HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

IFS INSTITUTE FOR SOFTWARE

# ToffiAnalyser

## Visualisierung und Analyse von NetFlow-Daten

# Studienarbeit

Abteilung Informatik

HSR Hochschule für Technik Rapperswil

Herbstsemester 2016


| | |
|---|---|
| Autor(en): | Matthias Gabriel und Philip Schmid |
| Betreuer: | Prof. Stefan Keller |
| Projektpartner: | - |

Revision 1, Dezember 2016

Bezugsquelle: https://eprints.hsr.ch

HSR Hochschule für Technik Rapperswil

Oberseestrasse 10

8640 Rapperswil

Schweiz



[1] https://creativecommons.org/licenses/by-nc-sa/3.0/ch/legalcode.de

# Abstract

Heute gibt es in der IT oft das Problem, dass nicht genau bekannt ist, was für Netzwerkverkehr im firmeninternen Netzwerk existiert. Das Firmennetzwerk stellt deshalb für viele IT Mitarbeiter eine Blackbox dar. Durch die entwickelte Webapplikation soll diese Situation verbessert werden, die aktuell in vielen IT Abteilungen vorhanden ist, indem dem Benutzer eine neue Darstellung des Netzwerkverkehrs geboten wird. Um diese verbindungsspezifischen Daten zu sammeln, existiert bei den Geräten der Firma Cisco die Funktionalität «NetFlow», welche die benötigten Daten direkt auf den Netzwerkkomponenten sammelt.

Im Gegensatz zu bereits existierenden Lösungen zur Analyse von NetFlow-Daten, sollen diese in vorliegender Webapplikation mittels Graphen dargestellt werden. Damit kann erreicht werden, dass der Benutzer eine völlig neue Ansicht auf die Daten bekommt und dadurch neue Zusammenhänge erkennen kann. Als separater Teil des Projektes wurde zudem ein ausführlicher Datenbank-Benchmark durchgeführt, welcher die drei Datenbank-Management-Systeme PostgreSQL, Neo4j und OrientDB unter der Verwendung von anwendungsspezifischen (Graph-)Daten und Abfragen vergleicht. Das Spezielle an diesem Benchmark ist der Vergleich von Graphdatenbanken (Neo4j und OrientDB) mit einer relationalen Datenbank (PostgreSQL).

Als Resultat des Projektes wurde eine funktionsfähige Webapplikation mit dem clientseitigen Framework Angular sowie dem serverseitigen Framework Spring Boot erstellt. Die Anwendung bietet eine gute Balance zwischen Übersichtlichkeit über die zu analysierenden Daten und dem Detaillierungsgrad. Der Graph-Daten-Benchmark hat hauptsächlich zwei Erkenntnisse gebracht: Bei Änderungsoperationen und «normalen» Abfragen waren alle drei Systeme vergleichbar mit leichtem Vorteil für PostgreSQL. Bei den Abfragen auf Graphen (Traversierung), die hier oft gebraucht werden, war die Performance bei den Graphdatenbanken bis zu 50% besser.

# Abstract Englisch

In today's IT-world it's often unknown what network traffic exists in the company's own network. The computer network therefore poses a black box for many employees.

Cisco offers NetFlow, a feature that collects the connection specific data directly from the network components. Our newly developed web application, the ToffiAnalyser, tries to minimise this problem through a new kind of visualisation of this data.

Contrary to already existing solutions, the ToffiAnalyser displays the NetFlow data as a graph. This provides a completely new way for the user to view the data and recognise correlations in it.

The project includes an extensive database benchmark, which compared three database management systems, PostgreSQL, Neo4j and OrientDB, using application specific data and queries. The peculiarity is the comparison of the graph databases, Neo4j and OrientDB, to the relational database PostgreSQL.

The ToffiAnalyser is a working web application balancing overview and a sufficient level of detail. It uses the client-side framework Angular as well as the server-side framework Spring which allowed for a fast development. The benchmark concludes that graph databases grant a great performance boost, when using graph-traversals, but using «normal» queries PostgreSQL is still a bit faster.

