

Visualisierung von Multicast-Strömen mit APIC-EM

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2016

Autor(en): Fabian Wirz, Etienne Georgy
Betreuer: Prof. Beat Stettler
Projektpartner: AnyWeb AG

Aufgabenstellung

Einführung

Mit Software Defined Networking (SDN) soll die Intelligenz des Netzwerkes in Controller ausgelagert werden. Dadurch sollen Netzwerkservices flexibler gesteuert und überwacht werden können. Cisco hat mit dem «Application Policy Infrastructure Controller (APIC) Enterprise Module» ein erstes Produkt auf den Markt gebracht, welches die Ideen und Konzepte von SDN umsetzen soll.

In dieser Studienarbeit soll ein typischer Use-Case, der mit herkömmlichen Methoden schwierig zu realisieren ist, mit dem APIC-EM implementiert und getestet werden. Nach einer theoretischen Analyse der Anforderungen soll anhand einer konkreten Testumgebung aufgedeckt werden, ob der SDN Ansatz im Allgemeinen und insbesondere die Cisco Lösung die Versprechungen einlösen kann, wo heute noch die Grenzen liegen und welche zukünftigen Entwicklungen nötig sind.

Aufgabe

In einem grossen Netzwerk mit ungefähr 600 Standorten, zwei grossen Rechenzentren und mehreren Tausend Cisco Netzwerkkomponenten wird IP Multicast eingesetzt. Der Wunsch des Kunden ist es, Informationen über die Ausbreitung der Multicast-Ströme in seinem Netzwerk in Echtzeit dargestellt zu haben. Die Multicast-Ströme sollen auf einer Netzwerktopologie abgebildet werden.

Abstract

Die Anzahl von Multimedia-Inhalten in Computernetzwerken hat in den letzten Jahren erheblich zugenommen. Das zur Verteilung solcher Inhalte genutzte Kommunikationsverfahren Multicast, gewinnt dadurch immer mehr an Bedeutung. Eine Analyse von Multicast-Datenströmen gestaltet sich jedoch schwierig. Es werden unter anderem fortgeschrittene Kenntnisse im Netzwerkbereich benötigt. In modernen Firmennetzwerken wird zudem häufig MPLS-VPN eingesetzt. Die Analyse wird dadurch zusätzlich erschwert, da sich Multicast-Datenströme im MPLS-VPN anders verhalten als in normalen Netzwerken.

Um eine solche Analyse einfacher zu gestalten, wurde eine Applikation entwickelt, die einen Multicast-Datenstrom in einer Netzwerktopologie darstellt. Die Topologie lässt sich mithilfe des APIC-EM erstellen, einem Software Defined Network Controller von Cisco. Zur Visualisierung dient die JavaScript Library NeXt UI. Informationen zu Multicast-Datenströmen werden mittels SSH auf den jeweils involvierten Netzwerkgeräten ausgelesen, persistiert und in ein visuelles Format gebracht.

Als Resultat ist eine Webapplikation entstanden, die einen Multicast-Strom in einem Enterprise-Netzwerk anzeigen kann. Im MPLS-VPN wird zudem erkannt, welchen Multicast Distribution Tree der Multicast-Strom verwendet. Ein Netzwerktechniker kann nun, ohne grossen Aufwand, einen Multicast-Strom in einer Netzwerktopologie ansehen und bei Problemen entsprechend reagieren.

Management Summary

Ausgangslage

Multicast-Verkehr hat in den letzten Jahren erheblich zugenommen und ist heute ein wichtiger Teil jedes Unternehmensnetzwerks. Mit Multicast ist es möglich, mehrere Geräte gleichzeitig zu erreichen, ohne das Netzwerk übermässig zu beanspruchen. Mehrheitlich Multimediadienste wie TV oder Radiosignale werden damit über das Netzwerk verteilt. Die Analyse, wie ein solcher Multicast-Strom durch das Netzwerk fließt, ist aber meistens mit sehr viel Aufwand verbunden. Zusätzlich zu einem guten Netzwerkverständnis sind gute Kenntnisse im Umgang mit der Kommandozeile erforderlich. Eine Applikation, die diese Ströme sammeln und darstellen könnte, würde die Analyse erheblich vereinfachen.

Vorgehen/Technologien

Damit die Applikation einen solchen Multicast-Strom visualisieren kann, braucht es einerseits Informationen zur Netzwerkinfrastruktur und andererseits Informationen zum Multicast-Strom selbst.

Um in Erfahrung zu bringen, welche Netzwerkgeräte im Unternehmen vorhanden sind, bietet das Telekommunikationsunternehmen Cisco den APIC-EM an. Der APIC-EM ist ein Software Defined Network Controller, mit dem ein Netzwerk zentral verwaltet werden kann. Eine Möglichkeit, Informationen zu Multicast-Strömen mit APIC-EM zu erhalten, ist leider nicht vorhanden. Aus diesem Grund ist evaluiert worden, wie die Informationen anderweitig bezogen werden können.

Nebst den klassischen Verfahren über die Kommandozeile wären auch SNMP denkbar gewesen. Mit SNMP können jedoch nicht alle Informationen abgerufen werden. Gerade spezifische Netzwerkabfragen und Infos zu Multicast-Strömen sind mit SNMP nicht möglich. Daher zeigte die Evaluation auf, dass nur über die Kommandozeile brauchbare Informationen ausgelesen werden können; mit dem Nachteil, dass die Ausgaben geparkt werden müssen.

Mithilfe der von APIC-EM erhaltenen Daten zu vorhandenen Netzwerkgeräten können nun auf jedem Gerät die Informationen zum Multicast-Strom abgerufen werden. Die dadurch gesammelten Multicast-Daten werden zusammen mit den APIC-EM Informationen ausgewertet und in einer Datenbank gespeichert.

Für die Anzeige der Daten dient eine Webapplikation, die auf die Datenbank zugreift und dem Benutzer die Netzwerktopologie mitsamt den vorhandenen Multicast-Strömen darstellt. Für eine saubere Darstellung wird das NeXt UI Framework verwendet. Dies ist für die Darstellung von Netzwerktopologien optimiert und eignet sich darum bestens für diesen Anwendungszweck.

Ergebnisse

Mit der entstandenen Applikation kann nun ganz einfach erkannt werden, über welche Netzwerkgeräte welche Multicast-Ströme fließen. Dank des APIC-EM werden die Topologie-Daten bezogen und dargestellt. Eine mühsame Analyse über die Kommandozeile entfällt, und der Netzwerktechniker kann sofort erkennen, wie der Strom im Netzwerk verläuft.

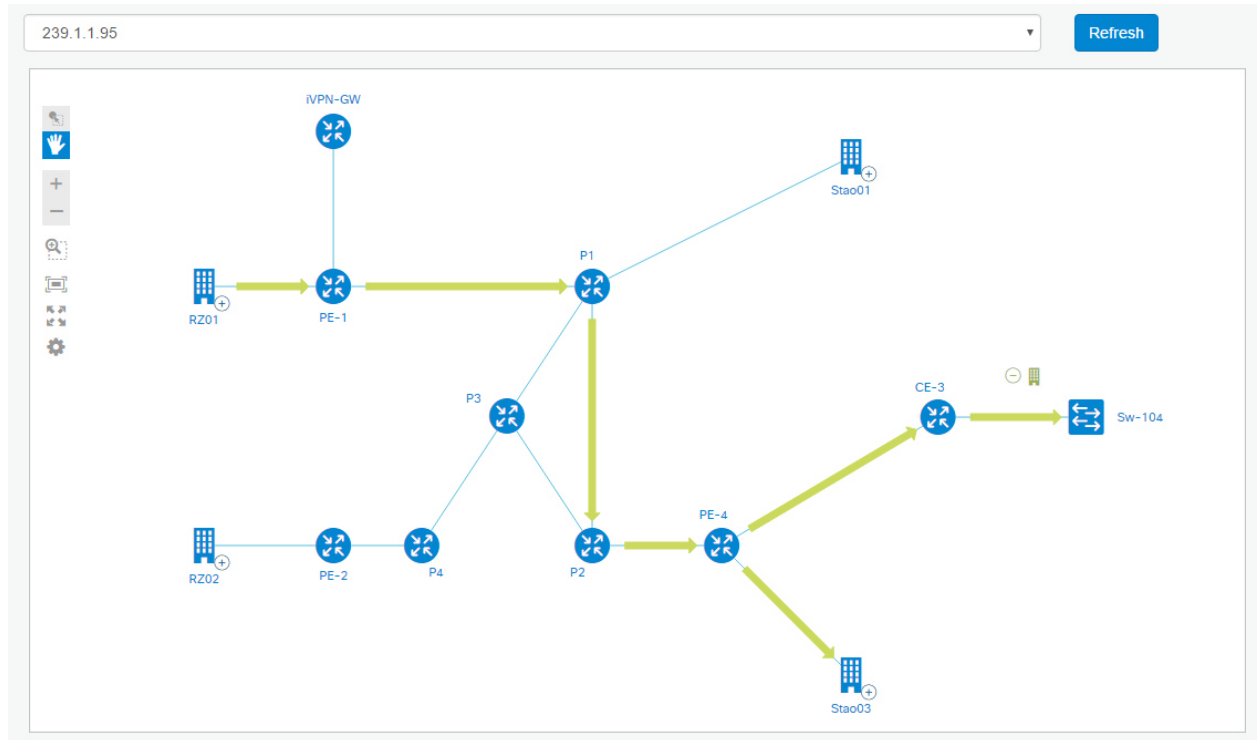


Abbildung 1: Multicast-Strom im Netzwerk

Inhaltsverzeichnis

I. Technischer Bericht	7
1 Anforderungsanalyse	8
1.1 Einführung.....	8
1.2 Allgemeine Beschreibung.....	10
1.3 User Stories	11
1.4 Anforderungen	12
1.5 Qualitätsmerkmale.....	16
2 Domainanalyse.....	17
2.1 Domainanalyse Topologie.....	17
2.2 Domainanalyse Multicast.....	19
3 Architekturentscheidung	21
3.1 Variante 1: Monolith-Architektur	22
3.2 Variante 2: Aufteilung in Services	23
3.3 Entscheidung Aufteilung	24
4 GUI Analyse	25
5 Technologieevaluation.....	27
5.1 Programmiersprache	27
5.2 SNMP oder SSH	27
5.3 Topologie-Informationen beziehen	30
5.4 MVC Framework	32
5.5 Netzwerk-Topologie.....	34
5.6 Datenbank.....	36
5.7 Zugriff auf die Datenbank	36
6 Implementation	37
6.1 Polling Service	37
6.2 Timer Service.....	47
6.3 Parser-Klassen.....	47
6.4 Testlabor	48
6.5 Datenbank.....	50
6.6 MC Viewer.....	54
6.7 NeXt UI	61
7 Testing.....	63
7.1 Unit Test.....	63
7.2 Systemtest.....	64

8	Ergebnisdiskussion & Ausblick	65
8.1	Ergebnisdiskussion	65
8.2	Ausblick	68
9	Statistiken	69
9.1	Codezeilen.....	69
9.2	Arbeitsaufwand.....	69
10	Glossar.....	70
11	Literatur und Quellenverzeichnis.....	72
12	Abbildungen	73
II.	Anhänge.....	74

I. Technischer Bericht

1 Anforderungsanalyse

1.1 Einführung

1.1.1 Multicast

Multicast ist eine Technologie in Computernetzwerken, die dazu verwendet wird, effizient Daten von einer Quelle an deren Empfänger zu senden. Im Normalfall muss eine Quelle zu jedem einzelnen Empfänger im Netzwerk eine eigene Verbindung aufbauen, um diesem Daten senden zu können. Sind nun viele Empfänger vorhanden, muss die Quelle für jeden Empfänger eine einzelne Verbindung aufbauen. Der Ressourcenverbrauch bei der Quelle ist dadurch entsprechend hoch und das Netzwerk wird unnötig belastet. Diese klassische Kommunikationsform wird Unicast genannt.

Multicast hingegen ermöglicht es, dass nur eine einzelne Verbindung benötigt wird. Die Quelle sendet dabei kontinuierlich einen Datenstrom aus, welcher im Netzwerk von den Routern jeweils multipliziert und an diejenigen Empfänger weitergeleitet wird, welche den Empfang dieses Datenstroms auch explizit wünschen. Die Netzwerkbelastung wird dadurch erheblich minimiert.

In dieser Studienarbeit wird oft der Begriff Multicast-Strom verwendet. Der Multicast-Strom bezeichnet hierbei den oben genannten Datenstrom zwischen einer Quelle und deren Empfänger.

Der Multicast-Strom wird durch eine eindeutige IP-Adresse identifiziert. Diese IP-Adresse wird Multicast-Gruppenadresse genannt. Alle IP-Adressen im Bereich 224.0.0.0 – 239.255.255.255 sind Multicast-Gruppenadressen und können für diesen Zweck gebraucht werden. Falls nun eine Adresse im Netzwerk verwendet wird - das heisst, es gibt einen Sender und mehrere Empfänger - so baut das Netzwerk einen Multicast-Baum auf. Jedes Netzwerkgerät, das diesen Multicast-Strom mit dieser Adresse weiterleitet, ist automatisch Teil dieses Baums. Für eine andere Multicast-Gruppe könnte dieser Baum aber schon wieder ganz anders aussehen. Es ist daher ganz interessant zu sehen, wie der Baum aussieht und wie der Multicast Verkehr durch das Netzwerk fließt.

1.1.2 MPLS-VPN

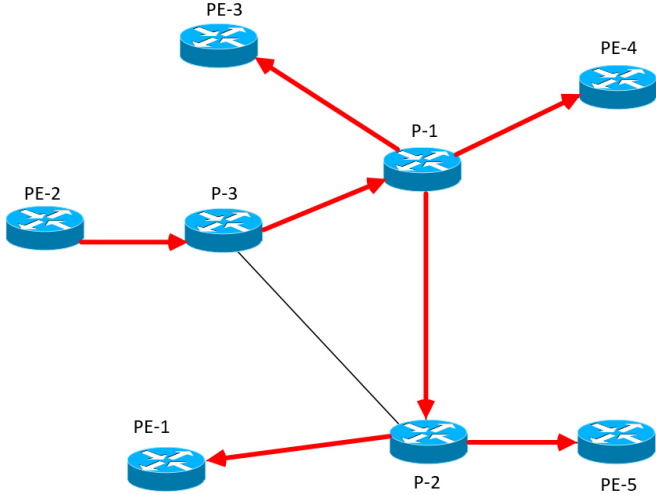
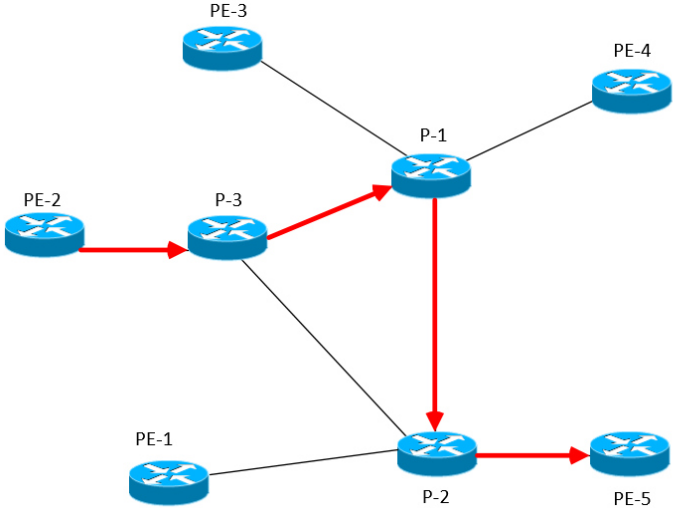
MPLS-VPN wird häufig in grösseren Firmennetzwerken eingesetzt. Die Datagramme werden dabei mit einem MPLS-Header erweitert. In diesem MPLS-Header ist das sogenannte Label enthalten. Das Paket wird nun im MPLS-Netzwerk nicht mehr anhand der IP-Zieladresse weitergeleitet, sondern auf Grund des Labels. Ein MPLS-fähiger Router wird daher als Label-Switched Router bezeichnet. [1]

Je nach Funktion im MPLS-Netzwerk übernimmt jeder Router eine andere Rolle.

In der folgenden Tabelle gibt es eine kurze Erläuterung, was die verschiedenen Rollen bedeuten.

Rolle	Beschreibung
PE Router (Provider Edge Router)	Dieser Router bildet den Übergang vom gerouteten Layer 3-Netzwerk ins MPLS-Netzwerk.
P Router (Provider Router)	Der P Router arbeitet im Core des MPLS-Netzwerk. Dieser ist dafür zuständig, dass die Pakete an den richtigen Empfänger weitergeleitet werden.
CE Router (Customer Edge Router)	Dieser Router ist ganz klassisch ohne MPLS konfiguriert. Er ist normalerweise das erste Netzwerkgerät an einem Standort.

Im MPLS-VPN Netzwerk gibt es im Bereich Multicast Unterschiede zu einem IP-Netzwerk. Der Multicast-Strom kann grundsätzlich auf zwei verschiedene Arten durchs MPLS-Netzwerk fließen.

<p>Default MDT</p>	<p>Beim Default MDT wird der Multicast-Strom an alle PE Router im MPLS-Netzwerk weitergeleitet. Es findet sozusagen ein Broadcast im ganzen MPLS-Netzwerk statt. Ein P Router ist Root und ist für die Verteilung an alle PE Router zuständig. Der Default MDT wird benutzt, falls die Bandbreite des Stroms nur sehr gering ist.</p>  <p><i>Abbildung 2: Beispiel Default MDT</i></p>
<p>Data MDT</p>	<p>Beim Data MDT fließt der Strom direkt vom eingehenden zum ausgehenden PE Router. Der Strom wird somit nicht an alle PE Router verteilt. Der Data MDT wird automatisch bei grösserer Bandbreite gebraucht und ist mit dem normalen Multicast-Strom im Layer 3-Netzwerk vergleichbar.</p>  <p><i>Abbildung 3: Beispiel Data MDT</i></p>

1.2 Allgemeine Beschreibung

1.2.1 Produktfunktion

Mithilfe der Webapplikation muss es einem Netzwerktechniker möglich sein, einen Multicast-Strom zu eruieren. Dazu wird eine Oberfläche benötigt, welche die Topologie eines Netzwerkes darstellt und darauf die verschiedenen Multicast-Ströme einzeichnen kann.

Die Applikation selbst muss dabei die notwendigen Topologie-Informationen aus dem Netzwerk beziehen. Als zentrale Informationseinheit im Netzwerk kommt hierbei der APIC-EM zum Einsatz.

Informationen über die Multicast-Verteilung müssen ebenfalls über das Netzwerk bezogen werden. Anders als bei APIC-EM können diese nicht von einer zentralen Einheit erhalten werden, sondern müssen je nach Möglichkeit per SNMP oder SSH-Verbindung auf den jeweiligen Geräten ausgelesen werden.

Alle gesammelten Daten werden danach innerhalb der Applikation korreliert und über die Benutzeroberfläche dargestellt.

Damit nicht jedermann auf die Benutzeroberfläche zugreifen kann, muss ausserdem eine einfache Benutzerauthentifizierung in Form eines Logins vorhanden sein.

1.2.2 Benutzercharakteristik

Mit dieser Website können Netzwerkadministratoren einfacher sehen, wo im Netzwerk welche Multicast-Ströme fließen. Anhand dieser Informationen können Verbesserungen am Netzwerk gemacht und Probleme besser behoben werden.

1.2.3 Einschränkungen

- Multicast über WLAN muss nicht berücksichtigt werden
- Nur die physikalische Sicht der Multicast-Ströme
- Perimeter und Inter-VPN Firewalls müssen nicht erfasst werden
- Die Applikation darf nichts am Netzwerk verändern

1.3 User Stories

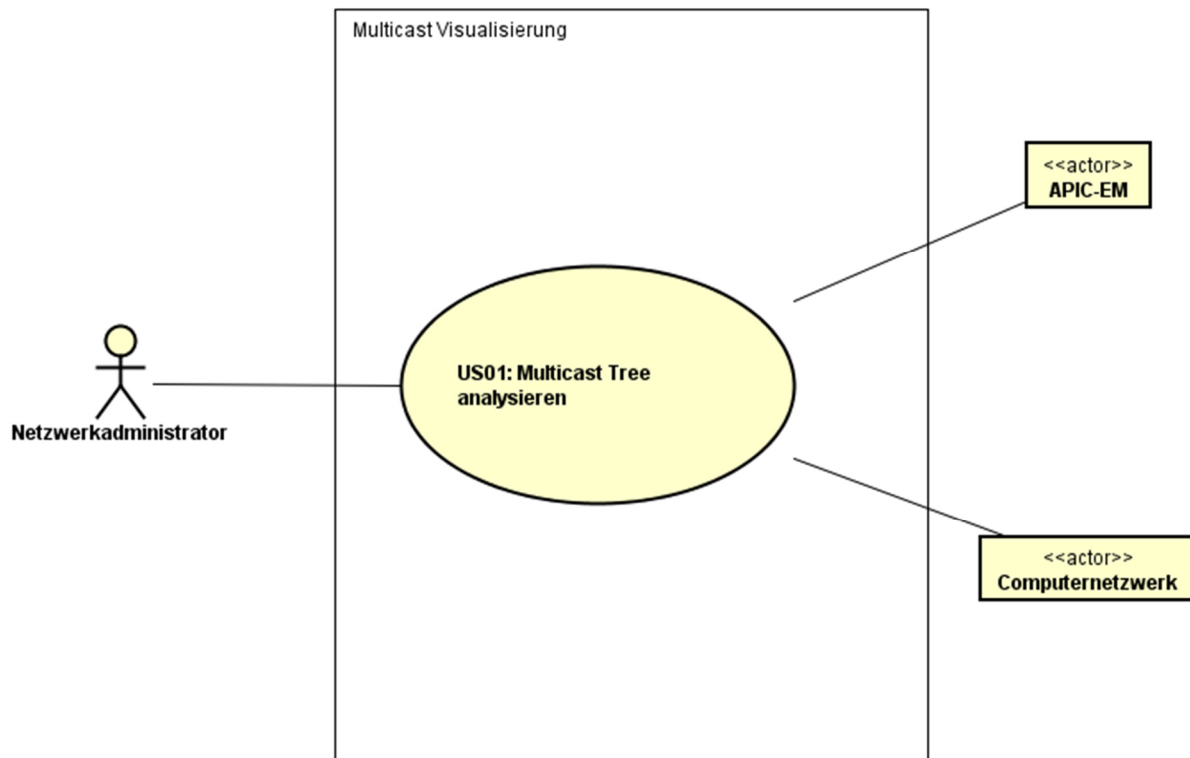


Abbildung 4: User Story US01

1.3.1 US01 Multicast Tree analysieren

Der Netzwerkadministrator als Benutzer möchte den Multicast-Baum analysieren. Als Voraussetzung für die Funktion muss der Benutzer beim System angemeldet sein. Dem Benutzer werden in einer ersten Phase alle Multicast-Ströme angezeigt. Will der User aber nur einen bestimmten Strom sehen, kann er in einer Liste diesen Strom auswählen und er wird ihm in der Applikation dargestellt.

1.3.2 Login Funktionalität

Die Applikation braucht einen einfachen Zugriffsschutz. Der Login sollte aber lediglich verhindern, dass unberechtigte Personen das Netzwerk anschauen können. Es braucht keine Benutzerverwaltung und die Verbindung muss auch nicht verschlüsselt werden.

