Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Dahlem Center for Machine Learning and Robotics

Kooperative Fahrstrategieplanung über IEEE 802.11p am Beispiel der Funktion Platooning

Julian Pfeifer
Matrikelnummer: 4207283
julian.pfeifer@fu-berlin.de

Betreuer: Oliver Sawade
Eingereicht bei: Prof. Dr. Daniel Göhring
Zweitgutachter: Dr. Ilja Radusch

Berlin, 22. Oktober 2017

Zusammenfassung

Die standardisierte Kommunikation zwischen Fahrzeugen eröffnet neue Möglichkeiten, Sicherheit, Komfort und Effizienz im Straßenverkehr zu verbessern. Eine dieser Möglichkeiten ist die Absprache und Ausführung gemeinsamer automatisierter Fahrfunktionen wie z. B. Platooning. Um eine solche kollaborative Fahrfunktion umzusetzen, reicht eine kurzfristige taktische Planung zwischen den Fahrzeugen nicht aus. Es wird eine längerfristige kooperative Fahrstrategieplanung benötigt. Allerdings ist der momentane Standard für Fahrzeugkommunikation in Europa (ETSI ITS G5) nicht dafür konzipiert, diese zu verwirklichen. In aktueller Forschung wird versucht diese Problematik zu lösen. Diese Arbeit stellt ein Konzept für die kooperative Fahrstrategieplanung aus dem Forschungsprojekt „Intelligente Manöver Automatisierung - kooperative Gefahrenvermeidung in Echtzeit“ (IMAGinE) vor. Durch die Ausarbeitung und Erweiterung dieses Konzepts ist das Collaborative Strategic Session-based Planning (CSSP) Protokoll entstanden. Es ermöglicht eine robuste kollaborative Fahrstrategieplanung über den Funkstandard IEEE 802.11p. Das CSSP Protokoll ist, mit nur einer verwendeten Nachrichtenart, als simple Erweiterung des ETSI ITS G5 Standards konzipiert. In dieser Arbeit stelle ich CSSP vor, zeige eine Beispielimplementierung anhand einer Platooning-Funktion und evaluiere das Konzept hinsichtlich Funktionsfähigkeit und Robustheit.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand Anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder Ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, 22. Oktober 2017

Julian Pfeifer

Inhaltsverzeichnis

1 Einleitung und Motivation

Vor über 130 Jahren wurde das erste Automobil mit Verbrennungsmotor von Carl Benz zum Patent angemeldet [1]. Seitdem wird das Automobil stetig verbessert, insbesondere in Hinsicht auf Sicherheit, Komfort und Effizienz. Im Jahr 2014 erreichte die Automobilindustrie die Rekordzahl von 57.000 Patenten für Automobile [2]. Ein Weg, die Automobile zu verbessern, sind Fahrerassistenzsysteme. Einige davon, wie die Servolenkung, unterstützen den Fahrer bei den grundlegenden Fahraufgaben. Diese lassen sich nach Winner et al. [3] in Navigation, Führung und Kontrolle unterscheiden. Erste aktive Fahrerassistenzsysteme wie das Antiblockiersystem unterstützen auf Kontrollebene. In den letzten Jahren wurden mit Hilfe von moderner Elektronik fortgeschrittene Fahrerassistenzsysteme (ADAS) wie zum Beispiel das elektronische Stabilitätsprogramm, der Abstandsregeltempomat oder der Notbremsassistent entwickelt. Ein Automobil mit Frontkollisionswarner und Notbremsassistent verringert Auffahrunfälle um zirka 40 Prozent [4], im Vergleich zu einem Automobil ohne diese Systeme. Aktuell findet eine weitere Entwicklung zu automatisiertem Fahren statt [5], [6]. Im Gegensatz zu den vorher genannten ADAS greifen automatisierte Funktionen auf mehreren Ebenen ein, also z.B. in lateraler und longitudinaler Führung. Ein Beispiel für zurzeit verfügbare automatisierte Funktionen sind Autobahnpiloten, die für eine gewisse Zeit die komplette Führung des Fahrzeugs übernehmen [7]. Um diese Funktionen zu realisieren wird eine maschinelle Wahrnehmung des umgebenden Verkehrs benötigt. Dies wird durch Sensoren wie Kameras, Radarsensoren und LiDAR-Systemen (light detection and ranging systems) gewährleistet. Mit diesen Sensoren konnten bereits viele Aufgaben erfolgreich gelöst werden. Inzwischen gibt es unter anderem Algorithmen für Spur- und Straßenerkennung, Objekterkennung und -verfolgung, Verhaltensanalyse und Szenenverständnis [8]. Die natürliche Grenze der Algorithmik und somit der Fahrfunktion ist jedoch die Erfassungsreichweite der Sensoren. Diese sind auf Sichtlinie („line-of-sight“) begrenzt. Wenn also z. B. ein Automobil hinter einer nicht einsehbaren Kurve liegen bleibt, dann kann kein heutiger Sensor dieses erfassen und der Fahrer kann nicht über die Gefahr informiert werden. Dieses Problem ist einer der Gründe für die aktuelle Forschung an der Kommunikation zwischen Fahrzeugen.

100 Jahre nachdem Carl Benz sein Patent einreichte, also vor ungefähr 30 Jahren, wurde in Europa die Forschung an Kommunikation zwischen Fahrzeugen begonnen. Damals wurden im Projekt PROMETHEUS [9] bereits die ersten Grundsteine für heutige Forschung im Bereich Fahrzeug-zu-Fahrzeug und Fahrzeug-zu-Infrastruktur Kommunikation (V2X-Kommunikation) gelegt. Heutzutage ist die V2X-Kommunikation in Europa mittels ETSI ITS G5 [10] standardisiert und wurde in Feldversuchen erprobt [11], [12].

Mit der in ITS G5 standardisierten V2X-Kommunikation kann die Verkehrssicherheit, Effizienz und der Komfort gesteigert werden, wie z. B. das sim^{TD} Projekt [13] gezeigt hat. Von 2009 bis 2013 war sim^{TD} der bislang größte europäische Feldversuch zur V2X-Kommunikation und bot neben über 100 ausgerüsteten Fahrzeugen, Roadside Units und angeschlossenen Lichtsignalanlagen sogar eine kooperative Verkehrszentrale, die an bestehende Verkehrszentralen angeschlossen war. Im Projekt wurden z. B. Warnungen vor Einsatzfahrzeugen, Unfällen, Stauenden oder liegengebliebenen

Fahrzeugen erprobt. Durch die Funkreichweite von bis zu einem Kilometer, sowie der Möglichkeit Nachrichten auch außerhalb der Sichtlinie zu empfangen, konnten Fahrer rechtzeitig gewarnt und eine deutliche Erhöhung der Sicherheit nachgewiesen werden. Zusätzlich war es möglich den Fahrern Informationen über die Verkehrslage und die Verkehrsinfrastruktur anzuzeigen, um die Verkehrseffizienz zu erhöhen. So wurde den Fahrern unter anderem die optimale Geschwindigkeit angezeigt, mit der sie die nächste Ampel passieren können. Dies sind nur einige der vielfältigen Anwendungsmöglichkeiten von V2X-Kommunikation, die man mit einer entsprechenden Ausrüstung von Fahrzeugen und Infrastruktur umsetzen könnte. Eine Einführung von V2X-Kommunikation auf Basis des ETSI ITS G5 zugrundeliegenden Funkstandards IEEE 802.11p wird bereits im Jahr 2019 erwartet [14]. In den USA wird in Erwägung gezogen, die nötige Technik für die V2X-Kommunikation in Fahrzeugen ab dem Jahr 2021 verpflichtend zu machen [15].

Obwohl die Automobile stetig sicherer gemacht werden, sterben ungefähr 1,25 Millionen Menschen weltweit jedes Jahr in Verkehrsunfällen [16]. Die aktuelle Forschung strebt danach Sicherheit, Komfort und Effizienz auf der Straße weiter zu steigern und beschäftigt sich verstärkt mit der Kombination von vernetztem und automatisiertem Fahren, den „CoDas“ - Cooperative Driver Assistance Systems, zu Deutsch „kooperative Fahrerassistenzsysteme“ [17]. So könnte man z. B. den Abstandsregeltempomat mit Kommunikation zu einem kooperativen Abstandsregeltempomat erweitern. Da die Fahrzeuge ihre Eigenbewegung selbst viel genauer erfassen können als die Sensoren des nachfolgenden Fahrzeuges, kann die Kommunikation hier direkt zur verbesserten Längsführung beitragen. Die Funktion ist dadurch effizienter (Durchschnittsgeschwindigkeit) und reduziert die Kosten (Bremsaufwand) [18]. Ein anderes Beispiel, an dem bereits lange geforscht wird, ist das Fahren in einem Platoon. Dabei fahren Fahrzeuge längs- und querautomatisiert mit sehr geringen Abständen hintereinander. Platooning ist als Forschungsthema unter anderem interessant, weil es den Platz auf Straßen effizienter nutzen und den Treibstoffverbrauch senken kann [19], [20]. Die momentanen ITS G5 Standards für die V2X-Kommunikation sehen jedoch keinerlei Konzept für längerfristige Kooperation durch Kommunikation vor, wie man sie für Platooning benötigt [21], [22].

In dieser Arbeit werde ich ein solches Konzept für kooperative Fahrstrategieplanung vorstellen. Mein Konzept setzt auf einen verteilten Zustandsautomaten zur Absprache und Verhandlung gemeinsamer kooperativer Fahrmanöver. Dieser Automat erlaubt spezifisch je Funktion eine synchronisierte Verhaltensweise der Fahrzeuge. Die genutzten Nachrichten sollen im Gegenteil dazu generisch ausgeführt werden, um zukünftig auch andere kooperative Funktionen zu ermöglichen. Im Gegensatz zu taktischer Kooperation (Planung für 5 - 10 Sekunden) ist die Absprache hierbei fokussiert auf einen Zeithorizont von 30 Sekunden bis 30 Minuten. Um das Konzept zu testen habe ich die fahrstrategischen Elemente einer Platooning-Funktion implementiert. In einer Simulationsumgebung zeige ich die Umsetzbarkeit der kooperativen Fahrstrategieplanung, die genutzten Nachrichten und schätze ab, ob die resultierende Funkkanallast in einem akzeptablen Bereich liegt. Des Weiteren evaluiere ich, wie das Konzept auf degradierende Kommunikationsbedingungen (insbesondere Paketverlust) reagiert.

2 Stand der Technik

Im folgenden Kapitel werde ich den aktuellen Stand der V2X-Kommunikation vorstellen und gebe einen Überblick über die Platooningkonzepte, auf die ich mich im Rest der Arbeit beziehe.

2.1 European Telecommunication Standards

Die maßgebliche Organisation für die Standardisierung von Kommunikation im Straßenverkehr innerhalb Europas ist das European Telecommunications Standards Institute (ETSI). Es hat die aktuellen Standards für die V2X-Kommunikation in Europa unter dem ETSI ITS G5 Standard zusammengefasst (siehe Abbildung 1). Vergleichbare Konzepte gibt es für den amerikanischen (SAE J2735) und asiatischen Markt, diese stehen jedoch nicht im Fokus meiner Arbeit.

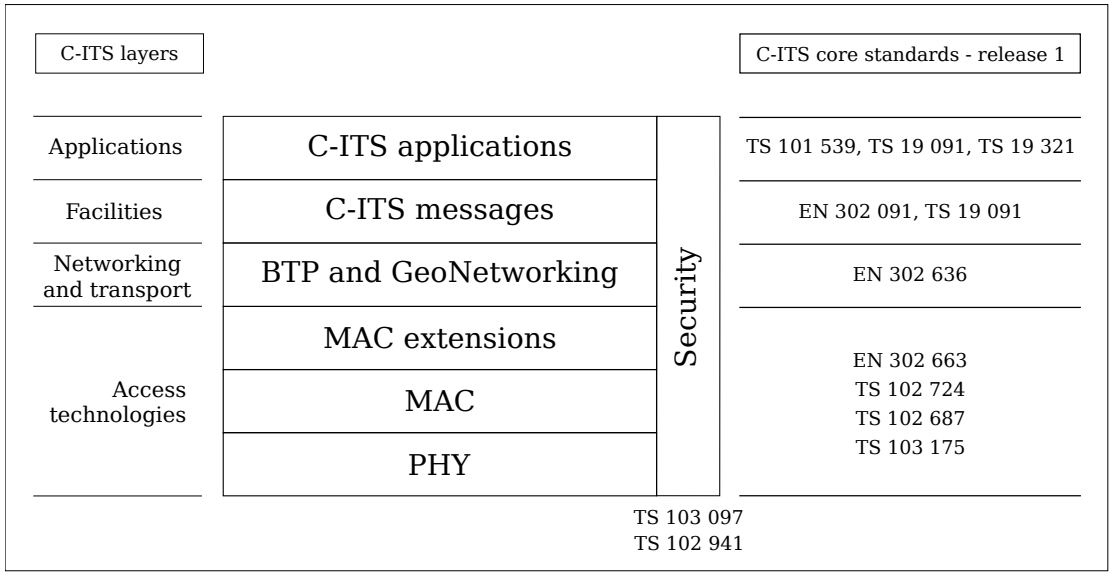

Abbildung 1: ETSI ITS G5 Protokollstapel nach Festag [23]

Die Kurzstreckenkommunikation zwischen Fahrzeugen ist im ETSI ITS G5 über den Standard IEEE 802.11p geregelt. Er ermöglicht es, Ad-hoc-Netzwerke für Verkehrsteilnehmer aufzubauen. Über periodisch ausgesendete Nachrichten wird eine Nachbarschaft aufgebaut, welche dann für geographische Netzwerke und Routing [24] benutzt werden kann. Dabei werden Nachrichten mit einem Zielort an einen anderen Ad-hoc-Teilnehmer gesendet (Geocast), der sich in der richtigen Richtung befindet. Damit die Nachricht das Ziel erreicht, kann sie so mehrere Male weitergereicht werden. Dies bezeichnet man als "multi-hop". Außerdem sind Fahrzeuge in der Lage Nachrichten "mitzunehmen", wenn sie sich in Richtung Zielort fortbewegen (z. B. mittels Gegenverkehr), um diese dort auszusenden.

Im ITS G5 Standard stehen zwei verschiedene Nachrichten im Mittelpunkt. Das ist zum einen die Cooperative Awareness Message (CAM) [25]. Sie wird regelmäßig (zwischen

1 und 10 Hertz), mit Statusinformationen über den Sender wie z. B. Geschwindigkeit, Position und Eigenzustand zusammengestellt. Die CAM Nachricht wird an alle Fahrzeuge im Umkreis ausgesendet und von diesen nicht weitergeleitet (Broadcast). Neue CAM Nachrichten überschreiben alte und werden, wenn keine Neuen empfangen werden, nach einer bestimmten Zeit bei den Empfängern gelöscht. Zum anderen gibt es die Decentralized Environmental Notification Message (DENM) [26]. Sie warnt vor Gefahren auf der Straße und informiert über abnormale Situationen sowie deren Art und Ort. Eine DENM kann z. B. von einem liegen gebliebenen Fahrzeug oder einem Fahrzeug im Stauende verschickt werden. Sie wird je nach Situation mittels Geocast oder Broadcast verschickt. Auch sie besitzt eine Lebenszeit, nach der sie ungültig und bei den Empfängern gelöscht wird. Alle aktuellen Nachrichten des ITS G5 Standards werden über das „Basic Transport Protocol“ (BTP) [27] versendet.

