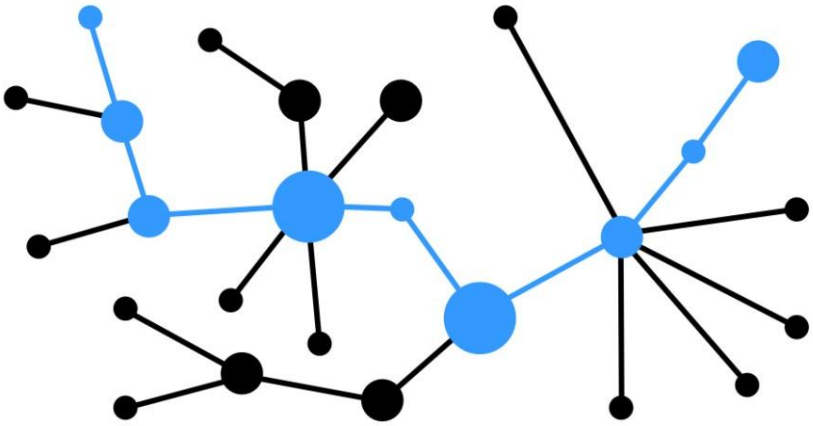
Network Path Visualization



Technischer Bericht der Semesterarbeit

Autoren:	André Ulrich
	Reto Gsell
Betreuer:	Prof. Beat Stettler
Projektpartner / Auftraggeber:	Rolf Schärer
Experte:	Prof. Beat Stettler
Abteilung:	Informatik
Themengebiet:	Internet-Technologien und –Anwendungen
Institut:	Institute for Networked Solutions (ins)
Zeitraum:	HS 2012 (17.09.2012 - 21.12.20012)

Änderungsgeschichte

Datum	Version	Änderung	Autor
04.12.2012	1.0	Initialisierung	André Ulrich
07.12.2012	1.1	Projektplan, Anforderungsanalyse und Domainmodel eingearbeitet.	André Ulrich
12.12.2012	1.2	Softwarearchitektur eingearbeitet und ausgearbeitet / Lab	André Ulrich
17.12.2012	1.2.5	Softwarearchitektur	Reto Gsell
18.12.2012	1.3	Webseite dokumentiert	Reto Gsell
19.12.2012	1.4	Sämtliche Punkte des Dokuments abgefüllt und erweitert.	Reto Gsell / André Ulrich
20.12.2012	1.5	Management-Summary und Abstract eingearbeitet / Korrekturlesen	André Ulrich
21.12.2012	2.0	Finalisiert / Ausgedruckt	André Ulrich / Reto Gsell

Aufgabenstellung

Einführung
In modernen Multipath-Datennetzen wird das Troubleshooting schwierig, sobald der Verkehr über mehrere Links parallel geführt wird. Es muss in mühsamer Handarbeit mit Konsolenbefehlen herausgefunden werden, welche Links für welche Kommunikationsbeziehung verwendet werden.

Ziel der Arbeit
Es soll eine Applikation als Webservice entwickelt werden, welche die Kommunikationswege innerhalb eines Netzwerkes aufzeigen kann.

Es soll ein Framework entstehen, welches erlaubt, Befehle auf Netzwerkgeräten (unabhängig vom Hersteller/Produkt) von einem Host im Netzwerk abzusetzen (auch via Konsole) und die Ergebnisse weiter zu verarbeiten. Dieses Framework soll verwendet werden, um eine Web-Applikation zu entwickeln, welche den Pfad einer Kommunikationsbeziehung visualisieren kann. Das Ganze soll automatisiert funktionieren - d.h. aufgrund des Resultates auf einem Netzwerkgerät wird ein Befehl auf dem nächsten Gerät ausgeführt.

Quelle: avt.hsr.ch

Abstract

Im Rahmen der Studienarbeit „Network Path Visualization“ (NPV) wurde eine, auf Java basierende Applikation mit Web-Front-End entwickelt. Anhand von Daten, welche von Netzwerkgeräten der Firma Cisco abgerufen werden, zeigt unsere Studienarbeit den Pfad zwischen einem beliebigen Gerät-A und Gerät-B im Netzwerk auf. Durch den dynamischen Aufbau der Software ist es möglich ohne grossen Mehraufwand eine Kompatibilität mit weiteren Herstellern zu gewährleisten.

Unsere Anwendung soll einem Troubleshooter das tägliche Leben erheblich erleichtern, indem es als unverzichtbares Instrument bei der Auffindung von Fehlern im Netzwerk den Pfad des Datenflusses darstellt. Monotone, wiederkehrende Arbeitsschritte beim Debugging gehören so der Vergangenheit an.

Die Logik zur Traversierung der einzelnen Knoten (Netzwerkgeräte) und zur Sammlung von Daten wurde als Library implementiert. Diese kann problemlos in Projekten von Drittanbietern zum Einsatz kommen.

Im Weiteren wurde eine Weboberfläche mit JavaServer Faces realisiert, um die Interaktion mit der Anwendung zu ermöglichen. Über diese Schnittstelle sollen gewünschte Informationen wie IP oder MAC-Adresse der beiden Endgeräte und allgemeine Parameter wie Login Informationen der Netzwerkgeräte definiert werden. Die Ergebnisse des Algorithmus werden sowohl in grafischer wie auch in textueller Form präsentiert.

Da die Anwendung von einem WebServer zur Verfügung gestellt wird, kann diese aus dem gesamten Netzwerk angesprochen und verwendet werden. Dies ermöglicht eine einfache Bedienung des Tools und die gewünschten Daten werden an einer zentralen Stelle bereitgestellt.

Management Summary

Ausgangslage

Das Institute for Networked Solutions (ins) der Hochschule für Technik Rapperswil (HSR) hat viele Aufträge aus der Privatwirtschaft. Im täglichen Arbeitsumfeld müssen Mitarbeiter oftmals Fehler suchen und beheben. Die Suche nach einem Fehler im Netzwerk gestaltet sich dank hochredundanter Netzwerke immer schwieriger und mühsamer. Unser Auftraggeber möchte dieser teils monotonen Arbeit ein Ende setzen. Die Applikation soll das Tagesgeschäft eines Troubleshooters erheblich erleichtern, indem es selbständig den Weg zwischen zwei Endgeräten aufzeigt und dabei Redundanzen mitbeachtet.

Die Anwendung soll lediglich zwei IP-Adressen, beziehungsweise zwei MAC-Adressen erhalten, um die weiteren Schritte zu bestimmen. Im Normalfall sind die Netzwerkgeräte mit einer Authentisierungsfunktion ausgestattet. Falls der Algorithmus keinen Zugriff auf die benötigten Ressourcen erhält, wird der Benutzer aufgefordert, die benötigte Information, wie Benutzername und Passwort nachzureichen.

Es wurde sehr viel Wert darauf gelegt, dass die Applikation analog zum Troubleshooter vorgeht. Sie soll sich also ebenfalls bei den einzelnen Geräten einloggen können und dieselben Kommandos absetzen, wie es der Troubleshooter machen würde. Die Ausgaben werden anschliessend interpretiert und auf Grund dieser Informationen entscheidet sich der Algorithmus für den nächsten Schritt.

Im Weiteren wurde der Wunsch geäussert, mehrere Hersteller und mehrere Softwareversionen der verschiedenen Hersteller zu unterstützen. Da diese teils unterschiedliche Kommandos voraussetzen und die Ausgaben der verschiedenen Geräte unterschiedlich formatiert sind und sogar unterschiedliche Informationen enthalten, muss die Interpretation extern beeinflusst werden können.

Die Ergebnisse der Anwendung sollen sowohl in textueller Form wie auch in grafischer Form ausgegeben werden.

Vorgehen, Technologien

Vorgängig wurde ein Redmine-Server bestellt, der es uns ermöglichte, stets über den aktuellen Stand des Projektes im Bild zu sein und unseren Aufwand auf die verschiedenen Teile des Projektes zu buchen.

In einer ersten Phase (Anforderungsanalyse) wurde analysiert, wie der Troubleshooter bei seiner Arbeit genau vorgeht. Dabei wurden alle Kommandos, die für die Problemstellung relevant sind, definiert und dokumentiert. Parallel dazu begann die Suche nach geeigneten Technologien und Libraries, die wir in unser Projekt einfliessen lassen konnten. Dazu gehören z.B. Technologien zur Authentisierung, zur grafischen Visualisierung oder auch die zugrundeliegende Programmiersprache an sich.

Zuerst wurde lediglich die Thematik des Layers 2 analysiert. Aus diesem Grund folgte rasch der Aufbau eines Netzwerks, das dank Etherchannel bereits eine gewisse Redundanz bot. Hier konnten grundlegende Techniken wie Telnet- und SSH-Zugriff auf verschiedenen Geräten evaluiert werden. Zu diesem Zeitpunkte stand bereits fest, dass die Applikation auf JAVA basieren soll und wir folgende Libraries einsetzen werden:

- SSH und Telnet (jcraft.com (jsch))
- SSH 2.0 (javatelnet.org)
- Testing (junit.org)
- XML-Dateien laden (jdom.org)

Sobald dieser Teil implementiert und funktionsfähig war, wurde die Applikation um die Problematik des Layers 3 und um die Funktionalität einer grafischen Interaktionsmöglichkeit Webseite erweitert. Auch hier galt es, zuerst das Testnetzwerk an die neuen Anforderungen anzupassen und nach passenden Technologien und Hilfsmitteln zu suchen. Mit JavaServer Faces (JSF) und JavaScript wurde schnell ein mächtiges Werkzeug gefunden um sowohl den Input, wie auch den Output zu gewährleisten.

- JSF components (primefaces.org)
- Visualisierung (jgraph.com (mxGraph))
- AJAX-Push (github.com/Atmosphere)

Parallel dazu wurden „WireFrames“ erstellt um auch gewährleisten zu können, dass in grafischer und interaktionstechnischer Sicht alle Wünsche unseres Auftraggebers erfüllt werden können. Schlussendlich galt es, all diese Komponenten zu implementieren, zusammenzufügen und natürlich auch zu dokumentieren.

Ergebnisse

Es gelang uns, alle geforderten Teile innert der geforderten Frist zu implementieren. Es können sowohl Pfade zwischen zwei MAC-Adressen, wie auch zwischen zwei IP-Adressen dargestellt werden. Alle benötigten Informationen werden von den Netzwerkgeräten abgefragt und interpretiert. Es werden Netzwerkgeräte unterstützt, die vom Hersteller Cisco sind und entweder IOS oder NEXUS als Betriebssystem installiert haben. Unterschieden werden können normale Links, Etherchannel-Links, redundante L3-Pfade und auch VRRP-Backup-Links erkannt werden.

Natürlich ist die Applikation noch nicht so weit ausgereift, dass sie in jedem Netzwerk zurechtkommt, doch sie kann einfach erweitert werden und wird mit minimalem Aufwand an andere Gerätschaften angepasst.

Die Webseite präsentiert sich mit modernem Layout und reagiert völlig unabhängig vom Algorithmus, der im Hintergrund ausgeführt wird. Des Weiteren kann sie sehr einfach installiert und betrieben werden.

Ausblick

Wie bereits beschrieben, kommt die Anwendung erst mit den im Auftrag beschriebenen Gegebenheiten zurecht. Sicherlich wäre es also interessant, weitere Hersteller, Betriebssystem-Versionen und Redundanztechniken zu unterstützen.

Technologien wie MPLS werden heute immer öfter eingesetzt und könnten natürlich von einer solchen Applikation ebenfalls analysiert werden. Diesbezüglich wäre ein Ausbau der Applikation ebenfalls sehr interessant.

Im Verlauf der Gespräche mit unserem Auftraggeber wurde uns bewusst, dass es nützlich wäre, wenn man aus der Applikation hinaus gleich auf eines der Geräte im Pfad verbinden könnte und auf diesem Kommandos absetzen könnte. Soweit also die Funktionalität eines Kommando-Zeilen-Interpreters, wie Putty auf der Webseite darstellen.

Ideen sind bestimmt genügend vorhanden, um hier gleich noch eine ganze Bachelorarbeit anzuhängen.

Inhaltsverzeichnis

1. Allgemein

1.1 Einführung

Dieses Dokument beinhaltet die gesamte Dokumentation der Semesterarbeit „Network Path Visualization“.

1.2 Gültigkeitsbereich

Das Dokument ist für die komplette Dauer des Projektes „Network Path Visualization“ gültig. Falls inhaltliche Änderungen am Dokument vorgenommen werden, muss dies dem Team mitgeteilt werden.

1.3 Abkürzungen

Der Einfachheit halber werden im Dokument teils Abkürzungen verwendet. Diese werden hier kurz erläutert.

Abkürzung	Bedeutung
NPV	Network Path Visualization
SA	Studienarbeit
JSF	JavaServer Faces
JS	JavaScript
Lab	Labor

1.4 Referenzen

Alle im Dokument verwendeten Texte sind selbst geschrieben und alle Grafiken wurden selber erstellt. Als Referenzen sind folgende Adressen zu nennen, die beim Aufbau des nötigen Knowhows behilflich waren.

- http://www.cisco.com
- http://stackoverflow.com
- http://www.primefaces.org/
- https://jcraft.com
- https://javatelnet.org
- https://junit.org
- https://jdom.org
- https://jgraph.com
- https://github.com/Atmosphere
- http://www.vogella.com
- http://www.regexplanet.com

2. Projektplan

2.1 Einführung

2.1.1 Zweck

Dieses Dokument enthält die Projektplanung der Studienarbeit „Network Path Visualization“, welche die Entwicklung einer webbasierten Software zur Darstellung von Netzwerkpfaden zum Ziel hat. Es sind sämtliche Angaben wie Arbeitspakete – in grober Ausführung – inklusive Meilensteine definiert. Eine Zeitplanung, basierend auf den Arbeitspaketen, die Beschreibung der verschiedenen Iterationsschritte und zusätzlich Angaben über die Qualitätssicherung und das Risikomanagement sind ebenfalls enthalten. Der Projektplan dient als Basis für die weitere Planung der Studienarbeit und ist der Grundstein für diese Studienarbeit.

2.1.2 Zweck und Ziel

Das Hauptaugenmerk dieses Projektes liegt auf einer erfolgreichen Semesterarbeit und der Umsetzung von gesammelten Erfahrungen. Das Ziel ist es, den Anforderungen entsprechend eine Lösung zu präsentieren.
Weiter interessiert uns die Entwicklung im Netzwerkbereich sehr und stellt sicher einen grossen Mehrwert in Bezug auf Erfahrungswerte dar.
Als Endprodukt wird ein funktionierendes Tool für die Unterstützung des Troubleshooters angestrebt. Es wird jedoch keinesfalls komplett fertig und einsetzbar sein, sondern einen Grundstein legen, der alle technischen Probleme löst und implementiert.

2.1.3 Lieferumfang

Die Semesterarbeit besteht aus einem Java Programm, das als Server fungiert und einer Webapplikation, die für die Darstellung der gesammelten Daten zuständig ist. Ausgeliefert wird dies einerseits in zwei Projekten, die für die Weiterentwicklung verwendet werden können. Zum anderen wird die Webapplikation mit integrierter Library als verwendbares Webprojekt ausgeliefert, das von allen kompatiblen Webservern verwendet werden kann,

2.1.4 Erweiterungen

Die Applikation kann beliebig erweitert werden, indem weitere Redundanztechnologien oder weitere Geräteunterstützungen hinzugefügt werden. Zusätzlich sind Technologien wie MPLS eine mögliche Erweiterung.

2.1.5 Annahmen und Einschränkungen

Für die Projektmitarbeiter wird mit einer Sollarbeitszeit von 16 Stunden pro Woche gerechnet. Bei insgesamt 14 Wochen Arbeitszeit ergibt dies (16*14 =) 224 Stunden pro Mitarbeiter. Da am Projekt zwei Mitglieder beteiligt sind, ergibt das einen totalen Aufwand von **448 Stunden**. Dieser Aufwand soll vollumfänglich durch die Semesterarbeit abgedeckt werden.

2.2 Projektorganisation

Das Team besteht aus zwei gleichgestellten Mitgliedern. Die Betreuung wird durch Herr Rolf Schärer wahrgenommen. Er steht dem Projektteam bei Fragen zur Seite.

2.2.1 Organisationsstruktur

Name	e-Mail	Aufgaben und Verantwortungen
André Ulrich	a1ulrich@hsr.ch	Entwicklung, Java und Webservice, Dokumentation
Reto Gsell	reto.gsell@hsr.ch	Entwicklung, Java und Webservice, Dokumentation

2.2.2 Externe Schnittstellen

Name	e-Mail	Rolle
Rolf Schärer	rolf.schaerer@ins.hsr.ch	Projektpartner
Beat Stettler	beat.stettler@ins.hsr.ch	Projektbetreuer / Examinator

2.3 Management Abläufe

2.3.1 Zeitliche Planung

Die zeitliche Planung, die Phasen, sowie auch die Arbeitszeiterfassung sind in Redmine genauer abgebildet. Hier eine knappe Übersicht:

Semesterwoche	Datum	Termine
-	17.09.2012	Semesterbeginn
Inception		
SW 01 / KW 38	17.02.2012 - 23.09.2012	25.02.2012 14:30 Uhr – Kick-Off-Meeting 26.09.2012 15:00 Uhr – Labor einrichten
SW 02 / KW 39	24.09.2012 -30.09.2012	
Elaboration		
SW 03 / KW 40	01.10.2012 - 07.10.2012	03.10.2012 13:00 Uhr – Troubleshooting 03.10.2012 14:00 Uhr – Meeting
SW 04 / KW 41	08.10.2012 - 14.10.2012	
SW 05 / KW 42	15.10.2012 - 21.10.2012	
Milestone 1 – Prototyp		
Construction 1		
SW 06 / KW 43	22.10.2012 - 28.10.2012	24.10.2012 15:00 Uhr – Meeting
SW 07 / KW 44	29.10.2012 - 04.10.2012	
Milestone 2 – Layer 2-Funkionalität		
Construction 2		
SW 08 / KW 45	05.11.2012 - 11.11.2012	07.11.2012 15:00 Uhr – Meeting
SW 09 / KW 46	12.11.2012 - 18.11.2012	
SW 10 / KW 47	19.11.2012 - 25.11.2012	
Milestone 3 – Layer 3-Funktionalität und Webseite		
Construction 3		
SW 11 / KW 48	26.11.2012 - 02.12.2012	28.11.2012 15:00 Uhr – Meeting
SW 12 / KW 49	03.12.2012 - 09.12.2012	
Milestone 4 – Optimierungen und Erweiterungen		
Transition		
SW 13 / KW 50	10.12.2012 - 16.12.2012	12.12.2012 15:00 Uhr – Meeting 17.12.12 – Abgabe Kurzfassung / Poster 21.12.12 – Abgabe der gesamten Arbeit
SW 14 / KW 51	17.12.2012 - 23.12.2012	
Milestone 5 – Projektabgabe (Final Release)		
-	21.12.2012	Semesterende

2.3.2 Meilensteine

Untenstehend eine Beschreibung der einzelnen Meilensteine:

#	Bezeichnung	Beschreibung	Meeting-Zeitpunkt

1	Prototyp	In einem Prototypen sollen folgende Funktionalitäten realisiert sein:
		• SSH und Telnet-Verbindung
		• Testpfad für Visualisierung
		• Parser für folgende commands
		  ○ show mac address-table
		  ○ show cdp neighbors
		  ○ show ip route
		  ○ show version
			24.10.2012 15:00 Uhr

2	Layer 2-Funkionalität	In diesem Stadium soll die Applikation im Stande sein, den Weg zwischen „GerätA“ und „GerätB“ zu erkennen, in einer Objektstruktur abzulegen. Dies jedoch erst in einem Layer2-Netzwerk	07.11.2012 15:00 Uhr
3	Layer 3-Funktionalität und Webseite	Im dritten Meilenstein soll die Applikation bereits in einem Layer3-Netzwerk zurechtkommen. Die Eingabe der Parameter und die Ausgabe des evaluierten Pfades erfolgt über eine WebSeite.	28.11.2012 15:00 Uhr
4	Optimierungen und Erweiterungen	Die Applikation soll nun redundante Pfade auf Layer 2 und Layer 3 aufzeigen können. Mögliche Erweiterungen sind auch: • Stacking • VSS • FabricPath Des Weiteren kann die Webseite um grafische Ansichten zur Visualisierung der Ergebnisse erweitert werden.	12.12.2012 15:00 Uhr
5	Projektabgabe (Final Release)	Die Applikation befindet sich in einem abgeschlossenen Zustand. Sämtliche Dokumente sind fertig erstellt, aktuell und wurden abgegeben.	Ende

2.3.3 Besprechungen

2.3.3.1 Mit Betreuer

Besprechungen mit beiden Betreuern sind jeweils im Abschnitt „2.3.1 Zeitliche Planung“ zu finden. Sitzungsprotokolle werden im Verzeichnis „Protokolle“ festgehalten.