1.4 Anforderungen

In diesem Kapitel werden die Anforderungen an die Applikation behandelt, welche durch den Industriepartner AnyWeb AG gestellt worden sind. Aus diesen kundenspezifischen Anforderungen sind die technischen Anforderungen entstanden.

1.4.1 Kundenspezifische Anforderungen

Aus dem Pflichtenheft und zusätzlichen Besprechungen ergaben sich folgenden Anforderungen.

Originaltext aus dem Pflichtenheft oder Beschlüsse aus Besprechungen	Interpretierte Anforderung	Priorität
Der Wunsch des Kunden ist es, Informationen über die Ausbreitung der Multicast-Ströme in seinem Netzwerk in Echtzeit dargestellt zu haben.	Die Daten werden regelmässig von den Quellen abgerufen und auf der Website aktualisiert dargestellt.	1
Die Netzwerk-Topologie mit den Multicast-Strömen soll auf einer Webseite dargestellt werden.	Zugriff auf die Applikation über einen Browser.	1
Die Änderungen sollen automatisch in die Darstellung übernommen werden / neue Standorte oder Netzwerkkomponenten sollen automatisch hinzugefügt, nicht mehr aktive Elemente automatisch entfernt werden.	Änderungen an der Topologie müssen in die Topologie übernommen werden.	1
Von der Applikation werden die Netzwerk-Topologie und die Geräteliste über die Rest-Schnittstelle (Rest API) vom APIC-EM abgefragt.	Informationen zur Topologie werden per REST vom APIC-EM abgerufen.	1
Informationen zu den Multicast-Strömen werden über SNMP oder SSH direkt auf den Cisco Geräten abgefragt.	Wenn möglich Multicast-Informationen mit SNMP auslesen, ansonsten direkt mit SSH.	1
Ein Pull down-Menü, welches alle MC-Ströme in Form der Gruppenadresse darstellt. Die Reihenfolge soll nach Zahlenwert aufsteigend sortiert sein.	Auf der Website muss ein Pull down-Menü existieren, in dem alle Multicast-Ströme aufsteigend sortiert sind.	1
Detaillierte Ansicht eines im Pull-down-Menü ausgewählten MC-Stromes.	Im Pull-down-Menü kann der gewünschte Multicast-Strom ausgewählt werden, und dieser wird danach im Netzwerk angezeigt. Die Linie soll dabei als Pfeil, farbig und in Richtung Empfänger dargestellt werden.	1
Anmerkung aus Besprechung: Im Netzwerk gibt es verschiedene VRFs, eine Multicast Gruppe geht über mehrere VRFs.	Applikation muss aus allen VRF im Netzwerk die Multicast-Informationen herauslesen.	1

Programmiersprache JAVA.	Als Programmiersprache muss JAVA verwendet werden, damit sie nachher vom Kunden weiterentwickelt und verbessert werden kann.	1
Die Standorte werden immer alphabetisch sortiert dargestellt. Kommt ein neuer Standort dazu, soll dieser ebenfalls am richtigen Ort eingereiht werden (Stufe 2).	Netzwerkgeräte an einem Standort müssen zusammengefasst und alphabetisch angeordnet werden, Multicast-Strom wird zusammengefasst zum Standort angezeigt.	2
In dieser Stufe sollen alle Netzwerkkomponenten am ausgewählten Standort angezeigt werden (Stufe 3).	Wird auf einen Standort geklickt, sollen alle Netzwerkgeräte mit dem zugehörigen Multicast-Strom angezeigt werden.	2
Die Multicast-Ströme werden in unterschiedlichen Netzwerkbereichen (MPLS-Core, Layer-3, Access) erkannt.	Applikation muss mit MPLS-VPN-Netzwerken umgehen können und die Ströme dort auslesen.	2
Topologie-Map soll durch Klicken auf einen Switch eine Liste mit allen Ports, welche den MC-Strom empfangen oder senden, angezeigt werden.	Multicast-Strom muss bis zum Switch angezeigt werden (IGMP Snooping).	Optional
Informationen zum angeschlossenen Endgerät wie Hostname und IP Adresse.	Angeschlossene Endgeräte müssen mit Informationen zum Hostname und der IP-Adresse auf der Website dargestellt werden.	Optional
Auf der Übersichts-Topologie sollen Zahlen auf einem Link die Anzahl aktiven MC-Ströme auf diesem anzeigen.	Es gibt eine Darstellung, auf der alle momentan aktiven Ströme dargestellt werden; wie viele Ströme auf einem Link aktiv sind, wird mit einer Zahl in der Topologie angezeigt.	Optional
MC-Gruppen, z.B. von Routing-Protokollen, sollen gefiltert werden können. Der Filter soll ein/ausschaltbar sein.	Link-Local Multicast Protokolle (OSPF) sollen aus der Liste der aktiven Multicast-Ströme gefiltert werden können.	Optional
Anmerkung aus Besprechung: Standorte in Standortgruppen zusammenfassen, siehe Pflichtenheft Stufe 1.	Die verschiedenen Gebäude zu einer Gruppe zusammenfassen und alphabetisch sortieren.	Optional

1.4.2 Technische Anforderungen

Aus den interpretierten Anforderungen wurden technische Anforderungen definiert.

Interpretierte Anforderung	Technische Anforderung
Die Daten werden regelmässig von den Quellen abgerufen und auf der Website aktualisiert dargestellt.	Applikation muss fähig sein, in gewissen Abständen Anfragen in das Netzwerk zu verschicken, die empfangenen Daten zu verarbeiten und danach abzuspeichern.
Zugriff auf die Applikation über einen Browser.	Im Hintergrund muss ein Webserver / Applikationsserver laufen, der eine Webseite zur Verfügung stellen kann.
Änderungen an der Topologie müssen in die Topologie übernommen werden	Datenbank muss unerreichbare Netzwerkgeräte nach 24h löschen.
Informationen zur Topologie werden per REST vom APIC-EM abgerufen.	APIC-EM Server muss im Netzwerk vorhanden und erreichbar sein.
Wenn möglich Multicast Informationen mit SNMP auslesen, ansonsten direkt mit SSH.	SNMP und SSH muss im Netzwerk bzw. auf den einzelnen Geräten erlaubt sein. Netzwerkgeräte müssen über eine einheitliche Management IP-Adresse erreichbar sein.
Auf der Website muss ein Pull-down-Menü existieren, in dem alle Multicast-Ströme aufsteigend sortiert sind.	Vorhandene Webapplikation mit gängigem User-Interface.
Im Pull-down-Menü kann der gewünschte Multicast-Strom ausgewählt werden, und dieser wird danach im Netzwerk angezeigt. Die Linie soll dabei als Pfeil, farbig und in Richtung Empfänger dargestellt werden.	Vorhandene Webapplikation mit gängigem User-Interface.
Applikation muss aus allen VRF im Netzwerk die Multicast-Informationen herauslesen.	Informationen zu Multicast-Strömen in VRFs müssen via SNMP oder wenn nötig per SSH ausgelesen werden können.
Als Programmiersprache muss JAVA verwendet werden, damit sie nachher vom Kunden weiterentwickelt und verbessert werden kann.	Programmiersprache JAVA.
Netzwerkgeräte an einem Standort müssen zusammengefasst und alphabetisch angeordnet werden, Multicast-Strom wird zusammengefasst zum Standort angezeigt.	Vorhandene Webapplikation, Architektur muss eine Speicherung von Gruppen/Standorten zulassen.
Wird auf einen Standort geklickt, sollen alle Netzwerkgeräte mit dem zugehörigen Multicast-Strom angezeigt werden.	Vorhandene Webapplikation, die aufgeklappten Netzwerkgeräte müssen sinnvoll auf der Topologie dargestellt werden.

Applikation muss mit MPLS-VPN-Netzwerken umgehen können und die Ströme dort auslesen.	Multicastinformationen in MPLS-Netzwerken müssen per SNMP oder SSH auslesbar sein.
Multicast-Strom muss bis zum Switch angezeigt werden (IGMP Snooping).	IGMP Snooping Table muss per SNMP oder SSH auslesbar sein und in Datenbank gespeichert werden können.
Angeschlossene Endgeräte müssen mit Informationen zum Hostname und der IP-Adresse auf der Website dargestellt werden.	DNS-Server muss beim APIC-EM eingetragen sein. Alle Hosts müssen mit Hostnamen und IP beim DNS-Server eingetragen sein.
Es gibt eine Darstellung, auf der alle momentan aktiven Ströme dargestellt werden; wie viele Ströme auf einem Link aktiv sind, wird mit einer Zahl in der Topologie angezeigt.	Vorhandene Webapplikation.
Link-Local-Multicast-Protokolle (OSPF) sollen aus der Liste der aktiven Multicast-Ströme gefiltert werden können.	Vorhandene Webapplikation.
Die verschiedenen Gebäude zu einer Gruppe zusammenfassen und alphabetisch sortieren.	Vorhandene Webapplikation, Datenbank muss diese Sortierung unterstützen.

1.5 Qualitätsmerkmale

Es gibt diverse Qualitätsmassnahmen, die die Applikation erfüllen muss. Nachfolgend sind alle Qualitätsanforderungen an die Applikation aufgelistet.

1.5.1 Erweiterbarkeit

- Alle optionalen Anforderungen müssen in den Architekturentscheidungen berücksichtigt werden
- Die Architektur der Applikation muss mehr als 20'000 Netzwerkgeräte behandeln können
- Die Applikation unterstützt nur IPv4 Adressen.

1.5.2 Zuverlässigkeit

- Die Applikation muss alle Multicast Ströme im Netzwerk erkennen können und diese mindestens alle zwei Stunden aktualisieren.

1.5.3 Benutzbarkeit

- Mindestens zwei User müssen die Applikation gleichzeitig bedienen können
- User müssen die Multicast-Ströme auf der Topologie eindeutig erkennen können
- Die ganze Topologie muss bei einer normalen Bildschirmgrösse dargestellt werden können.

1.5.4 Sicherheit

- Die Website muss mit einem Login vor unberechtigten Blicken geschützt werden
- Der Applikation-Server muss nach den heutigen Standards geschützt werden, eine Firewall und sichere Passwörter müssen verwendet werden.

1.5.5 Übertragbarkeit

- Verwendete Technologien müssen den Firmenstandards entsprechen, damit die Applikation später auch weiterentwickelt und verbessert werden kann
- Die Applikation sollte mit einem gängigen Browser (Firefox, Chrome, Internet Explorer) aufgerufen werden können.

2 Domainanalyse

Die Domainanalyse wurde im Projekt in zwei Teile gegliedert. Die erste Domainanalyse befasst sich mit der Darstellung der Topologie und die zweite mit den Multicast-Adressen.

2.1 Domainanalyse Topologie

Das folgende Bild zeigt das Domain-Modell der Topologie Analyse.

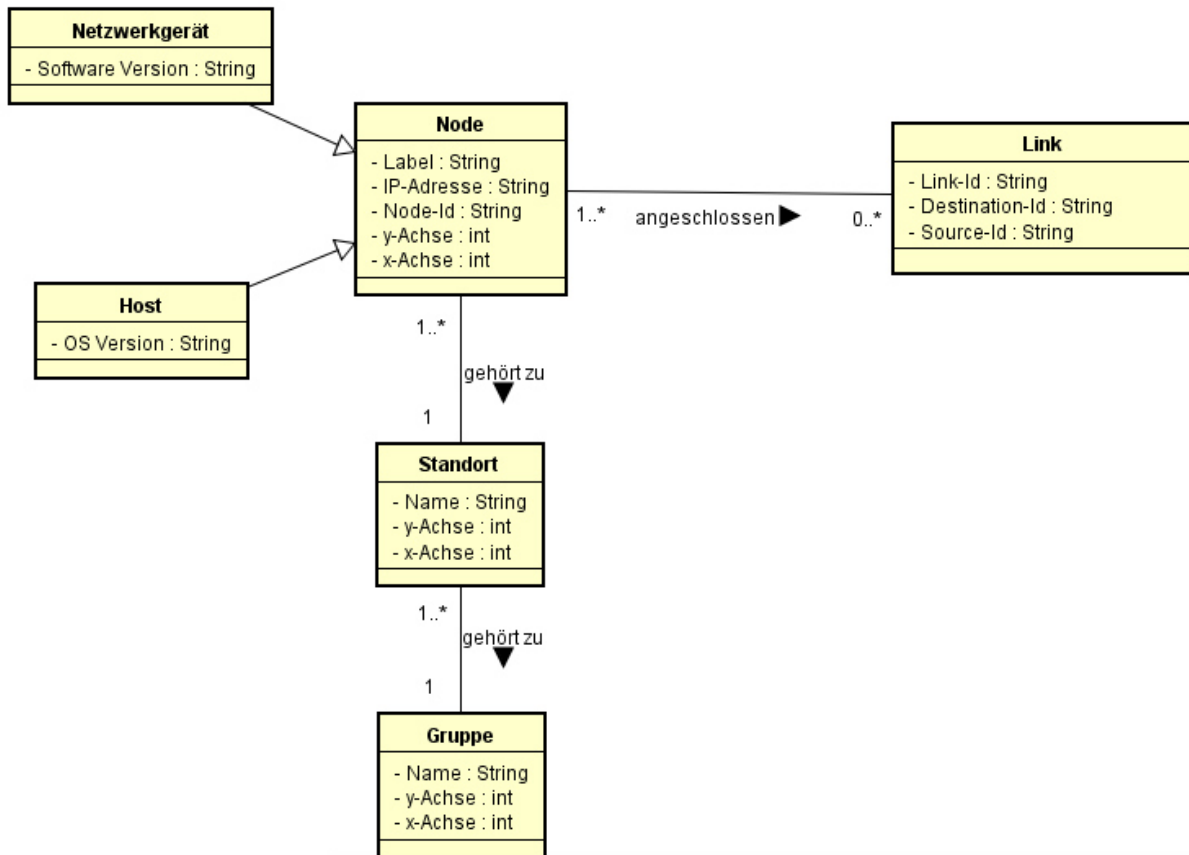


Abbildung 5: Domain-Modell Topologie

2.1.1 Link

Als Link wird die Verbindung zwischen zwei Netzwerkgeräten verstanden. Jedes Netzwerkgerät bekommt durch den APIC-EM eine eindeutige ID zugewiesen. Anhand dieser ID kann festgestellt werden, welche Netzwerkgeräte miteinander verbunden sind.

Attribut	Beschreibung
Link-Id	Eindeutige Link ID.
Destination-Id	ID vom Ziel-Netzwerkgerät.
Source-Id	ID vom Quell-Netzwerkgerät.

2.1.2 Node

Als Node ist ein Netzwerkgerät oder ein Endgerät gemeint, an dem ein oder mehrere Link(s) angeschlossen sind.

Attribut	Beschreibung
Label	Das Label ist sozusagen der Hostname des Gerätes im Netzwerk.
IP-Adresse	Management IP-Adresse.
Node-Id	ID zur eindeutigen Identifikation des Netzwerkgerätes.
y-Achse	y-Koordinate des Netzwerkgerätes auf der Topologie.
x-Achse	x-Koordinate des Netzwerkgerätes auf der Topologie.

2.1.3 Netzwerkgerät

Mit dem Netzwerkgerät ist ein bestimmter Typ eines Gerätes gemeint, wie zum Beispiel Router, Switch oder Access Point.

Attribut	Beschreibung
Software Version	Da nur Cisco Geräte als Netzwerkgeräte im gesamten Netz vorhanden sind, ist mit der Software Version die installierte IOS Version gemeint.

2.1.4 Host

Mit dem Host ist ein Endgerät im Netzwerk gemeint. Dies kann zum Beispiel ein Computer oder auch ein Handy sein.

Attribut	Beschreibung
OS Version	Mit der OS-Version ist das installierte Betriebssystem auf dem Endgerät gemeint.

2.1.5 Standort

Mehrere Nodes, die am gleichen Standort sind, werden zu einer Gruppe zusammengefasst.

Attribut	Beschreibung
Name	Name des Standortes.
y-Achse	y-Koordinate des Standortes auf der Topologie.
x-Achse	x-Koordinate des Standortes auf der Topologie.

2.1.6 Gruppe

Mehrere Standorte können zu einer Gruppe zusammengefasst werden. Diese Anforderung würde in der Anforderungsanalyse als optional deklariert.

Attribut	Beschreibung
Name	Name der Gruppe.
y-Achse	y-Koordinate der Gruppe auf der Topologie.
x-Achse	x-Koordinate der Gruppe auf der Topologie.

2.2 Domainanalyse Multicast

Das folgende Bild zeigt das Domain-Modell der Multicast Analyse.

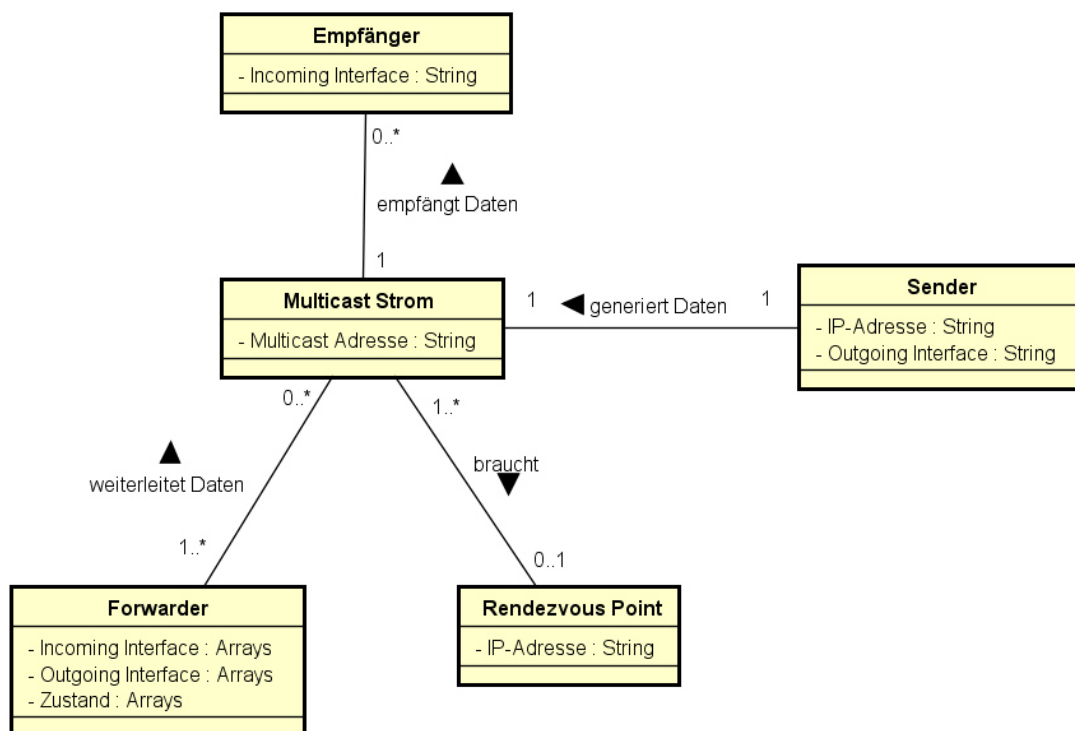


Abbildung 6: Domainmodel Multicast

2.2.1 Multicast-Strom

Der Multicast-Strom besteht aus einem Sender und mehreren Empfängern. Der Multicast-Strom wird im ganzen Netzwerk über eine eindeutige IP-Adresse (Multicast-Adresse) verteilt.

Attribut	Beschreibung
Multicast Adresse	Eindeutige IP-Adresse, die im Netzwerk bekannt ist.

2.2.2 Forwarder

Der Forwarder ist ein Netzwerkgerät, meistens ein Router oder auch ein Switch, der den Multicast-Strom vom Sender an die Empfänger weiterleitet.

Attribut	Beschreibung
Incoming Interface	Interface, bei dem der Multicast-Strom vom Sender hineinfließt.
Outgoing Interface	Interface, bei dem der Multicast-Strom zum Sender hinausfließt.
Zustand	In welchem Zustand bzw. Modus der Multicast-Strom im Netzwerk gerade ist.

2.2.3 Sender

Der Sender erzeugt den Multicast-Strom und sendet ihn ins Netzwerk.

Attribut	Beschreibung
IP-Adresse	Eindeutige IP-Adresse, damit festgestellt werden kann, woher die Daten kommen.
Outgoing Interface	Der Sender sendet den Multicast-Strom über ein bestimmtes Interface ins Netzwerk.

2.2.4 Empfänger

Mit Empfänger sind alle Teilnehmer gemeint, die den spezifischen Multicast-Strom von der Adresse empfangen wollen. Die IP-Adresse des Empfängers ist irrelevant für den Multicast-Strom und darum nicht aufgeführt.

Attribut	Beschreibung
Incoming Interface	Interface, auf dem die Sender die Daten empfangen können.

2.2.5 Rendezvous Point

Falls nicht bekannt ist, bei welcher Quelle ein Multicast-Strom bezogen werden kann, braucht es einen Rendezvous-Point. An diesem Punkt treffen sich Sender und Empfänger, und der Sender erfährt dadurch, von welcher Quelle er den Strom direkt beziehen kann.

Attribut	Beschreibung
IP-Adresse	Der Rendezvous Point hat eine eindeutige IP-Adresse, und diese muss im Netzwerk bekannt sein.

3 Architekturentscheidung

Aufgrund der Anforderungsanalyse sind wir zum Entschluss gekommen, dass für die Umsetzung der Webapplikation eine 3-Tier-Architektur am besten geeignet wäre. Es braucht in diesem Fall einen Webserver, der die Anfragen von einem Browser behandelt. Der Applikationsserver sammelt die Daten von den verschiedenen Quellen und speichert sie in der Datenbank. Der erste Gedanke war, dass die Datenbank auf demselben Server betrieben wird wie die Webapplikation und der Datenbank-Server selbst. In Zukunft ist es jedoch auch möglich, die verschiedenen Komponenten voneinander zu entkoppeln, um eine bessere Trennung zu erreichen.

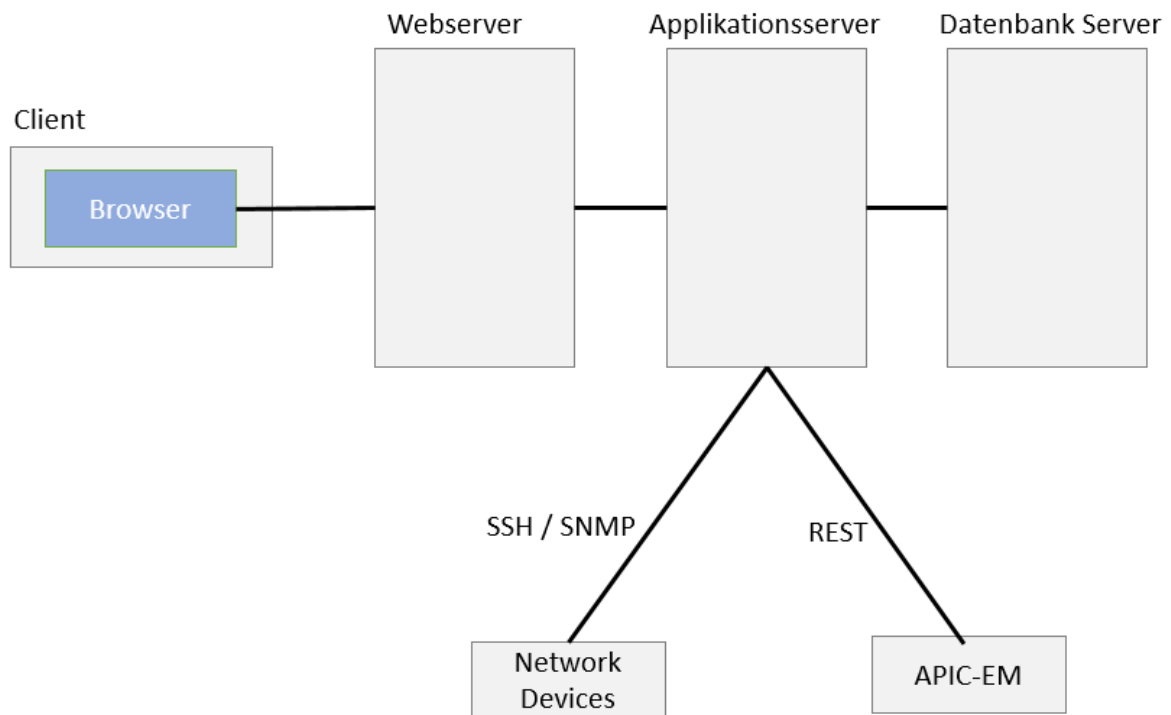


Abbildung 7: Grundlegende Architektur

Welche Komponenten nun aber zusammengefügt oder auch aufgeteilt werden sollten, musste zuerst evaluiert werden.

