DIPLOMARBEIT

Flexible Co-Simulationsumgebung zum Testen von Ladeinfrastrukturen für Elektromobilität

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

o.Univ.Prof. Dipl.-Ing. Dr.techn. Dietmar Dietrich
Dipl.-Ing. Mario Faschang

Dipl.-Ing. Dr.rer.nat. Johannes Stöckl (Austrian Institute of Technology)
Dipl.-Ing. Dr.techn. Friederich Kupzog (Austrian Institute of Technology)

am

Institut für Computertechnik (E384)
der Technischen Universität Wien

durch

Martin Nöhrer BSc
Matr.Nr. 0725589
8753 Fohnsdorf, Bahndammgasse 16/14

Fohnsdorf, am 14. Mai 2014 _____

Kurzfassung

Um die Umweltbelastung, vor allem die Emissionen in Städten, und die Abhängigkeit vom Energieträger Erdöl zu reduzieren, werden Elektrofahrzeuge in Zukunft eine wichtige Rolle im Verkehrssektor spielen. Durch eine höhere Marktdurchdringung der Elektromobilität könnte es zu Leistungsengpässen im elektrischen Verteilnetz kommen. Um dies zu verhindern, empfiehlt sich der Einsatz einer intelligenten Ladeinfrastruktur, die eine Koordinierung und Steuerung der Ladevorgänge von Elektrofahrzeugen durchführt. Für die Schaffung von steuerbaren und europaweit interoperablen Ladeinfrastrukturen werden in Zukunft Prüfsysteme benötigt, welche das korrekte Verhalten von Ladesäulen für Elektrofahrzeuge überprüfen. Gegenseitige Wechselwirkungen zwischen den Ladesäulen und den Elektrofahrzeugen müssen dafür analysiert werden. Diese Arbeit zeigt den Entwurf und die exemplarische Implementierung einer modularen Co-Simulationsumgebung, die in der Lage ist, gleichzeitig real aufgebaute und simulierte Ladesäulen in einer Umgebung zu vereinen. Dabei wird zum Austausch der Simulationsdaten auf den universell konfigurierbaren Simulation Message Bus gesetzt, der die Verbindung zwischen den ans System angeschlossenen Simulatoren herstellt. Ebenso wird auf das weit verbreitete Open Charge Point Protocol zurückgegriffen, um reale Ladesäulen in das System einbinden zu können. Die abschließende Untersuchung von Lademanagementalgorithmen, die unabhängig und lokal im Laderegler des Elektrofahrzeuges implementiert werden können, dient der Validierung der entworfenen Umgebung. Mit dieser Arbeit wird ein Grundstein für den späteren Ausbau einer Testinfrastruktur geschaffen, welcher bereits für verschiedene Zwecke eingesetzt werden kann. Neben der Untersuchung der Auswirkungen verschiedener Lademanagementstrategien, erlaubt die entworfene Umgebung Hardware-in-the-loop-Simulationen, um Wechselwirkungen und Beeinflussungen zwischen realen Ladesäulen beobachten zu können.

Abstract

In future, electrical vehicles will play an important role in reducing environmental pollution in the traffic sector, especially emissions in cities, and decreasing the dependence on fossil fuels. Increased market penetration of electric mobility can however cause congestions in electric power systems. Smart charging infrastructure that allows the coordination and control of many electric vehicles' charging processes is considered to avoid this problem. In order to successfully implement controllable and interoperable charging systems throughout Europe, test systems will be required to validate the correct behavior of charging stations for electric vehicles. Mutual interdependency of these charging stations and the vehicle itself has to be analyzed. This thesis presents the conceptual design and the exemplary implementation of a co-simulation framework that has the ability to simultaneously control real-world and simulated charging stations. To enable exchange of simulation data the universal configurable Simulation Message Bus will be used to establish communication channels between all connected simulation components within the system. Web services using the Open Charge Point Protocol will integrate the hardware-based charging stations into the simulation environment. The analysis of different charging strategies, which can be independently and locally implemented in the charging controller of an electric vehicle, is used to validate the designed concept. The implemented simulation system is a base for different future applications and can be used to build a hardware-in-the-loop simulation for testing the mutual influences of charging stations or for researching the effects of diverse charging algorithms.

Inhaltsverzeichnis

Abkürzungen

AC	Wechselstrom (alternating current)
ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
CO_2	Kohlenstoffdioxid
CP	Chargepoint
CS	Chargestation
DC	Gleichstrom (direct current)
EN	Europäische Norm
EXI	Efficient XML Interchange
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IP	Internet Protocol
JAXB	Java Architecture for XML Binding
JSON	JavaScript Object Notation
LAN	Local Area Network
Li-Ion	Lithium-Ionen
LR	Laderegler
LS	Ladestation
LTE	Long Term Evolution
NiMH	Nickel-Metallhydrid
NO_X	Stickoxide
OCHP	Open Clearing House Protocol
OCPP	Open Charge Point Protocol
OICP	Open InterCharge Protocol
ÖVE	Österreichischer Verband für Elektrotechnik
PLC	Powerline Communication
SCCP	Smart Charge Communication Protocol
SCL	Substation Configuration Language
SMB	Simulation Message Bus
SML	Smart Message Language
SOAP	veraltet: Simple Object Access Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UMTS	Universal Mobile Telecommunications System
WLAN	Wireless Local Area Network
WSDL	Web Services Description Language
XML	Extensible Markup Language

1 Einleitung

Keine andere Errungenschaft im letzten Jahrhundert beeinflusste den Menschen in so starker Weise als die Erfindung des Automobils. Johann-Günther König bringt dessen Einfluss als „wichtigstes Lebensmittel“ mit folgender Aussage auf den Punkt [Kön10, S.7]:

> „Es symbolisiert die Verheißung von grenzenloser individueller Mobilität, Macht über Zeit und Raum sowie hohe Geschwindigkeit. Es eignet sich als übersehbarer Bedeutungsträger für das Ausleben von Distinktionswünschen und zur Wahrung sozialer Hierarchien.“

Als primäres Antriebskonzept setzte sich dabei der mit Erdöl angetriebene Verbrennungsmotor durch. Ein erhöhtes Verkehrsaufkommen und eine große Anzahl an Individualfahrzeugen, die durch einen Verbrennungsmotor angetrieben werden, führen zu einigen Problemen, welche durch neue Antriebskonzepte eingedämmt werden müssen.

Der größte Ansporn, das Zeitalter der fossilen Brennstoffe hinter sich zu lassen und den Einsatz von alternativen Energieträgern voranzutreiben, ist sicherlich die *endliche Verfügbarkeit von Erdöl*. Dabei stellt sich nicht die Frage, ob in Zukunft noch genügend Ölressourcen vorhanden sein werden, sondern inwiefern die Versorgung mit kostengünstigem Erdöl gesichert ist. Heutzutage befindet sich der Rohöl- und Benzinpreis auf einem Höchststand und ist zirka viermal so hoch als noch in den 1970er Jahren [1], wo das günstige Benzin einer der Hauptgründe für den starken Anstieg des Individualverkehrs war. Da auch der asiatische Raum ein immer höheres Bedürfnis an fossilen Energieträgern entwickelt, wird dies in Zukunft den Importpreis von Rohöl weiter in die Höhe treiben [Hel09, S.7]. Diese Entwicklung ist ein global-ökonomisches Problem, da der Ölpreis massive Auswirkungen auf die Transportwirtschaft hat und somit auch die Preise anderer Wirtschaftsgüter weitgehend beeinflusst.

Ein weiteres Argument gegen fossile Treibstoffe ist die *Reduktion der Umweltbelastung*. Kraftfahrzeuge erzeugen durch die Verbrennung von Treibstoffen lokale Emissionen an Stickoxiden (NO_X), Feinstaub und Kohlenstoffdioxid (CO_2). Diese Schadstoffe können zu Allergien, Asthma oder Herzkreislauferkrankungen führen [Hel09, S.40]. Besonders in Ballungsräumen zählt der Verkehrssektor mit einem Anteil von 29 % an der gesamten produzierten Feinstaubmasse zu den größten Umweltverschmutzern [Hel09, S.51].

Das Verkehrsaufkommen, verursacht durch individuelle Fahrten mit dem eigenen Fahrzeug, führt bereits heute in vielen Regionen der Erde zu großen Schwierigkeiten. Besonders Städte sind davon betroffen, wo es durch den Individualverkehr zu einer Überfüllung und Platznot kommt. Die dabei

auftretenden Staubildungen fördern eine zusätzliche Schadstoff- und Lärmbelastung, die gesundheitsschädliche Folgen haben kann. Wie aktuelle Bevölkerungsveränderungen in Österreich zeigen [2], nimmt die Urbanisierung auch hierzulande weiterhin zu, wodurch diese Problematik sicherlich in Zukunft verstärkt wird.
Um diesen Individualverkehr zu reduzieren, bedarf es neuer Verkehrskonzepte, bei der es trotz Nutzung von öffentlichen Verkehrsmitteln nicht zu einer Einschränkung der „Individualmobilität von A nach B“ kommen darf [Lie12, S.11]. Dabei sind multimodale Verkehrskonzepte anzustreben, bei denen Personen je nach Situation unterschiedliche Verkehrsmittel benutzen [KVS13, S.67], um eine Fahrt mit dem eigenen Fahrzeug zu kompensieren, wodurch das Verkehrsaufkommen reduziert wird. Diese Konzepte könnten zum Beispiel durch den Ausbau von „Park & Ride“-Anlagen, die einen leichten Umstieg auf öffentliche Verkehrsmittel erlauben, oder durch die Ausweitung von „Carsharing“-Angeboten umgesetzt werden. Beide Konzepte könnten den Einsatz von alternativen Antriebstechnologien, hauptsächlich die Elektromobilität, fördern, da hier die Infrastrukturen und Konzepte zum Laden dieser Fahrzeuge sinnvoll geplant werden können.

Obwohl die Möglichkeit besteht, den Individualverkehr zu reduzieren und gemeinschaftliche Mobilitätskonzepte zu forcieren, wird kein Weg an alternativen Antrieben von Fahrzeugen vorbeiführen, da früher oder später fossile Energieträger nicht mehr wirtschaftlich förderbar sein werden. Eine derzeit vielversprechende Technologie um Verbrennungsmotoren in Zukunft zu ersetzen, ist sicherlich die Elektromobilität. Dabei bietet der Elektroantrieb einige Vorteile gegenüber konventionellen Verbrennungsmotoren. Er arbeitet lokal emissionsfrei und geräuscharm. Ebenso kann die elektrische Energie für den Antrieb aus verschiedensten Energieträgern, bevorzugt aus regenerativen Quellen, erzeugt werden, was wiederum zu einer Minderung der Schadstoffemission führt. Werden Elektrofahrzeuge mit dem heute typischen Strommix betrieben, ergibt sich bereits ein Einsparungspotential von 40 % CO_2 gegenüber den sparsamsten Benzinmotoren [Hel09, S.165]. Außerdem bietet der Elektroantrieb die höchste Gesamteffizienz, von der Quelle bis zum bewegten Rad („well to wheel“) [Hel09, S.110].

Trotzdem konnte sich der Elektroantrieb bis heute noch nicht durchsetzen, da für die Autonutzer die Einschränkungen derzeit überwiegen. Einer dieser Nachteile betrifft die Batterietechnologie, welche die Reichweite und das Gewicht der Fahrzeuge noch zu stark begrenzen. Die Übersicht der heutigen Fahrzeuge mit reinem Elektroantrieb [3] zeigt, dass diese eine durchschnittliche Reichweite von lediglich 100 – 150 km erreichen.

Auch in Zukunft wird der Individualverkehr einen wichtigen Stellenwert im gesamten Verkehrskonzept einnehmen. Der Elektromobilität werden nach heutigem Stand die größten Potentiale zum Lösen gesamtgesellschaftlicher Probleme, wie z. B. der Umweltbelastung in Städten, zugesprochen. Dieser Trend bildet die Basis für diese Diplomarbeit, mit welcher „ein weiterer Meter auf der noch langen Straße gebaut“ werden soll, um mit Elektrofahrzeugen in die Zukunft fahren zu können.

1.1 Problemumfeld

Grundsätzlich besteht zwischen dem Tank- bzw. Ladevorgang von Fahrzeugen je nach Antriebskonzept ein gravierender Unterschied. Bei Verbrennungsmotoren bildet der Tankvorgang einen abgeschlossenen Prozess mit geringer Beeinflussung der Außenwelt, da für jedes individuelle Fahrzeug nur eine Umlagerung von Treibstoff aus einem großen lokalen Tankbehälter an der Tankstelle, der als Zwischenspeicher dient, in den Fahrzeugtank erfolgt.

Das Laden von Elektrofahrzeugen kann diesbezüglich nicht als individueller und unabhängiger Vorgang betrachtet werden, da dies nur durch Anschluss an das elektrische Energienetz, welches keine Speichereigenschaft besitzt, erfolgen kann. Das Elektrofahrzeug ist aus der Sicht des Energienetzes ein zusätzlicher Strombezieher, welcher im Verbund mit vielen weiteren Verbrauchern steht, die sich gegenseitig beeinflussen können.

Unter den derzeit in Österreich zugelassenen 4,6 Millionen Personenkraftwagen werden knapp 12.000 entweder rein elektrisch oder mit elektrischem Zweitmotor angetrieben [4], was einem Anteil von gerade einmal 0,25 % entspricht. Die Durchdringungsrate von Elektrofahrzeugen in Österreich soll im Jahr 2030 bereits 40 % erreichen und sich bis ins Jahr 2050 auf 75 % erhöhen [PWL10, S.37]. Würden, hypothetisch betrachtet, alle derzeit zugelassenen Personenkraftwagen in Österreich rein elektrisch betrieben werden, würden diese nur 6,5 % des jährlichen Elektrizitätsbedarfs von Österreich benötigen. Diese zusätzlich benötigte Energiemenge steht bereits heute in den Nachtstunden zur Verfügung bzw. könnte leicht durch Effizienzsteigerungen im elektrischen Energiesystem erreicht werden [Bra08, S.383]. Die Herausforderung, die bei der vermehrten Durchdringung von Elektrofahrzeugen entsteht, stellt nicht die Bereitstellung der zusätzlich benötigten Energie dar, sondern die elektrische Leistung, die für die Ladung der Fahrzeuge aufgebracht werden muss.
Durch die bereits bekannten Verhaltensmuster die beim Nachtanken eines Fahrzeuges auftreten, kann man daraus schließen, dass auch die zukünftigen Nutzer von Elektrofahrzeugen zu kurzen Ladezeiten mit hoher Leistung (Schnellladung) tendieren. Dieser Umstand wiederum wird die Verteilnetzbetreiber zu einem zusätzlichen Netzausbau zwingen, da konventionelle Haushaltsanspeisungen für diese Leistungen nicht ausgelegt sind. Ladevorgänge mit längeren Ladezeiten führen hingegen zu einer Verringerung der Ladeleistung und könnten somit bereits bei bestehenden elektrischen Energienetzen eingesetzt werden (vgl. [LB08, S.391]).

Eine weitere Herausforderung stellt die Gleichzeitigkeit der Ladevorgänge von mehreren Fahrzeugen dar, die jedoch erst bei einer höheren Durchdringungsrate von Elektrofahrzeugen auftritt. Derzeit werden Elektrofahrzeuge nach dem „Anschließen an die Steckdose“ sofort und mit der maximal möglichen Leistung geladen. Dieser Vorgang wird als ungesteuertes oder unkoordiniertes Laden bezeichnet [RPL+12, S.7]. Die bereits vorhandene abendliche Leistungsspitze im Haushaltslastprofil wird bei dieser Art der Ladung erhöht und es könnte zu Überlastungen und Engpässen im elektrischen Verteilnetz kommen [Lit10, S.13].
Um dieser Situation entgegenzuwirken, ist ein massiver Netzausbau, wie zum Beispiel die Querschnittsvergrößerung der Zuleitungen oder die Kapazitätsvergrößerung des Ortstransformators, erforderlich. Um diesen Netzausbau hinauszuzögern, wird derzeit eine alternative Lösungsstrategie verfolgt, bei der versucht wird, den Ladevorgang „intelligenter“ zu gestalten. Dies erreicht man durch die Steuerung bzw. Regelung der Ladevorgänge mehrerer Fahrzeuge (siehe Kapitel 2.4). Die einfachste Möglichkeit, um die abendliche Leistungsspitze zu reduzieren, ist die gleichmäßige Verteilung der Ladevorgänge auf die Nachtstunden [LSL10, S.5], um bereits vorhandene Kapazitäten des elektrischen Verteilnetzes ausnützen zu können. Diese Strategie verursacht einen zusätzlichen Kommunikationsaufwand zwischen den Ladesäulen, den Fahrzeugen und einem Leitsystem, welches die Koordination der Ladevorgänge übernimmt.

1.2 Entwicklung einer Simulationsumgebung für Ladesäulen

Diese Diplomarbeit soll einen Beitrag zur Entwicklung einer Testmöglichkeit für Ladeinfrastrukturen, bestehend aus mehreren Ladesäulen, leisten. Ziel ist es dabei, die Interoperabilität zwischen

Komponenten verschiedener Hersteller zu gewährleisten, um eine einheitliche und kompatible Ladeinfrastruktur zu schaffen. Dadurch soll die Kundenakzeptanz und die internationale Verbreitung von Elektrofahrzeugen gefördert werden. Es ist daher erforderlich, neue Ansätze für die Testung und Funktionsüberprüfung dieser Anlagen zu entwerfen. Um die korrekte Funktionsweise einer einzelnen Ladesäule zu verifizieren, würde eine einfache automatisierte Testumgebung ausreichen, die alle Einzelfunktionen der entsprechenden Ladesäule überprüft. In dieser Diplomarbeit soll einen Schritt weitergegangen werden und eine Simulationsumgebung entwickelt werden, welche imstande ist, reale und simulierte Ladesäulen gleichzeitig anzusprechen. Bei simultanem Betrieb einer großen Anzahl von Ladesäulen in einem gemeinsamen Abschnitt des elektrischen Verteilnetzes, könnte es zu gegenseitigen Beeinflussungen kommen, welche die Netzqualität verschlechtern (z. B. Oberschwingungen). Die Simulationsumgebung ermöglicht die gemeinsame Ansteuerung von mehreren Ladesäulen und die gegenseitigen Wechselwirkungen können dadurch analysiert und überprüft werden.

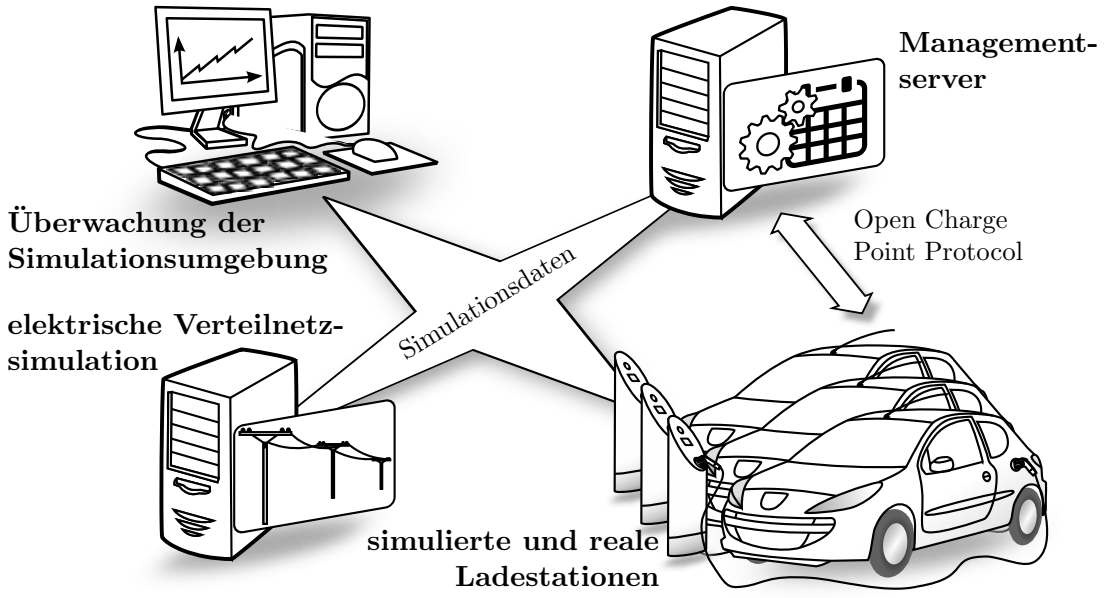

Abbildung 1.1: Grundlegende Architektur der Simulationsumgebung

In Abbildung 1.1 wird ein grundlegender Aufbau einer solchen Simulationsumgebung dargestellt, welche in dieser Diplomarbeit entwickelt wird. Der zentrale *Managementserver* stellt dabei den Mittelpunkt des Systems dar. Dieser übernimmt alle Funktionen, welche für die Koordination von mehreren Ladesäulen notwendig sind. Mit diesem Server sollen in späterer Folge *reale Ladesäulen* angesteuert werden, welche über das *Open Charge Point Protocol* kommunizieren. Um eine der Realität entsprechende Belastung des simulierten elektrischen Energienetzes zu generieren, mit der Ladesäulen im Verbund und deren gegenseitige Beeinflussung getestet werden kann, müssen zusätzliche *simulierte Ladestationen* in der Umgebung vorhanden sein. Neben der beschriebenen Ladeinfrastruktur verfügt das System über eine *elektrische Verteilnetzsimulation*, welche das elektrische Energienetz nachbildet. Die bezogenen Leistungen der realen und simulierten Ladesäulen werden dieser Netzsimulation zugeführt, um den Einfluss auf das elektrische Netz evaluieren zu können. Ein *Kontroll- und Überwachungssystem* dient sowohl der Bedienung, als auch der Steuerung der gesamten Simulations- bzw. Testumgebung.

1.3 Ziel der Arbeit

Das Ziel dieser Arbeit besteht in der Entwicklung einer Simulationsumgebung, welche die Fähigkeit besitzt, simultan real aufgebaute und am Computer simulierte Ladesäulen anzusteuern. Die über diese Ladesäulen bezogenen Leistungen werden einer Verteilnetzsimulation übergeben, um den Einfluss auf das elektrische Energienetz zu analysieren. Ebenso soll diese Umgebung in der Lage sein, verschiedene Managementalgorithmen einzusetzen, mit denen die Ladevorgänge der angeschlossenen Ladesäulen bzw. Elektrofahrzeuge koordiniert werden können.

Diese Diplomarbeit soll den Grundstein für weitere Untersuchungen bilden, bei denen die hier entworfene Umgebung zu einer automatisierbaren Testumgebung für Ladesysteme weiterentwickelt werden kann. Sie bildet somit die Basis, um später die korrekte Funktionsweise von realen Ladesäulen im kollektiven Betrieb überprüfen zu können. Dieser Modus zur Testung von Ladesäulen soll bereits beim Architekturentwurf der Softwarekomponenten durch geeignete Schnittstellen berücksichtigt werden.

Mit der entworfenen Simulationsumgebung soll am Ende dieser Diplomarbeit der Einfluss verschiedener Lademanagementkonzepte auf ein elektrisches Energienetz untersucht werden. Dabei werden besonders Algorithmen verwendet, welche sich durch ihre Einfachheit auszeichnen und ohne zusätzliche Informationen von Seiten des elektrischen Energienetzes auskommen. Daher könnten sich solche Algorithmen bereits in naher Zukunft durchsetzen, da sie in den Ladereglern der Elektrofahrzeuge implementiert werden können und sich der Kommunikationsaufwand zwischen Elektrofahrzeug und elektrischem Energienetz in Grenzen hält.

1.4 Methodik

Die Vorgehensweise im Zuge dieser Diplomarbeit beginnt mit einem ausführlichen Studium über den aktuellen Stand der Technik im Bereich der Elektromobilität und dessen Ladeinfrastrukturen. Dies bildet die Basis für die Untersuchung der Testmöglichkeit einer Ladeinfrastruktur und dem Festlegen der dafür benötigten Systemanforderungen. Die Kombination dieser Anforderungen mit den Komponenten eines Simulationssystem resultiert in verschiedenen Architekturansätzen, welche analysiert und diskutiert werden. Die Auswahl der geeignetsten Architektur bildet die Grundlage für die Implementierung einer Simulationsumgebung, welche den parallelen Einsatz von real aufgebauten und simulierten Ladesäulen erlaubt. Diese Implementierung, sowie die Analyse verschiedener Strategien zur Koordination mehrerer Ladevorgänge von Elektrofahrzeugen dienen der Validierung dieser ausgewählten Architektur. Aus den gewonnenen Erkenntnisse bei der praktischen Ausführung werden am Ende die Ergebnisse zusammengefasst und ein Ausblick auf die verschiedenen Einsatzmöglichkeiten der entworfenen Simulationsumgebung gegeben.

2 Stand der Technik

Die Elektromobilität wird derzeit stark politisch forciert. Der technologische Fortschritt ist dabei nicht nur in Bereichen, die in direktem Zusammenhang mit dem im Fahrzeug befindlichen elektrischen Antriebsstrang stehen, erkennbar. So werden neben der umfangreichen Forschung im Fahrzeugbereich und der Batterietechnologie, auch Ladekonzepte und Ladeinfrastrukturen entwickelt, die eine optimale Ausnutzung der bereits vorhandenen elektrischen Übertragungs- und Verteilnetzinfrastruktur erlaubt. Für eine erfolgreiche Marktdurchdringung und Nutzerakzeptanz der Elektrofahrzeuge sind Entwicklungen in diesem Bereich gleichbedeutend mit jenen in der Fahrzeugtechnologie. Dieses Kapitel behandelt den aktuellen Stand der Technik, der in die anschließende Entwicklung der Simulationsumgebung einfließen wird.

2.1 Ladetechnik für Elektrofahrzeuge

Die Ladetechnik von Elektrofahrzeugen kann prinzipiell als Zusammenspiel verschiedener Stakeholder mit unterschiedlichen Anforderungen an den Ladevorgang beschrieben werden. Aus physikalischer Sicht kann mit dem energetischen Prinzip

$$\text{geladene Energiemenge}\,[\text{kWh}] = \text{Ladeleistung}\,[\text{kW}] \times \text{Ladedauer}\,[\text{h}] \tag{2.1}$$

der Ladevorgang in Batteriesystemen, wie er auch in Elektrofahrzeugen eingesetzt wird, vereinfacht beschrieben werden. Aus Kunden- bzw. Nutzersicht sind in dieser Beziehung besonders die geladene Energiemenge als auch die Ladedauer von großer Bedeutung. Die geladene Energiemenge kann im Falle einer Vollladung und bei fast leerem Batteriestand der Batteriekapazität des Fahrzeuges gleichgesetzt werden und steht somit in direkter Beziehung zu der möglichen Reichweite des Fahrzeuges.
Die Dauer des Ladevorganges muss auf die Bedürfnisse und das Verhalten des Fahrzeugnutzers abgestimmt sein und kann grundsätzlich nach [Dor12] in vier Anforderungsprofile gegliedert werden:

- **Autobahn**
  Um den Ansprüchen der Nutzer an Autobahnraststationen gerecht zu werden, muss eine große Energiemenge in der kurzen Zeit einer Kaffeepause (15-30 min) bereitgestellt werden. Dafür wird die Möglichkeit einer Schnellladung mit entsprechend hoher Ladeleistung benötigt. Dieser Fall bedarf daher einer speziell dafür ausgelegten Ladeinfrastruktur.

- **Geschäftslokal, Restaurant, Einkaufszentrum**
  Mit durchschnittlichen Verweilzeiten der Kunden von 30-120 min können in diesem Anforderungsprofil kostengünstigere Ladetechnologien eingesetzt werden, da hier eine geringere Ladeleistung als auf Autobahnen bereitgestellt werden muss.

- **Wohnort**
  Für die private Anwendung kann eine geringe Ladeleistung vorgesehen werden, bei der ein Fahrzeug über eine längere Zeitspanne (Ladevorgang über die Nachtstunden) geladen werden kann. Dadurch kann an Wohnorten eine Ladelösung eingesetzt werden, welche kostengünstig ist und zusätzlich leicht ins Energienetz integriert werden kann.

- **Büro**
  Hier setzen sich die Anforderungen an die Ladeinfrastruktur aus den beiden bereits vorher beschriebenen Profilen zusammen. Einerseits kann eine kostengünstige Lösung mit längeren Ladezeiten (8 h Arbeitstag) für die Mitarbeiter eingesetzt werden. Auf der anderen Seite muss die Möglichkeit bestehen, wie bei Geschäftslokalen, Ladevorgänge für Kunden und häufig benötigten Firmenfahrzeugen mit kürzeren Ladezeiten durchführen zu können.

Die beschriebenen Anforderungsprofile erfordern daher unterschiedliche Ladetechnologien, die auf den jeweiligen Anwendungsfall abgestimmt sind.

2.1.1 Ladetechnologien

Um den vorher beschriebenen Anforderungsprofilen gerecht zu werden, befinden sich derzeit verschiedene Ladetechnologien am Markt, welche in diesem Abschnitt kurz erläutert werden. Dabei ist die grundsätzliche Einteilung zu treffen, ob der Energiefluss von der Ladestation zum Fahrzeug über Kabel (konduktiv) oder induktiv erfolgt. Die Ladung über ein Ladekabel mit Gleich- oder Wechselstrom unterliegt dem derzeitigen Stand der Technik.

Die verschiedenen Lademodi werden in der Norm IEC 61851-1:2010 [5, S.15] beschrieben und sind überblicksmäßig in Abbildung 2.1 dargestellt.

Im einfachsten Fall kann ein Elektrofahrzeug über die bereits im Haushalt vorhandenen Steckdosen, welche keine zusätzliche Infrastruktur benötigen, mit einem maximal vorgeschriebenen Wechselstrom (AC) von 16 A (Modus 1) bzw. 32 A (Modus 2) geladen werden. Der Modus 1 schreibt keine zusätzliche Schutzfunktion, wie Erdung und Fehlerstrom-Schutzeinrichtung, vor und ist daher in den USA und einigen anderen Ländern nicht erlaubt. Diese fehlende Schutzfunktion ist in Modus 2 durch einen im Kabel eingebauten Fehlerstromschutzschalter nachgerüstet. Ebenso kann in diesem Modus eine zusätzliche Steuerkomponente (in-cable control box) eingebaut werden, um den maximalen Ladestrom zwischen Fahrzeug und Energienetz festzulegen [5].

Im Modus 3 wird eine permanent installierte Ladestation eingesetzt, welche typischerweise mit dreiphasigem Wechselstrom bis 63 A Ladestrom arbeitet [VFCI12, S.4]. Solange bei dieser Betriebsart kein Fahrzeug an der Station angeschlossen ist, wird das Ladekabel spannungsfrei geschaltet, um die Sicherheit zu erhöhen. Ebenso ist bereits eine Kommunikation über das Ladekabel zwischen Fahrzeug und Ladestation vorgesehen [6, S.18].

Der Lademodus 4 unterscheidet sich stark von den anderen. In diesem Modus wird der Gleichrichter bzw. der Laderegler *nicht* im Auto verbaut, sondern in der Ladestation [7, S.9]. Dadurch wird

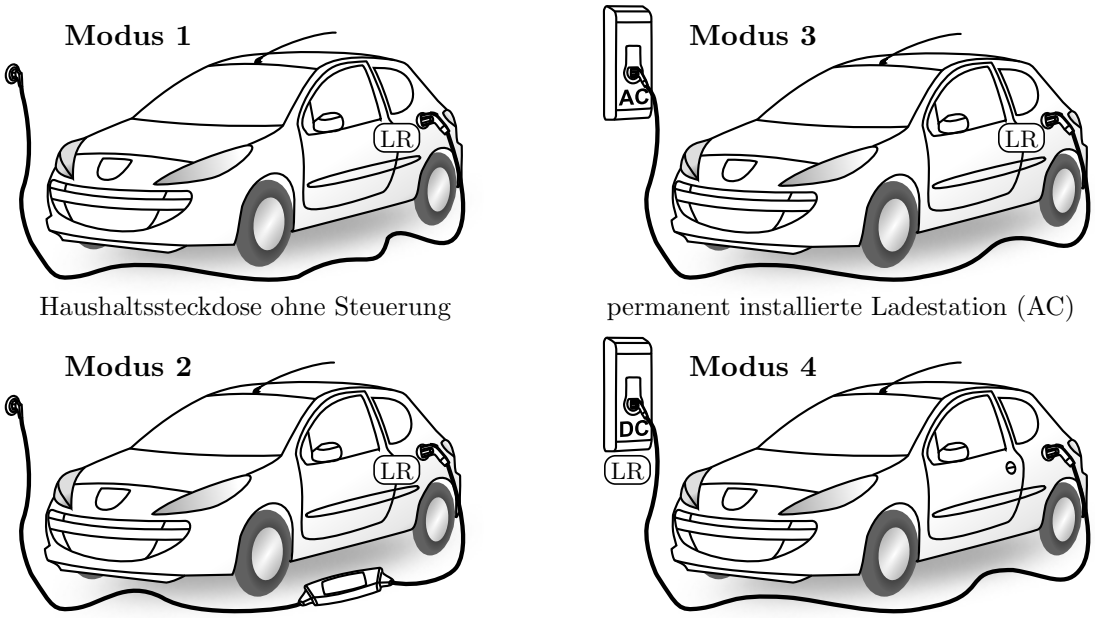

Abbildung 2.1: Lademodi nach IEC 61851-1

über das Ladekabel ein Gleichstrom (DC) mit hoher Leistung übertragen, mit dem die Fahrzeugbatterie geladen wird. Diese Form des DC-Ladens hat den Vorteil, dass bei hohen Ladeleistungen Ladekabel mit kleineren Leitungsquerschnitten verwendet werden können, wodurch die Handhabbarkeit verbessert wird [NPE10, S.12]. Zusätzlich ist in dieser Betriebsart die Sicherheitsfunktion, wie in Modus 3 beschrieben, integriert.

Die induktive Ladetechnologie stellt eine Alternative zum Ladekabel dar, die noch in den Kinderschuhen steckt. Hier kann ohne Eingriff durch den Fahrzeugnutzer das Fahrzeug am Abstellort direkt und ohne Ladekabel geladen werden. Durch das induktive Ladekonzept besteht die Wahrscheinlichkeit, dass ein Fahrzeug für das Vehicle2Grid-Konzept mit Rückspeisung und Lastverteilung (siehe Abschnitt 2.4) öfters zur Verfügung steht [NPE10, S.12], da ein versehentliches Vergessen der Verbindungsherstellung unterbunden werden kann.

2.1.2 Restriktion aufgrund der elektrischen Verteilnetzstruktur

Obwohl die maximal mögliche Ladeleistung grundsätzlich von der Batterietechnologie abhängig ist, spielen die Restriktionen der elektrischen Verteilnetzstruktur eine große Rolle. Ein Teil dieser Einschränkungen wird erst in Zukunft bei einer höheren Durchdringungsrate der Elektromobilität eine Rolle spielen. Bestimmte Restriktionen sind jedoch bereits heute vorhanden und müssen daher berücksichtigt werden.

Es besteht die Annahme, dass während der Einführungsphase der Elektromobilität, ein Großteil der Fahrzeuge am Wohnort des Besitzers geladen wird [Bra08, S.385]. Elektrische Anschlüsse und Leitungen in Hausinstallationen sind typischerweise für einen maximalen Strom von 16 A abgesichert. Theoretisch ergäbe sich hier eine Ladeleistung von 3,7 kW, welche aber in den seltensten Fällen erreicht werden kann. Dauerströme an dieser Belastungsgrenze, wie sie beim Laden

auftreten, könnten zu thermischen Überlastungen der Stromleitungen und somit bis zum Ausbruch eines Hausbrandes führen. In diesen Fällen ist ein Ausbau der Hausinstallation mit einem separaten Abgang für die Ladestation erforderlich [NPE10, S.11].

Die typische Auslegung heutiger Anschlussleistungen bei Haushalten liegt bei 16 kW bzw. 25 kW Spitzenleistung. Trotzdem sind wegen der Gleichzeitigkeit der Haushaltsbelastungen nur 2 kW Dauerbelastung pro Haushalt am Ortstransformator erforderlich und vorgesehen [Bra08, S.385]. Werden in Zukunft durch Elektrofahrzeuge alle Anschlüsse mit einer höheren, durch die Ladevorgänge der Elektrofahrzeuge verursachte, Leistung dauerhaft belastet, kommt es zur Überschreitung der Betriebsgrenzen von verschiedenen Betriebsmitteln im elektrischen Netz. Davon sind insbesondere der Transformator und jene Leitungen betroffen, bei denen durch die höhere thermische Belastung die Lebensdauer gesenkt wird [RHR12, S.85].

Der Ladevorgang von Fahrzeugen verursacht eine zusätzliche Belastung des elektrischen Energienetzes. In den Leitungen muss ein höherer Strom transportiert werden, welcher durch den elektrischen Widerstand einen höheren Spannungsabfall und somit höhere Netzverluste erzeugt. Nach ÖVE/ÖNORM EN 50160 muss an der Übergabestelle zum Endverbraucher die Versorgungsspannung innerhalb eines Spannungsbandes von ±10 % der Nennspannung liegen [SGM11, S.25]. Diese Grenzen müssen trotz des höheren Bedarfs an Ladeenergie mit geeigneten Mitteln, wie einem Netzausbau oder einem intelligenten Lademanagement, eingehalten werden.

2.1.3 Batterietechnologien

Aus der aktuellen Marktübersicht [3] kann entnommen werden, dass als Energiespeicher in Elektrofahrzeugen die Lithium-Ionen-Batterietechnologie (Li-Ion) am häufigsten eingesetzt wird. Im Gegensatz zu Nickel-Metallhydrid-Batterien (NiMH), die verstärkt in Hybridfahrzeugen Verwendung finden [KMB$^+$12, S.15], haben Lithium-Ionen-Batterien eine höhere spezifische Ausgangsleistung (W/kg) und einen um 20 % höheren spezifischen Energieinhalt (Wh/kg) [YWWS13, S.31]. Diese Parameter werden durch ein hohes elektrochemisches Potential und durch das geringe spezifische Gewicht von Lithium erreicht [KMB$^+$12, S.16].

Der typische Ladeverlauf einer Lithium-Ionen Batterie wird in Abbildung 2.2 dargestellt und unterteilt sich in drei Abschnitte. Der Laderegler, der für den Ladevorgang der Batterie zuständig ist, muss die Betriebsarten *Laden mit konstantem Strom* und *Laden mit konstanter Spannung*

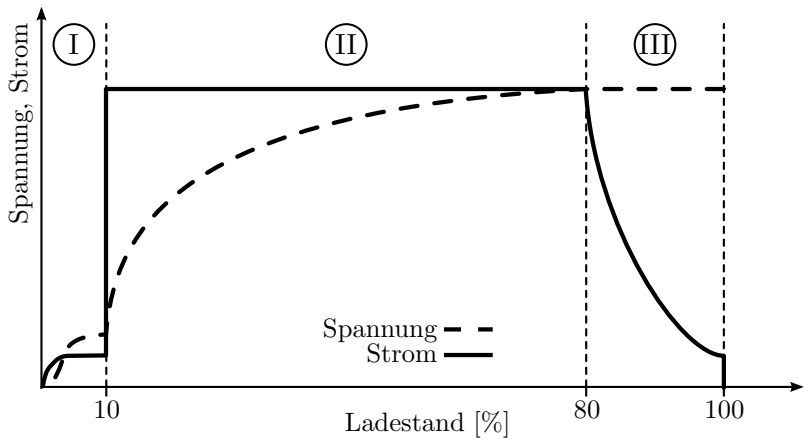

Abbildung 2.2: Typischer Ladeverlauf einer Lithium-Ionen-Batterie (vgl. [YWWS13, S.23], [8])

beherrschen. Bei einer geleerten Batterie korreliert der in der Abbildung angeführte Ladestand mit der Dauer des Ladevorganges.
Im ersten Abschnitt des Ladevorganges (I) – falls die Batterie stark entladen ist – muss mit einem kleinen Strom (∼10 % des maximalen Ladestromes) gearbeitet werden, um ein Überhitzen der Batteriezelle zu vermeiden [8]. Anschließend, im zweiten Abschnitt (II), wird die Batterie mit einem vorgegebenen konstanten Ladestrom geladen, bis die Klemmenspannung der Batterie erreicht ist. An diesem Punkt angelangt, muss der Laderegler auf den Modus „Laden mit konstanter Spannung“ umschalten, um die maximale Batteriekapazität zu erreichen, ohne die Batterie zu zerstören. In dieser Phase (III) verringert sich der Ladestrom kontinuierlich bis ein bestimmter Endstrom erreicht und der Ladevorgang beendet ist.

