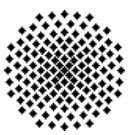
Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Kurt Rothermel
Betreuer:	Dipl.-Inf. Jing Tian
Begonnen am:	30.09.02
Beendet am:	30.04.03
CR-Nummer:	C.2.1, C2.2

Diplomarbeit-Nr.: 2058

Konzeption und Implementierung eines ortsbezogenen Ad-Hoc Routing Protokoll für die Linux Plattform

Richard Klinger



Universität Stuttgart

Institut für Parallele und Verteilte Systeme

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Gleichungsverzeichnis

1. Einleitung

In der modernen Gesellschaft spielt die Mobilität eine tragende Rolle. Deshalb existieren bereits weltweit über 500 Millionen Autos und die Anzahl wächst von Tag zu Tag. Ein ausgebautes und dichtes Straßennetz sichert den reibungslosen Ablauf des alltäglichen Personen- und Gütertransports. Leider kommt es trotzdem zu Verkehrsbehinderungen und sogar zu tödlichen Unfällen. Insbesondere in Ballungsräumen und auf Autobahnen führt das hohe Verkehrsaufkommen zu chronisch verstopften Straßen, was sich in den Hauptverkehrszeiten in kilometerlangen Staus widerspiegelt. Die immer länger und häufiger werdenden Staumeldungen deuten auf keine Entspannung in den nächsten Jahren hin. Tatsächlich sprechen einige Experten von einem drohenden Verkehrsinfarkt. Durch „intelligente" Fahrzeuge könnten die schlechten Bedingungen auf den Straßen entschärft werden, indem sie die Fahrer so sicher und schnell wie möglich durch den Verkehr zu ihren gewünschten Zielen leiten. Das „intelligente“ Auto unterstützt den Fahrer, indem es die aktuelle Situation auf den Straßen analysiert und mit den anderen „intelligenten“ Fahrzeugen kommuniziert. Dadurch ist das Auto in der Lage kritische Situationen zu erkennen und den Fahrer zu warnen bzw. andere Fahrzeuge auf die kritische Situation hinzuweisen. Somit wird die Sicherheit im Straßenverkehr erhöht.

Kommt es dennoch zu einem schweren Unfall, dann zählt jede Sekunde, um Leben zu retten. In einigen Fällen werden die Rettungsmannschaften durch den Verkehr behindert und können nur schwer zur Unfallstelle vordringen, da die Verkehrsteilnehmer die Rettungsmannschaften nicht schnell genug erkennen und dementsprechend nicht reagieren können. Ein Warn- und Informationssystem könnte solche Situationen vermeiden oder zumindest entschärfen, indem die Autofahrer über das Eintreffen der Rettungsmannschaften informiert werden.

Zusätzlich könnten die „intelligenten“ Autos den Fahrkomfort erhöhen, indem sie die menschliche Kommunikation zwischen Insassen in unterschiedlichen Fahrzeugen ermöglichen, selbst wenn diese sich ausserhalb der Funkreichweite befinden. Solche Autos sind daher in der Lage, Nachrichten weiterzuleiten bis sie ihren Empfänger erreichen. Diesen Vorgang nennt man *Multi-Hop Routing*. Daraus ergibt sich ein Kommunikationsnetzwerk, welches von Fahrzeugen aufgespannt und autonom verwaltet wird. Solche Kommunkationsnetzwerke werden in dieser Diplomarbeit als *Vehicular* MANET[a] bezeichnet.

a. Mobile Ad-Hoc Network [MANET]

1.1. Aufgabenbeschreibung

Die Aufgabe dieser Diplomarbeit besteht aus dem Entwurf und der Implementierung einer Plattform für eine Fahrzeug-Fahrzeug Kommunikation, die in das europäische Projekt „CarTALK 2000“ eingebunden werden kann. Die Fahrzeug-Fahrzeug Kommunikation basiert auf einem mobilen *Ad-Hoc* Netzwerk, welches allein durch Fahrzeuge auf den Straßen aufgespannt wird (*Vehicular* MANET). Das Netzwerk benötigt deshalb keine vorhandenen Netzwerke mit Infrastruktur.

Multi-Hop Ad-Hoc Routing Protokolle sind besonders gut für einen MANET geeignet. Trotzdem können sie nicht ohne weiteres in einem *Vehicular* MANET eingesetzt werden. Deshalb sollen vorhandene *Routing* Konzepte untersucht werden, ob sie die charakterlichen Besonderheiten eines mobilen *Ad-Hoc* Netzwerk aus Fahrzeugen berücksichtigen und den Anforderungen genügen.

Verschiedene Implementierungsstragien von vorhandenen *Routing* Protokolle sollen erläutert und bewertet werden. Anschließend soll ein *Position-based Multi-Hop Ad-Hoc Routing* Protokoll für die Linux 2.4 *Distribution* implementiert und auf einem Laptop getestet werden. Das Laptop ist mit einer IEEE 802.11b Funktnetzwerkkarte und einem GPS Empfänger ausgerüstet.

1.2. Motivation

Anwendungen, die Nachrichten an andere Autos schicken müssen, basieren auf einem Funknetzwerk von Fahrzeugen. Mögliche Netzwerklösungen wurden in der Diplomarbeit von Markus Legner [Le02] ausgiebig untersucht. Es wurde gezeigt, dass aufgrund der charakteristischen Eigenschaften ein MANET für Fahrzeuge am besten geeignet ist. In einem MANET können die mobilen Endgeräte nach dem „Einklicken“ im ganzen Netzwerk sich frei bewegen. Diese Unabhängigkeit führt leider auch zu großen Problemen beim *Routing*, da Fahrzeug-Netzwerke sehr stark partitioniert werden können und damit das Weiterleiten von Nachrichten schwieriger wird. Ein *Map-based Geographic Forwarding* (MGF) Protokoll wurde in [Le02] vorgestellt, das für ein *Vehicular* MANET sehr gut geeignet ist, um ein effizientes Weiterleiten von Nachrichten zu ermöglichen. Das MGF Protokoll verwendet dazu Straßeninformationen aus einer digitalen Landkarte, um Nachrichten an Fahrzeuge zu schicken, die für die Weiterleitung der Nachrichten am besten geeignet sind oder sie solange speichert bis

sich ein geeignetes Fahrzeug im Sendebereich befindet. Somit erhöht sich die Wahrscheinlichkeit einer erfolgreichen Nachrichtenzustellung.
Mit Hilfe des MGF Protokolls ist eine Kommunikation zwischen Fahrzeugen bzw. über mehrere Fahrzeuge hinweg möglich. Wünschenswert wäre dabei ein ähnliches Verhalten des *Vehicular* MANET wie bei festverdrahteten Netzwerken. Das Funknetzwerk sollte so konzipiert sein, dass es sich an das TCP/IP-Referenzmodell anlehnt und sich in die Linux-Architektur nahtlos einbindet. Dadurch können langjährig bewährte Konzepte und vorhandene Netzwerk-Technologien übernommen werden. Somit ist es möglich, in kürzester Zeit ein effizientes und stabiles *Vehicular* MANET aufzubauen. Je näher sich das MFG Protokoll an dem TCP/IP Referenzmodell anlehnt, desto einfacher haben es später die Entwickler Anwendungen für das *Vehicular* MANET zu entwerfen, zu implementieren und zu testen. Bereits vorhandene Kommunikationssoftware müsste nur geringfügig verändert werden, um es an die MGF-Protokoll Schnittstellen anzupasssen. Die Linux-Architektur bietet eine ideale Vorraussetzung für eine prototypische Implementierung eines Routing Protokolls aufgrund der „Open Source“ Eigenschaft.

Das von der Europäischen Kommission initiierte Projekt "CarTALK 2000" versucht, die oben genannten Ziele zu erreichen und gleichzeitig einen Standard für Fahrzeugkommunikationssystem zu schaffen, das den Fahrer sicher und bequem durch den Verkehr leitet. Das dafür notwendige extrem flexible *Ad-Hoc* Funknetzwerk für Fahrzeuge wird in dieser Diplomarbeit ausgearbeitet und implementiert, damit es für zukünftige Fahrassistenten als Kommunikationsplattform dienen kann.

1.3. Struktur der Diplomarbeit

Fachausdrücke, die in dieser Diplomarbeit verwendet werden, sind in englischer Sprache und kursiv gehalten und werden nach Möglichkeit nicht ins Deutsche übersetzt. Im “Abkürzungsverzeichnis” (siehe Seite 103) werden alle Abkürzungen, die in dieser Diplomarbeit verwendet werden aufgelistet. Die Vorgehensweise für die Implementierung eines Routing Protokolls für das *Vehicular* MANET sollen soweit wie möglich den Methoden des modernen Software Engineerings entsprechen, um am Ende eine möglichst effiziente und stabile Software

zu erhalten. Dazu soll das Wasserfall-Modell, das in die einzelnen Phasen (Analyse, Entwurf, Implementierung und Test) unterteilt ist, angewendet werden.

In Kapitel 2 wird das „CarTALK 2000“ Projekt vorgestellt. Danach erfolgt in Kapitel 3 eine Darstellung vorhandener Arbeiten über *Beaconing* Protokoll, *Location Service* Protokoll und Implementierungen von *Routing* Protokollen. Anschließend wird in Kapitel 4 eine Anforderungsanalyse an den beiden Protokollen und der Implementierung durchgeführt. Die Anforderungen werden dann für die Bewertung herangezogen. Danach erfolgt in Kapitel 5 die Konzeption für das *Beaconing* und *Location Service* Protokoll, sowie die Implementierung für den *Vehicular Routing Mechanism* (VRM). In Kapitel 6 werden die wichtigsten Klassen vom VRM anhand eines Diagramms dargestellt und erläutert. Nach der Implementierung erfolgt in Kapitel 7 ein Test über die Leistung des *Vehicular Routing Mechanism*. Eine Zusammenfassung und Ausblick über das *Routing* Protokoll in Kapitel 8 schließt die Diplomarbeit ab.

2. Einführung

In einer kurzen Einführung über das „CarTALK 2000“ Projekt in Kapitel 2.1 werden die vorhandenen Komponenten und Strukturen im einem „CarTALK 2000“ Fahrzeug beschrieben.
Anschließend erfolgt in Kapitel 2.2 die Darstellung der benötigten Hardwarekomponenten für ein *Vehicular* MANET. Die Verknüpfung der einzelnen Hardwarekomponenten wird durch die Software in Kapitel 2.3 "Softwarekomponenten" realisiert. Sie wird nur kurz für das allgemeine Verständnis erläutert, da in den nachfolgenden Kapiteln die einzelnen Softwarekomponenten ausführlich dargestellt werden.
In Kapitel 2.4 "Kommunikationsarchitektur" erfolgt eine Darstellung der Anordnung und Struktur von den Hardware- und Softwarekomponenten. Die Funktionsweisen der Komponenten werden anhand eines kleinen Beispiels beschrieben.

2.1. CarTALK 2000

Wie schon bereits in Kapitel 1.2 "Motivation" erwähnt, soll das europäische Projekt „CarTALK 2000“ eine Plattform für eine Fahrzeugkommunikation schaffen und gleichzeitig einen zukünftigen Standard für Fahrerassistentensysteme festlegen. Das Ziel dieses Projektes ist ein sicheres und komfortableres Fahren aufgrund einer *Inter-Vehicle* Kommunikation zu erreichen .
Das Projekt wurde im August 2001 von der Europäischen Kommission initiiert und endet im August 2004 [CAR00].

2.1.1. CarTALK 2000 Systemarchitektur

Die „CarTALK 2000“ Systemarchitektur besteht in erster Linie aus einem verteilten Computersystem. Mit der AMI-C[a] Architektur kann das System mit zusätzlichen Komponenten erweitert werden. In Abbildung 1 ist die „CarTalk 2000“ Architektur der Komponenten grob abgebildet.

OEM Gateway
Das OEM Gateway versteckt alle Implementierungsaspekte der *In-Car* OEM[b] Netzwerke und liefert eine offene Schnittstelle nach außen, um Daten an die *In-Car* OEM Netzwerke zu senden bzw. zu empfangen.

a. Automotive Multimedia Interface Collaboration
b. Original Equipment Manufacturer

Funkkommunikation und Ortsbestimmung
Die "Funkkommunikation und Ortsbestimmung" Komponente ist zuständig für die Kommunikation im *Vehicular* MANET und der Positionserkennung des Fahrzeugs.

Telematik und Fahrzeugsteuerung
Es gibt zwei verschiedene Anwendungsformen: Die Anwendung für Telematik und die Anwendungen für Fahrzeugsteuerung. Anwendung in der Fahrzeugsteuerung werden in Echtzeit ausgeführt.

In-Car OEM Netzwerke
Die *In-Car* Kommunikation wird über ein offenes Netzwerk abgewickelt, wobei die Bandbreite hier von langsamen Netzwerken bis zu Hochgeschwindigkeitsnetzwerken reicht.

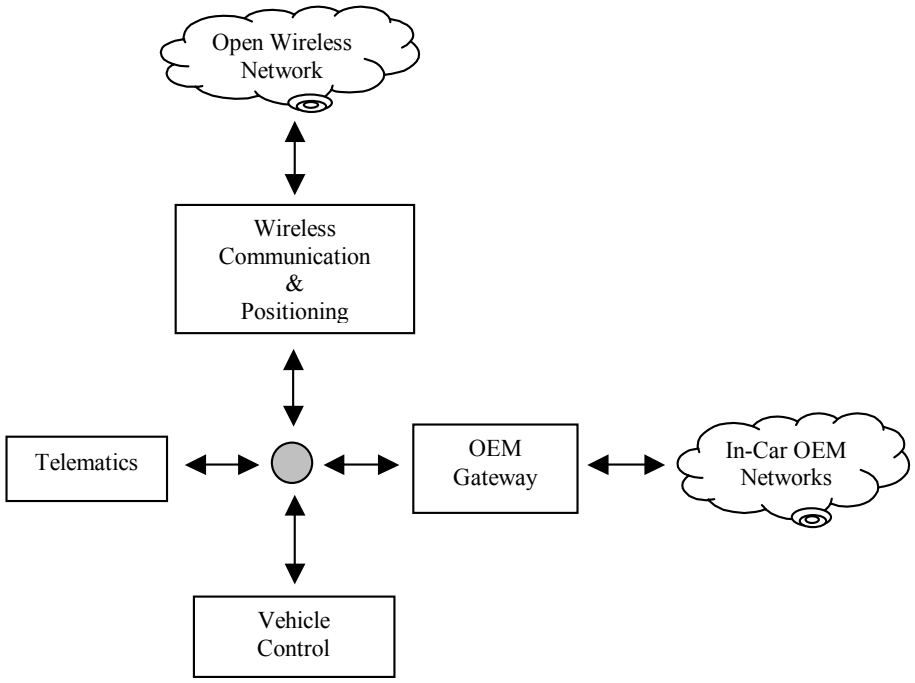


Abbildung 1: CarTALK 2000 Architektur

Der in dieser Diplomarbeit implementierte *Vehicular Routing Mechanism* (VRM) kann später nahtlos in die „CarTALK 2000“ Komponente "Funkkommunikation und Ortsbestimmung" eingebunden werden. Nähere und genauere Information über das „carTALK“ 2000 Projekt und über dessen Architektur kann in [CAR00] nachgelesen werden.

2.1.2. CarTalk 2000 Anwendungen

Anwendungskonzepte für das *Vehicular* MANET, die für Sicherheit und Fahrkomfort sorgen sollen werden in [CAR00] entwickelt.

In den folgenden Abschnitten werden drei typische Szenarien dargestellt, in denen der Fahrer durch eine carTALK Anwendung gewarnt, entlastet oder unterstützt wird. Die Anforderungen an die Anwendung werden kurz erläutert und zusammengefasst dargestellt. Weitere Informationen können in [CAR00] nachgelesen werden.

Szenario: Warnungen und Informationen an den Fahrer

Eine „CarTALK 2000“ Anwendung muss bei Unfällen Warnungen an andere Fahrzeuge schicken, damit der nachfolgende Verkehr rechtzeitig reagieren kann. Eine solche Situation ist in Abbildung 2 dargestellt. Die Anwendung muss den entgegenkommenden Verkehr informieren, um eventuelle Auffahrunfälle an unübersichtlichen Stellen zu verhindern.

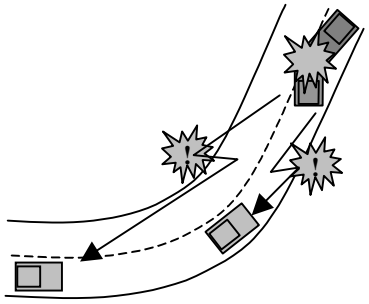


Abbildung 2: Szenario: Unfall auf einer Landstrasse

CarTALK ordnet das Szenario in den Bereich *Information and Warning Functions* (IWF) ein.

Die Nachrichten für eine IWF-Anwendung müssen kurz, mit hoher Priorität und geringer Verzögerungszeit durch einen *Broadcast* an andere Fahrzeuge übermittelt werden, die den Empfang bestätigen. Um Warnungen und Informationen schnellstmöglich übermitteln zu können, soll die Sendereichweite bis zu 1000 Meter betragen und es muss die Möglichkeit bestehen, durch *Multi-Hop* Übertragung die Nachrichten in ein größeres Gebiet zu verteilen.

Szenario: Entlastung für den Fahrer

Eine Anwendung kann den Fahrer entlasten, indem sie ihn über die aktuelle Situation der vorderen Fahrzeuge informiert. Der Fahrer ist dann in der Lage, möglichst vorrausschauend zu fahren, ohne dass er Sichtkontakt zu allen vorderen Fahrzeugen haben muss. Gerade beim abrupten Bremsen reagieren die Fahrer in den hinteren Autos zu langsam. Durch Unachtsamkeit des Fahrers kann es dann zum Auffahrunfall kommen.

In Abbildung 3 ist eine *Stop-and-Go* Situation dargestellt, die den Fahrer durch das ständige Anfahren und Bremsen unter Stress setzt. Wenn Auto B dem Fahrer C die Sicht versperren würde, könnte Auto C erst bremsen, wenn Auto B bremst. Wenn Auto A das Bremsen an Auto B und C senden würde, könnte sich der Fahrer in Auto C auf ein Bremsen vorbereiten und würde somit nicht vom plötzlichen Bremsen von Auto B überrascht.

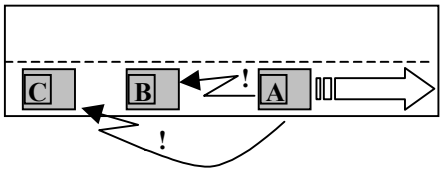


Abbildung 3: Szenario: Stop-and-Go

Die Nachrichten, die nur für die Fahrzeuge in der selben Fahrbahn bestimmt sind, fallen unter die Kategorie: *Communication-based Longitudinal Contro*l (CBLC). CBLC-Nachrichten sind mittellang, haben normale Priorität und werden in kurzen periodischen Intervallen mit *Broadcast* oder *Multicast* gesendet. Die Nachrichten müssen bis zu 500 Meter weit gesendet und durch *Multi-Hop* Übertragungen schnell an weitere Nachbarn geleitet werden können. Eine Empfangsbestätigung ist dabei nicht notwendig, was Bandbreite und Zeit spart.

Szenario: Unterstützung des Fahrers

Eine fahrerunterstützende Anwendung hilft dem Fahrer in kritischen Situationen den Überblick im Straßenverkehr nicht zu verlieren, indem sie ihm die notwendigen Informationen liefert, die er zur Bewältigung der kritischen Situation benötigt.

Ein Auto in einen fließenden Verkehr einzufädeln, besonders in Autobahnauffahrten, wäre beispielsweise eine kritische Situation für manche Autofahrer.

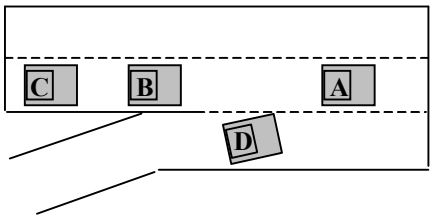


Abbildung 4: Szenario: Autobahnauffahrt

Anwendungen, die den Fahrer in solchen Situationen unterstützen, werden zu *Co-operative Driver Assistance* (CODA) eingeordnet. Es gibt bis zu drei verschiedene CODA-Nachrichtentypen mit unterschiedlichen Prioritäten (Notfall-, Assistent- und Informationsnachrichten). Alle CODA-Nachrichten sind mittellang und müssen mit geringer Verzögerung durch *Unicast* mit Empfangsbestätigung übermittelt werden. Die Sendereichweite sollte bis zu 500 Meter betragen. Eine *Multihop*-Übertragung ist nicht notwendig, es sollte aber die Möglichkeit vorhanden sein, Nachrichten mit *Broadcast* zu versenden.

	IWF	**CBLC**	**CODA**
Funkreichweite	1000m	500m	500m
Weiterleitung	Ja	Ja	Nein
Broadcast	Ja	Ja	Ja
Unicast	Nein	Nein	Ja
Priorität	Ja	Nein	Ja
Wiederholungsrate	1-10-25-50	1-10-20-50	20-50
Maximale Verzögerung	40-100ms	100ms	100ms
Nachrichtenlänge	Kurz	Mittel	Mittel

Tabelle 5: Anforderungen an das Kommunikationssystem

Die Anforderungen an das Kommunikationssystem für die drei Anwendungen sind in Tabelle 5 zusammengefasst.

2.2. Hardwarekomponenten

Die Hardwarekomponenten, stellen die physikalischen Komponenten dar, die für das Senden bzw. Empfangen von Nachrichten und Ermitteln der eigenen Position zuständig sind. Sie wurden in die „Wireless Communikation & Positioning“ Komponente in der Abbildung 1 auf Seite 6 eingefügt.

2.2.1. Funktechnologie

Fahrzeugnachrichten werden mit Hilfe von Funkgeräten gesendet bzw. empfangen. Das „CarTALK 2000“ Projekt sieht für die Übertragung von Nachrichten eine IEEE 802.11b Karte für ein prototypischen *Vehicular* MANET vor [WF01]. In Tabelle 6 werden die wichtigsten Eigenschaften einer IEEE 802.11b Karte aufgelistet.

Funkreichweite	Bandbreite	Netzwerk-topographie	Frequenz-bereich	Zugriffs-kontrolle
160m bis 550m[a]	1 Mbit/s bis 11 Mbit/s	BSS[b] / Ad-Hoc	2,4 GHz	CSMA/CD

Tabelle 6: Eigenschaften einer IEEE 802.11b Karte

a. siehe Tabelle 20
b. Basic Station Set

IEEE 802.11b Standard

Die Funkreichweite einer IEEE 802.11b Karte variiert je nach Einsatzort und verwendeter Bandbreite zwischen 160 Meter bis 550 Meter. Die Relation zwischen Funkreichweite und Bandbreite befindet sich weiter hinten in Tabelle 20 auf Seite 43. In der Netzwerktopographie unterscheidet man zwischen dem BSS und dem *Ad-Hoc* Modus. BSS Mode wird für Netzwerke mit Infrastruktur verwendet, die für ein *Vehicular* MANET aber eher ungeeignet sind, da die Teilnehmer sich bei den Basisstationen anmelden müssen bevor sie Nachrichten versenden können. Daher werden alle IEEE 802.11b Karten im *Ad-Hoc* Mode betrieben, damit eine Kommunikation zwischen den Teilnehmern im *Vehicular* MANET möglich ist, ohne sich vorher am Netzwerk anmelden zu müssen.

Der IEEE 802.11b Standard verwendet den CSMA/CA um Zugriff auf das Medium zu bekommen. Das bedeutet, dass kein *Quality of Service* (QoS) zur Verfügung steht. Nachrichten

werden mit der DSSS (Direct Sequence Spread Spectrum) Methode gesendet, um unempfindlicher gegenüber Interferenzen bei der Übertragung zu werden.

2.2.2. Positionierungssystem

Um Nachrichten erfolgreich weiterleiten zu können, muss das Fahrzeug seine eigene Position kennen. Dafür wird ein Positionierungssystem benötigt. Das Positionierungssystem für ein *Vehicular* MANET muss weltweit funktionieren. Satellitennavigationssysteme sind dafür sehr gut geeignet. Sie sind in der Lage, den Standort innerhalb einer kurzen Zeit bis auf wenige Meter genau zu ermitteln. Dabei spielen Umwelteinflüsse keine Rolle. In vielen Anwendungen haben sie sich schon bewährt, besonders bei Flug- und Fahrzeugnavigation.

GPS

Das GPS wurde von der U.S. Department of Defense entwickelt und aufgebaut. Es ist seit 1980 für den zivilen Gebrauch zugänglich und kann seit Mai 2000 uneingeschränkt d.h. ohne künstliche Verfälschung der Position (SA) verwendet werden. Das Verteidigungsministerium der USA behält sich aber das Recht vor in Krisenregionen die Benutzung von GPS für zivile Zwecke einzuschränken [GPS1].

Genauigkeit

Die Genauigkeit des GPS ist ausschlaggebend für ein erfolgreiches Weiterleiten von Nachrichten an Fahrzeuge. Wenn die aktuelle Position zu ungenau ist, dann könnten Nachrichten unnötiger- oder irrtümlicherweise weitergeleitet werden. Im schlimmsten Fall erreichen die Nachrichten ihren Zielort überhaupt nicht. Je höher die Genauigkeit, desto effektiver können die Nachrichten weitergeleitet werden. Das bedeutet weniger Kommunikationsverkehr im *Vehicular* MANET. Mit den heutigen im Handel erhältlichen GPS-Empfängern kann eine Genauigkeit von bis zu 15 Meter erreicht werden. Diese reicht aus, um Nachrichten an andere Fahrzeuge erfolgreich weiterzuleiten. Eine Aussage auf welcher Spur sich das Fahrzeug bei einer mehrspurigen Straße bzw. Autobahn befindet, ist aber aufgrund der ungenauen GPS Position nicht möglich.

DGPS

Das differentielle GPS (DGPS) kann die Genauigkeit der Positionsbestimmung mit GPS erhöhen, dazu werden zusätzlich zu den Satellitensignalen noch Korrekturdaten benötigt. Diese werden von einer Referenzstation erzeugt, die an einem genau vermessenen Punkt einen Soll/Ist-

Vergleich durchführt. Ein DGPS-fähiger GPS-Empfänger ist dann in der Lage eine genauere Positionsberechung durchzuführen [GPS2].

Genauigkeit
Die Genauigkeit liegt im Bereich von 1-3 Metern und ist unter anderem abhängig von der Distanz der Korrekturdatensender und der Signalqualität. Diese Genauigkeit reicht aus, um eine Aussage machen zu können, auf welcher Spur sich das Fahrzeug bei einer mehrspurigen Straße befindet. Da heutzutage kein genaueres Positionierungssystem erhältlich ist, das weltweit eingesetzt werden kann, soll das GPS bzw. DGPS verwendet werden.

Dead Reconing
Kann die Position des Fahrzeugs aufgrund einer Abschattung der Satellitensignalen (Tunnel, hohe Häuser usw.) nicht bestimmt werden, ist es möglich für eine kurze Zeit die Position anhand der Geschwindigkeit und Fahrtrichtung des Fahrzeugs zu berechnen.

2.2.3. Kommunikationsrechner

Das Herzstück für das *Routing* stellt der Kommunikationsrechner (Router) dar. Dazu ist ein Laptop vorgesehen, der die folgenden zwei Eigenschaften für das erfolgreiche *Routing* im *Vehicular* MANET mitbringt.

Leistung
Die Leistung des Latops muss ausreichend sein, um Nachrichten auch bei sehr hohem Verkehrsaufkommen innerhalb kürzester Zeit zu empfangen, auszuwerten und wenn nötig an andere Fahrzeuge weiterzuleiten. Ebenso können verschiedene Programme (Kommunikationsprogramme, Fahrassistenten usw.) auf dem Kommunikationsrechner ausgeführt werden, die ebenfalls Leistung für sich beanspruchen. Zusätzlich kann das Intranet des Fahrzeugs in der “CarTALK 2000 Architektur” (siehe Seite 6) noch Nachrichten versenden bzw. empfangen.

Schnittstelle
Das in dem Unterkapitel “Funktechnologie” (siehe Seite 10) vorgestellte Funkgerät und im Unterkapitel “Positionierungssystem” (siehe Seite 11) dargestellte GPS benötigen eine USB/ PCMCIA und eine serielle Schnittstelle, um eine Kommunikation zwischen den Geräten zu

ermöglichen. Das Laptop weißt diese Schnittstellen auf und kann daher mit diesen Geräten verbunden werden.

Das Laptop erfüllt diese zwei wichtigsten Eigenschaften und wird daher für den prototypischen Aufbau eines *Vehicular* MANET verwendet.

2.3. Softwarekomponenten

Die Softwarekomponenten verbinden die einzelnen Hardwarekomponenten zu einem Gesamtsystem und führen die Funktionen aus, die für das *Routing* im *Vehicular* MANET notwendig sind. In den folgenden Unterkapiteln werden die einzelnen Softwarekomponenten aufgelistet und kurz erläutert.

2.3.1. Betriebssystem

Die Anforderungen an ein Betriebssystem sind abhängig von den verwendeten Hardwarekomponenten, da die Schnittstellen von den Funk- und Positionierungsgeräten vom Betriebsystem unterstützt werden müssen. Zusätzlich muss das Betriebsystem flexibel sein, um es an die Bedürfnisse des *Routing* im *Vehicular* MANET anzupassen.

Das Betriebsystem Linux genügt diese Anforderungen. Es unterstützt alle gängigen Schnittstellen und kann darüber hinaus an die Bedürfnisse des *Routing* angepasst werden, da es die *Open Source* Eigenschaft besitzt. Deshalb wird der *Vehicular Routing Mechanism* auf einem Linux Rechner ausgeführt.

2.3.2. Routing Modul

Das *Routing* Protokoll soll hier nur für das allgemeine Verständnis kurz erläutert werden, um eine Vorstellung zu bekommen, welche Anforderungen an ein Protokoll für ein *Vehicular* MANET gestellt werden. In den folgenden Abschnitten wird das *Routing* im *Vehicular* MANET nur sehr kurz erläutert, da eine komplette Beschreibung den Rahmen dieser Diplomarbeit sprengen würde. Eine detailierte Erläuterung kann in [Le02] nachgelesen werden.

Das *Routing* Modul kann in die folgenden drei Bereiche, *Map-based Geographic Forwarding* (MGF), *Beaconing* und *Location Service* aufgeteilt werden, wie in Abbildung 7 zu sehen ist.

