ebök Institut GmbH

Vernetzung autonomer Agenten in einer Industrie 4.0-Umgebung mittels Distributed Ledger Technologien zu einem Virtuellen Kraftwerk

Projektkürzel: VK2G

Abschlussbericht über ein Entwicklungsprojekt,
gefördert unter dem Az: 34798/01 von der
Deutschen Bundesstiftung Umwelt

von

Prof. Dr. Claus Kahlert, ebök Institut GmbH
Prof. Dr. Antonio Notholt, Hochschule Reutlingen
Prof. Dr. Debora Coll-Mayor, Hochschule Reutlingen
Prof. Dr. Helmut Nebeling, Hochschule Reutlingen

Juni 2021

06/02

Projektkennblatt
der
Deutschen Bundesstiftung Umwelt

Az **34798/01**	Referat	Fördersumme **124.161 EUR**

Antragstitel	Vernetzung autonomer Agenten in einer Industrie 4.0-Umgebung mittels Distributed Ledger Technologien zu einem Virtuellen Kraftwerk
Stichworte	Energie

Laufzeit	Projektbeginn	Projektende	Projektphase(n)
24 Monate	**01.04.2019**	**31.03.2021**	**1**

Zwischenberichte

Bewilligungsempfänger	ebök Institut GmbH Stäudach 99 72074 Tübingen	Tel 07071/9597-19 Fax 07071/9597-21
		Projektleitung Prof. Dr. Claus Kahlert
		Bearbeiter Prof. Dr. Claus Kahlert
Kooperationspartner	Hochschule Reutlingen	

Zielsetzung und Anlass des Vorhabens

Das Ziel des Projekts besteht darin, in einer Industrie 4.0-Umgebung Last- und Erzeugungs-Flexibilität zu charakterisieren und für ein Virtuelles Kraftwerk zu nutzen. Dabei sollen netzdienliche Liefer- und Abnahmeversprechen mit hoher zeitlicher Auflösung (15min) mittels einer Distributed Ledger Technology (DLT) abgewickelt, überwacht und abgerechnet werden. Die Entwicklung strebt einen Technology Readiness Level (TRL) 4 an.

.

Darstellung der Arbeitsschritte und der angewandten Methoden

.

Das Projekt umfasst drei inhaltliche Arbeitspakete (APs) sowie „Berichte und Verbreitung“, AP4 und „Projektmanagement“, AP5. Das Projektkonsortium führt seine Untersuchungen in einer Simulationsumgebung sowie am Demonstrator Virtuelles Kraftwerk Neckar-Alb an der Hochschule Reutlingen durch.

AP1 untersucht möglichst einfache Netzmodelle, die für die Vorhersage von Lastverteilungen auf der Basis von Lieferverträgen möglichst präzise Aussagen erlauben, welche auf Preissignale an die Akteure abgebildet werden.

AP2 untersucht in einer I4.0-Umgebung Last und Erzeugungs-Flexibilität, die als netzdienliche Leistungen gehandelt werden können.

AP3 entwickelt ein Protokoll für einen Punkt-zu-Punkt-Handel und die Überwachung der Lieferung; weiterhin untersucht es Schwarmalgorithmen, welche für eine optimale Auslastung der Netz-Ressourcen sorgen.

Ergebnisse und Diskussion

.

Das Projekt demonstriert in einem Laboraufbau (TRL4) die Kommunikation autonomer Agenten mittels DLT-Technologie. Für die zwei Anwendungsfälle Stromhandel und Engpassmanagement optimieren die Netzwerk-Teilnehmer ihre Lastgänge und Erzeugung anhand von Preissignalen. Im gewählten Beispiel sind Haushalte mit PV-Anlagen und Elektrofahrzeugen sowie produzierende Betriebe dargestellt.

Das zentrale Instrument der Optimierung stellt die Nutzung von Flexibilität in der Produktionsplanung dar. Dadurch wird ein wesentlich größeres Potential erschlossen als bei einer Fernsteuerung via CLS-Schnittstelle, die auf unwichtige oder zufällig verfügbare Verbraucher zugreifen kann. Als weiterer großer Vorteil bleiben alle Daten in den Betrieben; nur Lastgänge und Preise werden ausgetauscht. Im gewählten Modell erhält jeder Teilnehmer via Smart Contract fast instantan eine Rückmeldung zu seinem angemeldeten Lastgang und kann damit entscheiden, ob sein geplantes Verhalten netzdienlich ist oder ein weiterer Iterationsschritt folgen sollte.

Mit dem beschriebenen Mechanismus optimiert der Schwarm der Teilnehmer gemeinsam das lokale Netz. Über Smart Contracts lässt sich dieser Prozess weitgehend automatisieren. Voraussetzung dafür ist, dass jeder Teilnehmer die interne Datenbank der Produktionsziele und Anlagenrestriktionen auf dem Laufenden hält.

Stadtwerke als Betreiber von Verteilnetzen können diesen Mechanismus zur Vermeidung von Netzengpässen als Alternative zu Investitionen in Betriebsmittel einsetzen. Unter den gegebenen Regularien und der verbreiteten Risiko-Aversität der Branche ist eine rasche Verbreitung in den nächsten Jahren allerdings nicht zu erwarten.

Öffentlichkeitsarbeit und Präsentation

Aufgrund der Pandemie-Situation fanden keine öffentlichen Veranstaltungen statt, bei denen Projektergebnisse vor Interessenten und Multiplikatoren vorgestellt werden konnten. Ein Teil der Projektergebnisse wurde auf den online durchgeführten Industriekolloquien der Semesterarbeiten unter Teilnahme einiger Unternehmensvertreter vorgestellt.

Bei internationalen Konferenzen wurden drei Beiträge sowie ein Zeitschriftenartikel eingereicht und akzeptiert.

Fazit

Das Projekt hat alle gesteckten Ziele erreicht. Mit dem Aufbau eines Modells gelang es auch, Studierende für das Thema zu interessieren, die Studienarbeiten dazu anfertigten

Die entwickelten Algorithmen sind für Unternehmen mit Einzel- oder Kleinserien-Fertigung auch ohne Teilnahme an Flexibilitätsmärkten von Interesse, da sie auch die Netzentgelte minimieren können.

DLT-Kommunikation zur Optimierung von Betriebsmitteln ist momentan für Stadtwerke noch kein vordringliches Thema, Mit der Zunahme von PV-Anlagen und privaten Ladestationen in Quartieren, wird das Thema zukünftig allerdings in den Vordergrund rücken.

Inhaltsverzeichnis

Anhänge

Abbildungsverzeichnis

Tabellenverzeichnis

Acronyme

A

ABI Application Binary Interface

AN Anschlussnutzer.

API Application Programming Interface.

B

BDEW Bundesverband der Energieund Wasserwirtschaft.

BEV Battery Electric Vehicle

BTC Bitcoin.

BTR Betreiber einer technischen Ressource.

C

CPU Central Processing Unit.

D

DLT Distributed-Ledger-Technologie.

E

ETH Ethereum.

EVM Ethereum Virtual Maschine.

EWF Energy Web Foundation.

G

GPU Graphics Processing Unit.

GUI Graphical User Interface.

I

IPC Inter Process Communication.

IPv4 Internet Protokoll Version 4.

M

M2M Machine to machine.

N

NAT Network Adress Translation.

NB Netzbetreiber.

P

P2P Peer-to-Peer.

PoA Proof of Authority.

PoS Proof of Stake.

PoW Proof of Work.

PV Photovoltaik.

R

RAM Random Access Memory.

S

SC Smart Contract.

SLP Standardlastprofil.

SSH Secure Shell.

T

TRL Technology Readiness Level

U

URL Uniform Resource Locator.

USB Universal Serial Bus.

V

V2G Vehicle to Grid.

V2H Vehicle to Home.

VK Virtuelles Kraftwerk

VKNA Virtueles Kraftwerk Neckar-Alb

VM Virtuelle Maschine.

VNB Verteilnetzbetreiber.

W

WLAN Wireless LAN.

Vorwort

Das vorgestellte Projekt entstanden zu großen Teilen in den Pandemie-Jahren 2020 und 2021. An die Stelle von persönlichen Treffen traten Video-Konferenzen und spontane Besprechungen im Labor mussten entfallen. Unter diesen erschwerten Bedingungen ist es der Disziplin und Motivation der Mitarbeiter und Studierenden zu verdanken, dass der Zeitplan weitgehend eingehalten werden konnte und fundierte Ergebnisse entstanden. Die Professoren der Hochschule Reutlingen bedanken sich herzlich bei den im Projekt beteiligten Mitarbeitern, Herrn Vasyl Matyniuk und Herrn Slavi. Außerdem bedanken sie sich bei den Studierenden Herrn Marco Arena, Felix Schaal, Jim Rebholz, Alexander Migunov und Nils Hägele für die ausführlichen Beiträge, die im Rahmen von Studienarbeiten entstanden sind und die Projektergebnisse vertieft haben.

ebök Institut bedankt sich bei den Kollegen der Hochschule für die konstruktive und vertrauensvolle Zusammenarbeit sowie bei Herrn Philemon Ballbach für die reibungslose administrative Abwicklung des Projekts.

Zusammenfassung

Das Projekt demonstriert in einem Laboraufbau (TRL4) die Kommunikation autonomer Agenten mittels DLT-Technologie. Für die zwei Anwendungsfälle Stromhandel und Engpassmanagement optimieren die Netzwerk-Teilnehmer ihre Lastgänge und Erzeugung anhand von Preissignalen. Im gewählten Beispiel sind Haushalte mit PV-Anlagen und Elektrofahrzeugen sowie produzierende Betriebe dargestellt.

Das zentrale Instrument der Optimierung stellt die Nutzung von Flexibilität in der Produktionsplanung dar. Dadurch wird ein wesentlich größeres Potential erschlossen als bei einer Fernsteuerung via CLS-Schnittstelle, die auf unwichtige oder zufällig verfügbare Verbraucher zugreifen kann. Als weiterer großer Vorteil bleiben alle Daten in den Betrieben; nur Lastgänge und Preise werden ausgetauscht. Im gewählten Modell erhält jeder Teilnehmer via Smart Contract fast instantan eine Rückmeldung zu seinem angemeldeten Lastgang und kann damit entscheiden, ob sein geplantes Verhalten netzdienlich ist oder ein Iterationsschritt folgen sollte.

Mit dem beschriebenen Mechanismus optimiert der Schwarm der Teilnehmer gemeinsam das lokale Netz. Über Smart Contracts lässt sich dieser Prozess weitgehend automatisieren. Voraussetzung dafür ist, dass jeder Teilnehmer die interne Datenbank der Produktionsziele und Anlagenrestriktionen auf dem Laufenden hält.

Stadtwerke als Betreiber von Verteilnetzen können diesen Mechanismus zur Vermeidung von Netzengpässen als Alternative zu Investitionen in Betriebsmittel einsetzen. Unter den gegebenen Regularien und der verbreiteten Risiko-Aversität der Branche ist eine rasche Verbreitung in den nächsten Jahren allerdings nicht zu erwarten.


Virtuelles Kraftwerk
der zweiten Generation

1 Einführung

Das vorliegende Projekt befasst sich mit der Konzeption einer neuen Elektrizitätsversorgung, welche über die Vernetzung autonomer Agenten in einer Industrie 4.0 Umgebung mithilfe von Distributed Ledger Technologien (DLT) zu einem Virtuellen Kraftwerke gebündelt wird.

1.1 Projektziele

Die zentrale Fragestellung bezieht sich auf den Einsatz von Industrie 4.0 und DLT um eine neuartige Stromversorgung zu konzipieren. Dabei wurden folgende Schwerpunkte untersucht:

1. Welche Lastflexibilität findet sich in einer I4.0-Umgebung, verglichen mit einer konventionellen Produktion?
2. Wie lassen sich Echtzeitfähigkeit und ein günstiges Skalenverhalten mit hoher Sicherheit und einem netzdienlichen Schwarmverhalten kombinieren?
3. Wie kann man sinnvoll Distributed Ledger Technologien in Verteilnetze integrieren und welcher Mehrwert entsteht dabei?

Ziel des Projekts ist es, die Grundlagen für eine Kooperation unabhängiger Agenten nach dem Vorbild einer industriellen Symbiose zu erarbeiten und die notwendigen Werkzeuge für den Betrieb eines so strukturierten Virtuellen Kraftwerks zu entwickeln.

1.2 Struktur des Projekts

Das Projekt wurde in Zusammenarbeit zwischen dem ebök Institut GmbH, den Fachbereich Werkzeugmaschinen und Fertigungssysteme und der Forschungsgruppe DLT-Lab bearbeitet.

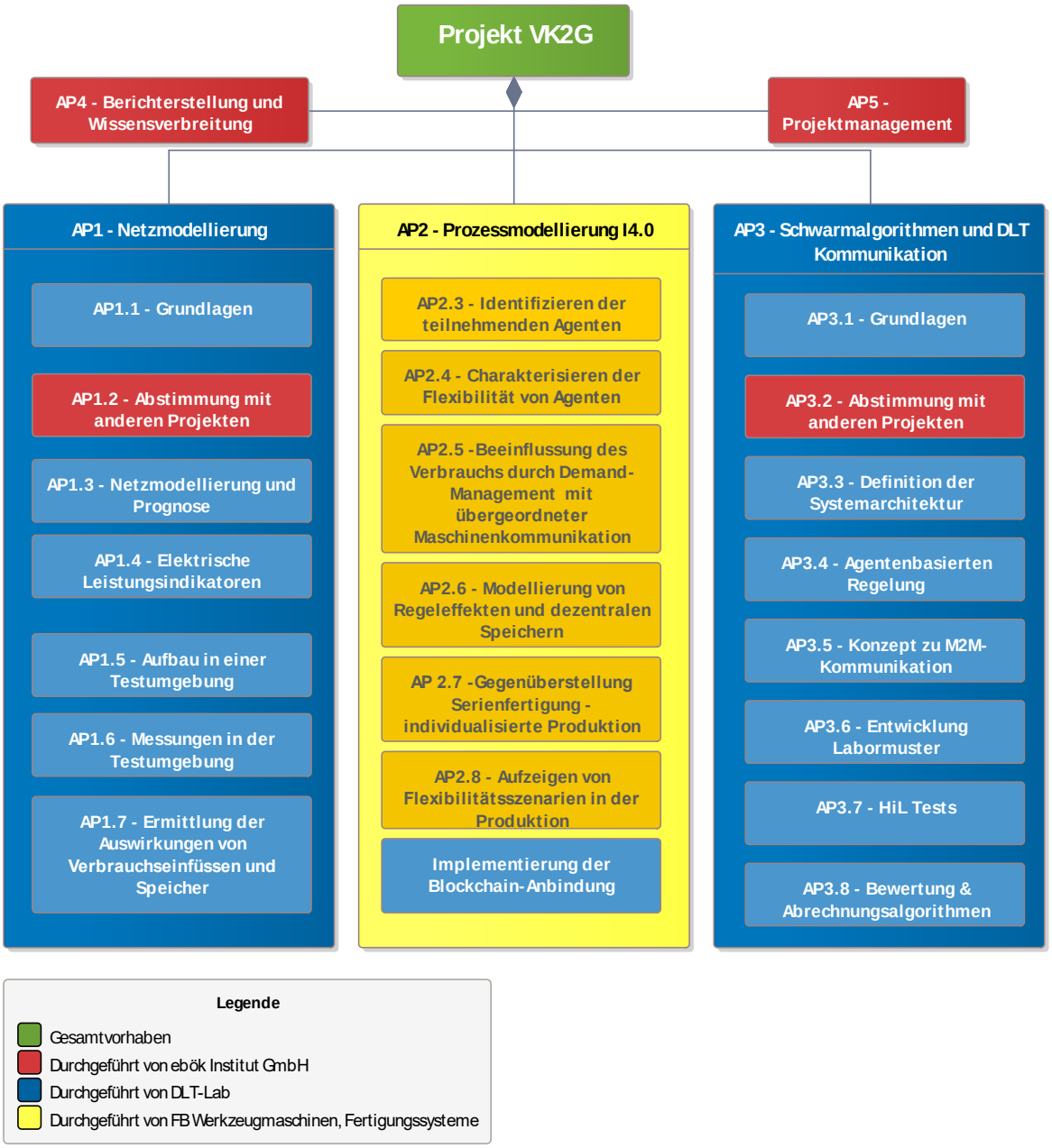


Abbildung 1.1: Strukturierung des Projekts. Die Zuordnung der Partner zu den unterschiedlichen Arbeiten ist farblich markiert

Dabei trugen neben den angestellten Mitarbeitern auch eine Reihe von studentischen Arbeiten zu den Ergebnissen bei.

Das Projekt ist in fünf Arbeitspakete gegliedert. Dabei sind drei Arbeitspakete für die Beantwortung der Leitfragen maßgeblich beteiligt. Eine Übersicht mit den wesentlichen Verknüpfungen und Interaktionen ist im Abbildung 1.1 dargestellt.

Der logische Aufbau des Berichts erfordert die Vorstellung der Arbeitspakete in eine andere Reihenfolge als den Projektstrukturplan vorsieht.

1.3 Gesamtkonzept des Vorhabens

Mit dem wachsenden Anteil von erneuerbaren Energien wie zum Beispiel Photovoltaik (PV), Windund Wasserkraft sowie dem Ausbau der Elektromobilität entstehen einige Herausforderungen für die Stromnetze der Zukunft. Zum einen wird die Energieerzeugung dezentraler und zum anderen sind Verbrauch sowie Erzeugung des elektrischen Stroms schwieriger zu synchronisieren. Dies erfordert neue Konzepte zur Regelung von Verbrauchern und Erzeugern. Ein Ansatz ist, eine Kommunikation der verschiedenen Erzeuger und Verbraucher im Netz über Blockchain herzustellen. Mit den dadurch zur Verfügung stehenden Informationen kann dann eine dezentrale Regelung umgesetzt werden. Außerdem ermöglicht die Blockchaintechnologie Abrechnungsvorgänge sicher durchzuführen.

In diesem Projekt soll anhand verschiedener Modelle von Verbrauchern und Erzeugern ein solches Konzept veranschaulicht werden. Ziel ist in einem lokalen Stromnetz autonom unter den Teilnehmern sowohl die lokale Optimierung als auch die Abrechnung durchzuführen.

1.4 Einführung in die Distributed-Ledger-Technologien

Unter Distributed Ledger Technology (DLT) versteht man eine Serie von IT-Konzepten, die auf eine verteilte, transaktions-basierte Datenvorhaltung basieren. Durch der Architektur und algorithmischen Vorrichtungen ist die Manipulation der Datenvorhaltung sehr schwer machbar. Aus diesem Grund gelten DLT-basierte Applikationen als besonders sicher gegen Angriffe von außen.

Mithilfe der DLT kann ein Informationssystem beschrieben werden, das als dezentrale Datenbank fungiert. Ins Deutsche übersetzt ergibt sich auch der Begriff „Verteiltes Kontenbuch", der die dezentrale Funktion der Technologie verdeutlicht. Abbildung 1.2 zeigt eine Gegenüberstellung zwischen einem sogenannten Centralized Ledger und dem Distributed Ledger. Im Vergleich zu einem Centralized Ledger ist hier keine dominante und zentrale Validierungsinstanz, die wie innerhalb der gezeigten Abbildung die Informationen speichert und verwaltet. Die Konsensbildung erfolgt stattdessen über ein demokratisches Prinzip mithilfe verschiedener Konsens-Algorithmen [SL19].

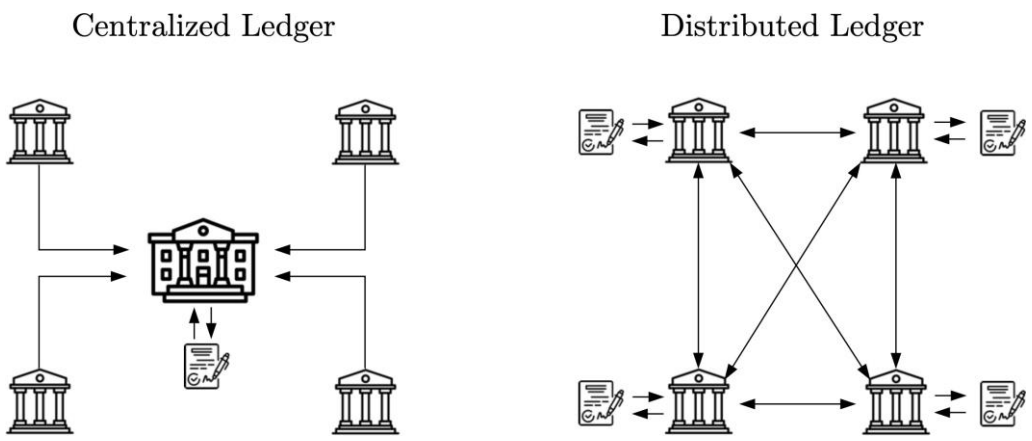


Abbildung 1.2: Vergleich Centralized Ledger (l.) und Distributed Ledger (r.)

Den Kern der Technologie stellen die Transaktionen dar, die in einem Hauptbuch (Ledger) auf einer Vielzahl unabhängiger Rechner gespeichert werden. Die innerhalb des dezentralen Netzwerks ansässigen Rechner werden auch als Knoten bzw. Nodes bezeichnet. Jeder Knoten erhält somit vollen Zugriff auf alle bereits getätigten Transaktionen. Soll

eine neue Transaktion durchgeführt werden, muss dies von den Knoten des Netzwerks zunächst verifiziert werden. Auch die Reihenfolge in der die Transaktionen erfasst werden, wird mithilfe des Konsens-Mechanismus verwaltet. Ein weiterer wichtiger Bestandteil der Distributed-Ledger-Technologie stellen die sogenannten Hash-Verfahren dar. Durch das Verfahren wird sichergestellt, dass die Person hinter einer Transaktion eindeutig nachgewiesen und während der Übertragung nicht verändert werden kann, wodurch der Schutz gegenüber dem Zugriff unberechtigter Personen gewährleistet wird [SL19].

Der Einsatz von DLT ist jedoch nicht für alle IT-Applikationen sinnvoll, daher stellt sich oft die Frage: Wann ist der Einsatz von DLT besonders gefragt? Grundsätzlich sollten folgende Kriterien erfüllt sein:

- Es besteht eine Geschäftsbeziehung mit mehr als zwei Parteien, z.B. Peer-to-Peer (P2P) Anwendungsfälle
- Das Vertrauen zwischen den Geschäftspartner ist nicht gegeben (weil sie sich z.B. vorab nicht kennen)
- Es werden Waren (in Form von Tokens) ausgetauscht

In der Elektrizitätswirtschaft werden bei vielen Anwendungsfällen die o.g. Kriterien erfüllt und somit ist ein Einsatz von DLT sinnvoll.

1.4.1 Blockchain

Eine der bekanntesten DLT ist die Blockchain-Technologie. Im Rahmen dieses Projekts wird das Ethereum Blockchain-Technologie eingesetzt. Eine ausführliche Einführung in die Blockchain-Technologie steht im Anhang A.2 zur Verfügung.

2

Anwendungsfälle und Konzept

In diesem Kapitel werden die grundlegenden Randbedingungen für die Entwicklung des Projekts gelegt. Als Referenz dienen die Anwendungsfälle und Struktur. Auf deren Basis wird eine Systemarchitektur erstellt, die anschließend für die Entwicklung der unterschiedlichen Arbeitspakete dient.

2.1 Akteure und Anwendungsfälle

2.1.1 Akteure

In den betrachteten Szenarien wurden sechs Akteuren identifiziert, welche im Abbildung 2.1 dargestellt sind. Die Beschreibung der einzelnen Akteure ist wie folgt:

Netzbetreiber (NB): Der Netzbetreiber ist verantwortlich für die Durchleitung und Verteilung von Gas bzw. Elektrizität sowie für den Betrieb, die Wartung und den Ausbau seines Netzes. Der Netzbetreiber ist zuständig für die Netzsicherheit. [BDE21]

Verteilnetzbetreiber (VNB) ist ein abgeleiteten Akteur, welche die Rechte und Pflichten vom NB erbt und dies für die Vierteilnetze umsetzt.

Anschlussnutzer (AN): Der Anschlussnutzer ist derjenige, der einen Netzanschluss einsetzt. Er ist die generelle Definition für alle Akteure, die von einem Netzanschluss Gebrauch machen.

Industrie Unternehmer: Bezeichnet die Unternehmen, welche an dre Optimierung teilnimmt, indem er seinen geplanten und tatsächlichen Lastgang kommuniziert. Er erbt die Eigenschaften eines AN.

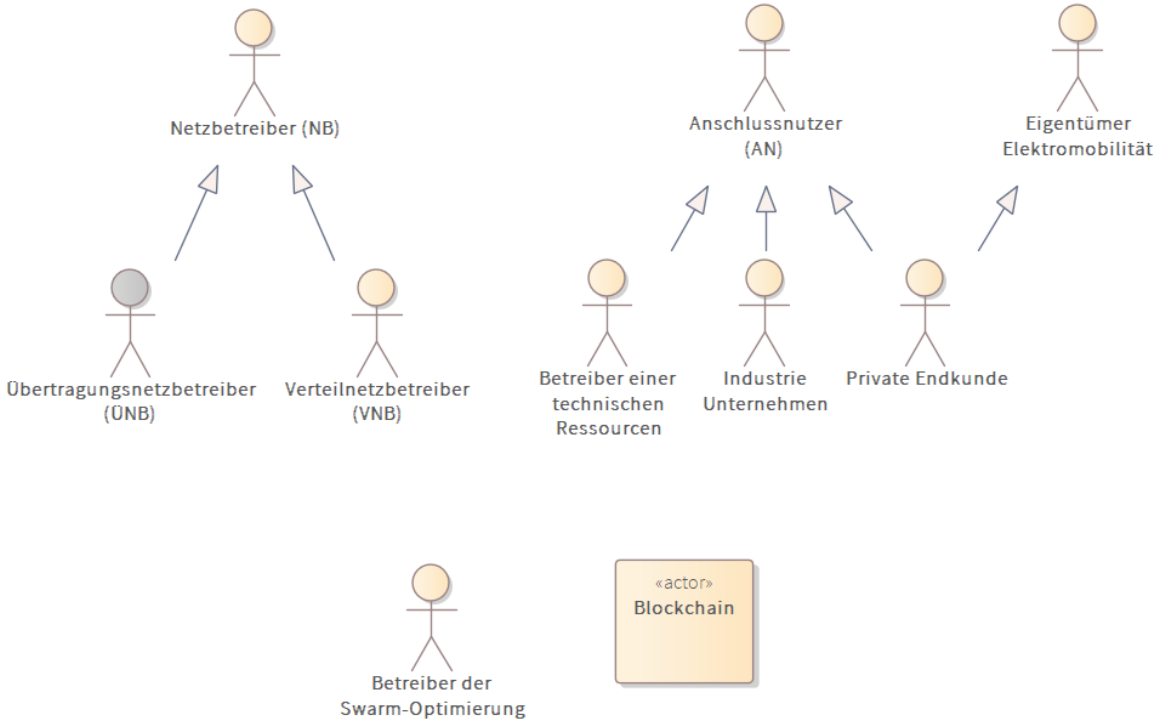


Abbildung 2.1: Akteure des Projekts

Privater Endkunde: Bezeichnet einen Haushalt, welcher neben den normalen Haushaltslasten auch PV und eventuell Elektromobilität einsetzen kann. Auch er erbt die Eigenschaften eines AN.

Betreiber einer technischen Ressource (BTR): Der Betreiber einer technischen Ressource ist verantwortlich für den Einbau, den Betrieb und die Wartung von technischen Ressourcen, in diesem Fall einer PV-Anlage.

Betreiber der Schwarm-Optimierung: Im Rahmen des Projekts wurde ersichtlich, dass nicht alle Funktionen sich als verteilte Applikationen realisieren lassen. Diese Funktionen werden durch einen Akteur abgewickelt, die ebenfalls als Teil des Ökosystems ist und die Schwarm-Optimierung durchführt. Er sendet Preissignale, auf die AN autonom (z.B. nach Optimierung ihres Lastgangs) reagieren.

Blockchain: **Technisches** System, welche alle Funktionen (inkl. Smart-Verträge) eines DLT beinhaltet.

Grundlage für die Benennung der Akteure ist das aktuelle Rollenmodell für die Marktkommunikation im deutschen Energiemarkt 2.0 [BDE21].

2.1.2 Anwendungsfälle

Abbildung 2.2 stellt die analysierten Anwendungsfälle dar. Der grundlegende Anwendungsfall ist die lokale Optimierung von Ressourcen. Die einzelnen Anwendungsfälle sind in den folgenden Abschnitten beschrieben.

Hauptziel ist die verteilte Optimierung der Lastgänge durch jeden Teilnehmer (Anschlussnutzer) auf Basis eines gemeinsamen Ziels mit vollster Transparenz im Prozess aber ohne Austausch von vertraulichen oder firmeninternen Informationen.

Die Anwendungsfälle (functional requirements) sind eng miteinander verzahnt. Daher sind neben der Beschreibung der Anwendungsfälle auch die unterschiedlichen Interaktionen dargestellt. Diese sind in Abbildung 2.3 dargestellt.

2.1.3 Netzgrenzwerte ermitteln und Restriktionen bekannt geben

Die Motivation für eine lokale Optimierung entspringt in der Regel aus knappen Ressourcen. Diese können in Form eines Engpasses im übergeordneten Verteilnetz vorhanden sein. Aus verschiedenen Initiativen, wie z.B. das Projekt Intelligente Wärme München [HLRS20], S. 158 besteht das Konzept, eine Art Leistungssollwert vom Netzbetreiber an einen bestimmten Netzanschlusspunkt zu vermitteln, welcher die Anlage selbstständig einhält. Dieses Projekt setzt auf die Analyse der geplanten und erwarteten Lastgänge aller AN mittels eines Netzmodells, das erwartete Restriktionen vorhersagen kann.

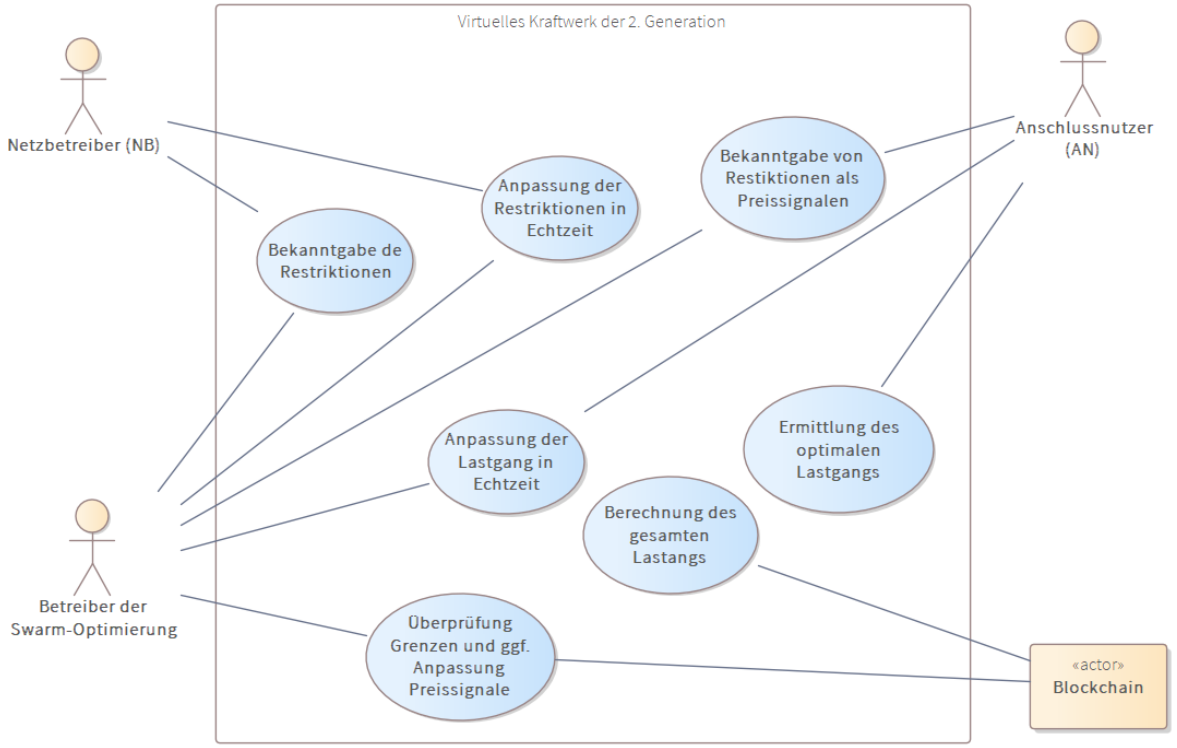


Abbildung 2.2: Hauptanwendungsfälle des Projekts

Diese Netzgrenzen werden für die jeweiligen Netzabschnitte über die Blockchain bekannt gegeben (publiziert). Die Restriktionen/Grenzen werden wie folgt bekannt gegeben:

- Maximale Bezug/Einspeiseleistung
- Maximaler Blindleistungsbezug/Einspeisung
- Maximaler/Minimaler Spannung im Umspannwerk (Zukunft)

2.1.4 Bekanntgabe von Restriktionen als Preissignale

Nach Bekanntgabe der Restriktionen kann der Betreiber der Schwarm-Optimierung die Restriktionen als Offset von Marktenergiepreise in den zur Optimierung notwendige Preissignale umsetzen. Diese werden in den Blockchain für einen bestimmten Netzabschnitt ebenfalls publiziert, damit die teilnehmenden Anschlussnutzer die Optimierung vornehmen können. Zur Kommunikation von Engpässen wird heute verbreitet ein Ampel-Modell eingesetzt. Die Übersetzung in ein Preissignal zeitigt jedoch mehrere Vorteile:

- Eine feinere Granularität gegenüber drei festen Stufen

- Ein methodischer Anschluss an den Stromhandel mit zeitvariablen Tarifen
- Die Quantifizierung der Vergütung für ein netzdienliches Verhalten der AN

Beim Anwendungsfall „Stromhandel" muss der „Engpass-Manager" stets mitlaufen. Dadurch wird verhindert, dass die Betriebsmittel überlastet werden, weil alle AN zu Zeiten der tiefsten Preise die höchsten Lasten anmelden.

2.1.5 Ermittlung des optimalen Lastgangs

In diesem Anwendungsfall wird die Flexibilität der einzelnen Verbraucher ermittelt. Grundlage sind die entsprechenden Preissignale. Die Ermittlung der Flexibilität in Produktionsanlagen wird im Abschnitt 4.2 ausführlich eingegangen.

Der optimierte geplanteLastgang wird im Anschluss in die Blockchain zurückgeschrieben.

Die Ermittlung des optimalen Lastgangs beinhaltet die Erzeugung von erneuerbare Energie. Für Anlagen, die nur ins Netz einspeisen und keine Speichermöglichkeit haben, ist der optimierte Lastgang eine negative Last, die aus der Prognose resultiert.

2.1.6 Berechnung des gesamten Lastgangs

Die Systemoptimierung und Führung wird im Rahmen dieses Projekts zum großen Teil in die Blockchain mittels sogenannten Smart Contracts (SCs) implementiert. In diesem Anwendungsfall wird der SC den konsolidierten Lastgang veröffentlichen.

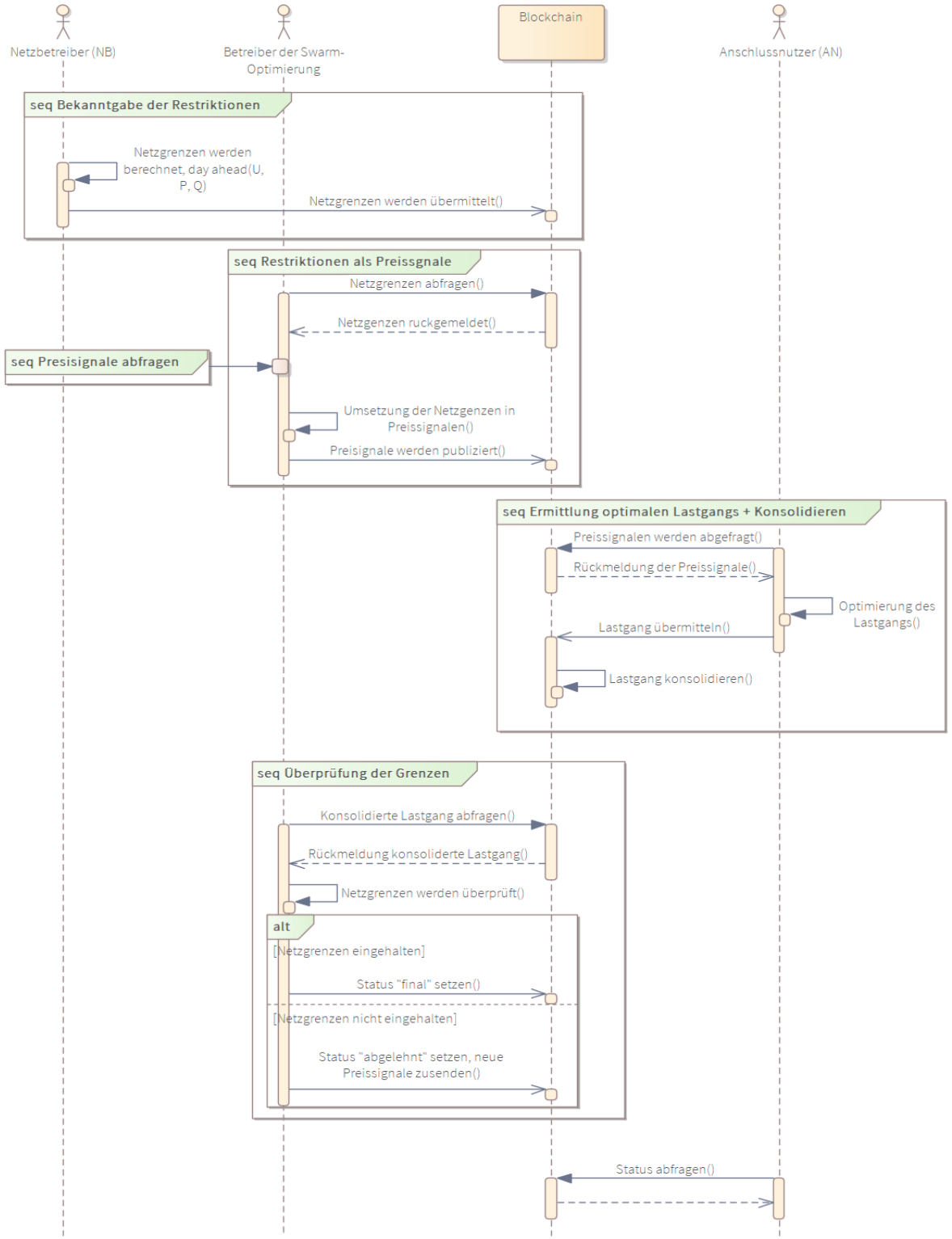


Abbildung 2.3: Interaktionen zwischen den verschiedenen Akteuren in einem Sequenzdiagramm. Im Normalfall werden keine Einschnitte vom Netzbetreiber vorgegeben.

2.1.7 Überprüfung der Grenzen und. ggf. Anpassung der Preissignale

Mit der konsolidierten Lastgang kann der Betreiber der Schwarm-Optimierung die vom Netzbetreiber publizierten Restriktionen überprüfen. Werden die Netzgrenzen eingehalten, wird der Fahrplan freigegeben.

Im Falle einer nicht Einhaltung kann der Betreiber der Schwarm-Optimierung die Preissignale anpassen und eine Optimierung anstoßen. Dieser iterative Prozess kann bis zu einer bestimmten Schlusszeit passieren. Sollte die Optimierung nicht konvergieren, greift eine Priorisierung, welche in Anhang C näher beschrieben wird.

2.1.8 Abrechnung (Vorschlag)

Nachdem alle AN einen Konsens über ihre geplanten Lastgänge gefunden haben, muss die Vertragserfüllung, die vom Netz gelieferten Strommengen, mittels Smart-Meter-Daten verifiziert werden; was sich ebenfalls in einem Smart Contract fassen lässt. Im Rahmen des Projekts wurden keine Abrechnungsmechanismen erarbeitet. Jedoch ein einfaches Konzept für die Abrechnung wird dargestellt. Durch den Einsatz von Blockchain lassen sich sehr kleine Abrechnungen implementieren. Diese sogenannten Micropayments sind durch die Technologie möglich.

Nachdem einen Ausgleichsvorgang abgeschlossen wurde, werden die im Smart-Contract verankerten Geldflüsse in Form von Ether ausbezahlt. [1]

[1] Im Rahmen der Bearbeitungskapazität des Projekts, wurden alternative Methoden zur Buchführung (z.B. durch den Austausch von Tokens) nicht berücksichtigt.

3

Systemarchitektur und DLT-Kommunikation

Dieses Kapitel beschreibt eine für das Projekt zugeschnittene Systemarchitektur, inklusive deren Implementierung in einem Demonstrator.

Die Anwendungsfälle wurden im vorherigen Kapitel beschrieben und dienen als Grundlage für die Erarbeitung der Systemarchitektur.

3.1 Strukturbeschreibung und Interaktionen

Die im Abschnitt 2.1 definierten Anwendungsfälle ergeben eine Übersicht der involvierten Akteure. Diese Akteure werden jedoch nicht selber handeln. Dafür kommen sogenannten Agenten zum Einsatz.

Agenten sind —im Rahmen des Projekts– technische Systeme die, die Rolle eines bestimmten Akteurs übernehmen und Handungen in deren Auftrag in der Blockchain ausführen.

Abbildung 3.1 stellt das Strukturdiagramm für die angestrebte Lösung vor. Dabei werden die Rollen der unterschiedlichen Akteure mittels Agenten implementiert. Diese Agenten haben Funktionen, die gezielt mit dem Blockchain/Smart Contracts interagieren. Die implementierten Funktionalitäten sind:

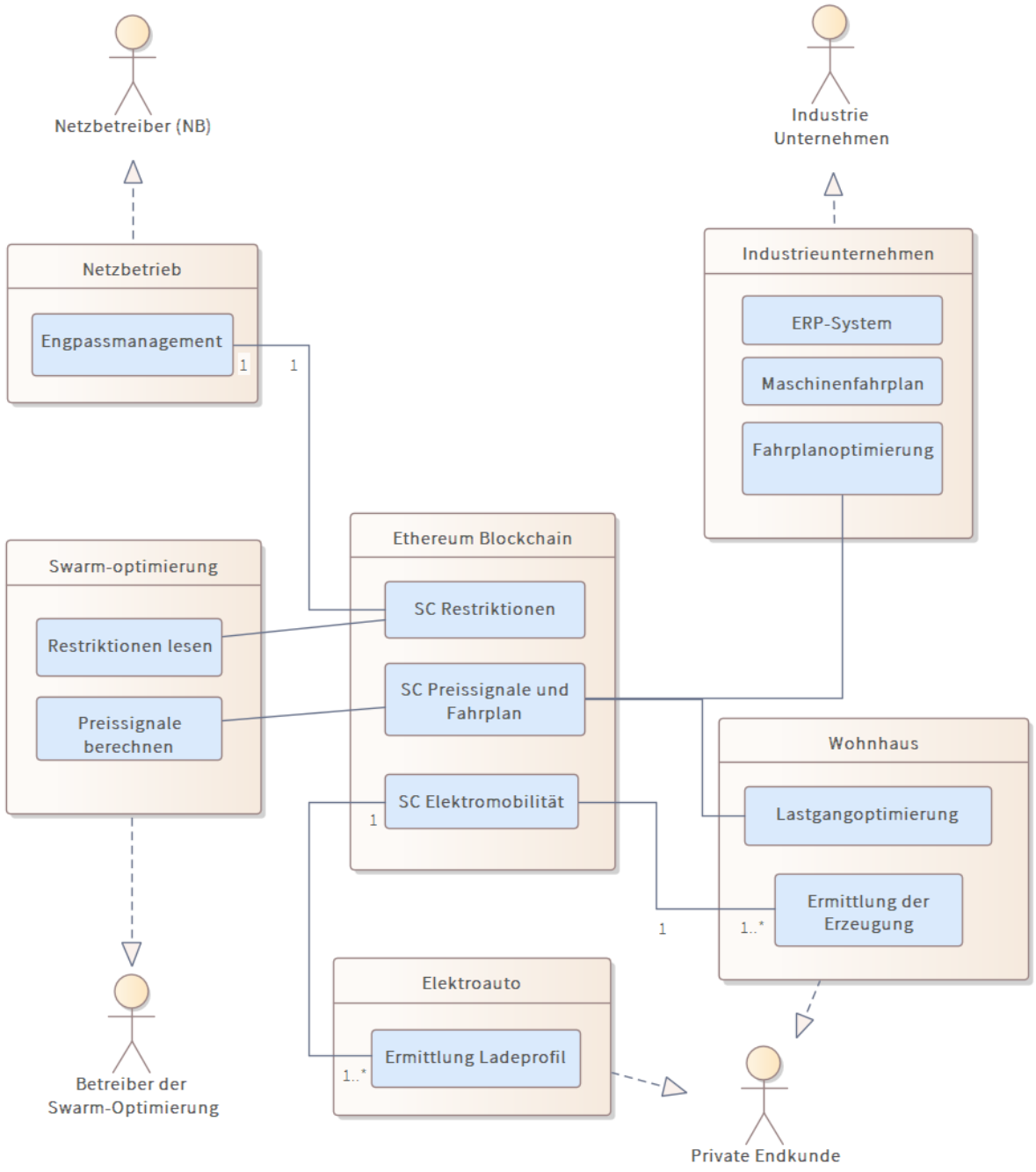


Abbildung 3.1: Komponenten-Struktur nach Funktionen

3.1.1 Engpassmanagement

Ermittelt die maximale Leistung, welche einen Netzknoten vertragen könnte (sowohl Last als auch Einspeisung) und gibt eine bestimmte Priorität an den Teilnehmer frei.

3.1.2 Fahrplanoptimierung

Aus dem Informationen des der Produktionsplanung und der notwendigen Betriebsmittel errechnet dieser Agent, einen optimalen Fahrplan und meldet diesen an den zugehörige SC.

3.1.3 Ermittlung der Erzeugung

Stellt die aktuelle (bzw. prognostizierte) Erzeugungskapazität fest und vermittelt es an die verschiedenen SC. Im Rahmen dieses Projekts ist die Solaranlage ein Teil des Wohnhauses und die Erzeugung wird nur für die Einspeisung (Erzeugung minus Eigenverbrauch) ermittelt.

3.1.4 Ermittlung Ladeprofil

Dieser Agent meldet die notwendige Energie und maximale Ladeleistung (Ladekurve) um seinen Fahrzeug laden zu können. Er bekommt von dem SC eine mögliche Ladeleistung zugewiesen.

3.2 Eingesetzte Systemarchitektur

In diesem Abschnitt wird die für dieses Projekt entwickelte Implementierung dargestellt. Eine ausführliche Beschreibung der Ergebnisse und entwickelte Plattformen stehen in den Anhängen zur Verfügung.

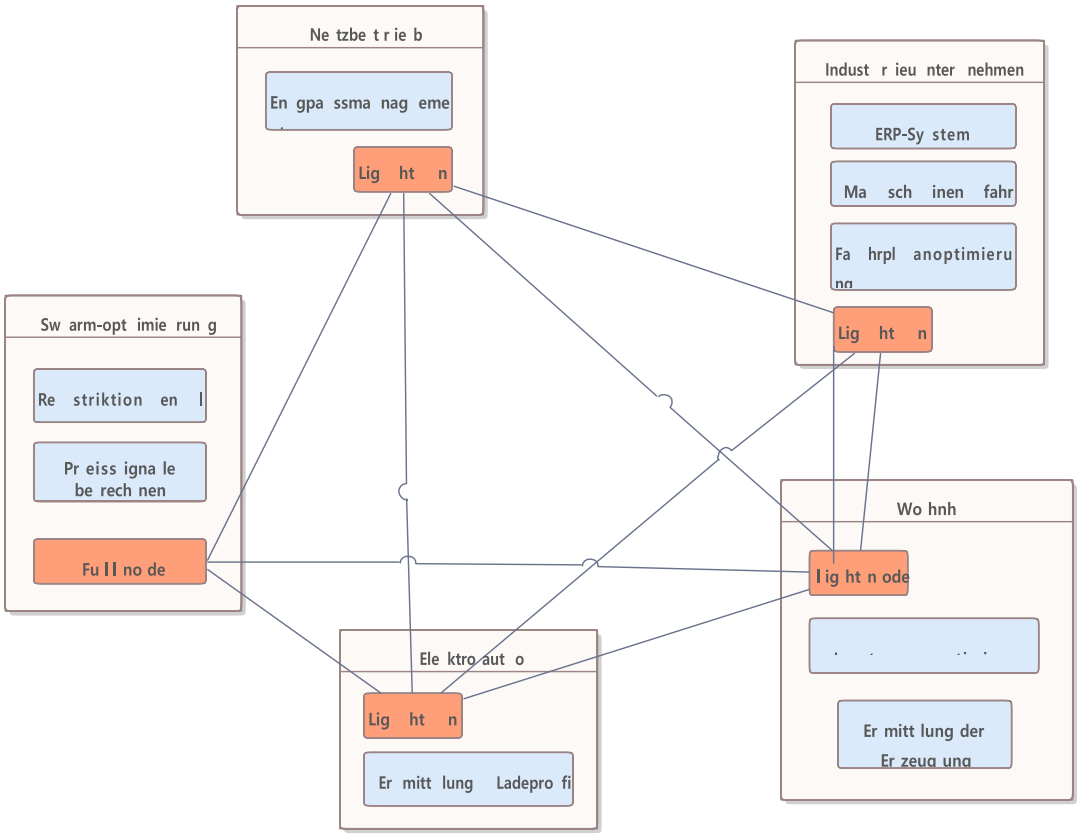


Abbildung 3.2: Übersicht des M2M Kommunikationskonzeptes

3.2.5 Konzept zur M2M-Kommunikation

Die in diesem Abschnitt eigesetzte Terminologie kann in Anhang A nachgelesen werden.

