



FOM Hochschule für Oekonomie & Management

Hochschulzentrum Düsseldorf

Seminararbeit

im Studiengang Wirtschaftsinformatik

zur Erlangung des Grades eines

Bachelor of Science (B.Sc.)

über das Thema

Resilientes Design von Edge Cloud Systemen

von

Aljosha Krüger

Betreuer : Dr. Aykut Bußian

Matrikelnummer : 639098

Abgabedatum : 5. Februar 2025

Inhaltsverzeichnis

| | |
|---|-------------|
| Abbildungsverzeichnis | IV |
| Tabellenverzeichnis | V |
| Abkürzungsverzeichnis | VI |
| Symbolverzeichnis | VII |
| Glossar | VIII |
| 1 Einleitung | 1 |
| 1.1 Zielsetzung | 1 |
| 1.2 Aufbau der Arbeit | 1 |
| 2 Theoretische Grundlagen von Edge Cloud Systemen | 2 |
| 2.1 Begriffserklärungen | 2 |
| 2.1.1 Edge Cloud Systeme | 2 |
| 2.1.2 Konzepte von Software Defined Networking | 2 |
| 2.2 Probleme der Resilienz in Edge Cloud Systemen | 4 |
| 3 Resiliente Edge Cloud Systeme in der Praxis | 6 |
| 3.1 3 Schichten Architektur | 6 |
| 3.2 Aufbau eines resilienten Edge Cloud Systems | 6 |
| 3.3 Verwendete Software, Editor und Zusatzpakete | 11 |
| 3.3.1 Windows 8+ | 11 |
| 3.3.2 Mac OSX und iOS | 11 |
| 3.3.3 Online | 11 |
| 3.4 Dokumentenklasse | 12 |
| 3.5 Grafiken | 12 |
| 3.6 Quellcode | 12 |
| 3.7 Tabellen | 14 |
| 3.8 Biblatex | 14 |
| 3.8.1 Erklärung | 14 |
| 3.8.2 Beispielfußnoten | 15 |
| 3.9 Abkürzungen | 15 |
| 3.10 Formeln | 16 |
| 3.11 Symbole | 17 |

| | |
|--|-----------|
| 3.12 Glossar | 17 |
| 3.13 Listen und Aufzählungen | 17 |
| 3.13.1 Listen | 17 |
| 3.13.2 Aufzählungen | 17 |
| 3.13.2.1 Tiefste Ebene 1 | 18 |
| 3.13.2.2 Tiefste Ebene 2 | 18 |
| 3.14 Skript zum Kompilieren | 18 |
| 3.15 PlantUML | 18 |
| 4 Fazit | 18 |
| Anhang | 20 |
| Literaturverzeichnis | 21 |

Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 1: Verzeichnisstruktur der \LaTeX -Dateien | 1 |
| Abbildung 2: Titel der Abbildung hier | 13 |

Tabellenverzeichnis

| | |
|--|----|
| Tabelle 1: Beispieltabelle 1 | 14 |
|--|----|

Abkürzungsverzeichnis

| | |
|-------------|-----------------------------------|
| ARP | Adress Resolution Protocol |
| DDOS | Distributed Denial of Service |
| DORA | Digital Operations Resilience Act |
| IP | Internet Protocol |
| IT | Informationstechnologie |
| MAC | Medium Access Control |
| SDN | Software Defined Networking |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |

Symbolverzeichnis

| | |
|--------------|---|
| A | Aufrechter Buchstabe |
| \mathbb{N} | Menge aller natuerlichen Zahlen ohne die Null |

Glossar

Glossar In einem Glossar werden Fachbegriffe und Fremdwörter mit ihren Erklärungen gesammelt.. 17

Glossaries Glossaries ist ein Paket was einen im Rahmen von LaTeX bei der Erstellung eines Glossar unterstützt.. 17

1 Einleitung

Dies soll eine \LaTeX -Vorlage für den persönlichen Gebrauch werden. Sie hat weder einen Anspruch auf Richtigkeit, noch auf Vollständigkeit. Die Quellen liegen auf Github zur allgemeinen Verwendung. Verbesserungen sind jederzeit willkommen.

1.1 Zielsetzung

Kleiner Reminder für mich in Bezug auf die Dinge, die wir bei der Thesis beachten sollten und \LaTeX -Vorlage für die Thesis.

1.2 Aufbau der Arbeit

Kapitel 2 enthält die Inhalte des Thesis-Days und alles, was zum inhaltlichen erstellen der Thesis relevant sein könnte. In Kapitel 3 Resiliente Edge Cloud Systeme in der Praxis findet ihr wichtige Anmerkungen zu \LaTeX , wobei die wirklich wichtigen Dinge im Quelltext dieses Dokumentes stehen (siehe auch die Verzeichnisstruktur in Abbildung 1).

Abbildung 1: Verzeichnisstruktur der \LaTeX -Dateien

| Name | Änderungsdatum | Typ | Größe |
|---|------------------|--------------------|-------|
|  abbildungen | 29.08.2013 01:25 | Dateiordner | |
|  kapitel | 29.08.2013 00:55 | Dateiordner | |
|  literatur | 31.08.2013 18:17 | Dateiordner | |
|  skripte | 01.09.2013 00:10 | Dateiordner | |
|  compile.bat | 31.08.2013 20:11 | Windows-Batchda... | 1 KB |
|  thesis_main.tex | 01.09.2013 00:25 | LaTeX Document | 5 KB |

Quelle: Eigene Darstellung

2 Theoretische Grundlagen von Edge Cloud Systemen

2.1 Begriffserklärungen

2.1.1 Edge Cloud Systeme

todo In den letzten Jahren haben Cloud-Systeme immer mehr an Popularität gewonnen. Firmen, die ihre Informationstechnologie (IT)-Systeme on-premise betrieben haben, also auf angemieteten Servern, haben teilweise ihre Aktivität in die Cloud verlegt. Die Cloud ist ein entferntes großes Rechenzentrum, das üblicherweise von einem großen Cloud-Betreiber betrieben wird. Die größten Cloudbetreiber im Jahr 2024 todo bitte Quelle finden, waren Amazon Web Services, Microsoft Azure und Google. Die Cloud bietet Unternehmen einige Vorteile, beispielsweise lässt sie sich einfach konfigurieren und den Bedürfnissen des Unternehmens anpassen. Die benötigte Rechenkapazität lässt sich außerdem einfach skalieren. Allerdings bringt die Cloud auch Nachteile mit sich. Das Rechenzentrum der Cloud ist meistens räumlich weit entfernt vom Unternehmen, das Cloud-Dienste in Anspruch nimmt. Dies resultiert in einer hohen Latenz, also Verzögerung bei der Übertragung des Datenverkehrs zwischen Endbenutzer und Cloud. Für den Großteil der Dienste ist diese Latenz zu vernachlässigen. Mit dem Aufkommen von modernen Diensten, die mit Real-Time-Verarbeitung arbeiten, ist Latenz allerdings zu einem Problem geworden. Beispiele für solche Dienste sind beispielsweise Fahrzeug-zu-Fahrzeug-Kommunikation, die unter anderem im teilautonomen Fahren eingesetzt wird, aber auch ortsabhängige Dienste. Jene Real-Time-Dienste benötigen eine möglichst niedrige Latenz, um ordnungsgemäß zu funktionieren. Um die Vorteile einer Cloud nutzen zu können, ohne dabei die Latenz in die Höhe zu treiben, kam das Konzept der Edge Cloud Systeme auf. Edge Cloud Systeme betreiben Edge Computing, also Rechenarbeit am Netzwerkrand. In diesem Konzept findet die Rechenarbeit nicht in der Cloud statt, allerdings auch nicht im Netzwerk des Unternehmens. Diese wird stattdessen in eine sogenannte Edge-Cloud verlagert. Dies ist ein kleineres Rechenzentrum, dass sich räumlich näher an der Datenquelle befindet. Die räumliche Nähe reduziert die Latenz, da Datenpakete nur noch einen Bruchteil der Strecke überwinden müssen. Dies ermöglicht es, auch Latenz-sensitive Dienste in einer Cloud-Umgebung zu betreiben, um alle Vorteile dieser zu nutzen.