3.1 Variante 1: Monolith-Architektur

Eine Möglichkeit wäre gewesen, einen Monolith zu erstellen. Das heisst, der Applikations- und Webserver werden zu einem System zusammengefasst. Anfragen vom Browser wären an diesen Monolith gegangen und auch die Datensammlung von den externen Quellen hätte dieser Applikationsserver behandelt. Nur die Datenbank wäre weiterhin unabhängig von den anderen Komponenten.

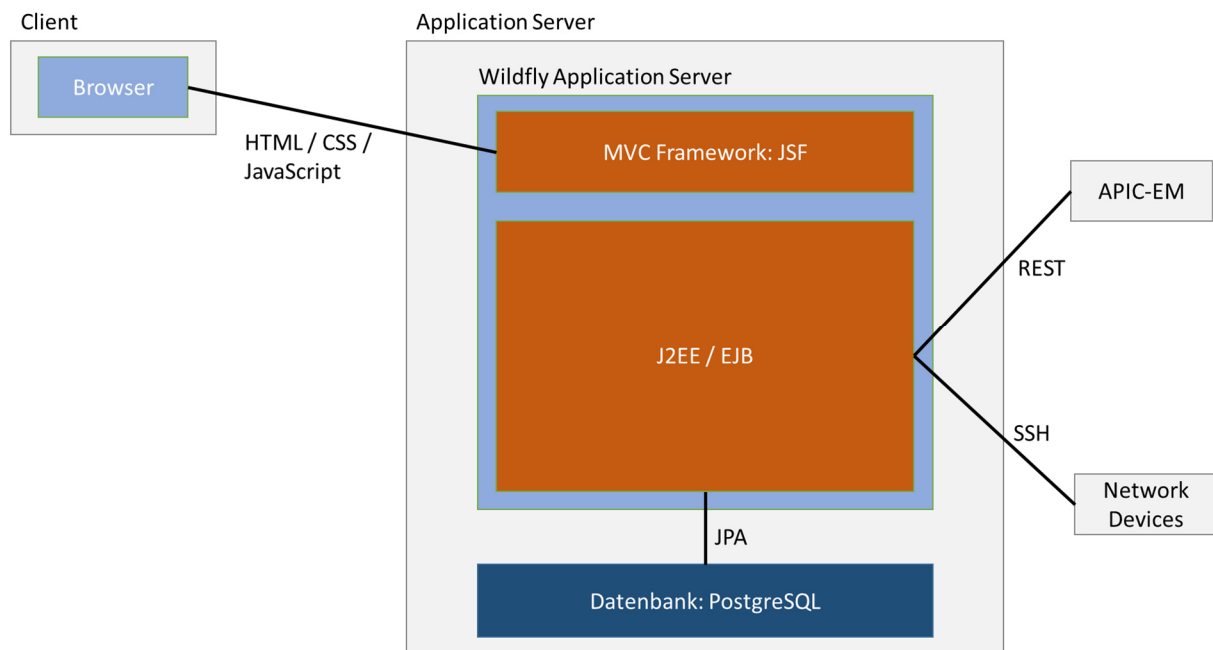


Abbildung 8: Monolith Architektur

Auf der Grafik sind mögliche Technologien abgebildet, die verwendet werden könnten. In einer späteren Evaluation wäre es aber weiterhin möglich, andere Technologien zu verwenden.

3.1.1 Vorteile

- Eine Applikation für das ganze Projekt
- Einzelne Klassen könnten mehrmals verwendet werden

3.1.2 Nachteile

- Blockiert ein Teil die Applikation, könnten die anderen Teile ebenfalls betroffen sein
- Die Skalierung einzelner Komponenten funktioniert nur bedingt
- Komplexität innerhalb des Applikationsservers ist sehr hoch

3.2 Variante 2: Aufteilung in Services

Bei dieser Variante wird der ursprüngliche angedachte Applikationsserver gespalten und in zwei voneinander unabhängige Komponenten aufgeteilt. Der Polling Service ist ein Service, der nur dafür zuständig ist, die Daten von den verschiedenen Quellen zu beziehen und zu korrelieren. Danach speichert der Polling Service die aufbereiteten Daten in einer Datenbank. Dies geschieht nach einem festen Zyklus und geschieht unabhängig vom Webserver. Der Webserver dagegen liest die Daten aus der Datenbank und bereitet sie für den User in einem ansprechenden Format auf. Durch diese Architektur können einzelne Komponenten gestoppt werden, ohne dass andere Komponenten dadurch beeinträchtigt werden.

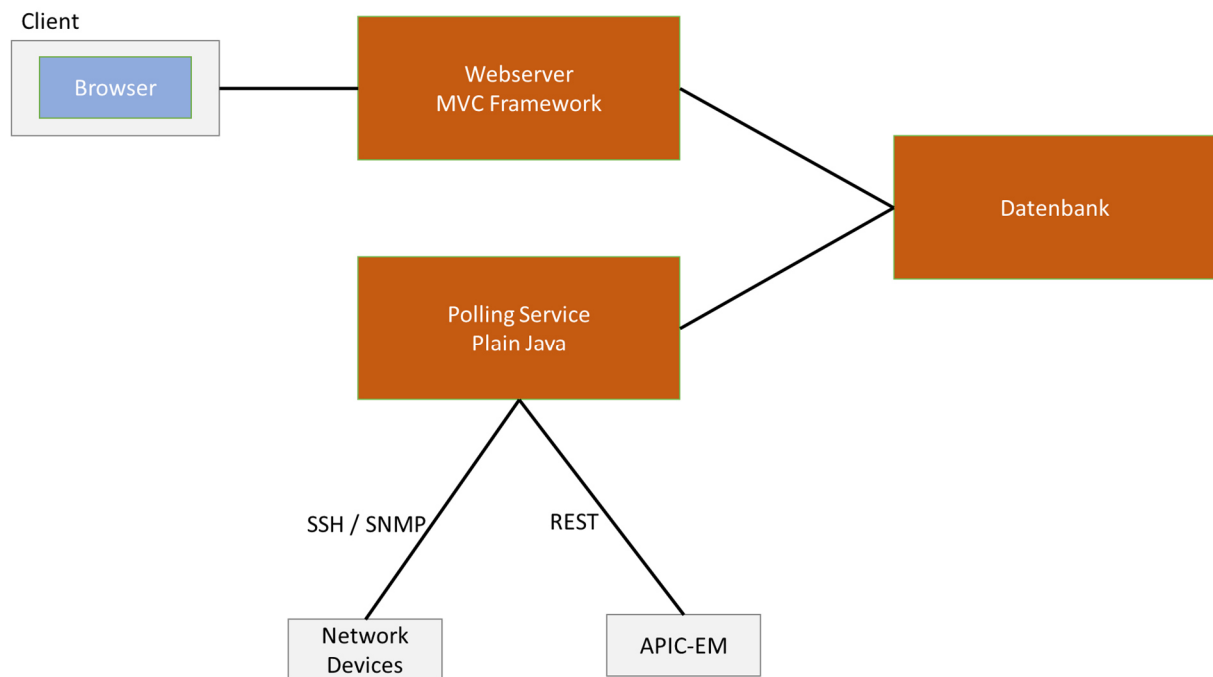


Abbildung 9: Unterteilte Services Architektur

3.2.1 Vorteile

- Skalierbarkeit der einzelnen Komponenten
- Unabhängigkeit
- Einzelne Komponenten können einfacher ausgetauscht werden

3.2.2 Nachteile

- Schnittstellen müssen genau definiert werden
- Klassen können nicht wiederverwendet werden

3.3 Entscheidung Aufteilung

Wir sind bei der Entscheidung dem wichtigen Grundsatz der Losen-Kopplung gefolgt. Bei Variante 1 sind alle Komponenten eng miteinander verbunden und die Kopplung dazwischen wäre dementsprechend sehr hoch. Einzelne Komponenten könnten nicht einfach so ausgetauscht oder erweitert werden. Bei einer späteren Weiterentwicklung würde die Komplexität zunehmen und die Wartung würde schwieriger werden. Der Vorteil mit der Wiederverwendbarkeit einzelner Klassen rückt damit in den Hintergrund.

Mit der Variante 2 wäre die Kopplung zwischen den einzelnen Komponenten minimal. Blockiert der Polling Service, könnte die Website immer noch Daten von der Datenbank beziehen und die Topologie darstellen.

Wir haben uns darum für Variante 2 entschieden, da wir der Meinung sind, eine bessere Lösung für unser Projekt gefunden zu haben.

4 GUI Analyse

Aus der Anforderungsanalyse geht hervor, dass das User Interface einfach zu bedienen sein muss. Der Zugriff auf das GUI findet lediglich von normalen Desktop Browsern statt.

Um die Webapplikation so einfach wie möglich zu halten, werden lediglich zwei Seiten zur Anzeige verwendet. Ruft man die Webapplikation auf, wird ein Login-Formular angezeigt, welches den Zugriff auf die eigentlichen Daten erstmals unterbindet. Nach einem erfolgreichen Login wird man auf die eigentliche Seite weitergeleitet, wo man sich die Netzwerktopologie mit den Multicast-Strömen anzeigen lassen kann.



The mockup shows a web interface titled "MC Viewer". Below the title is a login form with two input fields: "Username:" and "Password:". Below these fields is a "Login" button.

Abbildung 10: Mockup Login Screen

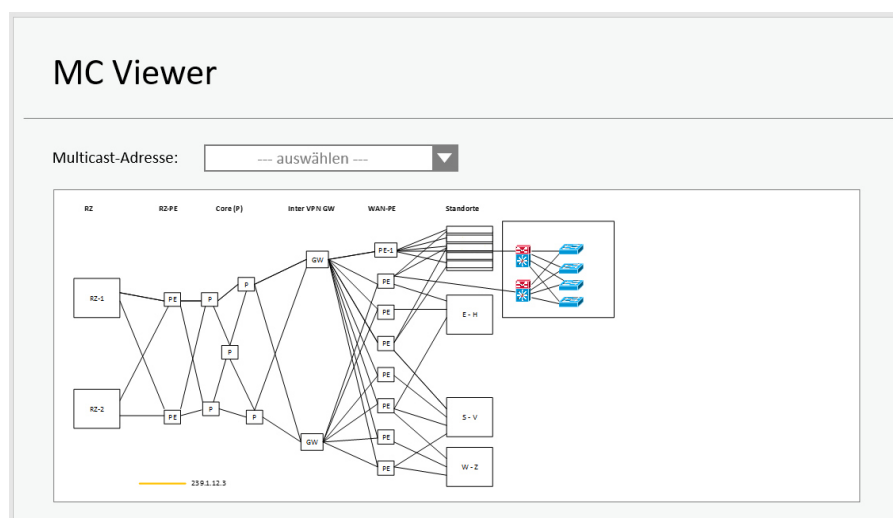


Abbildung 11: Mockup Topologieübersicht

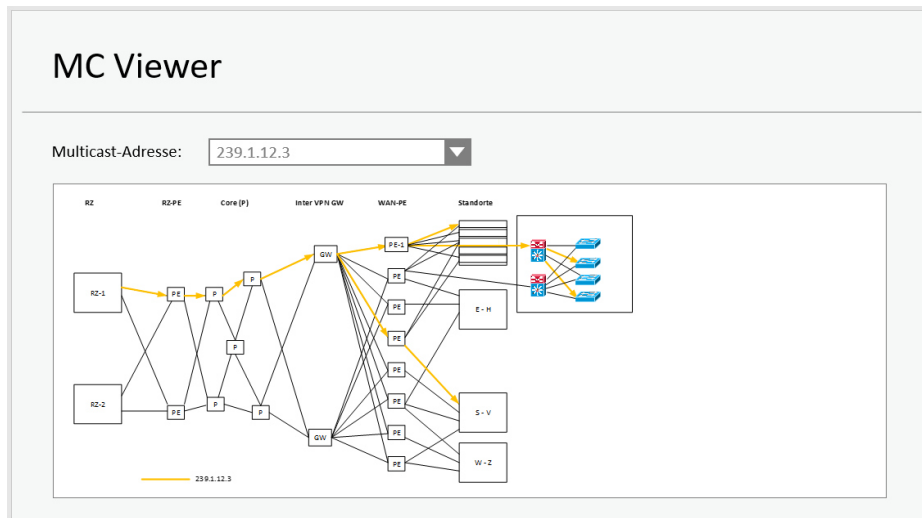


Abbildung 12: Mockup Multicast-Strom

5 Technologieevaluation

Während der Evaluationsphase des Projekts sind wir auf diverse Probleme gestossen, für die Lösungen evaluiert werden mussten.

5.1 Programmiersprache

Als Programmiersprache im Backend muss Java verwendet werden. Dies ist eine fixe Vorgabe durch die AnyWeb AG. Da an der HSR ebenfalls diese Programmiersprache primär behandelt wird, sollte dies für eine erfolgreiche Durchführung des Projekts keine Probleme darstellen.

5.2 SNMP oder SSH

5.2.1 Problem

Um die Multicast-Informationen von einem Netzwerkgerät zu beziehen, gibt es zurzeit zwei Möglichkeiten: unter anderem mittels SSH eine Verbindung zum Netzwerkgerät aufbauen und dort einen Shell-Befehl absetzen, oder eine SNMP-Abfrage an das Netzwerkgerät schicken. Die Netzwerkgeräte unterstützen standardmässig SSH und SNMPv2c.

5.2.2 Lösungsvarianten

5.2.2.1 SSH

Die Verbindung zum Netzwerkgerät per SSH ist eine gute und schnelle Möglichkeit, um Informationen aus dem Netzwerk auszulesen. Mit dem Befehl «show ip mroute» können die Multicast-Informationen sehr einfach abgerufen werden.

Auf dem folgenden Bild ist eine solche Abfrage mit Ausgabe abgebildet.

```
PE-4#show ip mroute vrf Red
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.97), 6d06h/00:03:28, RP 10.10.255.200, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    GigabitEthernet2.10, Forward/Sparse, 6d06h/00:03:28

(*, 239.1.1.96), 6d01h/stopped, RP 10.10.255.200, flags: SP
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(10.11.10.20, 239.1.1.96), 6d01h/00:01:36, flags: PA
  Incoming interface: Lspvif1, RPF nbr 10.0.254.21
  Outgoing interface list: Null
```

Abbildung 13: show ip mroute Ausgabe

Es ist unschwer zu erkennen, dass die Ausgaben meistens sehr unstrukturiert sind und darum ein Parsing zusätzlich erschwert wird. Mit dem «include»-Befehl und Regular Expressions können die Ausgaben auf der Konsole eingeschränkt werden.

Auf der nachfolgenden Abbildung ist der Befehl dargestellt, der alle Multicast-Adressen auf dem Netzwerkgerät abfragt und mit einer Regular Expression besser darstellt.

```
PE-4#show ip mroute vrf Red | include \
(*, 239.1.1.97), 6d06h/00:02:47, RP 10.10.255.200, flags: S
(*, 239.1.1.96), 6d01h/stopped, RP 10.10.255.200, flags: SP
(10.11.10.20, 239.1.1.96), 6d01h/00:02:57, flags: PA
```

Abbildung 14: show ip mroute Ausgabe mit Regular Expressions

Der grosse Vorteil einer SSH-Verbindung ist, dass sicher alle vorhandenen Informationen auf dem Netzwerkgerät ausgelesen werden können. Dies macht die SSH-Verbindung zu einer Allzweckwaffe. Zudem sind diese Daten sicher aktuell und stimmen mit der Situation im Netzwerk überein.

Vorteile	Nachteile
<ul style="list-style-type: none"> • Alle Informationen verfügbar • Einschränkungsmöglichkeiten mit Regular Expressions • Informationen werden live abgefragt und stimmen garantiert • Auf allen Netzwerkgeräten möglich 	<ul style="list-style-type: none"> • Verbindungsbehandlung mühsam • Parsing der Ausgabe schwierig • Unterschiedliche Kommandos bei unterschiedlichen Netzwerkgeräten

5.2.2.2 SNMP

Da durch den APIC-EM sowieso schon SNMP auf den Netzwerkgeräten konfiguriert ist, wäre es eine Möglichkeit, die Multicast-Informationen per SNMP abzufragen. Es gibt mehrere MIBs, die verwendet werden können, um diese Informationen auszulesen. Damit aber alle IOS-Versionen unterstützt werden, muss zwangsläufig die «CISCO-IPMROUTE-MIB.my» verwendet werden.

Nach diversen Abfragetests wurde aber festgestellt, dass diese SNMP-Abfragen nicht die gewünschten Resultate liefern. Die verschiedenen Multicast-Gruppenadressen werden direkt hinter die OID angehängt. Um nun die Multicast-Gruppe herauszulesen, müsste wie bei SSH ebenfalls geparst werden. Auch kann nicht ausgelesen werden, auf welchem Interface die Multicast-Ströme empfangen und danach weitergesendet werden. Die Abfrage gibt lediglich als Next Hop die MAC-Adresse zurück. Anhand der MAC-Adresse müsste darum zusätzlich noch herausgefunden werden, welches Interface verwendet wurde.

Das aber wohl grösste Problem ist, dass die MIB nicht mit VRF-Instanzen umgehen kann. Will man das Multicast-Routing einer bestimmten VRF-Gruppe sehen, kann zum Beispiel bei SSH die Gruppe angegeben werden. Da bei SNMP keine Argumente mitgegeben werden können, werden einfach nur die globalen Multicast-Informationen angezeigt. Diese Informationen sind aber für die Applikation wenig hilfreich.

Der nachfolgende Vergleich sollte dies verdeutlichen. Der Befehl wurde auf dem gleichen Netzwerkgerät ausgeführt.

Ausgabe: show ip mroute

```
PE-4#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode
```

Abbildung 15: show ip mroute ohne VRF

Ausgabe: show ip mroute vrf Red

```
PE-4#show ip mroute vrf Red
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.97), 6d06h/00:03:09, RP 10.10.255.200, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    GigabitEthernet2.10, Forward/Sparse, 6d06h/00:03:09

(*, 239.1.1.96), 6d01h/stopped, RP 10.10.255.200, flags: SP
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null
```

Abbildung 16: show ip mroute mit VRF

Das Anzeigen der Multicast-Informationen auf VRF-Ebene ist aber eine wichtige Anforderung. In grossen Enterprise-Netzwerken wird oft mit VRF-Instanzen gearbeitet.

Vorteile	Nachteile
<ul style="list-style-type: none"> • Einfache Verbindungsbehandlung (GET Request) • Einheitliches Format der Ausgabe 	<ul style="list-style-type: none"> • Multicast-Informationen nur beschränkt verfügbar • Next Hop-Adresse wird als Mac-Adresse angegeben • VRF-Abfrage nicht möglich

5.2.2.3 Andere Möglichkeiten

Da die Daten möglichst aktuell und dynamisch erfasst werden müssen, würde ein weiteres Programm wahrscheinlich ebenfalls die anderen beiden Möglichkeiten als Informationsquelle verwenden. Ein solches Tool würde dadurch nur eine zusätzliche Abstraktionsebene in unser Projekt legen, von dem wir danach abhängig sind. Es macht darum wenig Sinn, ein externes Programm zu evaluieren und zu verwenden, um die Multicast-Informationen herauszulesen.

5.2.3 Entscheidung

Da SNMP nur sehr wenige Informationen zu den Multicast-Strömen zurückgibt und diese nicht wirklich brauchbar sind, haben wir uns für die SSH-Möglichkeit entschieden. Obwohl das Parsing schwieriger wird, ist diese Lösung einfacher und besser erweiterbar. Braucht es später noch mehr Informationen zum Multicast-Strom, können diese sicher besser über SSH als per SNMP ausgelesen werden. Multicast-Abfragen per SNMP sind im Unternehmensnetzwerk zudem weniger gut geeignet, da eine Abfrage zu den VRFs nicht möglich ist.

5.3 Topologie-Informationen beziehen

5.3.1 Problem

Damit eine Topologie mit allen Netzwerkgeräten und Verbindungen erstellt werden kann, müssen diese Informationen aus dem Netzwerk herausgelesen werden. Da das vorgegebene Netzwerk nur aus Cisco Geräten besteht, dürfen Cisco-proprietäre Produkte verwendet werden, um die Topologie-Informationen zu erhalten. Es muss die Frage geklärt werden, wie alle Geräte im Netzwerk am einfachsten erkannt werden können.

5.3.2 Lösungsvarianten

5.3.2.1 CDP

Auf dem Netzwerkgerät können per SSH-Konsole und dem Befehl «show cdp neighbors» alle Netzwerkgeräte angezeigt werden, die mit dem Netzwerkgerät verbunden sind. Über einen Einstiegspunkt könnten so alle vorhandenen Cisco Geräte im Netzwerk gefunden werden. Dieses Traversieren über das Netzwerk bräuchte aber einen komplexen Algorithmus. Zudem müsste auf jedem Netzwerkgerät cdp aktiviert sein, was nicht immer gewünscht ist.

Vorteile	Nachteile
<ul style="list-style-type: none"> • Einfach Feststellung, wo welches Netzwerkgerät angeschlossen ist 	<ul style="list-style-type: none"> • Komplexer Algorithmus • Verbindungsbehandlung mit SSH • CDP muss aktiviert sein

5.3.2.2

5.3.2.3 APIC-EM

Eine einfachere Variante bietet hier der APIC-EM. Der APIC-EM ist ein Controller für Enterprise-Netzwerke und kann zum Beispiel die Netzwerktopologie-Informationen auslesen und sinnvoll auf einer Weboberfläche darstellen. Zusätzlich zu dieser Web-Oberfläche hat der APIC-EM eine REST-Schnittstelle. Über diese Schnittstelle können auch die Informationen über die Netzwerktopologie abgefragt werden. Als Übertragungsformat wird JSON verwendet. Dabei werden die Links und die Nodes jeweils als Array zurückgegeben.