Die Machine to machine (M2M) Kommunikation zwischen den unterschiedlichen Agenten erfolgt 100% über eine Blockchain Kommunikation. So wird sichergestellt, dass Transaktionen transparent und nachvollziehbar bleiben.

Eine wesentliche Designfrage ist der Auswahl der Blockchain, die eingesetzt werden soll. Aufgrund des Informationsinhalts und Anforderungen an der Geschwindigkeit (nach Tests

wurde ermittelt, dass Zykluszeiten von 5–10 min optimal wären) wird in diesem Konzept eine private Blockchain bevorzugt. Näheres über privaten und öffentlichen Blockchains sind in Abschnitt A.2.4 beschrieben.

Abbildung 3.2 stellt das Konzept des M2M (Agenten) Kommunikation dar. In den vorgeschlagenen Konzept haben alle teilnehmende Agenten ein einfaches Blockchain Node (light node) mit dem sie Transaktionen zwar lesen und schreiben können, aber diese nicht selber verifizieren (block mining). Die Verifikation und block mining findet in einer der Full-nodes, welche der Betreiber der SchwarmOptimierung hat. Um mehr Transparenz und Vertrauen zu schaffen, können alle Akteure jederzeit einen Full-node implementieren, wobei dann die entsprechende Rechenleistung vorhanden sein muss.

Die Blockchain-Technolgie hat heutzutage ein negatives Image aufgrund des extremen Energiebedarfs der Bitcoin Blockchain. Dieser Energiebedarf ist auf der Consensus-Mechanismus: «Proof of Work» zurückzuführen. Heutzutage stehen viele Alternative zur Verfügung, welche in Anhang A.2.2 näher erläutert werden.

3.2.6 Entwicklung des Labormusters

Grundlage für alle Ergebnisse ist eine verteilte Infrastruktur. So erfolgten die Tests der Vorgänge mit einer realen Blockchain Technologie (in diesem Fall eine private Ethereum Blockchain). Die Simulation der Komponenten wurde jedoch zentral abgelegt. Die Verteilung der Software Komponenten auf den unterschiedlichen technischen Anlagen ist in Abbildung 3.3 dargestellt.

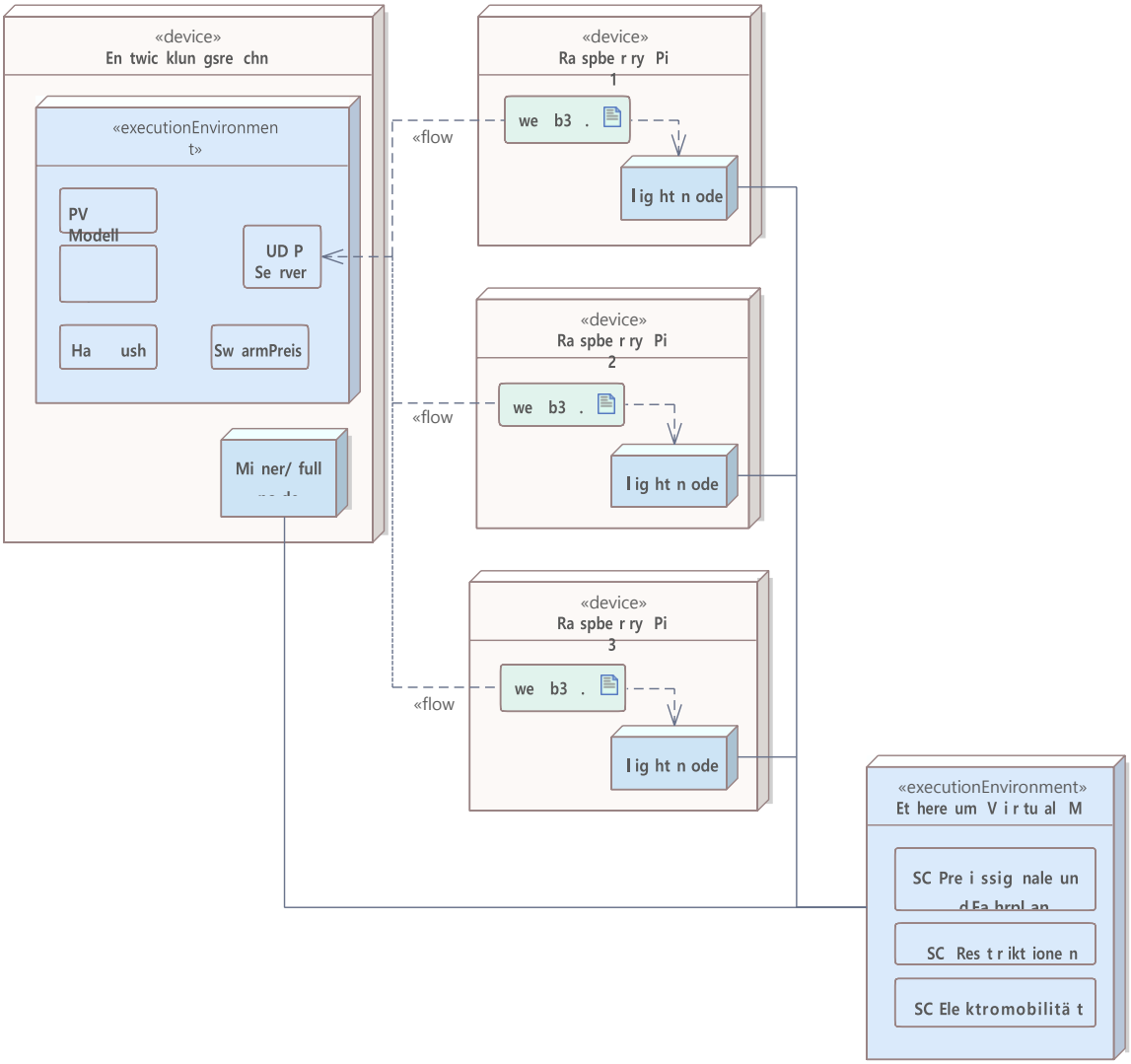


Abbildung 3.3: Deployment-Diagramm der implementierten Lösung. Um die verschiedenen Teilnehmern zu modellieren, wurden Rasperry Pis integriert.

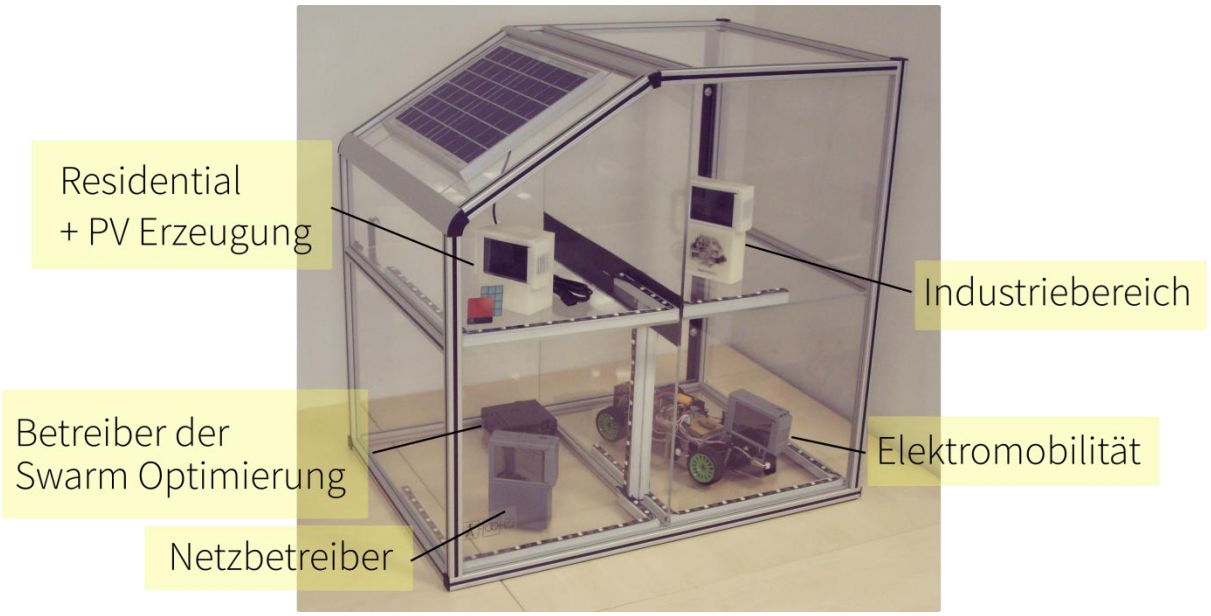


Abbildung 3.4: Der Versuchsaufbau im Labor für Regelungstechnik

Dieser Setup wurde physikalisch aufgebaut und als Showcase implementiert. Das Exponat steht im Labor für Regelungstechnik an der Hochschule Reutlingen für weitere Versuche zur Verfügung (siehe Abbildung 3.4).

3.3 Das Virtuelle Kraftwerk

Die beschriebenen Mechanismen erlauben ein koordiniertes netzdienliches Verhalten von Erzeugern und Verbrauchern ohne eine zentrale Instanz, die steuernd auf Anlagen zugreift. Alle Teilnehmer bleiben autonom und somit Herr ihrer Daten und Abläufe. Vergl. dazu auch [Kah20], im vorliegenden Fall entfiel auch noch die zentrale Kommunikationsplattform zugunsten einer dezentralen Datenhaltung.

Netzdienlichkeit resultiert allein aus dem Schwarmverhalten der Teilnehmer. Der gesamte Overhead reduziert sich auf den Schwarm-Manager, der eine zuverlässige Belastungsprognose der Betriebsmittel erstellen muss.

Die Innovation besteht beim gewählten Ansatz im Wegfall einer steuernden Zentrale und in der vollständig automatisierbaren Kommunikation der Teilnehmer.

Neben dem hier beschriebenen Fokus auf das lokale Verteilnetz kann der Mechanismus auch für den Handel mit Strommengen, z. B. am Spotmarkt, verwendet werden. Dabei müssen Vorkehrungen gegen sogenannte 50,2 Hz-Effekte getroffen werden. Wenn viele Abnehmer zum günstigsten Preis einkaufen wollen, können im Übertragungs- wie auch im Verteilnetz Überlastungen der Betriebsmittel auftreten.

4

Prozessmodellierung in einer I4.0 Umgebung

Die produzierenden Betriebe mit ihren flexiblen Lasten spielen eine zentrale Rolle im Virtuellen Kraftwerk, da sie

1. wesentlich größere Leistungen als Haushalte einbringen
2. Flexibilität aus Anlagen-Eigenschaften und Bearbeitungsaufgaben folgt
3. ihr Lastgang sich anhand der Produktionsplanung vorhersagen lässt

Am Beispiel der Bearbeitung von Metallen soll hier eine Modellbildung vorgestellt werden, die das Produktionsziel erreicht und anhand der übermittelten Preissignale einen Prduktionsplan mit optimiertem Lastgang generiert.

Bei der Nutzung der Flexibilität gibt es die Zielgrößen

- Reduzierung der Spitzenlast
- Reduzierung des Energiepreises

Diese beiden Zielgrößen können auch kombiniert werden. Dabei können unterschiedliche Gewichtungen erfolgen. Bei der Preisoptimierung werden die Preissignale des Strommarktes entweder durch Prognosen oder tatsächliche Preise verwendet. In Abbildung 4.1 ist der Verlauf des Stromverbrauchs und des Strompreises einer vollen Arbeitswoche dargestellt. An Werktagen (Montag bis Freitag) lassen sich morgens und abends jeweils zwei Spitzen bei den Preisen erkennen. Am Wochenende und Feiertagen treten teilweise Phasen extrem niedriger Preise auf. Zur Nutzung der Preissignale / Preisprognosen sind unterschiedliche

Randbedingungen zu berücksichtigen. Ein wesentlicher Bestandteil ist die Flexibilität der Maschinennutzung. Dabei sind die Arbeitszeiten und Schichtmodelle zusammen mit der Automatisierung und der Möglichkeit des autonomen Maschinenbetriebs zu berücksichtigen. Bei der Auftragsreihenfolge sind die Beherrschung der Technologie, die Gesamtdurchlaufzeit und der Liefertermin zu berücksichtigen. Sind automatisierte und nicht automatisierte Maschinen vorhanden, kann die Flexibilität die Verlagerung auf andere Maschinen teilweise mit anderen Technologieparametern erfordern. Weiterhin ist bei einer zeitlichen Verlagerung auch der Grundverbrauch, der unabhängig von der Bearbeitung auftritt, zu berücksichtigen. Zur Nutzung der Flexibilität ist die Prognose des Strompreises abhängig von der Laufzeit der Bearbeitungsprogramme pro Werkstück, die Anzahl der Werkstücke pro Los und der Betriebszeit erforderlich. Je mehr Parameter flexibel genutzt werden können, desto größer ist das Potenzial zur Reduzierung der Spitzenleistung und der Kosten. In einem konkreten Beispiel konnte bei zweischichtiger Nutzung ohne Automatisierung und ohne autarken Maschinenbetrieb eine Reduzierung der Spitzenleistung um 26 % und des Gesamtenergiebedarfs von 9 % ermittelt werden. Insbesondere die Verschiebung energieintensiver Operationen auf de günstigen Nachtstunden kann deutliche Kosteneinsparungen hervorrufen. Bei der Simulation wurden auch Fragen, wie die Nutzung von Eigenerzeugung und Speicherung berücksichtigt. Durch diese Maßnahmen lassen sich weitere Verbesserungen der Kosten und Spitzenleistung erzielen.

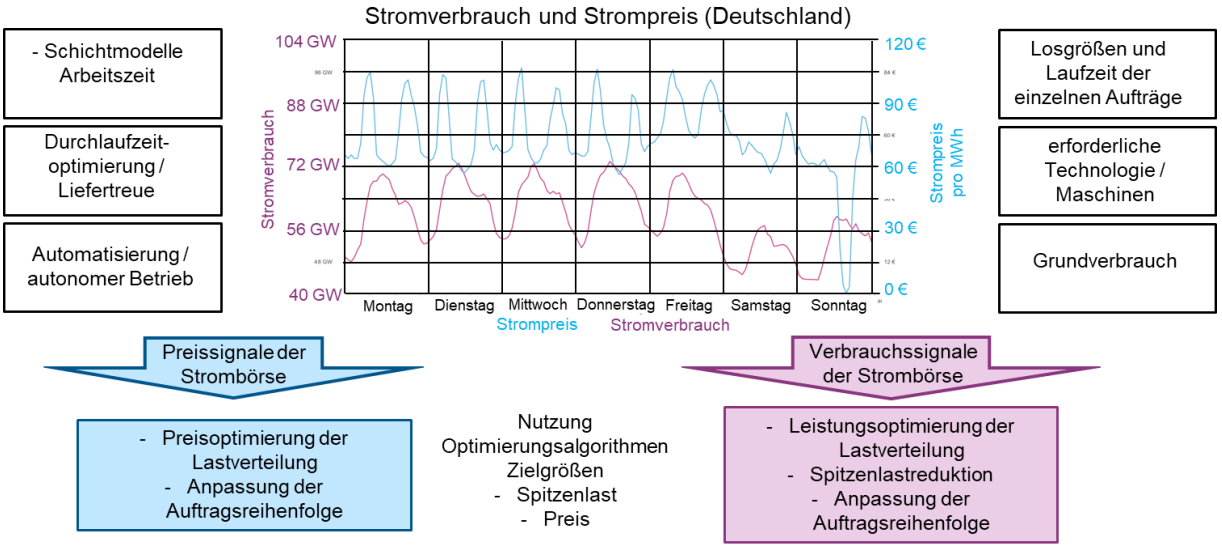


Abbildung 4.1: Leistungs und Stromverfügbarkeit mit Kenngrößen der Flexibilität

Auf der Basis von Vorgängerprojekten konnte in diesem Projekt auf Erkenntnisse bezüglich der Leistungscharakterisierung von Produktionsanlagen zurückgegriffen werden. Im Mittelpunkt der Analyse standen weitverbreitete Anlagen der metallverarbeitenden Industrie in der Prozesskette Urformen – Umformen – Trennen – Fügen – Beschichten – Stoffeigenschaften ändern, wie im Abbildung 4.2 dargestellt.

Bei den dort zusammengefassten Produktionsprozessen gibt es eine Vielzahl unterschiedlicher Verfahren, die sich nach Technologie und Anlagen sehr stark unterscheiden. Im Rahmen dieses Projekts wurden primär trennende Verfahren bezüglich der energetischen Nutzung analysiert. Während beim Urformen (z.B. Erschmelzen von Metallen) und Umformen (z. B. Erhitzen metallischer Bauteile) schon heute aufgrund der großen Anlagenleistungen energetische Optimierungen durchgeführt werden, ist dies beim Trennen häufig noch nicht der Fall. Für die Nutzung von Flexibilität zum Angleich des Verbrauchs an die verfügbare Leistung wurden unterschiedliche Ansätze verfolgt:

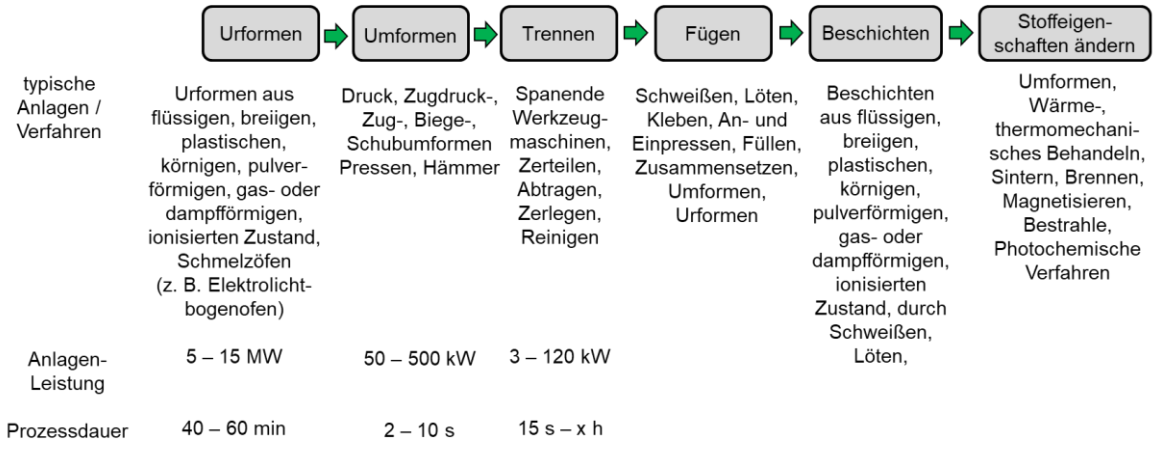


Abbildung 4.2: Prozesskette der metallverarbeitenden Industrie

- Charakterisierung der unterschiedlichen Prozesse bezüglich deren Leistungs-Zeit-Verlauf (Lastgang)
- Ermittlung netzdienlicher Prozesse

- Ermittlung der zeitlichen Flexibilität unterschiedlicher Prozesse
- Möglichkeiten der Prädiktion von Leistungsverläufen
- Integration von Speichern zur kurzzeitigen Verhinderung von Engpässen

4.1 Identifizieren der teilnehmenden Agenten

Bei der Identifikation der teilnehmenden Agenten wurden die unterschiedlichen Unternehmensstrukturen analysiert. Dabei wurden kleine Unternehmen mit flexibler Einzelteil- und Kleinserienfertigung auf universellen Maschinen ebenso wie größere Maschinenbauunternehmen mit geringerer und höherer Fertigungstiefe bis hin zur Serienfertigung bei großen Stückzahlen betrachtet. Als Branchen wurden Bauteile aus dem Maschinenbau, der Feinwerktechnik, der Investitionsgüterund der Automobilindustrie untersucht. Dabei wurden die maßgeblichen, an der Produktion beteiligten Anlagen bezüglich deren Leistungsund Energiebedarf ermittelt. Da die Unternehmen des Urund Umformens schon netzdienlichen Betrieb der Anlagen umsetzen, wurden im Rahmen dieses Projekts primär die spanenden Maschinen bezüglich der vorhandenen Flexibilität analysiert. Bei der Herstellung metallischer Komponenten auf spanenden Maschinen ist die Technologie (z.B. Schnittgeschwindigkeiten, verwendete Werkzeuge) sowie die zu bearbeitenden Bauteile durch die Fertigungsaufträge vorgegeben. Auch die Belegung der Maschinen richtet sich nach den vorhandenen Aufträgen sowie deren Lieferzeiten bzw. Verwendung in Folgeoperationen.

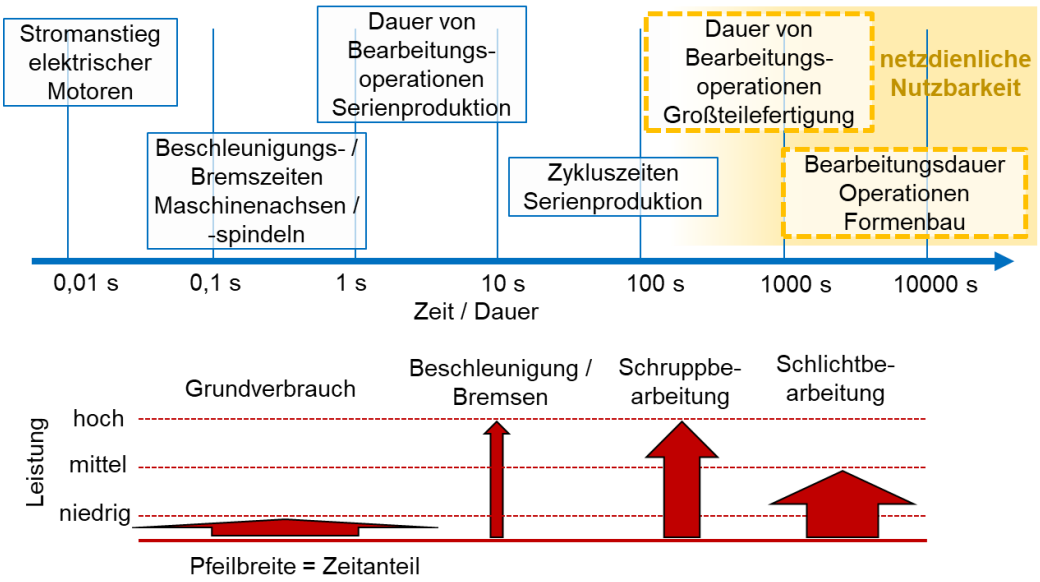


Abbildung 4.3: Energieinhalt und Leistung metallverarbeitender Maschinen und Operationen

4.2 Charakterisieren der Flexibilität von Agenten

Nach der Identifikation der unterschiedlichen Verfahren wurde die Möglichkeit zur Nutzung einzelner Prozesse und Maschinen für die netzdienliche Flexibilität analysiert. Dabei wurden die unterschiedlichen Zustände der Maschine während eines kompletten Bearbeitungszyklus analysiert. Wichtig für die Netzdienlichkeit ist die Reproduzierbarkeit, Leistung und Dauer der verschiedenen Zyklen. Diese Aspekte sind nicht bei allen Operationen gegeben. Die Spitzenleistungen beim Beschleunigen und Bremsen von Bewegungen liegen zeitlich deutlich unter einem sinnvoll nutzbaren Bereich. Auch die einzelne Operations und Zykluszeiten von Serienproduktionsanlagen bewegen sich im Bereich von einzelnen Sekunden bis max. wenige Minuten mit unterschiedlichen Leistungen. Insbesondere bei der Bearbeitung von größeren Bauteilen und bei der Bearbeitung von Einzelteilen mit größeren Anteilen von Leistungszerspanung gibt es nutzbare Potenziale, wie Abbildung 4.3 darstellt.

Im Rahmen dieses Projekts wurde mithilfe eines Simulationsprogrammes die Möglichkeit geschaffen, den grundverbrauchs-, bewegungsund prozessbedingten Energieund Leistungsbedarf von spanenden Produktionsanlagen für beliebige Prozesse zu

beschreiben. Dabei ist das Ergebnis der Leistungsbedarf einer Anlage über der Zeit. Die Simulation basiert auf der Nutzung von Modellen für die während der Bearbeitung auftretenden Zerspankräfte und der in den Antrieben für das Aufbringen der Kräfte erforderlichen Leistung. Die zeitliche Auflösung der Simulation liegt im Bereich von ms, so dass der die Beschleunigung induktiv bedingte begrenzende Stromanstieg in den Antrieben die untere Grenze darstellt (Abbildung 4.4).

Auf dieser Basis wurden die unterschiedlichen Bearbeitungsoperationen verschiedener spanender Bearbeitungsbeispiele simulativ abgebildet. Anhand dieser Last-Zeit-Verteilung wird die Netzdienlichkeit entsprechend dem folgenden Schema entschieden. Dabei wird immer vorausgesetzt, dass die einzelnen Verbraucher in der Maschine (z.B. Hauptund Nebenantriebe) schon optimiert wurden. Auch wird die steuerungstechnische Optimierung der unterschiedlichen Verfahren angenommen, die für einen sicheren und sparsamen Betrieb erforderlich sind (z.B. Nachlauf der Druckluft).

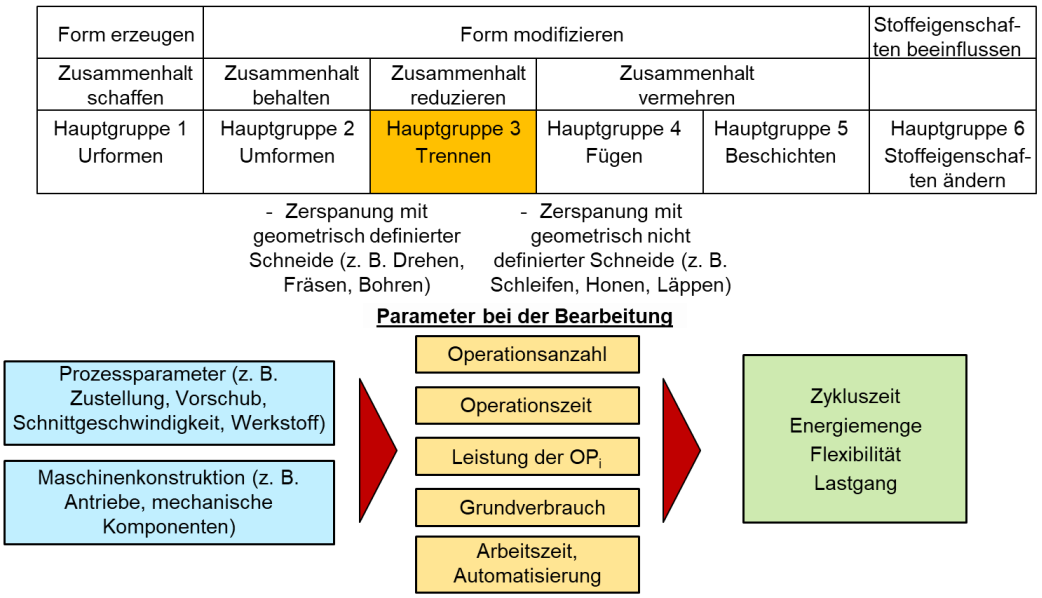


Abbildung 4.4: Energetisch relevante Kenngrößen in der Metallbearbeitung

Bei der Simulation der unterschiedlichen Maschinen und Prozesse wurde auf die exemplarisch an unterschiedlichen Produktionsanlagen (z.B. Bearbeitungsund Drehzentren

ermittelten Verbrauchskennwerte zurückgegriffen. Diese zeigen den Grund- und Leistungsverbrauch der unterschiedlichen Maschinen und wurden während typischer Bearbeitungsoperationen verifiziert.

4.3 Beeinflussung des Verbrauchs durch Demand-Management mit übergeordneter Maschinenkommunikation

Die Realisierung von Flexibilität ist nur dann möglich, wenn die einzelnen Verbraucher zeitlich und/oder bezüglich der Leistung flexibel sind. Da die Produktivität und Wirtschaftlichkeit der Produktionsanlagen von der Bearbeitungszeit der unterschiedlichen Bauteile und der damit umzulegenden Maschinenstundensätze zusammenhängen, ist eine Leistungsregelung der einzelnen Maschinen häufig nicht oder nur eingeschränkt möglich (Abbildung 4.5). Aus diesem Grund wurde bei der Optimierung die zeitliche Verteilung der Leistung unterschiedlicher Maschinen betrachtet. Im Folgenden sollen zwei Musterprozesse mit einem Potenzial der Flexibilität vorgestellt werden. Als Beispiele wird die Bearbeitung zweier unterschiedlicher Werkzeuge für das Spritzoder Druckgießen verwendet. Links oben sind die Bauteile schematisch im Schnitt dargestellt. Diese unterscheiden sich vor allem durch die Höhe der Formen. Dadurch resultiert ein unterschiedlich hohes Zerspanvolumen, welches die Bearbeitungszeit beim Schruppen beeinflusst, während die Zeit für das Schlichten aufgrund der nur wesentlich weniger unterschiedlichen Oberfläche nicht deutlich abnimmt. Bei dieser Bearbeitung gibt es einen nutzbaren Unterschied (Dauer) der Leistung zwischen dem Schruppen und Schlichten (Abbildung 4.6).

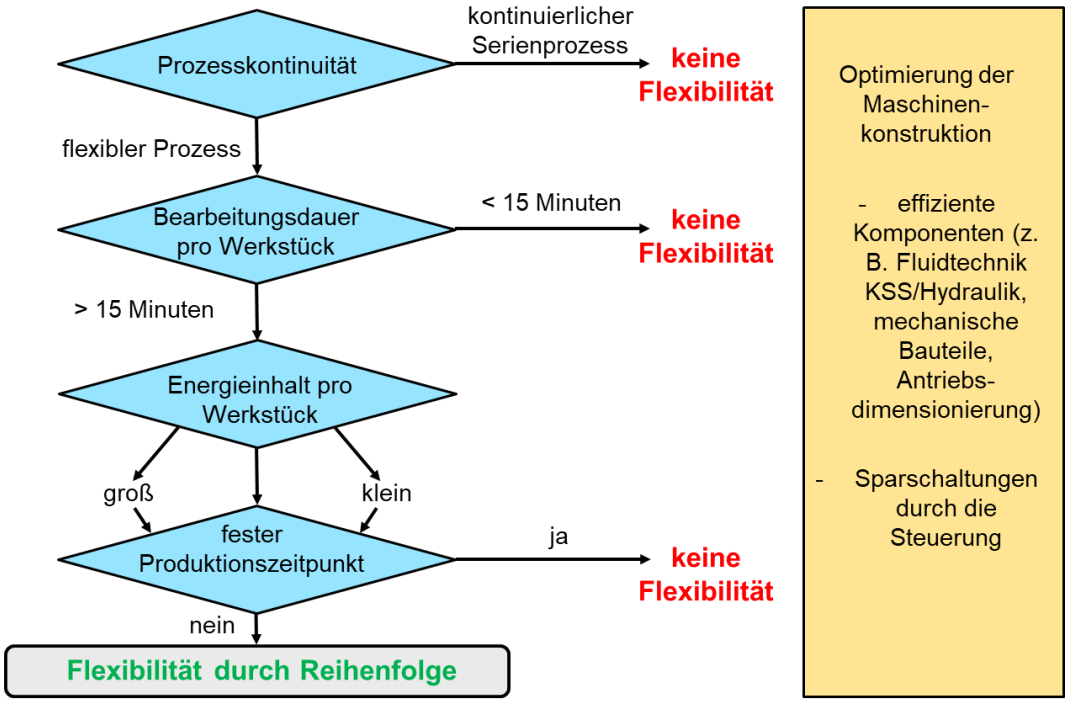
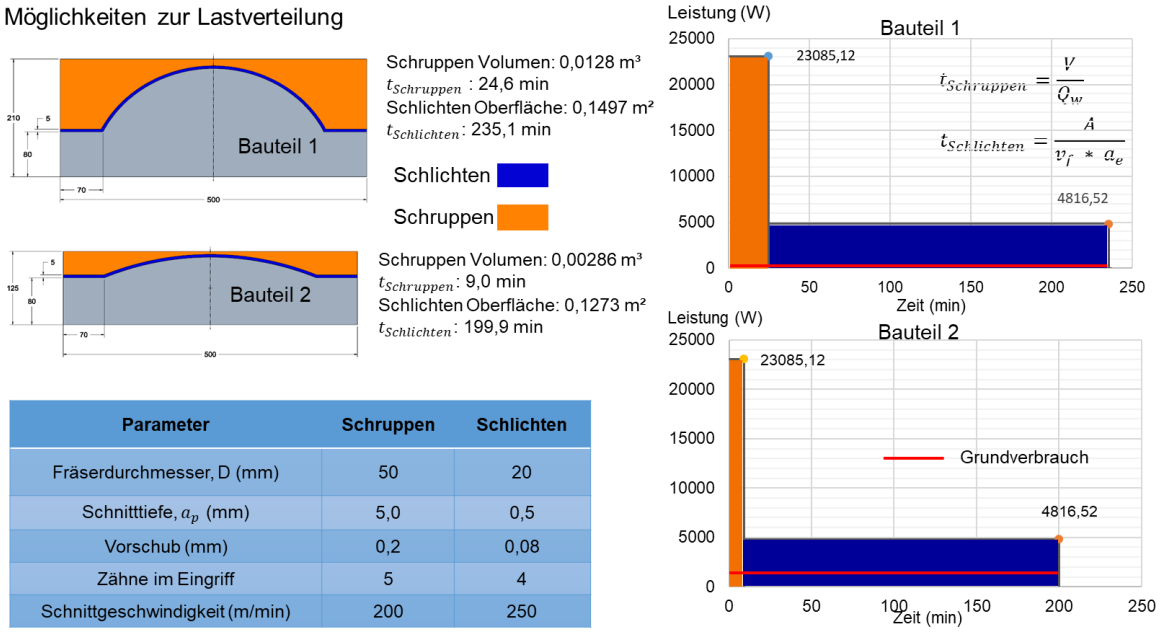


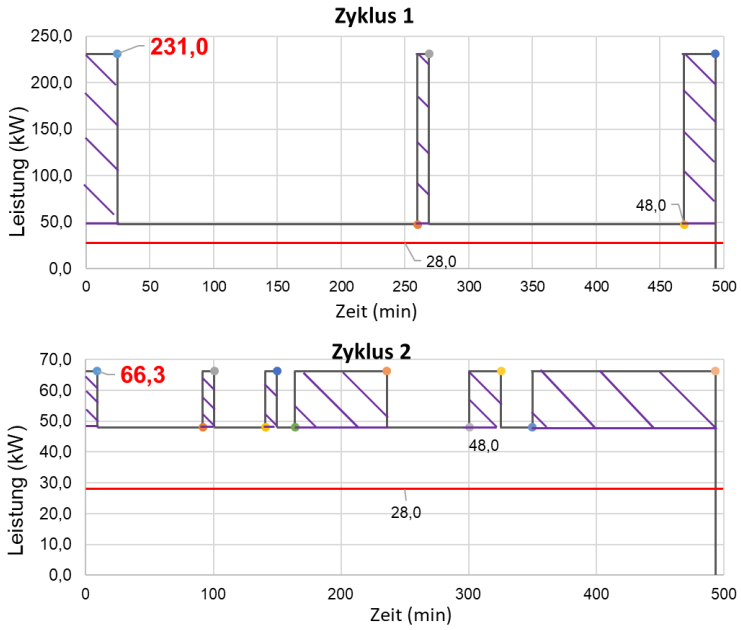
Abbildung 4.5: Nutzbarkeitsabfrage unterschiedlicher Flexibilitätskriterien



Parameter	Schruppen	Schlichten
Fräserdurchmesser, D (mm)	50	20
Schnitttiefe, a_p (mm)	5,0	0,5
Vorschub (mm)	0,2	0,08
Zähne im Eingriff	5	4
Schnittgeschwindigkeit (m/min)	200	250

Abbildung 4.6: Netzdienlich nutzbare Bearbeitungszyklen (Leistungs-Zeit-Verlauf)

Bei der Optimierung wurden 10 gleiche Maschinen mit diesen Werkstücken analysiert, die parallel arbeiten. Durch die zeitliche Steuerung der Aufträge auf den Maschinen konnte eine maximale Glättung bezüglich der Spitzenleistung (Optimierungskriterium Spitzenleistung) von 231 kW auf 66,3 kW erzielt werden (Abbildung 4.7). Dabei beträgt der Grundverbrauch der Maschinen 48 kW. Der Grundverbrauch ist leistungsunabhängig immer vorhanden. Ebenso ist bei einer derartigen Verteilung auch die erforderliche Gesamtenergie identisch, da alle Maschinen kontinuierlich laufen.



Randbedingung

- Die Bearbeitung kann ohne Störung durchlaufen
- Es ergibt sich bei beide Zyklen 10 gleiche Bearbeitungsmaschinen
- Nur Schnittleistung und Grundbedarf der Maschine werden betrachtet

Ergebnis

- minimale Leistungsmenge 48 kW
- Variable Energiemenge liegt zwischen 276 bis 639 MJ
- Flexibilität durch Steuerung von Produktionsabläufe
- Gesamtenergieverbrauch am Ende sind gleich wegen der gleichen Stückzahl von fertig bearbeiteten Bauteile

Abbildung 4.7: Spitzenleistungskappung und Zeitverteilung durch Flexibilitätsnutzung bei spanenden Maschinen

4.4 Modellierung von Regeleffekten und dezentralen Speichern

Nach der Optimierung der Spitzenleistung mit einfachen Prozessen wurden Optimierungsaufgaben mit unterschiedlichen Prozessen, feiner Verteilung und mehreren Optimierungskriterien verfolgt. Dabei war zunächst die Wahl eines geeigneten

Optimierungsalgorithmus zur bestmöglichen Gestaltung einer effizienten Berechnung der Lastgänge erforderlich. Als Optimierungsalgorithmen wurden

- Genetic Algorithm (GA)
- Particle Swarm Optimization (PSO)
- Tabu Search (TS)
- Simulated Annealing (SA)
- Ant Colony Optimization (ACO)

gegenübergestellt (Tabelle 4.1). Bei der Gegenüberstellung wurden die Algorithmen nach unterschiedlichen Eignungskriterien analysiert. Kriterien bei der Auswahl war die Eignung für die

- Vorgabe einer Preiskurve im Tagesgang mit dem Ziel minimaler Kosten
- Vorgabe einer nutzbaren Leistungskurve entsprechend des jeweils geltenden Strompreises
- Vorgabe einer maximal nutzbaren Anschlussleistung
- Vorgabe der maximal nutzbaren zeitlichen Flexibilität bezüglich der Fertigstellungszeit eines Auftrages.

als Ziele der Optimierung. Diese einzelnen Ziele wurden für die Optimierung auch in Kombination mit unterschiedlicher Gewichtung verwendet. Bei den Algorithmen ergibt sich die im Tabelle 4.1 dargestellte Gegenüberstellung.

Bei der Optimierung wurden ebenso unterschiedliche Flexibilitätsmodelle in Abhängigkeit der Arbeitszeitmodelle und der Fähigkeit des autarken Betriebes der Maschinen bei Ein-, Zwei- und Dreischichtbetrieb mit automatisierter und chaotischer Fertigung eingebunden.

Die Anwendung der Algorithmen wurden im Laufe des Projekts durch die Integration unterschiedlicher Maschinen, Prozesse und Bauteile mit verschiedenen Stückzahlen sukzessive erweitert.

Daraus resultiert eine Möglichkeit zur einer preisund leistungsorientierten optimierten Prozess- und Auftragsabfolge sowie Maschinenbelegung.

Tabelle 4.1: Gegenüberstellung unterschiedlicher Optimierungsalgorithmen

Algorithm	Genetic Algorithm (GA)	Particle Swarm Optimization (PSO)	Tabu Search (TS)	Simulated Annealing (SA)	Ant Colony Optimization (ACO)
Pros	• Bigger variety of solutions to be visited • Saving only the best solutions from available • Big amount of papers and deep coverage • Relatively easy implementation and data representation	• Particles are in the constant pursuit of the best fitting candidate • The path of particles to best fitting can spot better solutions in local search • Relatively small dependency on pseudorandom values	• Uses the short time and long time memory concepts to avoid the repetition of solutions, that do not lead to improvement of costs • Aspire the use of unvisited solutions instead of creating the elite group	• Relatively easy implementation • Constant "near optimum" value of variables • Small dependency on the random values • Usually can be used as the hybrid with other algorithms (GA or TS)	• Covers a big field of possible permutations of variables • Positive feedback for finding the better solutions • Can receive the optimum solutions relatively fast on early iterations
Cons	• In between the iterations can occur the situation with full population fitness dropped • Pseudorandom behavior • The variance of the solution is dependent on the rules of crossover/mutation gene exchange	• Can stuck in local minimum • In case of JSSP, it's hard to represent the velocity and path to the fittest candidate • Hard to separate the variable value that leads to improvement of costs	• Requires the complex understanding and analyze of the moves and how they affect the path to optimum • Pseudorandom behavior • The bigger the scale of input data – the bigger data storage for tabu list required and can become less effective	• Potentially, random value can discard the better fitting solution • Works with one candidate at the time, so less perturbations of variable can be visited • Without additional conditions can stuck in local optimum on the way to global optimum	• Require a long CPU time • The bigger scale of input data – the longer time is required

Aufgrund der durchgeführten Tests erscheint uns der Genetic Algorithm am besten für den Einsatz im Feld geeignet. Das im Vorgänger-Projekt VK_Koop [Kah20] verwendete MILP-Verfahren konnte leider nicht verglichen werden, da in der verwendeten Simulationsumgebung dafür keine Bibliotheken vorlagen.

4.5 Gegenüberstellung Serienfertigung und individualisierte Produktion

Die Serienfertigung bietet aufgrund der starren Abfolge der einzelnen Operationen, der kurzen Zykluszeiten der produzierten Bauteile im Bereich von 10 Sekunden bis wenige Minuten und der kontinuierlichen Produktion keine Möglichkeit für die Nutzung von Flexibilität auf einzelnen Maschinen. Bei nicht vollständiger Auslastung der einzelnen Produktionsmittel kann jedoch die vorhandene Kapazitätsreserve genutzt werden. Bei der Einzelteil und Kleinserienfertigung besitzen die zu produzierenden Bauteile unterschiedliche Bearbeitungsdauer, erforderliche Durchlaufzeiten und Leistungsbedarfe.

Dabei kann, unabhängig von der Dauer und Leistung der einzelnen Operation, Flexibilität durch flexible Auftragsreihung genutzt werden. Nicht vollständige Nutzung von Produktionskapazitäten kann bei der Optimierung für die Einbindung von Pausenzeiten zwischen der Beendigung eines und des darauf folgenden Auftrages genutzt werden. Dabei ist jedoch zu bedenken, dass die Maschinen mit ihrem Grundverbrauch weiterhin in die Leistung eingehen. Beispiele für die Verteilung der Aufträge auf unterschiedliche Maschinen ist in Abbildung 4.8 dargestellt.

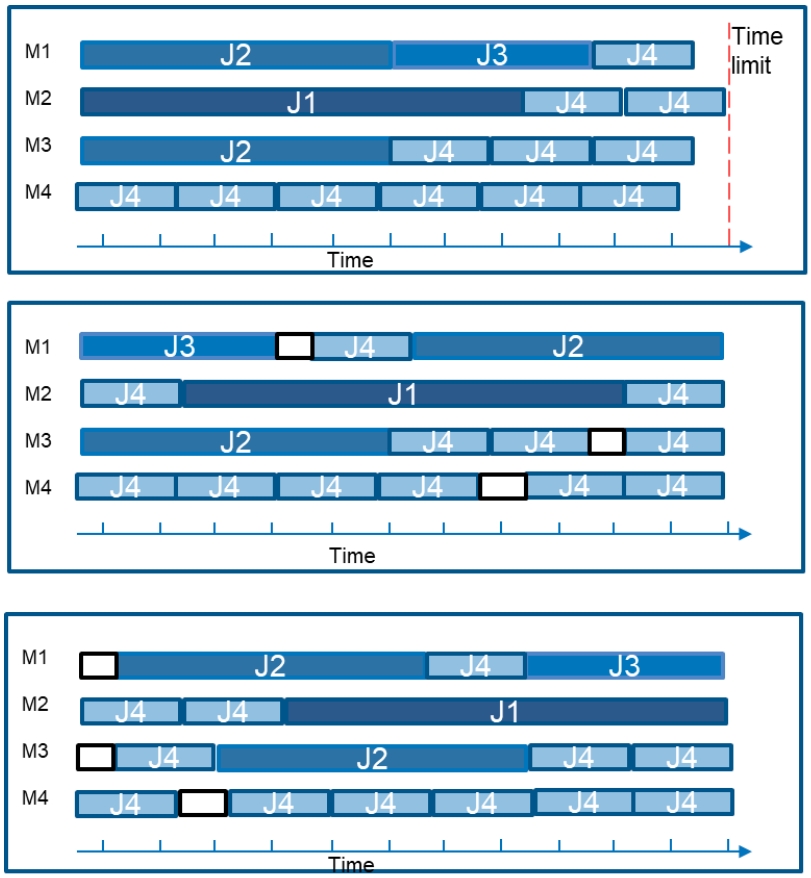


Abbildung 4.8: Verteilung der Aufträge auf unterschiedliche Maschinen

Unter Berücksichtigung der unterschiedlichen Optimierungskriterien wurde ein kleiner Musterbetrieb mit unterschiedlichen Maschinen und Aufträgen bezüglich der Vorgabe von minimalen Kosten und minimaler Spitzenleistung optimiert. Dabei wurden unterschiedliche Arbeitszeitmodelle und die termingerechte Auslieferung der einzelnen

Aufträge mit berücksichtigt. Insgesamt wurden in der Simulation 425 Aufträge auf 7 Maschinen berücksichtigt, welche im Abbildung 4.9 dargestellt sind.

In den Diagrammen ist die umgesetzte Energie pro Stunde (mittlere Leistung) angegeben. Ohne Optimierung steigt diese bis auf 240 kW an. Bei der leistungsoptimierten Berechnung kann eine Spitzenleistung von 180 kW erzielt werden. Durch Nutzung von Niedrigpreisphasen und die Möglichkeit zur bedienerunabhängigen Bearbeitung lassen sich erhebliche Kosteneinsparungen erzielen.

Der Vergleich der unterschiedlichen Optimierungen zusammen mit der nicht optimierten Produktion ergibt eine Kosteneinsparung von 9 %, Maximalleistungsreduzierung von 26 % und eine Senkung der durchschnittlichen Leistung von 33 % (Abbildung 4.10).

4.6 Aufzeigen von Flexibilitätsszenarien in der Produktion

Die Arbeiten in diesem Projekt haben unterschiedliche Werkzeuge für die Realisierung von Möglichkeiten zur Spitzenleistungsglättung, der Optimierung der durchschnittlichen Leistung und die Reduzierung der Energiekosten entsprechend vorgegebener Kriterien sowohl bezüglich des Netzanschlusses, der Preissignale als auch den Anforderungen nach Durchlaufzeiten, Personalverfügbarkeit, Schichtmodellen, Möglichkeiten zum autarken Betrieb von Maschinen sowie der chaotischen Fertigungsreihenfolge aufgezeigt. Dabei beschränkt sich die mögliche Nutzung von Flexibilität vor allem auf die Einzelteilund Kleinserienfertigung. Bei der Serienfertigung von Bauteilen ist die Nutzung aufgrund der begrenzten Ressourcen, deren Auslastung sowie der starren Abfolge nur eingeschränkt möglich. Weiterhin wurde die Möglichkeit zur Optimierung der Integration unterschiedlicher Speicher zur Pufferung kurzfristiger Überschüsse / Unterdeckungen ermöglicht. Bei diesen Speichern ist neben der Kapazität auch die Einund Ausspeicherleistung eine relevante Kenngröße.