# Management Summary

## Ausgangslage

Heute ist in der IT oft nicht genau bekannt, welche Art von Netzwerkverkehr im firmeninternen Netzwerk existiert. Das Firmennetzwerk stellt deshalb für viele IT Mitarbeiter eine Blackbox dar. Mit dem immer mehr aufkommenden Trend des Software-Defined-Networking (SDN), verschlechtert sich diese Situation sogar noch. Cisco bietet die Funktionalität NetFlow an, welche verbindungsspezifische Daten direkt von den Netzwerkkomponenten sammelt und zu Analysezwecken benutzt werden können.

Unsere neu entwickelte Webapplikation, der ToffiAnalyser, versucht durch eine neue Art der Darstellung dieser Daten, diese Situation zu verbessern und den Mitarbeitern einen Einblick in Ihr Netzwerk zu geben. Es existieren zwar bereits diverse Tools von namhaften Herstellern, welche eine Darstellung des Netzwerkverkehrs mittels diverser Grafiken bieten, der ToffiAnalyser setzt jedoch auf die Darstellung der Netzwerkinformationen als Graph.

Im Hauptteil unserer Arbeit wollen wir eine Softwarelösung entwerfen, bei welcher die Netzwerkverbindungen als Elemente eines Graphen dargestellt werden. Wir haben die Vermutung, dass dem Benutzer ein besserer Überblick über die Verbindungen zwischen den Netzwerkteilnehmern ermöglicht wird.

Ein weiteres Ziel unserer Arbeit besteht darin, herauszufinden welches von drei ausgewählten Datenbank-Management-Systemen (PostgreSQL, Neo4j, OrientDB) sich am besten für die Speicherung der Netzwerkdaten anbieten würde.

## Vorgehen und Technologien

In den ersten Wochen haben wir uns vor allem mit dem Thema Benchmarking beschäftigt und einen eigenen Benchmark mit einigen Kandidaten entworfen und durchgeführt. Die Entscheidung einen eigenen Benchmark zu erstellen haben wir getroffen, da wir für unsere spezifischen Anforderungen leider keinen passenden Benchmark gefunden haben. Wir konnten uns aber beim Aufbau auf die vorhandenen Grundstrukturen stützen. Der Benchmark ist als eigenständige Java-Applikation geschrieben und die verwendeten Daten sind identisch mit denen der Hauptapplikation.

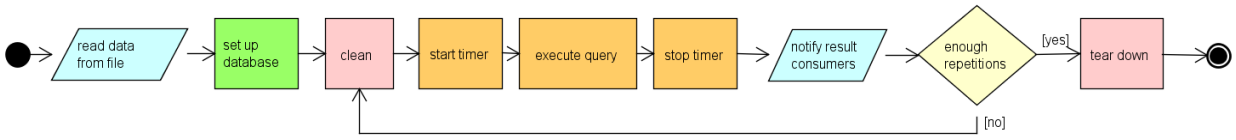


Abbildung 1-1 Ablauf einer Benchmarkabfrage

Um die benötigten Verbindungsdaten zu sammeln, existiert die bereits erwähnte Funktionalität NetFlow, welches auf grösseren Netzwerkgeräten von Cisco aktiviert werden kann. Das INS Institute for Networked Solutions der HSR hat uns die benötigten Daten freundlicherweise zur Verfügung gestellt.

Für die Softwarelösung haben wir uns für eine Webapplikation entschieden, welche auf der Clientseite mit Angular realisiert wurde. Serverseitig wurde auf Spring Boot gesetzt, welches die «convention over configuration»-Lösung des bekannten Spring Frameworks ist. Die Datenpersistierung erfolgt in Neo4j, einer der bekanntesten Graphdatenbanken.

# Ergebnisse

Als Resultate können wir einerseits einen spezifischen Benchmark mit ausführlicher Auswertung vorweisen, andererseits ist während des Projekts die gewünschte Softwarelösung entstanden. Der Benchmark vergleicht die Performance der drei Datenbanksysteme PostgreSQL, OrientDB sowie Neo4j aufgrund von diversen unterschiedlichen Datenbankabfragen. Eine Erweiterung der Benchmarksoftware, um weitere Datenbanksysteme oder Abfragen, wäre ohne weiteres möglich.