2.1.2 Konzepte von Software Defined Networking

todo Mit dem Aufkommen neuartiger Services verändern sich auch die Anforderungen an Netzwerke. Netzwerke müssen beispielsweise in der Lage sein, flexibel auf verschiedene

Situationen zu reagieren. Besonders Skalierbarkeit ist eine wichtige Eigenschaft moderner Netzwerke. Um dies zu ermöglichen, setzt man zunehmend auf sogenannte Software Defined Networking (SDN)-Ansätze statt klassischen Netzwerken. Klassische Netzwerke sind statisch. Dies bedeutet, dass die Netzwerktopologie, also der Aufbau des Netzwerkes fest ist. Soll diese Topologie verändert werden, so bedeutet dies einen hohen manuellen Konfigurationsaufwand. Auch sind in einem klassischen Netzwerk die sogenannten Flussregeln dezentral definiert. Eine Flussregel ist eine Regel, an welche Geräte ein Netzwerkgerät bestimmte Datenpakete weiterleiten soll. Dezentral definiert bedeutet, dass jedes Netzwerkgerät seine Flussregeln lokal gespeichert hat. Soll eine Flussregel also geändert werden, so muss zum einen auf das Netzwerkgerät zugegriffen werden, zum anderen müssen die Auswirkungen, die das Ändern der Regel mit sich bringt, betrachtet werden. SDN-Systeme verfolgen einen anderen Ansatz, diese trennen die Datenebene von der Kontrollebene. Die Datenebene bezeichnet die Komponente eines Netzwerkes, die die eigentliche Aufgabe des Netzwerkes verrichten. Teil der Datenebene sind Netzwerkgeräte, die Datenpakete nach bestimmten Regeln weiterleiten. Das können Router, Switches oder auch andere Komponenten des Netzwerkes sein. Die Kontrollebene auf der anderen Seite hat die Aufgabe, das Netzwerk kontinuierlich zu überwachen. Auf ihr befinden sich die sogenannten SDN-Controller, das sind Softwareprogramme, die die Aufgabe haben, Änderungen im Netzwerk zu erkennen und dieses dynamisch umzukonfigurieren, um dieser Änderung gerecht zu werden. Ein SDN-Controller könnte beispielsweise eine erhöhte Anzahl an Client-Anfragen registrieren und zusätzliche virtuelle Server hochfahren, um diese Anfragen zu bearbeiten. Die statische Netzwerktopologie der klassischen Netzwerke wird so durch ein sich dynamisch anpassendes Netzwerk ersetzt. (1.Quelle)todo Um die Potenziale von SDN in Gänze zu nutzen, wird es häufig mit Virtualisierung kombiniert. Virtualisierung ermöglicht Skalierbarkeit des Netzwerkes, da es so möglich ist, eine gewünschte Anzahl an Netzwerkgeräten virtuell zur Verfügung zu stellen. Werden diese virtuellen Instanzen nicht mehr benötigt, so kann man sie flexibel deaktivieren. Zur Virtualisierung von Netzwerkgeräten wie Switchen werden häufig Softwareprogramme wie beispielsweise OpenvSwitch eingesetzt (Ende 1. Quelle)todo. Um die dezentralen Flussregeln der Netzwerkgeräte durch ein zentrales Programm wie den SDN-Controller zu steuern, benutzen die Geräte ein Protokoll wie OpenFlow, das eine Kommunikation zwischen Netzwerkgeräten und SDN-Controllern und damit die dynamische Umkonfiguration des Netzes ermöglicht. SDN-Controller können so als Betriebssystem des Netzwerkes fungieren. Ähnlich wie ein Betriebssystem die Hoheit über Speicher und Prozessor besitzt und festlegt, welche Programme die Hardware nutzen dürfen, so besitzt der Controller die Hoheit über das Netzwerk und legt fest, wie Datenflüsse innerhalb des Netzwerkes weitergeleitet werden. SDN-Systeme bieten einige Vorteile gegenüber

klassischen Netzwerken. Dazu zählt neben der bereits erwähnten Skalierbarkeit auch, dass sie in der Lage sind, Strategien zur Lastverteilung umzusetzen. Zum einen können sie sich bei einer insgesamt erhöhten Datenlast anpassen, zum anderen reagieren sie flexibel, wenn einzelne Teile innerhalb des Netzwerkes überlastet sind und verteilen die Datenströme gleichmäßig innerhalb des Netzwerkes. Dies wird durch eine allgemein bessere Übersicht über das Netzwerk ermöglicht. SDN ermöglicht es also Netzwerke so flexibel anzubieten, wie Virtualisierung dies mit Instanzen von Betriebssystemen ermöglicht.

Die müssen hier solange stehen bis sie verwendet werden damit man das Abkürzungsverzeichnis testen kann

Internet Protocol (IP): Internet Protocol

2.2 Probleme der Resilienz in Edge Cloud Systemen

todo

Die Resilienz von IT-Systemen ist ein immer wichtiger werdendes Thema. Das zeigen unter anderem auch Gesetze wie der Digital Operations Resilience Act (DORA), der von der Europäischen Union erlassen wurde, um Resilienz in Unternehmen zu verbessern. Im Grund bedeutet Resilienz die Widerstandsfähigkeit von IT-Systemen gegen anormale Zustände (todo bitte Quelle finden). Diese Resilienz soll auch in Edge Cloud Systemen umgesetzt werden, denn da die Wichtigkeit von Edge Cloud Systemen immer weiterwächst, steigt auch der Bedarf an resilienten Lösungen.

Edge Cloud Systeme sind komplexe Systeme, was dazu führt, dass es viele Probleme für die Umsetzung einer resilienten Lösung gibt. Davon sollen allerdings 4 betrachtet werden. Zum einen besteht das Problem von Kommunikationsabbrüchen zwischen Switchen und ihren SDN-Controllern. Wie bereits beschrieben, steuert ein SDN-Controller die ihm zugewiesenen Netzwerkgeräte, überwacht diese und aktualisiert ihre Flussregeln. Dazu müssen Sie eine Netzwerkverbindung aufbauen. Führt ein nicht erwartetes Ereignis zu dem Ausfall der Netzwerkverbindung, so muss eine resiliente Lösung den Betrieb auch weiterhin zumindest für eine bestimmte Zeit lange weiterführen. Eine Lösung, in der keine Resilienz-steigernden Maßnahmen eingebaut wurden, könnte bei Abbruch der Verbindung den Betrieb nicht mehr fortsetzen. Da die SDN-Controller keine Kontrolle mehr über die Netzwerkgeräte besitzen, kann das Netzwerk nicht mehr dynamisch umkonfiguriert werden.