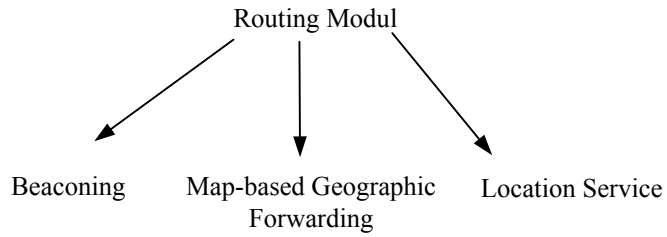


Abbildung 7: Routingkomponenten im Vehicular MANET

Beaconing

Das *Beaconing* Protokoll hat die Aufgabe, die Positionen und Eigenschaften der Nachbarfahrzeuge, die sich innerhalb des Senderadius aufhalten, zu erkennen. Dazu senden alle Fahrzeuge regelmäßig *Beacons* an ihre Nachbarn. Ein *Beacon* ist eine Nachricht, in der Informationen (Position, Geschwindigkeit, Fahrzeug-ID usw.) über das eigene Fahrzeug enthalten sind. Das *Beacon* wird mit *Broadcast* gesendet, dadurch empfängt jedes andere Fahrzeug innerhalb des Sendegebietes diese Nachricht und kennt somit die Position, Geschwindigkeit, Fahrzeug-ID usw. des sendenden Fahrzeugs. Die anderen Fahrzeuge senden ebenfalls *Beacons*, um sich bei ihren Nachbarn bemerkbar zu machen. Dadurch wird sichergestellt, dass jedes Fahrzeug von anderen Fahrzeugen erkannt wird.

Aufgrund der sehr begrenzten Bandbreite im *Vehicular* MANET darf das Beaconintervall nicht zu klein sein, da sonst zuviel Bandbreite für das *Beaconing* verwendet wird. Auf der anderen Seite, darf das Intervall nicht zu lang sein, da sonst zu schnelle Fahrzeuge das *Beacon* zu spät oder gar nicht empfangen würden. Diese Problematik wird in Kapitel 4.1.1 "Anforderungen an das Beaconing Protokoll" und Kapitel 4.1.2 "Bewertung der Beaconing Protokolle" ausführlich erläutert.

Map-based Geographic Forwarding

Für die Weiterleitung von Nachrichten wird das *Geographic Source Routing* (GSR) benötigt. Das GSR berechnet für jede zu versendende Nachricht die komplette Route, damit sie entlang diesem Weg zu ihrem Ziel weitergeleitet wird. Die einmal berechnete Route wird mit der Nachricht verschickt und bleibt bis zum Ziel unverändert. Eine Route besteht dabei aus Wegpunkten, die eine Strecke entlang den Straßen bis zum Ziel vorgeben.

Das Map-based Geographic Forwarding (MGF) Protokoll ist für die Weiterleitung der Nachrichten zuständig. Jede Nachricht wird solange an Fahrzeuge weitergeleitet bis sie entweder am Ziel bzw. im Zielgebiet ankommt, oder momentan kein passendes Fahrzeug in Sendereichweite gefunden werden kann. In letzteren Fall muss die Nachricht solange gespeichert werden, bis sich ein passendes Fahrzeug innerhalb der Sendereichweite befindet oder ein *Timer* abläuft.

Das MGF nützt die eingeschränkte Bewegungsfreiheit der Fahrzeuge aus, um Nachrichten effizient weiter zu leiten. Deshalb ist die Entfernung zwischen Fahrzeug und Zielposition nicht die Luftlinie, sondern die Summe der einzelnen Straßenabschnitte (geographische Entfernung).

Das MGF muss deshalb für jedes Fahrzeug, das sich innerhalb des Sendebreichs aufhält, die geographische Entfernung vom Fahrzeug bis zum Wegpunkt der Nachricht berechnen. Anschließend wählt es das Fahrzeug aus, das sich geographisch am nächsten Wegpunkt befindet und sendet die Nachricht mit *Unicast* an das nächstliegende Fahrzeug.

Location Service
Der *Location Service* hat die Aufgabe, den Aufenthaltsort für ein bestimmtes Fahrzeug innerhalb des *Vehicular* MANET zu ermitteln. Dieser Dienst wird immer dann benötigt, wenn Nachrichten an Fahrzeuge gesendet werden müssen, die sich nicht innerhalb der Sendereichweite des Fahrzeugs aufhalten. Eine Konzeption eines *Location Service* für ein *Vehicular* MANET wird in Kapitel 5.2 "Konzeption eines Vehicular Location Service" dargestellt und erläutert.

2.3.3. Digitale Straßenkarte

Die digitale Straßenkarte bietet Funktionen an, die für das Weiterleiten von Nachrichten notwendig sind. Diese Funktionen können aus dem vorherigen Abschnitt “Map-based Geographic Forwarding” (siehe Seite 15) abgeleitet werden. In Tabelle 8 sind die notwendigen Funktionen aufgelistet. Mit der `GetRoute`-Funktion erhält GSR die komplette Route für eine Nachricht anhand von Wegpunkten. Mit der `GetStreetInfo`-Funktion erhält der MGF den aktuellen Straßennamen, um festzustellen auf welcher Straße sich das Fahrzeug gerade befindet. Mit der `GetGeographicDistance`-Funktion kann die geographische Entfernung zwischen Fahrzeug und Wegpunkten ermittelt werden. Durch die `GetLaneNumber`-Funktion erhält man die Information, auf welcher Spur sich das Fahrzeug gerade befindet, dazu muss das GPS Positionierungssystem hinreichend genau sein (siehe Kapitel 2.2.2 "Positionierungssystem").

Funktion	Übergabeparameter	Rückgabewert
`GetRoute`	Own Position, Destination Position	Own Position bis zur Destination Position
`GetStreetInfo`	Own Position	Straßename und -nummer
`GetGeographicDistance`	Vehicle Position, Destination Position	Distance in km
`GetLaneNumber`	Own Position	Lane number

Tabelle 8: Anforderung an eine digitale Straßenkarte

Die Funktionen in Tabelle 8 sind frei erfunden und soll lediglich zur Veranschaulichung der Anforderungen von GSR und MGF an eine digitale Straßenkarte dienen.

2.3.4. Anwendungen

Anwendungen für ein *Vehicular* MANET sind noch nicht vorhanden und müssen erst entworfen und implementiert werden. Im Projekt „CarTALK 2000“ werden Konzepte für Anwendungen entwickelt, die den Fahrer vor Gefahren warnen und ihn in schwierigen Situationen entlasten. In “CarTalk 2000 Anwendungen” (siehe Seite 7) wurden die einzelnen Konzepte genauer beschrieben. Darüber hinaus können auch andere Anwendungen, wie Spiele, WWW-Programme (Email, Browser, Chat usw.) für ein *Vehicular* MANET verwendet werden.

2.4. Kommunikationsarchitektur

In diesen Kapitel wird die Kommunkationsarchitektur für das *Vehicular* MANET erläutert, die aus den Hardware- und Softwarekomponenten aus Kapitel 2.2 und Kapitel 2.3 aufgebaut ist.

In der Abbildung 9 ist die Kommunikationsarchitektur für ein *Vehicular* MANET dargestellt. Die einzelnen Hardwarekomponenten sind mit dem mobilen Rechner verbunden. Die Softwarekomponenten sind in die Hardware eingebettet, da die Software in der Hardware ausgeführt wird. Hardwarekomponenten können ohne Softwarekomponenten existieren und werden deshalb separat voneinander dargestellt.

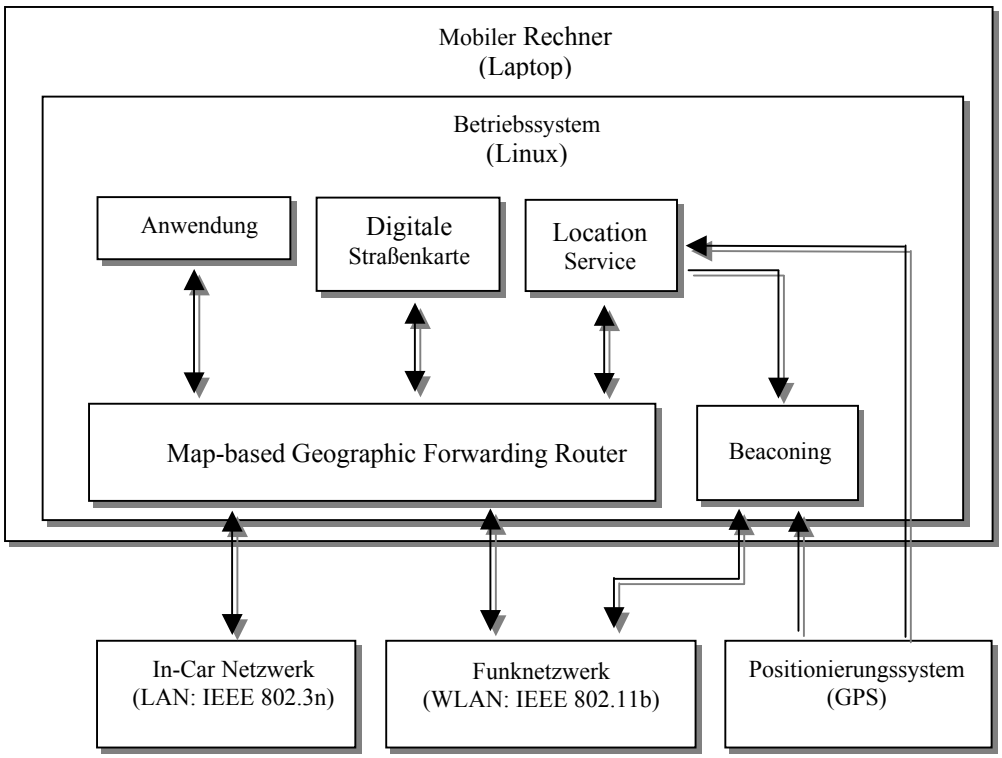


Abbildung 9: Kommunikationsarchitektur

Die Pfeile veranschaulichen die Kommunikationsverbindungen. Doppelpfeile weisen auf eine bidirektionale und einfache Pfeile auf eine unidirektionale Verbindung hin.

Anwendung
Anwendungen übergeben ihre Nachrichten an den *Map-based Geographic Forwarding Router*, damit die Nachricht an ein anders Fahrzeug gesendet wird. Empfangene Nachrichten, die für das Fahrzeug bestimmt sind, werden vom MGF-Router an die Anwendung übergeben.

Map-based Geographic Forwarding Router
Der MGF-Router ist für die korrekte Weiterleitung der Nachrichten zuständig. Empfangene Nachrichten müssen entweder an andere Fahrzeuge, die sich näher an der Zielposition bzw. am Zielgebiet befinden weitergeleitet werden, oder an die Anwendung übergeben werden, falls die Nachricht für das Fahrzeug bestimmt war.

Digitale Straßenkarte
Die digitale Straßenkarte liefert die geographische Entfernungen zwischen den Nachbarfahrzeugen und der Zielposition bzw. dem Zielgebiet der Nachricht. Außerdem kann sie anhand der GPS-Position des Fahrzeuges den Straßennamen bestimmen.

Location Service
Der *Location Service* liefert dem MGF-Router die GPS-Position des Fahrzeugs, an den die Nachricht gesendet werden soll. Darüberhinaus liefert er die aktuelle Position des eigenen Fahrzeugs. Dazu erhält er vom Positionierungssystem die aktuelle GPS Information.

Beaconing
Das *Beaconing* ist unabhängig vom MGF Router und benötigt nur Informationen über die Dichte der Fahrzeuge innerhalb des Sendegebiets und die eigene Position. Diese Informationen werden vom *Location Service* und dem Positionierungssystem geliefert. Dadurch läßt sich das Beaconinvertall berechnen. Nach dem Ablauf des Intervalls wird das *Beacon* versendet.

Positionierungssystem
Das GPS liefert die aktuelle Position, Fahrtrichtung und Geschwindigkeit des Fahrzeugs an das *Beaconing* und an den *Location Service*.

In-Car Netzwerk
Mit dem *In-Car* Netzwerk können ebenfalls Nachrichten an den MGF-Router versendet und empfangen werden. Dadurch können Anwendungen auf andere Rechner ausgeführt werden.

Funknetzwerk

Das Funknetzwerk besteht aus IEEE 802.11b Funkkarten. Die Funkkarte erhält die zu sendenen Nachrichten vom MGF-Router oder vom *Beaconing* Modul. Empfangene Nachrichten werden an den MGF-Router und empfangene *Beacons* an das *Beaconing* Modul übergeben.

3. Untersuchung der vorhandenen Arbeiten

In diesem Kapitel werden Untersuchungen über das *Beaconing* Protokoll in Kapitel 3.1, dem *Location Service* in Kapitel 3.2 und den vorhandenen *Routing* Implementierungen in Kapitel 3.3 durchgeführt.

3.1. Beaconing Protokolle

Die Aufgabe eines *Beaconing* Protokolls wurde bereits in Abschnitt "Beaconing" (siehe Seite 15) erläutert. In diesen Kapitel werden zwei *Beaconing* Protokolle vorgestellt, die aus einem *Routing* Prototkoll bzw. *Location Service* stammen.

3.1.1. Statisches Beaconing

In [W02] wird ein statisches *Beaconing* Protokoll beschrieben. Es sendet die *Beacons* nach einen fest vorgegebenen Intervall an seine Nachbarfahrzeuge durch *Broadcast*. In diesen *Beacon* befindet sich die aktuelle Position und ID des Fahrzeugs. Um Bandbreite zu sparen wird die *Piggy-backing* Methode angewendet, d.h. an jeder Nachricht wird ein *Beacon* angehängt. Somit müssen keine weitere *Beacons* mehr gesendet werden, falls Nachrichten innerhalb des Beaconintervalls gesendet wurden.

Damit die Nachbarfahrzeuge auch Nachrichten mit dem *Beacon* des Senders empfangen können, die aber nicht für sie bestimmt sind, müssen alle Funkkarten im *promiscuous* Modus betrieben werden.

3.1.2. Geschwindigkeitsabhängiges Beaconing

In [CB+02] berechnet das *Simple Location Service* (SLS) Protokoll die Intervalldauer des *Beacons* anhand der momentanen Geschwindigkeit des Fahrzeugs und der Sendereichweite des Senders. Die Formel ist den Gleichungen 1 abgebildet.

$$Beaconintervall = \frac{Trange}{\alpha v}$$

$Trange$: Sendereichweite

v : Geschwindigkeit des Fahrzeuges

α : Skalierungsfaktor

Gleichungen 1: Beaconintervall vom Simple Location Service

Das Beaconintervall wird länger je größer die Sendereichweite oder je langsamer die Geschwindigkeit des Fahrzeugs ist. Somit wird bei schnellen Fahrzeugen oder kleinerer Sendereichweite das Beaconintervall kürzer. Das *Beacons* wird bei kurzen Beaconintervallen häufiger gesendet, damit können auch schnell fahrende Fahrzeuge von langsam fahrenden Fahrzeugen erkannt werden. Das Erkennen von den Fahrzeugen, die sich innerhalb der Sendereichweite befinden ist für eine Kommunikation im *Vehiclar* MANET zwischen den Fahrzeugen zwingend notwendig

Der Skalierungsfaktor α muss experimentell bestimmt werden. Liegt er zu hoch kann das Netzwerk bei vielen Fahrzeugen überlastet werden, ist er zu niedrig können eventuell nicht alle Fahrzeuge erkannt werden.

3.2. Location Services

In diesen Kapitel werden die vorhandenen *Location Service* Protokolle klassifiziert und vorgestellt. Die Aufgabe eines *Location Service* wurde schon in dem Abschnitt “Location Service” (siehe Seite 16) erläutert.

Der *Location Service* kann in zwei Arbeitsbereiche unterteilt werden, in *Location Dissemination* und in *Location Discover*y. Im ersten Arbeitsbereich wird die Verbreitung (Dissemination) der geographischen Positionen von mobilen Knoten (Fahrzeugen) zu anderen mobilen Knoten mit den sogenannten *Location Update* Nachrichten durchgeführt. Der zweite Bereich ist zuständig für das Auffinden (Discovery) von geographischen Positionen bestimmter Fahrzeuge, die sich innerhalb eines p*osition-based* MANETs befinden.

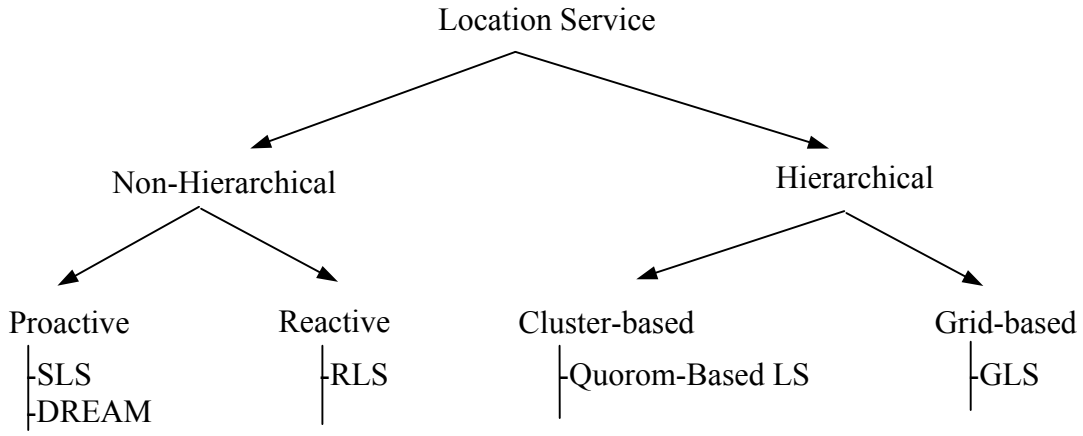


Abbildung 10: Klassifizierung der Location Service Strategien

Die Klassifikation der unterschiedlichen *Location Service*s werden in Abbildung 10 anhand einer Baumstruktur dargestellt. Entscheidend für die Klassifizierung der einzelnen *Location Service* Protokolle sind die Strategien, die sie für die *Location Dissemination* bzw. *Location Discovery* verwenden. Die Klassifikation erhebt keinen Anspruch auf die Allgemeingültigkeit für alle Protokolle. Die Baumstruktur soll helfen die vorgestellten Protokolle besser zu verstehen.

Location Dissemination
Die Aufgabe einer *Location Dissemination* besteht, wie schon oben erwähnt, in der Verbreitung von geographischen Positionen der mobilen Knoten im MANET durch *Location Update* Nachrichten. Die Mitteilung der Knotenposition an andere mobile Knoten ist notwendig, damit sie von der Existenz des mobilen Knotens erfahren. Das *Location Dissemination* Protokoll regelt, welche Knotenpositionen von anderen mobilen Knoten empfangen und ob die Knotenpositionen in der *Location Table* gespeichert werden. Eine *Location Table* ist meistens eine zwei-dimensionale Tabelle, in der die Fahrzeug-IDs und die geographischen Positionen der Fahrzeuge gespeichert werden.

Location Discovery
Reactive Location Service Protokolle benötigen die aktuelle geographische Position eines Knotens auf Anfrage. Dazu werden die *Location Discovery* Protokolle eingesetzt. Sie haben die Aufgabe einen bestimmten Knoten innerhalb des Netzwerks zu finden, um die aktuelle Position zu erfragen. Eine sogenannte *Location Request* Nachricht wird an die Nachbarknoten versendet, die wiederum die Nachricht an ihre Nachbarknoten weiterleiten, wenn sie die *Location Request* Nachricht nicht beantworten können. Kommt die *Location Request* Nachricht bei dem gesuchten Knoten an, sendet er seine aktuelle Knotenposition durch eine *Location Respond* Nachricht an den Sender zurück. Erhält darauf der Sender die gesendete Knotenposition, kann er seine Nachrichten an den Empfängerknoten schicken.
Beim *Reacvtive Location Service* tritt durch die Suche des Knotens eine gewisse Verzögerung beim Senden von Nachrichten auf, wenn die Position des Empfängers dem Sender noch nicht bekannt ist. Die Verzögerungsdauer hängt von der Antwortzeit der *Location Request* Nachricht bzw. von der *Timeout* Einstellung des Protokolls ab, wenn der Knoten nicht gefunden wurde. Der *Timeout* läuft ab, wenn der Sender innerhalb einer vorgegebenen Zeit keine Antwort auf seine *Location Request* Nachricht erhält. Bei *Proactive Location Service* Protokollen tritt diese Verzögerung nicht auf.

3.2.1. Non-Hierachical
Location Service Protokolle sind in der Kategorie “Non-Hierarchical” einzuordnen, wenn sie die Knotenpositionen mit Hilfe eines *Flooding*-Algorithmus im ganzen Netzwerk verteilen. “Non-Hierarchical” bedeutet, dass bei der Verbreitung keine hierarchischen Strukturen innerhalb des

Netzwerks berücksichtigt werden. Jeder Knoten, der eine Nachricht empfängt, sendet sie mit einem *Broadcast* an seine Nachbarknoten weiter, falls er die Nachricht noch nicht versendet hat.
"Non-Hierarchical" *Location Service* Protokolle können in *Proactive* und *Reactive Location Services* weiter untergliedert werden.

3.2.1.1. Proactive Location Service

Proactive Location Service Protokolle besitzen nur einen *Location Dissemination*, aber keinen *Location Discovery* Algorithmus. Das *Location Dissemination* Protokoll verteilt die *Location Update* Nachrichten im ganzen MANET. Jeder mobile Knoten verwaltet eine *Location Table*, in der Fahrzeug-IDs und Positionen aller mobilen Knoten eingetragen sind. Der Sender sucht nach dem Standort des Empfängers in seiner Location Table, damit er dem Empfänger eine Nachricht schicken kann. Somit ist jeder Knoten in der Lage, Nachrichten an jeden beliebigen Knoten zu verschicken, ohne vorher den Aufenthaltsort des Empfängers explizit zu erfragen. Deshalb verwenden *Proactive Location Service* Protokolle keine *Location Request* und *Location Respond* Nachrichten, sondern nur *Location Update* Nachrichten.

SLS

Das *Simple Location Service* (SLS) Protokoll [CB+02] sendet seine aktuellen *Location Update* Nachrichten[a] mit einem vorgegebenen Intervall an seine Nachbarknoten, die dann die Nachrichten wiederum an ihre Nachbarknoten weiterleiten usw. Die Intervalldauer hängt von der Geschwindigkeit des Knotens ab. Ist die Geschwindigkeit des Knotens hoch, müssen dementsprechend viele *Location Update* Nachrichten an die Nachbarknoten gesendet werden, damit sie die aktuelle Position des Knotens mitbekommen. Die *Location Update* Nachrichten werden mit einem *Flooding* Algorithmus im ganzen Netzwerk verteilt. Die empfangenden *Location Update* Nachrichten werden benötigt, um die *Location Table* laufend zu aktualisieren.
Der SLS ist ein *Proactive Location Service*, da in der *Location Table* die Positionen aller Knoten des Netzwerks gespeichert sind. Da der SLS einen *Flooding* Algorithmus zur Verteilung seiner *Location Update* Nachricht verwendet, kann er in die Kategorie "Non-Hierarchical" eingeordnet werden.

a. in [CB+02] werden sie Location Packet (LP) genannt

DREAM

Der *Distance Routing Effect Algorithm for Mobility* (DREAM) [BC+98] sendet ebenfalls periodisch seine aktuellen *Location Update* Nachrichten an seinen Nachbarn. Wie der SLS besitzt er auch eine *Location Table*, in der alle Netzwerkknoten enthalten sind. Der Unterschied bei der *Location Table* zwischen DREAM und SLS ist, dass nicht die globalen Knotenpositionen gespeichert werden, sondern die relativen Entfernungen und Richtungen zum eigenen Knoten.

Ein weiterer Unterschied liegt in der Verbreitungsstrategie der *Location Update* Nachricht. Ein mobiler Knoten ist in der Lage, seine Positionsgenauigkeit zu anderen mobilen Knoten zu steuern, indem er die Intervalldauer für die *Location Update* Nachrichten verändert (zeitliche Auflösung) und/oder angibt, wie weit seine *Location Update* Nachrichten durch das Netz befördert werden dürfen, bevor sie von den Knoten verworfen werden (räumliche Auflösung). Die zeitliche Auflösung ist meistens gekoppelt an die Geschwindigkeit des jeweiligen Knoten. Je schneller der Knoten sich bewegt, desto häufiger müssen *Location Update* Nachrichten gesendet werden. Die räumliche Auflösung ist bei Nachbarknoten sehr hoch, da sie jede *Location Update* Nachrichten empfangen und somit ihre *Location Table* ständig aktualisieren können. Knoten, die sich relativ weit weg vom ursprünglichen Sender befinden, erhalten die *Location Update* Nachrichten seltener. Die Position der relativ weit entfernten Knoten können deshalb häufiger veraltet bzw. ungenauer sein als die Position der Knoten, die sich in unmittelbarer Nähe des Senders befinden.

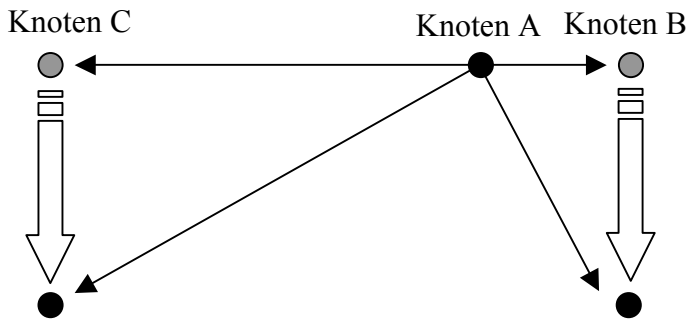


Abbildung 11: Der Distanzeffekt

Dieses Problem wird durch den Distanzeffekt [BCS99] entschärft, da die vom Sender weiter entfernten Knoten, die Richtungen und Entfernungen nicht so stark ändern wie bei näheren Knoten. In Abbildung 11: "Der Distanzeffekt" ist dieser Umstand nochmals dargestellt. Wenn

Knoten A sich nicht bewegt und Knoten B und C sich mit derselben Geschwindigkeit und in die gleiche Richtung bewegen, ändert sich aus der Sicht des Knotens A die Entfernung und Richtung zum Knoten B stärker als zum Knoten C. Durch den Distanzeffekt müssen die weiter entfernten Knoten weniger oft aktualisiert werden als die Knoten, die sich näher beim Sender (Knoten A) befinden.

3.2.1.2. Reactive Location Service

Reactive Location Service Protokolle müssen die Position des Empfängers explizit erfragen. Sie besitzen einen Algorithmus für *Location Dissemination* und *Location Discovery. Reactive Location Service* Protokolle, die zu den "Non-Hierarchical" Protokollen zugeordnet werden, verwalten keine *Location Table*, sondern nur einen *Location Cache*. Im *Location Cache* sind die Einträge von den Knoten enthalten, die sich momentan in der Funkreichweite des Knotens befinden. Die Einträge über die Knotenpositionen im *Location Cache* werden durch *Location Update* Nachrichten laufend aktualisiert. Knoten, die eine gewisse Zeit ihre Position nicht mehr senden, werden aus dem *Location Cache* gelöscht.

Kennt der *Location Service* nicht den aktuellen Aufenthaltsort des Empfängers, muss er eine *Location Request* Nachricht mit Hilfe eines *Flooding* Algorithmus an dem Empfänger schicken, der dann seine aktuelle geographische Position zurücksendet. Der Vorteil beim *Flooding* ist, wie schon in dem obigen Kapitel 3.2.1.1 "Proactive Location Service" erwähnt, die äußerst zuverlässige Verteilung der Nachrichten innerhalb eines MANETs. Große Probleme bereiten weiterhin das "Broadcast Storm Problem" und das "Hidden Terminal" Problem bei Funknetzwerken.

RLS

Das *Reactive Location Service* (RLS) [CB+02] Protokoll ist ein sehr einfaches Protokoll. Wenn die Knotenposition für einen Knoten nicht bekannt oder veraltet ist, sendet es zuerst eine *Location Request* Nachricht an seine Nachbarknoten. Reagieren die Nachbarknoten nicht innerhalb einer bestimmten Zeit auf seine Anfrage, dann verteilt das RLS Protokoll die *Location Request* Nachricht mit Hilfe eines *Flooding* Algorithmus über das ganze Netzwerk.

Empfängt ein Knoten eine *Location Request* Nachricht und der gesuchte Knoten ist im *Location Cache* enthalten, antwortet der Knoten mit einer *Location Respond* Nachricht. Dabei legt die *Location Respond* Nachricht den umgekehrten Weg von der *Location Request* Nachricht zurück.

Das bedeutet, dass in jeder *Location Request* Nachricht der komplette Weg gespeichert werden muss.

3.2.2. Hierarchical Location Service

Hierarchical Location Service Protokolle verwenden eine hierarchische Struktur, um die Nachrichten von mobilen Knoten weiterzuleiten. Eine weitere Untergliederung in “Cluster-based” und “Grid-based” kann anhand der hierarchischen Struktur vorgenommen werden. “Cluster-based” *Location Service* Protokolle haben im allgemeinen nur eine Hierarchieebene. Die “Grid-based” *Location Service* Protokolle können dagegen beliebig viele Hierarchieebenen verwenden.

3.2.2.1. Cluster-based Location Service

In der “Cluster-based” Kategorie befinden sich die *Location Service* Protokolle, die bei der Verteilung der *Location Update* Nachricht keinen dem Flooding ähnlichen Algorithmus verwenden, sondern gezielt ihre Position an einen bestimmten Netzwerkknoten senden. Die “Cluster-based” Protokolle unterscheiden dazu zwei Netzwerkknotentypen. Der erste Knotentyp ist ein gewöhnlicher mobiler Knoten, der in der Lage ist, Nachrichten an andere mobile Knoten weiterzuleiten. Der zweite Knotentyp (*Location Service* Knoten) ist ebenfalls ein mobiler Knoten, kann aber die Positionen von anderen mobilen Knoten in einer *Location Table* speichern, wenn sich die mobilen Knoten bei einem *Location Service* Knoten anmelden. Zusätzlich hat der *Location Service* Knoten Kenntnisse von den Positionen der umliegenden *Location Service* Knoten, damit er notfalls eine Positionsanfrage an diese *Location Service* Knoten weiterleiten kann, falls sich der gesuchte Knoten nicht in seiner *Location Table* befindet.

Durch die Aufteilung der einzelnen Knoten in mobile Knoten und *Location Service* Knoten entsteht meistens eine einstufige Hierarchie, die einer Organisation von kleinen Gruppen (Cluster) entspricht, die aber untereinander durch *Location Service* Knoten verbunden sind. Der Vorteil von “Cluster-based” Protokollen liegt in der effizienten Verbreitung der Knotenpositionen, da sie direkt an die *Location Service* Knoten gesendet werden. Im Gegensatz zu “Non-Hierarchical” Protokollen sind in der *Location Table* nur die Positionen der *mobilen Knoten* enthalten, die zu einer Gruppe eines *Location Service* Knoten gehören. Bei einer Kommunikation zwischen zwei *mobilen Knoten* wird immer ein *Location Service* Knoten benötigt, wenn der Sender die Position des Empfängers nicht kennt oder diese veraltet ist.

Quorum-based Location Service

Die Idee eines Quorumsystems kommt aus den Anwendungsbereichen Datenbanken und Verteilte Systeme. Ein Quorum besteht dabei aus einer bestimmten Anzahl von *Location Service* Knoten. Schreib- und Lesebefehle werden auf einem Quorum ausgeführt. Auch das Aktualisieren von Daten (Schreibbefehl) wird in einem Quorum von vorhanden *Location Service* Knoten ausgeführt. Wenn die Schnittmenge von Schreib- und Lese-Quorum nicht leer ist, dann ist gewährleistet, dass aktuelle Daten gelesen werden können. [MW+01].