Der Vergleich brachte zwei wesentliche Erkenntnisse: Bei Änderungsoperationen und «normalen» Abfragen waren alle drei Systeme vergleichbar mit leichtem Vorteil für PostgreSQL. Bei den Abfragen auf Graphen (Traversierung), die im ToffiAnalyser oft gebraucht werden, war die Performance bei den Graphdatenbanken bis zu 50% besser.

Unsere entwickelte Softwarelösung umfasst mehrere verschiedene Ansichten, um die Netzwerkverbindungen zu visualisieren. Der grösste Unterschied der ist der unterschiedlich gewählte Detaillierungsgrad. Durch die verschiedenen Visualisierungen der gleichen Daten ist es möglich diese unter verschiedenen Aspekten zu analysieren.

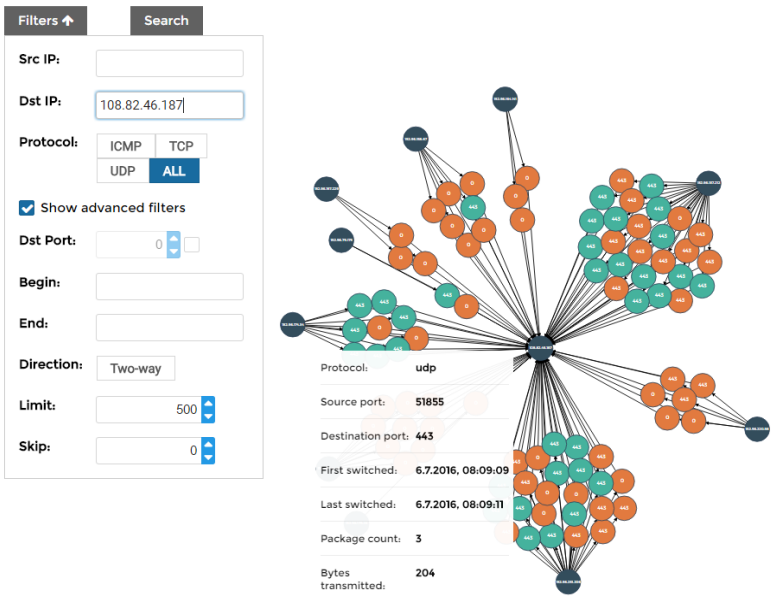


Abbildung 2-1 NetFlow Ansicht mit Tooltip

# Ausblick

Die im Laufe des Projektes entwickelte Software kann als Grundlage für eine komplette Netzwerkvisualisierungslösung gesehen werden. Neben den von uns entwickelten Ansichten besteht noch ein grosser Spielraum für Verbesserungen und Erweiterungen. In unseren Augen bietet sich insbesondere eine Kombination von klassischen Diagrammen und den spezifischen Graph-Ansichten an, um eine gute Balance zwischen Übersicht und Detaillierungsgrad zu erreichen. Ebenfalls als sehr wichtigen Erweiterungspunkt sehen wir den automatischen Import der NetFlow-Daten über eine Programmierschnittstelle. In einem längerfristigen Ausblick sehen wir vor allem das Potential für eine automatisierte Erkennung von auffälligen und somit potentiell gefährlichen Netzwerkverbindungen, sowie die Benachrichtigung der betreffenden Personen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Aufgabenstellung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

### Network Flow Analyzer

- Studienarbeit im Herbstsemester 2016/2017
- Autor/in: Matthias Gabriel und Philip Schmid
- Betreuer: Prof. Stefan Keller, Institut für Software, HSR
- Industriepartner: -

### Ausgangslage und Aufgabenstellung

Der Umfang dieser Semesterarbeit ist in zwei unterschiedliche grosse Teile unterteilt. Einerseits wird es im ersten und zugleich kleineren Teil darum gehen herauszufinden, wie das NoSQL Graph-Datenbanksystem Neo4J gegenüber den Datenbanksystemen PostgreSQL und OrientDB performt. Insbesondere sollten für diesen Benchmark explizit NetFlow-Testdaten verwendet werden, da diese von der Struktur her genau auf die von NoSQL Graph-Datenbanksystemen-Hersteller beworbenen Vorteile abzielen.