Weitere Besprechungen mit Herrn Rolf Schärer werden sporadisch abgehalten. Sofern nicht nur technische Dinge besprochen wurden, finden sich zu diesen Meetings ebenfalls im Verzeichnis „Protokolle“ die Sitzungsprotokolle.

Vorwiegend:

Meeting	Nach
Teilnehmer	Prof. Beat Stettler, Rolf Schärer , André Ulrich, Reto Gsell
Ort	HSR Rapperswil (oder nach Absprache)

Die Protokolle zu den Besprechungen sind im Anhang zu finden. Sofern ausserordentliche Besprechungen durchgeführt wurden, so ist dies ebenfalls den Protokollen zu entnehmen.

2.3.3.2 Ohne Betreuer

Vorgesehen ist ein wöchentliches Meeting. Die Meetings dienen dazu, erledigte Arbeiten, Pendenzen und allfällige Probleme zu besprechen. Während den Besprechungen wird kein Protokoll geführt.

Meeting	Montag 13:10 ca 1 Stunde (kann nach Absprache geändert werden)
Teilnehmer	André Ulrich, Reto Gsell
Ort	HSR Rapperswil (oder nach Absprache)

Hierzu wird kein Protokoll erstellt.

2.4 Risikomanagement

2.4.1 Risiken

Nachfolgend ist ein Auszug der Analyse der technischen Risiken zu sehen. Diese Tabelle enthält zusätzlich die Massnahmen, die eingeleitet werden, falls der beschriebene Fall eintreten sollte.

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Zugriff auf die CLI mit Java	Kein Teammitglied hat bis anhin Kenntnisse mit einer Java Library, die eine SSH oder Telnet Verbindung herstellt	50	70%	35	Etablierte Technologien verwenden (Libraries, die weit verbreitet sind)	Literatur oder entsprechende Dozenten zu Hilfe ziehen.
R2	Probleme beim Implementieren des Webservices	Programmierung eines Webservice nur grob bekannt und es sind nur Grundkenntnisse in der Webprogrammierung vorhanden.	40	70%	28	In dem Modul Internettechnologien die Unterlagen zu Webservices durcharbeiten.	Problem im Team besprechen und eventuell externe Hilfe hinzuziehen.
R3	Geeigneten Algorithmus für die Pfadfindung finden.	Die von Cisco verwendeten Technologien um die Links herzustellen sind den Teammitgliedern noch nicht alle bekannt.	40	20%	8	In die Literatur der einzelnen Technologien einarbeiten.	eventuell externe Hilfe zuziehen (Rolf Schärer)
R4	Krankheit oder Unfall eines Teammitgliedes	Arbeit wird aufgrund eines Arbeitsausfalles hinausgeschoben	40	5%	2	Reserven einplanen	Arbeiten nachholen
R5	Probleme mit dem Parsen der Antworten der CLI	Das Parsing gestaltet sich schwierig oder fast unmöglich	30	20%	6	Einarbeitung in Regex und Parser Objekte	Best mögliche Umsetzung oder einen anderen Weg zur Informationsbeschaffung ausarbeiten.
R6	Cisco Technologien können nicht über die CLI ausgewertet werden	Jeweilige Technologie nicht nutzbar für die Pfadfindung.	40	10%	4	Abklären was über die CLI möglich ist und was nicht	Jeweilige Technologie kann nicht genutzt werden.
Summe			**240**		**83**		

2.5 Arbeitspakete

Um die Risiken besser in den Griff zu bekommen wurden kleine Arbeitspakete erstellt.

Die Arbeitspakete sind zwecks Zeiterfassung der Arbeiten im Projektplanungstool „Redmine“ definiert. Dort sind alle Arbeitspakete ersichtlich. Näheres zum Zugriff auf diese Daten ist im Abschnitt „Projektmanagement“ auf der Folgeseite zu finden.

Hier findet sich nun ein Auszug aller Arbeitsschritte aus dem Redmine in Form eines Gantt-Diagramms:

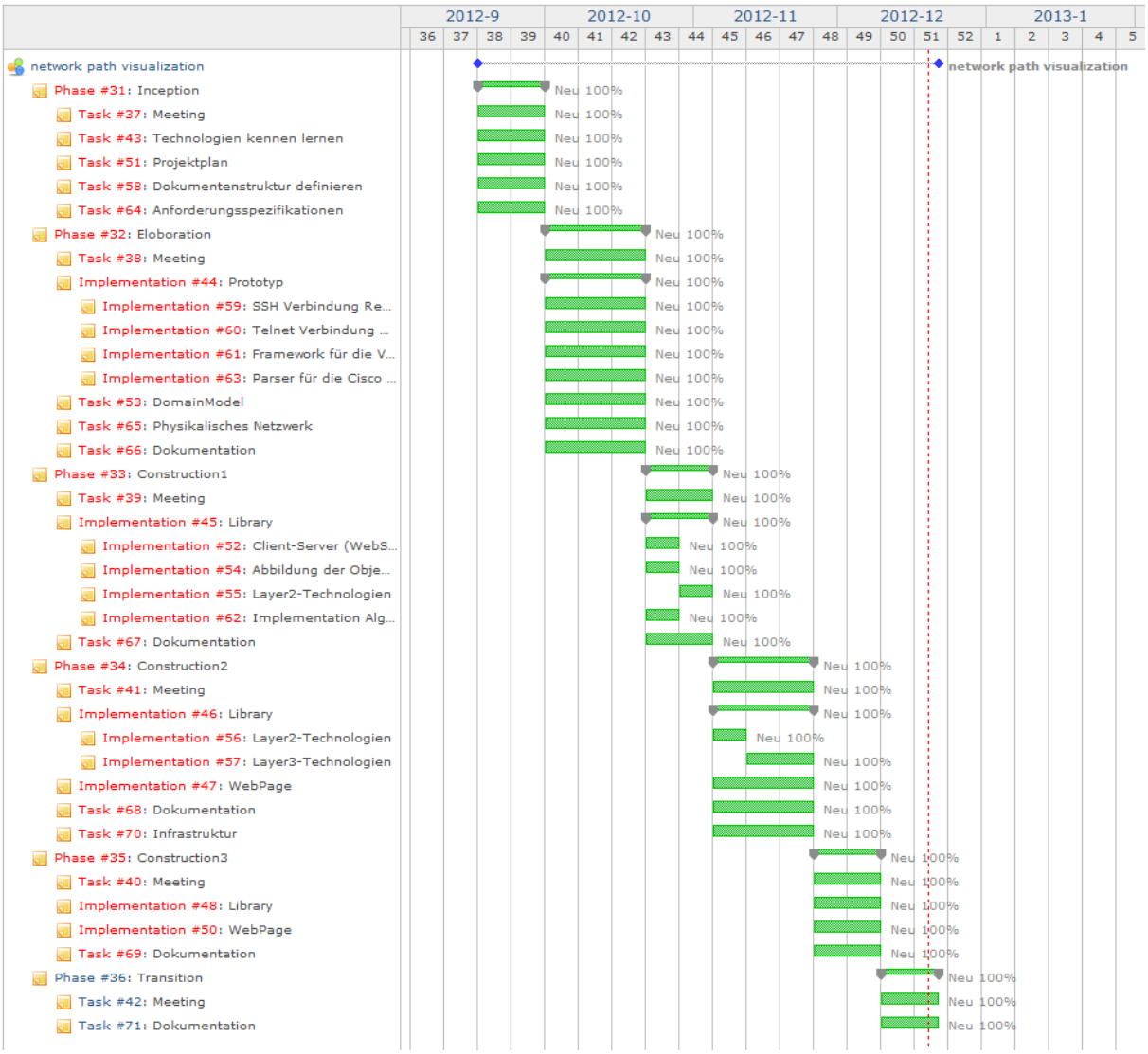


Abbildung 1: Arbeitspakete

2.6 Infrastruktur

2.6.1 Planungsumgebung

2.6.1.1 Redmine

Die Projektmanagementsoftware ist auf einer virtuellen Maschine, welche von der HSR zur Verfügung gestellt wird. Es ist ein Linux Ubuntu System, auf welchem das Projektmanagementtool „Redmine“ installiert ist. In Redmine sind die Arbeitspakete verwaltet, die Zeitplanung des Projektes dargestellt und die Aufwände der einzelnen Projektmitglieder dargestellt.

Folgend die Spezifikation der Maschine:

Betriebssystem	Linux Ubuntu 10.04 LTS
Lebenszyklus virtuelle Maschine	01.09.2012
Servername	sinv-56056.edu.hsr.ch
CPU	1x 2.27 GHz
RAM	1 GB
HDD	15 GB
LAN Adapter	1 VMWare Flexible NIC
Video Adapter	VMWare SVGA II

2.6.1.2 Dokumentenverwaltung

Für die Verwaltung der Projektdokumente wird Dropbox eingesetzt. Dropbox ermöglicht eine automatische Versionisierung, was im Zusammenhang mit der Projektplanung einen enormen Mehrwert darstellt.

➔ https://www.dropbox.com

2.6.2 Entwicklungsumgebung

2.6.2.1 Programmierinstrumente

Die Entwicklungsumgebung während des Projektes besteht aus den persönlichen Laptops der Projektmitglieder. Festzuhalten ist jedoch, dass die Entwicklung von den folgenden Eckpunkten gezeichnet ist.

- IDE ist Eclipse EE
- Java Version 1.7
- Versionskontrolle via Git

2.6.2.2 GIT (GitHub)

Für die Versionskontrolle wird das Git System von github verwendet. Dadurch wird die Versionisierung und Sicherung unseres Programmcodes sichergestellt.

Bei github wurde ein für verifizierte Studenten gratis verfügbares „Private Repo“ eingerichtet. Dieses ist für eingeladene Mitglieder unter folgender URL zu finden:

➔ https://github.com/retogsell/NetworkPathVisualization

2.6.2.3 Labor

Dem Team steht ein Labor zur Verfügung, welches die benötigten Geräte (Cisco Router und Cisco Switches(L2 / L3)) mit dem entsprechenden Equipment enthält. Das Team wird durch den Betreuer eingewiesen, kann dann aber selbständig Änderungen an der Konfiguration vornehmen.

Die Dokumentation dieser Testnetzwerke, die damit aufgebaut werden, sind im Anhang zu finden.

2.7 Qualitätsmassnahmen

2.7.1 Dokumentation

Die Dokumentation wird parallel mit der Entwicklung des Produkts vorangetrieben. In der Schlussphase wird dies aber natürlich überhand gewinnen.

Es ist wichtig, dass sich jedes Teammitglied auf die Aktualität der Dokumente verlassen kann. Aus diesem Grund sind Neuerungen durch die Teammitglieder stets zu aktualisieren. Dokument-Reviews sind von allen Teammitgliedern insbesondere vor Meetings durchzuführen.

2.7.2 Projektmanagement

Um eine hohe Qualität des Projektmanagements zu erreichen, setzen wir auf Redmine, wo die Arbeitspakete definiert werden können.

Damit auch Betreuer und Aussenstehende einen Einblick in die aktuelle Projektplanung erhalten, wurde ein Account eingerichtet, der lesenden Zugriff auf die Daten ermöglicht.

Zugang zum Redmine:

- Link: http://sinv-56056.edu.hsr.ch/redmine
- Benutzername: SA-Viewer
- Passwort: SA-Viewer2012

2.7.3 Entwicklung

Da Git eine dezentrale Versionisierung verfolgt, ist ein Backup des Codes gewährleistet, da jedes der Projektmitglieder eine lokale Kopie des Codes besitzt. Es werden nur Quellcodedateien, die lauffähig und getestet sind, eingecheckt („stable trunk“). Somit wird allen Projektmitgliedern der Zugriff auf den aktuellen und funktionsfähigen Code gewährt. Fälschliche Manipulationen des Quellcodes können durch ältere Versionen rückgängig gemacht werden.

2.7.3.1 Code Reviews

Teammitglieder können, für von ihnen entwickelte Codebausteine, Code Reviews beantragen. Diese Reviews werden dann, sofern der Antrag gutgeheissen wurde, jeweils am Montag anlässlich des wöchentlichen Teammeetings durchgeführt. Es soll infolge knapper Terminplanung darauf geachtet werden, dass Reviews primär für kritische Softwarekomponenten beantragt werden. Weiter ist eine gute Vorbereitung des Antragstellers unerlässlich, um einen speditiven Ablauf gewährleisten zu können.

2.7.3.2 Code Style Guidelines

Alle Entwickler haben sich beim Schreiben von Code an die im Styleguide für Sourcecode definierten Richtlinien zu halten.
Beispielcode:

```
/**
 * This class acts as a example for javadoc - documentations
 *
 * @author John Doe
 */
public class CodeGuidelines {

	//Variables & Constants
	private final int CONSTANT = 1;
	private int variableInCamelcase = 1;
	public static int STATIC_VARIABLE = 1;
	public String publicVarInCamelcase = "hello world";

	//Constructors

	/**
	 * This is the constructor of the CodeGuildelines class
	 *
	 * @author John Doe
	 */
	public CodeGuidelines(){
		//do something
	}

	//Methods

	/**
	 * This method does something.
	 *
	 * @author John Doe
	 * @param a String to do something.
	 * @param b Integer with is needed for something.
	 * @return Double from the operation.
	 */
	public double someMethod(String a, int b){
		return 1d;
	}
}
```

- Der ganze Code und der Kommentar wird ausschliesslich in englischer Sprache geschrieben.
- Jede Klasse enthält einen Kommentar, mit den Elementen aus dem Beispiel.
- Jede wichtige und nicht triviale Methode enthält einen Kommentar wie im Beispiel beschrieben.
- Abgeleitete Klassen enthalten den Namen der Vaterklasse am Ende der Bezeichnung.
- Paketnamen werden in Kleinbuchstaben geschrieben
- Zusammengesetzte Wörter in Klassen, Methoden und Variablen werden in CamelCase geschrieben.
- Static Variablen und Konstanten werden komplett in Grossbuchstaben geschrieben. Zusammengesetzte Wörter werden hier durch einen Unterstrich voneinander getrennt.

2.7.4 Testen

2.7.4.1 Unit Tests

Neben den von Hand gemachten Test, werden alle automatisierten Tests mit JUnit realisiert. Die Unit Tests werden lokal programmiert, dann lokal getestet. Sie sind sowohl nach jeder Änderung einer betroffenen Klasse zu testen und vor jedem Abgleich des Source-Codes mit den anderen Mitarbeitern.

2.7.4.2 Systemtests

Zum Zeitpunkt des Meilensteins 3, 4 und 5 werden die Systemtests durchgeführt um sicherzustellen, dass die bis dahin implementierten Funktionen erfolgreich im System realisiert wurden

3. Anforderungsanalyse

3.1 Einführung

3.1.1 Übersicht

Dieser Abschnitt soll die Anwendungsfälle(UseCases), die an die Software gestellt werden, aufzeigen und somit die Anforderungsspezifikation beinhalten. Die UseCases bündeln alle möglichen Szenarien, die beim Bedienen der Software eintreten können.

3.2 Allgemeine Beschreibung

3.2.1 Produktefunktion

Das Produkt wird in erster Line bei Troubleshooting-Einsätzen Verwendung finden. Da in der heutigen Zeit immer mehr mit Redundanz im Netzwerk gearbeitet wird, ist es oftmals schwierig nachzuvollziehen, welchen Pfad ein Paket von Rechner A nach Rechner B wählt. Es ist jedoch in gewissen Fällen wichtig, diesen Pfad zu finden und genau hier unterstützt das Produkt den Troubleshooter. Es wird auch darauf geachtet, dass das Programm lediglich die Arbeitsabläufe, die bisher mühsam in eigener Arbeit gemacht werden mussten, automatisiert.

In zweiter Linie kann es auch dafür eingesetzt werden, die Konfiguration des Netzwerks zu testen und zu überprüfen, ob Pfade so umgesetzt wurden, wie es in der Planung vorgesehen war.

3.2.2 Benutzer Charakteristik

Das Produkt soll natürlich einfach zu bedienen sein, doch wird ein Benutzer ohne netzwerktechnischen Hintergrund mit dem Output ohnehin nicht viel anfangen können. Darum besteht die Zielgruppe ganz klar aus Netzwerktechnikern, die mit dem Aufbau von redundanten Netzwerken vertraut sind.

3.2.3 Einschränkungen

Die Applikation soll lediglich den Pfad von Rechner A nach Rechner B in einem Netzwerk aufzeigen können, nicht jedoch Aufzeichnen wie das gesamte Netzwerk aussieht. Dafür werden weiterhin Programme wie WhatsUpGold eingesetzt.

Die Applikation bezieht sich lediglich auf Cisco-Geräte. Sobald ein Gerät im Pfad nicht von Cisco stammt, kann der Pfad nicht visualisiert werden. Des Weiteren werden lediglich Betriebssysteme (Cisco-OSs) unterstützt, die im Lab vorhanden sind. Dies sind cisco IOS und cisco Nexus. Eine Erweiterung ist relativ einfach machbar, da der eigentliche Traversierungsalgorithmus geräteunabhängig funktioniert. Doch dies ist nicht Teil dieser Semesterarbeit.

Im Rahmen der SA werden lediglich diese Redundanz-Technologien unterstützt, die mit Herrn Rolf Schärer zusammen im Labor aufgebaut werden. Weitere Technologien können aber unabhängig von dieser SA relativ leicht implementiert werden.

Da die verschiedenen Netzwerkgeräte über ihre IP-Adresse, oder Mac-Adresse (falls keine IP vorhanden ist) identifiziert werden muss also immer daran gedacht werden, dass bei einem manuellen Eingriff die Management-IP des Gerätes verwendet wird und keine andere. Konkret wird also bei der Angabe des Seed-Devices davon ausgegangen, dass die Management-IP im Manufactor.xml-File eingetragen wurde.

Die Applikation ist nach der SA nicht in einem komplett verkaufsbereiten Zustand. Es werden im Rahmen der SA alle Aufgaben implementiert, auf welche die Applikation angewiesen ist: z.B. Telnet-, SSH-Zugriff, asynchrone Interaktionsmöglichkeiten über die Webseite, usw. Zusätzlich wird ein Pfad mit all den im Lab gegebenen Technologien aufgebaut und dargestellt. Somit bietet die Applikation nach der SA eine ausgezeichnete Grundlage für die Weiterentwicklung und Verfeinerung der Anwendung.

3.2.4 Annahmen

Auf dem zu prüfenden Pfad werden nur Cisco Produkte (Router und Switches) eingesetzt.
Auf dem Client, auf dem die Webapplikation ausgeführt wird, ist eine lauffähige und aktuelle JRE installiert.

3.2.5 Abhängigkeiten

Der Parser arbeitet mit Daten, die zu diesem Zeitpunkt von Cisco-Geräten ausgegeben werden. Falls Cisco bei einem Update ihres OS die Art und Weise ihrer Ausgabe der verschiedenen Kommandos ändert, ist die Applikation nicht mehr im Stande, die Ausgaben zu interpretieren und muss angepasst werden.