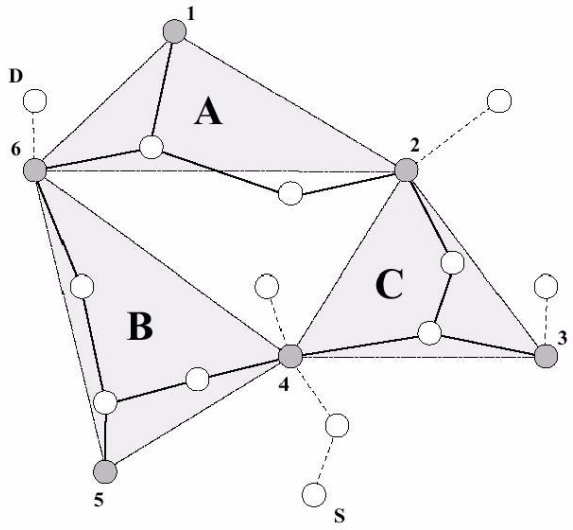


Abbildung 12: Beispiel eines Quorum-based Location Service

In [HL99] wird diese Idee verwendet, um einen *Location Service* für MANET zu entwickeln. Die Vorgehensweise für *Location Dissemination* und *Location Discovery* eines *Quorum-based Location Service* wird anhand der Abbildung 12 erläutert.

Location Dissemination

Die *Location Service* Knoten 1-6 bilden insgesamt 3 Gruppen. In der Gruppe A sind die Knoten 1, 2, 6, in der Gruppe B die Knoten 4, 5, 6 und in Gruppe C die Knoten 2, 3 und 4 enthalten. Diese Knoten stellen ein sogenanntes virtuelles *Backbone* dar.

Alle mobilen Knoten senden ihre aktuellen *Location Upate* Nachrichten an einen *Location Service* Knoten, der ihnen am nächsten steht. Der *Location Service* Knoten wählt dann ein Quorum von den *Location Service* Knoten aus, um die Knotenposition an diese zu verteilen. Dazu

sendet der Knoten D seine Position an den Knoten 6, der beispielsweise Gruppe A auswählt, um die Knotenposition von D an die Knoten 1 und 2 zu verteilen.

Location Discovery
Der Location Discovery Algorithmus sendet die *Location Request* Nachrichten gezielt mit einem *Unicast* an *Location Service* Knoten. Das Besondere daran ist, dass nur die *Location Service* Knoten *Location Table* führen in denen eine kleine Untermenge von mobilen Knoten (Gruppen) oder die benachbarten *Location Service* Knoten enthalten sind. Empfängt ein *Location Service* Knoten eine *Location Request* Nachricht, dann wird geprüft, ob sich der gesuchte mobile Knoten in der *Location Table* befindet. Befindet sich der gesuchte Knoten in der *Location Table*, dann sendet der *Location Service* Knoten eine *Location Respond* Nachricht an den Sender zurück, sonst leitet der Knoten die *Location Request* Nachricht an alle seine benachbarten *Location Service* Knoten weiter. Das "Broadcast Storm Problem" ist trotzt der beschränkten Anzahl an Knoten vorhanden.

Der *Location Discovery* Algorithmus soll anhand der Abbildung 12 kurz erläutert werden. Der Knoten S benötigt die Position von Knoten D, deshalb sendet Knoten S eine *Location Request* Nachricht an den nächsten *Location Service* Knoten 4. Dieser kennt die Position von Knoten D nicht und wählt daraufhin beispielsweise das Quorum B mit den Knoten 4,5 und 6 aus, um die Nachricht an die *Location Service* Knoten 5 und 6 weiterzuleiten. *Location Service* Knoten 6 kennt die Position von Knoten D, da er auch ein Mitglied vom Quorum A ist und sendet eine Location Respond Nachricht an den Knoten S zurück.
Knotenpositionen können auf verschiedenen *Location Service* Knoten in *Location Table* redundant gespeichert werden, daher ist ein Zeitstempel für jede Knotenposition notwendig. Dadurch ist es möglich, bei mehreren empfangenen Antworten die aktuelle Knotenposition herauszufinden.

3.2.2.2. Grid-based Location Service
Location Service Protokolle in der Kategorie “Grid-based” können beliebig viele Hierarchiestufen verwenden. Die Hierarchie wird durch eine Baumstruktur realisiert. Eine Hierarchiestufe entspricht einer Verbindung zwischen einem Vater- und Sohn-Knoten innerhalb einer Baumstruktur. Protokolle mit beliebig vielen Hierarchiestufen haben den Vorteil, sehr viele

Knoten effizient verwalten zu können. Beispielsweise haben die Protokolle für festverdrahtete Netzwerke, wie DNS[a], zum Teil eine sehr tiefe Hierarchieebene. Der Nachteil bei Protokollen mit vielen Hierarchiestufen ist der relativ lange Nachrichtenweg, der beim Weiterleiten in die höheren Ebenen entsteht. Je öfter eine Nachricht durch eine Hierarchieebene weitergereicht wird, desto größer ist die Wahrscheinlichkeit, dass die Nachricht auf dem Weg verloren geht, weil eine Verbindung zur nächsten Hierarchieebene nicht mehr vorhanden ist.

GLS

Der *Grid Location Service* (GLS) ist ein Teil vom Grid Projekt [LJ+00], [MJK+00]. Der GLS unterteilt die Umgebung in gleich große Quadrate, die dann hierarchisch zusammengesetzt werden. Die Aufteilung der Umgebung ist jedem Knoten bekannt. Dadurch kann jeder Knoten bestimmen, in welchem Quadrat er sich gerade aufhält.

Die Hierarchie eines Quadrates der n-ten Ordnung setzt sich aus genau vier Quadraten der (n-1)-ten Ordnung zusammen, daraus ergibt sich ein sogenannter Quadtree. Quadrate der 1. Ordnung sind die kleinsten Quadrate, die bei der Aufteilung der Umgebung vom GLS entstanden sind. Ein Quadrat der 2. Ordnung besteht aus genau vier Quadraten der 1. Ordnung. Ein Quadrat von der 3. Ordnung wird wiederum aus genau vier Quadraten der 2. Ordnung gebildet usw. Um Überlappungen von Quadraten zu verhindern, gehört ein Quadrat der n-ten Ordnung nur zu einem Quadrat der (n+1)-ten Ordnung.

Location Dissemination

In der *Location Dissemination* Phase verteilt ein Knoten seine ID und geographische Position an drei *Location Server* für jede Hierarchiestufe.

In Abbildung 13 sendet der Knoten B seine Informationen an drei *Location Server* aus dem Quadrat der 1. Ordnung (2, 23, 63), drei *Location Server* aus dem Quadrat der 2. Ordnung (26, 31, 43) und drei *Location Server* aus dem Quadrat der 3. Ordnung (19, 20, 37). Knoten, die die ID und Position vom Knoten B erhalten haben, sind an der fett gedruckten Nummer zu erkennen. Der Knoten B ist dann in ihrer *Location Table* aufgelistet. Die Auswahl eines Knotens innerhalb eines Quadrates erfolgt anhand der nächstgrößeren ID, ausgehend von der eigenen Knoten ID. Die IDs sind in einer zyklischen Liste geordnet, d.h. die nächstliegende ID ist gleichzeitig die nächstgrößere ID und der Nachfolger der letzten ID wäre die erste ID in der zyklischen Liste.

a. Domain Name Service

Deswegen sendet der Knoten 17 seine ID und Position an Knoten 43 und nicht an Knoten 12. Die Abbildung 13 zeigt nur einen kleinen Auszug einiger *Location Tables* der Knoten.

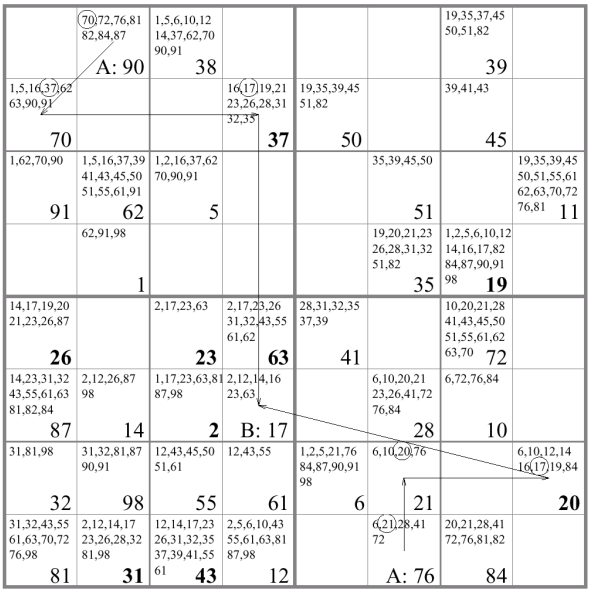


Abbildung 13: GLS Aufbau und Suche

Der GLS verwaltet zwei Tabellen, die *Location Table* und der *Location Cache*. Jeder Eintrag in einer der Tabellen enthält die ID und geographische Position des Knotens. Durch das *Broadcast*-Senden der aktuellen Position innerhalb eines Quadrates der 1. Ordnung werden die *Location Table* und *Location Cache* laufend aktualisiert. Knoten aus der 2. und 3. Ordnung in der *Location Table* müssen dagegen durch ein *Unicast* Senden über die aktuelle Position mitgeteilt werden. Wie in Abbildung 13 zu sehen ist, muss der Knoten B die Knoten in der 1. bis 3. Ordnung über seine Positionsveränderung durch *Unicast* Senden mit *Location Update* Nachrichten informieren.

GLS Location Discovery

Der *Location Discovery* Algorithmus soll anhand eines Beispiels in Abbildung 13 erläutert werden. Die Knoten IDs sind in jedem Quadrat unten rechts angegeben. Die Einträge der *Location Table* der Knoten sind links oben im selben Quadrat aufgelistet. Die Pfeile geben den Weg der *Location Request* Nachricht an. Die eingekreisten IDs in der *Location Table* zeigen an, welche Knoten zur Weiterleitung der *Location Request* Nachricht ausgewählt wurden.

Wenn Knoten A (76) die geographische Position von Knoten B (17) benötigt, dann sendet Knoten A eine *Location Request* Nachricht an Knoten 21, da Knoten A den Knoten mit der zyklisch nächsten ID zum Empfänger auswählt. Die ID 21 befindet sich zyklisch näher an der ID 17 als die übrigen IDs 6,28,41,72. Knoten 21 kann ebenfalls die Positionsanforderung nicht beantworten und sendet sie an Knoten 20, da die ID 20 zyklisch näher an 17 liegt als die IDs 6,10,76. Knoten 20 kann die Positionsanforderung auch nicht beantworten und leitet sie direkt zum Knoten 17 weiter. Knoten 17 kennt natürlich seine aktuelle Position und sendet sie zum Knoten A (76) zurück.

In Abbildung 13 ist ein weiteres Beispiel zu sehen. Knoten A, in diesem Fall mit der Knoten ID 90, benötigt die Knotenposition von Knoten B (17). Die *Location Request* Nachricht wird über Knoten 70 und 37 direkt an Knoten 17 gesendet. Dieser sendet seine aktuelle Position dann an Knoten A (90) zurück.

3.3. Implementierung der Ad-Hoc Routing Protokollen

In diesem Kapitel werden die vorhandenen Implementierungen von *Ad-Hoc Routing* Protokollen für das Unix bzw. Linux Betriebssystem vorgestellt und erläutert. Dazu werden die Implementierung der Protokolle in *User Space*, *Kernel Space* und *Hybrid* untergliedert. In Abbildung 14 sind die zu untersuchenden Protokollimplementierungen in die jeweilige Kategorie eingeordnent.

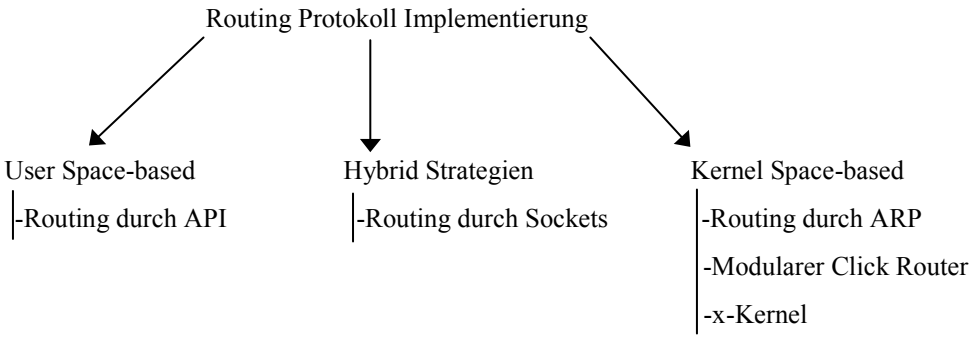


Abbildung 14: Klassifizierung der Routing Protokoll Implementierungen

3.3.1. User Space-based

Implementierte Protokolle, die den Standard *Kernel* nicht verändern, werden zu der *User Space* Kategorie zugeordnet. Solche Programme können ausgeführt werden, ohne den *Kernel* vorher modifizieren zu müssen.

Routing durch API

In [KZ+02] wird ein Konzept für die Implementierung eines *Reactive Ad-Hoc Routing* Protokolls vorgestellt. Das Konzept besteht aus einer *Packet-Routing Function* und einer *Packet-Forwarding Function* (siehe Abbildung 15).

In der *Packet-Forwarding Function* Einheit werden die ankommenden Pakete (packets in) anhand der *Kernel Routing Table* an das jeweilige Netzwerk *Interface* gesendet. Die *Paket-Routing Function* Einheit im *User Space* ist zuständig für die Wegberechnung der Pakete in der *Routing Tabl*e. Diese Einheit wird durch ein *Routing Daemon* realisiert. Die eingefärbten Bereiche stellen die erweiterten Komponenten des ursprünglichen Linux Systems dar.

Die Funktion des *Ad-Hoc Routing daemon* wurde in einen Modul implementiert. Es wird über die API angesprochen und kann in einem laufenden Kernel geladen werden, ohne ihn neu

kompilieren oder rebooten zu müssen. Das Modul wird in einer *Ad-Hoc Support Library* (ASL) zur Verfügung gestellt. Im *Kernel Space* wurde ein *Route_Check* Modul hinzugefügt, in dem eine Tabelle enthalten ist. Die Tabelleneinträge sind Zeitmarken, von den verwendeten Routen. Das Modul wird durch den Netfilter's[a] POST_ROUTING registriert und bei jedem Paket aus der *Packet-Forwarding Function* aktualisiert. Über Linux's */proc* Dateisystem kann der *Routing Daemon* im *User Space* die Zeitmarken in der *Route_Check* Tabelle im *Kernel Space* auslesen.

.

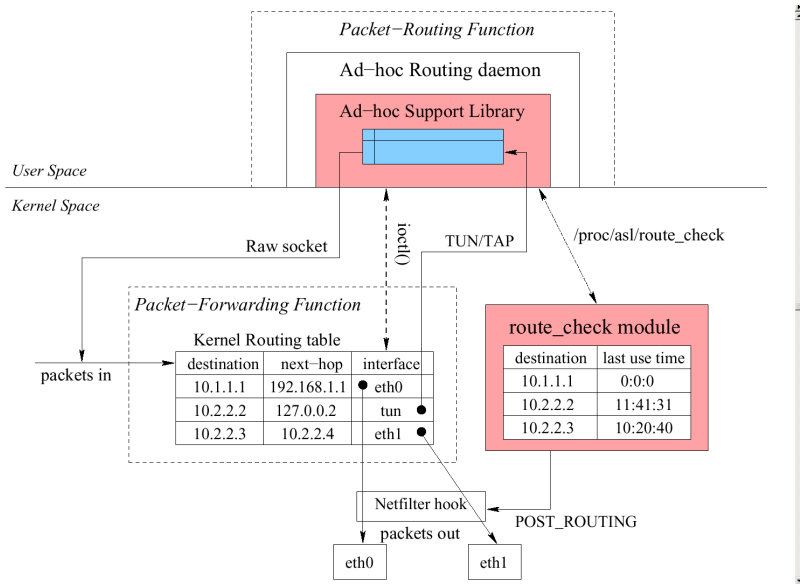


Abbildung 15: Erweiterungen für Reactive MANET Protokolle

Ein Paket, bei der die Route noch nicht bekannt ist, wird an die *Universal* TUN/TAP (tun) Geräteschnittstelle gesendet. Diese benachrichtigt den *Ad-Hoc Routing Daemon* durch einen asynchronen *Stream*, der durch einen *File Descriptor* realisiert. Der *Daemon* liest das Paket aus dem *Stream* und speichert es solange im *User Space* bis die Zeit abgelaufen, oder eine Route gefunden wurde. Im letzterem Fall wird es wieder in die *Kernel Routing Table* mit dem *Raw Socket*[b] Mechanismus eingefügt, sonst wird es verworfen. Mit Hilfe der ioclt() Schnittstelle werden die Einträge in der *Kernel Routing Table* hinzugefügt oder gelöscht.

a. http://www.netfilter.org
b. Raw Sockets werden normalerweise verwendet, um Pakete die nicht vom Kernel explizit unterstützt werden, zu behandeln.

Durch die zwei Mechanismen (ladbare Module und Netfilter) kann das Routing Protokoll ohne Änderung des *Kernels* implementiert werden.

3.3.2. Kernel Space-based

In der *Kernel Space* Kategorie befinden sich implementierte Protokolle, die nur im *Kernel Space* ausgeführt werden. Daher muss der Kernel modifiziert werden, bevor solche Programme ausgeführt werden können.

Routing durch ARP

In [DD00] wird ein Konzept für ein AODV Routing Protokoll unter Linux vorgestellt. Die Idee ist dabei, dass vorhanden *Address Resoulution Protocol* (ARP) [WS95][St94] zu einem AODV Protokoll in *Kernel* zu erweitern. ARP wird für *Broadcast* Netzwerke, wie *Ethernet* verwendet, um ein IP zu MAC Adressen *Mapping* durchzuführen. ARP verwaltet eine eigenen ARP Tabelle, in der die Informationen für das *Mapping* enthalten ist.

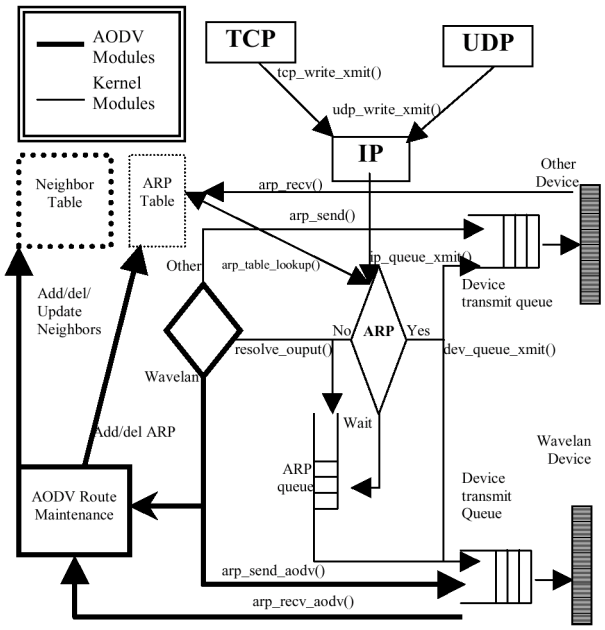


Abbildung 16: Blockdiagramm vom AODV

Abbildung 16 zeigt das modifizierte ARP Protokoll. Pakete werden von Anwendungen generiert und über *TCP* oder *UDP* an das IP übergeben. Das IP versucht ein *Mapping* der IP-Adresse zur MAC-Adresse durchzuführen. Ist dies möglich, wird das Paket an die Netzwerkkarte für das LAN

bzw. WLAN weitergeleitet. Kann kein *Mapping* durchgeführt werden, muss eine Suche für eine Route eingeleitet werden. Die IP Pakete werden während der Suche in der ARP *queue* gespeichert, und eine modifizierte ARP *Request* Nachricht wird an die umliegenden Nachbarknoten durch einem *Broadcast* gesendet. Mit Hilfe des *Flooding* wird diese Nachricht an alle weiteren Knoten verteilt. Der Knoten mit der gesuchten IP Adresse antwortet mit einem modifizierten ARP *Respond*. Empfängt der Sender die Antwort, dann aktualisiert der AODV *Route Maintenance* die ARP und *Neighbor Table*. Jetzt ist das ARP in der Lage ein *Mapping* von der IP zur MAC Adresse durchzuführen und das IP Paket in die *Device Transmit Queue* weiterzuleiten. Da die ARP *Queue* standardmäßig nur sehr wenige Pakete speichern kann, muss die Kapazität der *Queue* für *On-Demand Routing* Protokolle erhöt werden.

Modularer Click Router

Eine neue Softwarearchitektur für das Erstellen von flexiblen und konfigurierbaren *Routern* wird in [MK+99] vorgestellt. Eine ausführliche Erklärung würde den Rahmen dieser Diplomarbeit sprengen, deshalb soll eine sehr kurze Einführung in die neuen Softwarearchitektur einen groben Eindruck vermitteln. Genaueres kann in [MK+99] nachgelesen werden.

Click's Softwarearchitektur wird aus einzelnen paketverarbeitenden Modulen, sogenannten *Elements* zusammengesetzt. Eine Click-Konfiguration ist ein gerichteter Graph, dessen Knoten die *Elements* sind. Eine Kante, oder eine Verbindung, zwischen zwei Elementen stellt einen möglichen Weg für eine Paketübertragung dar.

Die wichtigsten Eigenschaften eines Elements sind:

- Element Klasse: Ähnlich wie Objekte in einem objekt-orientierten Programm. Jedes Element gehört einer Klasse an, die sein Verhalten festlegt.
- Input und Output Ports: Ports sind Endpunkte von Verbindungen zwischen Elementen. Ein Element kann beliebig viele Input und Output Ports besitzen.
- Configuration String: Einige Elementklassen verwenden zusätzliche Argumente, um den Zustand und das Verhalten festzulegen.

Abbildung 17 zeigt ein Beispielelement, "Tee(2)". "Tee" ist der Name der Klasse des Elements. Ein "Tee" Element sendet jedes Paket, das es von dem Eingang erhält an seine zwei Ausgänge. Die Zahl „2" ist der Konfiguration *String* der Klasse und gibt die Anzahl der Ausgänge an. Die dreieckigen Ports sind Eingänge und die rechteckigen Ports sind Ausgänge eines Elements.

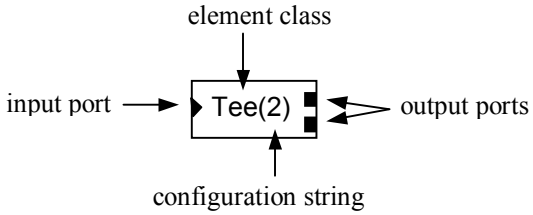


Abbildung 17: Click Beispielelement Tee

Jeder Vorgang in einer Click Softwarearchitektur wird in einem einzelnen Element abgebildet und die Kommunikationswege der Elemente werden durch Verbindungen festgelegt. In Abbildung 18 wird ein *Router* dargestellt, der eingehende Pakete zählt und danach verwirft. Die Elemente sind durch Kanten (Pfeile) verbunden, die die Ablaufrichtung der Pakete anzeigt. In diesem Fall werden die Pakete vom "FromDevice" zum "Counter" gesendet, der nach dem Zählen diese an das "Discard" Element weiterleitet, das dann die Pakete verwirft.

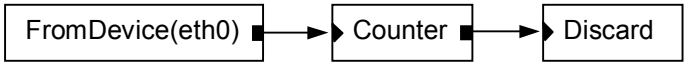


Abbildung 18: Ein zählender Router

Auf diese Weise kann ein kompletter IP *Router* zusammengestellt werden (siehe [MK+99]). Damit aber das Element "FromDevice" die eingehende Pakete erhält, muss der Kernel modifiziert werden. Der Click Code wird unter einem *bottom-half handler* ausgeführt; *bottom halves* führen Funktionen aus, die zu zeitintensiv für ein Software-Interrupt sind, aber nicht zu einem Benutzerprozess zugeordnet werden können. Wenn eine Netzwerkkarte ein Paket empfängt, dann puffert er es in einem Linux Paketpuffer und löst danach ein Software-Interrupt aus. Nach dem der *Interrupt* seine Arbeit erledigt hat, wird der *bottom-half* aufgerufen, der dann das Paket direkt an das "FromDevice" Element übergibt, ohne dass es vorher noch von Linux bearbeitet wird.
Das Click *Kernel* Modul verwendet das Linux's /proc Dateisystem, um mit Benutzerprozessen zu kommunizieren. Es ist möglich eine Click-Konfiguration während des laufenden Betriebs zu ändern. Click ist ein offenes, erweiterbares und konfigurierbares *Router Framework* und ist nur

unwesentlich langsamer als der Router in Linux 2.2.10. Click läuft als *Kernel Thread* innerhalb eines Linux 2.2 *Kernels*.

Der x-Kernel

In [HP91] wird ein neuer Betriebssystemkernel, der sogennante x-Kernel beschrieben. Er ermöglicht anhand einer expliziten Architektur, effiziente Netzwerkprotokolle schnell zu entwerfen, zu implementieren und zu testen. Der Hauptaspekt des x-Kernels ist die Bereitstellung der folgenden drei Eigenschaften:

1. Definition einer einheitlichen Menge von Abstraktionen, um Protokolle besser von den internen Vorgängen abkapseln zu können.
2. Struktuierung der Abstraktionen, damit die Interaktionen effizient sind.
3. Unterstützung von primitiven Routinen und Funktionen, die in den meisten Protokollen benötigt werden.

In der x-Kernel Architektur besteht ein Protokoll aus einer Spezifikation einer abstrakten Kommunikation, in der bestimmte Teilnehmer eine Menge von Nachrichten austauschen. Um dieses Modell zu realisieren liefert der x-Kernel drei primitive Kommunikationsobjekte: *protocols*, *sessions*, und *messages*.

Protocols Objekte haben zwei Hauptfunktionen, sie erstellen *session* Objekte und ordnen *messages*, die aus dem Netzwerk empfangen werden ihrem *session* Objekt zu.

Ein *session* Objekt ist eine Instanz von einem Protokoll, das während der Laufzeit erstellt wird. Es entspricht einem Endpunkt bei einer Verbindung durch ein Netzwerk. Es interpretiert beispielsweise Nachrichten oder verwaltet den Zustand der Verbindung.

Das *message* Objekt entspricht einer Nachricht, die mit Hilfe des *session* Objekts und *protocol* Objekts über das Netzwerk verschickt wird. Das *message* Objekt kommt entweder aus der unteren Schicht des Kernels (z.B. Netzwerkkarte) oder aus der oberen Schicht des Kernels (z.B. Anwendungen). Message Objekte werden durch **push** bzw. **pop** Operationen an obere bzw. untere *session* Objekte weitergeleitet. Das *message* Objekt wird dazu von einem Prozess von der untersten bis zur obersten Schicht bzw. von der obersten bis zu untersten Schicht durchgeschleust, ohne dass die Nachricht verändert wird, im Gegensatz zu den protokollbasierten Prozessen, bei denen die Nachricht bei jedem Wechsel der Protokollschicht in eine Warteschlange eingereiht und der Inhalt ausgetauscht wird.

Das ganze x-Kernel Kommunikationssystem ist im *Kernel Space* eingebettet. Die objekt-orientierte Infrastruktur formt einen “Kernel” im System mit individuellen Protokollen, die bei Bedarf konfiguriert werden können. Mit Hilfe eines x-Kernelsimulators können neue Protokolle getestet und die Fehler leichter gefunden werden, da das Debugging im *Kernel Space* schwieriger ist.

3.3.3. Hybrid Strategien

Hybrid implementierte Protokolle verwenden sowohl den *User Space* als auch den *Kernel Space* für das *Routing*. Protokolle, die nicht im *Kernel Space* ausgeführt werden aber den *Kernel* modifizieren, gehören ebenfalls zur diese Kategorie.

Routing durch Sockets

In [RP00] wird ein Konzept für die Implementierung eines AODV *Routing* Protokoll vorgestellt. In Abbildung 19 sind die logischen Implementierungsstrukturen dargestellt.

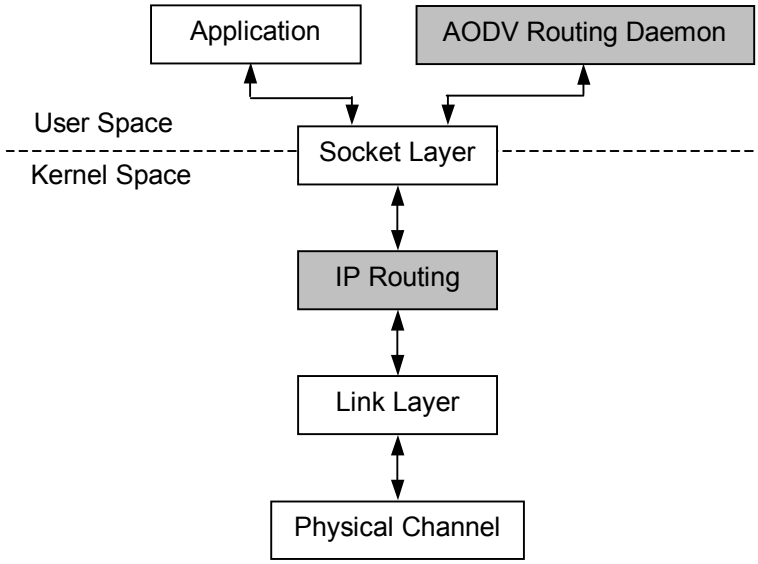


Abbildung 19: Implementierungsstruktur

Die eingefärbten Flächen entsprechen den Veränderungen bzw. Erweiterungen des ursprünglichen Linux *Codes*. Das AODV *Routing* wurde als *Daemon* im *User Space* implementiert. Er verwaltet seine eigene Route Tabelle von Empfängern bei denen eine Route vorhanden ist. Jeder Eintrag in der Route Tabelle besitzt ein *Lifetime* Feld. Ist die *Lifetime* abgelaufen, wird die Route ungültig.

Sie wird bei jeder Routenverwendung aktualisiert, damit der Eintrag nicht frühzeitig gelöscht wird. Bei Weiterleitungen von IP Paketen wird der AODV *Daemon* nicht benötigt und kann deshalb die *Lifetime* Einträge in seiner eigenen Route Tabelle nicht erneuern. Deshalb muss der *Kernel* so geändert werden, dass er die Zeitpunkte der verwendeten Routen in seiner *Kernel* Route Tabelle speichert. Läuft ein *Lifetime* Eintrag in der AODV Route Tabelle ab, dann fragt der *Daemon* in der IP nach dem Zeitpunkt des abgelaufenen Eintrags. Wenn die Route innerhalb der *Lifetime* verwendet wurde, wird der Eintrag in der AODV Route Tabelle nicht gelöscht. Statt dessen wird eine ein neuer Zeitpunkt festgelegt, zu dem der Eintrag verfällt.

