Karlsruhe Institute of Technology

Adaptive Routenplanung mit Hilfe eines Geocast Protokolls

Diplomarbeit
von

Matthias Preis

am Institut für Telematik
der Fakultät für Informatik

Erstgutachter:	Prof. Dr. M. Zitterbart
Zweitgutachter:	Prof. Dr. D. Wagner
Betreuende Mitarbeiter:	Dipl.-Inform. M. Florian M. Baum, M. Sc.

Bearbeitungszeit: 01. Januar 2014 – 30. Juni 2014

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 30. Juni 2014

Inhaltsverzeichnis

1. Einleitung

Aufgrund der stetig wachsenden Anzahl an Fahrzeugen kommt es auf Deutschlands Straßen immer häufiger zu Staus. Laut einer Staubilanz des ADAC von 2013 wurden auf den deutschen Autobahnen rund 415.000 Stauereignisse registriert [1] und es wird davon ausgegangen, dass es noch viele weitere Ereignisse gibt, die aufgrund mangelnder Informationsquellen nicht registriert werden konnten. Die Reduktion dieser Stauereignisse ist eine der großen Herausforderungen im heutigen Verkehrswesen. Eine Möglichkeit, dieses Problem zu lösen, ist die bessere Verteilung aller Fahrzeuge auf alle verfügbaren Routen. Anbieter von Navigationsgeräten wie z. B. TomTom greifen bereits auf die aktuell gemeldeten Verkehrssituationen zurück, um einen Stau zu umfahren. Diese Informationen liegen aber meist erst mit einer gewissen Verzögerung vor. Zudem werden alle Verkehrsteilnehmer, welche diese Geräte verwenden, oftmals auf dieselbe Strecke umgeleitet, was dazu führen kann, dass sich auf dieser Strecke ebenfalls ein Stau bildet. Um alle Fahrzeuge optimal auf alle verfügbaren Strecken zu verteilen, wird daher ein System benötigt, welches aktiv in der Lage ist, Verkehrsinformationen zu sammeln und anhand dieser Informationen die beste Strecke zu berechnen. Damit aber Verkehrsinformationen gesammelt werden können, muss das System mit anderen Verkehrsteilnehmern oder einer zentralen Instanz, welche über alle relevanten Informationen verfügt, kommunizieren und Informationen über die Verkehrslage an den jeweiligen Positionen der anderen Fahrzeuge erfragen. Bereits existierende Systeme verwenden in der Regel das sogenannte *Client-Server* Prinzip, bei dem es eine Instanz, den Server, gibt, welcher über alle Informationen verfügt und die Fahrzeuge, die Clients, diese Informationen von dem Server abfragen. Bei diesem Prinzip existieren aber Bedenken hinsichtlich der Ausfallsicherheit und des Datenschutzes. Fällt dieser Server aus, funktioniert das ganze System nicht mehr. Außerdem können anhand der Informationen auf diesem Server Bewegungsmuster von sehr vielen Personen erstellt werden. Dies macht den Server zu einem Ziel für Angriffe auf diese Daten. Ein sogenanntes dezentrales Geocast Protokoll kann die Ausfallwahrscheinlichkeit verringern, da es keine zentrale Instanz gibt, auf der alle Informationen gespeichert sind. In einem dezentralen System kennen alle Fahrzeuge die Position von einigen wenigen anderen Fahrzeugen. Dies erschwert das Erstellen von Bewegungsmustern, da ein Angreifer nicht wissen kann, welches Fahrzeug

[1] http://www.adac.de/_mmm/pdf/statistik_staubilanz_0514_68897.pdf

gerade die Informationen besitzt, die er haben möchte. Darüber hinaus ermöglicht ein Geocast Protokoll dem System, Nachrichten gezielt an geographische Positionen zu senden, ohne wissen zu müssen, welches andere Fahrzeug sich an dieser Position befindet. Durch das Protokoll wird die Nachricht so weitergeleitet, dass sie von dem Fahrzeug empfangen wird, welches den geringsten Abstand zu den Zielkoordinaten hat.

1.1 Zielsetzung der Arbeit

Das Ziel dieser Arbeit ist die Implementierung und Evaluierung eines adaptiven Routenplaners für die Simulationsumgebung OMNeT++ [2] [15] unter Zuhilfenahme des von dem Institut für Telematik entwickelten Rahmenwerkes OverSim [3]. Dieser Routenplaner soll in der Lage sein, in einem gegebenen Straßennetz mit verschiedenen Verkehrssituationen, den Pfad mit der kleinsten Reisezeit zu finden. Um dies zu leisten, erhält der Routenplaner die Möglichkeit, Verkehrsinformationen, wie z. B. die aktuell gefahrene Geschwindigkeit, von anderen Fahrzeugen zu erfragen. Zu diesem Zweck muss der Routenplaner in der Lage sein, mit anderen Verkehrsteilnehmern zu kommunizieren. In dieser Arbeit soll unter anderem ermittelt werden, ob sich Geocast Protokolle wie z. B. OverDrive [6] für diesen Einsatz eignen.

1.2 Gliederung der Arbeit

Diese Arbeit wird in die folgenden Kapitel aufgeteilt: Das Kapitel *Grundlagen* beschäftigt sich mit der Einführung wichtiger Definitionen und Begrifflichkeiten. Des Weiteren gibt es einen Einblick in die Funktionsweise von OMNeT++ und OverSim. Im Kapitel *Analyse* wird detailliert auf die Fragestellung und Zielsetzung dieser Arbeit eingegangen. Zudem werden Anforderungen an den Routenplaner gestellt. Anschließend werden verschiedene Lösungsansätze für diese Anforderungen und deren Vor- bzw. Nachteile erörtert. Der *Entwurf* erläutert die Umsetzung der verschiedenen Lösungsansätze. In diesem Kapitel wird im Detail die Arbeitsweise des Routenplaners präsentiert. Das Kapitel *Implementierung* befasst sich mit der eigentliche Implementierung. Es werden wichtige Funktionen und Datenstrukturen erklärt. Anschließend werden die verschiedenen Simulationsparameter genannt. In der *Evaluierung* wird der Simulationsaufbau detailliert beschrieben. Es wird erläutert, welche Parametereinstellungen zum Einsatz kommen und welche Auswirkungen diese Parameter auf die Effizienz des Routenplaners haben. Anschließend werden die Ergebnisse der Simulationen präsentiert und erläutert. Das letzte Kapitel *Zusammenfassung*, gibt einen Überblick über die gesamte Arbeit. Es wird kurz zusammengefasst, was in dieser Arbeit beschrieben wurde, welche Ergebnisse erzielt wurden und welches Fazit daraus gezogen werden kann. Zum Schluss wird noch ein Ausblick auf weitere Arbeiten zu diesem Thema gegeben.

[2] www.omnetpp.org

2. Grundlagen

In diesem Kapitel wird zu Beginn die Definition eines Graphen und eines Pfads gegeben. Des Weiteren wird erläutert, was unter einem zeitabhängigen Dijkstra-Algorithmus verstanden wird und was ein Overlay-Netzwerk ist. Als nächstes wird beschrieben, was ein Geocast Protokoll ist und anschließend werden Begriffe eingeführt, welche für diese Arbeit von Bedeutung sind. Abschließend wird die Simulationsumgebung OMNeT++ sowie das Rahmenwerk OverSim erläutert.

2.1 Erläuterung: Graph

In diesem Abschnitt wird nun die Definition eines *Graphen* und eines *Pfad* gegeben. Der Graph wird verwendet, um das Straßennetz zu modellieren.

Definition 1 *Gewichteter und gerichteter Graph*
Ein gewichteter und gerichteter Graph $G = (V, E, c)$ besteht aus einer Menge an Knoten V (V steht für vertices), einer Menge an Kanten E (E steht für edges, $E \subseteq V \times V$), sowie einer Kostenfunktion $c : E \mapsto \Re^+$. Eine Kante $e_{v_1,v_2} \in E$ verbindet genau den Knoten v_1 mit dem Knoten v_2. Die Kante e_{v_1,v_2} ist eindeutig, es darf also keine andere Kante von v_1 nach v_2 geben. Jeder Kante ist eine positive reelle Zahl $c(e_i)$ zugeordnet. Des Weiteren wird in dieser Arbeit verlangt, dass entweder die Anzahl an eingehenden oder die Anzahl an ausgehen Kanten für jeden Knoten positiv sein muss. Es darf also keine isolierten Knoten geben, welche weder über eingehende noch ausgehende Kanten verfügen.

Definition 2 *Pfad*
Gegeben sei ein gewichteter und gerichteter Graph $G = (V, E, c)$. Ein Pfad $P_{s,z}$ von einem Startknoten s zu einem Zielknoten z wird durch eine Folge von Knoten $s = v_0,\ldots,v_k = z \in V$ und eine Menge an Kanten $e_0,\ldots,e_{k-1} \in E$ bestimmt, wobei e_i die Kante zwischen den Knoten v_i und v_{i+1} beschreibt. Es gilt: Für jeden Knoten v_i aus $P_{s,z}$ muss es einen Knoten v_{i+1} und einen Knoten v_{i-1} sowie eine Kante $e_i(v_i, v_{i+1})$ und eine Kante $e_{i-1}(v_{i-1}, v_i)$ geben. Die Ausnahmen bilden nur der Startknoten s und der Zielknoten z.

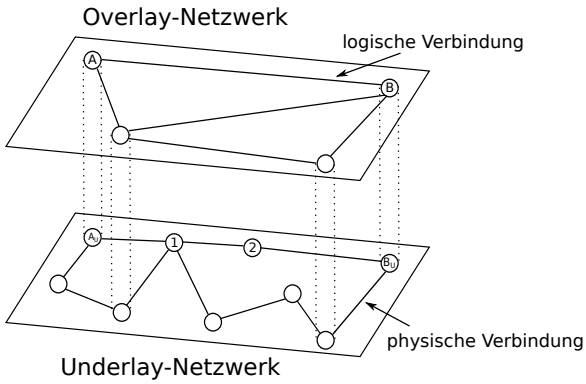

Abbildung 2.1: Beispiel eines Overlay-Netzwerkes

Definition 3 *Distanz eines Pfads und schnellster Pfad*
Anhand der Kostenfunktion jeder Kante in einem Pfad, ist es möglich, die Distanz eines Pfades zu berechnen. Gegeben sei ein gewichteter und gerichteter Graph $G = (V, E, c)$ *sowie ein Pfad P. Dieser Pfad enthält die Kanten* $e_0 \ \ldots \ e_i$. *In der Regel wird die Distanz d(P) wie folgt definiert:* $d(P) = \sum_i \ c(e_i)$. *In dieser Arbeit soll die Distanz eines Pfades aber über die Zeit t bestimmt werden, die benötigt wird, um diesen Pfad zu fahren. Dazu verfügt jeder Knoten über geographische Koordinaten. Mit Hilfe dieser Koordinaten kann der Abstand l zwischen zwei Knoten errechnet werden. Die Kosten einer Kante repräsentieren die Geschwindigkeit v, welche auf dieser Kante gefahren werden kann. Aus dieser Geschwindigkeit und dem Abstand zwischen den Knoten, welche durch diese Kante verbunden sind, wird nun die Zeit* $t = l/v$ *berechnet, welche benötigt wird, um von dem einen Knoten zu dem anderen zu fahren. Die Distanz eines Pfades wird nun bestimmt durch:* $d(P) = \sum_i \ t_i$, *wobei* t_i *die benötigte Zeit für Kante* e_i *darstellt. Mit Hilfe der Distanz lassen sich Pfade nun vergleichen. Der schnellste Pfad zeichnet sich dadurch aus, dass er die kleinste Distanz von allen Pfaden zwischen einem Start- und einem Zielknoten hat.*

2.2 Erläuterung: Zeitabhängiger Dijkstra-Algorithmus

Um in einem gewichteten Graphen den kürzesten Pfad zu berechnen, wird oftmals Dijkstras-Algorithmus [5] verwendet. Dieser Algorithmus berechnet für einen gewichteten Graph mit konstanten Kantengewichten garantiert den kürzesten Pfad von einem Startknoten zu einem Zielknoten, sofern so ein Pfad existiert. In Abbildung 2.2 wird der Algorithmus in Pseudo Code beschrieben. Das Straßennetz, welches in dieser Arbeit zum Einsatz kommt, wird durch einen gewichteten und gerichteten Graphen repräsentiert. Die Kanten des Graphen modellieren die verschiedenen Straßen und die Kantengewichte stehen für die Maximalgeschwindigkeit, welche auf dieser Straße gefahren werden kann. Auf dem Straßennetz sollen verschiedene Verkehrssituationen, wie z. B. ein Stau, simuliert werden. Dazu werden die Kantengewichte entsprechend der Verkehrssituation angepasst. Da die Verkehrssituationen aber ein dynamisches Verhalten aufweisen sollen, ändern sich die Kantengewichte mit der Zeit. Der Algorithmus von Dijkstra ist in so einem Fall aber nicht mehr in der Lage, den kürzesten Pfad zu berechnen. Somit wurde für diese Arbeit eine Abwandlung von Dijkstras-Algorithmus implementiert, welche in der Lage ist, den zeitabhängigen kürzesten Pfad in einem gewichteten Graphen zu berechnen. Dieser Algorithmus

```
Funktion Dijkstra(Graph, Source):
{
    initializeGraph(Graph,Source,distance[],predecessor[],Q)
     solange Q nicht leer:                  // Der eigentliche Algorithmus
        u := Knoten in Q mit kleinstem Wert in distance[]
        entferne u aus Q
        für jeden Nachbarn v von u:
           falls v in Q:
             distanz_update(u,v,distance[],predecessor[])
     return predecessor[]
}
Methode initializeGraph(Graph,Source,distance[],predecessor[],Q):
{
     für jeden Knoten v in Graph:
        distance[v] := unendlich
        predecessor[v] := null
     distance[Source] := 0
     Q := Die Menge aller Knoten in Graph
}
Methode distanz_update(u,v,distance[],predecessor[]):
{
     alternativ := distance[u] + abstand_zwischen(u, v)   // Weglänge vom Startknoten nach v über u
     falls alternativ < distance[v]:
        distance[v] := alternativ
        predecessor[v] := u
}
Funktion erstelleKürzestenPfad(Zielknoten,predecessor[])
{
   Weg[] := [Zielknoten]
   u := Zielknoten
   solange predecessor[u] nicht null:   // Der Vorgänger des Startknotens ist null
      u := predecessor[u]
      füge u am Anfang von Weg[] ein
return Weg[]
}
```

Abbildung 2.2: Pseudo Code des Dijkstra Algorithmus

wird verwendet, um den Pfad, in einem Straßennetz mit sich zeitlich ändernden Verkehrssituationen, zu berechnen, für den ein Fahrzeug die geringste Zeit benötigt, um von einem Startknoten zu einem Zielknoten zu fahren. Eine nähere Beschreibung der Implementierung dieses Algorithmus wird in Kapitel 5.1 gegeben.

2.3 Erläuterung: Overlay-Netzwerk

Während ein Graph eine rein theoretische Beschreibung ist, mit der reale Strukturen wie ein Straßennetz modelliert werden können, handelt es sich bei einem *Overlay-Netzwerk* um ein real existierendes Netzwerk. In einem Overlay-Netzwerk sind die Verbindungen zwischen zwei Teilnehmern logische Verbindungen. Das bedeutet, dass es keine direkte physische Verbindung, wie z. B. ein Glasfaserkabel, zwischen den beiden Teilnehmern gibt. Jedes Overlay-Netzwerk muss auf einem sogenannten *Underlay-Netzwerk* aufgebaut sein, welches über physische Verbindungen zwischen Teilnehmern verfügt. Eine logische Verbindung im Overlay-Netzwerk besteht in der Regel aus mehreren physischen Verbindungen im Underlay-Netzwerk. In Abbildung 2.1 ist ein solches Netzwerk zu sehen. Der obere Bereich repräsentiert das Overlay-Netzwerk und der untere Bereich Underlay-Netzwerk. Die gestrichelten Linien zwischen dem Underlay- und dem Overlay-Netzwerk sollen verdeutlichen, dass jeder Teilnehmer im Overlay-Netzwerk auch ein Teilnehmer des Underlay-Netzwerkes sein muss. Teilnehmer A versendet nun eine Nachricht an Teilnehmer B. Dazu übergibt er die Nachricht an Teilnehmer A_U. Die Nachricht wird dann von Teilnehmer A_U über Teilnehmer 1 und Teilnehmer 2 zu Teilnehmer B_U weitergeleitet. Dieser übergibt die Nachricht an Teilnehmer B im Overlay-Netzwerk. Aus der Sicht von Teilnehmer A und Teilnehmer B wurde die Nachricht direkt von A nach B geleitet, ohne dass ein anderer Teilnehmer die Nachricht weitergeleitet hätte.

2.4 Erläuterung: Geocast Protokoll

Bei einem Geocast Protokoll handelt es sich um ein Protokoll, in dem Nachrichten an geographische Koordinaten und weniger gezielt an andere Teilnehmer gesendet werden können. Eine der ersten Arbeiten zu Geocast Protokollen wurde von Navas und Imielinski [12] veröffentlicht. Das hier beschriebene Geocast Protokoll soll neuen Anwendungen ermöglichen, Dienste bereitzustellen, welche sich nur auf ein bestimmtes geographisches Gebiet beziehen. So soll z. B. jeder Autofahrer, welcher in ein bestimmtes Gebiet hineinfährt, die Meldung erhalten, dass aufgrund von Unwetterschäden eine bestimmte Straße gesperrt ist. Diese Meldung soll aber nur in diesem Gebiet empfangen werden können. Um dies zu ermöglichen, wird für jede Nachricht ein Zielgebiet definiert. Bei diesem Gebiet kann es sich um einen Punkt, einen Kreis mit festgelegtem Radius oder einem Polygon handeln. Jeder Teilnehmer in diesem Gebiet erhält die Nachricht, ohne dass der Sender der Nachricht weiß, welche anderen Teilnehmer sich in diesem Gebiet aufhalten.

2.5 Begriffsdefinitionen

In dieser Arbeit kommen hauptsächlich zwei Bereiche der Informatik zum Einsatz: Die Telematik und die Algorithmentechnik. Da es Begriffe gibt, die in beiden Bereichen eingesetzt werden, aber unterschiedliche Bedeutungen haben, werden in diesem Abschnitt nun Begriffe eingeführt, welche für diese Arbeit von Bedeutung sind: Unter einem *Knoten* wird der Knoten eines Graphen verstanden. Ein *Fahrzeug* beschreibt ein mobiles Objekt, das seine geographische Position verändern kann. Darüber hinaus verfügt ein Fahrzeug über die Möglichkeit, Nachrichten zu versenden und Nachrichten zu empfangen. Eine *Anfrage* ist eine Nachricht, welche von einem Fahrzeug versendet wird, um Informationen zur aktuellen Verkehrslage auf einer Straße zu sammeln. Eine *Kreuzung* bezeichnet einen Knoten im Straßengraph, der über mindestens zwei ausgehende Kanten verfügt. Die *Reisezeit* gibt die Zeitdauer an, welche ein Fahrzeug benötigt, um von einem Startknoten zu einem Zielknoten zu fahren. Unter der *Simulationszeit* wird die Zeitdauer verstanden, die benötigt wird, um eine Simulation durchzuführen.

2.6 OMNeT++ und OverSim

Die Implementierung des Routenplaners erfolgt mit der Simulationsumgebung OMNeT++ [1] [15]. OMNeT++ ist eine Event gesteuerte Simulationsumgebung, welche kostenlos[2] zu erhalten ist. OMNeT++ ist modular aufgebaut und bietet mit der Sprache *NED* die Möglichkeit, eigene Module zu erstellen. Es gibt zwei grundlegende Modularten: Die sogenannten *compound modules*, welche aus anderen Modulen zusammengesetzt sind und die sogenannten *simple modules*, welche direkt in C++ implementiert werden. OMNeT++ ermöglicht es, verschiedene Module über sogenannte *gates* zu verbinden. Module, die über ein solches gate verbunden sind, können Nachrichten austauschen. Des Weiteren bietet OMNeT++ die Möglichkeit, einem gate einen sogenannten *channel* hinzuzufügen. Mit einem channel kann die Übertragung einer Nachricht realistisch simuliert werden, da z. B. die Verzögerungszeit eines

[1] www.omnetpp.org

[2] kostenlos nur für nicht gewerbliche Zwecke

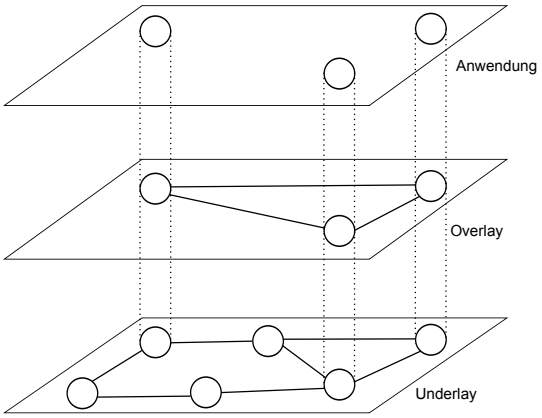

Abbildung 2.3: Hierarchischer Aufbau eines Protokolls, das mit OverSim simuliert werden kann. Die gestrichelten Linien stellen Teilnehmer dar, welche Teil mehrerer Schichten sind.

Kupferkabels simuliert werden kann. Oder ein channel verwirft mit einer gewissen Wahrscheinlichkeit eine Nachricht, die über ihn übertragen werden sollte und simuliert damit den Nachrichtenverlust, der in einem realen Netzwerk auftreten kann. Des Weiteren verfügt OMNeT++ über eine *GUI*, mit der eine Visualisierung des Netzwerkes und der Nachrichtenübertragung möglich ist. Diese GUI kann auf Wunsch abgeschaltet werden, was die Simulationszeit deutlich verringert.

Das Rahmenwerk OverSim [3] wurde vom Institut für Telematik entwickelt, um Overlay-Netzwerke zu simulieren. OverSim ermöglicht es hierarchisch aufgebaute Overlay Protokolle zu simulieren. Diese bestehen aus einem *Underlay-Netzwerk*, einem *Overlay-Netzwerk* und der *Anwendungsschicht*. In Abbildung 2.3 ist dieser Aufbau dargestellt. Die Protokolle für das Underlay- bzw. das Overlay-Netzwerk können beliebig ausgetauscht werden. OverSim ermöglicht es, Simulationen mit einer großen Anzahl an Teilnehmern durchzuführen und dabei nur eine akzeptable Simulationszeit zu benötigen. Des Weiteren unterstützt OverSim sowohl strukturierte als auch unstrukturierte Overlay-Netzwerke. OverSim ist ebenso in der Lage, bösartige Teilnehmer zu simulieren. Somit können auch Sicherheitsaspekte von Protokollen evaluiert werden. Weiter Anforderungen an OverSim sind *Flexibilität*, *interactive Visualisierung* und *erstellen von Statistiken*. Unter Flexibilität wird verstanden, dass OverSim mobile Teilnehmer simuliert und es auch zu Ausfällen von Teilnehmern kommen kann. Die interaktive Visualisierung soll es Entwicklern von neuen Protokollen ermöglichen ihre Protokolle anhand einer graphischen Darstellung auf Programmfehler hin zu untersuchen. Mit Hilfe der erstellten Statistiken sollen Entwickler in der Lage sein, ein neues Protokoll anhand von Metriken wie verbrauchte Bandbreite, fehlerhafte oder nicht übertragene Nachrichten und Stretch zu evaluieren. Für OverSim wurde bereits Underlay Modelle wie z. B. *INET* und Overlay Protokolle wie z. B. *Kademlia* [11] implementiert.

3. Analyse

In diesem Kapitel wird nun die Aufgabenstellung detailliert beschrieben und es werden Anforderungen an den Routenplaner gestellt. Anschließend werden existierende Lösungsansätze präsentiert und auf ihre Tauglichkeit evaluiert. Damit der Routenplaner in einer möglichst realitätsnahen Umgebung simuliert werden kann, werden neben dem Routenplaner noch zwei weitere Modelle implementiert: Das *Verkehrsmodell* und das *Kommunikationsmodell*. Das Verkehrsmodell soll das Straßennetz, die Fahrzeuge und die Verkehrssituationen bereitstellen. Das Kommunikationsmodell soll ein Geocast Protokoll bereitstellen. Die genauen Anforderungen an alle drei Modelle werden im Abschnitt 3.2 beschrieben. Im Folgenden wird nun die Aufgabenstellung dieser Arbeit erläutert.

3.1 Aufgabenstellung

Das Ziel dieser Arbeit ist die Implementierung eines Routenplaners, der mit Hilfe eines Geocast Protokolls in der Lage sein soll, in einem gegebenen Straßennetz die Route von einem Startpunkt zu einem Ziel zu finden, welche die kürzeste Reisezeit erfordert. Damit der Routenplaner dies erreichen kann, ist es ihm möglich, über das Geocast-Protokoll mit anderen Fahrzeugen zu kommunizieren. Dadurch kann er aktuelle Informationen über die Verkehrslage auf einer bestimmten Straße erfahren. Anhand dieser Informationen soll er in der Lage sein, Staus zu umfahren, sofern dies möglich ist. Darüber hinaus soll der Kommunikationsaufwand, der betrieben werden muss, um die Informationen über die aktuelle Verkehrslage zu erhalten, so gering wie möglich gehalten werden. Der Routenplaner muss daher einen geeigneten Trade-Off zwischen Kommunikationsaufwand und Reisezeit finden.

3.2 Anforderungen

Im Folgenden werden nun die Anforderungen an das Verkehrsmodell, das Kommunikationsmodell und den Routenplaner erläutert.