3.3 Use Cases

Da die Applikation aus Usersicht lediglich die Visualisierung des Pfades darstellt und der Nutzer keine weitere Interaktionsmöglichkeit hat, sind nur sehr wenige Usecases aufzuführen.

3.3.1 Use Case Diagramm

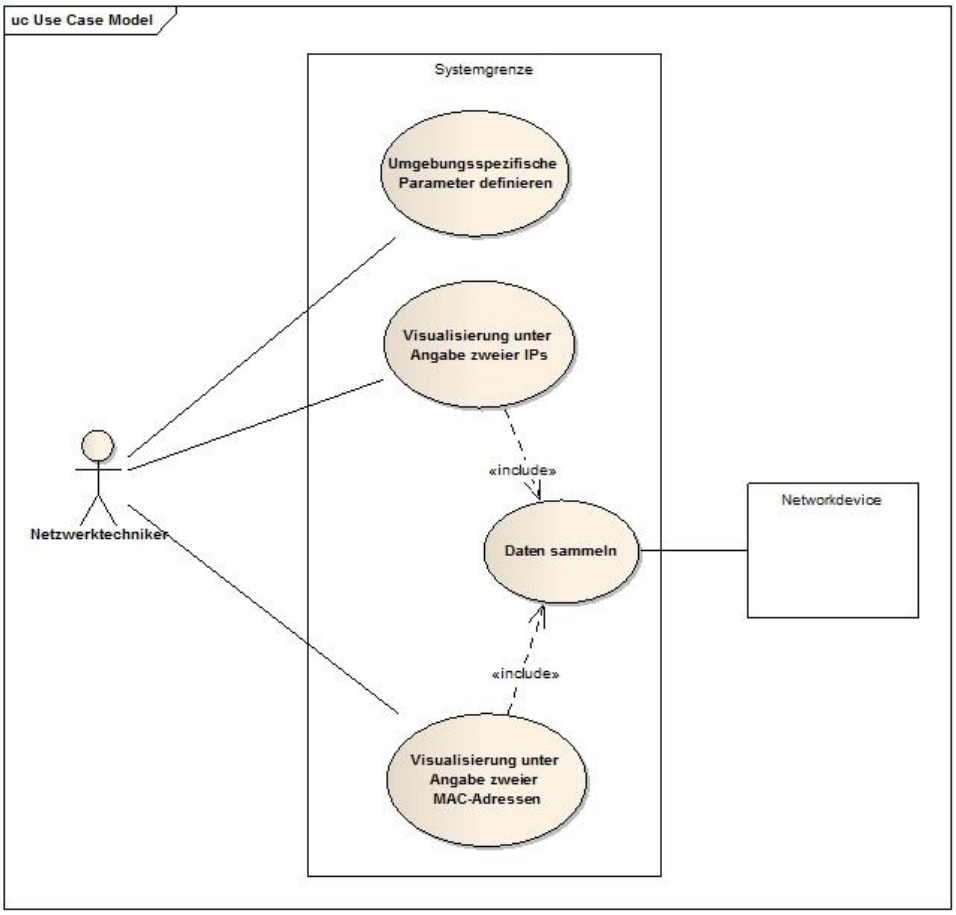


Abbildung 2: UseCase-Diagramm

3.3.2 Aktoren & Stakeholder

- Netzwerktechniker

3.3.3 Beschreibungen (Fully Dressed)

3.3.3.1 UC1: Umgebungsspezifische Parameter definieren

UC1: Bestellung erfassen	
Primary Actor	Netzwerktechniker
Description	Der Netzwerktechniker startet die Applikation in seinem Browser und definiert umgebungsspezifische Parameter, wie Usernamen und Passwörter.
Stakeholder and Interests	• Netzwerktechniker: Möchte die Applikation an die Gegebenheiten des jeweiligen Netzwerks anpassen.
Preconditions	1. keine
Postconditions	1. Die Applikation ist jetzt einfacher zu bedienen, weil gewisse notwendige Parameter bereits bekannt sind.
Main Success Scenario	1. Der Netzwerktechniker startet den Browser 2. Der Netzwerktechniker öffnet die Applikations-Startseite 3. Er wählt die Schaltfläche, die ihn zu den Einstellungen führt. 4. Er erfasst alle benötigten Parameter 5. Er speichert die Eingaben
Exceptions	5.a Falls gewisse Eingaben keinen Sinn ergeben, wird eine entsprechende Fehlermeldung ausgegeben.
Special Requirements	• Einen Internet-Browser
Technology and Data Variations List	keine
Frequency of Occurrence	Beim Auftreten eines Fehlers im Netzwerk oder bei allfälliger Überprüfung des Netzwerks.

3.3.3.2 UC2: Visualisierung unter Angabe zweier IPs

UC1: Bestellung erfassen	
Primary Actor	Netzwerktechniker
Description	Der Netzwerktechniker startet die Applikation in seinem Browser und gibt die IP-Adressen der beiden relevanten Geräte ein. Das System errechnet den Pfad zwischen diesen beiden Geräten und stellt diesen in einer Grafik dar.
Stakeholder and Interests	• Netzwerktechniker: Möchte den Pfad zwischen Gerät A und Gerät B visualisiert sehen.
Preconditions	1. Alle Switches und Router müssen via Telnet, oder SSH erreichbar sein. 2. Beide IPs müssen erreichbar sein. 3. Die involvierten Geräte müssen mit einem unterstützten Cisco-OS ausgestattet sein. 4. Die verwendeten Technologien zur Redundanz müssen von der Applikation unterstützt werden. 5. Alle angesprochenen Geräte (auch Switches) müssen eine IP zugewiesen haben, über welche man Zugriff auf das Gerät erhalten kann.
Postconditions	1. Die Grafik lässt erkennen, welche Geräte ein Paket zwischen Rechner A und Rechner B durchströmt.
Main Success Scenario	1. Der Netzwerktechniker startet den Browser 2. Der Netzwerktechniker öffnet die Applikations-Startseite 3. Er gibt die beiden gewünschten IP-Adressen ein 4. Er klickt auf „senden“ 5. Es erscheint eine Ansicht, die das Laden symbolisiert 6. Die Grafik, die alle involvierten Netzwerkgeräte beinhaltet, wird angezeigt.
Exceptions	5.a Falls die IP-Adressen nicht erreichbar oder fehlerhaft sind, wird ein Fehler angezeigt. 5.b Falls bei der Traversierung ein Fehler auftritt, wird ein Fehler angezeigt
Special Requirements	• IP-Connectivity zum Applikationsrechner • Einen Internet-Browser • Installierte Software zur Darstellung von Java-Applets im Browser
Technology and Data Variations List	keine
Frequency of Occurrence	Beim Auftreten eines Fehlers im Netzwerk oder bei allfälliger Überprüfung des Netzwerks.

3.3.3.3 UC3: Visualisierung unter Angabe zweier MAC-Adressen

UC1: Bestellung erfassen	
Primary Actor	Netzwerktechniker
Description	Der Netzwerktechniker startet die Applikation in seinem Browser und gibt die MAC-Adressen der beiden relevanten Geräte ein. Das System errechnet den Pfad zwischen diesen beiden Geräten und stellt diesen in einer Grafik dar.
Stakeholder and Interests	• Netzwerktechniker: Möchte den Pfad zwischen Gerät A und Gerät B visualisiert sehen.
Preconditions	1. Alle Switches und Router müssen via Telnet, oder SSH erreichbar sein. 2. Beide IPs müssen erreichbar sein. 3. Die involvierten Geräte müssen mit einem unterstützten Cisco-OS ausgestattet sein. 4. Die verwendeten Technologien zur Redundanz müssen von der Applikation unterstützt werden. 5. Alle angesprochenen Geräte (auch Switches) müssen eine IP zugewiesen haben, über welche man Zugriff auf das Gerät erhalten kann.
Postconditions	1. Die Grafik lässt erkennen, welche Geräte ein Paket zwischen Rechner A und Rechner B durchströmt.
Main Success Scenario	1. Der Netzwerktechniker startet den Browser 2. Der Netzwerktechniker öffnet die Applikations-Startseite 3. Er gibt die beiden gewünschten MAC-Adressen ein 4. Er klickt auf „senden“ 5. Es erscheint eine Ansicht, die das Laden symbolisiert 6. Die Grafik, die alle involvierten Netzwerkgeräte beinhaltet, wird angezeigt.
Exceptions	5.a Falls die MAC-Adressen in den AddressTables der Geräte nicht erreichbar sind oder fehlerhaft eingegeben wurden, wird ein Fehler angezeigt. 5.b Falls bei der Traversierung ein Fehler auftritt, wird ein Fehler angezeigt
Special Requirements	• IP-Connectivity zum Applikationsrechner • Einen Internet-Browser • Installierte Software zur Darstellung von Java-Applets im Browser
Technology and Data Variations List	keine
Frequency of Occurrence	Beim Auftreten eines Fehlers im Netzwerk oder bei allfälliger Überprüfung des Netzwerks.

3.4 Weitere Anforderungen

3.4.1 Qualitätsmerkmale

3.4.1.1 Änderbarkeit

Das Projekt kann beliebig erweitert werden und ist keinesfalls mit dem Ende der im Projektplan definierten Ziele abgeschlossen. Das Ziel ist, dass die Software beliebig erweitert werden kann und, dass dies auch von externen Personen übernommen werden könnte. Durch die Aufteilung der Arbeit in eine Library, welche die Logik beinhaltet und ein Servlet, das als User-Schnittstelle dient, kann leicht eine der Komponenten ausgetauscht oder erweitert werden.

3.4.1.2 Benutzbarkeit

Die benötigten Useraktionen sollen sich auf ein Minimum begrenzen. Die Applikation soll vollkommen selbständig arbeiten können und von überall im Netzwerk verwendbar sein.

3.4.1.3 Effizienz

Durch den geforderten Ansatz, die einzelnen Geräte des Pfades anzusprechen und die entsprechenden Daten auszulesen, ist die Applikation zeitlich abhängig von der Geschwindigkeit der einzelnen Geräte und von der Anzahl der Geräte im Pfad. Zusätzliche Zeitverzögerungen sollten aber nicht auftreten, weil die Auswertung und die Weiterverarbeitung der Daten auf dem Applikationsserver sehr schnell abgehandelt werden.

3.4.1.4 Funktionalität

3.4.1.4.1 Interoperabilität

Die Verwendung von Java und einem Webinterface für die Implementation deckt bereits alle von uns angestrebten Architekturen ab. Dies bezieht sich sowohl auf den Server, wie auch auf die Clients, welche die Applikation verwenden.

3.4.1.4.2 Sicherheit

Login Daten und ähnlich sensitive Daten werden lediglich für den Zeitraum des entsprechenden Requests gespeichert und danach verworfen. Da die Applikation also keinerlei sensitive Daten speichert, muss hier kein erhöhter Wert auf Sicherheit gelegt werden.

Durch die Verwendbarkeit von SSH werden bei der Authentifizierung keine Klartextdaten ausgetauscht, womit eine sichere Möglichkeit zur Authentifizierung geboten wird.

3.4.1.5 Übertragbarkeit

- Clientseitig ist keine Installation der Applikation notwendig.
- Serverseitig muss ein WebServer, wie Tomcat verfügbar sein, der mit Servlets umgehen kann. Somit kann die Applikation auf allen verbreiteten Betriebssystemen ohne weiteren Aufwand betrieben werden.

3.4.1.6 Zuverlässigkeit

Da keine Daten gespeichert werden, ist bei einem Ausfall der Applikation ohnehin kein Verlust von Informationen gegeben. Selbstverständlich soll die Applikation aber zuverlässig funktionieren und unterstützen.

3.4.2 Schnittstellen

Folgende Schnittstellen sind in der Anwendung vorhanden:

- Benutzerschnittstellen
  - Webapplikation
    - Zur Interaktion zwischen Benutzer und Anwendung
- Softwareschnittstelle
  - Telnet-/SSH-Library
    - Zur Kommunikation zwischen Anwendung und den verschiedenen Netzwerkgeräten, die angefragt werden.

4. Domainanalyse

4.1 Einführung

4.1.1 Übersicht

Der folgende Abschnitt beschreibt die Analyse der Domain für das Projekt NPV.

4.2 Domain Modell (Library)

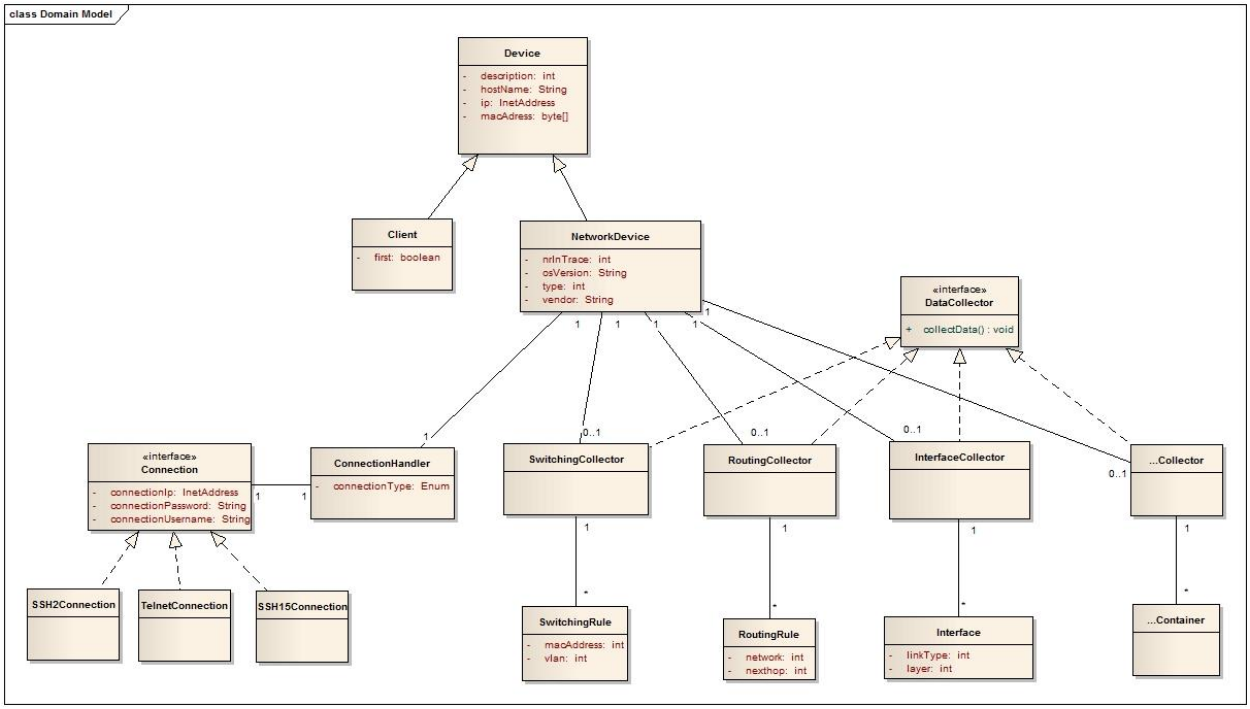


Abbildung 3: Domain Model

4.2.1 Klassen

Es werden der Einfachheit halber lediglich jene Klassen beschrieben, welche eine eigenständige Funktion besitzen. So sind sich die verschiedenen Collector z.B. so ähnlich, dass dies lediglich zu viel Text ohne neuen Inhalt führen würde.

4.2.1.1 Device

Device bildet eine physische Hardware ab, die auf dem Graphen dargestellt werden kann. Ein Device bildet die Oberklasse für „NetworkDevice“ und „Client“ und besitzt somit alle Attribute, die von beiden Klassen verwendet werden.

4.2.1.2 Client

Der Client stellt die zwei Geräte dar, zwischen denen der Pfad gefunden werden soll. Der Client leitet von Device ab.

4.2.1.3 NetworkDevice

Ein NetworkDevice repräsentiert ein Gerät, das auf dem Pfad zwischen Gerät A und Gerät B liegen kann. Es hält alle Funktionen des Gerätes zusammen und wird bei der Informationsbeschaffung auf dem Pfad initialisiert und bleibt erhalten, bis eine neue Berechnung in Auftrag gegeben wird.

4.2.1.4 …Collector (inkl. Interface)

Ein spezifischer Collector übernimmt die Aufgabe, Informationen vom Gerät abzufragen, diese zu interpretieren und sie dem Netzwerkgerät zur Verfügung zu stellen. Alle Collectoren leiten vom Interface „Collector“ ab und müssen somit darin definierten Methoden implementieren. Näheres dazu ist im Punkt Collector zu finden.

4.2.1.5 …Container

Der Collector legt seine Erkenntnisse im passenden Container ab. Meist besitzt der Collector eine Liste von Containern, um jederzeit alle Ausgaben des Netzwerkgerätes abfragen zu können.

4.2.1.6 ConnectionHandler

Der ConnectionHandler sorgt sich um die Verbindungen des Servers. Natürlich kann gewählt werden, welche Technologie beim Authentisieren verwendet werden soll und wie die verschiedenen Login Informationen lauten. Alles Weitere wird vom ConnectionHandler erledigt.

4.2.1.7 ...Connection (inkl. Interface)

Das Interface „Connection“ beschreibt lediglich ein paar der Methoden, die von den verschiedenen Connections implementiert werden. Die einzelnen Connection-Klassen sind gezwungen, das Interface, bzw. die darin definierten Methoden zu implementieren.

4.3 Systemsequenzdiagramm

4.3.1 Visualisierung (Pfad suchen)

In der Library ist die komplette Logik unserer Applikation abgebildet. Darum werden alle Operationen im Algorithmus ausgelöst und ausgewertet. Der Algorithmus erhält von seinem Creator einen Pfad, der dann durch die im Algorithmus implementierte Logik gefüllt wird.

4.3.1.1 Diagramm

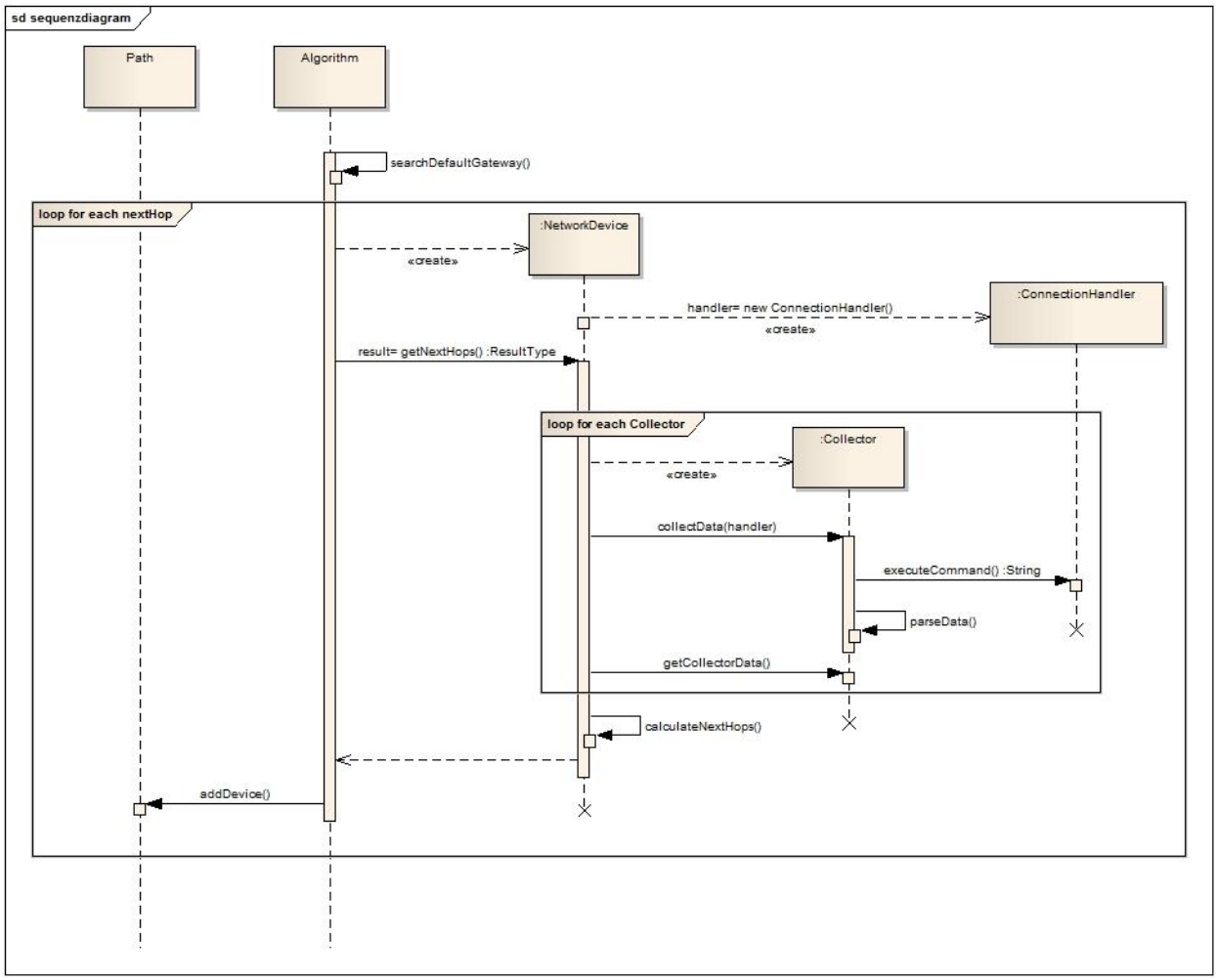


Abbildung 4: Sequenzdiagramm

4.3.1.2 Ablauf

Der Algorithmus ist für die Initialsuche des Default-Gateways zuständig. Er benötigt dazu ein Seed-Device, das im ManufactorData.xml File hinterlegt ist. Auf diesem Device sucht er den Default-Gateway für das erste Device (Host A). Sobald der Default-Gateway gefunden wurde, wird dieser instanziiert und dem Pfad hinzugefügt. Auf diesem Device wird nun das erste Mal nach den nächsten Hops (getNextHops()) in Richtung Host A gesucht. Nachdem der Pfad bis zum Host A abgebildet wurde, wird nun ausgehend vom Default-Gateway der Pfad in Richtung Host B in Angriff genommen. Sobald beide Richtungen abgebildet wurden ist der Pfad komplett im Pfad-Objekt abgebildet und kann dort ausgelesen werden.