Der erste und der dritte Abschnitt sind durch die Batterietechnologie vorgegeben und können daher nicht verändert werden. Nur im zweiten Abschnitt (II), beim *Laden mit konstantem Strom*, ist es möglich den Ladestrom vorzugeben. Somit ist dieser Bereich des Ladevorganges für die nachfolgenden Betrachtungen des Lademanagements von großer Bedeutung.

In [SL13, S.6ff] wurden die Ladeprofile mehrerer Elektrofahrzeuge unterschiedlicher Größe und Hersteller aufgezeichnet. Obwohl diese Profile in Ladedauer, Verlauf und maximaler Ladeleistung variieren, folgen sie alle der typischen, in Abbildung 2.2 dargestellten, Ladekurve.

2.2 Architektur eines intelligenten Ladesystems

Die Architektur eines Ladesystems für Elektrofahrzeuge ist nicht eindeutig von allgemeinen Anforderungen ableitbar. Vielmehr muss bei der Implementierung eines spezifischen Systems auf die Beteiligten und die örtlichen und strukturellen Gegebenheiten Rücksicht genommen werden. Daher unterscheidet sich die Architektur von Land zu Land bzw. von Projekt zu Projekt.

Grundsätzlich muss in einem intelligenten Ladesystem ein Informations- und Datenaustausch zwischen Stakeholdern aus unterschiedlichen Bereichen durchgeführt werden, welche in Abbildung 2.3 allgemein dargestellt sind.

Das Elektrofahrzeug und der Anwender sind grundsätzlich in jeder Architektur, die sich mit Elektromobilität beschäftigt, vorhanden. Dabei ist das *Fahrzeug* über eine Ladestation mit dem System verbunden und tauscht beispielsweise Informationen über den aktuellen Batteriezustand oder den maximal möglichen Ladestrom mit dem Infrastrukturbetreiber des Ladesystems aus. Ebenfalls interagiert der *Anwender*, der die Ladung seines Fahrzeuges durchführt, mit dem Infrastrukturbetreiber und übermittelt diesem seinen Ladewunsch, die nächste Abfahrtszeit sowie seine Identifikationsdaten.

Der *Infrastrukturbetreiber* stellt die wichtigste Komponente im System dar und ist für die Verwaltung der Ladeinfrastruktur verantwortlich. Er übernimmt verschiedene Aufgaben, von der Kundenverwaltung über die Authentifizierung und Kontrolle der Ladevorgänge bis zur Abrechnung der geladenen Energiemenge. Dieser Betreiber kann zusätzlich definierte Ladewünsche von Kunden entgegennehmen, auch wenn diese bei ihm nicht unter Vertrag stehen. Dabei stehen verschiedene Wege zur Verfügung, welche von lokaler Abrechnung mit Kreditkarte bis zu Roaming-Verträgen mit anderen Infrastrukturbetreibern reichen kann [FF13, S.4] (siehe Abschnitt 2.3.2).

Für die Vehicle2Grid-Konzepte (siehe Abschnitt 2.4) wird ein *Aggregator* benötigt. Dieser versucht durch geeignete Algorithmen, die Ladewünsche der Anwender zu respektieren und verfolgt eine Strategie, bei der vorhandenen Ressourcen optimal, bezogen auf ein gegebenes Ziel, ausgenutzt

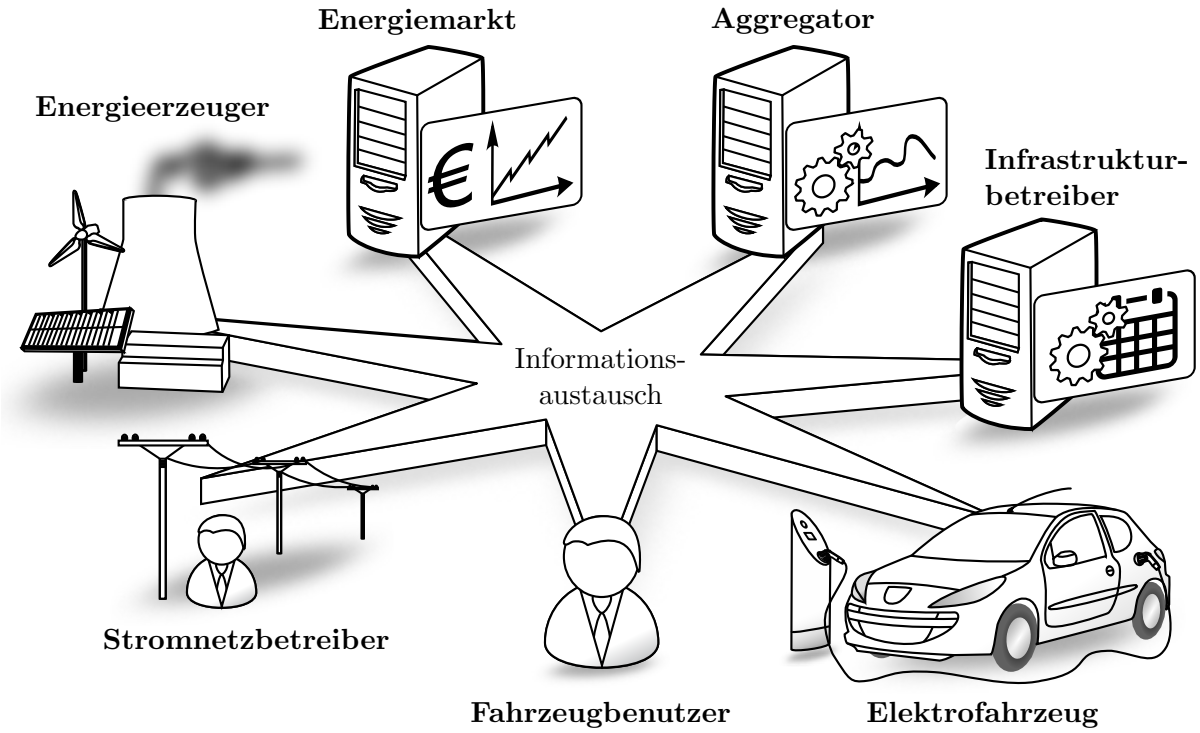

Abbildung 2.3: Informationsaustausch in einem Ladesystem [KSWH10, S.162][BGJ+10, S.4][Fas11, S.27]

werden. Ein gegebenes Ziel kann beispielsweise eine Marktoptimierung sein. Je nach Architektur kann dieser Stakeholder als eigener Teilnehmer auftreten oder seine Funktionalitäten werden durch den Stromnetzbetreiber bzw. Infrastrukturbetreiber übernommen. Im Kontext des hier beschriebenen intelligenten Ladesystems ist dieser Teilnehmer nur für das Management der variablen Lasten, welche Elektrofahrzeugen zuzuordnen sind, zuständig. Im zukünftigen intelligenten Energienetz kann dieser Aggregator auch andere variable Lasten, z. B. Kühlanlagen oder bestimmte Maschinen in Industrieanlagen, zur Optimierung seiner Lastmanagementstrategie verwenden.

Der *Stromnetzbetreiber* steuert und überwacht die elektrische Verteilnetzinfrastruktur. Dabei übermittelt dieser Informationen, wie zum Beispiel die aktuelle Netzauslastung, dem Aggregator, um den ordnungsgemäßen Betrieb des Verteilnetzes aufrecht erhalten zu können.

Neben dem Stromnetzbetreiber sind im elektrischen Energienetz noch die *Energieerzeugung* und der *Energiemarkt* von Bedeutung. Diese Stakeholder sorgen für die Bereitstellung der für die Ladung benötigten Energie. Dabei kann entweder lokal in Kraftwerken Energie erzeugt oder am Energiemarkt Strom zugekauft werden.

Die hier beschriebene und in Abbildung 2.3 dargestellte Architektur stellt nur einen allgemeinen Überblick dar. Je nach Anwendung können mehrere Stakeholder zu einem einzigen Teilnehmer zusammengefasst bzw. neue Stakeholder hinzugefügt werden.
In bestimmten Regionen wird der Energiedienstleister, der bereits heute für den Betrieb des elektrischen Energienetzes verantwortlich ist, die Koordination der Ladeinfrastruktur übernehmen und die für den Ladevorgang benötigte Energie bereitstellen. Da ihm bereits alle Informationen über das Energienetz zur Verfügung stehen, ist die Eingliederung der Elektromobilität mit möglichst einfachen Mitteln zu bewerkstelligen.
Als zweite Architektur wäre ein vom vorhandenen Energiedienstleister unabhängiger Ladeinfrastrukturbetreiber möglich. Dieser kann am Energiemarkt Strom zu günstigen Preisen einkaufen,

um seine Ladeinfrastruktur optimal betreiben zu können. Er tritt am Markt als eigener Teilnehmer auf und ist somit ein Konkurrent des Energiedienstleister [BGJ+10, S.4].

2.3 Kommunikationstechnologien für Ladeinfrastrukturen

Wie soeben erläutert wurde, beruht ein Ladesystem für Elektrofahrzeuge auf Interaktionen unterschiedlicher Stakeholder. Dabei ist es erforderlich, dass jedem Teilnehmer der Zustand des Ladesystems bzw. des elektrischen Energiesystems bekannt ist. Um dies zu bewerkstelligen, ist ein umfangreicher Informationsaustausch notwendig, der über verschiedenste Kommunikationstechnologien durchgeführt wird.

Im einfachsten Fall kann die Ladung eines Elektrofahrzeuges über die Haushaltssteckdose erfolgen (siehe Abschnitt 2.1.2 Modus 1). Da in dieser Anwendung keine Informationen zwischen Fahrzeug und Umgebung ausgetauscht werden, müssen jene Parameter, die das Energienetz beschreiben (z. B. maximal möglicher Ladestrom), vom Fahrzeug explizit angenommen werden. Diese Einstellungen beziehen sich auf eine Vielzahl von vorhandenen Ladeanschlüssen und sind daher unter den tatsächlich bestehenden Betriebsgrenzen dimensioniert. Zu Gunsten der Sicherheit gehen daher Übertragungskapazitäten verloren, welche für eine optimale Ausnutzung der Infrastruktur verwendet werden könnten.
Dieser Anwendungsfall zeigt bereits, dass für eine erfolgreiche Integration der Elektromobilität kein Weg am Austausch von Informationen vorbeiführt. Die Kommunikationstechnik, die in direktem Zusammenhang mit dem Elektrofahrzeug steht, kann grundsätzlich in drei Stufen eingeteilt werden. Abbildung 2.4 stellt die am Informationsaustausch betroffenen Teilnehmer und die Kommunikationsprotokolle dar, welche in diesem Abschnitt genauer erläutert werden.

Um zukünftigen Anforderungen entsprechen zu können, muss die Ladeinfrastruktur auf ein automatisiertes Laden vorbereitet werden. Dies ist nur mit einer bidirektionalen Kommunikation zwischen Fahrzeug, Ladesäule und Energiedienstleister bzw. -netzbetreiber möglich, bei der Informationen für verschiedene Dienste ausgetauscht werden. Diese Informationen umfassen einen weiten Bereich, von der Authentifizierung und Identifikation des Fahrzeuges, über Abrechnungsdaten der bezogenen Energie bis zur Überwachung des Ladevorganges [RHR12, S.90].

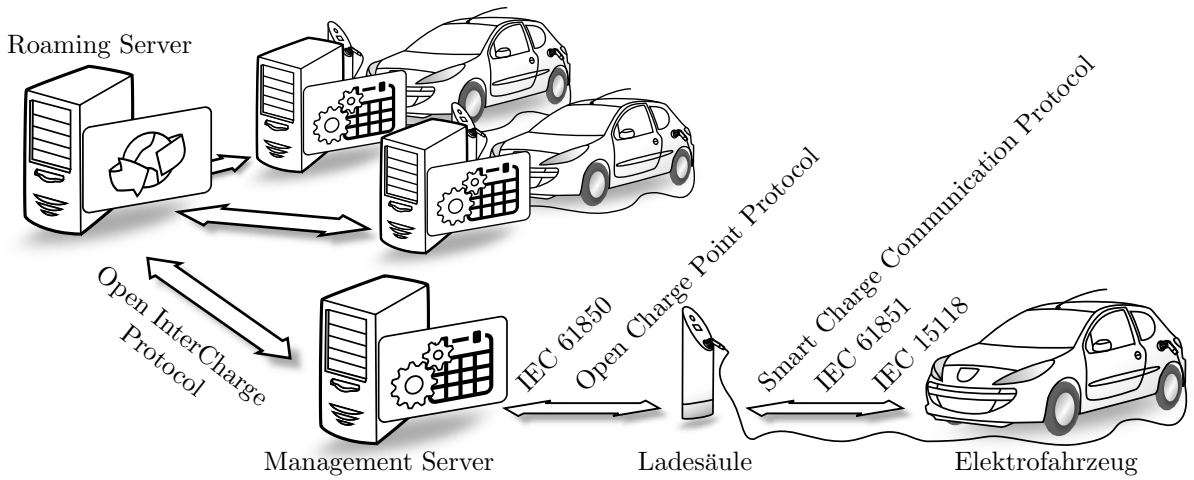

Abbildung 2.4: Kommunikationsprotokolle für Ladeinfrastrukturen

2.3.1 Kommunikation zwischen Fahrzeug und Ladesäule

Die Verbindung zwischen Fahrzeug und Ladesäule wird bereits durch verschiedene Normen beschrieben. Diese physikalische Schnittstelle (Ladekabel) wird durch die IEC 62196 [6] festgelegt und definiert die Kompatibilität der Signalisierungsleitungen und der verwendeten Steckertypen. Das Ladekabel dient nicht nur der Energieübertragung, sondern wird von den Kommunikationsprotokollen als Übertragungsmedium zum Datenaustausch verwendet.

IEC 61851

Mit der Norm IEC 61851 [5] wird der konduktive Ladevorgang standardisiert. Neben den Lademodi (siehe Abschnitt 2.1.1) wird auch eine einfache Signalisierung zwischen Fahrzeug und Ladesäule beschrieben. Dazu wird eine zusätzlich im Ladekabel vorhandene Steuerleitung, der Pilotkontakt, verwendet. Die grundlegende Aufgabe dieses Pilotkontaktes ist die Überwachung der korrekt verbundenen Schutzleitung bzw. die Prüfung des ordnungsgemäßen Anschlusses des Fahrzeuges [RHR12, S.89].

Die Funktionsweise dieser Signalisierungseinrichtung ist so einfach gestaltet, dass sie ohne digitale Recheneinheit arbeiten kann. Die Ladesäule legt dabei ein pulsweitenmoduliertes (1 kHz) Signal mit einem Spannungslevel von 12 V an den Pilotkontakt. Das Elektrofahrzeug verändert je nach Bereitschaft seines Ladereglers dieses Spannungslevel. Die Ladesäule wiederum misst die Spannung an dieser Leitung und ist somit in der Lage ein angeschlossenes Fahrzeug zu detektieren. Ebenso kann festgestellt werden, ob dieses für den Ladevorgang bereit ist. Durch das Tastverhältnis der Pulsweitenmodulation kann die Ladesäule dem Laderegler im Fahrzeug den maximal möglichen Strom vorgeben, den die Säule liefern kann [RSRW11, S.36].

Bereits dieses Protokoll ermöglicht es, eine einfache Ladesteuerung durchzuführen. Dabei gibt die Ladesäule über das pulsweitenmodulierte Signal den maximalen Strom während des Ladevorganges vor, welcher vom Laderegler im Fahrzeug eingestellt wird. Hier kann der Ladevorgang im einfachsten Fall mit Hilfe einer Zeitschaltuhr verschoben werden [RHR12, S.89]. Der Laderegler im Fahrzeug muss der Stromvorgabe innerhalb von fünf Sekunden folgen. Durch diese im Standard vorgeschriebene Zeitvorgabe („Hard realtime deadline“) entsteht ein echtzeitfähiges System, welches zum Schutz vor Überlastung des elektrischen Energienetzes verwendet werden kann [RSRW11, S.42].

IEC 15118 und Smart Charge Communication Protocol

Das bereits erläuterte Protokoll IEC 61851 ist sehr einfach gestaltet und verzichtet auf einen umfangreichen Datenaustausch. Daher wird dieses Protokoll auch als Low-Level-Protokoll bezeichnet. Um in Zukunft jedoch eine Ladeinfrastruktur zu etablieren, müssen zwischen Elektrofahrzeug und Ladesäule mehrere Informationen, wie zum Beispiel Fahrzeugidentifikation, Batterieladestand oder Zählerstände, ausgetauscht werden. Dafür werden sogenannte High-Level-Protokolle verwendet, welche einen umfangreichen Datenaustausch erlauben.
Das Smart Charge Communication Protocol (SCCP) und die Norm IEC 15118 beschreiben ein solches Kommunikationsprotokoll und sind im Allgemeinen sehr ähnlich aufgebaut. Das SCCP wurde 2008 von Daimler und RWE als Pilotprojekt entwickelt, mit dem Ziel die damals vorhandene Lücke einer höheren Kommunikation zwischen Fahrzeug und Ladesäule zu schließen [RSRW11,

S.34]. Es bildet die Grundlage für die internationale und im Frühjahr 2014 veröffentlichte Norm IEC 15118 [RHR12, S.90].

Beide Protokolle basieren auf einer Internet Protokoll (IP) gestützten Powerline-Kommunikation (PLC), welche als aufmoduliertes Signal über die Stromleitung zwischen Fahrzeug und Ladesäule übertragen wird. Der Nachrichtenaustausch zwischen beiden Teilnehmern erfolgt auf Basis des verbindungsorientierten und packetvermittelten Transport Control Protocol (TCP), bei dem die Verbindung über das Transport Layer Security (TLS) Protokoll gesichert und verschlüsselt wird [RSRW11, S.37].

Neben kleinen Veränderungen in der Abfolge der transportierten Nachrichten bzw. leicht abgeänderten Parametern unterscheiden sich beide Protokolle hauptsächlich in der Darstellung der übertragenen Daten. Das SCCP setzt auf die in Deutschland sehr weit verbreitete, aus dem Smart Metering Bereich stammende, Smart Message Language (SML), während die IEC 15118 eine mit der Efficient XML Interchange (EXI) komprimierte Extensible Markup Language (XML) verwendet [RHR12, S.90]. Beide Darstellungen benützen ein komprimiertes Byte-Format, wodurch die Größe einer Nachricht verringert und dadurch effizient übertragen werden kann.

2.3.2 Kommunikation mit Backend-Systemen

Der Standardisierungsgrad im Kommunikationsbereich zwischen Fahrzeug und Ladesäule ist bereits weit fortgeschritten. Nachdem 2013 der Ladestecker für die Verbindung genormt wurde, folgt im Frühjahr 2014 die High Level-Kommunikation IEC 15118 [9]. Somit unterliegt dieser Bereich einheitlichen internationalen Richtlinien. Im Gegensatz dazu, ist die Kommunikation hinsichtlich Management- bzw. Kontrollsystemen und Ladesäulen noch nicht geregelt. Als Folge davon entwickelten sich viele inkompatible und spezifische Systeme.
Dieser Abschnitt beschreibt Protokolle, die sich in Zukunft in diesem Bereich als Quasi-Standard etablieren könnten, da sie bereits heute schon von unterschiedlichen Herstellern in bestimmten Regionen eingesetzt werden.

IEC 61850

Ursprünglich wurde die Normenreihe IEC 61850 für die Stationsautomatisierung definiert und beschreibt ein Kommunikationsprotokoll in der Schutz- und Leittechnik von Schaltanlagen der Mittel- und Hochspannungstechnik. Im Laufe der Zeit wurde diese Normenreihe kontinuierlich erweitert, indem die Kommunikation generalisiert wurde, um für die dezentrale elektrische Energieversorgung mit Wasser- und Windkraftanlagen eingesetzt werden zu können [10].

Die IEC 61850 befasst sich nicht nur mit der Datenübertragung, sondern sie kann auch dazu verwendet werden, gesamte Schaltanlagen formal zu beschreiben. Diese Verwendungsmöglichkeit ergibt sich durch die eigens definierte Substation Configuration Language (SCL), welche auf der Extensible Markup Language basiert. Jede eingesetzte Komponente verfügt über eine SCL-Beschreibungsdatei, in der diese Komponente modelliert ist und damit eine konsistente Datendarstellung erreicht wird [Mac06, S.2]. Durch diese Art der Beschreibung entsteht ein abstrakter Zusammenhang, welcher verschiedene Übertragungsprotokolle verwenden kann. Dabei besteht die Möglichkeit neben anderen Übertragungsprotokollen, auch TCP/IP und Ethernet zu nutzen [Mac06, S.7].

Obwohl Elektrofahrzeuge als Energiespeicher für regenerative Energien dienen können, werden sie nicht von der IEC 61850 erfasst. In verschiedenen Projekten ([SWA12], [UOZ13]) wurde bereits versucht diesen Missstand zu beheben, indem IEC 61850-konforme Modelle für die Ladeinfrastruktur von Elektrofahrzeugen entwickelt wurden.

Open Charge Point Protocol

Das Open Charge Point Protocol (OCPP) wurde 2009 vom niederländischen Konsortium e-laad initiiert. Es beschreibt den Datenaustausch zwischen Ladesäule und einem zentralen Managementsystem, das von einem Energienetz- oder Ladeinfrastrukturbetreiber geleitet werden kann. Das OCPP steht zur freien Verwendung offen zur Verfügung und entwickelt sich dadurch zu einem De-Facto-Standard, welcher von einer europaweiten Initiative gefördert wird [9].

Dabei wird das Netzwerkprotokoll SOAP verwendet, mit welchem Nachrichten über das Internet ausgetauscht werden können. Dadurch kann eine rasche Implementierung des OCPPs erfolgen [11]. Zur Übertragung dienen verschiedene derzeit vorhandene Medien, die für den Internetzugriff ausgelegt sind. Darunter fallen neben kabelgebundenen Technologien, wie LAN oder ADSL, auch kabellose, wie WLAN, LTE oder UMTS [Luk13].

In der derzeit aktuellen Version 1.5 ist das Open Charge Point Protocol für das administrative Management der Ladevorgänge ausgelegt, was der Funktionsbeschreibung entnommen werden kann [12]. Daher ist dieses Protokoll derzeit noch nicht ausgereift genug, um in einem Lademanagementkonzept eingesetzt zu werden, da keine Informationen über den Energiebedarf oder den Abfahrtszeitpunkt übertragen werden können.
Der mit Ende 2013 veröffentlichte Entwurf der Protokollversion 2.0 lehnt sich stark an das Kommunikationsprotokoll IEC 15118 (siehe 2.3.1) an, und ist bereits in der Lage, ein solches Lademanagementkonzept umzusetzen [13].

Open InterCharge Protocol

Derzeit muss ein Fahrzeugnutzer, wenn er einen Ladevorgang, grundsätzlich an öffentlichen Ladesäulen, beginnen will, sich bei der entsprechenden Ladesäule bzw. beim Betreiber dieser authentifizieren. Die Authentifizierungs- und Abrechnungsmethoden sind bei jedem Betreiber unterschiedlich und beruhen meist auf individuell abgeschlossenen Verträgen. Dadurch ist es generell nicht möglich, für den Ladevorgang eine Ladesäule eines fremden Betreibers zu verwenden.

Dieser Mangel versucht das deutsche Joint-Venture Hubject [14] zu beheben, indem es ein Informations- und Transaktionsportal zur Verfügung stellt, um dem Endkunden eine Lademöglichkeit an allen beteiligten Ladesäulen zu gewährleisten. Die Gründungsmitglieder, unter anderem namhafte deutsche Firmen wie Siemens, RWE, BMWGroup oder Bosch, versuchen damit eine aus Kundensicht europaweite Ladeinfrastruktur zu errichten nach dem Vorbild des bekannten Roamings in Mobilfunknetzen [9].

Der Zugang und der Informationsaustausch zwischen den Ladesäulen- bzw. Elektromobilitätsbetreibern erfolgt über das frei verfügbare Open InterCharge Protocol (OICP). Dieses stellt mit Hilfe eines speziellen Roaming-Servers (Hubject Platform) die Verbindung zwischen beiden Betreibern her. Die Kommunikation erfolgt, wie beim Open Charge Point Protocol, über vordefinierte Webservices, welche den Nachrichtenaustausch beschreiben [15].

Ein ähnliches Protokoll ist das Open Clearing House Protocol (OCHP), welches von e-clearing.net [16] für denselben Zweck eingesetzt wird und somit als Konkurrenzprodukt des von Hubject entwickelten OICPs angesehen werden kann.

2.4 Lademanagementkonzepte

Bereits im Abschnitt 1.1 wurde die Herausforderung erläutert, die durch das gleichzeitige Auftreten mehrerer Ladevorgänge von Elektrofahrzeugen ergibt. Dabei kann das Ziel verfolgt werden, die abendliche Leistungsspitze zu reduzieren, indem Ladevorgänge an einen anderen Tageszeitpunkt verschoben werden. Dadurch kann eine gleichmäßige Auslastung des elektrischen Energienetzes erreicht werden. Diese Art der Lastverschiebung beschränkt sich dabei nicht nur auf die Ladevorgänge von Elektrofahrzeugen. Unter dem Begriff Demand Response werden unter anderem alle Maßnahmen bezeichnet, bei denen durch intelligente Steuerung des Leistungsverbrauchs die Leistungsspitzen im elektrischen Energienetz verringert und bestehende Energienetzinfrastrukturen effizienter ausgenutzt werden [DTU10]. Ebenso wird in Gebieten mit hohen regenerativen Anteil in der Energieerzeugung als alternative Strategie versucht, den Energiebezug von zeitlich flexiblen Lasten derart zu verschieben, dass eine Anpassung an die gegebene Erzeugung erreicht werden kann [PSF12].

Grundsätzlich unterscheidet man drei verschiedene Konzepte, welche eingesetzt werden können, um den Ladevorgang durchzuführen [RPL+12, S.7]. Dabei treten die Auswirkungen des jeweiligen Konzeptes nicht direkt an der entsprechenden Ladesäule auf, sondern der Erfolg lässt sich am Zustand des gesamten elektrischen Energienetzes und dessen Netzqualität messen.

2.4.1 Unkoordiniertes Ladekonzept

Beim unkoordinierten bzw. ungesteuerten Laden müssen keine Vorkehrungen beim Ladevorgang getroffen werden. Jedes Elektrofahrzeug arbeitet unabhängig und kann sich seinen benötigten Ladestrom selbst einstellen. Der Beginn des Ladeprozesses erfolgt sofort nach dem Anschließen des Fahrzeuges ans elektrische Energienetz.

Wenn geeignete Maßnahmen, wie eine niedrige Ladeleistung und eine symmetrische Lastverteilung auf drei Phasen, eingesetzt werden, stellt das ungesteuerte Laden derzeit eine sinnvoll nutzbare Methode dar. Erst bei höherem Durchdringungsgrad der Elektromobilität von mehr als 40 % wird es bei diesem Ladekonzept zu flächendeckenden Netzengpässen kommen [RPL+12, S.10].

2.4.2 Unidirektionales koordiniertes Ladekonzept

Unidirektionale Ladekonzepte beschreiben den Eingriff in den Ladevorgang von Elektrofahrzeugen. Diese Konzepte können auch als Lademanagement bezeichnet werden, da hier nur die Steuerung des Ladevorganges übernommen wird.

Prinzipiell unterliegen alle Lademanagementkonzepte der Idee die elektrische Last, seien es Haushalte oder elektrische Fahrzeuge, gleichmäßig über den Tag zu verteilen oder bereits vorhandene, regenerative Energieressourcen optimal auszunutzen. Elektrofahrzeuge können als flexible Lasten angesehen werden, bei denen der Zeitpunkt des Ladevorganges zeitlich verschoben werden kann. Abbildung 2.5 stellt diese Grundidee schematisch dar. Dadurch können Leistungsspitzen

der Gesamtbelastung (Grundbelastung durch Haushalte und Elektrofahrzeuge) an bestimmten Tageszeitpunkten, wie zum Beispiel am Abend, reduziert werden, indem die Ladevorgänge der Fahrzeuge auf die Nachtstunden verschoben werden. Die gesamte benötigte Energiemenge bleibt dabei konstant und die maximale Belastung des elektrischen Energienetzes kann vermindert werden. Als Voraussetzung müssen die Ladevorgänge von Elektrofahrzeugen möglichst flexibel gestaltet sein, indem die vorgegebene Zeitdauer für den Ladevorgang um einiges höher ist als die tatsächlich benötigte Ladedauer der Batterieladung.

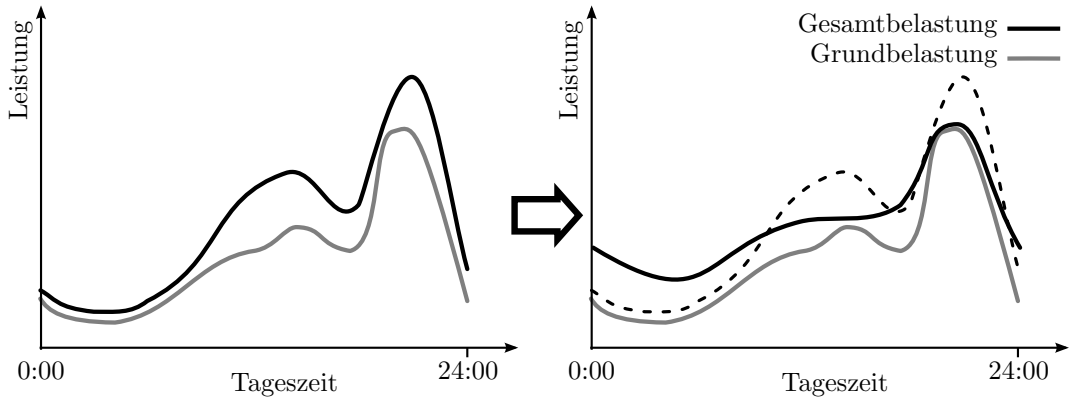

Abbildung 2.5: Konzept der netzgeführten Lastverteilung im elektrischen Energienetz

Das unidirektionale Ladekonzept kann in zwei Stufen eingeteilt werden, wobei der technische Aufwand unterschiedlich hoch ausfällt. Die erste Stufe wird als gesteuertes Laden bezeichnet. Hier wird versucht einen zeitlichen Ladeplan für die Elektrofahrzeuge aufgrund von Prognosen zu erstellen. Dabei können zwei Strategien mit unterschiedlichen Zielen verfolgt werden:

- Die *marktbasierte Strategie* orientiert sich am Energiemarkt und versucht die kostengünstigste Variante zu finden, um die Elektrofahrzeuge zu laden. Das bedeutet, dass die Elektrofahrzeuge zu Zeitpunkten, wo der Stromtarif niedrig ist, mit Energie versorgt werden. Wird dieses Ziel bei einer höheren Durchdringungsrate der Elektromobilität verfolgt, kommt es wieder zu einer hohen Gleichzeitigkeit der Ladevorgänge und die Netzbelastung wird dadurch nicht verbessert [RPL$^+$12, S.11].

- Die *netzgeführte Strategie* richtet sich nach der im Energienetz bereits vorhandenen Last bzw. Erzeugung. Hier werden die Fahrzeuge zu Zeitpunkten, in denen Erzeugungsreserven vorhanden sind bzw. das Netz wenig belastet ist, geladen (siehe Abb. 2.5 rechts). Durch Verfolgung dieses Zieles kann die Gleichzeitigkeit der Ladevorgänge verringert und das vorhandene Energienetz optimal ausgenutzt werden [RPL$^+$12, S.13].

Die zweite Stufe des unidirektionalen Ladekonzeptes beruht nicht nur auf längerfristigen Prognosen, sondern verwendet zeitnahe Informationen, wie Mess- und kurzfristige Prognosewerte, um das Energienetz optimal auszunutzen. Diese Strategie, auch als geregeltes Laden bezeichnet, kann daher die Netzintegration von regenerativen Energien fördern. Wegen eines höheren technischen Aufwandes wird diese Regelung erst im Zuge einer höheren Durchdringungsrate von Elektrofahrzeugen interessant, wenn das vorher beschriebene Konzept des gesteuerten Ladens an seine Grenzen stößt [RPL$^+$12, S.14].

2.4.3 Bidirektionales koordiniertes Ladekonzept

Im Gegensatz zum unidirektionalen Ladekonzept beschäftigt sich das bidirektionale Ladekonzept mit dem Lade- und dem Entladevorgang von Batterien. Da Elektrofahrzeuge den Großteil der Zeit nicht benutzt werden, können sie bei Anschluss ans Versorgungsnetz als verteilte Energiespeicher angesehen werden. Dadurch wäre es möglich, vor allem in kleineren elektrischen Energienetzen, die Regelausgleichsenergie aus diesen zu beziehen. Diese bidirektionalen Konzepte werden unter dem Begriff Vehicle2Grid zusammengefasst und benötigen einen, an diese Funktion angepassten Laderegler, der die Rückspeisung der Energie in Gegenrichtung erlaubt [KVS13, S.283].

Der Vehicle2Grid-Ansatz kann nur im längerfristigen Kontext gesehen werden, da derzeit die spezifischen Batteriekosten zu hoch und die Ladezyklenanzahl der Batterien zu gering sind [aca10, S.25]. Diese Systeme sind technisch aufwendig zu realisieren und durch die hohen Batteriekosten derzeit unrentabel. Momentan kann die benötigte Stromrückspeisung aus kostengünstigeren Quellen, wie Pumpspeicherkraftwerken, erfolgen [RPL+12, S.13], welche für diesen Zweck bereits technisch ausgelegt und ausgereift sind.

2.5 Simulation im Bereich der Elektromobilität

Elektrofahrzeuge sind heutzutage nur in beschränktem Ausmaß anzutreffen. Daher können viele Erkenntnisse nur durch ausführliche Simulationen gewonnen werden. Die Elektromobilität dringt dabei in verschiedene Domänen bzw. Forschungsbereiche ein, welche alle ihre eigenen, für die entsprechende Aufgabe angepassten, Werkzeuge verwenden.

Die Erforschung von Ladealgorithmen (z. B. [SC12], [BYBJ12]), mit denen eine zeitliche Koordinierung der Elektrofahrzeuge vorgenommen werden kann, unterliegt hauptsächlich dem Aufstellen und Lösen von mathematischen Optimierungsproblemen. Softwareprogramme in diesem Bereich, wie z. B. MathWorks MATLAB [17], dienen lediglich dem Lösen und Aufbereiten dieser mathematischen Probleme.

Um die Auswirkungen der Ladevorgänge von Elektrofahrzeugen in elektrischen Energienetzen nachbilden zu können, werden in diesem Bereich Simulatoren verwendet, welche in der Lage sind, Lastfluss- oder Stabilitätsberechnungen in einem elektrischen Energienetzmodell durchzuführen. DIgSILENT PowerFactory [18] oder GridLAB-D [19] sind Programme, die in diesem Bereich Anwendung finden.

Für diese Simulationen müssen Daten über Fahrzeugbewegungen zur Verfügung gestellt werden, welche möglichst der in der Realität durchgeführten Fahrten entsprechen. Hier kann bereits auf in der Vergangenheit durchgeführte Statistiken von Fahrzeugbewegungen [20] zurückgegriffen werden und die daraus gewonnenen Erkenntnisse auf Elektrofahrzeuge umgelegt werden.
Ebenso gibt es in diesem Bereich spezielle Projekte [21], die mithilfe von Mobiltelefonen den genauen Fahrzyklus, inklusive Beschleunigungs- und Bremsphasen, von echten, mit Verbrennungsmotor angetriebenen Fahrzeugen aufzeichnen. Die so erhobenen Daten werden anschließend unter Verwendung einer speziellen Software zu virtuellen Elektrofahrzeugen simulationstechnisch umgerüstet.

Zur Simulation von Auswirkungen der Ladevorgänge von Elektrofahrzeugen gibt es bereits Simulationssysteme, welche neben den Ladevorgängen hauptsächlich das Fahrverhalten und den damit einhergehenden Energieverbrauch während der Fahrten simulieren. Dabei werden die Fahrten

mithilfe der agenten-basierten Transport-Simulation MATSim [22] nachgebildet. Die darauf aufbauende Simulation zur Untersuchung der Auswirkungen auf das elektrische Energienetz erfolgt mit dem Ladeinfrastruktursimulator EVSim [SU13].

Da wie bereits eingangs erwähnt, viele Simulationstools unterschiedlicher Domänen bei der Simulation im Bereich der Elektrofahrzeuge zum Einsatz kommen, müssen diese miteinander verbunden werden, um Informationen austauschen zu können. Dadurch entsteht eine Co-Simulationsumgebung mit Simulatoren für verschiedenste Bereiche, die synchronisiert betrieben werden müssen. Der Simulation Message Bus (SMB), welcher für die Entwicklung von Regelungskonzepten in Smart Grids entworfen wurde, kann diese Aufgabe übernehmen, indem er eine einfache Schnittstelle bildet, mit der Daten selektiv zwischen unterschiedlichen Softwareprodukten ausgetauscht werden können [MKFS13]. Ebenso lassen sich diese angeschlossenen Simulationsumgebungen mithilfe des SMBs zeitlich synchronisieren.

2.6 Testen von Ladesäulen

Bei der Durchführung von Tests werden Informationen gewonnen, mit denen unter vorgegebenen Rahmenbedingungen der ordnungsgemäße Betrieb eines Systems festgestellt wird.
Derzeit befinden sich am Markt nur Elektrofahrzeug-Simulatoren, welche zur Überprüfung der Installation einer Ladesäule dienen. Diese Simulatoren ([23], [24]) verifizieren unter anderem die Signalisierung über den Pilotkontakt beim IEC 61851-Protokoll oder führen Tests der elektrischen Sicherheitseinrichtungen durch.

Grundsätzlich muss beim Testen eine strukturelle Vorgehensweise eingehalten werden, die auch nachvollziehbar dokumentiert wird. Das Überprüfen von Ladesäulen kann prinzipiell als Black Box-Test angesehen werden, bei dem nur die Funktion und das Verhalten der Ladesäule bekannt ist, jedoch nicht die interne Struktur. Jede vorhandene Schnittstelle der Ladesäule muss mit geeigneten Mitteln verschiedenen Überprüfungen unterzogen werden, damit die korrekte Funktionsweise und die Interoperabilität zwischen verschiedenen Ladesäulen und Ladeinfrastrukturen sichergestellt werden kann.

Aus der Softwareentwicklung stammend, gibt es viele unterschiedliche Testmethoden [Pil12, S.65], die auch auf Systeme, wie Ladesäulen, angewendet werden können. Diese Methoden beziehen sich vor allem auf die Funktionsweise des Systems. Hier können zur Sicherstellung des korrekten Betriebs neben Funktions- und Integrationstests auch Performance- oder Stresstests durchgeführt werden. Daneben müssen noch Überprüfungen für die Einhaltung der sicherheitsrelevanten elektrischen Parameter vollzogen werden, welche hier nicht weiter behandelt werden.

Die Kommunikation von Ladesäulen mit Backend-Systemen wird in Zukunft größtenteils auf Basis von Webservices aufgebaut sein. Diese, in Abschnitt 3.4.5 genauer erläuterten, Services basieren auf einer Client-Server-Architektur, bei der ein Client Funktionen an einem entfernten Server aufruft. Dieser Server kann durch eine Testumgebung ersetzt werden, welche mit entsprechenden Testparametern versehene Funktionsaufrufe abfängt und diese im Anschluss mit einem zu erwartenden Vergleichswert überprüft. Für das Open Charge Point Protocol, welches für diese Kommunikation verwendet werden kann, gibt es bereits die skriptfähigen Programme der *OCPP Testbench* [25], die für das automatisierte Testen des Protokolls eingesetzt werden können.

3 Entwicklung des Architekturmodells

Nachdem im letzten Kapitel ein Überblick über den derzeitigen Stand der Technik im Bereich der Elektromobilität und deren Ladeinfrastruktur gegeben wurde, beschäftigt sich dieser Abschnitt mit dem Entwurf der Softwarearchitektur und den Anforderungen an die einzelnen Komponenten der Simulationsumgebung.