3.2.1 Verkehrsmodell

Das Verkehrsmodell muss das Straßennetz, die einzelnen Fahrzeuge und die Verkehrslage verwalten. Das Straßennetz soll realistisch sein. Das heißt, es soll verschiedene Straßentypen wie z. B. Autobahnen und Bundesstraßen mit unterschiedlichen Maximalgeschwindigkeiten enthalten. Des Weiteren müssen alle Fahrzeuge auf dieselben Straßeninformationen zugreifen können, um z. B. die Geschwindigkeit erfahren zu können, welche auf dieser Straße maximal gefahren werden darf bzw. kann. Darüber hinaus soll das Verhalten der Fahrzeuge realistisch sein. Die Geschwindigkeiten von unterschiedlichen Fahrzeugen, welche sich auf derselben Straße befinden, sollen daher variieren. Zuletzt muss das Verkehrsmodell Verkehrssituationen auf dem Straßennetz erzeugen. Diese sollen sich zeitlich ändern. Eine Verkehrssituation soll somit während einer Simulation entstehen bzw. sich auflösen können.

Ein Programm, mit dem Verkehrssimulationen durchgeführt werden können, ist SUMO [8]. Mit SUMO können Verkehrsszenarien sehr detailliert simuliert werden. So kann z. B. das Verhalten der Fahrzeuge direkt festgelegt werden und darüber hinaus kann der Verkehrsfluss durch Ampeln gesteuert werden. Des Weiteren können verschiedene Fahrzeugtypen wie z. B. LKWs oder PKWs verwendet werden. Aufgrund dieser Detailtiefe ist es sehr aufwändig, ein Szenario in SUMO zu erstellen und zu simulieren. Der ausschlaggebende Grund, SUMO nicht zu verwenden, war die nicht vorhandene Kompatibilität mit dem in dieser Arbeit verwendeten Rahmenwerk OverSim. Darüber hinaus verfügte OverSim bereits über ein rudimentäres Verkehrsmodell welches während der Entwicklung von OverDrive [6] für OverSim implementiert wurde. Die Erweiterung dieses Verkehrsmodells wurde daher der Verwendung anderer Programme zur Verkehrssimulation vorgezogen. Das Verkehrsmodell von OverSim ermöglicht das Einlesen von Straßenkarten im XML Format und es erstellt daraus einen Straßengraphen. Des Weiteren ist dieses Rahmenwerk in der Lage, Fahrzeuge zu erzeugen und diese auf einem Straßennetz fahren zu lassen. Das Verkehrsmodell von OverSim erfüllt daher schon einige der Anforderungen. Für diese Arbeit wurde das Modell so erweitert, dass es sämtliche Anforderungen erfüllt. Diese Erweiterungen werden im nächsten Kapitel erläutert.

3.2.2 Kommunikationsmodell

Das Kommunikationsmodell muss den Fahrzeugen ermöglichen, miteinander zu kommunizieren. Dies soll über ein Geocast-Protokoll realisiert werden. Das in dieser Arbeit verwendete Geocast-Protokoll ist OverDrive [6]. Im nächsten Abschnitt wird nun ein Überblick über das Protokoll OverDrive gegeben. Das Protokoll wird nicht im Detail erläutert, sondern es wird grob die Funktionsweise erklärt.

3.2.2.1 OverDrive

OverDrive [6] wurde am Institut für Telematik am Karlsruhe Institut für Technologie entwickelt. Es handelt sich hierbei um ein sogenanntes *dezentrales Geocast Protokoll*, welches explizit für Anwendungen im Bereich der Fahrzeugkommunikation entwickelt wurde. Nachrichten werden bei einem Geocast Protokoll an eine geographische Position versendet und nicht an ein bestimmtes Fahrzeug. Ein Geocast Protokoll ermöglicht es einem Fahrzeug somit, eine Nachricht an eine geographische Position zu senden, ohne zu wissen, welches Fahrzeug sich an dieser Position befindet oder

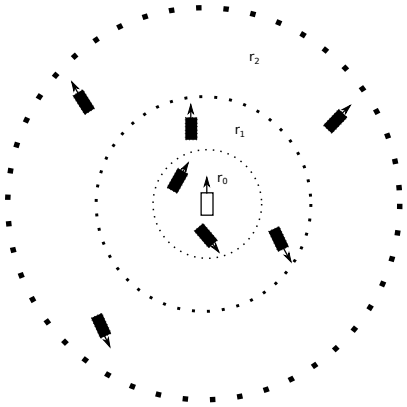

Abbildung 3.1: Ringe für die Gruppierung der Nachbarschafts-Tabelle Einträge

ob sich überhaupt ein anderes Fahrzeug an dieser Position aufhält. OverDrive erzeugt ein sogenanntes *Overlay-Netzwerk* und verwendet das Mobilfunknetz und das Internet als Underlay-Netzwerk. Nachrichten werden also mit Hilfe von Routing-Protokollen wie z. B. *IP* von einem Fahrzeug zum anderen übertragen. OverDrive ist darüber hinaus ein dezentrales Protokoll. Dies bedeutet, dass jedes Fahrzeug des Protokolls selbst dafür verantwortlich ist, genug Informationen zu sammeln, damit Nachrichten an andere Fahrzeuge versendet werden können. Es gibt also keine zentrale Stelle, welche über alle Informationen, wie z. B. die Anzahl an Fahrzeugen oder deren Adressen, verfügt. Im nächsten Abschnitt wird nun beschrieben, wie ein OverDrive-Fahrzeug die benötigten Informationen sammelt.

Jedes Fahrzeug besitzt eine sogenannte *Nachbarschafts-Tabelle*. In dieser Nachbarschafts-Tabelle werden die geographische Position, die Geschwindigkeit und die Bewegungsrichtung der anderen Fahrzeuge eingetragen. Ein Eintrag in die Nachbarschafts-Tabelle repräsentiert eine Kommunikationsbeziehung zwischen dem Besitzer der Nachbarschafts-Tabelle und dem Fahrzeug, das durch den Eintrag repräsentiert wird. Diese Kommunikationsbeziehung ist bidirektional und kann nur stattfinden, wenn beide Fahrzeuge dieser Beziehung zustimmen. Des Weiteren unterteilt jedes Fahrzeug den Bereich um seine Position in sogenannte *Ringe*. Diese Ringe dienen zur Gruppierung der Einträge in der Nachbarschafts-Tabelle nach der geographischen Entfernung. In Abbildung 3.1 ist dargestellt, wie die Ringe benutzt werden, um Fahrzeuge nach ihrer Entfernung zu gruppieren. Fahrzeuge mit einer sehr kurzen geographischen Entfernung werden Ring r_0 zugeordnet. Fahrzeuge mit einer mittleren Entfernung werden dem Ring r_1 zugeordnet und alle Fahrzeuge mit einer großen Entfernung werden Ring r_2 zugeordnet. Für jeden Ring gibt es eine gewünschte Anzahl an Einträgen n_{des} und eine maximale Anzahl an Einträgen n_{max}. Ist die Anzahl der Einträge in einem Ring kleiner als n_{des}, sucht OverDrive aktiv nach anderen Fahrzeugen. Die Größe von n_{des} nimmt mit der Entfernung eines Ringes ab. So ist n_{des} für r_2 deutlich geringer als für r_1. n_{max} gibt an, wie viele Einträge ein Ring maximal haben darf. Ist dieser Wert erreicht, wird weder nach anderen Fahrzeugen gesucht, noch wird eine neue Kommunikationsbeziehung zu einem Fahrzeug angenommen, welches sich in diesem Ring befindet.

Damit ein Fahrzeug F entscheiden kann, ob ein neuer Nachbar in die Nachbarschafts-Tabelle aufgenommen wird, gibt es die sogenannte *Bewertungsfunktion*. Mit Hilfe dieser Bewertungsfunktion wird jedem potentiellen neuen Nachbarn ein Wert zuge-

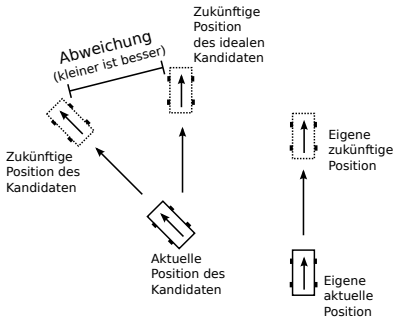

Abbildung 3.2: Bewertung eines Nachbarn anhand der Abweichung zwischen dem Kandidaten und einem idealen Kandidaten

teilt. Dieser Wert ist die Abweichung von einer *vorhergesagten zukünftigen Position* und der vorhergesagten Position eines *idealen Nachbarn*. Die vorhergesagte zukünftige Position nach einer Zeitspanne t_{fut} wird anhand der aktuellen Position, der Bewegungsrichtung und der Geschwindigkeit des potentiellen Nachbarn bestimmt. Es wird angenommen, dass sich Geschwindigkeit oder Bewegungsrichtung des potentiellen Nachbarn nicht ändern. Der konzeptionelle ideale Nachbar bewegt sich mit der gleichen Geschwindigkeit und Bewegungsrichtung wie F. Ideale Nachbarn führen zu einer stabilen Nachbarschaftsstruktur: Da sich die relative Position des Nachbarn zu F nicht ändert, wird der Nachbar immer dem gleichen Ring zugeordnet.

Das Konzept der Bewertungsfunktion wird in Abbildung 3.2 dargestellt.

Es wird nun erläutert, wie Fahrzeuge F mit Hilfe von OverDrive nach anderen Fahrzeugen suchen können: F überprüft periodisch, ob die gewünschte Anzahl an Einträgen n_{des} für alle Ringe erreicht ist. Existiert ein Ring r_i, für den dies nicht der Fall ist, so sendet F eine sogenannten *find_neighbor_request* Nachricht. Diese Nachricht wird an zufällig ausgewählte Koordinaten gesendet, welche innerhalb des Rings r_i liegen. Diese Nachricht wird von dem Fahrzeug X empfangen, welches die kürzeste Entfernung zu diesen Koordinaten hat. X sendet nun eine Antwort zurück an F. In dieser Antwort sind Einträge aus der Nachbarschafts-Tabelle von X enthalten. Sobald F die Antwort erhält, berechnet er den Wert für jeden Eintrag in der Antwort und sortiert die Einträge entsprechend dieser Werte. Anschließend versucht er nacheinander eine Kommunikationsbeziehung zu den besten Fahrzeugen, welche durch diese Einträge repräsentiert werden, (aufzubauen). Dies geschieht solange, bis Ring r_i mindestens n_{des} Einträge enthält.

Der Aufbau einer Kommunikationsbeziehung erfolgt durch sogenannte *neighbor connect request* Nachrichten. Empfängt ein Fahrzeug Y eine solche Nachricht von Fahrzeug F, überprüft Y ob der Ring r_j, welchem F zugeordnet werden würde, schon n_{max} Einträge enthält. Ist dies nicht der Fall, wird Y die Anfrage annehmen und es wird eine Kommunikationsbeziehung zwischen Fahrzeug Y und Fahrzeug F hergestellt. Enthält der Ring r_j bereits n_{max} Einträge, so wird der Wert von F berechnet. Ist dieser Wert kleiner als der größte Wert, der einem anderen Fahrzeug aus diesem Ring zugeordnet wurde, so wird die Kommunikationsbeziehung zu diesem Fahrzeug beendet und die Kommunikationsbeziehung zu F akzeptiert.

Damit Fahrzeuge Nachrichten korrekt weiterleiten können, müssen die Informationen über die Position der Nachbarn aktuell gehalten werden. Anstatt periodisch die aktuelle Position an alle Nachbarn zu senden, verwendet OverDrive eine effizientere

Methode: Ein Fahrzeug F kennt die Position, die Geschwindigkeit und die Bewegungsrichtung seiner Nachbarn. F weiß aber ebenso, dass diese Nachbarn seine Position, Geschwindigkeit und Bewegungsrichtung kennen, und dass sie mit Hilfe dieser Informationen vorhersagen, an welcher Position sich F befinden wird. F berechnet nun, an welcher Position seine Nachbarn ihn erwarten und vergleicht diese Position mit der aktuellen Position. Ist die Abweichung zwischen den beiden Positionen zu groß, wird eine sogenannte *location update* Nachricht gesendet.

OverDrive verwendet das sogenannte *rekursive, greedy routing* Schema, um Nachrichten weiterzuleiten: Ein Fahrzeug sendet eine Nachricht an das Fahrzeug aus seiner Nachbarschafts-Tabelle, welches die kleinste Entfernung zu den Zielkoordinaten der Nachricht hat. Gibt es kein Fahrzeug in der Nachbarschafts-Tabelle, welches näher an dem Ziel ist als das Fahrzeug selbst, dann sieht sich das Fahrzeug als Ziel dieser Nachricht und verarbeitet sie.

OverDrive bietet einer Anwendung zwei Möglichkeiten, Nutzerdaten zu übertragen: Das sogenannte *geographische Unicast* und das sogenannte *geographische Fluten*. Bei dem geographischem Unicast übergibt die Anwendung die Nutzdaten und die Koordinaten, an welche die Nutzdaten gesendet werden sollen. OverDrive erstellt eine geographische Unicast Nachricht, welche die Nutzdaten und die Koordinaten enthält und sendet die Nachricht an jenes Fahrzeug, welches die kleinste Entfernung zu den Koordinaten hat. Dieses Fahrzeug entpackt die Nutzdaten und leitet sie an die Anwendung weiter. Das geographische Fluten ermöglicht einer Anwendung, Nutzdaten an alle Fahrzeuge in einem bestimmen Gebiet zu senden. Dazu übergibt die Anwendung sowohl die Nutzdaten als auch die Koordinaten des Zentrums des Gebietes. Ein Gebiet wird als geometrische Form angesehen. Es kann z. B. ein Kreis oder Rechteck sein. Handelt es sich bei dem Gebiet um einen Kreis, übergibt die Anwendung noch zusätzlich den gewünschten Radius des Kreises an OverDrive. Die Nutzdaten und die Koordinaten inkl. der Form des Gebietes werden in eine geographische Fluten-Nachricht und an das Fahrzeug gesendet, welches die kleinste Entfernung zum Zentrum des Gebietes hat. Dieses Fahrzeug leitet die Nachricht an alle seine Nachbarn weiter, die sich in diesem Gebiet aufhalten, bevor es die Nachricht an die Anwendung übergibt. Die Nachbarn wiederum leiten die Nachricht an alle ihre Nachbarn weiter, welche sich in diesem Gebiet aufhalten. Es soll daher gewährleistet werden, dass alle Fahrzeuge in diesem Gebiet die Nachricht erhalten.

Ein Problem von Overlay-Netzwerken im Allgemeinen, aber von Overlay-Netzwerken mit mobilen Teilnehmern speziell, ist, dass nicht garantiert werden kann, dass Nachrichten korrekt oder zeitnahe übertragen werden. Aus diesem Grund verwendet OverDrive sogenannte *acknowledgements*. Jedes Fahrzeug, das eine Nachricht enthält, sendet an den Sender dieser Nachricht ein acknowledgement, um diesem den Empfang der Nachricht zu bestätigen. Wird das acknowledgement nicht in einer bestimmten Zeit empfangen, geht der Sender davon aus, dass die Nachricht nicht angekommen ist. Er versucht daraufhin, die ursprüngliche Nachricht erneut zu senden.

Durch die *find_neighbor_request*, *neighbor connect request* und *location update* Nachrichten wird bei OverDrive sehr viel sogenannter *Overhead* erzeugt. Als Overhead werden alle Nachrichten bezeichnet, welche gesendet werden müssen, um die Nachbarschaftsstruktur jedes Fahrzeugs aufzubauen und aufrecht zu halten. Es bezieht sich also auf alle Nachrichten, welche keine Nutzerdaten einer Anwendung enthal-

ten. Dieser Overhead ist für eine sehr lange *Simulationszeit* verantwortlich. Je mehr Nachrichten während der Simulation gesendet werden, desto länger ist die Simulationszeit. Aus diesem Grund wurde ein simples Geocast Protokoll implementiert, welches nur sehr wenig Overhead erzeugt und somit eine geringe Simulationszeit benötigt. Dieses Geocast Protokoll wird im nächsten Kapitel vorgestellt.

Weitere Protokolle welche für die Thematik Verkehr und Mobilität entwickelt wurden sind sogenannte *VANET* (vehicular ad hoc networks) Protokolle. Bei diesen Protokollen sind Fahrzeuge unter anderem in der Lage Nachrichten auszutauschen. Arbeiten zu diesem Thema sind z. B. Willke et. al [16], Nzouonta et. al [13] und Lochert et. al [9]. Die Übertragung der Nachrichten bei diesen Protokollen wird aber in der Regel mit *WLAN* (Wireless Local Area Network) erreicht. Dies birgt einen Nachteil, da die Distanz über die Nachrichten übertragen werden können in der Regel auf mehrere Hundert Meter beschränkt ist. Somit eignen sich VANET Protokolle nicht für den Einsatz in dieser Arbeit.

3.2.3 Routenplaner

Der Routenplaner ist für das Finden des *schnellsten Pfades* von einem Start- zu einem Zielknoten verantwortlich. Als schnellsten Pfad wird der Pfad verstanden, auf dem ein Fahrzeug die geringste Zeit benötigt, um vom Startknoten zum Zielknoten zu fahren. Der Routenplaner muss mit Hilfe des Geocast Protokolls aktiv die Verkehrslage auf den Straßen von anderen Fahrzeugen ermitteln und anhand dieser Daten den schnellsten Pfad berechnen. Dabei soll darauf geachtet werden, die Anzahl gesendeter Nachrichten so gering wie möglich zu halten. Der Routenplaner muss demnach einen Weg finden, die Anzahl gesendeter Nachrichten klein zu halten, aber dennoch genug Informationen zu sammeln, um den schnellsten Pfad zu finden.

Das Straßennetz wird als Graph repräsentiert. Der Routenplaner soll eine Kopie dieses Straßengraphen besitzen, welche aber nicht die Verkehrsinformationen enthält. Mit Dijkstras Algorithmus [5] wurde ein Algorithmus implementiert, welcher den kürzesten Pfad zwischen einem Start- und einem Zielknoten in einem Graphen berechnet, sofern ein solcher Pfad existiert. Es muss nun ein Weg gefunden werden, wie die Informationen zur Verkehrslage gewonnen und in den Straßengraphen des Routenplaners integriert werden können, so dass ein erkannter Stau umfahren werden kann. Somit muss ein Algorithmus implementiert werden, der Informationen zur Verkehrslage in einem Bereich zwischen einem Start- und Zielknoten sammelt und diese Informationen in den Straßengraphen des Routenplaners integriert. Die Anwendung von Dijkstras Algorithmus auf dem Straßengraphen des Routenplaners soll den Pfad liefern, welcher das Fahrzeug auf dem schnellsten Weg zum Zielknoten führt.

3.3 Zusammenfassung

In diesem Kapitel wurden mit dem Verkehrsmodell, dem Kommunikationsmodell und dem Routenplaner die drei Bereiche erläutert, in die diese Arbeit unterteilt ist. Darüber hinaus wurden für alle drei Bereiche Anforderungen gestellt, die erfüllt werden müssen. Anschließend wurden verwandte Arbeiten präsentiert, welche alle oder nur Teile der Anforderungen erfüllen und es wurde erläutert, ob diese verwandten Arbeiten geeignet sind, in dieser Arbeit eingesetzt zu werden.

4. Entwurf

In diesem Kapitel werden nun Lösungsansätze zu den Problemstellungen für die drei Bereiche aus dem vorherigen Kapitel aufgezeigt.

4.1 Verkehrsmodell

Das Verkehrsmodell hat drei zentrale Aufgaben. Zum einen muss es ein Straßennetz erzeugen, auf dem die Fahrzeuge fahren können. Des Weiteren müssen die Fahrzeuge erstellt und verwaltet werden und zuletzt müssen Verkehrssituationen simuliert werden. Zur Repräsentation des Straßennetzes wird ein gewichteter gerichteter Graph verwendet. Der Graph, welcher das Straßennetz abbildet, wird als Straßengraph bezeichnet. Damit der Routenplaner auf einem möglichst realistischen Straßennetz simuliert werden kann, wird der Straßengraph mit Hilfe von OpenStreetMap Daten [1] erstellt. OpenStreetMap ist ein kostenloser Internet-Dienst, welcher aktuelles Straßenkartenmaterial in Form von XML-Dateien zur Verfügung stellt. Das Straßennetz von OpenStreetMap besteht aus sehr vielen Knoten und Kanten. Eine Kante kann aber nur als gerade Linie gezeichnet werden. Damit ein Kurvenabschnitt dargestellt werden kann, muss dieser Abschnitt in viele kurze Abschnitte zerlegt werden, welche dann durch gerade Linien dargestellt werden. Dies hat zur Folge, dass ein Autobahnabschnitt zwischen einer Ausfahrt und der Nächsten nicht aus zwei Knoten und eine Kante besteht, sondern aus einer Vielzahl an Knoten und Kanten. Dies führt dazu, dass die geographische Entfernung von zwei Knoten, welche über eine Kante verbunden sind, nur wenige Meter betragen kann. Alle Knoten und Kanten im Straßengraphen enthalten eine Reihe von Attributen. Jedem Knoten wird eine einzigartige Identifikationsnummer (ID) zugeordnet. Des Weiteren wird jedem Knoten eine geographische Position zugeordnet. Ebenso kennt jeder Knoten alle von ihm ausgehenden Kanten und die IDs der Knoten, zu denen die Kanten führen. Ein Knoten, welcher über eine Kante erreicht werden kann, wird als Nachbarn bezeichnet. Jeder Kante wird ein Kostenwert zugeordnet. Dieser Wert repräsentiert die Geschwindigkeit, welche auf dieser Straße maximal gefahren werden kann. Ebenso sind die IDs der beiden Knoten bekannt, die von dieser Kante verbunden werden. Mit Hilfe der

[1] www.openstreetmap.org

geographischen Positionen von Knoten kann die geographische Entfernung zwischen zwei Knoten berechnet werden. Aus dieser Entfernung und der Geschwindigkeit, welche ein Fahrzeug auf der Kante zwischen diesen Knoten fahren darf, ergibt sich die Zeit, welche das Fahrzeug benötigt, um vom einen Knoten zu dem anderen zu fahren. Als schnellster Pfad wird in dieser Arbeit der Pfad verstanden, auf dem ein Fahrzeug die geringste Zeit benötigt, um vom Startknoten zu dem Zielknoten zu fahren. Das verwendete Rahmenwerk OverSim bietet bereit einen Mechanismus, um Straßenkarten im XML-Dateiformat einzulesen und daraus einen Straßengraphen zu bilden. Dieser Mechanismus wurde im Rahmen der OverDrive [6] Entwicklung für OverSim implementiert, um das Protokoll evaluieren zu können. Der Mechanismus wurde so erweitert, dass es nun möglich ist, jeder Kante einen Straßentyp wie z. B. *Autobahnauffahrt* oder *Landstraße* zuzuordnen. Des Weiteren wurde die C++ Datenstruktur, durch die der Graph repräsentiert wird, geändert. Näheres dazu wird im nächsten Kapitel *Implementierung* erläutert.

Eine weitere Aufgabe des Verkehrsmodells ist die Erzeugung und Verwaltung von Fahrzeugen. Als Fahrzeug wird ein Objekt verstanden, welches sich mit einer gewissen Geschwindigkeit auf einem Pfad fortbewegt und in der Lage ist, mit anderen Fahrzeugen zu kommunizieren. Die Geschwindigkeit ist nur von der Maximalgeschwindigkeit abhängig, welche auf der aktuellen Straße gefahren werden kann. In dieser Arbeit werden die Fahrzeuge nicht nach verschiedenen Typen wie z. B. LKWs, PKWs oder Motorräder unterschieden, da nur die Position und die Geschwindigkeit eines Fahrzeuges von Bedeutung für den Routenplaner sind. Im Rahmen der Entwicklung von OverDrive wurde auch hierfür ein Mechanismus für OverSim entwickelt, welcher Fahrzeuge erzeugt und auf einem Pfad fahren lässt. Allerdings fahren diese Fahrzeuge immer mit der Maximalgeschwindigkeit, welche auf der aktuellen Straße gefahren werden darf. Es ist nicht erwünscht, dass alle Fahrzeuge, welche sich auf derselben Straße befinden, mit der gleichen Geschwindigkeit fahren. Aus diesem Grund wurde der Mechanismus so erweitert, dass jedes Fahrzeug seine Geschwindigkeit zufällig aus einem festgelegten Intervall auswählt.

Die letzte Aufgabe des Verkehrsmodells ist die Erzeugung von Verkehrssituationen. OverSim bietet hierzu keinen Mechanismus. Es wurde daher ein entsprechender Mechanismus zur Erzeugung von Verkehrssituationen implementiert. Als Verkehrssituation wird ein Pfad im Straßengraphen verstanden. Die Fahrzeuge, welche auf diesem Pfad fahren, können nicht mit Maximalgeschwindigkeit fahren. Um die Komplexität des Verkehrsmodells gering zu halten, werden die Verkehrssituationen durch Anpassung der Kantengewichte im Straßengraphen erzeugt. Eine realistische Erzeugung der Verkehrssituationen aufgrund vieler Fahrzeuge auf den Straßen würde die Rechenzeit für eine Simulation deutlich erhöhen. Des Weiteren müsste dem Verkehrsmodell zu jeden Zeitpunkt die Position und Geschwindigkeit aller Fahrzeuge bekannt sein und es müsste verhindern, dass ein Fahrzeug durch ein anderes Fahrzeug hindurch fahren kann, indem es die Geschwindigkeiten der Fahrzeuge entsprechend beeinflusst. Dies würde zu noch höheren Rechenzeiten führen. Die Aufgabe einer Verkehrssituation besteht darin, die Geschwindigkeit der Fahrzeuge zu beeinflussen. Diese Aufgabe wird durch die Anpassung der Kantengewichte erfüllt. Die Verkehrssituationen sollen des Weiteren ein realistisches Verhalten aufweisen. Das heißt, eine Verkehrssituation muss sich zeitlich ändern. Zu diesem Zweck können sowohl konstante als auch dynamische Verkehrssituationen simuliert werden. Konstante Verkehrssituationen ändern sich nicht über die Zeit. Die Nicht-Konstanten

verringern bzw. erhöhen ihre Geschwindigkeit mit der Zeit. Das Verkehrsmodell erzeugt eine Verkehrssituation nun wie folgt: Zu Beginn wird zufällig ein Knoten aus dem Straßengraphen gewählt. Dieser Knoten bildet den Startpunkt der Verkehrssituation. Anschließend wird die Länge L der Verkehrssituation zufällig gewählt. Nun wird ein zufälliger Pfad zu einem zufälligen Zielknoten berechnet. Dieser Pfad hat eine geographische Länge von ungefähr L. Als nächstes wird aus einer Menge an Verkehrssimulationen zufällig bestimmt, um welche Verkehrssituation es sich handeln soll. Diese Verkehrssituation wird nun allen Kanten entlang des Pfades zugeordnet.