Die Methode getNextHops() liefert einen ResultType, welcher die Neighbors (die nächsten Network-Devices auf dem Pfad) beinhaltet. Die gefundenen Neighbors werden nun als Network-Device instanziiert. Auf diesen neu gefundenen Geräten wird wiederum die getNextHops() Methode ausgeführt. Sobald ein neues Network-Device gefunden wird, wird dieses im Path-Objekt eingetragen und, so baut sich der Pfad nach und nach auf.

Sobald ein neues Network-Device instanziiert wird, wird ein neuer Connection-Handler erstellt, welcher für die Verbindung zu den physischen Geräten zuständig ist. Über diesen Handler werden alle Abfragen (Commands auf den Geräten) ausgeführt.

Jedes Gerät hat eigene Collectors, die für die Datenbeschaffung des Devices zuständig sind. Entsprechend den Funktionen eines physischen Gerätes werden verschiedene Collectors dem Network-Device hinzugefügt. In den Collectors sind alle Parameter für das Finden der Neighbors enthalten. Über Methode collectData() wird der Collector angewiesen, die Daten vom physischen Gerät zu lesen. Für diesen Schritt benutzt der Collector den Connection-Handler, welcher über eine Connection die Commands ausführt. Die Connection liefert die gewünschten Daten, welche dann über den Connection-Handler an den Collector weitergegeben werden. Der Collector wertet die Daten aus und speichert diese in den entsprechenden Container-Objekten.

Sobald alle nötigen Collectors dem Network-Device hinzugefügt und die entsprechenden Daten gesammelt wurden, können die Daten aus den Collectors ausgelesen werden. Die Logik im Network-Device ist nun für die Verwertung dieser Daten zuständig und liefert die nächsten Hops. Gleichzeitig prüft diese Logik auch, ob das Ziel (Host A oder Host B) gefunden wurde. Die Resultate werden über einen ResultType an den Algorithmus übergeben.

Entsprechend den von den Network-Devices gelieferten Daten wird so im Algorithmus der Pfad aufgebaut.

4.3.2 Systemoperationen

4.3.2.1 searchDefaultGateways()

Diese Systemoperation ist für die Suche des Initial-Devices zuständig. Für diese Operation ist ein Seed-Device notwendig. Auf diesem wird der Default-Gateway gesucht und als Ausgangspunkt für den Pfad markiert. Nach dieser Operation ist ein Default-Gateway als Ausgangspunkt vorhanden, von wo aus Host A und Host B gesucht werden können.

4.3.2.2 getNextHops()

Mit dieser Operation wird auf dem Network-Device die Suche nach den Neighbors ausgelöst. Die im Network-Device implementierte Logik sucht nach den Neighbors. Für diesen Schritt muss ein Network-Device mit dem Connection-Handler instanziiert sein. Diese Methode liefert einen ResultType, welcher die Neighbors beinhaltet.

4.3.2.3 collectData()

Mit diesem Aufruf werden auf den Collectors die Daten gesammelt. Dazu verbindet sich das Network-Device mit dem physischen Device und der Collector übernimmt dann die Sammlung der Daten. Im Collector werden die erhaltenen Daten geparst und gespeichert. Bevor diese Operation ausgeführt werden kann muss eine Connection vorhanden sein, welche erfolgreich connected wurde. Nach dem diese Operation getätigt wurde, sind die Daten im Collector abgespeichert.

4.3.2.4 executeCommand()

Über diese Operation wird auf dem Connection-Handler die executeCommand() Methode auf der instanziierten Connection ausgelöst. Diese Operation liefert die Antwort eines Gerätes in der Form eines Strings. Voraussetzung für diese Operation ist ein Connection-Handler, welcher eine Connection enthällt, welche connected wurde.

4.3.2.5 getCollectorData()

Hier werden die Daten, welche in den einzelnen Collectors gespeichert sind, abgerufen und für die weitere Verarbeitung bereitgestellt.

4.3.2.6 calculateNextHops()

In dieser Operation ist die ganze Logik eines Network-Devices abgebildet. Hier werden die Layer 3 und Layer 2 Technologien anhand der gesammelten Daten überprüft und ausgewertet. Des Weiteren wird überprüft, ob das Ziel gefunden wurde. Voraussetzung für diese Operation sind korrekte Daten in den Collectors. Diese Operation liefert einen ResultType, welcher dem Algorithmus zur weiteren Verarbeitung übergeben wird.

4.3.2.7 addDevice()

Diese Operation fügt ein gefundenes Device dem Path-Objekt hinzu. Hier werden die entsprechenden Devices miteinander verknüpft.

5. Softwarearchitektur

5.1 Einführung

In diesem Dokument wird die gesamte Softwarearchitektur abgebildet und dokumentiert.

5.2 Systemübersicht

Die Systemübersicht sieht wie folgt aus. Unten folgend sind die einzelnen Komponenten genauer beschrieben.

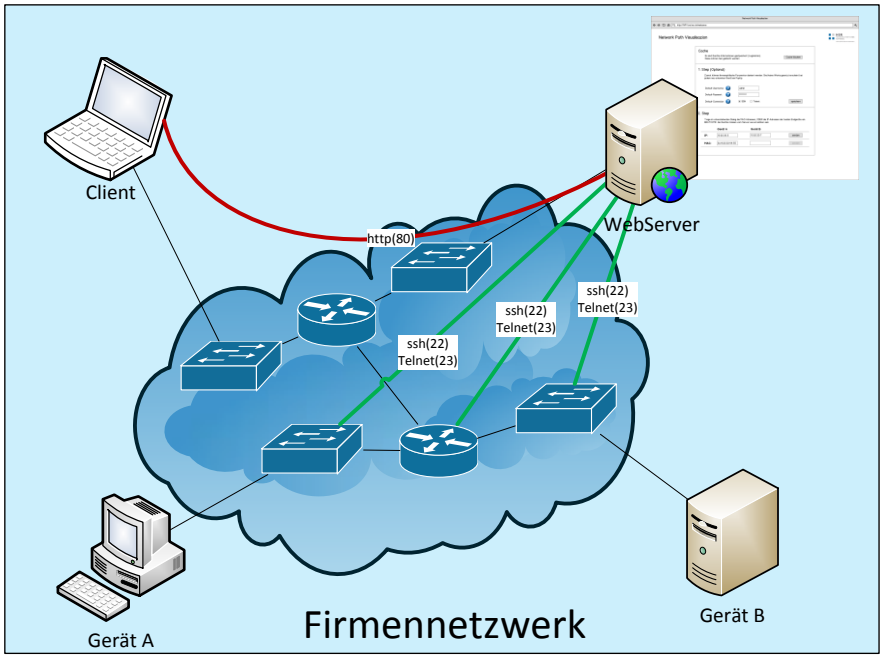


Abbildung 5: Systemübersicht

5.2.1 Webserver

Der Webserver stellt die zentrale Komponente unseres Projektes dar. Er enthält die Logik für die Traversierung und die nötigen Komponenten zur Darstellung der Webseite und des visualisierten Pfades.

5.2.2 Client

Beim Client handelt es sich um das Endgerät, das der Servicetechniker verwendet um via Webbrowser seine Anfragen an den Webserver zu senden.

5.2.3 Gerät A/B

Die Geräte A und B stellen Geräte dar, deren Pfad zueinander zu untersuchen ist. Gerät A wird jeweils das Startgerät sein, wobei Gerät B das Ende des Traversierungsalgorithmus bildet.

5.2.4 Netzwerkgeräte

Die Netzwerkgeräte innerhalb des Pfades zwischen Gerät A und B werden von der Applikation angesprochen. Dies passiert entweder über SSH oder Telnet. Im Weiteren ist auch eine Verbindung via Terminalserver möglich.

5.3 Architektonische Ziele & Einschränkungen

5.3.1 Ziele

5.3.1.1 Erweiterbarkeit

Da im Rahmen der SA unmöglich alle Technologien und alle Hersteller im Netzwerkbereich beachtet werden können ist es unter anderem unser Ziel, die Applikation so zu programmieren, dass man mit relativ wenig Aufwand diese Technologien/Hersteller ebenfalls unterstützt. Dies trifft vor allem auf die Library zu, da der Webservice dabei eine eher zweitrangige Rolle spielt und auf Wunsch auch komplett ausgetauscht werden könnte.

5.3.1.2 Usability

Das Programm soll schlussendlich einfach zu bedienen sein und den Nutzer schnell zum Ziel führen. Unnötige Aktionen sollen wenn immer möglich unterbunden werden.

5.3.1.3 Privacy und Security

Die Applikation speichert Daten lediglich im Rahmen der gerade aktiven Troubleshooting-Session. Danach werden alle Daten wieder vernichtet. Da also keine Daten persistent gespeichert werden, besteht kein erhöhter Bedarf an Sicherheit.

Durch SSH wird der Traffic zwischen WebServer und Netzwerkgerät verschlüsselt vollzogen, doch falls die Authentifizierung per Telnet gewählt wird, muss damit gerechnet werden, dass verwendete Benutzernamen und Kennwörter nachher bekannt sind.

5.3.2 Einschränkungen

Folgende Einschränkungen sind im Umgang mit dem Produkt gegeben:

- Man braucht zwingend IP- Connectivity zu den entsprechenden Netzwerkgeräten.
- Login-Informationen müssen vorhanden und eingegeben werden.
- Sämtliche Netzwerkgeräte, die traversiert werden, müssen von Cisco stammen.
- Die eingesetzten Cisco-Geräte müssen ein kompatibles Cisco-OS installiert haben.

5.4 Externes Design (Webseite)

Im Rahmen der Designanalyse wurden untenstehende Bilder entworfen, die zeigen sollen, wie man mit der Applikation interagieren kann. Auf der rechten Seite ist jeweils genauer beschrieben, was auf der entsprechenden Seite ersichtlich ist.

5.4.1 Mockups

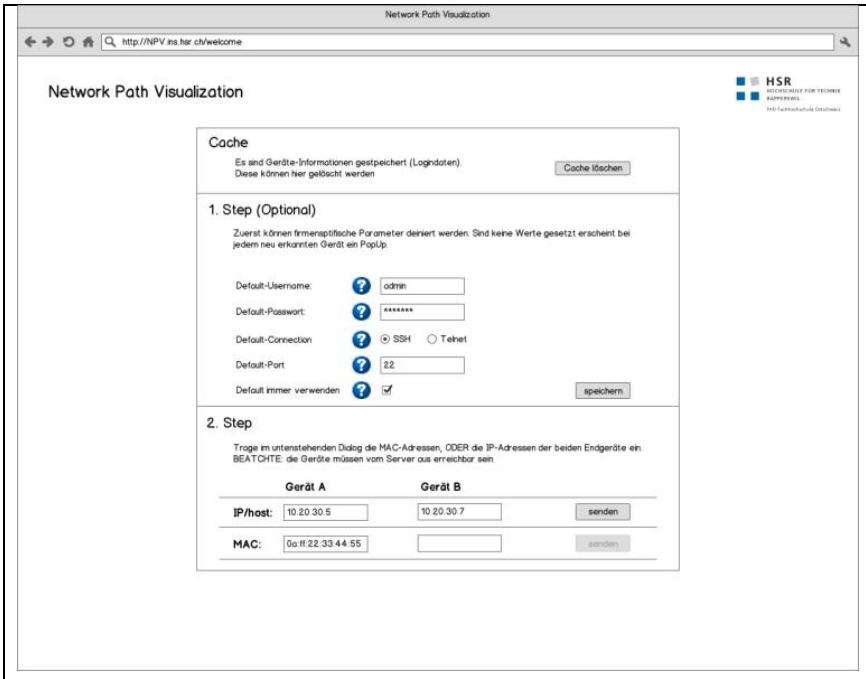


Abbildung 6: Welcome

Welcome-Page

Auf der Welcome-Page wird man landen, sobald man die Adresse des Webservers wählt. Diese lässt einem alle Parameter auf einen Blick definieren. Durch einen Klick auf senden unter „2. Step“ gelangt man nach getaner Kalkulation zu den unten folgenden Bildern.

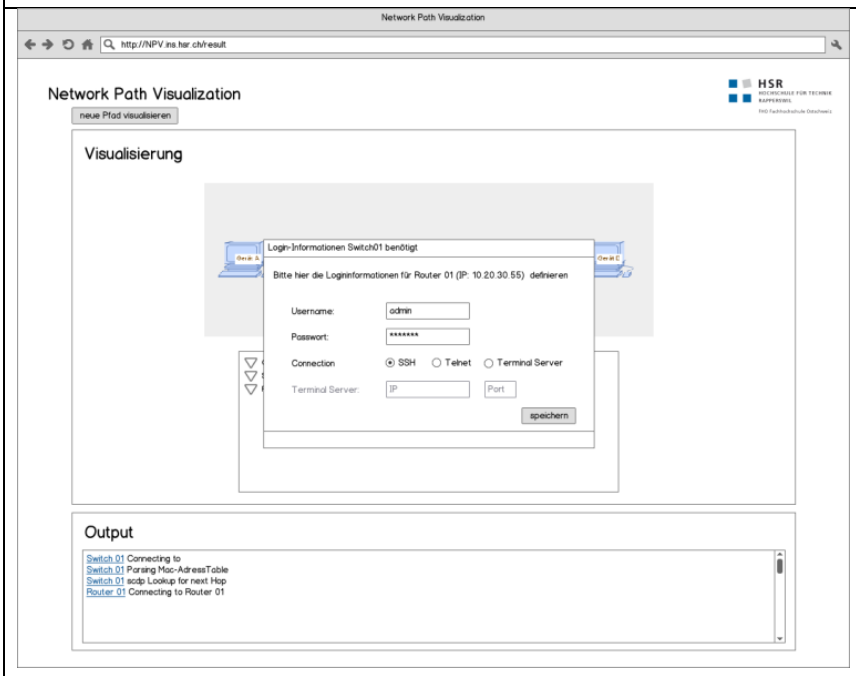


Abbildung 7: PupUp

PopUp

Sobald im System nicht genügend Parameter vorhanden sind, meldet es sich mit einem PopUp, um die fehlenden Informationen eingeben zu lassen. Dies kann im Extremfall also bei jedem Netzwerkgerät, das auf dem Pfad von Gerät A nach B traversiert wird, erscheinen.

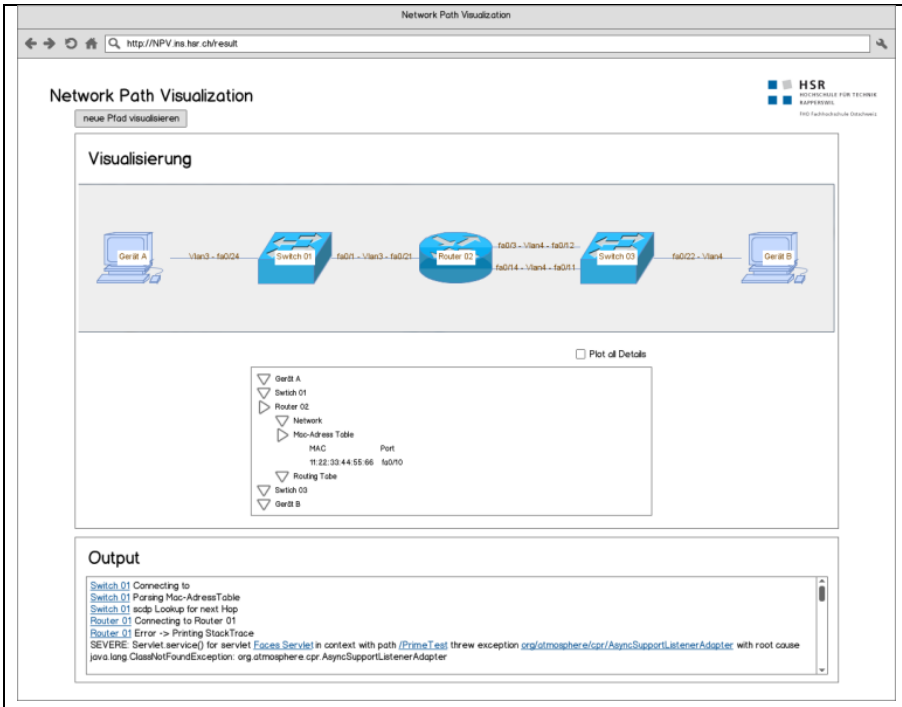


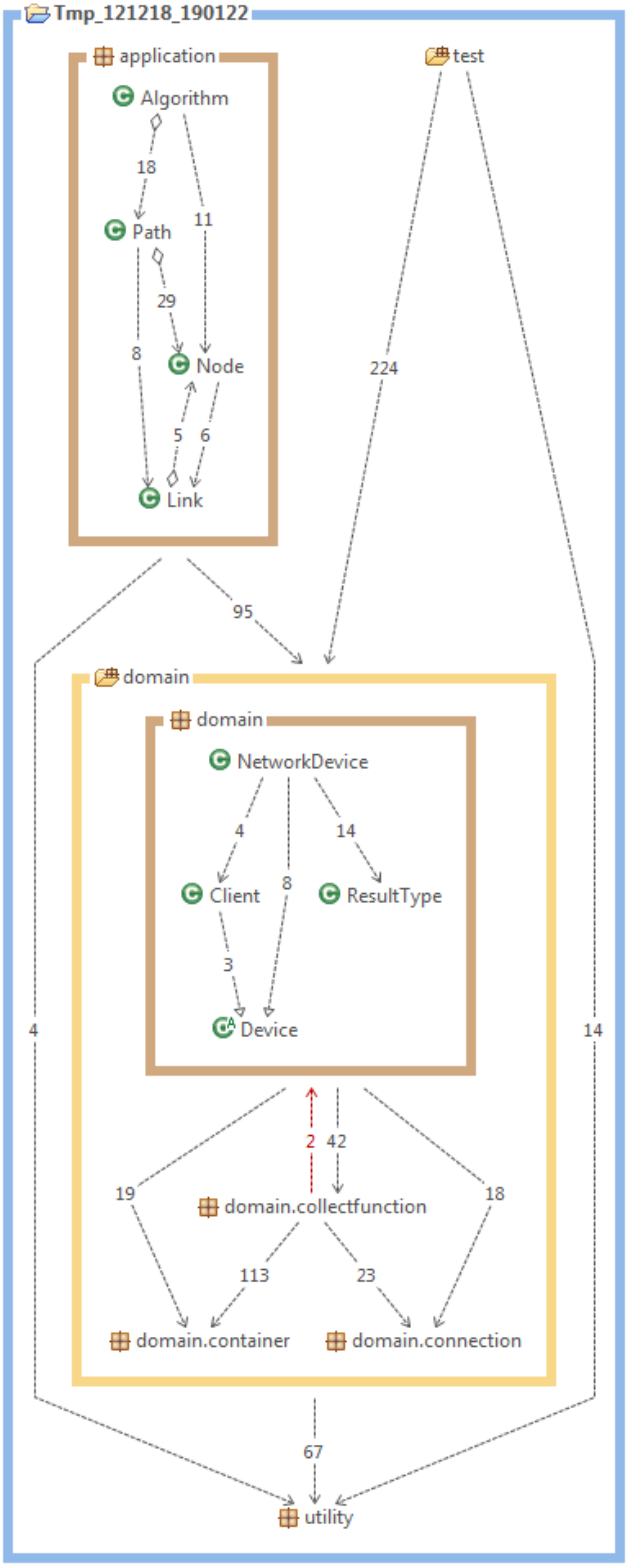
Abbildung 8: Visualisazion

Visualization

Sobald der Algorithmus abgeschlossen ist, präsentiert sich die Webseite mit allen nötigen Erkenntnissen. Im Abschnitt „Visualisierung“ werden der Graph, sowie eine hierarchische Ansicht der durchlaufenen Netzwerkgeräte angezeigt. Im unteren Abschnitt „Output“ wird ein Log angezeigt, das den Ablauf des Algorithmus nachvollziehen lässt. Falls ein Error auftreten sollte, ist dies ebenfalls hier ersichtlich.