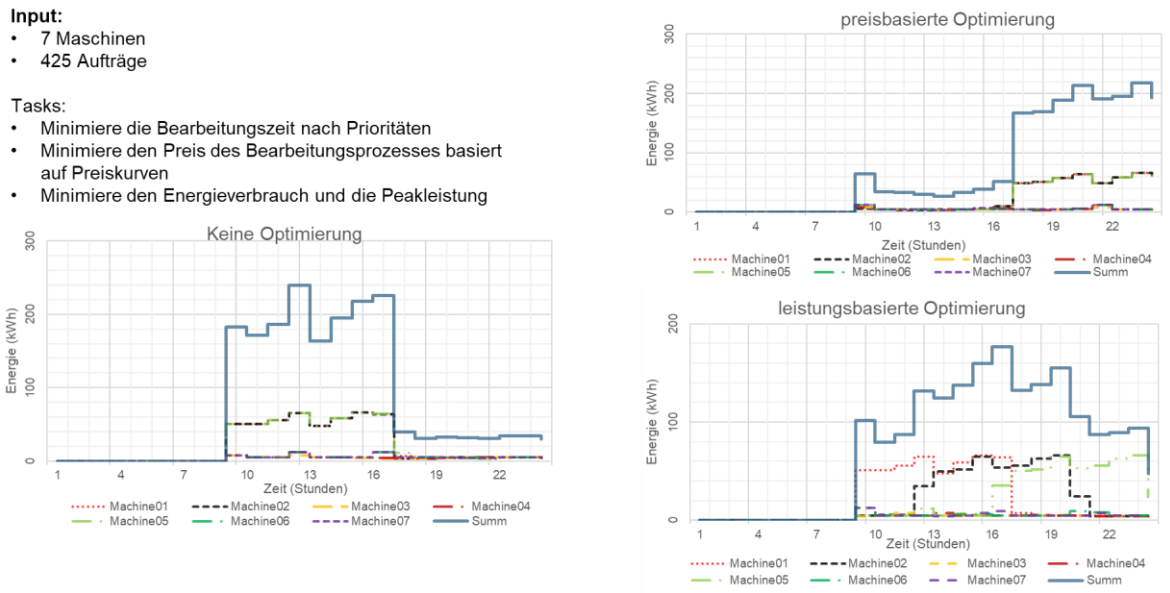
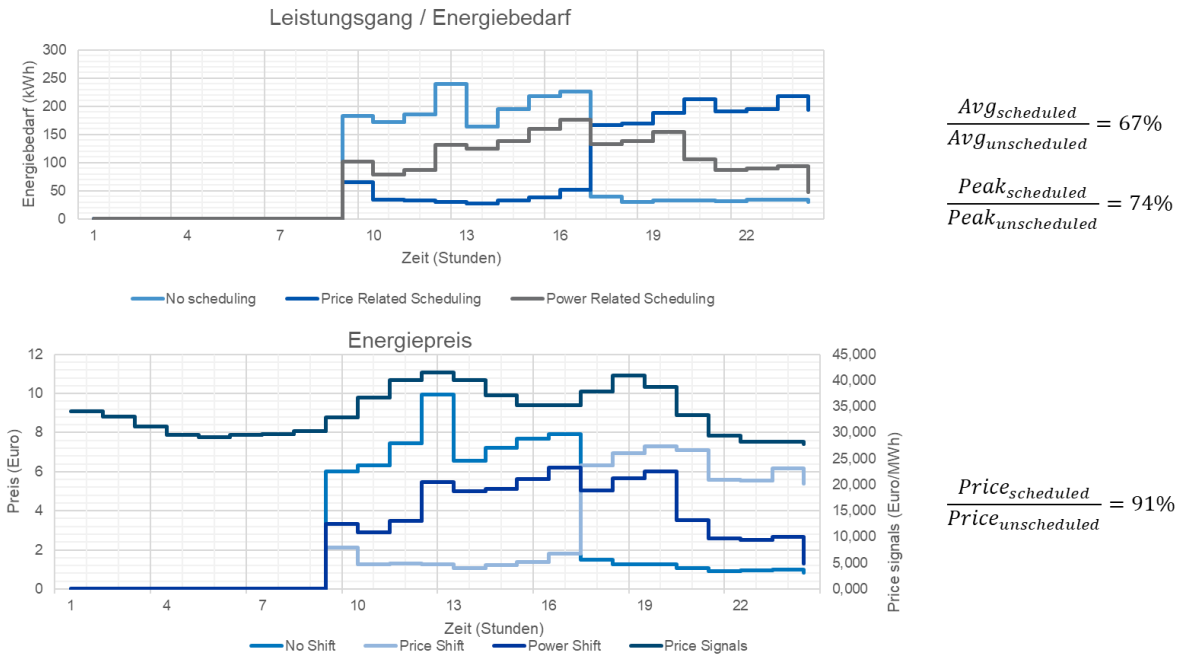


Abbildung 4.9: Preisund Leistungsoptimierung einer Beispielfertigung



$$\frac{Avg_{scheduled}}{Avg_{unscheduled}} = 67\%$$

$$\frac{Peak_{scheduled}}{Peak_{unscheduled}} = 74\%$$

$$\frac{Price_{scheduled}}{Price_{unscheduled}} = 91\%$$

Abbildung 4.10: Einsparpotenzial bei Leistung und Kosten

Mit den erarbeiteten Ergebnissen ist die Kopplung von Kapazitätskurven anderen Sektoren (z.B. privaten Haushalten) möglich.

Der nach dem externen Preissignal und betriebsinternen Kriterien optimierte geplante Lastgang wird in die Blockchain geschrieben und steht in Echtzeit allen Teilnehmern des Netzwerks zur Verfügung. Insbesondere der Schwarm-Manager kann die Netzprognose aktualisieren und ggf. ein neues Preissignal veröffentlichen.

4.7 Fazit

In dieser Arbeit wurde mit einer Industriesimulation erfolgreich ein weiterer Teilnehmer zum BlockchainNetzwerk hinzugefügt. Außerdem wurden realistische Lastprofile für die Netzteilnehmer in der Simulation umgesetzt. Weiterhin ist die Blockchain auf ein effizientes PoA-Verfahren umgestellt und das Beitrittsverfahren für neue Teilnehmer vereinfacht worden.

In einem Showcase werden die Funktionen des überarbeiteten Smart Contracts zur dezentralen Netzregelung gezeigt. Das Prinzip der Preissenkung als Anreiz zu einem Zuschalten von Verbrauchern ist bisher jedoch nicht vollständig umgesetzt. Außerdem sollten weitere Teilnehmer mit Stromspeichern hinzugefügt werden um die Netzregelung weiterzuentwickeln.

Durch die flexibel erweiterbare PoA-Blockchain und das neue Smart Contract Konzept wird mit dieser Arbeit ein weiterer Schritt auf dem Weg zu einer effizienten autonomen Netzregelung geschafft.

5

Geschäftsmodelle für Stadtwerke

Stadtwerke sind zumeist kommunale Eigenbetriebe, deren Entscheidungsgremien politisch zusammengesetzt sind. Deshalb sind sie häufig auch mit Aufgaben der Daseinsvorsorge (ÖPNV, Bäder etc.) betraut, die aus dem Energiehandel (Strom, Gas und Wärme) subventioniert werden müssen. Durch die Liberalisierung der Märkte verloren Stadtwerke so zahlreiche Kunden, insbesondere gewerbliche Abnehmer.

Mit der Digitalisierung der Elektrizitätswirtschaft können Stadtwerke nicht allein ihr Kerngeschäft modernisieren, durch innovative , digitale Geschäftsmodelle erhalten sie die Chance, verlorene Kunden wiederzugewinnen, indem sie ihre Vorteile wie Kundennähe und Kompetenz in Energiefragen in die Waagschale werfen. Die Entwicklung digitaler Dienstleistungen wird in [Kah19] am Beispiel eines Virtuellen Kraftwerks ausführlich beschrieben. Dabei sind Innovationsbereitschaft der gesamten Organisation und kluger Umgang mit den vorhandenen Ressourcen die entscheidenden Faktoren. Dies steht häufig im Widerspruch zu einer Firmenkultur, deren oberstes Ziel eine störungsfreie Versorgung darstellt, was durch zahlreiche Regularien gestützt wird.

DLT gehört zur Avantgarde der digitalen Technologien, von daher wird sie in der Elektrizitätswirtschaft noch selten angetroffen. Die ersten Anwendungen finden sich im Bereich des P2P-Stromhandels (meistens in der Region), s. z.B. [Thy20]. Hierfür ist DLT prädestiniert, da der Herkunftsnachweis verhindert, dass Strommengen mehrfach abgerechnet werden und bei einem 15min-Takt auch kleinste Beträge abgerechnet werden müssen (Micropayment). Ein Stadtwerk, das die Blockchain betreibt, generiert seine Einnahmen nicht aus dem Stromhandel, sondern aus Transaktionsgebühren.

Das aktuelle Projekt behandelt das Kerngeschäft eines Stadtwerks als Verteilnetzbetreiber. Die Idee dahinter ist, dass, zumindest in einer regulatorischen Innovationszone, die Verbraucher ins Netzmanagement eingebunden werden und so die vorhandenen

Betriebsmittel besser genutzt werden können. Eine bessere Auslastung des vorhandenen Netzes erspart Investitionen in Erweiterungen, die nur selten ausgenutzt werden. Ein anderer Ansatz aus dem Virtuellen Kraftwerk Neckar-Alb (VKNA), der vollständig in der Kontrolle des Stadtwerks bleibt, findet sich in [Hac19]. Mit den ersparten Betriebskosten kann das Stadtwerk die beteiligten Betriebe für ihre netzdienlichen Aktionen vergüten.

Das beschriebene Geschäftsmodell weist, zumindest in der Anfangsphase mit wenigen Teilnehmern, eine strukturelle Schwäche auf, die sich nicht mit Geld kompensieren lässt: eine Zeitskalen Fehlanpassung. Betriebsmittel für den Netzbetrieb sind überwiegend langlebige Investitionsgüter; dagegen sind organisatorische Lösungen, wie das oben beschriebene Engpassmanagement, oft an Vertragslaufzeiten von wenigen Jahren gebunden. Das Stadtwerk muss bei seiner Investitionsplanung also immer auch mit dem Ausfall einzelner Akteure im stabilisierenden Netzwerk rechnen.

Die Frage der Wirtschaftlichkeit eines solchen Modells konnte im laufenden Projekt nicht geklärt werden. Hierfür müssen auf der Seite des Schwarm-Managers (Stadtwerk) wie in den Betrieben die Planungstools auf ihre Erweiterbarkeit hin untersucht werden. Dabei sind die Investitionen in die Planungs-Infrastruktur wie die Kosten des Betriebs entscheidend. Auf der anderen Seite können sich durch die Automatisierung auch Kostenreduktionen ergeben.

5.1 Experteninterviews

Ursprünglich war vorgesehen, am Rande von Tagungen Experten anzusprechen und später Interviews mit diesen zu führen. Durch die digitalen Tagungsformate in Zeiten der Corona-Pandemie war dies nicht möglich. So wurden aufgrund von Literaturrecherchen und persönlichen Empfehlungen 15 Personen angeschrieben. Dabei antworteten acht von zehn Autoren, die in ihren Artikeln eine persönliche Mailadresse publiziert hatten. Von den fünf über Karrierenetzwerke (LinkedIn, XING) kontaktierten Personen antwortete niemand. Nach einer kurzen Beschreibung der Fragen, erklärten sich vier Experten zur Teilnahme bereit – ein Interview-Partner hatte noch einen Kollegen dazu eingeladen, um alle Fragen qualifiziert beantworten zu können. Die Interviews wurden zwischen dem 23.04.2021 und dem 12.05.2021 via Telefon oder Videokonferenz geführt. Der Interview-Leitfaden ist in Anhang F dokumentiert.

Obgleich ein Zeitrahmen von 30 min vorgesehen war, ergaben sich zahlreiche Anknüpfungspunkte an verwandte Themen oder gemeinsame Bekannte, so dass alle Gespräche erst nach ca. 1 h beendet waren.

5.1.1 Engpassmanagement in Verteilnetzen

Die Verteilnetze sind überwiegend in einem guten Zustand mit ausreichend Leistungsreserven. Dies kann sich perspektivisch durch zwei Faktoren ändern:

- PV-Einspeisung belastet in ländlichen Regionen die Transformatoren oft schon stärker als die Lastspitzen der Verbraucher. Hier sind Nutzungskonzepte für Gebäude und Quartiere gefragt.
- Elektroautos (BEV) können die Lastspitzen in einem Quartier in die Höhe treiben, wenn sie gleichzeitig am Hausanschluss geladen werden. Über die Anmeldepflicht für Wallboxen (bis 11 kW) bzw. Genehmigungspflicht (über 11 kW) haben die Stadtwerke die Möglichkeit, Fernwirktechnik zu installieren.

Die ersten Anwender für eine Schwarm-Optimierung wären Betreiber von „schwach" ausgelegten Netzen oder von Insel-Netzen.

5.1.2 Flexibilitätsmärkte

Der heute meistdiskutierte Ansatz für Nutzung von Flexibilität ist eine Fernwirkung via CLS-Schnittstelle des Smart Meter Gateway (nach §14a ENWG), womit allerdings lediglich ad-hoc-Flexibilität erreicht wird. Dieser Mechanismus ähnelt dem bei „abschaltbaren Lasten".

Darüber hinaus finden sich spezielle Beispiele, bei denen Abweichungen von der Planung (sehr) großer Abnehmer durch andere Abnehmer kompensiert werden, die Lastspitzen puffern können (Aluminiumschmelze, Kühlhäuser etc.).

Für eine vorausschauende Netzdienlichkeit existiert momentan kein regulatorischer Rahmen.

5.1.3 Regulatorik

Drei der vier Experten gehen davon aus, dass die regulatorischen Hürden für lokale und kleine Anbieter in Zukunft weiter erhöht werden. Nur einer sieht positive Signale für Innovationen „von unten".

5.1.4 Blockchain und deren Implementierung

Blockchain-Anwendungen sind bisher selten und wurden von wenigen Spezialisten implementiert. Anwendung findet die Technologie bisher ausschließlich im Bereich des regionalen Stromhandels. Aufgrund der zahlreichen Teilnehmer ist die Installation mit einem erheblichen logistischen und investiven Aufwand verbunden. Da sich der Betrieb einer privaten Blockchain sehr stromsparend gestalten lässt, ist auch der Handel mit haushaltsüblichen Strommengen mit einem geringen Overhead belastet. Die Teilnehmer, z.T. auch in Österreich, wo die Regularien einfacher sind, waren überzeugt von Ihren neuen Möglichkeiten der Marktteilnahme.

5.1.5 Fazit

Die Schwarm-Optimierung mittels DLT-Technologien führt drei Innovationen in ein konservatives und auf Sicherheit bedachtes Umfeld ein:

1. Nutzung von Flexibilität ohne steuernden Eingriff
2. Dezentrale Kommunikation ohne Kontrolle durch eine Zentrale
3. Automatisierte Abläufe durch Smart Contracts

Nach Ansicht der Experten überfordert dies heute die meisten Stadtwerke. Da sie sich unter erheblichem wirtschaftlichem Druck befinden, werden sie einzelne digitale Geschäftsmodelle implementieren und in wenigen Jahren auch Kombinationen davon, wie die Schwarm-Optimierung. Als Vorreiter werden große und mittlere Stadtwerke auftreten, die Innovationen in ihrer gesamten Organisation leben. Kleine Stadtwerke werden sich dagegen auf standardisierte Produkte verlassen. Von daher ist en Ansatz zur Verbreitung der Projektergebnisse die Kooperation mit Software-Herstellern, die in der Branche eingeführt sind.

6

Zusammenfassung und Ausblick

Das Projekt Virtuelles Kraftwerk der 2. Generation hatte als Ziel, ein Konzept zu entwickeln, die eine verteilte Optimierung von Lastgängen unter Betrachtung der Netzgrenzwerte ermöglicht. Dies hat folgende Mehrwerte:

- Die Optimierung erfolgt auf der Basis eines Schwarm-Verfahrens (jeder Teilnehmer führt eine lokale Optimierung aus) und somit muss keine firmeninterne oder persönliche Information ausgetauscht werden, lediglich die Lastprofile.
- Der Prozess ist trotzdem für alle beteiligte transparent. Jeder weiß zu jeden Zeitpunkt, was gefordert wurde und wie es umgesetzt wurde. Abweichungen können auditiert werden (außerhalb des Scope des Projekts).
- Die Plattform erlaubt eine direkte Abrechnung mittels Kryptowährung oder Tokens, die zu einem späteren Zeitpunkt in Geldflüsse umgewandelt werden können.

Im Projekt wurde ebenfalls ersichtlich, dass komplexe Berechnungen in der Blockchain mithilfe von einem Smart Contract wenig zielführend ist. Daher werden Teilprozesse der Schwarm-Optimierung aufgeteilt. Die Aufteilung von on-chain und off-chain Algorithmen ist eine wichtige Architekturentscheidung, die mittlerweile als Stand der Technik in vielen Bereichen anerkannt wird.

6.1 Ausblick

In der Bearbeitung des Projekts kamen verschiedene Fragestellungen, die eine tiefere Untersuchung benötigen. Vor allem folgende Aufgaben:

- Verifikation der Ergebnisse in einem Feldtest mit realen Unternehmen (vergl. [Kah20])
- Modellierung von Erzeugungseinheiten wie BHKWs oder Dampfturbinen in produzierenden Unternehmen.
- Die Einbindung von Elektromobilität wurde im Rahmen des Projekts nur eingeschränkt angegangen, jedoch sind Elektro-autos, die Vehicle to Grid Funktionen implementieren, eine wichtige Tragesäule der zukünftigen Optimierung von Netzen. Dabei sind Modelle für „Laden am Arbeitsplatz" in unserem Kontext von besonderem Interesse.
- Die Abrechnung im Falle einer vom Netzbetreiber vermittelte Einschränkung, bzw. für die Optimierung und Energieaus-tausch zwischen den verschiedenen Akteuren wurde zwar auf Basis einer Kryptowährung implementiert, jedoch eignet sich ein Token-basiertes Verfahren viel besser.
- Der Abrechnungszyklus (im Rahmen des Projekts je 15 min) wurde mit dem Regeltakt vom Netzbetreiber umgesetzt. Es kann jedoch sein, dass sich mit einer höheren zeitlichen Auflösung wesentlich bessere Ergebnisse im Hinblick auf Kosten erzielen lassen.

7

Kommunikation und Verbreitung

Unmittelbar vor Projektstart besuchte der DBU-Generalsekretär am 08.03.2019 den „Demonstrator Virtuelles Kraftwerk Neckar-Alb" an der Hochschule Reutlingen und informierte sich über die Vorarbeiten und das aktuelle Vorhaben.

Aufgrund der Pandemie-Situation fanden, bis auf die im kleinen Rahmen unter Industriebeteiligung abgehaltenen Industriekolloquien über die Semester- und Abschlussarbeiten an der Hochschule, keine öffentlichen Veranstaltungen statt, bei denen Projektergebnisse vor Interessenten und Multiplikatoren vorgestellt werden konnten.

Bei internationalen Konferenzen wurden drei Beiträge eingereicht und akzeptiert:

D. Coll-Mayor, A. Notholt: **Development and test of distributed ledger technologies applications in a microgrid distributed control**. International Conference on Renewable Energies and Power Quality (ICREPQ'19)Tenerife (Spain), 10th to 12th April, 2019

D. Coll-Mayor, A. Notholt: **Transforming the Energy System with P2P transactions between distributed generators and end consumers**. International Conference on Renewable Energy and Power Quality (ICREPQ'20). University of Granada. Spain. 2, 3, 4 of September 2020

D. Coll-Mayor, A. Notholt: **Transforming the Energy System with P2P transactions between distributed generators and end consumers**. International Conference on Renewable Energy and Power Quality (ICREPQ'20). University of Granada. Spain. 2, 3, 4 of September 2020

Weiterhin entstand eine Veröffentlichung:

P. H. Nebeling: **Energetische Analyse von Produktionssystemen**, VDE-Verlag, etz elektrotechnik & automation · 141. Jahrgang · S2-S3/2020, S. 24 – 29, ISSN 0948-7387, Berlin, 2020

Literaturverzeichnis

[BDE21] BDEW. Rollenmodell für die Marktkommunikation im deutschen Energiemarkt Version 2.0. March 2021.

[Bel18] Belin, Oliver. The difference between blockchain and distributed ledger technology. https://tradeix.com/distributed-ledger-technology/, January 2018. Abgerufen: 12. September 2020.

[Bey19] Stefan Beyer. What is Go Ethereum? A Detailed Guide. https://www.mycryptopedia. com/what-is-go-ethereum-a-detailed-guide/, December 2019. Abgerufen: 19. September 2020.

[Bin] Was ist eine digitale Signatur? https://academy.binance.com/de/blockchain/what-is-a-digital-signature. Abgerufen: 13. September 2020.

[Bit19] Bitcoin vs. Ethereum: Aussichten für die Kryptowährungen? https://coincierge.de/ bitcoin-vs-ethereum/, May 2019. Abgerufen: 19. September 2020.

[BNS09] Albrecht Beutelspacher, Heike B Neumann, and Thomas Schwarzpaul. Kryptografie in Theorie Und Praxis: Mathematische Grundlagen für Internetsicherheit, Mobilfunk und elektronisches Geld. Vieweg+teubner Verlag, 2 edition, 2009.

[Bre] Erneuerbare energien und stabile netze. https://www.bundesregierung.de/breg-de/aktuelles/erneuerbare-energien-und-stabile-netze-422384. Zugriff: 2020-01-15.

[But20] Vitalik Buterin. Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform, 2020. https://ethereum.org/en/whitepaper/, (Stand 09.09.2020).

[DLB+12] Dong Dong, Jin Li, Dushan Boroyevich, Paolo Mattavelli, Igor Cvetkovic, and Yaosuo Xue. Frequency behavior and its stability of grid-interface converter in distributed generation systems. In 2012 Twenty-Seventh Annual IEEE Applied Power Electronics Conference and Exposition (APEC), pages 1887–1893. IEEE, 2012.

[Ene19] Energy Web Foundation. The energy web chain: Accelerating the energy transition with an open-source, decentralized blockchain platform. Technical report, Energy Web Foundation, July 2019.

[Eth] Das Ether 1x1 – Teil 8: Smart Contracts auf der Ethereum Plattform. https://zertifikate.vontobel.com/DE/blog/Artikel/ das-ether-1x1-teil-8-smart-contracts-auf-der-ethereum-plattform. Abgerufen: 19. September 2020.

[Eth20] Ethereum. What is Ether (ETH)?, 2020. https://ethereum.org/en/eth/, (Stand 10.09.2020).

[FM20a] Hans-Georg Fill and Andreas Meier. Blockchain Grundlagen, Anwendungs-szenarien und Nutzungspotenziale. Springer-Verlag, Berlin Heidelberg New York, 2020.

[FM20b] Hans-Georg Fill and Andreas Meier. Blockchain kompakt: Grundlagen, Anwendungsoptionen und kritische Bewertung. Springer Fachmedien Wiesbaden, Wiesbaden, 2020.

[Fra] Energy-charts frauenhofer ise. https://energy-charts.info/charts/power/chart.htm?l=de&c=DE. Zugriff: 2020-01-15.

[GoE] Go-ethereum (Geth). https://docs.kaleido.io/kaleido-platform/protocol/ethereum/geth/. Abgerufen: 19. September 2020.

[Gro] Matthias Grote. Raspberry Pi OS. https://www.heise.de/download/product/raspbian-91329. Abgerufen: 19. September 2020.

[Hac19] Khalid Hachimy und Frank Truckenmüller, Demonstrator Automatisierte Kabelverteiler als Alternative zum regelbaren Ortsnetztrafo (DEMOrONT-Alternative), Statuskolloquium Umweltforschung Baden-Württemberg 2019, Fellbach, 19.03.2019

[HAT+20] 50 Hertz, Amprion, Tennet, TransnetBW, and EEX. Stromproduktion in deutschland in woche 52 2019, 2020. "https://www.energy-charts.de/power_de.htm?source=all-sources&year=2019&week=52, (Stand 01.01.2020)".

[Hen17] Jake Henningsgaard. Monitoring the Ethereum Blockchain. https://blog.ippon.tech/monitoring-the-ethereum-blockchain/, November 2017. Abgerufen: 19. September 2020.

[HLRS20] Dr. Birgit Haller, Dr. Ole Langniß, Dr. Albrecht Reuter, and Nicolas Spengler. 1,5° Csellsius: Energiewende Zellulär Partizipativ vielfältig umgesetzt. C/sells Selbstverlag c/o Dr. Langniß Energie und Analyse, 2020.

[HP12] Hans-Martin Henning and Andreas Palzer. 100% erneuerbare energien für strom und wärme in deutschland. Freiburg, Fraunhofer-Institut für Solare Energiesysteme, 2012.

[Hus18] Alen Huskanović. Params in ethereum genesis block explained async labs. https://www.asynclabs.co/blog/params-in-ethereum-genesis-block-explained/, July 2018. Abgerufen: 19. September 2020.

[ITF19] Unterschied zwischen Peer-to-Peer und Client Server? https://www.it-fabrik.gmbh/unterschied-zwischen-peer-to-peer-und-client-server/, November 2019. Abgerufen: 14. September 2020.

[Kah20] Claus Kahlert, Uwe Ziegler, Thomas Röger, Sabine Löbbe und Helmut Nebeling, Entwicklung einer Methodik zur Einbindung von KMU unterschiedlicher Branchen in ein Virtuelles Kraftwerk, DBU-Abschlussbericht-AZ-33154/01, 2020.

[Kam10] Andreas Kamper. Dezentrales Lastmanagement zum Ausgleich kurzfristiger Abweichungen im Stromnetz. KIT Scientific Publishing, 2010.

[KHK17] KfW, Dr. Holger Höfling, and Dr. Henrike Koschel. Smart grid schema, 2017. https://www.kfw.de/stories/kfw/stories/umwelt/erneuerbare-energien/ smart-grids-intelligentes-stromnetz/, (Stand 01.01.2020).

[Kom] Asymmetrische Kryptografie (Verschlüsselung). https://www.elektronik-kompendium.de/sites/net/1910111.htm. Abgerufen: 13. September 2020.

[Mah18] Till Antonio Mahler. Merkle Trees — Ensuring Integrity On Blockchains blockwhat? https://medium.com/blockwhat/ merkle-trees-ensuring-integrity-on-blockchains-508d6647d58e, December 2018. Abgerufen: 14. September 2020.

[McC] Gregory McCubbin. Intro to Web3.py – Ethereum For Python Developers. https://www. dappuniversity.com/articles/web3-py-intro. Abgerufen: 19. September 2020.

[Min17] Ethereum Mining: So minen Sie Ethereum am PC. https://www.bitcoinmag.de/mining/ethereum-mining, March 2017. Abgerufen: 19. September 2020.

[MS14] Christoph Meinel and Harald Sack. Sicherheit Und Vertrauen Im Internet: Eine Technische Perspektive. Springer Vieweg, 2014 edition, 2014.

[Net] Client-Server-Modell. https://www.netzorange.de/it-ratgeber/ client-server-modell-ein-klassisches-kommunikationsmodell/. Abgerufen: 14. September 2020.

[NLZ+04] Marcus VA Nunes, JA Pecas Lopes, Hans Helmut Zurn, Ubiratan H Bezerra, and Rogério G Almeida. Influence of the variable-speed wind generators in transient stability margin of the conventional generators integrated in electrical grids. IEEE Transactions on Energy Conversion, 19(4):692–701, 2004.

[Pan19] Alexander Panknin. Raspberry Pi 4 Mehr Power für Retro-Gaming. https://www.gaming-grounds.de/ jetzt-erhaeltlich-raspberry-pi-4-mehr-power-fuer-retro-gaming/, June 2019. Abgerufen: 19. September 2020.

[Pöt18a] Harald Pöttinger. Asymetrische Kryptographie ist für die Blockchain unverzichtbar. http://haraldpoettinger.com/blockchain-kryptographie/, June 2018. Abgerufen: 12.September 2020.

[Pöt18b] Harald Pöttinger. Die Integrität der Blockchain wird anhand von Hashwerten überprüft. http://haraldpoettinger.com/blockchain-hashwerte/, June 2018. Abgerufen: 12. September 2020.

[RA04] Cuauhtemoc Rodriguez and GAJ Amaratunga. Dynamic stability of grid-connected photovoltaic systems. In IEEE Power Engineering Society General Meeting, 2004., pages 2193–2199. IEEE, 2004.

[Rap] Raspberry pi 4 model B specifications. https://www.raspberrypi.org/products/ raspberry-pi-4-model-b/specifications/. Abgerufen: 19. September 2020.

[Rau20] Sebastian Rau. Ethereum 2.0 – Alles was Anleger wissen müssen. https:// blockchainwelt.de/ethereum-2-0/, July 2020. Abgerufen: 19. September 2020.

[Sch20a] Felix Schaal. Transactive control via ethereum blockchain, 2020.

[Sch20b] Peter Schmitz. Was ist Proof of Stake (PoS)? https://www.blockchain-insider.de/was-ist-proof-of-stake-pos-a-900657/, January 2020. Abgerufen: 13. September 2020.

[Sch20c] Peter Schmitz. Was ist Proof of Work (PoW)? https://www.blockchain-insider.de/was-ist-proof-of-work-pow-a-900636/, January 2020. Abgerufen: 13. September 2020.

[SIF07] Joe A Short, David G Infield, and Leon L Freris. Stabilization of grid frequency through dynamic demand control. IEEE Transactions on power systems, 22(3):1284–1293, 2007.

[SL19] Sigurd Schacht and Carsten Lanquillon. Blockchain und maschinelles Lernen Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren. Springer-Verlag, Berlin Heidelberg New York, 2019.

[Sol20] Solidity. Solidity introduction to smart contracts, 2020. https://solidity.readthedocs.io/en/v0.6.0/introduction-to-smart-contracts.html, (Stand 10.09.2020).

[Tab19] Vincent Tabora. A decomposition of the Bitcoin block header. https://www.datadriveninvestor.com/2019/11/21/ a-decomposition-of-the-bitcoin-block-header/, November 2019. Abgerufen: 14. September 2020.

[Thy20] E. Thyen, die Blockchain im energiewirtschaftlichen Einsatz – der Wuppertaler Tal.Markt, in O.D. Doleski (Hrsg.), Realisierung Utility 4.0 Dand 2, Springer, Wiesbaden, 2020.

[Tru20] What is truffle suite? Features, how to install, how to run Smart Contracts. https://www.upgrad.com/blog/what-is-truffle-suite/, March 2020. Abgerufen: 19. September 2020.

[Vir] Was ist ein virtuelles kraftwerk. https://www.next-kraftwerke.de/wissen/virtuelles-kraftwerk. Zugriff: 2020-01-16.

[WIK] Versorgungsqualität in elektrischen versorgungsnetzen. https://de.wikipedia.org/wiki/Versorgungsqualität. Zugriff: 2020-01-16.

[Zah] Das Ether 1x1 – Teil 2: Zahlen und Fakten zu Ethereum. https://zertifikate.vontobel.com/DE/blog/Artikel/ das-ether-1x1-teil-2-zahlen-und-fakten-zu-ethereum. Abgerufen: 19. September 2020.

[Zen20] Thorsten Zenner. Vorlesungskript Steuerungssysteme, 2020.

A Grundlagen

In diesem Kapitel soll die grundlegende Theorie, die für das Verständnis dieses Projekts notwendig ist, dargelegt werden. In einem ersten Schritt wird hierzu die Distributed-Ledger-Technologie genauer erläutert, welche die Basis für die Blockchain bildet. Neben der benötigten Hardware und Software wird zudem das Thema Elektromobilität, im Besonderen die Konzepte Vehicle to Grid (V2G) bzw. Vehicle to Home (V2H), näher beleuchtet.

A.1 Distributed-Ledger-Technologie

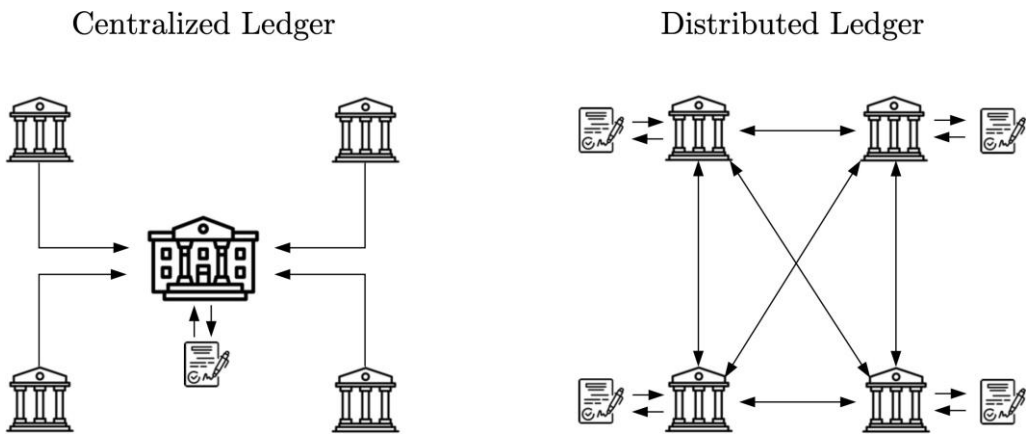


Abbildung A.1: Vergleich Centralized Ledger (l.) und Distributed Ledger (r.)

Mithilfe der DLT kann ein Informationssystem beschrieben werden, das als dezentrale Datenbank fungiert. Ins Deutsche übersetzt ergibt sich auch der Begriff „Verteiltes

Kontenbuch", der die dezentrale Funktion der Technologie verdeutlicht. Abbildung A.1 zeigt eine Gegenüberstellung zwischen einem sogenannten Centralized Ledger und dem Distributed Ledger. Im Vergleich zu einem Centralized Ledger ist hier keine dominante und zentrale Validierungsinstanz, die wie innerhalb der gezeigten Abbildung die Informationen speichert und verwaltet. Die Konsensbildung erfolgt stattdessen über ein demokratisches Prinzip mithilfe verschiedener Konsens-Algorithmen [SL19].

Den Kern der Technologie stellen die Transaktionen dar, die in einem Hauptbuch (Ledger) auf einer Vielzahl unabhängiger Rechner gespeichert werden. Die innerhalb des dezentralen Netzwerks ansässigen Rechner werden auch als Knoten bzw. Nodes bezeichnet. Jeder Knoten erhält somit vollen Zugriff auf alle bereits getätigten Transaktionen. Soll eine neue Transaktion durchgeführt werden, muss dies von den Knoten des Netzwerks zunächst verifiziert werden. Auch die Reihenfolge in der die Transaktionen erfasst werden, wird mithilfe des Konsens-Mechanismus verwaltet. Ein weiterer wichtiger Bestandteil der Distributed-Ledger-Technologie stellen die sogenannten Hash-Verfahren dar. Durch das Verfahren wird sichergestellt, dass die Person hinter einer Transaktion eindeutig nachgewiesen und während der Übertragung nicht verändert werden kann, wodurch der Schutz gegenüber dem Zugriff unberechtigter Personen gewährleistet wird [SL19]. Die Verfahren werden innerhalb des nachfolgenden Abschnitt A.2, in welchem das Thema Blockchain behandelt wird, näher beleuchtet.

A.2 Blockchain

Der bekannteste Ableger der Distributed-Ledger-Technologie ist die Blockchain. Die beiden Begrifflichkeiten werden oft synonym verwendet, jedoch stellen sie nicht den gleichen Sachverhalt dar. Demnach basiert die Blockchain grundlegend auf dem Prinzip der Distributed-Ledger-Technologie und erweitert diese um zusätzliche Funktionen. Zudem werden die Transaktionen in Form von Blöcken organisiert und mithilfe kryptografischer Verfahren miteinander verkettet, woraus sich auch der Name Blockchain ergibt [Bel18].

Insbesondere die fünf herausragenden Eigenschaften – dezentral, belastbar, verifizierbar, transparent und unveränderlich – zeichnen die Verwendung einer Blockchain aus. Die Dezentralität, die bereits in Abbildung A.1 thematisiert wurde, ermöglicht den Wegfall einer zentralen Kontrolle. Durch das breit gestreute Netz an Nodes wird zudem ein stabi-

les, ausfallsicheres und somit auch belastbares Netzwerk generiert. Da alle Teilnehmer Zugriff auf die verketteten Datenstrukturen haben und auf Basis bestehender Informationen neue Transaktionen verifizieren, entsteht ein transparentes und nachvollziehbares Netz an Daten. Der letzte und wichtigste Punkt stellt die Eigenschaft der Unveränderlichkeit dar. Da jede Transaktion in der Kette gespeichert und von den Teilnehmern verifiziert ist, macht dieses Prinzip ein nachträgliches Ändern bestehender Transaktionen unmöglich, da in einem solchen Fall ein bestimmtes Glied der Kette nicht mehr konsistent ist und mit den anderen Kopien auf den jeweiligen Nodes nicht mehr übereinstimmt. Dem manipulierten Knoten wird die weitere Teilnahme am Netzwerk nicht mehr gestattet [SL19].

A.3 Aufbau

Da jeder Blockchain-Ansatz auch in gewisser Weise einen differenzierten Aufbau aufweist, wird im Folgenden ein vereinfachtes Konzept der Bitcoin-Plattform vorgestellt, das sich jedoch in ähnlicher Weise auch bei Ethereum wiederfindet. Die Abbildung A.2 zeigt drei miteinander verkettete Blöcke einer Blockchain. Richtet man zunächst den Blick auf einen Block im Detail, so ist dieser in insgesamt zwei Teile untergliedert. Im unteren Bereich finden sich die Informationen zu der jeweiligen Transaktion, die innerhalb des Blocks durchgeführt wird. Geht es dabei z.B. um den Austausch einer virtuellen Währung (BTC, ETH , etc.), beinhalten die Transaktionen Angaben zur Übertragung in Form der öffentlichen Schlüssel der Teilnehmer, die über die privaten Schlüssel jeweils noch signiert und somit vor einem unberechtigten Zugriff durch Dritte geschützt werden [FM20a].

Innerhalb des Block-Headers befinden sich alle wichtigen Informationen, die den Block identifizieren und die Verkettung ermöglichen. In den folgenden Unterabschnitten soll daher auf die jeweiligen Elemente genauer eingegangen werden.

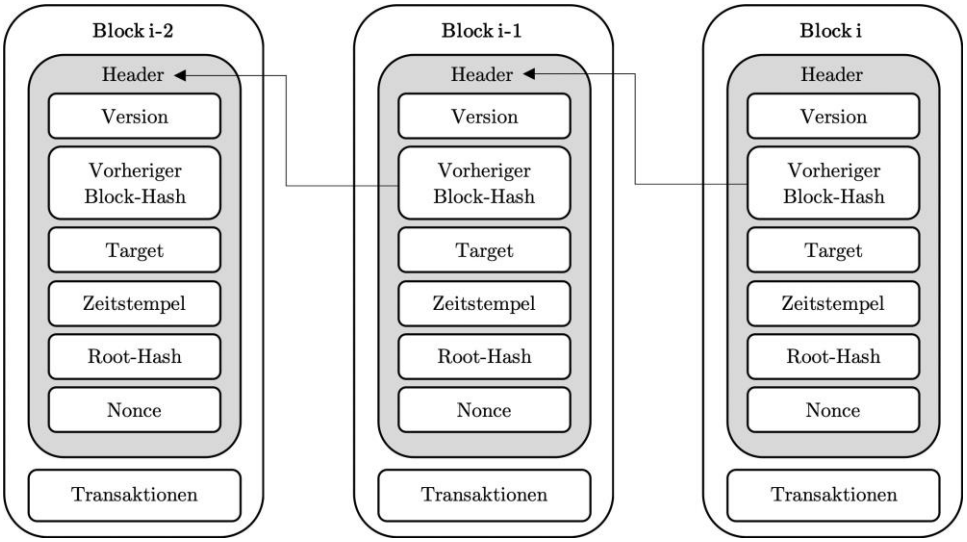


Abbildung A.2: Aufbau einer der einzelnen Blöcke innerhalb der Blockchain

A.3.1 Version

Die Versionsnummer gibt das Regelwerk bzw. Protokoll an, das für die Erstellung des Blocks verwendet wird. Wirft man den Blick auf Bitcoin, existieren hier z.B. ingesamt vier verschiedene Versionen, die innerhalb des Bitcoin Cores beschrieben werden. Ändert sich die Versionsnummer gegenüber einem vorherigen Block, können die folgenden Blöcke nicht mehr Teil der bestehenden Kette sein. In diesem Fall bildet sich eine Abspaltung der Blockchain, wobei auf einem neuen Zweig die hinzugefügten Blöcke angehängt werden. Hier spricht man auch von einem sogenannten Hard Fork [Tab19].

A.3.2 Vorheriger Block-Hash

Dieser Bereich beinhaltet, wie der Name bereits vermuten lässt, den Hash-Wert des vorherigen Block Headers. Er stellt damit eine Art Pointer auf den vorherigen Block dar und ist demnach der Grundbestandteil der Blockchain-Verkettung. Diese Information kann nicht geändert werden, ohne dass die Block-Hashes der vorherigen Header ebenfalls abgeändert werden. Es tritt der Lawineneffekt ein [FM20b][Tab19].

A.3.3 Target

Das Target gibt den Schwierigkeitsgrad, auch Difficulty genannt, der gestellten Aufgabe des Proofof-Work Konsens an. Je mehr Miner sich innerhalb des Netzwerks befinden, desto größer wird auch der gestellte Schwierigkeitsgrad [Tab19]. Auch mit der Anzahl an Validierungen und Konsensbildungen steigt der Schwierigkeitsgrad an, wie in Abbildung A.2.2 näher beschrieben wird.

A.3.4 Zeitstempel

Der Zeitstempel gibt den Zeitpunkt an, an dem mit dem Hashing des Headers begonnen worden ist. Dieser ist im Unix Format codiert und gibt somit die verstrichenen Sekunden seit dem 1. Januar 1970 an. Die Zeit, die zur Berechnung des Hashes benötigt wird, darf aufgrund der gestiegenen Komplexität der Aufgabe nicht kleiner sein als die mittlere Rechenzeit zum Lösen der vorangegangenen 11 Blöcke.

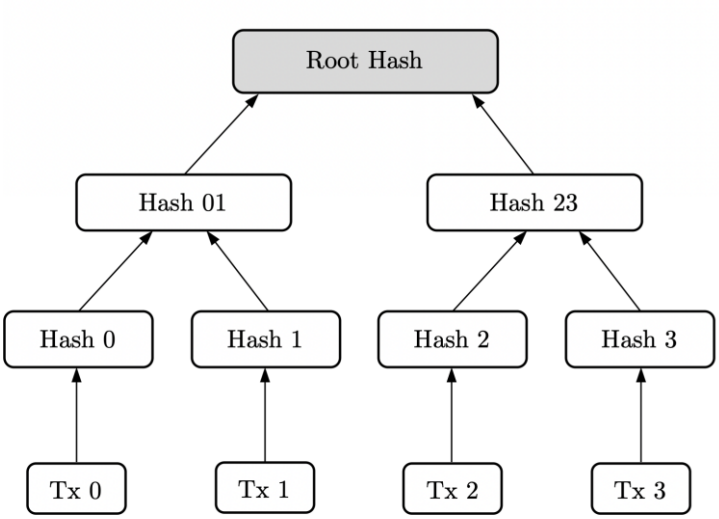


Abbildung A.3: Struktur eines Merkle Tree mit dazugehörigem Root-Hash

Vollständige Knoten akzeptieren zudem keine Blöcke, die laut ihrem Timestamp mehr als 2 Stunden in der Zukunft liegen [Tab19].

A.3.5 Root Hash

Innerhalb des Root Hashs werden alle Transaktionen bzw. die jeweiligen Hashes der einzelnen Transaktionen zu einem großen Hash zusammengefügt. Die Struktur die hierfür verwendet wird ähnelt einem Baum und wir daher auch nach dessen Erfinder Merkle Tree oder auch Hash-Baum bezeichnet. Die Abbildung A.3 zeigt exemplarisch einen Merkle Tree mit dem dazugehörigen Root Hash.

Auf der untersten Ebene sind die einzelnen Transaktionen 0 3 aufgelistet. Jede Transaktion wird mit einem Hash-Wert versehen, hier entsprechend Hash 0 3. Die jeweiligen Hashes der Transaktionen werden anschließend mit einem übergeordneten Hash versehen, der die Kombination aus den beiden vorherigen darstellt. Dies geschieht so lange, bis die Spitze des Baumes, in diesem Fall der Root Hash erreicht ist. Dieser Wert zeigt sich anschließend innerhalb des BlockHeaders.

Der Vorteil der sich hieraus ergibt ist, dass die Gültigkeit und Integrität einer bestimmten Transaktion des Blocks sehr einfach nachgewiesen werden kann. Soll so z.B. die Transaktion Tx 2 überprüft werden, wird ausschließlich Hash 3 sowie Hash 01 benötigt, um den Root Hash zu erzeugen. Weist der Baum eine deutlich größere Struktur auf, wie sie auch in einer echten Blockchain mit einigen tausend Transaktionen pro Block zu finden ist, wird dieser Effekt noch deutlicher. Zudem kann durch die Verwendung von Hashes ein großer Anteil an Speicherkapazität eingespart werden, da zur Verifizierung nicht die eigentlichen Daten sondern nur die jeweilige Kennung bzw. der eindeutige Fingerabdruck jeder Transaktion in Form des Hashes benötigt wird [Mah18].

A.3.6 Nonce

Innerhalb dieses Bereichs wird der Wert zur Lösung der Mining-Aufgabe, sprich zur Berechnung des Hash-Wertes, gespeichert. Wer als erstes diesen Nonce errechnet, kann den Block anschließend validieren und erhält hierdurch die Entschädigung bzw. Belohnung in Form der jeweiligen Kryptowährung. In Abhängigkeit der Gesamt-Hash-Rate des Netzwerks wird die Schwierigkeit bzw. Difficulty an gegebenen Bedingungen dynamisch

angepasst, wodurch auch die Rechenzeit zur Berechnung der Nonce variiert [Tab19]. Wie diese Validierung und Konsensbildung im Detail funktioniert, wird im nächsten Unterabschnitt (insbesondere innerhalb Abbildung A.2.2) genauer beleuchtet.

A.4 Grundlegende Verfahren

Im Nachfolgenden werden die grundlegenden Verfahren und Konzepte aufgezeigt, die die Datenstrukturen und Funktionen einer Blockchain ermöglichen.

A.4.1 Hash-Funktionen

Mittels der Hash-Funktion wird ein Block innerhalb der Blockchain mit einem bestimmten Fingerabdruck versehen, der eindeutig auf den Block und den darin enthaltenen Transaktionen zurückzuführen ist. Dieser Fingerabdruck oder auch Hash-Wert genannt wird zur Verkettung im nächsten Block ebenfalls aufgeführt und gespeichert(vgl. Abbildung A.2.1). Die Hash-Funktion ermöglicht dabei, dass aus einer beliebigen Menge an Eingangsdaten eine Ausgangsgröße fixer Länge erzeugt wird, wodurch Daten innerhalb des definierten Zielbereichs eingespart werden können. Die Ausgangsgröße kann zudem nicht mehr dem Eingangswert zugeordnet werden, der Hash-Wert ist demnach nicht invertierbar. Ein wichtiges Kriterium für die Verwendung von Hash-Funktionen ist eine starke Kollisionsresistenz. Eine Kollision entsteht dann, wenn zwei unterschiedliche Eingangsdaten zu einem gleichen Hash-Wert führen. Daher muss der Zielbereich durch die Funktion möglichst optimal ausgenutzt werden. Man spricht hier von einer Streuung, woraus sich auch der deutsche Begriff für die Hash-Funktion – Streuwert-Funktion – ableiten lässt. Bereits kleinste Änderung der Eingangsdaten führen daher zu einem gänzlich anderen Zielwert, was auch als Lawinen-Effekt bezeichnet wird. Wird innerhalb der Blockchain demnach eine Transaktion nachträglich manipuliert, ändert sich auch der jeweilige Hash-Wert der verketteten Blöcke, wodurch die Integritätsbedingung der Konsistenz nicht mehr gewährleistet werden kann. Die Manipulation wird sofort erkannt [Pöt18b].

Ein innerhalb der Blockchain verwendeter Algorithmus zur Generierung eines Hash-Wertes ist der SHA-256-Hash-Algorithmus. Dieser erzeugt aus einem beliebig großen Eingangswert einen Ausgangswert mit einer fixen Länge von 256 Bit, was einer Dezimalzahl

mit 78 Stellen entspricht. Der Lösungsraum der Ausgangsgröße weist somit eine ausreichende Größe auf, um Kollisionen in der Berechnung zu vermeiden [FM20a].

A.4.2 Asymmetrische Verschlüsselung

Jeder Teilnehmer des Netzwerks erhält bei dieser Verschlüsselung einen privaten und einen öffentlichen Schlüssel, der sich aus mathematisch generierten alphanumerischen Zeichen zusammensetzt. Durch das Schlüsselpaar können die Daten vor unberechtigtem Zugriff durch Dritte geschützt werden. Bei dieser Form der Kryptographie wird oft auch von einem sogenannten Falltür-System gesprochen. Dabei existieren zum einen öffentliche Informationen, die für jeden Nutzer des Netzwerks zugänglich sind. Diese werden durch den öffentlichen Schlüssel repräsentiert, wodurch andere Nutzer dem Eigentümer Datenpakete zukommen lassen können. Hierzu werden die Daten mithilfe des öffentlichen Schlüssels verschlüsselt, was in Abbildung A.4 zu erkennen ist [Kom].