4.2 Kommunikationsmodell

Die zentrale Aufgabe des Kommunikationsmodells besteht darin, Fahrzeugen die Möglichkeit zu bieten, miteinander Nachrichten auszutauschen. Durch diesen Austausch soll der Routenplaner in der Lage sein, Informationen zur aktuellen Verkehrslage zu sammeln. Es kommt das Geocast Protokoll OverDrive zum Einsatz. Da OverDrive aber, wie schon in Kapitel 3.2.2.1 beschrieben wurde, eine lange Simulationszeit benötigt, wurde ein weiteres Geocast Protokoll implementiert, das zentralisierte Geocast Protokoll.

4.2.1 Zentralisiertes Geocast Protokoll

Bei diesem Geocast Protokoll existiert eine Instanz, den sogenannten *Observer*, der die Position aller Fahrzeuge kennt und Nachrichten direkt an das richtige Fahrzeug weiterleitet. Jedes Fahrzeug meldet sich zu Beginn seiner Fahrt bei diesem Observer an und der Observer fragt periodisch nach der aktuellen Position aller Fahrzeuge. Möchte ein Fahrzeug nun eine Nachricht senden, übergibt es diese an den Observer zusammen mit den Koordinaten, an die die Nachricht gesendet werden soll. Der Observer ermittelt nun das Fahrzeug, welches die geringste Entfernung zu diesen Koordinaten hat und übergibt ihm die Nachricht. Wie schon bei OverDrive gibt es auch bei diesem Protokoll die Möglichkeit, eine Nachricht in einem Gebiet zu *fluten*. Dazu wird wieder ein Gebiet in einer geometrischen Form definiert und der Observer übergibt die Nachricht an alle Fahrzeuge, welche sich in diesem Gebiet aufhalten. Sowohl OverDrive als auch dieses Protokoll sind aber darauf angewiesen, dass es viele Fahrzeuge gibt, die Informationen austauschen können. Je mehr Fahrzeuge simuliert werden müssen, desto länger ist die Simulationszeit. Aus diesem Grund wurde ein Geocast Modell entworfen, bei dem nur ein einziges Fahrzeug simuliert werden muss und das dadurch deutlich geringere Simulationszeiten ausweist.

4.2.2 Probabilistisches Geocast Modell

In Abschnitt 4.1 wurde erläutert, dass ein Stau nicht durch viele Fahrzeuge erzeugt wird, sondern dadurch, dass die Maximalgeschwindigkeit, die auf einer Straße gefahren werden kann, verringert wird. Die Information zur aktuellen Verkehrslage kann daher vom Verkehrsmodell direkt erfragt werden. Möchte ein Fahrzeug nun Informationen zur Verkehrslage auf einer Straße erhalten, wird auf den Straßengraphen des Verkehrsmodells zugegriffen und die entsprechende Geschwindigkeit wird abgelesen. Zudem kann eine Wahrscheinlichkeit festgelegt werden, mit der dieser Zugriff erfolgreich ist. Mit dieser Methode kann nur ein einziges Fahrzeug simuliert werden,

welches den Routenplaner verwendet, um mögliche Staus zu erkennen und zu umfahren. Dieser Ansatz ermittelt, wie das zentralisierte Geocast Protokoll und OverDrive, eine Menge an Knoten von denen Informationen erfragt werden sollen. Statt Fahrzeugen, welche sich in der Nähe dieser Knoten befinden, Anfragen zu schicken, werden die Kantengewichte der gewünschten Straßen direkt abgefragt. Dadurch ist die Simulationszeit bei dieser Methode deutlich geringer als bei OverDrive oder dem zentralisierten Geocast Protokoll. Da nur ein Fahrzeug benötigt wird, handelt es sich bei diesem Verfahren um ein rein theoretisches Verfahren. Es wird angenommen, dass sich auf jeder Kante des Straßengraphen, mit einer festgelegten Wahrscheinlichkeit, ein Fahrzeug befindet, welches Informationen über die Verkehrslage geben kann.

4.2.3 Zusammenfassung

In diesem Kapitel wurden drei Verfahren präsentiert, welche dem Routenplaner ermöglichen, Informationen zur aktuellen Verkehrslage zu sammeln. Mit OverDrive wurde ein dezentrales Geocast Protokoll verwendet. Da der Overhead, der durch OverDrive erzeugt wird, eine hohe Simulationszeit mit sich bringt, wurde ein zentralisiertes Geocast Protokoll implementiert. Dieses Geocast Protokoll erzeugt nahezu keinen Overhead und hat dadurch eine geringere Simulationszeit. Beide Verfahren benötigen aber eine Menge an Fahrzeugen um Staus zu erkennen. Diese Menge an Fahrzeugen sorgt allerdings wieder für eine hohe Simulationszeit. Aus diesem Grund wurde das probabilistische Geocast Modell implementiert. Bei diesem Modell sind keine Fahrzeuge auf den Straßen notwendig, um einen Stau zu erkennen. Die Informationen zur Geschwindigkeit werden direkt von dem Straßengraphen des Verkehrsmodells abgelesen. So ist es möglich, ein einziges Fahrzeug auf den Straßen fahren zu lassen, welches den Routenplaner benutzt, um Staus zu erkennen. Dies führt zu einer deutlich geringeren Simulationszeit. Durch die Vereinfachung des Modells wird aber die Realität nicht mehr so detailliert abgebildet.

4.3 Routenplaner

Der Routenplaner hat zwei zentrale Aufgaben. Zum einen muss der schnellste Pfad auf einem gegeben Straßengraphen gefunden werden. Dieser Pfad wird mit Hilfe von Dijkstras Algorithmus berechnet. Die Zeit, die benötigt wird, um einen Pfad entlang zu fahren, wird Reisezeit genannt. Es existieren immer zwei Reisezeiten. Die Eine ist die geschätzte Reisezeit. Sie wird vom Routenplaner anhand der gesammelten Informationen über die Verkehrslage berechnet. Die Andere ist die tatsächliche Reisezeit, welche das Fahrzeug benötigt um einen Pfad entlang zu fahren. Der Routenplaner besitzt zu Beginn der Fahrt eine Kopie des Straßengraphen. Die Kantengewichte in diesem Straßengraphen entsprechen der gesetzlichen Maximalgeschwindigkeit, welche auf dieser Straße gefahren werden darf. Verkehrssituationen, welche von dem Verkehrsmodell erzeugt wurden, sind in diesem Straßengraph zu Beginn nicht enthalten. Damit die angenommene Reisezeit mit der realen Reisezeit übereinstimmt, müssen Informationen über die aktuelle Verkehrslage gesammelt werden und der Straßengraphen des Routenplaners muss entsprechend aktualisieren werden. Dieses Sammeln der Informationen zu der aktuellen Verkehrslage ist die zweite zentrale Aufgabe des Routenplaners. Um diese Informationen zu sammeln, werden Nachrichten gezielt an geographische Koordinaten geschickt. Da jeder Knoten im Straßengraphen mit geographischen Koordinaten versehen ist, kann die Position einer

Straße ermittelt werden und eine Nachricht kann an Fahrzeuge gesendet werden, die sich auf dieser Straße befinden. Zur Vereinfachung wird im Folgenden gesagt, dass Nachrichten an Knoten gesendet werden. Gemeint ist damit, dass Nachrichten an Fahrzeuge gesendet werden, welche sich auf einer Straße befinden, die mit diesem Knoten verbunden ist. Die Nachrichten enthalten die geographischen Koordinaten des Knotens, von dem Informationen über die Verkehrslagen gesammelt werden soll und die Identifikationsnummer des Absenders der Nachricht. Erhält ein Fahrzeug so eine Nachricht, vergleicht es die eigene Position mit den geographischen Koordinaten aus der Nachricht. Ist das Fahrzeug nahe genug an diesen Koordinaten, sendet es seine aktuelle Position und die aktuell gefahrene Geschwindigkeit an den Absender zurück. Eine Möglichkeit, die Verkehrslage auf dem gesamten Straßengraphen zu erfahren, wäre, an jeden Knoten eine Nachricht zu verschicken. Der Routenplaner soll aber so wenige Nachrichten wie möglich verschicken. Um die Anzahl an Nachrichten zu verringern, müssen die Knoten ausfindig gemacht werden, welche relevant für die Berechnung des schnellsten Pfades sind. Die Relevanz eines Knoten sei nun wie folgt definiert: Es gibt zwei Straßengraphen G_V und G_{RP}. Die Menge an Knoten und Kanten sei identisch. Graph G_V sei der Straßengraph, welcher alle Verkehrssituationen enthält, also die Gewichte der Kanten entsprechend angepasst hat. Graph G_{RP} sei der Straßengraph des Routenplaners, welcher die Verkehrssituationen nicht enthält. Nun sei P_V der schnellste Pfad vom Startknoten zum Zielknoten im Straßengraph G_V. Der Pfad P_{RP} sei dementsprechend der schnellste Pfad im Straßengraph G_{RP}. Es gilt im Allgemeinen P_V ungleich P_{RP}. Ein Knoten aus G_{RP} sei dann *relevant*, wenn die Informationen zur Verkehrslage an diesem Knoten benötigt werden, damit die Berechnung des schnellsten Pfades auf G_{RP} einen Pfad P'_{RP} ergibt, welcher gleich P_V ist.

4.3.1 Kriterien für die Relevanz von Knoten

In diesem Abschnitt werden nun einige Kriterien vorgestellt und bewertet, mit denen die Relevanz von Knoten ermittelt werden sollen. Das erste Kriterium, um zu entscheiden ob ein Knoten relevant ist, wäre die geographische Entfernung sowohl zum Startknoten als auch zum Zielknoten. Wenn ein Knoten mehrere 100 oder gar 1000 Kilometer sowohl von Start als auch von Ziel entfernt ist, wird die Verkehrslage an diesem Knoten nicht relevant sein für die Pfadberechnung. Es ist aber nur schwer möglich, eine geeignete Abschätzung zu finden, ab welcher geographischen Entfernung ein Knoten nicht mehr relevant ist. Des Weiteren kann auf Grund von Unfällen oder Straßensperrungen ein großer Umweg nötig sein, um zum Ziel zu gelangen. In diesem Fall müssten nun aber Knoten betrachtet werden, die aufgrund ihrer geographischen Entfernung fälschlicherweise als nicht relevant eingestuft wurden. Aus diesen Gründen ist die geographische Entfernung kein geeignetes Kriterium, um zu entscheiden, ob ein Knoten relevant für die Pfadberechnung ist oder nicht. Ein anderes Auswahlkriterium wäre die zeitliche Entfernung eines Knotens von der aktuellen Position des Fahrzeugs. Mit der zeitlichen Entfernung ist die Anzahl an Stunden bzw. Minuten gemeint, die das Fahrzeug zurücklegen muss, um diesen Knoten zu erreichen. Ein Knoten im Straßengraphen, der frühestens nach einigen Stunden erreicht werden kann, ist weniger relevant als ein Knoten, der in wenigen Minuten erreicht werden kann, da sich die Verkehrslage an dem weit entfernten Knoten noch drastisch ändern kann, bevor man ihn erreicht. Auch bei dieser Variante kann schlecht abgeschätzt werden, ab welcher zeitlichen Entfernung ein Knoten nicht mehr relevant ist.

Denn auch hier gibt es Szenarien, in denen ein zeitlich weit entfernter Knoten relevant sein kann. So muss man z.B. z. B. schon frühzeitig wissen, ob ein Gebirgspass befahrbar ist oder nicht. Daher ist auch diese Variante für sich genommen kein gutes Kriterium, um zu entscheiden, ob ein Knoten relevant für die Pfadberechnung ist oder nicht. Die Analyse dieser beiden Ansätze zeigt, dass ein fester vordefinierter Zahlenwert, wie die geographische oder zeitliche Entfernung, für sich genommen, kein geeignetes Kriterium ist, um die Relevanz eines Knotens zu bestimmen.

4.3.2 Betrachtung von Pfaden

Des Weiteren ist die Betrachtung jedes einzelnen Knotens für sich genommen nicht sinnvoll. Ein Knoten wird immer Teil eines Pfades sein, wenn das Fahrzeug ihn erreicht. Es ist daher sinnvoller zu überprüfen, ob ein Pfad relevant ist. Ein Pfad sei dann relevant, wenn mindestens ein Knoten entlang des Pfades relevant ist. Der Pfad muss nicht immer der gesamte Pfad von Start- zu Zielknoten sein. Es können auch Teilstücke betrachtet werden. Ein Ansatz hierbei besteht darin, sich nur auf den aktuell gefahrenen Pfad zu beschränken. Bei diesem Ansatz werden nur Knoten nach der Verkehrslage gefragt, welche Teil des aktuellen Pfades sind. Mit dem aktuellen Pfad ist der Pfad gemeint, dem das Fahrzeug gerade folgt. Der Vorteil dieses Ansatzes ist, dass die Anzahl an gesendeten Nachrichten gering bleibt, da nur an einen geringen Teil aller Knoten aus dem Straßengraphen Nachrichten gesendet werden. Dieser Ansatz wird im Folgenden als *Straight-Forward-Ansatz* bezeichnet. Der Nachteil des Straight-Forward-Ansatzes ist der nur sehr eingeschränkte Blick auf die Verkehrslage des gesamten Straßengraphen, da aktuelle Informationen zur Verkehrslage immer nur von einer geringen Anzahl an Knoten vorliegen. Eine andere Möglichkeit wäre, sich mehrere Alternativpfade im Vorfeld, also bevor das Fahrzeug überhaupt losfährt, zu berechnen und auch diese Pfade nach der Verkehrslage zu befragen. Anhand dieser Informationen ist es möglich, von mehreren Verkehrssituationen gleichzeitig zu erfahren und diese zu umfahren. Der Vorteil dieses Ansatzes ist der bessere Überblick über die Verkehrslage. Der Nachteil ist die größere Menge an Nachrichten, die versendet werden müssen. Dieser Ansatz wird im folgenden *Alternativ-Graph-Ansatz* genannt.

4.3.3 Gemeinsamkeiten der Ansätze

Bevor aber die beiden Ansätze beschrieben werden, wird im Folgenden noch auf weitere Punkte eingegangen, welche beide Ansätze gleichermaßen betreffen. Es ist nicht nur wichtig, dass Informationen zur Verkehrslage erhalten werden, sondern auch *wann* sie erhalten werden. Die Information, dass sich auf der Autobahn ein Stau bildet, sollte erhalten werden, bevor die letzte Ausfahrt vor dem Stau erreicht wird. Eine Möglichkeit wäre, die Informationen über die Verkehrslage periodisch zu erfragen. Die Zeit zwischen zwei Anfragen darf aber nicht zu groß sein (viele Minuten oder gar Stunden), sonst wird eine wichtige Information vielleicht zu spät erhalten. Sie darf aber auch nicht zu klein sein (wenige Minuten oder gar Sekunden), da sonst zu viele Nachrichten verschickt werden und viele dieser Nachrichten keine neuen Informationen enthalten. Wie schon bei der geographischen bzw. zeitlichen Entfernung ist ein vordefinierter absoluter Wert für das Zeitintervall zwischen zwei Anfragen keine geeignete Lösung, da nicht garantiert werden kann, dass alle wichtigen Informationen rechtzeitig erhalten werden und dabei die Anzahl an gesendeten

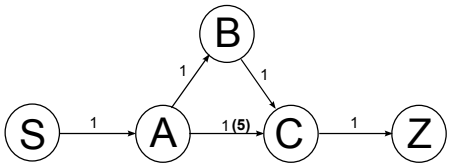

Abbildung 4.1: Beispiel Straßengraph. Knoten A ist der Startknoten. Knoten Z der Zielknoten. Die Zahl an einer Kante repräsentiert die Reisezeit für diese Kante

Nachrichten gering bleibt. Eine bessere Alternative ist, den Zeitpunkt einer Anfrage von der aktuellen geographischen Position des Fahrzeugs abhängig zu machen. Wenn sich das Fahrzeug auf einem langen Stück Autobahn befindet und die nächste Ausfahrt sowohl zeitlich als auch geographisch noch weit entfernt ist, müssen keine Anfragen gesendet werden, da das Fahrzeug in diesem Moment keine Möglichkeit hat, den Pfad zu ändern. Aktuelle Informationen zur Verkehrslage werden in diesem Beispiel erst dann benötigt, sobald sich das Fahrzeug der Ausfahrt nähert. Allgemein kann dieser Ansatz folgendermaßen formuliert werden: Eine Anfrage zur aktuellen Verkehrslage muss erst dann initiiert werden, sobald sich das Fahrzeug einem Knoten nähert, an dem sich der Pfad P des Fahrzeugs ändern könnte. Die Änderung eines Pfades P bedeutet in diesem Fall, dass ein Pfad P' von der aktuellen Position zum Zielknoten berechnet wird und dass es mindestens einen Knoten K gibt, welcher in P' aber nicht in P enthalten ist. Ein Knoten, an dem so eine Pfadänderung möglich ist, wird im Folgenden *Kreuzung* genannt. Der Zeitpunkt, wann eine Abfrage gesendet wird, ist abhängig von dem verwendeten Kommunikationsprotokoll. Es muss ausschließlich garantiert werden, dass die Zeit, bis der Knoten erreicht ist, ausreicht, um die Anfrage zu senden, die Antworten zu empfangen und der Fahrer des Fahrzeugs in der Lage ist, auf eine mögliche Pfadänderung zu reagieren. Ein weiterer Punkt, den beide Ansätze beachten müssen, ist der Informationsgehalt, den eine Antwort auf eine Anfrage hat. Jedes Fahrzeug fährt mit einer unterschiedlichen Geschwindigkeit. Sendet nun ein Fahrzeug, dass seine Geschwindigkeit deutlich langsamer ist, als auf dieser Straße möglich wäre, muss dies nicht automatisch bedeuten, dass ein Stau vorliegt. Dieses Fahrzeug kann aus den unterschiedlichsten Gründen langsamer fahren als eigentlich möglich wäre. Somit ist der Informationsgehalt einer Antwort nicht sehr groß. Um ein realistisches Bild der Verkehrslage auf einem Straßenabschnitt zu bekommen, müssen also Informationen von mehreren Fahrzeugen bezogen werden, welche sich auf diesem Abschnitt aufhalten. Dies wird mit Hilfe des geographischen Fluten-Mechanismus des Kommunikationsprotokolls erreicht. Dieser Mechanismus ermöglicht es dem Routenplaner, eine Anfrage an einen Knoten zu versenden, welcher aber von allen Fahrzeugen empfangen werden kann, die sich in einem bestimmten Gebiet um diesen Knoten aufhalten. Der Routenplaner kann also mit nur einer Anfrage viele verschiedene Informationen zu der Verkehrslage an einem Knoten erhalten. Nun muss noch ein geeigneter Weg gefunden werden, wie aus all diesen Informationen eine Geschwindigkeit ermittelt werden kann, die der tatsächlich möglichen Geschwindigkeit nahe kommt. Eine Möglichkeit wäre, den Durchschnitt über alle erhaltenen Geschwindigkeiten zu bilden. Da der Durchschnitt aber anfällig gegenüber extremen Abweichungen ist, wird in dieser Arbeit der Median der Geschwindigkeiten bestimmt. Im Folgenden werden nun die beiden Ansätze besprochen.

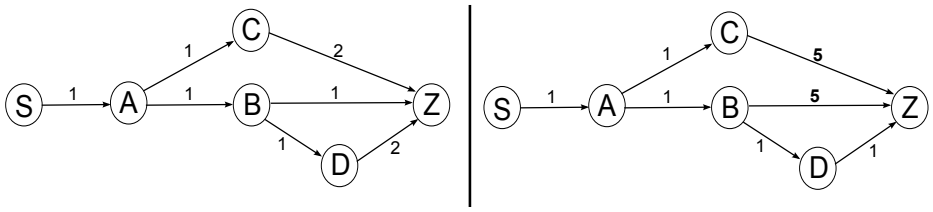

Abbildung 4.2: Beispiel Straßengraph. Knoten A ist der Startknoten. Knoten Z der Zielknoten. Links der Graph aus der Sicht des Routenplaners zu Beginn der Fahrt. Rechts der Graph mit der aktuellen Verkehrslage

4.3.4 Straight-Forward-Ansatz

Bei diesem Ansatz wird die Verkehrslage nur auf dem aktuellen Pfad ermittelt. Die Funktionsweise soll am folgenden Beispiel erläutert werden. Abbildung 4.1 zeigt einen Beispielgraphen. Die Zahlen an den Kanten sollen die Reisezeit darstellen, welche benötigt wird, um diese Kante zu passieren. Die schwarzen Zahlen repräsentieren die gesetzlichen Maximalgeschwindigkeiten. Die Zahl in Klammern repräsentiert einen Stau. Der Knoten S stellt den Start und der Knoten Z das Ziel dar. Zwischen Knoten A und Knoten C liegt eine Stausituation vor. Diese Stausituation ist dem Routenplaner nicht bekannt. Dieser Ansatz funktioniert nun wie folgt: Nachdem der Fahrer des Fahrzeugs das gewünschte Ziel, Knoten Z, eingegeben hat, wird mit Hilfe von Dijkstras Algorithmus der schnellste Pfad berechnet. Der schnellste Pfad sei jener Pfad, bei dem die Summe der Kantengewichte minimal ist. Dies ist in diesem Beispiel der Pfad $S \rightarrow A \rightarrow C \rightarrow Z$. Knoten A ist der erste Knoten, der eine Kreuzung darstellt. Daher werden die Anfragen nach der aktuellen Verkehrslage versendet, bevor Knoten A erreicht wird. Fahrzeuge, die in dem Stau zwischen Knoten A und Knoten C stehen, erhalten die Anfrage und senden ihre aktuelle Geschwindigkeit. Der Routenplaner aktualisiert seinen Straßengraphen anhand dieser Informationen und berechnet erneut den schnellsten Pfad. Aufgrund des Staus zwischen Knoten A und Knoten C ist der schnellste Pfad nun $A \rightarrow B \rightarrow C \rightarrow Z$. Der Routenplaner informiert den Fahrer über den neuen Pfad und dieser wird von Knoten A zu Knoten B fahren. Da es nun, bis der Zielknoten Z erreicht wird, keinen Knoten mehr gibt, welcher eine Kreuzung darstellt, werden keine neuen Anfragen mehr gesendet. Der Routenplaner war somit in der Lage, den Stau zu erkennen und zu umfahren. Wie gut der Routenplaner mit diesem Ansatz in der Lage ist, den schnellsten Pfad zu finden, hängt sehr stark von der Verkehrslage auf dem Straßengraphen ab, was das nächste Beispiel demonstrieren soll. In Abbildung 4.2 ist links ein Beispielgraph zu sehen. Der Knoten S markiert den Start und der Knoten Z markiert das Ziel. Dies ist der Zustand des Straßengraphen, wie ihn der Routenplaner zu Beginn der Fahrt kennt. Rechts ist der gleiche Straßengraph zu sehen, allerdings ist darüber hinaus noch die aktuelle Verkehrslage abgebildet. Nachdem der Fahrer des Fahrzeugs das gewünschte Ziel, Knoten Z, eingegeben hat, wird mit Hilfe von Dijkstras Algorithmus der schnellste Pfad berechnet. Dieser wird anhand des Straßengraphen im linken Teil der Abbildung ermittelt. Der schnellste Pfad in diesem Beispiel sei $S \rightarrow A \rightarrow B \rightarrow Z$. Knoten A ist der erste Knoten, welcher eine Kreuzung repräsentiert. Daher werden die Anfragen nach der aktuellen Verkehrslage versendet, bevor Knoten A erreicht wird. In der rechten Hälfte von Abbildung 4.2 ist zu sehen, dass zwischen Knoten B und Knoten Z ein Stau vorliegt. Fahrzeuge, die in diesem Stau stehen, erhalten die Anfrage und senden ihre aktuelle Geschwindigkeit.

Der Routenplaner aktualisiert seinen Straßengraphen anhand dieser Informationen und berechnet erneut den schnellsten Pfad. Aufgrund des Staus zwischen Knoten B und Knoten Z ist der schnellste Pfad nun $S \rightarrow A \rightarrow C \rightarrow Z$. Der Routenplaner informiert den Fahrer über den neuen Pfad und dieser wird von Knoten A zu Knoten C fahren. Da es nun, bis der Zielknoten Z erreicht wird, keinen Knoten mehr gibt, der eine Kreuzung darstellt, werden keinen neuen Anfragen mehr gesendet. Dies hat zur Folge, dass der Routenplaner nicht über den Stau zwischen Knoten C und Knoten Z informiert wird. Diese Information ist aber an diesem Punkt auch nicht mehr von Bedeutung, da es für den Fahrer keine Möglichkeit mehr gibt, den Stau zu umfahren. Dieses Beispiel zeigt die Schwäche dieses Ansatzes auf. Da dieser Ansatz nur einen sehr eingeschränkten Blick auf die gesamte Verkehrslage gibt, war es dem Routenplaner nicht möglich, alle Stausituationen zu erkennen und zu umfahren. Diese Schwäche kann wie folgt behoben werden: Im eben genannten Beispiel wird der Stau zwischen Knoten B und Knoten Z erkannt und es wird der schnellste Pfad $S \rightarrow A \rightarrow C \rightarrow Z$ berechnet. Statt diesen Pfad als den aktuellen Pfad zu bestimmen und ihm zu folgen, werden auch für diesen Pfad Anfragen versendet. Diese Anfragen ergeben, dass zwischen Knoten C und Knoten Z ebenfalls ein Stau existiert. Die erneute Anwendung von Dijkstras Algorithmus ergibt nun den schnellsten Pfad $S \rightarrow A \rightarrow B \rightarrow D \rightarrow Z$. Es wurden somit alle Staus erkannt und umfahren. Allgemein lässt sich diese Funktionsweise wie folge beschreiben: Der Routenplaner berechnet den schnellsten Pfad P, bestimmt ihn als aktuellen Pfad und versendet Anfragen zur aktuellen Verkehrslage auf diesem Pfad. Nachdem die Antworten empfangen wurden und der Straßengraph des Routenplaners entsprechend aktualisiert wurde, wird der schnellste Pfad P' berechnet. Unterscheiden sich P und P' so wird P' als aktueller Pfad bestimmt und es werden Anfragen für P' versendet. Dies wird solange wiederholt, bis sich der aktuelle Pfad nicht mehr vom schnellsten Pfad unterscheidet. Der Straight-Forward-Ansatz mit dieser Optimierung wird *iterativer Straight-Forward-Ansatz* genannt. Der Nachteil dieses Ansatzes liegt in der, unter Umständen, höheren Anzahl an Nachrichten. Aus diesem Grund wurden Simulationen sowohl mit dem Straight-Forward-Ansatz als auch mit dem iterativen Straight-Forward-Ansatz durchgeführt.