2.2 Platooning

Eine der Anwendungen, die zukünftig mit Hilfe der V2X-Kommunikation ermöglicht werden soll, ist das Platooning. Dabei stellen mindestens zwei Fahrzeuge, in der Regel auf der Autobahn, einen engen Verbund her. Es wird erwartet, dass Sicherheit, Fahrkomfort, Treibstoff- und Verkehrseffizienz durch Platooning erhöht wird [28]. Platooning wurde bereits in den 1990er Jahren erstmals erforscht und stand seitdem oft im Fokus von Forschungsprojekten [29]. Diese haben teilweise starke Unterschiede in Umsetzung und Konzept aufgewiesen. Dazu gehörten z. B. [19]:

- Grad der Automatisierung
- gemischte versus gleichartige Fahrzeuge
- Erstellung von spezieller Infrastruktur für den Anwendungsfall

Alle Forschungsprojekte haben ein grundlegendes Szenario gemein. Es existiert immer ein Führungsfahrzeug, das an der Spitze des Platoons fährt. Alle anderen Teilnehmer fahren in der gleichen Spur dahinter, ohne dass sich andere Fahrzeuge dazwischen befinden und sind zumindest längsautomatisiert. Die Fahrzeuge nehmen im Verbund einen sehr kurzen Abstand zueinander ein, der durch die Kommunikation ermöglicht wird. Indem das Führungsfahrzeug seine Geschwindigkeit und Beschleunigung übermittelt, können sich direkt alle folgenden Fahrzeuge daran orientieren. Ohne die Kommunikation müsste jedes Fahrzeug über eigene Sensorik die Geschwindigkeit und Beschleunigung des voranfahrenden Fahrzeugs messen beziehungsweise bestimmen. Dies würde bedeuten, dass eine Beschleunigung des Führungsfahrzeugs sich Schritt für Schritt durch den Platoonverbund fortpflanzt. Dadurch wäre das System anfällig für Verzögerungen und der Aufaddierung von Fehlern, welche die automatisierte Steuerung instabil machen und Schwingungen im Verbund begünstigen können [30].

Die früheren Forschungsprojekte zum Thema Platooning und anderen CoDAS, wie zum Beispiel das PATH Programm [29] (Forschungsprojekt zur Automatisierung auf US Amerikanischen Highways), setzten häufig auf anwendungsspezifische V2X-Kommunikationslösungen. Es wurden Anforderungen an die Kommunikation aus spezifischen Anwendungen ermittelt und als Mittel zum Zweck verwendet. Seit dem

Einsetzen der Standardisierung im Bereich der V2X-Kommunikation wurden aber zunehmend standardisierte Lösungen verwendet.

In den Jahren 2009 bis 2012 wurde das Forschungsprojekt „Safe Road Trains for the Environment“ (SARTRE) [31] mit dem Ziel durchgeführt, ein sicheres Platooningsystem für die offene Straße, genauer gesagt Autobahnen, zu entwerfen und dieses zu demonstrieren. Damit sollte eine Veränderung in der Benutzung des persönlichen Verkehrs unterstützt werden. Im Konzept von SARTRE ist das Führungsfahrzeug des Platoons immer ein nicht privates Fahrzeug, z. B. ein Lastkraftwagen. Der Rest des Platoons kann durch beliebige Fahrzeuge gebildet werden. Dabei befindet sich die benötigte Technik allein in den Fahrzeugen, es wird keine spezielle Infrastruktur benötigt. Für die Kommunikation wird zur physischen Übertragung der IEEE Standard 802.11p verwendet [32]. Um das Platoon zu realisieren wurde ein neuer SARTRE CAN-Bus in den Fahrzeugen eingerichtet. Der CAN-Bus (Controller Area Network Bus) ist ein serielles Bussystem, welches im Auto eingesetzt wird, um Daten zwischen Komponenten auszutauschen. Die V2X-Kommunikation wird hierbei als Gateway für den SARTRE CAN-Bus verwendet, um alle Fahrzeuge über diesen CAN-Bus miteinander zu verbinden und die nötigen Informationen für das Platooning auszutauschen. Dafür wird im SARTRE Projekt eine spezielle CAN-Nachricht verwendet, die sich in keinerlei Standards wiederfindet.

Schließlich führten Harmonisierungsbestrebungen 2016 zur European Truck Platooning Challenge (EU TPC)[33], dem ersten grenzüberschreitenden Platooning-Experiment. Daran nahmen fünf europäische Staaten und die sechs Europäischen Lastkraftwagenhersteller teil. Jeder dieser Hersteller startete einen Lastkraftwagenplatoon an einem anderen Ort in Europa mit dem gemeinsamen Ziel, dem Hafen von Rotterdam. Dabei ging es nicht um einen technischen Wettbewerb zwischen den Herstellern, sondern vielmehr um den Beginn einer Harmonisierung der neuen Fahrzeugtechnologien zwischen den Behörden der einzelnen Staaten. Ein Ziel des Experiments war es grenzüberschreitende Korridore bis zum Jahr 2020 zu ermöglichen, in denen es erlaubt wird, Platoons zu bilden.

2.3 Kooperative Strategieplanung

Weder SARTRE noch die EU TPC legten einen Fokus auf die kollaborative Verhandlungsebene, welche unbedingt erforderlich ist, um CoDAS wie z. B. Platooning in Serienmodelle zu integrieren. Um dies genauer zu erklären möchte ich zunächst auf eine Klassifizierung von CoDAS eingehen, die Sawade und Radusch [17] vorgeschlagen haben. Dabei werden diese in drei verschiedene Stufen eingeteilt:

1. implizite CoDAS - Diese Fahrerassistenzsysteme werden implizit durch bereits vorhandene standardisierte unilaterale Kommunikation ermöglicht, wie z. B. die CAM.

2. explizite CoDAS - Hier wird eine explizite unilaterale Kommunikation benötigt um diese CoDAS zu ermöglichen, was bedeutet, dass die benötigten Informationen zusätzlich ausgesendet werden müssen. Dafür ist es nötig, die expliziten

Funktionen in die aktuellen ITS G5 Standards zu integrieren.

3. kollaborative CoDAS - Für aktive Kollaboration wird eine bilaterale Kommunikation zwischen Fahrzeugen benötigt. In anderen Worten, um eine aktive Kollaboration zwischen Fahrzeugen zu ermöglichen, wird eine aktive Abstimmung über Entscheidungen benötigt, welche über ITS G5 noch zu standardisieren ist.

Für die kollaborativen CoDAS muss je nach Anwendungsfall mehr oder weniger zwischen den Autos ausgehandelt beziehungsweise koordiniert werden. Generell muss bei jeder kollaborativen Funktion zunächst entschieden werden, ob und mit wem diese durchgeführt wird. Der restliche Teil ist funktionsspezifisch. Beim Platooning muss zum Beispiel abgestimmt werden, wer in welcher Position fährt oder ob ein drittes Fahrzeug dem Platoon beitreten darf. Dies bezeichnet man als kooperative Fahrstrategieplanung.

Das aktuelle Forschungsprojekt „Intelligente Manöver Automatisierung - kooperative Gefahrenvermeidung in Echtzeit“ (IMAGinE) forscht an innovativen Assistenzsystemen für kooperatives Fahren und neuen Standards zur Informationsübermittlung zwischen Fahrzeugen. Es sieht eine kooperative Fahrstrategieplanung über IEEE 802.11p vor. Zusätzlich zu aktuellen Standards wird unter anderem eine strategische Kooperationsnachricht verwendet, um die Kooperation zu ermöglichen [34]. Des Weiteren wird in IMAGinE eine verteilte Perzeption vorgesehen, also die Übermittlung von in Eigensensorik erfassten Fahrzeugen an andere Kommunikationsteilnehmer.

3 Konzept

In diesem Kapitel stelle ich ein Konzept zur fahrstrategischen Kooperation zwischen Fahrzeugen vor. Dieses basiert auf verteilten Zustandsautomaten, die über ein von mir adaptiertes Protokoll mittels Nachrichten zwischen Fahrzeugen ausgetauscht werden. Das Konzept geht dabei über den Horizont von taktischer Planung mit z. B. Trajektorien hinaus. Als Beispielfunktion für mein Konzept wende ich dieses auf „Platooning“ an. Ich bezeichne das Protokoll im Folgenden mit CSSP (Collaborative Strategic Session-based Planning). Das Konzept basiert auf einem Entwurf aus dem IMAGinE Forschungsprojekt [34], den ich an einigen Stellen ausgearbeitet beziehungsweise erweitert habe. Sofern nicht anders gekennzeichnet beziehen sich die Ausführungen in diesem Kapitel auf das aus dem IMAGinE Projekt entnommene Protokoll. Meine Erweiterungen lege ich gesondert im Unterkapitel 3.2 dar.

Die der taktischen Ebene übergeordnete kooperative Fahrstrategieplanung ermöglicht es, kooperative Funktionen mit anderen Fahrzeugen zu starten, den aktuellen Zustand der Funktion zu halten und diesen durch Verhandlung zu verändern. Die kooperative Fahrstrategieplanung wird dabei nicht mit der taktischen Kooperation vermischt. Beides sind unabhängige und eigenständige Konzepte. Die strategische Kooperation ergänzt die taktische Kooperation um eine Verhandlungsmöglichkeit für einen gemeinsamen Fahrzustand wie z. B. das Aufbauen eines Platoons. Dieser

wiederum bestimmt das taktische Verhalten. Eine Kooperation auf taktischer Ebene (also z. B. zur Aushandlung konfliktfreier kooperativer Trajektorien) kann jedoch auch losgelöst von der strategischen Ebene erfolgen. In der weiteren Arbeit fokussiere ich die fahrstrategische Ebene.

Das von mir dargelegte Konzept ist also dazu entwickelt, einen gemeinsamen Zustand zu halten und gemeinsame Übergänge in Nachfolgezustände abzustimmen. Des Weiteren wird angenommen, dass sich alle Teilnehmer wohlwollend verhalten und wahrheitsgetreu handeln. Diese Annahme widerspricht Konzepten wie z.B. des Konsenses unter „byzantinischen“ Bedingungen [35]. Ich erläutere die möglichen Auswirkungen bösartiger Akteure in Kapitel 6.

3.1 IMAGinE Konzeptentwurf

Der Konzeptentwurf der fahrstrategischen Kooperation zwischen Fahrzeugen aus IMAGinE basiert auf einem über alle Fahrzeuge verteilten Zustandsautomaten. Die kurzfristige taktische Verhaltensweise der einzelnen Autos wird über den derzeit aktiven Zustand bestimmt. Sobald ein Fahrzeug aufgrund äußerer Einflüsse oder eigener Motive in einen anderen Zustand wechseln möchte, kann es einen Zustandswechsel anstoßen und den Wunsch mit den anderen Fahrzeugen verhandeln. Eine wichtige Anforderung an das Konzept ist die Synchronizität aller beteiligten Fahrzeuge. Diese besagt, dass sich zu jedem Zeitpunkt alle Fahrzeuge im gleichen internen Zustand befinden müssen, oder in den gleichen Nachfolgezustand wechseln. Es darf nicht möglich sein, dass Fahrzeuge gleichzeitig in verschiedene Zustände überwechseln. Da Netzwerkkommunikation, speziell in drahtlosen ad-hoc Netzwerken, stets verlustbehaftet ist, lässt sich diese Anforderung nie komplett erfüllen [36]. Daher muss es weiterhin einen Resynchronisationsmechanismus geben, falls durch Kommunikationsstörungen ein Wechsel in neue Zustände verpasst wurde. Um verschiedene Funktionen mit Hilfe dieses Konzepts zu realisieren, wird für jede Einzelfunktion ein Zustandsautomat festgelegt, welcher die Zustände innerhalb der Funktion sowie erlaubte Zustandsübergänge definiert. Durch das Festlegen weiterer Zustandsautomaten kann das Protokoll also im Nachhinein für weitere Funktionen erweitert werden. Die benutzten Nachrichten sollten daher funktions-agnostisch ausgelegt sein und den genutzten Zustandsautomaten z.B. über Versionierung referenzieren.

Grundlegend für die Kommunikation dieses Protokolls sind Sessions. Die Session ist ein Datencontainer, der sämtliche kooperative Parameter zwischen den Fahrzeugen hält. Ein Fahrzeug, welches initial die Kommunikation beginnt, erstellt dafür eine Session. Alle folgende Kommunikation wird dann innerhalb dieser Session ausgeführt. Die Abstimmung des initialen und aller Nachfolgezustände erfolgt mit „Wünschen“. Jedem Fahrzeug steht es frei, einen Wunsch zu äußern, wie die Session modifiziert werden soll (z.B. einen Zustandswechsel). Bei voller Zustimmung aller Sessionteilnehmer gilt der Wunsch als angenommen und die Session entsprechend modifiziert. Die Teilnehmer einer Session senden regelmäßig den aktuellen Zustand mit einer definierten Frequenz aus. Dies soll analog zur CAM in einer Frequenz von zehn Hertz über das BTP erfolgen. Dadurch können Desynchronisationen erkannt und ein korrekter einheitlicher Zustand

über alle Fahrzeuge hergestellt werden. Die gesamte Kommunikation dieses Protokolls stützt sich auf einen einzelnen Nachrichtentyp. Er besteht aus folgenden Teilen:

- Header
- Zustandsdaten
- Zustandswunsch (optional)

Der Header dieser Nachricht beschreibt die Session, in der die Nachricht verschickt wird, und setzt sich wie folgt zusammen:

- Version des Protokolls
- Session-ID
- Funktionscode
- Funktionsversion

Die Versionierung des Protokolls stellt sicher, dass es keine Probleme durch Unterschiede im Protokoll zwischen den Fahrzeugen gibt. Über Funktionscode und -version ist sichergestellt, dass alle Teilnehmer wissen, welche Funktion in dieser Session ausgeführt wird, also welcher Zustandsautomat verwendet werden muss. Die Session-ID besteht aus der Station-ID des Fahrzeugs (eindeutiger Identifikator für einen V2X-Kommunikationsteilnehmer), welches die Session initiiert hat und dem Zeitpunkt, zu dem sie dies getan hat. Dadurch kann die Session eindeutig von anderen Sessions unterschieden werden, falls z. B. in der Fahrzeugumgebung von anderen Teilnehmern die gleiche Funktion ausgeführt wird.