Über den privaten Schlüssel, der oftmals in einer speziell gesicherten Umgebung abgelegt ist, wird der berechtigte Empfänger verifiziert und kann so wiederum auf seine Daten zugreifen, die ansonsten nur in verschlüsselter Form vorliegen. Das Schlüsselpaar wird über eine einfache Berechnung generiert, die sich jedoch mathematisch nicht umkehren lässt (RSA-Verfahren – Einwegfunktionen auf Basis von Primzahl-Multiplikationen). Nur über Brute-Force-Methoden kann das Schlüsselpaar in der Theorie errechnet werden. Jedoch steigt der Rechenaufwand auch durch den Einsatz rechenstarker

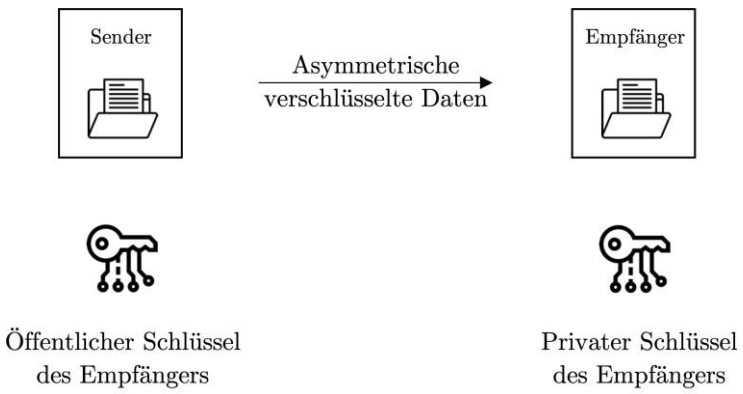


Abbildung A.4: Public-Key-Verfahren

Computer so stark an, dass der Schlüssel nicht innerhalb einer praktikablen Zeit errechnet werden kann (Shor-Algorithmus – Primfaktorzerlegung und Berechnung diskreter Logarithmen) [Pöt18a].

A.4.3 Digitale Signaturen

Neben der Verschlüsselung von Nachrichten bzw. Datenpaketen spielen auch digitale Signaturen eine entscheidende Rolle beim Austausch von Daten. Im Vordergrund steht dabei die Authentizität – der Verfasser der Nachricht ist auch der tatsächliche Absender – sowie die Integrität, die die unverfälschte Übermittlung der Nachricht beschreibt.

Die Abbildung A.5 zeigt den Prozess einer digitalen Signatur. Mittels eines Hash-Algorithmus wird zunächst die Prüfsumme des zu übermittelnden Datenpakets der Person A erzeugt. Anschließend wird der Hash-Wert über den Private Key (Person A) verschlüsselt und an das Ende der Nachricht gehängt, wodurch sich die digitale Signatur ergibt. Empfänger B kann nun den Hash-Wert in Kombination mit dem öffentlichen Schlüssel des Senders A auf die digitale Signatur anwenden und erhält hierdurch die Gültigkeit der übermittelten Signatur [BNS09] [Bin].

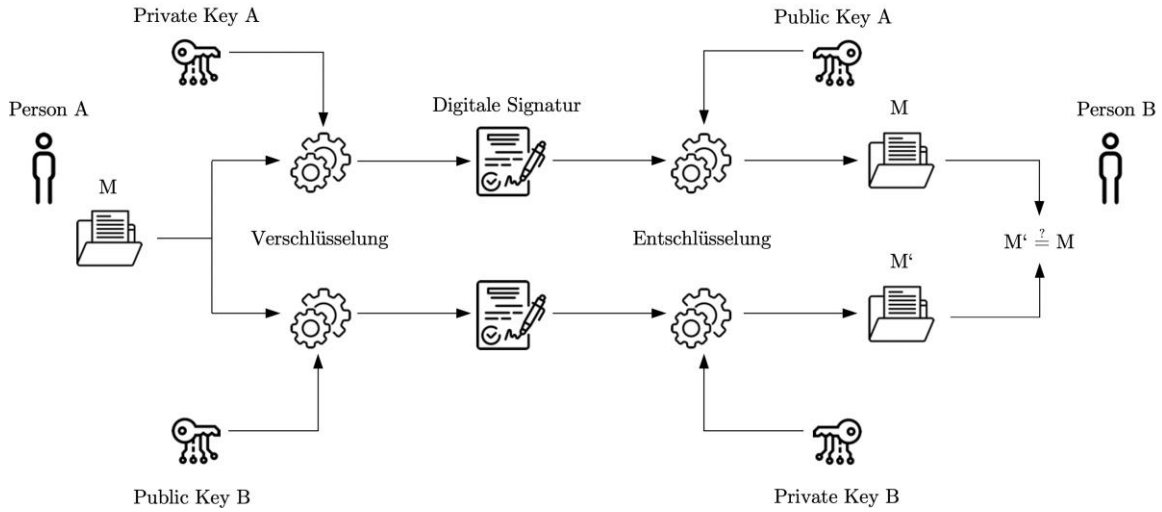


Abbildung A.5: Schematische Darstellung einer digitalen Signatur

Zusätzlich kann Person A über den Public Key des Empfängers B verschlüsselt und als Chiffrat verschickt werden. Nach der Entschlüsselung der Nachricht über den Private Key

der Person B kann der Empfänger die beiden gesendeten Nachrichten miteinander vergleichen. Hierdurch kann die Integrität der übermittelten Nachricht nachgewiesen und validiert werden [MS14].

A.4.4 Konsens-Algorithmen

Wesentlicher Bestandteil der Blockchain-Technologie stellen die sogenannten Konsens-Algorithmen dar, welche die Funktionsfähigkeit eines dezentralen Netzwerks mittels der Transaktionsverifizierung und Bewahrung der Integrität des Ledgers sicherstellen, vor allem mit dem Hintergrund der wegfallenden zentralen Kontrollinstanz. Um die Validität von Transaktionen zu verifizieren, können verschiedene Algorithmen eingesetzt werden. Vor allem die Ansätze Proof-of-Work sowie Proof-of-Stake stellen die bekanntesten Algorithmen bei einem Netzwerk dar, in welchem die Teilnehmer untereinander nicht bekannt sind, sich also nicht vertrauen können.

A.4.5 Proof-of-Work

Der Proof-of-Work Algorithmus basiert auf der Idee, dass die im Netzwerk befindlichen Teilnehmer eine relativ komplexe Aufgabe lösen müssen. Mithilfe einer kryptografischen Hash-Funktion soll ein gewisser Ausgabewert innerhalb eines definierten Wertebereich gefunden werden (Aufgabenstellung), was in Abbildung A.6 dargestellt ist. Aufgrund der SHA-256-Funktion kann jedoch nicht bereits im Voraus abgeschätzt werden, wie das Ergebnis auf Basis der Hash-Funktion erzielt werden kann. Durch einen gewissen Arbeitsaufwand (Proof-of-Work) werden solange zufällige Werte durch die Brute-Force-Methode durchprobiert, bis eine Lösung für die komplexe Aufgabe gefunden ist [FM20a]. Die Lösung wird also nicht algorithmisch, sondern durch „Würfeln" gefunden. Ziel dieser Methodik ist es, einen Missbrauch durch Teilnehmer des Netzwerks zu verhindern. Ein Teilnehmer könnte z.B. auch bestimmte Blöcke mit Transaktionen erzeugen, die einen gewissen Vorteil für den jeweiligen Teilnehmer erbringen. Durch den zunächst recht hohen Arbeitsbzw. Rechenaufwand und die damit verbundenen Kosten zur Berechnung eines Hashes sorgen dafür, dass Teilnehmer nicht beliebig viele bzw. auch manipulierte Transaktionen in die Blockchain einspeisen. Hierdurch wird jedoch nicht sichergestellt, dass solche Transaktionen den Weg in die Blockchain finden, dennoch ist die finanzielle

und vor allem zeitliche Hürde in vielen Fällen zu hoch, um eine nicht valide Transaktion einzuspeisen. Ein weiterer Grund hierfür ist vor allem die Verifikation eines Blockes,

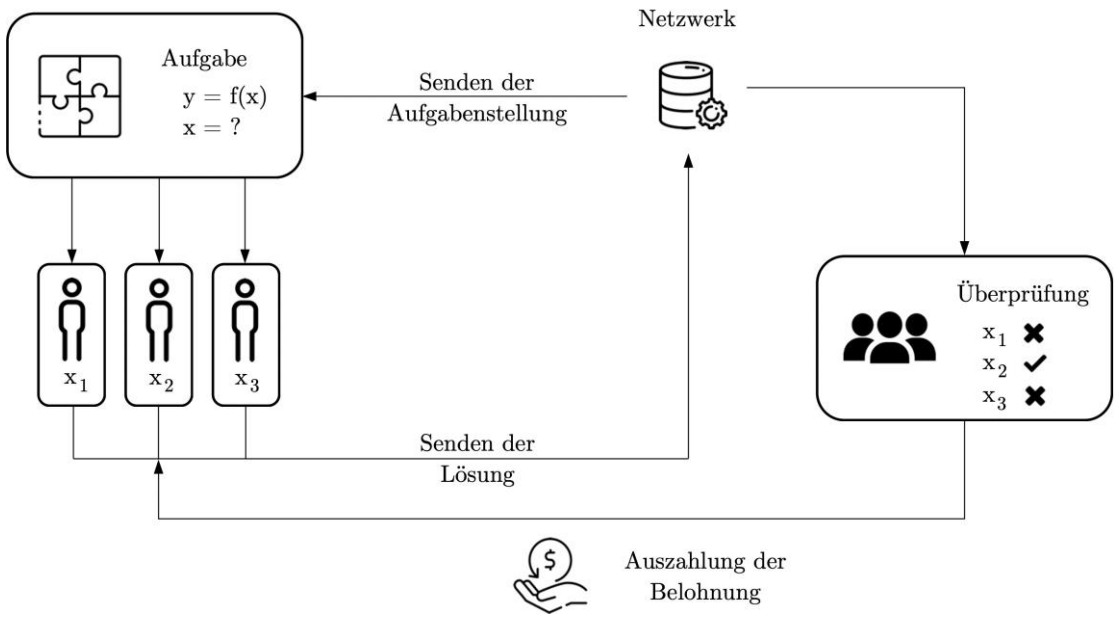


Abbildung A.6: Proof-of-Work Konsens

in welchem die neuen Transaktionen hinterlegt ist. Bevor ein Block demnach schlussendlich an die Blockchain angehängt wird, muss er von den Teilnehmern des Netzwerks verifiziert werden. Aufgrund der bekannten Hash-Funktion kann der erzeugte Hash im Gegensatz zu dessen Berechnung sehr einfach verifiziert werden. Eine fehlerhafte oder ungültige Transaktion kann so mithilfe der Verifizierung durch unterschiedliche und vor allem unabhängige Partner herausgefiltert werden [SL19].

Der Prozess aus Sicht des Teilnehmers, der die Arbeitsleistung erbringt (Miner), wird als mining bezeichnet. Erfüllt der errechnete Hash die Bedingung sowie eine erfolgreiche Verifizierung, wird der Miner mit einer bestimmten Belohnung vergütet. Da alle Teilnehmer innerhalb des Netzwerks an dem Mining-Prozess teilnehmen können, erhält derjenige Teilnehmer die Prämie, der als erstes den korrekten Hash des neuen Blocks erzeugt. In Abbildung A.6 wäre dies z.B. Teilnehmer 2 mit der Lösung x2. Je mehr Teilnehmer sich im Netzwerk befinden bzw. je mehr Rechenleistung für die Berechnung und Validierung eines neuen Blocks aufgebracht werden muss, desto mehr erhöht sich auch die Komplexität der Aufgabe. Die sogenannte Difficulty wird dabei künstlich erhöht, um die Generierung neuer Blöcke gezielt zu verzögern (Verhinderung von DDoS-Angriffen).

Neben vielen Vorteilen bringt die Methodik jedoch auch negative Aspekte mit sich. Da jeder Teilnehmer um die schnellstmögliche Berechnung des Hashes kämpft, existiert ein hoher Konkurrenzdruck. Die besten Möglichkeiten hat dabei der Teilnehmer, der auch die meisten und vor allem leistungsstärksten Ressourcen zur Verfügung hat. Kleine Teilnehmer mit nur sehr geringen Ressourcen haben kaum die Chance, einen Hash zu errechnen. Einhergehend mit den sehr rechenintensiven Berechnungen ist der Stromverbrauch, der zum Betrieb der Hardware benötigt wird. Daher befinden sich große Mining-Farmen in Ländern, in denen die Stromkosten sehr gering sind, wie z.B. China, Island, Georgien, etc. Auch die Außentemperaturen spielen bei der Standortwahl eine wichtige Rolle, da durch die Rechner auch eine enorme Wärmeenergie ausgeglichen werden muss. Insgesamt sind solche Farmen demnach kritisch bezüglich dem Umwelt-Aspekt zu betrachten, da das Ergebnis des Hashes abgesehen von dem Nachweis der Arbeitsleistung, keinem speziellen Zweck dient [Sch20c].

A.4.6 Proof-of-Stake

Der Konsens Proof-of-Stake macht sich zum Ziel, Transaktionen zu verifizieren und eine digitale Einigung über die Richtigkeit der Blockchain zu evozieren. Demnach liegt kein Unterschied zwischen den Zielsetzungen der beiden Konsens-Algorithmen vor. Innerhalb des Proof-of-Stake Verfahrens wird jedoch auf Basis eines Anteilsystems gearbeitet. Grundsätzlich kann jeder am Validierungsprozess teilnehmen, der einen Anteil an der Coinbase hat bzw. Token in seinem Wallet verzeichnen kann. Wer jedoch explizit validieren darf, wird mithilfe eines Zufallssystems ermittelt. Je mehr Token bzw. Coins der Teilnehmer aufweisen kann, bzw. je größer der Gesamtanteil an der Coinbase ist, desto größer ist auch die jeweilige Chance den Zuschlag für die Validierungsund Konsensbildung zu erhalten [Sch20b]. Da nicht wie beim Konsens-Algorithmus Proof-of-Work eine Aufgabe gelöst werden muss, entfällt die direkte Belohnung bei der Erstellung und Verkettung eines weiteren Blocks. Stattdessen wird der ausführende Nutzer hier über eine Transaktionsgebühr bezahlt, die bereits vor Beginn der Validierung bekannt ist. Der jeweilige Anteil der Coinbase wird während der Validierung und Konsensbildung als Kaution eingezogen, so dass der Nutzer zu einem transparenten und akkuraten Vorgehen verpflichtet ist. Sollte es zu einem nicht validen Block kommen, wird die Kaution dem Netzwerkteilnehmer entzogen [SL19].

Der Vorteil, der sich aus diesem Konsens-Algorithmus ergibt, ist vor allem gegenüber dem Proof-ofWork Ansatz der deutlich geringere Energiebedarf. Durch den fehlenden Konkurrenzdruck und den Entfall von parallelen Berechnungen der verschiedenen Netzwerkteilnehmern, reduziert sich der Rechenaufwand für die jeweiligen Ressourcen enorm. Zudem ergibt sich eine höhere Sicherheit, da Teilnehmer, die während eines Validierungsverfahrens Transaktionen manipuliert haben, durch den Wegfall der Kaution bzw. des hinterlegten Token-Anteils praktisch keine Chance mehr auf eine weitere Validierung erhalten. Durch das Anteilssystem ergibt sich hingegen der Nachteil, dass nur ein kleiner Teilnehmerkreis eine realistische Chance auf das Erzeugen eines neuen Blocks hat. Der Einfluss kann zwar über weitere Methoden wie Randomized Block Selection oder Coin Age-Based Selection reduziert werden, dennoch bleibt die deutlich höhere Wahrscheinlichkeit mit steigendem Anteil bestehen. Auch die Gefahr einer 51%-Attacke kann so in der Theorie bestehen, wobei hier mehr als 50% der verfügbaren Coins bzw. Tokens auf eine einzige Person zurückführen müssten, was jedoch in der Praxis aufgrund des hohen Risikos und den enormen Investition sehr unwahrscheinlich ist. Dieses Monopol würde dennoch bedeuten, dass die Blockchain nur von einer einzigen Person kontrolliert werden könnte. Der dezentrale Ansatz und somit auch grundlegende Kern der Technologie würde dabei verloren gehen [Sch20b].

A.4.7 Netzwerktypologie

Die Netzwerktypologie befasst sich mit Verfahren bzw. Architekturen zur Netzwerkschicht, die für die Kommunikation zwischen den einzelnen Teilnehmern bzw. den untereinander unabhängigen Nodes unabdingbar ist. Insgesamt werden im Folgenden zwei wesentliche Architekturen beschrieben.

A.4.8 Client-Server Architektur

Die Client-Server Architektur ist der aktuell noch am meisten verbreitete Netzwerktyp im IT-Bereich. Gängige Client-Server Anwendungen sind z.B. HTTP, FTP, rsync, etc. Das Modell setzt sich dabei grundlegend aus zwei verschiedenen Arten von Teilnehmern zusammen – der Client, sprich die Partei die eine bestimmte Anfrage bzw. ein Request hat, sowie der Server, der die Informationen oder auch Ressourcen zur Verfügung stellt. Die Abbildung A.7 verdeutlicht die Stellung des Servers gegenüber den innerhalb des Netzwerks

befindlichen Clients. Jeder Client kann unabhängig von den anderen Teilnehmern eine Anfrage an den Server stellen. Der Server hingegen beantwortet die Anfragen und

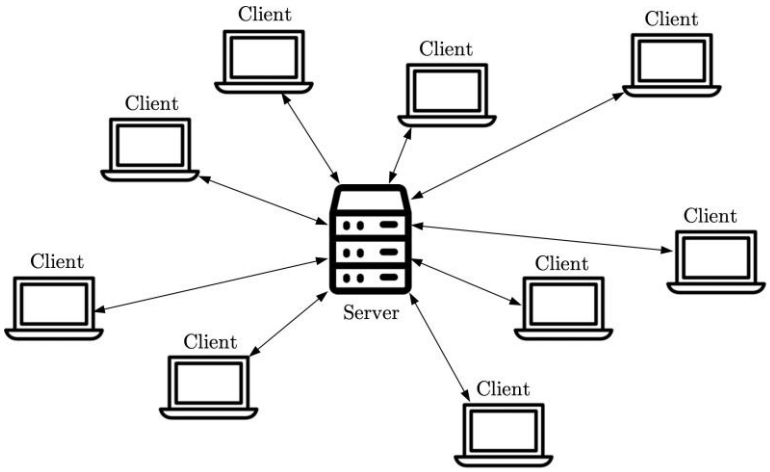


Abbildung A.7: Client-Server Architektur

stellt anschließend die benötigten Informationen oder Ressourcen zur Verfügung. Die Reihenfolge nach der die Anfragen beantwortet werden, obliegt dem Server selbst. Alle Teilnehmer bzw. Clients sind gleichberechtigt, es darf demnach niemand priorisiert oder benachteiligt werden [Net].

Das Problem, das sich jedoch aus der Client-Server Architektur heraus ergibt ist, dass der Server eine möglichst hohe Zuverlässigkeit gegenüber Ausfällen oder Problemen jeglicher Art aufweisen muss. Fällt demnach z.B. der Server aus, so ist das gesamte Netzwerk durch dessen Abhängigkeit zum Server nicht mehr arbeitsfähig. Um dieses Problem zu umgehen, werden Server-Hochverfügbarkeitslösung innerhalb des Netzwerks eingeplant, so dass bei einem Ausfall des Servers, andere Backup-Hardware eingesetzt werden kann. Die Daten zwischen den primären Servern und der Backup-Hardware muss hierzu jedoch immer auf dem aktuellen Stand gehalten werden, da ansonsten die Daten inkonsistent sind.

Ein weiterer Nachteil ist die Einschränkung der Anfragen durch die Clients, um den Server nicht zu überlasten. Das bedeutet jedoch gleichzeitig, dass die Prozesse des Clients oftmals schneller arbeiten könnten als der Server die Informationen oder Ressourcen über-

haupt zur Verfügung stellen kann. Der Server stellt demnach je nach Kapazität einen Flaschenhals im System dar.

Auch die Skalierbarkeit zeigt sich bei der Client-Server Architektur als negativer Aspekt, da hier der Server nur auf eine bestimmte Anzahl an Clients ausgelegt ist. Melden sich mehr Clients am Server an, so kann unter Umständen der Zugriff auf diesen zusammenbrechen und die Funktion nicht mehr gewährleistet werden. Dieses Problem kann zwar durch den Einsatz mehrerer Server eingedämmt werden, jedoch sind dann komplexe Verfahren zur Lastverteilung auf den unterschiedlichen Servern notwendig [ITF19].

A.4.9 Peer-to-Peer Architektur

Die am häufigsten verwendete Architektur im Bereich von Blockchain-Technologien stellt das Peerto-Peer Protokoll dar. Innerhalb dieses Verfahrens tauschen verschiedene Rechner in einem Netzwerk Daten untereinander aus, ohne dass ein Server hierfür notwendig ist. Die Abbildung A.8 verdeutlicht hierzu das eben genannte Prinzip.

Grund für den Einsatz der Peer-to-Peer (P2P) Architektur sind die drei wesentlichen Gesichtspunkte

Skalierbarkeit, Sicherheit sowie Zuverlässigkeit und Flexibilität. Vor allem die Skalierbarkeit spielt in internetbasierten Applikationen der Zukunft eine entscheidende Rolle, da durch immer größere

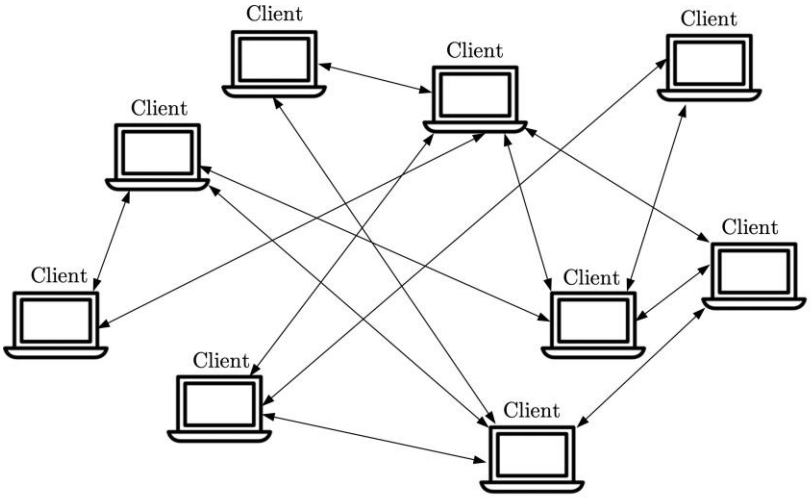


Abbildung A.8: Peer-to-Peer Architektur

Nutzerzahlen auch steigende Anforderungen an die Rechenleistung, Speicherkapazität, etc. gestellt werden. Auch die Sicherheit und Zuverlässigkeit stellen zwei wichtige Anforderungen an das Netzwerk dar, die durch die P2P Architektur umgesetzt werden können. Durch den Wegfall eines zentralen Servers wird neben der Erhöhung der Sicherheit auch die Zuverlässigkeit positiv beeinflusst, da ein technischer Ausfall eines Teilnehmers nahezu keine Auswirkungen auf das Gesamtnetzwerk hat. Durch die sich hieraus ergebenden Vorteile spricht man auch bei einem P2P Netzwerk von einer sogenannten dezentralen Ressourcennutzung, die die homogene Nutzung von Rechenleistung und Speicher im gesamten Verbund beschreibt, sowie die Selbstorganisation – die einzelnen Teilnehmer agieren direkt untereinander ohne eine zentrale Instanz [SL19].

Insgesamt lassen sich zwei grundlegende Ansätze bezüglich der Netzform erkennen, die sich in strukturierte und unstrukturierte Netze untergliedern. Innerhalb unstrukturierter Netzwerke wird jeder Node zufällig mit einem anderen Node verbunden, so dass in Summe eine logische Mesh-Topologie entsteht, die den Vorteil einer hohen Robustheit gegenüber Angriffen und Ausfällen mit sich bringt. Grund hierfür ist, dass sich durch dieses Prinzip kein zentraler Angriffspunkt ausbildet und durch die zufälligen Verbindungen auch viele Redundanzen innerhalb des Netzwerks entstehen. Bei einer strukturierten Anordnung der Nodes ist die Ordnung der einzelnen Teilnehmer hingegen festgelegt. Auch welcher Node welche Informationen führt und weitergibt ist vorgegeben,

wodurch der Informationsfluss innerhalb einer vordefinierten Zeit ermöglicht wird. Eine solche Netzform kann mithilfe sogenannter Distributed Hash Tables (DHT) abgebildet werden [SL19].

A.4.10 Netzwerkzugriffe

Neben der Typologie des Netzwerkes stellt sich bei Blockchain-Anwendungen zudem die Frage, wie mit Zugriffsrechten umgegangen wird. Speichert z.B. ein Unternehmen wichtige und sensible Daten über eine Blockchain, so bietet es sich an, dass diese nur für Mitglieder des Unternehmens zugänglich sind. Einen Überblick über die beiden Arten von Zugriffsrechten soll der nachfolgende Unterabschnitt bieten.

A.4.11 Öffentlicher Zugriff

Geht es vor allem um das Thema Kryptowährungen, so wird primär eine öffentliche Blockchain präferiert, wobei alle Teilnehmer unabhängig von den jeweiligen Rechten vollen Zugriff auf das Netzwerk erhalten. Jeder kann demnach das Netzwerk betreten und auch wieder verlassen. Zum Betrieben eines Nodes wird daher oftmals eine Open-Source-Software kostenlos zur Verfügung gestellt, über die die Teilnehmer Zugriff auch die Blockchain erhalten. Da sich die Teilnehmer untereinander nicht kennen und jeder Zugang zur Blockchain erhalten kann, werden Konsens-Algorithmen, wie in Abbildung A.2.2 beschrieben, eingesetzt [SL19].

Alle Informationen innerhalb des Netzwerks sind bei einem öffentlichen Zugriff frei zugänglich und daher besonders bei transparenten Applikationen sinnvoll, die vor Angriffen oder Manipulationen geschützt werden sollen [SL19].

A.4.12 Privater Zugriff

Bei einer Blockchain, die durch einen privaten Zugriff beschränkt ist, können externe Teilnehmer nicht das Netzwerk betreten. Der Zugang wird hierfür mittels eines Netzwerk-Operators, der die Rolle des Administrators einnimmt, bereitgestellt oder mithilfe spezieller Richtlinien innerhalb des Blockchain-Protokolls. Da hier eine Zentralisierung auf eine bestimmte Organisation erfolgt, kann der Verwalter bzw. Administrator der Blockchain

auch Einfluss auf die Blöcke oder Transaktionen nehmen – was dem eigentlichen Grundgedanken einer Blockchain bzgl. Dezentralität und Gleichberechtigung aller User widerspricht. Da jeder Zugang kontrolliert wird, kann davon ausgegangen werden, dass keine Angriffe bzw. Manipulationen an den Daten erfolgen, da es sich um ein vertrauenswürdiges Netzwerk handelt [SL19].

A.4.13 Permissioned- und Konsortiumszugriff

Der Hybrid aus den beiden bereits genannten Zugriffsformen stellt der Konsortiumszugriff in Kombination mit dem Blockchain Typ Permissioned dar, der hauptsächlich im Bereich von Unternehmen eingesetzt wird. Hierbei bleibt das Konzept bezüglich einem kontrollierten und verifizierten Zugriff bestehen, jedoch wird der eigentliche Betrieb im Vergleich zu einem privaten Zugriff nicht durch eine zentrale Autorität kontrolliert. Demnach wird der Hauptaspekt der Blockchain bezüglich der Dezentralität bei diesem Zugriffsverfahren weiterhin bewahrt. Um die Verwaltung der Zugriffsrechte nicht manipulieren zu können, wird ein bestimmter Verbund (Konsortium) eingesetzt, der die Verwaltung der Rechte übernimmt. Auch hier werden Konsens-Algorithmen angewendet, die jedoch in diesem Fall ausschließlich von den Mitgliedern des Konsortiums betrieben werden, wodurch die Konsensbildung an sich schneller vorangetrieben werden kann. Auch einzeln können die beiden Zugriffskonzepte angewendet werden [SL19].

A.4.14 Plattformen

Die beiden bekanntesten Plattformen, die auf der beschriebenen Distributed-Ledger-Technologie aufbauen, werden innerhalb des nun folgenden Abschnitts näher beschrieben. Beides stellen heutzutage etablierte Kryptowährungen im Finanzsektor dar, die aufgrund deren ersten und zweiten Ranglistenplatzes innerhalb der größten Kryptowährungen einen entscheidenden Marktanteil mit sich bringen.

Die Abbildung A.9 zeigt die Marktkapitalisierung (Stand 2019) anhand eines Diagramms. Die Kryptowährung Bitcoin macht laut diesem den größten Anteil mit rund 56 % aus. Mit einem deutlich geringeren Anteil von 11 % folgt die Ethereum-Plattform. Ripple und Bitcoin Cash bilden mit 7 % bzw. 3 % das Schlusslicht der Rangliste. Innerhalb des letzten Blocks Other befinden sich alle Plattformen, die aufgrund ihres geringen Marktanteils

nicht mehr in die Auflistung mit aufgenommen wurden. Hierunter zählen z.B. EOS, Litecoin, Binance Coin, etc.

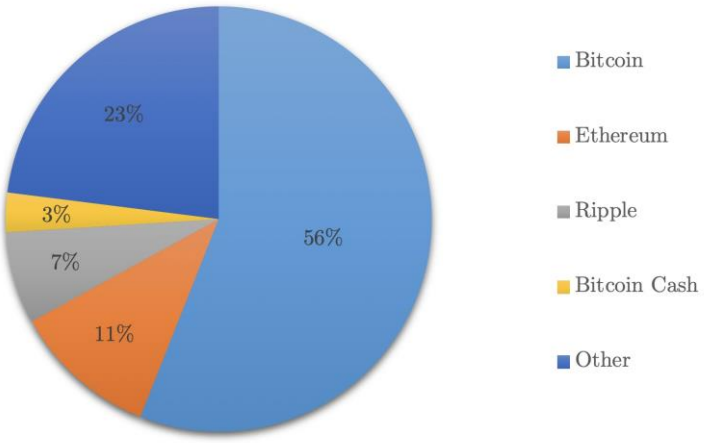


Abbildung A.9: Marktkapitalisierung der Kryptowährungen [Zah]

A.4.15 Bitcoin

Die 2009 auf dem DLT-basierten Grundgedanken entwickelte Idee von Satoshi Nakomoto, baut auf der Speicherung von Transaktionen innerhalb einer verschlüsselten Netzwerk-Kette auf. Die Transaktionen zielen dabei hauptsächlich auf den Kauf- und Zahlungsverkehr ab, die innerhalb eines dezentralen Netzwerks mithilfe von Minern validiert und der Kette hinzugefügt werden. Durch den öffentlichen Zugang des Bitcoin-Netzwerks kann jeder den Open-Source Client herunterladen und so direkt am Mining-Prozess teilnehmen.

Um innerhalb des Netzwerkes auf einen Konsens zu kommen, wird der Proof-of-Work-Mechanismus verwendet. Der Knotenpunkt, der den Hash als schnellstes berechnet, wird für dessen Errechnung entlohnt. In Bezug auf die Währungspolitik ist das Bitcoin-Konzept im Vergleich zu vielen anderen Kryptowährungen deflationär aufgebaut. Daher ist die maximale Anzahl an Bitcoins, die über das Netzwerk ausgegeben werden können, auf 21 Millionen Coins beschränkt. Wirft man einen Blick auf andere Fiat-Währungen, worunter z.B. auch die Plattform Ethereum fällt, so werden hier ständig neue Währungspakete ausgege-

ben, die sich neben einer Vielzahl weiterer Faktoren ebenfalls negativ auf die Inflation der Währung auswirken [Bit19].

A.4.16 Ethereum

Im Vergleich zur Plattform Bitcoin stellt Ethereum kein reines System zum Kauf- und Zahlungsverkehr dar. Vielmehr erweitert dieses Protokoll den Blockchain-Ansatz um zusätzliche Aspekte, die im November 2013 erstmalig vorgestellt wurden. Aufgrund der Erweiterung des bisherigen BlockchainProtokolls wird daher auch von der Blockchain zweiter Generation gesprochen. Die Währung, die als Belohnung für die zur Verfügung gestellte Rechenleistung dient, wird als Ether bezeichnet.

Durch den Einsatz von Ethereum soll das sogenannte Internet of Value ermöglicht werden. Innerhalb des neuartigen Protokolls sollen demnach programmierte Codes, auch Smart Contracts genannt, implementiert werden, die sich durch das Eintreten bestimmter Bedingungen selbst ausführen. Hierdurch wird es ermöglicht, dass sogar ganze Applikationen über die Blockchain gehostet werden können. Im Fachjargon wird daher bei solchen Applikationen auch von decentralized apps, kurz dApps, gesprochen. Auch ganze Organisationen können so über Decentralized Autonomous Organizations rein mittels Smart Contracts und dApps realisiert werden [Bit19].

ie Smart Contracts, die auch im weiteren Verlauf des hier vorliegenden Projekts näher thematisiert und angewendet werden, stellen selbst ausgeführte Verträge dar, die auf Basis bestimmter Ereignisse ausgeführt werden bzw. in Kraft treten. Innerhalb dieser intelligenten Verträge können automatisch und ohne die Aufsicht Dritter unabhängiger Personen, Transaktionen und Aktionen zwischen Parteien durchgeführt werden, die sich untereinander nicht kennen und dementsprechend auch nicht zwingend vertrauen. Nach Abschluss der Bedingungen kann eine weitere Transaktion oder z.B. auch ein weiterer Smart Contract angestoßen werden, was den Aufbau ganzer Strukturen und Organisationen ermöglicht, die unabhängig von Zensuren, Ausfallzeiten, Betrug oder Störungen durch Dritte sind. Das Vertrauens-Problem zwischen den unabhängigen Parteien bzw. Vertragspartner wird demnach durch das Wenn-Dann-Prinzip gelöst – nur wenn alle Bedingungen des Vertrags erfüllt worden sind, wie die Übermittlung des Geldes sowie die Bereitstellung der vereinbarten Gegenleistung, kann die Transaktion abgeschlossen werden. Durch die zusätzliche Validierung mithilfe des Netzwerks, wird der

Übermittlungsvorgang zusätzlich vor Manipulationen und unrechtmäßigen Änderungen innerhalb des Vertrags geschützt [Eth].

A.4.17 Ethereum 2.0

Die bisher größte Weiterentwicklung in der Geschichte der Plattform Ethereum, soll über die zweite Generation, auch Ethereum 2.0 oder ETH2, ausgerollt werden. Der neue Iterationsschritt zeichnet sich hierbei vor allem durch zwei wesentliche Änderungen im Vergleich zur ersten Generation der Bitcoin- Alternative aus. Als wahrscheinlich wichtigster Punkt soll der Konsens-Mechanismus Proof-of-Work abgelöst werden. Damit verabschiedet sich die Plattform von der klassischen Form des Mining. Hingegen soll das Staking über den bereits beschriebenen Algorithmus Proof-of-Stake als Alternative verwendet werden. Durch diese Umstellung soll die Plattform deutlich energieeffizienter gestaltet werden können und somit Ressourcen einsparen [Rau20].

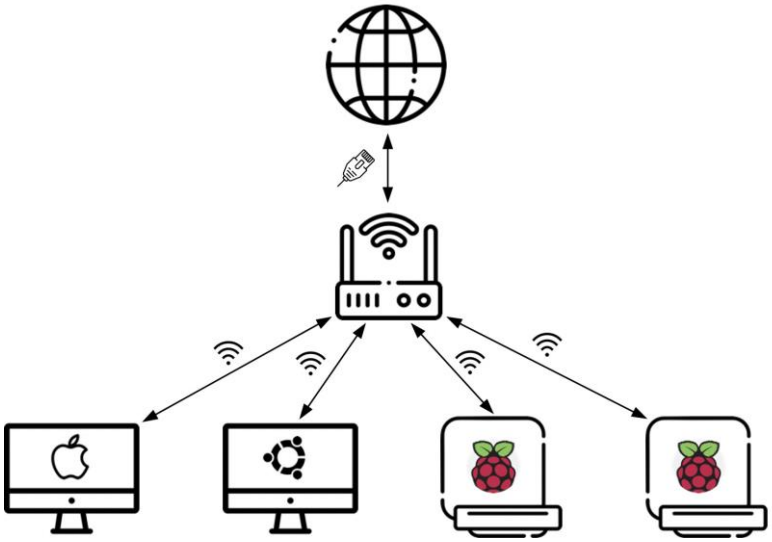


Abbildung A.10: Übersicht des aufgebauten Netzwerks

Ein weiterer Aspekt der zweiten Entwicklungsstufe stellt das sogenannte Sharding dar. Durch dieses Verfahren wird die Blockchain in mehrere Shards unterteilt, die sich somit auf bestimmte Anwendungsbereiche spezialisieren. Neben der zusätzlich deutlich

erhöhten Transaktionsrate auf einige tausend Transaktionen pro Sekunde, soll so eine verbesserte Skalierbarkeit erreicht werden [Rau20].

A.5 Benötigte Hardware

Um die in diesem Projekt beschriebene Software, insbesondere zum Betrieb der eigenen privaten Blockchain auf Basis von Ethereum, zu installieren, muss auch eine bestimmte Hardware zur Verfügung stehen. Die in diesem Fall beschriebene Hardware dient nur als exemplarisches Beispiel und kann demnach auch durch andere Komponenten ersetzt werden.

Abbildung A.10 zeigt den spezifischen Aufbau, der innerhalb des Projekts verwendet wird. Der Router stellt somit eine zentrale Stelle für die vier unterschiedlichen Teilnehmer dar und verbindet die jeweiligen Knoten mit dem Internet, in diesem Fall per WiFi. Um ein möglichst breites Spektrum an unterschiedlichen Betriebssystemen auf den Nodes zu verwenden, wird ein Computer unter macOS genutzt, ein virtuelle Maschine mit Linux Ubuntu und zwei Raspberry Pi mit dem Betriebssystem RaspberryOS. So ergeben sich ingesamt vier unabhängige Nodes, die innerhalb des BlockchainNetzwerks teilnehmen.

Um einen reibungslosen Ablauf in der Kommunikation zwischen den einzelnen Teilnehmern zu ermöglichen, soll demnach zuerst auf die Netzwerkkonfiguration der an der Blockchain beteiligten Knoten eingegangen werden.

A.5.1 Netzwerkkonfiguration

Für einen möglichst reibungslosen Ablauf mit den einzelnen Teilnehmern der Blockchain, bietet es sich an, zunächst einen Blick in die eigene Netzwerkkonfiguration des heimischen Routers zu werfen. Vor allem statische IP-Adressen erleichtern den Umgang zur Kommunikation zwischen den Teilnehmern. Da es von Zeit zu Zeit passieren kann, dass sich ohne eine statische Vergabe der Adressen die Kommunikationspfade zu den einzelnen Netzwerkknoten ändern, bietet sich diese Option zur Vermeidung von Verbindungsschwierigkeiten an. Viele Router besitzen eine solche Funktionalität bereits standardmäßig.

Abbildung A.11: Raspberry Pi 4 [Pan19]

A.5.2 Computer und Betriebssystem

Der für das Projekt verwendete Computer kann unabhängig von dessen Betriebssystem gewählt werden, da softwareseitig alle gängigen Systeme unterstützt werden. Da hier jedoch die Kompatibilität mit Linux-basierten Systemen sowie macOS am höchsten ist, werden die nachfolgenden Beschreibungen bzw. Anleitungen auch für diese beiden Systeme ausgelegt.

Bezüglich der Computer-Hardware erleichtert ein leistungsstarker Rechner auch den Workflow mit der Blockchain. Mindestvoraussetzung für den Betrieb einer geth-Node ist eine 64-Bit-Architektur sowie eine Grafikkarte mit mindestens einem Gigabyte an Arbeitsspeicher. Da die meisten Rechner heutzutage diese Voraussetzungen mehr als nur erfüllen, sollte der Betrieb einer privaten Blockchain nahezu auf jeder Hardware möglich sein [Min17].

Soll jedoch innerhalb einer öffentlichen Blockchain am Mining-Prozess teilgenommen werden, reicht ein normaler Rechner bzw. auch ein High-Performance Endgerät heutzutage nicht mehr aus, um effektiv minen zu können. Daher sind für das Mining spezielle Rechner bzw. Chipsätze entwickelt worden, sogenannte ASIC-Chips, die einzig und allein für die spezielle Anwendung des Lösens von Rechenoperationen zur Errechnung des Hash-Wertes hergestellt werden.

Da die Berechnungen innerhalb einer privaten Blockchain, die in erster Linie nur für Forschungszwecke genutzt wird, bereits innerhalb kurzer Zeit durchlaufen werden können, ist eine gängige StandardHardware vollkommen ausreichend.

A.5.3 Raspberry Pi

Wie bereits in Abbildung A.10 erwähnt, werden insgesamt zwei Raspberry Pis in das Netzwerk eingebettet. Somit können zwei Teilnehmer als Nodes simuliert werden. Die Abbildung A.11 zeigt einen solchen Mini-Computer. Die hier eingesetzte vierte Version des Raspberry Pis bietet gegenüber dem Vorgänger eine deutlich gestiegene Rechenleistung. Vor allem jedoch der vergrößerte RAM-Speicher von 4 GB ermöglicht somit eine flüssige und komfortable Verwendung des Raspberry Pi [Rap].

Mittels eines USB-C Steckers kann das Raspberry Pi mit Strom versorgt werden. Durch die intern vorhandene WiFi-Antenne kann mühelos eine Verbindung mit dem Internet hergestellt werden. Als Betriebssystem wird Raspberry Pi OS, ehemals Raspbian, verwendet. Soll der Raspberry Pi nur über einen anderen Computer bedient werden, so kann auch die Lite Version des Betriebssystems installiert werden [Gro].

A.6 Benötigte Software

Neben einiger zusätzlicher Software, die für die Nutzung oder auch Installation bestimmter Programme notwendig ist, soll in diesem Abschnitt die Software zum Betrieb bzw. auch zur initialen Aufsetzung einer Blockchain auf Basis des Ethereum-Protokolls vorgestellt werden.

A.6.1 Geth

Die Software Go Ethereum, auch oft einfach nur unter dem Namen Geth bekannt, basiert Google Sprache Go und ist neben C++ und Python eine der drei ursprünglichen Implementierungen des Ethereum-Protokolls [GoE]. Geth ist unter anderem eine der ausgereiftesten Umsetzungen der Ethereum-Knoten-Software.

Neben Geth kann z.B. auch die direkte Konkurrenz Parity Ethereum zur Verbindung mit dem Hauptnetzwerk von Ethereum, einer Reihe von Testnetzwerken, wie Ropsten oder Rinkeby, oder benutzerdefinierten Netzwerken verwendet werden [Bey19]. Da sich das hier vorliegende Projekt vor allem auf die Nutzung eines eigens erstellen Netzwerks zu Forschungszwecken bezieht, ist die Verbindung zum Hauptnetzwerk Ethereum bzw. auch zu anderen Testnetzwerken, wie z.B. Rinkeby oder Ropsten nur von geringerer Bedeutung.

Im Gegensatz zu Parity bietet Geth keinen umfassenden Support für privatwirtschaftliche Blockketten, jedoch Unterstützung für andere Web3-Konzepte, wie z.B. dezentrales privates Messaging und dezentralen Filestore. Vor allem mittels der Web3-Anwendungen sollen später die Smart Contracts realisiert werden, die es den Teilnehmern ermöglicht, Transaktionen durchzuführen und mit anderen intelligenten Verträgen auf der Ethereum-Blockkette zu interagieren [Bey19][GoE].

A.6.2 Puppeth

Das bereits im Go Ethereum Paket enthaltene Tool Puppeth, stellt einen Ethereum Private Network Manager dar, der die Interaktion und vor allem die Initialisierung einer privaten Blockchain deutlich vereinfacht. Über das Tool kann hierbei eine neue Etereum Blockchain aufgesetzt sowie der Genesis- Block, sprich der initiale Block der Kette, konfiguriert werden. Mit dem anschließenden Export der Genesis-Konfiguration kann die Blockchain mithilfe der genesis.json aufgesetzt werden. Neben den klassischen Konfigurationsparametern, die bereits in Abbildung A.2.1 thematisiert wurden, können hier auch ETH-Accounts vorerstellt sowie mit der Währung Ether aufgeladen werden [Hus18].

A.6.3 Netstats

Da die bisher beschriebene Software größtenteils nur auf Befehlen innerhalb des Terminals basiert, ermöglicht die Erweiterung Netstats eine Visualisierung der privaten Blockchain. Auch über den Private Network Manager Puppeth kann ein Netstats-Server bereits während der Initialisierung aufgesetzt werden. Wird die Blockchain gestartet, so kann das Netstats-Dashboard über eine bestimmte Adresse im Browser aufgerufen und dargestellt werden.

Die Abbildung A.12 zeigt die Visualisierung einer aktiven Ethereum Blockchain. Neben den aktiven Nodes können Informationen zu den zuletzt gemineten Blöcken aufgerufen werden. Zum Beispiel findet sich hier auch der zeitliche Verlauf der letzten benötigten Berechnungen, der Schwierigkeitsgrad, die letzten Transaktionen sowie die durchschnittliche Zeit für das Minen eines Blocks [Hen17].

Auch für private Netzwerke macht eine solche Visualisierung Sinn, da hierdurch die Aktivität der Blockchain überwacht werden kann. Vor allem als Einstieg in die Thematik liefert Netstats informative

Abbildung A.12: Ethereum Netstats [Hen17]

Einblicke in die Technologie Ethereum und zeigt, was hinter den einzelnen Befehlen innerhalb des Terminals steckt.

A.6.4 Truffle

Truffle ist eine Entwicklungsumgebung, ein Test-Framework und eine Asset-Pipeline in einem. Es basiert auf der Ethereum Blockchain und wird primär dazu verwendet, eine reibungslose Entwicklung von dApps zu erleichtern. Durch die Umgebung können im Terminal Smart Contracts, unabhängig von der Solidity-Version, kompiliert und auf der

Blockchain deployed werden. Zusätzlich ermöglicht Truffle auch ein umfangreiches Testing von neuen Smart Contracts, das durch die Integration des auf Java Script basierten Tools Jasmine Test Framework ermöglicht wird [Tru20]

A.6.5 Web3.py

Mithilfe Web3.py können Clients erstellt werden, die mit der Ethereum Blockchain kommunizieren und somit Daten aus der Blockchain auslesen, neue Transaktions-Daten schreiben, Smart Contracts ausführen sowie deren Ergebnisse über die Programmier-sprache Python interpretieren. Die Basis für die Kommunikation stellt das JSON RPC (Remote Procedure Call) Protokoll dar. Web3.py ermöglicht es, Anfragen an einen einzelnen Ethereum-Knoten im Namen des gesamten Netzwerks mit JSON RPC zu stellen. Auf diese Weise können über einen einzigen Knoten Daten im Netzwerk gelesen und in das Netzwerk geschrieben werden. Dies ist vergleichbar mit HTTP-Anforderungen an eine JSON-API, die über Webserver gestellt werden [McC].

In diesem Kapitel werden einige Themen vorgestellt, die zum Verständnis der Arbeit wichtig sind. Dabei handelt es sich im Wesentlichen um die Blockchain-Technologie und der Stand der Vorarbeit.