Auf dieser Seite sieht man die Antwort des HTTP-GET Request. Zuerst werden alle aktiven Nodes (Netzwerkgeräte) aufgelistet, danach kommen alle aktiven Links (Verbindungen zwischen den Netzwerkgeräten).

```
{
  "response": {
    "nodes": [
      {
        "deviceType": "Cisco Cloud Services Router 1000V",
        "label": "PE-3.sa-apic.ch",
        "ip": "10.0.254.23",
        "softwareVersion": "15.5(3)S2",
        "nodeType": "device",
        "family": "Routers",
        "platformId": "CSR1000V",
        "tags": [],
        "role": "BORDER ROUTER",
        "roleSource": "AUTO",
        "customParam": {},
        "id": "1856b84e-de61-4a12-b1db-c6e5027e431c"
      },
      {
        "deviceType": "Cisco Cloud Services Router 1000V",
        "label": "Pl.sa-apic.ch",
        "ip": "10.0.254.31",
        "softwareVersion": "15.5(3)S2",
        "nodeType": "device",
        "family": "Routers",
        "platformId": "CSR1000V",
        "tags": [],
        "role": "BORDER ROUTER",
        "roleSource": "AUTO",
        "customParam": {},
        "id": "39fd6d1e-0a59-4f13-8f89-4b54d9533fdb"
      }
    ],
    "links": [
      {
        "source": "39fd6d1e-0a59-4f13-8f89-4b54d9533fdb",
        "startPortID": "6e0f2492-1f80-47b9-b885-d12f720f3b80",
        "startPortName": "GigabitEthernet3",
        "startPortIpv4Address": "10.0.16.31",
        "startPortIpv4Mask": "255.255.255.0",
        "startPortSpeed": "1000000",
        "target": "1856b84e-de61-4a12-b1db-c6e5027e431c",
        "endPortID": "f40e018d-5331-49ff-aefd-77f03db595dd",
        "endPortName": "GigabitEthernet3",
        "endPortIpv4Address": "10.0.16.23",
        "endPortIpv4Mask": "255.255.255.0",
        "endPortSpeed": "1000000",
        "linkStatus": "up",
        "id": "97114"
      }
    ]
  },
  "version": "1.0"
}
```


Vorteile	Nachteile
<ul style="list-style-type: none"> • Einfache Abfrage (REST-Schnittstelle) • Rückgabe Format ist JSON • Änderungen im Netzwerk werden vom APIC-EM behandelt 	<ul style="list-style-type: none"> • Abhängigkeit von einer Schnittstelle • Allenfalls fehlen wichtige Informationen in der Rückgabe

5.3.3 Entscheidung

Da in den Anforderungen definiert wurde, dass ein APIC-EM im Netzwerk vorhanden sein muss, ist dies die bevorzugte Variante im Projekt. Zudem ist diese Variante einfacher und führt zu weniger Fehlern.

5.4 MVC Framework

5.4.1 Problem

Um einem Benutzer eine Weboberfläche zur Betrachtung der Multicast-Informationen anbieten zu können, wird ein Web Framework benötigt. Es gibt heutzutage diverse solcher Frameworks. Für die Evaluation wurden zwei Frameworks beigezogen.

5.4.2 Lösungsvarianten

5.4.2.1 JSF

Java Server Faces ist ein möglicher Ansatz zur Lösung des Problems. Es gehört zu den Webtechnologien der Java Enterprise Edition (Java EE) und basiert auf Servlets und JSP. JSF wurde vom Industriepartner vorgeschlagen, da das Know-how zu dieser Technologie bereits in der Industrie vorhanden ist. JSF wurde ausserdem in der Schule im Modul «Internettechnologien» behandelt.

Während der Evaluation dieses Frameworks trat jedoch ein grösseres Problem auf, welches sich nicht ganz einfach lösen lässt. Mit JSF ist es nicht ohne weiteres möglich, Daten aus der Datenbank in einer JavaScript Library für die Visualisierung zur Verfügung zu stellen. Eine mögliche Lösung wäre es, eine REST-Schnittstelle in Form von JAX-RS in die Applikation zu integrieren. Diese Integration macht jedoch die gesamte Applikation sehr komplex.

Vorteile	Nachteile
<ul style="list-style-type: none"> • Know-how beim Industriepartner vorhanden • Bewährte Technologie 	<ul style="list-style-type: none"> • Keine integrierte REST-Schnittstelle • Benötigt zusätzlichen Webserver

5.4.2.2 Play Framework

Das Play Framework [2] ist ein aktuelles MVC Framework, welches bereits von sich aus eine integrierte REST-Schnittstelle besitzt. Es ist, was die Komplexität betrifft, relativ schlank gehalten und hat bereits einen integrierten Webserver in Form von JBoss Netty. Dadurch wird ein zusätzlicher Webserver überflüssig.

In der Entwicklung besitzt es den Vorteil, dass wenn etwas am Code verändert wird, nicht die gesamte Applikation neu deployed werden muss. Nach einem Refresh im Browser sind die Änderungen sofort ersichtlich. Tritt eine Exception oder ein sonstiger Fehler während der Entwicklung auf, so wird eine entsprechende Meldung im Frontend ausgegeben. Dies ist um einiges übersichtlicher, als wenn man die Meldung anhand des Logfiles eines Webserver analysieren müsste.

Vorteile	Nachteile
<ul style="list-style-type: none">• Integrierte REST-Schnittstelle• Integrierter Webserver• Integriertes Testframework• Einfachere Programmierung (saubere Anzeige von Fehlern, einfaches Deployment)	<ul style="list-style-type: none">• Kein Know-how vorhanden• Kompilieren dauert lange, da diverse Abhängigkeiten bestehen

5.4.3 Entscheidung

Die Entscheidung fiel letztlich auf das MVC Framework Play, u.a. deshalb, da es bereits eine integrierte REST-Schnittstelle besitzt und so die zu visualisierenden Informationen aus der Datenbank problemlos ans Frontend übertragen werden können. Ausserdem wird Play auch in vielen Webprojekten von Studenten an der HSR eingesetzt. Bei etwaigen Problemen könnte somit ein Informationsaustausch mit anderen Studenten stattfinden.

Dank der übersichtlichen Darstellung von Exceptions und Fehlern ist die Programmierung und Fehlerbehebung auch etwas angenehmer.

5.5 Netzwerk-Topologie

5.5.1 Problem

Um die Netzwerktopologie bzw. den Multicast-Strom für den Benutzer visuell darstellen zu können, wird im Frontend eine Technologie benötigt, die dies realisiert. Heutzutage gibt es im Internet diverse OpenSource JavaScript Libraries, welche das Zeichnen von Graphen auf einer Webseite ermöglichen. Für die Visualisierung von Netzwerktopologien sind jedoch nicht alle gleich gut geeignet.

Folgende Anforderungen muss die JavaScript Library erfüllen, damit sie für die Visualisierung von Multicast-Strömen geeignet ist:

Nr	Anforderung	Grund
A01	Doppelte Links anzeigen.	Redundante Links im Netzwerk.
A02	Links müssen hervorgehoben werden können (Bsp. durch Farbe).	Unterscheidung Topologie <-> Multicast-Strom.
A03	Pfeile müssen gezeichnet werden können.	Datenfluss von Multicast.
A04	Verschiebung der Netzwerkgeräte.	Ansicht umändern, um Topologie besser zu verstehen. Speichern der veränderten Ansicht ist nicht Teil der Anforderungen.
A05	Verwendung von sinnvollen Symbolen (Switch, Router).	Netzwerkgeräte müssen unterschieden werden können.
A06	Gruppieren von verschiedenen Geräten	Übersichtlichkeit bei vielen Geräten bewahren.

5.5.2 Lösungsvarianten

Für die Evaluation des UI Frameworks wurden verschiedene JavaScript Libraries für Visualisierungen in Betracht gezogen und auf die oben genannten Anforderungen geprüft.

5.5.2.1 NeXt UI Framework

Das NeXt UI Framework ist eine von Cisco vertriebene OpenSource Library, welche primär dazu verwendet wird, Netzwerktopologien darzustellen. Sie bietet bereits eine Reihe nützlicher Funktionen der Netzwerkvisualisierung an. So ist es zum Beispiel möglich, Pfade in der Topologie einzuzichnen und einzelne Komponenten in Gruppen zusammenzufassen. Dies deckt die gestellten Anforderungen bestens ab.

NeXt UI wird ausserdem von Cisco selbst im Verbund mit APIC-EM verwendet.

5.5.2.2 vis.js

Die vis.js Library ist eine einfach gehaltene OpenSource Library, um grosse Datenmengen zu visualisieren bzw. zu bearbeiten. Sie besteht aus verschiedenen Komponenten, wobei sich eine davon auf die Visualisierung von Netzwerken im Allgemeinen spezialisiert.

Die Library ist ausserdem gut dokumentiert und wird durch die Community laufend erweitert.

5.5.2.3 D3 JS (Data-Driven Document)

Die D3 JS Library ist eine sehr weit verbreitete, ebenfalls gratis erhältliche Library. Sie bietet für allerlei Daten eine Visualisierungsmöglichkeit und ist deshalb sehr beliebt. Durch die vielen Möglichkeiten der Library steigt jedoch deren Komplexität.

Im Internet finden sich diverse Tutorials und unzählige Beispiele für die Verwendung von D3 JS, was bei der Entwicklung enorm von Vorteil sein kann.

5.5.3 Entscheidung

Anforderung	NeXt UI	vis.js	D3 JS
A01	strukturiert und übersichtlich	möglich aber unstrukturiert	möglich aber unstrukturiert
A02	möglich	möglich	möglich
A03	möglich	möglich	möglich
A04	möglich	möglich	möglich
A05	möglich aber nur vordefinierte	unbrauchbar	möglich
A06	möglich	eingeschränkt möglich	möglich

Alle JavaScript Libraries entsprechen den gestellten Anforderungen mit Ausnahme von vis.js. Aufgrund der Analyse ist es nicht möglich, brauchbare Netzwerk Icons (Router, Switch) zu verwenden. Es werden lediglich vordefinierte Icons von Drittanbietern unterstützt. Unter dieses befindet sich jedoch kein brauchbares Symbol. Das NeXt UI bietet von sich aus bereits diverse Netzwerksymbole an, und bei D3 JS können sogar eigene Symbole verwendet werden.

Ansonsten sind die Libraries betreffend Anforderungen in etwa gleich. Einen grossen Unterschied findet man in der Darstellung mehrerer Links. Während beim NeXt UI doppelte Links als Geraden und parallel dargestellt werden, ist bei den anderen Libraries die Darstellung immer gebogen. Dies wirkt zum Teil irritierend in Bezug auf Computernetzwerke.

Für die Visualisierung von Netzwerktopologien und Pfaden eignet sich das NeXt UI am besten. Dies vor allem aus dem Grund, da es speziell für solche entwickelt worden ist. Ein Pfad kann auf einer bestehenden Topologie eingezeichnet werden, ohne dass dabei etwas verändert wird. Bei D3 JS müssten die Links der Topologie einzeln verändert werden. Ausserdem wird es von Cisco selbst verwendet, um Netzwerktopologien darzustellen.

Aus diesen Gründen wird das Projekt mit NeXt UI durchgeführt.

5.6 Datenbank

5.6.1 Problem

Um die Informationen / Daten aus dem Netzwerk zu persistieren, muss im Hintergrund eine Datenbank vorhanden sein. Idealerweise sollte es eine relationale Datenbank sein, um das Domainmodell sauber abbilden zu können. Ansonsten gibt es keine direkten Anforderungen an die Datenbank.

5.6.2 Entscheidung

Es gibt diverse relationale Datenbanken, welche gratis verwendet werden können. Zwei weitverbreitete heutzutage sind PostgreSQL und MySQL.

Im Projekt wird nun PostgreSQL verwendet. Dies aus dem einfachen Grund, da es auch an der HSR in den Datenbank-Modulen Verwendung findet und das Know-how deshalb bereits vorhanden ist. Sollte es aus irgendwelchen Gründen (z.B. Performance-Probleme) trotzdem nicht möglich sein, PostgreSQL zu verwenden, so kann die Datenbank problemlos durch eine andere ersetzt werden.

5.7 Zugriff auf die Datenbank

5.7.1 JDBC oder OR-Mapper

Der Zugriff auf die Datenbank des Polling Service könnte über mehrere Varianten realisiert werden. Eine Variante wäre ein OR-Mapper, der die Objekte direkt persistiert. Da die Erfahrung mit OR Mapper aber nur sehr gering ist und ein OR Mapper doch sehr viele Eigenheiten mit sich bringt, ist ein Einsatz mit sehr viel Risiko verbunden. Zudem ist die Kommunikation zwischen Objekten und Datenbank in diesem Projekt nicht sehr eng miteinander verzahnt. Es ist somit problemlos möglich, die SQL Statements von Hand zu schreiben. Da aber gerade diese SQL Statements im Code eher mühsam zu schreiben sind, wird JOOQ als Zwischenschicht verwendet.

Der Datenbankzugriff der Webapplikation wird über JDBC gelöst, da das Play Framework JOOQ nicht korrekt unterstützt.

5.7.1.1 JOOQ

JOOQ ist eine Mischung aus der OR Mapper-Welt und der normalen JDBC-Welt. Der Abstraktionslevel von JOOQ ist somit etwa in der Mitte der beiden Welten. Mit JOOQ können viel einfacher Datenbank- Abfragen erstellt und die Objekte persistiert werden; deshalb bedeutet JOOQ auch «Java Object Oriented Querying. JOOQ ist daher sehr eng mit Java verbunden und bietet somit gute Typensicherheit. Mit JOOQ kann zudem die Datenbank relativ einfach ausgetauscht werden, ohne dass grosse Änderungen im Code gemacht werden müssen. [3]

6 Implementation

6.1 Polling Service

Der Polling Service ist das Backend der Applikation. Er sammelt die Topologie- und Multicast-Daten aus dem Netzwerk und persistiert sie in einer Datenbank.

6.1.1 Schichtenmodell

Der Polling Service ist in drei Schichten aufgebaut. Der Zugriff findet immer nur von oben nach unten statt, damit keine zyklischen Abhängigkeiten entstehen können.

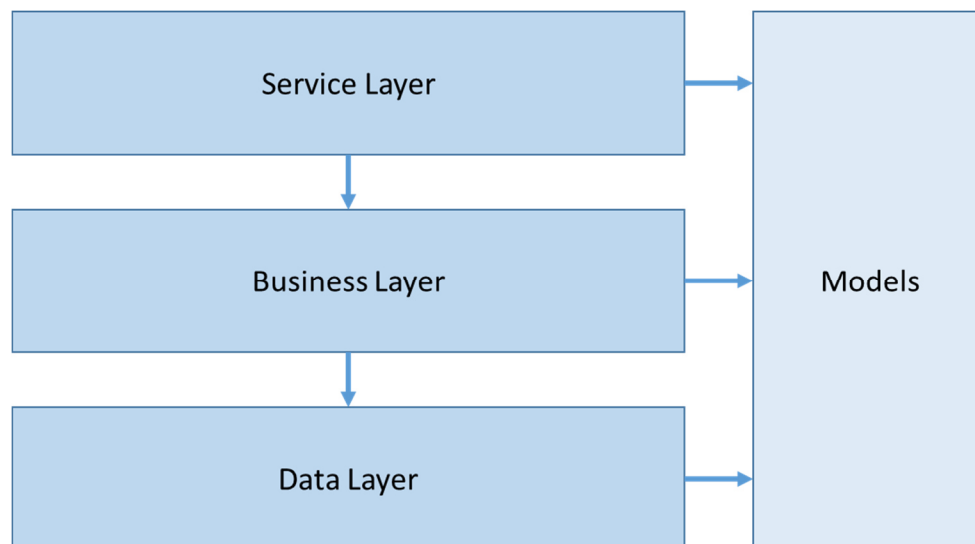


Abbildung 17: Schichtenmodell Polling Service

6.1.2 Service Layer

Im Service Layer sind alle Klassen gespeichert, die nacheinander die Klassen im Business Layer aufrufen. Es gibt drei verschiedene Service Layer-Klassen, die unterschiedlichen Funktionen aufrufen.

Klasse	Funktion
TopologyService	Startet die Funktionen im Package «topology» und ist dafür zuständig, dass die initiale Topologie aufgebaut wird. Zuerst werden alle Standorte generiert und danach die Netzwerkgeräte den Standorten zugewiesen.
MulticastService	Startet die Funktionen im Package «multicast». Bevor aber die Multicast-Abfragen über das Netzwerk laufen können, müssen zuerst alle VRF-Instanzen gesammelt und der Default MDT generiert werden.
CleaningUpService	Startet die Funktionen im Package «cleaningup». Mit diesem Service werden alle alten Netzwerkgeräte gelöscht, die für eine gewisse Zeit nicht mehr erreichbar waren. Ebenfalls werden nicht mehr verwendete VRF-Instanzen regelmässig gelöscht.

6.1.3 Business Layer

Im Business Layer befindet sich die gesamte Intelligenz der Applikation. Unter anderem werden hier die REST- und SSH-Abfragen getätigt, geparkt und ausgewertet.

6.1.3.1 Location Handling

Die Klasse Location Handling ist für die Initialisierung der verschiedenen Standorte zuständig. Hier wird eine CSV-Datei mit den Inhalten zu Hostname, Standort, etc. gelesen und geparkt. Die Standorte werden dabei ausgelesen und in sortierter Reihenfolge abgespeichert. Zusätzlich wird jedem Standort dynamisch eine X- und eine Y-Koordinate zugewiesen.

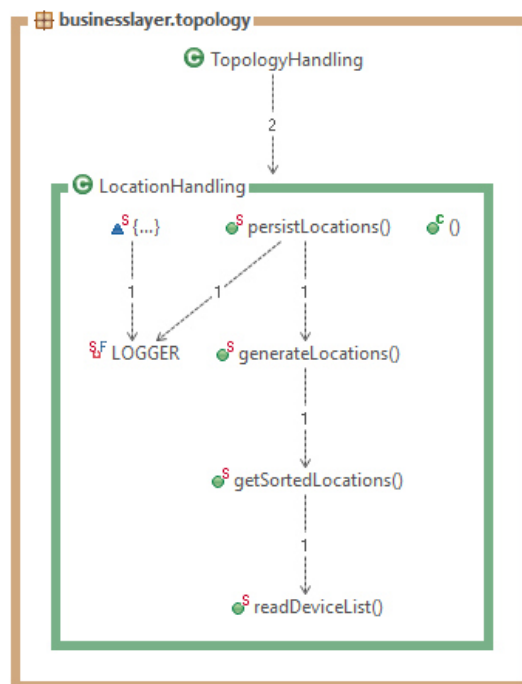


Abbildung 18: Klasse LocationHandling mit allen enthaltenen Funktionen

Damit das CSV korrekt geparkt wird, muss es folgende Struktur besitzen:

```

Hostname;Location;IsDeviceInDatacenter;x;y
CE-1;RZ01;true;;
CE-2;RZ02;true;;
CE-3;Stao02;;;
CE-4;Stao03;;;
IVPN-GW;;;-200;-220
  
```

Abbildung 19: CSV für Standortzuweisungen

Spalte	Bemerkung
Hostname	Hostname ohne FQDN des Gerätes.
Location	Name des Standortes.
IsDeviceInDatacenter	Gibt an, ob ein Gerät in einem Rechenzentrum steht oder nicht.
x	X-Koordinate für die Anzeige im NeXt UI. Wird verwendet, um ein Netzwerkgerät statisch in der Topologie platzieren zu können. Ist keine Koordinate gesetzt, wird diese aufgrund des Standortes dynamisch generiert.
y	Y-Koordinate für die Anzeige im NeXt UI. Wird verwendet, um ein Netzwerkgerät statisch in der Topologie platzieren zu können. Ist keine Koordinate gesetzt, wird diese aufgrund des Standortes dynamisch generiert.

6.1.3.2 Topology Handling

Die Klasse Topology Handling ist für den Aufbau der Topologie verantwortlich. Sie macht eine REST-Abfrage zum APIC-EM und erhält somit Informationen zu den verfügbaren Geräten und deren Links. Sie ist ebenfalls dafür zuständig, dass UI-spezifische Daten, wie Koordinaten, korrekt initialisiert werden. Informationen zu Koordinaten erhält das Topology Handling unter anderem über die bereits für die Generierung der Standorte verwendete CSV-Datei.

Alle Informationen werden danach in die Datenbank gespeichert, sodass der Multicast Service später darauf zugreifen kann.

6.1.3.2.1 generateDynamicCoordinates

Diese Methode sorgt dafür, dass jedes Netzwerkgerät, welches keine fixen Koordinaten in der CSV-Datei definiert hat, dynamische Koordinaten erhält. Damit für ein Gerät die Koordinaten generiert werden können, muss bekannt sein, an welchem Standort sich das Gerät befindet. Aufgrund der Koordinaten des Standortes können dann die Koordinaten für die Geräte innerhalb dieses Standortes generiert werden.

Der Algorithmus wurde so implementiert, dass im Falle von mehreren Netzwerkgeräten, diese in vier Spalten aufgeteilt werden. In der ersten Spalte befinden sich alle Layer 3-Geräte. In den Spalten zwei bis vier folgen dann die Layer 2-Geräte. Eine Spalte mit Switches muss mindestens 30 Switches enthalten, bis die nächste Spalte generiert wird.

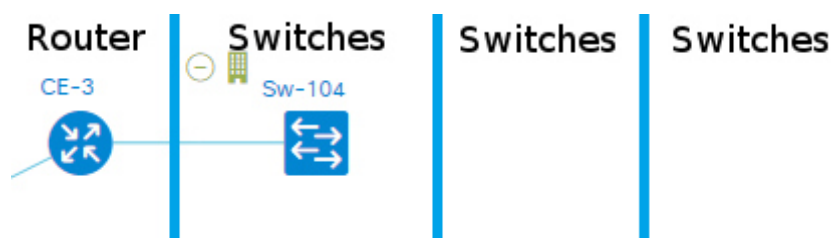


Abbildung 20: Geräteverteilung innerhalb Standort

6.1.3.3 VRF Handling

Bevor die Multicast-Informationen abgefragt werden können, muss bekannt sein, auf welchem Gerät welche VRF-Instanz aktiv ist. Dies wird über die VRF Handling-Klasse erledigt. Da die VRF-Informationen noch nicht wirklich sinnvoll aus dem APIC-EM ausgelesen werden können, müssen sie per Kommandozeile über SSH abgefragt werden.

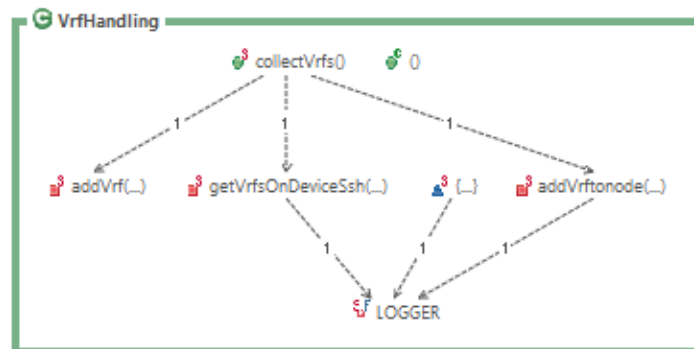


Abbildung 21: Klasse VrfHandling mit allen enthaltenen Funktionen

6.1.3.3.1 collectVrfs

Diese Funktion fragt alle PE und CE Router von der Datenbank ab und ruft danach auf jedem Router die Funktion «getVrfsOnDeviceSsh» auf. Die P Router werden nicht abgefragt, da sie im Core-Bereich des MPLS Netzwerks arbeiten und keine VRF-Instanzen konfiguriert haben.