Im Folgenden wird nun noch auf das eigentliche Anfragen von Informationen zur Verkehrslage eingegangen. Damit der Routenplaner entscheiden kann, ob ein Pfad gut ist, muss er die Verkehrslage auf jedem Abschnitt des Pfades kennen. Eine Lösung, diese Informationen zu erhalten, wäre somit, jeden Knoten auf diesem Pfad nach der aktuellen Verkehrslage zu befragen. Die Anzahl an Knoten in einem Pfad wird aber nur durch die Anzahl an Knoten im ganzen Straßengraphen beschränkt. Es würden daher sehr viele Nachrichten versendet. Dazu kommt, wie schon im Abschnitt 4.1 beschrieben, dass die geographischen Abstände zwischen zwei Knoten sehr gering sein können. Eine Stausituation wird aber nie nur auf einigen wenigen Metern existieren. Es ist daher sinnvoll, mehrere Knoten zu einem Gebiet zusammenzufassen und nur eine Anfrage für dieses Gebiet zu senden. Dies kann mit Hilfe des Fluten Mechanismus des Kommunikationsprotokolls erreicht werden. Anfragen, welche so übertragen werden, erreichen alle Fahrzeuge, die sich innerhalb dieses Gebietes befinden. Die Anfrage soll aber nur von Fahrzeugen beantwortet werden, welche auf dem richtigen Pfad fahren und sich nicht z. B. auf der Gegenfahrbahn befinden. Um dies zu erreichen, werden die IDs der Knoten, welche dieses Teilstück des Pfades bilden, mit in der Anfrage übertragen. Jedes Fahrzeug, welches diese

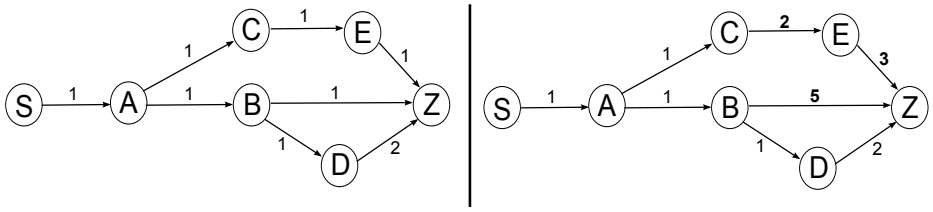

Abbildung 4.3: Beispiel Straßengraph. Knoten A ist der Startknoten. Knoten Z der Zielknoten. Links der Graph aus der Sicht des Routenplaners zu Beginn der Fahrt. Rechts der Graph mit der aktuellen Verkehrslage

Anfrage erhält, kann nun überprüfen, ob es auf dem richtigen Pfad fährt und gegebenenfalls antworten. Alle Antworten, die für dieses Gebiet empfangen wurden, werden zu einer Geschwindigkeit zusammengefasst. Diese Geschwindigkeit wird nun als maximale Geschwindigkeit in diesem ganzen Gebiet angesehen. Die Größe eines solchen Gebiets ist somit von Bedeutung. Wird das Gebiet zu groß gewählt, kann die Länge eines Staus überschätzt werden und ein potentiell guter Pfad wird verworfen. Wird das Gebiet jedoch zu klein gewählt, steigt die Anzahl an Nachrichten, die versendet werden müssen, um alle Informationen zur Verkehrslage auf diesem Pfad zu erhalten. Ein geeigneter Wert für die Größe des Gebietes wurde experimentell ermittelt und wird im Kapitel *Evaluierung* genannt. Ein Fokus dieser Arbeit ist die Reduzierung der Nachrichtenanzahl. Aus diesem Grund war eine weitere Überlegung, ob tatsächlich jeder Teil des Pfades nach der aktuellen Verkehrslage befragt werden muss oder ob es ausreicht, die Verkehrslagen stichprobenartig zu ermitteln. Mit dieser Methode geht zwar die Garantie verloren, dass jede Stausituation auf einem Pfad erkannt wird, es werden aber weniger Nachrichten übertragen. Es wird experimentell ermittelt, wie viele Bereiche des Pfades nicht nach der Verkehrslage befragt werden müssen, so dass ein Stau nicht erkannt wird. Auch diese Ergebnisse werden im Kapitel *Evaluierung* präsentiert.

4.3.5 Alternativ-Graph-Ansatz

Im Gegensatz zu dem Straight Forward Ansatz (SF-Ansatz) liegt der Fokus bei diesem Ansatz mehr auf dem Finden der schnellsten Pfade. Der Nachrichtenaufwand soll zwar weiterhin gering gehalten werden, es wird jedoch in Kauf genommen, dass möglicherweise mehr Nachrichten verschickt werden müssen, um einen besseren Überblick über die gesamte Verkehrslage zu erhalten. Zu diesem Zweck berechnet der Routenplaner mehrere Alternativpfade und es werden für jeden dieser Pfade Anfragen gesendet.

Es existieren mehrere Ansätze, wie solche sogenannten *Alternativpfade* berechnet werden können. Einer dieser Ansätze wurde von Abraham et al. [1] entwickelt. In ihrer Arbeit stellen sie drei Anforderungen an einen Pfad, damit dieser als Alternativpfad akzeptiert wird: Gegeben sei ein Graph G, ein Startknoten S und ein Zielknoten Z. Der kürzeste Pfad zwischen S und z sei $P_{s,z}$. Damit ein Pfad P' als Alternativpfad akzeptiert wird, darf P' nur eine kleine Anzahl an Knoten enthalten, welche auch Teil von P sind. P' darf also nur zu einem kleinen Teil mit P übereinstimmen. Des Weiteren sollen keine unnötigen Umleitungen existieren.

In Abbildung 4.4 ist ein Beispielgraph dargestellt, anhand dessen diese Anforderung erläutert werden soll. Gegeben sei der Pfad P':$s \rightarrow u \rightarrow v \rightarrow w \rightarrow z$. Da-

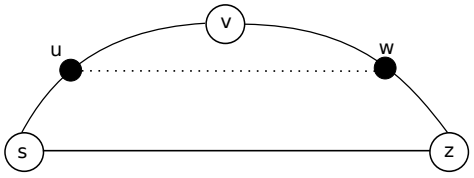

Abbildung 4.4: Beispielgraph. Damit der Pfad $s \rightarrow u \rightarrow v \rightarrow w \rightarrow z$ als Alternativpfad akzeptiert wird, darf es die gestrichelte Kante nicht geben.

mit P' als Alternativpfad akzeptiert wird, muss der schnellste Pfad von Knoten u zu Knoten z die beiden Knoten v und w enthalten. Würde der schnellste Pfad von Knoten u zu Knoten z z. B. den Knoten v nicht enthalten, würde P' nicht als Alternativpfad akzeptiert. Die letzte Anforderung betrifft die Anzahl an Knoten l des möglichen Alternativpfades: Die Anzahl an Knoten in einem Pfad P' $l(P')$ darf nicht größer sein als die Anzahl der Knoten in P $l(P) * (1 + \alpha)(0 \leq \alpha \leq 1)$. Durch diese drei Anforderungen werden Pfade ermittelt, welche sich nur gering mit dem schnellsten Pfad überdecken, die keine unnötigen Umleitungen enthalten und deren Anzahl an Knoten einen festgelegten Wert nicht überschreiten. Weitere Arbeiten zu dieser Möglichkeit Alternativpfade zu berechnen gibt es von Luxen et. al. [10] Damit der Routenplaner in der Lage ist, einen Pfad P' zu berechnen, der einen Stau umfährt, ist es jedoch nicht relevant, ob der schnellste Pfad P beinahe komplett mit P' übereinstimmt. Der Pfad P' darf nur nicht die Kante enthalten, auf der sich der Stau befindet. Aus diesem Grund wurde das via path Verfahren zur Berechnung von alternativen Pfaden für diese Arbeit nicht verwendet.

Ein weiteres Verfahren zu Berechnung von alternativen Pfaden ist das sogenannte *penalty Verfahren* [2]. Bei diesem Verfahren werden die Gewichte auf allen Kanten des schnellsten Pfades P um einen gewissen Wert erhöht. Anschließend wird mit Dijkstras Algorithmus der schnellste Pfad P' berechnet. Wurden die Kantengewichte von P weit genug erhöht, so unterscheidet sich P von P'. Wie groß der Unterschied zwischen P und P' ist, hängt zum größten Teil von dem Wert ab, um den die Kantengewichte erhöht wurden. Wird dieser Wert sehr niedrig gewählt, so können P und P' gleich sein oder sich nur gering unterscheiden. Wird dieser Wert sehr groß gewählt, unterscheiden sich P und P' ganz, sofern dies möglich ist. Mit Hilfe des penalty Verfahrens können, wie schon beim via path Ansatz, Pfade berechnet werden, welche sich deutlich von dem ursprünglich schnellsten Pfad unterscheiden. Es können aber auch Pfade berechnet werden, welche sich nur gering von dem schnellsten Pfad unterscheiden. Darüber hinaus ist die Implementierung des penalty Verfahrens weniger komplex, als die Implementierung des im vorherigen Abschnitt beschriebenen Verfahren. Daher wurde für diese Arbeit das penalty Verfahren implementiert. Weiter Arbeiten zum penalty Verfahren existieren von Paraskevopoulos et. al [14] und Kobitzsch et. al [7].

Mit Hilde der Alternativpfade werden Informationen zur Verkehrslage nicht nur für den aktuellen Pfad, sondern auch für andere Pfade zum Ziel erhalten. Die Berechnung der Alternativpfade erfolgt direkt nach der Eingabe des Ziels. Sie werden mit Hilfe des sogenannten *penalty Verfahren* berechnet, welches in Kapitel 3.2.3 beschrieben wurde. Der Ablauf dieses Ansatzes wird nun anhand von Abbildung 4.3 beschrieben: Der Fahrer gibt das Ziel Knoten Z in den Routenplaner ein. Mit Hilfe von Dijkstras Algorithmus wird der schnellste Pfad P zum Ziel berechnet. Dies ist

der Pfad $S \rightarrow A \rightarrow B \rightarrow Z$ mit einer Kantengewichtssumme von Drei. Als Grundlage für diese Berechnung wird ein Straßengraph links im Bild verwendet, der nur die gesetzlichen Maximalgeschwindigkeiten enthält. Anschließen wird das penalty Verfahren eingesetzt, um zwei Alternativpfade zu berechnen. Dies geschieht wie folgt: Zuerst wird zu jedem Kantengewicht einer Kante aus P der Wert 1 addiert. Die Kosten für P steigen somit auf sechs an. Anschließend wird erneut der schnellste Pfad P_{a1} berechnet. Dies ist nun der Pfad $S \rightarrow A \rightarrow C \rightarrow E \rightarrow Z$ mit Kosten von fünf. Nun werden die Kantengewichte für diesen Pfad um eins erhöht. Die Kosten für diesen Pfad steigen somit auf Neun. Als nächsten wird zum dritten Mal der schnellste Pfad P_{a2} berechnet. Dies ist der Pfad $S \rightarrow A \rightarrow B \rightarrow D \rightarrow Z$ mit Kosten von Acht. Nachdem nun die gewünschte Anzahl an Alternativpfaden berechnet wurde, werden die Kantengewichte wieder auf den Wert gesetzt, den sie vor der Berechnung hatten. Für jeden dieser Pfade werden nun Anfragen versendet. Nachdem die Antworten empfangen wurden, wird der verwendete Straßengraph mit den in den Antworten enthaltenen Geschwindigkeiten angepasst und es wird wieder mit Dijkstras Algorithmus der schnellste Pfad P' auf dem nun aktualisierten Straßengraphen berechnet. Der Pfad $P' = S \rightarrow A \rightarrow B \rightarrow D \rightarrow Z$ wird nun dem Fahrer vorgeschlagen. Dies ist ein Vorteil des AG-Ansatzes. Der erste Pfad, der dem Fahrer vorgeschlagen wird, enthält schon erste Verkehrsinformationen. Bei dem SF-oder ISF-Ansatz enthält der erste vorgeschlagene Pfad noch keine Verkehrsinformationen. Dies kann dazu führen, dass zu Beginn der Fahrt ein Pfad vorgeschlagen wird, dieser jedoch schon nach wenigen Minuten durch einen anderen Pfad ersetzt wird. Der Pfad P' sowie die alternativ Pfade P_{a1} und P_{a2} werden nun in einem separaten Graphen zusammengefasst. Dieser Graph wird *Alternativgraph* bezeichnet. In diesem Graphen sind nur die Knoten und Kanten enthalten, welche auch Teil der Pfade P', P_{a1} und P_{a2} sind. Das Fahrzeug folgt nun dem Pfad P'. Anders als bei dem SF- bzw. dem ISF-Ansatz werden Anfragen nur dann verschickt, wenn eine Kreuzung im Alternativgraph erreicht wird oder wenn eine Kreuzung im Straßengraph erreicht wird und das Fahrzeug einem Pfad folgt, der nicht Teil des Alternativgraph ist. Auf diese Art und Weise wird das Senden von Anfragen nicht so häufig ausgelöst. Wird eine Kreuzung erreicht, an der Anfragen gesendet werden sollen, so berechnet der AG-Ansatz erneut eine Menge an Alternativpfaden. Dies geschieht, da die Alternativpfade, welche zuvor berechnet wurden, ihre Gültigkeit verloren haben. Es wird anschließend genauso verfahren wie zu Beginn der Fahrt. Die Alternativpfaden und der schnellste Pfad werden wieder zu einem Alternativgraph zusammengefasst und das Fahrzeug folgt dem schnellsten Pfad, bis eine Kreuzung erreicht wird, an der Anfragen versendet werden sollen.

4.3.6 Zusammenfassung Routenplaner

Mit dem Straight Forward Ansatz, dem iterativen Straight Forward Ansatz und dem Alternativ Graph Ansatz wurden drei Ansätze präsentiert, die den Routenplaner in die Lage versetzen, Staus zu detektieren und die schnellste Route in einem Straßengraphen zu finden. Die Stärke des Straight Forward Ansatz liegt in der geringen Anzahl an versendeten Anfragen. Die Schwäche dieses Ansatzes liegt in der beschränkten Sicht auf die Verkehrslage des gesamten Straßengraphen. Der Vorteil des Alternativ Graph Ansatz liegt im detaillierteren Blick auf die Verkehrslage des gesamten Straßengraphen. Um dies zu erreichen, muss allerdings deutlich mehr Rechenaufwand betrieben werden und es müssen mehr Anfragen versendet werden. Um

die Vorteile von beiden Verfahren zu nutzen, bietet es sich an, beide Verfahren zu implementieren und immer das Verfahren zu benutzen, welches sich für die aktuelle Situation am besten eignet. So macht es zu Beginn einer Fahrt Sinn, mit dem Alternativ Graph Ansatz einen besseren Überblick über die gesamte Verkehrslage zu erhalten. Ergeben die ersten Anfragen aber, dass nur wenige Stausituationen vorliegen, so wird für die restliche Fahrt der Straight Forward Ansatz benutzt. Kommt es bei der Nutzung des Straight Forward Ansatz aber in kurzen zeitlichen Abständen zu häufigen Routenänderungen, so wird wieder der Alternativ Graph Ansatz aktiviert.

5. Implementierung

In diesem Kapitel wird nun auf die Implementierung eingegangen. Für jeden der drei Bereiche Verkehrsmodell, Kommunikationsmodell und Routenplaner werden wichtige Datenstrukturen bzw. Funktionen erläutert. Anschließend werden die Parameter präsentiert, welche das Verhalten des Routenplaners während einer Simulation beeinflussen.

5.1 Verkehrsmodell

Eine wichtige Datenstruktur im Verkehrsmodell ist der Graph, welcher das Straßennetz repräsentiert. Von der Implementierung dieser Datenstruktur hängt die Zeit ab, die benötigt wird, um mit Hilfe des Dijkstra Algorithmus den schnellsten Pfad in dem Graphen zu berechnen. Wie in Kapitel *Analyse* beschrieben, ermöglicht das Rahmenwerk OverSim, bereits Straßenkarten im XML-Dateiformat einzulesen und als Graph zu speichern. Diese Datenstruktur wird *RoadNetwork* genannt. In ihr werden die Knoten und Kanten des Graphen in Form von Vektoren von *pointern* gespeichert, welche auf die entsprechenden Knoten bzw. Kantenobjekte zeigen. Für jeden Knoten im Straßennetz wird ein Knotenobjekt erzeugt. Es wird allerdings nicht für jede Kante im Graphen ein Kantenobjekt erzeugt. Stattdessen werden nur so viele Kantenobjekte erstellt, wie es Kanten mit einzigartigen Attributen gibt. Gibt es nun zwei Kanten, welche die gleichen Attribute besitzen, so werden beide

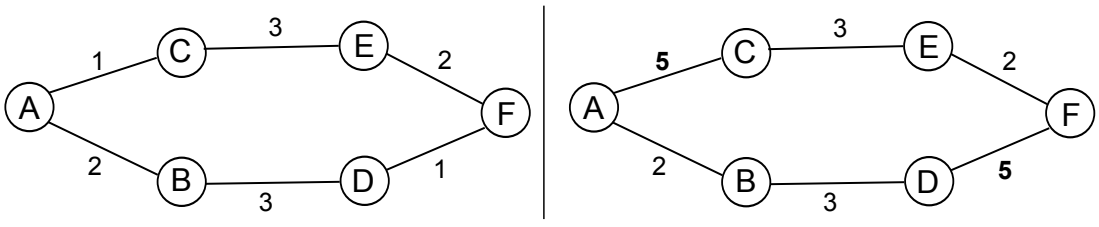

Abbildung 5.1: Beispielgraph, an dem veranschaulicht werden soll, weshalb die Datenstruktur RoadNetwork nicht geeignet ist. Links der ursprüngliche Graph, rechts der Graph, nachdem das Kantengewicht zwischen Knoten A und Knoten C geändert wurde

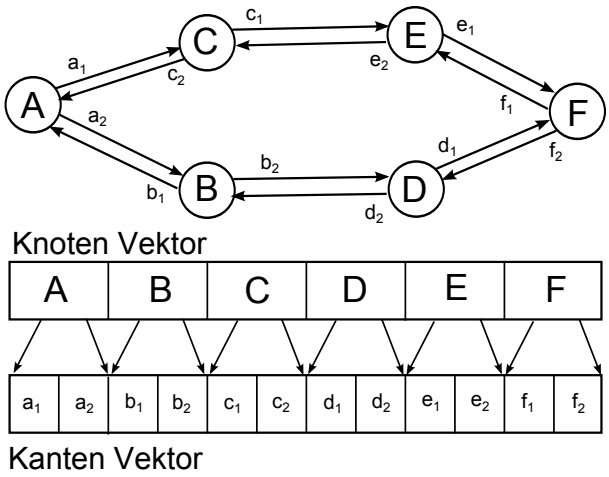

Abbildung 5.2: Beispielgraph, an dem die Datenstruktur Graph erläutert werden soll. Zu sehen ist sowohl ein Graph mit Kanten und Knoten als auch der Knoten- bzw. der Kantenvektor

durch dasselbe Kantenobjekt repräsentiert. Dies soll am folgenden Beispiel erläutert werden. In Abbildung 5.1 ist ein Beispielgraph zu sehen. Die Kosten für eine Kante seien in diesem Beispiel das einzige Attribut einer Kante. Alle Kanten, welche dieselben Kosten haben, werden daher durch dasselbe Kantenobjekt repräsentiert. Rechts in der Abbildung ist der ursprüngliche Graph zu sehen. Die Kante zwischen Knoten A und Knoten C wird durch dasselbe Kantenobjekt repräsentiert wie die Kante zwischen Knoten D und Knoten F. Werden nun aber die Kosten an der Kante zwischen Knoten A und Knoten C geändert, so ändern sich automatisch auch die Kosten auf der Kante zwischen Knoten D und Knoten F. Es ist mit dieser Datenstruktur nicht möglich, ein Attribut einer Kante zu ändern, ohne dabei auch das gleiche Attribut an anderen Kanten zu ändern, welche durch dasselbe Kantenobjekt repräsentiert werden.

Daher wurde die Klasse *Graph* implementiert, welche dieses Problem behebt. In dieser Datenstruktur werden alle Knoten bzw. Kanten eindeutig durch ein Knotenobjekt bzw. ein Kantenobjekt repräsentiert. Der Graph speichert sowohl die Knotenobjekte als auch die Kantenobjekte in einem Vektor. Im Knotenvektor werden die Knoten nacheinander gespeichert. Die Reihenfolge, wie die Knoten gespeichert werden, ist nicht relevant. Die Kanten werden allerdings so gespeichert, dass alle Kanten, welche von demselben Knoten ausgehen, nebeneinander in dem Vektor gespeichert werden. Dies wird in Abbildung 5.2 dargestellt. Darüber hinaus ist zu erkennen, dass jeder Knoten die Startposition und die Endposition seiner Kanten im Kantenvektor speichert. Des Weiteren ist die Berechnungszeit des schnellsten Pfades auf der Graph-Datenstruktur deutlich kürzer als auf der RoadNetwork-Datenstruktur. Dies liegt an der besseren Cache Effizienz der Graph-Datenstruktur. Wenn ein Prozess auf ein Feld in einem Vektor zugreifen muss, wird gleich ein ganzer Bereich des Vektors um dieses Feld herum in den Cache der CPU geladen. Möchte der Prozess nun anschließend auf ein benachbartes Feld des Vektors zugreifen, so liegen diese Daten bereits im CPU Cache und stehen zur Verfügung. Bei der RoadNetwork Datenstruktur werden nur die Pointer in den Cache geladen. Möchte ein Prozess aber auf ein Objekt zugreifen, so muss dieses Objekt erst noch in den Cache geladen werden. Aus diesem Grund ist die Berechnung des schnellsten Pfades auf der Graph-Datenstruktur

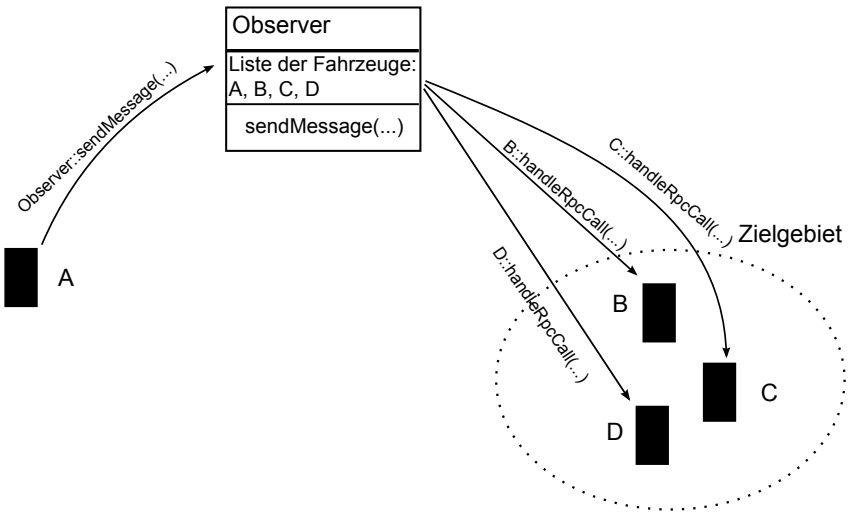

Abbildung 5.3: Beispiel anhand dessen die Funktion des Observers erläutert werden soll

deutlich schneller als auf der RoadNetwork-Datenstruktur. Auch wenn die Berechnungszeit für den schnellsten Pfad in der Graph-Datenstruktur deutlich kürzer ist, gibt es dennoch einige Methoden, um diese nochmal deutlich zu verkürzen. Einige dieser Methoden sind in einem Übersichtsartikel von D. Delling et al. [4] vorgestellt. Da der Routenplaner in dieser Arbeit aber nicht den Anspruch erhebt, in Echtzeit ausführbar zu sein, wurden diese Methoden nicht implementiert.

Erzeugung von Verkehrssituationen

Die *generateTrafficScenario* Funktion erzeugt die Verkehrssituationen auf dem Straßengraphen. Die Verkehrssituationen werden so erzeugt, dass sie sich, sowohl von Start- als auch von Zielknoten, nicht weiter entfernt befinden, also die geographische Entfernung vom Start- zum Zielknoten. Dies ist in Abbildung 5.4 dargestellt. Die Schraffierung markiert den Bereich, in dem eine Verkehrssituation erzeugt werden kann. Dies soll garantieren, dass sich die gewählte Anzahl an Verkehrssituationen in einem Bereich befindet, durch den das Fahrzeug fahren muss, um zum Ziel zu gelangen.