Die Aufgabe dieser Diplomarbeit ist, wie unter Abschnitt 1.3 bereits erläutert, die Entwicklung einer Simulationsumgebung, welche in der Lage ist, gleichzeitig reale und simulierte Ladesäulen zu betreiben und deren Einfluss auf das elektrische Energienetz simulationstechnisch nachbilden zu können. Ebenso soll diese Umgebung für den Ausbau zu einer vollautomatisierbaren Testumgebung für Ladesäulen vorbereitet werden. Um diesem Punkt Rechnung zu tragen, wird die mögliche Weiterentwicklung bereits von Beginn an beim Architekturentwurf berücksichtigt.

Als Erstes werden eine Anforderungsanalyse und die Entwicklung einer rudimentären Architektur für eine Testumgebung von Ladesäulen durchgeführt. Die dabei aufgezeigten Schnittstellen und deren Anforderungen werden beim endgültigen Entwurf der Simulationsumgebung berücksichtigt und bilden den Grundstein, um den Ausbau zur Testumgebung gewährleisten zu können. Anschließend wird ein allgemeines Architekturmodell entwickelt, welches die gleichzeitige Simulation von Elektrofahrzeugen und des elektrischen Energienetzes erlaubt. Die Kombination dieser Modelle resultiert in verschiedenen zu untersuchenden Architekturansätzen. Die Auswahl des geeignetsten Ansatzes bildet die Basis für die zu implementierende Umgebung, dessen Spezifikationen und Anforderungen in diesem Kapitel genauer erläutert werden.

3.1 Architektur einer Testumgebung

Wie schon in Abschnitt 1.2 dargestellt, müssen Ladesäulen getestet und überprüft werden, um Inkompatibilitäten zwischen Produkten unterschiedlicher Hersteller verhindern zu können. Dabei müssen neben der korrekten Funktionsweise der Ladesäule die kommunikations- und die energietechnischen Schnittstellen überprüft werden, um die Interoperabilität gewährleisten zu können.
Am Beginn wird eine Analyse aller vorhandenen Schnittstellen und aller interagierenden Systeme einer Ladesäule durchgeführt, aus der anschließend die erforderlichen Komponenten für das Architekturmodell der Testumgebung abgeleitet werden.

3.1.1 Schnittstellenanalyse einer Ladesäule

Eine Ladesäule für Elektrofahrzeuge besitzt grundsätzlich mehrere Schnittstellen unterschiedlicher Domänen, welche in Abbildung 3.1 dargestellt sind. Die Analyse baut auf einem grundlegenden und abstrakten Modell einer Ladesäule auf, da dies für die allgemeine Darstellung der Schnittstellen bereits ausreichend ist.

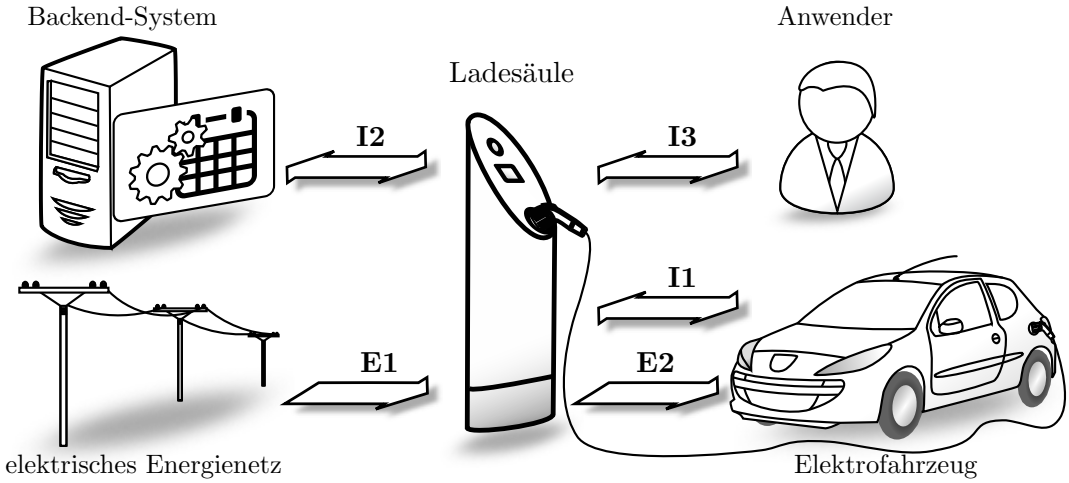

Abbildung 3.1: Schnittstellen einer Ladesäule

Die vorhandenen Schnittstellen lassen sich in zwei große Bereiche einteilen, welche voneinander unabhängige Aufgaben erfüllen. Dabei stellt der Energiepfad den ersten Bereich dar und verbindet das elektrische Energienetz über die Ladesäule mit dem Elektrofahrzeug. Über diesen Pfad wird die benötigte Energie für den Ladevorgang des Elektrofahrzeuges übertragen, welcher von der Ladesäule je nach Anwendung auf unterschiedliche Weise beeinflusst werden kann. Im einfachsten Fall misst die Ladesäule die übertragene Leistung über diesen Pfad. Im elektrischen Energiepfad befinden sich an der Ladesäule zwei Schnittstellen (E1, E2), welche die elektrischen, sowie die sicherheitsrelevanten Anschlussvorgaben erfüllen müssen.
Wie in Abbildung 3.1 ersichtlich ist, wird für die Analyse ein unidirektionaler Stromfluss vom elektrischen Energienetz zum Elektrofahrzeug angenommen, da dies dem Aufbau heutiger Ladesäulen entspricht. Im Falle des bidirektionalen Vehicle2Grid-Ansatzes würde der Energietransport über diesen Pfad in beide Richtungen erfolgen.

Die Schnittstellen des zweiten Bereiches führen die Übertragung von Informationen der Ladesäule zu unterschiedlichen Komponenten durch und werden nachfolgend beschrieben. Dabei wird bei jeder dieser Schnittstellen die für die jeweilige Anwendung angepasste Kommunikationstechnik verwendet. Allgemein betrachtet, gibt es an einer Ladesäule für Elektrofahrzeuge drei Kommunikationskanäle:
Die Schnittstelle I1 übermittelt Informationen zwischen Ladesäule und Elektrofahrzeug und wurde bereits im Abschnitt 2.3.1 genauer erläutert. Die Übertragung erfolgt beispielsweise durch eigene Signalkontakte im Ladekabel (z. B. IEC 61851) oder durch eine Powerline-Kommunikation (z. B. IEC 15118), bei der Informationen über die Fahrzeugidentifikation, erlaubte Ladestrombereiche oder Batteriezustände ausgetauscht werden.
Die Kommunikation über die Informationsschnittstelle I2 erfolgt überwiegend auf Basis des Übertragungsprotokolls TCP/IP und stellt eine Verbindung zu einem übergeordneten Backend-

System her (z. B. Open Charge Point Protocol). Diese Schnittstelle, welche unter Kapitel 2.3.2 bereits dargestellt wurde, wird für die Administration der Ladevorgänge bzw. für zukünftige Lademanagementkonzepte benötigt.
Die letzte zu analysierende Schnittstelle (I3), an der Informationen ausgetauscht werden, stellt die Verbindung zwischen Benutzer und Ladesäule (Mensch-Maschine-Schnittstelle) dar. Sie ermöglicht dem Nutzer die direkte Steuerung des Ladevorganges, sowie die Darstellung des aktuellen Zustandes.

3.1.2 Notwendige Komponenten für ein Testsystem

Die im vorherigen Abschnitt durchgeführte Analyse der Ladesäulenschnittstellen bildet die Ausgangslage, um die notwendigen Komponenten für ein Testsystem festzustellen und dessen erforderliche Architektur entwickeln zu können. Jede der angeführten Schnittstellen benötigt ein eigenes Gegenstück bzw. einen Emulator, mit dem die entsprechende Schnittstelle getestet und beeinflusst werden kann. Die Schnittstellen der unterschiedlichen Bereiche erfordern den Einsatz verschiedener Emulatoren, die zum Teil mit entsprechender Hardware für den Energiepfad ausgestattet sein müssen.
Abbildung 3.2 zeigt die notwendigen Komponenten, um eine Ladesäule für Elektrofahrzeuge testen zu können.

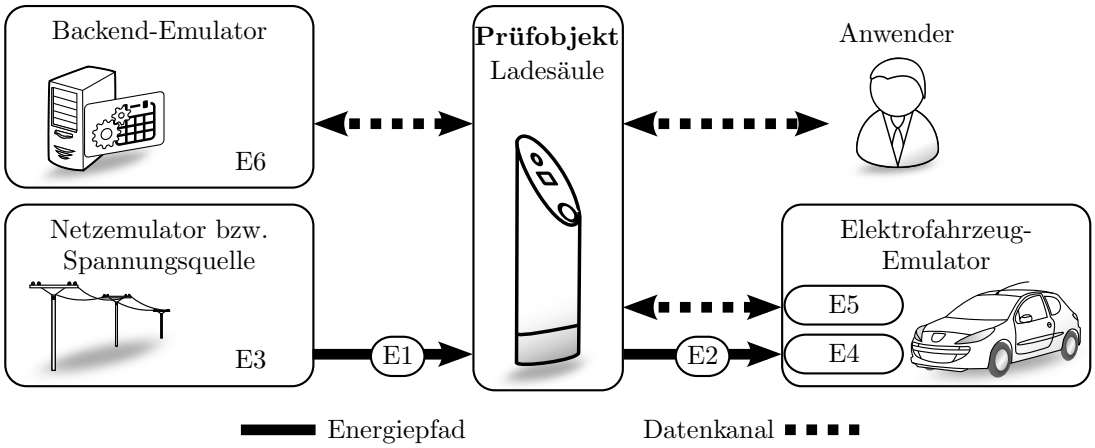

Abbildung 3.2: Notwendige Komponenten für das Testsystem einer Ladesäule

Am Energiepfad einer Ladesäule müssen verschiedene Parameter überprüft werden. Mit Hilfe von externen Messgeräten (E1, E2) wird die Funktion der in der Ladesäule verbauten und zur Abrechnung benötigten internen Messgeräte kontrolliert. Daneben muss die korrekte Funktionsweise der Ladesäule bei jedem möglichen Zustand des elektrischen Energienetzes getestet werden. Dafür sind spezielle Spannungsquellen bzw. Energienetzemulatoren (E3) notwendig, die die Netzspannung bzw. Frequenz am Eingang der Ladesäule vorgeben können. Diese Emulatoren (z. B. der Firma Spitzenberger [26]) sind in der Lage variable Schwankungen der Versorgungsspannung bzw. der Frequenz, sowie praxisrelevante Störungen (transiente Signale) und Unterbrechungen zu erzeugen, um das elektrische Verhalten von Testobjekten bei Störungen ermitteln zu können.
Ebenso muss am elektrischen Ausgang der Ladesäule ein Emulator (E4), welcher zur Simulation des im Fahrzeug verbauten Ladereglers bzw. der Fahrzeugbatterie dient, angeschlossen werden, um auch diese Schnittstelle überprüfen zu können. Dabei wird die für den Ladevorgang typische Last (siehe Kapitel 2.1.3) vorgegeben.

Neben der Überprüfung des elektrischen Energiepfades müssen auch alle kommunikationsrelevanten Schnittstellen getestet werden. Dazu werden Emulatoren benötigt, welche die informationsbasierte Verbindung zwischen Ladesäule und Elektrofahrzeug (E5) bzw. Backend-System (E6) nachbilden. Hierbei muss auf die korrekte Abarbeitung der Schnittstellenprotokolle geachtet werden.

Zusätzlich muss eine Person in die Testumgebung integriert werden, da bestimmte Funktionalitäten (z. B. Authentifizierung oder Anstecken des Ladekabels) nicht automatisiert durchgeführt werden können.

Die soeben erläuterten Komponenten müssen untereinander Nachrichten austauschen und von einer übergeordneten Stelle gesteuert werden, um die automatisierte Testaufgabe erfüllen zu können. Ebenso ist, wie in Abschnitt 1.2 bereits behandelt, nicht nur die Funktionsüberprüfung einer einzelnen Ladesäule notwendig, sondern es muss auch eine Analyse von gegenseitigen Wechselwirkungen mehrerer, an einem Netzabschnitt angeschlossenen, Ladesäulen durchgeführt werden. Daraus resultiert die in Abbildung 3.3 dargestellte Architektur eines Testsystems, die exemplarisch nur zwei Ladesäulen zeigt, um der Übersichtlichkeit Rechnung zu tragen.

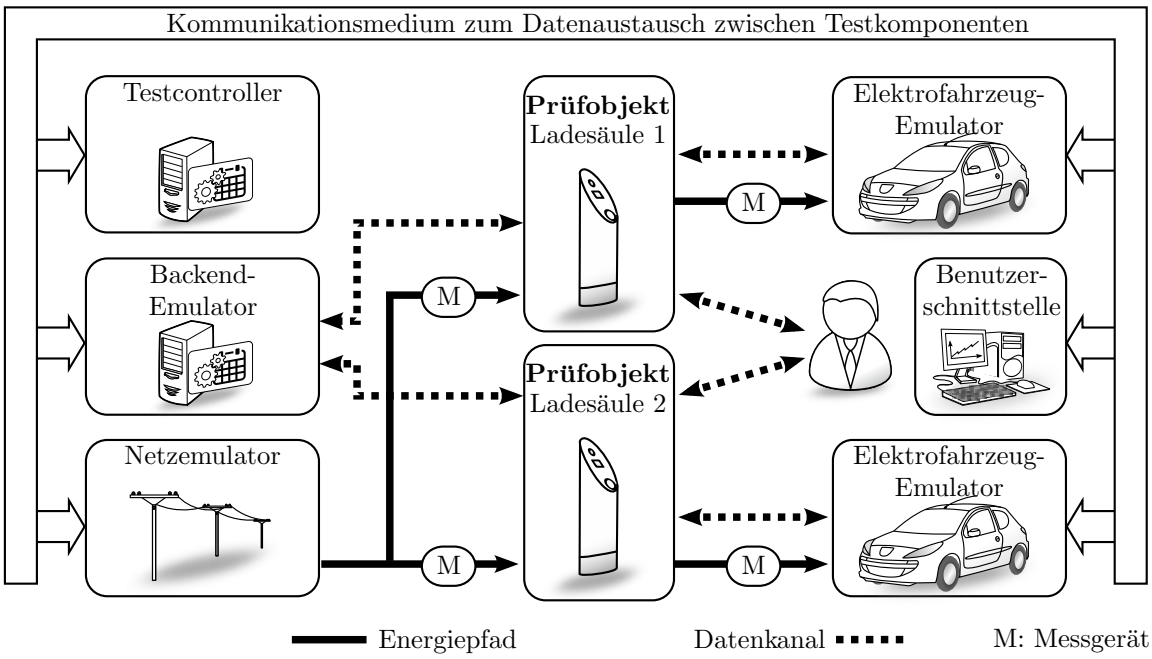

Abbildung 3.3: Architekturvorschlag für ein mögliches Testsystem

In dieser Architektur befinden sich neben den bereits beschriebenen Systemkomponenten noch zusätzliche, welche für die Steuerung und den Nachrichtenaustausch verantwortlich sind. Zu diesen Komponenten zählt ein Testcontroller, der die Abarbeitung und die Auswertung der Testfälle übernimmt, sowie eine grafische Benutzerschnittstelle, die den oben beschriebenen Anwender in die Testumgebung integriert. Zum Nachrichtenaustausch ist ein Kommunikationssystem notwendig, welches die verschiedenen Komponenten und Emulatoren miteinander verbindet.

Der Architektur der Testumgebung muss ein modulares Design zugrunde gelegt werden, damit die Umgebung zügig an neue Gegebenheiten der Ladesäulen angepasst werden kann. Dabei müssen einzelne Komponenten ausgetauscht werden, um z. B. neue Schnittstellenprotokolle einsetzen zu

können. Ebenso müssen unterschiedliche Komponenten verschiedener Hersteller kommunikationstechnisch miteinander verbunden werden, um die automatisierte Testaufgabe zu erfüllen. Diese Designrichtlinie muss beim Entwurf der zu implementierenden Architektur berücksichtigt werden, um die spätere Weiterentwicklung zu einer Testumgebung zu ermöglichen.

3.2 Komponenten einer Simulationsumgebung

Um den Einfluss von Elektrofahrzeugen auf ein elektrisches Energienetz simulieren zu können, wird ein abstrahiertes Modell einer realen Ladeinfrastruktur benötigt, bei der die wichtigsten Teilnehmer nachgebildet werden. Diese, in Abschnitt 2.2 dargestellten, Teilnehmer spiegeln sich zum Teil in der Simulationsumgebung wider. Abbildung 3.4 zeigt diese Komponenten mit ihren Beziehungen, die zur Simulation und zur Darstellung der Auswirkungen der Ladevorgänge von Elektrofahrzeugen auf das elektrische Energienetz eingesetzt werden.

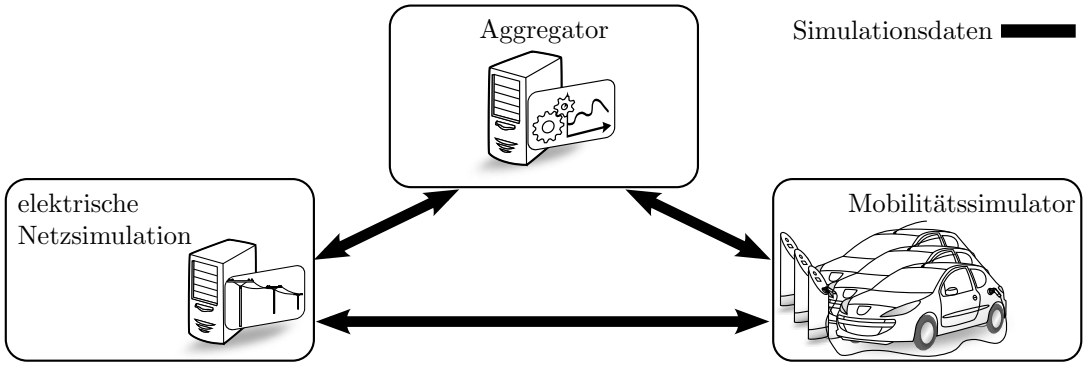

Abbildung 3.4: Komponenten einer Simulationsumgebung

Die in der Abbildung 3.4 gezeigten Bestandteile sind bereits nach ihren Aufgaben getrennt, welche im Folgenden allgemein beschrieben werden.

Netzsimulation

In der Netzsimulation wird ein elektrisches Energienetz nachgebildet, mit dem verschiedene Simulationen, wie Lastflussberechnungen bzw. RMS-Simulationen, durchgeführt werden können. Diesem Teilbereich der Umgebung können Lastprofile von Haushalten oder anderen Verbrauchern hinterlegt werden, um ein elektrisches Energienetz realitätsnah nachzubilden. Ebenso müssen die elektrischen Leistungen, die durch die Ladevorgänge der Elektrofahrzeuge verursacht werden, dieser Netzsimulation übergeben werden.

Mobilitätssimulator

In der Simulationsumgebung dient der Mobilitätssimulator zur Nachbildung der Fahrtbewegungen und der Ladevorgänge von Elektrofahrzeugen. Dafür müssen der Simulation Tagesprofile von Fahrzeugen hinterlegt werden, bei denen die Fahrtrouten und -zeiten bzw. die Standzeiten, in denen geladen werden kann, definiert werden. Ebenso muss die Umgebung die charakteristischen Eigenschaften der Elektrofahrzeuge, wie beispielsweise den durchschnittlichen Energieverbrauch

pro Kilometer oder den Energieinhalt der Fahrzeugbatterie, kennen. Basierend auf diesen Daten generiert die Simulation anschließend virtuelle Fahrten der Fahrzeuge und Lademöglichkeiten, in denen das Fahrzeug an simulierten Ladesäulen angeschlossen wird. Die virtuellen Ladevorgänge werden der vorher beschriebenen elektrischen Energienetzsimulation übergeben, um den durch die Ladevorgänge verursachten Einfluss der Elektrofahrzeuge auf das elektrische Verteilnetz darstellen zu können. Dabei werden die simulierten Ladesäulen in Form von variabel veränderbaren Lasten in der Energienetzsimulation nachgebildet.

Aggregator

Der Aggregator übernimmt die Koordination bzw. Steuerung der Ladevorgänge der Elektrofahrzeuge. Dadurch können verschiedene Algorithmen für ein Lastmanagement in die Simulationsumgebung eingebunden werden. Diese Komponente kann aus der Energienetzsimulation Daten, welche den Zustand des elektrischen Energienetzes beschreiben, verwenden, um reaktiv die Ladevorgänge der Elektrofahrzeuge im Mobilitätssimulator zu beeinflussen. Dadurch lässt sich ein kontinuierlicher Regelalgorithmus implementieren, welcher im Vehicle2Grid-Konzept benötigt wird (siehe Kapitel 2.4.3).

Je nach Anforderung können diese Teilnehmer in einzelne Softwarepakete, die über ein Kommunikationsmedium Daten austauschen, aufgeteilt werden. Es besteht aber auch die Möglichkeit, die gesamte Funktionalität der Komponenten in einem einzigen Softwareprogramm zu vereinen, um den Aufwand, der durch die Kommunikation entsteht, zu vermindern.

3.3 Architekturansätze

Das Ziel dieser Diplomarbeit ist die Entwicklung einer Simulationsumgebung, welche gleichzeitig reale und simulierte Ladesäulen betreiben kann. Außerdem soll die Umgebung auf eine mögliche Erweiterung zu einem Testsystem vorbereitet werden (siehe Abschnitt 1.3). Um diesen letzten Punkt gewährleisten zu können, werden die gewonnenen Erkenntnisse aus der Entwicklung einer Testsystemarchitektur (vgl. Kap. 3.1.2) mit den Komponenten der Simulationsumgebung zusammengeführt. Bei dieser Kombination entstehen verschiedene Möglichkeiten, wie die real aufgebauten Ladesäulen in die Architektur integriert werden können.

In den folgenden Darstellungen wird zur Kommunikation zwischen Ladesäulen und Managementsystem beispielhaft das Open Charge Point Protocol (OCPP) verwendet. Die vorgestellten Konzepte können direkt auf ähnliche Kommunikationsprotokolle für Ladesäuleninfrastrukturen umgelegt werden.

3.3.1 Konzept mit einem gemeinsamen Kommunikationsprotokoll

Das erste Architekturkonzept, welches in Abbildung 3.5 dargestellt ist, setzt das Open Charge Point Protocol zur Kommunikation in der gesamten Simulationsumgebung ein. Hier werden neben den real aufgebauten Ladesäulen auch die simulierten über dieses Protokoll angesprochen. Der in der Abbildung dargestellte Aggregator übernimmt dabei nicht nur die Funktionalität der Koordination der Ladevorgänge, wie im letzten Abschnitt beschrieben wurde, sondern er ist auch als Managementsystem für die Verwaltung der angeschlossenen Ladeinfrastruktur verantwortlich.

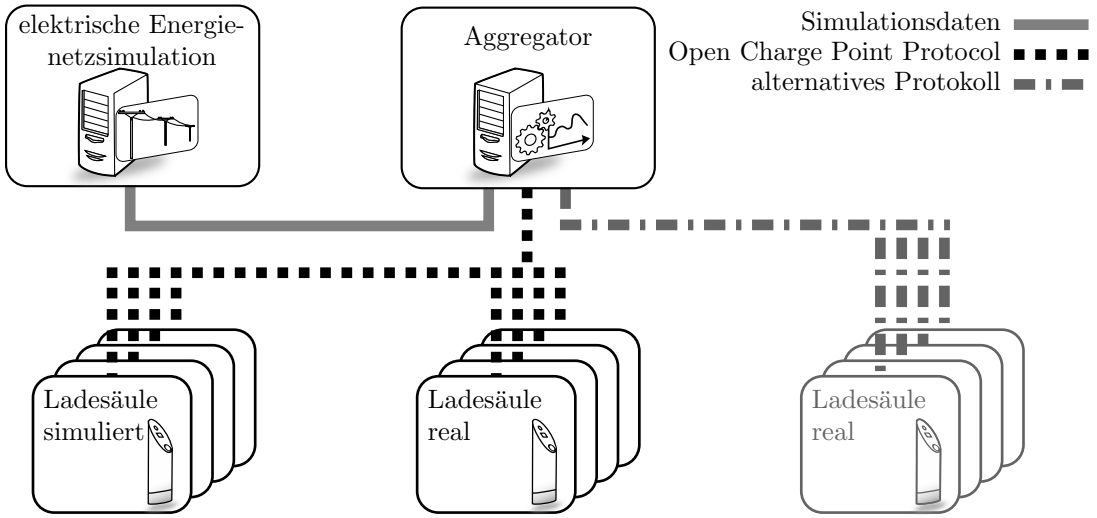

Abbildung 3.5: Konzept mit einem gemeinsamen Kommunikationsprotokoll

Dieses Konzept zeichnet sich besonders durch die Nähe der Simulation zur Realität aus. Unter anderem ist es bei dieser Architektur möglich, Eigenschaften der realen Kommunikation, wie Laufzeitverzögerungen oder Verluste von Nachrichtenpaketen, simulationstechnisch zu erfassen und nachzubilden.

Bei der Einflussanalyse von Ladevorgängen auf das elektrische Energienetz ergibt sich beim Einsatz eines bestehenden und nicht für die Simulation ausgelegten Protokolls ein zusätzlicher Kommunikationsaufwand (Overhead), welcher bei jeder Nachricht mitübertragen werden muss. Ebenso müssen Funktionen des verwendeten Kommunikationsprotokolls in der Simulationsumgebung implementiert werden, obwohl diese für die Anwendung nicht erforderlich sind. Zum Beispiel gehören die Übertragung von Diagnosedaten oder die Aktivierung eines Firmware-Updates der Ladesäule über das Open Charge Point Protocol zu dieser Gruppe der für die Simulation irrelevanten Funktionen.

Die Verwendung von OCPP als quasi-standardisiertes Kommunikationsprotokoll hat den Nachteil, dass Simulationsdaten, wie die Leistungen der Ladesäulen, nicht über dieses Protokoll an den Energienetzsimulator übertragen werden können. Hier muss der Aggregator diesen Nachrichtenaustausch zusätzlich über einen eigenen Kanal und parallel zur bestehenden Kommunikation durchführen.
Ebenso kann bei diesem Ansatz die Einbindung weiterer Protokolle für Ladesäulen nur schwer durchgeführt werden, da diese ebenfalls parallel zum bestehenden Open Charge Point Protocol implementiert werden müssen. Dadurch ergibt sich eine heterogene Kommunikationsstruktur, bei der dieselben Daten über verschiedene Protokolle ausgetauscht werden.

3.3.2 Konzept unter Verwendung eines Protokollkonverters

In der zweiten Architektur wird zur Kommunikation zwischen Ladesäulen und Verwaltungssystem ein eigens definiertes Protokoll eingesetzt. Dieses Protokoll implementiert die Funktionalität, welche für den Datenaustausch in einem Simulationssystem notwendig ist. Abbildung 3.6 zeigt diesen Ansatz.

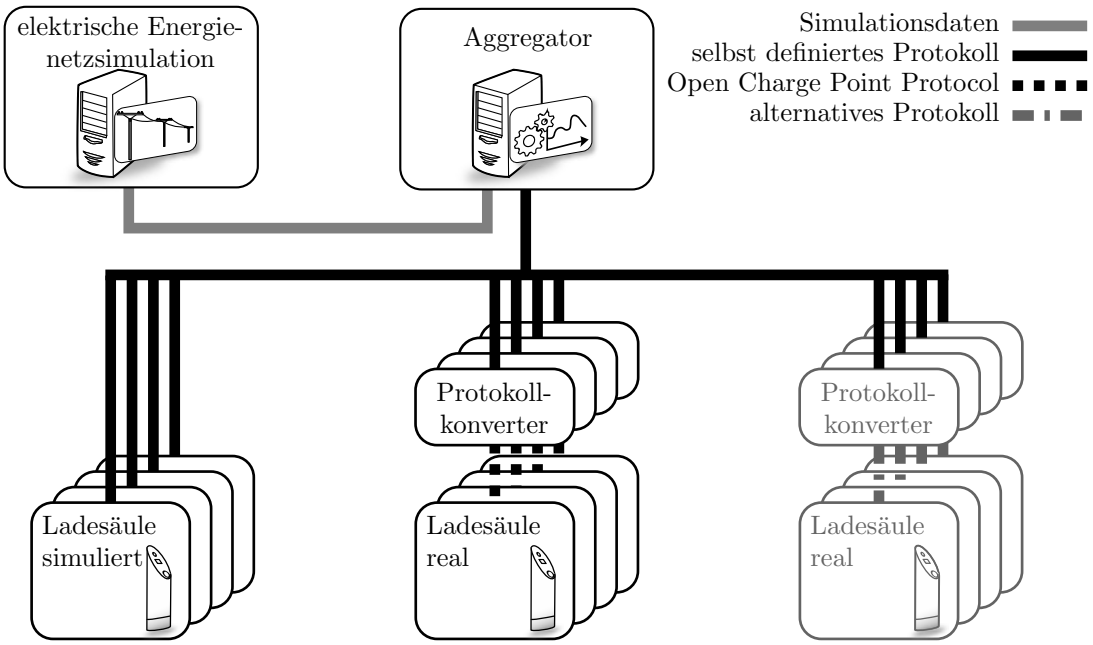

Abbildung 3.6: Konzept unter Verwendung eines Protokollkonverters

Simulierte Ladesäulen können dabei direkt über dieses Protokoll angesteuert werden, wodurch die Nachrichtenübertragung effizient durchgeführt werden kann, da kein unnötiger Protokoll-Overhead, wie beim vorherigen Ansatz, vorhanden ist. Die Kommunikation mit den real aufgebauten Ladesäulen erfolgt weiterhin über das Open Charge Point Protocol. Dabei wird jede Ladesäule mit einem eigenen Protokollkonverter an die Simulationsumgebung angeschlossen. Diese Konverter dienen lediglich der Übersetzung des Protokolls und übernehmen daher keine Verwaltungsaufgaben.

Bei der Verwendung eines speziell für die Aufgabe definierten Protokolls wird die Anbindung der systeminternen Komponenten, wie des Aggregators, vereinfacht, da nur eine einzige einheitliche Schnittstelle implementiert werden muss. In dieser Architektur übernimmt der Aggregator weiterhin, wie auch bei der zuvor beschriebenen Architektur, die Aufgabe der Koordination der Ladevorgänge, sowie die Verwaltung der Ladesäulen. Dabei entsteht die Schwierigkeit, dass unterschiedliche Kommunikationsprotokolle für die Anbindung der realen Ladesäulen über einen ähnlichen Funktionsumfang verfügen müssen, um die Verwaltung im Aggregator vereinheitlichen und den Implementierungsaufwand gering halten zu können. Deshalb ist auch bei diesem Architekturansatz eine Erweiterung der Umgebung, besonders durch die aufwendige Implementierung der Protokollfunktionalität, auf weitere Kommunikationsprotokolle schwer möglich.

3.3.3 Konzept mit getrennter Infrastrukturverwaltung

Die letzte hier dargestellte Architektur teilt die Aufgaben des Aggregators auf. Im Gegensatz zu den vorher beschriebenen Konzepten ist in der in Abbildung 3.7 dargestellten Architektur der Aggregator nur mehr für die Koordination und die Steuerung der Ladevorgänge verantwortlich. Dies entspricht der Grundidee des Aggregators, die in Abschnitt 3.2 bereits erläutert wurde.

Die Verwaltung der Ladesäulen wird in diesem Konzept von speziellen Infrastrukturmanagern übernommen. Durch die Zuordnung eines eigenen Infrastrukturmanagers für jedes Kommunika-

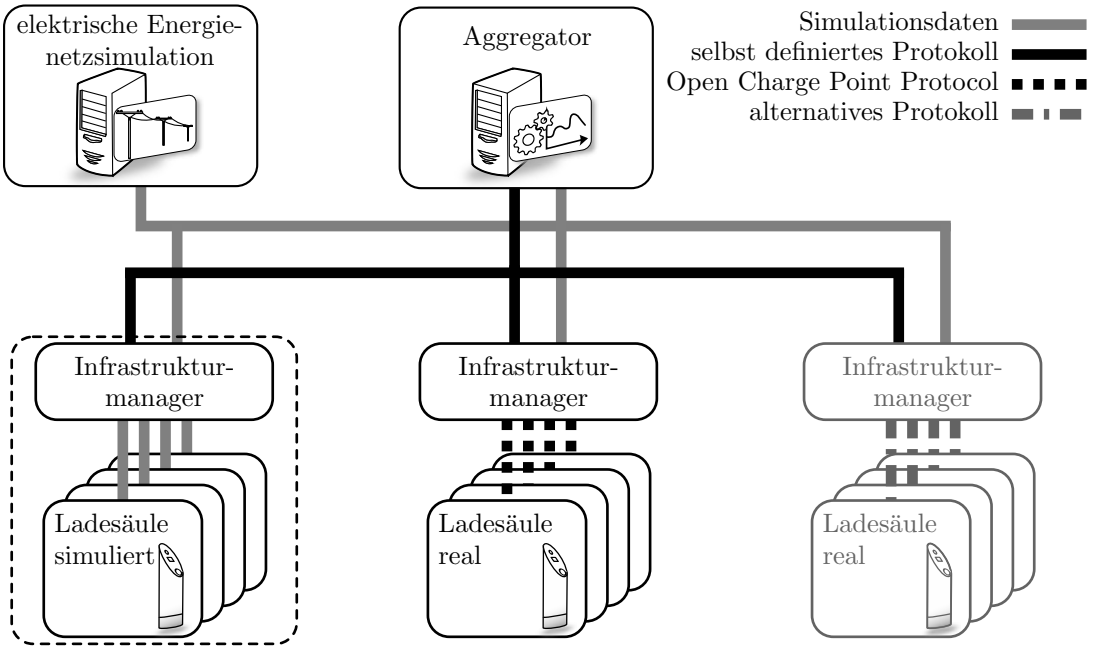

Abbildung 3.7: Konzept mit getrennter Infrastrukturverwaltung

tionsprotokoll kann die schlechte Erweiterbarkeit auf neue Protokolle der bisher beschriebenen Konzepten in dieser Architektur deutlich verbessert werden. Dabei wird jeder Manager auf das implementierte Protokoll abgestimmt. In Richtung Simulationsumgebung werden diese Manager über eine einheitliche Schnittstelle und eine eigens definierte Nachrichtenübertragung angesteuert, wodurch die Implementierung und die Verbindung zum Aggregator vereinfacht werden. Ebenfalls können Simulationsdaten, wie die Leistungen der angeschlossenen Ladesäulen, bereits von den Infrastrukturmanagern an die Energienetzsimulation übertragen werden, wodurch der Aggregator weiterhin von seinen Aufgaben entlastet wird. Durch diese zusätzlichen Komponenten und die Trennung der Aufgabenbereiche des Aggregators erhöht sich aber nachteilig der Kommunikationsaufwand der systeminternen Datenübertragung, da zwischen Infrastrukturmanagern und Aggregator zusätzliche Informationen, wie z. B. Ladepläne, ausgetauscht werden müssen.
Die Anbindung der simulierten Ladesäulen kann bei dieser Architektur unabhängig von den restlichen Komponenten durchgeführt werden. Dabei können, wie in der Abbildung 3.7 strichliert umrahmt dargestellt, der Ladeinfrastrukturmanager, die simulierten Ladesäulen und gegebenenfalls der Mobilitätssimulator für die Fahrprofile der Elektrofahrzeuge in einer einzigen Softwarekomponente implementiert werden. Dadurch entfällt die zusätzliche externe Kommunikation zwischen virtueller Ladesäule und Infrastrukturmanager.

Diese letzte Architektur zeichnet sich besonders durch ihr modulares Design aus, das durch die strikte Trennung der Aufgabenbereiche in eigene Softwarekomponenten die Ausbaufähigkeit und Erweiterbarkeit der Umgebung deutlich verbessert.

3.4 Systemarchitektur der Simulationsumgebung

Um den späteren Ausbau der Umgebung zu einem universellen Testsystem gewährleisten zu können, muss besonders die Möglichkeit zur Anpassung an neue Gegebenheiten im Vordergrund

stehen. Die zuletzt beschriebene Architektur mit getrennten Infrastrukturmanagern stellt für die zu entwickelnde Simulationsumgebung den besten Ansatz dar, weil bereits durch das modulare Design die Ausbaufähigkeit und die Flexibilität des Systems gegeben ist.

Durch die Analyse der notwendigen Komponenten und den Vergleich der verschiedenen Kombinationsmöglichkeiten wurde die Systemarchitektur herausgearbeitet, welche in Abbildung 3.8 gezeigt wird. Dieser schematische Aufbau stellt dabei die zu implementierende Version des Konzeptes mit getrennter Infrastrukturverwaltung dar. Daher kann diese Umgebung zukünftig für verschiedene Szenarien im Bereich der Elektromobilität eingesetzt werden (siehe Abschnitt 4.7).

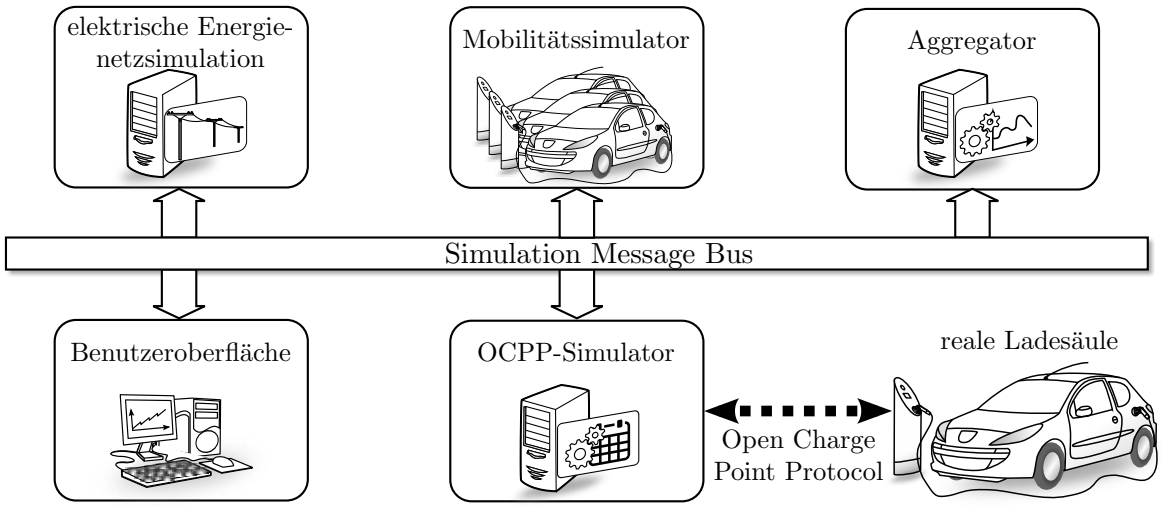

Abbildung 3.8: Systemarchitektur der Simulationsumgebung

Die Eigenschaften der einzelnen in Abbildung 3.8 ersichtlichen Komponenten werden im Anschluss ausführlich erläutert. Einerseits kann in diesem Projekt auf bestehende Softwareprodukte, wie dem Simulation Message Bus und die Netzsimulation, zurückgegriffen werden, bei denen im entsprechenden Abschnitt die Funktionsweise dargestellt wird. Andererseits müssen für diese Architektur eigene Komponenten implementiert werden, um die geforderte Funktionalität der Simulationsumgebung zu erhalten. Die Anforderungen und Aufgaben an diese Komponenten werden ebenfalls in diesem Kapitel ausführlich beschrieben. Die hier dargestellte Architektur mit ihren Anforderungen an die Komponenten bildet den Ausgangspunkt und die Grundlage für die Implementierung, welche im nächsten Kapitel behandelt wird.

3.4.1 Simulation Message Bus

Das modulare Design der Simulationsumgebung erfordert einen umfangreichen Austausch an Simulationsdaten zwischen den einzelnen Komponenten. Die bereits unter Abschnitt 3.1.2 dargestellte Notwendigkeit unterschiedliche Technologieplattformen miteinander zu kombinieren, erfordert ein flexibles System, welches auf einfache Art und Weise an diese unterschiedlichen Simulatoren angepasst werden kann.