Die Zustandsdaten enthalten den aktuellen Zustand, von dem das aussendende Fahrzeug ausgeht:

- Aktueller Zustand des Automaten
- Teilnehmer der Session und ihre Rolle
- Zähler für Zustandsübergänge

Der aktuelle Zustand bezieht sich auf den aktiven Zustand im funktionsspezifischen Zustandsautomaten. Zusätzlich gehört zum aktuellen Zustand eine Liste aller Teilnehmer der Session mit ihrer jeweiligen Rolle innerhalb der Funktion. Der Zähler für die Zustandsübergänge wird jedes Mal erhöht, wenn durch eine erfolgreiche Verhandlung ein neuer Zustand angenommen wird.

Das Konzept der Rollen bestimmt über das kurzfristige taktische Verhalten der Fahrzeuge innerhalb einer Funktion, also welches Verhalten vom jeweiligen Teilnehmer von den anderen erwartet werden kann. Beim Beispiel Platooning werden über die Rollen Positionen innerhalb des Platoons festgelegt. Da es möglich ist die Teilnehmer und die Rollen einer bestehenden Session durch Abstimmung zu modifizieren, können auch komplexere Manöver koordiniert werden, wie z. B. das Beitreten zu einem Platoon in der Mitte. Ein Spielraum für das genaue taktische Verhalten wird über Funktionscode, Funktionsversion, aktuellem Zustand und Rolle festgelegt. Durch Verwendung des

CSSP Protokolls erhält man somit eine funktionsspezifische Kooperation über ein generisches Protokoll.

Um einen Zustandswechsel einzuleiten, kann am Ende der Nachricht ein Wunsch angefügt werden. Dieser enthält:

- Time-out für die Abstimmung
- Wunschzustand
- Veränderte Teilnehmerliste und Rollen (optional)

Ein Wunsch besteht in der Regel mindestens aus einem Time-out und einem Wunschzustand. Es kann aber auch eine Änderung an den Teilnehmern beziehungsweise deren Rollen innerhalb der Funktion enthalten sein. Letzteres kann aus verschiedenen Gründen nötig sein. Wenn ein Fahrzeug die Session verlässt, kann es sein, dass nicht nur die Teilnehmerliste verändert, sondern auch dessen Rolle eingenommen werden muss.

Sobald ein Fahrzeug einen Wunsch äußert, wird dieser von den anderen Fahrzeugen innerhalb der Session evaluiert. Das initiierende Fahrzeug legt dabei den Time-out fest, nach dem die Abstimmung als gescheitert gilt. Die empfangenden Fahrzeuge prüfen den Wunsch und senden bei Zustimmung den gleichen Wunsch aus. Wenn sie dem Wunsch nicht zustimmen, haben sie zwei Optionen:

1. Den Wunsch ignorieren und selber keinen aussenden.
2. Einen Gegenvorschlag machen, also einen anderen Wunsch aussenden.

Die Abstimmung endet entweder sobald alle Fahrzeuge den gleichen Wunsch ausgesendet haben oder wenn der Time-out erreicht ist. Sobald erfolgreich ein Zustandswechsel durchgeführt wird, wird zudem der oben genannte Zähler für die Zustandsübergänge um eins erhöht. Fahrzeuge, die einen Zustandswechsel verpasst haben, können über diesen Zähler feststellen, ob sie einen Wechsel verpasst haben und sich darüber resynchronisieren. Dies funktioniert über das regelmäßige Aussenden des aktuellen Zustands aller Teilnehmer. Wenn ein Teilnehmer einen aktuellen Zustand empfängt, der von seinem eigenen abweicht und einen höheren Zähler für Zustandsübergänge hat, dann geht dieser davon aus, dass er einen Zustandswechsel verpasst hat und würde sich mit diesem neueren Zustand synchronisieren.

Diese Art der Synchronisation bietet im Vergleich zu Ansätzen wie Paxos [37] oder Turquois [38] keine Garantie der steten Synchronizität. Dies ist aber eine bewusste Konzeptentscheidung des Imagine Projekts zur Simplifikation des Protokolls und der Nachrichten. Das gesamte Konzept basiert darauf, dass sich alle Teilnehmer einer Session an die Regeln des Protokolls halten. Ein Teilnehmer könnte böswillig (beziehungsweise eigenwillig) einen neuen Zustand mit einem höheren Zähler für Zustandsübergänge aussenden. Das Protokoll würde dann die Synchronisation aller anderen Teilnehmer in diesen Zustand vorsehen. Daher müssen diese Resynchronisationen teilnehmerseitig ebenso wie Wünsche geprüft werden, um einen ungewollten Übergang in einen potentiell gefährlichen Nachfolgezustand zu vermeiden. So ist es nicht möglich Schaden anzurichten, indem andere Fahrzeuge in ungewollte Zustände

gezwungen werden. Es ist aber nur möglich die Auswirkungen, nicht jedoch die Ursache, zu verhindern, wenn ein solches Protokoll auf einer einzigen Nachricht basiert. Wenn eine aktivere Form der Synchronisation gewünscht wäre, dann würde man einen extra Layer für die Synchronisation und eigens dafür vorgesehene Nachrichten benötigen, wie es zum Beispiel das Collaborative Maneuver Protocol [39] vorsieht.

Um Robustheit gegen Störungen im Funknetzwerk zu erreichen, werden die regelmäßig ausgesendeten Nachrichten aller Teilnehmer beobachtet. Sobald eine gewisse Anzahl an aufeinander folgenden Nachrichten von einem Teilnehmer nicht empfangen werden, wird die Session aus Sicherheitsgründen abgebrochen. Dabei wird diese nicht ordnungsgemäß beendet, da davon ausgegangen wird, dass nicht mehr alle Teilnehmer erreichbar sind. Die Anzahl an Nachrichten, die für einen Sessionabbruch erforderlich ist, ist einstellbar.

3.2 Erweiterungen des Konzepts

Im Entwurf des IMAGinE Forschungsprojekts wird nicht explizit auf den Aufbau einer Session und dem damit verbundenen Anfragen von Funktionen eingegangen. Deswegen habe ich folgenden Mechanismus entwickelt, der möglichst viel von dem bereits vorhandenen Verhandlungskonzept verwendet um die Komplexität zu reduzieren. Dafür ist es nötig, dass nicht nur der Wunsch in der Nachricht optional ist. Während des Sessionaufbaus existiert noch kein gemeinsamer Zustand, sondern nur ein Wunsch. Im Fall von Platooning ist dies der Wunsch nach dem Zustand für den Aufbau eines Platoons. In diesem Fall dürfen die Zustandsdaten in der Nachricht leer sein. Um eine Anfrage für eine kollaborative Funktion über das CSSP Protokoll auszusenden verwendet man eine CSSP Nachricht, die nur mit Header und Wunsch gefüllt ist. Dafür erstellt das initiierende Fahrzeug eine neue (vorerst lokale) Session, in der die Verhandlung der Funktionsanfrage stattfindet. Allerdings ist diese nur eine temporäre Session und mit dem Time-out des Wunsches in der Anfrage begrenzt. Alle Fahrzeuge, die der Anfrage zustimmen treten ebenfalls der temporären Session bei. Falls nicht alle angefragten Fahrzeuge der Kooperation zustimmen oder nicht alle vor Ablauf des Time-outs antworten, verlassen alle Fahrzeuge, die bereits der Kooperation zugestimmt haben, die Session wieder. Sobald ein erster gemeinsamer Zustand verhandelt werden kann, wird der Time-out der Session entfernt. Durch dieses Verfahren ist es nicht nötig, einen extra Mechanismus zum Verhandeln von Kooperationsanfragen zu entwickeln. Der normale Algorithmus der Verhandlungen kann dafür verwendet werden. Allerdings müssen Fahrzeuge, die sich nicht in einer Funktion befinden, empfangene Nachrichten danach auswerten, ob dies mögliche Kooperationsanfragen für sie sind.

Damit eine Session gemeinsam von den Teilnehmern aufgelöst werden kann, muss dies als Wunsch ausgedrückt werden können. Diesen Wunsch definiere ich allgemeingültig für alle Funktionen. Er besteht aus einem Time-out, keinem Wunschzustand und einer leeren Teilnehmerliste. Diese Idee habe ich aus dem anwendungsspezifischen Konzeptentwurf für Platooning aus dem IMAGinE Projekt verallgemeinert.

Die Verhandlungsmechanik aus dem IMAGinE Entwurf ist sehr freizügig. Jeder Teil-

nehmer kann zu jedem Zeitpunkt unbegrenzt seinen Wunsch modifizieren. Ein Zustandswechsel wird daher nur erreicht, wenn alle Fahrzeuge zu einem bestimmten Zeitpunkt denselben Wunsch äußern. Dies hat in der Theorie eine Schwachstelle, falls auf der Netzwerkebene ein Nachrichtenverlust auftritt. Um dies zu verdeutlichen gebe ich folgende Beispiele:

Beispiel 1 (auflösbarer Konflikt)

Es existieren Fahrzeuge Alpha und Beta, die sich alleine innerhalb einer Session in Zustand X befinden. Alpha sendet einen Wunsch A aus. Nach einer gewissen Zeit stimmt Beta durch Aussenden des gleichen Wunsches A zu. Zugleich geht Beta in den Nachfolgezustand A über, da volle Zustimmung vorliegt. Durch Netzwerkstörungen erreicht Fahrzeug Alpha die Zustimmung Betas nicht vor Ablauf des Time-outs, es bleibt also im initialen Zustand X. Beide Fahrzeuge befinden sich nun in unterschiedlichen Zuständen. Dies kann durch den Übergangszähler aufgelöst werden, indem Alpha den Nachfolgezustand von Beta als solchen erkennt und selbst auch nach A wechselt.

Beispiel 2 (nicht auflösbarer Konflikt)

Es existieren Fahrzeuge Alpha und Beta, die sich alleine innerhalb einer Session befinden. Alpha sendet einen Wunsch A aus. Aber Beta sendet einen Gegenvorschlag B aus. Alpha ignoriert diesen, weil es seinen eigenen Wunsch erreichen möchte. Nun ändern sich z. B. äußere Umstände und Alpha stimmt dem Wunsch von Beta zu, während Beta dem Wunsch von Alpha zustimmt. In meinem Beispiel erreichen jedoch beide Nachrichten den jeweils anderen nicht (z.B. durch eine temporäre Kanalsättigung). Nun geht Alpha davon aus, dass alle dem Wunsch B zugestimmt haben und wechselt dorthin. Beta geht hingegen davon aus, dass dem Wunsch A zugestimmt wurde und wechselt in diesen. Beide Fahrzeuge befinden sich nun in unterschiedlichen Zuständen, haben aber einen gleich hohen Zähler für Zustandsübergänge.

Es ist nicht möglich den Konflikt aus Beispiel 2 aufzulösen. Deswegen habe ich einen strikteren Verhandlungsmechanismus entworfen. Er unterscheidet sich wie folgt von dem in IMAGinE entworfenen Mechanismus:

Wenn ein Fahrzeug einen Wunsch äußern möchte, muss er dafür eine Verhandlungsrunde starten, indem er eine Wunsch ID erstellt. In jeder Verhandlungsrunde wird nur über einen Wunsch abgestimmt und es kann nur eine einzige Verhandlungsrunde je Session parallel geben. Die Wunsch ID besteht wie die normale Session-ID aus Station-ID des initiierenden Fahrzeugs und initialem Timestamp. Sie wird dem Wunschteil der Nachricht hinzugefügt. Dadurch kann leicht festgestellt werden, welche Verhandlungsrunde zuerst erstellt wurde. Diese wird dann bevorzugt. Falls die Sessions zur exakt gleichen Zeit erstellt worden sind, wird stets die Session des Fahrzeugs mit der geringeren Station-ID bevorzugt. Wichtig ist an diesem Punkt, dass eine deterministische und eindeutige Entscheidung über die zu behaltende Verhandlung getroffen wird. Fahrzeuge können also keine neuen Wünsche äußern, während bereits über einen anderen Wunsch abgestimmt wird. Sie haben in einer Verhandlungsrunde nur zwei Möglichkeiten: Sie können entweder diesem Wunsch zustimmen und ihn in ihre eigenen Nachrichten übernehmen oder sie können ihn ignorieren. Eine Verhandlungs-

runde endet entweder wegen ihres Time-outs oder sobald alle Fahrzeuge zusammen den Wunsch annehmen. Durch dieses striktere Verfahren kann die oben beschriebene Situation nicht eintreten.

3.3 Platooning

Um kooperative Funktionen über CSSP nutzen zu können, muss für jede ein funktionsspezifischer Zustandsautomat definiert sein. Im Folgenden möchte ich dies am Beispiel von Platooning demonstrieren. Dafür verwende ich ebenfalls einen Entwurf aus dem IMAGinE Forschungsprojekt [34]. Ich werde das taktische Verhalten, das sich aus den Zuständen ergibt, nicht bis zum letzten Detail konkretisieren. Die Implementierung eines Platooning Algorithmus, der dem Stand der Technik entspricht, ist nicht Fokus dieser Arbeit.

Grundlegend benötigt man für das Platooning drei Hauptzustände. Zum einen muss ein Platoon auf- und abgebaut werden können und zum anderen muss man die Haupttätigkeit, das Fahren in einem Platoon, ermöglichen. Zusätzlich sollte es möglich sein, dass Fahrzeuge dem Platoon beitreten und ihn verlassen können. Damit sind die wichtigsten grundlegenden Möglichkeiten innerhalb eines Platoons abgedeckt. Komplexere Manöver, wie z. B. Spurwechsel, würden den Rahmen dieser Arbeit überschreiten. Als Rollen werden die Fahrzeugpositionen innerhalb des Platoons verwendet.

Jedes Platoon beginnt im Zustand „Aufbau“. Ziel dieses Zustandes ist es, die Fahrzeuge richtig zu positionieren, also alle hintereinander auf einem Fahrstreifen in einer bestimmten Reihenfolge anzuordnen, ohne dass sich dritte Fahrzeuge dazwischen befinden. Falls die verhandelte Konfiguration nicht erreicht wird, kann die Konfiguration durch Tausch der Rollen beim Wechsel in den Zustand „Fahren“ der tatsächlichen Reihenfolge angepasst werden.

Im zentralen Zustand „Fahren“ sind mindestens alle Fahrzeuge bis auf das Führungsfahrzeug längs- und querautomatisiert. Dabei werden die Abstände zwischen den Fahrzeugen auf ein technisches Minimum reduziert. Telemetrieinformationen werden zur Regelung verwendet.

Um die Teilnehmer am Platoon zu modifizieren werden die Zustände „Beitreten“ und „Verlassen“ verwendet. Ein Fahrzeug, das dem Platoon beitreten möchte, kann einen Wunsch mit dem Zustand „Beitreten“ nutzen um dies zu kommunizieren. Analog funktioniert dies für ein Fahrzeug, das die Funktion beenden möchte. Je nachdem, ob ein Fahrzeug in der Mitte, an der Spitze oder am Ende beitritt oder den Platoon verlässt, variiert das taktische Manöver leicht. Es ist auch möglich, dass dadurch die Führungsrolle übergeben werden muss. Diese kann wie jede andere Rolle bei den entsprechenden Zustandsübergängen verändert werden.