Beim zweiten und grösseren Teil geht es darum, eine Softwarelösung zu entwickeln, welche es erlaubt Netzwerkverbindungsdaten zeitnah analysieren, filtern, auswerten und visuell darstellen zu können. Als Basisfunktionalität dient dafür das NetFlow-Feature von Cisco, welches auf fast allen Cisco-Routern und auch einigen leistungsstarken Switches zur Verfügung steht. NetFlow erfasst dabei sogenannte NetFlows, welche grundsätzlich eine unidirektionale Netzwerkverbindung zwischen zwei Endpunkten abbilden.

Die Daten werden von den Netzwerkgeräten erfasst und an ein zentrales System geschickt. Das zentrale System kann in einem ersten Schritt eine Art von Logserver sein, welcher die NetFlow Daten in einem File abspeichert. Anschliessend können die Daten aus dem File manuell in die Datenbank importiert werden. Zu einem späteren, optionalen Schritt wäre es hier denkbar, dass die NetFlow Daten direkt an ein API des Network Flow Analyzers geschickt werden.

Die Verbindungsdaten sollen in einer Neo4J Graph-Datenbank abgelegt werden. Diese Daten können dann entsprechend ausgewertet und visualisiert werden. Neben einer Darstellung auf einer Webseite, wäre es eine Option, regelmässige Reports zu generieren. Hierzu kann ein Reporting Framework wie beispielsweise Jasper eingesetzt werden.

Einsatzzwecke für den Network Flow Analyzer könnten unter anderem Folgende sein:

- Eine Firma weiss häufig nicht, was genau für Netzwerkverbindungen in Ihrem Netzwerk stattfinden. Gerade auch im schnell aufkommenden Software Defined Networking (SDN) ist es teilweise extrem undurchschaubar, welche Verbindungen nun genau innerhalb des Netzwerkes existieren. Hier kann der Network Flow Analyzer für mehr Überblick sorgen.
- Die Auslastung des Netzwerkes oder spezifischer Services (Ports) soll ersichtlich werden, um somit Engpässe eruieren zu können.
- Firewall Administratoren stehen häufig vor dem Problem, dass sie nicht genau wissen, was für Verbindungen für einen bestimmten Host freigeschaltet/blockiert werden müssen. In einem solchen Fall könnte der Firewall Administrator den Network Flow Analyzer zur Hilfe nehmen, um zu evaluieren, welche Regeln er noch auf der Firewall ergänzen oder eventuell auch blockieren sollte.

### Lieferobjekte

1. Dokumentation, inkl. Textabstract (zusätzlich englisch), Management Summary (deutsch), technischer Bericht und Software Engineering-Projekt (deutsch); Anhänge (Literaturverzeichnis, CD-Inhalt).
2. Evaluation der Visualisierungs-Library

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

3. Die vom Studiengang geforderten bzw. empfohlenen Lieferobjekte: Broschüren-Abstract, Poster (nur digital), kein Kurzvideo.
4. Software Code (englisch).
5. Daten.

### Vorgaben/Rahmenbedingungen

SW-Infrastruktur mit kontinuierlichem Testen; optional/empfohlen: Docker.

Vorgehen und Arbeitsweise: Die Studierenden wählen nach Rücksprache ein Vorgehensmodell zur Softwareentwicklung. Es gibt wöchentliche Meetings mit vorbereiteten Unterlagen; wobei Ausnahmen vereinbart werden können.

### Dokumentation

Die Dokumentation ist auf Deutsch geschrieben wo nicht anders vermerkt und ist in den Lieferobjekten erwähnt.

Weitere Angaben:

- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind (einheitliche Nummerierung).
- Die Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen (nicht ausschliesslich Wikipedia-Links auflisten).
- Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung, Eigenständigkeitserklärung, Nutzungsrechte) gemäss Vorgaben des Studiengangs und Absprache mit dem Betreuer.

Form der Dokumentation:

- Bericht gebunden (1 Ex.), inkl. einer beschrifteten CD (plus 1 Ex. für den Studiengang).
- Alle Dokumente und Quellen der erstellten Software auf CDs.

### Bewertung

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Studienarbeit des Studiengangs Informatik mit besonderem Gewicht auf moderne Softwareentwicklung wie folgt:

- Projektorganisation (Gewichtung ca. 1/5)
- Bericht, Gliederung, Sprache (Gewichtung ca. 1/5)
- Inhalt inkl. Code (Gewichtung ca. 2/5)
- Gesamteindruck inkl. Kommunikation mit Industriepartner (Gewichtung ca. 1/5). Ein wichtiger Bestandteil der Arbeit ist, dass eine lauffähige, getestete Software abgeliefert wird (inkl. getesteter Installationsanleitung).