Der Nachrichtenaustausch in diesem System basiert auf dem Simulation Message Bus, welcher unter der Kooperation von Siemens Austria, Austrian Institute of Technology und der Technischen Universität Wien entstanden ist [MKFS13].

Der Simulation Message Bus (SMB) ist eine Server Software, die als flexibler Nachrichtenrouter agiert. Er bietet eine lose Kopplung von heterogenen Komponenten, deren gemeinsame Basis eine TCP/IP-Kommunikation bildet. Somit kann er sinnvoll für eine Co-Simulationsumgebung eingesetzt werden, in der unterschiedliche Simulatoren für verschiedene Domänen Nachrichten austauschen müssen. Ursprünglich wurde der SMB für den Einsatz in der Planung und Inbetriebnahme von Regelungskonzepten in Smart Grids entwickelt, um eine durchgängige Kommunikationsplattform von der Simulation bis zum Betrieb im elektrischen Energienetz zu ermöglichen [MKFS13].

Das Routing erfolgt im SMB über sogenannte Channels, welche als Zugriffspunkte für verschiedene Softwarekomponenten dienen. Jeder Channel kann aus mehreren sogenannten Proxies bestehen, welche als Einheiten mit bestimmten Aufgaben angesehen werden können. Im SMB sind bereits Proxies vordefiniert, welche das Routing von Nachrichten oder das Speichern von Logdateien übernehmen. Abbildung 3.9 zeigt eine beispielhafte Konfiguration dieses Kommunikationssystems, welches aus mehreren Channels mit verschiedenen Proxies besteht [MKFS13, S.3].

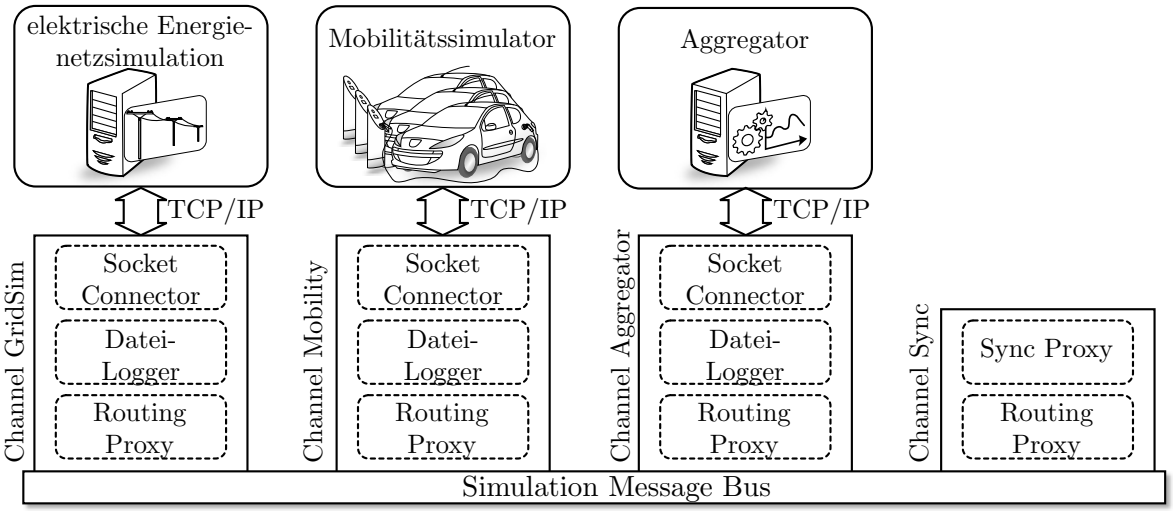

Abbildung 3.9: Beispielhafte SMB-Konfiguration (vgl. [MKFS13])

Die Konfiguration erfolgt statisch vor dem Betrieb des SMBs über eine XML-Konfigurationsdatei. Diese beschreibt den genauen Aufbau und enthält alle vorhandenen Channels und deren Proxies. Ebenso werden in dieser Datei Informationen für das Routing hinterlegt, mit denen die Nachrichten im System verteilt werden können (Beispielkonfiguration im Anhang A.2). Dabei kann auf verschiedene Mechanismen zurückgegriffen werden, bei denen Nachrichten entweder simultan an mehrere Channels bzw. Komponenten gesendet werden oder Hop-by-Hop von Teilnehmer zu Teilnehmer weitergereicht werden [FDLK13, S.9].

Der Simulation Message Bus ist neben der bereits beschriebenen Nachrichtenübermittlung auch für die zeitliche Synchronisation der angeschlossenen Simulatoren zuständig. Dafür kann in einem speziell dafür vorgesehenen Channel des SMBs ein Proxy zur Synchronisierung (Abb. 3.9: Channel Sync) instanziert werden. Der SMB unterscheidet dabei zwischen zwei verschiedenen Betriebsarten [MKFS13]:
Im *Simulationsmodus* wird die Simulationszeit in Zeitschlitze eingeteilt, wobei jeder Startzeitpunkt eines Zeitschlitzes mithilfe einer speziellen Nachricht vom Synchronisations-Proxy initiiert wird. Die Simulation läuft, abhängig von der Hardware, schnellstmöglich ab und es treten keine

Abweichungsfehler in der Synchronisation auf. Ein Bezug zur Realzeit ist in diesem Betriebsmodus nicht möglich.
Die zweite Betriebsart dient der *Emulation* und wird benötigt, falls Hardware-Komponenten an die Simulationsumgebung angeschlossen werden. Bei der Emulation wird lediglich der Startzeitpunkt aller Komponenten bestimmt und die Synchronisation der Einheiten erfolgt über die interne Rechnerzeit des jeweiligen Computers. Durch Angabe eines speziellen Faktors ist der SMB in der Lage die Emulation zu beschleunigen. In dieser Betriebsart kann es zu einem Auseinanderdriften der Zeiten zwischen den einzelnen Simulatoren und somit zu geringfügigen Abweichungen in der Synchronisation kommen.

3.4.2 Netzsimulation

Wie bereits in Abschnitt 3.2 erwähnt, dient die Netzsimulation zur Darstellung und Simulation des elektrischen Verteilnetzes. Im hier entworfenen System basiert diese auf der Software *PowerFactory* von DIgSILENT [18]. Durch den sogenannten „Engine Mode“ lässt sich PowerFactory durch externe Software steuern [27] und ermöglicht so die Einbindung in eine Co-Simulationsumgebung mit verschiedenen Programmen bzw. Simulatoren. In PowerFactory können über diese externe Schnittstelle die Leistungen der variablen Lasten während der andauernden Simulation gesetzt werden. Ebenso lassen sich in PowerFactory berechnete Spannungen auslesen, um sie für eine mögliche reaktive Regelung der Ladevorgänge, wie unter Abschnitt 3.2 dargestellt, verwenden zu können.

Die Netzsimulation erlaubt die Überwachung von kritischen Punkten im elektrischen Energienetz, um die vorgegebenen Restriktionen der Verteilnetzarchitektur, wie sie in Kapitel 2.1.2 erläutert wurden, einzuhalten. Dadurch können mögliche Überlastungen des elektrischen Energienetzes erkannt und bewertet werden.

Die Anbindung der elektrischen Netzsimulation an die Simulationsumgebung erfolgt durch die am Austrian Institute of Technology entwickelte Software *GridSim*, welche PowerFactory mit dem Simulation Message Bus verbindet [FDLK13]. Der Nachrichtenaustausch mithilfe des SMBs erlaubt das einfache ferngesteuerte Setzen der Leistungen in der Energienetzsimulation. Außerdem lassen sich auf diese Weise Spannungen von kritischen Knotenpunkten, welche in PowerFactory während einer Simulation berechnet werden, auslesen und für die restliche Simulationsumgebung zur Verfügung stellen.
Die Software GridSim ist ebenso in der Lage, die vorher beschriebene zeitliche Synchronisation zu ermöglichen. Dadurch arbeitet PowerFactory simulationstechnisch zeitgleich mit weiteren Simulatoren der Umgebung zusammen.

3.4.3 Mobilitätssimulator

Der Mobilitätssimulator, der in dieser Arbeit entwickelt werden muss, hat die Aufgabe eine grundlegende Ladeinfrastruktur für Elektrofahrzeuge nachzubilden. Damit ist es möglich fiktive Ladevorgänge in einem elektrischen Verteilnetz zu simulieren und dessen Belastung durch die Ladevorgänge darzustellen. Die Tatsache, dass heute noch zu wenige Elektrofahrzeuge auf den Straßen unterwegs sind, die für umfangreiche Beobachtungen der Auswirkungen verwendet werden können, macht diese Simulation notwendig. Außerdem verfügt die heute vorhandene Ladeinfrastruktur für Elektromobilität noch nicht über die Größe und Verbreitung, um zukünftige Belastungen in bestimmten Regionen vorhersagen zu können.

In der Mobilitätssimulation werden zwei grundlegende Bereiche unterschieden. Einerseits muss eine virtuelle Ladeinfrastruktur geschaffen werden, die in der Lage ist, eine Lademöglichkeit für simulierte Elektrofahrzeuge zu bieten. Auf der anderen Seite müssen die Fahrtbewegungen bzw. Ladezeiten von Elektrofahrzeugen generiert und simulationstechnisch abgearbeitet werden.

Virtuelle Ladeinfrastruktur

Nach Analyse des Open Charge Point Protocols (siehe Abschnitt 3.4.5) und der Fahrprofile der Software EVSim (siehe Kapitel 2.5), dessen Profildateien von der hier entwickelten Simulationsumgebung übernommen werden sollen, bildet sich ein zweistufiger hierarchischer Aufbau der Ladeinfrastruktur heraus. Dabei kann die Ladeinfrastruktur in Ladestationen und -punkte eingeteilt werden.
Ladepunkte können als Anschlussmöglichkeit (z. B. Ladekabel) für ein Elektrofahrzeug betrachtet werden und verfügen bereits über bestimmte Eigenschaften, wie den maximalen Ladestrom, der durch den Ladepunkt oder das angeschlossene Ladekabel fließen darf. Trotzdem sind Ladepunkte keine selbstständigen Objekte, sondern immer einer Ladestation zugeordnet.
Eine Ladestation bildet daher eine größere Einheit, in der mehrere Ladepunkte zusammengefasst werden. Diese Zusammenfassung beeinflusst positiv den Aufwand der Administration in der Initialisierungsphase der Simulation. Eine Ladestation ist in der Lage, die Leistungen, die durch die Ladevorgänge an den ihr zugeordneten Ladepunkten verursacht werden, zu summieren und als Nachricht an die elektrische Energienetzsimulation zu senden. Sie wird in der Energienetzsimulation (siehe Abschnitt 3.4.2) als extern steuerbare Last implementiert, bei der die Leistung gesetzt bzw. die berechnete Spannung an diesem Punkt ausgelesen werden kann.

Um den Bezug zur Realität herzustellen, kann die hierarchische Konstruktion aus Ladestationen und -punkten verschiedenen Ladeinfrastrukturtypen zugeordnet werden. Die Simulation der Auswirkungen, die in einem elektrischen Energienetz untersucht werden sollen, können je nach Anwendungsfall verschiedene Abstraktionslevel der Ladeinfrastruktur annehmen. Daher lässt sich die virtuelle Ladeinfrastruktur an diese unterschiedlichen Ebenen anpassen. Abbildung 3.10 zeigt zwei beispielhafte Ausprägungen der Infrastruktur.

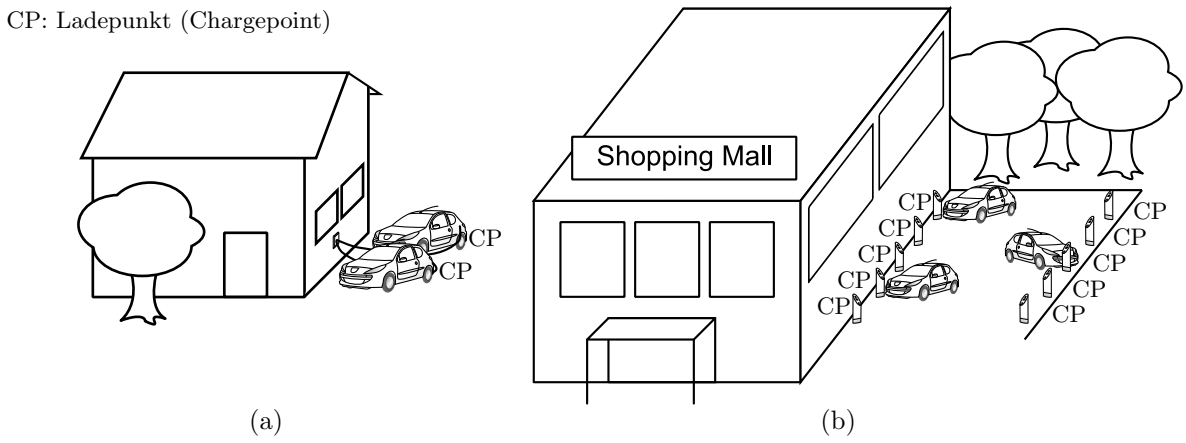

Abbildung 3.10: Anwendungsfälle für Ladestationen und -punkte

Eine virtuelle Ladestation kann zum Beispiel eine einfache reale Ladesäule mit zwei Anschlüssen (a) in einem Einfamilienhaus repräsentieren. Ebenso wäre es möglich, für eine weitere Untersuchung alle Lademöglichkeiten eines gesamten Parkplatzes (b) mit mehreren hundert Ladepunkten (Anschlüssen) einer einzigen virtuellen Ladestation zuzuordnen.

Virtuelle Elektrofahrzeuge

Neben der Ladeinfrastruktur müssen im Mobilitätssimulator auch die Eigenschaften der Elektrofahrzeuge simuliert werden. Dabei besitzen diese Fahrzeuge verschiedene Parameter, wie den durchschnittlichen Energieverbrauch pro Kilometer oder den maximalen und minimalen Ladestrom, welche den Betriebsbereich der Ladeleistung vorgegeben. Der Zustand des Elektrofahrzeuges wird über die Batteriekapazität und den Ladestand gekennzeichnet.
Zusätzlich zu den beschriebenen Eigenschaften muss für jedes Fahrzeug ein Fahrprofil hinterlegt werden, damit unterschiedliche Aktionen während der Simulation durchgeführt werden können. Ein typisches Profil für solch eine Fahrtbewegung wird in Tabelle 3.1 dargestellt.

Ort	Aktion	Zeitpunkt	Fahrtstrecke
Residential	Charging	2014-01-16 00:00:00	
Road	Moving	2014-01-16 06:15:00	18 km
Industrial	Charging	2014-01-16 06:33:00	
Road	Moving	2014-01-16 13:49:00	13 km
Residential	Charging	2014-01-16 14:02:00	

Tabelle 3.1: Typisches Fahrprofil

Grundlegender Simulationsablauf

Das Zusammenspiel der virtuellen Elektrofahrzeuge mit den entsprechenden Ladestationen beruht auf einer durch die Simulationszeit vorgegebenen Ablaufsteuerung. Dabei können die Fahrtbewegungen der Fahrzeuge als Eingangsgrößen betrachtet werden, die zum gegebenen Simulationszeitpunkt eine Zustandsänderung des Systems verursachen. Je nach Ort und Aktion wird dabei ein Elektrofahrzeug entweder an eine Ladestation angeschlossen oder von ihr entfernt. Ein Ladevorgang muss dem Aggregator (siehe Kapitel 3.4.4) gemeldet werden, um – wenn vorgesehen – einen individuellen Ladeplan für ein mögliches Lademanagement zu generieren.
Ebenso muss der Ladestand der Batterie nach jeder durchgeführten Fahrt mithilfe des durchschnittlichen Kilometerverbrauchs und der Fahrtstrecke, wie sie in der Tabelle 3.1 angeführt ist, neu berechnet werden.

Anforderungen

Nachdem der grundlegende Aufbau und die Funktionsweise des Mobilitätssimulators gezeigt wurden, wird hier auf die Anforderungen und Eigenschaften eingegangen, die von der Implementierung dieses Simulators (siehe Abschnitt 4.2) erfüllt werden müssen:

A-1 Es muss die Möglichkeit bestehen, bereits vorhandene Daten von Elektrofahrzeugen und ihren Fahrprofilen aus anderen Programmen, wie z. B. EVSim (siehe Kapitel 2.5), im Mobilitätssimulator verwenden zu können.

A-2 Der Simulator muss in der Lage sein, die Leistungen seiner instanzierten Ladestationen an den elektrischen Energienetzsimulator zu senden und von diesem die berechneten elektrischen Spannungen, die an den Anschlusspunkten der Ladestationen entstehen, zu empfangen bzw. abzufragen.

A-3 Wird ein virtuelles Elektrofahrzeug an eine Ladestation angeschlossen, muss der Aggregator mit aktuellen Informationen zu Abfahrtszeitpunkt und benötigter Energiemenge versorgt werden, um einen Ladeplan für den Ladevorgang generieren zu können, der anschließend vom Mobilitätssimulator empfangen und weiterverarbeitet wird.

A-4 Ebenso soll der Simulator im Stande sein, Ladepläne, die sich reaktiv während der Ladedauer des Elektrofahrzeuges ändern, zu übernehmen und für den weiteren Ladevorgang zu aktualisieren.

A-5 Für die Simulation der Ladeinfrastruktur ist es notwendig, dass verschiedene maximale Leistungsgrenzen der einzelnen Teilnehmer, wie zum Beispiel der Ladesäule oder des Ladereglers im Elektrofahrzeug, berücksichtigt werden, um die in der Realität auftretenden Restriktionen (siehe Abschnitt 2.1.2) in der Simulation erfassen zu können.

A-6 Die in Kapitel 2.1.3 angeführte Abhängigkeit der Ladeleistung vom Ladestand der Fahrzeugbatterie soll ebenfalls in der Simulation nachgebildet werden.

A-7 Die Profildaten der Fahrzeuge, welche in der Simulation verwendet werden, sowie weitere Programmparameter sollen in Form von Konfigurationsdateien gespeichert werden, um eine einfache Simulation ohne grafische Benutzeroberfläche zu erlauben.

3.4.4 Aggregator

Nachdem der Mobilitätssimulator keine Koordination der Ladevorgänge übernimmt, wurde in der Simulationsumgebung eine eigene Komponente – der Aggregator – eingeführt, um dieser Aufgabe nachzukommen. Diese Komponente, welche bereits in Abschnitt 2.2 kurz dargestellt wurde, beschränkt sich hier nur auf die Koordination von Ladevorgängen in einer Ladeinfrastruktur für Elektromobilität. Dabei wird nicht auf die Koordination aller, wie dies in zukünftigen Smart Grids unter dem Überbegriff Demand Side Management angedacht ist, flexibler Lasten, welche in einem elektrischen Energienetz vorkommen können, eingegangen.

Elektrofahrzeuge im Mobilitätssimulator, die an das simulierte elektrische Energienetz angeschlossen werden und den Ladevorgang starten wollen, melden sich beim Aggregator an und erhalten von diesem einen Ladeplan. Dieser Ladeplan setzt sich aus Zeitabschnitten mit vorgegebenen Ladeleistungen zusammen, welche vom Laderegler eingehalten werden müssen.

Der modulare Ansatz, wie er in Abschnitt 3.1.2 gefordert wird, findet auch bei dieser Komponente Anwendung. Die Modularität ermöglicht den universellen Einsatz, wodurch der Aggregator in der Lage ist, Ladevorgänge von mehreren Mobilitätssimulatoren gleichzeitig zu koordinieren. Abbildung 3.11 zeigt dieses Zusammenspiel der unterschiedlichen Komponenten. Neben einem Mobilitätssimulator lässt sich auch der im nächsten Abschnitt dargestellte OCPP-Simulator an das

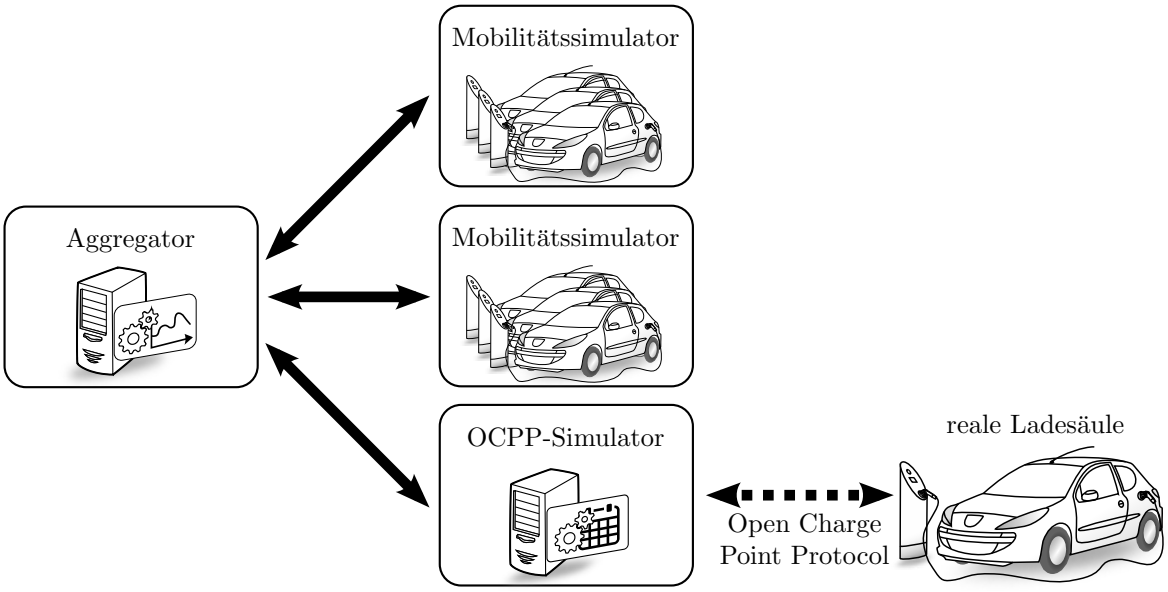

Abbildung 3.11: Modularer Aggregator

System anschließen und die von ihm verwalteten Ladevorgänge vom Aggregator koordinieren. Dadurch entsteht eine gemeinsame Koordination von simulierten und real aufgebauten Ladesäulen, die durch einen einzigen Algorithmus im Aggregator durchgeführt wird.

Die Lademanagementalgorithmen, welche im Aggregator implementiert werden, können sich je nach Anwendungsfall stark voneinander unterscheiden. Grundsätzlich haben diese Algorithmen ein Optimierungsziel, das von finanziellen Vorteilen des Kunden, über die maximale Nutzung von regenerativen Energien bis zur gleichmäßigen Auslastung des elektrischen Energienetzes reichen kann (siehe Abschnitt 2.4). Trotzdem müssen auch bei diesen Optimierungen die Restriktionen des Energienetzes, wie die Einhaltung des zulässigen Spannungsbandes oder die thermischen Grenzen der Übertragungskomponenten, berücksichtigt werden (siehe Kapitel 2.1.2).
Die eingesetzten Algorithmen können auf aktuelle Zustände des elektrischen Energienetzes (Spannungen an bestimmten Knotenpunkten) zurückgreifen und eine regelnde Strategie verfolgen, bei der die Ladepläne der Elektrofahrzeuge reaktiv angepasst werden. Ladepläne, die sich während der Laufzeit nicht verändern, können aber auch proaktiv am Beginn des jeweiligen Ladevorganges erstellt werden.

Anforderungen

Bei der Implementierung des Aggregators, welche im Abschnitt 4.3 gezeigt wird, müssen folgende Anforderungen erfüllt werden, um den universellen Einsatz unterschiedlicher Managementalgorithmen zu erlauben.

A-1 Der Aggregator muss bei der Anmeldung eines neuen Ladevorganges des Elektrofahrzeuges einen Ladeplan erstellen, welcher an den entsprechenden Mobilitätssimulator retourniert wird.

A-2 Elektrofahrzeuge, die sich beim Aggregator angemeldet haben, müssen im System gespeichert werden, um für Algorithmen, die den Ladeplan auch während des Ladevorganges

verändern können, zur Verfügung zu stehen. Nur dadurch sind reaktive Ladealgorithmen möglich, die den Ladeplan je nach Zustand des Gesamtsystems aktualisieren.

A-3 Spannungen an kritischen Punkten in der elektrischen Energienetzsimulation sollen aus dieser ausgelesen werden, um für reaktive Ladealgorithmen eingesetzt werden zu können. Dadurch kann der Aggregator, auf den aktuellen Zustand des simulierten Energienetzes zurückgreifen und die Ladepläne der Elektrofahrzeuge dementsprechend anpassen.

A-4 Der Aggregator soll verschiedene Lademanagementstrategien implementieren, welche für die entsprechende Simulation ausgewählt werden können. Dadurch soll eine einfache Adaptierung an neue Simulationen möglich sein, um den Einfluss verschiedener Koordinationsalgorithmen auf das elektrische Energienetz miteinander vergleichen zu können.

A-5 Der Aggregator soll, wie der bereits beschriebene Mobilitätssimulator, die Einstellungen der verwendeten Algorithmen in Konfigurationsdateien speichern, um ein Betreiben der Simulationsumgebung ohne grafische Oberfläche zu ermöglichen.

3.4.5 OCPP-Simulator

Die Simulationsumgebung muss die Möglichkeit bieten, reale Ladesäulen anzusprechen. Als Infrastrukturmanager übernimmt der OCPP-Simulator diese Aufgabe, indem er eine Kommunikationsschnittstelle zu hardwaremäßig aufgebauten Ladesäulen über das Open Charge Point Protocol bietet. Dadurch können eine Verbindung und ein Informationsaustausch zwischen Simulationsumgebung und real aufgebauter Ladeinfrastruktur mit mehreren Ladesäulen für Elektrofahrzeuge hergestellt werden.
Die Leistungen, die beim Ladevorgang an den realen Ladesäulen übertragen werden, werden der elektrischen Netzsimulation mitgeteilt und können in dieser weiterverwendet werden.

Bevor auf den Aufbau und die Anforderung für die Implementierung des OCPP-Simulators eingegangen wird, soll nachfolgend das Open Charge Point Protocol kurz erläutert werden.

Open Charge Point Protocol

Im Abschnitt 2.3.2 wurden bereits allgemeine Informationen zum Open Charge Point Protocol (OCPP) gegeben. Hier wird nun die technische Funktionsweise dieses Protokolls erläutert, welche für die Entwicklung des Simulators notwendig ist.

Das Open Charge Point Protocol ist für die Übertragung mittels SOAP (Simple Object Access Protocol, veraltet) über das Hypertext Transfer Protocol (HTTP) entworfen worden. Um eine Kommunikation aufzubauen, müssen Webservices definiert werden, die eine interoperable Maschine-Maschine-Kommunikation über ein Netzwerk erlauben [28, Kap. 1.4]. Beschrieben werden diese Webservices mithilfe der Web Services Description Language (WSDL), welche alle für die Kommunikation notwendigen Informationen zur Schnittstelle und zu den übertragenden Daten und Datentypen in maschinenlesbarer Form als Datei bereitstellt [29, Kap. 1]. Diese Dateien können durch entsprechende Frameworks in Bibliotheken konvertiert werden, um sie in verschiedenen Programmiersprachen verwenden zu können.

Das Open Charge Point Protocol wird in Form von zwei WSDL-Dateien ausgeliefert. Dabei werden verschiedene Webservices (*Chargepoint* und *Centralsystem*) definiert, welche die gesamte

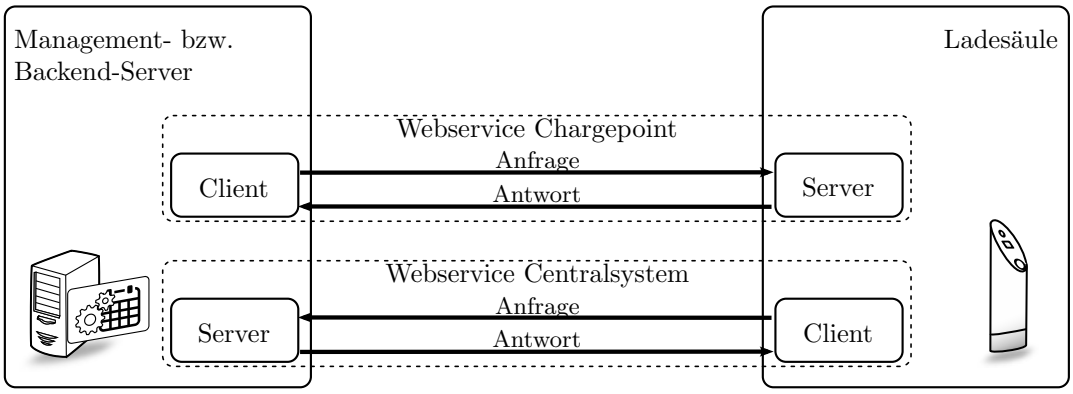

Abbildung 3.12: Webservices des OCPP

Kommunikation zwischen Ladesäulen für Elektrofahrzeuge und einem Managementserver bzw. Backend-Server beschreiben.

Abbildung 3.12 zeigt dieses Zusammenspiel beider Komponenten. Da ein Webservice nur entfernte Funktionsaufrufe durchführen kann, werden beim OCPP zwei Services eingeführt, um eine bidirektionale Kommunikation durchführen zu können. Jeder der beiden Komponenten, die Ladesäule und der Managementserver, implementieren jeweils einen Server, der das entsprechende Webservice zur Verfügung stellt und auf Anfragen des Clients reagiert. Ebenso fungiert jede Komponente als Client, um die Funktionen des gegenüberliegenden Servers aufrufen zu können.

In dieser Arbeit wird die derzeit aktuelle Protokollversion 1.5 verwendet, welche hauptsächlich für die Authentifizierung und die Abrechnung des Ladevorganges ausgelegt ist. Ebenso sind vorzeitige

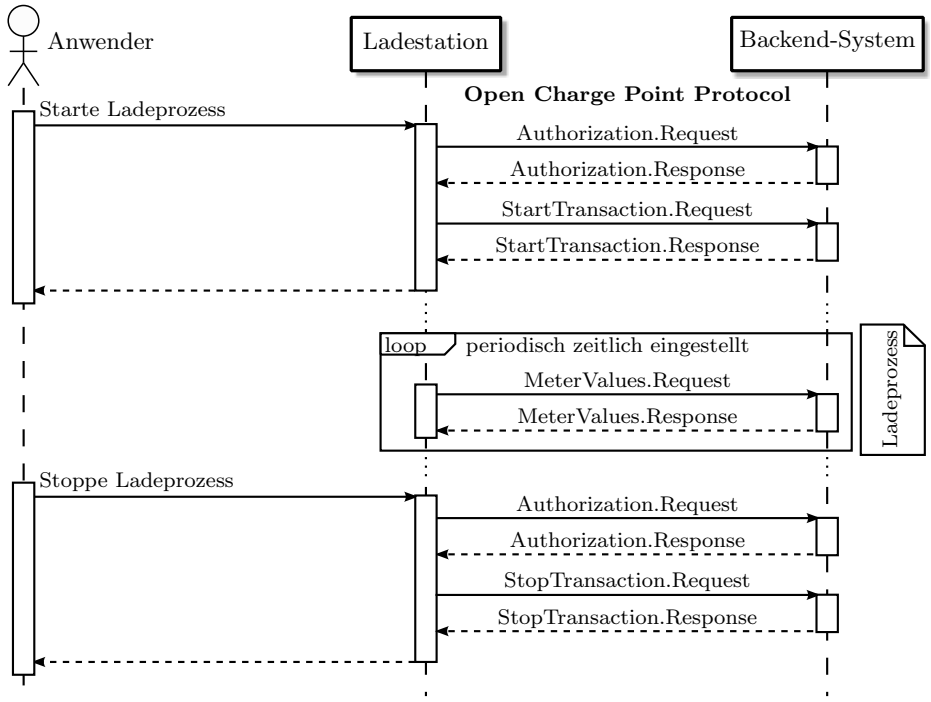

Abbildung 3.13: Typischer Kommunikationsablauf eines Ladevorganges (vgl. [12, S.13])

Reservierungen der Ladesäule möglich. Ein intelligentes Lastmanagement für die Koordination der Ladevorgänge kann mit dieser Version hingegen noch nicht durchgeführt werden (siehe Kapitel 2.3.2).

Abbildung 3.13 zeigt eine typische Abfolge der Kommunikation während eines Ladevorganges. Zu Beginn muss sich der Anwender authentifizieren (Authorize), um den Zugang zur Lademöglichkeit zu erhalten. Danach wird der Ladevorgang gestartet (StartTransaction), indem aktuelle Zustandsdaten der Ladesäule an den Managementserver übertragen werden. Zu diesen Daten gehören unter anderem der Startzeitpunkt, sowie die aktuellen Stände der Energiezähler. Während des Ladens werden laufend, nach einem voreingestellten Intervall, Messdaten an den Server übertragen (MeterValues). Um den Ladevorgang beenden zu können (StopTransaction), muss sich der Benutzer erneut authentifizieren.

Aufbau des OCPP-Simulators

Die interne Struktur und der Aufbau des OCPP-Simulators lehnen sich stark an den vorher beschriebenen Mobilitätssimulator (Abschnitt 3.4.3) an. Im Gegensatz zu diesem werden dabei keine virtuellen Elektrofahrzeuge nachgebildet, sondern der Simulator stellt eine Verbindungen zu realen Ladesäulen her.

Der Simulator stellt, vom Prinzip her, einen intelligenten Protokollumsetzer mit interner Datenverwaltung dar (Infrastrukturmanager). Neben den Komponenten aus dem Mobilitätssimulator, welche für die Kommunikation mit dem Simulation Message Bus notwendig sind, wird der Rest der Architektur vom Open Charge Point Protocol vorgegeben. Abbildung 3.14 zeigt diesen Aufbau, welcher dem OCPP-Simulator zugrunde liegt.

Das Konzept der virtuellen Ladeinfrastruktur mit ihren Ladestationen und -punkten, welches bereits im Mobilitätssimulator (siehe Abschnitt 3.4.3) Anwendung findet, kann auch im OCPP-Simulator eingesetzt werden. Dadurch entsteht ein einheitlicher Aufbau (Datenmodell) der beiden Simulatoren und der Verwaltungsaufwand in der Simulationsumgebung wird verringert.
Zur Kommunikation mit den realen Ladesäulen müssen die vorher erläuterten Webservices in den

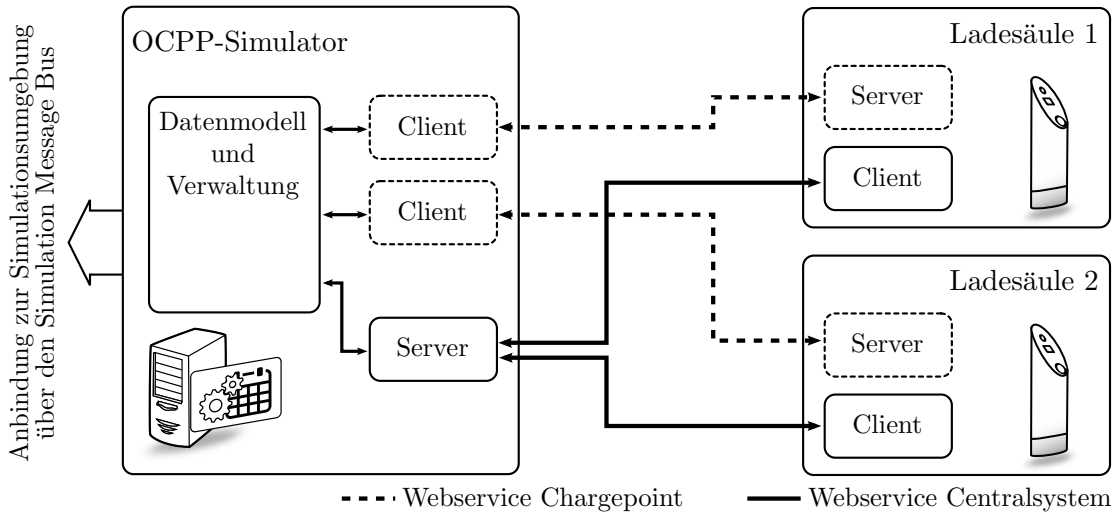

Abbildung 3.14: Architektur des OCPP-Simulators

Simulator eingebunden werden. Dabei stellt der OCPP-Simulator den Server für das Centralsystem-Webservice (Abb. 3.14: durchgezogene Linien) bereit, den die Ladesäulen für Anfragen über das Open Charge Point Protocol verwenden. Ebenso müssen die Funktionen der Chargepoint-Webservices (Abb. 3.14: strichlierte Linien), welche von den Ladesäulen bereitgestellt werden, abgefragt und angesteuert werden. Dafür muss für jede angeschlossene Ladesäule im Simulator ein eigener Client instanziert werden, der diese Kommunikation übernimmt.

Grundlegender Simulationsablauf

Der grundlegende Ablauf wird bei diesem Simulator in großen Teilen durch das Open Charge Point Protocol bestimmt. Dieser Ablauf wurde anhand eines Beispiels bereits in Abbildung 3.13 auf Seite 37 gezeigt und findet sich in leicht geänderter Form im Simulator wieder.

Ein Elektrofahrzeug wird an die reale Ladesäule angeschlossen. Der Anwender muss sich an der Ladesäule authentifizieren und diese sendet die entsprechende Nachricht an den OCPP-Simulator. Der Simulator erlaubt *jedem* den Ladevorgang, welcher anschließend gestartet werden kann. Während des Ladevorganges sendet die Ladestation laufend aktuelle Messdaten an den OCPP-Simulator. Dieser übernimmt diese Daten, konvertiert sie entsprechend und leitet sie an den elektrischen Energienetzsimulator weiter.
Das Ende des Ladevorganges wird wieder vom Anwender initiiert, wodurch sich das Simulationssystem bei der Anbindung von realen Ladesäulen rein passiv verhält und nur den Fortschritt der Ladevorgänge für den Energienetzsimulator aufbereitet.
Der OCPP-Simulator ist nur für die informationstechnische Anbindung der Ladesäule an das System zuständig. Daher muss die Beeinflussung der Versorgungsspannung bzw. der Lastseite der real angeschlossenen Ladesäulen über externe Emulatoren, wie sie als Spannungsquelle oder Elektrofahrzeug-Emulator unter Abschnitt 3.1.2 bereits erläutert wurden, durchgeführt werden.

Anforderungen

Nachdem der Aufbau und die Funktionsweise des OCPP-Simulators erläutert wurden, werden in diesem Abschnitt die Anforderungen an diesen aufgelistet:

A-1 Der OCPP-Simulator agiert als Bindeglied zwischen Simulationsumgebung und realen Ladesäulen. Er muss daher die Informationen, welche über das Open Charge Point Protocol übertragen werden, für die Simulationsumgebung aufbereiten, um die Daten auf einfache Weise verwenden zu können.

A-2 Der Simulator soll, ähnlich wie der Mobilitätssimulator, die Ladesäulen in einer Ladestation-Ladepunkt-Struktur speichern, um eine einheitliche Verwendung über den Simulation Message Bus gewährleisten zu können.

A-3 Die aktuellen Leistungen der realen Ladestationen muss der OCPP-Simulator an die elektrische Energienetzsimulation weiterleiten.

A-4 Obwohl das aktuelle OCPP kein Lastmanagement vorsieht, soll die Kommunikation des Simulators in Richtung Aggregator vorbereitet werden, um die einfache Adaption des Simulators für zukünftige Versionen des Open Charge Point Protocols zu ermöglichen.

3.4.6 Grafische Benutzeroberfläche

Zur Überwachung und Steuerung der angeschlossenen Komponenten wird in der Simulationsumgebung eine grafische Benutzeroberfläche eingesetzt. Besonderes während der Entwicklungs- und Implementierungsphase wird diese Oberfläche benötigt, um den Ablauf der Simulation beobachten und vorhandene Fehlfunktion schnellst möglichst auffinden zu können.

Der Mobilitätssimulator oder der OCPP-Simulator müssen zum Beispiel für Beobachtungen die aktuellen Daten der Simulation zur Verfügung stellen. Dazu gehören neben einer Auflistung der Ladesäulen auch Fahrzeuge, die gerade an diesen geladen werden. Ebenso müssen Informationen der virtuellen Elektrofahrzeuge, wie die Position im Zeitplan oder der aktuelle Batterieladestand auf der grafischen Oberfläche präsentiert werden.