Falls aus externen oder internen Gründen ein Auflösen des Platoons gewünscht wird, kann der Zustand „Abbau“ eingenommen werden. Dabei erhöhen die Fahrzeuge ihren Abstand zueinander und beenden letztendlich die Session, wenn sie genügend Abstand erreicht haben.

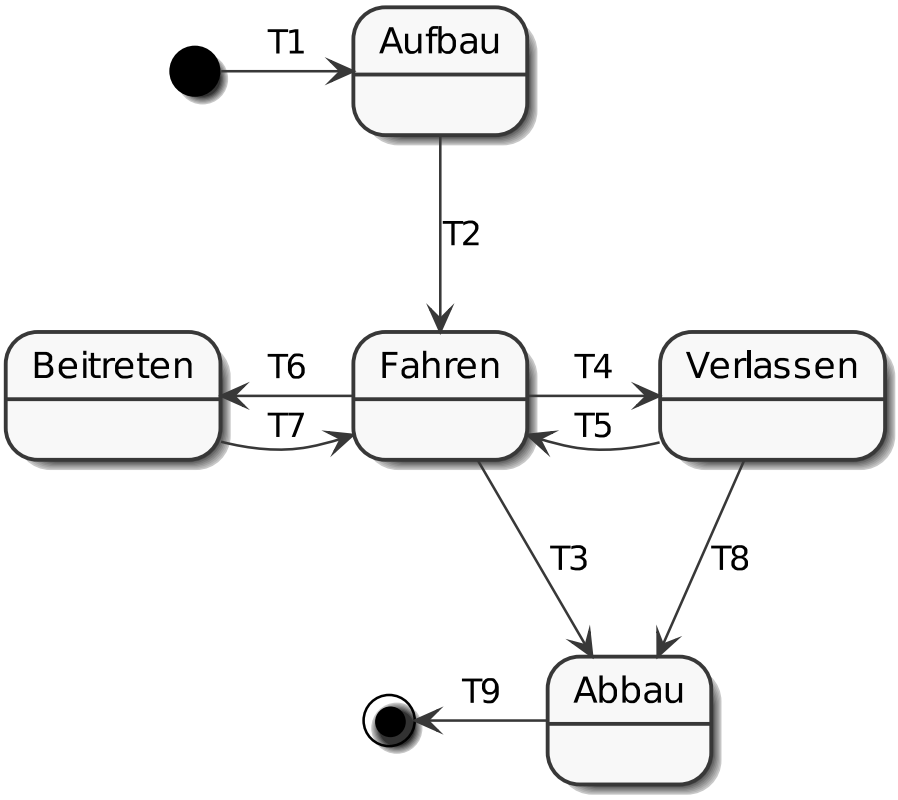

Abbildung 2: Platooning Zustandsautomat

Abbildung 2 zeigt den vollständigen Platooning-Zustandsautomaten von dem ich in dieser Arbeit ausgehe. Die Zustandsübergänge werden darin mit T1 bis T9 bezeichnet. Folgende Bedingungen müssen für die entsprechenden Zustandsübergänge erfüllt sein:

T1, T3, T4, T8: keine

T2, T7: alle Teilnehmer auf einem Fahrstreifen ohne dazwischen befindende Fahrzeuge

T5: regulärer Abstand des verlassenen Fahrzeugs zum Platoon (oder auf anderem Fahrstreifen)

T6: beitretendes Fahrzeug muss sich entweder direkt vor oder hinter dem Platoon oder auf einem angrenzenden Fahrstreifen befinden

T9: regulärer Abstand zwischen allen Fahrzeugen

4 Implementierung

In diesem Kapitel beschreibe ich die Implementierung des von mir adaptierten Protokolls anhand einer Beispielfunktion „Platooning“. Als Umgebung für die Implementierung des CSSP Protokolls habe ich den Simulator PHABMACS [40] (Physics-Aware Behavior Modeling Advanced Car Simulator) verwendet. Dies ist ein javabasierter

Simulator zum Entwickeln und Überprüfen von fortgeschrittenen Fahrerassistenzsystemen und -funktionen, der an der Technischen Universität Berlin entwickelt wird. Der Simulator PHABMACS stellt zusätzlich eine 3D Visualisierung der Simulation mittels OpenGL bereit. Es ist also möglich, die Simulation zu beobachten oder in sie einzugreifen, indem man z. B. die manuelle Steuerung eines Fahrzeugs übernimmt. Die simulierten Fahrzeuge können leicht spezialisiert und angepasst werden. Ein sogenanntes Driver-Modul des Fahrzeugs trifft die Entscheidungen und übernimmt die Steuerung eines simulierten Fahrzeugs. Der Simulator bietet bereits implementierte Driver, die grundlegende automatisierte Fahrfähigkeiten mitbringen und durch Spezialisierung weiter angepasst werden können. Außerdem kann man die Fahrzeuge mit Sensoren ausstatten, die z. B. Abstände messen, aber auch V2X-Kommunikation ermöglichen. Die Welt, in der sich die Fahrzeuge bewegen, kann aus Open Streetmap (OSM) Karten automatisch erstellt werden. Der PHABMACS ist gegenüber der Realwelt validiert und stellt ein realistisches Fahrzeugverhalten dar [40].

Als Programmiersprache verwende ich Kotlin [41]. Diese Sprache wird zu Bytecode kompiliert und dann in der Java Virtual Machine wie Java ausgeführt. Sie ist vollständig kompatibel mit der Programmiersprache Java und verwendet deren Standardbibliotheken. Beide Sprachen können gemeinsam in einem Projekt verwendet werden. Als Build Tool verwende ich Gradle [42]. Ich habe den generischen Teil des CSSP Protokolls vollständig implementiert. Zusätzlich habe ich den funktionsspezifischen Teil für Platooning umgesetzt, um das Protokoll zu evaluieren. Dabei habe ich die strategische Ebene fokussiert, deren Kern der Zustandsautomat darstellt. Die Regelung der Fahrzeuge und dessen taktische Entscheidungen sind dabei nur idealisiert implementiert. Um das Konzept zu demonstrieren habe ich ein Szenario entworfen, in dem erst zwei Fahrzeuge einen Platoon aufbauen, dann ein drittes Fahrzeug diesem beitritt und nach erfolgtem Aufbau alle drei Fahrzeuge kurz in den "Fahren“ Zustand übergehen. Dann verlässt das dritte Fahrzeug den Platoon wieder, anschließend wird der Platoon von den verbliebenen beiden Fahrzeugen aufgelöst. Dabei betrachte ich nur ein idealisiertes perfektes Weltmodell und vernachlässige potentiell störende weitere Fahrzeuge, da diese für den von mir betrachteten Kommunikationsteil irrelevant sind. Die Regelung, Parameter und Bedingungen für Zustandsübergänge innerhalb des Platoonings sind ebenfalls in der Simulation idealisiert. Sie sind nicht Fokus dieser Arbeit, entsprechen aber grob einem realistischen taktischen Verhalten und vor allem dessen Ablauf. Dadurch werden alle Funktionalitäten von CSSP in einer Art verwendet, wie es auch in der Realität der Fall wäre. Außerdem ist es möglich, der Verwendung des CSSP Protokolls visuell zu folgen. Dies habe ich vereinfacht, indem ich zwei verschiedene „Overlays“ in der 3D Visualisierung des PHABMACS implementiert habe. In einem solchen Overlay kann man den aktuellen Zustand aller Fahrzeuge einsehen und in dem anderen ist es möglich, Konsolenausgaben meines Protokolls zu verfolgen, ohne den Blick auf ein anderes Fenster richten zu müssen (siehe Abbildung 3).

Zusätzlich habe ich die komplette Implementierung auf die unterliegende Simulationszeit angekoppelt, die der PHABMACS Simulator bereitstellt. So ist es möglich, die Zeit in der Simulation schneller oder langsamer ablaufen zu lassen, ohne dass dies Auswirkungen auf die Verhandlung hat. Dies benutze ich auf zwei unterschiedliche Weisen in meiner Implementierung. Im oben genannten Demonstrationsszenario wird

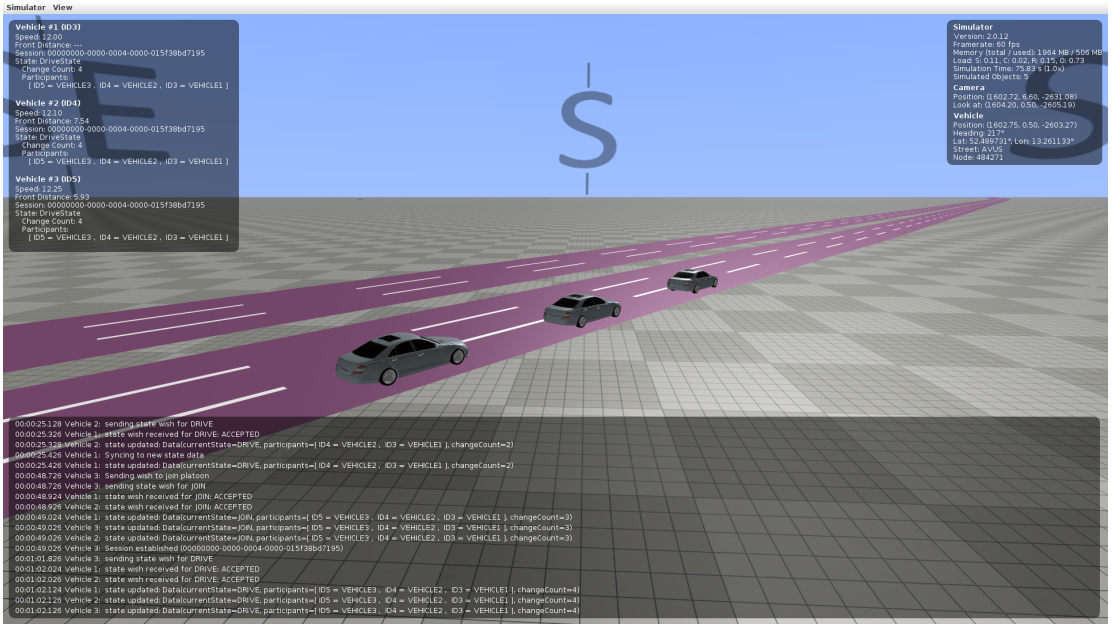

Abbildung 3: PHABMACS Simulator mit Overlays

die Zeit stark verlangsamt, während die Fahrzeuge Zustandswechsel verhandeln, um mehr Zeit zu haben, den Vorgängen als Zuschauer zu folgen. Während der Evaluation habe ich die Zeit mit höherer Geschwindigkeit ablaufen lassen, um mehr Daten in kürzerer Zeit sammeln zu können.

4.1 Architektur

Für mein erstelltes Szenario lade ich eine OSM Karte der AVUS[1] Berlins. Die drei Fahrzeuge, die ich der Simulation hinzufüge, werden initial mit geringem Abstand in den gleichen Fahrstreifen gesetzt. Die Fahrzeuge werden durch einen von mir spezialisierten Driver gesteuert. Er besitzt eine PD-Regelung um einen einstellbaren Abstand zum Frontfahrzeug zu halten. In einem realen Platooning Szenario würde man die Regelung auf die Geschwindigkeit des Führungsfahrzeugs beziehen. Dies war in der idealisierten Simulation nicht nötig. Außerdem habe ich die Fahrzeuge mit einem spezialisierten V2X Sensor „ausgestattet“. Dieser ermöglicht den simulierten Fahrzeugen zusätzlich zu einer Kommunikation, die konzeptuell dem 802.11p Standard entspricht, außerdem die Kollaboration über das CSSP Protokoll. Wie im Konzept Kapitel beschrieben, verwendet das Protokoll nur eine einzelne Nachricht und kann in Anhang A eingesehen werden.

Die Implementierung des CSSP Protokolls entspricht architektonisch einer Anwendung des Strategie-Entwurfsmusters [43] (siehe Abbildung 4). Dieses Muster ist dann nützlich, wenn eine Algorithmenfamilie existiert. Alle Algorithmen dieser Familie erfüllen dabei einen ähnlichen Zweck und können gegeneinander ausgetauscht werden. Dafür

[1] Die AVUS ist die ehemalige Automobil-Verkehrs- und Übungsstraße in Berlin und ist heute Teil der Autobahn A 115.

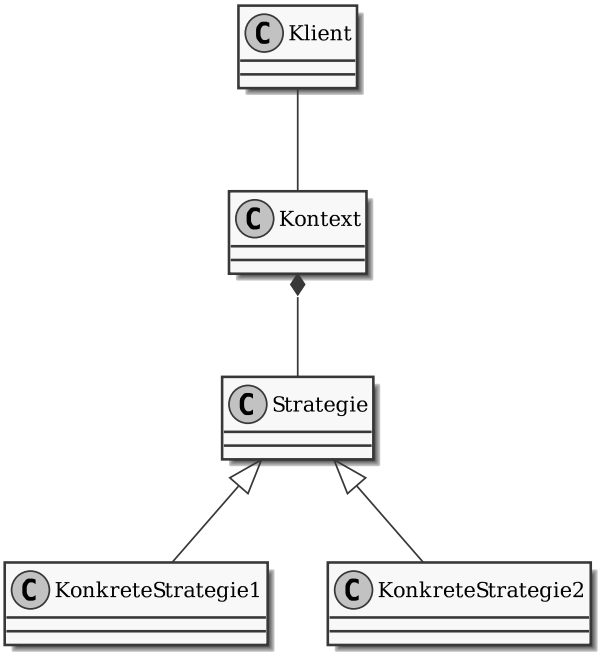

Abbildung 4: Stratgie-Entwurfsmuster

müssen diese die gleiche Schnittstelle bieten. Es existiert ein Kontextobjekt, welches das aktuell eingesetzte Strategieobjekt verwaltet. Dabei kennt das Kontextobjekt aber nur die Schnittstelle, die sich alle Algorithmen der Familie teilen. Ein Klient kann dann über den Kontext auf die Strategiefunktionalität zugreifen. Das Strategieobjekt kann entweder statisch zur Entwurfszeit oder dynamisch zur Laufzeit ausgetauscht werden.

Die Architektur meiner CSSP Implementierung (siehe Abbildung 5) vereint Kontext und Klient in einem Objekt, dem CSSPDirector. Die Strategien sind in diesem Fall die funktionsspezifischen Teile des CSSP Protokolls und implementieren das CSSPController Interface als Schnittstelle für die Familie der Funktionen. Sie werden beim CSSPDirector registriert und stellen bereit, welche kollaborative Funktion sie implementieren. Sobald eine kollaborative Funktion über das CSSP Protokoll angefragt wird, stellt der CSSPDirector fest, ob ein geeigneter CSSPController existiert um die Funktion auszuführen. Falls dieser existiert, wird die Anfrage an diesen CSSPController weitergeleitet. Falls der CSSPController der Funktionsanfrage zustimmt, wird dieser als aktiv vermerkt und die Akzeptanz ausgesendet. Der CSSPController wird über alle folgenden Kommunikationsereignisse die Funktion betreffend vom CSSPDirector informiert und muss die funktionsrelevanten Entscheidungen treffen. Er wird also als fahrstrategischer Entscheider verwendet.