### Weitere Beteiligte

Mitarbeiter vom Institut INS der HSR.

# I. Technischer Bericht

# 1. Einführung

## 1.1 Problemstellung und Vision

Heute ist in der IT oft nicht genau bekannt, welche Art von Netzwerkverkehr im firmeninternen Netzwerk existiert. Das Firmennetzwerk stellt deshalb für viele IT Mitarbeiter eine Blackbox dar. Mit dem immer mehr aufkommenden Trend des Software-Defined-Networking (SDN), verschlechtert sich diese Situation sogar noch. Oft steckt dann für viele Betrachter noch mehr Magie im Netzwerk, da nicht genau bekannt ist, was der SDN-Controller auf den SDN-Komponenten konfiguriert hat.

Cisco bietet die Funktionalität NetFlow an, welche verbindungsspezifische Daten direkt von den Netzwerkkomponenten sammelt und zu Analysezwecken benutzt werden können.

Unsere neu entwickelte Webapplikation, der ToffiAnalyser, versucht durch eine neue Art der Darstellung dieser Daten, diese Situation zu verbessern und den Mitarbeitern einen Einblick in Ihr Netzwerk zu geben. Der ToffiAnalyser unterstützt somit die IT-Mitarbeitenden, ihre Arbeit besser zu meistern indem Sie einen besseren Überblick über das Netzwerk haben.

## 1.2 Ziele und Unterziele

Das Hauptziel des ToffiAnalysers ist es, dass IT Mitarbeiter über ein Web-Interface in der Lage sind, den Netzwerkverkehr als Graphen anzeigen zu lassen. Somit haben Sie eine Möglichkeit besser zu verstehen, welcher Netzwerkverkehr in dem für sie interessanten Netzwerkteil vorhanden ist. Wir haben dafür folgende Unterziele.

- Möglichkeit zu einer Gesamt-Übersicht über das Netzwerk
- Möglichkeit den Netzwerkverkehr von spezifischen Netzknoten genau zu analysieren
- Bereitstellung von Filtermöglichkeiten
- Optional: Eine Reporting-Funktionalität zur Verfügung stellen
- Optional: Möglichkeit den nächsten Netzwerkknoten eines Flows zu sehen

## 1.3 Rahmenbedingungen

Die benötigten Daten wurden uns freundlicherweise vom INS Institute for Networked Solutions der HSR zur Verfügung gestellt. Diese wurde mittels NetFlow von einer zentralen Netzwerkkomponente erfasst und an uns weitergeleitet. Das Format dieser gelieferten Daten ist dabei von Cisco definiert. Genauere Informationen dazu finden sich im Kapitel IV 4 NetFlow Konfiguration.

Des Weiteren wurde der Wunsch geäussert, dass die Graphdatenbank «Neo4j» für die Persistierung der Daten verwendet werden soll.

Wir haben uns nach Rücksprache mit unserem Betreuer «Prof. Stefan Keller» entschieden im Verlaufe der Arbeit einen Datenbank-Benchmark für die Persistierung unserer Daten in verschiedenen Datenbanksystem durchzuführen. Die Ergebnisse werden jedoch keinen direkten Einfluss auf die Implementierung des Projektes haben. Mehr zum Thema Benchmark im Abschnitt III.

## 1.4 Aufbau der Arbeit

Grundsätzlich ist unsere Arbeit in zwei unabhängige Teile unterteilt. In einem ersten Teil wird ein Datenbank-Benchmark durchgeführt. In einem zweiten Teil wird das ToffiAnalyser-Projekt realisiert. Die beiden Teile werden teilweise mit zeitlichen Überschneidungen bearbeitet.

Der detaillierte Aufbau der Arbeit kann im Kapitel II 7 Projektmanagement nachgelesen werden.