Ein weiteres Problem ist die Überlastung eines Servers durch zu viele Client-Anfragen. Besonders bei schwankenden Nutzungsraten, muss eine resiliente Lösung fähig sein, mit

Überlastung umzugehen. Wird beispielsweise eine erhöhte Anzahl an Anfragen auf einen bestimmten Service registriert, so muss die Last automatisch auf verschiedene Server aufgeteilt werden, die den gleichen Service anbieten. Dies erhöht die Resilienz gegenüber einer großen Menge an Anfragen, die durch Distributed Denial of Service (DDOS)-Angriffe, aber auch durch legitime Nutzung zustande kommen können. Eine Lösung ohne solche Maßnahmen würde alle Anfragen an einen einzelnen Server weiterleiten, der aufgrund der Anfragen überlastet wäre. Dies resultiert in einer hohen Wartezeit. Diese hohe Wartezeit allerdings wäre ein Widerspruch gegen die Ziele von Edge Cloud Systemen, die eigentlich die Reduzierung von Latenz beinhalten. Eine höhere Wartezeit durch Überlastung eines Servers ist daher nicht tolerierbar.

Zusätzlich zu Servern können auch einzelne Netzwerkverbindungen überlastet werden. Im Prinzip führt dies zu den gleichen Folgen wie bei einer Überlastung eines Servers. Wird in einer nicht-resilienten Lösung ein Algorithmus verwendet, der alle Datenpakete zum selben Ziel auch denselben Weg durch das Netzwerk schickt, so führt dies bei einer hohen Anfragedichte dazu, dass eine bestimmte Verbindung schnell überlastet ist. Zur gleichen Zeit sind allerdings Alternativrouten kaum genutzt und wären zwar in der Theorie länger, in der Praxis durch den Stau auf der Hauptleitung allerdings schneller. Ein resilientes Design muss dies berücksichtigen und den Datenverkehr bei Auslastung einer Verbindung im Netzwerk verteilen. Dazu kann das System durch seinen SDN-basierten Aufbau dynamisch die Flussregeln der Netzwerkgeräte anpassen. Zu der Überlastung einzelner Leitung kommt noch, dass innerhalb einer Leitung Transmission Control Protocol (TCP) und User Datagram Protocol (UDP)-Datenpakete konkurrieren. UDP-Datenverkehr tendiert dazu, einen Großteil einer Verbindung in Anspruch zu nehmen. Bei normaler Auslastung stellt dies noch kein Problem dar, ist die Leitung allerdings überlastet, so wird TCP-Datenverkehr benachteiligt. Das resiliente System muss also durch die Verteilung des Datenverkehrs im Netzwerk dafür sorgen, dass TCP-Datenverkehr nicht benachteiligt wird.

Ein weiteres Problem ist die Orchestrierung der SDN-Controller. Eine redundante Menge an SDN-Controllern bietet an sich zwar schon Resilienz, bringt allerdings auch Probleme mit sich. Diese redundanten SDN-Controller müssen nämlich auch koordiniert werden. Ziel dabei ist es, dass jedes Netzwerkgerät einen Controller besitzt, der diesen kontrolliert, auf der anderen Seite dürfen allerdings nicht gleichzeitig mehrere Controller für ein Netzwerkgerät zuständig sein. Ein Netzwerkgerät ohne Controller wäre kein funktionierendes Teil im SDN-Netzwerk, denn es gäbe keine Möglichkeiten, die Flussregeln dieses Netzwerkgeräts zu ändern. Verarbeiten allerdings mehrere Controller die Nachrichten eines Netzwerkgeräts, so führt dies auch zu Problemen, da diese das Gerät mit widersprüchlichen Befehlen versorgen könnten.

3 Resiliente Edge Cloud Systeme in der Praxis

3.1 3 Schichten Architektur

Um Resilienz in Edge Cloud Systemen zu implementieren sind Resiliente Edge Cloud Systeme nach einer 3-Schichten Architektur aufgebaut. Dieses ist am 2-Schichten-Modell eines SDN-Systems angelehnt, es wird allerdings um eine zusätzliche Schicht ergänzt. Die unterste Ebene ist die Datenebene. Wie auch in klassischen SDN-Systemen befinden sich auf der Datenebene die Netzwerkgeräte, die den Datenverkehr weiterleiten, beispielsweise Router oder Switches. Die mittlere Schicht ist die Kontrollschicht, oder auch Verbindungslogikschicht. Dort wird die Logik definiert, nach der die Geräte der unteren Ebene die Datenpakete weiterleiten sollen. Dies wird durch die SDN-Controller durchgeführt. Ein klassisches Netzwerk besteht also aus nur einer Schicht, der Datenschicht. Um eine flexible Anpassung des Netzwerkes zu ermöglichen, setzt SDN eine 2-Schichten-Architektur ein, ergänzt also die Kontrollschicht. Um Resilienz in das System zu implementieren ist es logisch, diese 2-Schichten-Architektur wiederum, um eine Schicht zu erweitern. Um Resilienz zu gewährleisten, wird auf der mittleren Schicht nicht nur ein SDN-Controller eingesetzt, sondern eine Menge redundanter Controller. Um diese Menge zu koordinieren, wird ein Cluster Manager eingesetzt. Dieser ist Teil der dritten und damit höchsten Ebene, der Resilienzmanagementebene. Dieser Controller steuert die große Anzahl an SDN-Controllern.

3.2 Aufbau eines resilienten Edge Cloud Systems

Im Verlaufe dieser Arbeit wurden bereits einige Probleme der Resilienz von Edge Cloud Systemen erläutert. Diese 4 Hauptprobleme waren vor allem Kommunikationsabbrüche zwischen Controllern und Netzwerkgeräten, Überlastung eines Servers oder einer Verbindung und die Orchestrierung der SDN-Controller. Nun sollen mögliche Lösungen für diese Probleme betrachtet werden, die dazu beitragen können, Resilienz in Edge Cloud Systemen zu implementieren.

Ein großes Problem sind Abbrüche der Verbindung zwischen den SDN-Controllern und den Netzwerkgeräten. Neben Maßnahmen zur Vermeidung eines solchen Abbruchs müssen vor allem Regeln definiert werden, wie sich die Komponenten im Falle des Abbruchs verhalten sollen. In SDN-Systemen werden Switches häufig mit einer speziellen Software wie OpenvSwitch virtualisiert. Eine solche Software stellt für softwarebasierte Switches zwei Funktionsmodi zur Verfügung, also Einstellungen, die definieren, wie sich

der Switch verhalten soll. Diese beiden Funktionsmodi sind der secure-Modus, sowie der standalone-Modus. Ist bei einem Switch der secure-Modus ausgewählt, so wird dieser nach einer Verbindungstrennung versuchen, sich wieder mit dem zuständigen SDN-Controller zu verbinden. Alle einkommenden Pakete verwirft er allerdings. Wie die Bezeichnung also vermuten lässt, ist dieser Modus geeignet, um sicherzustellen, dass kein Paket fälschlicherweise an ein falsches Ziel weitergeleitet wird, weil die Flussregeln des Netzwerkgerätes veraltet sind und nicht durch den Controller aktualisiert werden konnten. Bevor dies geschehen kann, reagiert das Netzwerkgerät und stellt seine Tätigkeit ein. Der Standalone-Modus ermöglicht ein eigenständiges Handeln des Netzwerkgerätes. Auch hier registriert ein Switch die Verbindungstrennung und versucht sich wieder mit dem zuständigen Controller zu verbinden. Die Pakete allerdings verwirft er nicht, sondern sendet sie gemäß seiner Flussregeln weiter. Dies ermöglicht es, dass der Switch eine Zeit lang seine Funktion normal erfüllt, birgt allerdings das Risiko, dass sich in der Zwischenzeit die Flussregeln ändern und somit Pakete an einen falschen Empfänger geschickt werden. Bei diesem Funktionsmodus überwiegen die Vorteile allerdings die Nachteile, sodass dieser Modus in einem resilienten Design gewählt werden sollte. Besonders bei nur kurzen Verbindungsausfällen ist es unwahrscheinlich, dass sich Flussregeln zwischenzeitlich ändern. Solche kurzfristigen Verbindungsausfälle haben auf den Standalone-Modus nur einen geringen Einfluss, ein Netzwerk, das mit dem Secure-Modus konfiguriert wurde, ist allerdings bei jedem Ausfall in seiner Tätigkeit eingeschränkt, weil jene Komponenten des Netzwerkes, die von den Ausfällen betroffen sind ihre Tätigkeit einstellen.