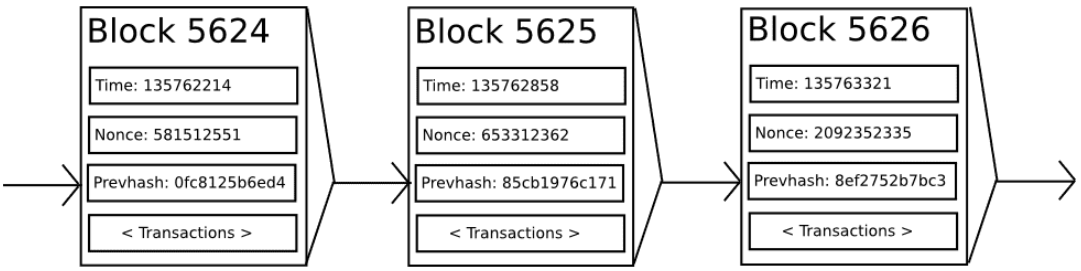


Abbildung A.13: Aufbau Blockchain Daten [But20]

Auszug aus Studienarbeit Marco Arena, Jim Rebholz, WiSe 2020/2021

B

Stabilität von elektrischen Strom-netzen und Netzmodellierug

Wie bereits in der Einleitung erwähnt unterliegt das elektrische Stromnetz immer größeren Herausforderungen [NLZ+04]. Eine besonders große Rolle spielt dabei die zunehmende Einspeisung von regenerativen Energien, die durch den Ausstieg aus Kohle und Atomkraftwerken immer wichtiger werden [RA04]. Eine weitere Belastung stellen die Privatverbraucher selbst dar. Immer mehr elektronische Verbraucher finden Einzug in Wohnungen, in denen Sie nicht direkt gesteuert werden können. Sie werden vom Nutzer

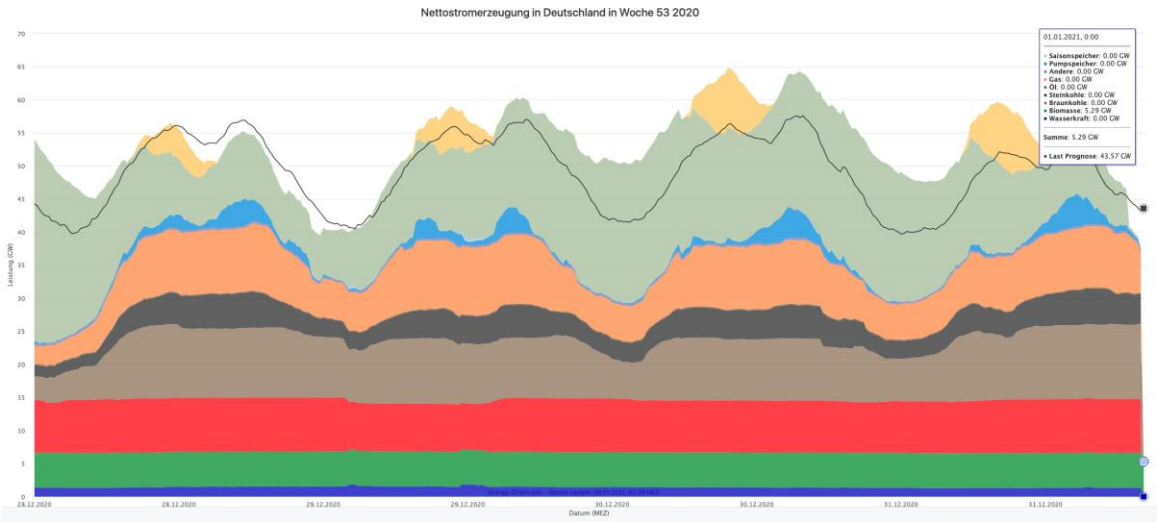


Abbildung B.1: Stromerzeugung und Verbrauch in Deutschland für KW53 im Jahr 2020 [Fra].

nach Belieben an das Stromnetz angeschlossen. Neben den Privatverbrauchern spielt auch die Industrie mit einem erheblichen Energieverbrauch eine bedeutende Rolle. Dort ist der Energieversorger allerdings in engem Kontakt zu den jeweiligen Unternehmen, so dass dieser sogar in der Lage ist, einzelne Anlagen vom Stromnetz zu trennen, um das Netz zu stabilisieren [Bre].

Zur Veranschaulichung der Herausforderungen an das Stromnetz ist in Abbildung B.1 der Verlauf bei der Energieerzeugung für die KW53 im Jahr 2020 dargestellt. Dementsprechend handelt es sich um einen sehr aktuellen Verlauf. Die Abbildung zeigt die großen Schwankungen im Verbrauch, die nur durch eine Regelung kompensiert werden können. Dabei lassen sich die Schwankungen im Verbrauch auf die unterschiedlichen Tageszeiten zurückführen. Allerdings gibt es auch erhebliche

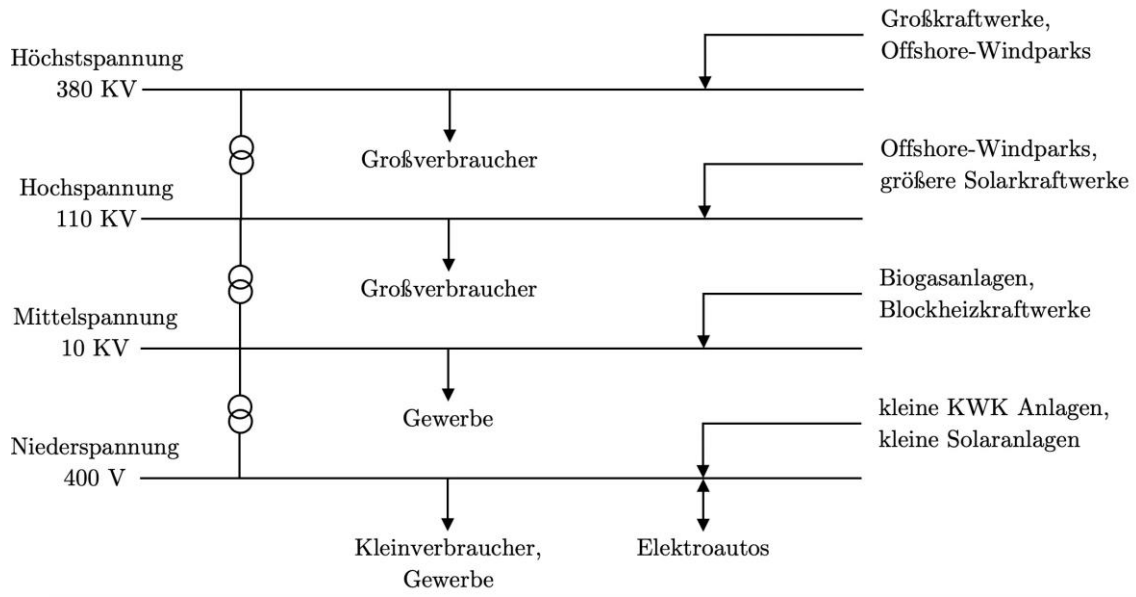


Abbildung B.2: Topologie der elektrischen Netzwerke im Verbundsystem. (Eigene Darstellung)

Schwankungen bei der Energieerzeugung. Im Folgenden werden Konzepte gezeigt, die als Maßnahmen gegen die starken Stromnetzbelastungen Einsatz finden.

B.1 Topologie der elektrischen Netzwerke

Bevor in die Stabilität von elektrischen Stromnetzen genauer analysiert wird, zeigt dieser Abschnitt den grundlegenden Aufbau mit den verschieden Spannungsebenen. In Abbildung B.2 ist der Aufbau des europäischen Verbundsystems in einer vereinfachten Form dargestellt. Das Ziel dieses Verbundsystems ist der Transport von elektrischer Energie über sehr lange Strecken in ganz Europa. Damit sollen Verbrauchsschwankungen und Ausfälle von Kraftwerken kompensiert werden. Aufteilen lässt sich das Stromnetz in vier Spannungsebenen. Jede dieser Ebenen wird im Folgenden kurz mit der dazugehörigen Funktionalität beschrieben.

- **Höchstspannung:** Auf dieser Ebene findet die Verteilung also der Ferntransport der el. Energie statt. Bedingt durch die weiten Distanzen und die hohen Spannungen werden hierfür spezielle Hochspannungsleitungen eingesetzt. Der Effektivwert der Spannung liegt bei 220 kV oder 380 kV und sorgt dafür, dass hohe Mengen an Energie mit geringen Verlusten übertragen werden. Größere Energiekraftwerke sind direkt an das Höchstspannungsnetz gekoppelt und speisen die Leistung über Transformatoren ein.

- **Hochspannungsebene:** Die nächste Ebene ist mit einer Effektivspannung von meist 110 kV die Hochspannungsebene. Daran sind z.B. die Leitungen der Deutschen Bahn gekoppelt und kleinere Kraftwerke können ihre Energie hier einspeisen. Genau wie das Höchstspannungsnetz übernimmt die Hochspannungsebene die Verteilung der elektrischen Energie.

- **Mittelspannung:** Auf der Mittelspannungsebene ist der Effektivwert je nach Region unterschiedlich. Mit einem Bereich von 10-30 kV ist die Spannung aber schon deutlich geringer als auch der Hochspannungsebene. Bereits jetzt befindet sich das Stromnetz auf der ländlichen bzw. städtischen Ebene. Direkte Abnehmer sind z.B. Krankenhäuser, Stadtwerke oder große Fabriken.

- **Niederspannung:** Bei der Niederspannungsebene ist der Effektivwert der Spannung bereits auf 230 V pro Phase abgesunken. Zwischen den

unterschiedlichen Phasen des Drehstromnetzes bestehend aus L1, L2 und L3 lässt sich eine Effektivspannung von 400 V messen. Die Niederspannungsebene dient dazu die Energie schlussendlich auf die einzelnen Verbraucher bzw. Häuser aufzuteilen. Dabei ist die Energie vor Ort an den diversen Steckdosen ohne weitere Transformatoren nutzbar.

Der Transport und die Transformation der Spannungslevel zwischen den einzelnen Spannungsebenen führt zu Energieverlusten, so dass unnötig langer Transport vermieden werden sollte. Die genaue Höhe des Verlusts ist durch einige Parameter zu beeinflussen. Jede elektrische Leitung besitzt einen elektrischen Widerstand, welcher durch das verwendete Material nicht zu vermeiden ist. Durch den Widerstand wird ein Teil der Energie beim Transport in Wärmeenergie umgewandelt und geht an die Umwelt verloren. Für die Leitung stellt diese Erwärmung eine Gefahr dar. Bestimmte Leitungen, die unterhalb der Erde verlegt sind, könnten durch die abgeführte Wärme schmelzen und ein klassischer Kurzschluss entsteht. Aber auch auf Freilandleitungen darf die Leitung nicht beliebig warm werden, da auch diese Schaden davon nimmt. Um die ohmschen Verluste möglichst gering zu halten, wird das Spannungslevel stark erhöht. Über den hohen Spannungspegel ist es möglich große Leistungen bei geringer Stromstärke zu übertragen. Durch die geringe Stromstärke verbessern sich die Wärmeverluste. Allerdings muss der Sicherheitsabstand der Leitungen beachtet werden. Mit steigender Spannung erhöht sich das Risiko für einen elektrischen Überschlag in Form eines Blitzes. Ein weiterer Anteil an Energie geht durch Blindströme verloren, die sich bei der Übertragung von Wechselstrom nicht vermeiden lassen. Ebenfalls nicht zu verhindern sind Verluste an den elektrischen Bauteilen, die in einem Stromnetz vorhanden sind. Dazu zählen zum Beispiel Gleichrichter, Wechselrichter und Transformatoren.

B.2 Regelenergie zum Ausgleich von Belastungen

In Europa und damit auch in Deutschland beträgt die Netzfrequenz im Verbundnetz genau 50 Hz. Für die Stabilität des Stromnetzes spielt die aktuelle Netzfrequenz eine erhebliche Rolle. Sie kann als direkte Qualitätskontrolle angesehen werden. Die Frequenz repräsentiert dabei das Verhältnis aus der erzeugten Leistung durch Kraftwerke und der entnommenen Leistung durch den Verbraucher im Netz. Das Stromnetz an sich kann

keine Energie speichern. Eine mögliche Art der Speicherung von Energie sind große Akkumulatoren, die allerdings sehr teuer sind und sich deshalb nicht bezahlt machen.

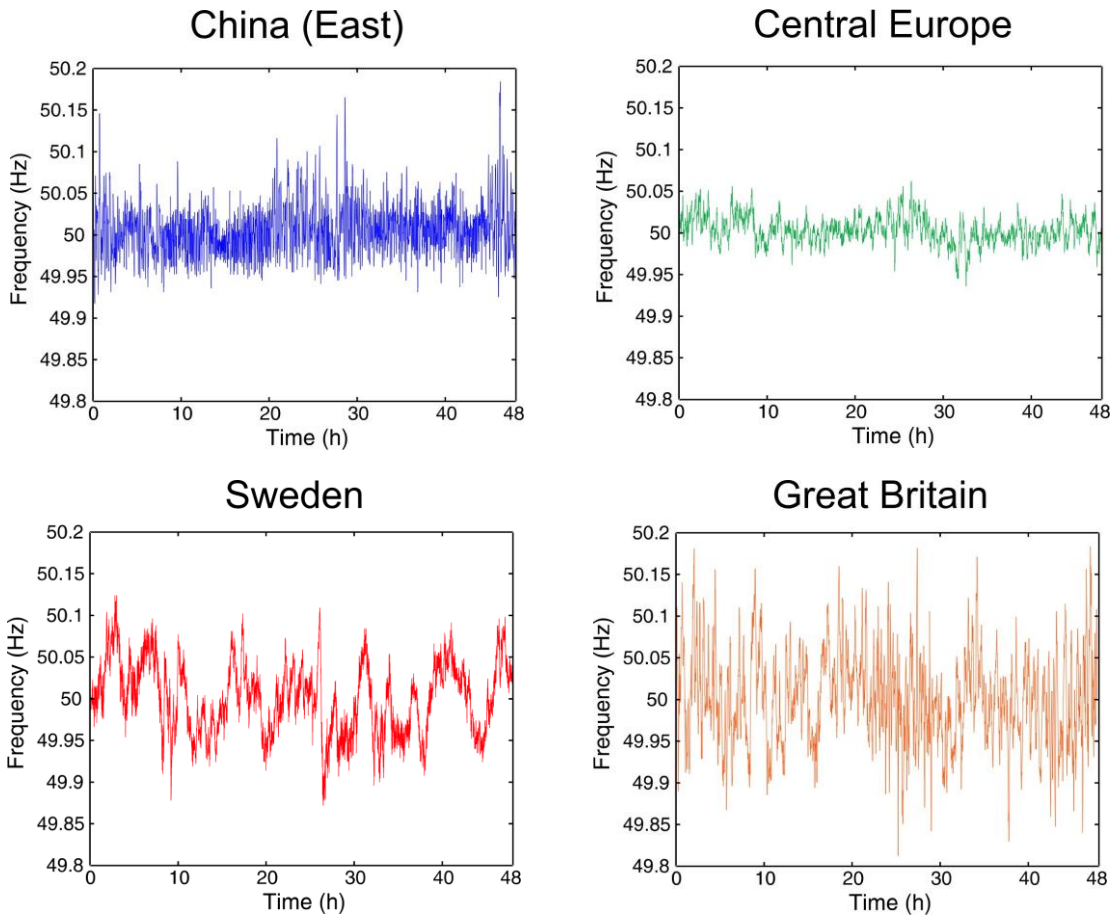


Abbildung B.3: Schwankungen der Netzfrequenz für ein Zeitfenster von 48 h für vier verschiedene Regionen [WIK].

Es gibt allerdings auch Alternativen, bei denen die elektrische Energie in eine andere Energieform transformiert wird. Das könnte zum Beispiel ein Pumpspeicherkraftwerk sein. Durch das schwierige Abspeichern von Energie muss die Ausgangsleistung deshalb zu jedem Zeitpunkt der Eingangsleistung entsprechen. Abweichungen in der Frequenz können in großen Stromausfällen resultieren, weshalb die Überwachung dieser eine wichtige Aufgabe für den Netzbetreiber darstellt.

Erzeugt wird der Strom, der durch das Verbundnetz fließt, in großen Kraftwerken. Dazu verwenden die Energieversorger große Generatoren, die durch eine Energiequelle in Rotation versetzt werden. Die Rotationsgeschwindigkeit des Generators resultiert in einer dazugehörigen Frequenz. Eine steigende Geschwindigkeit der Rotoren führt zu einer erhöhten Frequenz und umgekehrt. Die Netzfrequenz selbst hängt aber nicht nur von der Rotationsgeschwindigkeit, sondern auch von der Energie ab, die dem Netz zum aktuellen Zeitpunkt entnommen wird. Ist zu einem bestimmten ein Energieüberschuss im Netz vorhanden, so steigt die Netzfrequenz leicht an. Liegt ein Energiemangel vor, sinkt sie leicht ab. Geht der Netzbetreiber nicht aktiv gegen das Absinken der Netzfrequenz vor, führt die Situation zu einem Netzausfall [DLB+12]. Im Regelbetrieb sind diese Abweichungen minimal und belaufen sich bei gerade einmal $0{,}2$ Hz [SIF07]. Besonders herausfordernd beim Regeln einer stabilen Frequenz sind die Regenerativen Energien, die einen immer größer werdenden Anteil an Leistung im Gesamtnetz übernehmen. Im Prinzip ist eine Vorhersage der zur Verfügung stehenden regenerativen Energie möglich. Dennoch sind diese Prognosen nur in kurzen Zeitintervallen zuverlässig, so dass keine langfristige Vorplanung möglich ist.

In Abbildung B.3 ist der zeitliche Verlauf der Netzfrequenz für Europa und China, Schweden und Großbritannien dargestellt. In dem Schaubild ist zu erkennen, dass sich die Abweichungen in allen Regionen innerhalb der Rahmenbedingungen gehalten haben. Dennoch ist die Standardabweichung der Netzfrequenz in China deutlich höher als in Europa. Noch größere Schwankungen der Netzfrequenz finden sich in Schweden und Großbritannien wieder. Damit die Schaubilder diesen Verlauf annehmen können sind Schutzmaßnahmen nötig. Dafür gibt es die sogenannten Regelenergiearten [HP12]. Der Sinn dieser Regelenergie ist der Ausgleich von Schwankungen im Stromnetz. In Europa ist ein System mit vier verschiedenen Regelenergiestufen vorhanden.

Diese vier Regelenergiestufen sind:

1. Primärregelung
2. Sekundärregelung
3. Tertiärregelung (Minutenreserve)
4. Quartärregelung (Stundenreserve)

In Abbildung B.4 ist der zeitliche Verlauf beim Beziehen von Regelenergie dargestellt. Die Quartärregelung entspricht dabei dem Bilanzausgleich. Im Folgenden werden die einzelnen Stufen mit deren Funktion beschreiben.

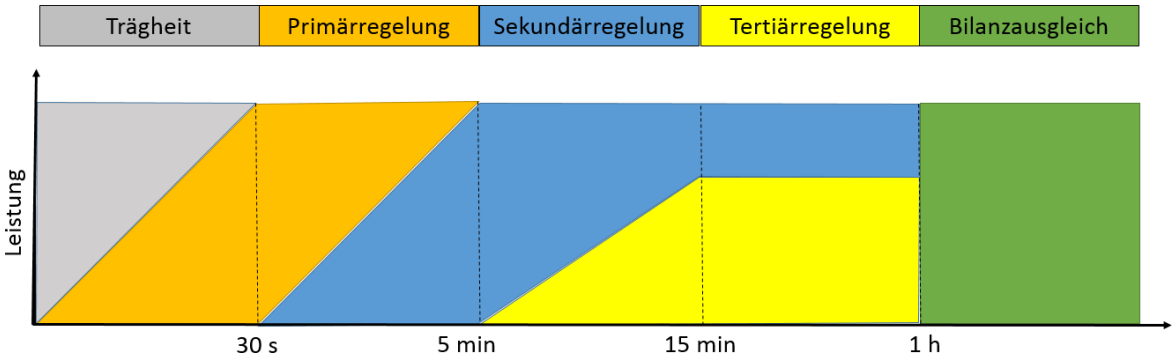


Abbildung B.4: Überblick des zeitlichen Verlaufs beim Beziehen von Regelleistung [Kam10].

B.2.1 Primärregelung

Treten Ungleichgewichte im Stromnetz auf wird die Primärregelung zur Kompensation eingesetzt. Dafür stehen in Deutschland spezielle Kraftwerke bereit. Für die Primärregelung kommen nur Kraftwerke in Frage, die innerhalb von weniger als 30 Sekunden ihre Regelleistung zur Verfügung stellen können. Die Auslastung der Kraftwerke wird bewusst niedrig gehalten, so dass diese in der Lage sind positive als auch negative Regelleistung zur Verfügung zu stellen. Bereits bei kleinen Frequenzabweichungen im Bereich von ± 10 mHz greift die Primärregelung und die Kraftwerke passen ihre Leistungen an. Richtig Ausgleichen kann diese Form der Regelenergie die Frequenzabweichung allerdings nicht. Es handelt sich hierbei um einen P-Regler, wodurch die die Regelleistung proportional zur Frequenzabweichung steigt. Ein P-Regler allein ist nicht in der Lage die Regelabweichung im System auf den Wert Null zu setzen. Folglich müssen die weiteren Regelleistungen eingreifen, um die Netzfrequenz wieder auf den exakten Wert von 50 Hz einzustellen. Da sich die Ausgangsleistungen der Kraftwerke nur grob Einstellen lassen ist die Primärregelung nicht dafür geeignet, kleine Abweichungen zu kompensieren. Sie ist eher als Begrenzung gedacht, um das Stromnetz vor einem Ausfall zu Schützen und die Netzfrequenz stabil zu halten.

B.2.2 Sekundärregelung

Die Primärregelung muss immer zur Verfügung stehen. Deshalb muss diese bei bleibenden Abweichungen entlastet werden. Da kommt die Sekundärregelung ins Spiel. Sie sorgt dafür, dass die Primärregelung wieder in ihren Normalbetrieb zurückkehren kann. Das Ziel der Sekundärregelung ist die Frequenz wieder auf den Sollwert von 50 Hz zu setzen. Sie ist als I-Regler implementiert und ist damit in der Lage bleibende Regelabweichungen zu kompensieren. Genau wie bei der Primärregelung gibt es spezielle Sekundärregelkraftwerke. Diese befinden sich in verschieden Zonen des Stromnetzes und können direkt vor Ort den Schwankungen entgegenwirken. Treten kleine Abweichungen im Rahmen von weniger als ±10 mHz auf, übernimmt in Normalfall die Sekundärregelung

Abbildung B.5: Schematischer Aufbau eines Virtuellen Kraftwerkes [Vir].

vor der Primärregelung, da sie sich viel feiner steuern lässt. Das hat zur Folge, dass die Abweichungen sehr schnell ohne Abweichung und Überschwingen kompensiert werden.

B.2.3 Teritärregelung und Quartärregelung

Genau wie die Primärregelung muss auch die Sekundärregelung nach einer gewissen Zeit von ihrer Last befreit werden, um für den nächsten Einsatz bereit zu sein. Dafür gibt es die Teritärregelung, die auch unter dem Synonym Minutenreserve bekannt ist. Im Gegensatz zu den vorherigen Regelungen geht die Minutenreserve nicht automatisch in den Betrieb, sondern muss manuell eingeschaltet werden. Die Minutenreserve kommt nur sehr selten zum Einsatz, da die Primär- und Sekundärregelung im Normalfall die Stabilität der Netzfrequenz gewährleisten.

Die Quartärregelung beschäftigt sich mit der langfristigen Einhaltung der Netzfrequenz im Stromnetz. Trotz der vielen Maßnahmen ist die Frequenz im Mittel nie ganz genau bei 50 Hz. Da die Netzfrequenz häufig als Taktgeber für Synchronuhren eingesetzt wird, können vorübergehend Abweichungen der Uhren von mehreren Sekunden $\pm$ auftreten. Überschreitet die Abweichung 20 s kommt die Quartärregelung zum Einsatz und passt die Netzfrequenz leicht an, bis die sogenannte Netzzeit wieder mit der Weltzeit übereinstimmt. Langfristig ist damit eine genaue Zeitbasis mit kurzfristigen Abweichungen gegeben.

B.3 Virtuelle Kraftwerke

Wie bereits in der Einleitung erwähnt, reicht der Einsatz von Regelenergie nicht aus, um das Stromnetz auch zukünftig bei dem steigenden Anteil von regenerativen Energien vor Überlastung zu schützen. Ein Ansatz zur Kompensation des Problems ist der Einsatz eines Virtuellen Kraftwerks. Ein virtuelles Kraftwerk bezeichnet den Zusammenschluss von dezentralen Erzeugern und Verbrauchern im Stromnetz. Die Teilnehmer des Virtuellen Kraftwerks sind dabei miteinander verbunden. Über diese Verbindung ist eine Kommunikation untereinander möglich. Teilnehmer können unter anderem Windkraft-, Photovoltaik-, Biogas- oder Wasserkraftanlagen sein ebenso wie (große) Abnehmer, die ihre Last flexibel ändern können.

In Abbildung B.5 ist ein vereinfachter Aufbau eines Virtuellen Kraftwerks dargestellt. Auf der rechten Seite ist das Virtuelle Kraftwerk mit seinen verschiedenen Energieerzeugern dargestellt. In der Mitte aller dezentralen Erzeuger ist eine zentrale Einheit, die alle Vorgänge innerhalb des Kraftwerks überwacht und steuert. Durch den Zusammenschluss

der Erzeugungsanlagen entstehen wirtschaftliche Vorteile wie zum Beispiel die Teilnahme an der Strombörse (siehe Abbildung). Durch die Kommunikation mit dem Netzbetreiber lässt sich außerdem Flexibilität anbieten, die mit einer ordentlichen Vergütung belohnt wird.

Durch den hohen Anteil an regenerativen Energiequellen innerhalb des Netzwerks haben Virtuelle Kraftwerke eine Schwäche gegenüber herkömmlicher Energieerzeugung durch fossile Brennstoffe. Die Versorgung kann nicht zu jedem Zeitpunkt garantiert werden. Die Erzeugung von Solar-/ und Windkraftenergie ist für einen Zeitraum von mehreren Stunden ziemlich präzise. Langfristige Prognosen sind allerdings mit größeren Unsicherheiten belastet. Infolgedessen ist es wichtig, dass im Virtuellen Kraftwerksverbund genug Erzeuger integriert sind, die nicht auf äußere Einflüsse angewiesen sind. Das kann zum Beispiel eine Biogasanlage sein. Ein gesunder Mix an verschieden Kraftwerkstypen ist also grundlegend für ein verlässliches Virtuelles Kraftwerk. Bedingt durch die Struktur sind Virtuelle Kraftwerke nicht in der Lage, sich am Regelenergiekreis zu beteiligen. Das gilt für die Primärregelung als auch für die Sekundärregelung und Minutenreserve, da die zur Verfügung stehende Leistung zu gering ist. Dennoch eignen sich die dezentralen Energieerzeuger dazu kurzfristig die benötigte Ausgleichsenergie bereitzustellen. Ein Energietransport ist an sich ebenfalls möglich. Durch die bereits erwähnten Verluste beim Transport ist dieser allerdings nicht wirtschaftlich. Aus diesem Grund ist es sinnvoll, die einzelnen Kraftwerke relativ nahe zueinander zu platzieren.

Auszug aus Studienarbeit Marco Arena, WiSe 2020/2021

C

Implementierung der Netzmodellierung im Falle eines Engpasses

C.1 Smart Grids

Ein Smart Grid ist ein Stromnetz welches durch das Zusammenspiel vieler intelligenter Teilnehmer gesteuert wird. Die Teilnehmer können je nach Art verschiedene Charakteristiken aufweisen. Gängige Teilnehmer sind beispielsweise Erzeuger, Verbraucher, Speicher oder Verteilnetze. Um eine intelligente Steuerung innerhalb des Netzes zu realisieren wird neben dem Verteilpfad für die Energie ein weiteres Netz für

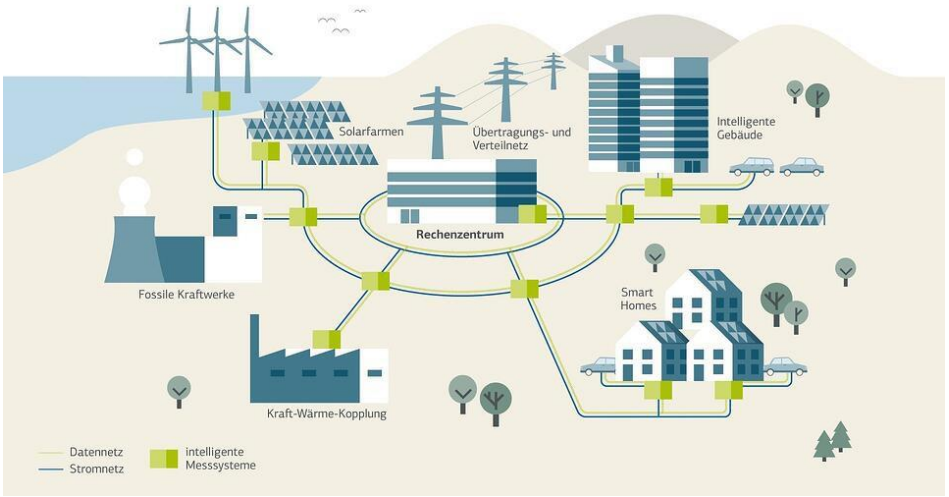


Abbildung C.1: Smart Grid Beispiel [KHK17]

den Datenaustausch der einzelnen Teilnehmer benötigt. Ein schematisches Smart Grid ist in Abbildung C.1 dargestellt. Dort sind die einzelnen Teilnehmer zu erkennen sowie die

Pfade zur Energieund Datenübertragung. Die in der Abbildung als intelligente Messsysteme bezeichneten Geräte sind sogenannte Smart Meter diese sind in Abschnitt C.1.2 beschrieben.

C.1.1 Motivation

Durch einen immer größer werdenden Anteil von erneuerbaren Energien und gleichzeitig einer steigenden Anzahl von Kleinsterzeugern müssen neue Konzepte zur dezentralen Regelung des Stromnetzes gefunden werden. Bei der Regelung von Stromnetzen sind nur geringe Margen für Fehler möglich, so darf die Frequenz von 50 Hz $\pm$ des Drehstromnetzes in Deutschland nur um 0,2 Hz $\pm$ abweichen. Dafür werden in Deutschland etwa 3000 MW an Regelleistung bereitgehalten. Diese Zusammenhänge werden durch die Betrachtung von Abbildung C.2 besonders deutlich.

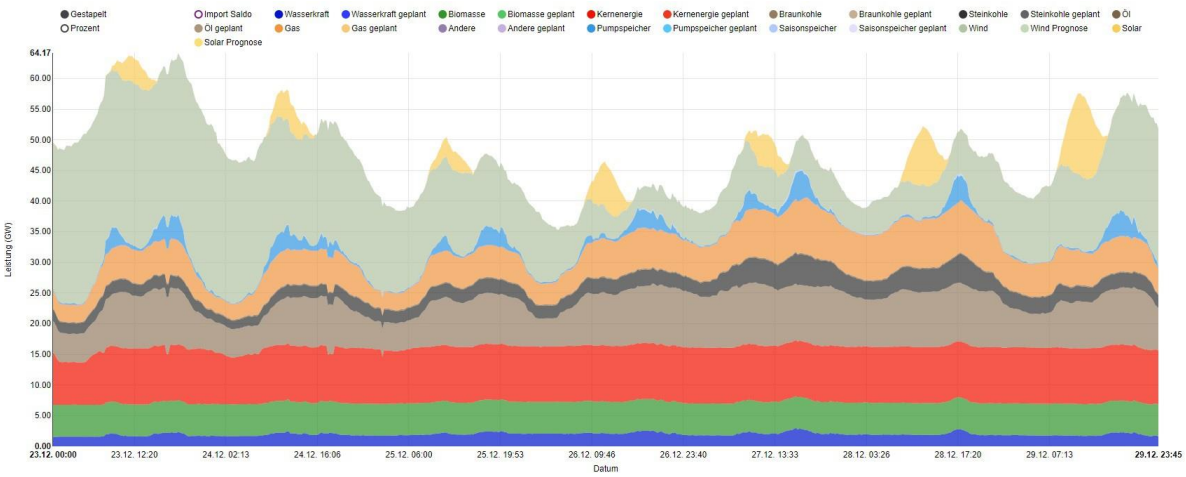


Abbildung C.2: Strommix Deutschland KW52 2019 [HAT+20]

Die Abbildung zeigt den Verlauf des deutschen Strommix in KW52 in 2019. Deutlich zu erkennen ist nicht nur die tägliche Fluktuation des Energieverbrauchs und die dahin bestehend notwendige Regelung, sondern auch eine starke Schwankung der erneuerbaren Energieträger. Die Schwankungen sorgen für erhöhten Regelaufwand bei schlecht regelbaren Großkraftwerken welche die Energie mit fossilen Energieträgern erzeugen. Diese Zusammenhänge führen zu erhöhten Kosten bei den Energieversorgern und Netzbetreibern, die Implementierung von Smart Grids könnte hier eine dezentrale

Lösung schaffen welche auch für die Endkunden durch niedrigere Strompreise von Vorteil wäre.

C.1.2 Smart Meter

Ein Smart Meter (intelligenter Zähler) ist ein Stromzähler welcher kommunikationsfähig ist und eine Intelligenz zur Datenverarbeitung besitzt. Smart Meter sind für die Umsetzung von Smart Grids unabdinglich. Diese stellen das Bindeglied zwischen Verbrauchern und Stromnetz dar. In Abbildung

sind diese grün dargestellt, zu erkennen ist, dass dadurch jeder Netzteilnehmer über eine Kommunikationsanbindung an das Datennetz des Stromnetzes verfügt. Die Stromzähler können auf Tarifänderungen oder den jetzigen und zukünftigen Stromverbrauch reagieren. Durch die Wichtigkeit ihrer Aufgabe müssen Smart Meter äußerst gut abgesichert sein und manipulationssicher gestaltet sein. Zusammenfassend kann gesagt werden, dass ohne den flächendeckenden Einsatz von Smart Metern kein autonomes Ressourcenund Lastmanagement sowie die daraus folgende Regelung möglich ist.

C.2 Implementierung

Im zweiten Teil dieser Arbeit werden die Erkenntnisse und Fortschritte des ersten Masterprojekts genutzt, um die Regelung der Netzauslastung in Smart Grids durchzuführen. Dafür ist zu Beginn das

System mit seinen dazugehörigen Teilnehmern beschrieben. Im Anschluss wird ein Smart Contract entwickelt, der in der Lage ist die Netzauslastung zu regeln. Außerdem wird auf das dazugehörige Matlab Simulinkmodell eingegangen, das ebenfalls einen großen Bestandteiles des Projekts darstellt. Nach einer ausführlichen Anleitung zur Inbetriebnahme des Gesamtsystems folgt die Diskussion der Ergebnisse.

C.3 Systembeschreibung

Über die Systembeschreibung sollen die Teilnehmer als auch deren Verbindungen zueinander visualisiert werden, um den Umfang der Projektarbeit zu definieren. In Abbildung C.3 ist der vereinfachte Systemaufbau dargestellt. Insgesamt umfasst das System drei verschiedene Aktoren. Der erste Aktor ist der Erzeuger, der im Schaubild als Kraftwerk abgebildet ist. Beim Erzeuger handelt es sich um einen Erzeuger, der in der Lage ist, seine Ausgangsleistung anzupassen. Wie bereits in den Grundlagen erwähnt, sind viele der Generatoren nicht in der Lage ihre Ausgangsleistung kurzfristig

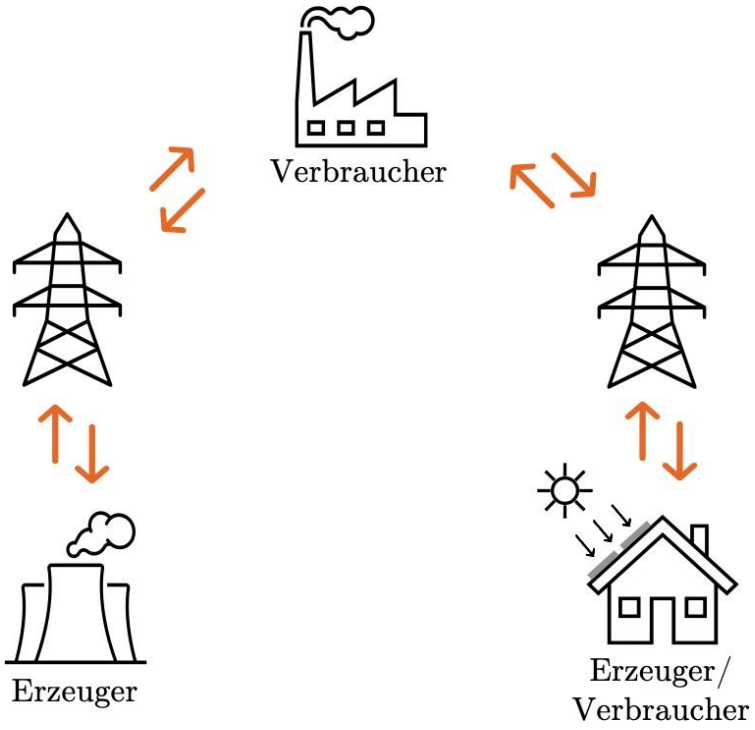


Abbildung C.3: Aufbau des Gesamtsystems mit den dazugehörigen Teilnehmern.

anzupassen, da sie selbst und auch das System träge sind. Um die Anzahl der Teilnehmer im Rahmen zu halten ist hier nur ein einzelner Erzeuger enthalten. Innerhalb von zukünftigen Projektarbeiten ist allerdings auch die Integration von statischen Generatoren durchaus denkbar. Durch das Stromnetz ist der Erzeuger im System an den ersten Verbraucher gekoppelt. Dieser Verbraucher repräsentiert eine Vielzahl an gewerblichen Unternehmen. Für die Beschreibung des Verbraucherverhaltens wird das Standardlastprofil (SLP) G0 verwendet. Dieses wird von dem Bundesverband der

Energieund Wasserwirtschaft (BDEW) zur Verfügung gestellt. Bei den Lastprofilen handelt es sich um einen vereinfachten Verlauf, der durchschnittlich von den jeweiligen Verbrauchergruppen bezogen wird.

Die Bezeichnung G0 steht für ein allgemeines Gewerbe. Neben G0 gibt es auch weitere SLPs, die spezielle Arten von Unternehmen betrachten. Auf den Industrieverbund folgt eine Ansammlung von

Haushalten, die sowohl Erzeuger als auch Verbraucher darstellen. Zu Beginn sind die Haushalte und die darin enthaltenden elektrischen Geräte als Verbraucher im System zu betrachten. Allerdings werden die Häuser mit Solaranlagen bestückt, um den steigenden Anteil an regenerativen Energien im Netz zu berücksichtigen. Übersteigt die erzeugte Energie der Solaranlagen den Verbrauch der Haushalte zu einem bestimmten Zeitpunkt, wird aus dem Verbraucher ein Erzeuger, der die überflüssige Leistung in das Netz einspeisen kann. Für die Simulation der Last ohne Solaranlage wird das Profil H0 (privater Haushalt) gewählt. Zur Realisierung der Solaranlage wird ein angepasstes Profil aus Alternative Energien 2 herangezogen.

Neben dem Systemaufbau müssen vor der Implementierung weitere Variablen definiert werden.

Dazu zählen z.B. die Grenzen des Systems. Es werden folgende Annahmen getroffen:

- Die maximale Ausgangsleistung des variablen Erzeugers beträgt 150 kW.
- Alle Teilnehmer sind in Reihe miteinander verschaltet.
- Das Stromnetz ist in der Lage maximal 100 kW Leistung zu transferieren.
- Zu jedem Zeitpunkt soll die erzeugte Leistung gleich der verbrauchten Leistung entsprechen.

Bei den beschriebenen Kenngrößen handelt es sich um fiktive Werte, die nicht mit der Realität in Verbindung stehen. Sie dienen lediglich dazu, dass System eindeutig für die anschließende Implementierung zu beschreiben. Daher können die Zahlenwerte beliebig angepasst werden, sofern die Änderungen auch in der Simulation berücksichtigt werden.

Bedingt durch die Reihenschaltung ist zu beachten, dass sich die Leistungen der Verbraucher addieren. In Summe darf die Leistung der beiden Verbraucher weder die maximale Erzeugerleistung noch die maximal übertragbare Leistung im Stromnetz übersteigen. Da die Projektaufgabe die Regelung der Netzauslastung vorsieht, ist die maximale Erzeugerleistung bewusst größer als die maximale Übertragungsleistung des Netzes gewählt. Begrenzt ist die Übertragungsleistung im Netz durch den Leitungsquerschnitt. Mit zunehmender Leistung steigt auch die Temperatur der Leitung an. Zu jeder Tageszeit muss die Temperatur der Leitungen des Netzes innerhalb eines bestimmten Bereichs liegen, um Schäden an den Komponenten durch zu hohe Temperaturen zu vermeiden. Um das Netz stabil zu halten ist es wichtig, dass die eingespeiste Leistung zu jedem Zeitpunkt der entnommenen Leistung entspricht. Nur so kann die Netzfrequenz von 50 Hz konstant gehalten werden. Kommt es zu Engpässen soll anhand von Prioritäten, die den Verbrauchern zugeordnet sind, die Verteilung der Energie übernommen werden.

C.4 Entwickeln des Smart Contract

Mithilfe des Smart Contracts soll die Lösung für das im vorherigen Abschnitt gezeigte System realisiert werden. Dieser Smart Contract muss so gestaltet sein, dass die verschiedenen Teilnehmer darauf zugreifen können, um mit diesem zu kommunizieren.

C.4.1 Anforderungen an den Smart Contract

Im Folgenden sind die Anforderungen an den Smart Contract und dessen Umfang aufgelistet. Jede der Anforderungen muss durch den Inhalt des Smart Contracts erfüllt werden.

- Der Industrieverband muss in jedem Zeitzyklus seinen aktuellen Leistungsbedarf an den Smart Contract melden. Dieser Leistungsbedarf berücksichtigt keine Leistungseinbrüche durch ein überlastetes Stromnetz. Der tatsächliche Wert berechnet sich aus dem G0 SLP und einem Faktor zur Skalierung, um den Leistungspegel auf die internen Systemvariablen anzupassen.

- Die Haushalte melden genau wie die Industrie ihren aktuellen Leistungsbedarf in jedem Zeitzyklus. Dieser berechnet sich aus dem H0 SLP. Im Gegensatz zur Industrie muss allerdings berücksichtigt werden, dass der Leistungsbedarf unter Umständen negative Werte annehmen kann.

- Der Erzeuger bekommt die Leistung, die er erbringen soll, in jedem Zeitzyklus mitgeteilt. Diese setzt sich zusammen aus dem Leistungsbedarf der Industrie und der Haushalte abzüglich der Leistung durch die Solaranlage. Im Smart Contract muss stets überprüft werden, dass der Erzeuger dabei nicht über sein Limit hinaus belastet wird.

- Im Smart Contract soll die Gesamtleistung auf die verschiedenen Teilnehmer verteilt werden. Überschreitet der Gesamtbedarf die maximal übertragbare Leistung findet im Smart Contract selbst eine Entscheidung statt, die über die einzelnen Ausgangsleistungen auf Grundlage der dazugehörigen Prioritäten bestimmt. Die Prioritäten sollen dabei flexibel und zur Laufzeit anpassbar sein. Weist ein Verbraucher eine höhere Priorität auf bekommt dieser die benötigte Leistung komplett, während der andere Verbraucher einen Teil der Geräte vorübergehend abschalten muss. Um Schäden am Menschen und der Umwelt zu vermeiden, ist in zukünftigen Projekten auch die Einführung von Gerätabhängigen Prioritäten denkbar.

C.4.2 Klassendiagramm des Smart Contract

Da der Smart Contract bereits über 200 Zeilen Code umfasst ist die Darstellung der internen Variablen und Funktionen in einem Klassendiagramm von großem Nutzen. In Abbildung C.4 ist das Klassendiagramm für den Smart Contract mit dem Namen ElectricalGrid dargestellt. Viele der darin enthaltenen Methoden sind dabei nicht zwingend für die grundlegende Funktion von Relevanz. Sollen allerdings spezielle Anwendungsszenarien untersucht werden, kommen auch diese zum Einsatz. Im oberen Teil sind die privaten Variablen gelistet, auf die nur mithilfe der Set-/ bzw. Get-Methoden zugegriffen werden kann. Die Funktionen sind um unteren Teil des Klassendiagramms gelistet. Über das Plus bzw. Minus vor der jeweiligen Funktion wird die Sichtbarkeit der

Funktion von außerhalb beschrieben. Beim Deployvorgang kommt der Konstruktor zum Einsatz und initialisiert den Smart Contract mit seinen Initialwerten. Aufgerufen wird der Konstruktor nur einmalig und nicht bei jedem Start der Blockchain. Damit für jeden Simulationsdurchlauf die gleichen Voraussetzungen gelten, muss vor dem Start einmalig die reset-Funktion aufgerufen werden. Anschließend muss der Anwender abwarten, bis die Informationen im Smart Contract aktualisiert wurden. Je nach aktueller Länge der Blockchain und Leistung des Computers kann der Vorgang bis zu 10 Sekunden dauern, da unter Umständen alle Variablen ihren Wert verändern. Jedes Mal wenn eine

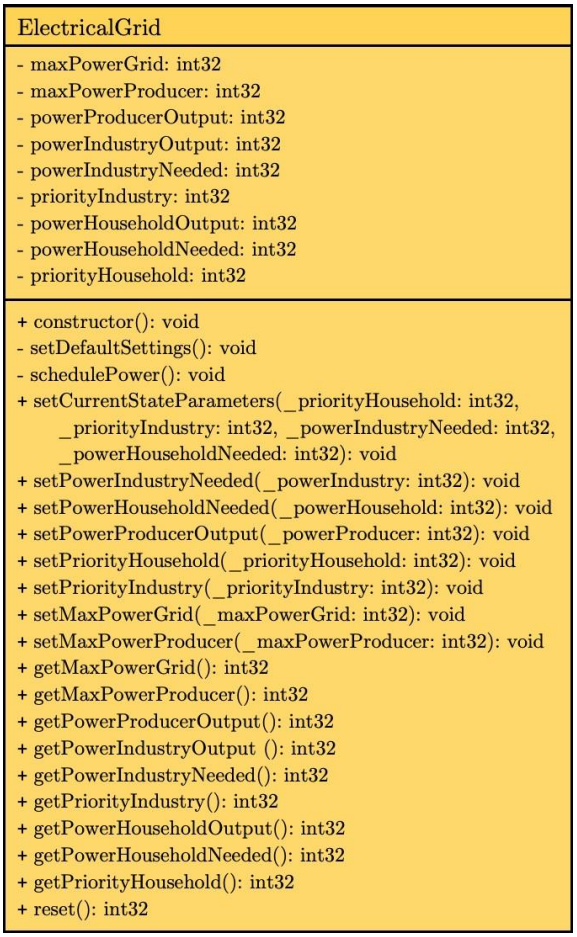


Abbildung C.4: Klassendiagramm des Smart Contracts ElectricalGrid.

Funktion aufgerufen wird, die Änderungen im Smart Contract verursacht (z.B. Set-Methoden), kommt es zu Verzögerungen in der Simulation. Darum ist die Funktion setCurrentStateParameters in dem Smart Contract integriert. Mit dieser können alle Eingangsvariablen auf einmal gesetzt werden, um zu vermeiden, dass viele

unterschiedliche Funktionsaufrufe die Dauer der Verzögerung erhöhen. Zu DEBUG Zwecken sind dennoch Funktionen vorhanden, die es erlauben einzelne Variablen anzupassen. Für die im weiteren Verlauf der Arbeit gezeigten Simulationsergebnisse sind diese aber nicht von Relevanz. Der Aufruf von Get-Methoden ist bei der Blockchain mit keiner Verzögerung verbunden, da der Smart Contract nicht angepasst wird. Ein Lesezugriff auf die interne Variable ist demnach nicht zeitkritisch, weshalb zu jeder Variable eine eigene Get-Funktion gehört.

Den größten Anteil des Smart Contracts übernimmt die Funktion schedulePower. In dieser steckt die eigentliche Intelligenz des Algorithmus. In schedulePower berechnet der Smart Contract die

Ausgangsleistungen auf Grundlage der internen Variablen und Eingangsvariablen. Innerhalb der Funktion werden unter anderem die Aspekte Prioritäten, Netzauslastung und die maximale Ausgangsleistung des Erzeugers berücksichtigt, um die zur Verfügung stehende Leistung fair zu verteilen. Neben der Verteilung der Leistung betrachtet die Funktion in jedem Zeitschritt außerdem die maximal übertragbare Leistung, um Schäden im Stromnetz durch thermische Energie zu vermieden. Kann die benötigte Leistung zu einem beliebigen Zeitpunkt nicht übertragen werden, bekommt der Verbraucher mit der höheren Priorität die benötigte Leistung. Alle Verbraucher mit einer niedrigeren Priorität erleiden ein Leistungsdefizit. Sind die Verbraucher gleich hoch priorisiert erfolgt das Drosseln der Leistungen im Verhältnis zur benötigten Leistung, so dass keiner der Verbraucher benachteiligt wird.

C.5 Aufbau der Simulation

Das Gesamtsystem zur Simulation besteht aus mehreren Teilnehmern, die zur Laufzeit miteinander interagieren. Für eine einwandfreie Interaktion müssen daher Schnittstellen zu den jeweiligen Teilnehmern implementiert werden. In Abbildung C.5 ist der schematische Aufbau der Simulation dargestellt. Das Modell umfasst fünf verschiedene Teilnehmer bzw. Komponenten.

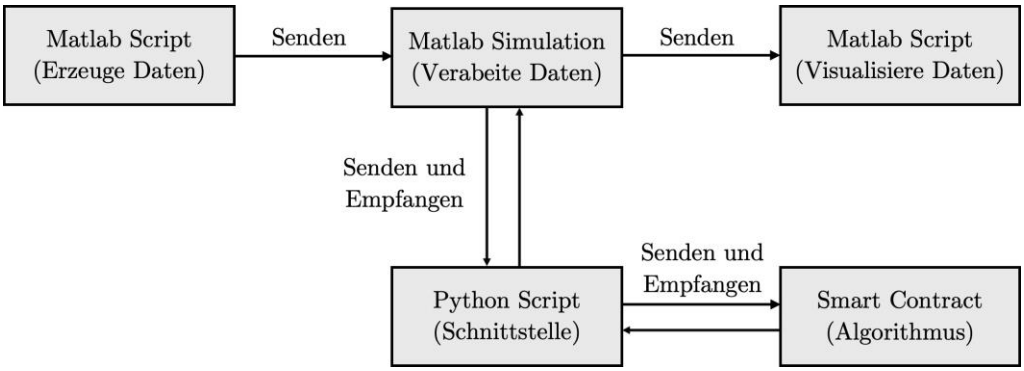


Abbildung C.5: Aufbau und Zusammenspiel der einzelnen Komponenten für die Simulation.