Der *Daemon* kommuniziert mit dem Linux *Kernel* über *Sockets*. Bei der Initialisierung öffnet der AODV ein UDP *Socket* zum *Kernel*. Dieser *Socket* wird für das Senden und Empfangen von Kontrollnachrichten verwendet. Der AODV *Daemon* verändert die IP *Route Table* durch die Verwendung eines *Netlink*[a] *Sockets*. Wann immer AODV eine Route hinzufügen, modifizieren oder löschen möchte, sendet er eine Nachricht an den IP durch den *Netlink Socket*. Damit die Einträge nicht zu früh aus der *Kernel Routing Table* gelöscht werden, schickt der AODV regelmäßig eine *Refresh* oder *Update* Nachricht an das IP.

Die Anwendung öffnet eine *Session* und sendet ihre Pakete an das IP *Routing*. Ist der Weg für ein Paket noch nicht bekannt, benachrichtigt das IP den Routing *Daemon* über den *Netlink Socke*t für einen *Location Discovery.* Dazu übergibt das IP die IP-Adresse des Empfängers und die Portnummer der Anwendung. Kann der *Daemon* keine Route finden, dann benachrichtigt er die Anwendung anhand der Portnummer ihre *Session* abzubrechen.

Das IP wird so verändert, das während der Suche einer Route einen *Dummy* Route Tabellen Eintrag mit einer relativen kurzen Auslaufszeit generiert wird. Das IP trägt das Paket in die Warteschlange mit der *Dummy* Route ein. Wenn der AODV *Daemon* nicht in der Lage war eine Route zufinden, dann läuft die Zeit des *Dummy* Eintrags ab und das Paket wird verworfen. Ansonsten aktualisiert der Daemon den *Dummy* Eintrag mit der korrekten Route und leitet das Paket an die *Link Layer* Schicht weiter.

a. *Netlink Sockets* sind Socket, die mit dem Parameter AF_NETLINK erzeugt wurden. (Anm. des Autors)

4. Anforderungsanalyse

In diesem Kapitel werden die Anforderungen und Bewertungen an das *Beaconing* Protokoll, an den *Location Service* Protokoll und an die Implementierung für das *Routing* Protokoll durchgeführt. In den Bewertungen wird überprüft in wieweit sie den Anforderungen genügen, und ob sie für den Einsatz in ein *Vehicular* MANET geeignet sind.

4.1. Beaconing Protokoll

Erst durch die Anforderungen an das *Beaconing* Protokoll kann eine objektive Bewertung der vorhanden *Beaconing* Protokolle, die in Kapitel 3.1 vorgestellt wurden, erfolgen.

4.1.1. Anforderungen an das Beaconing Protokoll

Die wichtigste Anforderung an ein *Beaconing* Protokoll ist eine möglichst geringe Belastung der Bandbreite. Trotzdem muss gewährleistet sein, dass Nachbarfahrzeuge bis zu einer Geschwindigkeit von 250 km/h erkannt werden. Nachbarfahrzeuge sind die Fahrzeuge, die sich innerhalb der Sendereichweite eines Fahrzeugs aufhalten.

Die Auslastung der Bandbreite wird durch den Beaconintervall geregelt. Bei einem kurzen Intervall werden häufiger *Beacons* gesendet, dadurch können auch schnell fahrende Fahrzeuge erkannt werden. Dazu wird aber mehr Bandbreite benötigt. Bei einem langen Intervall werden weniger *Beacons* gesendet, deswegen wird auch weniger Bandbreite benötigt. Ist das Intervall aber zu lang, dann können schnell fahrende Fahrzeuge nicht erkannt werden. Mit nicht erkannten Fahrzeugen kann auch keine Kommunikation stattfinden.

Außerdem muss gewährleistet sein, dass selbst bei einer großen Ansammlungen von Fahrzeugen (z.B. beim Stau auf einer Autobahnkreuzung) das *Beaconing* Protokoll das Funknetzwerk nicht überlastet. In den folgenden Unterkapiteln werden die Anzahl der Fahrzeuge für eine Autobahnkreuzung und auf einem Autobahnabschnitt statistisch berechnet.

Fahrzeuganzahl innerhalb der Funkreichweite

Bevor statistische Erhebungen gemacht werden können, müssen Kenngrößen ermittelt werden [Ne00]. Die Verkehrsdichte ist eine räumliche Kenngröße, die angibt, wie stark eine Straße belastet wird. Sie wird in der Einheit [Fz/km] angegeben. Bei einer Verkehrsdichte von über 30 Fz/km werden die Straßen umgangssprachlich als “voll” bezeichnent [Ki01].

Aus der Kenntnis, welchen Raum Fahrzeuge typischerweise in einem Stau einnehmen, lässt sich eine maximale Dichte von $p_{max} \approx 140\,Fz/km$ ableiten. Dies entspricht einem sogenannten Bruttoabstand von 7.14 Meter (inklusive Fahrzeuglänge).
Gemäß [HN94] beträgt die mittlere Fahrzeuglänge der Pkw-Fahrzeugflotte in Deutschland (einschließlich Kleinbussen) 4,5 Meter. Für Lkw kann eine durchschnittliche Länge von 12 Meter angenommen werden. Bei einem Lkw-Anteil von 10%, was als Durchschnitt für deutsche Autobahnen angenommen werden kann, ergibt sich eine mittlere Fahrzeuglänge über alle Fahrzeuge von 5,25 Meter.

Sendereichweite
In Kapitel 2.4 wird eine IEEE 802.11b Karte zur Funkübertragung verwendet. Die Funkreichweite variiert von 160 bis 550 Meter in offener Umgebung, je nach Bandbreite der Verbindung. Von einer offenen Umgebung spricht man, wenn zwischen Empfänger und dem Sender direkter Sichtkontakt besteht oder sich nur gering dämpfende Materialien (wie Holz, Gips oder Glas) befinden [EN01]. Die maximale Bandbreite wird nur unter günstigen Bedingungen erreicht. Bei größeren Entfernungen bzw. starken Dämpfungen oder stärkeren Störungen aus der Umgebung kann eine Verbindung auch auf geringere Datenraten zurückfallen, ohne dass die Applikation oder die höheren Protokoll-Schichten davon unterrichtet werden.

IEEE 802.11b Bandbreite	Sendereichweite in einer offenen Umgebung
Hoch (11 Mps)	160 m
Mittel (5,5 Mps)	270 m
Standard (2 Mps)	400 m
Niedrig (1 Mps)	550 m

Tabelle 20: Funkreichweite von IEEE 802.11b Karte

Angenommen der Funkradius einer WLAN Karte wäre kreisförmig und die Ausmaße für einen durchschnittlichen Pkw wären 5 Meter für die Länge und 4 Meter für die Breite inklusive des Sicherheitsabstands beim Parken, dann kann mit Hilfe der Tabelle 20: "Funkreichweite von IEEE 802.11b Karte" und der Gleichungen 2 die theoretisch maximale Anzahl der Autos innerhalb der

Sendereichweite einer IEEE 802.11b Karte berechnet werden. Die Gleichungen gelten nur für eine 2 dimensionale Ebene. Befindet sich das Fahrzeug zum Beispiel in einem Parkhaus, so könnte sich die Anzahl der Fahrzeuge erhöhen oder sogar verringern.

Funkfläche einer WLAN Karte:

$$F_{WLAN} = \pi \times Senderadius^2$$

Fläche eines Kleinautos:

$$F_{Kfz} = Länge \times Breite = 5m \times 4m = 20m^2$$

Theoretische maximale Anzahl der Autos innerhalb einer WLAN Karte:

$$A_{Theo} = F_{WLAN} / F_{Kfz}$$

n-spurige Autobahn (stark vereinfachte Formel):

$$A_{3S} = 2 \times Senderadius / Länge_{Kfz.} \times Anzahl_{Spuren}$$

Annahmen:
Fahrzeuge mit 110km/h benötigen 60 m Sicherheitsabstand (inkl. Autolänge)
Fahrzeuge mit 250 km/h benötigen 130m Sicherhei tsabstand (inkl. Autolänge)

Gleichungen 2: Berechnung der Autoanzahl

Theoretische Anzahl von Fahrzeugen

Die in der Tabelle 21 berechneten Autoanzahlen sind nur theoretische, gerundete Maximalwerte, die in der Praxis kaum auftreten. Die Anzahl der Fahrzeuge veranschaulicht das Problem, wenn alle 50.000 Autos ihre *Beacons* senden würden.

IEEE 802.11b Senderadius	Theoretischer Wert[a]	Autobahnkreuzug mit 12 Spuren (Stau)[a]	Autobahnabschnitt mit 6 Spuren (fliesender Verkehrt)[a]
160 m	4.000	500	30
270 m	10.000	900	50
400 m	25.000	1.300	80
550 m	50.000	1.800	110

Tabelle 21: Autoanzahl innerhalb der Sendereichweite vom IEEE 802.11b

a. Die Autoanzahl ist start gerundet. Annahme: Alle Fahrzeuge sind mit dem CarTALK 2000 Kommunikationsystem ausgestattet.

Autobahnkreuzung

In Abbildung 22 ist ein Stau auf der Autobahn dargestellt. Der Kreis stellt die Funkreichweite eines Fahrzeugs dar. Er kann von 160 bis 550 Meter schwanken, je nach Verbindungsqualität zum Funkpartner. Der durchschnittliche Bruttoabstand für jedes Fahrzeug beträgt zum nächsten Fahrzeug 7,14 Meter, somit können die Anzahl der Autos leicht berechnet werden. Die Formeln stehen in Gleichungen 2. Die stark gerundeten Ergebnisse sind in der Tabelle 21 in der Spalte "Autobahnkreuzung" aufgelistet. Damit weiterhin *Beacons* von den stehenden Fahrzeugen gesendet werden, wird angenommen, daß die Durchschnittsgeschwindigkeit der Fahrzeuge 20 km/h beträgt.

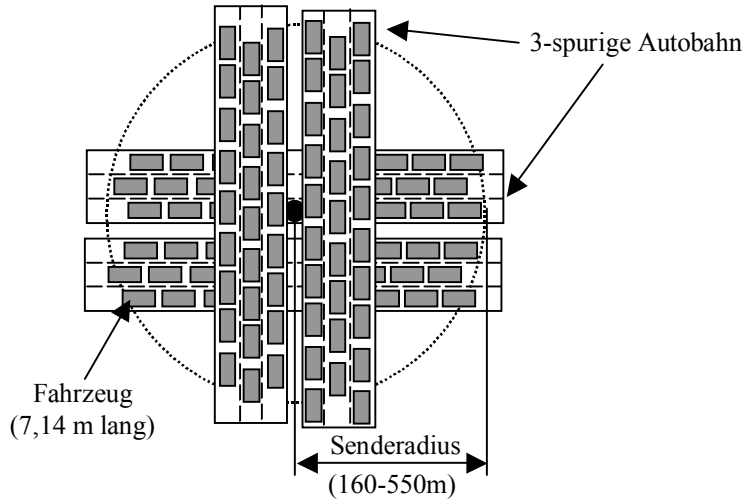


Abbildung 22: Stau auf einer Autobahnkreuzung

Autobahnabschnitt

Im Allgemeinen stehen die Fahrzeuge nicht im Stau, sondern fahren auf relativ freien Straßen. Das folgende Szenario soll eine Altagssituation darstellen, um eine grobe Abschätzung von der Autoanzahl machen zu können, die sich innerhalb des Senderadius vom IEEE 802.11b befinden. Auf einer 3-spurigen Autobahn können folgende Eigenschaften der Autos angenommen werden. Im Durchschnitt fahren die Autos 110 km/h und halten einen korrekten Sicherheitsabstand von 55 Meter[a] ein. Der Pkw ist durchschnittlich 5,25 Meter lang, daraus ergibt sich eine Gesamtlänge von ca. 60 Meter. Mit insgesamt 6 Bahnen (einschließlich dem Gegenverkehr) kann aus der

a. Sicherheitsabstand ist die halbe Tachometerzahl in Metern.

Formel, die in Tabelle 2 im Abschnitt "n-spurige Autobahn" dargestellt ist, die Autoanzahl innerhalb des Sendebereichs ausgerechnet werden. Das Szenario ist in Abbildung 23 zu sehen. Die gerundeten Werte sind in der Tabelle 21 in der Spalte "Autobahnabschnitt" aufgelistet.

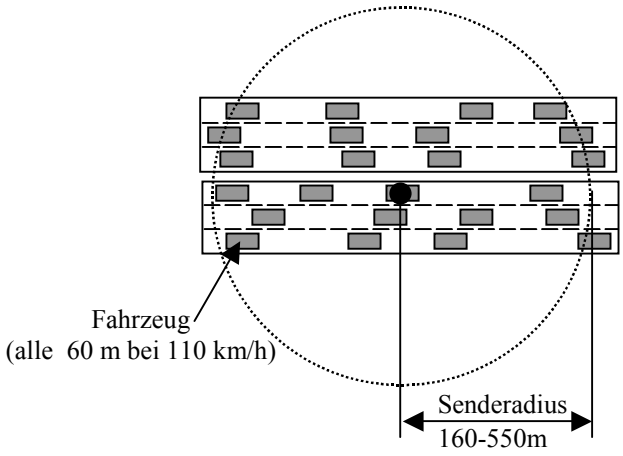


Abbildung 23: Fließender Autobahnverkehr

Bemerkung
Die in Tabelle 21 auf Seite 44 berechnete Anzahl von Autos innerhalb einer IEEE 802.11b Funkreichweite dürfen nur als grobe Richtwerte angesehen werden. Viele Faktoren, die in der realen Umgebung existieren, wurden in die Berechnung aufgrund der einfacheren Formel weggelassen.

Fazit
Die Anforderung an das *Beaconing* Protokoll ist eine möglichst geringe Belastung der Bandbreite. In den Szenarien „Stau auf einer Autobahnkreuzung“ bzw. Autobahnabschnitt mit bis zu 1.800 bzw. 110 Fahrzeugen innerhalb der Sendereichweite, darf das *Beaconing* Protokoll das Funknetzwerk durch *Beacons* nicht überlasten.

4.1.2. Bewertung der Beaconing Protokolle

In diesem Kapitel werden die *Beaconing* Protokolle, die in Kapitel 3.1 vorgestellt wurden, für den Einsatz in einem *Vehicular Routing Mechanism* untersucht und bewertet.

Damit eine möglichst objektive Bewertung der *Beaconing* Protokolle durchgeführt werden kann, sollen die Szenarien (Autobahnkreuzung und Autobahnabschnitt) in Kapitel 4.1.1 für die Berechnung der Bandbreitenauslastung herangezogen werden.
Die Bandbreitenauslastung berechnet sich aus der Beacongröße, der Übertragungsgeschwindigkeit und die Anzahl der Fahrzeuge innerhalb des Sendegebietes. Die Bandbreitenauslastung wird prozentual von der gesamten Bandbreite berechnet.

Beacongröße und -inhalt
In der Tabelle 24 sind die notwendigen Informationen aufgelistet, die in einem *Beacon* enthalten sein können.

Becon	Bemerkung	Länge in Byte
Fahrzeug ID	Fahrzeug ID für das Senden mit Unicast notwendig	16
GPS-Position	Aktuelle Ortsinformation für das Routing.	8
Fahrtrichtung	Ausrichtung des Fahrzeugs für das Routing.	2
Straßenname	Name der Straße auf dem sich das Fahrzeug befindet.	4
Fahrzeuggeschwin-digkeit	Aktuelle Fahrzeuggeschwindigkeit. Für Location Service und Beconintervallberechnung.	2
Sonstiges	MAC Adresse, Paketinformationen, Zeitpunkt usw.	20

Tabelle 24: Beacongröße und -inhalt

Die Fahrzeug ID ist notwendig, um Nachrichten gezielt an Fahrzeuge zu schicken. Die GPS-Position gibt die geographische Lage des Fahrzeugs an und wird beim *Forwarding* durch *Geocast* und *Unicast* benötigt. Die Fahrtrichtung entspricht der Kompassrichtung. Mit Hilfe des Straßennamens wissen die Empfänger, auf welcher Straße sich das Fahrzeug befindet.

Rahmenformat
Das Rahmenformat von einem IEEE 802.11b in der Bitübertragungsschicht ist in Abbildung 25 dargestellt.

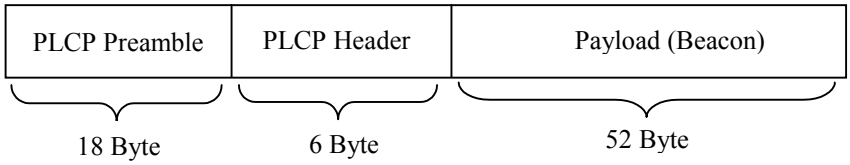


Abbildung 25: IEEE 802.11b PHY-Rahmenformat[a]

PLCP *Preamble* und PLCP *Header* sind insgesamt 24 Byte groß [Sc00]. Das *Beacon* wird in den *Payload* eingesetzt. Das ergibt eine Gesamtgröße von 76 Byte. Im Allgemeinen besteht das *Beacon* aus dem IEEE 802.11b Rahmen mit dem *Payload*.

Übertragungsdauer
Die Übertragungsdauer für das *Beacon* in der Praxis ist schwierig zu bestimmen, da der IEEE 802.11b Standard die Übertragungsgeschwindigkeit an die aktuelle Verbindungsqualität angleicht. Es ist allerdings möglich eine Unter- und Obergrenze der Übertragungszeit für ein *Beacon* zu berechnen (Beacondauer). Die Übertragungsgeschwindigkeiten bei 160 Meter Senderadius und 550 Meter Senderadius liegen bei 11 Mbps und 1 Mbps (siehe Tabelle 20).

Bei 1 Mps:

$$\text{Beacondauer\_1Mps} = \frac{\text{Preamble + Header + Beacon}}{\text{Bitübertragungsrate}_{550m}} = \frac{18\text{ Byte} + 6\text{ Byte} + 52\text{ Byte}}{1\text{ Mps}} = 0.57983\text{ ms}$$

Bei 11 Mps:

$$\text{Beacondauer\_11Mps} = \frac{\text{Preamble + Header}}{\text{Bitübertragungsrate}_{550m}} + \frac{\text{Beacon}}{\text{Bitübertragungsrate}_{160m}} = \frac{18\text{ Byte} + 6\text{ Byte}}{1\text{ Mps}} + \frac{52\text{ Byte}}{11\text{Mps}} = 0.22278\text{ ms}$$

Gleichungen 3: Übertragungsdauer für Beacons

Aus den Gleichungen 3 kann die Übertragungsdauer für 550 Meter (1 Mps) Sendereichweite berechnet werden. Bei 160 Meter Sendereichweite (bei 11Mps) muss berücksichtig werden, dass

a. im Direct Sequenze Spread Spectrum (DSSS)

die gesamte PCLP (*Preample* und *Header*) nur mit 1 Mps aber das *Payload* mit 11 Mps übertragen wird.

Die Annahme, alle Fahrzeuge würden ohne Kollision ihre *Beacons* senden ist bei hoher Auslastung des Netzwerks unwahrscheinlich, aber für die einfachere Berechnung notwendig. Bei niedriger Auslastung der Bandbreite kommt es aber selten zu Kollisionen, daher sind dann die Werte relativ genau. Damit die Bandbreitenbelastung berechnet werden kann, muss die Länge des *Beacons* und die Dauer der Übertragung festgelegt werden.

Beaconing auf der Autobahn

In Abbildung 26 ist ein Autobahnabschnitt dargestellt, in der das Fahrzeug A periodisch *Beacons* mit zwei unterschiedlichen Funkreichweiten sendet und die Fahrzeuge B und C das *Beacon* empfangen können, solange sie sich innerhalb des Senderadius befinden. Alle Fahrzeuge fahren mit einer Geschwindigkeit von 250 km/h.

Fahrzeug A fährt nach rechts und Fahrzeug B und C nach links. Offensichtlich ist die Aufenthaltsdauer der Fahrzeuge B und C innerhalb des Senderadius von Fahrzeug A abhängig von der Geschwindigkeit des Senders und Empfängers, der Funkreichweite des Senders und die Breite der Autobahn. Allgemein kann man sagen: Je breiter die Autobahn, je kürzer die Sendereichweit und je höher die Geschwindigkeitsdifferenz zwischen Sender und Empfänger ist, desto kürzer muss der Intervall sein

.

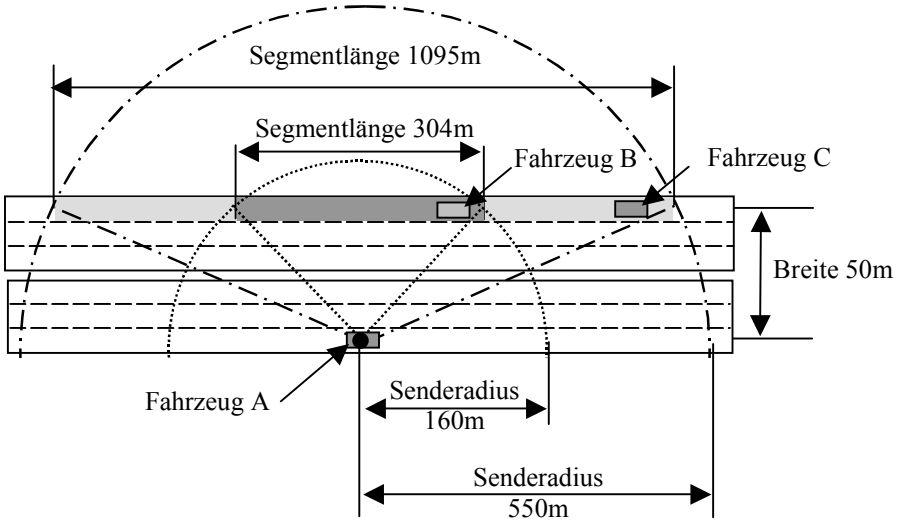


Abbildung 26: Beacon-Reichweite auf der Autobahn

Mit der Formel aus Gleichungen 4 kann die Aufenthaltsdauer im Sendegebiet vom Fahrzeug A für die Fahrzeuge B und C berechnet werden. Die Geschwindigkeitsdifferenz ergibt sich einfach durch die Addition der beiden Geschwindigkeiten von den Fahrzeugen von A und B bzw. C. Dies ist nur möglich, wenn die Fahrzeuge B und C in die entgegengesetzte Richtung des Fahrzeugs A fahren.

$$\text{Aufenthaltsdauer} = \text{Segementlänge} / \text{Geschwindigkeitsdifferenz}$$

$$\text{Segmentlänge} = \sin\left(\frac{180° - 2 \times \operatorname{asin}\left(\frac{\text{Breite}}{\text{Senderadius}}\right)}{2}\right) \times 2 \times \textit{Senderadius}$$

$$\text{Beaconintervall} = \frac{\text{Aufenthaltsdauer}}{4}$$

Bei Senderadius 160 m:

$\text{Beacondauer}_{\text{Gesamt}} = \text{Beacondauer}_{1\text{Mps}} \times \text{Fahrzeuganzahl}$

Geschwindigkeitsdifferenz = Geschw. Fahrzeug A + Geschw. Fahrzeug B

Segementlänge ≈ 304 m

Bei Senderadius 550 m:

$\text{Beacondauer}_{\text{Gesamt}} = \text{Beacondauer}_{11\text{Mps}} \times \text{Fahrzeuganzahl}$

Geschwindigkeitsdifferenz = Geschw. Fahrzeug A + Geschw. Fahrzeug C

Segementlänge ≈ 1095 m

Gleichungen 4: Berechnung der Aufenthaltsdauer

Anhand der Aufenthaltsdauer der Fahrzeuge und der verwendeten Bitübertragungsgeschwindigkeit kann die Bandbreitenauslastung für *Beacons* berechnet werden. Ausgangspunkt für die Berechnungen sind die Szenarien auf der Autobahn, die im Kapitel 4.2.1.1 dargestellt wurden.

Das Beaconintervall darf höchstens ein Viertel von der Aufenthaltsdauer des Fahrzeugs in Sendereichweite sein, damit wird gewährleistet, dass jedes Fahrzeug mindestens zwei *Beacon* von allen Nachbarfahrzeugen erhält und dadurch zwischen den Beaconintervall noch genügend Zeit für das Senden der Nachrichten zu diesem Fahrzeug übrig bleibt.

4.1.2.1. Statisches Beaconing

Das statische *Beaconing* sendet wie im Kapitel 3.1.1 beschrieben periodisch seine *Beacons* an die Nachbarfahrzeuge. Für die Berechnungen der Bandbreitenbelastung des Netzwerks ist anzunehmen, dass Fahrzeuge, die mit 70 m/s fahren (Geschwindigkeitsdifferenz von 140 m/s) sich gegenseitig noch erkennen können. Ebenfalls muss mindestens die Zeit von einem Beaconintervall für das Senden von Nachrichten vorhanden sein. Daraus ergibt sich nach den Gleichungen 4 auf einen Beaconinvertvall von 1.9555 Sekunden bei einer Segmentlänge von 1095 Metern. Die berechneten Bandbreitenauslastungen für die einzelnen Szenarien sind in Tabelle 27 aufgelistet.

Anzahl der Fahrzeuge[a] (Szenario)	Geschwindigkeits differenz	statischer Beaconintervall[b]	$Beacondauer_{Gesamt}$ [c]	Bandbreiten-auslastung[d]
1.800 (Stau auf Autobahnkreuz)	11 m/s	1.9555 s	1043.69 ms	53.4 %
1.300 (Stau auf beiden Seiten)	11 ms/s	1.9555 s	753.78 ms	38.5 %
690[e] (Stau auf einer Seite)	75.5 m/s	1.9555 s	400.08 ms	20.5 %
80 (geringer Autobahnverkehr)	140 m/s	1.9555 s	46.39 ms	2.4 %

Tabelle 27: Bandbreitenauslastung beim statischen Beaconing

a. nach der Formel aus 2“Berechnung der Autoanzahl” (siehe Seite 44) und stark gerundet.
b. Bei einer Segmentlänge von 1095 m
c. Dieser Wert ist ein optimaler Wert, d.h. es dürfen keine Kollisionen beim Senden stattfinden.
d. Wert aus der Formel: Auslastung = $Beacondauer_{Gesamt}$ / Beaconintervall
e. aus der Tabelle 31, “Autobahnverkehr,” auf Seite 59

Aufgrund der Bandbreitenauslastungen bis zu 53.4 % eignet sich das statische Beaconing Protokoll bei einer hohen Anzahl von Fahrzeugen nicht. Fällt die Anzahl von Fahrzeugen auf bis zu 80 Fahrzeugen liegt die Bandbreitenauslastungen bei 2.4 %.

Das statische *Beaconing* Protokoll kann durchaus am Anfange im Straßenverkehr eingesetzt werden. Wächst die Anzahl der Fahrzeuge, die mit „CarTALK 2000“ ausgestattet sind an, kann das statische *Beaconing* Protokoll nicht mehr verwendet werden.

4.1.2.2. Geschwindigkeitsabhängiges Beaconing

Das geschwindigkeitsabhängige *Beaconing* berechnet das Beaconintervall anhand der Geschwindigkeit des Fahrzeugs, der Sendereichweite der Funkkarte und einem Skalierungsfaktor α, wie aus den 1 “Beaconintervall vom Simple Location Service” (siehe Seite 21) zu erkennen ist. Die angenommene Sendereichweite beträgt 550 Meter und die Geschwindigkeit der Fahrzeuge entspricht der Hälfte der Geschwindigkeitsdifferenz aus der Tabelle 27. Der Skalierungsfaktor α mit dem Wert 4.02 führt zu den gleichen Beaconintervall wie sie beim statischen *Beaconing* bei einer Geschwindigkeitsdifferenz von 140 m/s berechnet wurden. Somit können die Bandbreitenauslastungen zwischen dem statischen und dem geschwindigkeitsabhängigen *Beaconing* besser verglichen werden.

Anzahl der Fahrzeuge[a] (Szenario)	Geschwindigkeit der Fahrzeuge	geschw.abh. Beaconintervall[b]	$Beacondauer_{Gesamt}$[c]	Bandbreite-auslastung[d]
1.800 (Stau auf Autobahnkreuz)	5.5 m/s	12.321 s	1043.69 ms	8.5 %
1.300 (Stau auf beiden Seiten)	5.5 ms/s	12.321 s	753.78 ms	6.1 %
650 (Stau Seite) 40 (geringer Verkehr) (Stau auf einer Seite[e])	5,5 m/s[f] 70 m/s[g]	12.321 s 1.9555 s	376.89 ms 23.19 ms	3.1 % 1.2 % = 4.3 %
80 (geringer Autobahnverkehr)	70 m/s	1.9555 s	46.39 ms	2.4 %

Tabelle 28: Bandbreitenauslastung beim geschwindigkeitsabhängigen Beaconing

a. nach der Formel aus 2“Berechnung der Autoanzahl” (siehe Seite 44) und stark gerundet.
b. Bei einer Segmentlänge von 1095 m und einen Skalierungsfaktor α von 4.02
c. Dieser Wert ist ein optimaler Wert. D.h. es dürfen keine Kollisionen beim Senden stattfinden.
d. Wert aus der Formel: Auslastung = $Beacondauer_{Gesamt}$ / Beaconintervall
e. aus der Tabelle 31, “Autobahnverkehr,” auf Seite 59
f. Fahrzeug A aus Tabelle 29, “Problem der Fahrzeugerkennung,” auf Seite 53
g. Fahrzeug B aus Tabelle 29, “Problem der Fahrzeugerkennung,” auf Seite 53

Die Bandbreitenauslastung bei einer Sendereichweite von 550 Metern sind in der Tabelle 28 aufgelistet.

Das geschwindigkeitsabhängige *Beaconing* verwendet bei einer hohen Anzahl von Fahrzeugen (Autobahnkreuz) und einer Reichweite von 550 Meter bis zu 8.5 % der Bandbreite.

Erkennungsproblem bei langsam fahrenden Fahrzeugen

Obwohl die Auslastung der Bandbreite bei 8.5 % (Stau auf Autobahnkreuz) liegt, ist das Erkennen von langsamfahrenden Fahrzeugen nicht mehr gewährleistet, da aufgrund der niedrigen Geschwindigkeit das Beaconintervall für schnell fahrende Fahrzeuge zu lange dauert. In der Abbildung 29 sind zwei Fahrzeuge (A und B) dargestellt die mit unterschiedlichen Geschwindigkeiten (5,5 ms/s und 70 m/s) vorbei fahren.

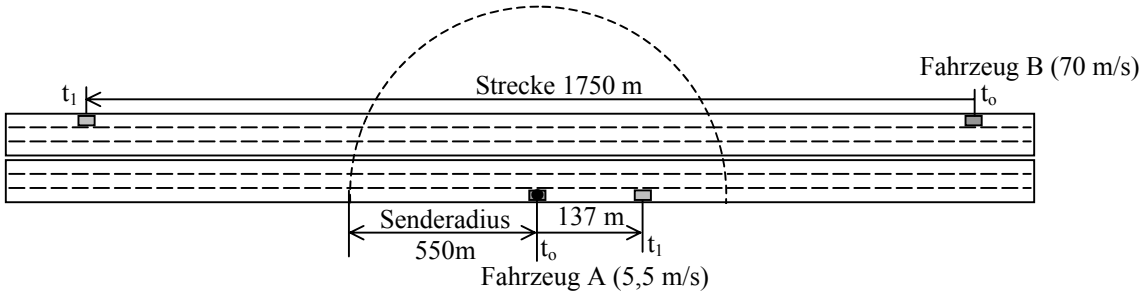


Abbildung 29: Problem der Fahrzeugerkennung

Aufgrund der Geschwindigkeit des Fahrzeugs A beträgt das Beaconintervall 25 Sekunden, aufgrund der 1 “Beaconintervall vom Simple Location Service” (siehe Seite 21) bei einem Senderadius von 550 Meter und dem Skalierungsfaktor von 4.02. Im Zeitpunkt t_o wird das erste *Beacon* und nach 25 Sekunden das zweite *Beacon* zum Zeitpunkt t_1 gesendet. In diesem Zeitraum legt das Fahrzeug A 137 Meter und Fahrzeug B 1750 Meter zurück. Fahrzeug B kann aufgrund der begrenzten Sendereichweite von 550 Meter das *Beacon* vom Fahrzeug A nicht empfangen und somit das Fahrzeug auch nicht wahrnehmen. Somit ist nicht gewährleistet, dass alle Fahrzeuge erkannt werden.