4.2 Protokoll

Der CSSPDirector implementiert alle generischen Teile des CSSP Protokolls und stellt alle relevanten Schnittstellen zum Aussenden von Kooperationsanfragen und Wünschen bereit (siehe Anhang B). Er ist außerdem verantwortlich für:

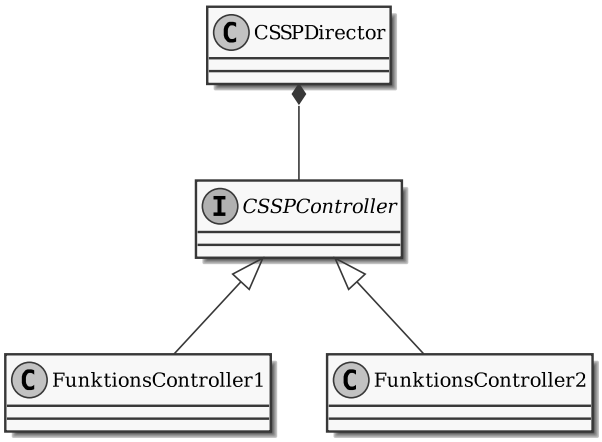

Abbildung 5: CSSP-Teilarchitektur

- Verwaltung und Einsetzen der CSSPController als Strategien für die Funktionen
- Verwaltung der Funktionssession
- Verhandlung der Wünsche und Verwaltung der Verhandlungsrunde
- Vorhalten und regelmäßiges Aussenden des aktuellen Zustands
- Verarbeitung eingehender CSSP Nachrichten

Angefangen bei der Entscheidung, ob eine Funktion zusammen ausgeführt werden soll, werden alle relevanten Ereignisse an den entsprechenden CSSPController weitergeleitet. Er stellt für jedes davon eine Methode zur Verfügung, welche der CSSPDirector entsprechend verwendet. Einige Ereignisse sind rein informativ, damit er die Übersicht über den aktuellen Zustand der Funktion behält. Sie umfassen folgende Ereignisse:

- die Erstellung von temporären beziehungsweise vollständig etablierten Sessions
- das Ende einer Session
- der Übergang in einen erfolgreich verhandelten Zustand
- der Start einer neuen Verhandlungsrunde

Bei fast allen Methoden werden über die Parameter Informationen mit übergeben, um das Ereignis genauer zu spezifizieren. So werden bei dem Ereignis, das die erfolgreiche Einigung eines neuen Zustands angibt, die neuen Zustandsdaten mit übergeben. Das exakte Interface dazu kann in Anhang C betrachtet werden.

Bei anderen Ereignissen muss der CSSPController einen Wunsch auswerten. Letztendlich muss die Zustimmung oder Ablehnung über das Ergebnis der Methode erfolgen. Das Modell ist dabei bewusst synchron gehalten um die Komplexität der Implementierung zu reduzieren, denn wenn der CSSPDirector eine Zustimmung erhält, nimmt er den entsprechenden Wunsch automatisch in den eigenen Zustand auf und sendet sie aus. Es existiert aber die Möglichkeit, einen Wunsch noch nachträglich anzunehmen. Solange die Verhandlungsrunde ihren Time-out noch nicht erreicht hat, kann man die

Schnittstelle des CSSPDirectors nutzen um dem Wunsch zuzustimmen.

Synchronisationen, die über einen neueren empfangenen Zustand erkannt werden, werden in meiner Implementierung direkt vom CSSPDirector angenommen und an den CSSPController gemeldet. Mögliche Sicherheitsmechanismen, wie im Konzept Kapitel diskutiert, wurden in dieser Implementierung ausgeklammert.

Wie im Konzept vorgesehen, wird im Fall von Störungen des Funknetzwerks ab einer definierten Anzahl von verpassten Nachrichten die Session durch den CSSPDirector abgebrochen. Es ist möglich, einen Time-out-Verzögerungsfaktor einzustellen, um die Anzahl der Nachrichten, deren Verlust toleriert wird, zu verändern. Es wird eine Pause von

$$TimoutVerzoegerungsfaktor * CSSPPeriodendauer + 1/2 * CSSPPeriodendauer$$

toleriert. Die CSSPPeriodendauer bezeichnet hierbei die Zeit zwischen zwei ausgesendeten CSSP Nachrichten. Es wird eine halbe Periodendauer zusätzlich toleriert, um nicht eine kurz verzögerte Nachricht als Verlust anzunehmen. Allerdings wird hierbei jeder Sessionteilnehmer einzeln betrachtet. Falls ein einzelnes Fahrzeug diese Zeit ohne ankommende Nachrichten überschreitet, wird die Session abgebrochen.

4.3 Verhandlung

Die Verwaltung von Verhandlungsrunde und Verhandlungsmechanismus wurde im CSSPWishSessionManager implementiert. Dies ist eine Komponente des CSSPDirectors, die als abstrakte Grundklasse die grundlegenden Aufgaben implementiert. Dazu gehört vor allem die Verwaltung der Verhandlungsrunde und die Vorverarbeitung und Filterung von empfangenen Wünschen. Beim Spezialisieren dieser Klasse muss das Verarbeiten, Verwalten und Senden von aktuellen Wünschen implementiert werden, um einen konkreten Verhandlungsmechanismus abzubilden. Dadurch kann die Verhandlung leicht angepasst beziehungsweise ausgetauscht werden. Ich habe die restriktivere Variante der Verhandlung implementiert, die ich aus dem IMAGinE Entwurf entwickelt habe. Es ist also pro Session immer nur eine Verhandlung parallel möglich.

4.4 Platooning

Die funktionsspezifische Implementierung des Platoonings habe ich in einer Klasse unter Verwendung des CSSPController Interfaces implementiert. Diese besteht vor allem aus dem im Konzept Kapitel beschriebenen Zustandsautomaten, den Bedingungen, die für einen Zustandswechsel erfüllt sein müssen, und einiger Steuerlogik, die notwendig ist, um das am Anfang dieses Kapitels beschriebene Szenario richtig ablaufen zu lassen.

Den Zustandsautomaten habe ich mit Hilfe der Bibliothek „Stateful for Kotlin“ [44] implementiert. Sie stellt Funktionalitäten bereit um Zustandsautomaten und zugehörige Ereignisse zu implementieren. Damit habe ich die Zustände und dessen erlaubte

Übergänge für Platooning abgebildet. Zusätzlich habe ich zwei Arten von Ereignissen definiert. Die eine Art der Ereignisse löst einen Zustandsübergang in einen festgelegten Zielzustand aus, der aber nur dann ausgeführt wird, wenn der entsprechende Übergang im Zustandsautomat vorgesehen ist. Die anderen Ereignisse bewirken eine Prüfung der Bedingungen, die nötig sind um den Zustandswechsel zuzulassen. Dadurch können eingehende Wünsche nach einem neuen Zustand ausgewertet werden, indem man ein entsprechendes Stateful for Kotlin Ereignis im Zustandsautomaten auslöst. Das Ergebnis steht anschließend im Ereignisobjekt. Damit der CSSPController für Platooning auch die Initiative ergreifen kann, besitzt jeder Zustand eine Methode, in der eine Prüfung der Situation erfolgt und falls nötig einen Wunsch zurückgibt. So konnte ich leicht verschiedenes Verhalten für die verschiedenen Zustände implementieren, indem diese Methode regelmäßig auf dem aktiven Zustand aufgerufen wird.

Ich habe die gesamte Implementierung strukturell realistisch gestaltet, um die Umsetzbarkeit des Konzepts zu unterstreichen.

5 Evaluierung

In diesem Kapitel stelle ich die Ergebnisse meiner Auswertung des CSSP Protokolls hinsichtlich Datenbedarf und Robustheit gegenüber Störungen dar. Ich berechne zunächst die Funkkanallast, um die Anforderungen, die 802.11p an das CSSP Protokoll stellt, zu überprüfen. Anschließend evaluiere ich die Robustheit von CSSP bei degradierenden Kommunikationsbedingungen.

5.1 Funkkanallast

Die Berechnung der Funkkanallast in verteilten ad-hoc Netzwerken ist durch die nicht strukturierte Umgebung (z.B. bei überlappenden Funkbereichen) komplex. Um dies zu verdeutlichen werde ich zunächst auf aktuelle Forschung von Shagdar et al. [45] eingehen. Darin wurde die Erfolgschance eines Platoon in hoher Verkehrsdichte betrachtet. Hierfür wurde zunächst ein theoretisches Modell aufgestellt, um die Erfolgsrate für das Ankommen von CAM Nachrichten innerhalb eines Platoons zu bestimmen. Anschließend wurde dieses Modell durch eine Simulation evaluiert. Eine Erkenntnis aus dieser Arbeit ist, dass in solch einer Situation die Wahrscheinlichkeit, dass CAM Nachrichten ankommen, teilweise nur 20 Prozent beträgt. Des Weiteren wurde festgestellt, dass durch den Capture Effekt [46] ein Platoon in dieser Situation trotzdem mehr als 65 Prozent Erfolgschance haben kann. Der Effekt beschreibt, wie Nachrichten trotz hoher Interferenz im Funkkanal bei manchen Fahrzeugen mit höherer Wahrscheinlichkeit ankommen. Um eine Funkkanallast bei der Nutzung von 802.11p zu analysieren, müssen also diverse Effekte berücksichtigt werden. Im Rahmen dieser Arbeit nutze ich daher eine Abschätzung, ob die Datenrate des CSSP Protokolls in einem üblichen Bereich für Applikationen liegt, die eine V2X-Kommunikation über den 802.11p Standard nutzen. Dafür werde ich die Kommunikation des CSSP Protokolls mit der Kommunikation des

Cooperative Awareness Basic Service [25] vergleichen. Dieser stellt Informationen eines Fahrzeugs für andere Fahrzeuge in der Umgebung über die CAM Nachricht bereit. Nicht betrachtet werden verschiedene Umgebungsszenarien mit ihrer Wechselwirkung auf verschiedene kooperative Funktionen.

Die Nachricht des CSSP Protokolls muss in einem regelmäßigen Intervall von einem Teilnehmer einer Session ausgesendet werden. Dies ist nötig, da nur ein einzelner Nachrichtentyp existiert. Die Nachricht dient also neben der Verhandlung auch der Synchronisationsüberwachung. Der „Cooperative Awareness Basic Service“ sendet ebenfalls regelmäßig eine Nachricht aus, nämlich die CAM Nachricht. CAM Nachrichten werden mit bis zu zehn Hertz in einem Kanal des Frequenzbereichs ITS-G5A ausgesendet, der gleichen Frequenz, die CSSP verwendet. Dieser Bereich ist für sicherheitsrelevante Anwendungen vorgesehen, wäre also auch der Frequenzbereich für das CSSP Protokoll. Die zugehörigen Kanäle haben eine standardmäßige Datenrate von 6 oder 12 Mbit pro Sekunde [10]. Die Frequenz für das Aussenden von CAM Nachrichten kann bei drohender Überlastung des Kanals gedrosselt werden. Dies verbietet sich für die CSSP Nachrichten, da sonst eine periodische Überwachung der anderen Teilnehmerzustände nicht mehr möglich wäre.

Die reine Datenmenge bei der aktuellen Implementierung einer CSSP Nachricht ist im Grundzustand bei einer Session-Teilnehmerzahl von sechs Fahrzeugen 136 Byte groß. Mit Grundzustand ist die Nachricht für das Aussenden des aktuellen Zustands ohne einen Wunsch gemeint. Falls ein Wunsch angehangen wird, vergrößert sich die Datenmenge auf 196 Byte.

Javed et al. [47] haben festgestellt, dass durch die benötigten Sicherheitsmaßnahmen der V2X-Kommunikation noch einmal 106 Byte für die Signierung auf einen Payload von 100 Byte hinzukommen würden. Die Datenmenge dieser Signatur ist konstant, da nach Javed et al. [47] die Signierung auf einem Public Key Algorithmus basiert, der auf einen „Message Digest“ der Nachricht angewandt wird. Dieser Message Digest wird durch eine Hash Funktion wie z. B. SHA-256 erstellt.

Nach Kloiber et al. [48] beträgt der Overhead der Kommunikation beim Versenden von CAM Nachrichten, durch unter anderem den Header, 39 Byte. Wenn ich die gleiche Größe für CSSP Nachrichten annehme und alle Bestandteile aufaddiere, ergibt sich eine Gesamtgröße der CSSP Nachricht bei sechs Sessionteilnehmern inklusive Wünschen von unter 350 Byte. Dies entspricht also einer Datenrate von 28 Kbit pro Sekunde und Fahrzeuge bei einer Frequenz von 10 Hz. Angenommen einer Netto-Datenrate von 6 Mbit pro Sekunde für CSSP lassen sich selbst bei dichtem Verkehr bis zu 200 Fahrzeuge in einer Session abbilden. Da aber in der Realität die Datenrate keineswegs alleine ausreicht um die Funkkanallast zu bestimmen, werde ich nun den Vergleich mit den CAM Nachrichten fortsetzen. Das Standardisierungsinstitut ETSI gibt eine Größe von 400 Byte für die Simulation von CAM Nachrichten an [49]. Da die CSSP Nachrichten also eindeutig kleiner sind als die CAM Nachrichten, ergeben sich also bessere Lastverhalten im Vergleich zum Aussenden von CAM Nachrichten. Zusätzlich befinden sich im Gegensatz zur CAM nicht immer alle Fahrzeuge in einer kollaborativen Funktion, senden also nicht durchgehend CSSP Nachrichten aus.

5.2 Robustheit

Um die Robustheit bei degradierenden Kommunikationsbedingungen zu evaluieren, habe ich eine Metrik nach Sawade et al. [39] verwendet. Dafür wird die Wahrscheinlichkeit betrachtet, mit der eine Nachricht von einem anderen Fahrzeug empfangen wird. Diese Ereignisse für das Empfangen einer Nachricht sind unabhängig voneinander. Die Metrik M besteht aus dem Verhältnis zwischen stabilen und instabilen Session-Situationen. Instabile Session-Situationen entstehen, indem genügend Nachrichten von einem anderen Fahrzeug nicht empfangen werden und somit eine Session von einem Teilnehmer abgebrochen wird. Ab diesem Moment wird die Session als instabil bezeichnet. Die anderen Fahrzeuge, falls sie sich noch in der Session befinden, werden jetzt nach und nach die Session verlassen, weil das Fahrzeug, das die Session abgebrochen hat, keine Nachrichten für diese Session aussendet. Nun versuchen die Fahrzeuge eine neue Session aufzubauen. Erst wenn dies erreicht ist, wird die Session wieder als stabil bezeichnet. Das Verhältnis für stabile Sessions, wird aus dem Quotient von Zeit in stabiler Session zu Zeit in instabiler Session gebildet und das Verhältnis für instabile Sessions äquivalent aus dem Kehrwert:

$$M_{\text{stabil}} = \frac{t_{\text{stabil}}}{t_{\text{instabil}}} \qquad M_{\text{instabil}} = \frac{t_{\text{instabil}}}{t_{\text{stabil}}}$$

Um dieses Verhältnis zu bestimmen, verwende ich das gleiche minimale Szenario wie Sawade et al. [39], in welchem sich zwei Fahrzeuge in einer Platooningsession verbinden. Dabei sind zwei Faktoren einstellbar. Die Wahrscheinlichkeit, mit der die Nachrichten bei einem Fahrzeug ankommen und der Time-out-Verzögerungsfaktor, der bestimmt, wie viele nicht empfangene Nachrichten geduldet werden. Nach der Aufzeichnung und Auswertung von circa 150 Simulationsstunden ergibt sich der Graph in Abbildung 6.