Eine weitere Herausforderung für die Resilienz in Edge Cloud Systemen ist die Überlastung einzelner Server, die bestimmte Services anbieten. Während manche Server kaum genutzt werden, können andere überlastet sein, da ihre Services gefragt sind und sie daher zahlreiche Anfragen bearbeiten müssen. SDN-Systeme sind dynamischer als klassische Netzwerke und können daher gut geeignet sein, um Mechanismen zur Lastverteilung zu implementieren. Solche Mechanismen helfen bei der Implementierung eines resilienten Edge Cloud Systems. Wichtig ist es zunächst, die Kopplung von Software und Hardware so weit wie möglich aufzuheben. Dazu kann Virtualisierung verwendet werden. Die Server einer Serverfarm eines Edge Cloud Systems werden daher virtualisiert, so können sie je nach Bedarf erzeugt oder verworfen werden. Dies ermöglicht eine flexible Skalierbarkeit und Anpassbarkeit an die momentane Datenlast. Allein Virtualisierung ist allerdings nicht ausreichend, um die Resilienz des Systems gegenüber Datenlast zu steigern. Die einkommende Last muss weitergehend fair auf die vorhandenen virtuellen Server aufgeteilt werden, denn eine ungleiche Verteilung führt zu unterschiedlicher Belastung der Server und damit zu nicht notwendigen Wartezeiten. Dieser Verteilung muss innerhalb des Edge-Cloud-Netzwerkes erfolgen, kann aber nicht von den Servern selbst durchgeführt

werden. Der Client hat keine Kenntnis von der internen Struktur des Edge Cloud Netzwerkes, kann also seine Daten nicht direkt auf die verschiedenen Server adressieren. Für diesen muss also eine Möglichkeit bestehen, seine Anfrage an eine zentrale Stelle zu senden, an der diese dann verteilt wird. Auch eine Verteilung bei den Servern ist nicht praktikabel, da es sich um virtuelle Server handelt, die nach Belieben aktiviert und deaktiviert werden. Die Verteilung muss daher bei den SDN-Controllern stattfinden, die somit zwischen dem Client, also der Quelle der Anfrage und dem Server, also dem Ziel der Anfrage stehen. Die SDN-Controller verteilen die Daten gleichmäßig auf die Server. Dieses Konzept wird mithilfe des Address Resolution Protocol (ARP) realisiert. Das Protokoll dient der Umwandlung von logischen IP-Adressen in physische Medium Access Control (MAC)-Adressen. Dazu wird in einem ARP-Request vom Client aus, eine Anfrage an alle Netzwerkegeräte innerhalb eines Netzwerkes geschickt, die die gesuchte IP-Adresse enthält. In einem ARP-Reply schickt das Gerät mit der gesuchten IP-Adresse seine MAC-Adresse zurück an den Client. Nun können beide Geräte über die MAC-Adresse kommunizieren. Dieses Protokoll kann man nutzen, um den SDN-Controller als Man-in-the Middle zu etablieren, also einen Zwischenmann, der zwischen Client und Server steht. In der Praxis könnte dies beispielsweise aussehen, dass ein Client, der einen Service der Edge Cloud in Anspruch nehmen möchte, einen ARP-Request absetzt, um herauszufinden, wohin er seine Datenpakete schicken soll. Dieser ARP-Request wird von einem der SDN-Controller verarbeitet. Dieser ermittelt die Auslastung der virtualisierten Server und entscheidet darauf basierend, welcher Server die Anfrage des Clients bearbeiten soll. Dieser virtuelle Server wird automatisch aktiviert. Dem Client schickt der SDN-Controller allerdings nicht die direkte IP-Adresse des virtuellen Servers. Stattdessen schickt er eine virtuelle IP-Adresse, die alle Server der Serverfarm repräsentiert. Damit entsteht für den Client der Eindruck, er würde die gesamte Zeit über mit demselben Server kommunizieren, in der Realität kommuniziert er allerdings mit dem SDN-Controller, seine Anfragen werden nicht immer vom selben Server bearbeitet, sondern immer von dem, der die höchste Kapazität hat. Für den Client bietet dies den Vorteil, dass er seine Anfrage immer nur an diese IP-Adresse schicken muss, der Lastverteilungsmechanismus wird für den Client unsichtbar ausgeführt. Mithilfe dieses Mechanismus lassen sich Edge Cloud Systeme resilienter gestalten, da Schwankungen in der Datenlast häufig zu Überlastung und Wartezeiten führen, was dem Ziel einer möglichst geringen Latenz entgegenläuft. Ein funktionierender Lastverteilungsmechanismus sollte also Teil eines resilienten Edge Cloud Systems sein.

Neben Servern können auch Verbindungen überlastet werden. Das Resultat ist das allerdings das gleiche, es kommt zu langen Wartezeiten und Stau. Im Prinzip folgt die Lösung für dieses Problem dem der Serverüberlastung. Statt hier den Datenverkehr zwischen mehreren Servern aufzuteilen, teilt man hier die Daten auf verschiedene

Verbindungen auf, um der Überlastung einzelner Verbindungen entgegenzuwirken. Das angestrebte Konzept nennt sich Multipath-Routing, also einer Routing-Art, bei der nicht jedes Paket zum selben Ziel zwangsläufig auch denselben Weg nimmt. In der klassischen Alternative dem Single-Path Routing ist dies genauso. Zunächst wird dort durch einen Routing-Algorithmus der kürzeste, beziehungsweise der schnellste Weg ermittelt und anschließend werden alle Pakete über diesen Weg geleitet. Das führt allerdings dazu, dass dieser Weg schnell überlastet wird, die anderen Verbindungen aber nicht genutzt werden. Mit Multi-Path-Routing beugt man dem entgegen, da die Pfadkosten dynamisch den realen Bedingungen im Netzwerk angepasst werden und jedes Paket immer den schnellsten Weg nimmt. Auch wenn das bedeutet, dass nicht alle Pakete des Datenstroms dieselbe Verbindung nutzen. Ein solcher Multipath-Routing-Ansatz kann mit Hilfe des OpenFlow Protokolls und der Python-Bibliothek NetworkX realisiert werden. OpenFlow stellt Select Groups zur Verfügung. Eine Select Group ist eine Sammlung verschiedener Aktionen, die auf ein Datenpaket angewendet werden können, um es zu seinem Ziel zu leiten. Diese einzelnen Aktionen werden Buckets genannt. Ein Bucket hat ein spezifisches Gewicht, also eine Wahrscheinlichkeit, mit der diese Aktion ausgewählt wird. Die Gewichte der Buckets werden durch einen Routing-Algorithmus bestimmt, der die Pfadkosten der Netzwerkverbindungen ermittelt. Die zu versendenden Pakete werden gemäß den Gewichten auf die einzelnen Buckets verteilt. So bekommen Verbindungen mit hoher Auslastung weniger Pakete zugeteilt als Verbindungen mit niedrigerer Auslastung. Dieser Mechanismus allein sorgt allerdings nur für eine Umverteilung der Pakete. Es wird schnell dazu kommen, dass die Alternativverbindungen durch die hohe Zahl der ihnen zugewiesenen Pakete auch überlastet werden, während sich die zuvor überlasteten Verbindungen erholen. Um den Datenfluss dauerhaft gleichmäßig zu verteilen, müssen die Pfadkosten kontinuierlich aktualisiert werden. Ziel ist es, dass die Pfadkosten so realistisch wie möglich die realen Bedingungen des Netzwerkes beschreiben. Dazu müssen die SDN-Controller regelmäßig Statistiken zur Performance des Netzwerkes von den Switchen abrufen. Ist eine vorher überlastete Verbindung nun weniger ausgelastet, so sinken die Pfadkosten dieser Verbindungen, was dann wiederum zu einem höheren Bucket-Gewicht und damit einer wieder steigenden Anzahl an Paketen führt. Im Umkehrschluss führt eine Erhöhung der Auslastung zu höheren Pfadkosten, damit einem niedrigeren Bucket-Gewicht und weniger Paketen. Dieser Mechanismus verteilt automatisch den Datenverkehr innerhalb des Netzwerkes und verringert somit das Risiko von überlasteten Verbindungen. Dies bringt außerdem einen weiteren positiven Effekt. Innerhalb einer Verbindung konkurrieren verschiedene Datenströme um die Verbindung. In überlasteten Verbindungen kann es daher dazu kommen, dass bestimmte Datenströme einen unfairen Vorteil bekommen. UDP-Pakete nutzen beispielsweise die gesamte Bandbreite, die sie benötigen, ohne dabei Rücksicht