Bewegung der Fahrzeuge

Im Folgenden wird beschrieben, wie die Fortbewegung der Fahrzeuge gesteuert wird. Jedes Fahrzeug fährt entlang eines Pfades. Dieser Pfad besteht aus einer bestimmten Anzahl an Knoten, welche über Kanten verbunden sind. Ein Fahrzeug befindet sich immer genau auf einer Kante zwischen genau zwei Knoten. Die IDs dieser Knoten werden von jedem Fahrzeug gespeichert. Der Impuls, sich von einer Kante zur nächsten zu bewegen, erfolgt durch den Observer. Dieser speichert die Position aller Fahrzeuge und aktualisiert diese Positionen jede Sekunde. Ein Fahrzeug selbst aktualisiert seine Position erst, wenn diese von außen erfragt wurde. Kommt eine solche Anfrage, ermittelt das Fahrzeug die Zeit, die seit der letzten Anfrage vergangen ist. Anschließend wird die Geschwindigkeit bestimmt, welche auf der Kante gefahren werden kann, auf der sich das Fahrzeug zum Zeitpunkt der letzten Anfrage befand. Mit dieser Geschwindigkeit und der Zeit wird nun die Länge der Strecke berechnet,

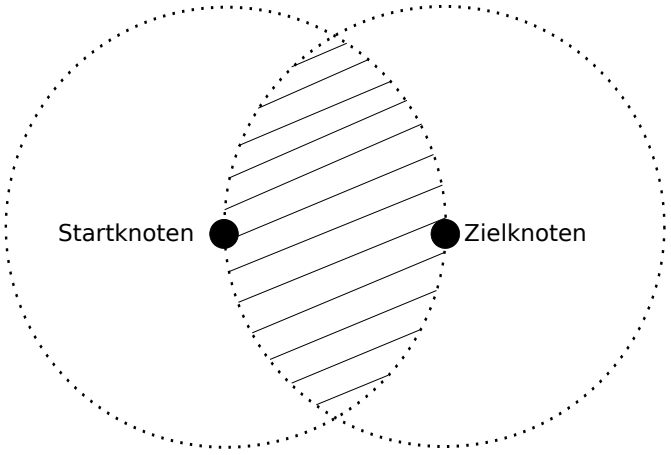

Abbildung 5.4: Schraffierter Bereich markiert einen Bereich, in dem eine Verkehrssituation erzeugt werden kann

welche das Fahrzeug in dieser Zeit zurückgelegt hat. Ist diese Strecke länger als der geographische Abstand zwischen den beiden Knoten, zwischen denen sich das Fahrzeug befunden hat, so bewegt sich das Fahrzeug zur nächsten Kante und die IDs der Knoten, welche durch diese Kante verbunden werden, ersetzen die IDs der alten Knoten. Diese Neuberechnung der Position wird immer dann ausgelöst, wenn die aktuelle Geschwindigkeit oder die Position des Fahrzeugs erfragt wird.

Zeitabhängiger Dijkstra-Algorithmus

Dieser Abschnitt beschreibt, wie der Dijkstra-Algorithmus angepasst wurde, damit er in der Lage ist, auch bei sich ändernden Kantengewichten noch den schnellsten Pfad in einem Graphen zu berechnen. Die Kantengewichte des Graphen werden durch die Funktion *getSpeedAtTime(double t)*$= a + (b * t)$ repräsentiert, wobei a ein konstanter Startwert ist und b entweder den Wert 1 oder -1 annimmt und somit festlegt, ob sich das Kantengewicht mit der Zeit erhöht oder verringert. Mit Hilfe dieser Funktion ist der Dijkstra-Algorithmus in der Lage, das Gewicht jeder Kante zu jedem beliebigen Zeitpunkt abzufragen. Die Funktionsweise des Algorithmus ist in Abbildung 5.5 in Pseudo Code dargestellt. Für jeden Knoten wird die Reisezeit berechnet, die benötigt wird, um zu ihm zu gelangen. Diese Zeit wird verwendet, um mit Hilfe der getSpeedAtTime-Funktion die tatsächliche Maximalgeschwindigkeit zu erfahren, welche auf allen von diesem Knoten ausgehenden Kanten gefahren werden kann. Mit Hilfe des zeitabhängigen Dijkstra-Algorithmus ist es möglich, den schnellsten Pfad zwischen einem Start- und einem Zielknoten für einen gegebenen Startzeitpunkt zu berechnen, auch wenn sich die Kantengewichte zeitlich ändern.

5.2 Kommunikationsmodell

Eine wichtige Datenstruktur des Kommunikationsmodells ist der sogenannte *Observer*. Der Observer ist eine globale Instanz, bei der sich alle Fahrzeuge anmelden. Der Observer speichert einen pointer zu jeder Fahrzeug-Instanz und greift periodisch auf deren Positionsdaten zu. Um den Fahrzeugen die Möglichkeit zu geben, Nachrichten auszutauschen, bietet der Observer die sogenannte *sendMessage* Funktion an. Die Funktionsweise der Nachrichtenübertragung soll an folgendem Beispiel demonstriert

```
Funktion Dijkstra(Graph, Source):
{
    initializeGraph(Graph,Source,travelTime[],predecessor[],Q)
     solange Q nicht leer:                    // Der eigentliche Algorithmus
        u := Knoten in Q mit kleinstem Wert in travelTime[]
        entferne u aus Q
        für jeden Nachbarn v von u:
           falls v in Q:
              distanz_update(u,v,travelTime[],predecessor[])
     return predecessor[]
}
Methode initializeGraph(Graph,Source,travelTime[],predecessor[],Q):
{
     für jeden Knoten v in Graph:
        travelTime[v] := unendlich
        predecessor[v] := null
     travelTime[Source] := 0
     Q := Die Menge aller Knoten in Graph
}
Methode distanz_update(u,v,travelTime[],predecessor[]):
{
     // Reisezeit vom Startknoten nach v über u
     alternativ := travelTime[u] + ( abstand_zwischen(u, v) / getSpeedAtTime(travelTime[u]) )
     falls alternativ < travelTime[v]:
        travelTime[v] := alternativ
        predecessor[v] := u
}
Funktion erstelleKürzestenPfad(Zielknoten,predecessor[])
{
  Weg[] := [Zielknoten]
  u := Zielknoten
  solange predecessor[u] nicht null:   // Der Vorgänger des Startknotens ist null
     u := predecessor[u]
     füge u am Anfang von Weg[] ein
return Weg[]
}
```

Abbildung 5.5: Pseudo Code des zeitabhängigem Dijkstra-Algorithmus

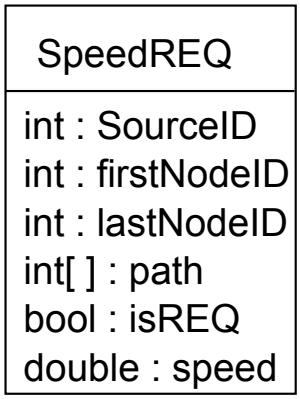

Abbildung 5.6: Klassendiagramm einer Anfrage-Nachricht

werden. In Abbildung 5.3 sind der Observer und vier Fahrzeuge dargestellt. Möchte ein Fahrzeug A nun eine Nachricht senden, so ruft es die sendMessage des Observers auf und übergibt dieser Funktion die Nachricht, die Koordinaten an welche die Nachricht gesendet werden soll und die Größe des Gebietes, in dem die Nachricht empfangen werden soll. Der Observer ermittelt anhand der Positionsdaten der Fahrzeuge, an welche dieser Fahrzeuge er die Nachricht weiterleiten muss. Damit der Observer eine Nachricht weiterleiten kann, existiert in jedem Fahrzeug die sogenannte *handleRpcCall* Funktion. Der Observer ruft die handleRpcCall Funktion der Fahrzeug-Instanz, an die die Nachricht weitergeleitet werden soll, auf und übergibt die Nachricht.

Aufbau einer Anfrage-Nachricht

Im Folgenden wird nun der Aufbau einer Anfrage-Nachricht beschrieben. Abbildung 5.6 zeigt das Klassendiagramm einer solchen Anfrage. Eine Anfrage enthält die IDs des Senders sowie des ersten und des letzten Knotens des Gebiets, auf das sich diese Anfrage bezieht. Darüber hinaus werden alle Knoten-IDs des Pfades gespeichert, für den die Anfrage gelten soll. Dies soll es einem Fahrzeug, welches die Anfrage empfängt, ermöglichen, zu entscheiden, ob die Anfrage tatsächlich für es bestimmt war und es antworten muss. Nur wenn sich ein Fahrzeug auf diesem Pfad befindet, wird es eine Antwort senden. Eine Anfrage-Nachricht kann auch zum Antworten auf eine Anfrage verwendet werden. Dazu wird das Feld *isREQ* auf *false* gesetzt und die aktuelle Geschwindigkeit wird in das Feld *speed* eingetragen.

5.3 Routenplaner

Der Routenplaner benutzt die Funktion *sendSpeedREQ*, um Anfragen an bestimmte Knoten des aktuellen Pfades zu senden. Dazu werden zuerst alle Knoten ermittelt, an die eine Anfrage gesendet werden soll. Hierfür wird der erste Knoten auf dem aktuellen Pfad ermittelt, der eine Kreuzung repräsentiert. Dies ist der erste Knoten k_1 in der Menge. Der nächste Knoten k_2, der in die Menge aufgenommen wird, muss einen festgelegten Mindestabstand zu k_1 aufweisen. Allgemein kann die Bestimmung der Knotenmenge wie folgt beschrieben werden: Ein Knoten wird in die Menge aufgenommen, wenn seine geographische Entfernung zu dem Knoten, der als letztes in die Menge aufgenommen wurde, einen festgelegten Mindestabstand

hat. An jeden Knoten in dieser Menge wird anschließend eine Anfrage gesendet. Kommt das Statistische Geocast-Protokoll zum Einsatz, wird keine Anfrage gesendet. Es wird stattdessen direkt auf die Verkehrsinformationen an diesem Knoten zugegriffen, indem die Daten direkt von dem Straßengraphen abgelesen werden. Ist der Alternativ-Graph-Ansatz im Einsatz, so wird zunächst eine festgelegte Anzahl an Alternativpfaden berechnet. Anschließend werden für alle diese Alternativpfade nach dem eben beschriebenen Prinzip Mengen von Knoten ermittelt und an diese Knoten Anfragen gesendet.

Eine weitere wichtige Funktion des Routenplaners ist die *checkBestRoute* Funktion. Diese Funktion wird immer dann aufgerufen, wenn Anfragen verschickt wurden und genug Zeit vergangen ist, um Antworten auf diese Anfragen erhalten zu haben. Die Hauptaufgabe dieser Funktion ist die Validierung des aktuellen Pfades. Dazu wird die Reisezeit des aktuellen Pfades ermittelt. Um diese Zeit zu ermitteln, werden ausschließlich die Informationen verwendet, welche durch Anfragen gesammelt wurden. Die dadurch ermittelte Reisezeit muss nicht der realen Reisezeit entsprechen. Anschließend wird mit Hilfe von Dijkstras Algorithmus der schnellste Pfad auf dem Straßengraphen des Routenplaners errechnet und dessen Reisezeit ermittelt. Ist die Reisezeit des aktuellen Pfades gleich der Reisezeit des schnellsten Pfades, so wird nichts verändert, da sich das Fahrzeug schon auf dem schnellsten Pfad befindet. Ist die Reisezeit des aktuellen Pfads jedoch größer, so wird der neu berechnete Pfad als aktueller Pfad bestimmt. Für den Fall, dass der Iterative Straight-Forward-Ansatz verwendet wird, verhält sich die checkBestPath Funktion wie folgt: Wurde festgestellt, dass der aktuelle Pfad nicht der schnellste Pfad ist, so wird der schnellste Pfad als aktueller Pfad bestimmt. Anschließend wird sofort die sendSpeedREQ Funktion aufgerufen, um Verkehrsinformationen zu dem neuen aktuellen Pfad zu sammeln.

5.4 Simulationsparameter

In Tabelle 5.1 sind die Simulationsparameter dargestellt. Im folgenden Abschnitt werden die Parameter vorgestellt und ihre Funktion erläutert. Der Parameter *useAltGraph* bestimmt, ob der Alternativ-Graph-Ansatz verwendet wird. Mit dem Parameter *altGraphSize* wird festgelegt, wie viele Alternativ-Routen berechnet werden. Der Parameter *useAltStraightForward* gibt an, ob der optimierte Straight-Forward-Ansatz verwendet wird. Mit Hilfe der Parameter *useStatTrafficInfo*, *useObserver* und *useOverDrive* wird festgelegt, welches Kommunikationsprotokoll zum Einsatz kommt. Durch den Parameter *minDistance* wird der Abstand zweier Knoten definiert, an die jeweils eine Anfrage zur aktuellen Verkehrsinformation gesendet wird. Die *REQSuccessRate* gibt die Wahrscheinlichkeit an, dass auf eine Anfrage auch eine Antwort erhalten wird, wenn das zentralisierte Geocast Protokoll bzw. das probabilistische Geocast Modell zum Einsatz kommen. Der Parameter *numberOfTrafficSituations* legt die Anzahl an Verkehrssituationen fest, welche vom Verkehrsmodell erzeugt werden. Durch den Parameter *floodMsgAreaRadius* wird der Radius des Gebiets bestimmt, für das eine Anfrage versendet wird.

Tabelle 5.1: Eine Liste der Simulationsparameter

Name	Wertebereich
useAltGraph	Gibt an ob der AG-Ansatz verwendet wird
altGraphSize	Bestimmt die Anzahl an zu berechnenden Alternativpfaden
useAltStraightForward	Legt fest ob der ISF-Ansatz benutzt wird
useStatTrafficInfo	Entscheidet ob das probabilistische Geocast Modell zum Einsatz kommt
useObserver	Gibt an ob das zentralisierte Geocast Protokoll verwendet wird
useOverDrive	Bestimmt ob OverDrive zum Einsatz kommt
minDistance in km	Legt den Abstand fest den ein Knoten zu einem anderen Knoten, an den bereits eine Anfrage gesendet wurde, habe muss damit an ihn auch eine Anfrage gesendet wird
REQSuccessRate	Bestimmt die Übertragungswahrscheinlichkeit einer Anfrage
numberOfTrafficSituations	Gibt an wie viele zusätzliche Verkehrssituationen erzeugt werden
floodMsgAreaRadius in km	Legt den Radius des Gebiets fest für das eine Anfrage versendet wurde

6. Evaluierung

Zu Beginn dieses Kapitels wird der Simulationsaufbau beschrieben. Anschließend werden die Ergebnisse der verschiedenen Ansätze und Protokolle für den Routenplaner bzw. das Kommunikationsmodell aus Kapitel 4 präsentiert und evaluiert.

6.1 Simulationsaufbau

Zunächst werden die Parametereinstellungen für das Verkehrsmodell präsentiert und erläutert. In Kapitel 4.1 wurde beschrieben, dass Kartenmaterial von OpenStreetMap verwendet wird. Dieses Kartenmaterial wurde so verändert, dass es nur noch die Autobahnen und Bundesstraßen in Baden-Württemberg enthält. Abbildung 6.1 zeigt das verwendete Straßennetz. Diese Einschränkung wurde eingeführt, da die Anzahl an Fahrzeugen, wie in Kapitel 4.2.3 erläutert wurde, für eine hohe Simulationszeit verantwortlich ist und somit nur begrenzt viele Fahrzeuge zum Einsatz kommen können. Mit einer begrenzten Anzahl an Fahrzeugen ist eine Simulation auf einem größeren Straßennetz, wie z. B. einem kontinentalen Straßennetz, nicht sinnvoll. Die Wahrscheinlichkeit, dass zwei Fahrzeuge in einem solchen Straßennetz auf derselben Straße fahren, ist sehr gering und der Routenplaner hätte nur eine geringe Chance, einen Stau zu erkennen und zu umfahren.

Eine Aufgabe des Verkehrsmodells ist die Erzeugung von Verkehrssituationen. Im Folgenden werden nun die drei verschiedenen Arten von Verkehrssituationen präsentiert, welche bei den Simulationen zum Einsatz kommen: Die drei Arten sind *freie Fahrt*, *stockender Verkehr* und *Stau*. Da die Verkehrssituationen dynamisches Verhalten aufweisen sollen, existieren zu jeder Art mindestens zwei Unterarten, welche entweder ein dynamisches oder ein konstantes Verhalten ausweisen. Daraus ergeben sich insgesamt sieben verschiedene Verkehrssituationen, welche in Tabelle 6.1 dargestellt sind. Ebenfalls zu sehen sind die Geschwindigkeiten, welche bei den entsprechenden Verkehrssituationen gefahren werden können. Die Evaluierung des Routenplaners und der verwendeten Protokolle auf reale, im täglichen Verkehr vorkommende, Verkehrssituationen war kein Ziel dieser Arbeit. Das Verkehrsmodell sollte verschiedene dynamische Verkehrssituationen erzeugen, um die Funktionalität des Routenplaners bzw. der eingesetzten Protokolle zu evaluieren. Aus diesem Grund

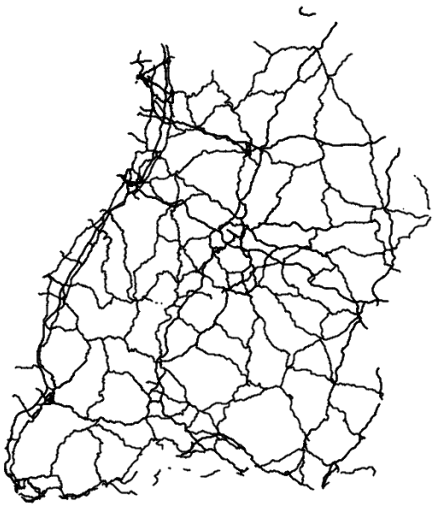

Abbildung 6.1: Verwendetes Straßennetz. Es kommen nur die Autobahnen und Bundesstraßen in Baden-Württemberg zum Einsatz

Tabelle 6.1: Liste der verschiedenen Verkehrssituationen und deren Geschwindigkeiten

Name	**Geschwindigkeit**	**dynamisches Verhalten**
konst. freie Fahrt	gesetzliche Maximalgeschwindigkeit	kein dynamisches Verhalten
freie Fahrt, Tendenz zu Stau	Zu Beginn gesetzliche Maximalgeschwindigkeit	Geschwindigkeit verringert sich, bis sie Stauniveau erreicht
konst. stockender Verkehr	20% der gesetzliche Maximalgeschwindigkeit	kein dynamisches Verhalten
stockender Verkehr, Tendenz zu Stau	Zu Beginn 20% der gesetzliche Maximalgeschwindigkeit	Geschwindigkeit verringert sich, bis sie Stauniveau erreicht
stockender Verkehr, Tendenz zu freie Fahrt	Zu Beginn 20% der gesetzliche Maximalgeschwindigkeit	Geschwindigkeit erhöht sich, bis sie die gesetzliche Maximalgeschwindigkeit erreicht
konst. Stau	$5km/h$	kein dynamisches Verhalten
Stau, Tendenz zu freie Fahrt	Zu Beginn $5km/h$	Geschwindigkeit erhöht sich, bis sie die gesetzliche Maximalgeschwindigkeit erreicht

Tabelle 6.2: Mögliche Längen einer Verkehrssituation und die Häufigkeit von Verkehrssituationen

	Wertebereich
Länge einer Verkehrssituation	$1km$-$15km$
Anzahl Verkehrssituationen in einer Simulation	0, 10, 20, 30

wurden sowohl die Arten der Verkehrssituationen als auch deren Geschwindigkeiten frei gewählt. Die Evaluierung des Routenplaners auf realen Verkehrssituationen könnte das Thema einer weiteren Arbeit sein. In jeder Simulation wird auf dem zu Beginn der Fahrt gewählten Pfad eine konstante Stausituation vom 15km erzeugt. Dies soll garantieren, dass der Routenplaner mindestens einmal einen Stau erkennen und umfahren muss.

In Tabelle 6.2 ist dargestellt, welche Werte für die Länge einer Verkehrssituation in dieser Arbeit verwendet werden und wie viele Verkehrssituationen in einer Simulation vorkommen können. Diese Werte wurden anhand der ADAC-Staubilanz von 2013 [1] gewählt. In dieser Bilanz werden für Baden-Württemberg 46.687 verschiedene Staus gemeldet, welche zusammen eine Gesamtlänge von 108.003 Staukilometern ergeben. Nach diesen Werten hatte somit jeder Stau eine durchschnittlich Länge von $2,3$km. Die Staulänge von 15km wurde in dieser Arbeit als Maximum gewählt, um auch Ausnahmesituationen wie die Vollsperrung einer Autobahn zu simulieren. Der Wert von 46.687 Staus in einem Jahr bedeutet durchschnittlich 128 Staus pro Tag. Dies wiederum bedeutet durchschnittlich 6 unterschiedliche Staus pro Stunden. Die Fahrtzeit auf den in dieser Arbeit verwendeten Strecken betrug ohne Stau maximal drei Stunden. Anhand dieser Werte wurde der Wertebereich der Anzahl an Verkehrssituationen pro Simulation geschätzt. Mit 30 Verkehrssituationen wurde eine hohe Anzahl an Verkehrssituationen gewählt, um Ausnahmesituationen, wie z. B. der Beginn der Sommerurlaubszeit, zu simulieren, bei denen ein höheres Verkehrsaufkommen existiert.

In Kapitel 3.2.1 wurde gefordert, dass die Geschwindigkeiten von Fahrzeugen, welche auf derselben Straße fahren, variieren. Um dies zu realisieren, wählt jedes Fahrzeug seine Geschwindigkeit zufällig aus einem Intervall aus. Die untere Grenze des Intervalls liegt bei 70% der Maximalgeschwindigkeit und die obere Grenze liegt bei der Maximalgeschwindigkeit, welche auf dieser Straße gefahren werden kann. Wann ein Fahrzeug seine Geschwindigkeit wählt und wie lange diese Geschwindigkeit gültig ist, wurde im Kapitel 5.1 beschrieben. Auch die Grenzen für die Geschwindigkeit wurden frei gewählt. Fahrzeuge wählen ihre Geschwindigkeit so, dass sie entweder langsamer oder genauso schnell fahren, wie maximal auf dieser Straße möglich. Der Grund dafür liegt in der verwendeten Metrik, mit der der Routenplaner evaluiert wird. Damit evaluiert werden kann, ob der Routenplaner alle Staus erkannt und umfahren hat, wird die Reisezeit des gefahrenen Pfads mit der Reisezeit des schnellsten Pfads verglichen. Der schnellste Pfad stellt das Optimum dar. Um zu garantieren, dass die Reisezeit des schnellsten Pfads nicht unterboten wird, muss es eine Maximalgeschwindigkeit geben, welche von den Fahrzeugen nicht überschritten werden kann.

Als Nächstes werden die Parametereinstellungen für das Kommunikationsmodell vorgestellt. Es kommen die in Kapitel 3.2.2 als auch in Kapitel 4.2 beschriebenen Pro-

[1] http://www.adac.de/_mmm/pdf/statistik_staubilanz_0514_68897.pdf

tokolle *OverDrive* und *zentralisiertes Geocast Protokoll* sowie das *probabilistisches Geocast Modell* zum Einsatz. Für jede Simulation kommt immer genau eines der zwei Protokolle oder das Modell zum Einsatz. Ein wichtiger Parameter, der sich auf alle drei Möglichkeiten bezieht, ist die Anzahl an Fahrzeugen. Für *OverDrive* werden 5000 Fahrzeuge simuliert. Für das *alternative Geocast Protokoll* werden 5000 10000 und 20000 Fahrzeuge simuliert und für das *probabilistische Geocast Modell* wird nur ein Fahrzeug simuliert. Die Anzahl an Fahrzeugen, welche bei OverDrive eingesetzt wird, wurde experimentell bestimmt. Bei 5000 Fahrzeugen benötigt eine Simulation mit OverDrive bis zu mehreren Tagen. Eine größere Menge war somit nicht praktikabel. Die Anzahl an Fahrzeugen für das zentralisierte Geocast Protokoll wurde so gewählt, dass die Ergebnisse zum Einen mit den Ergebnissen von OverDrive verglichen werden können und zum Anderen evaluiert werden kann, wie sich die Anzahl an Fahrzeugen auf die Funktionalität des Routenplaners auswirkt. Ein weiterer Parameter, der sich nur auf das zentralisierte Geocast Protokoll sowie auf das probabilistische Geocast Modell bezieht, ist die Übertragungswahrscheinlichkeit *REQSuccessRate* für Nachrichten. In dieser Arbeit wurden die vier Übertragungswahrscheinlichkeiten 100%, 80%, 50% und 30% gewählt. Diese Wahrscheinlichkeiten beziehen sich auf das Übertragen von einer Anfrage. So entscheidet beim zentralisierten Geocast Protokoll der Observer, ob eine Anfrage an andere Fahrzeuge weitergeleitet wird oder nicht.

Zum Abschluss werden nun die Parametereinstellungen für den Routenplaner erläutert. In Kapitel 4.3 wurden drei Ansätze präsentiert, welche es dem Routenplaner ermöglichen, einen Stau zu erkennen und zu umfahren. Bei diesen Ansätzen handelt es sich um den *Straight-Forward-Ansatz* (SF), den *iterativen Straight-Forward-Ansatz* (ISF) und den *Alternativ-Graph-Ansatz* (AG). In jeder Simulation kommt immer genau einer dieser Ansätze zum Einsatz. Der Parameter *minDistance* ist für alle drei Ansätze von Bedeutung, da er stark dafür verantwortlich ist, wie viele Anfragen der Routenplaner für einen gegebenen Pfad versendet. Er legt den Mindestabstand fest, den zwei Knoten haben müssen, damit an beide Knoten eine Anfrage gesendet wird. Je größer dieser Parameter gewählt wird, desto weniger Anfragen werden für einen gegebenen Pfad versendet. Der Einfluss von minDistance wurde im Kapitel 5.3 detailliert beschrieben. Ein weiterer Parameter, welcher von allen drei Ansätzen verwendet wird, ist *floodRadius*. Dieser Parameter bestimmt den Radius eines Gebietes, auf welchen sich eine Anfrage bezieht. Damit sich die Gebiete aus zwei Anfragen nicht überschneiden können, wird gefordert, dass der Wert für minDistance größer oder gleich dem *Durchmesser* (2∗floodRadius) des Gebietes ist. Der Parameter *altGraphSize* ist hingegen nur für den AG-Ansatz von Bedeutung. Er bestimmt, wie viele Alternativpfade zum Ziel berechnet werden. Ein weiterer Parameter, welcher nur für den AG-Ansatz eine Bedeutung hat, ist der *penaltyFactor*. Dieser bestimmt, wie stark die Kantengewichte des Straßengraphen verringert werden, um einen alternativen Pfad zu berechnen. Der Wert dieses Parameters ist für alle Simulationen gleich 0.5. In Tabelle 6.3 ist der Wertebereich für die drei Parameter *minDistance, floodRadius* und *altGraphSize* dargestellt. Dieser Wertebereich ergab sich durch Simulationen während der Entwicklungsphase der verschiedenen Ansätze.