Die Beobachtung dieser Daten wird auch während der Initialisierungsphase eines neuen Lademanagementalgorithmus benötigt, um dessen korrekte Funktionsweise verifizieren zu können.

Anforderungen

Die grundsätzlichen Anforderungen an die grafische Benutzeroberfläche werden wie folgt dargestellt:

A-1 In der Initialisierungsphase des Systems muss über die grafische Oberfläche eine Steuerung der Simulationsumgebung möglich sein, um die Simulation ordnungsgemäß und ohne Datenverlust starten und stoppen zu können.

A-2 Die Parameter und aktuellen Zustände aller in der Simulationsumgebung vorhandenen Komponenten müssen zentral über die Benutzeroberfläche dargestellt werden, damit die Applikation über mehrere Computer verteilt ausgeführt werden kann. Dies erlaubt die Beobachtung und Steuerung der gesamten verteilten Simulationsumgebung von einem einzigen Computer aus.

A-3 Der Betrieb der Simulationsumgebung muss auch ohne grafische Benutzeroberfläche und ohne Benutzereingriffe möglich sein, um einen automatisierbaren Ablauf von mehreren hintereinander ausgeführten Simulationen zu ermöglichen.

4 Implementierung der Umgebung

Nachdem im letzten Kapitel ausführlich der Weg zur finalen Systemarchitektur beschrieben und die Anforderungen an die einzelnen Simulationskomponenten aufgestellt wurden, befasst sich dieses mit der Implementierung der einzelnen Komponenten.
Am Beginn wird der Aufbau und der Ablauf der Kommunikation zwischen den Simulatoren erläutert, welche den Simulation Message Bus zum Nachrichtentransport verwendet. Aufbauend auf die Kommunikation werden anschließend die Komponenten der Simulationsumgebung behandelt. Das Ende dieses Kapitels gibt einen Überblick über die möglichen Anwendungsbereiche, in denen diese Umgebung eingesetzt werden kann.

Die implementierten und hier beschriebenen Programme der Simulationsumgebung werden in der objektorientierten und plattformunabhängigen Programmiersprache Java geschrieben. Der Simulation Message Bus kann dadurch mit voller Funktionalität in das System integriert werden, da eine einheitliche Programmiersprache im System verwendet wird. Ebenso erlauben umfangreiche Bibliotheken die schnelle Implementierung der Webservices, welche für das Open Charge Point Protocol (OCPP) benötigt werden.

4.1 Kommunikation zwischen Komponenten

Die Komponenten der Simulationsumgebung müssen untereinander Nachrichten austauschen. Dies erfolgt über den Simulation Message Bus, dessen Funktionsweise bereits ausführlich in Kapitel 3.4.1 erläutert wurde. Um die Kommunikation in jeden Teilnehmer auf einfache Art und Weise integrieren zu können, wird eine Softwareschnittstelle benötigt, welche die Anbindung an den Nachrichtentransport in einem eigenen Modul kapselt.

Der Simulation Message Bus verwendet ein eigenes Datenformat, mit dem die Nutz- bzw. Simulationsdaten übertragen werden. Dieses basiert auf Zeichenketten (Strings) und besteht aus mehreren Teilen. Die Verwendung der Programmiersprache Java erlaubt den direkten Zugriff auf diese String-Objekte. Die Variablen können zwar für die eigene Anwendung frei definiert und verwendet werden, trotzdem bestimmen bereits die Variablennamen einen gewissen Anwendungszweck:

- **ID** kann verwendet werden, um eine bestimmte Komponente eindeutig zu identifizieren (z. B. Gerätenummer: M01).

- **Var** legt eine bestimmte Variable fest (z. B. Strom in Phase 1).
- **Val** bestimmt den zu übertragenen Wert der Variable (z. B. 5 A).
- **Timestamp** enthält einen Zeitstempel, der mit der Nachricht übertragen wird.
- **Tags** sind eine Ansammlung von zusätzlichen String-Objekten, welche der Nachricht zugeordnet sind und die zur genaueren Spezifikation der Nachricht eingesetzt werden können.

Diese Variablen dienen nicht nur der Informationsübertragung, sondern der Simulation Message Bus verwendet sie auch für das Routing der Nachrichten. Dabei kann jede einzelne dieser Variablen (ausgenommen Timestamp) oder Kombinationen davon verwendet werden, um die Nachrichtenverteilung zu steuern. Dadurch ist es möglich einen flexiblen Datenaustausch zu gewährleisten.

Schon in Kapitel 3.4.1 wurden die verschiedenen Ablaufmodi erläutert. Der SMB unterscheidet diesbezüglich zwischen Simulation und Emulation und setzt dafür eine Zustandsmaschine ein, mit der unter anderem die zeitliche Synchronisation reguliert werden kann. Um diese Funktionalität in eigenen Programmen einsetzen zu können, stellt der SMB die abstrakte Java-Klasse `SimClient` zur Verfügung. Diese enthält die passende Ablaufsteuerung, welche bei der synchronisierten Kommunikation eingehalten werden muss.
Der `SimClient` erfüllt nicht alle Anforderungen, welche in der Simulationsumgebung benötigt werden und muss deshalb entsprechend erweitert und angepasst werden.

Die Kommunikationsmöglichkeit soll, wie bereits eingangs erwähnt, gekapselt werden, um sie einfach in die nachfolgend beschriebenen Simulatoren integrieren zu können. Dafür wird der `SimClient` entsprechend erweitert (Vererbung). Abbildung 4.1 zeigt den schematischen Aufbau des Kommunikationsmoduls, sowie dessen vorhandene Schnittstellen.

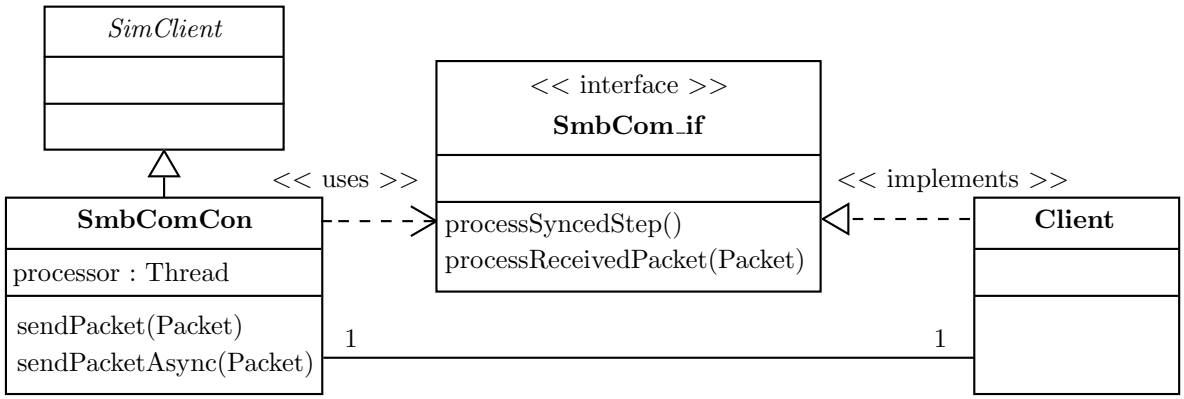

Abbildung 4.1: Aufbau des Kommunikationsmoduls

Die in der Abbildung ersichtliche Klasse `SmbComCon` stellt den zentralen Zugriffspunkt für die Kommunikation eines Clients (z. B. Mobilitätssimulator, Aggregator) mit dem Simulation Message Bus dar. Diese baut eine TCP/IP-Verbindung zu einem konfigurierten Channel des SMB auf und übernimmt das Senden und Empfangen von Nachrichten. Ebenso übernimmt diese Komponente die Synchronisierungsaufgabe, wie sie in Abschnitt 3.4.1 erläutert wurde.

Die `SmbComCon`-Klasse enthält einen eigenständigen, bis zum Programmende laufenden Thread (`processor`), welcher die Synchronisation und den zeitlichen Ablauf der Kommunikation steuert. Dabei wird von der bereits im `SimClient` definierten Zustandsmaschine Gebrauch gemacht,

mit der sich die am SMB angeschlossenen Clients zeitlich synchronisieren lassen. Dieser Thread wird im Emulationsmodus periodisch aktiviert und führt dabei einen vordefinierten Prozessablauf durch, der im Anschluss genauer erläutert wird. Im Simulationsmodus hingegen wird dieser Prozessablauf durch eine spezielle Synchronisationsnachricht ausgelöst (siehe Kapitel 3.4.1).

Der Simulation Message Bus dient innerhalb der Simulationsumgebung als allgemeines Kommunikationsmedium. Über ihn müssen neben den Simulationsdaten, auch Daten zur Konfiguration und Parametrierung der angeschlossenen Simulatoren bzw. Clients übertragen werden. Die verschiedenen Nachrichten lassen sich deshalb in zwei große Gruppen einteilen:

Zur Übertragung der Konfigurationsparameter ist ein *asynchroner Nachrichtenaustausch* notwendig, bei dem die Synchronisation der verschiedenen Simulatoren keine Relevanz spielt. Diese Nachrichten werden nach dem Empfang durch das `SmbComCon`-Modul direkt an den Client zur Verarbeitung weitergeleitet. Ebenso werden ausgehende Nachrichten mit der aktuellen Zeit des Computers versehen und anschließend gesendet. Diese Art des Nachrichtenaustausches erlaubt eine bessere Organisation der Simulationsumgebung, wenn diese über mehrere Computer verteilt ausgeführt wird. Zur Kennzeichnung der asynchronen Nachrichten werden die übertragenen Pakete mit dem Tag `async` versehen, wodurch der SMB und der empfangende Client diese Nachrichten auf einfache Weise filtern können.

Synchrone Nachrichten stellen den zweiten Typ des Nachrichtenaustausches dar, der für die Übertragung der Simulationsdaten, die zwischen den verschiedenen Simulatoren ausgetauscht werden müssen, eingesetzt wird. Zu dieser Gruppe zählen alle Nachrichten, welche nicht mit dem Tag `async` gekennzeichnet sind. Ebenso werden synchrone Nachrichten nicht mit der aktuellen Rechnerzeit versehen, sondern sie verwenden eine am Beginn festgelegte Simulationszeit. Der Datenaustausch erfolgt im Gegensatz zur asynchronen Übertragung zu bestimmten Zeitpunkten, wobei zu sendende Pakete vom `SmbComConf` zwischengespeichert werden. Der synchronisierte Ablauf des `SmbComCon`-Threads steuert anschließend den Versand dieser Nachrichten.

Das `SmbComCon`-Modul stellt Funktionen zur Verfügung, die vom angeschlossenen Client aufgerufen werden können. Dabei erlauben die Funktionen `sendPacket()` und `sendPacketAsync()` den Versand von synchronen bzw. asynchronen Nachrichten. Ebenso greift das Kommunikationsmodul auf das `SmbCom_if`-Interface zurück, welches vom Client implementiert werden muss. Dadurch wird der Klasse `SmbComCon` ermöglicht, die über den Simulation Message Bus empfangenen Nachrichten an den Client weiterzuleiten. Wird eine Nachricht empfangen, ruft der `SmbComCon` die vorhandene Funktion `processReceivedPacket()` des Clients auf.

Der interne Prozessablauf des Kommunikationsmoduls wird in Abbildung 4.2 dargestellt. Dieser kann je nach Betriebsart unterschiedlich aktiviert werden. Im Emulationsmodus wird zwischen den periodischen Abläufen eine konfigurierbare Zeitdauer abgewartet, im Simulationsmodus hingegen triggert der Empfang einer speziellen Synchronisierungsnachricht diesen Prozess.
Am Beginn werden alle empfangenen synchronen Pakete verarbeitet und über die `processReceivedPacket()`-Funktion dem angeschlossenen Client übergeben. Anschließend wird die Funktion `processSyncedStep()` ausgeführt, mit der allgemeine Operationen am Client durchgeführt werden können. Diese Methode ermöglicht, die bereits vorhandene Ablaufsteuerung des Kommunikationsmoduls im Client verwenden zu können. Einfache Clients können dadurch ohne eigenständig ablaufende Threads auskommen und die Programmierung kann wiederum stark vereinfacht werden.
Das Ende des Ablaufes wird durch den Versand aller in der letzten Zeitperiode gespeicherten

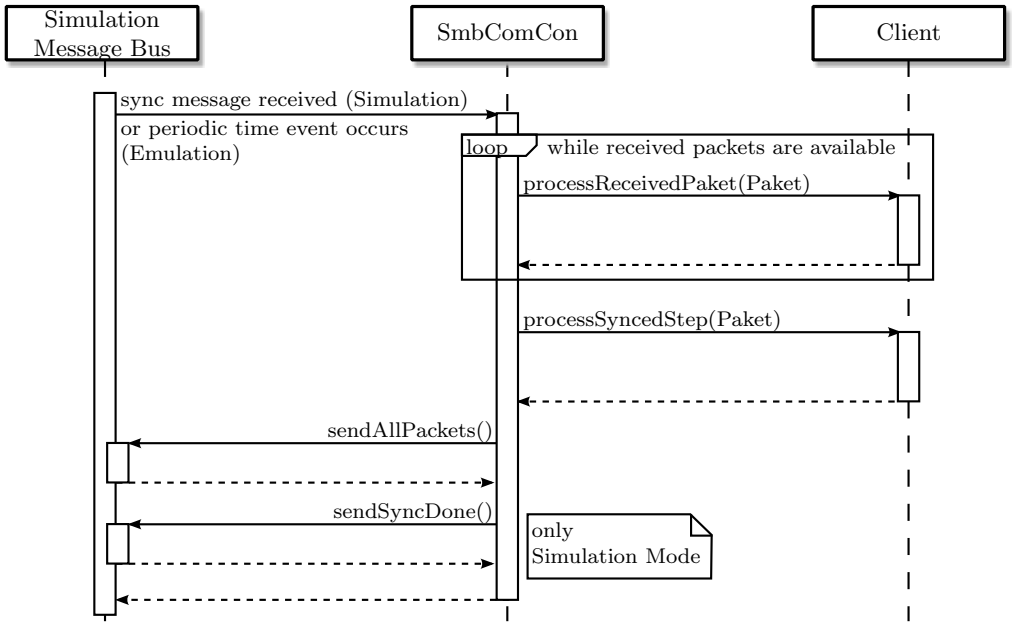

Abbildung 4.2: Interner Ablauf im Kommunikationsmodul

Nachrichten gekennzeichnet. Ebenso wird am Ende des Simulationsschrittes eine Synchronisierungsnachricht gesendet, welche der Simulationsumgebung mitteilt, dass dieser Client seinen periodischen Prozess beendet hat.

Eingangs wurde bereits auf das Datenformat des SMB eingegangen, welches auf mehreren String-Variablen basiert. In der Simulationsumgebung ist es notwendig, oft komplexe Datentypen mit mehreren Variablen, z. B. für die Anfrage eines Ladewunsches an den Aggregator, zu übertragen. Diese komplexen Typen müssen in eine Zeichenkettendarstellung konvertiert werden. Dafür wird das JSON-Protokoll (JavaScript Object Notation) eingesetzt, welches die Serialisierung von strukturierten Daten in ein Textformat erlaubt [30, S.3]. Daten werden dabei mithilfe von Klammern strukturiert, wodurch eine effiziente Darstellung mit geringem Overhead ermöglicht wird. Die hier durchgeführte Implementierung benutzt dafür die quelloffene Java-Bibliothek *JSON.simple* [31], um die zu übertragenden Daten in Zeichenketten konvertieren zu können.

Die Implementierung der Kommunikation in einem eigenen Modul beschränkt die Verwendung des Simulation Message Bus auf wenige Funktionen und die Programmierung der einzelnen Simulatoren bzw. Clients lässt sich dadurch vereinfachen.

4.2 Mobilitätssimulator

Der Mobilitätssimulator bildet eine virtuelle Ladeinfrastruktur nach, welche zur Analyse von Lademanagementstrategien eingesetzt werden kann. Die Aufgabe und die Anforderungen wurden bereits im Abschnitt 3.4.3 ausführlich behandelt. Ebenso wurde dort bereits eine allgemeine Architektur dieses Simulators, sowie die dafür benötigten Bestandteile aufgezeigt.

Der Aufbau des implementierten Mobilitätssimulators wird in Abbildung 4.3 gezeigt. Im Klassendiagramm werden dabei nur die Zusammenhänge zwischen den funktionalen Hauptklassen beschrieben, wodurch die Übersichtlichkeit in der Darstellung gesteigert wird.

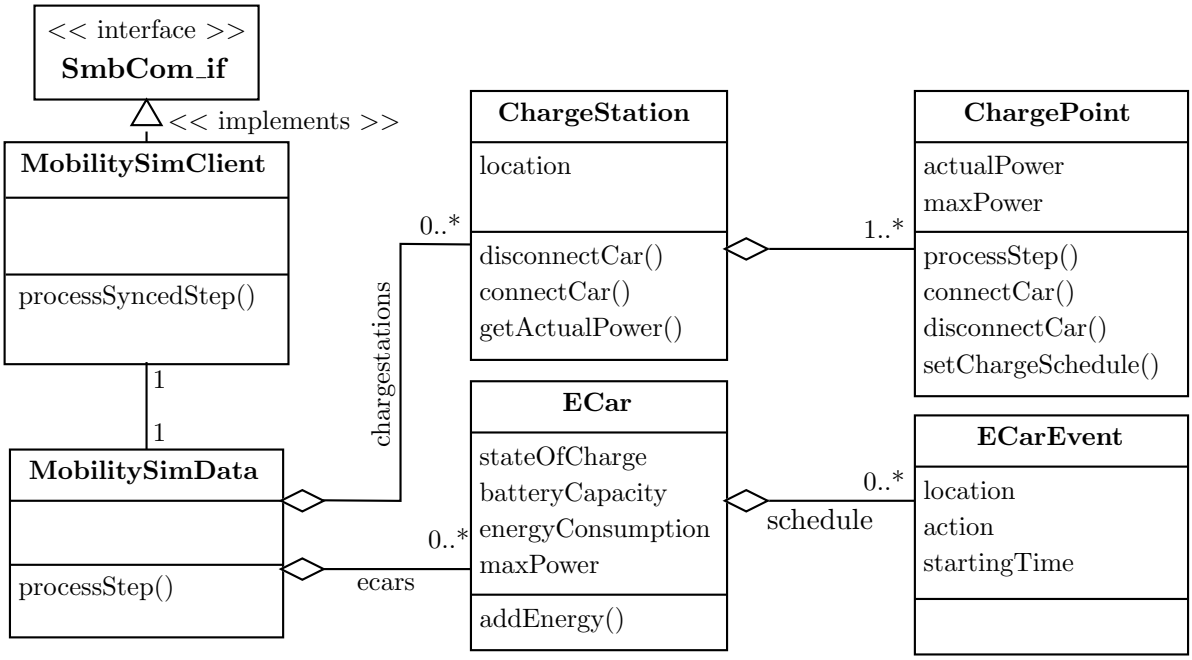

Abbildung 4.3: vereinfachtes Klassendiagramm der Hauptkomponenten

Die in Kapitel 3.4.3 bereits angeführte virtuelle Ladeinfrastruktur und die virtuellen Elektrofahrzeuge werden im Simulator in Form eines Datenmodells (`MobilitySimData`) implementiert. Dieser Entwurf erleichtert die im Anschluss beschriebene Möglichkeit, die Fahrprofile in Dateien zu speichern.
Die Klasse `MobilitySimClient` übernimmt die Verbindung zum Kommunikationskanal und dient der Verwaltung des Simulators. Dabei kann diese Klasse auf die Schnittstellenfunktionen des Kommunikationsmoduls `SmbComCon` zugreifen, um Nachrichten des Mobilitätssimulators über den Simulation Message Bus senden zu können (siehe Abschnitt 4.1). Ebenso implementiert diese Klasse die notwendigen Funktionen für den Empfang von Nachrichten (`SmbCom_if`). Diese werden, wie bereits im letzten Abschnitt erläutert, vom Kommunikationsmodul aufgerufen, wenn neue Daten für diesen Client empfangen wurden.

Das Datenmodell (`MobilitySimData`) beinhaltet neben den Parametern der Ladestationen auch jene der Elektrofahrzeuge, welche in der Simulation benötigt werden. Ein Elektrofahrzeug (`ECar`) kann diesbezüglich durch verschiedene Parameter, wie beispielsweise Batteriekapazität, Ladestand und durchschnittlicher kilometerbezogener Energieverbrauch, beschrieben werden. Außerdem wird für jedes Fahrzeug ein Fahrprofil (`schedule`) hinterlegt. Dieses Profil, welches in Tabelle 3.1 auf Seite 33 bereits dargestellt wurde, wird in Form einer Liste von Ereignissen (`ECarEvent`) gespeichert. Diese Klasse enthält neben dem Aufenthaltsort des Fahrzeuges die Aktion, wie Fahren oder Laden, und den Startzeitpunkt.

Die in Abschnitt 3.4.3 bereits entwickelte Ladeinfrastruktur, wird im Mobilitätssimulator direkt mit den Klassen `ChargeStation` und `ChargePoint` implementiert. Die Ladestationen (`ChargeStation`) dienen der Organisation dieser Struktur und bestehen aus mehreren Ladepunkten. Sie sind in der Lage, die Leistungen der angeschlossenen Ladepunkte zusammenzufassen und diese als summierten Leistungswert an den Netzsimulator (PowerFactory) zu senden. Ebenso sind sie in der Lage, von diesem die zugeordneten Spannungswerte auszulesen (vgl. Anforderung A-2).

An einer Ladestation sind Ladepunkte (`ChargePoint`) vorhanden, welche für die Simulation und die Durchführung des Ladevorganges eines Elektrofahrzeuges erforderlich sind. Wird ein Elektrofahrzeug (`ECar`) an eine Ladestation angeschlossen, so kommt es zu einer Verbindung mit einem freien Ladepunkt. Dieser Ladepunkt übernimmt die zeitliche Steuerung des Ladevorganges, welcher mit Hilfe eines vom Aggregator erstellten Ladeplanes erfolgt.

Die im Simulator vorhandenen Eigenschaften der Ladesäulen und -punkte, als auch die Fahrprofile der Elektrofahrzeuge werden in Dateien gespeichert (vgl. Anforderung A-7) und beim Softwarestart ausgelesen. Diese Konfigurations- oder Profildateien beinhalten alle relevanten Informationen zur Simulation der Ladevorgänge. Eine verwendete Konfiguration ist auszugsweise im Anhang A.1 angeführt. Beim Start des Programmes werden aus diesen Daten automatisch direkt verwendbare Objekte des Datenmodells (`MobilitySimData`) erzeugt. Dieser Prozess erfolgt durch den Einsatz der Java-Bibliothek *JAXB* (Java Architecture for XML Binding) [32], bei der Java-Klassen automatisch aus vorhandenen Dateien erstellt werden. Die Konfiguration erfolgt dabei in Form von XML-Dateien (Extensible Markup Language).

In Kapitel 3.4.3 wurde die Anforderung definiert, dass Daten aus anderen Simulationsprogrammen, wie EVSim, ausgelesen und verwendet werden können (vgl. Anforderung A-1). Da sich die Konfigurationsdateien verschiedener Simulatoren unterscheiden, müssen diese Informationen für den Einsatz im Mobilitätssimulator konvertiert werden. Dabei muss besonders die Zuordnung der Ladestationen zu den variablen Lasten im Energienetzsimulator erfolgen, was meist nur manuell durchgeführt werden kann. Daher ist die Konvertierungsmöglichkeit nicht direkt im Mobilitätssimulator implementiert, sondern wird von der Software der grafischen Benutzeroberfläche (siehe Abschnitt 4.5) übernommen.

Der Simulationsablauf im Client wird über das Kommunikationsmodul initiiert. Dieses ruft peri-

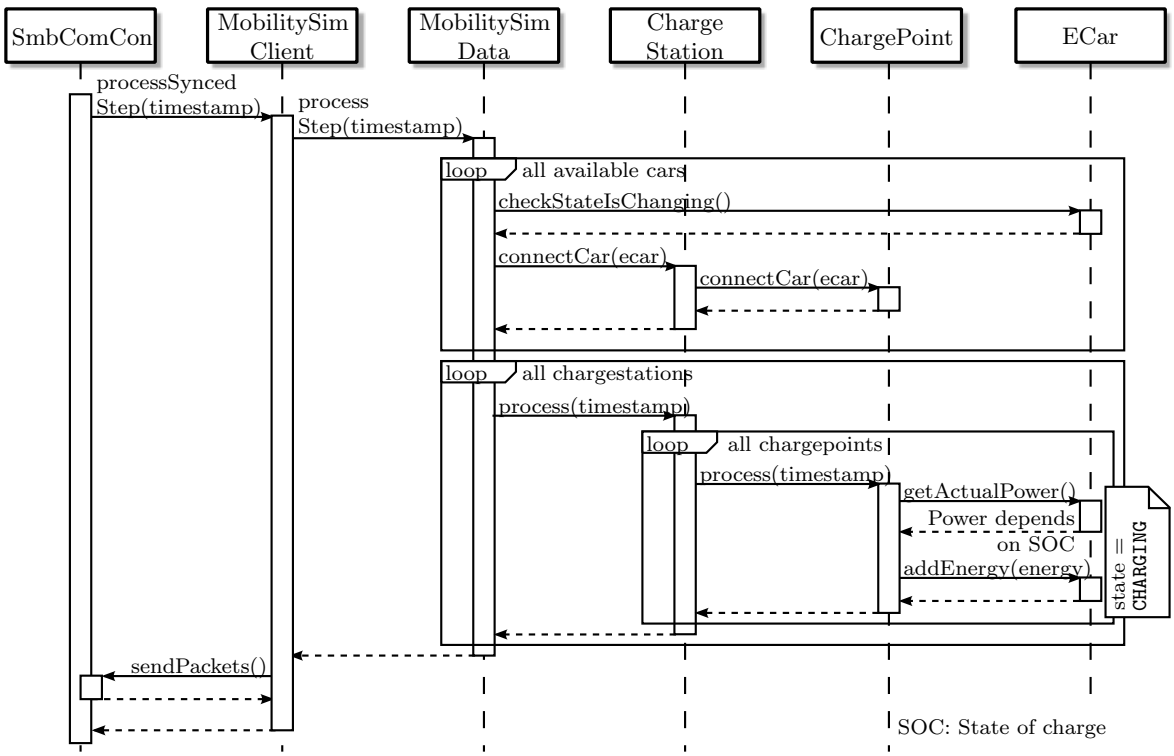

Abbildung 4.4: schematischer Ablauf eines Simulationsschrittes

odisch die Funktion `processSyncedStep()` der Klasse `MobilitySimClient` auf und startet somit die Durchführung eines Simulationsschrittes. Abbildung 4.4 zeigt den typischen Ablauf.

Die Prozessfunktion wird mit einem bestimmten vom Kommunikationsmodul festgelegten Zeitpunkt (`timestamp`) geladen. Zu Beginn werden die Zustände aller Elektrofahrzeuge mit diesem neuen Zeitpunkt aktualisiert. Dabei werden die Events im Fahrprofil auf Änderungen untersucht (`checkStateIsChanging()`). Falls dabei ein Fahrzeug seinen Zustand ändert und einen Ladevorgang starten will, wird dieses Fahrzeug mit einem freien Ladepunkt der entsprechenden Ladestation verbunden (`connectCar()`). Ebenso kann ein Elektrofahrzeug von seiner Ladestation entfernt werden, wenn dieses bei der Zustandsänderung den Ladevorgang beendet.

Im nächsten Schritt werden die Ladestationen aktualisiert. Dabei wird die `process()`-Funktion aller Ladepunkte aufgerufen, um die Simulation des Ladevorganges durchzuführen. Jeder Ladepunkt hat eine eigene Zustandsmaschine, deren Zustände den Ladevorgang des Elektrofahrzeuges an einer Ladesäule beschreiben:

- `IDLE`: Dieser Zustand kennzeichnet einen freien Ladepunkt, an dem kein Elektrofahrzeug angeschlossen ist.
- `WAITING_FOR_CHARGING`: Wird ein Elektrofahrzeug an den Ladepunkt angeschlossen, geht der Ladepunkt in diesen Zustand über. Dabei sendet er an den Aggregator eine Anfrage für einen neuen Ladeplan, um das Elektrofahrzeug laden zu können. Anschließend bleibt der Ladepunkt solange in diesem Zustand, bis der neue Ladeplan empfangen wurde (vgl. Anforderung A-3).
- `CHARGING`: Dieser Zustand führt die Simulation des tatsächlichen Ladevorganges durch, wobei das virtuelle Elektrofahrzeug mit Energie geladen wird (`addEnergy()`). Im Zuge dessen wird die Ladeleistung von verschiedenen, nachfolgend angeführten Parametern beeinflusst.
- `FINISHED_CHARGING`: Der letzte Zustand kennzeichnet ein vollständig geladenes Fahrzeug, welches weiterhin am Ladepunkt angeschlossen bleibt. Beim Wechsel in diesen Zustand wird dem Aggregator mitgeteilt, dass das Elektrofahrzeug vollständig geladen ist und für das Verlassen der Ladestation bereit ist.

Der Mobilitätssimulator ist bereits in der Lage ein Lademanagement durchzuführen. Vom Aggregator werden Ladepläne empfangen, welche die Ladung des angeschlossenen Fahrzeuges vorgeben. Ein Ladeplan besteht aus einer sortierten Liste von Ladeevents. Dabei setzt sich ein Ladeevent aus einem Zeitpunkt und einem Leistungswert, welcher die Höhe der maximalen Ladeleistung der

Event	Zeitpunkt	Leistung
1	16.1.2014 13:10:00	8 kW
2	16.1.2014 13:15:00	3 kW
3	16.1.2014 13:42:00	5 kW
4	16.1.2014 13:49:00	4 kW

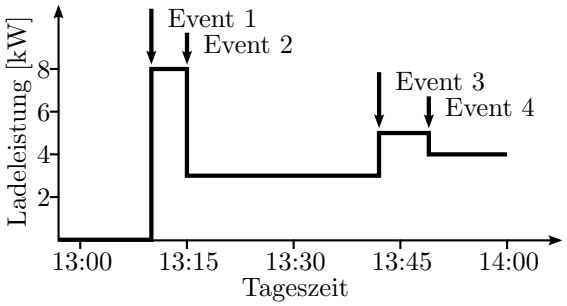

Abbildung 4.5: Ladeplan und -verlauf

Ladestation vorgibt, zusammen. Dieser Leistungswert wird zum gegebenen Zeitpunkt gesetzt und bleibt solange aktiv, bis er vom nächsten in der Liste befindlichen Event überschrieben wird. Abbildung 4.5 zeigt einen typischen Ladeplan und den dazugehörigen Ladeverlauf, welcher sich auch reaktiv während des Zustandes `CHARGING` ändern kann und vom Mobilitätssimulator übernommen wird (vgl. Anforderung A-4).

Im Zustand `CHARGING` wird der simulierte Ladestrom für das Elektrofahrzeug nicht ausschließlich vom soeben erläuterten Ladeplan bestimmt. Die durch den Plan vorgegebene Leistung muss innerhalb von maximalen, in der Konfigurationsdatei festgelegten, Grenzwerten liegen, die den möglichen Leistungsbereich des Ladepunktes oder des Elektrofahrzeuges beschränken (vgl. Anforderung A-5). Außerdem hängt der Ladestrom, wie dies in Kapitel 2.1.3 ausführlich behandelt wurde, vom Ladestand der im Fahrzeug befindlichen Batterie ab. Dieser Strom kann gegebenenfalls niedriger sein als die vorgegebene Leistung des Ladeplans und wird daher im Mobilitätssimulator mithilfe einer zusätzlichen Funktion (`getActualPower()`) berücksichtigt (vgl. Anforderung A-6).

4.3 Aggregator

Der Aggregator hat in dieser Simulationsumgebung die Aufgabe, die Ladevorgänge der angeschlossenen Fahrzeuge zu koordinieren. Die Anforderungen an die hier beschriebene Implementierung des Aggregators wurden bereits in Abschnitt 3.4.4 aufgestellt und die interne Struktur dieser Software wird in der Abbildung 4.6 gezeigt.

Die Klasse `AggregatorClient` hat dieselbe Aufgabe wie die, im letzten Abschnitt beschriebene Klasse `MobilitySimClient`. Sie stellt den Zugriffspunkt für das Kommunikationsmodul durch Implementierung des Interfaces `SmbCom_if` dar und verwaltet die Scheduler, welche die Koordination der Ladevorgänge durchführen. Obwohl mehrere Scheduler in der Software vorhanden sein können (vgl. Anforderung A-4), lässt der Aggregator nur die Ausführung eines einzigen Algorithmus zu, dessen Parameter aus einer Konfigurationsdatei ausgelesen werden (vgl. Anforderung A-5).

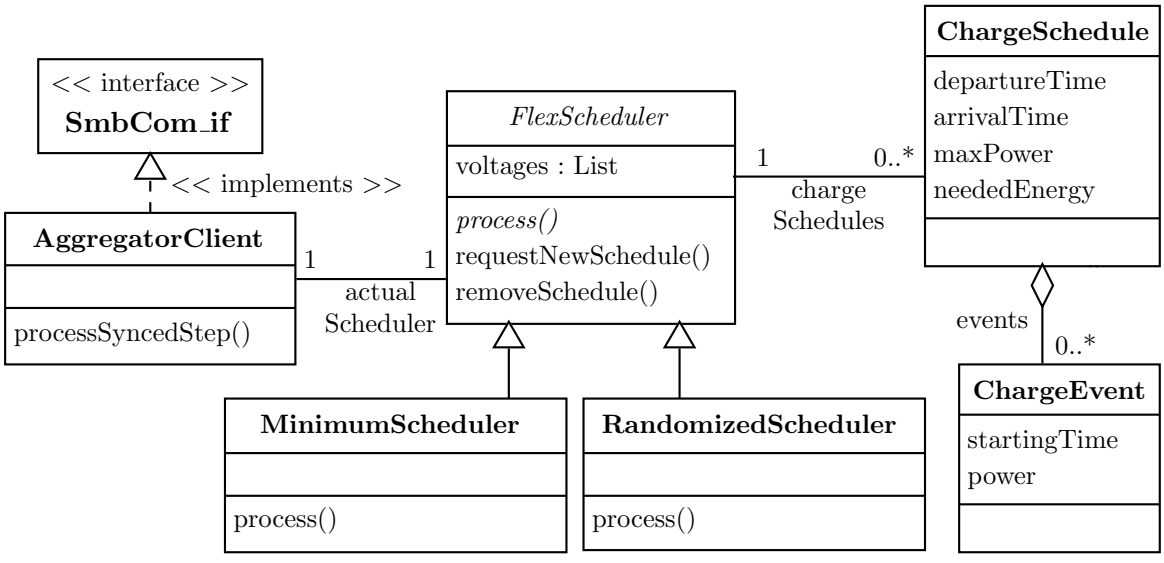

Abbildung 4.6: vereinfachtes Klassendiagramm des Aggregators

Die grundlegende Funktionalität der Scheduler wird in der abstrakten Basisklasse `FlexScheduler` zusammengefasst. Sie übernimmt empfangene Ladewünsche des Mobilitätssimulators und speichert diese zwischen (`chargeSchedules`). Ebenso übernimmt diese Klasse die Aufgabe, Spannungen vom Energienetzsimulator abzufragen und sie für reaktive Algorithmen zur Verfügung zu stellen. Um einen Algorithmus implementieren zu können, muss diese `FlexScheduler`-Klasse spezialisiert werden und die Methode `process()` überschrieben werden. Diese wird bei jedem periodischen Ablauf aufgerufen und dient als Zugriffspunkt für die Implementierung neuer Algorithmen. Die Klassen `RandomizedScheduler` bzw. `MinimumScheduler`, welche in Abbildung 4.6 dargestellt sind, implementieren die spezifischen Lademanagementalgorithmen zum Beispiel in der zu überschreibenden `process()`-Funktion. Dieses Design schafft eine einfache und rasche Implementierungsmöglichkeit neuer Lademanagementstrategien.

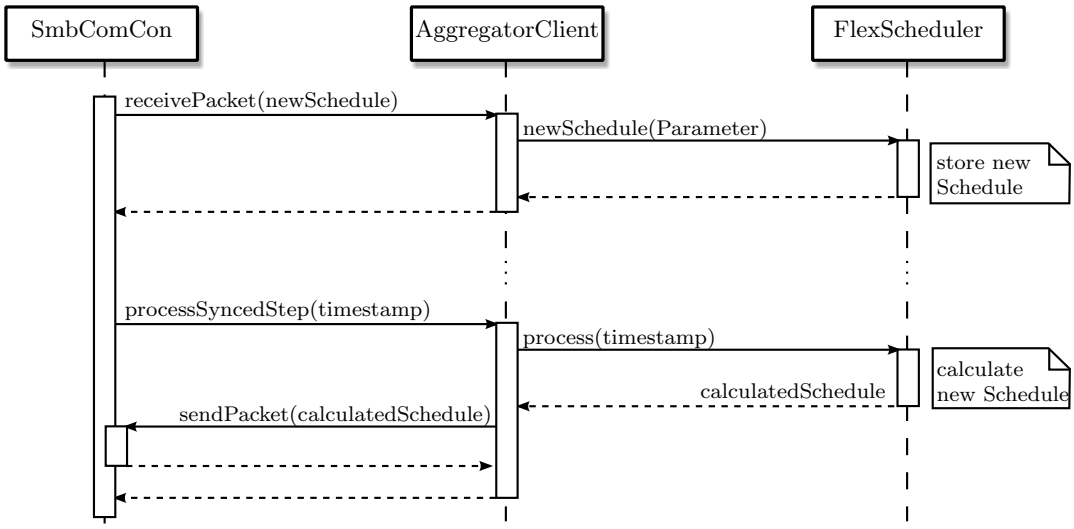

Abbildung 4.7: Ablauf beim Erstellen eines neuen Ladeplans

Ein Elektrofahrzeug im Mobilitätssimulator sendet zu Beginn des Ladevorganges einen Ladewunsch an den Aggregator. Diese Nachricht, welche unter anderem den geplanten Abfahrtszeitpunkt, sowie die benötigte Energiemenge enthält, wird vom Aggregator übernommen und in der aktiven `FlexScheduler`-Klasse zwischengespeichert. Beim nächsten zyklischen Aufruf der `process()`-Funktion wird ein Ladeplan erstellt und an den anfragenden Mobilitätssimulator retourniert. Abbildung 4.7 verdeutlicht diesen Ablauf (vgl. Anforderung A-1). Während des gesamten Ladevorganges werden die spezifischen Informationen des Elektrofahrzeuges gespeichert (vgl. Anforderung A-2).

Der Aggregator kann auch für den Einsatz von reaktiven Ladestrategien verwendet werden, deren Ablauf in Abbildung 4.8 dargestellt ist. Dabei müssen zyklische Spannungswerte von kritischen Punkten (z. B. Endpunkte von Leitungssträngen) vom Energienetzsimulator ausgelesen werden. Diese werden anschließend auf Einhaltung der vorgegebenen Betriebsgrenzen (siehe Kap. 2.1.2) überprüft. Dabei kann es passieren, dass Ladepläne von Fahrzeugen während des Ladevorganges angepasst werden müssen. Der dazu eingesetzte Scheduler im Aggregator berechnet dann die Ladepläne der im System gespeicherten Fahrzeuge neu (vgl. Anforderung A-3).