6.1.3.3.2 getVrfsOnDeviceSsh

Die Abfrage zu den vorhandenen VRF-Instanzen auf dem Router ist unterschiedlich zwischen PE und CE Router. Grundsätzlich würde auf allen Routern der Befehl «show vrf» funktionieren. Zusätzlich wird zur Abfrage noch das «include» Statement hinzugefügt, dadurch wird die Ausgabe schon ein erstes Mal gefiltert.

```

CE-1#show vrf | include :
Blue          65100:2011      ipv4      Gi2.20
Red           65100:1011     ipv4      Gi2.10
CE-1#
    
```

Abbildung 22: Ausgabe Befehl "show vrf | include :"

Da der PE Router im Gegensatz zum CE Router auch mit dem MPLS-Netzwerk verbunden ist, kann hier der Befehl «show vrf id» verwendet werden. Dieser Befehl ist auch nötig, da in der Ausgabe die VPN Id angegeben wird. Diese wird später bei der Abfrage vom Data oder Default MDT benötigt und darum ebenfalls in die Datenbank gespeichert.

```

PE-1#show vrf id
VPN Id      Name      RD
65000:10    Red       65000:1021
65000:20    Blue      65000:2021
65000:30    Green     65000:3021
PE-1#
    
```

Abbildung 23: Ausgabe Befehl show vrf id

6.1.3.4 Multicast Handling

Die Klasse «MulticastHandling» sucht im Netzwerk nach verfügbaren Multicast-Strömen. Dazu wird mit jedem vorhandenen Netzwerkgerät ein Verbindungsaufbau versucht. Welche Netzwerkgeräte im Netzwerk aktiv sind, wird aus der Datenbank ausgelesen.

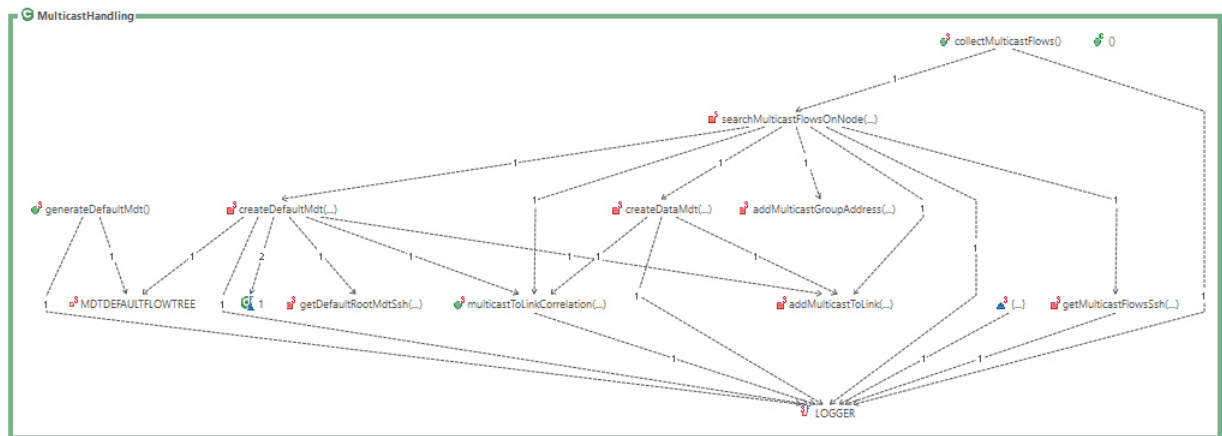


Abbildung 24: Klasse Multicast Handling mit allen enthaltenen Funktionen

6.1.3.4.1 collectMulticastFlows

Diese Methode ist die initiale Funktion der Multicast Handling-Klasse. Zuerst werden alle PE und CE Router von der Datenbank abgefragt, um danach von jedem Router die Multicast-Ströme sammeln zu können. Ist ein Router nicht erreichbar, springt die Applikation automatisch zum nächsten Router.

6.1.3.4.2 searchMulticastFlowsOnNode

Zuerst werden in dieser Funktion die Multicast-Ströme auf Ebene Layer 3 über SSH abgefragt und geparkt. Mit Layer 3 sind IP-geroutete Netzwerke gemeint. Der Name wurde gewählt, damit bei einem späteren Ausbau die Namensgebung einheitlicher wirkt.

Um Multicast-Ströme in einem normalen Layer 3-Netzwerk zu finden, wird nachstehende Abfolge durchlaufen.

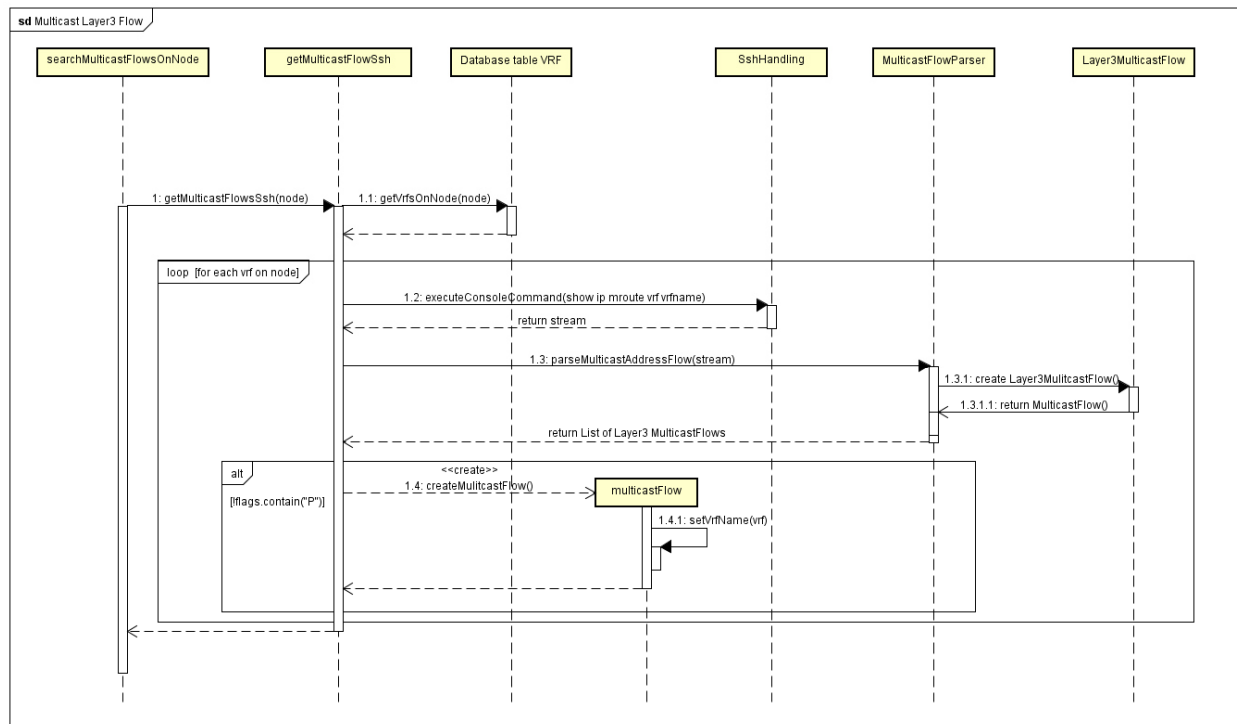


Abbildung 25: Sequenzdiagramm - Multicast-Ströme im Layer 3 Netzwerk sammeln

Grundsätzlich wird jeder PE oder CE Router angefragt, ob auf dem Gerät ein Multicast-Strom läuft. Da die Multicast-Ströme pro VRF-Instanzen unterschiedlich sind, muss pro VRF-Instanz der Befehl ausgeführt werden.

Die Multicast-Adressen werden nach dem Abfragen und Parsen in der Datenbank persistiert. Damit ein Multicast-Strom nun einem Link zugordnet werden kann, wird die Funktion «multicastToLinkCorrelation» aufgerufen.

6.1.3.4.3 MPLS-Netzwerk

Ist ein Teil des Netzwerks aber mit MPLS konfiguriert, so ist die Abfrage noch nicht zu Ende. Bis jetzt wurden nur alle Ströme im Layer 3-Netzwerk gefunden. Das heisst, alle Multicast-Ströme, die über ein geroutetes IP-Netzwerk fliessen.

Damit die Multicast-Ströme nun auch in einem MPLS-Netzwerk erkannt werden können, muss die Applikation wissen, welche Rolle ein Router im MPLS-Netzwerk übernimmt. Je nachdem, ob der Router die Rolle PE oder P hat, müssen andere Abfragen ausgelöst werden. Die Rolle wird beim Erstellen des Nodes ermittelt und in die Datenbank gespeichert. Die Spalte «role» in der Tabelle «node» enthält diese Informationen.

6.1.3.5 createDataMdt

Mit dieser Funktion wird der Data MDT im MPLS-Netzwerk gefunden. Es wird beim jedem P Router geprüft, ob dieser Data MDT aktiv ist. Dafür wird die Data MDT Id benötigt. Diese Id kann geparst werden, wenn auf dem PE Router der Befehl «show ip mroute» ausgeführt wird. Das folgende Sequenzdiagramm zeigt auf, wie der Data MDT-Strom aus dem MPLS herausgelesen und geparst wird.

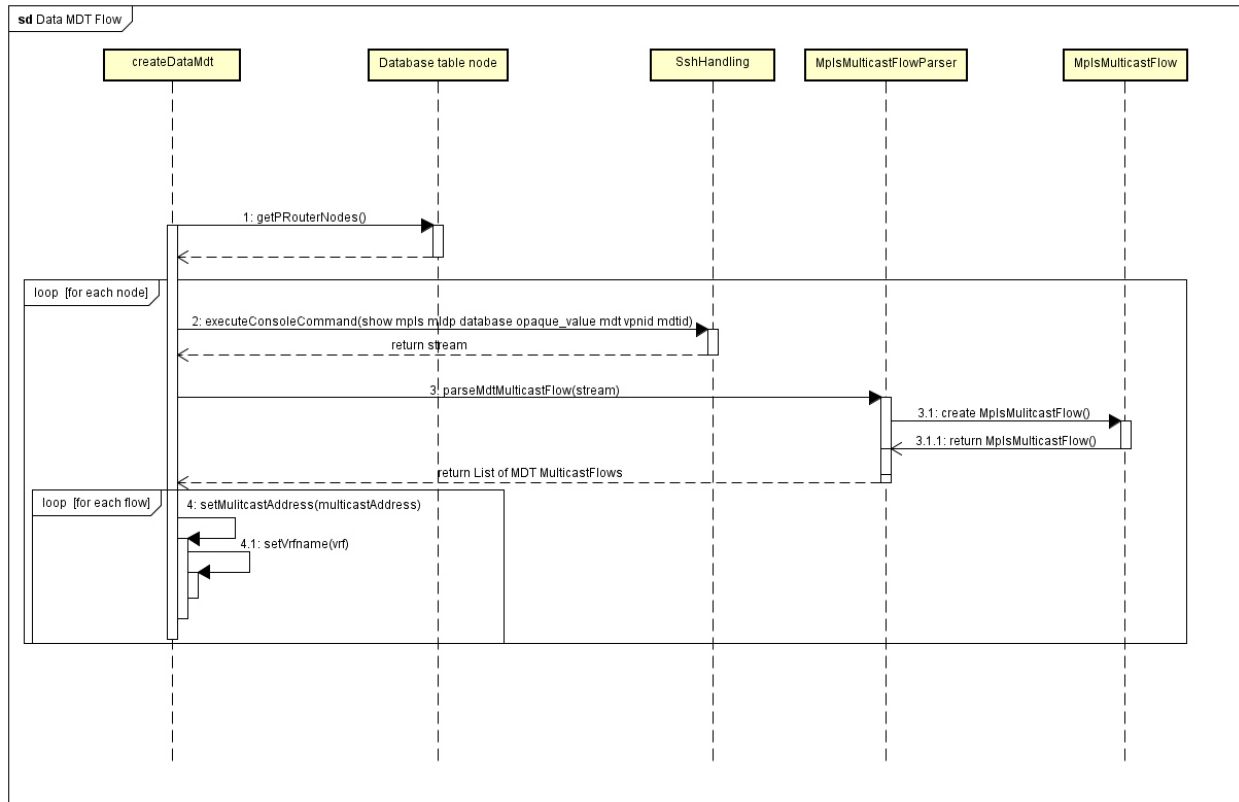


Abbildung 26: Sequenzdiagramm - Finden des Data MDT-Stroms im MPLS-Netzwerk

6.1.3.6 generateDefaultMdt

Wie bereits erwähnt, gibt es im MPLS-Netzwerk den Data MDT und den Default MDT. Der Data MDT ist ähnlich zum normalen Layer 3-Multicast-Strom und kann gerade beim Erkennen generiert werden. Der Default MDT hingegen ist statisch und ändert sich nicht ständig. Es macht darum Sinn, diesen zu generieren, bevor das Netzwerk nach Multicast-Strömen abgesucht wird. Die gesammelten Multicast- Ströme werden in eine HashMap gespeichert. Als Key wird dabei der Node verwendet und als Werte werden alle gefundenen Multicast-Ströme auf dem jeweiligen Node abgespeichert. Der Node kann nur ein P Router sein, da nur auf diesem die Multicast-Ströme verteilt werden.

Das folgende Sequenzdiagramm zeigt auf, wie der Default MDT generiert wird.

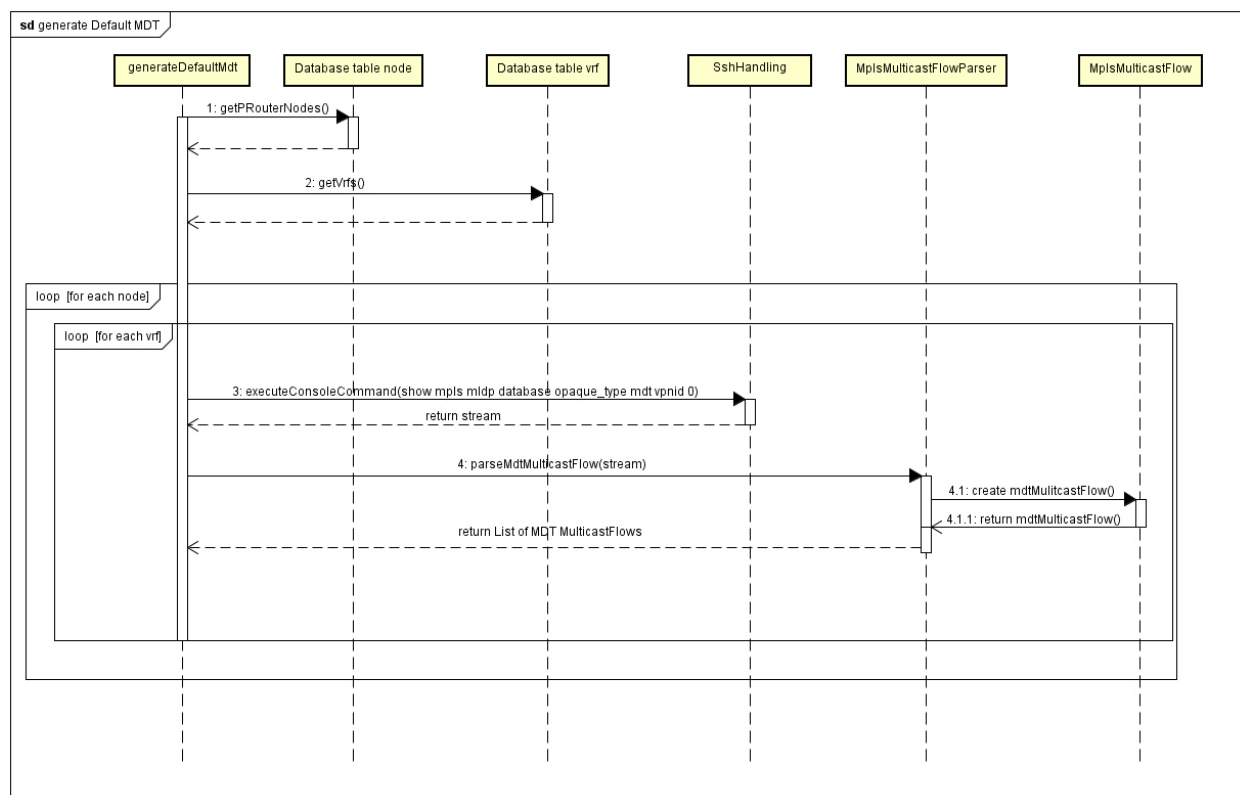


Abbildung 27: Sequenzdiagramm - Generieren des Default MDT

6.1.3.7 createDefaultMdt

Diese Methode wird benötigt, um herauszufinden, welchen Default MDT der Multicast-Strom im Netzwerk verwendet. Pro VRF-Instanz gibt es zwei mögliche Default MDTs. Der Multicast-Strom entscheidet beim Eingang ins MPLS-Netzwerk, welcher Default MDT verwendet wird. Für spätere Abfragen im MPLS-Netzwerk muss daher die Applikation wissen, welcher Default MDT verwendet wird. Mit der Methode «getDefaultRootMdtSsh» auf dem PE Router kann der verwendete Default MDT herausgefunden werden.

Die Methode «createDefaultMdt» greift nun auf die zuvor generierte HashMap zu. Ein Algorithmus sucht auf jedem P Router, ob der Multicast-Strom mit der spezifischen VRF-Instanz aktiv ist.

6.1.3.8 *multicastToLinkCorrelation*

Sind alle aktiven Multicast-Ströme auf einem Netzwerkgerät gefunden, müssen diese Informationen mit den Topologie-Daten abgeglichen werden. Der folgende Algorithmus wird dabei durchlaufen, um festzustellen, ob auf einem Link der Multicast-Strom mit der spezifischen Multicast-Adresse und der VRF-Instanz aktiv ist. Die Namen der Interfaces sind dabei entscheidend, ob ein Multicast-Strom über einen Link verläuft oder nicht. Der Link hat dabei immer eine spezifische Richtung von Source zu Destination. Verläuft nun der Multicast-Strom aber in die andere Richtung, muss in der Datenbank vermerkt werden, dass der Pfeil in entgegengesetzter Richtung zeigen soll. Grundsätzlich müsste nur immer von einer Seite geprüft werden, ob der Link zum Multicast-Strom gehört oder nicht. Es würde theoretisch reichen, dass nur die Links überprüft werden, bei denen auf dem Link die Source ID die Router ID ist. Da es aber Situationen gibt, in denen das gegenüberliegende Netzwerkgerät nicht abgefragt werden kann, müssen auch alle Links mit der Destination-ID geprüft werden.

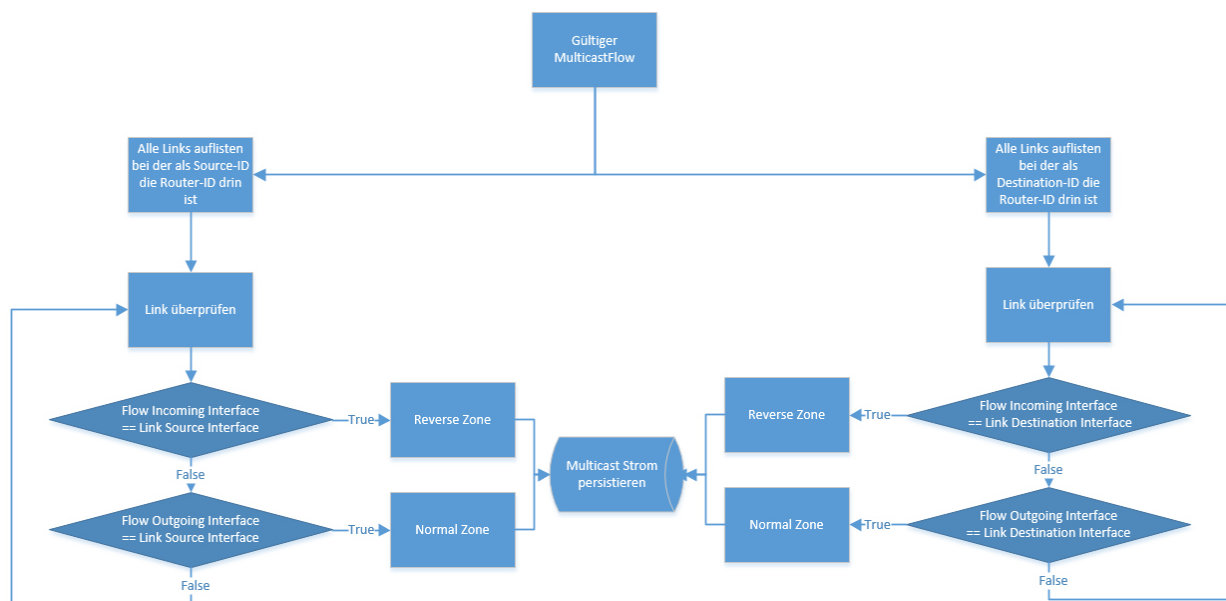


Abbildung 28: Algorithmus Korrelation Multicast-Strom mit Link

6.1.4 Data Layer

Im Data Layer findet der Zugriff auf die Datenbank oder andere Informationsquellen statt.

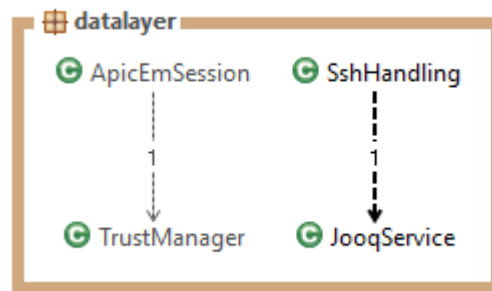


Abbildung 29: Klassen im Data Layer

6.1.4.1 JooqService

Diese Klasse «JooqService» ist für den Zugriff auf die Datenbank verantwortlich. Jeder Teil der Applikation kann Daten von der Datenbank lesen, speichern oder aktualisieren. Der Zugriff geschieht aber immer über diese Klasse und nie direkt. Alle verfügbaren SQL-Abfragen sind mit JOOQ in dieser Klasse modelliert.

6.1.4.2 ApicEmSession

Möchte die Applikation eine REST-Abfrage vom APIC-EM machen, so geschieht dies über diese Klasse. Über die Methode «restGetCall» kann auf die REST-Schnittstelle zugegriffen werden. Das Resultat wird im JSON-Format zurückgegeben. Die Klasse «TrustManager» wird für die Verifikation des Zertifikates vom APIC-EM benötigt.

6.1.4.3 SshHandling

Über diese Klasse werden Abfragen per SSH an die Netzwerkgeräte ausgelöst. Damit die Abfragen funktionieren, muss ein Account verwendet werden, der das Privilege Level 15 besitzt. Ist ein Netzwerkgerät nicht erreichbar, hat die Session ein Timeout von zwei Sekunden konfiguriert. Damit wird verhindert, dass bei einem grossen Netzwerk das Sammeln der Ströme zu lange dauert.

6.2 Timer Service

Damit in regelmässigen Zeitabständen die Topologie- und Multicast-Daten abgerufen werden, benötigt die Applikation einen Timer Service. Der Timer Service wird nach der Initialisierung der Applikation instanziiert und wiederholt aufgerufen.