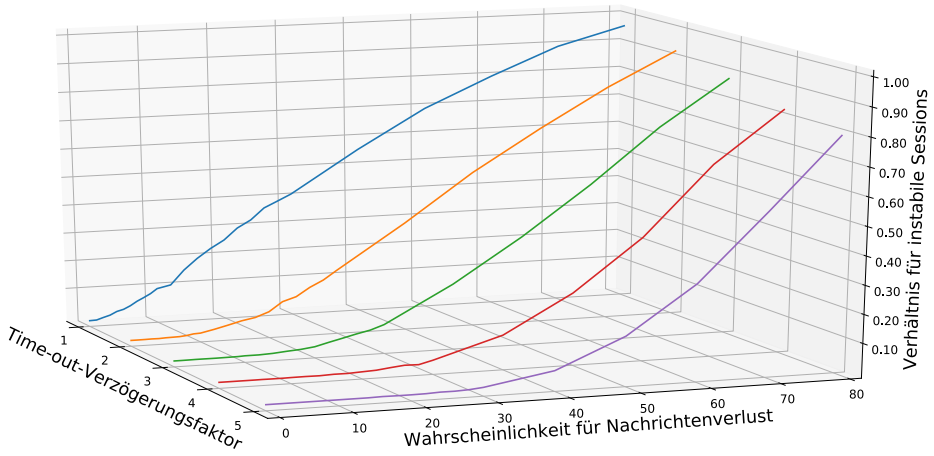

Abbildung 6: Verhältnis von instabilen zu stabilen Sessionsituationen

Für die Auswertung habe ich den Time-out-Verzögerungsfaktor bis zu einem Wert von fünf verwendet, der bereits einem Ausfall von mehr als 500 Millisekunden entspricht. In einem Platoon auf der Autobahn bei 40 Metern pro Sekunde entspricht dies bereits einer Fahrt von mehr als 20 Metern ohne definierten Session-Status. Pro Messpunkt habe ich jeweils 60 Minuten in der Simulation aufgezeichnet, die beginnen, sobald die Fahrzeuge das erste Mal erfolgreich eine Session aufbauen. Dabei sind die Fahrzeuge permanent in Reichweite zueinander. Die Tabelle mit den aufgezeichneten Messungen kann in Anhang D eingesehen werden.

Abbildung 6 verdeutlicht den Einfluss des Time-out-Verzögerungsfaktors. Mit sinkendem Faktor sinkt auch die Toleranzschwelle des CSSP Protokolls für den maximalen Nachrichtenverlust. Dadurch erhöht sich die Instabilität der Session. Gleichzeitig reagieren die Sessionteilnehmer auch schneller auf Kommunikationsabbrüche anderer, nehmen diese aber auch häufiger an, obwohl in Wirklichkeit nur kurz die Verbindung gestört ist. Das Gegenteil ist der Fall, wenn der Time-out-Verzögerungsfaktor erhöht wird.

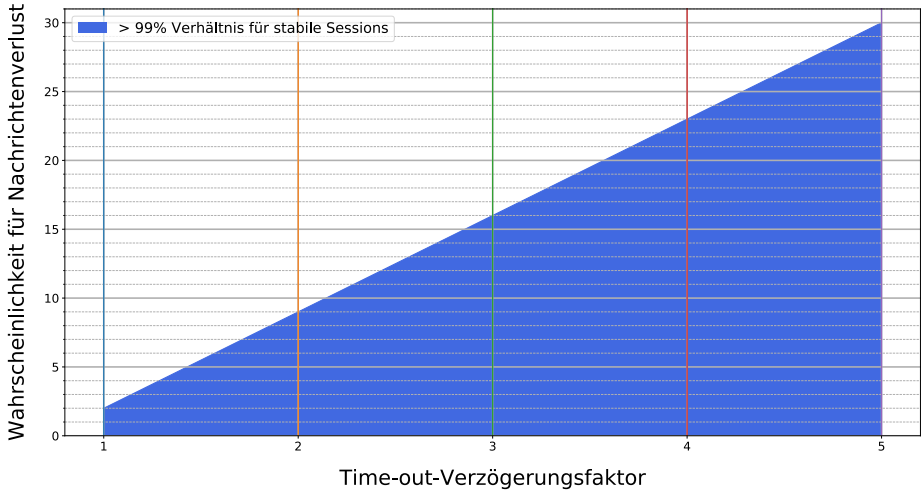

Abbildung 7: Stabiler Bereich

Ich nehme ebenso wie Sawade et al. [39] an, dass eine Toleranz für Desynchronisationen von 99 Prozent angemessen ist und bezeichne Sessions dann als stabil. Abbildung 7 verdeutlicht diesen stabilen Bereich und zeigt, dass dieser für einen Time-out-Verzögerungsfaktor von 1 bis 5 linear größer wird. Bei einem Faktor von 5 ist zu beachten, dass die Trägheit mit fünf geduldeten Nachrichten (550 Millisekunden) auf einen relativ hohen Wert steigt und damit eventuell nicht für jede kollaborative Funktion geeignet ist. Der Faktor 1 hingegen bietet keine hohe Robustheit (bis zu 2 Prozent Nachrichtenverlust), aber in zuverlässigen Kommunikationsumgebungen die schnellste Überwachung von Netzabbrüchen (150 Millisekunden). Somit erhalte ich je nach Konfiguration einen Bereich von 2 bis 30 Prozent Nachrichtenverlust ohne instabil zu werden. Wenn ein mittlerer Time-out-Verzögerungsfaktor von 3 gewählt wird, erhalte ich eine Toleranz von bis zu 16 Prozent Nachrichtenverlust in einer Session von zwei Fahrzeugen. So vergehen 350 Millisekunden, bis ein Verbindungsverlust realisiert

wird.

Um die Bedeutung einer stabilen Session genauer zu analysieren, habe ich die Anzahl der maximalen Sessionabbrüche im stabilen Bereich pro Time-out-Verzögerungsfaktor betrachtet. Tabelle 1 veranschaulicht, dass für einen Faktor von 1 und 2 knapp über einmal pro Minute die Session abgebrochen wird. Im restlichen Bereich brechen die Sessions weniger als einmal die Minute ab. Weitere Messungen ergaben, dass die Zeit, die für den Wiederaufbau einer Session benötigt wird, abhängig von Time-out-Verzögerungsfaktor und Nachrichtenverlust ist. Diese liegt im stabilen Bereich in der Regel unter einer Sekunde.

Tabelle 1: Maximale Sessionabbrüche im stabilen Bereich

Time-out-Verzögerungsfaktor	Wahrscheinlichkeit für Nachrichtenverlust in Prozent	Verbindungsabbrüche pro Stunde
1	2	71
2	9	62
3	16	48
4	23	36
5	30	39

Falls sich mehr als zwei Teilnehmer in einer Session befinden, sollte die Wahrscheinlichkeit für instabile Sessions ansteigen, da jeder Teilnehmer unabhängig beobachtet wird.

Es bietet sich an, den Time-out-Verzögerungsfaktor nicht allgemein festzulegen. Er sollte für die unterschiedlichen Funktionen, die über CSSP verwendet werden, unterschiedliche Werte annehmen können, denn Platooning stellt andere Anforderungen als z. B. ein kollaborativer Spurwechsel. Die Funktionen sollten den aktuellen Trägheitswert mit einbeziehen und könnten sich daran anpassen, indem sie z. B. mehr Abstand halten, falls dies dadurch nötig wird. Idealerweise müsste es möglich sein den Verzögerungsfaktor dynamisch an die Kommunikationsbedingungen anzupassen. Dafür könnte dieser wie ein Wunsch zwischen den Teilnehmern verhandelt werden.

Nach Shagdar et al. [45] ist neben der Nachrichtenkollision, also dem Verlust von Nachrichten, auch die Verzögerung ein entscheidender Faktor für die Leistung in einem V2X-Kommunikationssystem. Allerdings haben sie in ihrer Simulation für die Kommunikation in einem Platoon ermittelt, dass CAM Nachrichten nicht länger als ein paar Millisekunden verzögert werden. Ich nehme aufgrund der Ähnlichkeiten zwischen dem Aussenden von CAM Nachrichten und dem Aussenden von CSSP Nachrichten an, dass die Verzögerung ähnlich ist. Dies sollte also kaum Einfluss auf das CSSP Protokoll haben.

6 Zusammenfassung und Ausblick

Diese Arbeit befasst sich mit der kooperativen Fahrstrategieplanung zwischen automatisierten Fahrzeugen. Diese Ebene der Planung ermöglicht es Fahrzeugen eine längerfristige Kooperation herzustellen. Sie ergänzt die taktische Kooperation, welche sich für kurzfristige Planung eignet.

In dieser Arbeit habe ich das Collaborative Strategic Session-based Planning Protokoll vorgestellt. Dies ist ein funktions-generisches Protokoll für die kooperative Fahrstrategieplanung über den Funkstandard IEEE 802.11p. Das CSSP Protokoll basiert auf Vorarbeiten aus dem IMAGinE Forschungsprojekt. Es ermöglicht die strategische Ebene für kooperative Fahrfunktionen zu realisieren, ohne dabei auf einen vordefinierten Satz von Fahrfunktionen beschränkt zu sein. Dabei kommt das Protokoll mit einer einzelnen Nachricht aus, welche periodisch ausgesendet wird. Das Protokoll basiert auf Sessions, in denen sich Autos für spezifische Funktionen zusammenfinden können. Die Fahrstrategie wird über einen verteilten Zustandsautomaten festgelegt. Das Protokoll sieht Verhandlungen zwischen den Session-Teilnehmern vor, um gemeinsam in einen abgestimmten neuen Zustand zu wechseln. Dafür ist eine vollständige Zustimmung aller Teilnehmer nötig. Der Vorschlag eines neuen Zustands ebenso wie die Zustimmung erfolgt über „Wünsche“, die an die Session angehängt werden können.

Das von mir vorgestellte Konzept habe ich implementiert und in einer Simulationsumgebung evaluiert. Dies ist anhand einer exemplarischen Platooning-Funktion erfolgt. Ein Vergleich der erzeugten Datenlast mit dem Cooperative Awareness Basic Service aus dem Standard ETSI ITS G5 hat ergeben, dass es möglich ist, die CSSP Nachrichten wie vorgesehen in einem 802.11p Funkkanal zu versenden. Ein wesentlicher Bestandteil meiner Auswertung ist die Analyse des Verhaltens gegenüber Nachrichtenverlust. Da die Verhandlungen in einem ad-hoc Funknetzwerk stattfinden, ist eine Robustheit gegenüber solchen Störungen essentiell für die sichere Funktion eines Kooperationsprotokolls. Das Protokoll sieht einen periodischen Nachrichtenversand vor, der von allen Fahrzeugen zur Überwachung der Synchronizität und des Funktionsstatus genutzt wird. Ein wichtiger Parameter ist hier die Toleranz gegenüber verlorenen Nachrichten, also nach wie vielen Nachrichtenverlusten in Reihe das Protokoll abbricht. Meine Auswertung hat ergeben, dass je nach Einstellung der durchschnittliche Nachrichtenverlust 2 Prozent (geringe Toleranz) bis 30 Prozent (hohe Toleranz) betragen kann, um die Sessions zu über 99 Prozent stabil zu halten. Durch erhöhte Toleranz wird jedoch die Erkennung von einem Kommunikationsabbruch langsamer. Diese Trägheit liegt je nach Einstellung zwischen 150 und 550 Millisekunden und sollte auf die aktive Funktion angepasst werden.

Das vorliegende Konzept ist erweiterbar. Im Moment ist es z. B. nicht möglich eine Verhandlungsrunde im Vorhinein zu beenden. Es muss stets abgewartet werden, bis das Time-out eines Wunsches abgelaufen ist, um einen Wunsch zu verhandeln. Dies könnte zeitkritische Entscheidungen verhindern. Man könnte aber eine Ablehnung des verhandelten Wunsches durch einen Wunsch für den aktuellen Zustand ermöglichen. Sobald ein Teilnehmer ablehnt, könnten die anderen Teilnehmer die Ablehnung übernehmen, weil die vollständige Zustimmung für einen Zustandswechsel benötigt wird. So könnte

man die Runde vorzeitig beenden, sobald alle Teilnehmer eine Ablehnung aussenden. Es kommt allerdings zu einem Problem, falls ein Teilnehmer das vorzeitige Ende der Verhandlung durch Kommunikationsstörungen nicht erfasst hat. Dieser würde dann keine neuen Verhandlungen akzeptieren, bis das Time-out der Verhandlung abgelaufen ist. Dieses Problem könnte man lösen, indem der Zähler für Zustandsübergänge um eins erhöht wird, wenn die Teilnehmer die Verhandlung durch Ablehnung beenden. So könnte ein Teilnehmer, der die Ablehnung verpasst hat, ebenfalls feststellen, dass es zu einem Ergebnis der Verhandlung gekommen ist.

Des Weiteren könnte man einen Vorteil aus einer zweiten niedrigeren Grenze für Kommunikationsprobleme ziehen, die nicht sofort die Session beendet, sondern stattdessen die aktuelle Funktion über die Instabilität benachrichtigt. Das Protokoll sieht im Moment vor, dass eine Session aus Sicherheitsgründen abgebrochen wird, sobald eine gewisse Anzahl an CSSP Nachrichten von einem Teilnehmer nicht empfangen wurde. Die zweite niedrigere Grenze könnte die aktuelle Funktion in einen besonderen Zustand versetzen anstatt die Session zu beenden. Diese könnte dann extra Sicherheitsmaßnahmen, wie z. B. erhöhten Abstand zum vorausfahrenden Fahrzeug ergreifen. Dies würde allerdings zusätzlichen Aufwand für die Definition und Implementierung der funktionsspezifischen Teile der Funktionen erfordern. Es könnte auch sein, dass es nicht immer möglich ist, einen solchen vorsichtigeren Zustand anzunehmen. Durch diese Erweiterung könnte die harte Grenze für das Beenden der Session nach hinten verlagert werden.