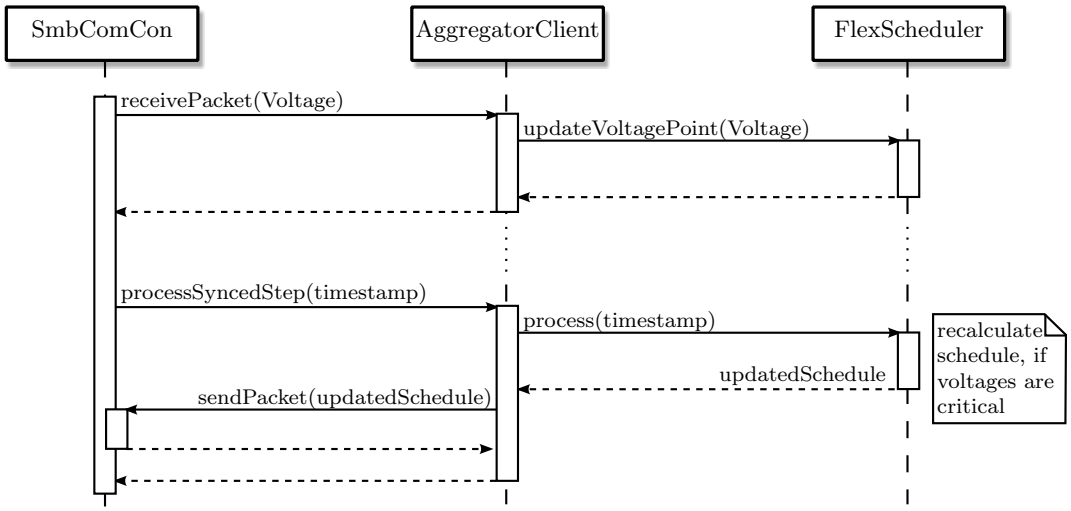

Abbildung 4.8: Reaktive Ladeplanerstellung

4.4 OCPP-Simulator

Der OCPP-Simulator ermöglicht den Anschluss und die Ansteuerung von real aufgebauten, über das Open Charge Point Protocol kommunizierende Ladesäulen an die Simulationsumgebung (vgl. Anforderung A-1). Die allgemeine Beschreibung der Aufgaben und der Anforderungen an diesen Simulator wurden bereits in Kapitel 3.4.5 behandelt.

Zur Bereitstellung der Webservices, welche vom Open Charge Point Protocol benötigt werden, wird in diesem Simulator die Java-Bibliothek *Apache CXF* [33] eingesetzt. Diese Bibliothek ist in der Lage, mithilfe des mitgelieferten Programms *wsdl2java*, aus den WSDL-Dateien, in denen OCPP spezifiziert ist, automatisch die entsprechenden Java-Klassen zu erzeugen. Diese Klassen werden in den OCPP-Simulator eingebunden, wobei die Bibliothek Apache CXF eine Programmierschnittstelle (API: Application Programming Interface) zur Verfügung stellt, mit der ein Webserver auf einfache Weise generiert werden kann.

Der OCPP-Simulator besitzt einen ähnlichen Aufbau, wie der bereits unter Abschnitt 4.2 beschriebene Mobilitätssimulator, um die Verwaltung der realen Ladesäulen vornehmen zu können (vgl. Anforderung A-2). Zusätzlich muss dieser Simulator um notwendige Komponenten für die Kommunikation mit den Ladesäulen erweitert werden. Abbildung 4.9 zeigt die vorhandenen Klassen, welche in diesem Simulator implementiert sind.

Die Klasse `Ocpp15Client` stellt, wie dies bereits in der Klasse `MobilitySimClient` (siehe Abschnitt 4.2) erläutert wurde, die Verbindung zum Simulation Message Bus her. Ebenso übernimmt diese Klasse die Erzeugung des Servers (`Endpoint`) für das OCPP-Webservice *Centralsystem*. Dieses greift während des Betriebes auf die implementierten Funktionen des Java-Inferfaces `CentralSystemService` zurück. Die Klasse `OcppCentralSystem` verwaltet die vorhandenen Ladestationen und stellt die notwendigen Funktionen für das Webservice-Interface bereit.

Die zu verwaltenden Ladesäulen werden in Konfigurationsdateien gespeichert und mit der Java-Bibliothek *JAXB* [32] automatisch beim Starten des Programmes instanziert (siehe Kapitel 4.2).

Wie bereits in Kapitel 3.4.5 erläutert, muss der OCPP-Simulator neben dem Server für das CentralSystem auch einen Client für das *Chargepoint*-Webservice für jede vorhandene Ladesäule

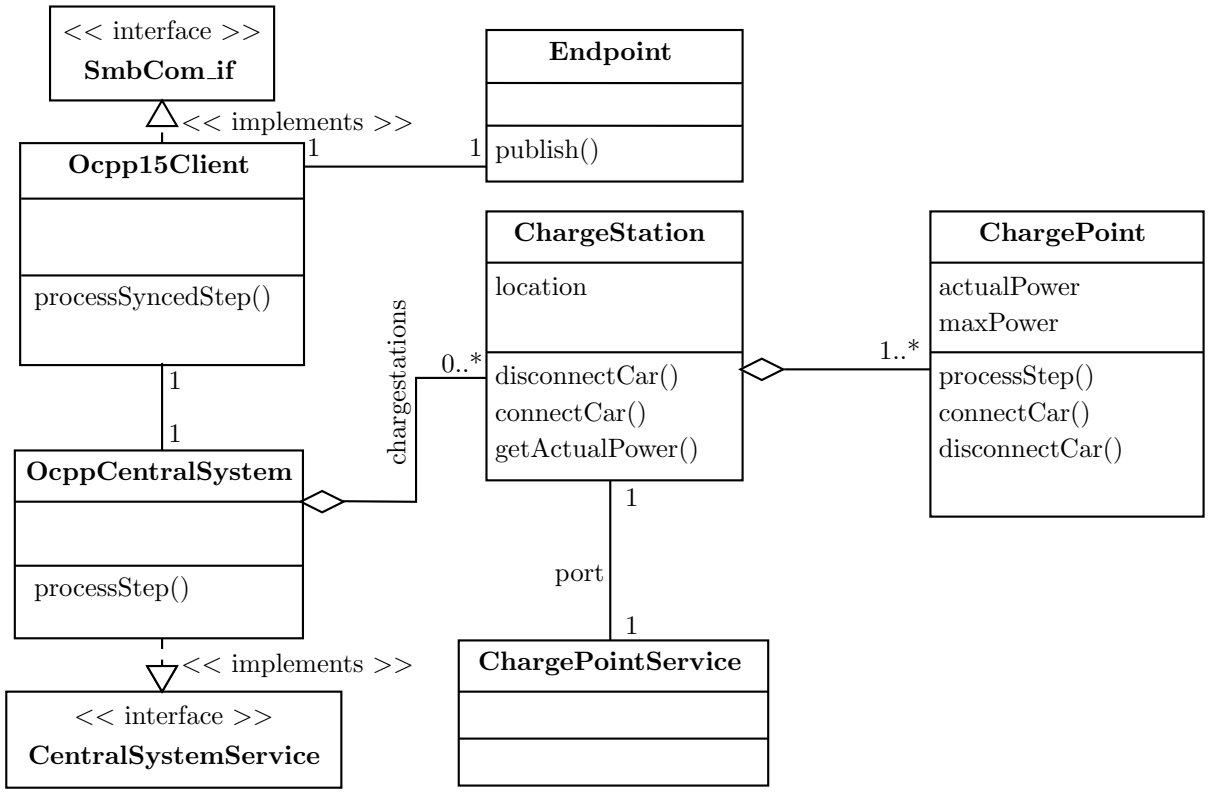

Abbildung 4.9: vereinfachtes Klassendiagramm des OCPP-Simulators

instanzieren. Nur so kann die Kommunikation mit der realen Ladesäule aufgebaut werden. Der benötigte Client (Klasse `ChargePointService`) wird dabei von jeder Ladesäulen-Klasse (`ChargeStation`) erzeugt und ermöglicht dadurch die Datenübertragung in Richtung real angeschlossener Ladesäulen.

Die Klasse `ChargePointService` wird automatisch vom beschriebenen WSDL-Generator erstellt und kann daher im Simulator auf einfache Weise verwendet werden. Dieses implementierte Webservice wird beispielsweise zur Abfrage von Diagnosedaten der real aufgebauten Ladesäule oder zum Verändern von Konfigurationseinstellungen benötigt. Im OCPP-Simulator wird von der Übertragung von Konfigurationen Gebrauch gemacht, um die Intervalle für die periodische Übermittlung der Messdaten während des aktiven Ladevorganges an der Ladesäule festzulegen. Der Ablauf des Sendens von Konfigurationsänderungen ist in Abbildung 4.10 dargestellt.

Das Open Charge Point Protocol in der hier verwendeten Version 1.5 dient hauptsächlich der

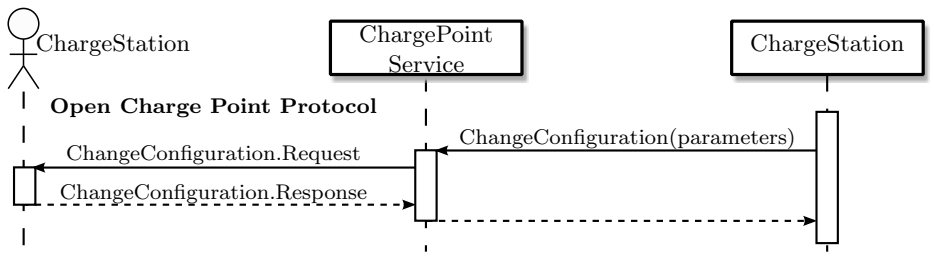

Abbildung 4.10: Änderung der Konfiguration an einer Ladesäule

Verwaltung von Ladesäulen, als auch der Abrechnung von Ladevorgängen. Für die Simulation sind viele Funktionalitäten des OCPP daher nicht erforderlich und somit im OCPP-Simulator auch nicht implementiert. Ebenso wurde die Überprüfung der Authentifizierung am Anfang eines Ladevorganges unterbunden. Dadurch kann jeder Benutzer, der sich an der Ladesäule anmeldet, den Ladevorgang des Elektrofahrzeuges starten (siehe Abbildung 4.11). Obwohl die verwendete OCPP-Version kein Lademanagement ermöglicht, sind im OCPP-Simulator bereits alle Funktionalitäten zum Empfangen von Ladeplänen implementiert (vgl. Anforderung A-4).

Die Ladesäule sendet zu Beginn eines Ladevorganges eine Authentifizierungsnachricht an den OCPP-Simulator. Diese wird vom gestarteten *Centralsystem*-Server (`Endpoint`) entgegen genommen und an die implementierten Funktionen des Interfaces `CentralSystemService` übergeben. Hier wird jede Nachricht akzeptiert und die entsprechende Antwort an die anfragende Ladesäule retourniert. Anschließend kann vom Benutzer der Ladesäule der Ladevorgang gestartet werden. Dieser wird vom OCPP-Simulator ebenfalls akzeptiert und der aktuelle Zustand der Ladesäule wird mithilfe der internen Klassen `ChargeStation` und `ChargePoint` im Simulator gespeichert. Auch Messdaten, welche während des Ladevorganges übertragen werden, können in diesen Klassen verwaltet werden. Dadurch ist der OCPP-Simulator in der Lage, die bezogenen Leistungen der realen Ladesäulen, wie der Mobilitätssimulator, an den elektrischen Energienetzsimulator weiterzuleiten (vgl. Anforderung A-3).

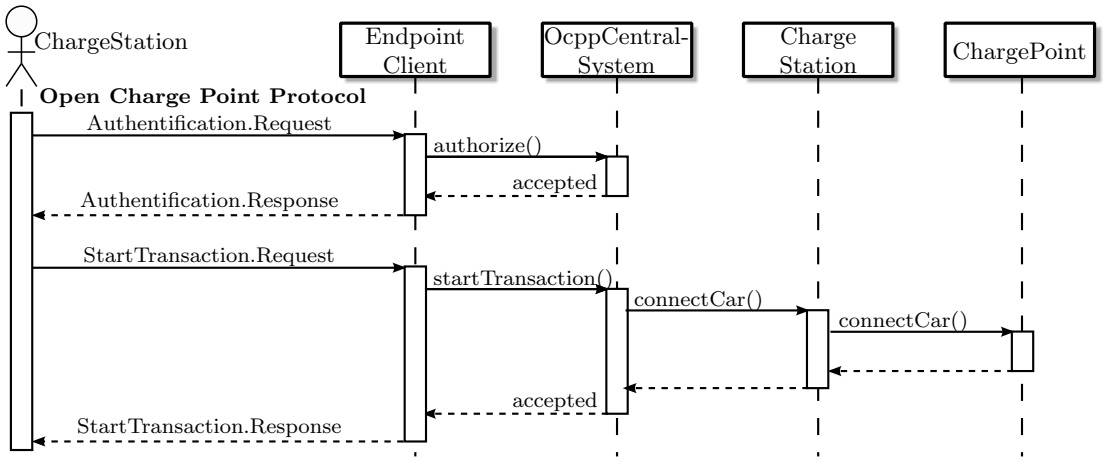

Abbildung 4.11: Ablauf beim Starten eines Ladevorganges

4.5 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche ist für die Überwachung und die Steuerung der gesamten Simulationsumgebung zuständig, wie dies bereits unter Kapitel 3.4.6 erläutert wurde. Dabei ist sie in der Lage, aktuelle Zustandsdaten aller angeschlossenen Simulationskomponenten während des laufenden Betriebes in einfacher Form darzustellen.

Abbildung 4.12 zeigt ein Bildschirmfoto der Software, bei der die Anzeige zweigeteilt ist. Beim Starten der Software wird mithilfe einer globalen Nachricht über den Simulation Message Bus das Vorhandensein der Komponenten abgefragt. Die angeschlossenen Komponenten antworten auf diese Anfrage und die Oberfläche stellt anschließend die in der Umgebung vorhandenen Simulatoren auf der linken Seite in einer Baumstruktur dar. Dabei werden zusätzlich die Ladestationen in

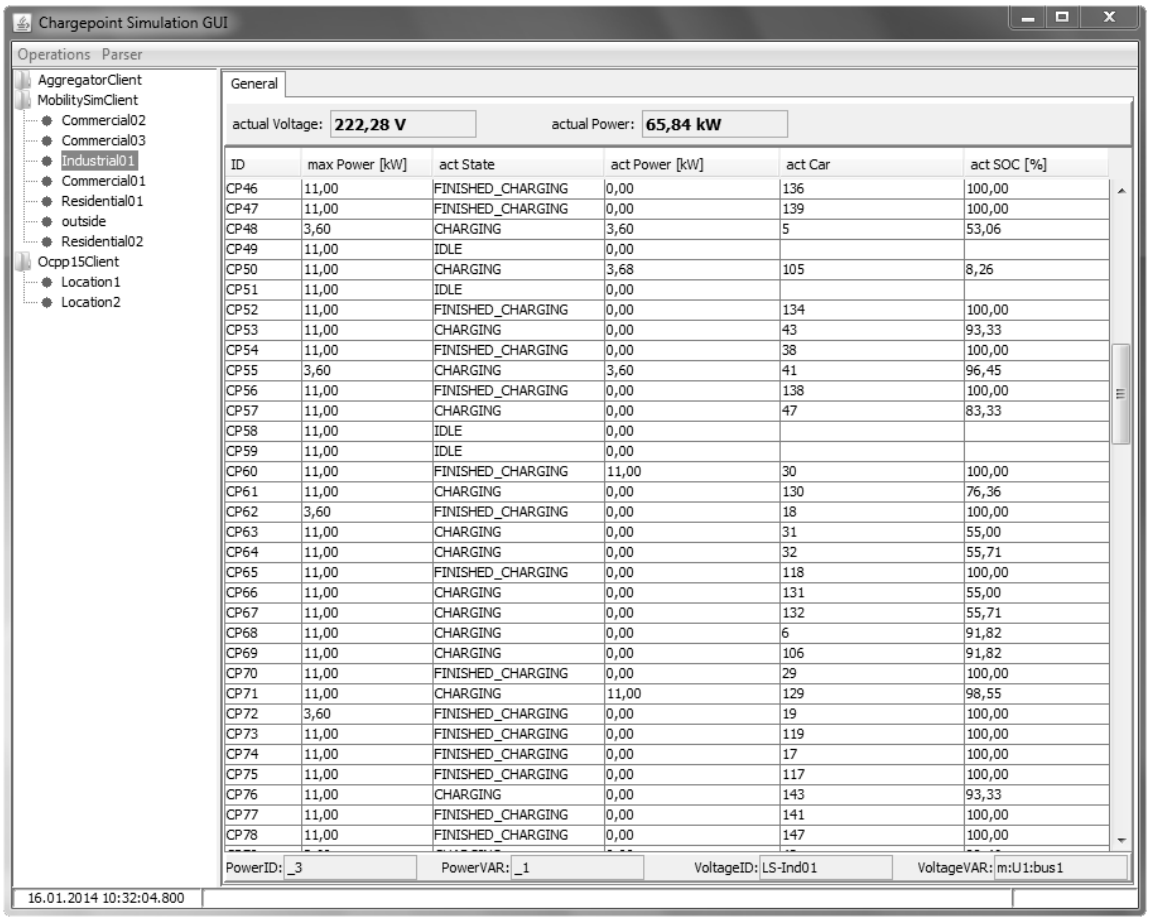

Abbildung 4.12: Bildschirmfoto der grafischen Oberfläche

den Simulatoren (Mobilitäts- und OCPP-Simulator) in diese Darstellungsform integriert. Durch Aktivierung eines Baumelementes wird auf der rechten Seite der Oberfläche ein Fenster mit den objektspezifischen Informationen angezeigt. Exemplarisch ist in der Abbildung 4.12 der aktuelle Zustand während der Simulation einer Ladestation des Mobilitätssimulators dargestellt. Im Fenster werden alle Ladepunkte der Station, mit den aktuellen Zuständen, den angeschlossenen Fahrzeugen und den entsprechenden Ladeleistungen, aufgelistet.

Die Simulationsumgebung wird über verschiedene Menüeinträge an der oberen Seite der Oberfläche gestartet bzw. gestoppt. Dies ist besonders in der Initialisierungsphase notwendig, um das System ordnungsgemäß ohne Datenverlust beenden zu können (vgl. Anforderung A-1). Der Informationsaustausch der grafischen Oberfläche mit den restlichen Simulatoren erfolgt, wie bei allen Komponenten, über den Simulation Message Bus, wodurch das Simulationssystem über mehrere Computer verteilt werden kann (vgl. Anforderung A-2). Die Software der grafischen Oberfläche sendet zyklisch Anfragen an jeden angeschlossenen Simulator. Dieser antwortet daraufhin mit den aktuellen Zustandsdaten. Diese Daten sind besonders durch ihre umfangreiche Nachrichtengröße gekennzeichnet, wodurch der Simulation Message Bus beim Einsatz der grafischen Oberfläche erheblich belastet wird. Das Anfrage-Antwort-Verfahren, das bei der Kommunikation zwischen Oberfläche und Umgebung eingesetzt wird, hat dabei den Vorteil, dass beim alleinigen Betrieb der Simulationsumgebung – ohne Oberfläche – die Systemlast des SMB bzw. die Nachrichtenübertragung reduziert werden kann (vgl. Anforderung A-3).

Parser für EVSim-Konfiguration

In Kapitel 4.2 wurde die Problematik, die beim Konvertieren von Konfigurationsdateien aus anderen Simulationsprogrammen auftritt, erläutert. Die Simulationsumgebung kann Fahrprofile aus der Software EVSim für den Einsatz im Mobilitätssimulator umwandeln. Dafür ist ein eigenes Werkzeug in der grafischen Oberfläche vorhanden, welches über den Menüeintrag *Parser* ausgewählt werden kann. Dieser Übersetzer, welcher in Abbildung 4.13 dargestellt ist, liest die Konfigurationsdateien aus EVSim ein. Jeder Eintrag kann anschließend über die Oberfläche bearbeitet und geändert werden. Dabei müssen die Ladestationen, welche in der linken, oberen Tabelle aufgelistet sind, manuell den variablen Lasten in der elektrischen Energienetzsimulation (DIgSILENT PowerFactory) zugewiesen werden. Am Ende können die adaptierten Einstellungen in einer neuen Konfigurationsdatei gespeichert werden, welche direkt im Mobilitätssimulator eingelesen und verwendet werden kann.

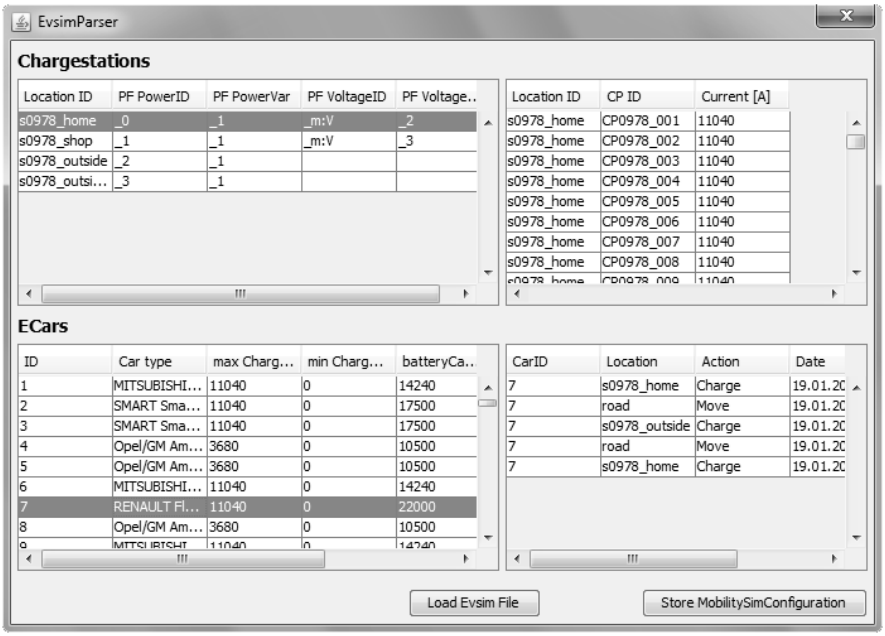

Abbildung 4.13: Parser für EVSim-Konfiguration

4.6 Datenauswertung und -speicherung

Neben der in Abschnitt 4.5 beschriebenen grafischen Oberfläche, welche der Darstellung der aktuellen Daten und der Überwachung während der Simulation dient, gibt es weitere Möglichkeiten, wie der Zustand der Simulationsumgebung angezeigt und gespeichert werden kann.

Jedes implementierte Programm stellt seinen internen Programmstatus in einer Konsole dar. Dafür wird eine Kombination der Java-Bibliotheken *Slf4J* [34] und *Log4J* [35] eingesetzt. Diese Bibliotheken können die Meldungen in verschiedene Hierarchieebenen einteilen und anschließend ausgeben. Zusätzlich können diese Daten in Textdateien gespeichert werden, um für spätere Ablaufanalysen zur Verfügung zu stehen.

Der Simulation Message Bus hat ebenfalls die Möglichkeit, übertragene Nachrichten in Log-Dateien zu speichern. Dafür können an verschiedenen Stellen in den Channels Datei-Logger in

Form von Proxies eingefügt werden, welche die Nachrichten genau an der eingefügten Stelle entnehmen (siehe Abbildung 3.9 auf Seite 30).

Die bis hierher beschriebenen Möglichkeiten zur Informationsspeicherung beziehen sich auf die Zustände der Programme. Dies gewinnt besonders während der Implementierungs- und Initialisierungsphase einer neuen Simulation an Relevanz und stellt sicher, dass die Arbeitsvorgänge der Simulationsumgebung beobachtet werden können. Das Wichtigste einer Simulationsumgebung ist jedoch die Ausgabe der simulierten Daten. In der Simulationsumgebung geht es primär um den Einfluss der Fahrzeugladungen auf das elektrische Energienetz. Die implementierten Komponenten (z. B. Mobilitätssimulator) übermitteln ihre simulierten Leistungsdaten dem Energienetzsimulator (DIgSILENT PowerFactory). Dieser speichert alle Informationen und stellt diese anschaulich in Diagrammen dar. Dieser Simulator kann bereits verwendet werden, um diese Daten für spätere Analysen in externen Programmen (z. B. MATLAB) in Ergebnisdateien zu speichern.

Mit dem Energienetzsimulator können hingegen nicht alle für eine spätere Analyse benötigten Simulationsdaten gespeichert werden. Daher sammelt der Aggregator (siehe Abschnitt 4.3) die Ladeanfragen der Elektrofahrzeuge und stellt diese am Ende der Simulation als Textdatei zur Verfügung. Mit diesen Informationen können statistische Analysen der Ladevorgänge, wie die Verteilung der Ankunftszeiten oder der Stehzeiten an den Ladesäulen, durchgeführt werden.

4.7 Anwendungen der Simulationsumgebung

Die modulare Architektur erlaubt vielfältige Einsatzmöglichkeiten der entworfenen Simulationsumgebung. Die implementierten Komponenten können dank des Simulation Message Bus beliebig kombiniert und mehrfach eingebunden werden. Dieser Abschnitt demonstriert mögliche Konfigurationen der Simulationsumgebung, mit welchen verschiedene Untersuchungen durchgeführt werden können.

Der einfachste Aufbau der Umgebung, welcher in Kapitel 5 bei der dortigen Analyse eingesetzt wird, besteht aus je einer Instanz der jeweiligen Komponente (Abbildung 5.1 auf Seite 58) und dient der reinen Simulation der Auswirkungen von Ladevorgängen der Elektrofahrzeuge auf ein nachgebildetes elektrisches Verteilnetz. Dafür müssen keine Komponenten mit Anbindung an reale Ladestationen verwendet werden. Ebenso kann in dieser Verwendungsart der Simulationsmodus des Simulation Message Bus ausgenutzt werden, um den schnellst möglichen Simulationsablauf zu ermöglichen (siehe Kapitel 3.4.1).

Die Anforderung den parallelen Betrieb zwischen simulierten und hardwaremäßig aufgebauten Ladesäulen, welche über das Open Charge Point Protocol kommunizieren, zu ermöglichen, stellte den Ausgangspunkt des Architekturentwurfes dar. Diese Betriebsart wurde schon ausführlich in Kapitel 3.4 beschrieben und ist bereits in Abbildung 3.8 auf Seite 29 dargestellt.

Bei den beiden beschriebenen Möglichkeiten handelt es sich um generische Umgebungen, welche für verschiedene Simulationen eingesetzt werden können. Eine weitere, realitätsnahe Variante wird in Abbildung 4.14 gezeigt. Dieser Aufbau kann zum Beispiel zur Untersuchung der Auswirkungen auf ein elektrisches Verteilnetz einer Stadt oder Ortschaft eingesetzt werden. In diesem Energienetz betreiben zwei oder mehrere Firmen jeweils eine Ladeinfrastruktur für Elektrofahrzeuge (Mobilitätssimulator 1 und 2). Dabei werden die Ladevorgänge der Fahrzeuge in der jeweiligen Infrastruktur unabhängig voneinander koordiniert. In der Simulationsumgebung kann dies durch den Einsatz mehrerer Aggregatoren ermöglicht werden.

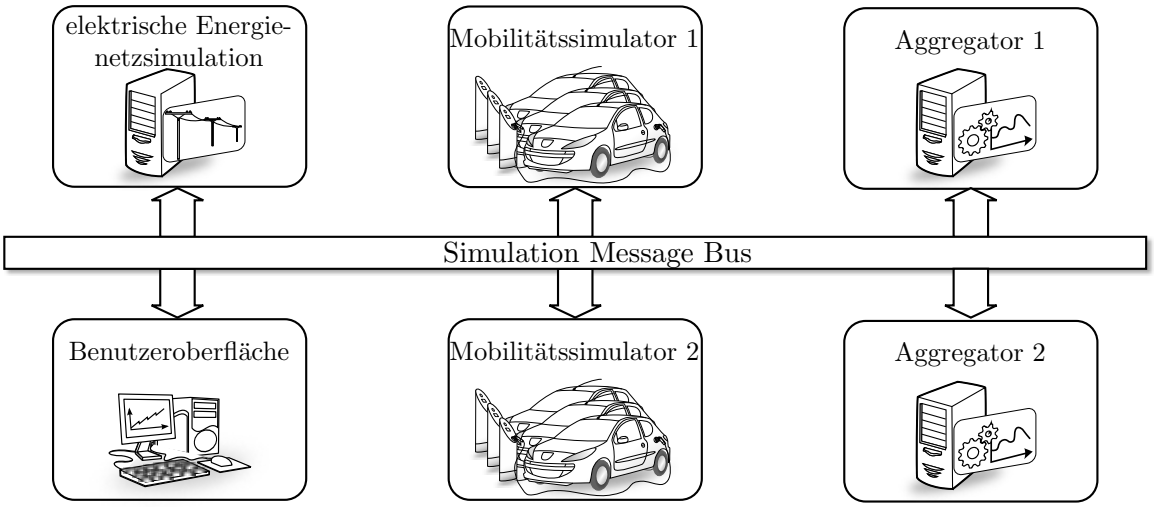

Abbildung 4.14: Energienetz mit mehreren Ladeinfrastrukturbetreibern

Ein weiteres Einsatzgebiet erlaubt die Simulation eines Lademanagements, welches sich über die Verteilnetze mehrerer Gebiete oder Ortschaften erstreckt und in Abbildung 4.15 dargestellt ist. Dazu wird jede Ortschaft mithilfe eines eigenen und unabhängigen elektrischen Energienetzsimulators nachgebildet. Ein einziger Aggregator übernimmt die Strategie zur gemeinsamen Koordination der Ladevorgänge in diesen beiden Ortschaften. Die Verdoppelung der Mobilitätssimulatoren in der Simulationsumgebung dient der Aufteilung der Ladeinfrastrukturen. Dadurch wird eine übersichtliche Verwaltung, sowie eine Beschleunigung des Simulationsprozesses durch Verteilung erreicht.

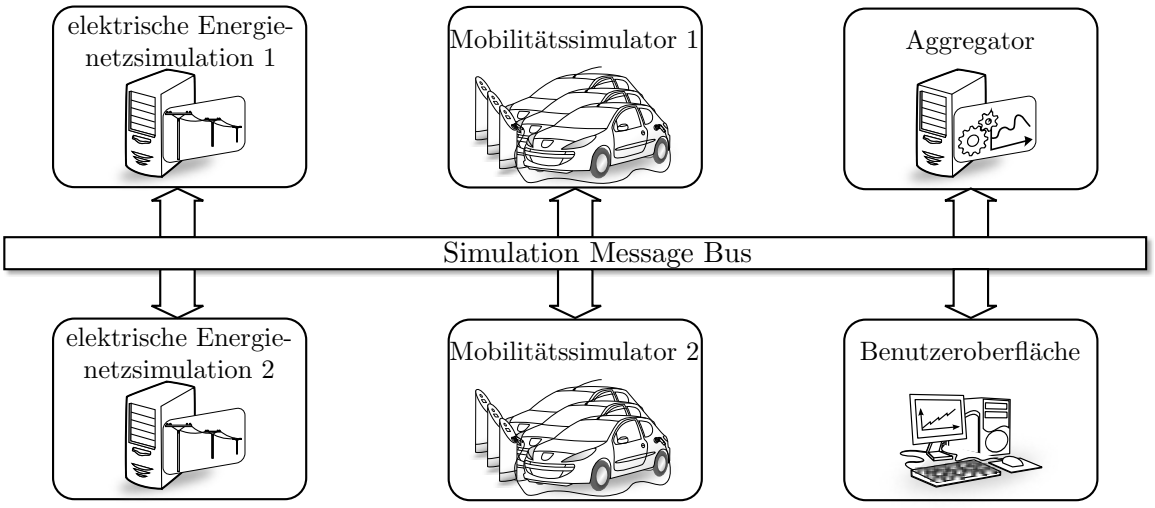

Abbildung 4.15: Lademanagement in mehreren Energienetzen

Die letzte Architektur zeigt die Anbindung der real aufgebauten Ladesäulen an die Simulationsumgebung (Abbildung 4.16). Dabei können verschiedene Schnittstellenprotokolle gleichzeitig betrieben werden, wenn für jedes verwendete Protokoll ein eigener Backend-Simulator eingesetzt wird. Diese Simulatoren haben dieselbe Schnittstelle zum Simulation Message Bus, wie der

Mobilitäts- oder OCPP-Simulator. Diese Konfiguration wird benötigt, um gegenseitige Wechselwirkungen der Ladesäulen zu testen, wobei diese über verschiedene Kommunikationsprotokolle angesteuert werden.

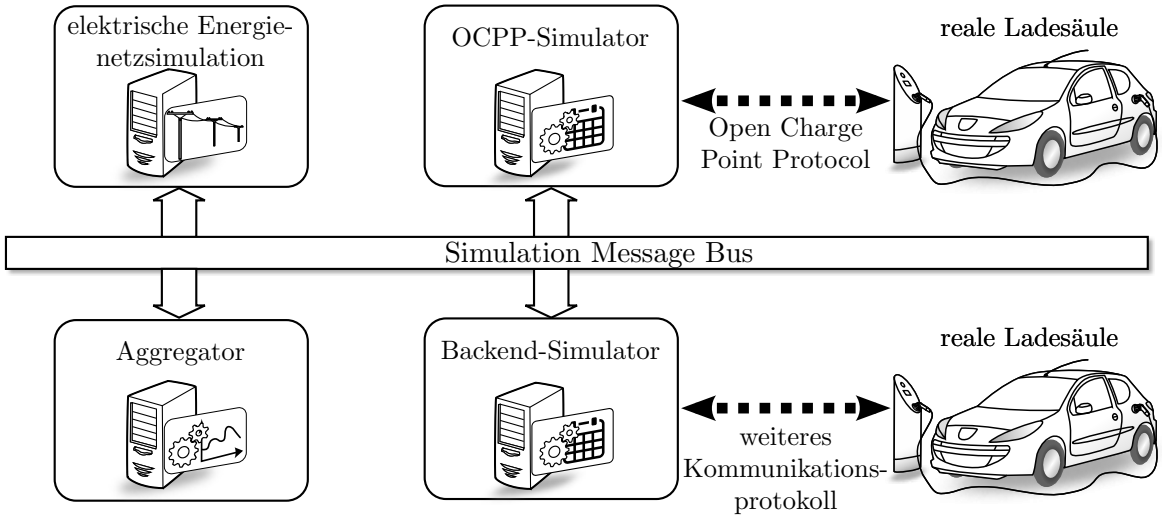

Abbildung 4.16: Einsatz verschiedener Kommunikationsprotokolle

Die hier beschriebenen Möglichkeiten dienen lediglich als Bespiele für verschiedene Verwendungszwecke. Dank des modularen Designs können die dargestellten Architekturen zu beliebigen Simulationsumgebungen kombiniert werden.

5 Einflussanalyse von Elektrofahrzeugen

Die implementierten Komponenten der Simulationsumgebung, welche im letzten Kapitel dargestellt wurden, werden in diesem Abschnitt herangezogen, um die Auswirkungen der Ladevorgänge von Elektrofahrzeugen auf ein elektrisches Verteilnetz zu analysieren. Dabei wird ein Vergleich zwischen unterschiedlichen Strategien gezogen, welche beim Lademanagement zum Einsatz kommen können. Ebenso dienen diese Simulationen zur Funktionsüberprüfung bzw. Validierung der gesamten Simulationsumgebung, um ein mögliches Fehlverhalten dieser aufzeigen zu können.

Der derzeitige Fokus des Demand Side Managementes liegt besonders in der optimalen Ausnutzung der regenerativer Energieerzeugung. Dabei wird der Energieverbrauch an die vorhandene Erzeugung dynamisch angepasst. Dies erfordert eine bidirektionale Kommunikation zwischen Verbrauchern und Energienetzbetreiber, um den aktuellen Zustand des elektrischen Netzes berücksichtigen zu können. Mit dieser Simulation soll aber gezeigt werden, dass es bereits mit einfachen Mitteln möglich ist, die Spitzenleistung zu bestimmten Tageszeiten zu reduzieren und den Energieverbrauch über den Tag gleichmäßiger zu verteilen. Es werden Ladealgorithmen eingesetzt, welche ohne Informationen über den aktuellen Zustand des elektrischen Energienetzes auskommen und daher keine zentrale Regelungs- oder Steuerungseinheit benötigen. Somit ist es möglich, diese Algorithmen lokal im Laderegler des Elektrofahrzeuges zu implementieren und dadurch die Installationskosten für die gesamte Ladeinfrastruktur, welche durch den Aufwand einer zusätzlichen Kommunikation entstehen, zu verringern.

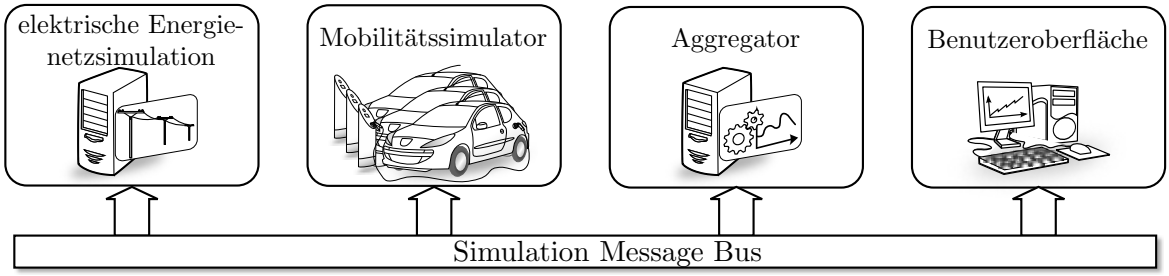

Abbildung 5.1: Benötigte Komponenten für die Simulation

Da nur eine Einflussanalyse durchgeführt wird, ist eine hardwaremäßige Anbindung einer realen Ladesäule an die Simulationsumgebung nicht vonnöten, wodurch die Simulationsgeschwindigkeit drastisch gesteigert werden kann. Daher findet der in Kapitel 3.4.5 erläuterte OCPP-Simulator in dieser Simulation keine Anwendung. Um die Simulation durchführen zu können, werden die Elemente benötigt, welche in Abbildung 5.1 dargestellt sind.

Die Funktionsweise und der Aufbau der dargestellten Komponenten wurde bereits in den letzten beiden Kapiteln ausführlich behandelt. Diese Komponenten verwenden ein Modell des elektrischen Energienetzes bzw. Tagesfahrprofile von Elektrofahrzeugen, welche im Anschluss genauer beschrieben werden, um das Verhalten verschiedener Managementalgorithmen im evaluierten Energienetz analysieren zu können. Die Kommunikation zwischen den Teilnehmern der Umgebung erfolgt über den Simulation Message Bus, dessen genaue Einstellungen – in Form der Konfigurationsdatei – im Anhang (A.2) angeführt sind.

Um in den verschiedenen Simulationen reproduzierbare Ergebnisse zu erhalten, müssen bestimmte Vorbedingungen gegeben sein:

- Das Elektrofahrzeug wird bei jedem Stopp an einer Ladesäule geladen.
- Die benötigte Energie des jeweiligen Ladevorganges hängt nur vom Ladestand der Batterie ab. Es wird versucht ein Elektrofahrzeug bei jedem Ladevorgang vollständig, d.h. bis zum Erreichen des maximalen Batterieladestandes, aufzuladen.
- Es müssen an jeder Ladestation genügend Ladepunkte vorhanden sein, um jedes in der Simulation befindliche Elektrofahrzeug aufladen zu können. Das Verhalten des Fahrzeugbenutzers bei einer vollkommen belegten Ladestation kann schlecht simulationstechnisch erfasst werden und wird durch diese Vorkehrung unterbunden. Genügend Ladepunkte gewährleisten, dass ein Elektrofahrzeug jederzeit geladen werden kann und dessen Einfluss auf jeden Fall in der Simulation berücksichtigt wird.

5.1 Elektrisches Verteilnetzmodell

Für die Einflussanalyse wird ein elektrisches Verteilnetz verwendet, das sich an das Energienetz in [PHS+05] anlehnt. Dabei wird die Grundstruktur mit den entsprechenden Leitungen und deren Parametern direkt übernommen. Dieses adaptierte Netz wird in Abbildung 5.2 dargestellt. Die größte Änderung stellt dabei die Umstellung der vorhandenen unsymmetrischen Komponenten auf symmetrische dar. Dadurch kann dieses Testnetz in der Simulationsumgebung auf einfachere Weise modelliert und verwendet werden.
Dieses Energienetzmodell wird für den Simulator DIgSILENT PowerFactory [18] angepasst, um anschließend von der Simulationsumgebung gesteuert werden zu können. Ebenso wurden an bestimmten Punkten im Energienetz, wie dies aus der Abbildung ersichtlich ist, variable Lasten angeschlossen, welche die Ladestationen (Abb. 5.2: LS-xx) repräsentieren. Die Leistungsparameter dieser Lasten in der Netzsimulation werden dabei vom externen Mobilitätssimulator über den Simulation Message Bus angesteuert.