# 2. Stand der Technik

## 2.1 Bestehende Lösungsansätze

Da NetFlow, welches von Cisco spezifiziert ist, relativ breit eingesetzt wird, gibt es hierfür natürlich auch diverse Auswertungsmöglichkeiten. Es gibt Softwarelösungen von verschiedensten grossen und kleinen Herstellern wie Paessler oder Nagios. Im nächsten Kapitel gehen wir auf einige dieser Produkte kurz ein und im Kapitel I 2.3 Defizite werden kurz darlegen, was uns an all diesen Lösungen gefehlt hat.

## 2.2 Beschreibung

Da es zu viele verschiedene Software-Lösungen zur Netzwerkanalyse gibt, haben wir uns hier auf einige wenige Beispiele von bekannten Herstellern bezogen. Diese haben einen ähnlichen Nutzen für den User, wie der ToffiAnalyser. Wir haben von jeder Lösung zwei Screenshots eingefügt, welche einmal eine Übersicht über das Netzwerk und einmal eine Detailansicht beinhalten. Die Art der Lizenzierung sowie die allfälligen Lizenzkosten haben wir ausser Acht gelassen, da die Lizenzierungsarten sehr vielfältig und undurchsichtig sind.

### 2.2.1 All-in-One NetFlow Analyzer & Monitoring von Paessler

Paessler hat sich mit dem PRTG einen guten Namen im Network-Monitoring-Bereich erarbeitet. Die «all-in-one NetFlow Analyzer & Monitoring Software» ist dabei lediglich ein kleiner Teil davon und wird vom Hersteller als zusätzlicher Sensortyp beschrieben.

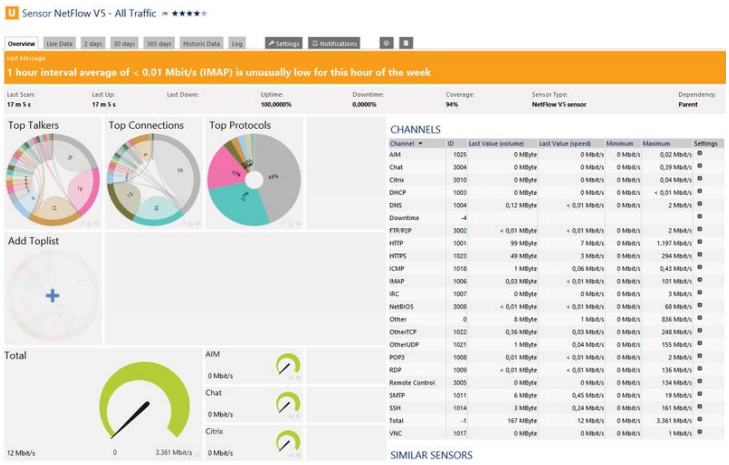


Abbildung 2-1 PRTG Übersicht[2]

[2] (Paessler, kein Datum)

Die Übersicht ist unterteilt in verschiedene Teile. Auf der linken Seite eine mehrheitlich graphische Zusammenfassung der vergangenen Flows sowie auf der rechten Seite eine etwas detailliertere Auflistung des Netzwerkverkehrs nach Protokoll. Generell finden wir die Ansicht gleichzeitig sehr übersichtlich und trotzdem informativ.

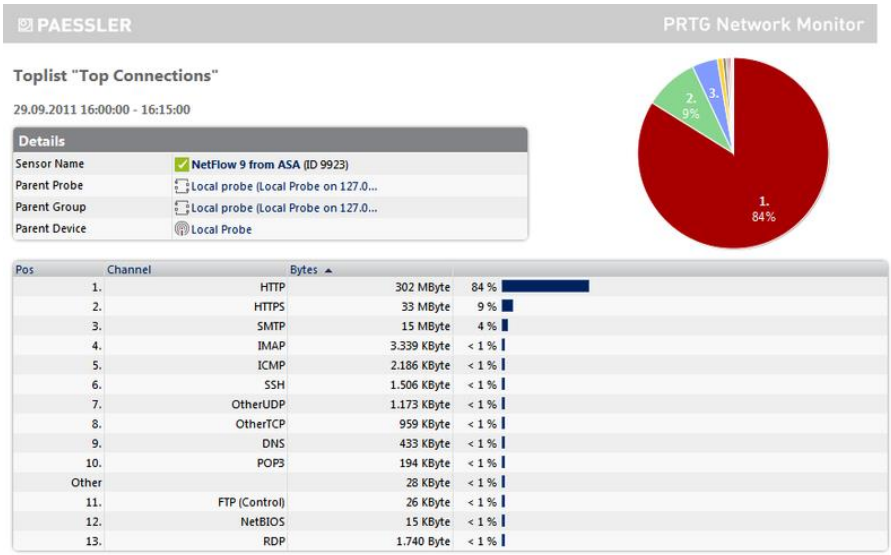


Abbildung 2-2 PRTG Detailansicht[3]

Die Detail-Ansicht ist wiederum unterteilt in einen kleineren graphischen Teil in Form eines Kreisdiagramms sowie einer genauen tabellarischen Auflistung des Verkehrs.