auf andere Datenströme zu nehmen. TCP-Datenpakete könnten daher in überlasteten Verbindungen von UDP verdrängt werden, obwohl sie eigentlich eine höhere Priorität haben könnten. Der Lastverteilungsmechanismus hilft dabei, überlastete Verbindungen zu verhindern und damit auch den unfairen Wettbewerb der Datenströme einzudämmen.

Um die Resilienz zu verbessern, wird das SDN-System nicht nur mit einem SDN-Controller, sondern mit mehreren redundanten Controllern betrieben. Dies bringt eine bessere Skalierbarkeit und Ausfallsicherheit, allerdings auch eine Herausforderung mit sich. Die Menge an redundanten SDN-Controllern muss orchestriert und organisiert werden, damit sich die einzelnen Controller nicht gegenseitig behindern. Befugnisse und Eingriffsbereiche eines jeden Controllers müssen also zu jedem Zeitpunkt klar definiert sein. Um diese Orchestrierung zu gewährleisten, müssen die Controller verschiedene Rollen annehmen, die ihre Einsatzbereiche festlegen. Ein Controller mit der Master-Rolle kontrolliert und steuert die ihm zugewiesenen Switches. Er kann außerdem wichtige Protokollnachrichten empfangen, beispielsweise Packet In. In einem Cluster kann immer nur ein Controller zum Master werden. Ein Controller, dem die Rolle Slave zugewiesen wurde, überwacht zwar auch das System, erhält allerdings nicht die Protokollnachrichten wie Packet In. So fungiert immer nur ein Controller des Netzwerkes als Hauptcontroller. Die Zuweisung der Rollen der Controller erfolgt durch einen Cluster Manager und nach einem festgelegten Verfahren. Jeder SDN-Controller des Netzwerkes schickt eine zufällig gewählte Ganzzahl an den Clustermanager. Dieser vergleicht die erhaltenen Nummern nun miteinander. Dem Controller mit der niedrigsten Zahl weist er den ersten Platz in der Reihenfolge, also Platz 0 zu, dem mit der zweitniedrigsten Zahl den zweiten Platz. Dies wird für jeden SDN-Controller durchgeführt, bis jeder SDN-Controller eine eindeutige Nummerierung besitzt. Außerdem wird die Gesamtzahl der Controller im Netzwerk abgespeichert. Der Controller mit der höchsten Nummerierung wird zum Master. Alternativ können die verschiedenen SDN-Controller die Arbeitslast auch so aufteilen, dass kein zentraler Master notwendig ist, um Nachrichten zu verarbeiten. In diesem Konzept hat jeder Controller die Rolle Equal. So ist allerdings ein verteilter Entscheidungsalgorithmus notwendig, der bestimmt, welcher Controller eine eingehende Nachricht verarbeiten soll. In diesem Algorithmus prüft jeder Controller, ob folgende Bedingung zutrifft:

$$dpid \bmod (\text{num_server}) == \text{order}.$$

Dpid, also Datapath-Identifizierer ist die eindeutige Zuordnung eines Switches. Es bezeichnet also, von welchem Switch die einkommende Nachricht gesendet wurde. Num_server ist die Gesamtanzahl aller SDN-Controller im System. Order ist die Nummerierung des Controllers, auf dem der Algorithmus jeweils ausgeführt wird. Die Modulo Operation teilt also die Menge der Switches auf die Gesamtzahl der SDN-Controller auf. Ein Controller ist damit immer für die Nachrichten zuständig, die von einem Switch einer bestimmten Gruppe kommen.

Jeder Controller vergleicht das Ergebnis der Operation mit seiner Nummerierung. Diese Bedingung wird also immer nur bei einem Controller gleichzeitig wahr. Dieser Controller verarbeitet dann die Nachricht. Trotzdem ist auch diese Lösung nicht komplett fair, was die Lastverteilung angeht. Kommen von manchen Switchen statistisch öfter Nachrichten, so werden diese auch immer demselben Controller zugewiesen. Um dies zu vermeiden, kann man statt nach Switches nach den Nachrichten an sich aufteilen. Dazu muss die Dpid in der Bedingung durch eine eindeutige Nummerierung der ankommenden Nachricht ersetzt werden. So wird jede ankommende Nachricht immer einem anderen Controller zugewiesen. Dieser Entscheidungsalgorithmus sorgt für absolut faire Verteilung der zu bearbeitenden Nachrichten auf die einzelnen Controller.

Siehe auch Wissenschaftliches Arbeiten¹. Damit sollten alle wichtigen Informationen abgedeckt sein ;-)² Hier gibt es noch ein Beispiel für ein direktes Zitat³ TODO das hier sind andere Zitierarten, die bitte entfernen wenn nicht mehr gebraucht

3.3 Verwendete Software, Editor und Zusatzpakete

3.3.1 Windows 8+

- MikTex: 2.9, 32-bit
- Biblatex: 3.5, Zusatz: Biber.exe
- Editor: TexStudio (kann ich empfehlen), Notepad++

3.3.2 Mac OSX und iOS

- MacTeX: <https://tug.org/mactex>
- Editor: TexPad <https://www.texpadapp.com>

3.3.3 Online

Overleaf ist eine Online-Anwendung mit der Ihr direkt im Browser an eurer Thesis schreiben könnt. Bis 1GB Größe und maximal 60 Einzeldateien könnt ihr Overleaf kostenlos nutzen: <https://www.overleaf.com/>

¹ Vgl. Balzert, H., Bendisch, R., Kern, U. et al., Wissenschaftliches Arbeiten, 2008, S. 1.

² Vgl. ebd., S. 1.

³ Ebd., S. 1.