Aufgrund dieser Eigenschaft ist das geschwindigkeitsabhängige *Beaconing* für ein *Vehicular* MANET nicht geeignet.

Fazit

Durch die Berechnung der Bandbreitenauslastung in den unterschiedlichen Szenarien konnte ein objektiver Vergleich zwischen den beiden *Beaconing* Protokolle durchgeführt werden. Eine Berechnung der Bandbreitenauslastung für alle Szenarien, die in Abbildung 30 dargestellt wurden, würde den Rahmen dieser Diplomarbeit sprengen. Deshalb wurden nur drei Szenarien auf der Autobahn für die Berechnung der Bandbreitenauslastung ausgewählt.

Die berechnete Anzahl der Fahrzeuge stützen sich auf statistische Werte und können daher in der Praxis erheblich voneinander abweichen. Das statische *Beaconing* ist dem geschwindigkeitsabhängigen *Beaconing* aufgrund der Gewährleistung, dass alle Fahrzeuge erkannt werden trotz der höheren Bandbreitenauslastung vorzuziehen.

In der Realität können Kollisionen durch Nachrichten und anderen *Beacons* die Auslastung noch drastisch weiter erhöhen. Durch ein verkehrsabhängiges *Beaconing* Protokoll können die Vorteile beider *Beaconing* Protokolle vereint werden. In Kapitel 5.1 wird das Konzept für ein verkehrsabhängiges *Beaconing* Protokoll vorgestellt und erläutert.

4.2. Location Service

Die Anforderungen an den *Location Service* werden in Kapitel 4.2.1 anhand den „CarTALK 2000“ Anwendungen und den *Vehicular* MANET Eigenschaften abgeleitet. Danach erfolgt eine Bewertung der vorhanden *Location Service* Protokolle in Kapitel 4.2.2.

4.2.1. Anforderungen an den Location Service

Die Anforderungen sind in „CarTALK 2000“ Anwendungen und in Verkehrsszenarien unterteilt.

CarTalk 2000 Anwendungen

Die Anforderungen für ein *Location Service* Protokoll wird anhand der Tabelle 5 auf Seite 9 für die „CarTALK 2000“ Anwendungen abgeleitet.

Ein *Location Service* wird nur dann benötigt, wenn ein Sender eine Nachricht an einen Knoten (Empfänger) senden möchte, der sich aber außerhalb der Funkreichweite befindet.

Die CODA-Anwendung versendet mit *Unicast* ihre Nachrichten an den Empfänger. Die Entfernung zwischen Sender und Empfänger darf dabei nicht länger als 500 Meter sein (siehe Tabelle 5). Die Funkreichweite der IEEE 802.11b Karte kann bis zu 550 Meter betragen, in diesem Fall muss die Nachricht dann nicht weitergeleitet, sondern kann direkt an den Empfänger übermittelt werden.

IWF-Anwendungen senden ihre Nachrichten durch *Broadcast* bis zu einer Reichweite von 1000 Meter an andere Fahrzeuge. Ein *Location Service* wird beim Senden durch B*roadcast* nicht benötigt, da die IWF-Nachrichten an alle Fahrzeuge gerichtet sind, die sich in einem bestimmten geographischen Gebiet aufhalten. Da IWF-Nachrichten in 1000 Meter noch empfangen werden müssen, aber der IEEE 802.11b Standard „nur“ bis zu 550 Meter weit senden kann, ist eine Weiterleitung notwendig. Ein *Location Service* ist aber nicht notwendig, da IWF-Nachrichten mit *Broadcast* gesendet werden.

Ein *Location Service* sind für CBCL-Anwendungen ebenfalls nicht notwendig, da ihre Nachrichten nicht mit *Unicast,* sondern mit *Broadcast* versendet werden.

Fazit
„CarTALK 2000“ Anwendungen benötigen keinen *Location Service*. Daher gibt es keine Anforderungen an den *Location Service* für solche Anwendungen. Dennoch sollte ein *Location Service* für zukünftige Anwendungen im *Vehicular* MANET vorhanden sein.

Klassifizierung der Verkehrsszenarien
Die Anforderungen für ein *Location Service* Protokoll, das für zukünftige Anwendungen im *Vehicular* MANET eingesetzt werden soll, kann aus der Verkehrsszenarien abgeleitet werden. Dazu werden die Verkehrsszenarien in die folgenden drei Kategorien unterteilt: Autobahnverkehr, Verkehr auf der Landstraße und Stadtverkehr (siehe Abbildung 30).

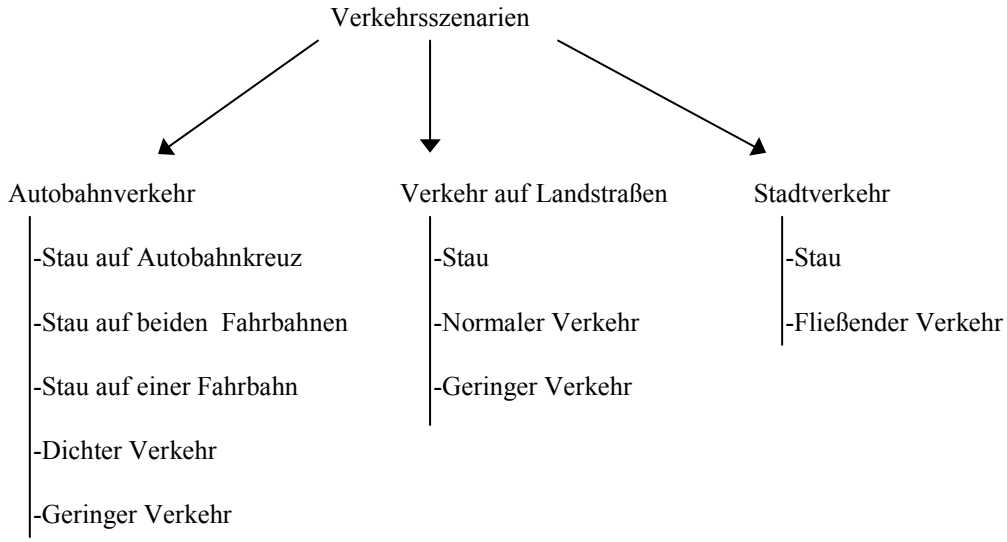


Abbildung 30: Klassifizierung der Verkehrsszenarien

Die Kategorien werden nochmals unterteilt in die einzelnen Verkehrszustände. Jeder Verkehrszustand wird in den folgenden Unterkapiteln bzw. Abschnitten dargestellt und erläutert. Dabei werden die Verkehrsdichte und die Durchschnittsgeschwindigkeit der Fahrzeuge statistisch berechnet. Dazu werden Kenngrößen eingesetzt, die in Abschnitt “Fahrzeuganzahl innerhalb der Funkreichweite” (siehe Seite 42) erläutert wurden. Aufgrund der extremen Geschwindigkeitsunterschiede zwischen den Fahrzeugen auf der Autobahn wird der Autobahnverkehr stärker untergliedert.

4.2.1.1. Autobahnverkehr (3-spurig)

Ein *Vehicular* MANET auf einer Autobahn kann die folgende Eigenschaften haben:

- Knotenausfall ist sehr selten. Meistens werden sie durch einen Unfall oder durch das Abstellen der Fahrzeuge auf den Seitenstreifen bzw. auf Autobahnraststätten hervorgerufen.
- Fahrtrichtungswechsel ist nicht möglich. Fahrzeuge können auf eine Autobahn auffahren oder sie verlassen, aber sie können die Fahrtrichtung nicht ändern.
- Fahrzeuge können eine Geschwindigkeit von bis zu 250km/h erreichen.
- Autobahnabschnitte können meistens über mehrere Kilometer lang sein.
- Kraftfahrzeuge fahren häufig in Kolonnen oder parallel zueinander (bis zu drei Fahrzeugen).

Stau auf einem Autobahnkreuz mit insgesamt 12 Spuren

Auf fast allen Spuren stehen die Fahrzeuge still. Sie stehen Stoßstange an Stoßstange und bewegen sich kaum vorwärts. Einige Autofahrer stellen ihr Fahrzeug ab. Fahrzeuge, die sich auf der rechten Spur eingeordnet haben, wollen wahrscheinlich die Autobahn wechseln und versuchen, auf die Ausfahrt zu gelangen. Andere Fahrzeuge, die von der Autobahnauffahrt her kommen, versuchen sich in den Stau einzuordnen. Es gibt für jede Fahrtrichtung eine Aus- und eine Auffahrt. Dadurch ergeben sich insgesamt Acht Aus- bzw. Auffahrten innerhalb eines Autobahnkreuzes.

Besonderheiten des Netzwerks:

Knotenausfall ist für Autobahnverhältnisse relativ hoch. Die Knoten verweilen minuten- bzw. stundenlang am selben Ort. Es können extrem viele Knoten innerhalb der Sendereichweite vorhanden sein.

Stau auf beiden Fahrbahnen der Autobahn (mit 6 Spuren einschließlich Gegenverkehr)

Auch hier stehen die Fahrzeuge Stoßstange an Stoßstange und können sich kaum vorwärtsbewegen. Aber dieses Mal ist kein Autobahnkreuz vorhanden und der Stau kann über mehrere Kilometer gehen. Fahrzeuge können nicht von der Autobahn herunter- oder neue Fahrzeuge auf die Autobahn auffahren. Fahrzeuge können aus ökonomischen Gründen abgestellt werden.

Besonderheiten des Netzwerks:

Knotenausfall ist für Autobahnverhältnisse relativ hoch. Die Knoten verweilen minuten- bzw. stundenlang am selben Ort.

Stau auf einer Fahrbahn der Autobahn (mit 6 Spuren einschließlich Gegenverkehr)
Der Stau ist nur auf einer Seite der Autobahn vorhanden, auf der Gegenfahrbahn können die Fahrzeuge unter umständen bis zur Höchstgeschwindigkeit fahren.

Besonderheiten des Netzwerks:
Knotenausfall ist für Autobahnverhältnisse relativ hoch bei Fahrzeugen, die im Stau stehen. Sie verweilen lange an einer Position. Fahrzeuge, die sich dagegen auf der staufreien Straßenseite befinden, können sich äußert schnell bewegen.

Dichter Verkehr auf einer Autobahn (mit 6 Spuren einschließlich Gegenverkehr)
Es gibt keine Aus- bzw. Auffahrten. Fahrzeuge verteilen sich auf allen Spuren. Die Dichte der Fahrzeuge ist zu gering für einen Stau, aber zu hoch, als dass Fahrzeuge ungehindert fahren könnten. Dadurch staut sich kurzzeitig der Verkehr hinter den langsam fahrenden Fahrzeugen (meistens LKWs oder PKWa mit Anhänger), die sich meistens auf der mittleren bzw. rechten Autobahnspur aufhalten. Schnell fahrende Fahrzeuge (meistens PKWs), die auf derselben Spur fahren, müssen abbremsen, wenn das Überholen aufgrund des zu dichten Verkehrs auf der linken Spur nicht möglich ist. Sie fahren dann so lange in der Kolonne, bis eine Möglichkeit besteht, über die linke Spur die langsamen Fahrzeuge zu überholen.

Besonderheiten des Netzwerks:
Es bilden sich Fahrzeugkolonnen, die in einem *Vehicular* MANET als ein Gruppenverband angesehen werden können. Die langsamen Fahrzeuge legen meistens große Entfernungen mit einer relativ konstanten Geschwindigkeit zurück.

Geringer Verkehr auf einer Autobahn (mit 6 Spuren einschließlich Gegenverkehr)
Es sind weder Aus- noch Auffahrten der Autobahn vorhanden. Alle Fahrzeuge können bis zur vorgeschriebenen Höchstgeschwindigkeit fahren, ohne dass sie von anderen Fahrzeugen behindert werden. Da die Höchstgeschwindigkeit von LKWs und PKWs mit Anhänger 80 km/h beträgt, bilden sie nur kleine Kolonnen, die von schnelleren Fahrzeugen problemlos überholt werden können.

Auf der folgenden Seite sind in der Tabelle 31 die wichtigsten Eigenschaften über einzelnen Autobahnszenarien für das *Vehicular* MANET aufgelistet.

	Autobahnkreuz (Stau auf allen Spuren)	Autobahnabschnitt (Stau auf beiden Seiten)	Autobahnabschnitt (Stau auf einer Seite)	Dichter Autobahnverkehr (mit 6 Spuren)	Geringer Autbahnverkehr (mit 6 Spuren)
Geschwindigkeitsbereich	0-20km/h	0-20km/h	0-20km/h 0-250km/h PKW[ab] 0-80km/h LKW[a]	60-130km/h (PKW) 60-80km/h (LKW)	90-250km/h (PKW) 80km/h (LKW)
Fahrzeuge innerhalb des Funkradius	2.600	1.300	690	110[b]	80[c]
Ausdehnung	1km x 1km	6 Spuren x 10km	6 Spuren x 10km	6 Spuren x 20km	6 Spuren x 20km
Maximale Anzahl der beteiligten Fahrzeuge	2.400	12.000	6.500	2.000	1.500

Tabelle 31: Autobahnverkehr

a. Auf der freien Straßenseite.
b. Durchschnittsgeschwindigkeit der PKWs liegt bei 110km/h
c. Durchschnittsgeschwindigkeit der PKWs liegt bei 150km/h

Geschwindigkeitsbereich
Der Geschwindigkeitsbereich ist ein geschätzter Bereich, in dem sich die meisten Fahrzeuge aufhalten. Wenn keine explizite Bezeichnung hinter dem Geschwindigkeitsbereich vorhanden ist, dann beziehen sich die Werte sowohl auf PKWs als auch auf LKWs. Die Höchstgeschwindigkeit für LKWa liegt in Deutschland bei 80 km/h. PKWs unterliegen keiner allgemeinen Geschwindigkeitsbegrenzung und können in diesem Szenario eine Geschwindigkeiten bis zu 250 km/h erreichen. Diese Geschwindigkeitswerte sind nur in Deutschland gültig und werden für diese Diplomarbeit verwendet, da noch keine europäischen Einheitsnormen für Höchstgeschwindigkeiten für PKWs und LKWs vorhanden sind.

Fahrzeuge innerhalb des Funkradius
Der Funkradius entspricht einer IEEE 802.11b Karte mit einer maximalen Sendereichweite von 550 Meter. IEEE 802.11b Karten können ihre Sendereichweite dynamisch verändern (siehe Tabelle 20: "Funkreichweite von IEEE 802.11b Karte". Da kleinere Senderadien das Kollisionsproblem beim Senden entschärfen würden, soll deshalb die maximale Sendereichweite (der *Worst-Case*) angenommen werden. Durch die Verwendung der maximalen Sendereichweite wird eine Obergrenze für die Anzahl der Fahrzeuge innerhalb des Senderadius festgelegt. Die verwendeten Formeln stammen aus den Gleichungen 2 auf Seite 44.

Ausdehnung
Die Ausdehnung gibt die geschätzten Dimensionen der Netzwerktopographie an, in der sich die dargestellte Szene abspielt. Dabei wird auch angenommen, dass die Nachrichten sich hauptsächlich auf dieses Gebiet beschränken.

Maximale Anzahl der beteiligten Fahrzeuge
Die maximale Anzahl von Kraftfahrzeugen kann nur sehr grob ermittelt werden. Die Annahme ist, dass jedes Fahrzeug mit dem CarTALK System ausgerüstet ist. Doch bei der Einführung des Systems werden nur sehr wenige Fahrzeuge damit ausgestattet sein, so dass die ermittelte Anzahl der beteiligten Fahrzeuge für die kommenden Jahre viel zu hoch ist. Trotzdem ist es wichtig zu prüfen, ob es später zu Problemen kommen kann, wenn die Autoanzahl bis zu diesem Wert ansteigt. Die verwendeten Formeln stammen aus den Gleichungen 2 auf Seite 44.

4.2.1.2. Verkehr auf Landstraßen

Ein *Vehicular* MANET auf einer Landstraße hat mitunter folgende Eigenschaften:

- Knotenausfall ist sehr selten. Meistens werden sie durch einen Unfall oder durch das Abstellen des Fahrzeugs bei einer langen Rotphase der Ampel hervorgerufen.
- Straßenwechsel ist häufiger. Fahrzeuge kommen am Ende einer Landstraße häufig an einer Kreuzung oder Verzweigung an.
- Die Höchstgeschwindigkeit der Kraftfahrzeuge liegt bei 100 km/h für PKWs und 80km/h für LKWs.
- Landstraßen sind meistens kürzer als Autobahnstraßen.
- Kraftfahrzeuge können in Kolonnen fahren.

Stau auf der Landstraße (auf beiden Fahrbahnen)

Situation ist ähnlich wie der "Stau auf einer Fahrbahn der Autobahn (mit 6 Spuren einschließlich Gegenverkehr)" (siehe Seite 58), aber es gibt nur eine Fahrbahn und die Gegenfahrbahn ist nicht baulich voneinander getrennt.

Besonderheiten des Netzwerks:

Die Knoten verweilen über mehrere Minuten am selben Ort. Spurwechsel ist kaum möglich, es sei denn, ein Hindernis auf der Straße wird umfahren.

Normaler Verkehr

Langsame Kraftfahrzeuge können überholt werden, wenn der Gegenverkehr es zulässt. LKWs führen meistens eine Fahrzeugkolonne an, da ein Überholen auf Landstraßen riskant und schwierig sein kann.

Besonderheiten des Netzwerks:

Fahrzeugkolonnen bleiben bis zur nächsten Kreuzung oder Verzweigung relativ stabil. Das *Vehicular* MANET kann aufgrund der zu großen Entfernungen der Fahrzeuge partitioniert werden.

Geringer Verkehr

Fahrzeuge können ohne Behinderungen durch andere Teilnehmer fahren. Die Höchst

Besonderheiten des Netzwerks:

Kolonnen sind instabil, da durch den geringen Gegenverkehr langsame Fahrzeuge ohne große Probleme überholt werden können. Durch die spärliche Anzahl von Fahrzeugen wird das Netzwerk häufig partitioniert.

In der Tabelle 32 sind die wichtigsten Eigenschaften vom *Vehicular* MANET auf einer Landstraße aufgelistet.

	Landstraße (Stau auf einer Spur)	Normaler Verkehr auf einer Landstraße	Geringer Verkehr auf einer Landstraße
Geschwindigkeitsbereich	0-20 km/h	60-90 km/h PKW 60-80 km/h LKW	90 - 100 km/h PKW 80 km/h LKW
Fahrzeuge innerhalb des Funkradius	400	40	<10
Ausdehnung	2 Spuren x 2km	2 Spuren x 10km	2 Spuren x 10km
Maximale Anzahl der beteiligten Fahrzeuge	800	400	100

Tabelle 32: Verkehr auf einer Landstraße

4.2.1.3. Stadtverkehr

Ein *Vehicular* MANET innerhalb einer Stadt hat mitunter folgende Eigenschaften:

- Knotenausfall ist sehr hoch. Meistens werden sie durch das Parken oder Abstellen des Fahrzeugs hervorgerufen.
- Straßenwechsel ist sehr wahrscheinlich. Fahrzeuge können theoretisch an jeder Kreuzung oder Verzweigung abbiegen.
- Die Höchstgeschwindigkeit der Kraftfahrzeuge kann zwischen 50 km/h und 60 km/h liegen.
- Straßen in den Städten sind meistens nur mehrere hundert Meter lang.
- Aufgrund der hohen und stark dämpfenden Baumaterialien (wie Beton, Stahl usw.) ist der Funkradius einer IEEE 802.11b Karte stark begrenzt. Deswegen werden die Fahrzeuge, die sich in einem Funkschatten aufhalten, nicht berücksichtigt. Funkschatten werden meistens von Gebäuden "geworfen", bei denen die Funkwellen nicht durch das Gebäude kommen und somit dahinter liegende Fahrzeuge nicht erreicht werden. Das gilt auch bei GPS Signalen.
- Es ist sehr wahrscheinlich, dass durch Interferenzen durch andere IEEE 802.11b Funkteilnehmer während einer Kommunikation Probleme auftreten, da der IEEE 802.11b Standard in einigen Gebäuden verwendet wird. Es ist davon auszugehen, dass die Bandbreite von den anderen Netzwerken so groß sein kann, dass eine Kommunikation zwischen den Autos kaum mehr möglich ist.
- Durch die häufigen Rotphasen der Ampeln wird der Verkehrsfluss in regelmäßigen Abständen unterbrochen. Dadurch kommt es immer wieder zur Aufteilung der Fahrzeugkolonne.

Stau

Fahrzeuge stehen Stoßstange an Stoßstange. Die Hauptstraßen und teilweise die Nebenstraßen sind überlastet. Die Hauptstraßen können aus drei Fahrspuren bestehen. Die Nebenstraßen sind kleiner und bestehen deshalb meistens nur aus einer Fahrspur.

Besonderheiten des Netzwerks:
Gruppierungen der Fahrzeuge wären möglich, da sie ihre Position nicht oder nur sehr begrenzt verändern. Das Netzwerk wird selten partitioniert.

Fließender Stadtverkehr
Beim fließenden Stadtverkehr ist es sehr schwierig, das Verhalten des Verkehrs abzuschätzen. Mit den folgenden Annahmen soll deshalb die Abschätzung erleichtert werden. Die Durchschnittsgeschwindigkeit der Fahrzeuge liegt bei 30 km/h. Sie fahren bis zu 50 km/h auf den Hauptstraßen und bis zu 40km/h auf den Nebenstraßen. Vor roten Ampeln staut sich der Verkehr, der als eine stabile Gruppe von Fahrzeugen angesehen werden kann. Bei einer grünen Phase können die meisten Autos die Ampel passieren. Sie fahren dabei meistens in unterschiedliche Richtungen.

Besonderheiten des Netzwerks:
Knotenausfälle sind sehr hoch, da Fahrzeuge häufig abgestellt werden. Gruppierungen sind vor roten Ampeln möglich, aber bei einer Grünphase zerfällt sie in kleineren Gruppen. Die Anzahl der Gruppen ist abhängig von der Anzahl der Abbiegemöglichkeiten. Je mehr Abbiegemöglichkeiten vorhanden sind, desto größer kann die Anzahl der Gruppen werden.

In der Tabelle 33 sind die Eigenschaften vom *Vehicular* MANET innerhalb einer Stadt aufgelistet.

	Stadtverkehr (Stau auf allen Straßen)	Fließender Stadtverkehr
Geschwindigkeitsbereich	0-20km/h	0-50km/h
Fahrzeuge innerhalb des Funkradius	800	400
Ausdehnung	5km x 5km	5km x 5km
Maximale Anzahl der beteiligten Fahrzeuge	15.000	7.500

Tabelle 33: Stadtverkehr

Fazit
Allgemeine Anforderungen für ein *Location Service* für ein *Vehicular* MANET sind kurze Verzögerungen bei der Bestimmung der geographischen Position des Empfängers und eine geringe Bandbreitenbelastung des *Vehicular* MANETs für *Location Dissemination* bzw. *Location Discovery.*
Die Anforderungen für ein *Location Service* sind nach den folgenden Szenarien Autobahnverkehr, Verkehr auf Landstraße, Stadtverkehr untergliedert.

Autobahnverkehr
Im Autobahnverkehr muss der *Location Service* in der Lage sein, alle Fahrzeuge in einem Gebiet von ca. einen Quadratkilometer beim Autobahnkreuz in kurzer Zeit zu erreichen, ohne dass es zum bekannten "Broadcast Storm Problem" kommt. Ebenfalls muss er die geographischen Positionen von allen Fahrzeugen, die sich auf einem Autobahnabschnitt von 10km bis 20km befinden, ermitteln können, falls das Netzwerk nicht partitioniert ist. Der Umgang von 80 bis 2.600 Fahrzeugen innerhalb des Senderadius und die Teilnahme von bis zu 12.000 Fahrzeugen auf einem Autobahnabschnitt darf für ein *Location Service* kein Problem darstellen. Geschwindigkeitsdifferenzen von bis zu 500km/h zwischen Fahrzeugen muss der *Location Service* mit umgehen können.

Verkehr auf Landstraßen
Die Eigenschaften des Verkehrs auf den Landstraßen kann der *Location Service* nutzen, indem er bei einer stabilen Kolonne die Fahrzeuge in einer Gruppen zusammenfasst, um damit Bandbreite zu schonen und die Verzögerung bei der *Location Discovery* gering zu halten.

Stadtverkehr
Im einem Stadtverkehr muss der *Location Service* mit ständigen Knotenausfällen rechnen. Der *Location Service* muss in der Lage sein, 15.000 Fahrzeuge in einem Gebiet von 5 km² zu verwalten.

4.2.2. Bewertungen der vorhandenen Location Services
Die im Kapitel 3.2 vorgestellten *Location Service* Protokolle müssen den Anforderungen, die im Kapitel 4.2.1 aufgestellt wurden, genügen.
Hauptkriterien sind Bandbreitenverbrauch bei *Location Dissemination* bzw. *Location Discovery* und die Verzögerungsdauer bei *Location Discovery.* Dazu werden die Szenarien in

“Klassifizierung der Verkehrsszenarien” (siehe Seite 56) verwendet, damit die Bewertungen der *Location Service* Protokolle möglichst objektiv ausfallen.

4.2.2.1. Simple Location Service

Das *Simple Location Service* Protokoll “SLS” (siehe Seite 25) ist ein *Proactive Location Service*, das seine *Location Update* Nachrichten (*Beacons*) mit einem *Flooding* Algorithmus im ganzen MANET verteilt. Die Aktualisierungsrate der Nachrichten hängt von der Geschwindigkeit der mobilen Knoten (Fahrzeuge) ab.

Flooding ist äußerst robust gegenüber sehr dynamischen Netzwerken wie beim *Vehicular* MANET. Bei einer geringen Anzahl von Fahrzeugen innerhalb der Funkreichweite kann *Flooding* eingesetzt werden. Hohe Geschwindigkeitsdifferenzen zwischen den Fahrzeugen bereitet Flooding kaum Probleme, da die Übertragungsdauer der *Beacons* im Millisekundenbereich liegt (siehe Gleichungen 3 auf Seite 48). In dieser Zeit legen die Fahrzeuge nur ein paar Zentimeter zurück. Die Wahrscheinlichkeit ist groß, dass *Location Update* Nachrichten übertragen werden können, ohne dass der mobile Knoten während dem Empfangen der Nachricht aus dem Funkbereich hinausfährt.

Der SLS könnte gut für die Szenarien auf der Autobahn (normaler, geringer Verkehr) und Landstraße (normaler, geringer Verkehr) eingesetzt werden. *Flooding* kann aber ab einer gewissen Dichte von Knoten zu einem sogenannten "Broadcast Storm Problem" [NT+02] führen. Deshalb ist SLS nicht so gut geeignet für Autobahn (bei Staus), Landstraße (Stau) und Stadtverkehr, da zu viele Fahrzeuge sich innerhalb der Reichweite befinden. Knotenausfälle beeinträchtigen dagegen das *Flooding* kaum, da die anderen Fahrzeuge die *Location Update* Nachrichten weiterleiten können.

4.2.2.2. DREAM

Mit der Strategie von *Distance Routing Effect Algorithm for Mobility* “DREAM” (siehe Seite 26) ist es möglich, die Anzahl der *Location Update* Nachrichten beim *Flooding* stark zu minimieren, indem die räumliche Auflösung verringert und die zeitliche Auflösung erhöht wird, d.h. die Nachrichten werden nach einer gewissen Entfernung vom mobilen Knoten verworfen und die Intervalldauer vergrößert.

Das DREAM Protokoll eignet sich sehr gut für MANETs, bei denen die Bewegung der mobilen Knoten niedrig ist, da die Intervalldauer von *Location Update* Nachrichten von der

Geschwindigkeit der einzelnen Knoten abhängt. Eine hohe Dichte von stehenden Fahrzeugen in einem begrenzten Bereich ist für das Protokoll kein Problem, da nur sehr wenige *Location Update* Nachrichten lokal verteilt werden.
DREAM ist sehr robust gegenüber einzelnen Knotenausfällen, da kein anderer Knoten von ihm abhängt. Die Verzögerung bei der Suche nach der Position des Empfängers kann durch die *Location Table* vernachlässigt werden.
In den Szenarien Autobahn (Stau auf beiden Seiten, Stau auf dem Autobahnkreuz, normaler und geringer Verkehr), Landstraße (Stau, normaler und geringer Verkehr) und Stadverkehr kann das DREAM Protokoll eingesetzt werden. In den Szenarien Autobahn (Stau auf einer Spur) kann es durch die hohe Geschwindigkeit der Fahrzeuge auf der anderen Straßenseite zu Problemen führen, da viele *Location Update* Nachrichten erzeugt und bis zu 20 km weit verteilt werden müssen.

4.2.2.3. Reactive Location Service

Das *Reactive Location Service* Protokoll "RLS" (siehe Seite 27) leitet keine *Location Update* Nachrichten an Nachbarknoten weiter, sondern verwaltet ein *Location Cache*. Darin sind die IDs und Positionen von mobilen Knoten enthalten, die sich innerhalb der Funkreichweite befinden. Durch das *Flooding* der *Location Request* Nachricht wird nach der Position des Empfängers im Netzwerk gesucht.
Wenn sehr wenige *Location Request* Nachrichten über das Netzwerk versendet werden, kann das RLS Protokoll eingesetzt werden. Eine Begrenzung der *Location Request* Nachrichten ist aber zwingend notwendig. Eine unbegrenzte Ausbreitung der Nachrichten würde das MANET schnell "verstopfen".
Hohe Geschwindigkeitsdifferenz zwischen den mobilen Knoten stellt für ein RLS Protokoll kein Problem dar. Entscheidend für einen erfolgreichen Einsatz des Protokolls im *Vehicular* MANETs ist die Häufigkeit der gesendeten *Location Request* Nachrichten vom Knoten. Bei sehr wenigen Nachrichten kann das Protokoll in jedem Szenario eingesetzt werden. Kommt es aber zu vielen *Location Request* Nachrichten, kann es bei den Szenarien mit einer hohen Knotendichte zum "Broadcast Storm Problem" kommen (z. B. Stau auf der Autobahn oder in der Stadt). Einzelne Knotenausfälle spielen bei dem RLS Protokoll kaum eine Rolle, da *Flooding* verwendet wird.