Im folgenden Codeausschnitt ist der Timer Service beschrieben. Jeder Task wird dem Scheduler hinzugefügt. Der «TopologyService» läuft alle 30 Minuten, im Gegensatz zum «MulticastService», der alle 15 Minuten durchs Netz geht. Die Funktionen werden alle nacheinander im gleichen Thread aufgerufen. Benötigt ein Service länger als geplant und läuft noch, wenn ein neuer starten will, kommt der Service automatisch in die Warteschlange. Es gibt in der Applikation keine Nebenläufigkeit, damit weniger Fehler in der Topologie erscheinen.

```
Timer timer = new Timer();

TopologyService topologyTask = new TopologyService();
timer.schedule(topologyTask, 0, 30*60*1000);

MulticastService multicastTask = new MulticastService();
timer.schedule(multicastTask, 0, 15*60*1000);

CleaningUpService cleanupTask = new CleaningUpService();
timer.schedule(cleanupTask, 0, 24*60*60*1000);
```

6.3 Parser-Klassen

Im Package «parsing» sind alle Parsing-Methoden enthalten. Der Rückgabewert der Methode «executeConsoleCommand» ist ein String mit allen Leerzeichen und Zeilenumbrüchen. Die verschiedenen Parser-Klassen lesen die Informationen aus der SSH-Rückgabe. Da die Parser nicht viel mit Leerzeichen oder Zeilenumbrüchen anfangen können, werden bei den meisten Parser-Klassen diese gerade am Anfang entfernt. Damit das Parsing funktioniert, werden Regular Expressions eingesetzt.

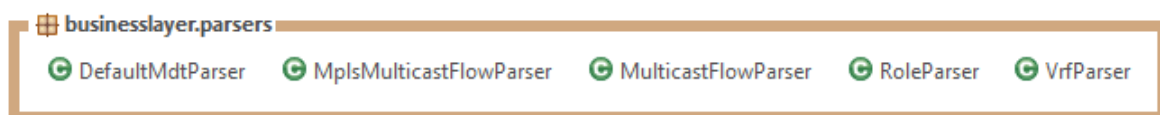


Abbildung 30: Parser-Klassen

6.4 Testlabor

Damit die Applikation in einem realen Umfeld getestet werden kann, wurde ein Labor aufgebaut. Das Labor soll die Gegebenheiten in einem grossen Unternehmensnetzwerk bestmöglich abbilden. Es wurde in Zusammenarbeit mit der AnyWeb AG erstellt und betrieben.

6.4.1 Topologie

Das Labor hat folgende Topologie. Auf der linken Seite ist das Rechenzentrum und auf der rechten Seite die verschiedenen Standorte. In der Mitte befindet sich ein MPLS-VPN-Netzwerk. Normalerweise startet eine Multicast-Strom im Rechenzentrum und wird an die verschiedenen Standorte verteilt. Da dies aber nicht zwingend ist, muss die Applikation auch mit einem Multicast-Strom umgehen können, der bei einem Standort startet.

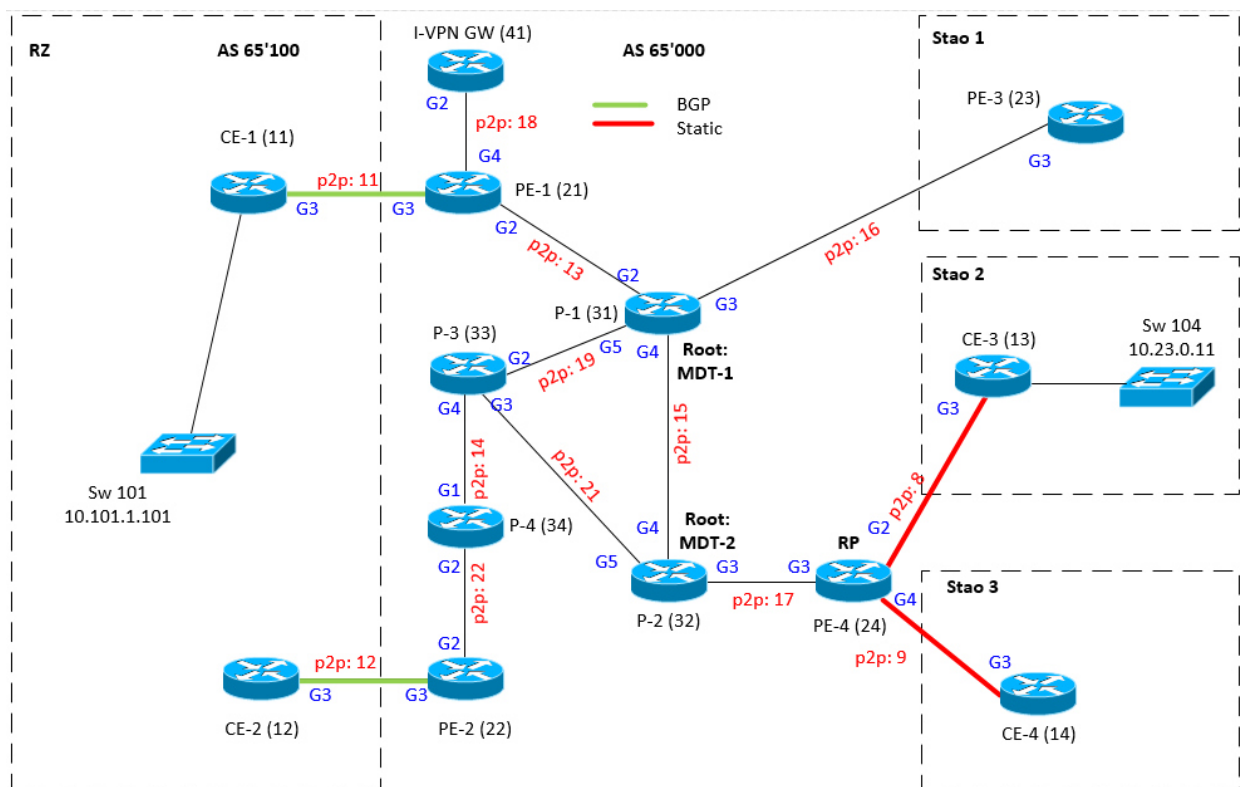


Abbildung 31: Labor Topologie

6.4.2 Hardware

Damit Änderungen am Netzwerk einfach gemacht werden können, sind die meisten Geräte virtualisiert. Als Router wird der Cisco CSR 1000 V Router verwendet. Dieser bietet ein vollwertiges IOS und unterstützt alle benötigten Funktionen.

Im Gegensatz zu den Routern können die Switches nicht wirklich gut virtualisiert werden. Darum werden zwei physische Cisco Catalyst 3650 verwendet.

6.4.3 Konfiguration

Das Labor ist einem normalen Unternehmensnetzwerk nachempfunden. Es gibt ein MPLS-VPN-Netzwerk und verschiedene VRF-Instanzen.

6.4.3.1 VRF

Im Netzwerk gibt es insgesamt drei verschiedene VRF-Instanzen. Die VRF-Instanzen mit den Namen «Blue» und «Red» sind im ganzen Netzwerk konfiguriert. Für Testzwecke ist zudem die VRF-Instanz mit dem Namen «Green» auf dem PE-2 Router konfiguriert.

Der I-VPN Gateway wird benötigt, damit Multicast-Kommunikation zwischen den verschiedenen VRF-Instanzen möglich ist.

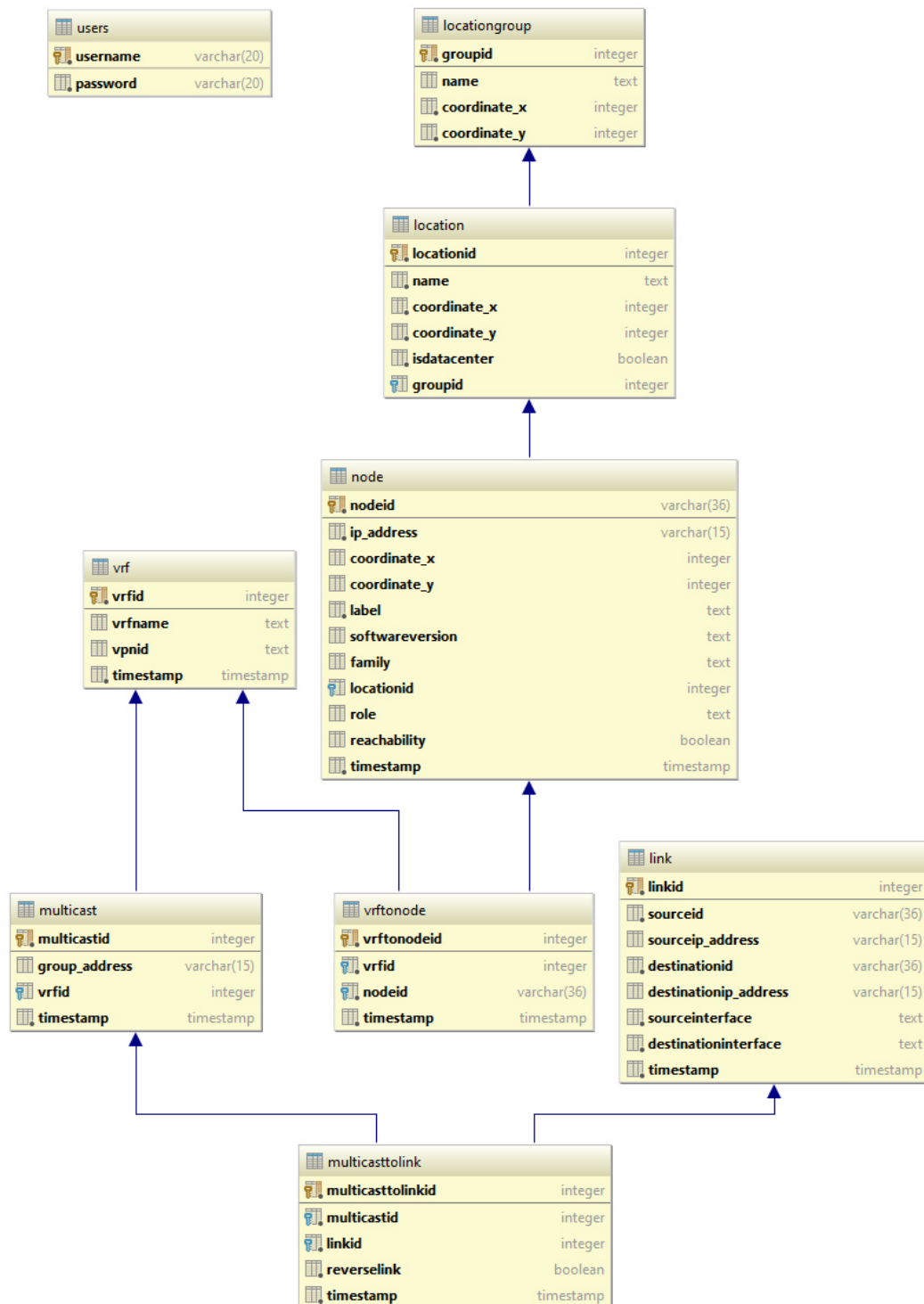
6.4.3.2 MPLS

Die bekannten Rollen aus dem MPLS-Netzwerk sind auch in dieser Topologie ersichtlich. Der Router PE-3 am Standort 1 ist eine Kombination aus PE und CE Router. Da dieser Router mit dem MPLS-Netzwerk verbunden ist, wird er als PE Router bezeichnet.

6.5 Datenbank

6.5.1 Datenbankschema

Die Applikation persistiert die Informationen in das folgende Datenbankschema.



Powered by yFiles

Abbildung 32: Datenbankschema

6.5.2 Eigenschaften

Einzelne Eigenschaften der Datenbank werden in diesem Abschnitt kurz erläutert.

6.5.2.1 *Timestamp*

Damit keine alten Einträge in der Datenbank existieren, gibt es in den meisten Tabellen die Spalte «timestamp». Die Cleanup-Funktionen verwenden diesen Timestamp als Informationsquelle. Als Wert wird das aktuelle Datum mit der Uhrzeit eingetragen.

6.5.2.2 *Varchar*

Ist bekannt, wie lange ein String Wert maximal werden kann, wird der Charaktertyp «varchar» verwendet. Dies ist beispielsweise bei der IP-Adresse der Fall. Diese kann maximal 15 Zeichen lang sein, weshalb der Charakter auf 15 Zeichen beschränkt ist.

6.5.2.3 *Sequenz*

Sobald der Primärschlüssel nicht mehr eindeutig über den APIC-EM ausgelesen werden kann, generiert eine Sequenz innerhalb der Datenbank eine eindeutige Id.

6.5.3 Tabellen

6.5.3.1 *users*

Die Applikation muss mit einem einfachen Login geschützt sein, damit nicht jeder die Topologie des Netzes sehen kann. Dafür wird über diese Tabelle eine einfache Userverwaltung realisiert. Das Passwort und der Username dürfen nur maximal 20 Zeichen lang sein.

```
CREATE TABLE public.users (  
  username CHARACTER VARYING(20) PRIMARY KEY NOT NULL,  
  password CHARACTER VARYING(20) NOT NULL  
);
```

6.5.3.2 *node*

In der Tabelle «node» sind alle Netzwerkgeräte im Netzwerk gespeichert. Jeder Node hat eine eindeutige Node ID, die vom APIC-EM vergeben wird. Die Tabelle Node hat über den Fremdschlüssel „locationid“ eine Relation zur Tabelle «location». Damit weiss die Applikation, zu welcher Location welche Nodes gehören, und kann sie entsprechend auf der Topologie platzieren.

```
CREATE TABLE public.node (  
  nodeid CHARACTER VARYING(36) PRIMARY KEY NOT NULL,  
  ip_address CHARACTER VARYING(15) NOT NULL,  
  coordinate_x INTEGER,  
  coordinate_y INTEGER,  
  label TEXT NOT NULL,  
  softwareversion TEXT,  
  family TEXT,  
  locationid INTEGER,  
  role TEXT,  
  reachability BOOLEAN,  
  timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL,  
  FOREIGN KEY (locationid) REFERENCES public.location (locationid)  
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE  
);
```

6.5.3.3 *location*

Damit es möglich ist, mehrere Netzwerkgeräte zu einem Standort zusammenzufassen, braucht es die Tabelle «location». Mit den Koordinaten wird dem Frontend angezeigt, wo der Standort in der Topologie platziert werden muss.

```
CREATE TABLE public.location (
  locationid INTEGER PRIMARY KEY NOT NULL DEFAULT
nextval('location_locationid_seq'::regclass),
  name TEXT NOT NULL,
  coordinate_x INTEGER NOT NULL,
  coordinate_y INTEGER NOT NULL,
  isdatacenter BOOLEAN NOT NULL,
  groupid INTEGER,
  FOREIGN KEY (groupid) REFERENCES public.locationgroup (groupid)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE
);
```

6.5.3.4 *locationgroup*

Gibt es auf der Topologie sehr viele Standorte, können sie mit der Tabelle «locationgroup» nochmals zusammengefasst werden.

```
CREATE TABLE public.locationgroup (
  groupid INTEGER PRIMARY KEY NOT NULL DEFAULT
nextval('locationgroup_groupid_seq'::regclass),
  name TEXT,
  coordinate_x INTEGER NOT NULL,
  coordinate_y INTEGER NOT NULL
);
```

6.5.3.5 *vrf*

In der Tabelle «vrf» werden alle VRF-Instanzen, die im Netzwerk gefunden wurden, gespeichert. Zusätzlich zum VRF-Namen wird noch die VPN ID gespeichert. Die VPN ID wird für die Abfragen im MPLS-Netzwerk benötigt.

```
CREATE TABLE public.vrf (
  vrfid INTEGER PRIMARY KEY NOT NULL DEFAULT nextval('vrf_vrfid_seq'::regclass),
  vrfname TEXT,
  vpnid TEXT,
  timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL
);
```

6.5.3.6 *vrftonode*

Damit festgestellt werden kann, auf welchem Node welches VRF konfiguriert ist, braucht es die Tabelle «vrftonode». Sie hat jeweils eine Assoziation mit einem Node und einer VRF-Instanz. Wird ein Fremdschlüssel gelöscht, muss auch der Eintrag in dieser Tabelle gelöscht werden, ansonsten wäre die Datenbank inkonsistent.

```
CREATE TABLE public.vrftonode (
  vrftonodeid INTEGER PRIMARY KEY NOT NULL DEFAULT
nextval('vrftonode_vrftonodeid_seq'::regclass),
  vrfid INTEGER NOT NULL,
  nodeid CHARACTER VARYING(36) NOT NULL,
  timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL,
  FOREIGN KEY (nodeid) REFERENCES public.node (nodeid)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE,
  FOREIGN KEY (vrfid) REFERENCES public.vrf (vrfid)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE
);
```

6.5.3.7 *multicast*

In dieser Tabelle sind alle vorhandenen Multicast-Adressen im Netzwerk enthalten. Da die gleiche Multicast-Adresse in zwei verschiedenen VRF-Instanzen gleichzeitig vorkommen kann, muss eine Assoziation mit der Tabelle «vrf» existieren. Wird zudem ein VRF-Instanz gelöscht, ist auch die zugehörige Multicast-Adresse zu löschen.

```
CREATE TABLE public.multicast (
  multicastid INTEGER PRIMARY KEY NOT NULL DEFAULT
nextval('multicast_multicastid_seq'::regclass),
  group_address CHARACTER VARYING(15),
  vrfid INTEGER,
  timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL,
  FOREIGN KEY (vrfid) REFERENCES public.vrf (vrfid)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE
);
```

6.5.3.8 *multicasttolink*

Diese Tabelle beschreibt, auf welchen Links welche Multicast-Adresse konfiguriert ist. Sobald der Link oder die Multicast-Adresse gelöscht wurde, muss der Eintrag in dieser Tabelle gelöscht werden.

```
CREATE TABLE public.multicasttolink (
  multicasttolinkid INTEGER PRIMARY KEY NOT NULL DEFAULT
nextval('multicasttolink_multicasttolinkid_seq'::regclass),
  multicastid INTEGER NOT NULL,
  linkid INTEGER NOT NULL,
  reverselink BOOLEAN NOT NULL,
  timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL,
  FOREIGN KEY (linkid) REFERENCES public.link (linkid)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE,
  FOREIGN KEY (multicastid) REFERENCES public.multicast (multicastid)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE
);
```

6.6 MC Viewer

Die Abkürzung MC Viewer steht für Multicast Viewer. Dies ist eine eigenständige Applikation, welche verwendet wird, um die im Polling Service erhaltenen Daten visuell darzustellen. Sie greift über die Datenbank auf die Informationen zu und bereitet diese visuell auf.

6.6.1 Aufbau Play Framework

Der Aufbau eines Play-Projektes bedingt eine gewisse Ordnerstruktur. Das Projekt MC Viewer ist folgendermassen aufgebaut:

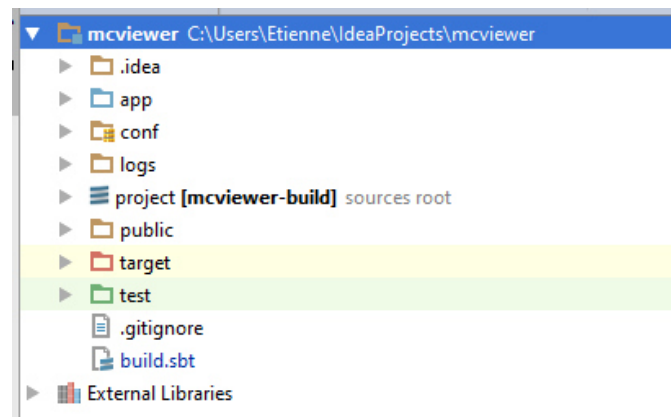


Abbildung 33: Ordnerstruktur Play Framework

Die wichtigsten Ordner werden in nachfolgender Tabelle erläutert.

Ordner	Beschreibung
app	Beinhaltet die gesamte Logik der Applikation. Der Ordner ist unterteilt in die verschiedenen Packages für verschiedene Layer.
conf	Hier sind die Konfigurationsdateien für das Play-Projekt hinterlegt. In der Datei «application.conf» werden die Verbindungsdaten zur Datenbank konfiguriert. Die Datei «routes» beinhaltet die einzelnen Routenkonfigurationen zum Ansteuern der REST-Schnittstelle.
public	Hier befinden sich alle öffentlichen Dateien, die zur Anzeige der Webseite benötigt werden. Der Ordner beinhaltet Stylesheets, Schriftarten, Bilder, eigene JavaScript Dateien sowie JavaScript Libraries (NeXt UI).
test	Beinhaltet die Unit-Tests der Applikation.

6.6.2 Schichtenmodell

Im Ordner «app» befindet sich die Server-seitige Logik der Applikation. Diese wird unterteilt in die verschiedenen Schichten, wie sie auch der Polling Service besitzt. Anstatt eines Service Layer kommt hierbei jedoch der Presentation Layer zur Geltung, welcher für das konkrete Erscheinungsbild der Webseite dient.

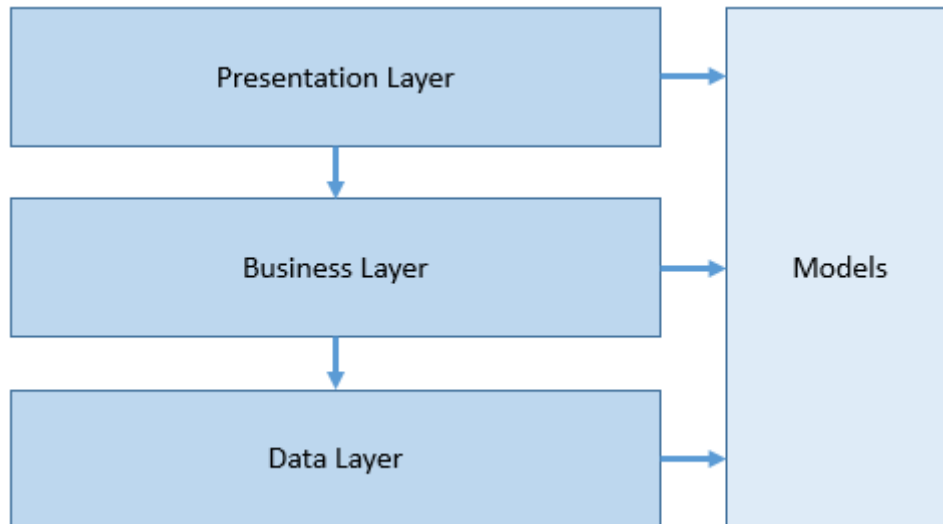


Abbildung 34: Schichtenmodell Frontend

6.6.3 Presentation Layer

Der Presentation Layer selbst ist in kein Package gegliedert. Er bildet das Grundkonstrukt des MVC Frameworks Play. Es beinhaltet unter anderem folgende Packages und Klassen:

Package	Funktionen
controllers	Beinhaltet die Controllerklassen, welche die Steuerung der Webseite übernehmen und die REST-Zugriffe auf das Backend koordinieren.
views	Beinhaltet alle HTML-Dateien, welche die Struktur der Webseite nach aussen hin wiedergeben.
views.formdata	Beinhaltet Java Klassen, welche aktive Formulardaten der Webseite repräsentieren.

Klassen / Dateien	Funktion
Global.java	Enthält allgemeine Methoden zum Abfangen gewisser Play Events wie Start und Stop der Applikation.
Application.java	Enthält die einzelnen Controllermethoden. Diese regeln das Rendern der Webseite und stellen REST-Funktionen zur Verfügung. Methoden, welche nur für authentifizierte Benutzer zur Verfügung stehen sollen, sind mit «@Security.Authenticated» annotiert.
Secured.java	Ist für die Kontrolle über authentifizierte bzw. nicht authentifizierte Benutzer zuständig. Registriert die authentifizierten Benutzer und übergibt sie der Frontend View.
LoginFormData.java	Repräsentiert das Loginformular. Die Methode «validate» wird automatisch beim POST Request ausgeführt und überprüft Benutzername und Passwort mit den in der Datenbank hinterlegten Daten. Bei einem fehlerhaften Login- Versuch werden hier die Error-Meldungen generiert und an das Frontend übertragen.
index.scala.html	Definiert den konkreten Webseiteninhalt, welcher angezeigt werden soll, je nachdem, ob ein Benutzer eingeloggt ist oder nicht.
loginform.scala.html	Stellt das Loginformular zur Verfügung. Wird gerendert, falls kein User eingeloggt ist.
main.scala.html	Definiert das Grundgerüst der Webapplikation, welche auf jeder Seite genau gleich sein soll. Unter anderem wird hier die Titelleiste generiert, wo sich der Logout Button befindet. Ausserdem werden hier die JavaScript Libraries sowie Stylesheets verlinkt, welche der Gestaltung der Webseite dienen.