3.4 Dokumentenklasse

Eigentlich hatte Prof. Finke empfohlen die Dokumentklassen „Book“ oder „Report“ für die Erstellung der Bachelor-Thesis zu verwenden, da diese über weitere Gliederungsebenen verfügen. Ich verwende dennoch eine leicht modifizierte Komaskript-Klasse „scrartcl“, mit der Erweiterung um eine Ebene. Siehe (skripte/weitereEbene.tex). Das Skript stammt irgendwo aus den Netz und übersteigt meine \LaTeX -Fähigkeiten. Dadurch kann ich über eine weitere Ebene in der Arbeit verfügen, ohne mich mit der Modifikation von Kapitel-Seiten rumschlagen⁴ zu müssen. Diese Quelle ist nur zur Demonstration und hat keinen inhaltlichen Bezug hierzu. Es werden übrigens nur die Quellen im Literaturverzeichnis angezeigt, die auch referenziert sind.

3.5 Grafiken

Das Paket `\usepackage{float}` ermöglicht es die Grafiken und Tabellen an der Stelle im Text zu positionieren, wo diese im Quelltext stehen (Option H). Ansonsten würde \LaTeX diese dort unterbringen, wo es typographisch sinnvoll wäre - das wollen wir ja nicht ;-).

Die Breite der Grafiken am Besten relativ zum Text angeben.

3.6 Quellcode

Quellcode kann auf unterschiedliche Arten eingebaut werden. Zum einen kann es hier durch direktives Einbinden in der Kapitel-Datei geschehen.

```
1 | % Hier wird aufgezeigt, wie man eine Grafik einbindet, es wird also in der PDF
   | angezeigt,
2 | % da es in einem Quellcode-Listing steht.
3 | % Auch wenn es hier faelschlicherweise als LaTeX-Befehl angezeigt wird.
4 | \includegraphics[width=0.9\textwidth]{sup}
```

Bei längeren Quellcode-Listings empfiehlt es sich jedoch auf eine externe Datei im Ordner Quellcode zu verlinken und diese einzubauen:

```
1 | <!-- So können Tabs definiert werden -->
2 | <ul class="tabs">
3 |     <li class="tab-title">
4 |         <div class="tab-content">
5 |             </div>
6 |     </li>
7 | </ul>
```

⁴ Vgl. Tanenbaum, A., Computernetzwerke, 2003, S. 5.

Statt dem Package Istlisting, welches direkt auf Tex basiert, kann auch das Package minted verwendet werden. Dieses Package basiert auf python-pygments und unterstützt weit mehr Sprachkonstrukte als Istlisting. Um das Paket zu verwenden muss es eingebunden werden und zusätzlich python-pygments installiert sein. (Dies ist mit im Dockerfile vorhanden. Für die anderen Compile-Methoden, wie das native verwenden von Tex Live findet sich hier die Installationsanleitung für das minted Paket: <https://ctan.org/pkg/minted?lang=de>)

Damit das kompilieren ohne Python trotzdem möglich ist, ist die Funktion standardmäßig ausgebaut. Deshalb muss zusätzlich in der Datei

```
thesis_main.tex \usepackage{minted}
```

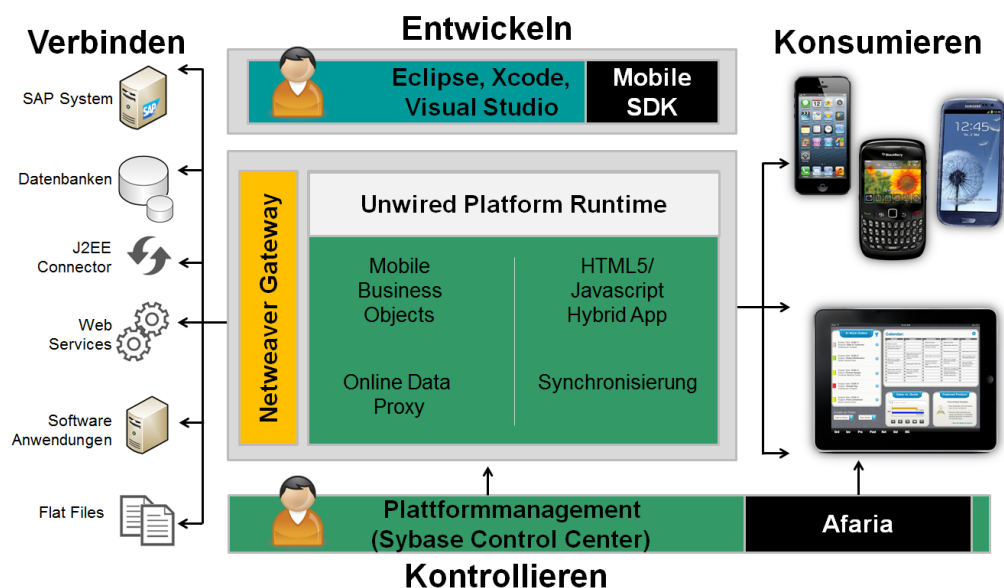
wieder einkommentiert werden.

Minted lässt sich dann ganz ähnlich zu Istlisting verwenden:

```
1 \begin{minted}{c}
2     int main() {
3         printf("hello, world");
4         return 0;
5     }
6 \end{minted}
```

Da der Pfad zu den Abbildungen im Hauptdokument definiert wurde, muss hier nur noch der Name des Bildes ohne Dateiendung stehen (sup).

Abbildung 2: Titel der Abbildung hier



Quelle: Eigene Darstellung

3.7 Tabellen

Tabelle 1: Beispieltabelle 1

| Abkürzung | Beschreibung | Berechnung |
|------------------------------------|-----------------------------------|------------|
| MEK | Materialeinzelkosten | |
| MGK | Materialgemeinkosten | + ↑ * |
| FEK | Fertigungseinzelkosten | |
| FGK | Fertigungsgemeinkosten | + ↑ * |
| SEKF | Sondereinzelkosten der Fertigung | |
| = Herstellungskosten | | |
| VwGK | Verwaltungsgemeinkosten | + ↑ * |
| VtGK | Vertriebsgemeinkosten | + ↑ * |
| SEKvt | Sondereinzelkosten des Vertriebes | |
| = Selbstkosten | | |
| + Gewinnaufschlag | | |
| + Rabatte | | |
| = Nettoverkaufspreis (NVP) | | |
| + Umsatzsteuer | | |
| = Bruttoverkaufspreis (BVP) | | |

Quelle: In Anlehnung an *Beckert, A., Beckert, S., Escherich, B.*, Mobile Lösungen, 2012a, S. 4

3.8 Biblatex

3.8.1 Erklärung

Von den vielen verfügbaren Literatur-Paketen habe ich mich für Biblatex entschieden. Die Anforderungen der FOM sollten hiermit erfüllt sein. Ich habe bisher nur Einträge „@book“ getestet. Wie immer steckt der Teufel hier im Detail und es wird sich später herausstellen, ob Biblatex eine gute Wahl war. Die Anpassungen hierfür liegen unter skripte/modsBiblatex. Ich verwende das Backend Biber, welches bib-Dateien in UTF-8 verarbeiten kann.

In der für den Leitfaden 2018 aktualisierten Version sind außerdem Beispiele für „online“,⁵ also Webseiten, und „article“,⁶ also wissenschaftliche Artikel, enthalten.

Laut Leitfaden sollen maximal 3 Autoren genannt werden und danach mit „et. al.“ bzw. „u.a.“ ergänzt werden. Damit im Literaturverzeichnis auch nur max. 3 Autoren stehen, muss man beim Füllen der literatur.bib-Datei darauf achten auch nur 3 einzutragen. Weitere Autoren

⁵ Vgl. *Brink, S.*, AngularJS, 2018.