4.2.2.4. Quorum-based Location Service

Der “Quorum-based Location Service” (siehe Seite 29) baut ein Netzwerk von virtuellen *Backbones* auf, damit die *Location Update* und *Location Request* bzw. *Location Respond* Nachrichten nicht über das ganze MANET verteilt werden. Diese Nachrichten werden gezielt an die *Backbones* gesendet. Somit wird die Bandbreite effizient verwendet.

Die Ausnutzung der Bandbreite hängt auch von einer optimale Quorumgröße ab. Eine optimale Quorumgröße zu finden ist aber nicht einfach, da auf der einen Seite ein großes Quorum viel Bandbreite für das Aktualisieren und Anfragen benötigt und auf der anderen Seite aber eine große Anzahl von *Location Service* Knoten innerhalb einer Schnittmenge von zwei Quoren die Ausfallsicherheit von *Location Service* Knoten erhöhen würde. In [HL99] werden mehrere Methoden erläutert, wie ein Quorum System mit den gewünschten Eigenschaften erstellt werden kann.

Ein Nachteil vom *Quorum-based Location Service* ist der zusätzliche Kommunikationsaufwand, der bei der Bildung bzw. bei der Reorganisation von Gruppen auftritt. Wenn die Gruppen nicht lange genug stabil bleiben, kann der zusätzliche Kommunikationsaufwand die Einsparung von Bandbreite bei der Verbreitung der Knotenpositionen wieder zunichte machen.

Das Protokoll ist in der Lage, viele Knoten zu verwalten und dabei die Bandbreite zu schonen. Deshalb sind die Szenarien auf der Autobahn (Stau auf dem Autobahnkreuz, Stau auf beiden Seiten) und Stadtverkehr (Stau) besonders gut geeignet, wenn die Knotenausfallrate gering ist.

Das *Quorum-based Location Service* Protokoll benötigt für ein effektives Arbeiten virtuelle *Backbones*, die relativ zuverlässig sind und sich kaum bewegen. Beim Ausfall eines virtuellen *Backbone* oder beim Entfernen aus dem Quorum, kommt es zu einer Reorganisation des Quorums, was meistens zusätzliche Kommunikation bedeutet. Deshalb ist eine Verwendung des *Qurom-based Location Service* Protokolls für die Szenarien Autobahn (Stau auf einer Seite, normaler und fließender Verkehr), Landstraße (Stau auf einer Spur, normaler und geringer Verkehr) und fließender Stadtverkehr ungeeignet.

4.2.2.5. Grid Location Service

Durch die hierarchische Aufteilung der Umgebung in Quadrate, kann der Grid-based Location Service “GLS” (siehe Seite 31) viele mobilen Knoten verwalten, ohne unnötig viel Bandbreite zu verwenden. Die *Location Update* Nachrichten werden gezielt an andere mobilen Knoten

verschickt und die *Location Request* Nachrichten werden mit einem *Unicast* an den zyklisch nächsten Knoten gesendet.

Je weniger die mobilen Knoten sich bewegen, desto effektiver kann das GLS Protokoll die *Location Request* Nachrichten verarbeiten. Deshalb bieten die Szenarien Autobahn (Stau auf dem Autobahnkreuz, Stau auf beiden Seiten) und Stau in der Stadt gute Bedingungen, wenn die Fahrzeuge nicht abgestellt werden (Knotenausfall).

Bewegen sich die mobilen Knoten aus einer Region hinaus, dann muss die Hierarchie teilweise aktualisiert werden. Je mehr und je schneller die mobilen Knoten sich bewegen, desto häufiger kommt es zu einer Reorganisation der Hierarchie, d.h. es werden mehr Nachrichten bzw. mehr Bandbreite benötigt. Deshalb sind Szenarien wie Autobahn (Stau auf einer Seite, normaler und geringer Verkehr), Landstraße und Stadt (fließender Verkehr) für ein GLS Protokoll nicht so gut geeignet.

Ein weiteres Problem sind die Knotenausfälle, die besonders häufig im Stadtverkehr auftreten, da Fahrzeuge häufig geparkt oder durch "Funklöcher" (Tunnel, Parkhaus usw.) fahren. Bei einem Knotenausfall müssen die *Location Table* von den anderen mobilen Knoten, in der sich der ausgefallene Knoten befindet, aktualisiert werden.

Angenommen, die Ausmaße des kleinsten Quadrates für die hierarchische Aufteilung wäre so groß wie der Funkbereich einer IEEE 802.11b Funkkarte (Quadratlänge = 1100m), dann müsste ein mobiler Knoten, der sich mit 250km/h bewegt, spätestens alle 16 Sekunden seinen *Location Service* Knoten in den verschiedenen Ebenen benachrichtigen. Das wäre für ein *Vehicular* MANET nicht praktikabel.

Fazit

Die vorhandenen *Location Services* sind nur bedingt einsetzbar. Es gibt keinen *Location Service*, der alle Anforderungen für die Szenarien genügen kann. Einige *Location Services* können aufgrund der sehr dynamischen mobilen Knoten oder der hohen Knotendichte nicht ohne weiteres eingesetzt werden. Viele *Location Services* sind auf einen globalen Dienst ausgelegt, d.h. sie versuchen möglichst alle mobilen Knoten in die *Location Dissemination* bzw. *Location Discovery* miteinzubeziehen, daher arbeiten sie in einem *Vehicular* MANET nicht effizient genug. Zukünftige Anwendungen, die in den obigen definierten Szenarien (siehe "Klassifizierung der Verkehrsszenarien" (siehe Seite 56)) eingesetzt werden, benötigen aber nur einen lokalen

Location Service. Daher ist ein neue *Location Service* Strategie für ein *Vehicular* MANET notwendig.

Ein effizienter *Location Service* soll aus den vorhandenen *Location Service* Strategien entworfen werden, indem die positiven Eigenschaften übernommen und die negativen Eigenschaften soweit wie möglich eliminiert werden. Die *Location Service* Protokolle RLS und DREAM eignen sich noch am besten für ein *Vehicular* MANET, da sie die Anforderungen in den meisten Szenarien erfüllen konnten. Im Kapitel 5.2 wird ein neuer *Location Service* für ein *Vehicular* MANET vorgestellt.

4.3. Vehicular Routing Mechanism Implementierung

In diesem Kapitel werden die Anforderungen an die Implementierung für einen *Vehicular Routing Mechanism* aufgestellt. Anschließend werden die in Kapitel 3.3 vorgestellten *Ad-Hoc Routing* Protokolle aufgrund den Anforderungen bewertet.

4.3.1. Anforderungen an die VRM Implementierung

Bei der Implementierung für ein *Vehicular Routing Mechanism* Protokoll sollten folgende Gesichtspunkte berücksichtigt werden:

Implementierung des Prototyp

Ein Prototyp für einen *Vehicular Routing Mechanism* (VRM) sollte innerhalb kurzer Zeit realisierbar sein. Die Verwendung von vorhandenen Protokollen und Funktionen, die vom Standard Linux *Kernel* und von Bibliotheken angeboten werden, unterstützen den Programmierer bei seiner Implementierung und verkürzt die Implementierungsphase erheblich. Die vorhandenen Protokolle und Bibliotheksfunktionen von Linux sind im allgemeinen gut dokumentiert und können dadurch gut gewartet werden.

Modifizierung des Linux Kernel

Jede Modifizierung des Standard Linux *Kernels* kann das System instabil machen und sollte daher vermieden werden. Ein Linux *Kernel*, der modifiziert wurde muss neu kompiliert und rebootet werden, was beim Debuggen sehr zeitaufwendig werden kann.

Kernel Space versus User Space

Der *Vehicular Routing Mechanism* kann sowohl im *Kernel Space* als auch im *User Space* ausgeführt werden. Programme, die im *Kernel Space* laufen haben den Vorteil, dass sie schneller ausgeführt werden und im allgemeinen einen leichteren Zugang zu den *Header* Daten von den Paketen in den einzelnen Netzwerkschichten haben. Die Nachteile bei einer Kernelprogrammierung liegen beim erschwerten Debuggen und zeitintensiven Warten des Kernelcodes. Außerdem kann das Portieren des Kernelcodes in ein anderes System zu Problemen führen. Ab einer gewissen Programmgröße kann das *Memory Management* vom *Kernel* beeinträchtigt werden.

Aufgrund der überwiegenden Nachteile sollte das *Routing* hauptsächlich im *User Space* ausgeführt werden.

Paketweiterleitung und -speicherung
Die Weiterleitung der Pakete an die Netzwerkkarte oder Anwendung sollte effizient erfolgen, da ein Paket bis zu 1.5 kByte groß sein kann. Ein unnötiges Kopieren zwischen *User Space* und *Kernel Space* muss daher vermieden werden. Das Pakete sollte während der *Location Discovery* im *User Space* zwischengespeichert werden, da der Kernelspeicher nur begrenzt vorhanden ist und nicht ausgelagert werden kann.

4.3.2. Bewertungen der Ad-Hoc Routing Implementierungskonzepte

Die vorhandenen Implementierungskonzepte werden auf die Verwendbarkeit für ein *Vehicular Mechanism Routing* Protokoll hin untersucht. Die Bewertung richtet sich nach den Anforderungen, die in den vorherigen Kapitel herausgearbeitet worden sind.

Routing durch API
Die Idee hinter diesem Konzept ist das Aufsetzen eines neuen *On-Demand Ad-Hoc Routing* Protokolls auf einen Standard Linuxkernel, ohne die Funktionen von vorhandenen Protokollen zu behindern. Der *Kernel* muss dafür nicht verändert werden, lediglich das Laden eines kleinen Moduls für die *Route_Check* Tabelle in den *Kernel Space* ist notwendig. Das ODRM[a] läuft als *Daemon* im *User Space* und übernimmt die Funktion der *Routing Discovery*. Dadurch bleibt der Kernel "schlank" und effizient, da sich nur das *Route_Check* Modul im Kernel Speicher befindet. Fehlerhafte Programmierung beim *Routing* Daemon führen nicht gleich zum Systemabsturz und können schnell korrigiert und getestet werden, ohne den Kernel neu zu kompilieren und zu rebooten.
Durch das Umkopieren der Pakete vom *Kernel Space* in den *User Space* bei der Wegberechnung, wird der Kernelspeicher entlastet. Dafür benötigt das Zurückkopieren in den *User Space* und das Wiedereinfügen der Pakete in die *Kernel Routing Table* Zeit und Rechenleistung. Trotzdem könnte die Strategie für die Implementierung eines *Vehicular Routing Mechanism* teilweise übernommen werden, indem der *Routing Mechanism* als Modul in User Space ausgeführt wird.

Routing durch Sockets
Die Probleme bei *Routing* durch *Sockets* für eine AODV *Routing* Implementierung sind ähnlich wie in *Routing* durch API. Die hauptsächlichen Unterschiede liegen in der Kommunikation

a. On Demand Routing Module

zwischen *User Space* und *Kernel Space* und die Pufferung der Pakete bzw. die signifikante Veränderung des bestehenden *Kernels*. Der AODV wurde als *Daemon* im *User Space* implementiert, um die Änderungen im *Kernel* gering zu halten. Die veränderte IP Schicht generiert *Dummy Route* Tabelleneinträge für Pakete, bei denen noch kein Weg vorhanden ist. Dann wird über das *Netllink Socket* der AODV *Routing Daemon* mitgeteilt, dass eine *Location Discovery* benötigt wird. Die Pakete bleiben im *Kernel* solange gepuffert, bis die *Lifetime* ausläuft oder eine Route gefunden wurde. Dadurch wird das Paket nicht verworfen, sondern bleibt solange im *Kernel*. Mit dieser Strategie entfällt das Umkopieren der Pakete vom *Kernel Space* zu *User Space* und das Zurückkopieren vom *User Space* zum *Kernel Space*, wenn die *Location Discovery* erfolgreich war. Dafür wird bei langer Wegsuche und/oder vielen Paketen der Speicher vom *Kernel Space* unnötig lange reserviert. Besonders dann, wenn die *Location Discovery* keine Route für das Paket finden konnte. Damit der AODV *Daemon* seine Routen löschen kann, muss das IP eine zusätzliche Spalte in seiner *Kernel Route* Tabelle hinzufügen, in der die Zeitpunkte für die verwendeten Routen gespeichert werden.
Die Kommunikation über *Netlink* Socket ist eine Alternative zum */proc* Dateisystem unter Linux. *Socket*. Bei dem *Vehicular Routing* Mechanism werden mehrer Dienste benötigt (MGF, VLS), um eine *Route* bzw. *Location Discovery* durch zu führen. Wenn aufgrund der Ausfallsicherheit das System verteilt werden muss, kann das */proc* Dateisystem nicht mehr verwendet werden. Daher ist eine Kommunikation mit *Socket* vorzuziehen. Dieser Ansatz ist vielversprechend und soll daher im Kapitel 5.3 "Konzeption für die Implementierung eines VRM" teilweise übernommen werden.

Routing durch ARP
Routing durch ARP unterscheidet sich von den anderen Implementierungen (*Routing* durch API und *Routing* durch *Sockets*), indem es das vorhandene ARP in Linux auf ein AODV *Routing* Protokoll erweitert. Dadurch muss die IP Schicht nicht verändert werden. Das Umkopieren der IP-Pakete entfällt, da es während der *Location Discovery* in der ARP *queue* gespeichert wird. Ein *Trigger* Mechanismus (wie *Lifetime* in Routing durch API) ist ebenfalls nicht notwendig, da auf die vorhandenen Ablaufstrukturen von ARP gebaut wurde.
Dieser Entwurf hat aber einen Nachteil, da der *Kernel* nur ARP *Request* Nachrichten generiert, wenn der Empfänger zu dem selben *Subnet* gehört, oder ein host-spezifischer Routeneintrag für

den Empfänger existiert. Dadurch begrenzen sich die Anwendungen auf eine bestimmte Netzwerkkonfiguration. Ein weiteres Problem ist, dass die IP-Pakete in der ARP *queue* während der *Location Discovery* Phase verworfen werden könnten bevor die *Location Discovery* abgeschlossen werden kann.
Aufgrund dieser Nachteile ist dieser Ansatz für eine *Vehicular Routing Mechanism* Protokoll nicht zu empfehlen.

Modularer Click Router
Der modulare Click Router in [MK+99] wurde für schnelle und konfigurierbare *Router* entwickelt. Die Click-Architektur baut nicht auf vorhandenen Protokollimplementierungen von Linux auf, sondern ist weitgehend eigenständig. Dadurch sind Änderungen an der IP Implementierung im Linux *Kernel* kaum notwendig.
Durch das modulare Konzept können Änderungen in den Elementen schnell vorgenommen und getestet werden, ohne den Linux *Kernel* vorher neu zu kompilieren. Das spart Zeit und macht das System zuverlässiger und stabiler. Konfigurationsänderung des *Routers* können während der Laufzeit durchgeführt werden. Bei prototypischen Anwendungen sind diese sehr flexiblen Strukturen sehr vorteilhaft.
Nachteilig ist, dass die vorhandenen Strukturen oder Protokolle, die schon unter Linux implementiert worden sind, nicht mehr verwendet werden können, und daher viele Funktionen in der Click-Architektur nochmals implementiert und getestet werden müssen.
Ein weiteres negatives Merkmal ist, dass das ISO-OSI Schichtenmodell in der Click-Architektur nicht vorhanden ist. Schnittstellen für Anwendungen werden nicht übernommen und müssen erst neu definiert werden. Vorhandene Anwendungen, die auf der Anwendungsschicht laufen, müssen unter umständen an die Click-Schnittstelle angepasst werden.
Trotz der Vorteile überwiegen die Nachteile für eine Realisierung eines *Vehicular Routing Mechanism* in der Click Router Technik.

Der x-Kernel
Mit dem in [HP91] beschriebene x-Kernel mit der expliziten Architektur und den zur Verfügung gestellten *protocol, session* und *message* Objekten, kann in kurzer Zeit ein Netzwerkprotokoll erstellt und zusammengesetzt werden. Da nur ein Prozess für ein *message* Objekt benötigt wird, um es durch die unterschiedlichen Protokollschichten durchzuschleusen, ist der x-Kernel

effinzienter als der ursprüngliche Kernel. Dies liegt vermutlich auch daran, das alle Protokolle im *Kernel Space* laufen und daher ein aufwendiges austauschen zwischen *User* und *Kernel Space* entfällt. Dadurch wird aber der *Kernel* komplexer und größer, was das Debuggen der Software erschwert. Durch einen x-Kernelsimulator können viele Fehler lokalisiert und eliminiert werden, aber die Fehler, die mit dem x-Kernelsimulation nicht entdeckt werden können, kann das ganze System instabil oder zum Absturz bringen. Eine Korrigierung oder Änderung im *Source Code* führt jedesmal zu einem Test mit dem x-Kernelsimulator, bevor er auf einem realen Rechner ausgeführt werden kann. Ein weiterer Nachteil ist der fehlende Schutz zwischen den Protokollen, da alle Protokolle im Kerneladressraum ausgeführt werden. Aufgrund der neuen Architektur müssen vorhandene Protokolle neu implementiert und getestet werden, was einen zusätzlichen Aufwand bedeutet.
Die x-Kernelarchitektur ist für Implementierung des *Vehicular Routing Mechanism* Protokolls aufgrund der oben aufgeführten Nachteile nicht besonders gut geeignet.

Fazit
Keines der oben genannten Konzepte kann ohne Bedenken in die Implementierung eines *Vehicular Routing Mechanism* übernommen werden. Zuviele Änderungen im Linuxkernel und zu komplexe Lösungen schaden der Übersichtlichkeit und Wartungsfreundlichkeit des Systems.
Ein einfaches und effizientes Konzept für ein *Vehicular Routing Mechanism* wird im Kapitel 5.3 vorgestellt.

5. Konzeption

In diesem Kapitel werden die Konzepte für ein *Beaconig* Protokoll, ein *Location Service* Protokoll und die Implementierung des *Vehicular Routing Mechanism* (VRM) vorgestellt. Die Konzepte versuchen den Anforderungen, die an den Protokollen bzw. an der Implementierung des VRM gestellt sind zu genügen.

5.1. Verkehrsbezogene Beaconing Protokoll

Ein verkehrsbezogenes *Beaconing* Protokoll berechnet das Beaconintervall anhand von dem Szenario in der sich das Fahrzeug gerade befindet. Dadurch kann die Belastung der Bandbreite gegenüber eines statischen *Beaconings* erheblich gesenkt werden und gewährleistet trotzdem das Erkennen von Nachbarfahrzeugen.

Das verkehrsbezogene *Beaconing* Protokoll versucht das momentane Verkehrszenario in die folgenden drei Szenarien einzuordnen: Autobahn, Landstraße und Stadtverkehr (siehe Abbildung 30 auf Seite 56). Anschließend kann das Beaconintervall durch die Formel aus den Gleichungen 1 auf Seite 21 bestimmt werden, indem die zulässige Höchstgeschwindigkeit für die Fahrzeuge innerhalb des Szenarios festgestellt wird. Somit ist beispielsweise das Beaconintervall in dem Szenario „Stau auf beiden Fahrbahnen auf der Autobahn“ länger als in dem Szenario „Stau auf einer Fahrbahn auf der Autobahn“, da die Höchstgeschwindigkeit in „Stau auf beiden Fahrbahnen“ bei 20 km/h liegt, aber in „Stau auf einer Fahrbahn“ sie bis zu 250 km/h betragen kann.

5.1.1. Erkennen von Szenarien

Ein einfaches und effizientes Erkennen von Szenarien kann anhand einer digitalen Straßenkarte, der Fahrzeugdichte, der Geschwindigkeit und der Fahrtrichtung erreicht werden.

Die digitale Straßenkarte liefert den Namen der Straße und die Stadt- bzw. Ortsgrenzen, dadurch ist es möglich das Fahrzeug in die drei Szenarien (Autobahn, Landstraße und Stadt) einzuordnen. Durch die Fahrzeugdichte, der Geschwindigkeit und der Fahrtrichtung der einzelnen Fahrzeuge kann auf die aktuelle Verkehrslage geschlossen werden. Beträgt beispielsweise die Durchschnittsgeschwindigkeit der Fahrzeuge nicht über 20 km/h und die digitale Karte zeigt an, dass das Fahrzeug auf einer Autobahn befindet, dann liegt wahrscheinlich das Szenario „Autobahn: Stau auf beiden Seiten“ vor. Beträgt die Durchschnittsgeschwindigkeit auf der

gegenüberliegenden Seite weit über 20km/h, dann ist es sehr wahrscheinlich, dass dort kein Stau vorhanden ist und somit in das Szenario „Autobahn: Stau auf einer Seite“[a] zugeordnet werden kann. Mit dieser Vorgehensweise kann das Klassifizieren der Umgebung, in der sich das Fahrzeug befindet relativ einfach durchgeführt werden.

5.1.2. Bandbreitenauslastung

Anhand des Szenarios “Autobahnverkehr (3-spurig)” (siehe Seite 57) soll berechnet werden, wieviel Prozent der Bandbreite für das Senden von Beacons verwendet wird. Es muss gewährleistet sein, dass alle Fahrzeuge sich gegenseitig erkennen können. Für die Beconintervallberechung wird die Formel aus dem geschwindigkeitsabhänigen *Beaconing* Protokoll (aus den Gleichungen 1 auf Seite 21) verwendet. Der Skalierungsfaktor α ist 4.21 und die Sendereichweite beträgt 550 Meter. Die Geschwindigkeit des Fahrzeugs *v* ist die Hälfte der Geschwindigkeitsdifferenz aus den Szenarien in Tabelle 31: "Autobahnverkehr" auf Seite 59. Das Beaconintervall wird durch die Gleichungen 4 auf Seite 50 berechnet.

.

Anzahl der Fahrzeuge[a] (Szenario)	Geschwindigkeit des Fahrzeugs	szenarioabh. Beaconintervall[b]	$Beacondauer_{Gesamt}$[c]	Bandbreiten-auslastung[d]
1.800 (Stau auf Autobahnkreuz)	5.5 m/s	12.321 s	1043.69 ms	8.5 %
1.300 (Stau auf beiden Seiten)	5.5 ms/s	12.321 s	753.78 ms	6.1 %
690[e] (Stau auf einer Seite)	75.5 m/s	3.461 s	400.08 ms	11.6 %
80 (geringer Autobahnverkehr)	140 m/s	1.9555 s	46.39 ms	2.4 %

Tabelle 34: Bandbreitenauslastung beim szenarioabhängigen Beaconing

a. nach der Formel aus den Gleichungen 2 “Berechnung der Autoanzahl” (siehe Seite 44) und stark gerundet.
b. Bei einer Segmentlänge von 1095 m und einen Skalierungsfaktor α von 4.02
c. Dieser Wert ist ein optimaler Wert, d.h. es dürfen keine Kollisionen beim Senden stattfinden.
d. nach der Formel: Auslastung = $Beacondauer_{Gesamt}$ / Beaconintervall
e. aus Tabelle 31, “Autobahnverkehr,” auf Seite 59

a. “Stau auf einer Fahrbahn der Autobahn (mit 6 Spuren einschließlich Gegenverkehr)” (siehe Seite 58)

In Tabelle 34 sind die berechneten Bandbreitenauslastungen für die verschiedenen Szenarien aufgelistet.

Beim Szenario „Stau auf einer Seite“ fällt der hohe Auslastungswert von 11.6 % auf. Er kommt aufgrund des kurzen Beaconintervalls von 3.461 Sekunden zustande, da die Fahrzeuge, die im Stau stehen auch von den schnellfahrenden Fahrzeugen auf der gegenüberliegenden Fahrbahn wahrgenommen werden müssen. Dadurch werden langsamfahrende Fahrzeuge von schnellfahrenden Fahrzeugen erkannt. Dies war durch das geschwindigkeitsabhängige *Beaconing* auf Seite 52 nicht möglich.

Piggy-backing

Um weiter Bandbreite zu sparen wird das *Piggy-backing* angewendet. D.h. an die Nachricht wird das *Beacon* des Versenders angehängt, damit entfällt das explizite Senden von *Beacons* an die Nachbarfahrzeuge, wenn innerhalb des Beaconintervalls Nachrichten gesendet werden. Durch das *Piggy-backing* erhalten auch die Fahrzeuge das *Beacon*, die sich auf den Kommunikationspfad aber außerhalb der Sendereichweite des Senders befinden.

Fazit

Das verkehrsbezogene *Beaconing* Protokoll vereint die Vorteile des statischen und des geschwindigkeitsabhängigen *Beaconing* Protokolls. Es gewährleistet, dass Wahrnehmen aller Fahrzeuge, die sich innerhalb der Sendereichweite befinden, wie beim statischen *Beaconing* Protokoll. Trotzdem ist die Bandbreitenauslastung fast so gering wie beim geschwindigkeitsabhängigen *Beaconing* Protokoll.

5.2. Konzeption eines Vehicular Location Service

In diesem Kapitel soll ein neues Konzept für ein *Vehicular Location Service* entwickelt werden. Der *Location Service* muss den Anforderungen aus dem Kapitel 4.2.1 genügen.

Ein vielversprechender Ansatz für ein *Location Service* ist eine Mischung zwischen dem *Reactive Location Service "RLS" (siehe Seite 27)* und dem *Distance Routing Effect for Mobility "DREAM" (siehe Seite 26)*. Die Vorteile der beiden Protokolle sollen zusammengeführt werden, um einen effizienten *Vehicular Location Service* zu bekommen. Im folgenden Kapitel 5.2.1 wird das Konzept für die *Location Dissemination* dargestellt und danach der Algorithmus für den *Location Discovery* in Kapitel 5.2.2 erläutert. Abschließend wird in Kapitel 5.2.3 über die Erfüllung der Anforderungen diskutiert. Der Begriff „mobile Knoten", der in Kapitel 3.2 verwendet wurde, soll aufgrund des besseren Verständnis durch den Begriff „Fahrzeug" ersetzt werden.

5.2.1. Location Dissemination

Die Intervalldauer der *Location Update* Nachrichten (*Beacon*) ist nicht abhängig von der Geschwindigkeit des jeweiligen Fahrzeugs (wie beim RLS), sondern szenarienabhängig. Die szenarienabhängige Beaconintervallberechnung wurde im Kapitel 5.1 dargestellt. Der Vorteil liegt in der effizienten Bandbreiteausnutzung, d.h. bei Stau auf beiden Seiten auf der Autobahn können die Fahrzeuge die Beaconintervalle erhöhen, um Bandbreite zu sparen. Trotzdem ist gewährleistet, dass die Fahrzeuge von anderen Fahrzeugen erkannt werden, wenn sie sich innerhalb des Funkradius aufhalten.

Location Table / Location Cache

Der *Vehicular Location Service* verwaltet eine *Location Table* und eine *Neighborhood Table*. Die empfangenen *Location Update* Nachrichten werden in der *Neighborhood Table* gespeichert, und dann in die *Location Table* übertragen, sobald sich das Fahrzeug außerhalb der Sendereichweite befindet oder sich nach einer bestimmten Zeit (*Timeout*) nicht mehr meldet. Der Zeitpunkt, in der das Fahrzeug außerhalb der Sendereichweite fährt, kann anhand der Funkreichweite und Geschwindigkeitsdifferenz zwischen den beiden Fahrzeugen berechnet werden, da im *Beacon*[a]

a. In der Tabelle 24 "Beacongröße und -inhalt" (siehe Seite 47)

die Geschwindigkeit und Fahrtrichtung enthalten sind. Die empfangenen *Location Update* Nachrichten werden nicht an andere mobilen Knoten weitergeleitet.

In der *Location Table* sind Einträge über Fahrzeuge enthalten, die sich außerhalb der Funkreichweite befinden und deren aktuelle Position anhand der alten Position, Richtung und Geschwindigkeit geschätzt werden kann. Jede geschätzte Position wird mit einer Wahrscheinlichkeit bewertet. Sie gibt an wie gut die geschätzte Position des Fahrzeugs ist. Sie ist stark abhängig von dem Szenario in der sich das Fahrzeug gerade befindet. Steht ein Fahrzeug beispielsweise im Stau auf einer Autobahn, kann sehr genau die Position für die nächsten Minuten oder sogar Stunden vorhergesagt werden, solange der Stau sich nicht auflöst. Es ist ebenfalls eine gute Abschätzung möglich, wenn das Fahrzeug auf einem freien Autobahnabschnitt mit einer konstanten Geschwindigkeit fährt und es dabei die Autobahn nicht verlässt. In den Szenarien mit normalem und geringem Verkehr auf der Landstraße oder fließendem Stadtverkehr, kann die Position des Fahrzeugs nur in einem sehr kleinen Zeitraum voraus berechnet werden. Häufige Geschwindigkeitsänderungen und ein Straßenwechsel des Fahrzeugs machen eine Berechnung für eine längeren Zeitraum kaum möglich. Allgemein gilt, je größer der Zeitraum für die geschätzte Position wird, desto ungenauer wird sie. Der Faktor Szenario spielt dabei auch noch eine wesentliche Rolle. Der *Location Service* muss daher in der Lage sein ein Szenario zu erkennen, um dadurch die Wahrscheinlichkeit für die geschätzte Position der Fahrzeuge besser berechnen zu können. Fällt die Wahrscheinlichkeit unter einen bestimmten Wert wird der Eintrag aus der Location Table gelöscht und muss explizit durch eine *Location Discovery* erfragt werden, wenn mit diesem Fahrzeug kommuniziert werden soll.

5.2.2. Location Discovery

Der Vorteil bei reaktiven *Location Service* Protokollen liegt an den effizienten Ausnutzung der Bandbreite, da *Location Request* Nachrichten nur auf Anfrage (On-demand) verschickt werden. Sind alle Positionen der Empfänger bekannt, dann werden auch keine *Location Request* Nachrichten von den Sendern gesendet. Somit wird das MANET nicht durch unnötige *Location Dissemination* Nachrichten belastet, was bei proaktiven *Location Service* Protokollen durch *Location Update* (*Beacons*) Nachrichten der Fall wäre.

Flooding

Die *Location Request* Nachrichten werden mit Hilfe eines *Flooding*-Algorithmus verteilt, da sie ausgesprochen gut im sehr dynamischen MANET funktionieren. Um ein "Broadcast Storm Problem" zu vermeiden, soll die in [BS+00] vorgestellte WaitTime Funktion (WT) verwendet werden. Durch die WT Funktion leiten nur die Fahrzeuge Nachrichten weiter, die sich am Rand der Funkreichweite befinden. Dadurch wird die Nachricht schneller verbreitet. Knoten, die nach einer Wartezeit von WT(d) die gleiche Nachricht mindestens noch einmal Mal empfangen haben, leiten diese Nachricht nicht mehr weiter. Durch die Formel in den Gleichungen 5 wird die Wartezeit WT(d) für das Weiterleiten der Nachricht berechnet.

$$\mathrm{WT}(d) = \mathrm{MaxWT} \times \left(1 - \frac{\Theta}{\mathrm{Range}}\right)$$
$$\Theta = \min(d, \mathrm{Range})$$

MaxWT: maximale Wartezeit
Range: Funkreichweite

Gleichungen 5: Wartezeit Funktion

Anhand der GPS Positionen von Sender und Empfänger kann die Entfernung d berechnet werden. Die Reichweite für Range liegt bei maximalen 550 Meter. Der MaxWT Wert kann nur geschätzt werden, er ist abhängig von der Übertragungsdauer der *Location Request* Nachricht und der Anzahl der Fahrzeuge, die sich zwischen dem Sender und dem Empfänger aufhalten. Je mehr Fahrzeuge und je größer die *Location Request* Nachrichten sind, desto länger muss der MaxWT Wert sein. Bei einem zu kleinen Wert würden die innenliegenden Fahrzeuge zu früh die Nachricht versenden. Ist der Wert zu groß gewählt, dann würden die innenliegenden Fahrzeuge unnötig lange warten. Anhand vom *Location Cache* bzw. von der *Location Table* in denen die umliegenden Fahrzeuge gespeichert sind, könnte ein MaxWT abgeleitet werden.
Das "Broadcast Storm Problem" dürfte sich aufgrund der WT Funktion und der Tatsache, dass die Straßen eindimensional verlaufen, nicht auftreten.