Aus den im letzten Abschnitt genannten Protokollen, Ansätzen und deren Parametern ergibt sich eine Vielzahl an verschiedenen Konfigurationen für Simulationen. Darüber hinaus wird jede Konfiguration fünfmal mit einem anderen Seed für den Zufallsgenerator wiederholt. Durch die lange Simulationszeit für sowohl OverDrive

Tabelle 6.3: Wertebereiche für die drei Parameter des Routenplaners

Parameter	**Wertebereich**
minDistance	1km, 2km, 4km, 8km, 16km, 32km
floodRadius	0.125km, 0.25km, 0.5km, 1km, 2km, 4km, 8km, 16km
altGraphSize	3, 5, 7

Tabelle 6.4: Werte der Konfigurationen für den SF-Ansatz mit dem besten Trade-Off

	minDistance	**floodRadius**	**Reisezeit**	**Bandbreite**
Konfig 1	16km	2km	102.24%	58.65kB
Konfig 2	32km	4km	103.81%	31.76kB

als auch für das zentralisierte Geocast Protokoll, ist die Simulation von derart vielen verschiedenen Konfigurationen nicht realisierbar. Aus diesem Grund wurde mit Hilfe des probabilistischen Geocast Modells eine geeignete Konfiguration für jeden der drei Ansätze SF,ISF und AG ermittelt. Diese Konfiguration kam anschließend sowohl für OverDrive als auch für das zentralisierte Geocast Protokoll zum Einsatz.

6.2 Evaluierung der Ergebnisse

In diesem Kapitel werden nun die Ergebnisse der einzelnen Simulationen evaluiert. Für die Evaluierung kommen zwei Metriken zum Einsatz: Die Reisezeit und die verbrauchte Bandbreite. Für die Reisezeit wird die relative Reisezeit des gefahrenen Pfades zu dem schnellsten Pfad in Prozent gebildet. Die Reisezeit gibt somit den Faktor in Prozent an um der der gefahrene Pfad zeitlich Länger ist, als der schnellste Pfad. Je kleiner diese Reisezeit ist desto besser hat der Routenplaner gearbeitet. Wird eine relative Reisezeit von 100% erreicht, so wurde der schnellste Pfad zum Ziel gefunden. Da die Reisezeit des schnellsten Pfades nicht vom Zufall abhängen soll, wird zur Berechnung der Reisezeit die Maximalgeschwindigkeit zu Grunde gelegt, welche auf den Straßen des Pfades gefahren werden kann. Des Weiteren soll dieser Pfad das Optimum darstellen. Es soll daher nicht möglich sein, eine Reisezeit zu erzielen, die kleiner ist als die Reisezeit des schnellsten Pfades. Aus diesem Grund wählen die Fahrzeuge ihre Geschwindigkeit so, dass sie höchstens die Maximalgeschwindigkeit fahren, welche auf einer Straße erlaubt ist. Mit der Reisezeit soll gemessen werden, in wie weit der Routenplaner in der Lage war, den schnellsten Pfad zu finden. Wie der schnellste Pfad berechnet wird, wurde in Kapitel 5.3 beschrieben. Im folgenden Abschnitt wird die geeignete Konfiguration für die drei Ansätze bestimmt. Anschließend werden die Ergebnisse der Simulationen von OverDrive und dem zentralisierten Geocast Protokoll präsentiert, welche diese Konfigurationen verwendet haben.

6.2.1 Einschränken der Wertebereiche

Im Folgenden wird nun anhand von Diagrammen gezeigt, für welche Wertebereiche die verschiedenen Parameter einen guten Trade-Off zwischen kleiner Reisezeit und kleiner Bandbreite erreichen. Es werden die Konfigurationen für jeden der drei Ansätze SF, ISF und AG ermittelt, die den besten Trade-Off zwischen Reisezeit und Bandbreite erreichen. Zu Beginn wird die Konfiguration für den SF-Ansatz ermittelt.

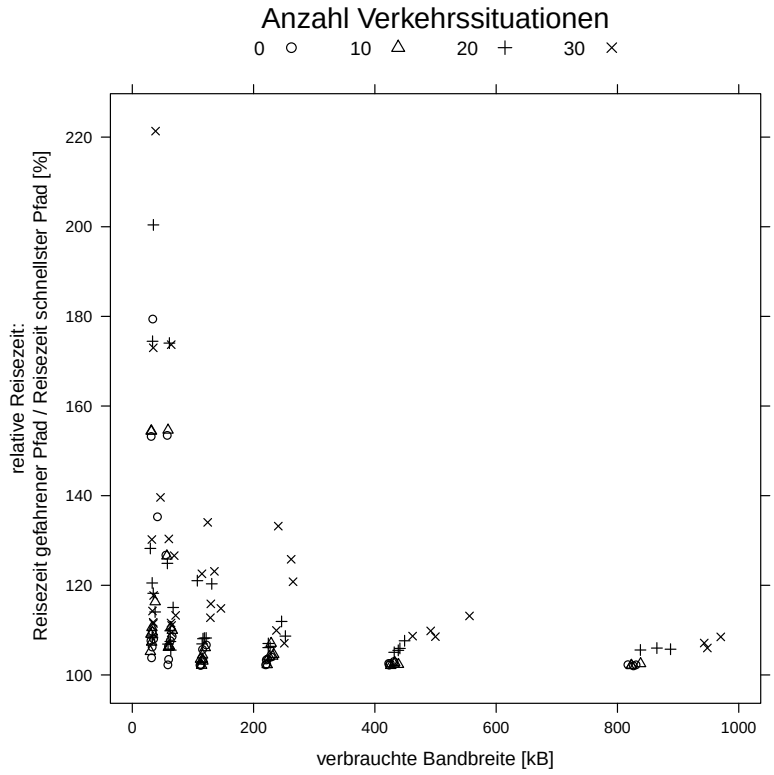

Abbildung 6.2: Ergebnisse aller Simulationen für den SF-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Die X-Achse zeigt die verbrauchte Bandbreite in kB. Die Y-Achse zeigt die relative Reisezeit in Prozent

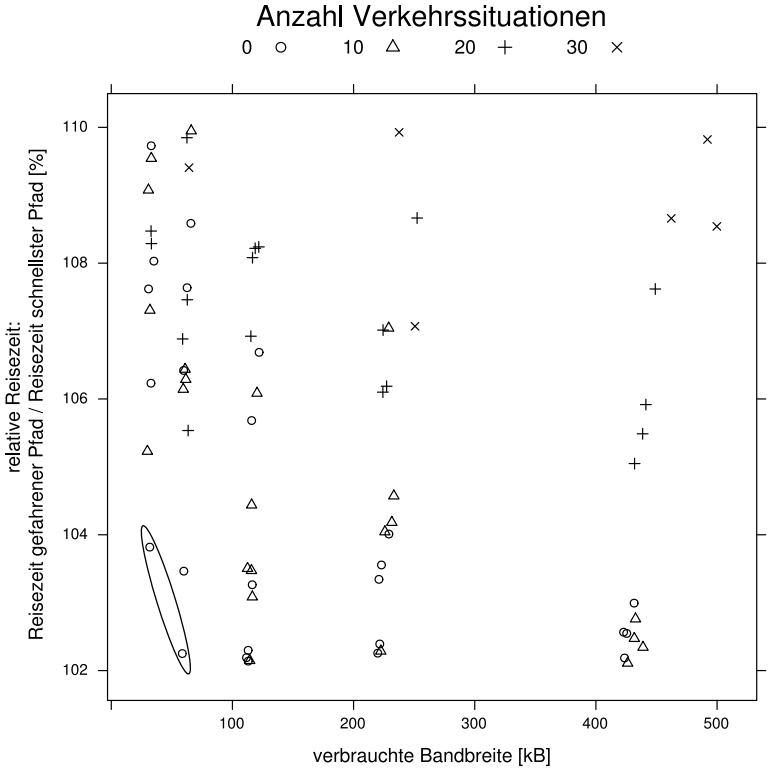

Abbildung 6.3: Ergebnisse aller Simulationen für den SF-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Die X-Achse zeigt die verbrauchte Bandbreite in kB. Die Y-Achse zeigt die relative Reisezeit in Prozent. Es werden nur die Ergebnisse angezeigt, welche einen Bandbreitenverbrauch von weniger als 500kB und eine Reisezeit von weniger als 110% bewirken.

SF-Ansatz

Abbildung 6.2 zeigt die Ergebnisse für alle Simulationen des SF-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Auf der Y-Achse wird die relative Reisezeit des schnellsten Pfads zu dem gefahrenen Pfad in Prozent gezeigt. Die X-Achse zeigt die verbrauchte Bandbreite in Kilobyte. Wie zu erkennen ist, erzielen die Simulationen entweder eine kleine Reisezeit mit stark schwankendem Bandbreitenverbrauch oder sie verbrauchen wenig Bandbreite bei sehr unterschiedlichen Reisezeiten. Es existiert keine Konfiguration, bei der sowohl eine hohe Bandbreite verbraucht wird als auch eine große Reisezeit benötigt wird. Es wird nun die Konfiguration gesucht, welche die kleinste Reisezeit bei geringem Bandbreitenverbrauch mit sich bringt. In Abbildung 6.3 ist ein Ausschnitt der vorherigen Grafik zu sehen. In diesem Ausschnitt werden nur die Konfigurationen gezeigt, welche einen Bandbreitenverbrauch von weniger als 500kB und eine Reisezeit von weniger als 110% bewirken. Das schwarze Oval markiert die beiden Konfigurationen, welche für den SF-Ansatz die kleinste Reisezeit bzw. den geringsten Bandbreitenverbrauch haben. Die Werte der verschiedenen Parameter für die einzelnen Konfigurationen sind in Tabelle 6.4 dargestellt. Es ist wenig verwunderlich, dass in beiden Konfigurationen nur der Stau auf dem ursprünglichen Pfad existiert. Wenn es nur einen Stau gibt, muss auch nur ein Stau erkannt und umfahren werden. Interessant hingegen ist die Tatsache, dass der Wert für floodRadius nicht genau der Hälfte von minDistance entspricht. Durch den kleineren Wert werden an einigen Abschnitten entlang eines Pfads keine Anfragen versendet bzw. gibt es Abschnitte, die zu keinem Gebiet gehören, für welches eine Anfrage versendet wurde. Dies hat den folgenden Grund: Kommt für das probabilistische Geocast Modell eine Übertragungswahrscheinlichkeit von 100% zum Einsatz, so bedeutet dies, dass angenommen wird, dass auf jeder Straße mindestens ein Fahrzeug fährt, welches Informationen zur Verkehrslage geben kann. Jedes Fahrzeug fährt aber mit einer Geschwindigkeit, welche kleiner oder gleich der Maximalgeschwindigkeit ist. In Kapitel 4.3.3 wurde beschrieben, dass über alle Geschwindigkeiten, welche auf eine Anfrage hin empfangen worden sind, der Median gebildet wird und diese Geschwindigkeit für alle Straßen des Gebiets, für das die Anfrage gesendet wurde, als Maximalgeschwindigkeit angenommen wird. Die Geschwindigkeit, welche auf einer Straße ohne Stau detektiert wurde, wird in der Regel kleiner sein als die tatsächliche Maximalgeschwindigkeit. Wird der Parameter floodRadius nun so gewählt, dass er der Hälfte von minDistance entspricht, so werden Anfragen für jede Straße versendet, was wiederum dazu führt, dass die Geschwindigkeit auf allen Straßen angepasst wird. Dies kann dazu führen, dass ein Pfad schlechter bewertet wird als er ist und ein anderer Pfad, der eigentlich schlechter ist, ausgewählt wird. Ist der Parameter floodRadius hingegen kleiner als die Hälfte von minDistance, so existieren Straßen, für die keine Anfragen gesendet werden. Die Geschwindigkeiten für diese Straßen werden somit auch nicht angepasst und können daher auch nicht schlechter bewertet werden als sie sind. In einem Szenario, in dem nur ein Stau existiert, werden daher mit einer Konfiguration, welche es dem Routenplaner ermöglicht, den Stau zu erkennen ohne die Geschwindigkeit auf jeder Straße anzupassen, in der Regel die besseren Ergebnisse erzielt, als mit einer Konfiguration, bei der die Geschwindigkeiten für alle Straßen angepasst werden. Des Weiteren kann in Tabelle 6.4 gesehen werden, dass die Werte für den minDistance Parameter die beiden größten Werte sind, welche dieser Parameter annehmen kann. Wie in Kapitel 5.4 beschrieben, hängt die Anzahl an gesendeten Anfragen und somit auch die

verbrauchte Bandbreite stark von dem minDistance Parameter ab. Je größer dieser Parameter ist, desto weniger Anfragen werden pro Pfad gesendet. Daher sinkt die verbrauchte Bandbreite mit größer werdendem minDistance Wert. Sowohl Konfiguration 1 als auch Konfiguration 2 haben einen geringen Bandbreitenverbrauch. Daher ist es nicht verwunderlich, dass der Wert von minDistance in diesen Konfigurationen der größte bzw. zweitgrößte Wert ist, den dieser Parameter annehmen kann. Je größer die Werte von minDistance sind, desto größer sind die Abstände zwischen zwei Knoten, an die Anfragen gesendet werden. Wird dazu der Parameter floodRadius so gewählt, dass er kleiner als die Hälfte von minDistance ist, gibt es Gebiete, die nicht nach ihrer aktuellen Verkehrslage befragt werden. Sind diese Gebiete zu groß, ist es möglich, dass ein in einem solchen nicht beachteten Gebiet vorhandener Stau nicht erkannt wird. Daher sollte sich ein großer Wert für minDistance negativ auf die relative Reisezeit auswirken. Da aber für beide Konfigurationen gilt, dass nur ein Stau mit einer Länge von 15km existiert, kann auch mit einem großen minDistance Wert dieser Stau noch erkannt werden. Um zu entscheiden, welche der beiden Konfigurationen die bessere ist, wurden die Ergebnisse dieser Konfigurationen sowohl bei geringeren Übertragungswahrscheinlichkeiten als auch bei einer größeren Anzahl an Verkehrssituationen betrachtet.

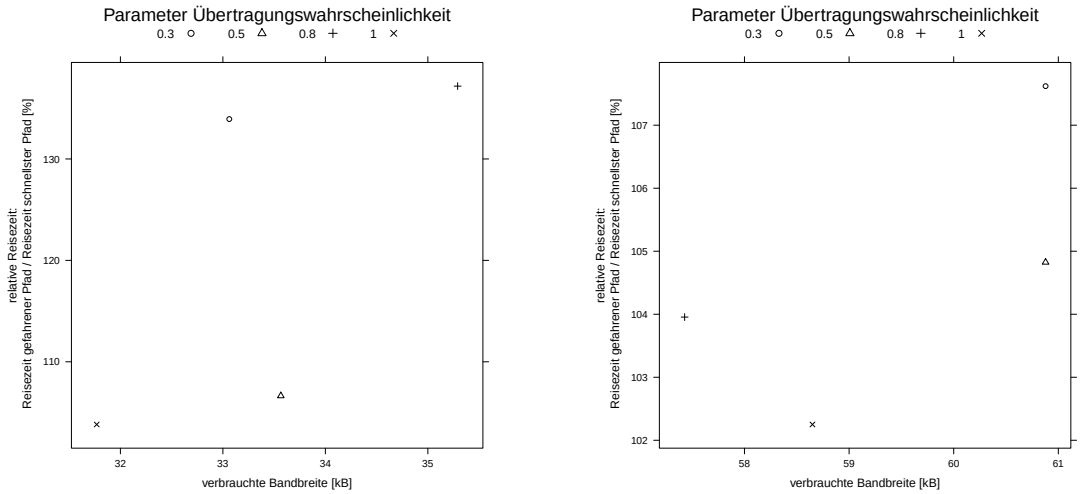

Abbildung 6.4: Vergleich der beiden Konfigurationen hinsichtlich des Verhaltens bei unterschiedlichen Übertragungswahrscheinlichkeiten. Links: Konfiguration 2. Rechts: Konfiguration 1.

In Abbildung 6.4 sind nun die Ergebnisse für beide Konfigurationen bei allen vier möglichen Übertragungswahrscheinlichkeiten zu sehen. Auf den Achsen werden wieder die Bandbreite und die relative Reisezeit dargestellt. Im linken Teil ist Konfiguration 2 und im rechten Teil ist Konfiguration 1 zu sehen. Es ist bereits an dem Wertebereich der Y-Achse zu erkennen, dass Konfiguration 2 deutlich schlechter mit nicht übertragenen Anfragen zurechtkommt, als Konfiguration 1. Dies kann mit dem Verhältnis der Staulänge von 15 zu der Länge des Gebiets, für das keine Anfragen gesendet werden, erklärt werden. Bei Konfiguration 1 hat das Gebiet, für das keine Anfragen gesendet werden, die Länge: $minDistance - 2 * floodradius = 16km - 2*2km = 12km$. Bei einer Staulänge von 15km ist es somit nicht möglich, dass der Stau komplett in einem Gebiet liegt, für welches keine Anfragen gesendet werden. Daher wird sich jedes Mal, wenn Anfragen gesendet werden, mindestens eine Anfrage

Tabelle 6.5: Werte der Konfigurationen für den ISF-Ansatz mit dem besten Trade-Off

	minDistance	**floodRadius**	**Reisezeit**	**Bandbreite**
Konfig 1	16km	1km	102.09%	108.31kB
Konfig 2	32km	4km	103.64%	64.4kB

auf eine Straße beziehen, auf der Stau herrscht. Somit fällt der Verlust einer solchen Anfrage nicht so stark ins Gewicht. Bei Konfiguration 2 hingegen hat das Gebiet ohne Anfrage die Länge: $minDistance - 2 * floodradius = 32km - 2 * 4km = 24km$. Bei dieser Konfiguration ist es somit möglich, dass der Stau komplett in dem Gebiet liegt, für das keine Anfragen gesendet werden. Somit fällt der Verlust einer Anfrage, welche sich auf eine Straße bezieht, auf der der Stau vorliegt, stärker ins Gewicht und die Konfiguration ist anfälliger für die erfolglose Übertragung von Anfragen.

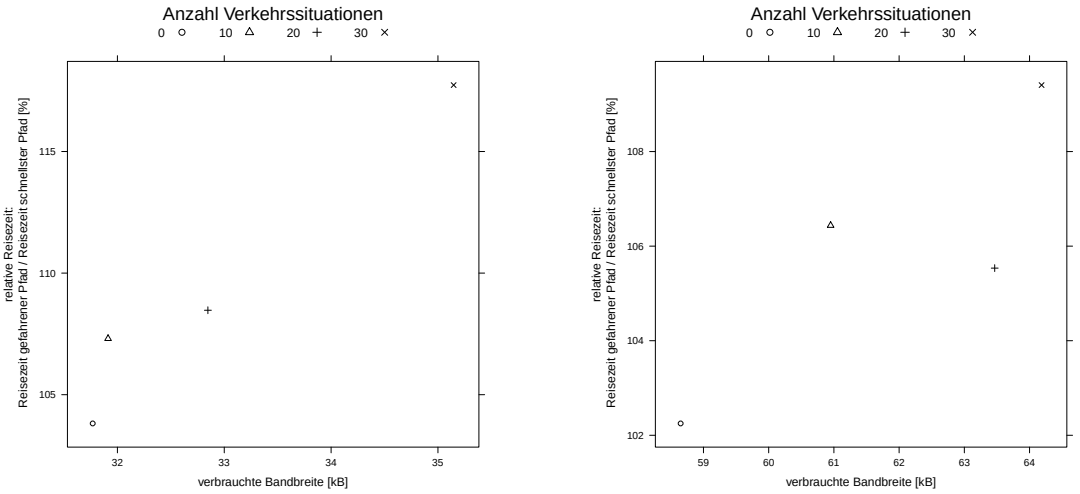

Abbildung 6.5: Vergleich der beiden Konfigurationen hinsichtlich des Verhaltens bei unterschiedlicher Anzahl an Verkehrssituationen. Links: Konfiguration 2. Rechts: Konfiguration 1.

Des Weiteren wurden die Ergebnisse beider Konfigurationen verglichen, wenn sich die Anzahl an Verkehrssituationen ändert. Abbildung 6.5 zeigt links wieder Konfiguration 2 und rechts wieder Konfiguration 1. Auf den Achsen sind wieder die Reisezeit und die verbrauchte Bandbreite dargestellt. Für diese Grafik wird eine Übertragungswahrscheinlichkeit von 100% gewählt. Auch hier ist zu erkennen, dass Konfiguration 1 hinsichtlich der Reisezeit mit vielen Verkehrssituationen besser zurecht kommt, als Konfiguration 2. Dies liegt wieder an der Länge der einzelnen Verkehrssituationen und der Länge des Gebietes, für das keine Anfrage gesendet wird. Da die Länge, die eine Verkehrssituation in dieser Arbeit haben kann, zwischen 1km und 15km liegt, wird die durchschnittliche Länge deutlich geringer sein als 15km. Somit ist es möglich, dass bei Konfiguration 2 nicht nur eine Verkehrssituation, sondern gleich mehrere Verkehrssituationen in dem Gebiet liegen können, für welches keine Anfrage gesendet wird. Durch die Vergleiche der beiden Konfigurationen hinsichtlich der Anfälligkeit gegenüber nicht übertragener Anfragen und des Einflusses der Anzahl an verschiedenen Verkehrssituationen, konnte gezeigt werden, dass Konfiguration 1 die besser Wahl ist und daher für OverDrive und das zentralisierte Geocast Protokoll zum Einsatz kommt.

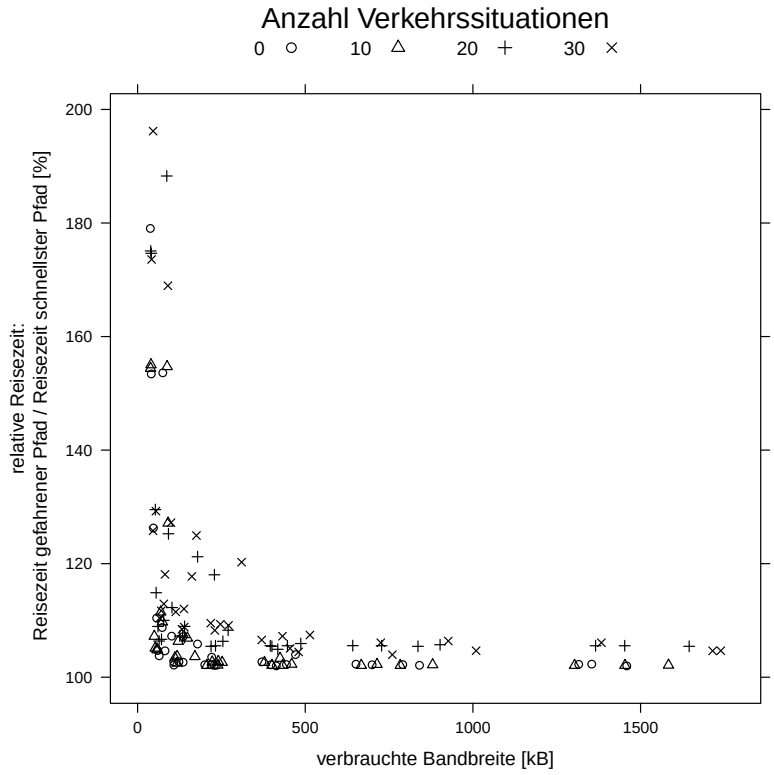

Abbildung 6.6: Ergebnisse aller Simulationen für den ISF-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Die X-Achse zeigt die verbrauchte Bandbreite in kB. Die Y-Achse zeigt die relative Reisezeit in Prozent

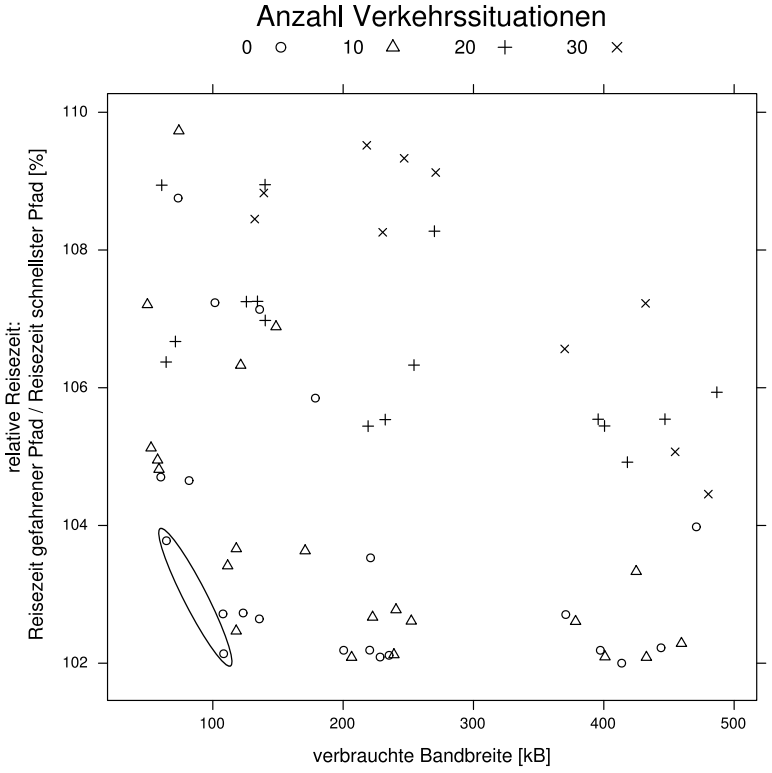

Abbildung 6.7: Ergebnisse aller Simulationen für den ISF-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Die X-Achse zeigt die verbrauchte Bandbreite in kB. Die Y-Achse zeigt die relative Reisezeit in Prozent. Es werden nur die Ergebnisse angezeigt, welche einen Bandbreitenverbrauch von weniger als 500kB und eine Reisezeit von weniger als 110% bewirken.