5.5 Logische Architektur

5.5.1 Dependency graph



Wie der Grafik auf der linken Seite zu entnehmen ist, wurde der Klassenstruktur sehr viel Beachtung geschenkt.

Zugriffe finden stets in halachischer Weise (von oben nach unten) statt. Damit werden zirkuläre Abhängigkeiten verhindert.

Im Vordergrund stehen natürlich die Packages „application“ und „domain“. Aus diesem Grund sind hier die Abhängigkeiten innerhalb der Packages ebenfalls dargestellt.

Wie dem Diagramm zu entnehmen ist, werden lediglich für die verschiedenen Klassen in der Domain und für die Hilfsklassen im Package „utility“ getestet.

Abbildung 9: Dependency

5.6 Prozesse und Threads

Die Software besteht, wie bereits erwähnt, aus den zwei Komponenten (Webseite und Library). Die Webseite dient lediglich als Schnittstelle zur Library und der Interaktion mit dem Benutzer. Interessante Prozesse finden also vorwiegend in der Library statt. Um mit der Webseite unabhängig vom Algorithmus arbeiten zu können, wird der Algorithmus jeweils in einem eigenen Thread gestartet.

Dies ist im folgenden Bild illustriert. In der Schlaufe (loop) werden nun völlig unabhängig von der Webseite neue Netzwerkgeräte initialisiert und dem Pfad hinzugefügt. Bei jeder Änderung im Pfad wird das Bean informiert und lässt die Webseite des Betrachters erneuern.

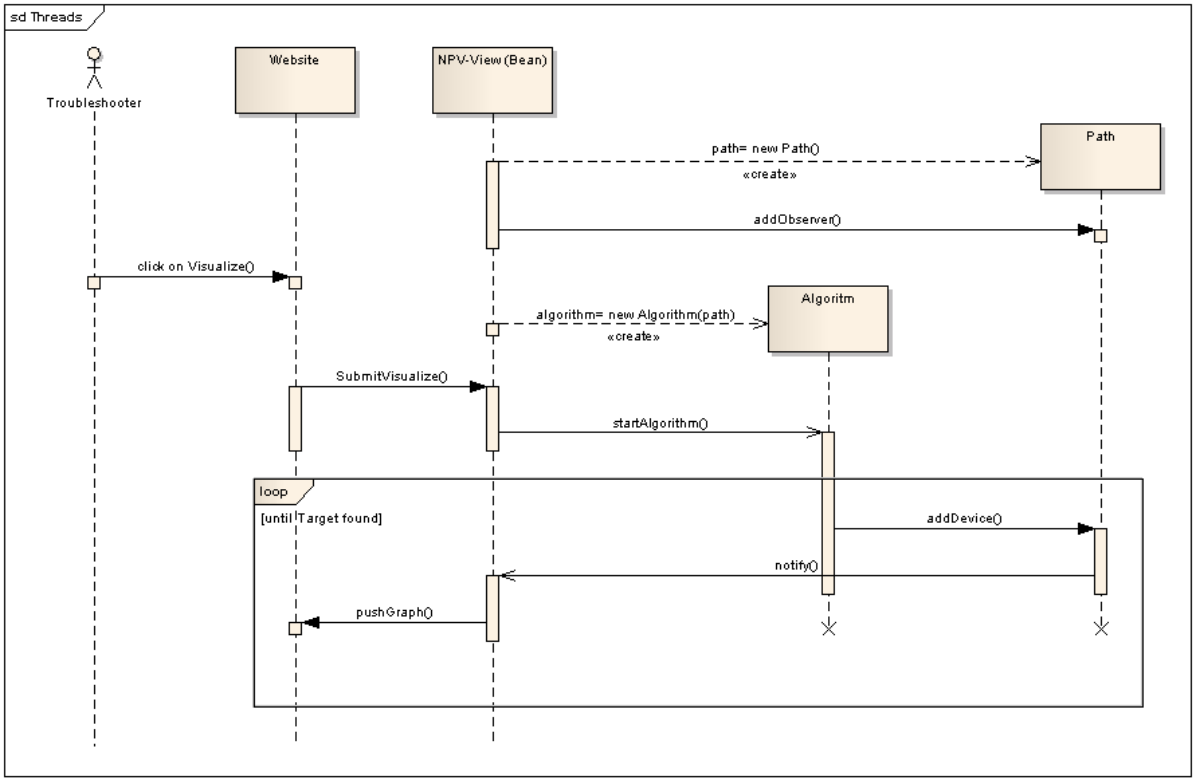


Abbildung 10: Threads

5.7 Erweiterbarkeit

Wie bereits in den Zielen der Applikation festgehalten, soll es einfach möglich sein, den Aufgabenbereich zu erweitern. Zum einen ist dies durch die Verwendung des XML-Dokuments bereits jetzt möglich, aber in diesem Abschnitt soll beschrieben werden, wie vorgegangen werden muss, um der Applikation eine vollkommen neue Funktion beizubringen.

5.7.1 Connection

Der ConnectionHandler ist für die Verwaltung der verschiedenen Connection-Typen eines Network-Devices zuständig. Jedes Network-Device besitzt einen eigenen ConnectionHandler, welcher eine Instanz einer Connection enthält. Mit dem ConnectionHandler wird gewährleistet, dass der richtige Connection-Typ (z.B. SSH2, SSH 1.5 oder Telnet) zur Verfügung gestellt wird. Die Auswahl des richtigen Connection-Typen geschieht über einen ENUM, es können also jederzeit neue Connection-Typen hinzugefügt werden. Die entsprechenden Connection-Informationen werden dann in der entsprechenden Connection, welche vom ConnectionHandler instanziert wurde, konsistent gehalten.

Um einen neuen Connection-Typen zu implementieren, muss lediglich eine neue Klasse erstellt werden, die von dem Interface Connection ableitet. Falls die Verarbeitung der Daten in den Collectors nicht angepasst wird, müssen die Daten, welche von executeCommand() geliefert werden, identisch mit einer Ausgabe aus einer SSH- oder Telnet-Verbindung sein. Sobald die Klasse existiert, kann über den ENUM ConnectionType im ConnectionHandler in der Methode connect() die neue Klasse instanziert werden. Fehlgeschlagene Verbindungsversuche werden über die Exceptions ConnectionAuthException (falsche Logininformationen) und ConnectionException (fehlgeschlagene Verbindung) gehandelt.

Momentan sind SSHv2, SSHv1.5, Telnet und eine Terminalserver Connection implementiert, welche verschiedene Libraries verwenden. Die Meisten neueren Geräte unterstützen SSHv2, aus Kompatibilitätsgründen wurden aber auch noch „alte“ Standards zur Verfügung gestellt.

SSHv2 wird mit der JSch Library von JCraft implementiert:
http://www.jcraft.com/jsch/

SSHv1.5 wird mit der JTA Library realisiert:
http://www.javatelnet.org

Die Telnet und Terminalserver Connection verwenden die Apache Commons Net Library:
http://commons.apache.org/net/

Anmerkung: Die SSHv1.5, Terminal und Terminalserver Verbindung kann nicht zu 100% gewährleistet werden, da eingesetzte Geräte eventuell andere Syntax verwenden, welche momentan nicht in unserer Library implementiert wurden. Die Parameter können aber in den verschiedenen Connection Klassen angepasst werden.

5.7.2 Collector

Interaktionen mit einem Netzwerkgerät werden über die sogenannten „Collectoren“ realisiert. Ein Netzwerkgerät beinhaltet eine Liste von Collectoren und arbeitet mit diesen. Ein Kollektor bildet eine Funktionalität eines Netzwerkgeräts ab, bzw. repräsentiert eine Instanz, die funktionsspezifische Informationen vom Netzwerkgerät abfragt und auswertet. Meist findet die Interaktion über einen Command statt, der auf dem Netzwerkgerät abgesetzt wird. Die Antwort wird dann interpretiert und für die Zeitdauer des Algorithmus konsistent gehalten.

Um einem Netzwerkgerät eine neue Funktionalität beizubringen muss also ein neuer Collector implementiert werden. Dazu muss zuerst eine Klasse mit einem aussagekräftigen Namen erstellt werden. Zusätzlich muss die Klasse das Interface „Collector“ implementieren. Dadurch wird man gezwungen, folgende Methoden zu implementieren:

```
- public boolean collectData(ConnectionHandler connectionHandler);
- public void deleteData();
- public String printData();
- public boolean isExisting();
```

Die erste Methode wird verwendet um einen Command über den connectionHander abzusetzen und die Antwort konsistent zu speichern. Sie gibt dem Aufrufer zurück, ob der Aufruf erfolgreich war und ob mindestens eine Information im Collector gespeichert wurde.

Die zweite Methode (deleteData()) wird lediglich verwendet, um bereits gespeicherte Daten wieder zu verwerfen.

„printData()“ gibt jeweils die gesammelten Daten, wie z.B. eine Mac-Adress-Table oder eine Routing-Table als String zurück.

Mit der Methode „isExisting()“ kann geprüft werden, ob das Netzwerkgerät Daten des entsprechenden Collectors besitzt. Natürlich kann dies auch verwendet werden, um zu prüfen, ob das Netzwerkgerät diese Funktion überhaupt besitzt.

Jeder Collector erhält im XML seine eigene Sektion. In dieser werden die gerätespezifischen Parameter festgehalten. Falls ein neuer Collector erstellt wird, muss diese Sektion also jedem Gerätetyp (Z.B. Cisco mit IOS) hinzugefügt werden.

Schlussendlich sollten Tests mit Testdaten der verschiedenen Geräte geschrieben werden, damit auch sichergestellt werden kann, dass der Collector unter jedem Gerätetyp voll funktionsfähig ist.

5.8 Deployment

Da sich die Applikation lediglich auf einem Gerät befindet, ist das Deployment relativ einfach. Die Library wird komplett abgetrennt vom Webservice implementiert. Bei Änderungen wird die Library neu in das darauf aufbauende Webprojekt integriert und genutzt. Schlussendlich wird der Webservice komplett exportiert und in einem Webserver, der Servlets unterstützt, eingebettet. Somit ist die Anwendung von überall mit allen tauglichen Browsern zugreifbar.

Dies ist auch der oben beschriebenen Systemübersicht zu entnehmen.

5.9 Datenspeicherung

Daten werden aus Sicherheitsgründen niemals persistent gespeichert, weshalb auch keine Datenbank zum Einsatz kommt.

5.10 Grössen und Leistung

Die Applikation kann theoretisch unbegrenzt grosse Netzwerke darstellen, doch ab einer gewissen Anzahl an Geräten übersteigt die Visualisierung die Grösse des Monitors. Dadurch wird es nötig werden mit der Scrollbar von links nach rechts zu navigieren, um gesuchte Elemente erkennen zu können.

6. Lab

Dieser Abschnitt dokumentiert das Labor, wie wir es während der gesamten Studienarbeit nutzen durften. Im Anhang ist die gesamte Konfiguration der verschiedenen Geräte dokumentiert. Hier werden die verschiedenen Geräte und ihre Verbindung mit den anderen Geräten erläutert. Zuerst in einer physikalischen Ansicht, welche die effektive Verkabelung zeigt und nachfolgend in einer logischen Ansicht, welche die Layer3-Informationen (für Routing relevante Daten) aufzeigt.

Es ist lediglich das Netzwerk des zweiten Aufbaus dokumentiert, da es zu gewissen Teilen das des ersten Netzwerks ersetzt. Im Anhang sind alle Informationen zum ersten Aufbau zu finden.

6.1 Physical View

6.1.1 Theoretisch

Die folgende Grafik zeigt, wie die Geräte theoretisch konfiguriert sein sollten. Reine Layer2-Geräte haben also keine IP zugewiesen. Das Netzwerk ist keinesfalls eine Best-Practice-Konfiguration, doch die Applikation soll auch in einem solchen Netzwerk zurechtkommen.

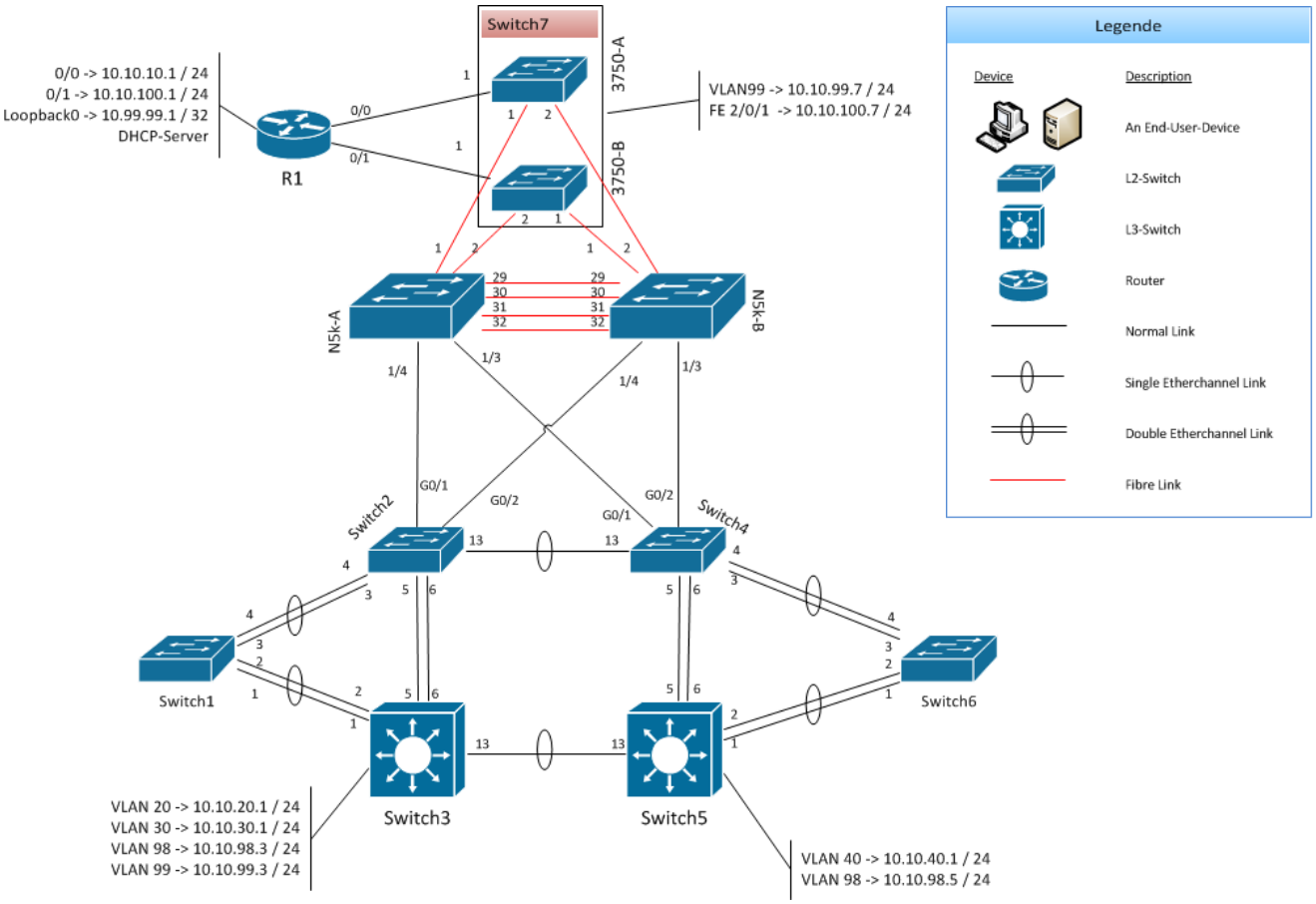


Abbildung 11: Physical-Theoretisch Ansicht

6.1.2 Effektiv

Da der Algorithmus auch auf Layer2-Geräten (reine Switches) einloggen können muss, wurde jedem Switch eine IP-Adresse aus dem VLAN 10 zugewiesen. Natürlich nicht aus dem Bereich, in dem der DHCP-Server die IP-Adressen verteilt.

Diese Grafik soll also zusätzlich die einzelnen IP-Adressen des VLAN 10 zeigen, über welche die Switches identifiziert und administriert werden.