C.5.1 Simulationskomponenten in Matlab/Simulink

Zu Beginn der Simulation übernimmt ein kleines Matlab Script die Erzeugung der Daten, die im weiteren Verlauf als Simulationseingang genutzt werden. Zu den erzeugten Daten gehören:

- Verlauf der Prioritäten von Industrie und Haushalten
- Verlauf der benötigten Leistung von beiden Verbrauchern zu beliebigen Zeitpunkten (H0-/ bzw. G0-SLP)
- Ertrag einer optionalen Photovoltaikanlage
- Zeitvektor für eine Laufzeit von 48 h mit Zeitschritten von je 15 min
- Auswahl der verschiedenen Wochentage und Jahreszeiten

Nach erfolgreicher Erzeugung der Simulationsdaten kann die Simulation selbst über das Skript aufgerufen oder von Hand manuell gestartet werden. Die Photovoltaikanlage lässt sich über die Variable PV ein-/ bzw. ausschalten. Da der Code auf verschiedene Dateien zugreift, muss die Codeausführung aus dem Matlab Ordner erfolgen. Dieser ist in der Projektabgabe hinterlegt.

Die Simulation selbst ist in Simulink implementiert. Simulink beschafft sich die Variablen aus dem Matlabskript über FromWorkspace Blöcke und verarbeitet diese weiter. Mit UDP-Send Blöcken aus dem Package Simulink Support Package for Raspberry Pi Hardware sendet die Simulationsumgebung die anstehenden Daten an das Pythonskript. In diesen Blöcken muss die IPv4-Adresse des Empfängers und der dazugehörige Port angegeben werden. Dabei gilt es zu beachten, die Ports für die verschiedenen Pakete zu variieren, da es sonst zu Kollisionen kommt. Gleichzeitig findet das Empfangen von Daten statt, die von Python an die Simulation gesendet werden. Auch dafür bietet das Package einen Block. In dem Block UDP Recieve muss der Datentyp der ankommenden Pakete eingetragen sein. Außerdem ist auch wieder ein individueller Port für jede Variable, die empfangen werden soll anzugeben. Für alle Daten innerhalb der Simulation reicht der Datentyp int32 aus. Zur Analyse des Systemzustands werden zur Simulationszeit zusätzliche Berechnungen durchgeführt. Zum Schluss werden alle Variablen noch im Matlab Workspace abgespeichert. Das Abspeichern ist über den Simulationsblock ToWorkspace realisiert. Dieser hinterlegt den zeitlichen Verlauf der Signale in einem Array. Aus diesem Array können im Schlussteil der Simulation die dazugehörigen Ergebnisse visualisiert werden. Der Vorteil hierbei ist die gute Anpassbarkeit der Schaubilder. In Matlab Simulink selbst sind nur begrenz Möglichkeiten zur Erzeugung qualitativ hochwertiger Schaubilder

integriert. Jedes Schaubild aus Abschnitt C.7 ist daher auf dieselbe Art und Weise mithilfe eines weiteren Skripts erstellt. Dieser Teil des Skripts wird automatisch unmittelbar nach Abschluss des letzten Zeitschrittes aus Simulink ausgeführt.

C.5.2 Python Simulationsschnittstelle

Im Gesamtsystem spielt das Python Skript eine bedeutende Rolle als Schnittstelle. Python ermöglicht die Interaktion zwischen Matlab Simulink und dem Smart Contract. Leider bietet Simulink selbst keine Möglichkeit auf einen Smart Contract bzw. auf dessen Inhalte zuzugreifen. Wird diese Funktionalität zukünftig integriert, schrumpft der Umfang der Simulation und der Aufbau wird deutlich kompakter. Das Ziel dieses Abschnittes ist es, dem Leser das Pythonskript näher zu bringen, so dass es leicht für die eigene Funktion angepasst werden kann.

In Listing C.1 ist der erste Teil des Skripts dargestellt. Zu Beginn werden alle Pakete importiert, die für die Programmausführung relevant sind. Anschließend erfolgt die Definition einiger grundlegender Variablen. Dazu zählen unter anderem die IPv4-Adressen der Teilnehmer, als auch die Informationen zum Smart Contract. In der Variable IP_Ubuntu muss die IPv4 Adresse eingetragen sein, aus der dieses Skript aufgerufen wird. In diesem Fall ist das die virtuelle Maschine unter Ubuntu. Um eine Verbindung zum Smart Contract aufzubauen muss dessen Adresse als auch die ABI bekannt sein. In Abschnitt D.5.2 ist beschrieben, wie die Variablen bestimmt werden können.

```
1
2
from web3.auto import w3
from web3 import Web3 , HTTPProvider , IPCProvider
from web3.contract import ConciseContract
import     time
import   struct
import socket

# IP Address of Ubuntu
IP_Ubuntu = '192.168.178.91'
# IP Address of MacOS
IP_Simulink = '192.168.178.68'

# Address of Contract in Blockchain
CONTRACTADDRESS = '0x82e7908d2a932EFBA9a8783D1a3680dD570b5DBd'

# Contract ABI
CONTRACTABI = '[{"inputs":[],"payable":false,"stateMutability":"nonpayable","
    type":"constructor"},{"constant" .....]'
```

Listing C.1: Importieren der Pakte und definieren wichtiger Variablen über Python.

Mithilfe von Listing C.2 stellt Python die Verbindung zum Smart Contract her. In Zeile 6 ist es wichtig, dass der Anwender den IPC Pfad angibt, den die Konsole beim Starten der Blockchain im Terminal ausgibt. Auch dieser Prozess ist bereits im ersten Teil des Masterprojekts beschreiben. Läuft dieser Codeausschnitt ohne Fehlermeldung durch ist die gewünschte Verbindung vorhanden. Wirft das Skript einen Fehler liegt das erfahrungsgemäß an einer veralteten ABI oder Smart Contract Adresse. Damit der Anwender beim tatsächlichen Ausführen nicht den Überblick verliert, werden Textnachrichten an das Terminal gesendet.

```
1
def main():

#                   Set                     IPC                     Provider
print("*********************************************************")
print("--> INFO: Python Script Started")
web3 = Web3(IPCProvider('/home/marco/Library/Ethereum/geth.ipc'))

# Get coinbase and set default account
consumerAccount     =     w3.eth. coinbase
w3.eth.defaultAccount = consumerAccount
contract = w3.eth.contract(address=w3.toChecksumAddress(CONTRACTADDRESS), abi
    =CONTRACTABI , ContractFactoryClass=ConciseContract)
print("--> INFO: Successfully Connected to Blockchain")
```

Listing C.2: Aufbauen einer Verbindung zum Smart Contract über Python.

```
3
print('-->       INFO:       Init       required       UDP       Sockets')
sendOutputPowerHouseholdSocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
sendOutputPowerIndustrySocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
sendOutputPowerProducerSocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)

recieveNeededPowerHouseholdSocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
recieveNeededPowerHouseholdSocket.bind((    IP_Ubuntu    ,             20183))
recieveNeededPowerIndustrySocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
recieveNeededPowerIndustrySocket.bind((    IP_Ubuntu    ,             20182))
recievePriorityIndustrySocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
recievePriorityIndustrySocket.bind((    IP_Ubuntu    ,              20181))
recievePriorityHouseholdSocket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
recievePriorityHouseholdSocket.bind((IP_Ubuntu, 20180))
print("--> INFO: Initialized Required UDP Sockets")
```

Listing C.3: Aufsetzen der UDP-Sockets zur Kommunikation mit Simulink über Python.

Im nächsten Schritt müssen vor dem ersten Simulationsschritt alle UPD-Sockets initialisiert werden. Insgesamt umfasst das Skript sieben verschiedene UDP-Sockets, wovon drei Stück Daten an Simulink senden. Die restlichen nehmen ankommende Datenpakete von Simulink entgegen. Der Prozess beim Initialisieren läuft immer gleich

ab. In Listing C.3 ist die gewählte Konfiguration dargestellt. Wie bereits erwähnt hat jeder UDP-Socket seine eigene Portnummer. Bei den drei UDP-Sockets, die Daten aus Python

heraus versenden, ist kein Port angegeben. Dieser muss erst beim Aufruf der socket.sendto Funktion angegeben werden.

Bevor die Simulation innerhalb einer Endlosschleife starten kann, wird die Reset Funktion des Smart Contracts getriggert. Da sich durch den Aufruf der Funktion die internen Variablen ändern, muss die Coinbase übergeben werden. Dieser Account übernimmt die anstehenden Gaskosten in der Blockchain. Damit der Miner unter macOS genug Zeit hat diesen Reset auf der Blockchain zu verarbeiten ist eine Verzögerung von 10 Sekunden vorgesehen. Jetzt folgt der eigentliche Kern des Algorithmus. Zyklisch werden vier UPD-Pakete empfangen und in lokalen Variablen abgespeichert. Die Funktion socket.recvfrom wartet in jedem Schritt so lange, bis ein neues Paket an dem jeweiligen Port ankommt. Dieses Warten ermöglicht ein Synchronisieren von Simulink und Python. Sobald alle der vier benötigten Pakete angekommen sind, werden die Daten weiterverarbeitet.

```
1
    print("--> INFO: Reset Smart Contract Parameters")
    print("")
    print("************************************************************")
    contract.reset(   transact={'from':          w3.eth.   accounts[0]})
    time.sleep(10)

currentTimeStamp = 1;

while True:
# Read out upd packages
data, addr = recieveNeededPowerHouseholdSocket.recvfrom(64)
data = struct.unpack('d',data)
recievedPowerHousehold = int(data[0])

data, addr = recieveNeededPowerIndustrySocket.recvfrom(64)
data = struct.unpack('d',data)
recievedPowerIndustry = int(data[0])

data, addr = recievePriorityIndustrySocket.recvfrom(64)
data = struct.unpack('d',data)
recievedPriorityIndustry = int(data[0])

data, addr = recievePriorityHouseholdSocket.recvfrom(64)
data = struct.unpack('d',data)
recievedPriorityHousehold = int(data[0])
print("--> LOG: Recieved Required Data in UDP Packages")
25
26
```

Listing C.4: Empfangen von Daten aus Simulink über Python.

In Listing C.5 ist der letzte Teil des Pythonskripts dargestellt. Darin enthalten ist der Aufruf der Funktion setCurrentStateParameters, über den die Berechnung der einzelnen Leistungen

im Smart Contract stattfindet. Genau wie für den Reset ist eine Verzögerung vorgesehen, in der macOS die Transaktion in einem gültigen Block platzieren kann. Es sind bewusst keine 10 Sekunden Verzögerung gewählt, so dass das System immer noch ca. 1 Sekunde lang auf die ankommenden UDP-Pakete wartet. Damit soll sichergestellt werden, dass keine Pakete verloren gehen und die Synchronisation bestehen bleibt. Nach Ablauf der Verzögerung werden die neuen Ausgangsleistungen über die GetMethoden abgefragt. Zwischen den Abfragen sind keine Verzögerungen mehr nötig. Zum Schluss werden die Ausgangsleistungen der Verbraucher und Erzeuger noch an Simulink übertragen. Dies übernehmen die letzten Zeilen des Listings mithilfe der socket.sendto Funktion.

```
print("--> LOG: Set Current Data in  SmartContract")
print("--> LOG: Calculate new Output Data in SmartContract")
contract.setCurrentStateParameters(recievedPriorityHousehold,
    recievedPriorityIndustry, recievedPowerIndustry, recievedPowerHousehold,
    transact={'from': w3.eth.accounts[0]})
time.sleep(9)

powerProducerOutput      =      contract. getPowerProducerOutput()
powerHouseholdOutput     =     contract. getPowerHouseholdOutput()
powerIndustryOutput = contract.getPowerIndustryOutput()

sendOutputPowerHouseholdSocket.sendto(struct.pack('i',powerHouseholdOutput),(
    IP_Simulink, 20194))
sendOutputPowerIndustrySocket.sendto(struct.pack('i',powerIndustryOutput),(
    IP_Simulink, 20193))
sendOutputPowerProducerSocket.sendto(struct.pack('i',powerProducerOutput),(
    IP_Simulink, 20192))
print("--> LOG: Read Out and Send SmartContract Data to Matlab")
```

Listing C.5: Aufrufen des Smart Contracts zur Berechnung der Ausgangsleistungen über Python.

C.6 Inbetriebnahme des Gesamtsystems

Bevor Ergebnisse in Form von Schaubildern präsentiert werden, soll in diesem Abschnitt die Inbetriebnahme des Systems erfolgen. Folgende Schritte müssen dafür in der vorgegebenen Reihenfolge durchgeführt werden.

1. Starten der Blockchain auf den verschiedenen Nodes (Raspberry Pi optional)
2. Verbinden der Nodes
3. Starten des Miningvorgangs in der Konsole unter macOS
4. Ausführen des Pythonskripts im Terminal unter Ubuntu
5. Start der Simulink Simulation
6. Stoppen aller Komponenten nach Ablauf der Simulationszeit

Da die ersten drei Schritte bereits im ersten Teil des Masterprojekts abgearbeitet werden, sind diese nur kurz erläutert. Zum Starten der Blockchain wird das `startnode.sh` Skript in der Konsole ausgeführt. In einem zweiten Terminal wird die Geth Konsole geöffnet, um die Node mit den anderen Teilnehmern bekannt zu machen. In Abbildung C.6 ist der Stand nach Abschluss der ersten drei Schritte unter macOS dargestellt.

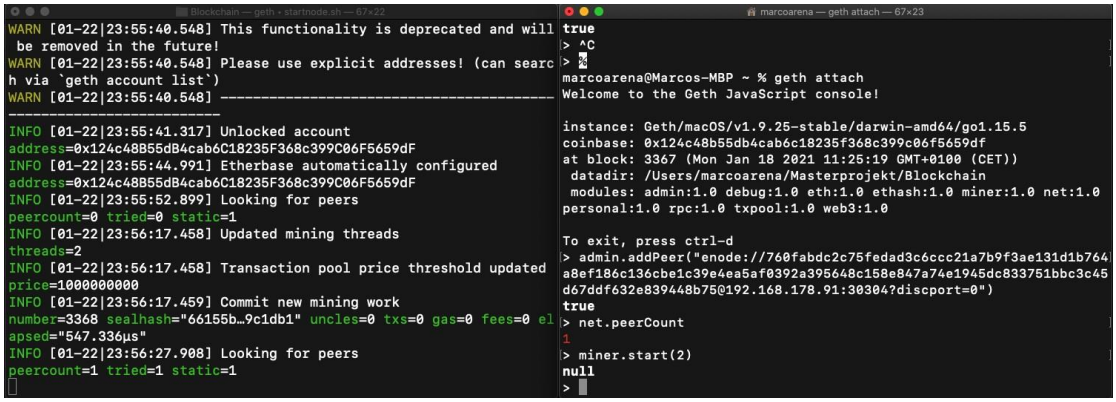


Abbildung C.6: Starten des Miningvorgangs in einem Blockchain Netzwerk.

Im nächsten Schritt wird das Pythonskript in Ubuntu aufgerufen. In Abbildung C.7 sind die beiden Konsolen unter Ubuntu abgebildet. Die obere Konsole stellt die Blockchain und deren Ausgaben dar. In der unteren Konsole wird das Skript mit dem Befehl `Python3` ausgeführt. In den Logs der Blockchain ist die Transaktion zu erkennen, die durch das Triggern der `Reset` Funktion hervorgerufen wird. Ab jetzt befindet sich das Skript im gewünschten Wartezustand, da keine UPD-Packete in Python vorhanden sind.

Damit die Kommunikation zwischen Simulink und Python funktioniert, müssen die Solver-Einstellungen in Simulink angepasst werden. Diese sind im Menü unter Simulation -> Model Settings -> Solver zu finden. Alle wichtigen Parameter sind in Abbildung C.8 in rot markiert. Die Stoptime von 1910 Sekunden resultiert in einer Simulationsdauer von genau 48 Stunden. Die Stoptime wie auch die Stepsize kann selbstverständlich angepasst werden. Allerdings funktioniert das Matlab Skript nur mit genau dieser Konfiguration einwandfrei. Soll die Stepsize geändert werden, ist das Anpassen

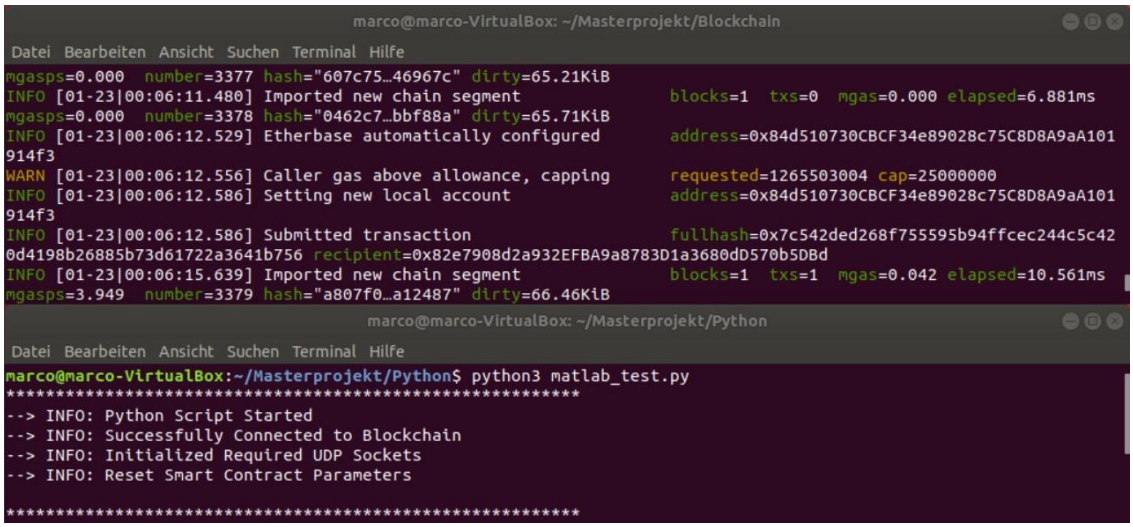


Abbildung C.7: Starten des Pythonskripts in Ubuntu bei laufender Blockchain.

des Matlabskripts bzw. der darin enthaltenden Zeitvektoren nötig. Außerdem gilt es zu beachten keine Simulationsbeschleunigung zu aktivieren. Ist diese Einstellung mit Accelerator oder Rapid Accelerator konfiguriert hält Simulink die absoluten Schrittweiten in nicht mehr ein, sondern versucht die Simulation so schnell wie möglich durchzuführen. Je nach Rechenleistung des Computers wäre die Simulation demnach innerhalb von Sekunden abgeschlossen und die Synchronisation zwischen Simulink und Python wäre nicht gegeben.

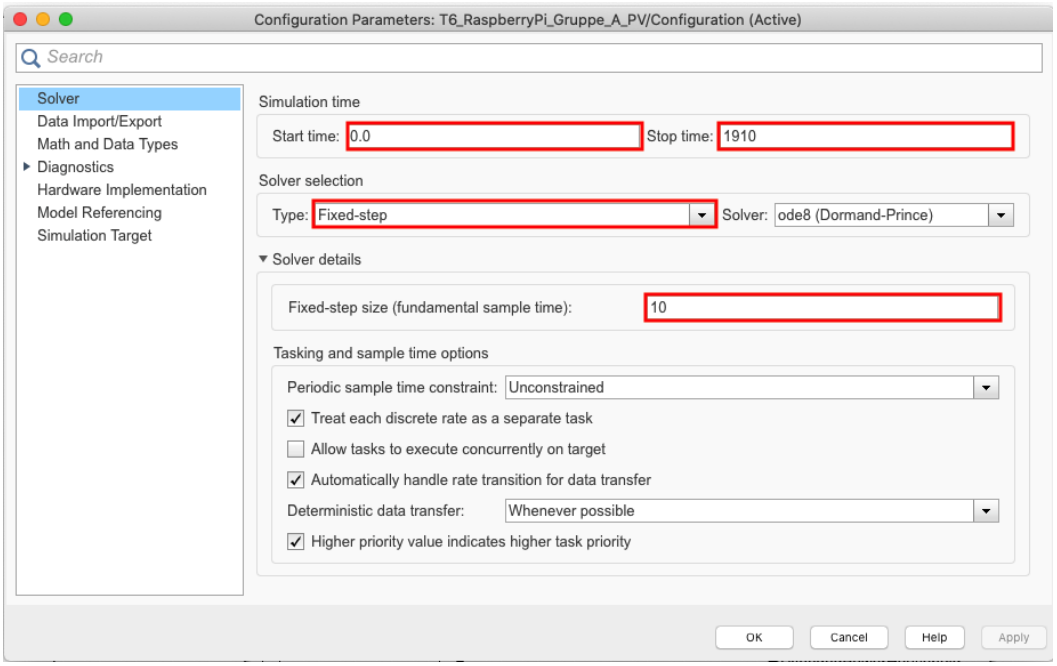


Abbildung C.8: Konfiguration der Simulation in Simulink.

Nach Übernehmen der Einstellungen kann die Simulation entweder manuell oder per Matlabskript gestartet werden.

In Abbildung C.9 ist die laufende Blockchain und das Pythonskript in Ubuntu dargestellt. Sind alle gezeigten Schritte durchgeführt muss die Ausgabe bis auf die konkreten Zahlenwerte genauso aussehen. Alle 10 Sekunden bekommt die Blockchain eine neue Transaktion. Die einzelnen Transaktionen lassen sich mit den Statusmeldungen Submitted transaction verfolgen. Beim Betrachten der Zeitstempel kann die Synchronisation der gesamten Simulation bestätigt werden. Zwischen zwei übermittelten Transaktionen muss der Miner einige Blöcke erzeugen, so dass nach jedem Zeitschritt die Werte im Smart Contract aktualisiert vorliegen. In diesem Fall liegen fünf erzeugte Blöcke zwischen den beiden Transaktionen. Sinkt diese Zahl allerdings auf unter drei erzeugte Blöcke benötigt der Miner mehr Rechenleistung. Diese noch fehlende Rechenleistung bekommt der Miner durch mehr Threads, die ihm zur Verfügung gestellt werden. Im unteren Teil von Abbildung C.9 sind die LOG bzw. DEBUG Nachrichten dargestellt. Die LOG Nachrichten dienen zur Überwachung der Funktion, während in den DEBUG Nachrichten alle relevanten Werte für jeden Simulationsschritt ausgegeben werden. Tritt

ein Fehler auf lässt sich dieser bereits zur Laufzeit und nicht erst im Nachhinein identifizieren.

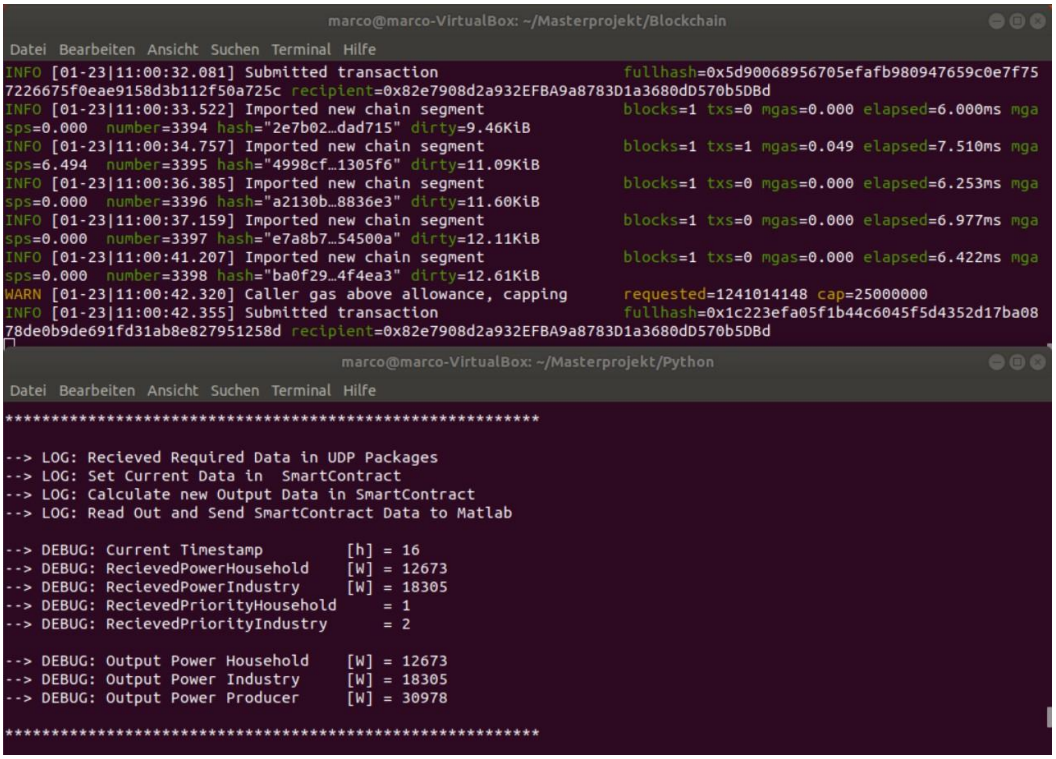


Abbildung C.9: Ausgabe der Konsolen bei erfolgreicher Simulation.

C.7 Simulationsergebnisse

Mit der erfolgreichen Inbetriebnahme der Simulation folgt die Visualisierung der Ergebnisse, um die einwandfreie Funktion des Smart Contracts zu verifizieren. Insgesamt gibt es zwei verschiedene Anwendungsszenarien, die in diesem Abschnitt untersucht werden. Bei dem ersten Szenario gibt es nur den Erzeuger (Kraftwerk) und die beiden Verbraucher (Industrie und Haushalte). Im zweiten Schritt folgt die Integration der angekündigten Photovoltaikanlage in das System, wodurch der Leistungsbedarf der Haushalte sinkt und sogar negative Leistungen auftreten können.

C.7.1 Szenario 1: System ohne PV-Anlage für 48h

Für das erste Szenario wird das Lastprofil für die Jahreszeit Winter gewählt. In der Simulation kann die Jahreszeit über die Variable ChangeSeason zwischen Sommer und Winter variiert werden. Dadurch lassen sich für die beiden Szenarien verschiedene Schaubilder generieren, die sich deutlich voneinander unterscheiden. Ebenfalls variabel ist der Zeitraum, in dem die Simulation spielt. Der Anwender kann wählen, ob es sich bei dem jeweiligen Tag um einen Werktag, Samstag oder Sonntag bzw. Feiertag handeln soll. Da die Simulation den Verlauf von 48h simuliert, können beliebe Abfolgen der Lastprofile realisiert werden. In diesem Fall handelt es sich bei den ersten 24h um einen Werktag (z.B. Freitag) und bei den zweiten 24h um einen Samstag. Es sind bewusst nicht zwei Werktage gewählt, so dass der Unterschied im Lastprofil für verschiedene Tage in der Woche deutlich wird. Da die Freiheitsgrade in der Simulation sehr hoch sind kann in jedem Szenario immer nur eine Konfiguration mit den dazugehörigen Schaubildern gezeigt werden.

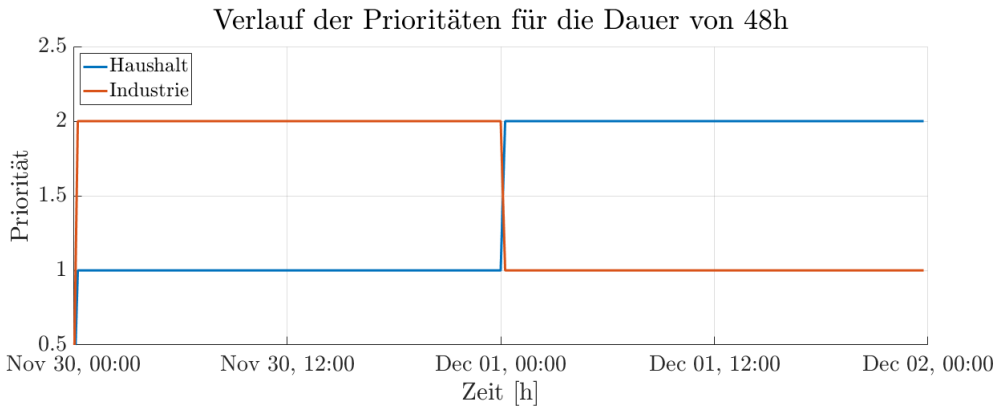


Abbildung C.10: In Matlab Simulink erstellter Verlauf der Prioritäten von Haushalten und Industrie.

In Abbildung C.10 ist der Verlauf der Prioritäten dargestellt. Auch dieser ist nicht fest vorgegebenen, sondern lässt sich in dem Matlabskript beliebig für jeden einzelnen Zeitschritt anpassen. Damit die Schaubilder übersichtlich bleiben, ist nur ein Wechsel der Prioritäten innerhalb der 48h vorgesehen. Bei Interesse kann die Priorität allerdings mit deutlich höherer Frequenz den Wert ändern. Zu Beginn des Schaubildes hat die Industrie eine höhere Priorität als die Haushalte. Der Gedanke dahinter ist, dass während eines

Werktags große und komplexe Maschinen im Einsatz sind, die nicht einfach so abgeschaltet werden können ohne Schäden an Mensch und Umwelt zu verursachen. Ein privater Kühlschrank, der vorübergehend keine Energiezufuhr hat, ist dagegen eher unkritisch bezüglich auftretender Schäden. Außerdem sind im Normalfall viele der Privatpersonen nicht zuhause, sondern bei der Arbeit und spüren daher keine Auswirkungen. Am Wochenende wechseln die Prioritäten und die Haushalte sind höher priorisiert. Die Auslastung der Industrie sinkt, da einige Unternehmen nur Werktags zur Verfügung stehen. Außerdem befinden sich die Privatpersonen gerade in den Wintermonaten viel zuhause. In dieser Zeit nutzen diese gerne Unterhaltungselektronik und schalten häufiger das Licht ein. Das Reduzieren der zur Verfügung stehenden Leistung resultiert demnach in der Unzufriedenheit der Verbraucher. Daher muss die Industrie über das Wochenende mit Leistungseinbußen zurechtkommen. Innerhalb der Industrie ist daher eine weitere Priorisierung der Unternehmen nötig, um wichtige Anlagen und Maschinen zu schützen. Diese kann ein Teil der zukünftigen Projektarbeiten sein. Im Weiteren sollen die einzelnen Lastprofile für das erste Szenario gezeigt werden.

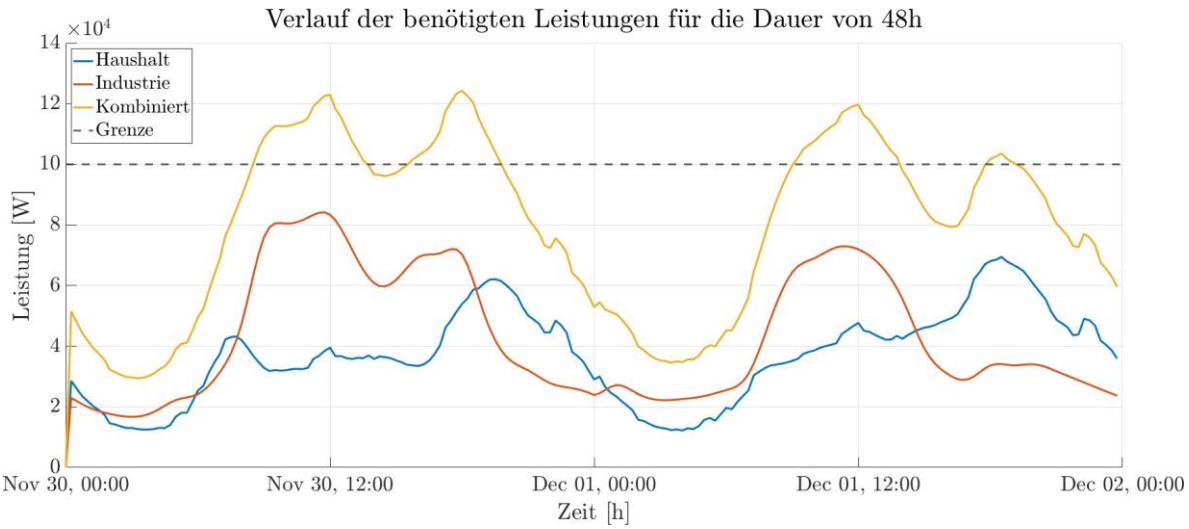


Abbildung C.11: Übersicht der Leistungsprofile G0 und H0 , sowie der Kombination der Teilnehmer für den Zeitraum von 48h im Winter ohne Photovoltaikanlage.

Dafür ist in Abbildung C.11 der Verlauf der benötigten Leistungen der beiden Verbraucher dargestellt. Bereits auf den ersten Blick ist der Unterschied der Lastprofile für den Werktag und den Samstag zu erkennen. Ebenfalls unterscheidet sich das Profil der Haushalte deutlich von dem der

Industrie. Der daraus resultierende Gesamtleistungsbedarf ist in Gelb dargestellt. Bei einer Leistung von 100 kW ist die maximale Auslastung des Stromnetzes erreicht. Insgesamt übersteigt der Bedarf die maximal übertragbare Leistung zu vier Zeitpunkten. Besonders zur Mittagszeit und gegen Abend steigt der Leistungsbedarf sehr schnell an. Das Ziel des Smart Contracts ist es diese Überlastungen zu verarbeiten und die Leistungen so zu verteilen, dass keine Überlastung das Stromnetz beschädigt. Bei Bedarf kann der Betrag der Lastprofile individuell über einen Faktor angepasst werden. In der Nacht sinkt der Leistungsbedarf sowohl für die Industrie als auch für die Haushalte stark ab und hält für mehrere Stunden ein nahezu konstantes Level. Erst in den Morgenstunden ist der erneute Anstieg der benötigten Leistung zu erkennen.

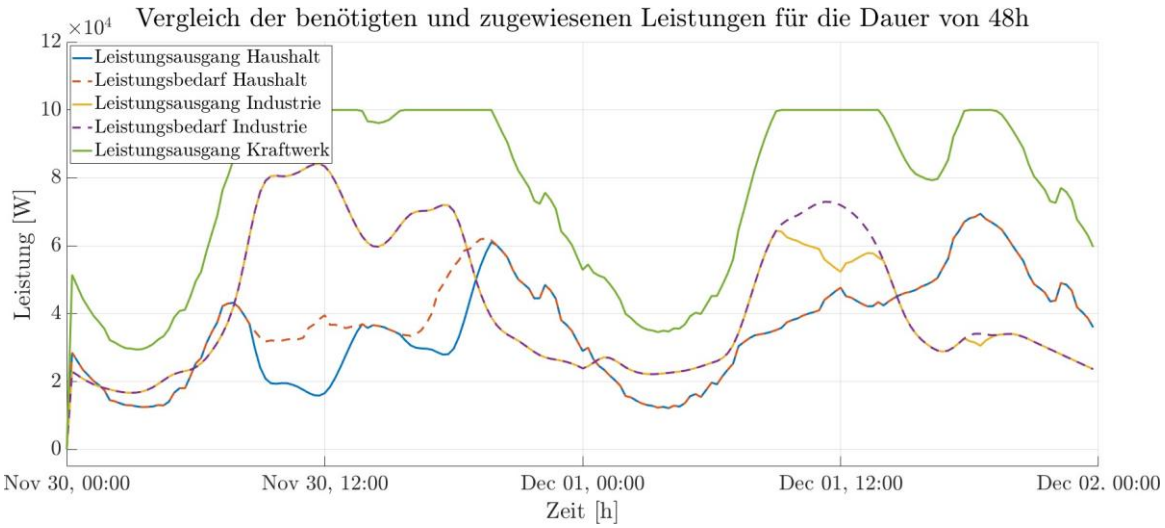


Abbildung C.12: Ausgangsleistungen der Simulationsteilnehmer aus dem Smart Contract für das erste Szenario.

In Abbildung C.12 ist der endgültige Verlauf der einzelnen Leistungen dargestellt. Die gestrichelten Linien repräsentieren das Lastprofil aus Abbildung C.11 für die Industrie und die Haushalte. Bei den durchgezogenen Linien handelt es sich um die Leistungen, die der Smart Contract den einzelnen

Verbrauchern zugeordnet hat. Für die ersten 24h entspricht die berechnete Leistung der Industrie im Smart Contract zu jedem Zeitpunkt auch dem vorgesehenen Bedarf. Dieses Verhalten ist durch die höhere Priorität begründet (siehe Abbildung C.10). Zu den Zeitpunkten an denen die benötigte Gesamtleistung das Limit übersteigt müssen demnach die Haushalte mit weniger Leistung als gefordert auskommen. Im zweiten Zeitintervall wechselt das eben erläuterte Verhalten aufgrund der Änderung der Prioritäten. Die Haushalte werden mit 100% der geforderten Leistung bedient, während die Industrie gedrosselt wird. Genau das ist das Verhalten, welches der Smart Contract hervorrufen sollte. In Grün ist der Verlauf der erzeugten Leistung des Kraftwerks dargestellt. Mit dieser Kurve lassen sich die Zeiträume identifizieren, in denen das Stromnetz die maximal mögliche Leistung überträgt. Durch diesen Verlauf wird deutlich welchen Schwankungen der Erzeuger ausgesetzt ist. Die einwandfreie Funktion für dieses Szenario ist damit gegeben.

C.7.2 Szenario 2: System mit PV-Anlage für 48h

Das zweite Szenario findet nicht mehr im Winter, sondern im Sommer statt. Dabei ist die Jahreszeit bewusst so gewählt, dass mit einem möglichst großen Ertrag der Photovoltaikanlage zu rechnen ist. Erreicht wird der hohe Ertrag durch die vielen Sommerstunden von ca. 06:00 22:00 und dem steilen Winkel, mit dem die Sonnenstrahlen auf die Atmosphäre treffen. In Abbildung C.13 ist das Erzeugerprofil der Photovoltaikanlagen dargestellt. Die Daten dafür stammen aus dem Projekt Alternative Energien. Dabei sind folgende Annahmen getroffen. Alle Photovoltaikanlagen befinden sich im Raum Stuttgart und haben eine Ausrichtung von 150° Süd-Ost mit einer durchschnittlichen Dachneigung von 35°. Zu Beginn befindet sich die Ausgangsleistung bei 0 kW, da in der Nacht keine Sonnenstrahlen auf die Anlage treffen. Bereits ab 06:00 Uhr des ersten Tages beginnt die Leistung kontinuierlich anzusteigen. Zwischen 10:00 und 12:00 ist die Ausgangsleistung mit am höchsten innerhalb der 48h. Während des gesamten Tages sind die enormen Schwankungen der Leistung sichtbar, die eine Integration von erneuerbaren Energien zum Problem werden lassen. In der Nacht sinkt die Ausgangsleistung erneut auf null. Beim zweiten Tag handelt es sich um einen weniger sonnigen Tag. Kurz nach dem Mittagspeak bleibt der Ertrag im Vergleich zum Vortag relativ niedrig und konstant.

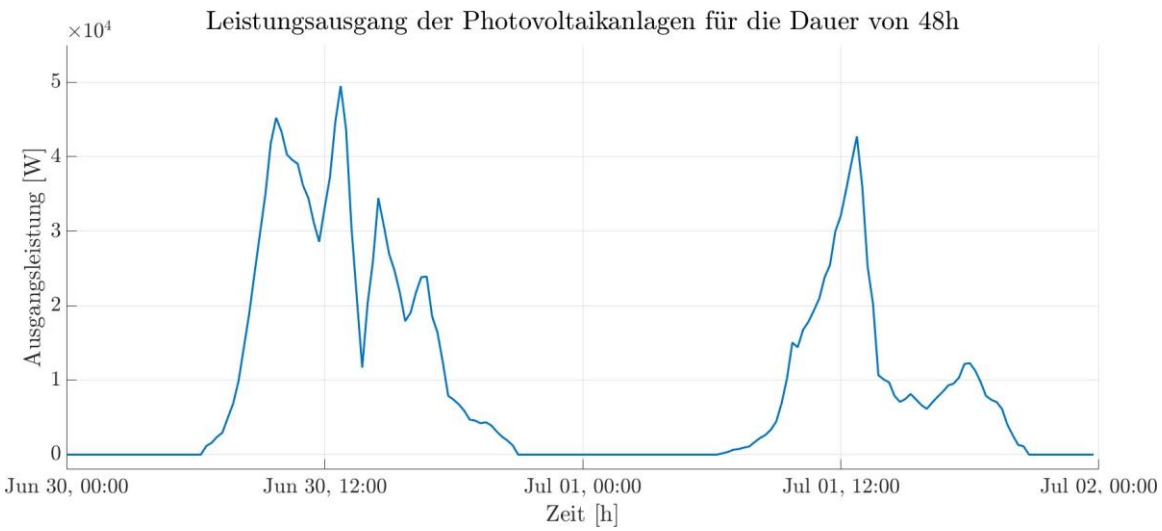


Abbildung C.13: Leistungsausgabe einer Vielzahl an Photovoltaikanlagen auf Grundlage von Alternative Energien 2.

Die Verteilung der Prioritäten wird für diesen Abschnitt zur besseren Übersicht nicht verändert. Es gelten dementsprechend immer noch die Verläufe der Prioritäten aus Abbildung C.10 ohne Anpassungen.

Der Verlauf der Lastprofile für das zweite Szenario ist in Abbildung C.14 visualisiert. Genau wie im ersten Szenario bestehen die 48h Simulationsdauer aus einem Werktag und einem darauffolgenden Samstag. Dabei ist das Lastprofil der Industrie unverändert geblieben. Eine deutliche Änderung des Lastprofils ist hingegen bei den Haushalten zu erkennen. Es ist deutlich dynamischer und weist kurzzeitig sogar einen negativen Bedarf an Leistung auf. Dadurch werden die Haushalte aus Sicht des Netzbetreibers kurzfristig zum zweiten Erzeuger im System. Verursacht wird dieser negative Leistungsbedarf durch den hohen Leistungspeak der Photovoltaikanlage um 10:00 und dem relativ geringen Verbrauch der Haushalte zu diesem Zeitpunkt. Beim Betrachten des kombinierten Leistungsbedarfs ist zu erkennen, dass die maximale Auslastung des Stromnetzes nicht mehr ganz so oft und stark überschritten wird. Da es sich bei den zweiten 24h um einen weniger sonnigen Tag handelt, ist der Leistungsbedarf der Haushalte im Durchschnitt höher.

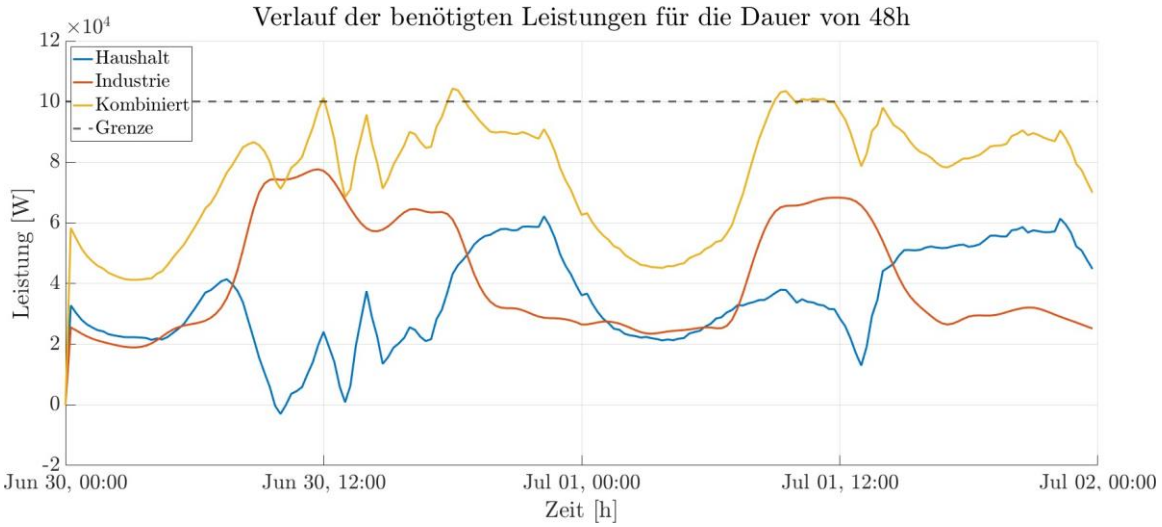


Abbildung C.14: Übersicht der Leistungsprofile G0 und H0 , sowie der Kombination der Teilnehmer für den Zeitraum von 48h im Sommer mit Photovoltaikanlage.

In Abbildung C.15 ist die Übersicht der benötigten und über den Smart Contract zugewiesenen Leistungen dargestellt. Durch den geringeren Verbrauch der Haushalte aufgrund der Photovoltaikanlagen treten Engpässe im Vergleich zu Abbildung C.12 deutlich seltener auf. Am ersten Tag muss kaum Leistung durch den Smart Contract gedrosselt werden. Zum Zeitpunkt 17:00 übersteigt die benötigte Leistung das Limit des Stromnetzes. Da die Priorität der Industrie zu Beginn der Simulation hoch ist, findet die Drosselung auf Seiten der Haushalte statt. Auch für den zweiten Tag bleibt das Stromnetz überwiegend im grünen Bereich. Kurz vor der Mittagszeit übersteigt die benötigte Leistung erneut das Limit und die Industrie wird gedrosselt, da sich die Prioritäten bei der Hälfte der Simulationszeit vertauscht haben. Ebenfalls im Schaubild in grün dargestellt ist die Leistung, die von dem Kraftwerk im Stromnetz erzeugt werden muss, damit das Netz stabil bleibt. Auch diese wird im Smart Contract berechnet und überschreitet den Wert von 100 kW zu keinem Zeitpunkt. Das Schaubild zeigt, dass auch dieses Szenario einwandfrei funktioniert und die Integration der Photovoltaikanlage ohne Probleme möglich ist, auch wenn der Leistungsbedarf des Verbrauchers dadurch kurzzeitig das Vorzeichen ändert und als zusätzlicher Erzeuger im System agiert.

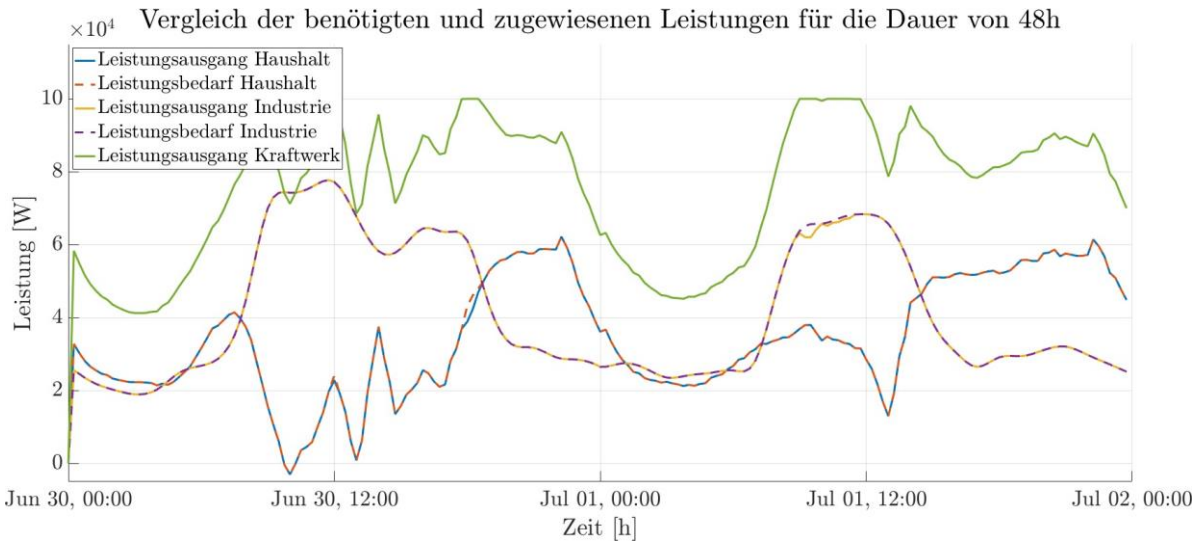


Abbildung C.15: Ausgangsleistungen der Simulationsteilnehmer aus dem Smart Contract für das zweite Szenario.