⁶ Vgl. *Decker, F.*, Koalitionsaussagen, 2009, S. 140.

kann man einfach mit „and others“ ergänzen. Siehe Eintrag für „Balzert.2008“. Zitiert man dann diese Werk, werden auch in der Fussnote alle Autoren korrekt genannt wie in dieser Fußnote⁷ zu sehen ist.

Hat man dagegen mehr als 3 Autoren in der bib-Datei hinterlegt, stehen im Literaturverzeichnis alle drin. In der Fussnote dagegen, steht nur einer⁸, was dem Leitfaden widerspricht.

Die Anzahl von 3 wird übrigens über die Option „maxcitenames=3“ des biblatex-Packages gesetzt. Man muss selbst schauen, dass die Anzahl der Autoren in den Bib-Dateien mit der Optionseinstellung übereinstimmt.

3.8.2 Beispielfußnoten

Diese Fussnote soll zeigen, wie mit einem „von“ vor dem Namen des Autors umgegangen wird⁹. Man muss für die korrekte Sortierung eines solchen Namens im Literaturverzeichnis einen „sortkey“ setzen.

Diese Fussnote soll zeigen, wie mit einer Online-Quelle ohne Jahresangabe umgegangen wird¹⁰.

Diese Fußnote¹¹ ist nur dazu da zu zeigen, wie mit mehreren Quellen des selben Autors aus dem selben Jahr umgegangen wird, wenn das Stichwort gleich bleibt¹² oder sich ändert¹³. Laut Leitfaden sollte bei gleichem Autor, Jahr und Stichwort ein Buchstabe an die Jahreszahl gehen. Zum Beispiel 2012a.

Die folgenden Fußnoten dienen dazu zu zeigen, dass die Nummern von zwei direkt aufeinanderfolgende Fußnoten mit Komma getrennt werden.^{14,15}

3.9 Abkürzungen

Abkürzungen werden mithilfe des Pakets Acronym eingebunden. Alle Abkürzungen sollten in der Datei acronyms.tex mithilfe des

⁷ Vgl. Balzert, H., Bendisch, R., Kern, U. et al., Wissenschaftliches Arbeiten, 2008, S. 1.

⁸ Vgl. Balzert, H. et al., XYZWissenschaftliches Arbeiten, 2008, S. 1.

⁹ Vgl. von Lucke, J., Heuermann, R., Poder, H. et al., Treiber, 2018, S. 1.

¹⁰ Vgl. Belastungsdienst, Bürgerservicenummer, o. J.

¹¹ Vgl. Beckert, A., Beckert, S., Escherich, B., Mobile Lösungen, 2012a, S.1.

¹² Vgl. Beckert, A., Beckert, S., Escherich, B., Mobile Lösungen, 2012b, S.2.

¹³ Vgl. Beckert, A., Beckert, S., Escherich, B., Mobile Lösungen2, 2012, S.3.

¹⁴ Vgl. Beckert, A., Beckert, S., Escherich, B., Mobile Lösungen, 2012b, S.2.

¹⁵ Vgl. von Lucke, J., Heuermann, R., Poder, H. et al., Treiber, 2018, S. 1.

`\acro`

Befehls festgelegt werden. Im Text werden diese dann mit

`\ac{Abkürzung}`

benutzt. Bei der ersten Verwendung einer Abkürzung wird der Begriff in beiden Formen dargestellt. So wie hier: **WYSIWYG!** (**WYSIWYG!**). Nur wenn eine Abkürzung tatsächlich verwendet wird erscheint sie auch im Abkürzungsverzeichnis.

Sollte es im Abkürzungsverzeichnis zu Anzeigefehlern kommen kann dies daher rühren, dass eine Abkürzung verwendet wird, die länger ist als **WYSIWYG!**. In diesem Fall müsst ihr in der Datei `acronyms.tex` den Parameter `[WYSIWYG]` durch eure längere Abkürzung ersetzen.

3.10 Formeln

Um eine Formel nach links auszurichten muss sie zwischen `&` und `&` eingesetzt werden:

Formel 1: Erste Formel

$$L_P = 10 \lg \cdot \frac{P}{1mW} \quad (1)$$

Quelle: In Anlehnung an *Beckert, A., Beckert, S., Escherich, B.*, Mobile Lösungen, 2012a, S. 4

Etwas mehr Text.

Quelle: Vgl. *Hochschule für Ökonomie & Management*, Onlinecampus, 2018

Ansonsten wird sie mittig ausgerichtet test.

Formel 2: Zweite Formel

$$L_P = 10 \lg \cdot \frac{P}{1mW} \quad (2)$$

Quelle: In Anlehnung an *Beckert, A., Beckert, S., Escherich, B.*, Mobile Lösungen, 2012a, S. 4

3.11 Symbole

Das hier ist ein definiertes Symbol: \mathbb{N} und das hier auch A . Symbole werden in der Datei Skripte symboldef.tex zentral definiert.

3.12 Glossar

Begriffserklärungen bzw. das Glossar wird mithilfe des Pakets Glossaries eingebunden. Alle Begriffe die erklärt werden sollen, sollten in der Datei glossar.tex mithilfe des

```
\newglossaryentry
```

Befehls festgelegt werden. Im Text werden diese dann mit

```
\gls{Begriff}
```

benutzt.

3.13 Listen und Aufzählungen

3.13.1 Listen

- ein wichtiger Punkt
- noch ein wichtiger Punkt
- und so weiter

3.13.2 Aufzählungen

1. Reihenfolge ist hier wichtig
2. Dieser Punkt kommt nach dem ersten
3. Da sollte jetzt eine 3 vorne stehen

3.13.2.1 Tiefste Ebene 1

Dies ist die tiefste Gliederungsebene. Sollten doch mehr Ebenen benötigt werden, muss eine andere Dokumentenklasse verwendet werden.

3.13.2.2 Tiefste Ebene 2

Der zweite Punkt in dieser Ebene ist zur Erinnerung daran, dass es nie nie niemals nur einen Unterpunkt geben darf.

3.14 Skript zum Kompilieren

Latex will ja bekanntlich in einer bestimmten Reihenfolge aufgerufen werden:

```
1 | lualatex thesis_main.tex
2 | biber thesis_main
3 | lualatex thesis_main.tex
4 | lualatex thesis_main.tex
5 | thesis_main.pdf
```

Dies ist der Inhalt der Batchdatei „compile.bat“.

3.15 PlantUML

```
1 | \begin{plantuml}
2 | @startuml
3 | Class01 <|-- Class02
4 | Class03 *-- Class04
5 | Class05 o-- Class06
6 | Class07 .. Class08
7 | Class09 -- Class10
8 | @enduml
9 | \end{plantuml}
```