Räumliche Auflösung
Damit die *Location Request* Nachrichten nicht das ganze Netzwerk überfluten, wird die Reichweite und die Anzahl der Weiterleitungen begrenzt. Dadurch wird eine räumliche Auflösung erreicht, ähnlich wie beim DREAM Protokoll. Die Reichweite kann dynamisch von der Anwendung festgelegt werden, darf aber einen Maximalwert (*MaxRange*) nicht überschreiten.

5.2.3. Eigenschaften des Vehicular Location Service

Der *Vehicular Location Service* muss den Anforderungen genügen, die in "Anforderungen an den Location Service" (siehe Seite 55) aufgelistet wurden.

Szenarien
Der *Vehicular Location Service* leitet seine *Location Update* Nachrichten (*Beacons*) nicht weiter, somit "überschwemmt" er nicht das MANET durch die ständige *Location Update* Nachrichten.
Location Request Nachrichten werden mit der WT Funktion durch einen *Flooding* Algorithmus lokal verteilt. Dadurch ist der *Location Service* sehr robust gegenüber Knotenausfällen, hohen Geschwindigkeiten und einer hohen Fahrzeugdichte, die in den Szenarien Autobahn, Landstraße und Stadtverkehr in unterschiedlichen Häufigkeiten auftreten.

Verzögerung
Die Verzögerung beim Senden der Nachrichten beim *Location Discovery* Vorgang ist sehr gering, wenn der Empfänger nicht allzuweit entfernt ist. Als Beispiel zur Berechnung der Verzögerungsdauer soll eine Nachricht insgesamt vier mal mit einer Übertragungsrate von 1 Mps weitergeleitet werden.

$$\begin{aligned}\text{Verzögerung}\,[\text{ms}] &= 2 \times \frac{\text{Preamble} + \text{Header} + \text{LoReq}}{\text{Bitübertragungsrate}} \times \text{Weiterleitungen} \\ &= 2 \times \frac{18\,\text{Byte} + 6\,\text{Byte} + 104\,\text{Byte}}{1\,\text{Mps}} \times 4 = 7.8125\,\text{ms}\end{aligned}$$

Gleichungen 6: Verzögerung bei Location Discovery

Die Verzögerung errechnet sich aus doppelter Beacongröße[a] für eine *Location Request* und *Location Response* Nachricht nach der Formel in den Gleichungen 6.

a. "Beacongröße und -inhalt" (siehe Seite 47)

Fazit

Der *Location Service* ist für ein *Vehicular* MANET konzipiert und kann aufgrund seiner Robustheit und der effizienten Verwendung der Bandbreite in allen Szenarien eingesetzt werden. Wie Effektiv der *Location Service* in der realen Umgebung sein wird, häng ab von dem verwendeten MaxWT Wert der Wartezeitfunktion und der Genauigkeit für die berechneten Positionen der einzelnen Fahrzeuge in der *Location Table*.

5.3. Konzeption für die Implementierung eines VRM

Das *Vehicular Routing Mechanism* (VRM) setzt sich, wie schon aus dem Kapitel "Kommunikationsarchitektur" (siehe Seite 18) bekannt, aus den folgenden Komponenten zusammen:

- "Map-based Geographic Forwarding" (siehe Seite 15)
- "Location Service" (siehe Seite 16)
- "Beaconing" (siehe Seite 15)

Die Arbeitsweise zwischen den einzelnen Komponenten und deren Anordnung wurde im Kapitel 2.4 "Kommunikationsarchitektur" dargestellt. In den nachfolgenden Unterkapiteln wird ein Konzept über die Kommunikation zwischen der Anwendung und dem *Vehicular Routing Mechanism* vorgeführt. Anschließend erfolgt eine ausführliche Erläuterung der Kommunikation zwischen *Kernel Space* und *User Space* mit Hilfe von Datagrammen und Paketen.

5.3.1. Die VRM Kommunikationsschicht

Die Kommunikation zwischen Anwendung und VRM erfolgt über das UDP Protokoll der Transportschicht durch kompatible BSD[a] *Socket* APIs. *Sockets* sind Mechanismen um Daten zwischen Prozessen auszutauschen. Diese Prozesse können entweder auf dem eigenen Rechner oder auf anderen Rechnern, die durch einen Netzwerk verbunden sind, ausgeführt werden. Der VRM *Router* kommuniziert mit dem Linuxkernel ebenfalls über das *Socket* API, aber verwendet dazu nicht die Transportschicht, sondern sendet bzw. empfängt die Nachrichten direkt von der Datensicherungsschicht. Diese *Sockets* werden *Raw Sockets* genannt. In einem *Raw Socket* Paket ist weder eine IP Adresse noch eine Portnummer enthalten, da sie unterhalb der Transport- bzw. Internetschicht erzeugt werden. *Raw Socket* Pakete können von der Datensicherungsschicht anhand der MAC Empfängeradresse und des Protokolltyps an den jeweiligen VRM *Router* weitergeleitet werden.

In Abbildung 35 werden die einzelnen Schichten des Internet-Referenzmodell und deren Verwendung mit dem VRM dargestellt.

Der Datenaustausch zwischen der Anwendung und dem VRM erfolgt durch Datagramme mit einer Port- und IP-Adresse, wie sie auch bei UDP Kommunikationsanwendungen verwendet werden. Bei einer Kommunikation zwischen einer „CarTALK 2000" Anwendung und dem VRM

a. Berkeley Software Distribution

übergibt die Anwendung ihr Datagramm an die Transportschicht. Das Datagramm wird anschließend durch die Transportschicht und dem VRM weitergeleitet. Damit das Datagramm an das nächste Fahrzeug gesendet werden kann, muß es vorher in ein *Raw Socket* Paket konvertiert werden. Diese Konvertierung ist innerhalb des VRMs durch die Port/IP- und Typ/MAC-Einheit dargestellt. Nach der Konvertierung in ein *Raw Socket* Paket, kann es über die Datensicherungsschicht an die IEEE 802.11b Funkkarte übergeben werden, die dann das Paket versendet.

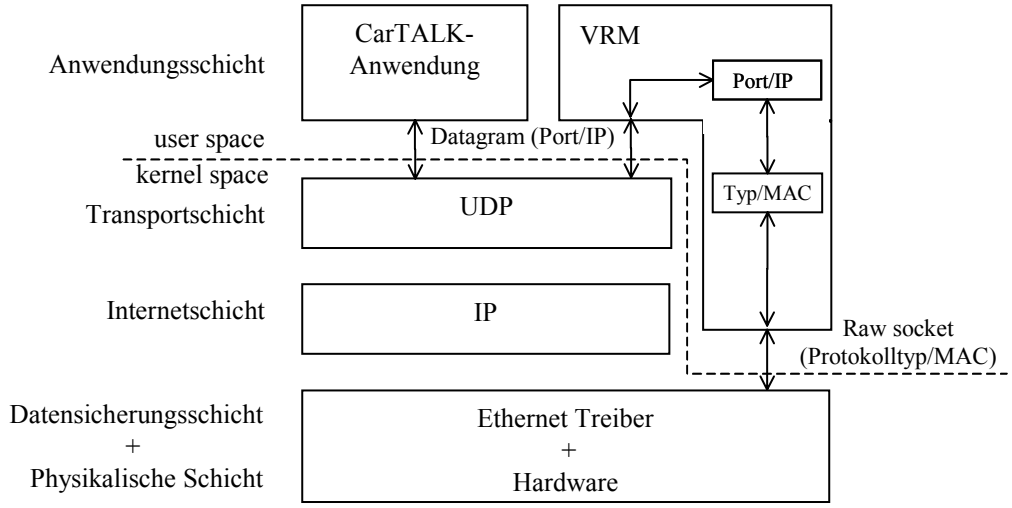


Abbildung 35: VRM Kommunikationsschicht

Empfängt die Funkkarte ein *Raw Socket* Paket, dann muss es in ein Datagramm mit Port und IP Adresse konvertiert werden, bevor es über die Transportschicht an die „CarTALK 2000“ Anwendung weitergeleitet werden kann

5.3.2. Datagramme und Pakete

In diesen Kapitel wird der Paketweg innerhalb des VRM dargestellt und erläutert. Damit zwischen den Begriffen „Datagramm“ und „Paket“ keine Verwechslung auftritt, sind „Datagramm“ und „Paket“ wie folgt definiert: Ein Datagramm ist eine Nachricht aus der Transportschicht und ein Paket ist eine Nachrichten aus der Datensicherungsschicht. In einem Paket sind zusätzliche Informationen, wie MAC-Adresse und Protokolltyp enthalten

In Abbildung 35 müssen die Datagramme bzw. die Pakete vom *kernel space* in den *user space* kopiert werden, um sie anschließend konvertieren zu können. Bei einer Weiterleitung der Datagramme an die Anwendung bzw. der Pakete an die Funkkarte müssen die Nachrichten aus dem *user space* in den *kernel space* kopiert werden. Das Kopieren erfolgt durch die Systemaufrufe sendto und recvfrom. sendto kopiert die Datagramme und Pakete vom *user space* in den *kernel space* und recfrom aus dem *kernel space* in den *user space*.
Aufgrund der Anforderung in Kapitel 2.1.2 "CarTalk 2000 Anwendungen" müssen die Pakete hinsichtlich ihrer Priorität sortiert werden. Deshalb werden alle empfangenen Datagramme und Pakete an eine zentrale Warteschlange eingereiht. Die Warteschlange befindet sich im *user space* und liefert immer das Paket oder Datagramm mit der höchsten Priorität zurück. Erst wenn das Paket verarbeitet wurde, wird das nächste Paket oder Datagramm aus der zentrale Warteschlange angefordert.

Fazit
Die „CarTALK 2000“ Anwendungen kommunizieren nicht direkt über die IP/UDP Netzwerkschicht miteinander, sondern senden ihre Nachrichten an den *Vehicular Routing Mechanism* durch einen UDP Port, der dann die Nachrichten über das *Vehicular* MANET an die jeweiligen Zielfahrzeuge schickt. Der VRM in einem Zielfahrzeug empfängt die Nachrichten und leitet sie an die „CarTALK 2000“ Anwendung weiter. Der Nachteil an dieser Vorgehensweise liegt an den spezifizierten Datagrammformaten (für *Unicast* und *Geocast*), die für die Kommunikation zwischen der Anwendung und dem VRM verwendet werden müssen. „CarTALK 2000“ Anwendungen können aufgrund der Kommunikation durch Datagrammen keine verbindungsorientierte Kommunikation herstellen (TCP).
Der Vorteil für dieses Konzept ist, dass auf die Modifizierung des Linuxkernels komplett verzichtet wurde, und damit das Debuggen und Warten der Software erheblich vereinfacht wird. Dadurch kann in einer relativ kurzen Zeit ein prototypischer *Vehicular Routing Mechansim* implementiert werden. Durch *Socket* Schnittstellen können Anwendungen und VRM unabhängig von einanderander entwickelt, implementiert und getestet werden.

6. Implementierung des Vehicular Routing Mechanism

In diesem Kapitel werden die notwendigen Änderungen aufgrund der fehlenden digitalen Straßenkarte für den MGF, *Location Service* und *Beaconing* Protokoll durchgeführt. Anschließend Werden die Implementierungen des *Vehicular Routing Mechanism* (VRM) anhand der wichtigsten Klassen und deren Funktionen mit Hilfe eines UML Diagramms dargestellt und erläutert.

6.1. Modifikation des MGF, Location Service und Beaconing

Bei der Implementierung des prototypischen *Vehicular Routing Mechanism* können nicht alle Softwarekomponenten, wie sie in Kapitel 2.3 "Softwarekomponenten" aufgelistet wurden, eingesetzt werden, da die notwendige digitale Straßenkarte, wie sie in Kapitel 2.3.3 beschrieben wurde für die Implementierung nicht zur Verfügung stand. Die folgenden Protokolle müssen deshalb so modifiziert werden, dass sie auch ohne der digitalen Straßenkarte eingesetzt werden können.

MGF

Das *Map-based Geographic Forwarding* Protokoll kann ohne Straßenkarte und Wegpunkte die Pakete nicht anhand der geographischen Entfernungen weiterleiten, sondern muss die Weiterleitung aufgrund der GPS Position der Fahrzeuge durchführen (Entfernung entspricht dann Luftlinie). Eine Weiterleitung der Pakete aufgrund der GPS-Position kann durch einen *Greedy* Algorithmus realisiert werden. Der *Greedy* Algorithmus leitet das Paket an das Fahrzeug weiter, das sich näher an der Zielposition bzw. näher an den Zielgebiet des Paketes befindet.

Location Service und Beaconing

Das *Location Service* Protokoll, wie es in Kapitel 5.2 "Konzeption eines Vehicular Location Service" entworfen wurde, kann größtenteils in die Implementierung übernommen werden. Lediglich die *Location Dissemination* aus Kapitel 5.2.1 muss modifiziert werden, da dazu das verwendete “Verkehrsbezogene Beaconing Protokoll” (siehe Seite 75) eine digitale Straßenkarte benötigt. Die Länge des Beaconintervalls hängt von dem aktuellen Szenario ab, in der sich das Fahrzeug befindet. Das Erkennen von Szenarien erfolgt anhand des Straßennamens, der von der digitalen Straßenkarte geliefert werden muss. Für die Implementierung wird das statische

Beaconing Protokoll aus Kapitel 3.1.1 aufgrund der sehr geringen Anzahl von „CarTALK 2000“ Fahrzeugen verwendet.

Überblick des Vehicular Routing Mechanism

Der prototypische VRM besteht aus den folgenden drei Softwarekomponenten: MGF (mit Greedy Algorithmus), *Location Service* und *Beaconing*. Diese sind in die VRM als gestrichelte Komponenten eingezeichnet, wie in Abbildung 36 zu sehen. Die äußere Form des VRMs entspricht der Abbildung 35 auf Seite 84, damit die Arbeitsweise des VRMs leichter verständlich wird.

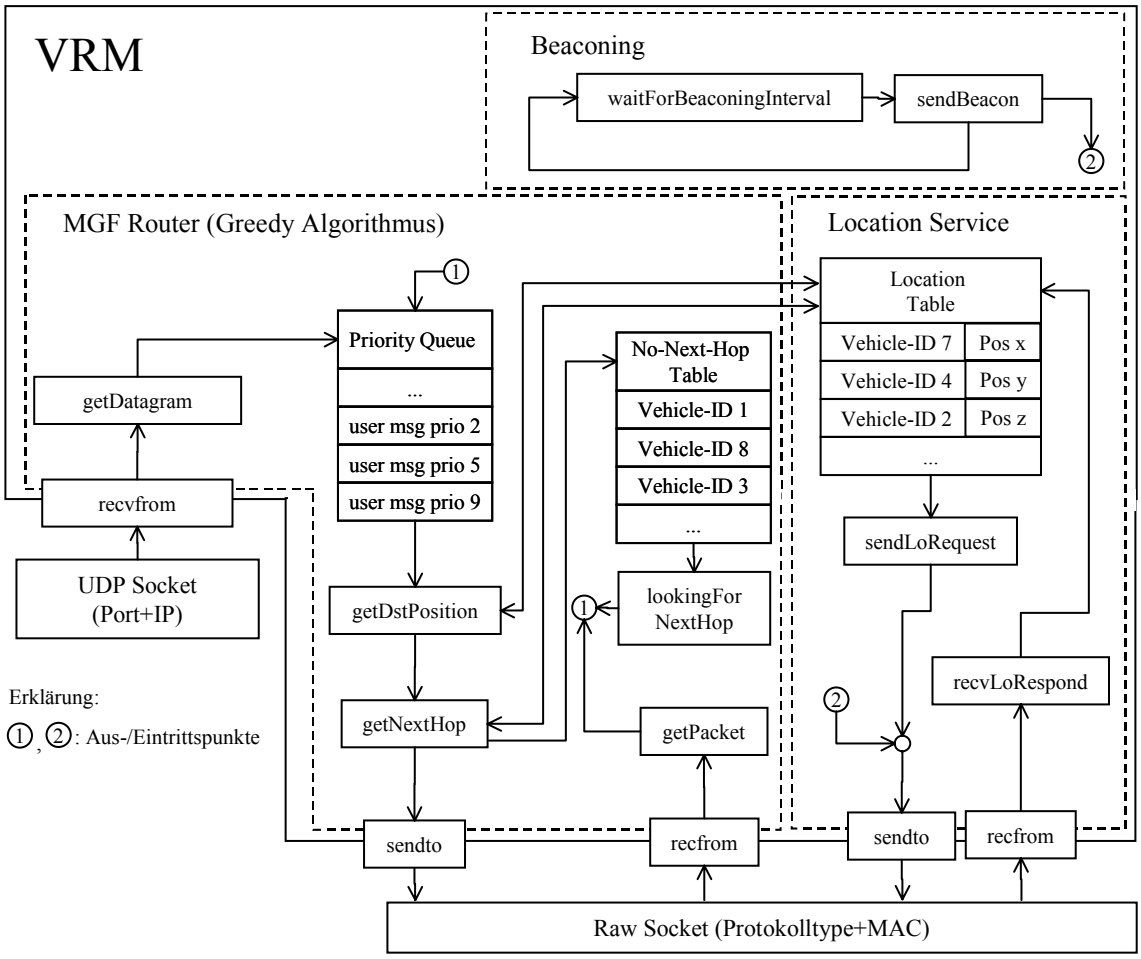


Abbildung 36: VRM - MGF Router, Location Service und Beaconing

Die Funktionalitäten der drei Komponenten sind nur grob dargestellt und sollen einen Überblick darüber verschaffen, welche Funktionen in welcher Softwarekomponenten ausgeführt wird.

Die Schnittstelle des VRMs zur Kommunikationsschicht läuft über BSD-kompatible *Sockets*. Sie sind durch das UDP- und Raw *Socket* dargestellt. Datagramme und Pakete werden ausschließlich durch die Systemaufrufe sendto und recvfrom versendet und empfangen.

Anwendung

Der MGF *Router* empfängt mit der Funktion getDatagram *Unicast* und *Geocast* Datagramme von den „CarTALK 2000“ Anwendungen und übergibt sie an die Priority Queue. Die Priority Queue liefert Datagramme und Pakete mit einer höheren Priorität zuerst aus.

Die Funktion getDstPosition erhält aus der Priority Queue ein *Unicast* Datagramm, in der eine *Vehicle-ID* enthalten ist (*Geocast* Datagramme überspringen diese Funktion, da sie nicht an ein bestimmtes Fahrzeug, sondern an ein Zielgebiet gesendet werden). Die Funktion fordert vom *Location Service* die GPS-Position des Fahrzeugs abhänging von der *Vehicle-ID* an. Der *Location Service* durchsucht daraufhin die *Location Table*. Befindet sich die Position in der *Location Table*, dann wird diese umgehend an den MGF übergeben, ansonsten leitet der *Location Service* eine *Location Discovery* ein. Dazu generiert die Funktion *sendLoRequest* ein *Location Request* Paket, das mit *Broadcast* über den Systemaufruf sendto an alle Nachbarfahrzeuge versendet wird. War die LD erfolgreich, d.h. das gesuchte Fahrzeug hat auf das *Location Request* Paket mit einem *Location Respond* Paket geantwortet, dann empfängt die Funktion recvLoRespond im *Location Service* die aktuelle Position des gesuchten Fahrzeugs. Der *Location Service* trägt daraufhin die Position in die *Location Table* ein und übergibt sie dem MGF.

Der MGF konvertiert das Datagramm in ein *Raw Socket* Paket und fügt dann die Position des Zielfahrzeugs in das *Raw Socket* Paket hinzu. Die Funktion getNextHop versucht ein geeignetes Fahrzeug zu finden, welches sich innerhalb der Sendereichweite und am nächsten zur Zielposition befindet. Die Funktion fordert dazu von dem *Location Service* das geeignete Fahrzeug an. Der *Location Service* muss dann in der *Location Table* prüfen, ob ein solches Fahrzeug vorhanden ist. Verlief die Suche erfolgreich, liefert der *Location Service* dem MGF die *Vehicle-ID*. Findet der *Location Service* kein Fahrzeug zur Weiterleitung wird das *Raw Socket* Paket in die *No-Next-Hop Table* zwischengespeichert, solange bis entweder die Funktion lookingForNextHop ein geeignetes Fahrzeug findet, oder durch einen *Timeout* das Paket verworfen wird.

Übergibt der *Location Service* die *Vehicle-ID* an die Funktion getNextHop, fügt sie die *Vehicle-ID* in das *Raw Socket* Paket ein und sendet es über die Systemfunktion sendto an das Fahrzeug.

Weiterleitung

Bei der Weiterleitung empfängt die Funktion getPacket die *Raw Socket* Pakete von der Funkkarte und leitet sie an ein Fahrzeug weiter, das sich näher an der Zielposition befindet. Dazu wird das *Raw Socket* Paket zuerst an die zentrale Warteschlange Priority Queue übergeben, damit Pakete mit höherer Priorität zuerst verarbeitet werden. Die Funktion getDstPosition wird bei *Raw Socket* Paketen übersprungen, da die Zielposition schon in den Paketen enthalten ist. Die getNextHop Funktion versucht ein geeignetes Fahrzeug zu finden, um das Paket weiterzuleiten. Raw Socket Pakete, die für dieses Fahrzeug bestimmt sind müssen, in Datagramme konvertiert und dann über die UDP Schnittstelle durch einen Socket an die „CarTALK 2000“ Anwendung geschickt werden. Dieser Vorgang ist in der Abbildung zugunsten der Übersichtlichkeit nicht eingezeichnet.

Beaconing

Das *Beaconing* läuft autonom zum VRM. Lediglich die Daten für das *Beacon* (Tabelle 24, “Beacongröße und -inhalt,” auf Seite 47) werden vom *Location Service* benötigt, um ein *Beacon* zu initialisieren. Nach der Initialisierung wird das *Beacon* mit der sendBeacon Funktion mit *Broadcast* durch die Systemfunktion sendto versendet. Danach wartet der Prozess in der Funktion waitForBeaconingInterval solange bis die Zeit für das Beaconintervall abgelaufen ist, um dann erneut ein *Beacon* mit den aktuellen Daten zu versenden.

6.2. Klassenüberblick

Insgesamt werden fast dreissig Klassen und über hundert Methoden für die Realisierung eines prototypischen VRM verwendet. Eine Darstellung aller Klassen und Methoden würde den Rahmen der Diplomarbeit sprengen, deshalb werden nur die wichtigsten Klassen und deren Methoden, die zum allgemeinen Verständnis beitragen, in Abbildung 37 dargestellt.

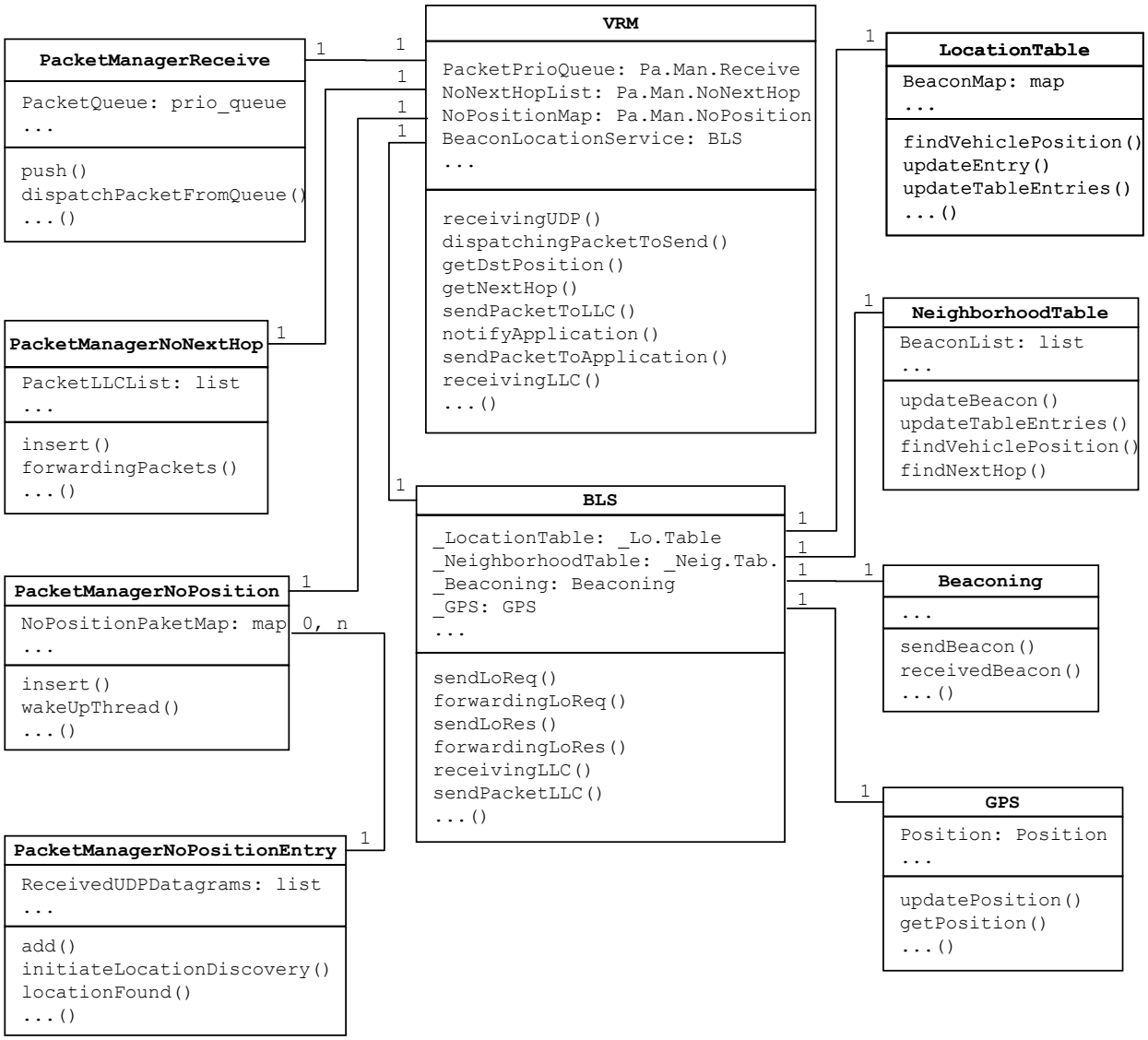


Abbildung 37: UML Klassendiagramm des VRM Routers

Die Attribute `prio_queue, list` und `map` sind Klassen aus der STL[a] und werden deshalb nicht explizit erläutert. Die Klasse `Position` besteht mitunter aus den zwei Attributen Latitude und Longitude, die eine GPS-Position beschreiben. Sie wurde zugunsten der Übersichtlichkeit nicht eingezeichnet.
Die Reihenfolge der Erläuterungen der einzelnen Klassen und ihrer Methoden entspricht ungefähr der in Kapitel 6.1 "Modifikation des MGF, Location Service und Beaconing" vorgestellten Softwarekomponenten, somit wird das Zusammenspiel der einzelnen Komponenten leichter verständlich.

VRM
Die `VRM` Klasse ist die zentrale Klasse des VRM *Routers* und beinhaltet alle Objekte, die für einen protypischen VRM benötigt werden.
Mit der Methode `receivingUDP` werden *Unicast* und *Geocast* Datagramme von den „CarTALK 2000“ Anwendungen entgegengenommen und an die `PacketPrioQueue` übergeben. Die `dispatchingPacketToSend` Methode erhält von der `PacketPrioQueue` ein Datagramm oder ein Paket. Datagramme müssen in *Raw Socket* Pakete konvertiert werden, bevor die `getDstPosition` Methode versucht, die Position des Fahrzeugs für das Paket von dem *Location Service* zu erhalten. Kann die Position nicht geliefert werden, wird das Paket dem Attribut `NoPositionMap` übergeben und die Methode beendet. Liefert der *Location Service* die Position des gesuchten Fahrzeugs, dann wird versucht mit der `getNextHop` Methode ein geeignetes Fahrzeug zu bestimmen. Wenn kein geeignetes Fahrzeug vorhanden ist, übergibt die Methode `dispatchingPacketToSend` das Paket an das Attribut `NoNextHopList` und beendet sich. Waren die Methoden `getDstPosition` und `getNextHop` jedoch erfolgreich, dann wird das Paket mit `sendPacketToLLC` weitergeleitet. *Geocast* Pakete, die sich im *forwarding* Modus befinden, rufen nur die Methode `getNextHop` auf. *Geocast* Pakete, die sich im *Broadcast* Modus befinden, benötigen weder `getDstPosition` noch `getNextHop` und werden deshalb sofort an `sendPacketToLLC` weitergeleitet.

a. Standard Template Library (siehe Website: http://www.infosys.tuwien.ac.at/Research/Component/tutorial/prwmain.htm)

Die Methode notifyApplication meldet der lokalen bzw. im Intranet befindlichen „CarTalk 2000“ Anwendung, dass entweder die Position für die Nachricht des Fahrzeugs nicht gefunden wurde oder kein Fahrzeug für das Weiterleitung der Nachricht vorhanden war.
Mit der sendPacktToApplication Methode werden Datagramme durch *Broadcast* an einen *Port* zu den „CarTALK 2000“ Anwendungen gesendet.
Die receivingLLC Methode empfängt alle Pakete aus der Datensicherungsschicht, die für den VRM bestimmt sind und übergibt sie dem PacketPrioQueue Attribut. Mit dem Attribut BeaconLocationService erhält das VRM Objekt Zugriff auf dessen Methoden, die für die getDstPosition und getNextHop notwendig sind.

PacketManagerReceive
Diese Klasse enthält eine Warteschlange, die die Priorität der Datagramme bzw. Pakete beim Auslesen berücksichtigt. Die Klasse hat einen eigenen *Thread*, der ständig versucht mit der dispatchPacketFromQueue Methode aus der Warteschlange Datagramme bzw. Pakete zu entnehmen, um sie dann der dispatchingPacketToSend Methode von VRM zu übergeben. Kehrt der *Thread* aus der dispatchingPacketToSend Methode zurück, wird das nächste Paket aus der Warteschlange entnommen, solange bis keine Pakete bzw. Datagramme mehr in der Warteschlange enthalten sind.
Mit der push Methode kann der VRM Paket oder Datagramme an die Warteschlange übergeben.