Im originalen elektrischen Energienetz werden drei Lasttypen definiert, für die ein normiertes Tageslastprofil hinterlegt ist [PHS+05, S.8]. Jedes dieser normierten Profile wird mit der maximalen Scheinleistung (S_{max}), welche für jede Last in Abbildung 5.2 angegeben ist, skaliert. Durch die speziell vorgegebene Anordnung zerfällt dieses Netz in die folgenden Teilbereiche:

- Wohngegend
- Industriegegend
- Einkaufszentren

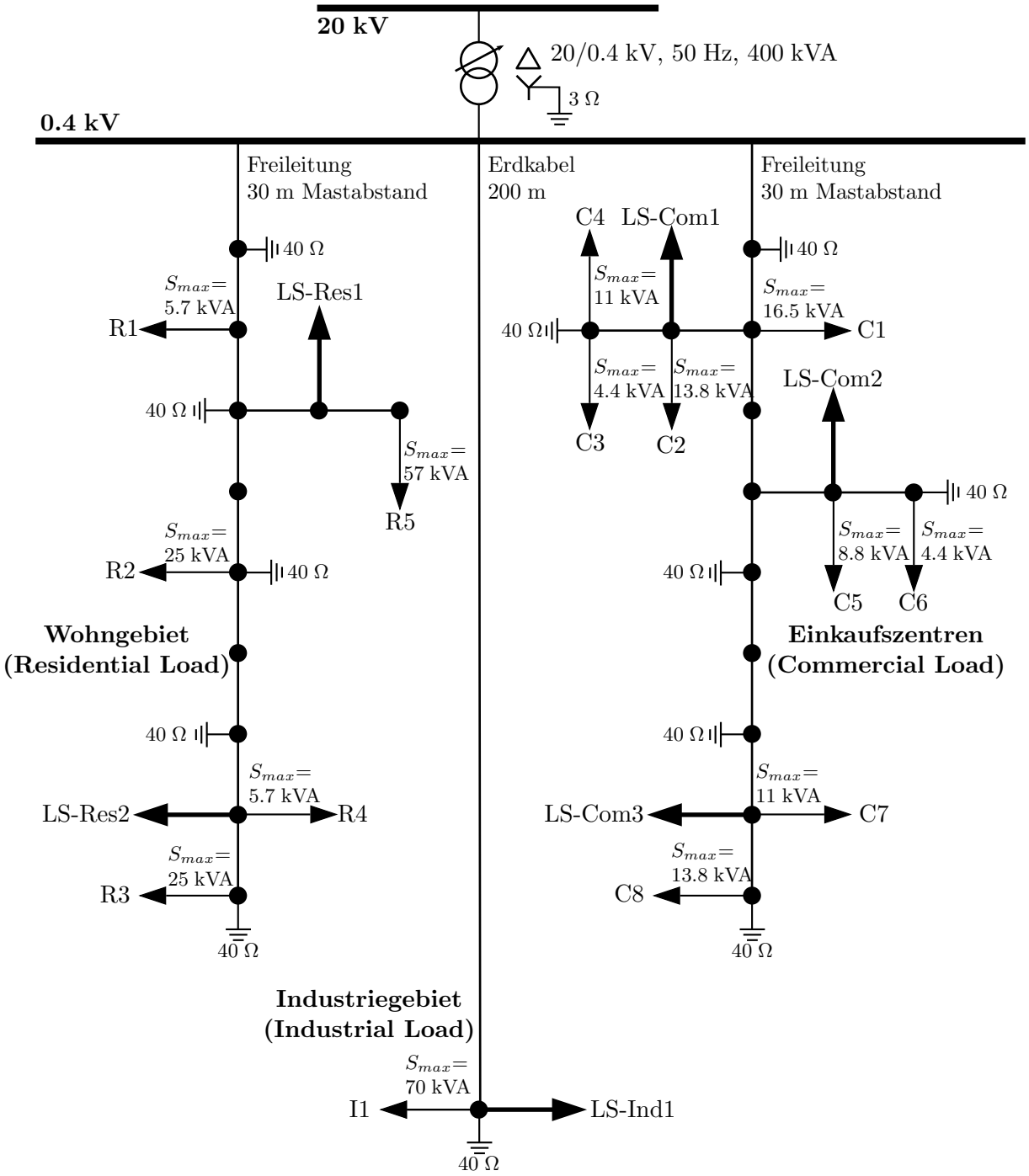

Abbildung 5.2: Modell des elektrischen Verteilnetzes (vgl. [PHS+05, S.7])

Diese drei Bereiche, jeweils durch einen eigenen Leitungsstrang repräsentiert, stimmen ebenfalls mit den unter Kapitel 2.1 bereits dargestellten Anforderungsprofilen, welche den Ladevorgang charakterisieren, überein. Die Grundbelastung, erzeugt durch die vorgegebenen Lastprofile der einzelnen Lasten, wird zusammengefasst und in den Ergebnissen (Abb. 5.9 – 5.13) ebenfalls angeführt.

5.2 Fahrzeugdaten

Neben den Tagesprofilen der Lasten im elektrischen Verteilnetzmodell werden auch Fahrprofile der Elektrofahrzeuge benötigt. Diese Profile wurden vom Austrian Institute of Technology zur Verfügung gestellt und mithilfe der Software MatSim/EvSim erzeugt (siehe Abschnitt 2.5). Aufgrund des implementierten Parsers können diese Daten direkt mit kleinen Anpassungen im Mobilitätssimulator verwendet werden (siehe Abschnitt 4.5).

Für die Simulation werden Tagesfahrprofile, wie exemplarisch in Tabelle 3.1 auf Seite 33 bereits dargestellt, von 268 Elektrofahrzeugen hinterlegt. Diese Profile stellen typische Fahrbewegungen, wie z. B. Wohnort–Arbeitsstätte–Wohnort oder Wohnort–Einkaufszentrum–Wohnort, dar. Ein Auszug aus dieser Profildatei, welche in dieser Simulation verwendet wird, befindet sich im Anhang A.1. Diese 268 Elektrofahrzeuge verursachen in der Simulation 586 Ladevorgänge, wobei 28 % auf die Einkaufszentren, 31 % auf die Industriegegend und 41 % auf die Wohngegend entfallen.

Die Profile wurden jeweils für drei aufeinanderfolgende Tage erstellt, wobei sich die Ergebnisse, welche später elaboriert werden, nur auf den mittleren Tag beziehen. Dieser Ansatz ist notwendig, da bestimmte Ladealgorithmen die Ankunfts- bzw. Abfahrtszeit der Fahrzeuge für die Koordination der Ladevorgänge verwenden. Um reproduzierbare Ergebnisse für einen gesamten Tag gewährleisten zu können, dürfen daher der Beginn bzw. das Ende der durchgeführten Simulation diese Koordination nicht beeinflussen.

5.3 Lademanagementalgorithmen

Wie bereits eingangs erwähnt, werden in dieser Simulation Lademanagementstrategien miteinander verglichen, die sich durch ihre geringe Komplexität auszeichnen. Diese Strategien benötigen keine Informationen über den Zustand des elektrischen Energienetzes und können daher lokal und unabhängig im Laderegler des Elektrofahrzeuges implementiert werden.
Im Folgenden wird, zum besseren Verständnis, die Funktionsweise jedes einzelnen Algorithmus, der in dieser Simulation eingesetzt wird, erläutert. Dabei wird zur Darstellung ein einzelner Ladevorgang verwendet, bei dem die Energiemenge (graue Fläche) bzw. die Verweilzeit an der Ladestation bei jedem Algorithmus identisch sind.

Sofortladung

Die Sofortladung stellt jenes Konzept dar, welches momentan am häufigsten bei der Ladung von Elektrofahrzeugen zum Einsatz kommt. Der Ladevorgang, welcher in Abbildung 5.3 beschrieben ist, erfolgt sofort nach dem Anschließen des Elektrofahrzeuges an das Stromnetz. Dabei wird die maximal erreichbare Ladeleistung verwendet, wodurch die Ladedauer möglichst kurz gehalten wird.

Abbildung 5.3: Sofortladung

Spätladung

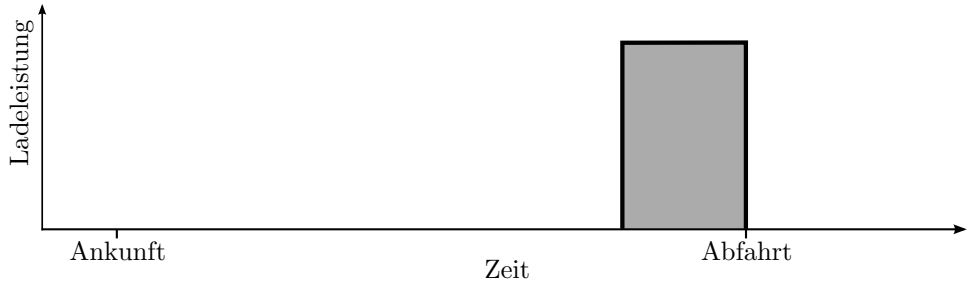

Abbildung 5.4: Spätladung

Die Spätladung ist dem Konzept der Sofortladung sehr ähnlich und ist in Abbildung 5.4 dargestellt. Hier wird der Ladevorgang aber an das Ende der Verweildauer an der Ladestation verschoben. Der Startzeitpunkt des Ladevorganges wird dabei so gewählt, dass die benötigte Energie zum spätest möglichen Zeitpunkt der Standzeit des Fahrzeuges übertragen wird.

Minimale Ladeleistung

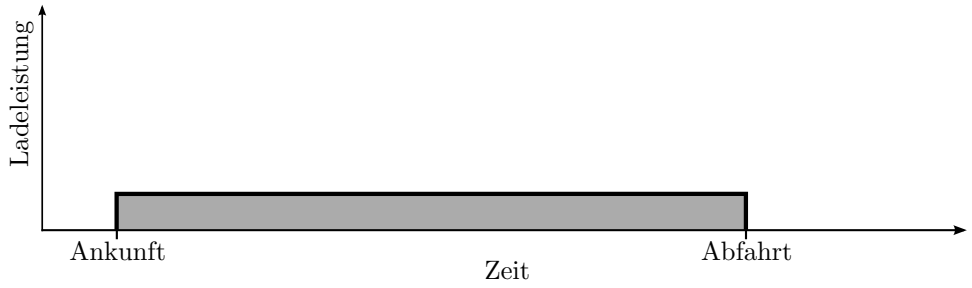

Abbildung 5.5: Minimale Ladeleistung

Die Ladestrategie „Minimale Ladeleistung“ kann auch als kontinuierliche Ladung bezeichnet werden, da der Ladevorgang des Elektrofahrzeuges über die gesamte Standzeit an der Ladesäule erfolgt. Abbildung 5.5 veranschaulicht diese Strategie, bei der die Ladeleistung das Niveau von Haushaltsgeräten erreichen kann.
Da die benötigte Ladeleistung von der angeforderten Energie und der Verweildauer an der Ladesäule abhängig ist, muss der Laderegler in der Lage sein, diese über einen weiten Leistungsbe-

reich effizient bereitstellen zu können. Dies ist meist technisch nur mit großem Aufwand möglich, da die Effizienz des Ladereglers nur für einen bestimmten Bereich optimiert werden kann. Daher ist die Verwendung dieses Ladealgorithmus oft mit höheren Verlusten behaftet.

Zufälliger Ladebeginn

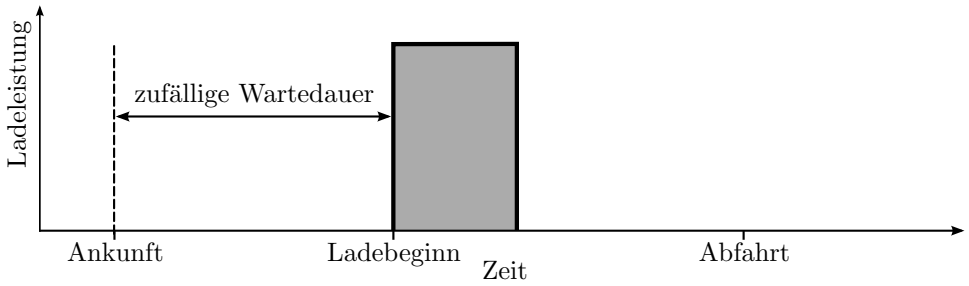

Abbildung 5.6: Zufälliger Ladebeginn

Bei dieser Strategie wird, wie bereits bei der Sofort- bzw. Spätladung, die maximal mögliche Ladeleistung vorgegeben, wodurch sich die Zeitdauer des Ladevorganges auf ein Minimum verkürzt. Die Verwendung des maximalen Ladestroms begünstigt die optimale Ausnutzung der Energieübertragung, da der Laderegler meist für diesen Bereich bestmöglich ausgelegt ist und die Verluste dadurch minimiert werden können.
Bei der Sofort- oder Spätladung korreliert der Ladevorgang mit der Ankunft bzw. der Abfahrt des Elektrofahrzeuges von der Ladesäule. Dabei werden mehrere Elektrofahrzeuge gleichzeitig geladen, da oft mehrere Anwender denselben Tagesrhythmus besitzen. Dieses Problem wurde bereits in Abschnitt 1.1 ausführlich behandelt. Die Grundidee dieser Ladestrategie beruht auf der bereits beschriebenen optimalen Ausnutzung der Infrastruktur und versucht dabei die Gleichzeitigkeit der Ladevorgänge zu verhindern. Dies wird durch das Zufallsprinzip erreicht, indem der Startzeitpunkt des Ladevorganges zufällig gewählt wird. Abbildung 5.6 veranschaulicht dieses Prinzip.

Zufälliger Ladebeginn mit Gewichtung

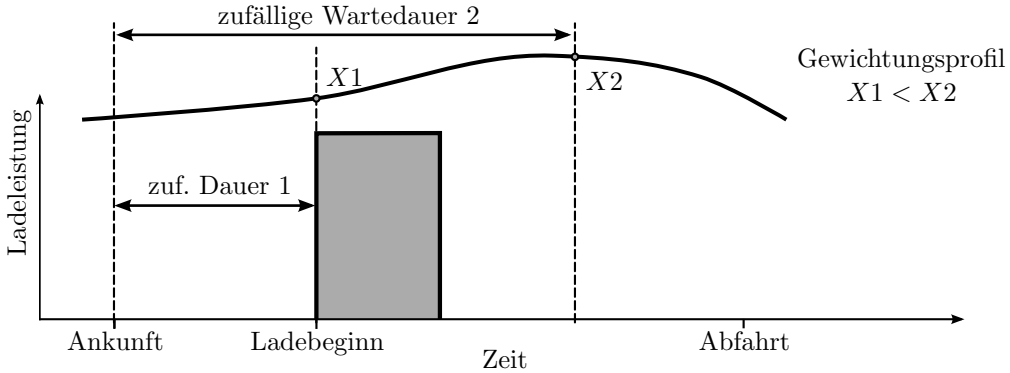

Abbildung 5.7: Zufälliger Ladebeginn mit Gewichtung

Die letzte zu untersuchende Strategie, welche in Abbildung 5.7 dargestellt ist, führt den vorher beschriebenen Ansatz fort. Dabei werden zwei zufällige Startzeitpunkte erzeugt, von denen einer für den Ladevorgang ausgewählt wird. Die Selektion dieses Startzeitpunktes erfolgt mithilfe eines generischen Grundlastprofiles, welches in Abbildung 5.8 dargestellt ist. Dieses Profil entspricht der Leistung der Sekundärseite des speisenden Netztransformators ohne der Belastung durch Elektrofahrzeuge. Wie in Abbildung 5.2 ersichtlich ist, setzt sich dieser Wert aus der Summe der einzelnen Teilbereiche zusammen.
Dieser Algorithmus arbeitet nach dem Prinzip, dass die bereits vorhandenen Leistungen im Grundlastprofil an den generierten Startzeitpunkten miteinander verglichen werden und jener Zeitpunkt gewählt wird, bei dem die Leistung geringer ist. Dadurch ist es möglich, Tageszeiten, bei denen das elektrische Energienetz weniger belastet wird, zu bevorzugen.

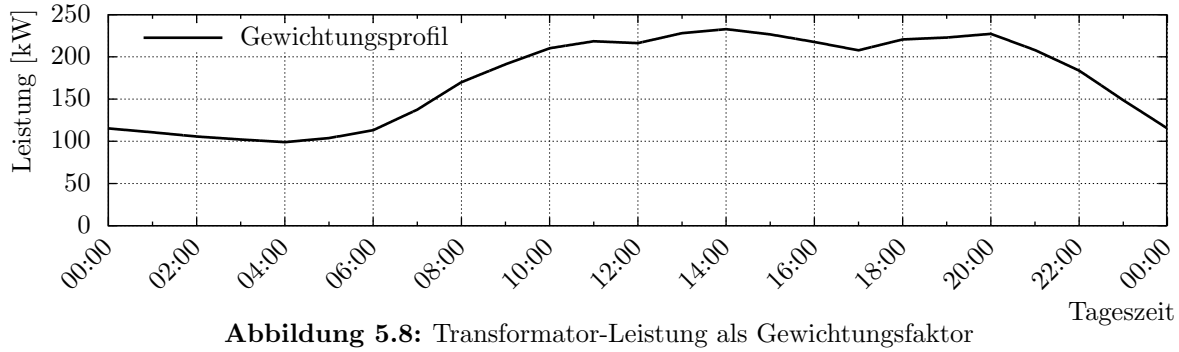

Abbildung 5.8: Transformator-Leistung als Gewichtungsfaktor

5.4 Simulationsergebnisse

Da bereits in Abschnitt 5.1 erläutert wurde, dass das hinterlegte elektrische Verteilnetz in drei Teilbereiche zerfällt, erfolgt auch die Auswertung der Simulationsergebnisse für jeden Ladealgorithmus getrennt. Ebenso repräsentieren diese Bereiche die verschiedenen Anforderungsprofile aus Kapitel 2.1.

In den Diagrammen wird die Grundbelastung des jeweiligen Teilbereiches, welche durch Haushalte oder Industrieanlagen erzeugt wird, durch die strichlierte Linie visualisiert. Diese Kurve enthält also die reine Belastung des elektrischen Energienetzes ohne den Ladevorgängen der Elektrofahrzeuge. Die durchgezogene Kurve wiederum beschreibt die gesamte Belastung, welche durch alle Verbraucher, einschließlich der Elektrofahrzeuge, des jeweiligen Netzstranges erzeugt wird.
Die Fläche zwischen diesen beiden Kurven drückt die Energiemenge aus, welche für die Ladevorgänge der Elektrofahrzeuge vom Energienetz bereitgestellt werden muss und ist in allen durchgeführten Simulationen äquivalent.

Sofortladung

Die Verteilung der Ladevorgänge erfolgt durch die unterschiedlichen Ankunftszeiten der Fahrzeuge an den Ladestationen und ist in Abbildung 5.9 dargestellt. Im Wohngebiet werden dabei die Fahrzeuge über den Tag und nicht in den Nachtstunden geladen. Eine starke Gleichzeitigkeit der Ladevorgänge tritt besonders im Industriegebiet auf, wo sich der Arbeitsbeginn der Arbeitnehmer auf wenige Stunden am Vormittag verteilt. In Einkaufszentren müssen die Fahrzeuge während der Geschäftsöffnungszeiten mit Energie versorgt werden.

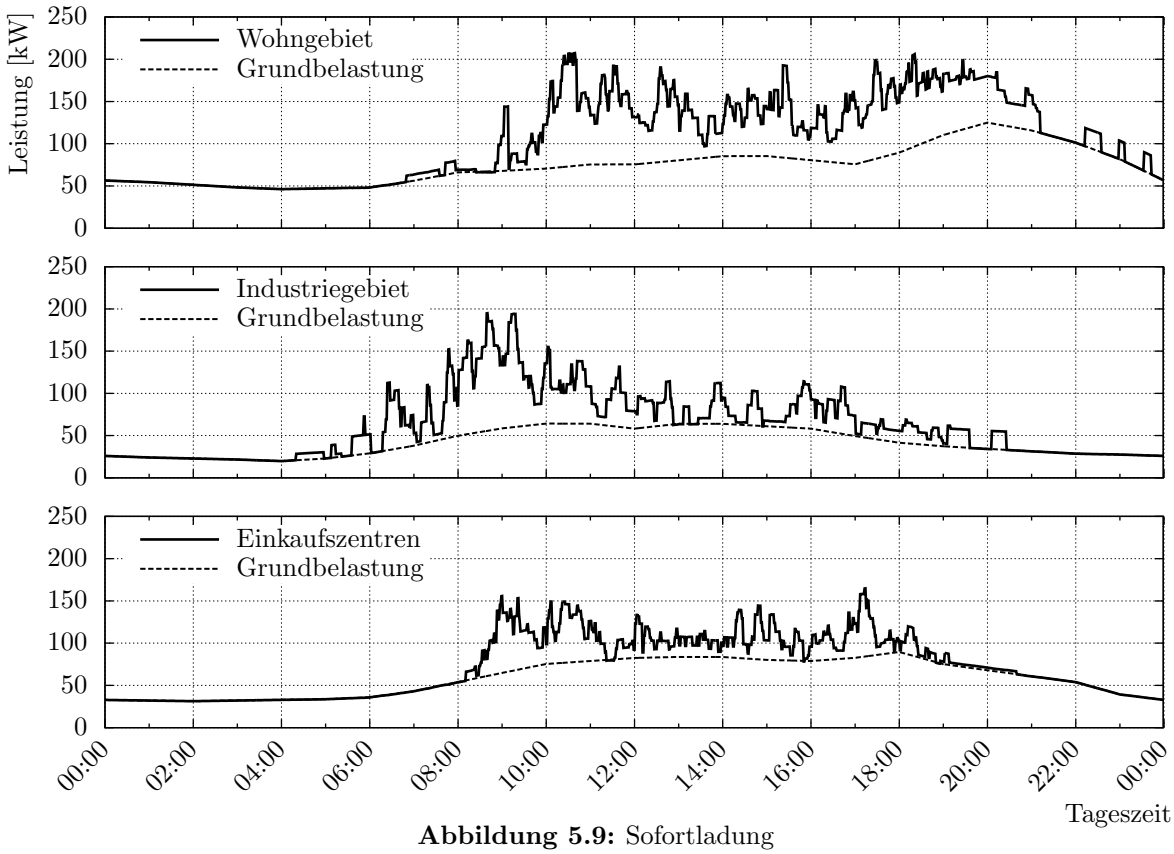

Abbildung 5.9: Sofortladung

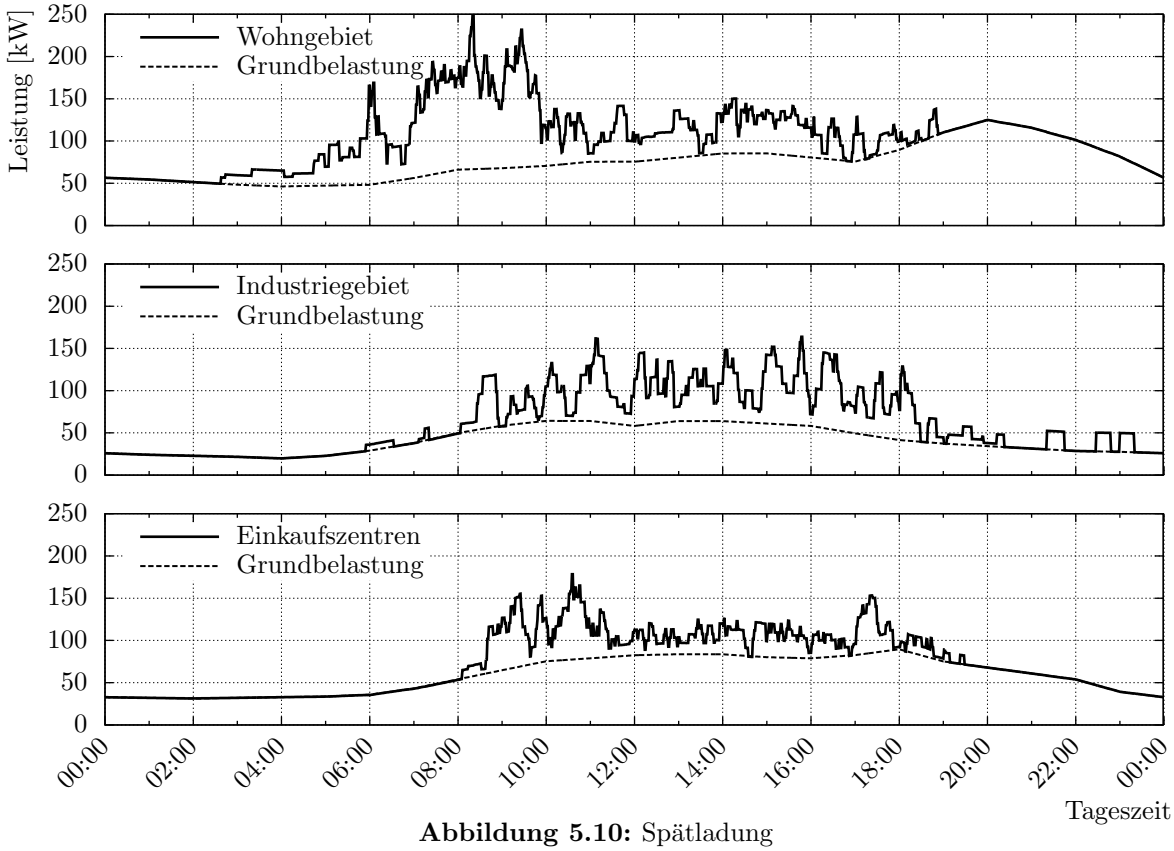

Abbildung 5.10: Spätladung

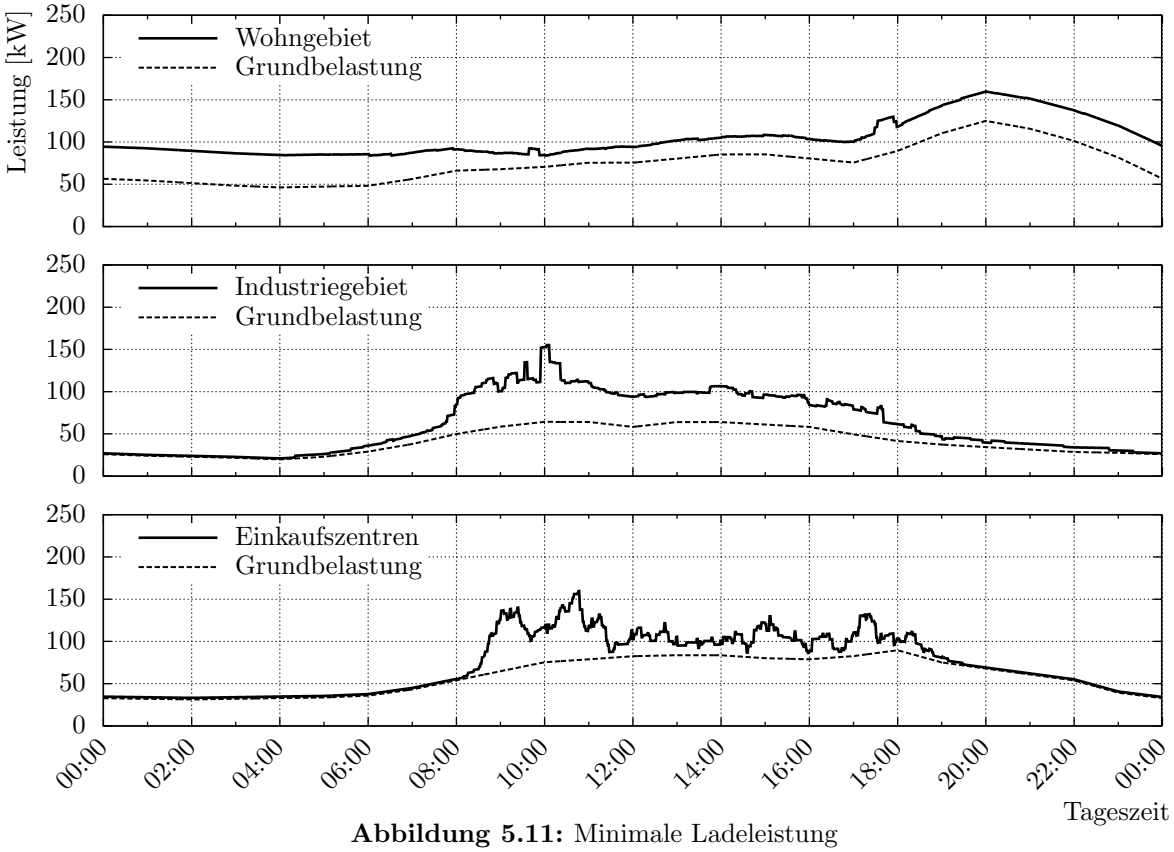

Abbildung 5.11: Minimale Ladeleistung

Spätladung

Bei der Spätladung, welche in Abbildung 5.10 dargestellt ist, korrelieren die Ladevorgänge mit der Abfahrtszeiten der Fahrzeuge von der Ladesäule. Im Wohngebiet entsteht dabei – im Vergleich zur Sofortladung – eine Lastspitze in den Morgenstunden. Diese kann als Resultat der, bereits oben erwähnten Gleichzeitigkeit des Arbeitsbeginns, angesehen werden. Gegensätzlich dazu, kann im Industriegebiet eine sichtbare Steigerung der Anzahl der Ladevorgängen zu den Morgenstunden beobachtet werden. Hier kann auf unterschiedliche und zeitlich verteilte Beendigungen der täglichen Arbeit geschlossen werden. In Einkaufszentren lässt sich hingegen keine große Veränderung zum vorherigen Algorithmus feststellen.

Minimale Ladeleistung

Die Ladung mit minimaler Leistung bedingt eine konstante Aufteilung aller Ladevorgänge über die Zeit, sodass das elektrische Energienetz eine gleichmäßige Belastung erfährt. Das Wohngebiet in Abbildung 5.11 stellt dies besonders deutlich dar, da hier lange Standzeiten an den Ladesäulen vorhanden sind. Trotzdem treten zwischenzeitlich verhältnismäßig geringe Lastspitzen auf, die auf unbedingt notwendige Ladevorgänge mit kurzen Ladedauern zu bestimmten Tageszeiten von einzelnen Fahrzeugen zurückführen sind. Bei diesem Algorithmus lassen sich die Schaltvorgänge der einzelnen Fahrzeuge kaum feststellen, da die Ladeleistung des Fahrzeuges einen geringeren Wert aufweist als in den bereits zuvor angeführten Strategien. Die Gleichzeitigkeit, die durch An- oder Abfahrt der Fahrzeuge entsteht, kann bei dieser Methode bereits gut unterbunden werden.

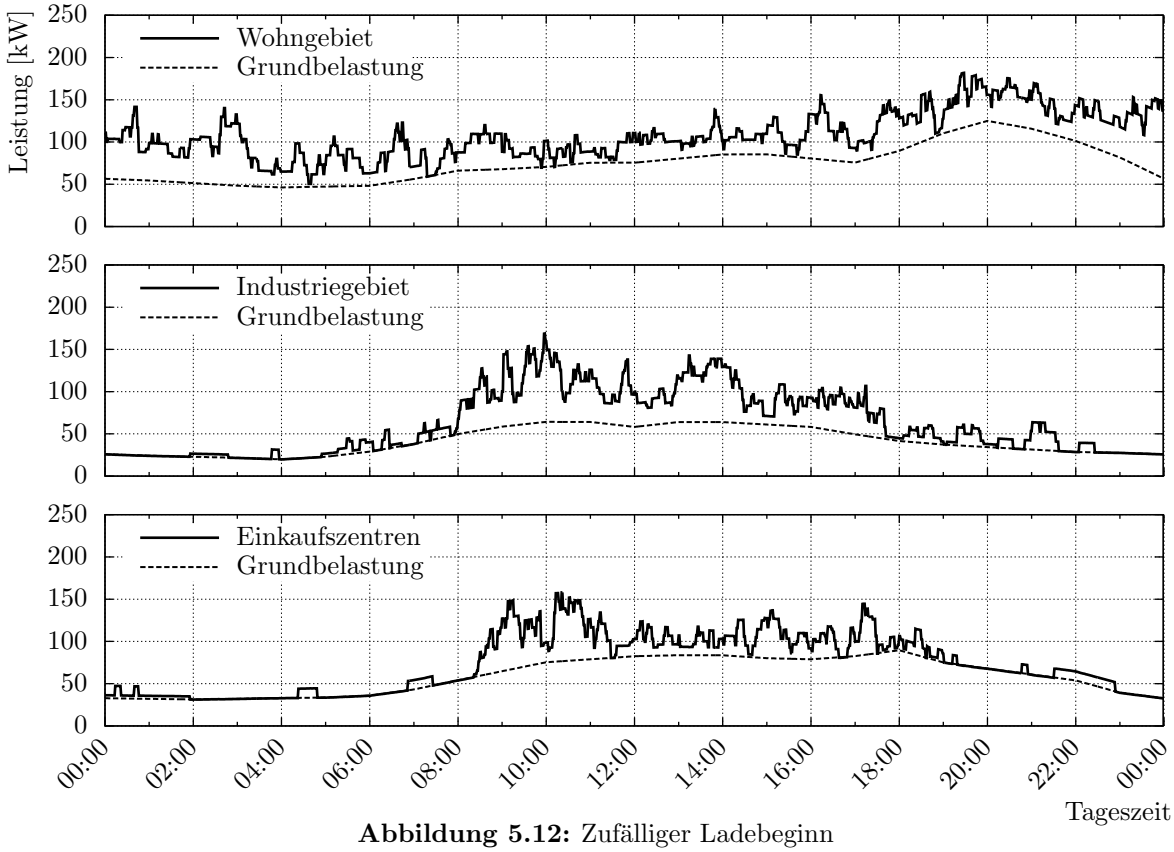

Abbildung 5.12: Zufälliger Ladebeginn

Zufälliger Ladebeginn

Der hier verwendete Algorithmus, welcher in Abbildung 5.12 gezeigt wird, bewirkt eine Ladung der Fahrzeuge mit dem maximal möglichen Ladestrom. Trotz dieser Gegebenheit entstehen nur geringere Spitzenbelastungen des elektrischen Energienetzes, im Vergleich zur Sofortladung. Besonders in Wohngebieten nähern sich die Auswirkungen dieser Methode dem kontinuierlichen Ladevorgang (minimale Leistung) an.

Zufälliger Ladebeginn mit Gewichtung

Die Gewichtung bei der Auswahl des Ladebeginns verschlechtert die Zufälligkeit der Ladevorgänge wieder. In Abbildung 5.13 wird dies besonders im Industriegebiet deutlich, wo es gegenüber der letzten Methode, zu einer stärkeren zeitlichen Häufung kommt. Hingegen hat dieser Algorithmus in Wohngebieten den Vorteil, dass die Ladevorgänge stärker in die Nachtstunden verschoben werden und die gesamte Spitzenbelastung während der Tagesstunden des elektrischen Verteilnetzes reduziert werden kann.

Zusammenfassung der Ergebnisse

Der Einsatz von einfachen Algorithmen, welche lokal im Laderegler des Elektrofahrzeuges implementiert werden können, erlaubt bereits eine starke Beeinflussung der Ladevorgänge. Dabei kann

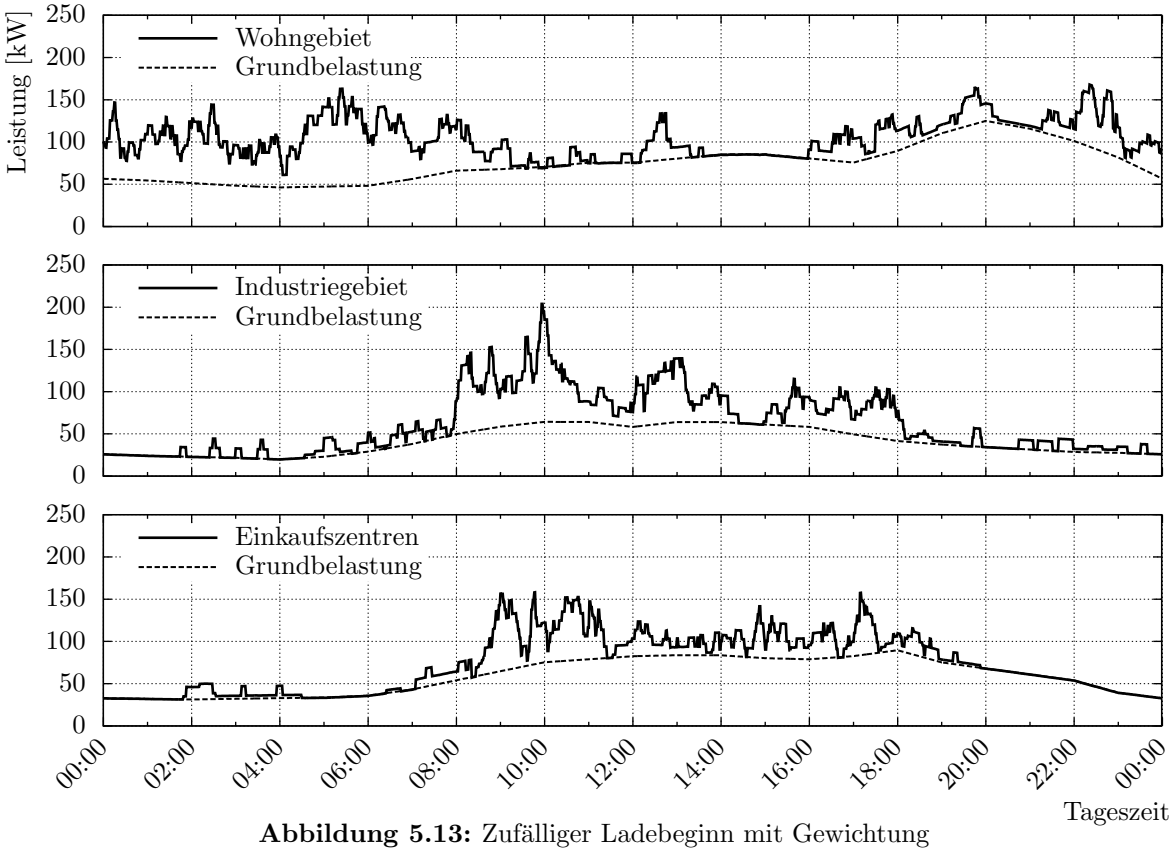

Abbildung 5.13: Zufälliger Ladebeginn mit Gewichtung

nur auf lokale Daten, wie Ladedauer und benötigte Lademenge, zurückgegriffen werden, um eine Steuerung des Ladevorganges zu erlauben.
Ohne zeitnahe oder prognostizierte Informationen über die bereits vorhandene Belastung des elektrischen Energienetzes kann nur die Strategie verfolgt werden, die Gleichzeitigkeit der Ladevorgänge zu unterbinden. Dadurch können Spitzenlasten reduziert und das elektrische Energienetz möglichst konstant über den Tag hinweg belastet werden. Um in zukünftigen Smart Grids die regenerativen Energien optimal auszunutzen, muss der elektrische Energieverbrauch an die Erzeugung angepasst werden. Dies benötigt den zusätzlichen Austausch von Informationen über den aktuellen Zustand des elektrischen Versorgungsnetzes und erfordert daher eine bidirektionale Kommunikation zwischen Energienetzbetreiber und der Ladeinfrastruktur für Elektrofahrzeuge.
Ebenso zeigen diese Simulationen bereits, dass sich die verwendeten Ladealgorithmen unterschiedlich stark auf die einzelnen Anwendungsgebiete auswirken. Die größten Potentiale zur Lastverschiebung bieten dabei Wohngebiete, wo die langen Standzeiten der Elektrofahrzeuge sehr effektiv ausgenutzt werden können. Hingegen können, in Bezug auf Einkaufszentren, keine signifikanten Unterschiede zwischen den einzelnen Lademanagementmethoden festgestellt werden. Daraus lässt sich bereits schließen, dass der Aufwand für die Implementierung und den Betrieb eines komplexen Lademanagementsystems, das die Koordination und die Regelung der Ladevorgänge übernimmt, in diesem Bereich wenig rentabel ist. Hier genügt bereits die heute gängige Methode der Sofortladung, um die Anforderungen in diesem Anwendungsbereich zu erfüllen.

6 Zusammenfassung

Dank der aktuell vorhandenen politischen Forcierung, um die Umweltbelastung durch den Verkehrssektor zu verringern, wird sich die Elektromobilität in Zukunft vermehrt durchsetzen und einen immer größeren Stellenwert im individuellen Personenverkehr einnehmen. Die momentan vorhandenen Einschränkungen von Elektrofahrzeugen, wie die geringe Reichweite, verhindern derzeit jedoch eine große Verbreitung. Trotzdem könnte sich dieser Umstand durch den technischen Fortschritt und die stetige Weiterentwicklung in den nächsten Jahrzehnten rasch ändern.