ISF-Ansatz

Als nächstes wird nun die Konfiguration für den ISF-Ansatz ermittelt, welche das beste Trade-Off zwischen Bandbreitenverbrauch und relativer Reisezeit hat. Abbildung 6.6 zeigt die Ergebnisse für alle Konfigurationen. Auf der X-Achse ist die verbrauchte Bandbreite dargestellt und auf der Y-Achse die relative Reisezeit des schnellsten Pfads im Vergleich zu dem gefahrenen Pfad. Es wurde wieder eine Übertragungswahrscheinlichkeit von 100% gewählt. Es ist zu erkennen, dass die Konfigurationen entweder einen geringen Bandbreitenverbrauch bei sehr unterschiedlichen Reisezeiten haben oder eine geringe Reisezeit bewirken, dabei aber sehr unterschiedliche Werte für den Verbrauch an Bandbreite haben. Um die Konfiguration zu finden, welche den besten Trade-Off bietet, wurde erneut ein näherer Blick auf die Ergebnisse aller Konfigurationen geworfen. Abbildung 6.7 zeigt die Konfigurationen, welche einen Bandbreitenverbrauch von weniger als 500kB und eine relative Reisezeit von weniger als 10% haben. Wieder markiert das schwarze Oval die beiden Konfigurationen, welche näher untersucht wurden. Tabelle 6.5 zeigt die Parameterwerte dieser beiden Konfigurationen und deren Ergebnisse für die verbrauchte Bandbreite sowie die relative Reisezeit. Es kommt, wie schon bei dem SF-Ansatz, der größte bzw. der zweitgrößte Wert für minDistance zum Einsatz. Dies liegt erneut daran, dass mit größeren minDistance Werten weniger Anfragen verschickt und somit weniger Bandbreite verbraucht wird. Da erneut nur ein Stau in diesen Simulationen existierte, war der Routenplaner trotz großen minDistance Wertes in der Lage, den Stau zu detektieren und zu umfahren. Des Weiten sind die Werte für floodRadius wieder geringer als die Hälfte von minDistance, was dazu führt, dass es auch hier Gebiete gibt, für die keine Anfragen versendet werden. Auch dies lässt sich wieder auf die Funktionsweise des Routenplaners zurückführen. Die durchschnittliche Geschwindigkeit, welche für ein Gebiet anhand der Anfragen errechnet wird, ist in der Regel langsamer als tatsächlich möglich. So kann es sein, dass ein Pfad, auf dem kein Stau vorliegt, schlechter bewertet wird als er eigentlich ist und ein Pfad gewählt wird, der eigentlich schlechter ist. Es werden nun beide Konfigurationen hinsichtlich ihrer Anfälligkeit für nicht übertragene Anfragen verglichen.

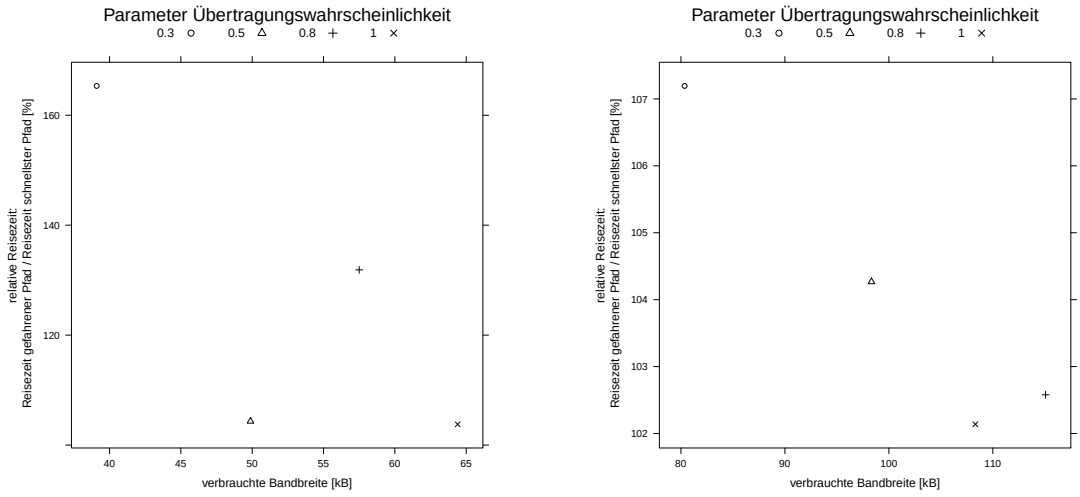

Abbildung 6.8: Vergleich der beiden Konfigurationen hinsichtlich des Verhaltens bei unterschiedlichen Übertragungswahrscheinlichkeiten. Links: Konfiguration 2. Rechts: Konfiguration 1.

Abbildung 6.8 zeigt links die Ergebnisse für Konfiguration 2 und rechts die Ergebnisse für Konfiguration 1. Auf den Achsen werden wieder die Reisezeit und die verbrauchte Bandbreite dargestellt. Auch bei dem ISF-Ansatz ist die Konfiguration 2 mit einem minDistance Wert von 32km deutlich anfälliger gegenüber nicht erfolgreich übertragenen Anfragen als Konfiguration 1 mit einem minDistance Wert von 16km. Auch hier kann dies mit der Länge des Gebiets, für das keine Anfragen versendet werden, im Vergleich zur Länge des einzigen Staus erklärt werden. Wie schon beim SF-Ansatz ist es bei Konfiguration 1 nicht möglich, dass der gesamte Stau von 15km Länge in einem Gebiet liegt, für welches keine Anfrage gesendet wird, da dieses Gebiet eine Länge von $minDistance - 2 * floodradius = 16km - 2 * 1km = 14km$ hat. Für Konfiguration 2 hingegen ist dies möglich, da die Länge dieses Gebiets $minDistance - 2 * floodradius = 32km - 2 * 4km = 24km$ beträgt. Aus diesem Grund ist Konfiguration 2 anfälliger für nicht übertragene Anfragen als Konfiguration 1.

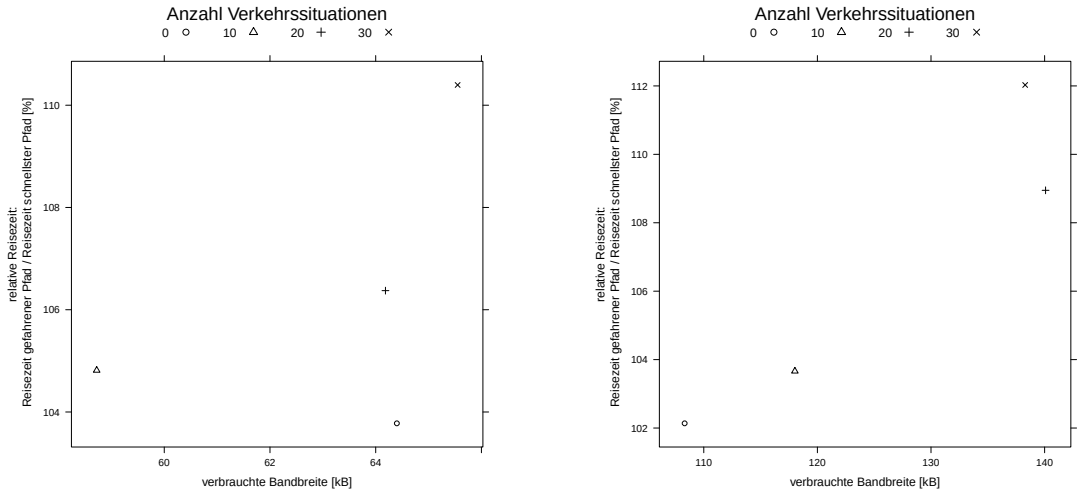

Abbildung 6.9: Vergleich der beiden Konfigurationen hinsichtlich des Verhaltens bei unterschiedlicher Anzahl an Verkehrssituationen. Links: Konfiguration 2. Rechts: Konfiguration 1.

Des Weiteren werden nun auch die Konfigurationen für den ISF-Ansatz hinsichtlich ihrer Ergebnisse bei unterschiedlichen Mengen an Verkehrssituationen verglichen. In Abbildung 6.9 ist links wieder Konfiguration 2 und recht Konfiguration 1 zu sehen. Die Achsen stellen erneut die relative Reisezeit und die verbrauchte Bandbreite dar. Es ist zu erkennen, dass Konfiguration 2 für eine Anzahl von 30 bzw. . 20 Verkehrssituationen die geringere Reisezeit bewirkt und somit besser abschneidet, als Konfiguration 1. Dies kann folgendermaßen erklärt werden: Wird bei Konfiguration 1 eine Verkehrssituation erkannt, so wird die Länge dieser Verkehrssituation auf $2 * floodRadius = 2km$ geschätzt. Da die Länge einer Verkehrssituation aber zwischen 1km und 15km liegt, wird die durchschnittliche Länge einer Verkehrssituation bei ungefähr 7km liegen. Konfiguration 1 unterschätzt somit die erkannten Verkehrssituationen. Bei Konfiguration 2 hingegen wird ein erkannter Stau auf die Länge 8km geschätzt, was deutlich näher an dem durchschnittlichen Wert von 7km liegt. Existieren nun eine Vielzahl an Verkehrssituationen, so werden bei Konfiguration 1 viele Verkehrssituation deutlich unterschätzt, bei Konfiguration 2 hingegen nur leicht überschätzt. Aus diesem Grund ist Konfiguration 2 besser hinsichtlich der

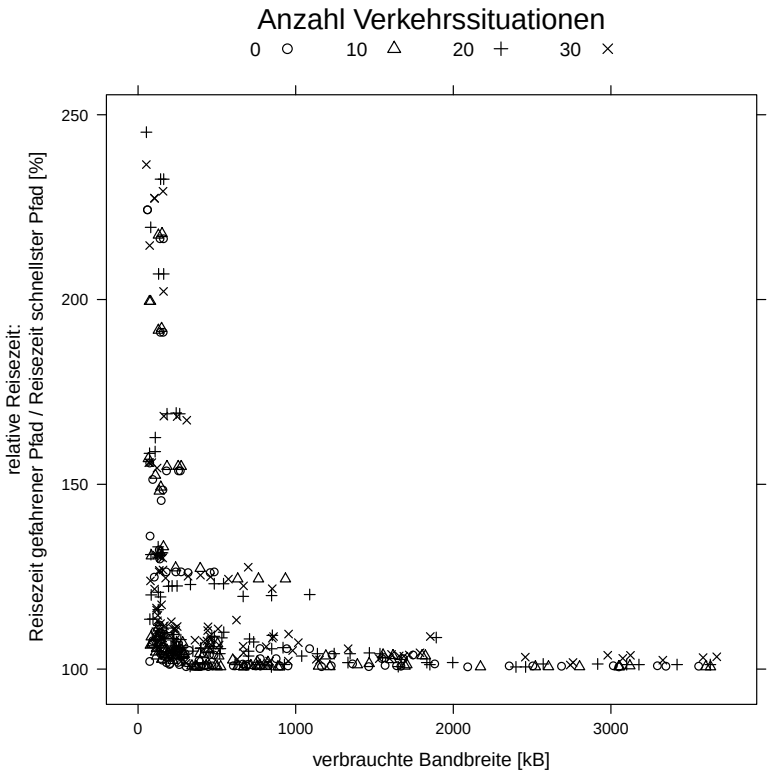

Abbildung 6.10: Ergebnisse aller Simulationen für den AG-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Die X-Achse zeigt die verbrauchte Bandbreite in kB. Die Y-Achse zeigt die relative Reisezeit in Prozent

Tabelle 6.6: Werte der Konfigurationen für den AG-Ansatz mit dem besten Trade-Off.

	minDistance	**floodRadius**	**altGraphSize**	**Reisezeit**	**Bandbreite**
Konfig 1	16km	8km	5	101.25%	199.45kB
Konfig 2	32km	8km	3	102.03%	73.74kB

Reisezeit für 20 bzw. 30 Verkehrssituation als Konfiguration 1. Bei weniger Verkehrssituationen ist das Erkennen einer Verkehrssituation wichtiger als das Abschätzen der Länge dieser Verkehrssituation. Aus diesem Grund ist Konfiguration 1 hinsichtlich der Reisezeit besser als Konfiguration 2 bei 0 bzw. 10 Verkehrssituation. Da Konfiguration 1 nicht so anfällig hinsichtlich der Übertragungswahrscheinlichkeit ist, wurde diese als die bessere der beiden Konfigurationen gewertet und kommt für OverDrive und das zentralisierte Geocast Protokoll zum Einsatz.

AG-Ansatz

Nun wird die Konfiguration ermittelt, bei welcher der AG-Ansatz den besten Trade-Off zwischen Reisezeit und Bandbreitenverbrauch erzielt. Abbildung 6.10 zeigt alle Ergebnisse der unterschiedlichen Konfigurationen bei einer Übertragungswahrscheinlichkeit von 100%. Auf den Achsen sind wieder die Reisezeit und die verbrauchte Bandbreite zu sehen. Auch beim AG-Ansatz zeigen die Ergebnisse, dass die Konfigurationen in der Regel entweder eine geringe Reisezeit mit unterschiedlichem Bandbreitenverbrauch oder einen geringen Bandbreitenverbrauch mit unterschiedlichen Reisezeiten bewirken. Um auch hier die Konfiguration zu finden, welche den

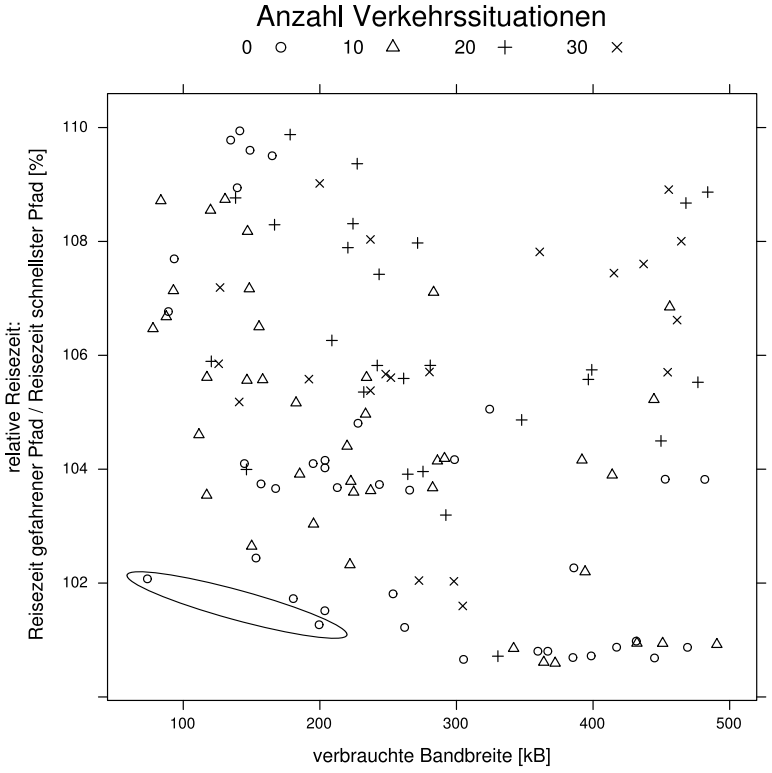

Abbildung 6.11: Ergebnisse aller Simulationen für den AG-Ansatz mit einer Übertragungswahrscheinlichkeit von 100%. Die X-Achse zeigt die verbrauchte Bandbreite in kB. Die Y-Achse zeigt die relative Reisezeit in Prozent. Es werden nur die Ergebnisse angezeigt, welche einen Bandbreitenverbrauch von weniger als 500kB und eine Reisezeit von weniger als 110% bewirken.

besten Trade-Off bietet, wird ein näherer Blick auf die Ergebnisse aller Konfigurationen geworfen. Abbildung 6.11 zeigt die Ergebnisse der Konfigurationen, welche einen Bandbreitenverbrauch von weniger als 500kB und eine relative Reisezeit von weniger als 10% haben. Auch hier markiert das schwarze Oval die beiden Konfigurationen, welche näher betrachtet werden. In Tabelle 6.6 sind die Werte der verschieden Parameter für beide Konfigurationen dargestellt. Auch für den AG-Ansatz kommt der größte bzw. der zweitgrößte Wert, welchen der Parameter minDistance annehmen kann, zum Einsatz. Dies kann erneut zum Einen damit erklärt werden, dass nur ein Stau der Länge 15km existiert und dieser mit so großen Werten für minDistance noch erkannt werden kann und zum Anderen, dass mit großen minDistance Werten weniger Anfragen versendet werden und daher weniger Bandbreite verbraucht wird. Es ist außerdem zu sehen, dass bei dem AG-Ansatz größere Werte für den Parameter floodRadius zum Einsatz kommen, als es bei dem SF- bzw. ISF-Ansatz der Fall ist. Dies kann folgendermaßen erklärt werden: In Kapitel 4.3.5 wurde beschrieben, dass Anfragen erst dann versendet werden, wenn entweder eine Kreuzung im Alternativgraph erreicht wird oder eine Kreuzung im Straßengraph erreicht wird und sich das Fahrzeug auf einem Pfad befindet, welcher nicht Teil des Alternativgraph ist. Es kommt somit seltener vor, dass das Senden von Anfragen initialisiert wird. Durch die größeren zeitlichen Abstände zwischen dem Senden von Anfragen muss ein Stau so früh wie möglich erkannt und richtig eingeschätzt werden, da es möglich ist, dass das nächste Senden von Anfragen erst wieder initialisiert wird, wenn der Stau bereits erreicht ist. Durch einen floodRadius von 8km wird ein erkannter Stau direkt auf eine Länge von 16km geschätzt. Da bei den hier betrachteten Konfigurationen nur ein Stau mit einer Länge von 15km existiert, wird dieser Stau von beiden Konfigurationen leicht überschätzt. Wäre der Wert von floodRadius aber kleiner, so würde dieser Stau deutlich unterschätzt werden. Aus diesem Grund wird mit einem floodRadius Wert von 8km der beste Trade-Off erzielt. Es wird nun untersucht, wie sich beide Konfigurationen verhalten, wenn die Übertragungswahrscheinlichkeit bzw. die Anzahl an Verkehrssituationen variiert.

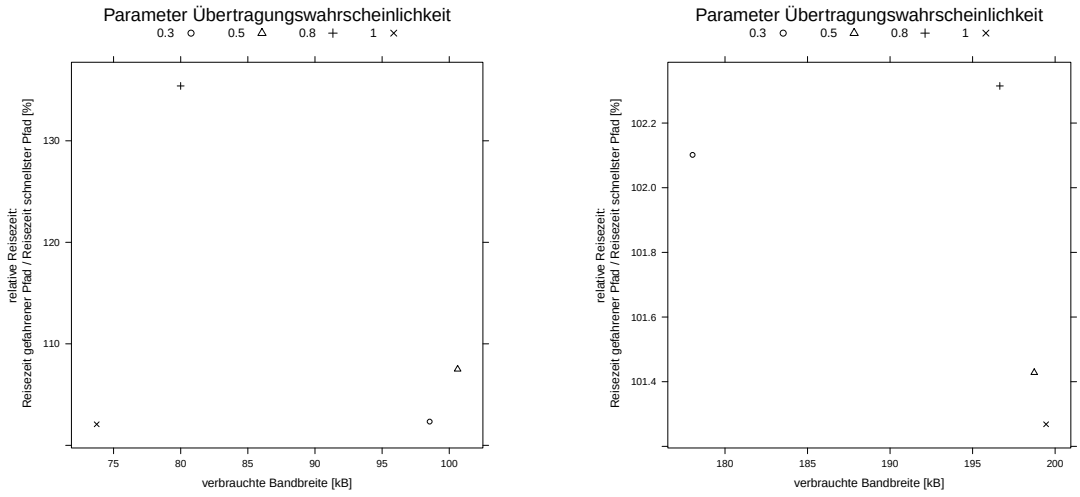

Abbildung 6.12: Vergleich der beiden Konfigurationen hinsichtlich des Verhaltens bei unterschiedlichen Übertragungswahrscheinlichkeiten. Links: Konfiguration 2. Recht: Konfiguration 1.

Abbildung 6.12 zeigt die Ergebnisse für beide Konfigurationen bei den unterschiedlichen Übertragungswahrscheinlichkeiten. Die Achsen stellen wieder die Reisezeit bzw. die verbrauchte Bandbreite dar. Es ist erneut zu erkennen, dass Konfiguration 1 weniger anfällig für nicht übertragene Anfragen ist als Konfiguration 2. Die kann erneut mit der Länge des Gebietes erklärt werden, für welches keine Anfrage gesendet wird. Bei Konfiguration 1 gibt es kein solches Gebiet, da der Wert von floodRadius genau der Hälfte des Wertes von minDistance entspricht. Bei Konfiguration 2 gibt es aber so ein Gebiet und es hat die Länge: $minDistance - 2 * floodradius = 32km - 2 * 8km = 16km$. Es ist somit bei Konfiguration 2 möglich, dass der Stau vollständig in einem solchen Gebiet liegt. Aus diesem Grund ist es wichtig für Konfiguration 2, dass jede Anfrage, welche sich auf ein Gebiet bezieht, in dem ein Stau vorkommt, erfolgreich übertragen wird. Daher ist Konfiguration 2 deutlich anfälliger für nicht übertragene Anfragen als Konfiguration 1.

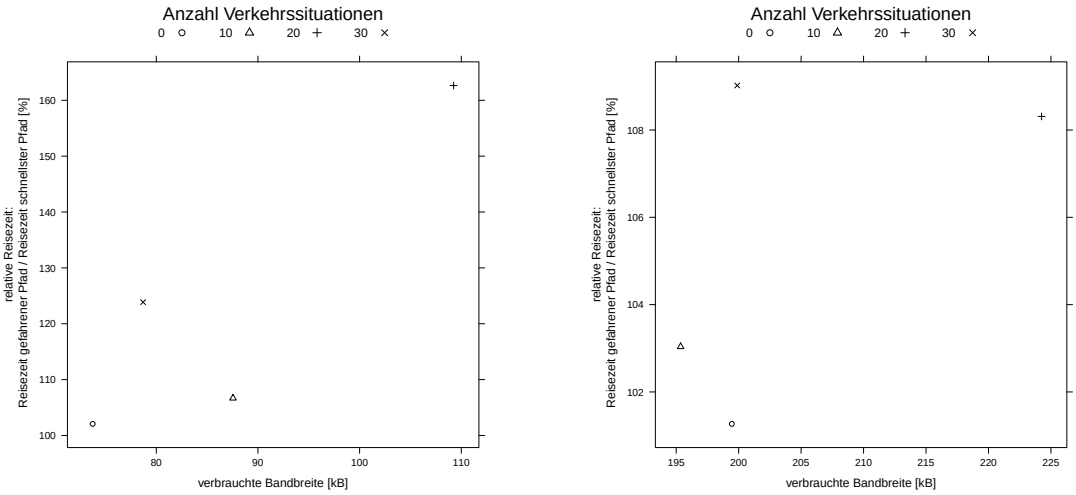

Abbildung 6.13: Vergleich der beiden Konfigurationen hinsichtlich des Verhaltens bei unterschiedlicher Anzahl an Verkehrssituationen. Links: Konfiguration 2. Recht: Konfiguration 1.

Es werden nun die Ergebnisse beider Konfigurationen hinsichtlich unterschiedlicher Mengen an Verkehrssituationen verglichen. Abbildung 6.13 zeigt rechts die Ergebnisse von Konfiguration 1 und links die Ergebnisse von Konfiguration 2. Auf den Achsen werden wieder die relative Reisezeit und die verbrauchte Bandbreite dargestellt. Es ist zu erkennen, dass Konfiguration 1 deutlich besser mit vielen Verkehrssituationen zurechtkommt als Konfiguration 2. Die kann erneut damit erklärt werden, dass es bei Konfiguration 1 kein Gebiet gibt, für welches keine Anfrage gesendet wird. Wird bei Konfiguration 1 ein Pfad, auf dem eine Verkehrssituationen vorliegt, überprüft, so wird diese Verkehrssituation auch erkannt. Die erkannte Verkehrssituation wird aber auf Grund des großen Wertes für floodRadius im Schnitt deutlich überschätzt. Bei Konfiguration 2 ist es jedoch nicht gegeben, dass jede Verkehrssituation erkannt wird, da es Gebiete der Länge 16km gibt, für welche keine Anfragen gesendet werden. Der Vergleich der beiden Konfigurationen hinsichtlich sich ändernder Übertragungswahrscheinlichkeit und Anzahl an Verkehrssituationen ergibt, dass auch bei dem AG-Ansatz Konfiguration 1 die bessere Wahl ist. So kommen die Werte von Konfiguration 1 für die verschiedenen Parameter bei OverDrive und dem zentralisierten Geocast Protokoll zum Einsatz.

Tabelle 6.7: geeignete Werte für den Einsatz bei OverDrive und dem zentralisierten Geocast Protokoll

Parameter	SF-Ansatz	ISF-Ansatz	AG-Ansatz
minDistance	16km	16km	16km
floodRadius	2km	1km	8km
altGraphSize	-	-	5

Aus der Analyse des Einflusses der Parameter minDistance, floodRadius und altGraphSize auf die Bandbreite und die Reisezeit ergeben sich die Werte für diese Parameter wie sie in Tabelle 6.7 dargestellt sind. Diese Werte werden für die Simulationen mit OverDrive und dem zentralisierten Geocast Protokoll verwendet.