Aktuell wird das Problem der byzantinischen Generäle [35] nicht im CSSP Protokoll berücksichtigt. Dabei geht es um eine Einigung über Kommunikation alleine. Allerdings ist dabei unklar, welche Nachrichten wahrheitsgemäße Informationen beinhalten. Ein bösartiger Akteur in einer CSSP Session könnte einen willkürlichen Zustand mit einem höheren Zähler für Zustandswechsel als aktuellen Zustand aussenden. Die anderen Teilnehmer seiner Session würden annehmen, dass sie einen Wechsel in den neuen Zustand verpasst haben. Momentan kann das CSSP Protokoll lediglich jegliche Folgen dieses Angriffs verhindern, weil ein vollständiger Konsens für Zustandswechsel benötigt wird. Dadurch kann jede Synchronisation wie ein Wunsch eines anderen Fahrzeugs ausgewertet werden und jeder Teilnehmer kann leicht überprüfen, ob er dem Wunsch nach einem zu synchronisierenden Zustand zugestimmt hatte. Wenn nicht, kann der Teilnehmer die Session verlassen und wird nicht in einen unsicheren, ungewollten Zustand gezwungen. Allerdings ist dann die Session unbrauchbar.

Es ist möglich, das in dieser Arbeit vorgestellte Protokoll weiterzuentwickeln, um Verhandlungen nicht nur auf Basis einer vollständigen Zustimmung auszuführen. Stattdessen könnte man Mehrheitsentscheide verwenden, bei der nur ein gewisser Anteil an Zustimmung nötig ist. Dies würde eine signifikante Erhöhung der Protokollkomplexität voraussetzen, wäre jedoch vor allem bei größeren Teilnehmergruppen von Vorteil. Der Synchronisationsmechanismus müsste dann unter anderem auch komplett ersetzt werden. Stattdessen wäre z. B. eine eigene Synchronisationsschicht oder ein geschicktes Konzept an Signaturen möglich.

Der aktuelle Standard für V2X-Kommunikation ETSI ITS G5 bietet momentan noch keine Möglichkeiten für echte Kollaboration. Das CSSP Protokoll stellt ein simples

Konzept zur Erweiterung dieses Standards dar, ohne unzählige Nachrichten zu etablieren und komplexe Algorithmen einzusetzen. Trotz dieser Einschränkungen wird über CSSP eine fahrstrategische Kollaboration ermöglicht.

A CSSPMessage

```
package com.dcaiti.phabmacs.c2x.CSSP

import com.dcaiti.phabmacs.c2x.C2XMessage
import com.dcaiti.phabmacs.c2x.StationID
import com.dcaiti.phabmacs.apps.platooning17.controller.
    CSSPPlaRoleNames
import com.dcaiti.phabmacs.apps.platooning17.controller.
    CSSPPlaStateName
import com.dcaiti.phabmacs.sim.GeoLocation
import java.time.LocalDateTime
import java.time.LocalTime
import java.util.*
import kotlin.properties.Delegates

/**
 * Message for strategic cooperation over C2X
 * with the CSSP protocol
 *
 * @author J.Pfeifer
 * @created 7/16/17.
 */
class CSSPMessage(senderLocation: GeoLocation,
                  senderID: Long, messageID: Long,
                  val header: Header, val data: Data?,
                  val wish: Wish? = null)
    : C2XMessage(senderLocation, senderID, messageID) {
    // set the message type to CSSP Message
    override val type: MessageType = MessageType.CSSPM

    /**
     * The CSSPMessage Header class includes
     * everything to fully specify a CSSP session
     */
    data class Header(val csspVersion: Int,
                      val sessionId: SessionID,
                      val functionCode: Int,
                      val functionVersion: Int)

    /**
     * The CSSPMessage Data class includes
     * the current session state data
     */
    data class Data(val currentState: Short,
                    val participants: List<Participant>,
                    val changeCount: Int) {
        override fun toString(): String {
            return "Data(currentState=${CSSPPlaStateName.values()[
                currentState.toInt()]}, participants=$participants,
                changeCount=$changeCount)"
        }
    }

    /**
     * The CSSPMessage Wish class includes everything
```

```
     * needed to negotiate about session state changes
     */
    data class Wish(val voteTimeout: LocalDateTime,
                    val state: Short,
                    val participants: List<Participant>) {
        var wishID: WishID by Delegates.notNull()
    }

    /**
     * A Session participant has a [stationId] for
     * identification and a [role] in respective
     * to the active function
     */
    data class Participant(val stationId: Long,
                           val role: Short) {
        override fun toString(): String {
            return " ID$stationId = ${CSSPPlaRoleNames.values()[role.
                toInt()]} "
        }
    }

    /**
     * Use this message to check if a received
     * message is a function request
     */
    fun isRequest(self: StationID): Boolean {
        return !hasData() && wish != null && wish.participants.map(
            Participant::stationId).contains(self)
    }

    fun hasWish() = wish != null

    fun hasData() = data != null
}

/**
 * 64bit stationId of initiating vehicle + 64bit timestamp
 */
typealias SessionID = UUID
typealias WishID = UUID
```

B CSSPDirector

```
package com.dcaiti.phabmacs.c2x.CSSP

import com.dcaiti.phabmacs.apps.platooning17.PlaCar
import com.dcaiti.phabmacs.c2x.StationID
import com.dcaiti.phabmacs.c2x.utils.simulationTime
import com.dcaiti.phabmacs.sim.GeoLocation
import java.time.Duration

/**
 * Generic director for strategic cooperation
 * over C2X using the CSSP protocol
```

```
 *
 * @author J.Pfeifer
 * @created 7/16/17.
 */
interface CSSPDirector {
    val self: StationID
    val vehicle: PlaCar
    val FREQUENCY: Int
    val csspVersion: Int
    // currently active session
    val currentSession: SessionID?
    // currently active function
    val currentFunction: Function?
    // current function state
    val currentStateData: CSSPMessage.Data?
    // used header for current session
    val currentHeader: CSSPMessage.Header?
    val vehicleLocation: GeoLocation
    // currently active controller for executing function
    val currentFunctionController: CSSPController?

    /**
     * Checks if a function is supported by this CSSPDirector
     */
    fun functionSupported(function: Function): Boolean

    /**
     * Send out a function/session request. The targeted
     *  vehicles have to be participants in the [wish]
     * If no [sessionID] is given, a new one is created
     */
    fun sendFunctionRequest(wish: CSSPMessage.Wish,
                function: Function,
                sessionID: SessionID =
                    SessionID(vehicle.id, System.currentTimeMillis()))

    /**
     * Use this function to send a state wish to the active
     * session participants.
     * If [wish].state is -1 it is equal to no state wished.
     * So a dissolving of the session is wanted
     */
    fun sendStateWish(wish: CSSPMessage.Wish)

    /**
     * Used to process everything besides newly incoming
     * messages. Simulator thread calls this method
     * e.g. sending out current state
     */
    fun process()

    companion object {
        // used to get a wish signaling the
        // intention to dissolve the session
        val WISH_TO_DISSOLVE
            get() =
```

```
                CSSPMessage.Wish(simulationTime() + Duration.ofSeconds
                    (1), -1, listOf())
    }
}

/**
 * [functionCode] and [functionVersion] describe together the
 * specific cooperative function
 */
data class Function(val functionCode: Int, val functionVersion: Int)
```

C CSSPController

```
package com.dcaiti.phabmacs.c2x.CSSP

import java.time.LocalDateTime
import java.time.LocalTime

/**
 * @author J.Pfeifer
 * @created 7/16/17.
 */
interface CSSPController {
    /**
     *  reference to the CSSP Director for protocol API
     */
    var csspDirector: CSSPDirector
    /**
     * the function this controller supports
     */
    val supportedFunction: Function

    /**
     * called when another vehicle wants to engage into a cooperative
     * function
     *
     * [message] received message with the function request
     */
    fun newFunctionRequest(message: CSSPMessage): Boolean

    /**
     * Called when another session participants expresses a new wish
     * to change state
     *
     * [message] received message with the wish to change state
     */
    fun newSessionWish(message: CSSPMessage): Boolean

    /**
     * Called when a wish to end a function and dissolve the session
     * is received
     */
    fun sessionWishToDissolveSession(): Boolean
```

```
    /**
     * Called when a new state has been successfully negotiated
     *
     * [stateData] data of the new state which was negotiated
     */
    fun stateUpdate(stateData: CSSPMessage.Data)

    /**
     * Called when a session is opened to negotiate if a function
     * should be executed together
     *
     * [currentSession] the [SessionID] which reflects the newly
     * joined trial session
     */
    fun sessionTemporarilyEstablished(currentSession: SessionID)

    /**
     * Called when all participants agree to execute a function
     * together
     *
     * [currentHeader] header used for each session message
     *     with all information to the newly established session
     * [currentStateData] the state in which the agreed upon
     *     function is started
     */
    fun sessionSuccessfullyEstablished(
                            currentHeader: CSSPMessage.Header,
                            currentStateData: CSSPMessage.Data)

    /**
     * Called when a session is dissolved, left or when a trial
     * session reaches its timeout
     *
     * [session] the [SessionID] from the session that ended
     */
    fun sessionEnd(session: SessionID?)

    /**
     * Called when a new wish session is started to negotiate
     * about a state change
     *
     * [currentWishID] the [WishID] for the newly started wish session
     * [voteTimeout] the timeout for the new wish session
     */
    fun newWishSession(currentWishID: WishID,
                       voteTimeout: LocalDateTime)

    /**
     * Called regularly to give the controller the opportunity
     * to take action e.g. initiate a state change through wish
     */
    fun checkForInitiativeAction()
}
```

D Messdaten

T Time-out-Verzögerungsfaktor
L Wahrscheinlichkeit für Nachrichtenverlust in Prozent
M_{stabil} Verhältnis für stabile Sessions
$M_{instabil}$ Verhältnis für instabile Sessions
t Gesamtzeit für den Messpunkt in Millisekunden
t_{stabil} Zeit in Session in Millisekunden
$t_{instabil}$ Zeit außerhalb der Session oder im Sessionaufbau in Millisekunden
B Anzahl Verbindungsabbrüche
W Durchschnittliche Zeit, die für den Sessionwiederaufbau benötigt wird in Millisekunden

Tabelle 2: Messdaten mit einem Time-out-Verzögerungsfaktor von 1

T	L	M_{stabil}	$M_{instabil}$	t	t_{stabil}	$t_{instabil}$	B	W
1	1	0.9991	9.0E-4	3600897	3597487	3410	7	487
1	2	0.9914	0.0086	3607673	3576573	31099	71	438
1	3	0.9843	0.0157	3607410	3550910	56500	124	455
1	4	0.9725	0.0275	3607432	3508082	99350	221	449
1	5	0.9661	0.0339	3607422	3484972	122450	272	450
1	6	0.9546	0.0454	3607468	3443718	163750	358	457
1	7	0.9397	0.0603	3607468	3390068	217400	468	464
1	8	0.9278	0.0722	3607512	3347212	260300	559	465
1	9	0.915	0.085	3607522	3300922	306600	656	467
1	10	0.8981	0.1019	3607504	3240004	367500	784	468
1	12	0.8876	0.1124	3607432	3202108	405324	915	442
1	14	0.8366	0.1634	3607330	3017980	589350	1235	477
1	16	0.7969	0.2031	3607420	2874900	732520	1508	485
1	18	0.7637	0.2363	3607542	2755142	852400	1733	491
1	20	0.736	0.264	3607448	2655248	952200	1904	500
1	22	0.6952	0.3048	3607290	2507640	1099650	2155	510
1	24	0.6688	0.3312	3607378	2412650	1194728	2325	513
1	26	0.628	0.372	3607200	2265200	1342000	2559	524
1	28	0.6059	0.3941	3607707	2185955	1421751	2636	539
1	30	0.5833	0.4167	3607249	2103999	1503249	2752	546
1	40	0.4353	0.5647	3607481	1570199	2037281	3278	621
1	50	0.3005	0.6995	3606671	1083799	2522871	3271	771
1	60	0.1962	0.8038	3607509	707699	2899809	2760	1050
1	70	0.1007	0.8993	3607715	363151	3244563	1799	1803
1	80	0.038	0.962	3607659	136999	3470659	799	4343

Tabelle 3: Messdaten mit einem Time-out-Verzögerungsfaktor von 2

T	L	M_{stabil}	$M_{instabil}$	t	t_{stabil}	$t_{instabil}$	B	W
2	1	1.0	0.0	3607349	3607349	0	0	0
2	2	0.9998	2.0E-4	3607465	3606865	599	1	599
2	3	0.9993	7.0E-4	3607539	3604839	2699	5	539
2	4	0.9996	4.0E-4	3607479	3605879	1599	3	533
2	5	0.9983	0.0017	3607541	3601241	6299	11	572
2	6	0.9978	0.0022	3607445	3599525	7919	13	609
2	7	0.9972	0.0028	3607367	3597167	10199	18	566
2	8	0.9948	0.0052	3607347	3588729	18617	34	547
2	9	0.9905	0.0095	3607687	3573587	34100	62	550
2	10	0.9893	0.0107	3607723	3569173	38549	67	575
2	12	0.9788	0.0212	3607695	3531295	76399	136	561
2	14	0.9669	0.0331	3607504	3487954	119550	209	572
2	16	0.9543	0.0457	3607438	3442438	165000	283	583
2	18	0.9453	0.0547	3607651	3410451	197199	330	597
2	20	0.9278	0.0722	3607494	3346896	260598	462	564
2	22	0.8945	0.1055	3607156	3226556	380600	644	590
2	24	0.8796	0.1204	3607711	3173463	434247	706	615
2	26	0.8492	0.1508	3607062	3063162	543900	884	615
2	28	0.8256	0.1744	3607426	2978376	629050	1007	624
2	30	0.7933	0.2067	3607306	2861656	745650	1180	631
2	40	0.6388	0.3612	3607236	2304136	1303100	1908	682
2	50	0.4725	0.5275	3607106	1704500	1902606	2407	790
2	60	0.3287	0.6713	3607228	1185650	2421578	2406	1006
2	70	0.1945	0.8055	3607653	701599	2906053	1965	1478
2	80	0.0781	0.9219	3605876	281600	3324276	1041	3193

Tabelle 4: Messdaten mit einem Time-out-Verzögerungsfaktor von 3

T	L	M_{stabil}	$M_{instabil}$	t	t_{stabil}	$t_{instabil}$	B	W
3	1	1.0	0.0	3607480	3607480	0	0	0
3	2	0.9998	2.0E-4	3607436	3606886	550	1	550
3	3	0.9995	5.0E-4	3607494	3605844	1650	3	550
3	4	0.9995	5.0E-4	3607364	3605564	1800	3	600
3	5	0.9996	4.0E-4	3607673	3606373	1300	2	650
3	6	0.9996	4.0E-4	3607562	3605962	1600	2	800
3	7	0.9993	7.0E-4	3607546	3605096	2450	4	612
3	8	0.9991	9.0E-4	3607340	3604040	3300	5	660
3	9	0.9989	0.0011	3607528	3603478	4050	6	675
3	10	0.9977	0.0023	3607474	3599124	8350	12	695
3	12	0.9976	0.0024	3600717	3592005	8712	14	622
3	14	0.9954	0.0046	3600565	3583953	16612	25	664
3	16	0.9911	0.0089	10816093	10720125	95966	143	671
3	18	0.9876	0.0124	10808696	10674710	133985	204	657
3	20	0.9814	0.0186	3607286	3540086	67200	97	692
3	22	0.969	0.031	3607452	3495474	111978	168	666
3	24	0.9574	0.0426	3607156	3453456	153700	219	701
3	26	0.9454	0.0546	3607378	3410478	196900	278	708
3	28	0.9349	0.0651	3606692	3371842	234850	323	727
3	30	0.9185	0.0815	3607352	3313264	294088	421	698
3	40	0.7903	0.2097	3607166	2850916	756250	971	778
3	50	0.6378	0.3622	3606546	2300264	1306282	1574	829
3	60	0.4676	0.5324	3606852	1686674	1920178	1947	986
3	70	0.2799	0.7201	3607671	1009899	2597771	1714	1515
3	80	0.1235	0.8765	3603640	444900	3158740	1076	2935