Da die Durchdringungsrate von Elektrofahrzeugen, wie bereits erwähnt, noch relativ gering ist, kann auch die dafür benötigte Energie ohne entsprechende Adaptionen bereitgestellt werden. In Zukunft, muss jedoch auch ein etwaiger Mehrverbrauch, der bei einem Zuwachs der Elektromobilität prognostiziert wird, durch das bestehende Versorgungssystem bewältigt werden. Die größte Herausforderung stellt dabei die Gleichzeitigkeit der Ladevorgänge dar, bei der das Energienetz an bestimmten Tageszeiten stark beansprucht wird. Diese kurzfristige Belastung muss daher in Zukunft unterbunden und eine optimale Ausnutzung der bereits vorhandenen Kapazitäten des Energienetzes garantiert werden. Dafür wird eine intelligente Ladeinfrastruktur benötigt, welche eine Koordinierung der Ladevorgänge erlaubt.

Um die Kundenakzeptanz für Elektrofahrzeuge zu erhöhen, wird eine einheitliche und europaweit interoperable Ladeinfrastruktur benötigt. Dafür müssen Ladesäulen im Verbund getestet und überprüft werden, um die gegenseitigen Beeinflussungen und Wechselwirkungen analysieren zu können. Die hier entwickelte Simulationsumgebung stellt den ersten Schritt in diese Richtung dar, indem ein System aus rechnerunterstützter Energienetzsimulation und real aufgebauten Ladesäulen entworfen wurde.
Die Analyse der notwendigen Komponenten eines Test- und Simulationssystems bildet den Ausgangspunkt für verschiedene Architekturansätze, welche untersucht und miteinander verglichen werden. Die Resultate dieser Untersuchung münden in einem Architekturvorschlag für eine modular entworfene Simulationsumgebung.
Diese Umgebung besteht aus mehreren Simulatoren, welche die Beeinflussung von Elektrofahrzeugen auf das elektrische Energienetz nachbilden können. Dabei ist das System in der Lage, die Ladevorgänge der Fahrzeuge zu koordinieren. Der verwendete Simulation Message Bus bewerkstelligt den einfachen Datenaustausch zwischen den unterschiedlichen Simulatoren, sowie den Aufbau eines verteilten Systems, welches sich über mehrere Computer erstrecken kann. Dieses verteilte System und der modulare Architekturentwurf erlauben den Einsatz der Simulationsumgebung in verschiedenen Anwendungsbereichen. Hier lassen sich gleichzeitig mehrere Energienetze oder mehrere Ladesäuleninfrastrukturen simulieren, in denen verschiedene Lademanagementstrategien zum Einsatz kommen. Ebenso ermöglicht die Anbindung an reale Ladesäulen eine Messung der

elektrischen Eigenschaften, als auch eine Verifizierung von Wechselwirkungen und gegenseitigen elektrischen Beeinflussungen.

Die durchgeführte Analyse verschiedener Lademanagementalgorithmen wurde zur Validierung der entwickelten Simulationsumgebung und deren Architekturentwurf eingesetzt. Es zeigt sich, dass die untersuchten Methoden die Gleichzeitigkeit der Ladevorgänge und das Auftreten von Spitzenlasten effektiv unterbinden. Die dabei eingesetzten Algorithmen zeichnen sich besonders durch ihre Einfachheit aus. Der nicht notwendige Informationsaustausch über den Zustand des elektrischen Energienetzes erlaubt zwar keine reaktive Beeinflussung, jedoch können diese Algorithmen bereits heute unabhängig in den Ladereglern der Elektrofahrzeuge eingesetzt werden.

6.1 Diskussion

Der dargelegte Entwurf der Architektur gibt bereits die Richtung zur einfachen Erweiterung und einem Ausbau der Simulationsumgebung vor. Die darauf basierende Implementierung bestätigt diese Möglichkeiten und zeigt, dass das entwickelte System durch das modulare Design für verschiedene Simulationen im Bereich der Elektromobilität eingesetzt werden kann. Diese Modularität und der Ansatz zur Verteilung der Umgebung über mehrere Simulationscomputer hinweg, erfordern aber einen umfangreichen Informationsaustausch zwischen den einzelnen Komponenten. Die Aufbereitung und Verteilung der dabei zu übertragenden Daten wiederum benötigt mehr Rechenleistung, wodurch die Simulationsgeschwindigkeit reduziert wird. Ebenso muss durch die Kommunikation mit Abweichungen in der Synchronisation gerechnet werden, wodurch Simulationsfehler verursacht werden können.

Die Implementierung der Hardwareanbindung einer Ladesäule im OCPP-Simulator erfolgte nur auf rudimentäre Weise. Dabei wurden nur jene Teile des Protokolls verwendet, welche für die Einbindung der Ladesäule in die Simulationsumgebung unbedingt notwendig sind. Dem System werden dabei nur die aktuell bezogenen Leistungen der realen, an das System angeschlossenen Ladesäulen bereitgestellt. Die Bedienung der Ladesäule kann dabei nicht automatisch durchgeführt werden, sondern bedarf immer einer manuellen Steuerung durch eine Person.

Die Simulationen der verschiedenen Lademanagementstrategien, welche in dieser Arbeit durchgeführt wurden, arbeiten auf Basis eines generisch erstellten elektrischen Energienetzes und einer fiktiv angenommenen Anzahl an Elektrofahrzeugen. Durch die Verwendung dieses angenommenen Netzes können nur die Auswirkungen der Algorithmen dargestellt und miteinander verglichen werden. Jedoch ist man nicht in der Lage, Rückschlüsse auf mögliche reale Ortschaften oder Gebiete zu ziehen.

Die Spitzenlast im elektrischen Verteilnetz wird durch die gleichzeitige Ankunft der Elektrofahrzeuge an den Ladesäulen verursacht. In den Simulationen wurde dabei festgestellt, dass die Einbringung des Zufalles die zeitliche Verteilung der Ladevorgänge bereits stark verbessert. Die analysierten Algorithmen verwenden aber keine Informationen der Energienetzsimulation, obwohl die Simulationsumgebung dazu in der Lage wäre. Dadurch kann nur die Strategie einer gleichmäßigen Verteilung der Ladevorgänge angestrebt werden. Mögliche vorhandene Belastungen des Energienetzes durch Haushalte oder Industrieanlagen werden dabei nicht berücksichtigt. Ebenso ist mit diesen Algorithmen keine netzgeführte Managementstrategie möglich und bevorzugte regenerative Energien können nicht optimal eingesetzt werden.

Die Analyse der Simulationen hat bereits gezeigt, dass es in bestimmten Bereichen, wie den Einkaufszentren, nicht von Relevanz ist, ob Lademanagementalgorithmen eingesetzt werden. In

diesen Bereichen erspart man sich den Aufwand und die Kosten für eine Implementierung einer intelligenten Steuerung der Ladevorgänge, da die Standzeiten der Elektrofahrzeuge zu kurz für eine mögliche Verschiebung des Ladevorganges sind. Im Gegensatz dazu, können Lademanagementalgorithmen in Wohngegenden sinnvoll eingesetzt werden, da hier lange Standzeiten der Fahrzeuge vorhanden sind, welche zur Lastverschiebung effektiv genutzt werden können.

6.2 Resümee und Ausblick

Bereits durch die entwickelte Software im Zuge dieser Diplomarbeit konnte die Möglichkeit geschaffen werden, die Simulationsumgebung für die Untersuchung von verschiedenen Szenarien im Bereich der Ladeinfrastrukturen für Elektrofahrzeuge einzusetzen (siehe Kapitel 4.7). Dies wird besonders durch das modulare Design der Architektur und des Simulation Message Bus gewährleistet. Besonders für den Entwurf und die anschließende Analyse von verschiedenen Lademanagementstrategien, die zur optimalen Ausnutzung des elektrischen Energienetzes dienen, kann diese Software auf einfache Weise verwendet werden.

Die in dieser Arbeit entworfene Architektur der Simulationsumgebung bildet den Grundstein für ein zukünftiges Testsystem für Ladeinfrastrukturen und die implementierte Software kann in verschiedene Richtungen ausgebaut und weiterentwickelt werden.
Zu den möglichen Einsatzgebieten gehört die bereits beschriebene Anwendung in der Analyse von Lademanagementalgorithmen. Hier können neue Algorithmen zur Verwendung kommen, welche reaktiv auf den aktuellen Zustand des simulierten Energienetzes reagieren. Ebenso lässt sich eine Verbindung zum elektrischen Energiemarkt herstellen, wobei je nach aktueller Marktlage der Ladevorgang gesteuert wird.
Der Ausbau zu einer automatisierbaren Testumgebung, welcher bereits in der Aufgabenstellung gefordert wird, lässt sich aufgrund des modularen Designs problemlos realisieren. Für den Einsatz des Systems in einem Testlabor, müssen zusätzliche Emulatoren in die Testumgebung integriert werden, welche die Beeinflussung der Versorgungsspannung einer realen Ladesäule erlauben. Ebenso müssen am elektrischen Ausgang einer Ladesäule Emulatoren eingesetzt werden, die den Ladevorgang eines Elektrofahrzeuges nachbilden.
Die Durchführung von Hardware-in-the-loop-Simulationen beschränkt sich derzeit auf die Verwendung des Kommunikationsprotokolls OCPP v1.5, welches noch nicht in der Lage ist, eine Lademanagementfunktion zu übertragen. Die Umgebung kann für den Einsatz der, derzeit im Entwurf befindlichen, Version 2.0 vorbereitet und ausgebaut werden.

Der Einsatz von intelligenten Steuerungen und Regelungen in zukünftigen Smart Grids ermöglicht die Ausschöpfung neuer Potentiale bei der optimalen Ausnutzung der Energienetzinfrastruktur. Besonders im Bereich der Elektromobilität können Lastmanagementstrategien sinnvoll eingesetzt werden. Trotz der steigenden Komplexität, die bei einer intelligenten Ladeinfrastruktur benötigt wird, muss die Zuverlässigkeit der eingesetzten Systeme im Vordergrund stehen. Nichts würde der Verbreitung von Elektrofahrzeugen mehr schaden, als dringend benötigte, jedoch aufgrund von unzuverlässigen Ladesystemen nicht geladene Fahrzeuge.

Wissenschaftliche Literatur

[aca10] *Wie Deutschland zum Leitanbieter für Elektromobilität werden kann.* Springer Berlin Heidelberg, 2010 (acatech BEZIEHT POSITION). – ISBN 978–3–642–13828–7

[BGJ+10] Binding, C. ; Gantenbein, D. ; Jansen, B. [u. a.]: Electric vehicle fleet integration in the danish EDISON project - A virtual power plant on the island of Bornholm. In: *Power and Energy Society General Meeting, 2010 IEEE*, 2010. – ISSN 1944–9925, S. 1–8

[Bra08] Brauner, G.: Infrastrukturen der Elektromobilität. In: *e&i Elektrotechnik und Informationstechnik* 125 (2008), Nr. 11, S. 382–386. – ISSN 0932–383X

[BYBJ12] Bingliang, Zhang ; Yutian, Sun ; Bingqiang, Li ; Jianxiang, Li: A Modeling Method for the Power Demand of Electric Vehicles Based on Monte Carlo Simulation. In: *Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific*, 2012. – ISSN 2157–4839, S. 1–5

[Dor12] Dorresteijn, Steven: Herausforderungen der Ladeinfrastruktur-Branche. In: *e&i Elektrotechnik und Informationstechnik* 129 (2012), Nr. 5, S. 362–363. – ISSN 0932–383X

[DTU10] Davito, Brandon ; Tai, Humayun ; Uhlaner, Robert: The smart grid and the promise of demand-side management. In: *McKinsey on Smart Grid* (2010), S. 38–44

[Fas11] Faschang, Mario: *System für die koordinierte Ladung von Elektrofahrzeugen*, Technische Universität Wien, Diplomarbeit, 2011

[FDLK13] Faschang, Mario ; Dimitriou, Pavlos ; Leber, Thomas ; Kupzog, Friederich: Projekt DG DemoNetz Smart LV Grid - Deliverable zu AP2 / ICT TU Vienna, AIT Energy Department. 2013. – Forschungsbericht

[FF13] Fries, Steffen ; Falk, Rainer: Electric Vehicle Charging Infrastructure–Security Considerations and Approaches. In: *International Journal On Advances in Internet Technology* 6 (2013), January, S. 57–67

[Hel09] Helmers, Eckard: *Bitte wenden Sie jetzt.* 1. Aufl. Weinheim : Wiley-VCH, 2009. – ISBN 978–3–527–32648–8

[KMB+12] Kleine-Möllhoff, Peter ; Benad, Holger [u. a.]: Die Batterie als Schlüsseltechnologie für die Elektromobilität der Zukunft / ESB Business School Reutlingen. 2012. – Forschungsbericht. `http://www.esb-business-school.de/fileadmin/_research/dokumente/Diskussionsbeitraege/2012-3-Reutlinger-Diskussionsbeitraege-Mark-Mngmt-E-Mobility-Batterie.pdf`

[Kön10] König, 1952: *Die Geschichte des Automobils.* Stuttgart : Reclam, 2010 (Reclam-Taschenbuch ; 20214). – ISBN 978–3–15–020214–2

[KSWH10] Käbisch, S. ; Schmitt, A. ; Winter, M. ; Heuer, J.: Interconnections and Communications of Electric Vehicles and Smart Grids. In: *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, S. 161–166

[KVS13] Kampker, Achim ; Vallée, Dirk ; Schnettler, Armin: Elektromobilität. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. – ISBN 978–3–642–31986–0

[LB08] Leitinger, C. ; Brauner, G.: Nachhaltige Energiebereitstellung für elektrische Mobilität. In: *e&i Elektrotechnik und Informationstechnik* 125 (2008), Nr. 11, S. 387–392. – ISSN 0932–383X

[Lie12] Lienkamp, Markus: *Elektromobilität - Hype oder Revolution?* Springer Berlin Heidelberg, 2012 (VDI-Buch). – ISBN 978–3–642–28548–6

[Lit10] Litzlbauer, Markus: Erstellung und Modellierung von stochastischen Ladeprofilen mobiler Energiespeicher. In: *11.Symposium Energieinnovation, Graz*, 2010. – `http://publik.tuwien.ac.at/files/PubDat_185241.pdf`

[LSL10] Leitinger, Christoph ; Schuster, Andreas ; Litzlbauer, Markus: Smart Electric Mobility - Speichereinsatz für regenerative elektrische Mobilität und Netzstabilität. In: *11. Symposium Energieinnovation, Graz*, 2010. – `http://publik.tuwien.ac.at/files/PubDat_184996.pdf`

[Luk13] Lukas, Martin: *Infrastructure for Integration of Smart Grids and E-Mobility*, University of Applied Sciences Technikum Wien, Diplomarbeit, August 2013

[Mac06] Mackiewicz, R.E.: Overview of IEC 61850 and benefits. In: *Power Engineering Society General Meeting, 2006. IEEE*, 2006, S. 8 pp.

[MKFS13] Mosshammer, R. ; Kupzog, F. ; Faschang, M. ; Stifter, M.: Loose coupling architecture for co-simulation of heterogeneous components. In: *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, 2013. – ISSN 1553–572X, S. 7570–7575

[NPE10] Zwischenbericht der Arbeitsgruppe 3 Lade-Infrastruktur und Netzintegration / Nationale Plattform Elektromobilität. 2010. – Forschungsbericht

[PHS+05] Papathanassiou, Stavros ; Hatziargyriou, Nikos ; Strunz, Kai [u. a.]: A benchmark low voltage microgrid network. In: *Proceedings of the CIGRE Symposium: Power Systems with Dispersed Generation*, 2005

[Pil12] Pilorget, Lionel: *Testen von Informationssystemen.* Wiesbaden : Vieweg+Teubner Verlag, 2012. – ISBN 978–3–8348–8677–4

[PSF12] Pina, André ; Silva, Carlos ; Ferrão, Paulo: The impact of demand side management strategies in the penetration of renewable electricity. In: *Energy* 41 (2012), Nr. 1, S. 128 – 137. – 23rd International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems, {ECOS} 2010. – ISSN 0360–5442

[PWL10] Pötscher, F. ; Winter, R. ; Lichtblau, G.: *Elektromobilität in Österreich: Szenario 2020 und 2050.* Umwelbundesamt, 2010. – ISBN 9783990040584

[RHR12] Rolink, Johannes ; Horenkamp, Willi ; Rehtanz, Christian: Ladeinfrastrukturen für die Netzintegration von Elektrofahrzeugen. 60 (2012), S. 84–

[RPL+12] Rezania, Rusbeh ; Prüggler, Wolfgang ; Litzlbauer, Markus [u. a.]: V2G-Strategies - Empfehlungen für Elektromobilität in Österreich / Technische Universität Wien. 2012. – Forschungsbericht. `http://publik.tuwien.ac.at/files/PubDat_184996.pdf`

[RSRW11] Ruthe, Sebastian ; Schmutzler, Jens ; Rehtanz, Christian ; Wietfeld, Christian: Study on V2G Protocols against the Background of Demand Side Management. In: *IBIS - Interoperability in Business Information Systems*,

2011. – http://www.ibis.uni-oldenburg.de/download/issues/11/Study_on_V2G_Protocols_against_the_Background_of_Demand_Side_Management.pdf

[SC12] Su, Wencong ; Chow, Mo-Yuen: Performance Evaluation of an EDA-Based Large-Scale Plug-In Hybrid Electric Vehicle Charging Algorithm. In: *Smart Grid, IEEE Transactions on* 3 (2012), March, Nr. 1, S. 308–315. – ISSN 1949–3053

[SGM11] SGMS - V2G-INTERFACES -Publizierbarer Endbericht / NEUE ENERGIEN 2020. 2011. – Forschungsbericht. http://www.smartgridssalzburg.at/fileadmin/user_upload/downloads/Endbericht_V2G-Interfaces.pdf

[SL13] Schuster, Andreas ; Litzlbauer, Markus: Begleitforschung der TU Wien in „ElectroDrive Salzburg“ - Endbericht / Technische Universität Wien – Institut für Energiesysteme und Elektrische Antriebe. 2013. – Forschungsbericht. http://www.ea.tuwien.ac.at/fileadmin/t/ea/projekte/EDS/TUWien_Begleitforschung_EDS_Endbericht_Anhang.pdf

[SU13] Stifter, M. ; Übermasser, S.: Dynamic simulation of power system interaction with large electric vehicle fleet activities. In: *PowerTech (POWERTECH), 2013 IEEE Grenoble*, 2013

[SWA12] Schmutzler, J. ; Wietfeld, C. ; Andersen, C.A.: Distributed energy resource management for electric vehicles using IEC 61850 and ISO/IEC 15118. In: *Vehicle Power and Propulsion Conference (VPPC), 2012 IEEE*, 2012, S. 1457–1462

[UOZ13] Ustun, T.S. ; Ozansoy, C.R. ; Zayegh, A.: Implementing Vehicle-to-Grid (V2G) Technology With IEC 61850-7-420. In: *Smart Grid, IEEE Transactions on* 4 (2013), Nr. 2, S. 1180–1187. – ISSN 1949–3053

[VFCI12] Veneri, O. ; Ferraro, L. ; Capasso, C. ; Iannuzzi, D.: Charging infrastructures for EV: Overview of technologies and issues. In: *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS), 2012*, 2012. – ISSN 2165–9400, S. 1–6

[YWWS13] Young, Kwo ; Wang, Caisheng ; Wang, LeYi ; Strunz, Kai: Electric Vehicle Battery Technologies. In: Garcia-Valle, Rodrigo (Hrsg.) ; Peças Lopes, João A. (Hrsg.): *Electric Vehicle Integration into Modern Power Networks*. Springer New York, 2013 (Power Electronics and Power Systems). – ISBN 978–1–4614–0133–9, S. 15–56

Internet Referenzen

[1] Mineralölwirtschaftsverband e.V. (Association of the German Petroleum Industry). *Zusammensetzung des Preises für Superbenzin 1972 bis 2011*, 2013. http://www.mwv.de/index.php/daten/statistikenpreise/?loc=10, abgerufen am 27.9.2013.

[2] Statistik Austria. *Schnellbericht 10.7 - Registerbasierte Statistiken Kalenderjahr 2013 - Demographie (RS) Registerzählung 2011*, 2013. https://www.statistik.at/web_de/static/registerbasierte_statistiken_2013_-_demographie_rs_sb_10.7__071439.pdf, abgerufen am 27.9.2013.

[3] ADAC Fahrzeugtechnik. *Elektroautos: Marktübersicht/Kenndaten*, April 2013. www.adac.de/_mmm/pdf/27373_46583.pdf, abgerufen am 3.10.2013.

[4] Statistik Austria. *Fahrzeug-Bestand 30. September 2013 in Österreich*, September 2013. http://www.statistik.at/web_de/static/fahrzeug-bestand_30._september_2013_062059.pdf, abgerufen am 30.10.2013.

[5] Austrian Standards Institute. *ÖVE/ÖNORM EN 61851-1: 2012 03 01: Elektrische Ausrüstung von Elektro-Straßenfahrzeugen - Konduktive Ladesysteme für Elektrofahrzeuge - Teil 1: Allgemeine Anforderungen (IEC 61851-1:2010)*. https://lesesaal.austrian-standards.at/search/Details.action?dokkey=412182, abgerufen am 5.11.2013.

[6] Austrian Standards Institute. *ÖVE/ÖNORM EN 62196-1: 2012 12 01: Stecker, Steckdosen, Fahrzeugkupplungen und Fahrzeugstecker - Konduktives Laden von Elektrofahrzeugen - Teil 1: Allgemeine Anforderungen (IEC 62196-1:2011)*. https://lesesaal.austrian-standards.at/search/Details.action?dokkey=452531, abgerufen am 5.11.2013.

[7] Union of the Electricity Industry-EURELECTRIC. *European electricity industry views on charging Electric Vehicles*, April 2011. http://www.eurelectric.org/media/26100/2011-04-18_final_charging_statement-2011-030-0288-01-e.pdf, abgerufen am 23.11.2013.

[8] Digi-Key Corporation. *A Designer's Guide to Lithium Battery Charging*, September 2012. http://www.digikey.com/us/en/techzone/power/resources/articles/a-designer-guide-lithium-battery-charging.html, abgerufen am 8.11.2013.

[9] Marc Mültin, editor. *Wie IKT die eMobilität und das Smart Grid verbindet*, Neue Mobilität 13, Oktober 2013. http://www.neue-mobilitaet.info/neue-mobilitaet/neue-mobilitat-13, abgerufen am 18.11.2013.

[10] Karlheinz Schwarz, editor. *Kostengünstige IEC-61850-Lösung für kurze Entwicklungszeiten.* etz - Elektrotechnik + Automation, June 2010. http://www.etz.de/files/e00626zfe_beck.pdf, abgerufen am 13.3.2014.

[11] Open Charge Alliance. *OCPP v1.5 A functional description*, volume v2.0, 2012. http://www.ocppforum.net/sites/default/files/ocpp%201%205%20-%20a%20functional%20description%20v2%200_0.pdf, abgerufen am 18.11.2013.

[12] Open Charge Alliance. *Open Charge Point Protocol - Interface description between Charge Point and Central System*, 2012. http://www.ocppforum.net/sites/default/files/ocpp_specification_1.5_final.pdf, abgerufen am 18.11.2013.

[13] Open Charge Alliance. *Open Charge Point Protocol 2.0 - Interface description between Charge Point and Central System*, volume RC1, 2013. http://openchargealliance.org/resources/ocpp20RC1.zip, abgerufen am 12.1.2014.

[14] *Hubject GmbH Homepage*. http://www.hubject.com/, abgerufen am 20.11.2013.

[15] Hubject GmbH. *Open Intercharge Protocol (OICP) Version 1.1*, Juli 2013. http://www.hubject.com/pdf/closed/v_1.1_Open_Intercharge_Protocol_(OICP).pdf, abgerufen am 20.11.2013.

[16] *e-clearing.net Homepage*. http://www.e-clearing.eu/, abgerufen am 20.11.2013.

[17] MathWorks. *MATLAB - Die Sprache für technische Berechnungen*. http://www.mathworks.de/products/matlab/, abgerufen am 7.2.2014.

[18] DIgSILENT. *PowerFactory*. http://www.digsilent.de/index.php/products-powerfactory.html, abgerufen am 7.2.2014.

[19] U.S. Department of Energy. *GridLab-D*. http://www.gridlabd.org/, abgerufen am 7.2.2014.

[20] *Regional Eco Mobility 2030 (REM 2030) - Fahrpofile*. http://www.rem2030.de/rem2030-de/REM-2030-Fahrprofile.php, abgerufen am 7.2.2014.

[21] *Virtuelle Elektromobilität in München - Forschungsprojekt simuliert Elektroautos*. Dr. Ulrich Marsch Corporate Communications Center - Technische Universität München, 2013. http://www.idw-online.de/de/news563920, abgerufen am 7.2.2014.

[22] *MATSim - Multi-Agent Transport Simulation*, 2014. http://www.matsim.org/, abgerufen am 20.3.2014.

[23] *Electric Vehicle Simulator*. Eaton, 2013. http://www.eaton.com/ecm/idcplg?IdcService=GET_FILE&allowInterrupt=1&RevisionSelectionMethod=LatestReleased&noSaveAs=0&Rendition=Primary&dDocName=PA00406004E, abgerufen 25.11.2013.

[24] *EVE-100S EV Charger Test System*. Gridtest Systems Inc., 2013. http://www.gridtest.com/assets/EVE-100S-Data-Sheet.pdf, abgerufen am 25.11.2013.

[25] *OCPP Testbench*. EURISCO - Research & Development, 2014. http://www.eurisco.dk/en/technologies/evs/ocpp-testbench, abgerufen am 20.3.2014.

[26] Spitzenberger & Spies GmbH & Co. KG. *Versorgungsnetz Simulationssysteme 1-phasig und 3-phasig*, 2014. http://www.spitzenberger.de/Versorgungsnetz-Simulationssysteme.aspx, abgerufen am 5.2.2014.

[27] DIgSILENT. *PowerFactory Basic Software Features*. http://www.digsilent.de/index.php/products-powerfactory-basic_software_features.html, abgerufen am 10.2.2014.

[28] World Wide Web Consortium (W3C). *Web Services Architecture*, 2004. http://www.w3.org/TR/ws-arch, abgerufen am 18.2.2014.

[29] World Wide Web Consortium (W3C). *Web Services Description Language (WSDL) 1.1*, 2001. http://www.w3.org/TR/wsdl, abgerufen am 18.2.2014.

[30] Internet Engineering Task Force. *The JavaScript Object Notation (JSON) Data Interchange Format*, number 7159 in Request for Comments. IETF, March 2014. http://www.ietf.org/rfc/rfc7159.txt, abgerufen am 11.3.2014.

[31] Yidong Fang, editor. *JSON.simple - A simple Java toolkit for JSON*, 2014. https://code.google.com/p/json-simple/, abgerufen am 11.3.2014.

[32] *Java Architecture for XML Binding (JAXB)*. Sun Microsystems, 2014. https://jaxb.java.net/, abgerufen am 21.3.2014.

[33] *Apache CXF: An Open-Source Services Framework*. Apache Software Foundation, 2014.

http://cxf.apache.org/, abgerufen am 24.3.2014.

[34] *Simple Logging Facade for Java (SLF4J)*, 2014. http://www.slf4j.org/, abgerufen am 17.3.2014.

[35] *Apache log4j 1.2*. Apache Software Foundation, 2014. http://logging.apache.org/log4j/1.2/, abgerufen am 17.3.2014.

Anhang

A.1 Auszug der Fahrzeugprofile und Ladesäulenkonfiguration

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:mobilitySimData xmlns:ns2="at.ac.ait.CpSimEnv">
  <ecars>
    <ecar id="1" type="MITSUBISHI i-Miev" maxChargePower="11040" minChargePower="0"
        batteryCapacity="14240" relConsumption="129.45454" initBatterySOC="100">
      <event location="Residential02" action="CHARGE" date="2014-01-16 00:00:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-16 08:12:00" km="28" />
      <event location="Industrial01"  action="CHARGE" date="2014-01-16 08:40:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-16 09:48:00" km="30" />
      <event location="Residential02" action="CHARGE" date="2014-01-16 10:18:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-17 08:12:00" km="28" />
      <event location="Industrial01"  action="CHARGE" date="2014-01-17 08:40:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-17 09:48:00" km="30" />
      <event location="Residential02" action="CHARGE" date="2014-01-17 10:18:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-18 08:12:00" km="28" />
      <event location="Industrial01"  action="CHARGE" date="2014-01-18 08:40:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-18 09:48:00" km="30" />
      <event location="Residential02" action="CHARGE" date="2014-01-18 10:18:00" km="0" />
    </ecar>

    <ecar id="2" type="SMART Smart ED" maxChargePower="11040" minChargePower="0"
        batteryCapacity="17500" relConsumption="140.0" initBatterySOC="100">
      <event location="Residential02" action="CHARGE" date="2014-01-16 00:00:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-16 09:16:00" km="37" />
      <event location="Industrial01"  action="CHARGE" date="2014-01-16 09:53:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-16 10:20:00" km="8" />
      <event location="Residential02" action="CHARGE" date="2014-01-16 10:28:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-17 09:16:00" km="37" />
      <event location="Industrial01"  action="CHARGE" date="2014-01-17 09:53:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-17 10:20:00" km="8" />
      <event location="Residential02" action="CHARGE" date="2014-01-17 10:28:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-18 09:16:00" km="37" />
      <event location="Industrial01"  action="CHARGE" date="2014-01-18 09:53:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-18 10:20:00" km="8" />
      <event location="Residential02" action="CHARGE" date="2014-01-18 10:28:00" km="0" />
    </ecar>

    <ecar id="S1" type="NISSAN Leaf" maxChargePower="11040" minChargePower="0"
        batteryCapacity="24000" relConsumption="200.0" initBatterySOC="100">
      <event location="Residential01" action="CHARGE" date="2014-01-16 00:00:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-16 08:31:00" km="8" />
      <event location="Commercial01"  action="CHARGE" date="2014-01-16 08:39:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-16 08:54:00" km="8" />
      <event location="Residential01" action="CHARGE" date="2014-01-16 09:02:00" km="0" />
      <event location="road"          action="MOVE"   date="2014-01-17 08:31:00" km="8" />
      <event location="Commercial01"  action="CHARGE" date="2014-01-17 08:39:00" km="0" />
```

```
      <event location="road"           action="MOVE"   date="2014-01-17 08:54:00" km="8" />
      <event location="Residential01" action="CHARGE" date="2014-01-17 09:02:00" km="0" />
      <event location="road"           action="MOVE"   date="2014-01-18 08:31:00" km="8" />
      <event location="Commercial01"  action="CHARGE" date="2014-01-18 08:39:00" km="0" />
      <event location="road"           action="MOVE"   date="2014-01-18 08:54:00" km="8" />
      <event location="Residential01" action="CHARGE" date="2014-01-18 09:02:00" km="0" />
    </ecar>

    <!--  ... -->
  </ecars>

  <chargestations>
    <chargestation locationId="Residential01"
        powerFactoryPowerId="_1" powerFactoryPowerVar="_1"
        powerFactoryVoltageId="LS-Res01" powerFactoryVoltageVar="m:U1:bus1">
      <chargepoint id="CP01" maxPower="11000" />
      <chargepoint id="CP02" maxPower="11000" />
      <!--  ... -->
    </chargestation>

    <chargestation locationId="Residential02"
        powerFactoryPowerId="_2" powerFactoryPowerVar="_1"
        powerFactoryVoltageId="LS-Res02" powerFactoryVoltageVar="m:U1:bus1">
      <chargepoint id="CP01" maxPower="11000" />
      <chargepoint id="CP02" maxPower="11000" />
      <!--  ... -->
    </chargestation>

    <chargestation locationId="Industrial01"
        powerFactoryPowerId="_3" powerFactoryPowerVar="_1"
        powerFactoryVoltageId="LS-Ind01" powerFactoryVoltageVar="m:U1:bus1">
      <chargepoint id="CP01" maxPower="11000" />
      <chargepoint id="CP02" maxPower="11000" />
      <!--  ... -->
    </chargestation>

    <chargestation locationId="Commercial01"
        powerFactoryPowerId="_4" powerFactoryPowerVar="_1"
        powerFactoryVoltageId="LS-Com01" powerFactoryVoltageVar="m:U1:bus1">
      <chargepoint id="CP01" maxPower="11000" />
      <chargepoint id="CP02" maxPower="11000" />
      <!--  ... -->
    </chargestation>

    <chargestation locationId="Commercial02"
        powerFactoryPowerId="_5" powerFactoryPowerVar="_1"
        powerFactoryVoltageId="LS-Com02" powerFactoryVoltageVar="m:U1:bus1">
      <chargepoint id="CP01" maxPower="11000" />
      <chargepoint id="CP02" maxPower="11000" />
      <!--  ... -->
    </chargestation>

    <chargestation locationId="Commercial03"
        powerFactoryPowerId="_6" powerFactoryPowerVar="_1"
        powerFactoryVoltageId="LS-Com03" powerFactoryVoltageVar="m:U1:bus1">
      <chargepoint id="CP01" maxPower="11000" />
      <chargepoint id="CP02" maxPower="11000" />
      <!--  ... -->
    </chargestation>
  </chargestations>
</ns2:mobilitySimData>
```

A.2 Konfiguration der SMB-Channels

```
<Channel id="channel_sync" >
  <Proxy id="sync_routing" type="at.siemens.smb.proxy.impl.DynamicRoutingProxy" >
    <param name="mode" value="packet" />
    <param name="destination" value="*:sync_status.*=*    -> channel_logging" />
    <param name="destination" value="*:config.*=*         -> channel_logging" />
    <param name="destination" value="* #FILE_CTRL1_CONFIG -> channel_logging" />
    <param name="destination" value="* #FILE_CTRL2_CONFIG -> channel_logging" />
    <param name="destination" value="* #FILE_CTRL3_CONFIG -> channel_logging" />
    <param name="destination" value="* #FILE_CTRL4_CONFIG -> channel_logging" />
    <param name="destination" value="*                    -> *" />
  </Proxy>
  <Proxy id="proxy_sync_filelog" type="at.siemens.smb.proxy.impl.FileLogProxy" >
    <param name="file" value="./log/sync.log" />
  </Proxy>
  <Proxy id="sync_proxy" type="at.siemens.smb.proxy.impl.SyncProxy" >
    <param name="clients" value="6" />
    <param name="trigger" value="true" />
    <param name="sourceid" value="SyncProxy" />
    <param name="scenariofile" value="./resources/sync_scenarios.csv" />
    <param name="sym_start" value="load" />
  </Proxy>
</Channel>

<!-- PowerFactory -->
<Channel id="channel_grid" >
  <Proxy id="proxy_grid_routing" type="at.siemens.smb.proxy.impl.DynamicRoutingProxy" >
    <param name="mode" value="packet" />
    <param name="destination" value="*:sync.*=*     -> channel_sync" />
    <param name="destination" value="*:_system.*=* -> channel_logging" />
    <param name="destination" value="*             -> channel_logging,channel_mobsim,
                                                      channel_ocpp15,channel_flexop" />
  </Proxy>
  <Proxy id="proxy_grid_stream" type="at.siemens.smb.proxy.impl.ObjectStreamingProxy" >
    <param value="at.siemens.smb.data.packet" name="context"/>
  </Proxy>
  <Proxy id="proxy_grid_connect" type="at.siemens.smb.proxy.impl.SocketConnectorProxy" >
    <param name="port" value="52200" />
    <param name="binary" value="false" />
  </Proxy>
</Channel>

<!-- MobilitySimClient Client -->
<Channel id="channel_mobsim" >
  <Proxy id="proxy_mobsim_routing" type="at.siemens.smb.proxy.impl.DynamicRoutingProxy">
    <param name="mode" value="packet" />
    <param name="destination" value="*:sync.*=*     -> channel_sync,channel_logging" />
    <param name="destination" value="*:_system.*=* -> channel_logging" />
    <param name="destination" value="*:*.FlexReq=* -> channel_flexop" />
    <param name="destination" value="* #voltage    -> channel_grid" />
    <param name="destination" value="* #power      -> channel_grid" />
    <param name="destination" value="* #GUI        -> channel_simgui" />
  </Proxy>
  <Proxy id="proxy_mobsim_filelog" type="at.siemens.smb.proxy.impl.FileLogProxy" >
    <param name="file" value="./log/MobilitySimClient.log" />
  </Proxy>
  <Proxy id="proxy_mobsim_connect" type="at.siemens.smb.proxy.impl.SocketConnectorProxy">
    <param name="port" value="13003" />
    <param name="binary" value="true" />
  </Proxy>
</Channel>

<!-- Ocpp15 Client -->
<Channel id="channel_ocpp15" >
  <Proxy id="proxy_ocpp15_routing" type="at.siemens.smb.proxy.impl.DynamicRoutingProxy">
    <param name="mode" value="packet" />
    <param name="destination" value="*:sync.*=*     -> channel_sync,channel_logging" />
```

```
    <param name="destination" value="*:_system.*=*  -> channel_logging"                />
    <param name="destination" value="*:*.FlexReq=*  -> channel_flexop"                 />
    <param name="destination" value="* #voltage     -> channel_grid"                   />
    <param name="destination" value="* #power       -> channel_grid"                   />
    <param name="destination" value="* #GUI         -> channel_simgui"                 />
  </Proxy>
  <Proxy id="proxy_ocpp15_filelog" type="at.siemens.smb.proxy.impl.FileLogProxy" >
    <param name="file" value="./log/Ocpp15Client.log" />
  </Proxy>
  <Proxy id="proxy_ocpp15_connect" type="at.siemens.smb.proxy.impl.SocketConnectorProxy">
    <param name="port" value="13005"    />
    <param name="binary" value="true"   />
  </Proxy>
</Channel>

<!-- FlexOpClient -->
<Channel id="channel_flexop" >
  <Proxy id="proxy_flexop_routing" type="at.siemens.smb.proxy.impl.DynamicRoutingProxy">
    <param name="mode" value="packet" />
    <param name="destination" value="*:sync.*=*     -> channel_sync,channel_logging"   />
    <param name="destination" value="*:_system.*=*  -> channel_logging"                />
    <param name="destination" value="*:*.FlexResp=* -> channel_mobsim,channel_ocpp15"  />
    <param name="destination" value="* #voltage     -> channel_grid"                   />
    <param name="destination" value="* #GUI         -> channel_simgui"                 />
  </Proxy>
  <Proxy id="proxy_flexop_filelog" type="at.siemens.smb.proxy.impl.FileLogProxy">
    <param name="file" value="./log/flexopclient_msg.log" />
    <param name="max_records" value="10000" />
    <param name="max_history" value="10"    />
  </Proxy>
  <Proxy id="proxy_flexop_connect" type="at.siemens.smb.proxy.impl.SocketConnectorProxy">
    <param name="port" value="13004"    />
    <param name="binary" value="true"   />
  </Proxy>
</Channel>

<!-- SimGui -->
<Channel id="channel_simgui" >
  <Proxy id="proxy_simgui_routing" type="at.siemens.smb.proxy.impl.DynamicRoutingProxy">
    <param name="mode" value="packet" />
    <param name="destination" value="*:sync.*=*     -> channel_sync,channel_logging"   />
    <param name="destination" value="*:_system.*=*  -> channel_logging"                />
    <param name="destination" value="* #voltage     -> channel_grid"                   />
    <param name="destination" value="* #GUI         -> channel_logging,channel_mobsim,
                                                       channel_ocpp15,channel_flexop" />
  </Proxy>
  <Proxy id="proxy_simgui_connect" type="at.siemens.smb.proxy.impl.SocketConnectorProxy">
    <param name="port" value="13002"    />
    <param name="binary" value="true"   />
  </Proxy>
</Channel>

<!-- Logging channels -->
<Channel id="channel_logging" >
  <Proxy id="logging_filelog" type="at.siemens.smb.proxy.impl.FileLogProxy">
    <param name="file" value="./log/recordAll.log" />
    <param name="max_records" value="10000" />
    <param name="max_history" value="10"    />
  </Proxy>
</Channel>
```