6.6.4 Business Layer

Der Business Layer übernimmt, wie der Name schon sagt, die Business-Logik der Applikation. Er enthält zwei Klassen, welche die Generierung der im Frontend verwendeten Informationen wie Topologie und Multicast-Strom übernimmt. Eine dritte Klasse ist für die Authentifizierung von Benutzern zuständig.

Klasse	Funktion
MulticastHandler.java	Stellt Funktionen im Zusammenhang mit dem Multicast-Strom bereit. Hier werden die einzelnen Multicastgruppen sortiert, um sie in der richtigen Reihenfolge im Dropdown Menü der Webseite anzuzeigen. Ausserdem werden hier die Daten generiert, welche vom «NeXt UI» benötigt werden, um einen Pfad in der Topologie anzuzeigen.
TopologyHandler.java	Konvertiert die über den Data Layer ausgelesenen Topologie-Daten (Nodes, Links, Standorte) in ein vom «NeXt UI» interpretierbares Format.
AuthenticationHandler.java	Enthält die Methode validateUser, die den Benutzer authentifziert.

6.6.5 Data Layer

Der Data Layer dient als Schnittstelle zur Datenbank. Er enthält eine einzige Klasse, welche alle benötigten Methoden zum Zugriff auf die Datenbank anbietet.

Klasse	Funktion
DatabaseConnection.java	Einzelne Methoden greifen mittels JDBC auf die Datenbank zu und füllen die Informationen in die verschiedenen Models ab. Ebenfalls wird hier die Validierung des Logins durchgeführt, um einen User zu authentisieren.

6.6.6 Models

Die Models werden für den Datenaustausch zwischen den verschiedenen Layer benötigt. Sie werden, mit Ausnahme der Models im Sub-Package «nextuimodels», im Data Layer instanziiert. Die speziellen NeXt UI Models dienen der Übergabe der Daten an das NeXt UI. Sie müssen ein bestimmtes Format besitzen, damit sie vom NeXt UI interpretiert werden können.

6.6.7 REST-Schnittstelle

Im Ordner «conf» befindet sich die Datei «routes». Hier werden die verschiedenen Routen der Webseite definiert, und es wird definiert, welche Methoden im Controller ausgeführt werden müssen.

```
# Routes
# This file defines all application routes (Higher priority routes first)
# ~~~~

# Home page
GET / controllers.Application.index()
POST /login controllers.Application.postLogin()
GET /logout controllers.Application.logout()
GET /getTopology controllers.Application.getTopology()
GET /getAvailableMulticastGroups controllers.Application.getAvailableMulticastGroups()
GET /getNodeLinksForMulticastGroup/:multicastGroup controllers.Application.getNodeLinksForMulticastGroup(multicastGroup: String)

# Map static resources from the /public folder to the /assets URL path
GET /assets/*file controllers.Assets.at(path="/public", file)
```

Abbildung 35: Ausschnitt aus Datei "routes"

Eigens definierte Routen:

HTTP Action	Route	Beschreibung
GET	/	Lädt die Indexseite.
POST	/login	Übermittelt Login Credentials.
GET	/logout	Führt Logout durch.
GET	/getTopology	Gibt Geräte, Links und Standorte zurück.
GET	/getAvailableMulticastGroups	Gibt alle zurzeit verfügbaren Multicast-Adressen zurück.
GET	/getMulticastLinksWithVrf/:multicastGroup/:vrf	Gibt alle Links zurück, auf denen der gewünschte Multicast-Strom in einer VRF-Instanz läuft.
GET	/getMulticastLinksWithVrf/:multicastGroup	Gibt alle Links zurück, auf denen der gewünschte Multicast-Strom über alle VRF-Instanzen läuft.

6.6.8 Client

Im Ordner «public/javascripts» befindet sich die Datei «application.js», welche für die Client-seitige Logik zuständig ist. Hier wird der gesamte Umgang mit dem NeXt UI definiert. Server-seitige Informationen, zum Beispiel verfügbare Multicast-Adressen, werden jeweils über AJAX-Aufrufe auf die definierten REST-Routen erhalten und entsprechend gehandhabt.

Um einfacheres JavaScript zu schreiben, wird ausserdem jQuery eingesetzt.

6.6.9 Ablaufsequenzen

6.6.9.1 Laden der Webseite

Diese Sequenz wird jeweils ausgeführt, wenn die Webseite geladen oder auf den «Refresh» Button gedrückt wird.

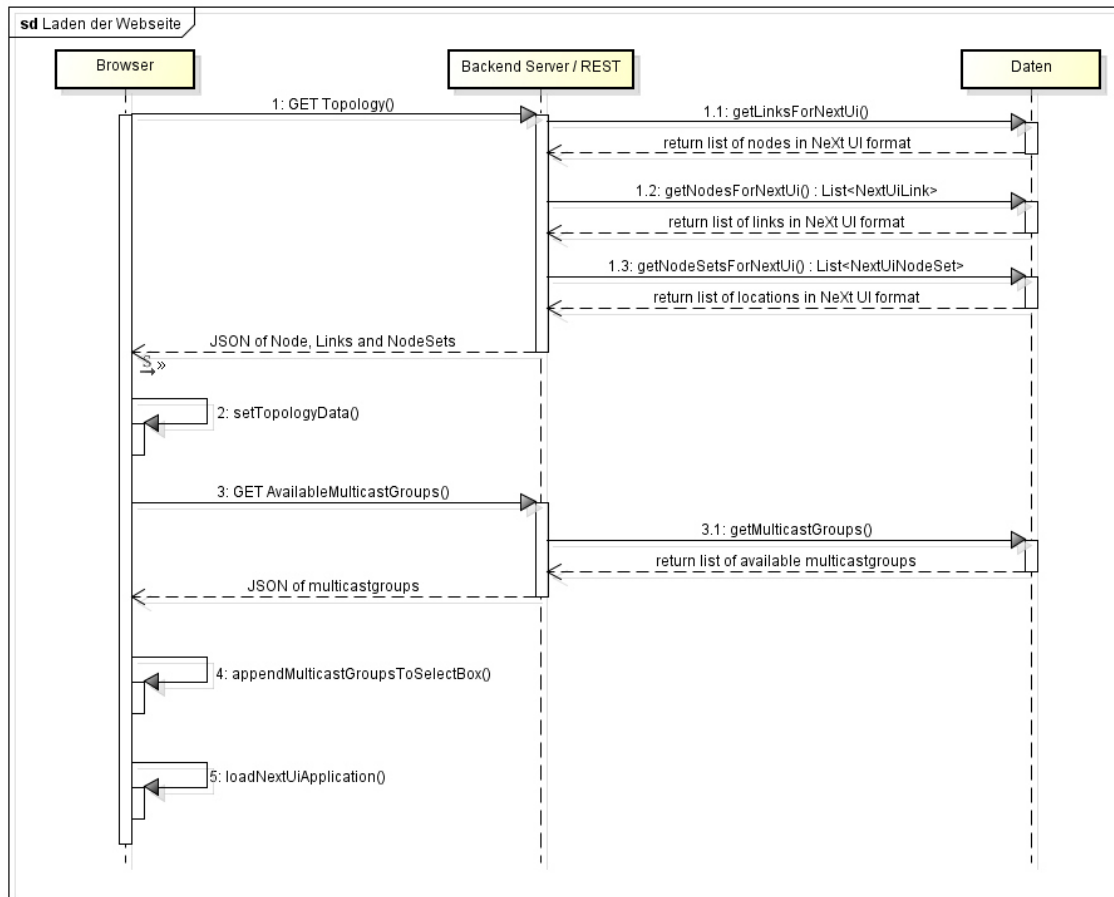


Abbildung 36: Sequenzdiagramm - Laden der Webseite

Zuerst fordert der Browser mittels einem GET Request Informationen zur Topologie an. Diese erhält er in Form eines JSON. Diese Daten werden dann im NeXt UI als Topologie-Daten hinterlegt. In einem zweiten Schritt werden verfügbare Multicast-Adressen angefordert, welche danach in das Dropdown- Menü geladen werden. Zuletzt wird die NeXt UI-Applikation gestartet.

6.6.9.2 Multicast-Strom auswählen

Diese Sequenz wird ausgeführt, sobald ein Multicast-Strom aus der Dropdown-Liste ausgewählt wird.

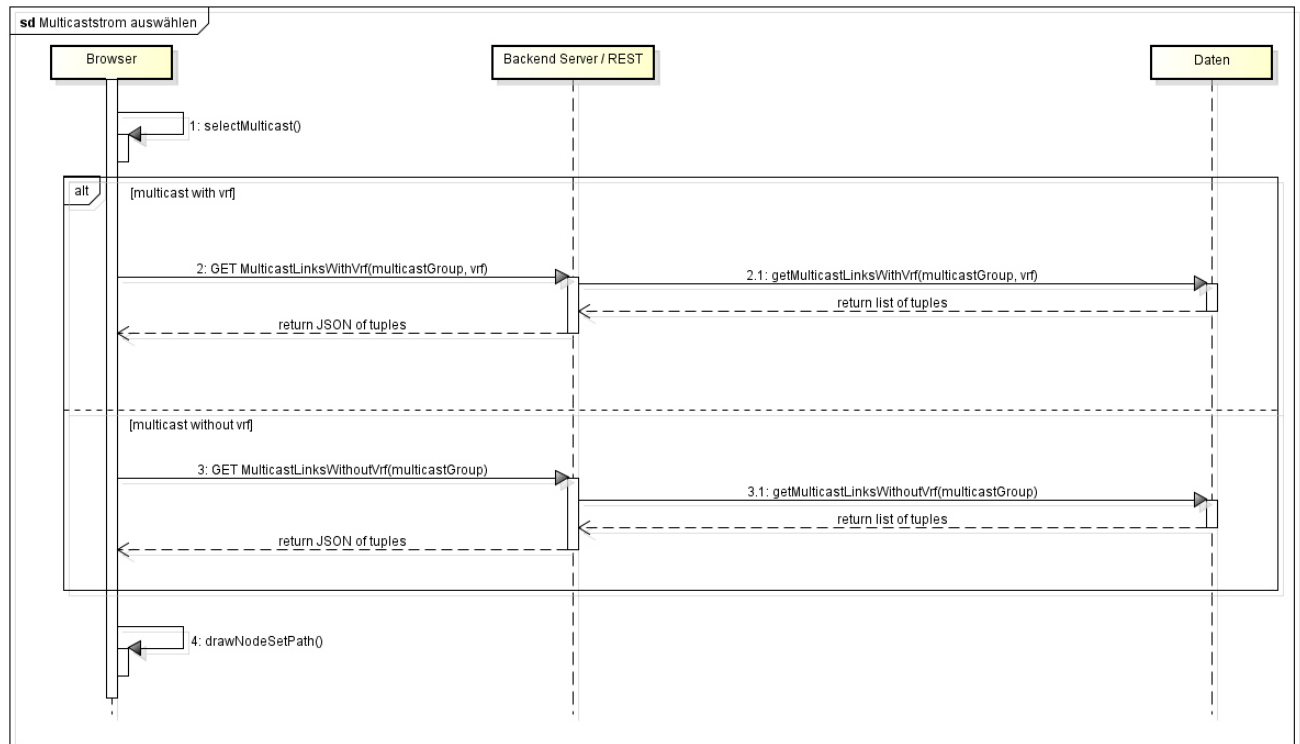


Abbildung 37: Sequenzdiagramm – Multicast-Strom auswählen

Bei der Auswahl eines Multicast-Stroms werden zwei verschiedene Ströme unterschieden. Je nachdem, ob der Multicast-Strom für eine bestimmte VRF-Instanz angezeigt werden soll oder nicht, wird ein anderer GET Request gesendet. Als Antwort auf die Anfrage an den Backend Server erhält der Browser jeweils eine Liste, die definiert, über welche Links der Multicast-Strom fließt. Die Liste besteht aus einem 2-Tupel. Ein solches Tupel besteht aus Source-Node-ID und Destination-Node-ID und definiert, von wo bis wo der Multicast-Strom fließt. Jedes Tupel entspricht somit einem Link in der Topologie, über den ein Multicast-Strom eingezeichnet werden muss.

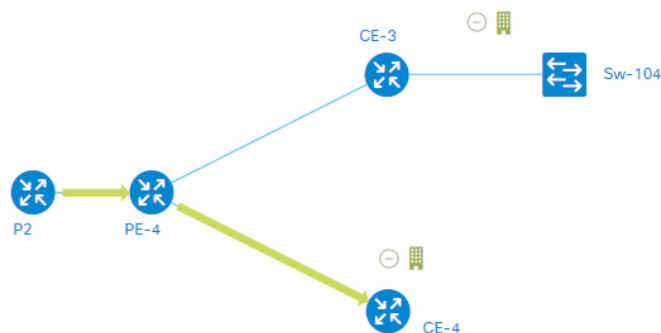


Abbildung 38: Beispiel für eine Return-Liste bestehend aus zwei definierten Tupel

6.7 NeXt UI

Innerhalb des MC Viewer wird die JavaScript Library NeXt UI verwendet. Sie dient der Anzeige der Topologie und der einzelnen Multicast-Ströme auf der Webseite. Die Library wird stetig weiterentwickelt und ist daher nicht ganz frei von Fehlern. Im offiziellen Release [4] der Library von Cisco fehlt unter anderem eine wichtige Funktion, die für das Projekt benötigt wurde. Es handelt sich hierbei um die Funktion «NodeSetPath», die es ermöglicht, einen Pfad über verschiedene «NodeSets» einzuzichnen. Diese Funktion ist jedoch in einem Release auf dem offiziellen GitHub Repository der Entwickler verfügbar. Deshalb wurde für die Umsetzung der Applikation die GitHub Version von NeXt UI verwendet. [5]

6.7.1 Begrifflichkeiten

Durch die Verwendung von NeXt UI sind einige Begrifflichkeiten aufgetaucht, die im Zusammenhang mit der Webapplikation eventuell unklar erscheinen. Die nachfolgende Tabelle soll daher diese Begrifflichkeiten erklären.

Begriff	Erklärung / Verwendung
Node	Ein einzelnes Gerät innerhalb der Topologie.
Link	Ein Link zwischen zwei Geräten.
NodeSet	Eine Gruppierung verschiedener Geräte. In der Webapplikation wurde dies verwendet, um einzelne Geräte in Standorte zusammenzufassen.
NodeSetPath	Einen Pfad in der Netzwerktopologie, der über eine Gruppierung von Geräten (NodeSet) hinausgeht.

6.7.2 Problem mit NodeSetPath

Wie bereits erwähnt bietet die GitHub-Version von NeXt UI die Funktion «NodeSetPath» an, welches ermöglicht, einen Pfad durch ein «NodeSet» zu zeichnen. In der Applikation wird diese Funktion verwendet, um einen Multicast-Strom über einen Standort hinweg einzeichnen zu können.

Ein solcher Pfad kann jedoch nur eingezeichnet werden, wenn sich dieser über mindestens drei verschiedene Nodes erstreckt. Da die Multicast-Strom-Logik der Webapplikation jedoch auf zwei Nodes basiert (2-Tupel), war dies keine Option und es musste eine manuelle Anpassung an der JavaScript Library vorgenommen werden. Die Änderungen sind in untenstehenden Code-Ausschnitten gelb markiert.

Original Code

```

for (var i = 0; i < visibleNodesLength - 1; i++) {
  var sourceNode = visibleNodes[i];
  var targetNode = visibleNodes[i + 1];
  var line = new Line(sourceNode.vector(), targetNode.vector());
  // padding start
  if (i === 0) {
    line = line.pad(padding, 0);
    d.push('M', line.start.x, line.start.y);
  }
  else if (i == visibleNodesLength - 2) {
    line = line.pad(0, arrow ? padding + strokeWidth : padding);
    d.push('L', line.start.x, line.start.y);
    d.push('L', line.end.x, line.end.y);
  }
  else {
    d.push('L', line.start.x, line.start.y);
  }
}
this._drawArrow();
return d.join(" ");

```

Geänderter Code

```

for (var i = 0; i < visibleNodesLength - 1; i++) {
  var sourceNode = visibleNodes[i];
  var targetNode = visibleNodes[i + 1];
  var line = new Line(sourceNode.vector(), targetNode.vector());
  // padding start
  if (i === 0) {
    line = line.pad(25, 25);
    d.push('M', line.start.x, line.start.y);
    d.push('L', line.end.x, line.end.y);
    this._drawArrow();
  }
  else if (i == visibleNodesLength - 2) {
    line = line.pad(0, arrow ? padding + strokeWidth : padding);
    d.push('L', line.start.x, line.start.y);
    d.push('L', line.end.x, line.end.y);
    this._drawArrow();
  }
  else {
    d.push('L', line.start.x, line.start.y);
  }
}
//this._drawArrow();
return d.join(" ");

```

Nach der Anpassung konnte die Funktion wie gewünscht verwendet werden.

7 Testing

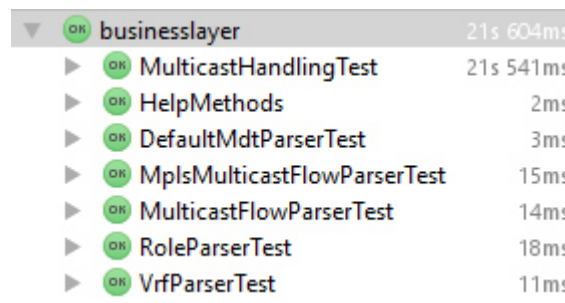
Das Testing der Applikation wurde in zwei Abschnitte aufgeteilt. Das Zusammenspiel aller Komponenten wurde mit Systemtests überprüft und einzelne kritische Funktionen wurden zusätzlich mit Unit Test ergänzt. Viele Komponenten konnten jedoch nicht einzeln getestet werden, da sie zum Beispiel Datenbankzugriffe oder SSH-Verbindungen benötigen. Eine aufwändige Mocking-Infrastruktur wäre nötig gewesen, die sich für diesen Rahmen der Arbeit aber nicht gelohnt hätte. Darum waren die Systemtests im Fokus, und diese konnten mit dem aufgebauten Lab auch intensiv überprüft werden.

7.1 Unit Test

Es wurden primär Parsing Funktionen mit Unit-Tests überprüft. Diese sind ein wichtiger Teil der Applikation und müssen daher richtig funktionieren.

7.1.1 Polling Services Tests

Die folgenden Tests wurde im Polling Service erstellt.

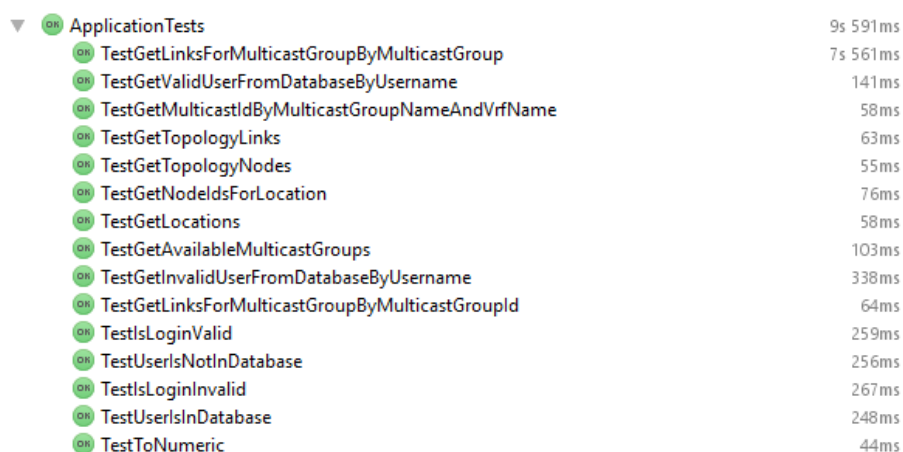


OK	businesslayer	21s 604ms
OK	MulticastHandlingTest	21s 541ms
OK	HelpMethods	2ms
OK	DefaultMdtParserTest	3ms
OK	MplsMulticastFlowParserTest	15ms
OK	MulticastFlowParserTest	14ms
OK	RoleParserTest	18ms
OK	VrfParserTest	11ms

Abbildung 39 Unit Test für Polling Service

7.1.2 Frontend Tests

Der MC Viewer greift im Hintergrund oft auf die Datenbank zu. Deshalb ist es sinnvoll, dass die Datenbank-Zugriffsmethoden mit Unit Tests abgesichert sind. Dazu wurde eine Testdatenbank mit vordefinierten Inhalten angelegt. Damit die Unit Tests funktionieren, muss zwingend diese Testdatenbank verwendet werden, da sich die Tests auf diese Inhalte beziehen.



OK	ApplicationTests	9s 591ms
OK	TestGetLinksForMulticastGroupByMulticastGroup	7s 561ms
OK	TestGetValidUserFromDatabaseByUsername	141ms
OK	TestGetMulticastIdByMulticastGroupNameAndVrfName	58ms
OK	TestGetTopologyLinks	63ms
OK	TestGetTopologyNodes	55ms
OK	TestGetNodeIdsForLocation	76ms
OK	TestGetLocations	58ms
OK	TestGetAvailableMulticastGroups	103ms
OK	TestGetInvalidUserFromDatabaseByUsername	338ms
OK	TestGetLinksForMulticastGroupByMulticastGroupId	64ms
OK	TestIsLoginValid	259ms
OK	TestUserIsNotInDatabase	256ms
OK	TestIsLoginInvalid	267ms
OK	TestUserIsInDatabase	248ms
OK	TestToNumeric	44ms

Abbildung 40: Unit Tests für MC Viewer

7.2 Systemtest

In einem ersten Teil der Systemtests wurde zuerst geprüft, ob alle Komponenten korrekt miteinander funktionieren: Macht das Backend regelmässig Abfragen beim APIC-EM und werden die Multicast-Ströme im Netzwerk gesammelt. Der zweite Teil befasste sich mehr mit der korrekten Anzeige der Ströme.

Die Applikation wurde dahingehend geprüft, ob folgende Testszenarios korrekt funktionieren und der Multicast-Strom korrekt dargestellt wird.

Testszenario	Erfolgreich
Anzeige: Kein Multicast-Strom	Ja
Anzeige: Data MDT Multicast-Strom	Ja
Anzeige: Default MDT Multicast-Strom	Ja
Anzeige: Mehrere Empfänger, gleiche VRF-Instanz, Data MDT	Ja
Anzeige: Mehrere Empfänger, unterschiedliche VRF-Instanz, Data MDT	Ja
Anzeige: Mehrere Empfänger, gleiche VRF-Instanz, Default MDT	Ja
Anzeige: Mehrere Empfänger, unterschiedliche VRF-Instanz, Default MDT	Ja
Anzeige: Mehrere Sender, ein Empfänger, Data MDT	Ja
Anzeige: Mehrere Sender, ein Empfänger, Default MDT	Ja
Netzwerkbearbeitung: Zusätzlicher P-Router, Default MDT	Ja
Netzwerkbearbeitung: Node ist nicht erreichbar	Ja

Weitere Informationen und Ergänzungen zu den Systemtests sind im Anhang zu finden.