PacketManagerNoPosition
In der PacketNoPosition Klasse befinden sich alle Datagramme in einer *Hash*-Tabelle (NoPositionPaketMap) bei denen die Position der *Vehicle-ID* nicht in der *Location Table* stand. Die *Vehicle-ID* wird als *Hash-Key* verwendet, damit ein direkter Zugriff auf vorhandene Datagramme möglich ist.
Der VRM übergibt die Datagramme mit der Methode insert. Sie prüft, ob sich schon ein Datagramm mit der selben *Vehicle-ID* in der *Hash*-Tabelle vorhanden ist. Befindet sich schon ein Eintrag in der *Hash*-Tabelle, dann fügt sie das Datagramm mit der Methode add von der Klasse PacketManagerNoPositionEntry hinzu. Ist der Eintrag nicht vorhanden, dann wird ein neuer Eintrag (PacketNoPositionEntry) in der *Hash*-Tabelle angelegt. Dieser Eintrag besitzt einen eigenen Thread, der durch die initiateLocationDiscovery Methode von der PacketManagerNoPositionEntry Klasse gestartet wird.

Mit der `wakeUpThread` Methode teilt der *Location Service* dem `PacketManagerNoPosition` mit, dass eine *Location Discovery* erfolgreich war, indem die Methode den Thread mit der *Vehicle-ID* von `PacketManagerNoPositionEntry` aufweckt.

PacketManagerNoPositionEntry
Die Klasse `PacketNoPositionEntry` ist ein Element der *Hash*-Tabelle in `PacketManagerNoPosition`. Die Klasse besitzt eine Liste, in der diejenigen empfangenen Datagramme (`ReceivedUDPDatagrams`) gespeichert sind, die alle zu einer *Vehicle-ID* (Fahrzeug) gehören. Ein eigene *Thread* dieser Klasse löst mit der Methode `initiateLocationDiscovery` eine *Location Discovery* beim *Location Service* aus und geht dann für eine bestimmte Zeit in den *suspend* Zustand über. Bei erfolgreicher *Location Discovery* wird der Thread durch die Methode `locationFound` „aufgeweckt“. Sie übergibt dann alle Datagramme, die sich in der Liste befinden an die `PacketPrioQueue` mit der `push` Methode. Wenn die *Location Discovery* nicht erfolgreich war, dann läuft der *Timer* für den Thread aus, der daraufhin alle Datagramme in der Liste verwirft und die „CarTALK 2000“ Anwendung mit der `notifyApplication` Methode des VRMs benachrichtigt.

PacketManagerNoNextHop
Diese Klasse ist für die Speicherung der Datagramme bzw. Pakete zuständig, die nicht an andere Fahrzeuge weitergeleitet (*forwarding*) werden konnten. Sie werden in Liste `PacketLLCList` solange gespeichert, bis entweder ein geeignetes Fahrzeug gefunden wurde oder ein *Timer* ausläuft.
Der VRM übergibt mit der Methode `insert` die Datagramme bzw. Pakete. Ein eigenständiger Thread versucht periodisch mit der Methode `forwardingPackets` die Datagramme und Pakete weiterzuleiten, indem er für alle Fahrzeuge in der `NeighborhoodTable` die Entfernung zwischen Fahrzeug und Zielposition berechnet und anschließend das Fahrzeug mit der kürzesten Entfernung auswählt. Läuft der *Timer* für ein Paket bzw. Datagramm ab, verwirft er es.

BLS
Die BLS (*Beaconing* und *Location Service)* Klasse ist zuständig für alle Positionsanfragen der Fahrzeuge, die in dem VRM eine Rolle spielen. Deshalb sind die Attribute `_NeighborhoodTable` und `_LocationTable` enthalten. Für die Lokalisierung der eigenen

Position steht das Attribut `_GPS` zur Verfügung, das immer die aktuelle GPS-Position liefert. Mit dem `_Beaconing` Attribut werden die Nachbarfahrzeuge ständig über die eigene Position des Fahrzeugs informiert.

Mit der Methode `sendLoReq` wird ein *Location Request* Paket erstellt und die *Vehicle-ID* von dem gesuchten Fahrzeug eingefügt. Danach wird es mit *Broadcast* an die Nachbarfahrzeuge durch die `sendPacketLLC` Methode versendet. Die Nachbarfahrzeuge empfangen das *Location Request* Paket durch die `receivingLLC` Methode und leiten es mit der Methode `forwardingLoReq` weiter, oder reagieren auf das *Location Request* Paket durch ein *Location Respond* Paket durch die `sendLoRes` Methode, falls das *Location Request* Paket an dieses Fahrzeug gerichtet war.

Die `forwardingLoReq` Methode leitet die *Location Request* Pakete effizient weiter, indem es die Wartezeit mit der WT-Funktion für jedes Paket errechnet und dann solange wartet, bis die berechnete Wartezeit abgelaufen ist. Wenn nach der Wartezeit dasselbe Paket nicht noch einmal empfangen wurde, wird es mit *Broadcast* weiter geleitet. Dadurch wird das „Broadcast Storm Problem“ vermindert (“Flooding” (siehe Seite 80)).

Durch die `receivingLLC` Methode werden auch die *Location Respond* Pakete empfangen und durch die `forwardingLoRes` Methode weitergeleitet. Kann kein *next-hop* Fahrzeug für die Weiterleitung des Pakets ermittelt werden, wird es verworfen.

NeighborhoodTable

In der NeighborhoodTable Klasse befinden sich alle Fahrzeuge, die sich innerhalb der Sendereichweite aufhalten.

Dazu werden die empfangenen *Beacons* in der `BeaconList` mit der Methode `updateBeacon` gespeichert bzw. aktualisiert, falls das *Beacon* schon in der Liste vorhanden war. Ein eigener *Thread* sorgt für eine aktuelle `BeaconList`, indem er durch die `updateTableEntries` Methode periodisch die veralteten *Beacons* aussortiert und sie der LocationTable übergibt.

Die `findVehiclePosition` Methode durchsucht die `BeaconList` nach der Position einer *Vehicle-ID*, die der VRM benötigt. Die `findNextHop` Methode durchsucht die `BeaconList` nach einem geeigneten Fahrzeug, das sich näher an der Zielposition bzw. dem Zielgebiet befindet, als das eigene Fahrzeug.

LocationTable
In der `LocationTable` Klasse befinden sich alle Fahrzeuge, die sich ausserhalb der Sendereichweite aufhalten und deren Position durch ein *Location Discovery*, *Beconing,* oder *Piggy-backing* bekannt wurde. Sie werden in einer `LocationTableMap` gespeichert. Steht die Position eines Fahrzeugs in der `LocationTableMap`, dann erspart sich der der *Location Service* eine *Location Discovery.* Die `LocationTableMap` ist eine *Hash-Tabelle* mit der *Vehicle-ID* als *Hash-Key.*
Durch die Methode `updateEntry` werden die Einträge (*Beacons*) in der *Hash-Tabelle* durch den VRM bzw. *Location Service* aktualisiert. Ein eigener Thread löscht durch die `updateTableEntries` Methode die veralteten Einträge aus der `LocationTableMap`.
Die `findVehiclePosition` Methode gibt die Position des Fahrzeugs mit der *Vehicle*-ID zurück.

Beaconing
Die *Beaconing* Klasse hat die Aufgabe, regelmäßig Daten über das eigene Fahrzeug an die Nachbarfahrzeuge zu senden.
Die Methode `sendBeacon` sendet mit *Broadcast* nach Ablauf des Beaconintervalls die *Beacons* mit den aktuellen Fahrzeugdaten (“Beacongröße und -inhalt” (siehe Seite 47) an die Nachbarfahrzeuge, wie es in Kapitel 3.1.1 "Statisches Beaconing" auf Seite 21 schon erläutert wurde.
Durch die Methode `receivedBeacon` werden die *Beacons* empfangen und an die `NeighborhoodTable` durch die Methode `updateBeacon` übergeben.

GPS
Die GPS Klasse liefert die aktuellen Daten über das Fahrzeug, welches für das *Beaconing* und für den *Location Service* benötigt werden. Die GPS Klasse erhält die Daten über einen GPS-Empfänger, welches über eine serielle Schnittstelle an das Laptop verbunden ist. Die Methode `updatePosition` aktualisiert die Position des Fahrzeugs periodisch und wird bis zum Abrufen der Daten durch `getPosition` in dem Attribut `Position` zwischengespeichert. Kann das GPS-Empfänger die aktuelle Position nicht bestimmen, werden die letzten gültigen Daten an das *Beaconing* bzw. an den *Location Service* gesendet.

7. Testergebnisse von Vehicular Routing Mechanism

In diesem Kapitel werden die Verzögerungen gemessen, die bei der Versendung und der Weiterleitung von Nachrichten durch den *Vehicular Routing Mechanism* auftreten.

Aufgrund der sehr begrenzten Anzahl von vorhandenen mobilen Rechnern, Funkkarten und GPS-Empfängern können nur kleine Verkehrsszenarien simuliert werden, die aus zwei bis drei Fahrzeugen bestehen. Eine Aussage über die Eigenschaften eines *Vehicular* MANET, wie sie in den Kapitel 4.2.1 dargestellt wurden, kann dieser Test nicht machen. Getestet wird mit einem Laptop und einen Desktop Rechner in einem Raum, die über IEEE 802.11b Funknetzwerkkarten im Ad-Hoc Modus verbunden sind. Die Funkstrecke beträgt ca. einen Meter, damit Interferenzen möglichst gering bleiben und die Bandbreite von 11 MBit/s beim Testen gewährleistet wird. Obwohl die Testversuche in einem Raum durchgeführt werden, können die Resultate auf die unten dargestellen Testszenarien übertragen werden. Die Geschwindigkeiten und Entfernungen von Fahrzeugen für diese Testversuche sind irrelevant und würden die gemessenen Verzögerungs- und Antwortzeiten nicht beeinflussen, da sich Funkwellen mit Lichtgeschwindigkeit ausbreiten. Deshalb kann auf Fahrzeugen bei diesen Testversuchen verzichtet werden. Das Testen erfolgt mit den folgenden Hardware- und Softwarekomponenten:

Desktop Rechner

Der Desktop ist mit 512 MB RAM und einem AMD Athlon Prozessor, der mit 1.4 GHz getaktet wird ausgerüstet. Außerdem enthält er eine 30 GB Festplatte und eine 10/100 MBit Ethernetkarte.

Laptop

Der IBM ThinkPad A31 ist mit 512 MB RAM und einem Intel Pentium 4 Prozessor, der mit 1.8 GHz getaktet wird ausgesrüstet. Das Laptop enthält außerdem eine 40 GB Festplatte, eine interne WI-FI *Wireless* und 10/100 MBit Ethernetkarte.

IEEE 802.11b Funknetzwerkkarte für den Desktop

Der Wireless Network USB Adapter von Sitecom ist IEEE 802.11b kompatibel und sendet die Daten mit einer Frequenz von 2.4 GHz. Die Übertragungsgeschwindigkeit liegt zwischen 1 und 11 MBit/s. Er wird über einen USB Anschluss an den Desktop angeschlossen.

IEEE 802.11b Funknetzwerkkarte für das Laptop
Der Orinoco Silver Wireless LAN Adapter von Lucent Technologies erfüllt den IEEE 802.11b Standard und sendet die Daten mit einer Frequenz von 2.4 GHz. Die Übertragungsgeschwindigkeit liegt zwischen 1 und 11 MBit/s. Er besitzt einen Anschluss für eine externe Antenne, damit die Funkreichweite erhöht werden kann. Der Adapter ist mit einer PCMCIA Schnittstelle ausgesrüstet.

GPS-Empfänger
Der eTrex Summit GPS-Empfänger von GARMIN berechnet seine Position bis zu 15 Metern genau pro Sekunde. Die Auflösung des Kompasses beträgt +/- 5 Grad. Die Genauigkeit der berechneten Geschwindigkeit liegt bei +/- 0.2 km/h. Mit der vorhandenen RS-232 Schnittstelle können Daten ausgelesen werden. Das Datenformat entspricht dem NMEA-0183 Format.

Betriebssystem
Das verwendete Betriebssystem ist von Red Hat Version 7.3. Die Konfiguration entspricht der Standardeinstellung, die bei der Installation des Betriebsystems vorgeschlagen wurde. Der *Vehicular Routing Mechanism* wird unter Linux 2.4.18-3 Distribution ausgeführt.

Softwaretools
Für die Implementierung des *Vehicular Routing Mechanism* wurde die Software „KDevelop“ mit der Version 2.1.3 für die Entwicklungsumgebung verwendet.
Die Antwort- und Verzögerungszeiten wurde mit der Software „Ethereal“ Version 0.9.8 gemessen.

Testszenario mit zwei Fahrzeugen
In Abbildung 38 kommunziert Fahrzeug A mit Fahrzeug B, indem es eine Nachricht an das Fahrzeug B sendet und anschließend Fahrzeug B durch eine Rückmeldung antwortet.

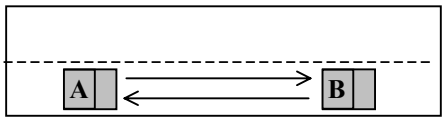


Abbildung 38: Kommunikation zwischen zwei Fahrzeugen

7.1. Antwortzeit von VRM und Anwendung

In diesem Test soll die Antwortzeit von Nachrichten zwischen zwei Fahrzeugen festgestellt werden. Die Nachricht von Fahrzeug B ist dabei immer gleich groß, wie die empfangene Nachricht von Fahrzeug A.

Das Fahrzeug A sendet unterschiedlich große Nachrichten, um festzustellen wie stark sich das Volumen der Nachricht auf die Antwortzeit auswirkt. In der ersten Versuchsreihe wird in Fahrzeug B die Kommunikationsanwendung auf dem *Routing* (VRM) Rechner ausgeführt, vergleichbar zu Abbildung 9 auf Seite 18. Bei der zweiten Versuchsreihe wird die Kommunikationsanwendung auf einem externen Rechner ausgeführt, der mit dem *Routing* (VRM) Rechner durch das *In-Car* Netzwerk verbunden ist. Damit ist es möglich, die zusätzliche Kommunikationsverzögerung im *In-Car* Netzwerk zu bestimmen.

In Abbildung 39 sind die gemessenen Resultate in einem Diagramm dargestellt. Die Byteangaben auf der x-Achse entsprechen den *Payload* im PHY-Rahmenformat, das in Abbildung 25 auf Seite 48 dargestellt wurde.

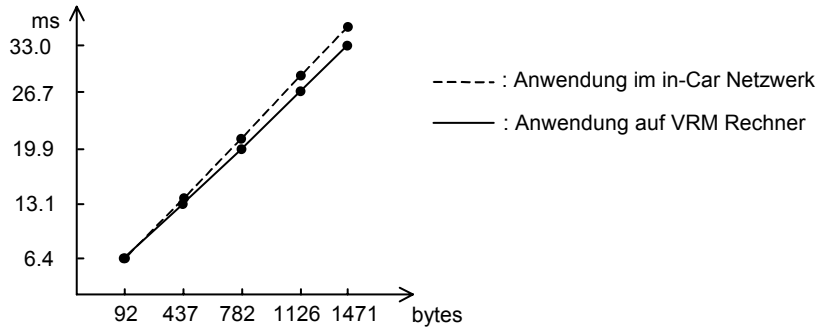


Abbildung 39: Antwortzeit zwischen zwei Fahrzeugen

Gemessen wird der Zeitraum zwischen dem Versenden und Empfangen der Nachrichten von Fahrzeug A. Jeder Messpunkt im Diagramm besteht dabei aus zehn gemessenen Verzögerungswerten. Das Diagramm zeigt einen linearen Anstieg der Antwortzeit bei größer werdenen Nachrichten. Die Antwortzeit fällt trotzdem sehr kurz aus. Bei einer Nachrichtengrößen von 92 Bytes beträgt die Antwortzeit nur 6.4 ms, selbst wenn die Anwendung auf einem anderen Rechner ausgeführt wird. Erst bei einer gewissen Größe der Nachricht gibt es einen messbaren

Unterschied, ob die Anwendung auf dem *Routing* (VRM) Rechner oder auf einem externen Rechner ausgeführt wird. Die Differenz liegt aber maximal bei 2.5 ms.

7.2. Antwortzeit beim Location Discovery

In diesem Test soll die Anwortzeit beim *Location Discovery* gemessen werden. Das Testszenario entspricht dem in Abbildung 38 in der zwei Fahrzeuge miteinander kommunizieren. Fahrzeug A sendet eine *Location Request* Nachricht an Fahrzeug B. Fahrzeug B sendet daraufhin eine *Location Respond* Nachricht an Fahrzeug A zurück. Gemessen wird der Zeitraum zwischen dem Versenden der *Location Request* Nachricht und dem Empfangen der *Location Respond* Nachricht in Fahrzeug A. Die Antwortzeit beträgt 0.135 ms, wobei insgesamt 5 *Location Discovery* Antwortzeiten gemittelt wurden. Dies ist eine erstaunlich kurze Reaktionszeit, wenn man sie mit der Reaktionszeit des Pings mit 5.8 ms vergleicht.

Testszenario zwischen drei Fahrzeugen

In Abbildung 40 ist eine Konstellation von drei Fahrzeugen (A, B und C) dargestellt. Der Senderadius der einzelnen Fahrzeuge erstreckt sich immer bis zum nächsten Nachbarfahrzeug. Daher kann Fahrzeug A nicht direkt mit Fahrzeug C kommunizieren, sondern benötigt Fahrzeug B zur Weiterleitung seiner Nachrichten an Fahrzeug C. Die Pfeile veranschaulichen den Kommunikationsweg.

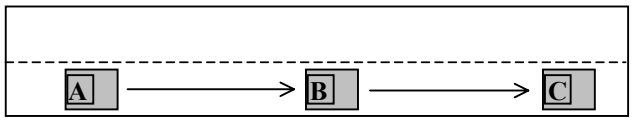


Abbildung 40: Kommunikation zwischen drei Fahrzeugen

7.3. Weiterleitung von Nachrichten

In diesem Test soll die Verzögerung von weitergeleiteten Nachrichten untersucht werden. Dabei sendet das Fahrzeug A eine Nachricht an das Fahrzeug B, welches die Nachricht an das Fahrzeug C weiterleitet. Gemessen wird der Zeitraum zwischen dem Empfangen der Nachricht von Fahrzeug A und dem Weiterleiten der Nachricht an Fahrzeug C durch Fahrzeug B.

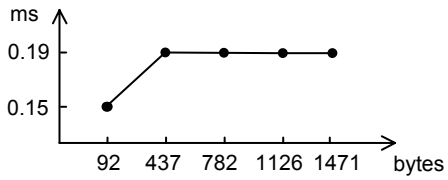


Abbildung 41: Verzögerungen bei der Weiterleitung von Nachrichten

Das Diagramm in Abbildung 41 zeigt die gemessenen Resultate bei der Weiterleitung von Nachrichten mit unterschiedlichen Größen. Die Nachrichtengröße entspricht dem *Payload* im PHY Rahmenformat, das in Abbildung 25 auf Seite 48 dargestellt wurde. Jeder Messpunkt besteht aus zehn gemessenen Werten. Die Verzögerung von 0.15 ms für das Weiterleiten von 92 Byte großen Nachrichten ist sehr kurz. Größere Nachrichten von bis zu 1471 Bytes benötigen für die Weiterleitung im Durchschnitt 0.19 ms, was ebenfalls sehr niedrig ist.

Fazit

Die gemessenen Verzögerungen beim Beantworten bzw. Weiterleiten von Nachrichten liegen in Millisekunden- und sogar in Nanosekundenbereich. Somit können Nachrichten innerhalb kürzester Zeit über mehrere Kilometer weitergeleitet werden. Es kann keine Aussage auf die Verzögerungen und Antwortzeiten von Nachrichten gemacht werden, wenn sich mehrere Fahrzeuge im *Vehicular* MANET befinden, wie in den Szenarien Autobahnverkehr, Verkehr auf der Landstraße und Stadtverkehr in Abbildung 30 dargestellt wurde.

8. Zusammenfassung und Ausblick

Das Ziel dieser Diplomarbeit war die Implementierung eines *Vehicular Routing Mechanism* (VRM) für ein Linux Betriebssystem. Dabei musste der VRM den Anforderungen von „CarTALK 2000“ Anwendungen und einzelnen Einsatzumgebungen (Szenarien) genügen. Darüber hinaus musste die Implementierung des VRM in einer kurzen Zeit möglich, und trotzdem einfach zu warten sein. Außerdem sollte der V*ehicular Routing Mechanism* ohne großen Aufwand auf andere Linux-Systeme installiert und zum Laufen gebracht werden können.

In dieser Diplomarbeit wurde deutlich, dass die vorhandenen Konzepte für das *Beaconing* und *Location Service* Protokoll nur bedingt in einem *Vehicular* MANET eingesetzt werden können. Deshalb wurden für das *Beaconing* und *Location Service* neue Protokolle entworfen, die die Eigenschaften des *Vehicular* MANET berücksichtigen, und daher auch unter schwierigen Bedingungen erfolgreich funktionieren können.

Vehicular Routing Mechanism

Die vorgestellten Konzepte für die Implementierung von *Ad-Hoc Routing* Protokollen konnten kaum übernommen werden, da sie häufig den *Kernel Code* modifizieren, um Zugriff auf die Datagramme von der TCP/UDP Schicht zu bekommen. Einen ganz anderen Weg schlägt die Implementierung des *Vehicular Routing Mechanism* ein. Er verzichtet völlig auf die Modifizierung des Linux *Kernel*, indem es BSD[a] kompatible *Socket* zur Kommunikation zwischen „CarTALK 2000“ Anwendungen und dem *Router* (VRM) verwendet. Die Anwendungen kommunizieren nicht direkt miteinander, wie es sonst bei herkömmlichen Anwendungen durch IP- und Portadresse üblich wäre, sondern sie senden ihre Nachrichten über einen bestimmten Port direkt an den *Vehicular Routing Mechanism* des Fahrzeugs. Dieser empfängt die Nachricht als ein Datagramm und leitet es, durch ein sogenanntes *Raw Socket* Paket an ein anderes Fahrzeug weiter. Erreicht ein *Raw Socket* Paket den Empfänger, dann konvertiert der VRM das Paket in ein Datagramm und sendet es über einen Port, der in dem Paket enthalten war, an die lokale „CarTALK 2000“ Anwendung. Durch diese Vorgehensweise wird die vorhandene *Routing* Tabelle im Linux *Kernel* elegant umgangen, ohne den Linux *Kernel* selbst modifizieren zu müssen.

a. Berkeley Software Distribution

Implementierung

Der prototypische *Vehicular Routing Mechanism* überzeugt durch sehr kurze Antwort- und Weiterleitungszeiten, dadurch ist eine effiziente Kommunikation im *Vehicular* MANET möglich. Zwischen dem *Location Discovery* und dem Empfangen von Nachrichten vergehen nur Millisekunden, wenn sich die Zielfahrzeuge in einigen Kilometer Entfernung aufhalten. Da sich innerhalb von ein paar Millisekunden auch schnellfahrende Fahrzeuge kaum bewegen, ist es sehr wahrscheinlich, dass Nachrichten nach dem *Location Discovery* bis zum Empfänger weitergeleitet werden können.

Obwohl die digitale Straßenkarte für diese Diplomarbeit leider nicht zur Verfügung stand und deshalb alternative Protokolle für das Weiterleiten von Nachrichten und das *Beaconing* verwendet werden mussten, wird das Funknetzwerk trotz des statischen *Beaconing* Protokolls kaum überlastet. Der Grund liegt in der sehr geringen Anzahl von prototypischen Fahrzeugen, die mit einer IEEE 802.11b Funkkarte ausgerüstet sind. Durch die Verwendung einer digitalen Straßenkarte kann eine effizientere Weiterleitung und Bandbreitenauslastung mit Hilfe des *Map-based Geographic Forwarding* und verkehrsbezogenes *Beaconing* erreicht werden.

Ausblick

Repräsentative Aussagen über das Verhalten des *Vehicular Routing Mechanism* in der Praxis sind erst ab einer gewissen Anzahl von „CarTALK 2000“ Fahrzeugen möglich. Stehen genügend Fahrzeuge zur Verfügung, können Szenarien wie Autobahn-, Landstraßen- und Stadtverkehr für die Testumgebung nachgebildet und das Verhalten des *Vehicular Routing Mechanism* in der Praxis getestet werden.

Der *Vehicular Routing Mechanism* bietet eine Plattform für Anwendungen. Sie sind dadurch in der Lage, Nachrichten an andere Fahrzeuge zu schicken. Aber erst durch „intelligente“ Anwendungen ist das Fahrzeug im Stande dem Fahrer die Sicherheit und den Fahrkomfort bieten zu können, wie er es von einem „intelligenten“ Auto erwartet. Solche Anwendungen müssen von den „CarTALK 2000“ Partnern bereitgestellt werden, damit das „CarTALK 2000“ System vernünftig eingesetzt kann. Entscheidend für einen erfolgreichen Einsatz des Systems in der Praxis ist das Erreichen einer „kritischen Masse“ von „CarTALK 2000“ Fahrzeugen, damit einen *Vehicular* MANET aufgebaut werden kann, um Nachrichten erfolgreich weiterleiten zu können.

Abkürzungsverzeichnis

Abkürzung	Bedeutung
BSD	Berkeley Software Distribution
AMI-C	Automotive Multimedi Interface Collaboration
API	Application Program Interface
ASL	Ad-Hoc Support Library
BSS	Basic Station Set
CBLC	Communication-based Longitudinal Control
CODA	Co-operative Driver Assistenace
CSMA/CA	Carrier Sense Multiple Access / Carrier Avoidance
EC	European Commission
GPS	Global Positioning System
ISO	International Organization for Standardization
IWF	Information and Warning Functions
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MGF	Map-based Geographic Forwarding
OEM	Original Equipment Manufacturer
OSI	Open Systems Interconnection
SA	Selective Availability
STL	Standard Template Library
WLAN	Wireless Local Area Network

Literaturverzeichnis

Abkürzung	Literaturquelle
[BS+00]	L. Briesemeister, L. Schäfers und G. Hommel: Disseminating Messages among Highly Mobile Hosts based on Inter-Vehicle Communication. Berlin-Germany; DaimlerChrysler AG Research and Technology; IEEE Intelligent Vehicles Symposium. Seite 522-527; Jahr 2000
[CAR00]	CarTALK 2000 Projekt: siehe Website http://www.cartalk2000.net
[CB+02]	T. Camp, J. Boleng und L. Wilcox: Location Information Services in Mobile Ad Hoc Networks. 2002 IEEE International Conference on Communication Conference Proceedings; Vol. 5, Seite 3318-3324, Jahr 2002
[DD00]	S. Desilva, S. Das: Experimental Evaluation of a Wireless Ad Hoc Network. Proceedings of the 9th International Conference on Computer Communications and Networks (IC3N); Las Vegas, October 2000
[EN01]	E. Nett, M. Mock, M. Gergeleit: Das drahtlose Ethernet: Der IEEE 802.11 Standard. Addison-Wesley, ISBN: 3-8272-1741-X; Jahr 2001
[GPS1]	GARMIN-What is GPS?: siehe Website http://www.garmin.com/aboutGPS
[GPS2]	GARMIN-Differenzielles GPS: siehe Website http://www.garmin.de/index-Begriffserklaerungen.html
[HL99]	Z. Haas, B. Lian: Ad hoc mobility management with uniform quorum systems. IEEE/ACM Transactions on Networking; Vol. 7, Nr. 2, Seite 228-240, April 1999
[HN94]	G. Hoffman, S. Nielsen: Beschreibung von Verkehrsabläufen an signalisierten Knotenpunkten. Bonn: Bundesministerium für Verkehr, Abt. Straßenbau, 1994; Forschung Straßenbau und Straßenverkehrstechnik - Heft 693, Jahr 1994

Abkürzung	Literaturquelle
[HP91]	H. Norman, P. Larry: The x-Kernel: An Architecture for Implementing Network Protocols. IEEE Transactions on Software Engineering; Vol. 17, Nr. 1, Seite 64-76, Jahr 1991
[Ki01]	J. Kienzle: Analyse von Einzelfahrzeugdaten - Verkehr verstehen. Institut für Strassen- und Verkehrswesen; Universität Stuttgart, Jahr 2001
[KZ+02]	V. Kawadia, Y. Zhang und B. Gupta: System Services for Implementing Ad-Hoc Routing Protocols; International Conference on Parallel Processing Workshops; Vancouver, Canada, August 2002
[Le02]	M. Legner: Map-based geographic forwarding in vehicular networks. Diplomarbeit-Nr: 1994, Universität Stuttgart Informatik, Prof. Rothermel, Jing Tian, 31. Juil 2002
[LJ+00]	J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger und R. Morris: A scalable location service for geographic ad hoc routing. In Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM) 2000; Seite 120-130, Boston, MA, USA, 2000
[MANET]	Aus der Charta der IETF MANET-Arbeitsgruppe: siehe Webseite: http://www.ietf.org/html.charters/manet-charter.html
[MJK+00]	R. Morris, J. Jannotti, F. Kaashoek, J. Li und D. S. J. De Couto Carnet: A scalable ad hoc wireless network system. In Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System; Kolding, Denmark, September 2000
[MK+99]	R. Morris, E. Kohler, J. Jannotti und F. Kaashoek: The Click modular router. Symposium on Operating Systems Principles; Seite 217-231, Jahr 1999
[MW+01]	M. Mauve, J. Widmer, und H. Hartenstein: A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. IEEE Network, Vol. 15, Nr. 6, Seite 30-39, Jahr 2001

Abkürzung	Literaturquelle
[Ne00]	L. Neubert: Statistische Analyse von Verkehrsdaten und die Modellierung von Verkehrsfluss mittels zellularer Automaten. Technologie der Gerhard-Mercator-Universität Duisburg; Oelsnitz (Vogtl.) Deutschland, Jahr 2000
[NT+02]	S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, und J.-P. Sheu: The Broadcast Storm Problem in a Mobile Ad Hoc Network. International Conference on Mobile Computing and Networking; ACM Press, Seite 151-162, New York, USA, Jahr 1999
[Pr00]	R. Pressman: Software Engineering - A Practitioner's Approach. McGraw-Hill Publishing Company, ISBN: 0-07-709677-0, Jahr 2000
[RP00]	E. Royer, C. Perkins: An Implementation Study of the AODV Routing Protocol. Proceedings of the IEEE Wireless Communications and Networking Conference; Chicago, IL, September 2000
[Sc00]	J. Schiller: Mobile Communications. Addison Wesley Professional; ISBN-0-201-39836-2; Jahr 2000
[St94]	W. R. Stevens: TCP/IP Illustrated, Volume I: The Protocols. Addison Wesley, ISBN: 0-201-63346-9, Jahr 1994
[W02]	G. Wépiwé: Design and Implementation of a Routing Mechanism for a Wireless Ad Hoc Network. Departement of Electrical and Computer Engineering; Institute of Open Communciation Systems; Technical University of Berlin, Februray 2002
[WF01]	W. Franz, H. Hartenstein und B. Bochow: Internet on the Road via Inter-Vehicle Communications. Workshop der Informatik 2001: Mobile Communication over Wireless LAN; Wien, 26.-29. September 2001
[WS95]	G., Wright, W. R. Stevens: TCP/IP Illustrated, Volume II: The Implementation. Addison Wesely, ISBN: 0-201-63354-X, Jahr 1995

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

Stuttgart, den 30.4.03

(Richard Klinger)