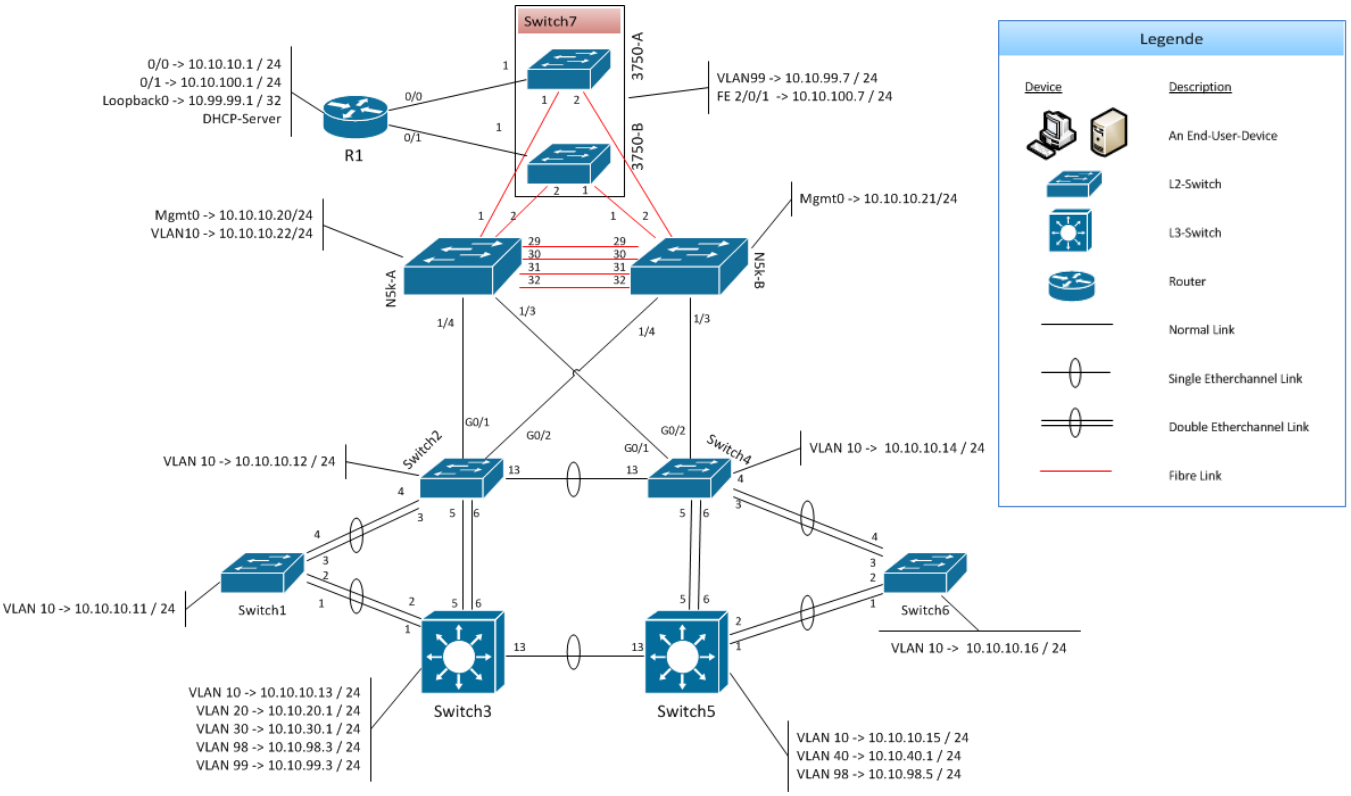


Abbildung 12: Physical-Effektiv Ansicht

6.2 Logical View

Die logische Ansicht des Netzwerks lässt schnell erkennen, dass das Netzwerk linear angeordnet ist. Die VLAN 10-20 sind jeweils mit ihrem Default-Gateway verbunden, was den Eintritts- und Austrittspunkt aus den entsprechenden VLANs zeigt.

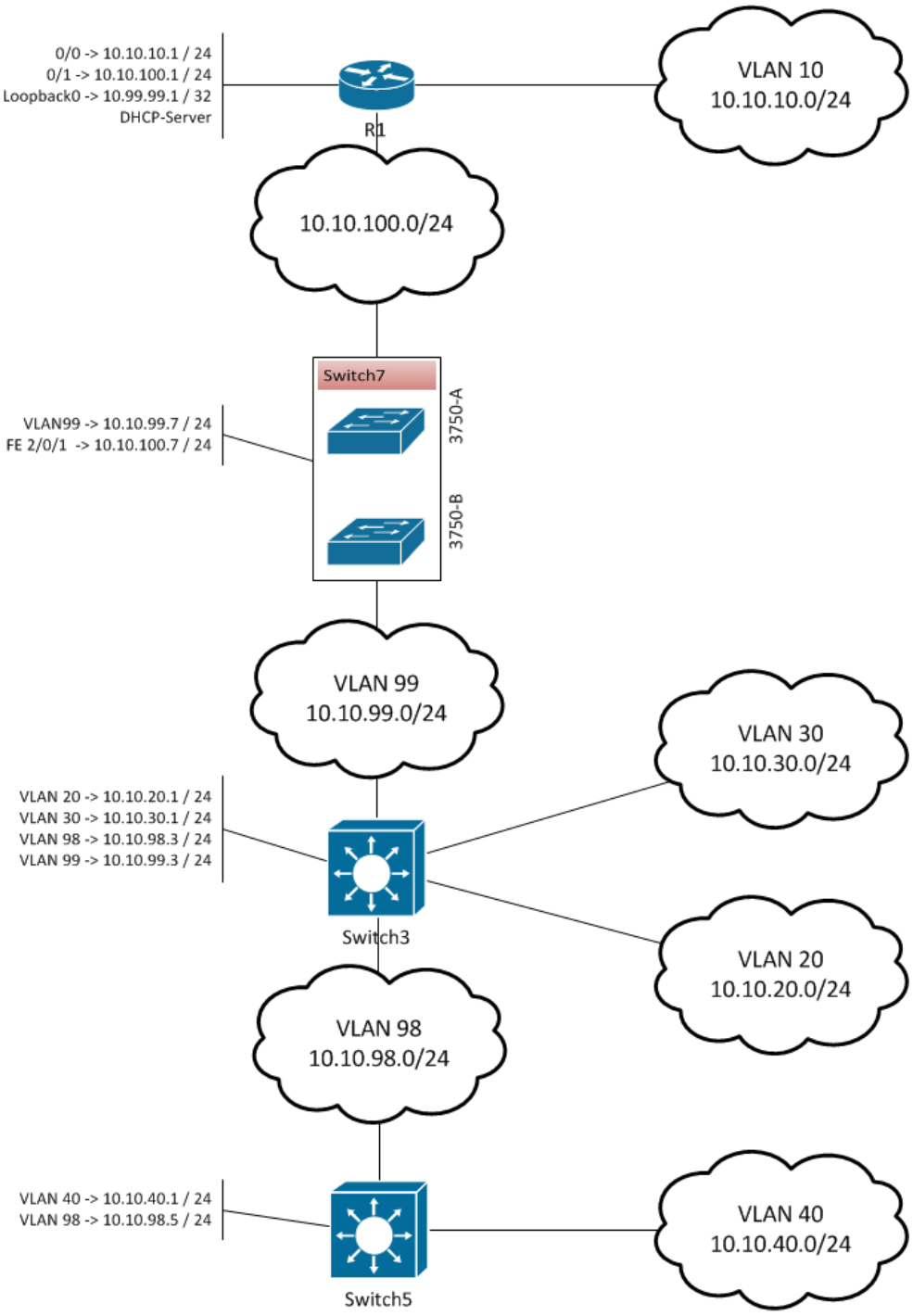


Abbildung 13: Logische Ansicht

6.3 Informationen

6.3.1 VLANS

Auf sämtlichen Switches, ausser auf den Nexus-Geräten, sind die folgenden VLANS auf den dahinter aufgeführten Ports definiert:

- VLAN 10 Port: 17, 18
- VLAN 20 Port: 19, 20
- VLAN 30 Port: 21, 22
- VLAN 40 Port: 23, 24

6.3.2 DHCP-Server

Der Router R1 übernimmt die DHCP-Funktionalität für das gesamte Netzwerk. Er verteilt jeweils Adressen aus den VLANs 10, 20, 30 und 40. Ausgeschlossen sind alle IP-Adressen zwischen 0 und 99 im letzten Oktett. Diese Adressen sind für manuell konfigurierte Geräte gedacht.

6.3.3 Devices / Credentials

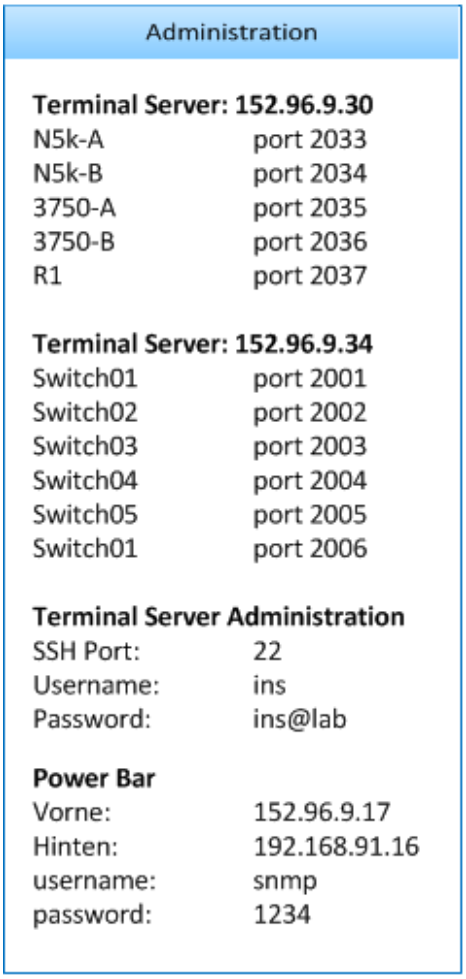
Mit Ausnahme des Switch7 sind alle Geräte genau gleich konfiguriert. Die abweichende Konfiguration ist auch ganz bewusst so gewählt, weil der Algorithmus bei diesem Gerät jeweils das Login-Fenster anzeigen soll. Es soll auch lediglich das Passwort um eine Ziffer erweitert werden.

Devices / Credentials

Allgemein (Alle Geräte)			**Ausnahme (Switch 7)**		
ip domain name: npv.hsr.ch			ip domain name: npv.hsr.ch		
Enabled Login:	SSH	√	Enabled Login:	SSH	√
	Telnet	√		Telnet	X
Default Privileges			Default Privileges		
Username:	user		Username:	user	
secret:	cisco		secret:	cisco2	
priv 15			priv 15		
Username:	admin		Username:	admin	
secret:	cisco		secret:	cisco2	
enable-Command			enable-Command		
secret:	cisco		secret:	cisco2	

Abbildung 14: Devices / Crendentials

6.3.4 Administrativ Informations



Für den direkten Zugriff auf die Geräte wurden uns zwei Terminal Server zur Verfügung gestellt. Wie links zu sehen ist, kann über den entsprechenden Port das entsprechende Netzwerkgerät erreicht werden. Der Zugriff findet dann unabhängig von der Netzwerkkonfiguration direkt über den Konsolen-Port statt.

Falls ein Zugriff auf ein Netzwerkgerät bereits belegt ist, kann die entsprechende Line (Port) durch ein Login am Port 22 des Terminal Servers die Line wieder frei geben.

Dazu muss das Kommando „clear line xx“ abgesetzt werden, wobei xx natürlich durch den entsprechenden Port zu ersetzen ist.

Die Power Bar ermöglicht es uns, den Strom der verschiedenen Geräte zu kappen. „Vorne“ bezieht sich hierbei auf die Geräte, die im vorderen Bereich des LABs (Cisco-Schulungsraum) stehen, wobei sich „Hinten“ auf die Geräte bezieht, die im Serverraum stehen.

Abbildung 15: Administrative Informationen

7. Implementation/Umsetzung

7.1 Library

7.1.1 XML

Die Applikation soll möglichst dynamisch sein. Dies beinhaltet auch die Kompatibilität mit verschiedenen Herstellern und verschiedenen Softwareversionen der Hardware-Hersteller. Aus diesem Grund werden sämtliche Parameter, die zur Kommunikation mit dem Netzwerkgerät und zur Analyse der Daten des Geräts dienen, in einem XML-Dokument definiert.

Im Weiteren dient es auch der Konfiguration des Algorithmus. So kann das SeedDevice definiert werden, welches vom Algorithmus als Erstes angesprochen wird.

7.1.1.1 Seed-Device

Mittels des “SeedDevice“-Tag wird definiert, welches Gerät vom Algorithmus zuerst angesprochen wird. Dazu sind eine IP, der Name des Herstellers, sowie die Software Version nötig.

```
<?xml version="1.0"?>
<Root>
	<!-- the Seed-Device -->
	<SeedDevice>
		<ip>10.10.10.5</ip>
		<manufactor>cisco</manufactor>
		<version>ios</version>
	</SeedDevice>
</Root>
```

7.1.1.2 Global

Innerhalb des TAGs „global“ finden sich Parameter, die entweder für alle Geräte gleich sind, oder für Aktionen benötigt werden, bei denen noch gar nicht bekannt ist, um welches Gerät es sich handelt.

```
<?xml version="1.0"?>
<Root>
	<global>
		<VersionCollector>
			<command>show version</command>
			<startLine>0</startLine>
			<footerLines>0</footerLines>
			<manufactorRegex>[Cc][Ii][Ss][Cc][Oo]</manufactorRegex>
			<versionRegex>[Ii][Os][Ss]|[Nn][Ee][Xx][Uu][Ss]
			</versionRegex>
		</VersionCollector>
	</global>
</Root>
```

7.1.1.3 Allgemein

Folgend sind die verschiedenen Stufen des XML-Dokuments erläutert. Die verschiedenen möglichen Parameter sind dem Beispiel im Anhang zu entnehmen. Dort sind alle möglichen Funktionen und Parameter definiert.

```
<?xml version="1.0"?>
<Root>
      <!-- explanation -->
      <Manufactor>
            <version>
                  <Function>
                        <param>someString</param>
                        <startLine>someNumber<startLine>
                        <FooterLines>someNumber</FooterLines>
                        <-more->some functionspecific String</-more->
                  </Function>
            </version>
      </Manufactor>
</Root>
```

Der Root-Tag definiert den Bereich des Dokuments. Es werden lediglich Sub-Tags innerhalb dieses Tags involviert. Der Aufbau ist stets hierarchisch in der Form „Hersteller“ (z.B.: cisco), „Software-Version“ (z.B.: IOS), „Funktion“ (z.B.: RoutingCollector). So können beliebige Geräte ganz einfach unterstützt werden.

Ein Auszug des aktuellen Files „ManufactorData.xml“ ist im Anhang hinterlegt. Es ist jedoch erst für Cisco-IOS und Cisco-Nexus konfiguriert.

7.1.1.4 Erläuterungen

7.1.1.4.1 Regex

Hier werden lediglich einige der eingesetzten Regular Expressions deklariert und erläutert, wie diese angepasst werden. In den verschiedenen Java-Klassen wird natürlich ein RegexMatcher verwendet, der mit all den gebräuchlichen Ausdrücken arbeiten kann.

Weiterführende Infos sind hier zu finden:

- http://www.vogella.com/articles/JavaRegularExpressions/article.html
- http://www.regexplanet.com/advanced/java/index.html

7.1.1.4.1.1 Allgemein

Expression	Bedeutung
\s	Ein Lehrzeichen / Kurzform für [\t\n\x0b\r\f]
^regex	Pattern muss am Anfang der Zeile stehen.
regex$	Pattern muss am Ende der Zeile stehen
`[a-f0-9]`	Definiert das Alphabet / Hier also alle Hex-Werte (0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f)
`{2}`	Zeichen des definierten Alphabets erscheinen genau 2 Mal
`{4,7}`	Zeichen des definierten Alphabets erscheinen zwischen 4 und 7 Mal
regex1|regex2	Entweder muss regex1 oder regex2 zutreffen

7.1.1.4.1.2 Oft verwendete Regex

Anwendung	Beispiele	Regex

IP
Netzadresse
Subnet in Oktalschreibweise
	10.20.30.40
	215.0.0.254
		`([01]?\d\d?|2[0-4]\d|25[0-5])\.`
		`([01]?\d\d?|2[0-4]\d|25[0-5])\.`
		`([01]?\d\d?|2[0-4]\d|25[0-5])\.`
		`([01]?\d\d?|2[0-4]\d|25[0-5])`

Subnet in Dezimalschreibweise	/24 /8	`/[0-9]\s|/[0-3][0-9]`
Mac-Adresse in Cisco-Schreibweise	0123.4567.89ab	`([a-fA-F0-9]{4}+[.]){2}` `([a-fA-F0-9]{4})`
Interfacebezeichnung	Fa0/5 FastEthernet0/5 ChannelGroupe2	`([a-zA-Z]{2})([0-9]+)([/])([0-9]+)|` `([a-zA-Z]{2})([0-9]+))`

7.1.1.4.2 Platzhalter

Platzhalter sind jeweils mit „$“ gekennzeichnet und werden von der Applikation beim Aufruf automatisch ersetzt. Diese Möglichkeit wird lediglich bei Kollektoren eingesetzt, die einen spezifischen Funktionsaufruf auf dem Endgerät voraussetzen. Untenstehend nun zu sehen.

Wo	Platzhalter	Beschreibung
`<NeighborCollector>-><command>`	`$interface$`	Dieser Platzhalter wird beim Aufruf durch die Bezeichnung des jeweiligen Interfaces ersetzt.
`<TraceRouteCollector>-><command>`	`$ip$`	Dieser Platzhalter wird durch die entsprechende IP ersetzt.
`<EtherChannelCollector>-><command>`	`$ChannelNr$`	Wird ersetzt durch den entsprechenden Chanel

7.1.2 Algorithmus

7.1.2.1 Ablaufdiagramm

Der Algorithmus unserer Applikation basiert auf folgendem Diagramm.

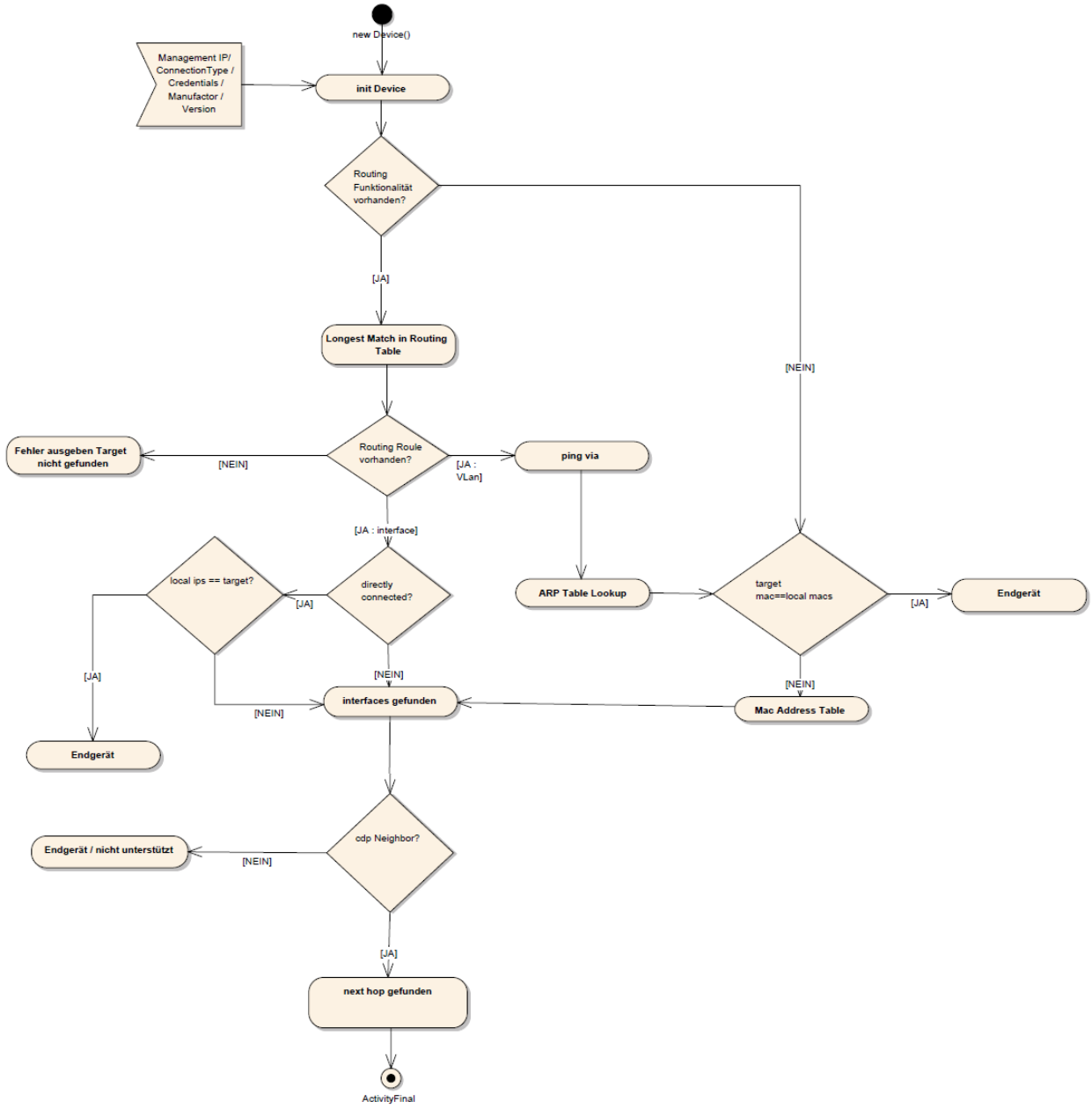


Abbildung 16: Ablaufdiagramm Algorithmus

7.1.2.2 Beschreibung

1. Dem Algorithmus werden verschiedene Parameter übergeben, die für die Berechnung des Pfades notwendig sind. Die wichtigsten sind die Daten zum Host A und Host B.
2. Das Seed-Device wird initialisiert. Auf diesem Device wird nun nach dem Default-Gateway gesucht. Dieser wird initialisiert und als Initial-Device vermerkt.
3. Nun wird jedes gefundene Device nach folgendem Ablauf ausgewertet.
4. Es wird überprüft ob ein Routing auf dem Device implementiert ist.
    a. Falls ein Routing vorhanden ist werden die Funktionalitäten des Routing überprüft und ausgewertet. Zusätzlich wird ein Ping auf das Ziel ausgeführt.
        i. Falls keine Routing Einträge gefunden werden kann das Gerät von diesem Gerät aus nicht erreicht werden, somit ist das Ende des Pfades erreicht.
        ii. Falls eine Routing-Roule gefunden wurde welche ein VLan enthält, wird die ARP-Tabelle abgefragt. Und danach geht es weiter mit der Switching-Function.
        iii. Falls das Gerät direkt mit dem Subnet verbunden ist, werden direkt die Interfaces überprüft.
    b. Falls kein Routing implementiert ist wird die Switching-Function aufgerufen. Diese kann auch aus der Routing-Function aufgerufen werden. Hier wird die Mac-Address-Table überprüft und ausgewertet. Danach geht es weiter mit dem Auswerten der Interfaces. Falls die Ziel-Mac-Adresse auf einem Interface gefunden wurde, ist das Ziel gefunden.
5. Beim Überprüfen der Interfaces werden für die gefundenen Interfaces die Neighbors gesucht. Diese werden später dem Algorithmus zur Weiterverarbeitung übergeben um neue Devices zu initialisieren. Falls ein Interface die Ziel-IP-Adresse besitzt ist das Endgerät gefunden.
6. Falls keine Neighbors gefunden wurden, ist entweder das Ziel gefunden worden oder der Neighbor ist ein Gerät, welches nicht von unserer Applikation unterstützt wird. In diesem Fall ist das Ende des Pfades erreicht.