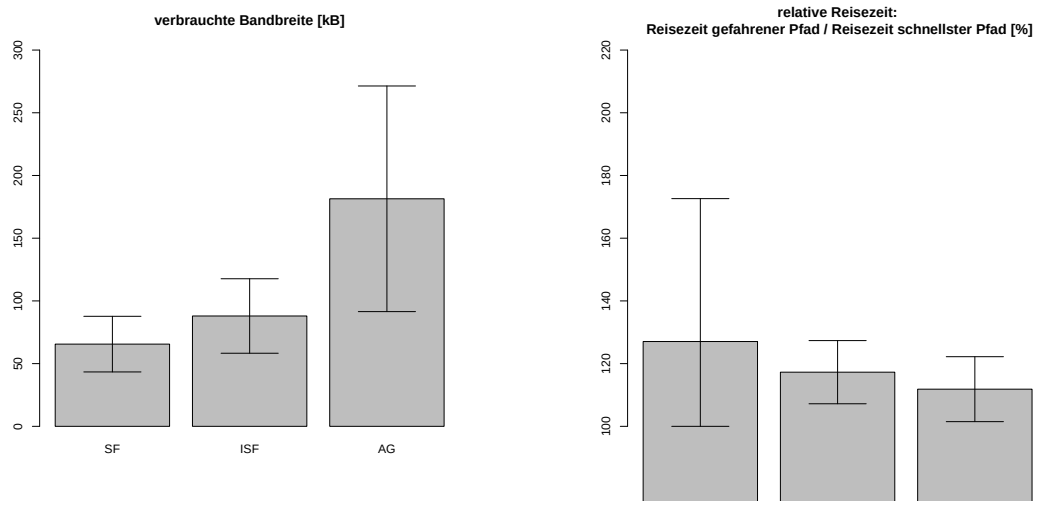

Abbildung 6.14: Vergleich der drei Ansätze. Rechts hinsichtlich der relativen Reisezeit und links hinsichtlich der verbrauchten Bandbreite

Vergleich der drei Ansätze SF, ISF und AG

Nachdem der Wertebereich der Parameter eingeschränkt wurde, werden nun die drei Ansätze einander gegenübergestellt. Vergleicht man die Tabellen 6.4, 6.5 und 6.6, ist zu erkennen, dass alle drei Ansätze bei optimaler Parameterkonfiguration ungefähr dieselben Reisezeiten bewirken. Um zu entscheiden, welcher der drei Ansätze nun aber die besten Resultate liefert, wurde Wert darauf gelegt, wie die Ansätze auf extremen Verhältnisse reagieren. Aus diesem Grund wurde eine Übertragungswahrscheinlichkeit von 30% und eine Anzahl an Verkehrssimulationen von 30 gewählt. Abbildung 6.14 zeigt links die Ergebnisse der drei Ansätze hinsichtlich der verbrauchten Bandbreite und rechts die Ergebnisse hinsichtlich der relativen Reisezeit. Es ist zu erkennen, dass der SF-Ansatz zwar am wenigsten Bandbreite verbraucht, dafür aber auch mehr Reisezeit benötigt als die anderen beiden Ansätze. Der ISF benötigt ungefähr doppelt so viel Bandbreite wie der SF-Ansatz, benötigt aber ungefähr 4% weniger Reisezeit. Der AG-Ansatz benötigt ungefähr vier Mal so viel Bandbreite wie der SF-Ansatz, braucht aber ungefähr 8% weniger Reisezeit. Anhand dieser beiden Grafiken erreicht der ISF-Ansatz den besten Trade-Off zwischen Reisezeit und Bandbreitenverbrauch. Die durchschnittliche Reisezeit eines Fahrzeuges betrug in dieser Arbeit 160 Minuten. Ein Unterschied in der Reisezeit vom ISF-Ansatz zum AG-Ansatz beträgt ungefähr 4%. Dies bedeutet in diesem Fall eine Zeitersparnis von

ungefähr 7 Minuten. In dieser Zeit benötigt der AG-Ansatz mit 200kB nur 100kB mehr als der ISF-Ansatz. Würde ein Fahrzeug, z. B. ein LKW eines Logistikunternehmens, 24 Stunden am Tag unterwegs sein und die ganzen Zeit den Routenplaner mit AG-Ansatz verwenden, so würde der Routenplaner für dieser LKW einen Bandbreitenverbrauch von ungefähr 54 Megabyte *pro Monat* erzeugen. In der heutigen Zeit, in der beim streamen Full-HD-Filme von mehreren Gigabyte in wenigen Stunden übertragen werden, fallen diese 54MB kaum ins Gewicht. Solange die Bandbreite nicht gezwungenermaßen so gering wie möglich sein muss, wird mit dem AG-Ansatz die beste Reisezeit erreicht und es wird dafür nur eine relativ geringe Menge an Bandbreite verbraucht.

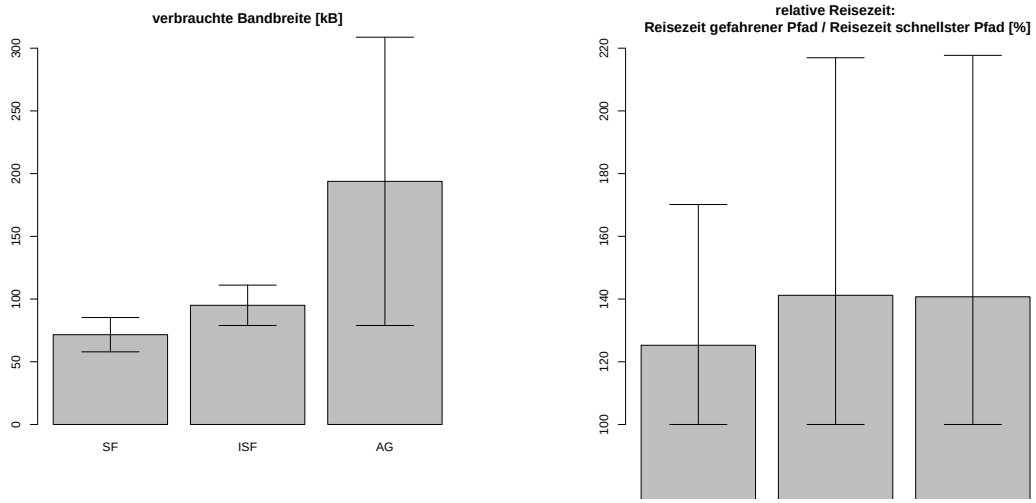

Abbildung 6.15: Vergleich der drei Ansätze. Rechts hinsichtlich der relativen Reisezeit und links hinsichtlich der verbrauchten Bandbreite

6.2.2 Evaluierung des zentralisierten Geocast Protokolls

In diesem Abschnitt werden nun die Ergebnisse für das zentralisierte Geocast Protokoll präsentiert. Abbildung 6.16 zeigt die Ergebnisse aller drei Ansätze für die unterschiedlichen Mengen an Fahrzeugen die Simuliert wurden. Links ist der SF-Ansatz, in der Mitte der ISF-Ansatz und rechte der AG-Ansatz abgebildet. Es ist deutlich zu erkennen, das für den SF- und den ISF-Ansatz kleinere Reisezeiten erzielt werden, wenn mehr Fahrzeuge simuliert werden. Nur beim AG-Ansatz werden mit 20000 Fahrzeugen größere Reisezeiten benötigt als mit 10000. Dies kann folgendermaßen erklärt werden: Für den AG-Ansatz wurde eine Wert von 8km für den Parameter floodRadius gewählt. Wird mit dieser Parametereinstellung ein Stau erkannt, wird dessen Länge auf 16km geschätzt. Dies ist aber deutlich länger als ein Stau durchschnittlich ist. Dies führt somit dazu, dass die Länge eines erkannten Staus überschätzt wird. Durch die größere Anzahl an Fahrzeugen steigt die Wahrscheinlichkeit auf eine Anfrage eine Antwort zu erhalten. Somit steigt auch die Wahrscheinlichkeit einen Stau zu erkennen und diesen zu überschätzen. Für die Simulationen mit 20000 Fahrzeugen werden aber auch für den AG-Ansatz kleiner Reisezeiten erzielt als mit 5000 Fahrzeugen. Daraus lässt sich schließen, dass eine größere Anzahl an Fahrzeugen sich positiv auf die Reisezeit auswirkt. Abbildung 6.15 zeigt die Ergebnisse der drei Ansätze hinsichtlich Bandbreitenverbrauch und Reisezeit. Bei diesen Simulationen kamen 5000 Fahrzeuge zum Einsatz und es wurde eine Übertragungswahrscheinlichkeit von 100% gewählt. Darüber hinaus gab

es nur einen konstanten Stau während der Simulation. Rechts sind die Ergebnisse des Bandbreitenverbrauchs und links die der Reisezeit dargestellt. Es ist zu erkennen, dass auch beim zentralisierten Geocast Protokoll der AG-Ansatz am meisten Bandbreite verbraucht. Jedoch wird mit diesem Ansatz bei diesem Protokoll nicht die kleinste Reisezeit erzielt. Die kleinste Reisezeit wird mit dem SF-Ansatz erzielt. Wie in Kapitel 4.3.4 beschrieben wurde, berechnet der SF-Ansatz, nachdem ein Stau erkannt wurde, den schnellsten Pfad anhand der Informationen welche erfragt wurden. Diesem Pfad folgt das Fahrzeug ohne, dass dieser Pfad überprüft wird. In einem Szenario in dem nur ein Stau existiert, wird dieser entweder der schnellste Pfad zum Ziel sein, oder diesem schnellsten Pfad sehr ähnlich sein. Der ISF-Ansatz überprüft jeden Pfad bevor diesem Pfad gefolgt wird. Wie bereits beschrieben wurde, werden die Kosten für einen Pfad in der Regel überschätzt. Dies kann dazu führen, dass ein potentiell guter Pfad verworfen wird und einem eigentlich schlechteren Pfad gefolgt wird. Dies verhält sich beim AG-Ansatz ähnlich. Deshalb erzielt in diesem Szenario der SF-Ansatz die kleinste Reisezeit.

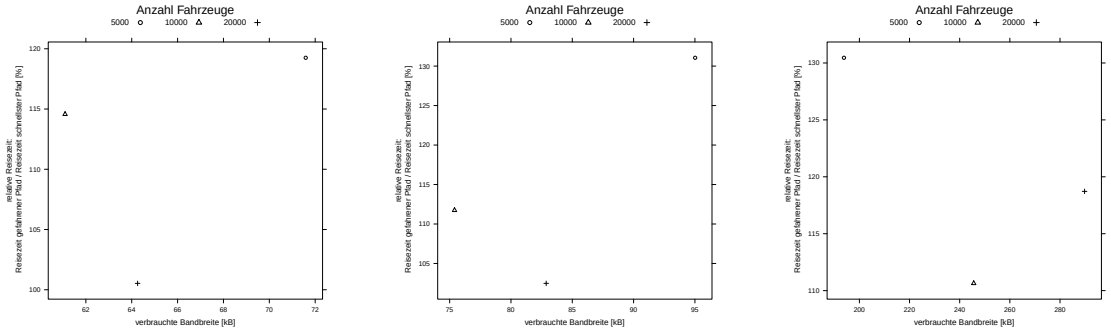

Abbildung 6.16: Verhalten der drei Ansätze bei unterschiedlicher Anzahl an Fahrzeugen. Links = SF; Mitte = ISF; Rechts = AG

6.2.3 Evaluierung von OverDrive

Im Folgenden werden nun die Ergebnisse für OverDrive präsentiert. Für die Simulationen mit OverDrive kamen 5000 Fahrzeuge zum Einsatz. Die Ergebnisse können daher direkt mit den Ergebnissen des zentralisierten Geocast Protokoll, bei denen auch 5000 Fahrzeuge simuliert wurden, verglichen werden. Abbildung 6.17 zeigt die Ergebnisse von OverDrive bei nur einem konstanten Stau. Links die Ergebnisse hinsichtlich der Bandbreite und rechts hinsichtlich der Reisezeit. Auch bei OverDrive benötigt der AG-Ansatz die meiste Bandbreite. Im Gegensatz zu dem zentralisierten Geocast Protokoll erzielen der SF- und der ISF-Ansatz nahezu die gleichen Reisezeiten. Im direkten Vergleich von Abbildung 6.17 mit Abbildung 6.15 ist zu erkennen, dass OverDrive im Durchschnitt ähnliche Ergebnisse erzielt wie das zentralisierte Geocast Protokoll bei einer Übertragungswahrscheinlichkeit von 100%. Daraus kann geschlossen werden, dass OverDrive als dezentrales Protokoll keine schlechteren Ergebnisse liefert als ein zentralisierten Protokoll.

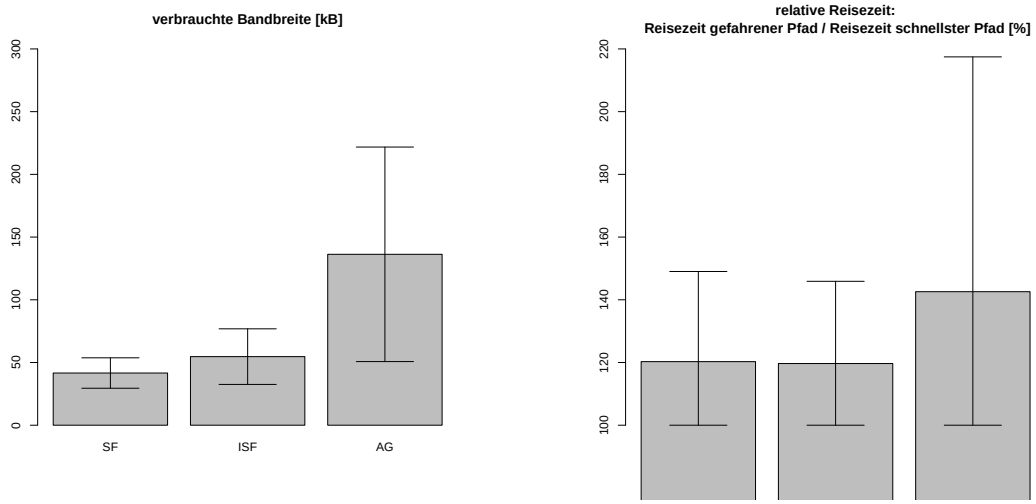

Abbildung 6.17: Vergleich der drei Ansätze. Rechts hinsichtlich der relativen Reisezeit und links hinsichtlich der verbrauchten Bandbreite

6.2.4 Zusammenfassung der Ergebnisse

In diesem Kapitel wurden zuerst die Wertebereiche einiger Parameter eingeschränkt, um die Anzahl an unterschiedlichen Simulationen zu verringern. Dies war erforderlich, da eine große Menge an unterschiedlichen Simulationen mit OverDrive und dem zentralisierten Geocast Protokoll nicht realisierbar war. Die Einschränkung erfolgte aufgrund der Ergebnisse des probabilistischen Geocast Modells. Mit den so ermittelten Werten für die verschiedenen Parameter wurden Simulationen mit OverDrive und dem zentralisierten Geocast Protokoll durchgeführt. Anschließend wurden die Ergebnisse dieser Simulationen genutzt, um die drei Ansätze aus Kapitel 4.3 zu bewerten. Es konnte dargestellt werden, dass der Alternativ-Graph-Ansatz der effektivste Ansatz ist sofern es genügend Fahrzeuge gibt, die den Routenplaner mit Informationen versorgen können. Ist dies der Fall benötigt der AG-Ansatz zwar mehr Bandbreite es wird aber auch eine kleinere Reisezeit erzielt. Anhand der Ergebnisse des zentralisierten Geocast Protokolls wurde gezeigt, dass sich eine größere Anzahl an Fahrzeugen positiv auf die Reisezeit auswirkt. Daraus lässt sich ableiten, dass die Anzahl an Fahrzeugen, die hier simuliert wurden, nicht ausreicht, um alle relevanten Staus auf dem Straßennetz zu erkennen. Zum Schluss wurden die Ergebnisse von OverDrive präsentiert, und sie wurden mit den Ergebnissen des zentralisierten Geocast Protokolls verglichen. Es konnte gezeigt werden, dass OverDrive nahezu die gleichen Resultate liefert wie das zentralisierte Geocast Protokoll.

6.3 Zusammenfassung

In diesem Kapitel wurde zu Beginn der Simulationsaufbau erläutert. Es wurde dargestellt, auf welchem Straßennetz die Simulationen durchgeführt wurden. Darüber hinaus wurden die Wertebereiche für die verwendeten Parameter definiert und es wurde gezeigt, dass diese eingeschränkt werden müssen. Die Einschränkung erfolgte mit Hilfe der Ergebnisse des probabilistischen Geocast Modells. Anschließend wurden die drei Ansätze Straight-Forward, iterativer Straight-Forward und Alternativ-Graph anhand der Metriken relative Reisezeit und Bandbreite evaluiert. Es wurde

ersichtlich, dass der Alternativ-Graph-Ansatz am effektivsten arbeitet sofern genügend Informationen vorhanden sind. Anschließend wurden die Ergebnisse für jedes der drei Protokolle OverDrive, zentralisiertes und probabilistisches Geocast präsentiert und miteinander verglichen.

7. Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Implementierung und Evaluierung eines adaptiven Routenplaners, der mit Hilfe eines Geocast Protokolls in der Lage ist, Staus zu erkennen und gegebenenfalls zu umfahren. Es folgte eine Analyse der Problemstellung, bei der Anforderungen an den Routenplaner gestellt wurden. Während dieser Analyse wurde registriert, dass das Geocast Protokoll OverDrive [6] sehr viel Overhead erzeugt und bei einer Fahrzeugmenge von 5000 Fahrzeugen eine Simulationszeit von über 24 Stunden benötigt. Für die Entwicklungsphase des Routenplaners war diese Simulationszeit viel zu lange und so wurde ein zentralisiertes Geocast Protokoll entwickelt, welches deutlich kleinere Simulationszeiten hatte. Damit der Routenplaner in der Lage ist, einen Stau zu erkennen, müssen bei beiden Protokollen viele Fahrzeuge simuliert werden. Dies führt aber zu längeren Simulationszeiten, welche für die Entwicklungsphase des Routenplaners wiederum zu lange waren. Deshalb wurde das probabilistische Geocast Modell entwickelt, dessen Simulationszeit so gering ist, dass es für die Entwicklungsphase des Routenplaners geeignet war. Es wurden anschließend die drei Ansätze *Straight-Forward*, *iterativer Straight-Forward* und *Alternativ-Graph* entwickelt, die es dem Routenplaner ermöglichen, gezielt Anfragen an Fahrzeuge zu versenden und mit den Informationen, die von diesen Fahrzeugen erhalten wurden, den Pfad zum Ziel zu berechnen, der die kürzeste Reisezeit hat. Bei dem Straight-Forward wird nur der aktuell gefahrene Pfad auf einen Stau hin untersucht. Wird ein Stau erkannt, so wird ein Pfad mit kürzerer Reisezeit berechnet und es wird diesem Pfad gefolgt. Der Nachteil dieses Ansatzes liegt darin, dass, wenn ein Stau erkannt wurde und einem anderen Pfad gefolgt wird, dieser andere Pfad noch nicht auf eine Stausituation hin untersucht wurde. Es ist somit möglich, dass das Fahrzeug, um einen Stau zu umfahren, in einen anderen Stau hinein fährt. Dieser Nachteil wird mit dem iterativen Straight-Forward Ansatz behoben. Bei diesem Ansatz wird jeder Pfad auf eine Stausituation hin untersucht und nur dem Pfad gefolgt, welcher auch nach der Untersuchung noch die kürzeste Reisezeit hat. Dieser Ansatz hat den Nachteil, dass potentiell viele Nachrichten versendet werden müssen, da eventuell viele Pfade untersucht werden müssen. Mit dem Alternativ-Graph-Ansatz wurde ein neues Konzept entwickelt, welches auf diesen Nachteil eingeht. Bei diesem Ansatz werden direkt beim Start mehrere verschiedene Pfade zum Ziel berechnet und alle diese Pfade werden untersucht. Dadurch, dass sich diese Pfade deutlich

voneinander unterscheiden, bietet dieser Ansatz dem Routenplaner einen besseren Überblick über die Verkehrslage auf dem ganzen Straßengraphen. Sowohl das zentralisierte Geocast Protokoll als auch das probabilistische Geocast Modell und die drei Ansätze wurden anschließend implementiert. Die drei Ansätze wurden anschließend für jedes der beiden Protokolle OverDrive und das zentralisierte Geocast Protokoll sowie für das probabilistische Geocast Modell simuliert. Darüber hinaus wurden für das zentralisierte Geocast Protokoll Simulationen mit einer unterschiedlichen Anzahl an Fahrzeugen durchgeführt. Die Auswertung der Ergebnisse dieser Simulationen zeigte, dass für das probabilistische Geocast Protokoll der Alternativ-Graph-Ansatz die kleinste Reisezeit bewirken konnte, dafür aber auch den größten Bandbreitenverbrauch hatte. Des Weiteren konnte durch die Ergebnisse des zentralisierten Geocast Protokoll gezeigt werden, dass sich eine größere Anzahl an Fahrzeugen positiv auf die erreichte Reisezeit auswirkt. Der direkte Vergleich der Ergebnisse des zentralisierten Geocast Protokolls und OverDrive - wenn für beide 5000 Fahrzeuge simuliert wurden - zeigte, dass OverDrive annähernd die gleichen Resultate liefert wie das zentralisierte Geocast Protokoll und somit geeignet ist, um als Kommunikationsprotokoll für den Routenplaner eingesetzt zu werden.

Ein Aspekt, der in dieser Arbeit nicht betrachtet wurde, war der Datenschutz und Sicherheit im Allgemeinen. Damit ein Routenplaner„ wie er in dieser Arbeit vorgestellt wurde jemals in der realen Welt zum Einsatz kommen kann, muss garantiert werden, dass ein Angreifer keine Möglichkeit hat, die Pfadwahl eines Fahrzeuges zu beeinflussen, indem er falsche Angaben zur Verkehrssituation macht. Neben den Sicherheitsaspekten gibt es noch weitere Aspekte, die zukünftige Arbeiten zu diesem Thema behandeln können. So wäre z. B. die Zusammenarbeit mehrerer Routenplaner interessant. Ein Problem, das aktuelle Routenplaner wie z. B. TomTom zur Zeit haben, besteht darin, dass sie bei einem erkannten Stau alle Fahrzeuge auf die gleiche Ausweichroute schicken und es somit auf dieser Route wieder zu einem Stau kommen kann. Würden die Routenplaner der Fahrzeuge aber Informationen über die gefahrenen Routen austauschen, so könnten sie damit verhindern, dass alle Fahrzeuge die gleiche Ausweichroute wählen und ein erneuter Stau könnte vermieden werden. Des Weiteren könnten die schon vorhandenen Fahrerassistenzsysteme verwendet werden, um z. B. die Verkehrsdichte auf einer Straße besser einzuschätzen. Ist ein Fahrzeug auf der Autobahn nur mit 100km/h unterwegs, obwohl es keine Geschwindigkeitsbegrenzung gibt, so könnte mit Hilfe des Abstandsmessers ermittelt werden, ob eine hohe Verkehrsdichte den Fahrer zwingt, langsamer zu fahren oder ob der Fahrer nur nicht schneller fahren möchte. Mit Hilfe dieser Information könnte besser abgeschätzt werden, wie die tatsächliche Verkehrslage auf einer Straße aussieht.

Literaturverzeichnis

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Alternative routes in road networks. *J. Exp. Algorithmics*, 18, Apr. 2013.

[2] R. Bader, J. Dees, R. Geisberger, and P. Sanders. Alternative route graphs in road networks. In *Proceedings of the First International ICST Conference on Theory and Practice of Algorithms in (Computer) Systems*. Springer-Verlag, 2011.

[3] I. Baumgart, B. Heep, and S. Krause. OverSim: A Flexible Overlay Network Simulation Framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, pages 79–84. IEEE, 2007.

[4] D. Delling, P. Sanders, D. Schultes, and D. Wagner. Algorithmics of large and complex networks. chapter Engineering Route Planning Algorithms. Springer-Verlag, 2009.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, pages 269–271, 1959.

[6] B. Heep, M. Florian, J. Volz, and I. Baumgart. OverDrive: An Overlay-based Geocast Service for Smart Traffic Applications. In *Proceedings of the 10th Annual Conference on Wireless On-Demand Network Systems and Services (WONS)*. IEEE, 2013.

[7] M. Kobitzsch, M. Radermacher, and D. Schieferdecker. Evolution and Evaluation of the Penalty Method for Alternative Graphs. In D. Frigioni and S. Stiller, editors, *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 33 of *OpenAccess Series in Informatics (OASIcs)*, pages 94–107, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4), December 2012.

[9] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve. Data aggregation and roadside unit placement for a vanet traffic information system. In *Proceedings of the Fifth ACM International Workshop on VehiculAr Inter-NETworking*, VANET '08, pages 58–65, New York, NY, USA, 2008. ACM.

[10] D. Luxen and D. Schieferdecker. Candidate sets for alternative routes in road networks. In *Proceedings of the 11th International Conference on Experimental Algorithms*, SEA'12, pages 260–270, Berlin, Heidelberg, 2012. Springer-Verlag.

[11] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. pages 53–65, 2002.

[12] J. C. Navas and T. Imielinski. Geocast—geographic addressing and routing. In *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '97, pages 66–76. ACM, 1997.

[13] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea. Vanet routing on city roads using real-time vehicular traffic information. *Vehicular Technology, IEEE Transactions on*, 58(7):3609–3626, Sept 2009.

[14] A. Paraskevopoulos and C. Zaroliagis. Improved Alternative Route Planning. In D. Frigioni and S. Stiller, editors, *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 33 of *OpenAccess Series in Informatics (OASIcs)*, pages 108–122, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[15] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[16] T. Willke, P. Tientrakool, and N. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *Communications Surveys Tutorials, IEEE*, 11(2):3–20, Second 2009.