8 Ergebnisdiskussion & Ausblick

8.1 Ergebnisdiskussion

Die entstandene Applikation erfüllt alle Anforderung aus der Anforderungsanalyse. Von der Anforderungsliste wurden jedoch keine optionalen Features implementiert.

8.1.1 Multicast-Ströme

Laufende Multicast-Ströme werden im MPLS-Netzwerk und im normalen Layer 3-Netzwerk erkannt. Sobald der Multicast-Strom nur noch auf Layer 2-Ebene fließt, kann die Applikation den Strom nicht mehr weiterverfolgen. Im MPLS-Netzwerk kann zwischen Default MDT- und Data MDT-Strom unterschieden werden. Die Multicast-Ströme werden regelmässig gesammelt und in die Datenbank persistiert.

8.1.1.1 Default MDT

Im MPLS-Netzwerk wird der Default MDT an alle PE Router weitergeleitet; dies kann auf dem folgenden Bild gut erkannt werden. Alle Pfeile vom PE-2 Router bis zum Default Root Router P2 sind in entgegengesetzter Richtung, da der Sender am PE-2 Router angeschlossen ist. Alle anderen Pfeile im MPLS-Netzwerk müssen vom P2 Router zu den PE-Routern zeigen.

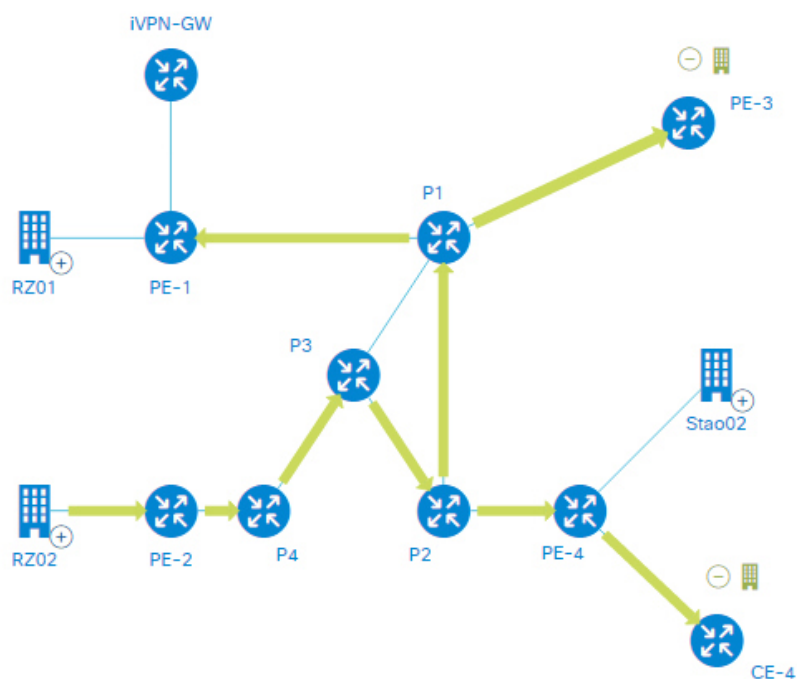


Abbildung 41: Default MDT über mehrere P Router

8.1.1.2 Data MDT

Auf dem folgenden Bild ist ein Data MDT abgebildet. Der Strom wird nicht an alle PE Router weitergesendet. Der Multicast-Strom hat seinen Sender im RZ01 und zwei Empfänger am Stao02 und Stao03.

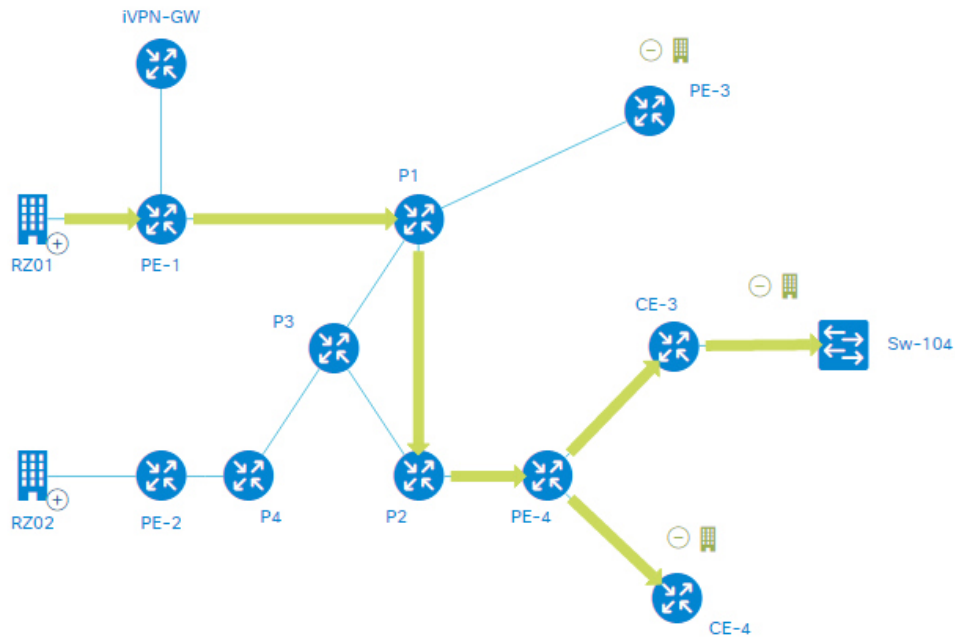
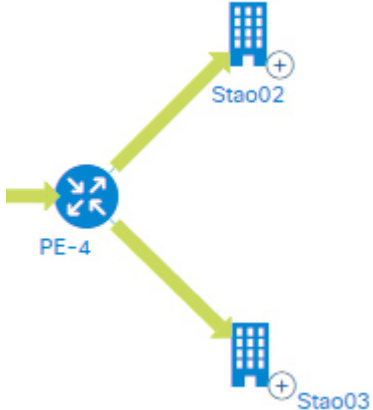
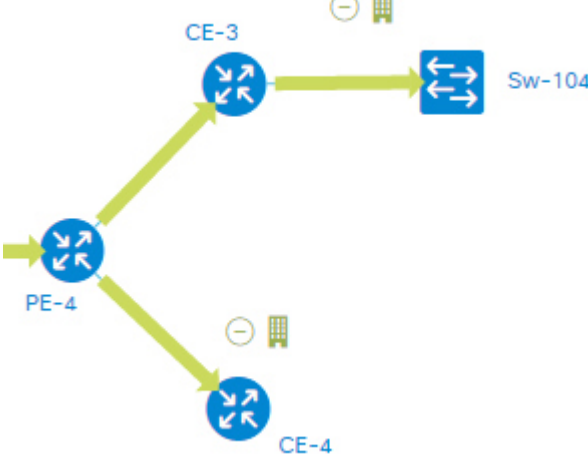


Abbildung 42: Data MDT im MPLS-Netzwerk

8.1.2 Topologie

Die Topologie-Daten werden vom APIC-EM abgefragt und in der Datenbank abgespeichert. Mehrere Netzwerkgeräte können zu einer Gruppe bzw. einem Standort zusammengefasst werden. Die Netzwerkgeräte an einem Standort und im Rechenzentrum werden dabei in der Topologie dynamisch verteilt. Netzwerkgeräte im Core-Bereich des Netzwerks sind fest definiert. Alle Konfigurationen zur Topologie können in einem CSV definiert werden und werden nach dem Starten der Applikation eingelesen.

Zusammengefasst Auf diesem Bild ist der Standort zusammengefasst, aber es ist erkennbar, dass ein Multicast-Strom zu diesem Standort fließt.	Aufgeklappt Der Standort ist aufgeklappt und der Multicast-Strom wird bis zum nächsten Netzwerkgerät innerhalb des Standortes angezeigt.
 <p>Abbildung 43: Multicast-Strom zu einem zusammengefassten Standort</p>	 <p>Abbildung 44: Multicast-Strom zu einem aufgeklappten Standort</p>

8.1.3 Performance

Mit Hilfe der Applikation können nun Netzwerktechniker viel schneller erkennen, über welche Netzwerkgeräte sich ein Multicast-Strom bewegt. Ein kurzer Vergleich soll zeigen, wieviel weniger Aufwand mit der Applikation nötig ist.

8.1.3.1 Ohne Applikation

Will ein Netzwerktechniker den Multicast-Strom ohne die Applikation erkennen können, muss zuerst die Netzwerktopologie bekannt sein. Danach muss ein Netzwerkgerät als Einstiegspunkt gewählt werden. Auf dem Gerät wird per SSH eingeloggt und anhand der Befehlsausgaben analysiert, zu welchem Netzwerkgerät der Multicast-Strom als nächstes fließt. Auf dem nächsten Netzwerkgerät muss das gleiche wieder gemacht werden, bis das Ende des Stroms gefunden wurde. Dies dauert je nach Größe des Netzwerkes und Kenntnissen des Technikers mehrere Minuten.

8.1.3.2 Mit Applikation

Beim Starten der Applikation wird die Netzwerktopologie ausgelesen. Danach wird auf allen Netzwerkgeräten nach Multicast-Strömen gesucht. Der ganze Vorgang dauert für einen einzelnen Multicast-Strom im Netzwerk weniger als 40 Sekunden. Der Multicast-Strom wird gerade mit Pfeilen in der Topologie angezeigt.

8.2 Ausblick

Mit der Applikation ist es im Moment möglich, in einem Layer-3 und MPLS-VPN Netzwerk alle vorhandenen Multicast-Ströme anzuzeigen. Die folgenden Punkte könnten die Applikation aber noch erheblich verbessern.

8.2.1 Parallelisierung

Im Moment verläuft das Sammeln der Multicast-Ströme rein sequentiell. In einem kleinen Netzwerk mit wenigen Multicast-Strömen ist dies weniger ein Problem. In einem riesigen Enterprise-Netzwerk mit vielen Sendern und Empfängern könnte jedoch eine Abfrage sehr lange dauern. Damit die Applikation aber immer noch alle Multicast-Ströme in akzeptabler Zeit sammeln könnte, müssten die Abfragen an die Netzwerkgeräte parallelisiert werden. Somit könnten auf mehreren Netzwerkgeräten die Analysen parallel verlaufen und der Zeitgewinn wäre dadurch erheblich.

8.2.2 Hosting-Infrastruktur

Das Backend wird im Moment noch über eine klassische JAR-Datei gestartet. Gibt es Probleme oder stürzt die Applikation ab, muss sie auf dem ausgeführten Server neu gestartet werden. Durch die Log-Meldungen ist ein gewisses Monitoring möglich, welches aber nicht gerade sehr komfortabel ist. Würde der Polling Service im Kontext eines Applikationsservers gestartet, wären die Möglichkeiten erheblich höher. Das Monitoring der Applikation wäre besser und einfacher. Auch könnten allfällige Konfigurationen einfacher gemacht werden.

8.2.3 Polling-Mechanismus

8.2.3.1 Log Analyzer

Der Zugriff über SSH ist suboptimal und birgt erhebliche Gefahren und Fehler. Heraus zu finden, welche Multicast-Ströme im Netzwerk gerade aktiv sind, wäre vielleicht auch mit einem Log-Analyzer möglich gewesen. Alle Netzwerkgeräte würden Meldungen an den Log-Analyzer schicken, der diese analysiert und weiterverarbeitet. Somit wäre eine Echtzeit-Anzeige der Multicast-Ströme möglich. Das Netzwerk würde aber unnötig mit vielen Log-Meldungen belastet werden.

8.2.3.2 REST-Schnittstellen

Der APIC-EM und auch die Router Software wird ständig weiterentwickelt. Allenfalls wäre es möglich, dass die Router ebenfalls eine REST-Schnittstelle anbieten. Somit könnte der mühsame und unsichere Umweg über SSH verhindert werden. Es könnten auch zusätzliche Funktionen zum APIC-EM hinzukommen, die den ganzen Mechanismus erleichtern könnten.

8.2.4 Datenstruktur

Wird ein Multicast-Strom auf einem Netzwerkgerät erkannt, hat dieser keinen Zusammenhang zu einem gleichen Strom mit gleicher Adresse auf einem anderen Gerät. Erst das Frontend baut aus den Informationen aus der Datenbank den kompletten Strom. Gäbe es im Backend eine Datenstruktur, welche die Multicast-Ströme direkt abbilden könnte, wären die Möglichkeiten viel grösser. Zum Beispiel könnte jedes Netzwerkgerät als Knoten abgebildet werden. In diesem Knoten wäre gespeichert, welches seine Vorgänger- und welches seine Nachfolger-Netzwerkgeräte sind. Die Applikation könnte dadurch besser überprüfen, ob der Multicast-Strom korrekt verläuft und so direkt allfällige Massnahmen ergreifen.

9 Statistiken

9.1 Codezeilen

Beim Polling Service sind es 3'340 Codezeilen. Die von Jooq generierten Klassen sind aber nicht mit eingerechnet, da diese nicht selber geschrieben wurden.

Beim Multicast Viewer sind es insgesamt 1294 Codezeilen, wobei nur 973 Codezeilen in Java geschrieben sind. Die restlichen 321 verteilen sich auf die verschiedenen HTML / CSS und JavaScript Dateien.

9.2 Arbeitsaufwand

Die Studienarbeit ist ein Modul welches 8 ECTS Punkte gibt. Rechnet man mit einem Aufwand von ca. 30h pro ECTS, so ergibt dies einen effektiven Arbeitsaufwand von 240 Stunden pro Person. Aufgeteilt auf 15 Wochen sind dies 32 Stunden pro Woche.

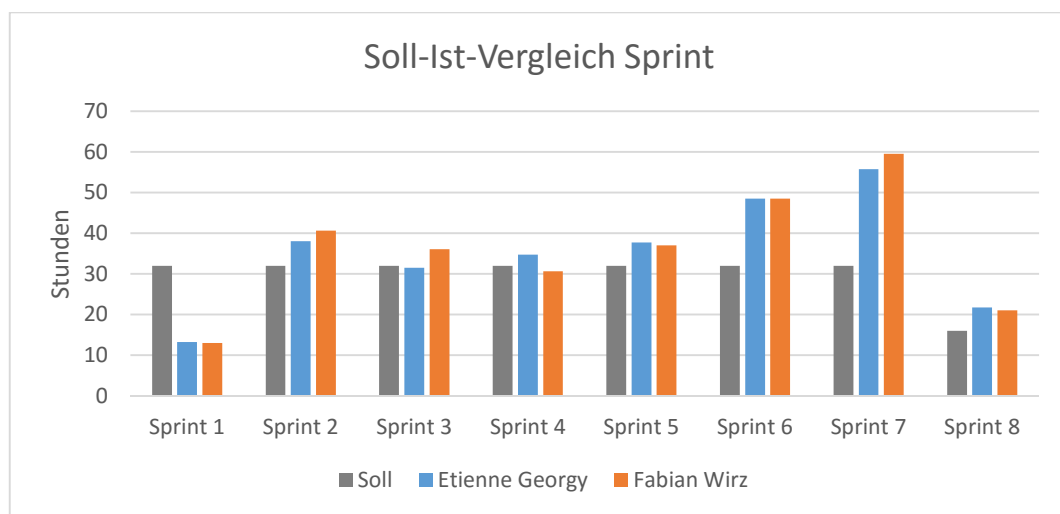


Abbildung 45: Soll-Ist-Vergleich

Wie man am Diagramm erkennen kann, wurde mit Ausnahme des ersten Sprints die jeweilige Zeitvorgabe erfüllt und meistens auch übertroffen. Gegen Ende der Arbeit wurde um einiges mehr gearbeitet als ursprünglich geplant. Dies, weil die Applikation fertiggestellt werden musste.

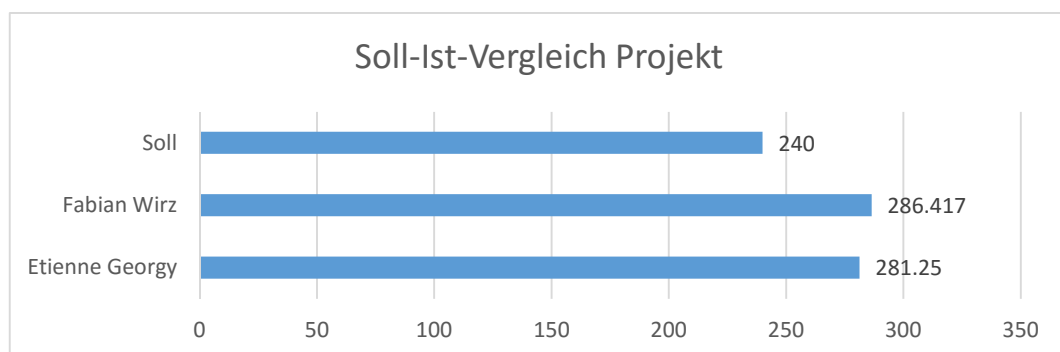


Abbildung 46: Total Aufwand pro Person

10 Glossar

APIC-EM	Application Policy Infrastructure Controller Enterprise Module <i>Software Defined Network Controller von Cisco</i>
MPLS-VPN	Multiprotocol Label Switching Virtual Private Network <i>WAN Technologie um verschiedene Computernetzwerke miteinander zu verknüpfen</i>
MDT	Multicast Distribution Tree <i>Baum über welcher ein Multicast-Strom im MPLS Netzwerk fließt</i>
OR-Mapper	Objektrelationale Abbildung <i>Technik um Datenbanken in Java Klassen abbilden zu können</i>
JPA	Java Persistence API <i>OR-Mapper für Java</i>
JDBC	Java Database Connectivity <i>Einfache Art um mittels Java auf eine Datenbank zugreifen zu können</i>
JSF	Java Server Faces <i>Framework für Java Webapplikationen</i>
JAX-RS	Java API for RESTful Web Services <i>Spezifikation einer Programmierschnittstelle mit der in Java REST Services programmiert werden können</i>
VRF	Virtual Routing and Forwarding <i>Virtuelle Routing-Instanz auf einem physischen Router</i>
CDP	Cisco Discovery Protocol <i>Protokoll das zur Identifizierung benachbarter Netzwerkgeräte dient</i>
SSH	Secure Shell <i>Netzwerkprotokoll um Befehle auf externem Netzwerkgerät abzusetzen</i>
CSV (Dateiformat)	Comma-separated values <i>beschreibt den Aufbau einer Textdatei zur Speicherung strukturierter Daten</i>
SNMP	Simple Network Management Protocol <i>Protokoll um Netzwerkgeräte zu managen</i>
REST	Representational State Transfer <i>Programmierparadigma für verteilte Systeme, insbesondere Webservices</i>

JOOQ	Java Object Oriented Querying <i>Eine Software Library die den Zugriff auf Datenbanken in Java erleichtert</i>
FQDN	Fully Qualified Domain Name <i>Kompletter Hostname in einer Netzwerkumgebung mit DNS</i>
AJAX	Asynchronous JavaScript And XML <i>Konzept für asynchrone Datenübermittlung zwischen Webbrowser und Webserver</i>
MVC	Model View Controller <i>Programmier-Pattern für Frontend Applikationen</i>
JAR	Java Archive <i>Enthält alle Informationen zu einer Java Applikation und dient als ausführbare Datei um diese zu starten</i>
OSPF	Open Shortest Path First <i>Link-State-Routing Protokoll</i>

11 Literatur und Quellenverzeichnis

- [1] J. Kurose und K. Ross, Computernetzwerke - Der Top-Down-Ansatz, Addison-Wesley, 2008, pp. 537-540.
- [2] „Play Framework Dokumentation,“ [Online]. Available: <https://www.playframework.com/documentation/2.5.x/Home>. [Zugriff am 15 März 2016].
- [3] Jooq, „Jooq Dokumentation,“ [Online]. Available: <http://www.jooq.org/doc/3.8/manual-single-page/>. [Zugriff am 29 5 2016].
- [4] „NeXt UI Cisco Release,“ [Online]. Available: <https://developer.cisco.com/site/next/>. [Zugriff am 26 März 2016].
- [5] „GitHub Repository NeXt UI,“ [Online]. Available: <https://github.com/opendaylight/next>. [Zugriff am 4 April 2016].

12 Abbildungen

Abbildung 1: Multicast-Strom im Netzwerk	4
Abbildung 2: Beispiel Default MDT	9
Abbildung 3: Beispiel Data MDT	9
Abbildung 4: User Story US01	11
Abbildung 5: Domain-Modell Topologie	17
Abbildung 6: Domainmodel Multicast	19
Abbildung 7: Grundlegende Architektur	21
Abbildung 8: Monolith Architektur	22
Abbildung 9: Unterteilte Services Architektur	23
Abbildung 10: Mockup Login Screen	25
Abbildung 11: Mockup Topologieübersicht	25
Abbildung 12: Mockup Multicast-Strom	26
Abbildung 13: show ip mroute Ausgabe	27
Abbildung 14: show ip mroute Ausgabe mit Regular Expressions	28
Abbildung 15: show ip mroute ohne VRF	29
Abbildung 16: show ip mroute mit VRF	29
Abbildung 17: Schichtenmodell Polling Service	37
Abbildung 18: Klasse LocationHandling mit allen enthaltenen Funktionen	38
Abbildung 19: CSV für Standortzuweisungen	38
Abbildung 20: Geräteverteilung innerhalb Standort	39
Abbildung 21: Klasse VrfHandling mit allen enthaltenen Funktionen	40
Abbildung 22: Ausgabe Befehl "show vrf include : "	40
Abbildung 23: Ausgabe Befehl show vrf id	40
Abbildung 24: Klasse Multicast Handling mit allen enthaltenen Funktionen	41
Abbildung 25: Sequenzdiagramm - Multicast-Ströme im Layer 3 Netzwerk sammeln	42
Abbildung 26: Sequenzdiagramm - Finden des Data MDT-Stroms im MPLS-Netzwerk	43
Abbildung 27: Sequenzdiagramm - Generieren des Default MDT	44
Abbildung 28: Algorithmus Korrelation Multicast-Strom mit Link	45
Abbildung 29: Klassen im Data Layer	46
Abbildung 30: Parser-Klassen	47
Abbildung 31: Labor Topologie	48
Abbildung 32: Datenbankschema	50
Abbildung 33: Ordnerstruktur Play Framework	54
Abbildung 34: Schichtenmodell Frontend	55
Abbildung 35: Ausschnitt aus Datei "routes"	57
Abbildung 36: Sequenzdiagramm - Laden der Webseite	59
Abbildung 37: Sequenzdiagramm – Multicast-Strom auswählen	60
Abbildung 38: Beispiel für eine Return-Liste bestehend aus zwei definierten Tupel	60
Abbildung 39 Unit Test für Polling Service	63
Abbildung 40: Unit Tests für MC Viewer	63
Abbildung 41: Default MDT über mehrere P Router	65
Abbildung 42: Data MDT im MPLS-Netzwerk	66
Abbildung 43: Multicast-Strom zu einem zusammengefassten Standort	67
Abbildung 44: Multicast-Strom zu einem aufgeklappten Standort	67
Abbildung 45: Soll-Ist-Vergleich	69
Abbildung 46: Total Aufwand pro Person	69

II. Anhänge

Anhang I:	Persönliche Berichte
Anhang II:	Projektplan
Anhang III:	Risikoanalyse
Anhang IV:	Installationsanleitung
Anhang V:	Systemtest
Anhang VI:	Eigenständigkeitserklärung