7.1.3 Der Path und seine Nodes

Die Daten, welche der Algorithmus sammelt, werden in einer eigens dafür entwickelten Datenstruktur abgelegt. Diese basiert einerseits auf einer HashMap um sicherzustellen, dass Network-Devices lediglich einmal initialisiert werden und die bereits initialisierten Geräte schnell über die ManagementIP zu finden sind. Zum anderen besteht diese Datenstruktur aus einer DubleMultiLinkedNodeList. Damit wird sichergestellt, dass jedes Device jeweils mehrere Nachbarn in beide Richtungen (In Richtung Gerät A oder in Richtung Gerät B) haben kann.

Ein Node hat eine Liste von Nodes vor und nach ihm in der Traversierung stehen. Jeder Node hat ein eindeutiges Netzwerkgerät zugewiesen und fungiert diesbezüglich als Wrapper. Folgende Grafik soll zeigen, welche Abbildungen mit dieser Datenstruktur möglich sind.

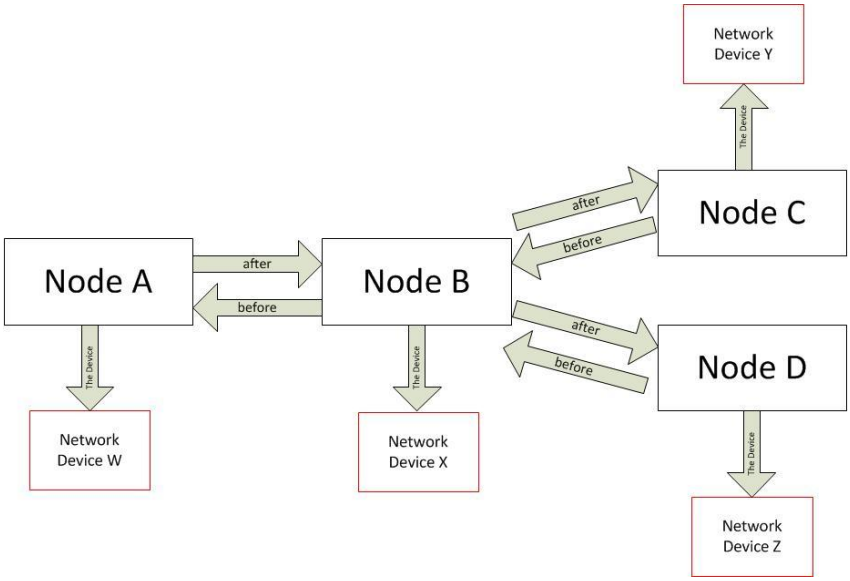


Abbildung 17: Nodelist allgemein

Im Pfad wird bereits bei der Initialisierung hinterlegt, wo sich der Node des Gerätes A, des Gerätes B und des Start-Gerätes befinden. Das Start-Gerät ist hierbei natürlich wieder der Default Gateway des Netzwerkes, in dem sich das Gerät A befindet.

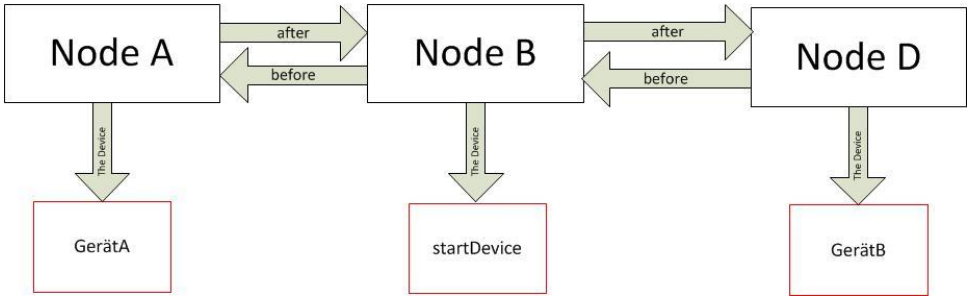


Abbildung 18: Initialisierung

Im Weiteren kennt der Algorithmus stets den Node des aktuellen Netzwerkgerätes. Sobald nun ein neues Gerät im Algorithmus initialisiert wird, wird dieses dem Pfad hinzugefügt. Dank dieser Datenstruktur kann zu jedem Zeitpunkt eine Map erstellt werden, die aufzeigt, wie die verschiedenen Netzwerkgeräte miteinander verbunden sind.

7.1.4 ResultType

Jedes Network-Device besitzt eigene Collectors. Damit der Algorithmus die Daten aus dem Network-Devices verarbeiten kann gibt es einen ResultType. Der ResultType beinhaltet Informationen über die nächsten Devices, die im Algorithmus initialisiert werden müssen und ob das Ziel-Device gefunden wurde. Das Ziel des ResultTypes ist es, die Network-Device Logik vom Algorithmus zu trennen. Damit kann gewährleistet werden, dass die Resultate über alle Netzwerk Devices gleich aussehen. Dies ist für die weitere Verarbeitung im Algorithm notwendig. In der Abbildung ist zu sehen, dass sich die ganze Logik für die Verarbeitung der Collectors im Network-Device befindet. Der Algorithmus hat so nur noch den ResultType auszuwerten.

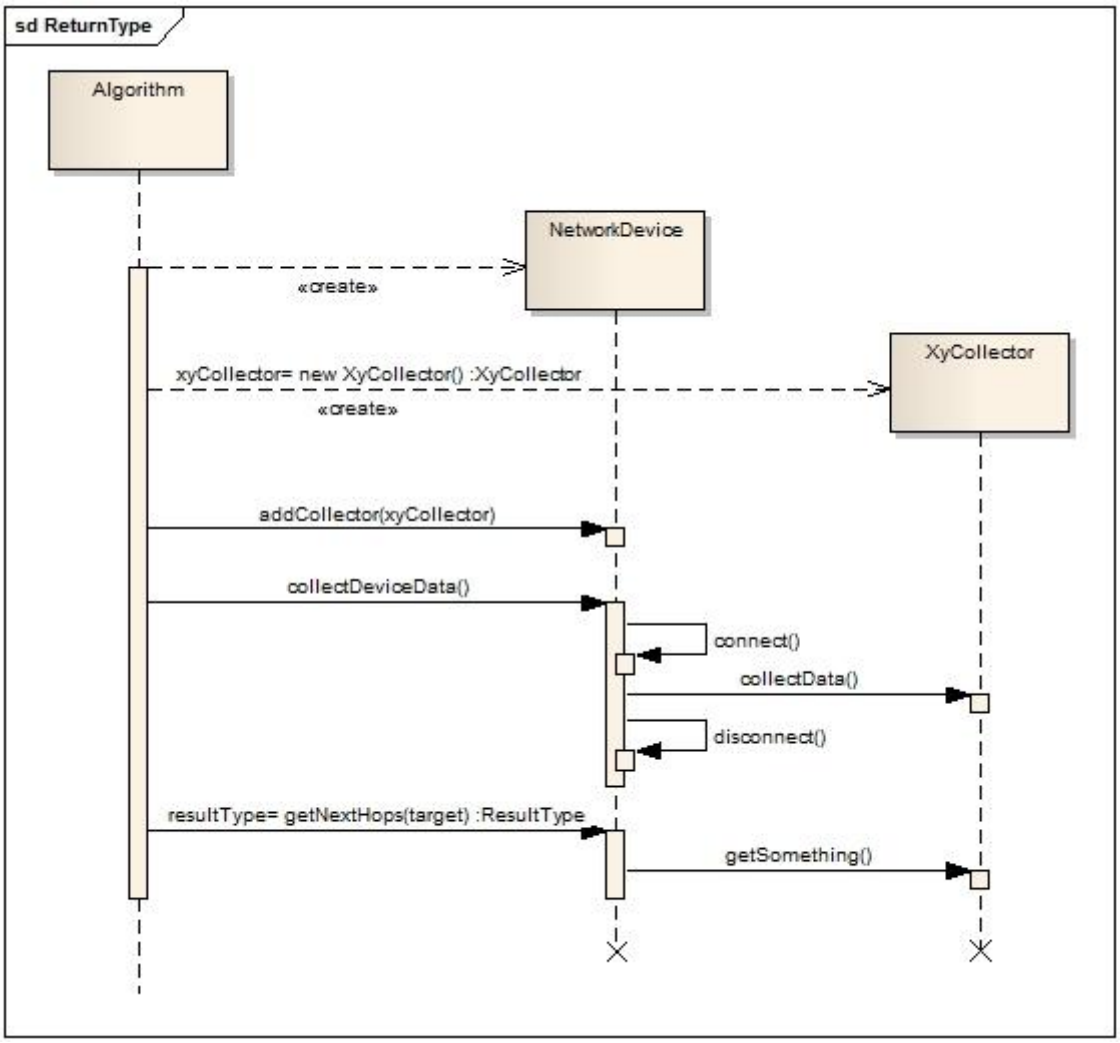


Abbildung 19: ReturnType

7.2 Website

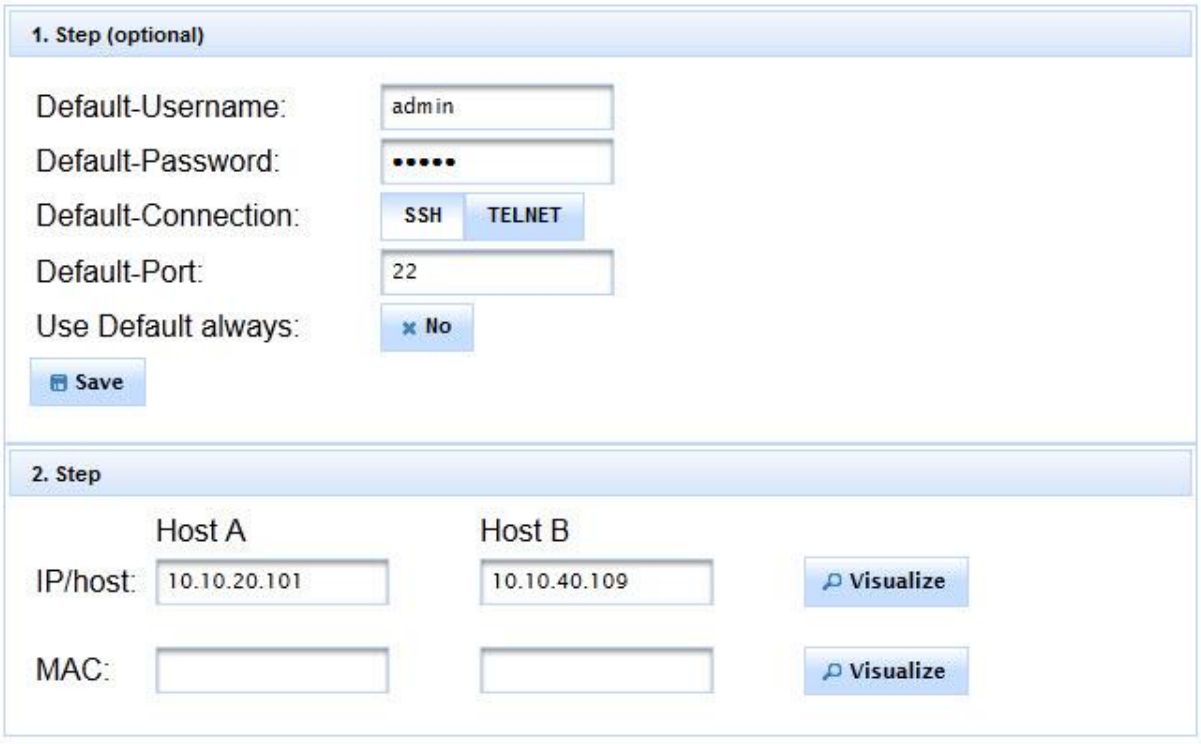
Für die Umsetzung der Benutzerschnittstelle wurde ein Webinterface gewählt. Die Technologie dahinter ist ein Tomcat Server (Version 1.7) welcher mit dem Java Runtime Environment (JRE 1.7) läuft. Für die Umsetzung der Webapplikation wird Java Server Faces (JSF) eingesetzt. Dazu wird das Mojarra JSF Framework verwendet. Die Webapplikation ist eine eigenständige Applikation und implementiert die Logik der NPV-Library. Für die visuelle Darstellung der Webseite wird mit PrimeFaces (Version 3.4.1) gearbeitet. Diese Component Suite bietet viele Grundelemente für die Webentwicklung mit JSF, unter anderem auch erweiterte Funktionalitäten für die Kommunikation zwischen Client und Server.

7.2.1 UI

Die Webapplikation besteht aus zwei Views. Beide Views greifen auf dasselbe Backingbean zu. Somit ist gewährleistet, dass die Daten zu jedem Zeitpunkt zur Verfügung stehen. Um das UI darzustellen werden Standardelemente von PrimeFaces benutzt.

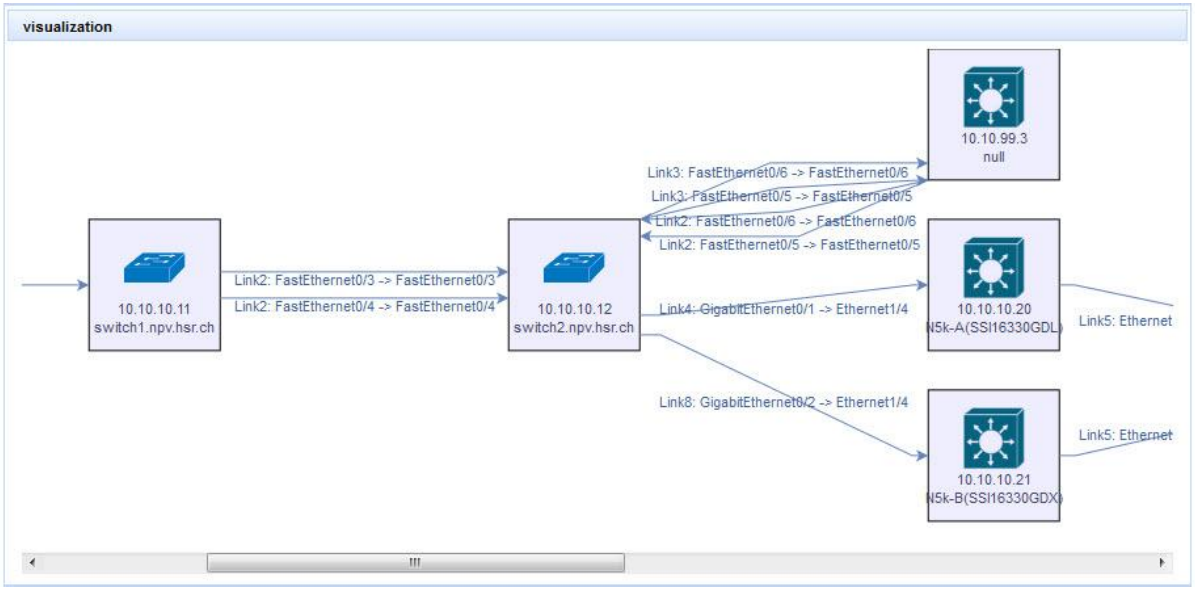
7.2.1.1 Startseite

Auf der Startseite werden die Angaben für die Pfadfindung definiert.



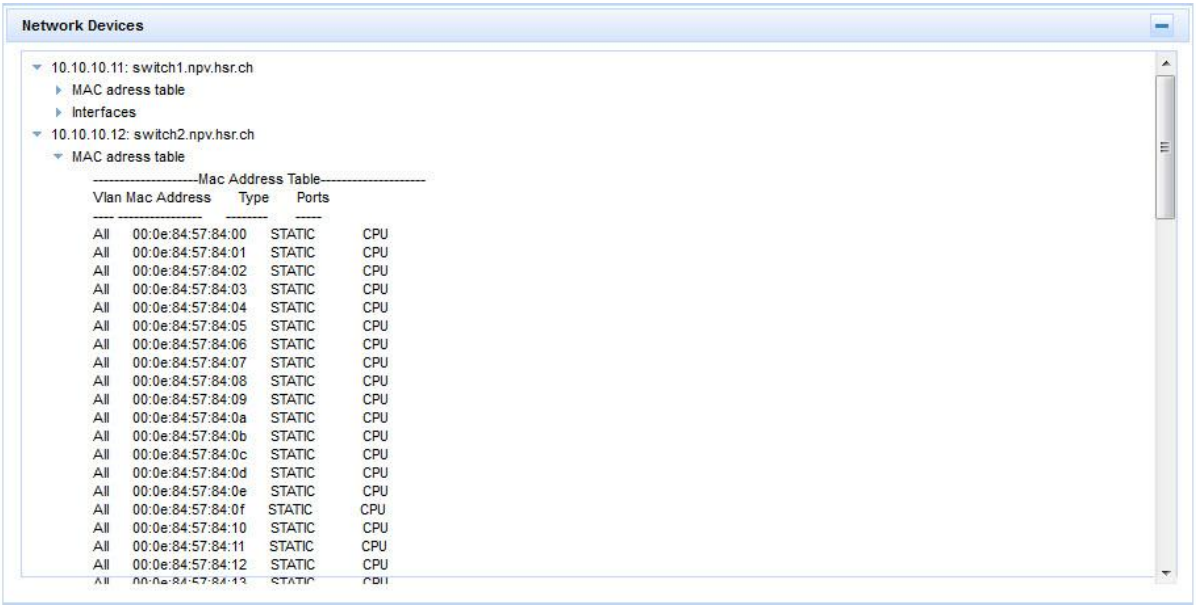
7.2.1.2 Visualisierung

Auf der Resultatseite wird im „Visualization“ Bereich die Visualisierung des Pfades dargestellt.