Tabelle 5: Messdaten mit einem Time-out-Verzögerungsfaktor von 4

T	L	M_{stabil}	$M_{instabil}$	t	t_{stabil}	$t_{instabil}$	B	W
4	1	1.0	0.0	3607637	3607637	0	0	0
4	2	1.0	0.0	3607429	3607429	0	0	0
4	3	1.0	0.0	3607431	3607431	0	0	0
4	4	1.0	0.0	3607477	3607477	0	0	0
4	5	1.0	0.0	3607339	3607339	0	0	0
4	6	0.9998	2.0E-4	3607505	3606805	699	1	699
4	7	1.0	0.0	3607447	3607447	0	0	0
4	8	1.0	0.0	3607413	3607413	0	0	0
4	9	1.0	0.0	3607519	3607519	0	0	0
4	10	0.9996	4.0E-4	3607637	3606237	1399	2	699
4	12	0.9993	7.0E-4	3607465	3605051	2413	3	804
4	14	0.9998	2.0E-4	3607503	3606803	699	1	699
4	16	0.9979	0.0021	3607437	3599837	7599	10	759
4	18	0.9981	0.0019	3607337	3600437	6899	9	766
4	20	0.9957	0.0043	3607443	3591953	15489	19	815
4	22	0.9934	0.0066	3607437	3583637	23799	30	793
4	23	0.9923	0.0077	3607653	3579853	27799	34	817
4	24	0.9876	0.0124	3607353	3562611	44741	54	828
4	26	0.9825	0.0175	3607533	3544433	63099	76	830
4	28	0.9836	0.0164	3607403	3548095	59307	71	835
4	30	0.9715	0.0285	3607379	3504579	102799	130	790
4	40	0.8996	0.1004	3607431	3245331	362099	414	874
4	50	0.7682	0.2318	3607417	2771117	836299	860	972
4	60	0.5893	0.4107	3607351	2125799	1481551	1308	1132
4	70	0.3516	0.6484	3607737	1268449	2339287	1410	1659
4	80	0.1767	0.8233	3607106	637250	2969856	1090	2724

Tabelle 6: Messdaten mit einem Time-out-Verzögerungsfaktor von 5

T	L	M_{stabil}	$M_{instabil}$	t	t_{stabil}	$t_{instabil}$	B	W
5	1	1.0	0.0	3607599	3607599	0	0	0
5	2	1.0	0.0	3607467	3607467	0	0	0
5	3	1.0	0.0	3607585	3607585	0	0	0
5	4	1.0	0.0	3607279	3607279	0	0	0
5	5	1.0	0.0	3607421	3607421	0	0	0
5	6	1.0	0.0	3607285	3607285	0	0	0
5	7	1.0	0.0	3607447	3607447	0	0	0
5	8	1.0	0.0	3607535	3607535	0	0	0
5	9	1.0	0.0	3607521	3607521	0	0	0
5	10	1.0	0.0	3607443	3607443	0	0	0
5	12	1.0	0.0	3607149	3607149	0	0	0
5	14	1.0	0.0	3607487	3607487	0	0	0
5	16	0.9989	0.0011	3607455	3603555	3899	4	974
5	18	0.9995	5.0E-4	3607307	3605607	1699	2	849
5	20	0.9995	5.0E-4	3607277	3605577	1699	2	849
5	22	0.999	0.001	3607187	3603567	3619	4	904
5	24	0.9969	0.0031	3607379	3596179	11199	13	861
5	26	0.9971	0.0029	3607427	3597027	10399	12	866
5	28	0.9951	0.0049	3607301	3589701	17599	20	879
5	30	0.9905	0.0095	3607367	3573267	34099	39	874
5	40	0.9494	0.0506	3607285	3424885	182399	177	1030
5	50	0.8469	0.1531	3607113	3055013	552099	488	1131
5	60	0.6828	0.3172	3607013	2462699	1144313	891	1284
5	70	0.4502	0.5498	3606917	1623817	1983099	1181	1679
5	80	0.2087	0.7913	3607595	752847	2854747	907	3147

Literatur

[1] Daimler, *Unternehmensgeschichte | Benz Patent-Motorwagen: Das erste Automobil (1885–1886)*. [Online]. Available: https://www.daimler.com/konzern/tradition/geschichte/1885-1886.html (Accessed: 10. 09. 2017).

[2] WELT. (2015). Immer mehr Erfindungen: Patente der Automobilhersteller - WELT, [Online]. Available: https://www.welt.de/motor/news/article144664614/Patente-der-Automobilhersteller.html (Accessed: 10. 09. 2017).

[3] H. Winner, S. Hakuli und G. Wolf, *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. 2009.

[4] J. B. Cicchino, „Effectiveness of Forward Collision Warning Systems with and without Autonomous Emergency Braking in Reducing Police-Reported Crash Rates“, *Insurance Institute for Highway Safety*, 2016.

[5] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein und E. Zeeb, „Making Bertha Drive-An Autonomous Journey on a Historic Route“, *IEEE Intelligent Transportation Systems Magazine*, Jg. 6, Nr. 2, S. 8–20, 2014.

[6] Waymo, *Journey*. [Online]. Available: https://waymo.com/journey/ (Accessed: 11. 09. 2017).

[7] Tesla, *Dual Motor Model S and Autopilot | Tesla Deutschland*, 2014. [Online]. Available: https://www.tesla.com/de_DE/blog/dual-motor-model-s-and-autopilot?redirect=no (Accessed: 09. 09. 2017).

[8] H. Zhu, K. V. Yuen, L. Mihaylova und H. Leung, *Overview of Environment Perception for Intelligent Vehicles*, Okt. 2017.

[9] A. Kemeny, „PROMETHEUS: Design technics“, in *Transportation Electronics, 1990. Vehicle Electronics in the 90's: Proceedings of the International Congress on*, IEEE, 1990, S. 201–207.

[10] *Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*, ETSI EN 302 663, 2012.

[11] Bundesministerium für Verkehr und digitale Infrastruktur. (2014). Cooperative ITS Corridor - Joint deployment, [Online]. Available: http://www.bmvi.de/SharedDocs/DE/Anlage/VerkehrUndMobilitaet/Strasse/flyer-eurokorridor-cooperative-its-corridor-in-deutsch.pdf?__blob=publicationFile (Accessed: 10. 09. 2017).

[12] R. Stahlmann, A. Festag, A. Tomatis, I. Radusch und F. Fischer, „Starting European Field Tests for Car-2-X Communication : the Drive C2X Framework“, in *18th ITS World Congress and Exhibition*, 2011.

[13] sim^{TD} Projekt. (2013). Deliverable D5.5 – TP5-Abschlussbericht – Teil A, [Online]. Available: http://www.simtd.de/index.dhtml/object.media/enEN/8154/CS/-/backup_publications/Projektergebnisse/simTD-TP5-Abschlussbericht_Teil_A_Manteldokument_V10.pdf (Accessed: 09. 09. 2017).

[14] Volkswagen AG. (2017). Mit dem Ziel, die Sicherheit im Straßenverkehr zu erhöhen, lässt Volkswagen Fahrzeuge ab 2019 miteinander kommunizieren,

[Online]. Available: https://www.volkswagenag.com/de/news/2017/06/pwlan.html (Accessed: 25. 09. 2017).

[15] S. Abuelsamid. (2016). New Cars Could Be Required To 'Talk' To Each Other As Soon As 2020, [Online]. Available: https://www.forbes.com/sites/samabuelsamid/2016/12/13/nhtsa-finally-issues-draft-v2v-communications-rule-could-be-mandatory-from-2021/#3dd734687581 (Accessed: 25. 09. 2017).

[16] W. H. Organization. (2017). Road traffic injuries, [Online]. Available: http://www.who.int/mediacentre/factsheets/fs358/en/ (Accessed: 10. 09. 2017).

[17] O. Sawade und I. Radusch, „Survey and classification of cooperative automated driver assistance systems", in *Proc. 82nd IEEE Vehicular Technology Conf.*, Boston, MA, USA, 2015, S. 1–5.

[18] Q. Xu und R. Sengupta, „Simulation, analysis, and comparison of ACC and CACC in highway merging control", in *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, IEEE, 2003, S. 237–242.

[19] C. Bergenhem, H. Pettersson, E. Coelingh, C. Englund, S. Shladover und S. Tsugawa, „Overview of platooning systems", in *Proceedings of the 19th ITS World Congress*, Vienna, Austria, Okt. 2012.

[20] U. Franke, F. Bottiger, Z. Zomotor und D. Seeberger, „Truck platooning in mixed traffic", in *Intelligent Vehicles' 95 Symposium., Proc.*, IEEE, 1995, S. 1–6.

[21] O. Sawade und I. Radusch, „Session-based communication over IEEE 802.11p for novel complex cooperative driver assistance functions", in *Proc. of the 21th ITS World Congress*, Detroit, USA, Sep. 2014, S. 1–11.

[22] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer und A. Kovacs, „Enhancements of V2X Communication in Support of Cooperative Autonomous Driving", *IEEE Communications Magazine*, Jg. 53, Nr. 12, S. 64–70, 2015.

[23] A. Festag, „Cooperative intelligent transport systems standards in Europe", *IEEE Communications Magazine*, Jg. 52, Nr. 12, S. 166–172, 2014.

[24] A. Festag, R. Baldessari, W. Zhang, L. Le, A. Sarma und M. Fukukawa, „CAR-2-X communication for safety and infotainment in Europe", *NEC Technical Journal*, Jg. 3, Nr. 1, S. 21–26, 2008.

[25] *Vehicular Communications ; Basic Set of Applications; Part 2 : Specification of Cooperative Awareness Basic Service*, ETSI EN 302 637-2, 2014.

[26] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*, ETSI EN 302 637-3, 2014.

[27] *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*, TS 102 636-5-1, 2011.

[28] L. Alvarez und R. Horowitz, „Safe platooning in automated highway systems", Institute of Transportation Studies, University of California, Berkeley, CA, USA, Tech. Rep. UCB-ITS-PRR-97-46, 1997.

[29] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, S. Member, W.-b. Zhang, D. H. Mcmahon, H. Peng, S. Sheikholeslam, S. Member und N. Mckeown, „Automatic Vehicle Control Developments in the PATH Program", *IEEE Trans. Veh. Technol.*, Jg. 40, Nr. 1, S. 114–130, Nov. 1991.

[30] G. Naus, R. Vugts, J. Ploeg, R. Molengraft und M. Steinbuch, „String-stable CACC design and experimental validation, a frequency-domain approach“, *IEEE Trans. Veh. Technol.*, Jg. 59, Nr. 99, S. 1, 2010.

[31] A. Dávila und M. Nombela. (2010). SARTRE: SAfe Road TRains for the Environment, [Online]. Available: `http://www.princeton.edu/~alaink/Orf467F10/PRT@LHR10_Conf/Arturo_Davila_prez_SafeTrains.pdf` (Accessed: 15. 09. 2017).

[32] C. Bergenhem, E. Hedin und D. Skarin, „Vehicle-to-Vehicle Communication for a Platooning System“, *Procedia - Social and Behavioral Sciences*, Jg. 48, S. 1222–1233, 2012.

[33] L. AARTS und G. FEDDES, „European Truck Platooning Challenge“, in *HVTT14: International Symposium on Heavy Vehicle Transport Technology, 14th, 2016, Rotorua, New Zealand*, 2016.

[34] IMAGinE. (2017). Imagine Projekt, [Online]. Available: `http://imagine-online.de/` (Accessed: 25. 09. 2017).

[35] L. Lamport, R. Shostak und M. Pease, „The Byzantine Generals Problem“, *ACM Transactions on Programming Languages and Systems*, Jg. 4, Nr. 3, S. 382–401, Juli 1982.

[36] M. J. Fischer, N. A. Lynch und M. S. Paterson, „Impossibility of distributed consensus with one faulty process“, *Journal of ACM*, Jg. 32, Nr. 2, S. 374–382, 1985.

[37] L. Lamport, „Generalized Consensus and Paxos“, Microsoft Research, Tech. Rep. MSR-TR-2005-33, 2005.

[38] H. Moniz, N. F. Neves und M. Correia, „Turquois: Byzantine consensus in wireless ad hoc networks“, in *IEEE/IFIP Proc. Int. Conf. Dependable Systems and Networks*, Chicago, IL, 2010, S. 537–546.

[39] O. Sawade, M. Schulze und I. Radusch, „Robust communication for cooperative driving maneuvers“, *IEEE Trans. Intell. Transp. Syst.*, to be published.

[40] D. Becker, A. Munjere, J. Einsiedler, K. Massow, F. Thiele und I. Radusch, „Blurring the border between real and virtual parking environments“, in *Intelligent Vehicles Symposium (IV)*, IEEE, Juni 2016, S. 1205–1210.

[41] JetBrains, *Kotlin Programming Language*. [Online]. Available: `https://kotlinlang.org/` (Accessed: 30. 09. 2017).

[42] Gradle Inc., *Gradle Build Tool*. [Online]. Available: `https://gradle.org/` (Accessed: 30. 09. 2017).

[43] E. Gamma, R. Helm, R. Johnson und J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[44] M. Krajewski, *Stateful for Kotlin*. [Online]. Available: `https://github.com/MiloszKrajewski/stateful4k` (Accessed: 02. 10. 2017).

[45] O. Shagdar, F. Nashashibi und S. Tohme, „Performance study of CAM over IEEE 802.11p for cooperative adaptive cruise control“, in *Wireless Days*, IEEE, März 2017, S. 70–76.

[46] M. Zorzi und R. R. Rao, „Capture and Retransmission Control in Mobile Radio“, *IEEE Journal Selected Areas Communications*, Jg. 12, Nr. 8, S. 1289–1298, 1994.

[47] M. A. Javed, E. B. Hamida und W. Znaidi, „Security in intelligent transport systems for smart cities: From theory to practice“, *Sensors*, Jg. 16, Nr. 6, S. 879, 2016.

[48] B. Kloiber, T. Strang und F. D. P. Müller, „Analytical Performance Considerations of IEEE 802.11p“, DLR, Tech. Rep. D2400.4.1, 2010.

[49] ETSI, „Intelligent Transport Systems (ITS); Cross Layer DCC Management Entity for operation in the ITS G5A and ITS G5B medium; Report on Cross layer DCC algorithms and performance evaluation“, Tech. Rep. TR 101 612, 2014.