Auszug aus Studienarbeit Felix Schaal, SoSe 2020, Jim Rebholz und Marco Arena, WiSe 2020/2021

D

Einrichtung einer privaten Blockchain

Dieser Kapitel beschäftigt sich mit der Inbetriebnahme eines privaten Blockchainnetzwerks, welches im zweiten Teil als Grundlage zur Realisierung der gewünschten Funktion vorausgesetzt wird. Dafür folgt im Weiteren eine Anleitung, mit der es innerhalb geringer Zeit möglich ist, ein Blockchainnetzwerk für Testzwecke aufzubauen. Um für die folgenden Projekte flexible Lösungsansätze anzubieten, wird diese Anleitung für den Raspberry Pi, Mac OS und Ubuntu aufgeführt. Anschließend wird dem Leser der Umgang mit Smart Contracts und die mögliche Kommunikation über eine Python Schnittstelle näher gebracht. Nach abarbeiten dieses Kapitels sollte der Leser in der Lage sein alle eben genannten Aspekte für seine Projektarbeit anzuwenden.

D.1 Erste Schritte

In diesem Abschnitt sind einige Vorbereitungen für die weiteren Abschnitte aufgezeigt. Dazu zählt zum Beispiel das Einrichten eines Raspberry Pi's sowie die Neuinstallation eines Ubuntu Betriebssystems innerhalb einer Virtuellen Maschine (VM).

D.1.1 Einrichten einer virtuellen Maschine

Zur Installation eines externen Betriebssystem benötigt der Computer vorerst zusätzliche Hardware. Zwischen vielen Alternativen wird die Software Virtual Box genutzt, da sie sowohl auf Mac OS als auch auf Windows zur Verfügung steht. Gewählt wird für das weitere Vorgehen die Installation der Linux-Distribution Ubuntu in der Version 18.04. Um

sicher zu gehen, dass die folgenden Schritte wie in der Anleitung beschrieben funktionieren, sollte die gleiche Version installiert werden. Ist die ISO Datei heruntergeladen muss diese in Virtual Box integriert werden. Über den Button Neu wird das Betriebssystem aufgesetzt und installiert. Für einen reibungslosen Prozess unterstützen die Anweisungen aus Virtual Box. Nach erfolgreicher Installation ist ein Neustart der VM erforderlich, um diese in den Einstellungen zu konfigurieren. Zwei Parameter entsprechen zum Installationszeitpunkt noch nicht den Anforderungen. Zum einen die Leistung samt verfügbarem Random Access Memory (RAM) und die Netzwerkkonfiguration. Beide Änderungen lassen sich im Startbildschirm von Virtual Box über den Button Ändern anpassen. In Abbildung D.1 sind die beiden Virtual Box Fenster zur Konfiguration dargestellt. Der Hauptgrund für diese Maßnahme liegt in dem PoW Konzept. Dieses ist in Teilabschnitt A.2.2 innerhalb der Grundlagen genauer erklärt. Soll das Finden und Erzeugen neuer Blöcke in Ubuntu stattfinden, senkt die Steigerung der Performance die benötigte Zeit bis eine Transaktion erfolgreich in einem Block an die Blockchain angehängt wird. Für einen Testzweck ist es sinnvoll möglichst kurze Wartezeiten für anstehende Transaktionen einzuhalten. Bei nicht ausreichend RAM ist es außerdem mehrfach aufgetreten, dass der Miningprozess mit einer Arbeitsspeicher-fehlermeldung abgestürzt ist. Daraufhin ist ein manueller Neustart nötig.

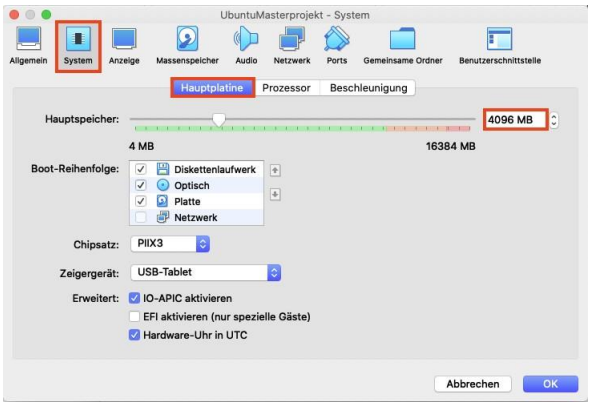


Abbildung D.1: Anpassen des für die VM zur Verfügung stehenden RAM auf mindestens 4GB

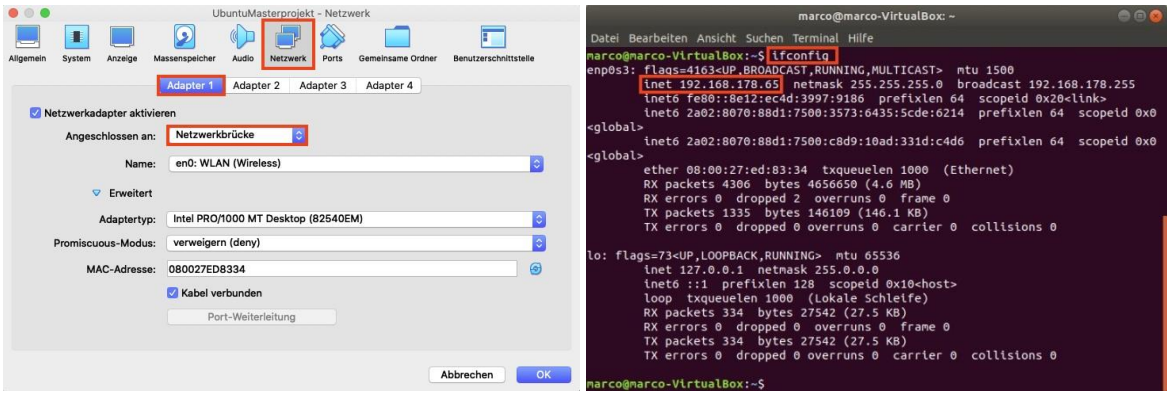


Abbildung D.2: Anpassen der Netzwerkeinstellung von Network Adress Translation (NAT) auf Netzwerkbrücke, um der VM eine eigene IPv4 Adresse zuzuweisen (links). Überprüfen der IPv4 Adresse in einem Ubuntu Konsolenfenster über den Befehl ifconfig (rechts).

Neben der Performance ist die Netzwerkkonfiguration ein weiterer Aspekt, welcher noch nicht mit den benötigten Einstellungen übereinstimmt. Relevant ist diese Änderungen immer dann, wenn die VM Teil eines Blockchain Netzwerks sein soll. Damit ein neuer Teilnehmer an einem bereits vorhandenen Netzwerk teilnehmen kann, braucht dieser eine eigene Internet Protokoll Version 4 (IPv4) Adresse.

Standardmäßig ist die Netzwerkkonfiguration auf NAT gesetzt. In dem NAT Modus kommuniziert der Host mit der VM und tauscht Netzwerkdaten aus. Das hat zur Folge, dass innerhalb der VM keine separate IPv4 Adresse zur Verfügung steht. Durch das Anpassen von NAT auf Netzwerkbrücke ist die VM netzwerktechnisch nicht von einem normalen physischen Rechner zu unterscheiden. Das genaue Vorgehen zur Anpassung der Netzwerkkonfiguration ist in Abbildung D.2 dargestellt. Ebenfalls gezeigt ist das Überprüfen der korrekten Einstellungen mit der Konsole. Dafür muss nach Bestätigen mit dem OK Button die VM gestartet werden. In einer Konsole können über den Befehl ifconfig die Netzwerkinformationen abgefragt werden. Stimmt die IPv4 Adresse sind alle Vorbereitungen in Ubuntu abgeschlossen.

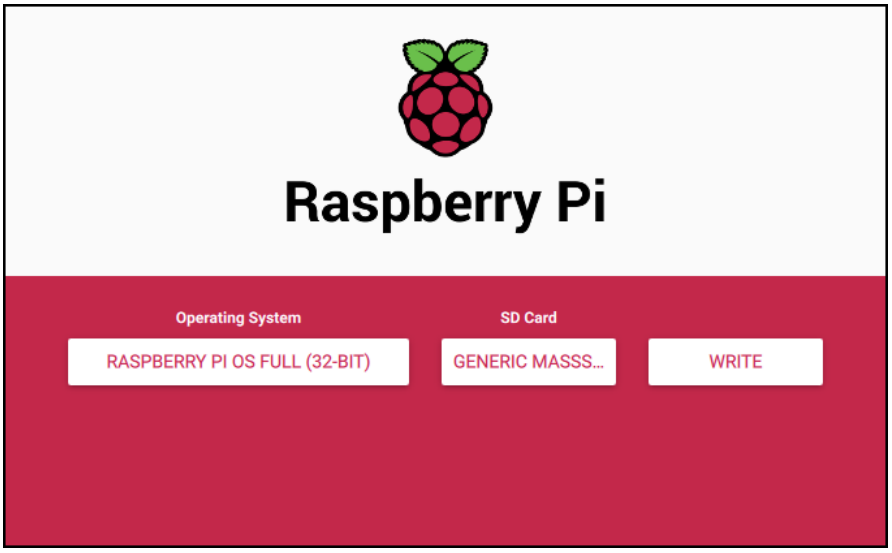


Abbildung D.3: Übersicht des Hilfstools Raspberry Pi Imager um die Micro-SD Karte mit dem gewünschten Image zu flashen.

D.1.2 Inbetriebnahme eines Raspberry Pi

Für die Inbetriebnahme des Raspberry Pi's sind ebenfalls Vorbereitungen nötig. Darunter fällt zum Beispiel das Bespielen der Micro-SD-Karte mit dem gewünschten Image. Während des Projekts hat sich herausgestellt, dass 16GB nicht genug Speicherplatz ist. Allein das Betriebssystem sowie das Speichern der Blockchain benötigt eine große Menge an Speicher auf der Micro-SD Karte. Gewählt wird dementsprechend eine nächst größere Micro-SD Karte mit 32GB. Bei der Wahl des Betriebssystems hat sich Raspberry Pi OS durchgesetzt (vorher Raspbian). Für das Flashen der Micro-SD Karte gibt es viel Hilfssoftware. Benutzt wird innerhalb dieser Arbeit das Programm Raspberry Pi Imager. Dieses ist sowohl für Mac OS als auch für Windows kostenlos verfügbar. Mit diesem Tool ist der Nutzer in der Lage über eine einfaches Graphical User Interface (GUI) ein bestimmtes Image zu wählen und dieses auf die jeweilige Micro-SD Karte zu spielen. In Abbildung D.3 ist der Aufbau der Software dargestellt. Über das Operating System auf der linken Seite lässt sich das gewünschte Image auswählen. Für die innerhalb dieser Arbeit genutzten Raspberry Pi's ist das Betriebssystem Raspberry Pi OS Full auf beiden Geräten installiert. Um die Kompatibilität mit den folgenden Schritten zu gewährleisten sollte ebenfalls dieses Betriebssystem gewählt werden. Nach dem Auswählen der SD-

Karte startet der Flashvorgang über den Button Write. Mit dem Abschließen des Flashvorgangs ist der Raspberry Pi einsatzbereit.

Um den Komfort zu erhöhen soll sich der Raspberry Pi selbstständig beim Einschalten in das Wire- less LAN (WLAN) Netzwerk des Routers einloggen. Dafür sind zwei zusätzliche Schritte nötig. Alternativ ist es auch möglich den Raspberry mit einem Universal Serial Bus (USB) Kabel zu verbinden. Im ersten Schritt folgt die Aktivierung von Secure Shell (SSH) , da über dieses Protokoll die Kommunikation zwischen Computer und Raspberry stattfindet. Um den Zugriff zu erlauben ist eine leere Datei mit dem Namen ssh nötig. Allerdings darf diese kein Dateiformat aufweisen. Im Dateimanager wird diese Datei auf die Micro SD Karte kopiert. Nach dem Flashvorgang ist der Name der Speicherkarte im Normalfall boot. Sobald der Raspberry Pi hochfährt liest und löscht er diese und gibt die SSH Funktion frei. Im Weiteren soll der Raspberry Pi noch mit dem WLAN Netzwerk kommunizieren. Ermöglicht wird diese Kommunikation über eine weitere Datei im boot Ordner. Auch dieses mal ist der Name der Datei mit wpa_supplicant.conf bereits festgelegt. Der Inhalt dieser Datei ist in Listing D.1 aufgezeigt.

```
1
country=DE


ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
  ssid="  NETWORK  -NAME"
  psk="NETWORKPASSWORD"
}
7
```

Listing D.1: Aufbau der Datei wpa_supplicant.conf

Lediglich die zwei Parameter ssid und psk sind noch nicht an das jeweilige WLAN Netzwerk angepasst. In diese Felder muss zum einen der Name des WLAN Routers als auch das dazugehörige Passwort eingetragen sein. Sind beide Dateien im boot Ordner untergebraucht, sind die Vorbereitungen am Computer für den Raspberry Pi abgeschlossen. Sobald der Raspberry Pi ans Stromnetz angeschlossen wird fährt dieser selbstständig das Betriebssystem hoch. In der Regel dauert dieser Vorgang nur wenige Sekunden. Bis sich der Raspberry schließlich in das WLAN Netzwerk eingeloggt hat

vergehen weitere Sekunden. Die IPv4 Adresse, die der Router dem Raspberry zuweist, findet sich

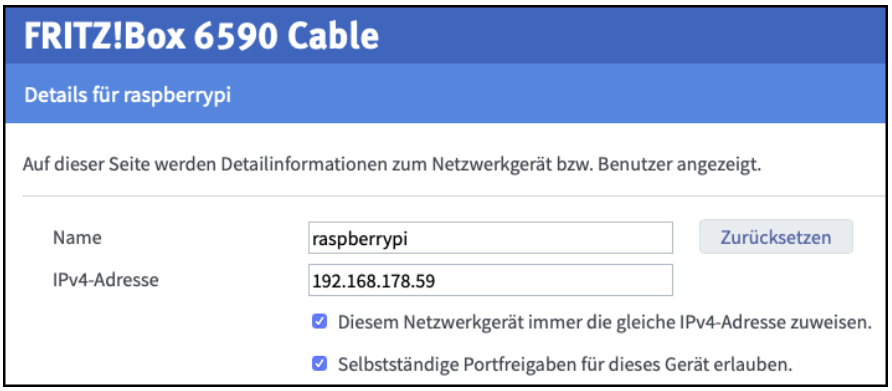


Abbildung D.4: Übersicht der Netzwerkdetails eines Raspberry Pi's mithilfe des Routers samt Namen und IPv4 Adresse.

im Menü des Routers wieder. In diesem Fall ist der Router über die URL fritz.box ansprechbar. In Abbildung D.4 ist die IPv4 Adresse des Raspberry Pi's im Konfigurationsmenü der Fritzbox dargestellt. Taucht in der Gesamtliste aller Geräte nach längerem warten kein Gerät mit dem Namen raspberrypi auf wird der Vorgang wiederholt. Erscheint der Raspberry wie in Abbildung D.4 dargestellt in der Liste aller verbundenen Geräte ist die Inbetriebnahme des Raspberry Pi's abgeschlossen.

D.2 Installation der benötigten Software

Da die Hardware auf dem aktuellen Stand ist, folgt im nächsten Schritt die Installation von zusätzlicher Software. Damit der Computer, die VM und der Raspberry Pi überhaupt in der Lage sind eine Ethereum Blockchain aufzusetzen wird das Hilfstool Geth oder Go-Ethereum verwendet. Diese Tools sind Schnittstellen, die unter anderem Folgendes ermöglichen:

- Minen von Blöcken und generieren von Ether
- Erstellen und Verwalten von Accounts
- Bereitstellen und Interagieren mit Smart Contracts

- Durchführen von Transaktionen bzw. Überweisungen
- Erzeugen eines privaten Ethereum Netzwerks

Neben Geth bzw. Go-Ethereum fehlen aber noch weitere Hilfsprogramme. Dazu zählt unter anderem NodeJS und NPM. NodeJS ist eine JavaScript Plattform, die es einem ermöglicht mit dem jeweiligen Netzwerkteilnehmer zu kommunizieren. NPM ist ein NodeJS Paketmanager, welcher wichtige zusätzliche Datenpakete zur Verfügung stellt. Außerdem wird Truffle installiert. Hinter Truffle verbirgt sich ein Framework, welches dazu verwendet wird Smart Contracts auf einfache Art und Weise zu kompilieren, zu testen und in der Blockchain bereitzustellen.

Um nicht auf ein Betriebssystem begrenzt zu sein, wird die Installation für das Macbook, die Ubuntu VM und den Raspberry Pi aufgezeigt. Auf diese Weise kann mit jedem dieser Geräte ein neues Ethereum Netzwerk in Betrieb genommen werden.

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install
    /master/install.sh)"
$ brew update
$ brew tap ethereum/ethereum
$ brew install ethereum
```

4

Listing D.2: Installation von Go-Ethereum unter Mac OS.

Zu Beginn folgt die Installation unter Mac Os. Wichtig dabei ist, dass alle Konsolenbefehle in vorgegebener Reihenfolge eingegeben werden. Auf diesem Betriebssystem ist die Installation von GoEthereum erst nach hinzufügen von Homebrew möglich. Dieses Tool ist eine Paketverwaltung zur Vereinfachung von Softwareinstallationen auf Mac OS oder Linux. Über ein Konsolenfenster lässt mit dem ersten Befehl aus Listing D.2 Homebrew installieren.

Bei erfolgreicher Installation gibt die Konsole nach kurzer Zeit die Information Installation successful! aus. Ist das der Fall erfolgt im nächsten Schritt die Installation Go-Ethereum ebenfalls über die Konsole. Mit dem Befehl in Zeile zwei ist sichergestellt, dass Homebrew in der aktuellen Version vorliegt. Anschließend fügt der Nutzer das Ethereum

Package zum Homebrew Katalog hinzu und installiert Go-Ethereum unter Mac OS. Abschließend folgt die Einrichtung von NodeJs, NPM und Truffle. Dafür sind die in Listing D.3 aufgezeigten Kommandos nötig.

```
$ brew install node
$ node -v
$ npm -v
$ npm install -g truffle ganache-cli live-server
```

Listing D.3: Installation von NodeJS und NPM unter Mac OS.

Über den Zusatz -v gibt die Konsole die Version der jeweiligen Software aus. In dieser Arbeit wird NodeJS in der Version v14.4.0 und NPM in der Version v6.14.4 verwendet. Teilweise kann es vorkommen, dass durch Updates verschiedene Features nicht mehr oder nur noch in abgeänderter Form vorhanden sind. Ist das der Fall ist es sinnvoll die Spezifikationen der Updates auf den dazugehörigen Websites nachzulesen. Dort wird auf aktuelle und folgende Änderungen hingewiesen. Die Installation bzw. Einrichtung der benötigten Software ist damit abgeschlossen. Die eigentliche Verwendung der Tools ist in den dazugehörigen Abschnitten behandelt.

D.2.1 Installation unter Ubuntu

Da Mac OS nicht auf jedem Computer vorhanden ist, bietet Ubuntu in einer VM eine einfache Alternative. Der große Vorteil dabei liegt in der Unabhängigkeit der Hardware. Unter Ubuntu ist die Installation noch einfacher, da keine Zusatzsoftware erforderlich ist. In dem bereits vorhandenen apt Paketmanager von Ubuntu sind Go-Ethereum, NodeJS, NPM und Truffle schon für den Download bereit. In Listing D.4 sind alle Befehle in geordneter Reihenfolge gelistet. Das Schlüsselwort sudo verlangt das einmalige Eingeben des Passworts.

```
$ sudo apt update
$ sudo apt install nodejs
$ sudo apt install npm
$ nodejs -v
$ npm -v
$ sudo apt install software-properties-common
$ sudo add-aptrepository -y ppa:ethereum/ethereum
$ sudo apt update
$ sudo apt install ethereum
$ sudo npm install -g truffle
```

Listing D.4: Installation von Go-Ethereum, NodeJS, NPM und Truffle unter Ubuntu.

Den Start bildet die Installation von NodeJS und NPM. Truffle wird nur über den Paketmanager NPM von Nodejs selbst bereitgestellt. Genau wie unter Mac OS lässt sich die Versionen über den Zusatz -v im Befehl ausgeben. In Zeile sechs beginnt das Einrichten von Geth. Im ersten Schritt findet eine Aktualisierung der Softwareeinstellungen statt. Anschließend wird ein sogenanntes Repository zum Paketmanager hinzugefügt, welches im Folgenden eine Installation von Geth in Zeile neun erlaubt. Aufgelistet und dennoch optional ist das Einrichten von Truffle. Truffle ist im gesamten Netzwerk nur auf einem Teilnehmer erforderlich. Diese Node ist anschließend im Netzwerk verantwortlich dafür, dass die individuellen Smart Contracts über eine Transaktion in der Blockchain zur Verfügung gestellt werden. Wahlweise ist eine Truffle Installation dementsprechend auf MacOS oder Ubuntu möglich.

D.2.2 Installation unter Raspberry Pi OS

Da der Raspberry ein nicht ganz so weit verbreitetes Hardwaresystem ist, gestaltet sich die Installation etwas aufwändiger. Um mit dem Raspberry zu kommunizieren wird die Verbindung über eine Konsole mit dem folgenden Befehl initialisiert:

```
$ ssh pi@<<IPv4-Adresse>>
```

Um die Verbindung zu erlauben fragt der Raspberry nach dem Passwort. Nach der Inbetriebnahme ist das Passwort standardmäßig auf raspberry gesetzt. Allerdings lässt es sich einfach in ein benutzerdefiniertes Passwort abgeändert. Mit

```
$ sudo raspi-config
```

gelangt der Nutzer in ein kleines grafisches Menü, zur Anpassung einiger Einstellungen auf dem Raspberry Pi. Das Startmenü ist in Abbildung D.5 dargestellt. Darunter fallen zum Beispiel das Ändern des Passworts und das Konfigurieren der Netzwerkeinstellungen. Außerdem lässt sich in diesem Menü auch der Arbeitsspeicher etwas erweitern. Diese Option lässt sich unter 7 Advanced Options

A3 Memory Split finden. Da an den Raspberry kein Bildschirm, etc. angeschlossen ist, reichen auch 16 MB Speicher für die GPU aus. Erst jetzt startet die eigentliche Installation der benötigten Software für den Raspberry Pi. In Listing D.5 sind die dazugehörigen Befehle für die Konsole dargestellt.

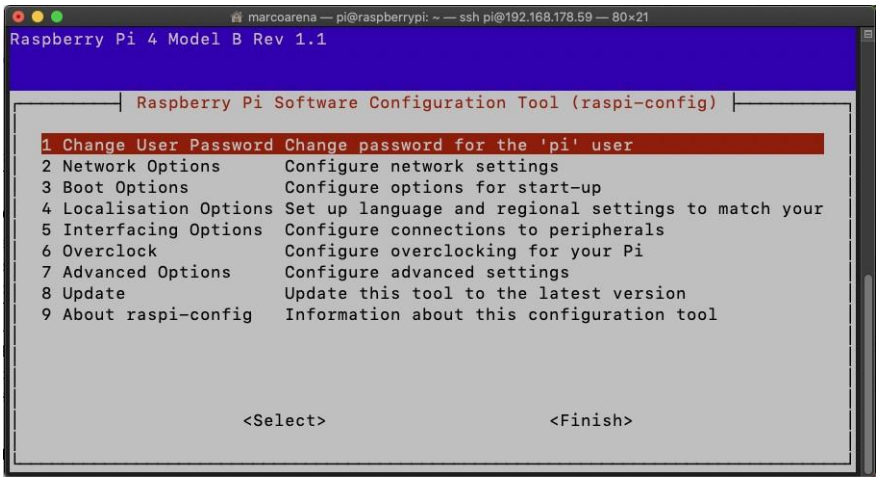


Abbildung D.5: Überlick des graphischen Konfigurationsmenüs im Raspberry Pi'.

Zu Beginn sorgen die ersten beiden Befehle dafür, dass sich die Dateipackete auf dem neuesten Stand befinden. Für die Installation von Geth ist die kompilierbare Programmiersprache Go nötig. Mit den Befehlen drei bis vier ist das erledigt. Mit den Schlüsselwörtern mkdir und cd erstellt der Anwender einen neuen Ordner mit dem Namen src und wechselt direkt in diesen. Dieser ist zum jetzigen Zeitpunkt erstmal noch

```
$ sudo apt-get update
$ sudo apt-get distupgrade
$ sudo apt-get install git golang libgmp3-dev
$ curl -sSL https://git.io/ginstall | bash
$ mkdir src
$ cd src
$ git clone -b release/1.8 https://github.com/Ethereum/goEthereum.git
$ cd goEthereum
$ make
$ sudo cp build/bin/geth /usr/local/bin/
```

Listing D.5: Installation von Go-Ethereum, NodeJS, NPM und Truffle unter Raspberry Pi OS.

leer. Mit make sorgt der User dafür, dass der Raspberry Pi die heruntergeladenen Daten für Geth kompiliert. Nach Abschluss des Kompiliervorgangs entsteht daraus eine fertige Anwendung. Mit erfolgreicher Installation von Geth sind alle Softwarekomponenten auf dem Betriebssystem vorhanden. Sind mehrere Raspberry Pi's im Netzwerk vertreten empfiehlt es sich die eben genannten Schritte nur einmal durchzuführen und die SD-Karte anschließend zu klonen. Unter MacOS unterstützt das Programm ApplePiBaker diesen Vorgang. In Ubuntu empfiehlt es sich den Prozess innerhalb der Konsole durchzuführen.

D.3 Inbetriebnahme einer privaten Blockchain

Im Prinzip ist die Inbetriebnahme einer eigenen und privaten Blockchain für alle drei Betriebssysteme gleich, weshalb hierfür nur eine Anleitung vorhanden ist.

D.3.1 Erzeugen und Initialisieren einer Genesis-Datei

Der Generierungsprozess einer Genesis File kann nicht auf dem Raspberry Pi durchgeführt werden, da dort das Hilfsprogramm Puppeth nicht vorhanden ist. Unter Mac Os als auch unter Ubuntu ist Puppeth bereits durch die im vorherigen Abschnitt gezeigten Schritte auf dem System installiert. Um später eine ordentliche Struktur hinter der Blockchain zu behalten, empfiehlt es sich einen neuen Ordner über

```
$ mkdir Blockchain && cd Blockchain
```

einzurichten. In diesem sind nach Abschluss des Projekts alle weiteren Datein abgespeichert. Puppeth lässt sich ganz einfach durch die Eingabe von puppeth in der Konsole starten. Daraufhin öffnet

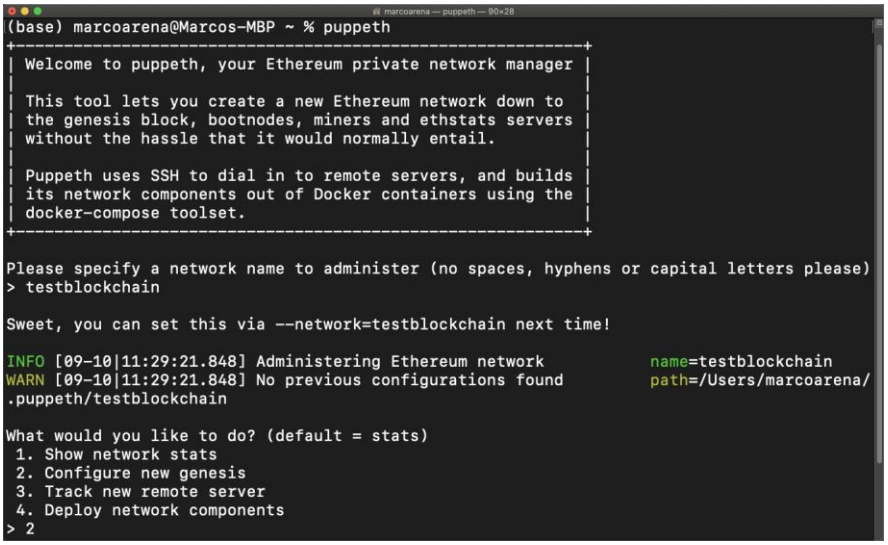


Abbildung D.6: Startbildschirm von Puppeth zur Unterstützung bei der Erzeugung einer Genesis-Datei.

sich das in Abbildung D.6 dargestellte Fenster, in dem die Anwendung nach einem Netzwerknamen fragt. Der Nutzer kann sich für einen beliebigen Namen entscheiden. In

```
Option 2 --> Configure new genesis
Option 1 --> Create new genesis from scratch
Option 1 --> Ethash proof-of-work
```

Listing D.6: Generieren einer eigenen Genesis-Datei.

Listing D.6 sind die zu wählenden Optionen chronologisch aufgelistet, bis die eigene Genesis-Datei generiert erzeugt ist.

Daraufhin fragt Puppeth den Benutzer nach bereits bestehenden Accounts, die mit einem Startguthaben versehen werden sollen. Diese Frage kann der Benutzer mit Enter überspringen und muss diese Entscheidung mit einem no bestätigen. Die letzte offene Information ist die Network ID. Im Rahmen dieser Arbeit ist diese standardmäßig als 4224 definiert. Mit dem Bestätigen der Network ID hat Puppeth bereits die Genesis-Datei

erstellt. Allerdings ist sie noch nicht in einem Ordner abgespeichert. Um die Genesis-Datei im aktuellen Verzeichnis abzulegen folgen die in Listing D.7 genannten Eingaben.

```
Option 2 --> Manage existing genesis Option
2 --> Export genesis configurations
<Enter>  --> Choose current directory
```

Listing D.7: Vorgehen zum Abspeichern der Genesis-Datei im gewünschten Verzeichnis.

Puppeth bestätigt das Abspeichern der Genesis-Datei innerhalb der Konsole. In dem aktuellen Ordner sind vier verschiedene Dateien mit der Dateiendung .json vorhanden. Davon wird lediglich die Datei mit dem Namen networkname.json benötigt. Der Inhalt ist im folgenden Listing dargestellt.

```
{    "config": {
     "chainId": 4224,
     "homesteadBlock": 0,
     " eip150Block": 0,  "
     eip150Hash": "0x0", "
     eip155Block": 0,
     "eip158Block": 0,
     "byzantiumBlock": 0,
     "constantinopleBlock": 0,
     "petersburgBlock": 0,
     " istanbulBlock": 0,
     "ethash": {}



   },
   "nonce": "0x0",
   "timestamp": "0x5f3b8f24",
   "extraData": "0x0",
   "gasLimit": "0x47b760",
   "difficulty": "0x80000",
   "mixHash": "0x0",
   "coinbase": "0x0",
   "alloc": {},
   "number": "0x0",
   "gasUsed": "0x0",
   "parentHash": "0x0"
}
```

Listing D.8: Übersicht der in Puppeth erzeugten Genesis-Datei.

Der Aufbau und die Schlüsselwörter wie timestamp oder difficulty sind im Theorieteil in Teilabschnitt A.2.1 erläutert. Puppeth hat für die Parameter automatisch geeignete Werte für ein privates Blockchainnetzwerk gewählt. Im ersten Teil in dem diverse Blöcke aufgezählt sind, die für diese Arbeit allerdings nicht weiter von Relevanz sind. Dennoch darf keines dieser Statements fehlen, da die Initialisierung der Blockchain ansonsten in Fehlermeldungen resultiert.

Erst jetzt kann der Nutzer die Blockchain bzw. den sogenannte Genesis-State initialisieren. Dabei ist zu beachten, dass sich die Konsole noch in dem Verzeichnis mit der Genesis-Datei befindet. In Geth funktioniert das Aufsetzen der Blockchain über den in Listing D.9 gezeigten Befehl.

```
1
$ geth init --datadir ~/Folder <networkname>.json
```

Listing D.9: Initialisierung einer Blockchain mithilfe von Geth.

Ist die Genesis-Datei frei von Fehlern folgt die in Abbildung D.7 gezeigte Ausgabe. In diesem Beispiel findet die Initialisierung in Ubuntu statt. Auf Mac OS und auf dem Raspberry ist die Ausgabe allerdings identisch. Ganz wichtig ist, dass am Ende die Meldung Successfully wrote genesis state erscheint. Beim Betrachten des Ordnerinhalts über das Kommando ls tauchen die zwei neuen Ordner Geth und Keystore auf. In Geth sind

```
$ geth account new --datadir ~/<Foldername>
```

Listing D.10: Erstellen eines Accounts in Geth.

Informationen über die Blockchain abgespeichert, während in Keystore Daten über die verschiedenen Accounts abgelegt sind. Um einen Account in Geth zu erstellen ist der in Listing D.10 gezeigte Befehl nötig.

Beim Anlegen des Accounts fragt die Konsole nach einem dazugehörigen Passwort, welches dafür verwendet wird den privaten Schlüssel bzw. Key zu verschlüsseln. Mit der öffentlichen Adresse ist der

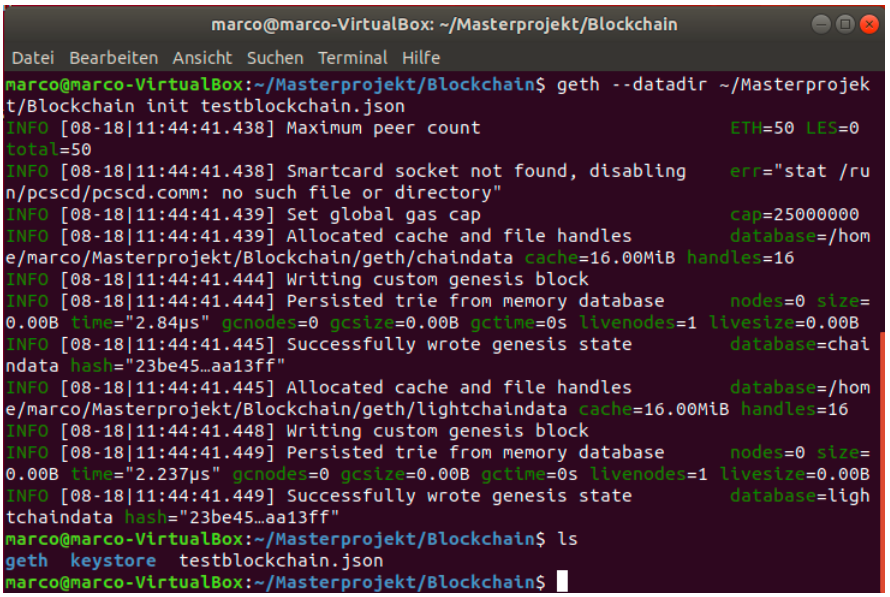


Abbildung D.7: Initialisieren einer privaten Blockchain mit anschließender Ordnerübersicht.

Sender und Empfänger einer Transaktion eindeutig referenzier bar. Daher müssen die Adressen der einzelnen Accounts auf jeden Fall bekannt sein. Um nicht durcheinander zu kommen, empfiehlt es sich, alle Accountpasswörter mit test zu belegen.

```
1
Your new key was generated
Public key address:0 x84E408dfAFf00C54038Be6019518Cee63b249A2d
Path of the secret key file: <Folder>/keystore/UTC--2020-09-10T13
    -13-59.315963000Z--84e408dfaff00c54038be601 9518cee63b249a2d
```

Listing D.11: Ausgabe der Konsole beim Erstellen eines neuen Accounts.

Anwendung findet das Passwort bei der Entriegelung von Accounts zur Durchführung von Transaktionen. Im Keystore Ordner liegt pro erzeugten Account eine Datei mit dem in Listing D.11 gezeigten Namen. Bis auf den Teil mit Puppeth sind die eben gezeigten Schritte zur Initialisierung der GenesisDatei auf allen Plattformen genau gleich und können ohne Veränderungen übernommen werden.

D.3.2 Starten der ersten Blockchain Node

Für das Starten einer Blockchain Node gibt es mehrere Möglichkeiten. In diesem Fall erleichtert ein eigenes Skript dem Anwender die Inbetriebnahme der Blockchain. In Listing D.12 ist das kleine Skript dargestellt, welches in die Datei mit dem Namen startnode.sh eingefügt wird. Dafür eignet sich zum Beispiel der Editor Nano. Ist der Inhalt aus Listing D.12 in die Datei kopiert, muss der Nutzer die Datei abspeichern.

```
$ geth --identity "macOS" --networkid 4224 --allow-insecure-unlock  --datadir
    "~/<Foldername>" --nodiscover --rpc --rpcport "8545" --port "30303" -
    rpccorsdomain "*"  --rpcapi eth,web3,personal,net --nat extip
    :192.168.178.63 --netrestrict 192.168.178.63/24 --unlock 0 --password ~/<
    Foldername>/password.sec --ipcpath "~/Library/Ethereum/geth.ipc"
```

Listing D.12: Inhalt des ausführbaren Skripts zur Inbetriebnahme der Blockchain Node.

In dem Skript sind viele bisher unbekannte Befehle vorhanden. Daher sind diese im Folgenden für den Anwender genauer beschreiben.

- **–identity:** Mit der Identity bekommt jede Node eine eigene Identität. Befinden sich im Netzwerk mehrere Teilnehmen müssen die Identitäten sich von den anderen Teilnehmern unterscheiden. Daher empfiehlt es sich einen Namen zu wählen, über den die Node eindeutig zugeordnet werden kann. Das kann zum Beispiel der Name der Hardware oder des Betriebssystems sein.
- **–networkid:** Hinter diesem Begriff verbirgt sich die Netzwerk ID des Blockchains, welche bereits in Puppeth definiert worden ist. Im Prinzip ist diese variabel. Allerdings muss Sie mit der Genesis-Datei übereinstimmen.
- **–allow-insecure-unlock:** Dieses Flag gestattet es dem Anwender von außerhalb auf die Accounts zuzugreifen und diese zu entsperren. Im

Raspberry führt dieses Flag zu einem Fehler, weshalb die startnode.sh für den Raspberry nochmal separat im Anhang aufgelistet ist.

- **–datadir:** Auf diesen Befehl folgt das Verzeichnis, in dem die Blockchain abgespeichert ist. Das ist der Ordner, in dem der Befehl aus Listing D.9 ausgeführt wird.

- **–nodiscover:** Über diesen Zusatz ist das Finden und Beitreten des Blockchain Netzwerks für einen fremden Teilnehmer nicht erlaubt. Das ist nur dann sinnvoll, wenn alle Netzteilnehmer im Vorhinein bekannt sind. Manuell lassen Sie diese trotz des Flags hinzufügen.

- **–rpc:** Mit rpc folgt die Aktivierung eines HTTP-PRC Servers, über den ein Benutzer von extern mit der Blockchain kommunizieren kann. Sobald dieser Befehl in der startnode vorhanden ist, muss auch ein dazugehöriger Port definiert sein. Standardmäßig ist das der Port 8545.

- **–port** Der hier definierte Port ist für jeden Netzwerkteilnehmer individuell und dient zur Kommunikation der einzelnen Nodes untereinander. Um einen Teilnehmer ins Netzwerk einzuladen ist die Angabe des Ports notwendig. Als Defaultport ist 30303 festgelegt.

- **–rpcapi:** Alle darauffolgenden Argumente stehen für die API, die für den Umgang mit der Node erlaubt sein sollen.

- **–nat:** Mithilfe von diesem Parameter bekommt die Node ihre eigene IPv4-Adresse mitgeteilt.

- Wie die IPv4-Adresse eines Teilnehmers ermittelt werden kann ist in Abbildung D.4 gezeigt.

- **–netrestrict:** Um den Zugang der Teilnehmer einzuschränken ist dieser Zusatz integriert. Über diesen ist die sogenannte Netmask des Subnetzes begrenzt. Die gewählte Netmask setzt sich aus der eigenen IPv4-Adresse und einem /24 am Ende zusammen.

- **–unlock 0:** Für einen sofort entsperrten Account sorgt dieser Teil. Alternativ lässt sich das entsperren eines Accounts auch über die

Konsole durchführen. Wie auch in Arrays beginnt der erste Account beim Index Null.

- **–password:** Wie bereits erwähnt ist zum entsperren eines Accounts ein Passwort nötig, welches beim Erstellen eines Accounts abgefragt wird. In der Datei password.sec muss deshalb das Passwort ohne Zusatz eingetragen sein.
- **–ipcpath:** Über den ipcpath lässt sich über den Befehl geth attach eine Konsole starten, die schon mit der Node verbunden ist.

Da der Inhalt der Datei startnode.sh etwas komplexer ist, folgt im Anhang das individuelle Skript für Mac OS, Ubuntu und dem Raspberry Pi. Bevor das Skript ausgeführt werden kann, fehlt die Datei, in der das Passwort des ersten Accounts hinterlegt ist. Am einfachsten lässt sich eine Datei password.sec mit dem Editor Nano erstellen. In dieser darf nur das Passwort des dazugehörigen Accounts stehen.

Im Weiteren soll der Anwender das individuell angepasste Skript ausführen. Dafür fehlen dem Skript allerdings noch die Rechte zur Ausführung. In allen drei Betriebssystemen sorgt der in Listing D.13 genannte Befehl für das Hinzufügen der notwendigen Berechtigungen.

```
$ chmod +x startnode.sh
$ ./startnode.sh
```

Listing D.13: Verändern der Dateirechte zur Ausführung eines Skripts.

Mit der Ausführung des Skripts erscheinen in der Konsole einige Informationen zum aktuellen Stand der Blockchain. In Abbildung D.8 ist die Ausgabe der Konsole dargestellt. Die wichtigen Informationen für die weiteren Schritte sind in Rot hervorgehoben.

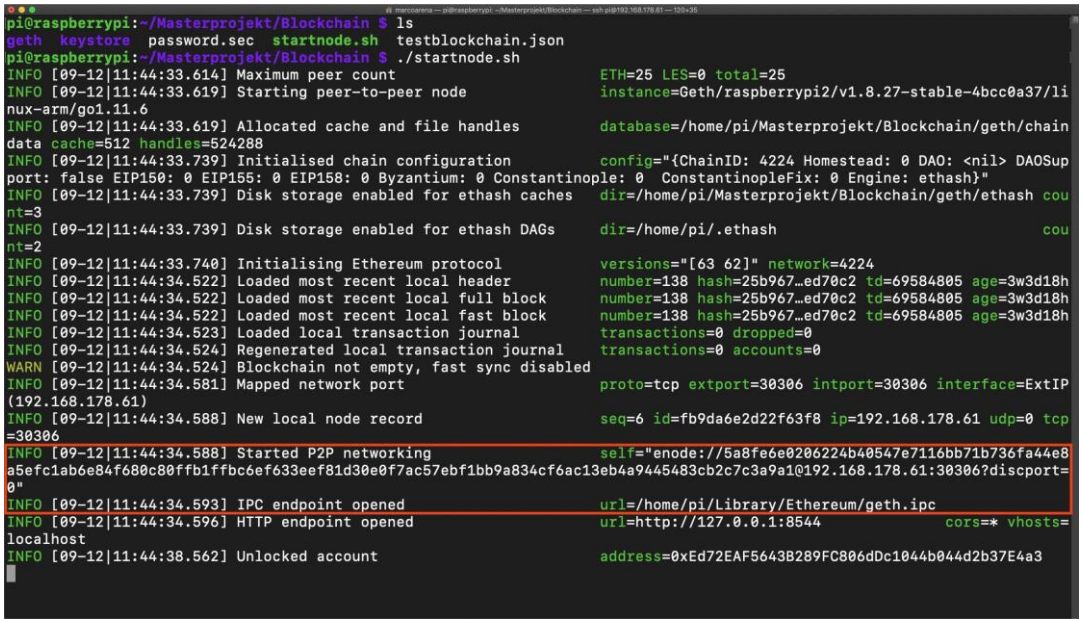


Abbildung D.8: Konsolenausgabe nach dem Starten der startnode.sh auf dem Raspberry Pi.

D.3.3 Interaktion mit der Blockchain Node

In diesem Schritt folgt die Kommunikation mit der Blockchain um beispielsweise eine Testtransaktion durchzuführen oder um den Miningvorgang zu starten. Dafür ist eine separate Konsole nötig. Alternativ bietet die Konsole innerhalb der laufenden Blockchain die Möglichkeit der Interaktion. Das ist aber mit einem Nachteil verbunden, da die Konsole sich andauernd durch neue Blöcke, etc. aktualisiert. Auf das Öffnen einer neuen Konsole folgt der in Listing D.14 dargestellte Befehl.