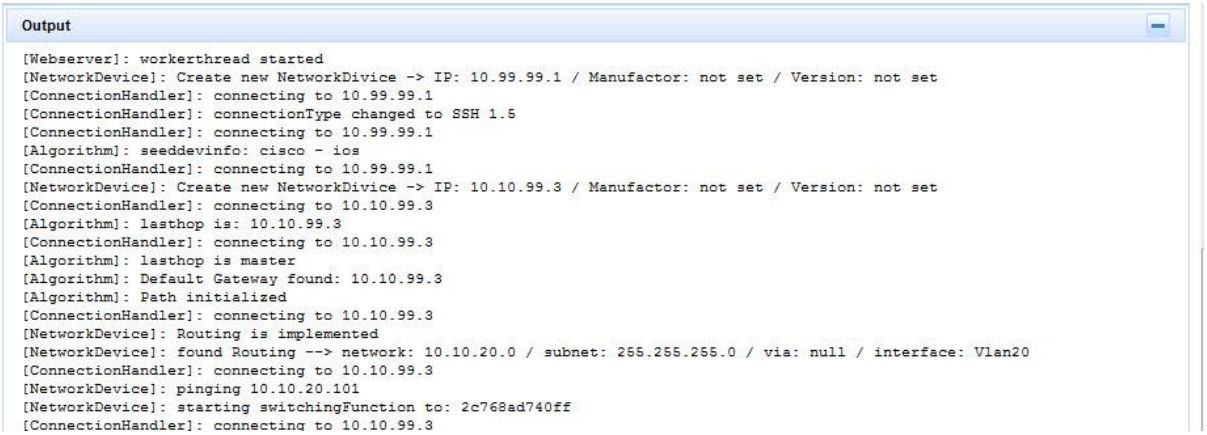
7.2.1.3 Anzeige der Netzwerk-Geräte

Im Bereich „Network Devices“ werden die gefundenen Netzwerkgeräte angezeigt. Zu den Netzwerkgeräten können auch die geparsten Informationen abgerufen werden.



7.2.1.4 Output

Der Bereich „Output“ zeigt den aktuellen Zustand der Applikation.



7.2.2 PrimeFaces Push

Immer wenn ein neuer Content im Pfad aus der Library generiert wird oder wenn zu einem Device nicht verbunden werden konnte, muss dies dem Benutzer mitgeteilt werden. Für das Pushen des Contens auf die Webseite wird die Push-Implementation von PrimeFaces verwendet. Der Push basiert auf der Atmosphere Runtime. Die Webapplikation wird über einen Observer von der Library über die entsprechende Aktion informiert. Die Webapplikation führt nun einen Push auf die Webseite durch, die dann die entsprechenden Elemente aktualisiert. Bei all jenen Elementen, die nur bei einer Änderung aktualisiert werden sollen, wird der Push-Ansatz verwendet. Hier wäre eine dauernde Aktualisierung der Elemente wie es bei einem Polling geschieht störend.

7.2.3 Polling

Für die Aktualisierung des Logs, welches den aktuellen Zustand des Algorithmus darstellt, wird ein Polling eingesetzt, welches das Log-Element auf der Webseite jede Sekunde aktualisiert. Diese Implementation ist einfach umzusetzen und reicht für die Aktualisierung des Logs völlig aus, da eine dauernde Aktualisierung hier nicht störend ist.

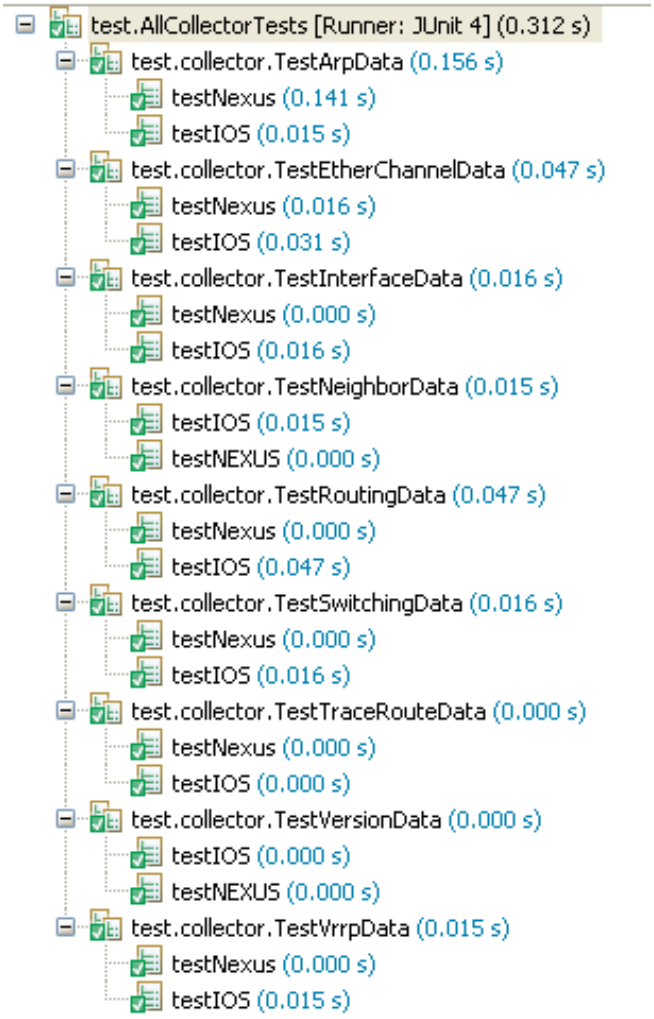
7.2.4 Visualisierung

Für die Visualisierung des Pfades wird das jgraph Framework mxGraph eingesetzt. Diese JavaScript Implementation erlaubt es, Knotenpunkte und Verbindungen zu visualisieren. Der Code für den Graph wird auf dem Webserver generiert und dann auf die Webseite gepusht. Auf der Webseite wird der Code ausgeführt. Dies führt zu einer Darstellung des Graphen auf der Webseite. Mit dieser Methode können ganze JavaScript Codeteile nachgeladen werden. Das Auslesen des Graphen geschieht über eine (JSGraphVisualizer) welche im Backingbean implementiert wird. Diese generiert auch den JavaScript Code.

7.3 Testing

Im Folgenden werden die Tests beschrieben, die sicherstellen, dass die implementierten Funktionen wie gewollt funktionieren. Dies betrifft die verschiedenen Collector und die Utility-Funktionen, die überall ihre Verwendung finden.

7.3.1 Collector



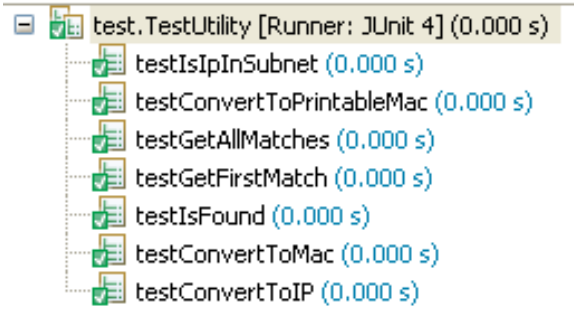
Alle Collector werden einzeln in einer Testklasse geprüft. Pro Software-Version gibt es eine Methode, die authentische Daten testet. Diese Daten wurden so direkt von den verschiedenen Geräten im Lab abgerufen.

Falls weitere Funktionen implementiert werden sollten, dann soll an dieser Stelle auch ein entsprechender Test erstellt werden.

Natürlich gilt dies auch, falls eine neue Softwareversion hinzugefügt wird.

Abbildung 20: CollectorTest

7.3.2 Utility



Da die Utility-Kasse aus sehr vielen Klassen verwendet wird, ist es hier besonders wichtig, dass alle Funktionen ordnungsgemäss laufen.

Abbildung 21: UtilityTest

7.4 Metrik

Folgend ein paar Kennzahlen die unsere Software, bzw. die Funktionen unserer Anwendung in Zahlen abbildet.

7.4.1 Library (npvLib)

Number of Packages	10
Number of Classes	68
Number of Methodes	279
Total Lines of Code	4064

7.4.2 Website (npvWebApp)

Number of Packages	2
Number of Classes	2
Number of Methodes	53
Total Lines of Code	475

7.4.3 Gesamt

Number of Packages	12
Number of Classes	70
Number of Methodes	332
Total Lines of Code	4539

8. Projektmanagement

Wir haben uns im Team immer super verstanden und waren uns gegenseitig immer eine grosse Hilfe, wenn Fragen oder Probleme anstanden. Die Stunden wurden stets zeitnah eingetragen und zu Beginn des Projektes konnten wir uns auch gut an den Zeitplan halten.

Ab Meilenstein 2 in der Semesterwoche 08, bzw. Kalenderwoche 45 wurde die Applikation um die Funktionalität des Layer3 und um die Webseite erweitert. Da gerade das Layer3 die Komplexität des Algorithmus exorbitant ansteigen liess, hat der Projektplan leider nicht mehr perfekt mit unseren Ergebnissen übereingestimmt.

Wenn man zusätzlich noch beachtet, dass uns die ersten zwei Wochen verloren gingen, weil das Kick-Off-Meeting leider erst in der Mitte der zweiten Woche stattfand wurde die Zeit extrem knapp.

Zur Erinnerung hier noch einmal dargestellt, was theoretisch für die Semesterarbeit aufgewendet werden sollte.

Zeit pro Woche pro Mitglied:	16 Stunden
Zeit pro Woche im Team:	32 Stunden
Zeit komplett pro Mitglied:	224 Stunden
Zeit komplett im Team:	448 Stunden

Unten nachfolgend nun unser effektiver Aufwand.

Mitglied	2012-38	2012-39	2012-40	2012-41	2012-42	2012-43	2012-44	2012-45	2012-46	2012-47	2012-48	2012-49	2012-50	2012-51	Gesamtzahl
Reto Gsell		4.00	12.50	10.50	19.50	11.00	14.00	13.00	11.00	24.00	39.00	31.00	60.00	55.00	304.50
André Ulrich	3.00	9.00	12.50	11.00	14.00	18.00	16.00	15.00	13.00	18.00	37.00	31.00	64.00	55.00	316.50
Gesamtzahl	3.00	13.00	25.00	21.50	33.50	29.00	30.00	28.00	24.00	42.00	76.00	62.00	124.00	110.00	621.00

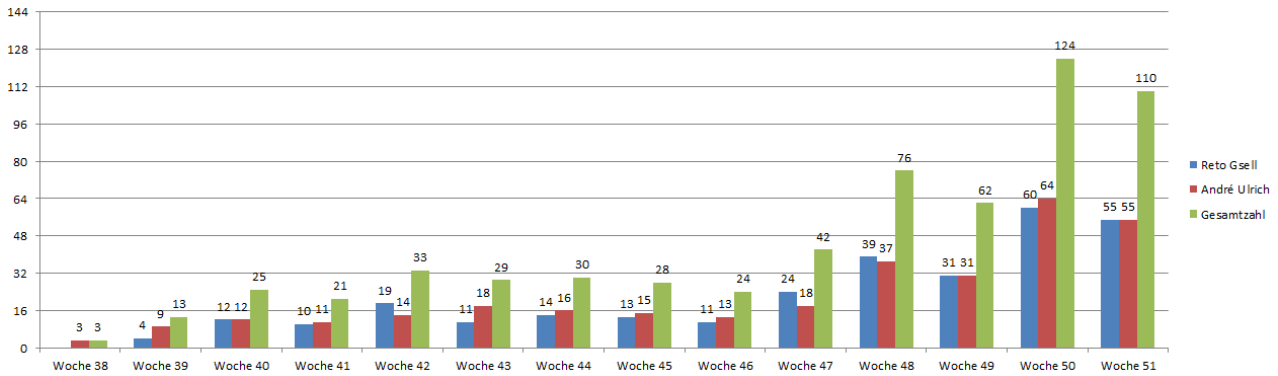


Abbildung 22: Redmine Auswertung Zeitbedarf

Mit 621 Stunden gegenüber den 448 angedachten Stunden sind wir also 173 Stunden über der Sollvorgabe. Das entspricht einem Mehraufwand von 38.6%. Wenn man zusätzlich noch beachtet, dass leider nicht alle geforderten Funktionen komplett implementiert werden konnten, zeigt uns dies, dass ein drittes Projektmitglied sicher eine gute Sache gewesen wäre.

9. Projektstand

Wir haben sehr vieles erreicht, aber leider nicht ganz so viel, wie wir uns erhofft hatten. Es wurde von Beginn weg erklärt, dass die Applikation die in den Anforderungen erklärten Funktionen bieten soll, dies aber nicht komplett in einer Semesterarbeit zu schaffen ist. Trotzdem hätten wir uns erhofft, dass wir es schaffen, alles zu implementieren, was geplant war.

Folgend nun eine Übersicht der grössten Errungenschaften und nachfolgend die ausstehenden, bzw. wünschenswerten Funktionalitäten.

Was funktioniert:

- Telnet-Zugriff auf Geräte
- SSH SSH2-Zugiff auf Geräte
- Cisco-Ios und Cisco-Nexus-Geräte werden unterstützt
- Sammlung der folgenden Daten von einem Netzwerkgerät
  - ArpCollector — ArpTable
  - EtherChannelCollector — EtherChannelListe
  - InterfaceCollector — Interfaceauflistung
  - NeighborCollector — cdp-Neighbor-Informationen
  - RoutingCollector — RoutingTable
  - SwitchingCollector — Mac-Address-Table
  - TraceRouteCollector — Traceroute-Output
  - VersionCollector — Versionsinformationen
  - VrrpCollector — VRRP- Konfiguration
- Interaktion über die Webseite
- Default-Werte definieren
- Ausgabe der gesammelten Daten auf der Webseite
- Grafische Darstellung des Pfades auf der Webseite
- Eine Pfadberechnung von Gerät A nach Gerät B unter Eingabe von IP-Adressen
- Eine Pfadberechnung von Gerät A nach Gerät B unter Eingabe von MAC-Adressen

Was steht noch aus / Was hätten wir gerne implementiert:

- IPv6
- Weitere Collectoren
  - MPLS
  - vPC
  - Fabric
- AccessListen beachten
- Speicherung von Logindaten bei erneutem Aufruf einer Visualisierung
- Visualisierung (Grafik) verbessern
  - Loadbalancing / Backuppfade, usw. in verschiedenen Farben zeichnen

Abschliessend sind wir der Meinung, dass wir ein sehr gutes Fundament gelegt haben, um die Idee eines Network-Path-Visualizers zu realisieren. Eine weitere Arbeit an dieser Applikation könnte daraus eine wirklich brauchbare Applikation hervorbringen. Leider ist dies momentan erst begrenzt möglich.

10. Erfahrungsbericht

10.1 André Ulrich

Ich habe bei dieser Arbeit sehr viel gelernt und bin froh, dass ich eine Arbeit finden konnte, die im Netzwerkbereich angesiedelt ist. Gerade dieses netzwerktechnische Wissen wird leider an der HSR zu knapp unterrichtet und konnte durch diese Arbeit gestärkt werden. Das Gelernte werde ich sicher auch in meinem späteren Arbeitsleben gut einsetzen können.

Die Arbeit am Projekt selber und auch die Zusammenarbeit mit meinem Projektpartner Reto Gsell sowie den Betreuern Beat Stettler und Rolf Schärer hat mir sehr zugesagt. Wir haben uns auf Anhieb gut verstanden und gut miteinander zusammengearbeitet. Einzig die Terminfindung mit den Betreuern hat sich einige Male etwas schwierig gestaltet, doch wurde immer rasch eine Alternative gefunden. Sehr positiv überrascht hat mich die Flexibilität und Kompetenz von Rolf, der uns jederzeit nützliche Tipps und Erklärungen liefern konnte.

Folgende zwei Punkte haben mein sonst sehr positives Erlebnis hauptsächlich negativ beeinflusst. Zum einen war es der doch beträchtliche Mehraufwand, weshalb wir in den letzten zwei Wochen komplett auf Vorlesungen und Übungen anderer, auch wichtiger Fächer, verzichten mussten. Auf der anderen Seite der Fakt, dass wir trotz des geleisteten Mehraufwandes nicht alle Ziele erreichen konnten. Auch wenn dies zum Teil bereits vor Beginn der Arbeit klar war, so ist es doch enttäuschend.

Aus implementationstechnischer Sicht haben uns vor allem die folgenden zwei Knackpunkte aus dem Zeitplan geworfen: Erstens wurde es notwendig, den gesamten Algorithmus neu zu entwickeln, als wir das Layer 3 im Algorithmus implementierten, was so nicht zu erwarten war. Zweitens waren wir durch die Inkompatibilität von SSH und SSH2 gezwungen, verschiedene Libraries zu verwenden, was uns übermässig viel Zeit gekostet hat.

Nichtsdestotrotz bin ich mit meiner Leistung und meinem Engagement sehr zufrieden. Wir haben in nur 14 Wochen eine Anwendung entwickelt, die effektiv einen Nutzen hat und in Vollendung ein tolles Instrument sein wird. Speziell gut hat mir gefallen, dass ich zum ersten Mal etwas implementieren durfte, wo ich den Nutzen ganz klar erkennen kann und auch die Idee dahinter für sinnvoll und spannend erachte. Das hat meine Motivation sicherlich enorm gefördert.

10.2 Reto Gsell

Die Arbeit an diesem Projekt hat mir im Grossen und Ganzen gut gefallen. Dabei habe ich sehr viel Neues dazu gelernt. Ich finde die Kombination von Netzwerkbereich und Software sehr interessant. Viele Grundlagen aus dem Netzwerkbereich wurden mit dieser Arbeit vertieft und dies hat mir viel neues Wissen gebracht. Auch der Kontakt mit neuen Webtechnologien war interessant und ich konnte Stoff aus dem Unterricht direkt anwenden.

Die Zusammenarbeit mit meinem Projektpartner André Ulrich war sehr gut, was als gute Grundlage für das Projekt notwendig war. Die Kommunikation innerhalb des Projektteams war sehr gut, Probleme konnten schnell und effizient abgearbeitet werden. Die Betreuung der Arbeit durch Beat Stettler und Rolf Schärer war kompetent und hat mir sehr gut gefallen. Insbesondere die Flexibilität von Rolf Schärer hat mir sehr gut gefallen. Wir konnten bei Problemen jederzeit auf sein fundiertes Wissen im Netzwerkbereich zurückgreifen.

Wie schon erwähnt hat mir das Projekt im Grossen und Ganzen gut gefallen. Der Rahmen des Projektes war aber meiner Meinung nach von Anfang an etwas hoch gesteckt, vielleicht etwas zu hoch. Wir mussten darum einiges an Mehraufwand betreiben. Schlussendlich konnten wir aber trotzdem nicht alle geforderten Funktionalitäten korrekt implementieren.

Das Projekt an sich war sehr interessant, da man mit verschiedensten Technologien in Kontakt gekommen ist. Dies hat aber die Komplexität der Arbeit erhöht, da wir uns in die verschiedenen Technologien einarbeiten mussten und die Applikation für jeden Bereich anpassen mussten. Vor allem die Implementation des Algorithmus hat sich recht schwierig gestaltet, da die Abbildung von Router- und Switch-Logik in der Software recht komplex ist. In unserer Arbeit mussten wir uns zudem in verschiedenste Java-Libraries einarbeiten und diese zum Teil für unsere Zwecke anpassen, was uns viel Zeit gekostet hat.

Die Arbeit hat mir aber trotzdem Spass gemacht und wir haben trotz einiger Schwierigkeiten mit der Implementation einen guten Einsatz geleistet. Ich kann mir gut vorstellen, dass das Tool effektiv eingesetzt werden kann, falls es weiterentwickelt wird.

11. Glossar

Wort	Bedeutung
Poll	Ein Ansatz, bei dem in einem bestimmten Zeitintervall eine Aktion ausgeführt wird. Meist eine Abfrage eines Wertes vom Server
Push	Dieser Ansatz löst das Problem, dass ein Polling auch statt finde, wenn keine Änderungen vorhanden sind. Hier wird der Client bei einer Änderung direkt informiert und je nach dem auch gleich die Daten angehängt.

12. Abbildungsverzeichnis