4 Fazit

Zusammenfassend ist Resilienz in Edge Cloud Systemen ein wichtiges anzustrebendes Ziel. Das Aufkommen moderner Services und das damit verbundene Aufkommen von Edge Cloud Services sind der Grund hierfür. Der Aufbau von Resilienz ist auch politisch gewünscht, was verschiedene Gesetze zeigen. Eine große Herausforderung

für die Resilienz in solchen Systemen ist die Lastverteilung, sowohl zwischen den Netzwerkverbindungen, als auch den einzelnen Knoten des Netzwerkes. Ansätze von SDN sind gut geeignet, um diese Herausforderung zu bewältigen. Mithilfe dieser Ansätze können verschiedene Strategien zur Lastverteilung dynamisch implementiert werden. Dabei ist zwingend zu unterscheiden, in welchem Bereich die Lastverteilung eingesetzt werden soll. Strategien zur Lastverteilung zwischen den Servern eines Netzwerkes beugen nur bedingt gegen Überlastung einzelner Verbindungen vor. Diese beiden Arten von Lastverteilung sind wichtige Komponenten und müssen in einer Strategie zur Lastverteilung berücksichtigt werden. Darüber hinaus bilden allerdings auch Koordinationsprobleme eine Herausforderung für Edge Cloud Systeme da. Beispielsweise führt die Redundanz der SDN-Controller zwar wie gewollt zu höherer Resilienz, allerdings auch zu Managementaufwand, das im schlechtesten Fall zu Abschlägen in der Leistung führt. Durch moderne Koordinierungsmethoden, beispielsweise bestimmten Algorithmen mit geringem Overhead kann der Koordinierungsaufwand weitestgehend reduziert werden. Ein weiteres Problem ist das der Verbindungsabbrüche zwischen SDN-Controllern und Netzwerkgeräten. Der Zusammenhalt des SDN-Netzwerkes macht das System dynamisch, allerdings auch anfälliger für Störungen. Die negativen Auswirkungen einer Störung können minimiert werden, indem jedes Netzwerkgerät einen notwendigen Grad an Eigenständigkeit erhält. Diese Eigenständigkeit führt zwar nicht zum optimalen Ergebnis, reicht aber als Übergangslösung in einer Störungsphase, wobei die Alternative einen teilweisen oder ganzen Ausfall des Netzwerkes bedeutet. Die behandelten Herausforderungen stellen keine abgeschlossene Liste dar. Ein wichtiger Teil der Resilienz ist auch die Widerstandsfähigkeit gegen Cyber-Attacken, diese wurde hier nicht behandelt. Stattdessen wurde sich rein auf die nicht-böswilligen Herausforderungen fokussiert. Trotzdem bieten Strategien zur Erhöhung der IT-Sicherheit in Edge Cloud Systemen weitere Forschungsschwerpunkte, die in zukünftigen Arbeiten erarbeitet werden können.

Anhang

Anhang 1: Beispielanhang

Dieser Abschnitt dient nur dazu zu demonstrieren, wie ein Anhang aufgebaut sein kann.







Anhang 1.1: Weitere Gliederungsebene

Auch eine zweite Gliederungsebene ist möglich.

Anhang 2: Bilder

Auch mit Bildern. Diese tauchen nicht im Abbildungsverzeichnis auf.

Abbildung 3: Beispielbild

| Name | Änderungsdatum | Typ | Größe |
|---|------------------|--------------------|-------|
|  abbildungen | 29.08.2013 01:25 | Dateiordner | |
|  kapitel | 29.08.2013 00:55 | Dateiordner | |
|  literatur | 31.08.2013 18:17 | Dateiordner | |
|  skripte | 01.09.2013 00:10 | Dateiordner | |
|  compile.bat | 31.08.2013 20:11 | Windows-Batchda... | 1 KB |
|  thesis_main.tex | 01.09.2013 00:25 | LaTeX Document | 5 KB |

Literaturverzeichnis

- Balzert, Helmut, Bendisch, Roman, Kern, Uwe et al.* (Wissenschaftliches Arbeiten, 2008): Wissenschaftliches Arbeiten: Wissenschaft, Quellen, Artefakte, Organisation, Präsentation, Soft skills, Herdecke [u.a.]: W3L-Verl., 2008
- Balzert2, Helmut, Bendisch, Roman, Kern, Uwe, Schäfer, Christian, Schröder, Marion, Zeppenfeld, Klaus* (XYZWissenschaftliches Arbeiten, 2008): Wissenschaftliches Arbeiten: Wissenschaft, Quellen, Artefakte, Organisation, Präsentation, Soft skills, Herdecke [u.a.]: W3L-Verl., 2008
- Beckert, André, Beckert, Sebastian, Escherich, Bernhard* (Mobile Lösungen, 2012a): Mobile Lösungen mit SAP, 1. Aufl., Bonn: Galileo Press, 2012
- Beckert, André, Beckert, Sebastian, Escherich, Bernhard* (Mobile Lösungen, 2012b): Mobile Lösungen mit SAP, 1. Aufl., Bonn: Galileo Press, 2012
- Beckert, André, Beckert, Sebastian, Escherich, Bernhard* (Mobile Lösungen2, 2012): Mobile Lösungen mit SAP, 1. Aufl., Bonn: Galileo Press, 2012
- Decker, Frank* (Koalitionsaussagen, 2009): Koalitionsaussagen der Parteien vor Wahlen. Eine Forschungsskizze im Kontext des deutschen Regierungssystems, in: Zeitschrift für Parlamentsfragen, 40 (2009), S. 431–453
- von Lucke, Jörn, Heuermann, Roland, Poder, Helmut et al.* (Treiber, 2018): Treiber, Ratgeber, Meinungsmacher, in: *Heuermann, Roland, Tomenendal, Matthias, Bressemer, Christian* (Hrsg.), Digitalisierung in Bund, Ländern und Gemeinden, Berlin: Springer Gabler, 2018, S. 153–213
- Tanenbaum, Andrew* (Computernetzwerke, 2003): Computernetzwerke, 4. Aufl., München: Pearson Studium, 2003

Internetquellen

Belastingdienst (Bürgerservicenummer, o. J.): Was ist eine Bürgerservicenummer (BSN)?, <[https://www.belastingdienst.nl/wps/wcm/connect/bldcontentde/belastingdienst/privatpersonen / sonstige _ themen / buergerservicenummer / was _ ist _ eine _ buergerservicenummer_bsn](https://www.belastingdienst.nl/wps/wcm/connect/bldcontentde/belastingdienst/privatpersonen/sonstige_themen/buergerservicenummer/was_ist_eine_buergerservicenummer_bsn)> (keine Datumsangabe) [Zugriff: 2019-02-26]

Brink, Sascha (AngularJS, 2018): AngularJS - Was ist Angular?, <<https://angularjs.de/buch/was-ist-angularjs>> (2018-12-20) [Zugriff: 2019-01-02 23:30 Uhr]

Hochschule für Oekonomie & Management (Onlinecampus, 2018): Onlinecampus, <<https://www.campus.bildungszentrum.de>> (2018) [Zugriff: 2018-11-01]

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die angemeldete Prüfungsleistung in allen Teilen eigenständig ohne Hilfe von Dritten anfertigen und keine anderen als die in der Prüfungsleistung angegebenen Quellen und zugelassenen Hilfsmittel verwenden werde. Sämtliche wörtlichen und sinngemäßen Übernahmen inklusive KI-generierter Inhalte werde ich kenntlich machen.

Diese Prüfungsleistung hat zum Zeitpunkt der Abgabe weder in gleicher noch in ähnlicher Form, auch nicht auszugsweise, bereits einer Prüfungsbehörde zur Prüfung vorgelegen; hiervon ausgenommen sind Prüfungsleistungen, für die in der Modulbeschreibung ausdrücklich andere Regelungen festgelegt sind.

Mir ist bekannt, dass die Zuwiderhandlung gegen den Inhalt dieser Erklärung einen Täuschungsversuch darstellt, der das Nichtbestehen der Prüfung zur Folge hat und daneben strafrechtlich gem. § 156 StGB verfolgt werden kann. Darüber hinaus ist mir bekannt, dass ich bei schwerwiegender Täuschung exmatrikuliert und mit einer Geldbuße bis zu 50.000 EUR nach der für mich gültigen Rahmenprüfungsordnung belegt werden kann.

Ich erkläre mich damit einverstanden, dass diese Prüfungsleistung zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Düsseldorf, 5.2.2025

(Ort, Datum)

A handwritten signature in black ink, consisting of a large, stylized 'H' followed by a series of loops and a final flourish.

(Eigenhändige Unterschrift)