```
1
$ geth attach /home/pi/Libary/Ethereum/geth.ipc
```

Listing D.14: Verknüpfen der Konsole mit der Blockchain zur anschließenden Interaktion.

Der Angegebene Pfad in Listing D.14 ist dabei von Gerät zu Gerät unterschiedlich. In Abbildung D.8 ist der Pfad in dem rot markierten Bereich wieder zu finden. Innerhalb dieses Bereichs erfolgt die Freischaltung des Blockchainzugangs via IPC. Die folgende URL, welche nicht für jede Hardware gleich ist, findet sich in dem Befehl wie im Beispiel gezeigt wieder. Nach Eingabe des Befehls begrüßt die Konsole den Anwender zur Geth

JavaSkript Konsole. Zur Validierung der erfolgreichen Kommunikation sind in Tabelle D.1 ein paar wichtige Befehle aufgelistet.

Tabelle D.1: Nützliche Befehle zur Kommunikation mit der Blockchain.

Befehl	Anwendung und Funktion
miner.start(2)	Starten des Miners mit dem Teilnehmer, auf welchem der Befehl ausgeführt wird. Die Zahl in den Klammern steht für die Anzahl der verwendeten CPU Kerne.
miner.stop()	Unterbrechen der Mining Aktivität unabhängig von der Anzahl an genutzen Kernen.
eth.accounts	Ausgabe aller bekannten Accounts, die an diesem Teilnehmer erstellt worden sind.
admin.peers()	Ausgabe der mit dem Teilnehmer verbundenen Nodes. Da das Netzwerk erst im nächsten Schritt aufgesetzt wird gibt die Funktion noch den Wert Null zurück.

D.4 Einrichten eines privaten Netzwerks

Für das Errichten eines Netzwerks muss der vorherige Schritt auf mindestens zwei Teilnehmern durchgeführt sein. Voraussetzung dafür ist die identische Genesis-Datei auf allen Geräten. Zusätzlich ist das Öffnen der Konsolen zur Kommunikation mit den Nodes vorausgesetzt. Zugleich befinden sich alle Teilnehmer im gleichen WLAN Netzwerk. Beispielhaft wird das Verbinden des Raspberry Pi's mit Mac OS in diesem Abschnitt gezeigt. Bei Bedarf können beliebig viele weitere Teilnehmer am Netzwerk teilnehmen, bis das ein festgelegtes Limit erreicht ist. Das Vorgehen dabei bleibt gleich. Die einfachste Methode zur Verknüpfung zweier Nodes ist der Koppelvorgang in der Konsole zur Laufzeit. Die API Funktion

```
$ admin.addPeer(String enode)
```

erlaubt es dem Anwender unter Angabe einer Enode einen Teilnehmer ins Netzwerk einzuladen. Eine Enode ist eine hexadezimale Node ID, die neben der eindeutigen ID weitere Informationen wie zum Beispiel die IPv4 Adresse und den Port beinhaltet. Beim

Start der Blockchain in Abbildung D.8 ist die Enode im rot markierten Bereich zu sehen. Allein dieser String reicht aus, um den neuen Teilnehmer einzuladen. Welche der Nodes eine weitere Node hinzufügt spielt keine Rolle. In Abbildung D.9 ist der Grundaufbau bestehend aus vier Konsolen dargestellt. Die linke Hälfte ist an die Blockchain unter Mac OS gebunden. Auf der rechten Hälfte ist der Computer an den Raspberry Pi gekoppelt. Im oberen Teil läuft die Blockchain demnach auf zwei verschieden Teilnehmern ohne Verbindung. Der untere Teil spiegelt jeweils die im vorherigen Abschnitt erwähnte Geth Konsole zur Interaktion mit den Nodes wieder. Über den Befehl admin.peers() aus Tabelle D.1 oder alternativ mit net.peerCount lässt sich die Anzahl der hinzugefügten P2P Nodes im Netzwerk anzeigen. Sowohl unter Mac OS als auch unter Raspberry Pi OS gibt die Konsole den zu erwartenden Wert 0 zurück. In diesem Beispiel fügt die Node unter Mac OS die Raspberry Pi Node zum Netzwerk hinzu. Allerdings kann dieser Vorgang bei Bedarf auch anders herum durchgeführt werden. Für das Hinzufügen der Node ist der im rot markierte Bereich der linken Geth Konsole aus Abbildung D.9 verantwortlich. Als Übergabeparameter der Funktion admin.addPeer() muss der Benutzer die Enode des zweiten Teilnehmers eintragen. Das entspricht in diesem Beispiel der Enode des Raspberry Pi's, die ebenfalls in rot markiert ist. Als Antwort auf den Befehl gibt die Konsole den Wert true zurück. Zum Schluss wird ein weiteres Mal überprüft, wie viele P2P Nodes an dem Netzwerk teilnehmen. In Abbildung D.9 ist das im unteren Bereich abgebildet. In beiden Geth Konsolen hat sich der Wert von 0 auf 1 erhöht. Diese beiden Nodes sind jetzt verknüpft. Bei einer weiteren dritten Node fügt der Nutzer erneut den Befehl

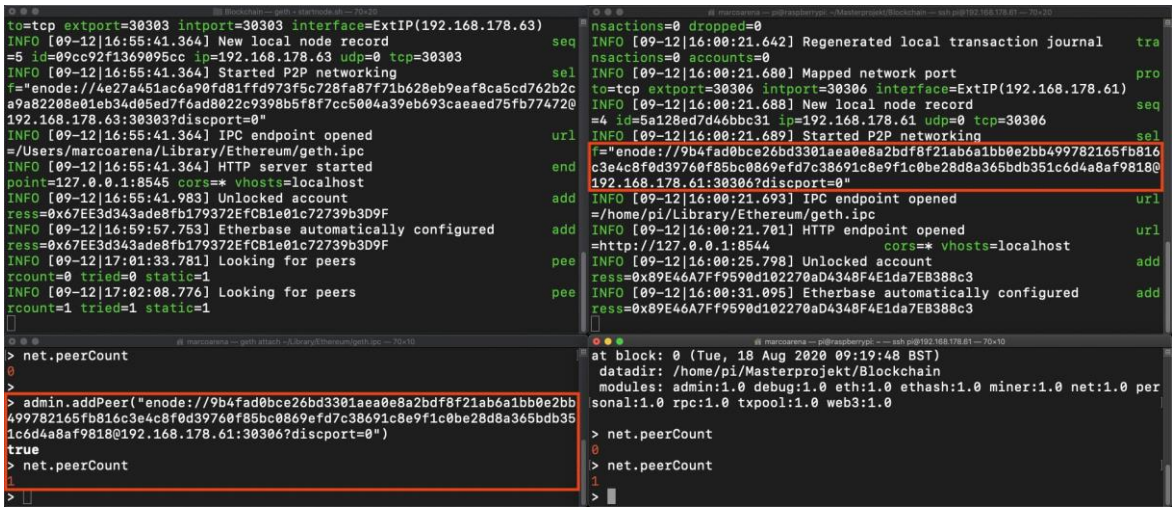


Abbildung D.9: Hinzufügen einer neuen Node ins Blockchain Netzwerk zwischen Mac OS und einem Raspberry Pi.

admin.addPeer() mit der neuen Enode in der Konsole ein. Jede Node muss zum Abschluss mit jeder weiteren Node im Netzwerk eine Bekanntschaft geschlossen haben. Um die einwandfreie Funktion der Verbindung zu demonstrieren kann über die Konsole unter Mac OS der Mining Vorgang gestartet werden. Sind die Nodes wie beschrieben

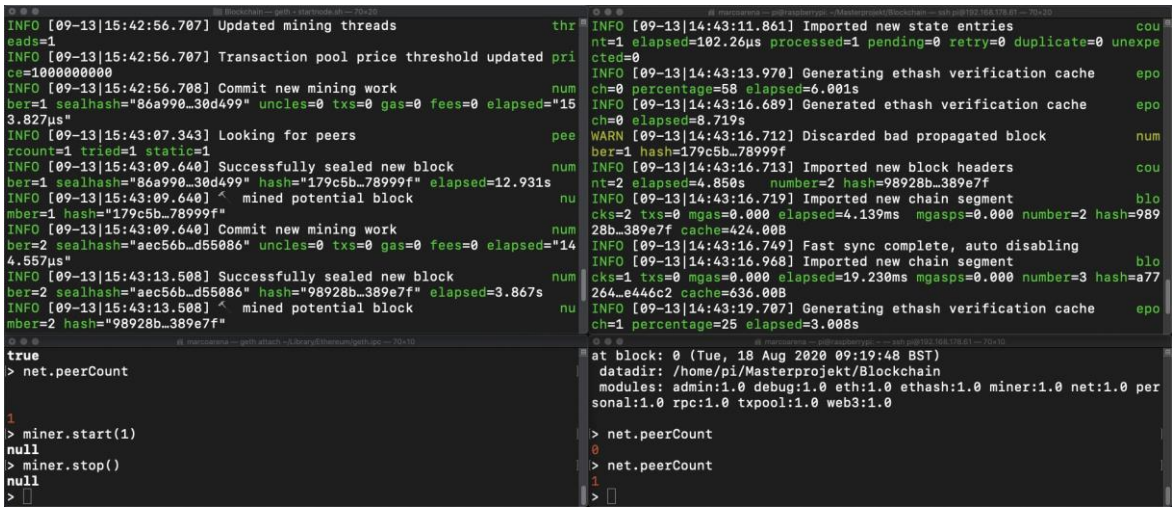


Abbildung D.10: Starten des Miningvorgangs in einem Blockchain Netzwerk zur Validierung der einwandfreien Kommunikation zwischen den Nodes.

gekoppelt, so ist die Erzeugung neuer Blöcke auf allen Nodes zu sehen. Die Visualisierung der erfolgreichen Kommunikation innerhalb des Netzwerks zeigt Abbildung D.10. In der Konsolenübersicht ist zu erkennen, wie der Miningvorgang über den in Tabelle D.1 gezeigten Befehl gestartet wird. Unter Mac OS sucht der Miner jetzt nach neuen gültigen Blöcken. Sind diese neuen Blöcke erzeugt dauert es nicht lange, bis diese an die Blockchain angehängt sind. Der Raspberry Pi agiert hier erstmal nur als Zuhörer. Jedoch validiert er die neuen Blöcke und fügt sie an die lokale Kopie seiner Blockchain an. In der Konsole kann das auf der rechten Seite über die Information Imported new chain segment nachvollzogen werden. Um das Netzwerk vorübergehend zu deaktivieren ist es wichtig die Konsolen nicht einfach so zu schließen, da die Blockchain sich ansonsten abrupt beendet. Passiert das während gerade ein Schreibprozess im Gange ist, kann es bei einer erneuten Verbindung der Teilnehmer zu Synchronisationsproblemen kommen. Daher ist es zu empfehlen die Blockchain auf allen Teilnehmern immer über die Tastenkombination STRG + C zu beenden.

D.5 Verwendung von Smart Contracts

Dieser Abschnitt behandelt den Umgang mit Smart Contracts. Im ersten Schritt sind dafür wichtige Aspekte erläutert, die für das Entwerfen eines eigenen Smart Contracts wichtig sind. Daraufhin folgt eine Anleitung, um einen Smart Contract in der Blockchain bereitzustellen bzw. zu deployen. Um das Vorgehen dabei zu veranschaulichen wird ein Minimalbeispiel durchgeführt. Zum Schluss zeigt dieser Abschnitt die Interaktion mit dem Smart Contract durch Aufrufen der zur Verfügung gestellten Funktionen.

D.5.1 Erstellen eines eigenen Smart Contracts

Für das Bereitstellen eines Smart Contracts innerhalb der Blockchain ist bereits das Hilfsprogramm Truffle unter Mac OS oder Ubunu in Teilabschnitt D.2.1 installiert. Das folgende Beispiel ist unter Mac Os durchgeführt. Alle Schritte sind aber genauso auf Ubuntu übertragbar. Zu Beginn folgt die Initialisierung von Truffle. In Listing D.15 ist der Grundbefehl zur Initialisierung dargestellt.

```
$ mkdir newFolder && cd newFolder
$ truffle init
```

Listing D.15: Erstellen eines neuen Ordners und initialisieren von Truffle um Smart Contracts zu deployen.

Mit der Ausführung des Befehls füllt sich der Ordner mit einigen Unterordnern und einer Konfigurationsdatei. Bevor ein Smart Contract deployed werden kann, muss dieser vorher in Form einer Datei vorliegen. Dafür eignet sich beispielsweise der Editor Atom, da dieser die Schlüsselwörter in Solidity hervorhebt. In Abbildung D.11 ist die Ordnerübersicht nach erfolgreicher Truffle Initialisierung dargestellt. Alle fertiggestellten Smart Contracts

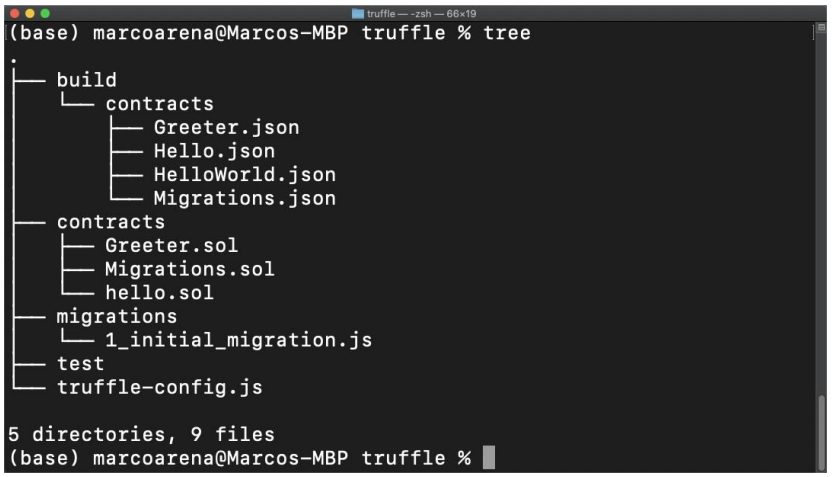


Abbildung D.11: Übersicht der Ordnerstruktur mit dem Befehl tree nach der Initialisierung von Truffle.

müssen im Ordner contracts abgespeichert mit der Solidity Dateiendung sein. Bei dem hier gezeigten Smart Contract handelt es sich um ein abgewandeltes Hello World Minimalbeispiel. In Listing D.16 ist der Smart Contract, welcher als Greeter.sol abgespeichert ist, dargestellt. Diese Datei ist auch in Abbildung D.11 in dem Ordner contracts wieder zu finden

.

```
pragma solidity ^0.5.0;
contract Greeter {
  string public greeting;
   constructor () public {
    greeting = 'Hello';
  }
  function setGreeting(string memory _greeting) public {
    greeting = _greeting;
  }
  function greet() view public returns (string memory) {
    return greeting;
  }
}
```

Listing D.16: Einfaches Smart Contract Beispiel mit Getter und Setter Methode zur anschließenden Validierung.

Er enthält lediglich einen Konstruktor und eine Get bzw. Set-Methode der internen Variable greeting vom Typ String. Zeile eins gibt die verwendete Solidity Version an. Für dieses Beispiel wird die Version 0.5.0 in der ersten Zeile des Codes über pragma festgelegt. Bei neueren Versionen sind teilweise Schlüsselwörter nicht mehr gültig und deshalb ist der hier gezeigte Code nicht mit jeder Solidity Version kompilierbar. Um die eigene Funktionalität in den Smart Contract zu verpacken sind auf der Website https://solidity.readthedocs.io/en/v0.4.24/introduction-to-smart-contracts. html sowie in der Arbeit des Vorgängers wichtige Hinweise zu Datentypen, etc. vorhanden.

D.5.2 Hinzufügen eines Smart Contracts zur Blockchain

```
var Migrations = artifacts.require("Migrations");
var Greeter = artifacts.require("Greeter");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
  deployer.deploy(Greeter);
};
```

7

Listing D.17: Anpassen der Datei <1_initial_migrations.js> zur Bekanntmachung der neuen Smart Contracts.

Ist der Smart Contract mit der individuellen Funktion fertiggestellt und wie beschrieben abgespeichert, folgt der nächste wichtige Schritt. In diesem findet die Vorbereitung für den Deployvorgang statt. Dafür muss die Datei 1_initial_migration.js aus dem Ordner migrations angepasst werden. Bei diesem Vorgehen wird Truffle mit dem neuen Smart Contract bekannt gemacht. Lediglich ein paar wenige Zeilen erfordern eine Anpassung in der erwähnten Datei. Listing D.17 zeigt das Vorgehen der Veränderungen für den Greeter Contract.

Der Inhalt ist zum größten Teil bereits beim Öffnen der Datei vorhanden. Nur die Zeilen zwei und sechs sind vom Anwender zu ergänzen. Der Name der Variablen muss mit dem Namen des Smart Contracts übereinstimmen. Daher ist die Variable mit Greeter initialisiert. Auf das Bearbeiten folgt das Speichern der Datei. Im nächsten Schritt kompiliert Truffle den Smart Contract, um den Nutzer auf eventuell angefallene Fehler hinzuweisen. Für den Kompiliervorgang stellt Truffle die in Listing D.18 genannte Kommandozeile zur Verfügung.

```
$ truffle compile
```

Listing D.18: Starten des Kompiliervorgangs für alle vorhandenen Smart Contracts.

Allerdings funktioniert der Befehl nur, wenn der Benutzer vorher mit der Konsole in den Ordner contracts navigiert. In der Konsole bestätigt Truffle den erfolgreichen Kompiliervorgang. Ist das nicht der Fall gibt die Fehlermeldung eine Zeile im Smart Contract an, in welcher ein Fehler vorliegt.

Anschließend kann der kompilierte Smart Contract im Prinzip auf eine beliebige Ethereum Blockchain deployed werden. Dafür muss Truffle noch mit einer bestimmten Blockchain in Verbindung treten. Auch das lässt sich mit Truffle einfach über die bereits vorhandene Datei truffle-config.js konfigurieren. Diese befindet sich immer auf der obersten Ebene in dem Initialverzeichnis (siehe Abbildung D.11). Innerhalb der Datei befinden sich unzählige Befehle und Einstellungen, von denen nur ein kleiner Bruchteil von Relevanz ist. Daher ist es sinnvoll den Inhalt erstmal komplett zu löschen. Anschließend folgt das Einfügen der in Listing D.19 gezeigten Konfiguration.

```
module.exports = {
  networks: {
    development: {
      host: "localhost",
      port:       8545  ,
      network_id :   "*",
      gas: 3721975,
      from: "0x351799a0Dc645650297d8DB901ba3AF2c7766c9D"
    }
  }
};
```

Listing D.19: Übersicht der Truffle Konfigurationsdatei zur Bekanntmachung von Truffle mit der Blockchain.

Zwei Aspekte sind individuell und erfordern noch ein Abändern der Truffle Konfigurationsdatei. Das ist zum einen der Port, welcher mit dem Port aus der startnode.sh übereinstimmen muss. Zum anderen ist es die Adresse nach dem Schlüsselwort from. Diese wird ersetzt durch die Adresse des ersten Accounts. Die Geth Konsole gibt diese über den Befehl aus Listing D.20 aus.

```
$ eth.accounts[0]
```

Listing D.20: Ausgabe der Adresse den ersten Accounts.

Die Angabe des ersten Accounts ist unumgänglich, da dieser die Gebühren in Form von Ether übernimmt, die es benötigt den Smart Contract auf der Blockchain zu deployen. Dementsprechend ein ausreichend hohes Grundkapital des Accounts ein wichtiger Punkt. Aufstocken lässt sich das Guthaben durch einen kurzen Miningvorgang auf dem Deployaccount. Eine geringe Schwierigkeit sorgt dabei für ein schnelles generieren von ausreichend Ether. Ist das Grundkapital an Ethern gesammelt kann der Deployvorgang mit dem in Listing D.21 dargestellten Kommando aus dem Ordner contracts gestartet werden. Mindestens ein Teilnehmer in der Blockchain muss zum Deployzeitpunkt innerhalb der Blockchain die ankommenden Smart Contracts in Transaktionen packen. Ebenfalls im Listing enthalten ist die dazugehörige Ausgabe in der Konsole, die wichtige Informationen beinhaltet.

```
$ truffle migrate --reset
Starting migrations...
======================
> Network name:    'development'
> Network id:      4224
> Block gas limit: 4930178 (0x4b3a82)
1_initial_migration.js
======================
Replacing 'Migrations'
----------------------
> transaction hash:    0xe4a399038bb9359d3da0e4dc455731bdb4103304cc32ee4582c9c9dcae5f83
> Blocks: 1            Seconds: 4
> contract address:    0x44bB614973a59F55d59b21490517e46e239F9fe2
> block number:        50
> block timestamp:     1600180897
> account:             0x67EE3d343ade8fb179372EfCB1e01c72739b3D9F
> balance:             100
> gas used:            191943 (0x2edc7)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.00383886 ETH

Replacing 'Hello'
-----------------
> transaction hash:    0x1b81b5410e7b06bcaad79b000274769773f37fed4e0f d50efb341b43e36376c7
> Blocks: 0            Seconds: 0
> contract address:    0x19D8EC5be3E5b54Aa56FA461D06d719B3801d51a
> block number:        51
> block timestamp:     1600180905
> account:             0x67EE3d343ade8fb179372EfCB1e01c72739b3D9F
> balance:             102
> gas used:            221402 (0x360da)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.00442804 ETH

Replacing 'Greeter'
-------------------
> transaction hash:    0xccb5de9271845570bd84e623cebfa29dd864a242b36fdb462f2f6de23f49914
> Blocks: 0            Seconds: 4
> contract address:    0x52C18C18FAb3A3e9aAE7d6bc657F6fa98d96b8a9
> block number:        52
> block timestamp:     1600180909
> account:             0x67EE3d343ade8fb179372EfCB1e01c72739b3D9F
> balance:             104
> gas used:            311675 (0x4c17b)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.0062335 ETH

> Saving migration to chain.
> Saving artifacts
-------------------------------------
> Total cost:          0.0145004 ETH
```

```
55
56  Summary
57  =======
58  > Total deployments:      3
59  > Final cost:             0.0145004 ETH
```

Listing D.21: Starten des Deployvorgangs aller Smart Contracts in Truffle.

Mit diesen Informationen kann der Benutzer auf den Smart Contract zugreifen und gegebenenfalls die vorhandenen Funktionen nutzen. Bei diesem Beispiel hat Truffle drei Smart Contracts deployed bzw. ersetzt. Die Angabe contract address ist besonders wichtig. Mit dieser kann der Anwender im weiteren Verlauf der Arbeit zum Beispiel über Python auf den Smart Contract zugreifen. Demensprechend ist es wichtig diese Adresse abzuspeichern. Ebenfalls interessant ist die Menge an Währung, die der Vorgang gekostet hat. Circa 0.0145 Ether hat die Blockchain dem angegebenen Account abgezogen. In der Blockchain selbst ist auch eine Verfolgung des Deployvorgangs möglich. Für jeden Smart Contract gibt die Blockchain die Information submitted contract creation in der Konsole aus. Ist das nicht der Fall liegt ein Fehler vor und die Verbindung zwischen der Blockchain und Truffle könnte abgebrochen sein.

D.5.3 Interaktion mit dem Smart Contract

Im letzten Abschnitt zu Smart Contracts ist eine mögliche Art der Interaktion dargestellt. Diese eignet sich besonders dann, wenn der Smart Contract auf grundlegende Funktionalität getestet werden soll. Genau wie der Deployvorgang findet die Interaktion über truffle statt. Allerdings nicht mehr in der normalen Konsole, sondern in einer speziellen Entwicklerkonsole, die sich über den Befehl aus Listing D.22 öffnet.

```
$ truffle console
1
```

Listing D.22: Starten der Entwicklerkonsole von Truffle.

Als Beispiel dient wieder der erweiterte Greeter Smart Contract, welcher in Listing D.16 abgebildet ist. Auch wenn darin lediglich zwei einfache Funktionen vorhanden sind, ist das hier beschriebene Vorgehen auf jede beliebige Funktion bzw. jeden Smart Contract übertragbar. In Abbildung D.12 ist der gesamte Ablauf samt Öffnen der Entwicklerkonsole dargestellt. Für den Zugriff auf Methoden eines Smart Contracts muss Truffle vorher eine Instanz des Objektes bekannt sein. Einrichten lässt sich das über das folgende Kommando.

```
$ let instance = await Greeter.deployed()
```

Listing D.23: Speichern des Smart Contract Objekts in einer Variablen.

Hinter der Variable instance verbirgt sich der Smart Contract Greeter. Genau wie in der Programmiersprache Java kann auf die Methoden eines Objekts mit dem Punktoperator zugegriffen werden. Da es sich bei der Methode greet um einen Funktionsaufruf handelt, muss der Befehl mit zwei Klammern abgeschlossen sein. In Abbildung D.12 ist der Funktionsaufruf sowie die Rückgabe der Konsole abgebildet. Da durch diesen Aufruf keinerlei Änderungen im Smart Contract stattfindet, ist dieser Funktionsaufruf ohne laufenden Miningvorgang möglich. Auch die Ausgabe mit "Hello" ist plausibel, da die interne Variable greeting bereits im Konstruktor auf diesen Defaultwert gesetzt ist.

Im nächsten Schritt folgt die Validierung der Set-Methode. Da diese Änderungen im Smart Contract und damit auch in der Blockchain verursacht, muss der Miningvorgang bereits vor dem Funktionsaufruf laufen. Als Übergabeparameter erwartet die Funktion setGreeting einen String. In Abbildung

D.12 wird die Set-Methode mit dem String "Test" aufgerufen. Als Rückgabe folgt eine längere Ausgabe. In dieser sind Details zur Transaktion enthalten, welche die Änderung durchführt. Erst wenn die Transaktion in einem gültigen Block untergebracht ist, wird die Variable tatsächlich überschrieben. Da die Schwierigkeit der Blockchain sehr niedrig ist geht das sehr zügig. Da hierfür genau wie beim Deployvorgang Ether verbraucht wird, muss der Standartaccount genügend Ressourcen aufweisen. Zum Schluss zeigt das erneute Aufrufen der Get-Methode die erfolgreiche Änderung der Variable greeting. Diese hat den Wert von "Hello" zu "Test" gewechselt.

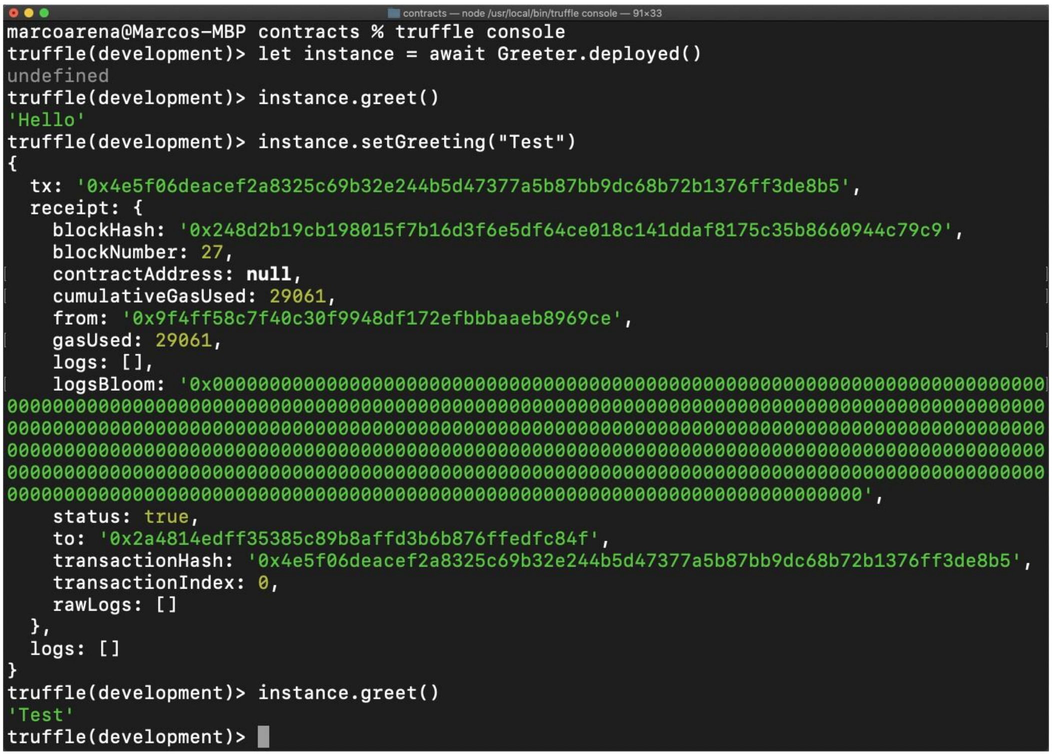


Abbildung D.12: Validierung der Get-/ Set-Methode des Smart Contracts Greeter in der Truffle Entwicklerkonsole.

D.6 Schnittstelle zur Blockchain über Python

Am bequemsten ist die Interaktion mit der Blockchain über ein Python Skript. In diesem Abschnitt folgt eine Zusammenfassung der dafür notwendigen Schritte. Da es sich bei den dafür relevanten Bibliotheken nicht um bereits vorinstallierte handelt, müssen diese manuell vom Benutzer heruntergeladen werden. In Listing D.24 sind die dazugehörigen Befehle abgebildet.

```
$ sudo apt install python3-pip
$ pip3 install web3
$ pip3 install solc
$ pip3 install py-solc
```

Listing D.24: Herunterladen von Python Bibliotheken zur Interaktion mit der Blockchain.

Mit der Installation aller eben gezeigten Bibliotheken steht der Interaktion mit der Blockchain nichts mehr im Weg. Um die Funktionsweise an einem konkreten Beispiel zu zeigen, ist es das Ziel dieses Abschnitts mit dem Smart Contract Greeter zu interagieren. Soll extern auf einen Smart Contract zugegriffen werden, sind zwei Informationen nötig. Das ist zum einen die Smart Contract Adresse, die Truffle beim Deployvorgang bekannt gibt. Zum anderen ist es die ABI, die einen Zugriff auf den Bytecode hinter dem Smart Contract erlaubt. Da die Smart Contract Adresse bereits bekannt ist folgt eine kurze Anleitung, die es dem Nutzer erlaubt die ABI in der Truffle Konsole auszugeben. In Abbildung D.13 sind die einzelnen Schritte vom Start der Truffle Entwicklerkonsole bis zur Ausgabe der ABI aufgezeigt. Dabei ist lediglich zu beachten, dass die Konsole aus dem Truffle Initialverzeichnis gestartet wird. Startet die Truffle Konsole aus einem anderen Verzeichnis stimmt die Angabe der relativen Pfade nicht mehr. Das resultiert anschließend in Fehlermeldungen. Je nach Smart Contract muss im zweiten Befehl der Name von Greeter auf den jeweils zutreffenden Namen abgeändert wer-

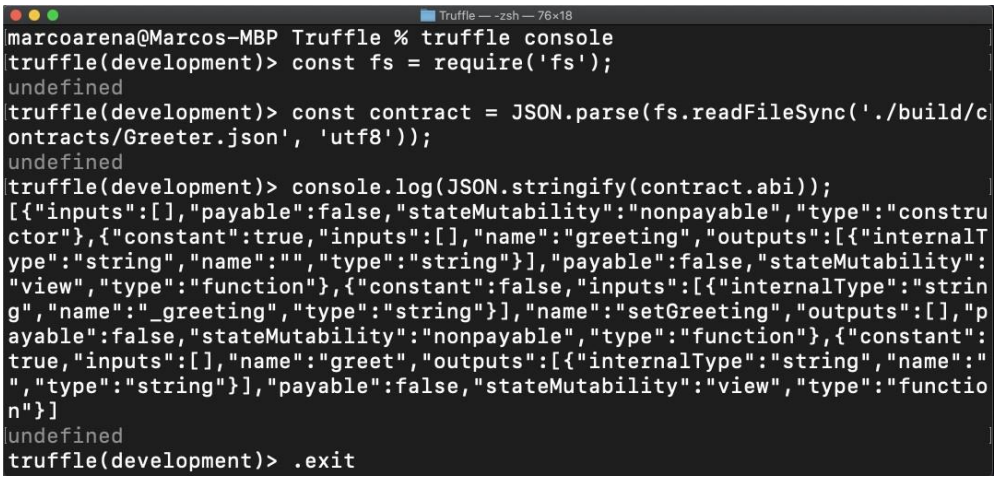


Abbildung D.13: Extrahieren der Contract ABI in der Truffle Entwicklerkonsole.

den. Nach Ausgabe der ABI in der Konsole ist es sinnvoll den ausgegebenen Code für den weiteren Verlauf abzuspeichern.

Bevor mit der ABI und der Smart Contract Adresse auf die darin enthaltene Funktionalität zugegriffen wird, gilt es einen Grundzustand herzustellen. In diesem Grundzustand

```
from web3.auto  import w3
from web3   import Web3, HTTPProvider, IPCProvider
from web3.contract import ConciseContract

import time

# Address of published contract
CONTRACTADDRESS = '0xC75C1B3C512dFf28F6aC700BEccBf096Eda0a5fe'

# ABI copied from simple.js
CONTRACTABI = '[{"inputs":[],"payable":false,"stateMutability":"nonpayable","
    type":"constructor"},{"constant":true,"inputs":[],"name":"greeting","
    outputs":[{" internalType":" string"," name":""," type":" string"}]," payable":
    false,"stateMutability":"view","type":"function"},{"constant":false,"inputs
    ":[{" internalType":" string"," name":" _greeting"," type":" string"}]," name":"
    setGreeting"," outputs":[]," payable": false ," stateMutability ":" nonpayable","
    type":"function"},{"constant":true,"inputs":[],"name":"greet","outputs":[{"
    internalType":" string"," name":""," type":" string"}]," payable": false ,"
    stateMutability":"view","type":"function"}]'



def main():
web3 = Web3(IPCProvider('/home/marco/Library/Ethereum/geth.ipc'))
consumerAccount    =    w3.eth. coinbase
w3.eth.defaultAccount = consumerAccount

print("************************************")
print("DefaultAccount erfolgreich zugewiesen")

contract  =  w3.eth.contract(address=w3.toChecksumAddress(CONTRACTADDRESS),  abi=
    CONTRACTABI, ContractFactoryClass=ConciseContract)
print("SmartContract erfolgreich zugewiesen")

print("Testen  der  SmartContract  Funktionen")
print("************************************")
print(" ")
print("Auslesen der Variable: greet = {}".format(contract.greet()))
print(" ")
print("Anpassen der Variablen")
print(" ")
contract.setGreeting('<< New String >>', transact={'from': w3.eth.accounts[0]})
print("Warten bis die Transaction verarbeitet wurde")
print("     ")
time.sleep(10)
print("Auslesen der Variable: greet = {}".format(contract.greet()))



if __name__ == '__main__':
main()
```

Listing D.25: Beispielprogrammcode in Python zur Interaktion mit dem Smart Contract Greeter.

können beliebig viele Nodes teilnehmen. Allerdings muss auf jeden Fall eine Node zur Laufzeit der Python Applikation mit dem Miningvorgang beschäftigt sein. Für dieses

Beispiel wird das Macbook mit Mac OS mit der Ubuntu VM gekoppelt wie in Abschnitt D.4 bereits erläutert. Unter Mac OS wird der Miningvorgang gestartet, während in Ubuntu der Zugriff per Python erfolgen soll. Ist dieser Zustand hergestellt kann der Anwender das Python Skript über die Konsole aufrufen. Im Prinzip ist der Aufbau bis auf den Aufruf der Funktionen immer ähnlich. Innerhalb dieser Arbeit ruft Python nur die Set-/Get-Methode auf um eine String Variable zu verändern. Je nach gewünschter Funktionalität kann der Umfang aber beliebig angepasst sein. In Listing D.25 ist der Beispielcode in der Programmiersprache Python dargestellt. Dieser wird im Folgenden analysiert und anschließend in der Konsole ausgeführt.

Da der Code bereits ca. 40 Zeilen umfasst sind viele Ausgaben in dem Code vorhanden, die den Code übersichtlich zu gestalten. Im oberen Teil sind die heruntergeladenen Bibliotheken eingebunden. Ohne den import Befehl kann das Skript nicht ausgeführt werden, da die genutzten Funktionen dem System ansonsten nicht bekannt sind. Damit zeitliche Verzögerungen möglich sind dient die Bibliothek Time. Anschließend folgt die Deklaration der wichtigen Variablen ContractAddress und ContractABI. Die Main Funktion startet mit einem Verbindungsaufbau zur Blockchain via IPC . Damit die Blockchain einen Account hat, von dessen Guthaben Sie Ether abbuchen kann ist in Zeile 18 der Defaultaccount definiert. Für den Zugriff auf den Smart Contract und dessen Methoden ist eine Objektvariable nötig. Als Übergabeparameter erhält die dafür verwendetete Funktionen unter anderem die bereits definierte ABI und Smart Contract Adresse. Mit den ständigen Ausgaben in der Konsole kann der Anwender live den aktuellen Stand mitverfolgen. Erst jetzt folgt die eigentliche Interaktion mit dem Smart Contract. Das Vorgehen ist dabei wie folgt:

1. Auslesen der Variable Greeting mithilfe der Get-Methode. Da die Variable noch nicht anderweitig verändert ist, muss die Rückgabe dem Initialstring entsprechen. Dieser ist im Konstruktor des Smart Contracts auf den String ”Hello” gesetzt.

2. Verändern der Variable Greeting auf den String ”« New String »”. Da diese Operation etwas an der Blockchain verändert ist eine gewisse Wartezeit nötig. In dieser Zeit wird die Transaktion vom Miner in einen gültigen Block verpackt. Eingeplant ist dafür eine Wartezeit von 10 s.

3. Erneutes Auslesen und Überprüfen der Variablen Greeting mit der Get-Methode. Wenn alles Funktioniert hat ist die Ausgabe jetzt nicht mehr "Hello" sondern "« New String »".

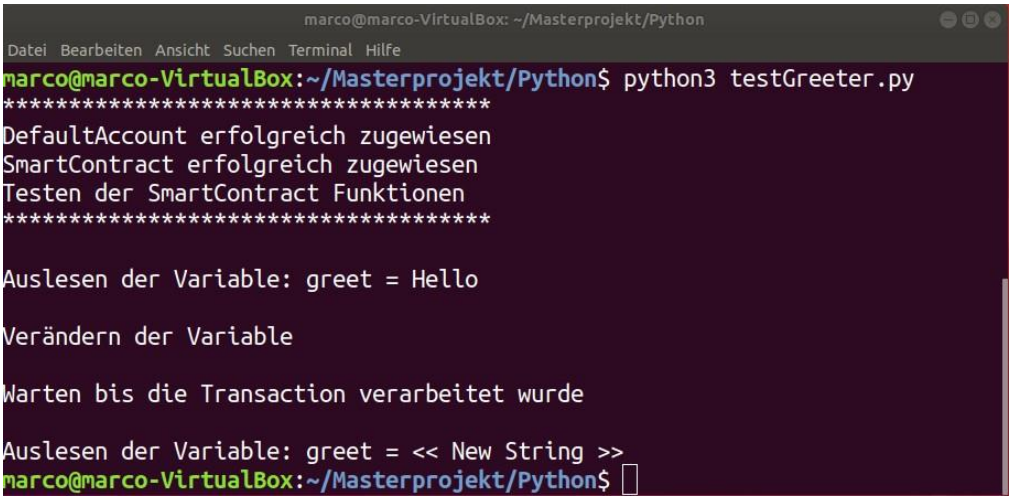


Abbildung D.14: Validierung der Get-/ Set-Methode des Smart Contracts Greeter.

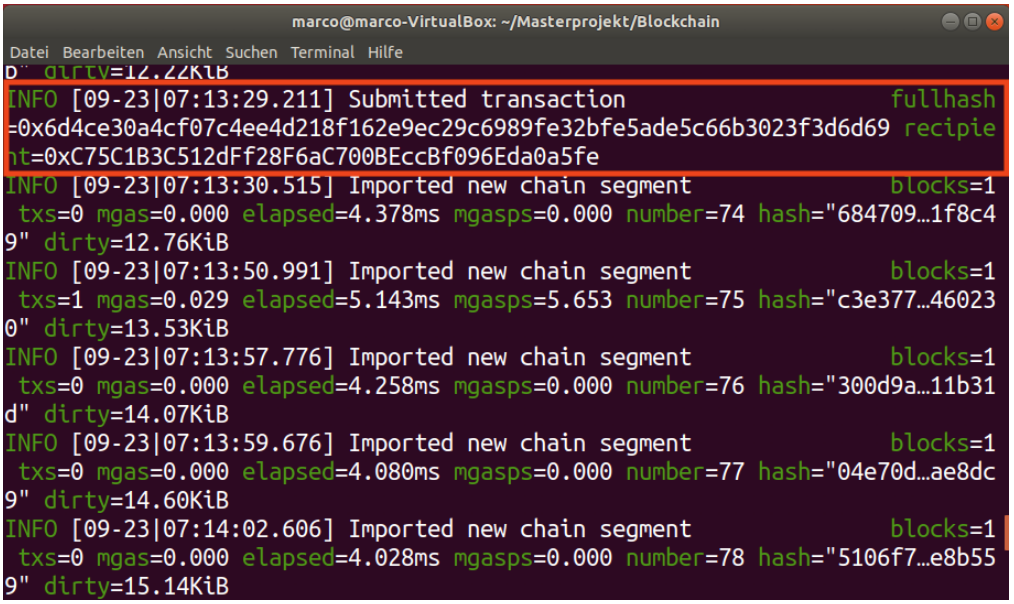


Abbildung D.15: Ausgabe der Blockchain beim Ausführen des Pythonprogramms.

In Abbildung D.14 ist die Ausgabe der Konsole in Python dargestellt. Nach dem Start der Ausführung erscheinen die Ausgaben zur Information des aktuellen Stands. Anschließend startet der Test der Set-/ Get-Methode. Beim ersten Auslesen entspricht der String wie bereits erwartet "Hello". Gegen Ende des kleinen Testprogramms wechselt die Ausgabe allerdings auf den gewünschten Wert "« New String »". Mit dem zweiten Aufruf der Get-Methode ist der Programmablauf beendet. Zur Kontrolle ist in Abbildung D.15 parallel der Verlauf innerhalb der Geth Konsole abgebildet. Sobald die Set-Methode aufgerufen wird, folgt die in rot markierte Ausgabe. Diese Transaktion packt der Miner unter Mac OS in eine der folgenden Blöcke. Mit der erfolgreichen Validierung des Smart Contract Greeter via Python ist die Projektdurchführung Teil 1 damit abgeschlossen.

Auszug aus Studienarbeit Jim Rebholz und Marco Arena, WiSe 2020/2021

E

Implementierung bei der Industrieanwendung

E.1 Ergebnisse des Showcases

Das Ergebnis dieser Arbeit ist ein Showcase, in dem die Umsetzung der dezentralen Regelung und Abrechnung der Netzteilnehmer gezeigt wird. Das Vorgehen wird in einer Anleitung beschrieben.

Sobald die Simulink Simulation gestartet wird können hier die Einund Ausgabewerte der einzelnen Teilnehmer über die Displayund Scope-Blöcke beobachtet werden. Nach Beenden der Simulation kann das Ergebnis als Matlab-Plot gespeichert werden. Zwei Durchläufe mit verschiedenen Einstellungen werden im Folgenden Vorgestellt.

E.2 Lastprofile

In Abbildung E.1 ist die Simulation der Leistung des Haushalts und Industrie über zwei Tage zu sehen. Diese Lastprofile bleiben für beide Versuche gleich.

Der speedup Faktor ist hier mit 360 eingestellt, dadurch kann die Simulationszeit mit einem Verhältnis von 10 Sekunden pro Stunde interpretiert werden (Bsp. 150s entspricht 15h). Das Lastprofil des Haushalt (oben) zeigt deutlich die Überproduktion der PV-Anlage über die Mittagszeit (ca 100-150s bzw. 10Uhr bis 15Uhr) durch die „negative'"Leistung. Das Industrie-Lastprofil ist meistens mit 10 kW konstant bis auf einige Lastspitzen und eine Pause über die Nacht (20Uhr 24Uhr).

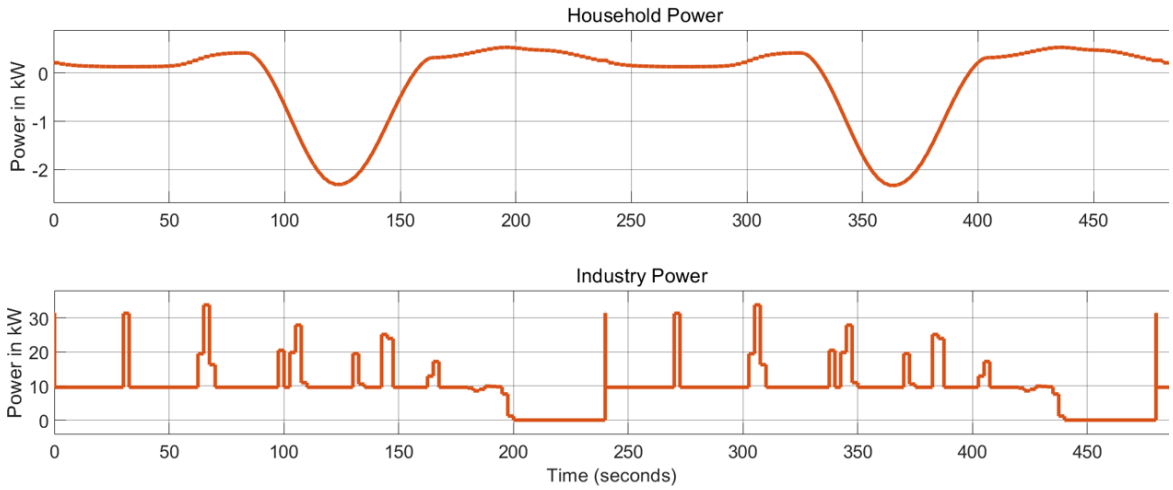


Abbildung E.1: Showcase Ergebnis Lastprofil

E.2.1 Kontostand ohne Bezahlen

Im ersten Versuch wird die Schwelle, ab der bezahlt wird, so hoch gesetzt, dass keine Begleichung der Beträge stattfindet. Der Resultierende Kontostand der Teilnehmer ist in Abbildung E.2 zu sehen.

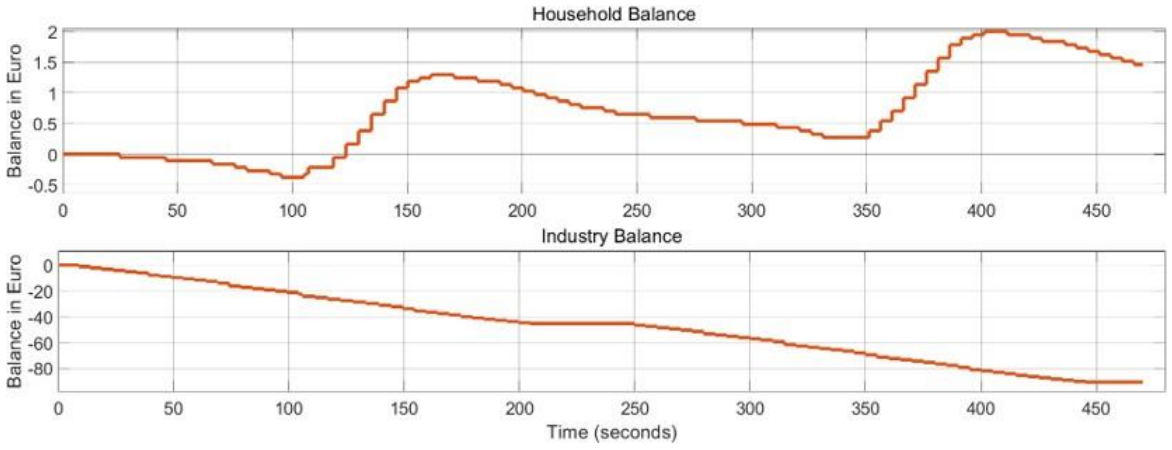


Abbildung E.2: Showcase Ergebnis Balance ohne Bezahlen

Man erkennt, dass der Haushalt durch die Überproduktion über die Mittagszeit dazuverdient. Hier bleibt jedoch der Preis konstant, da der Strom im Netz direkt verbraucht werden kann (10 kW Last der Industrie). Das ist am beinahe linearen Verlauf des Industrie Kontostands zu erkennen. Eine Preissenkung tritt erst auf wenn eine globale Überproduktion vorliegen würde.

E.2.2 Kontostand mit Bezahlen

Nun wird die Schwelle ab der bezahlt wird bei Positivbeträgen auf 2 Euro gesetzt und bei Negativbeträgen auf 10 Euro. Das Ergebnis ist in Abbildung E.3 zu sehen.

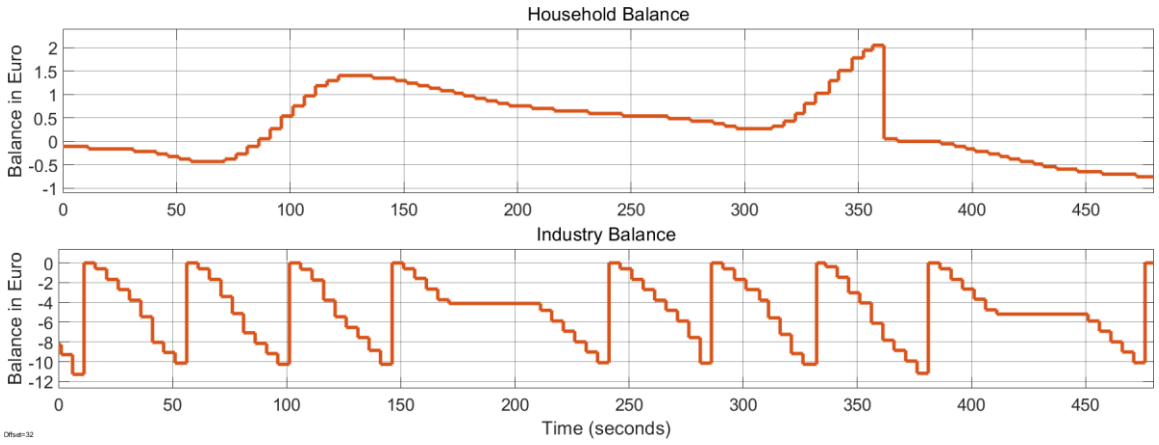


Abbildung E.3: Showcase Ergebnis mit Bezahlen

Durch den hohen Verbrauch in der Industrie wird hier häufig bezahlt, das erste Mal nach einigen Stunden. Dabei handelt es sich um Transaktionen an den Netzbetreiber, da mehr verbraucht als erzeugt wurde. Beim Haushalt wird vom Netzbetreiber auf das Haushaltskonto zum Zeitpunkt 36 Stunden eine Transaktion getätigt, da der Kontostand 2 Euro übersteigt. Die Transaktionen werden jeweils über Smart-Contract Funktionen ausgelöst. Dabei wird Ether als Währung versendet.

Auszug aus Mechatronikprojekt 2 – Nils Hägele, WiSe 2020/2021

F

Leitfaden für Experteninterviews

F.1 Projektvorstellung

Vorstellung ebök Institut: Ressourceneffizienz und Lebenszyklusanalyse Smart Grid: „Kupfer durch Hirnschmalz (sprich Algorithmen) ersetzen" Methodischer Ansatz: Flexibilität wirkt wie ein physischer Speicher (geringinvestiver Ansatz)

Das Projekt folgt 3 Prinzipien:

1. Strom fließt nach den Kirchhoff Gesetzen (Netzdienlichkeit).
2. Regenerative Erzeugung lässt sich nicht steuern aber gut vorhersagen.
3. In der Produktionsplanung besteht viel Flexibilität, die sich für optimierte Fahrpläne (Lastgänge) nutzen lässt. Das wird heute schon häufig beim Lastmanagement eingesetzt (Systemgrenze Netzanschlusspunkt). Zentraler Vorteil: Alle Daten und Prozesskontrolle bleibt in den Betrieben.

Am Fallbeispiel Engpassmanagement (day ahead Stromhandel, EEX und OTC, lokal, wird analog behandelt) nutzen die Teilnehmer eine Blockchain, in die sie ihre geplanten Lastgänge (Verbrauch und Erzeugung) für den nächsten Tag eintragen und (fast) in Echtzeit von einem Smart Contract ein Resultat in Form einer Ampel erhalten. Zu Zeiten von „Grün" sind höhere Lasten möglich, bei „Gelb" sollte die Last gehalten oder vermindert werden und bei „Rot" ist mit Versorgungsunterbrechungen (d.h. Produktionsausfällen) zu rechnen. Mit dieser Information lässt sich der Fahrplan iterativ erneut optimieren.

Der Schwarm der Teilnehmer optimiert die Auslastung der Betriebsmittel ohne steuernden Eingriff des VNB.

F.2 Fragen

F.2.1 Geschäftsmodelle

Wir befinden uns in einer regulatorischen Innovationszone, können also Ideen verfolgen, die aktuell nicht regelkonform sind.

1. Die Teilnehmer eines VK sollten eine Vergütung für Netzdienlichkeit erhalten. Wie lässt sich netzdienliches Verhalten incentivieren? Kennen Sie Beispiele oder Ideen für Flexibilitätsmärkte bei Strombeschaffung oder Engpassmanagement?
2. Der Betrieb der IT-Infrastruktur bedeutet für viele SW Neuland. Wir haben drei Strategien identifiziert:
   – Den Integrator, der die neuen Aufgaben in seine Abläufe einbaut,
   – den Orchestrator, der die Schnittstellen zu Dienstleistern schafft und
   – den White Label Nutzer, der die gesamte Leistung unter seinem Namen abwickeln lässt
   Welche Merkmale wie Größe, Unternehmenskultur, Portfolio, Digitalisierungsgrad etc. sehen Sie bei diesen drei Typen oder kennen Sie noch andere Strategien?

F.2.2 Regulatorik

Welche Hindernisse sehen Sie bei lokalem Stromhandel (Mieterstrom, Energiegenossenschaften, …)?

Welche Hindernisse sehen Sie bei der Vergütung von netzdienlichem Verhalten?

Welche Entwicklungen sehen Sie, die genannten Hemmnisse zu überwinden?

F.2.3 Blockchain

Welche Erfahrungen haben Sie mit BC-Projekten (Ressourcen, Planung, Implementation, Betrieb)?

Wurden Ihre Erwartungen erfüllt?

Wie war das Feedback der Kunden?

Autor: Claus Kahlert