zwei oder mehr Endgeräte zu einem vermaschten Netz verbindet. 46, 76, 92

Vermittlungsschicht Die Vermittlungsschicht sorgt bei leitungsorientierten Diensten für das Schalten von Verbindungen und bei paketorientierten Diensten für die Weitervermittlung von Datenpaketen. 11, 12, 14

Wireless LAN Drahtloses Netzwerk auf Basis von IEEE 802.11. 10, 50, 93

Wireless Mesh Network Vermaschtes, drahtloses Netzwerk auf Basis von WLAN. 4, 93

Wireless Sensor and Actor Network Ein Netzwerk aus intelligenten Sensoren und Aktoren, die geografisch verteilt und über ein WLAN miteinander verbunden sind. 3, 93

Abbildungsverzeichnis

Literaturverzeichnis

[ALF04] Arun Avudainayagam, Wenjing Lou, and Yuguang Fang. Dear: A device and energy aware routing protocol for heterogeneous ad hoc networks. *Journal of Parallel and Distributed Computing*, 63(2):228–236, Februar 2004.

[BCGS04] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic. *IEEE 802.11 AD HOC Networks: Protocols, Performance, and Open Issues*, page 416ff. WileyIEEE Press, Piscataway, NJ, USA, 2004.

[BHSW07] Rainer Baumann, Simon Heimlicher, Mario Strasser, and Andreas Weibel. A survey on routing metrics. TIK Report 216, Computer Engineering and Networks Laboratory, ETH-Zentrum, Zürich, Schweiz, Februar 2007.

[Bra89] Robert Braden. Requirements for internet hosts - communication layers. STD 3, RFC Editor, Oktober 1989. `http://www.rfc-editor.org/rfc/rfc1122.txt`.

[BSC13] Sourangsu Banerji and Rahul Singha Chowdhury. On ieee 802.11: Wireless lan technology. *International Journal of Mobile Network Communications and Telematics*, 3(4):45–64, August 2013.

[BTA+11] Azzedine Boukerchea, Begumhan Turgut, Nevin Aydin, Mohammad Ahmad, Ladislau Bölöni, and Damla Turgut. Routing protocols in ad hoc networks: A survey. *Computer Networks*, 55(13):3032–3080, September 2011.

[BTL13] Apidet Booranawong, Wiklom Teerapabkajorndet, and Chusak Limsakul. Energy consumption and control response evaluations of aodv routing in wsans for building-temperature control. *Sensors — Open Access Journal*, 13(7):8303–8330, Juli 2013.

[CFML08] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. Ospf for ipv6. RFC 5340, RFC Editor, Juli 2008. `http://www.rfc-editor.org/rfc/rfc5340.txt`.

[CH10] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *USENIX '10 Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, page 21ff., New York, NY, USA, Juni 2010. ACM.

[CJ03] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). RFC 3626, RFC Editor, Oktober 2003. `http://www.rfc-editor.org/rfc/rfc3626.txt`.

[CM99] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. RFC 2501, RFC Editor, Januar 1999. `http://www.rfc-editor.org/rfc/rfc2501.txt`.

[CWKS97] B.P. Crow, I. Widjaja, J.G. Kim, and P.T. Sakai. Ieee 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–126, September 1997.

[Hed88] C. Hedrick. Routing information protocol. RFC 1058, RFC Editor, Juni 1988. `http://www.rfc-editor.org/rfc/rfc1058.txt`.

[HQG+12] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *MobiSys '12 Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 225–238, New York, NY, USA, Juni 2012. ACM.

[JNA08] David Johnson, Ntsibane Ntlatlapa, and Corinna Aichele. A simple pragmatic approach to mesh routing using batman. In *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, Pretoria, Südafrika, Oktober 2008. WCITD.

[KAASYS13] Nitin Kumar, Parvez Ahmed Alvi, Ajay Singh Yadav, and Anupam Swami. A study of routing protocols for ad-hoc network. *International Journal of Application or Innovation in Engineering and Management*, 2(6):154–159, Juni 2013.

[LZL11] Erwu Liu, Qinqing Zhang, and K.K. Leung. Clique-based utility maximization in wireless mesh networks. *IEEE Transactions on Wireless Communications*, 10(3):948–957, März 2011.

[PBRD00] C. Perkins, E. Belding-Royer, and S. Das. Aodv for ipv6. Draft draft-perkins-aodv6-01, IETF, November 2000. `https://tools.ietf.org/id/draft-perkins-aodv6-01.txt`.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. RFC 3561, RFC Editor, Juli 2003. http://www.rfc-editor.org/rfc/rfc3561.txt.

[Pos81] Jon Postel. Internet protocol. STD 5, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc791.txt.

[RL95] Yakov Rekhter and Tony Li. A border gateway protocol 4 (bgp-4). RFC 1771, RFC Editor, März 1995. http://www.rfc-editor.org/rfc/rfc1771.txt.

[SK91] Theodore John Socolofsky and Claudia Jeanne Kale. Tcp/ip tutorial. RFC 1180, RFC Editor, Januar 1991. http://www.rfc-editor.org/rfc/rfc1180.txt.

[SWR98] Suresh Singh, Mike Woo, and C.S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98 Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, New York, NY, USA, Oktober 1998. ACM.

[TET16] Mohammad Tawalbeh, Alan Eardley, and Loai Tawalbeh. Studying the energy consumption in mobile devices. *Procedia Computer Science*, 94:183–189, August 2016.

[Zim80] Hubert Zimmermann. Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, April 1980.