### 2.2.2 Nagios Network Analyzer

Auch Nagios hat sich im Bereich Monitoring-Lösungen zu einem der führenden Anbietern entwickelt. Nagios bietet für den Netzwerkbereich den «Nagios Network Analyzer» an, welcher unter anderem Auswertungen von NetFlow-Daten durchführt und visualisiert.

[3] (Paessler, kein Datum)

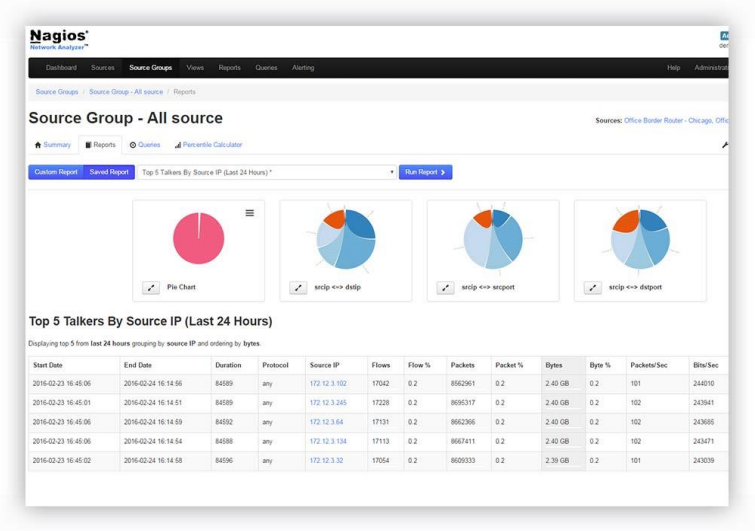


Abbildung 2-3 Nagios Übersicht[4]

In der Übersicht von Nagios wird wiederum mit Kreisdiagrammen gearbeitet, um dem User einen ersten Gesamtüberblick zu verschaffen. Darunter ist wie auch bei PTRG eine etwas detailliertere Auflistung des Netzwerkverkehrs zu sehen.

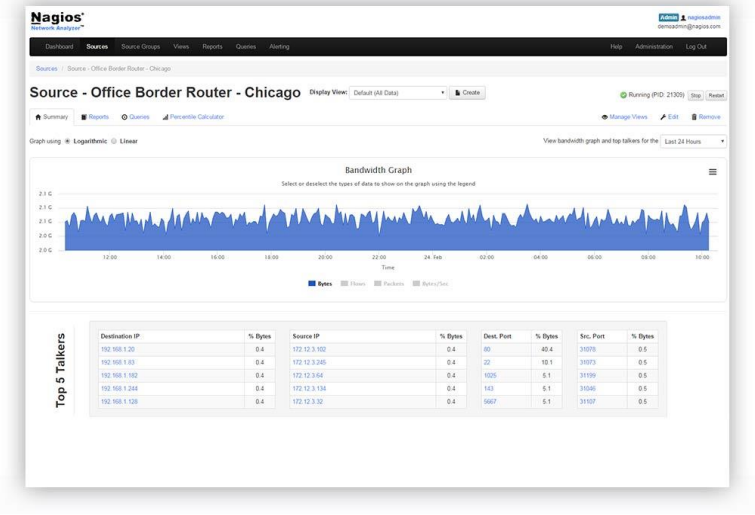


Abbildung 2-4 Nagios Detailansicht[5]

In der Detailansicht wird im Gegensatz zu PRTG mit einem zeitbasierten Graphen gearbeitet, um den laufenden Netzwerkverkehr darzustellen.

[4] (Nagios, kein Datum)

[5] (Nagios, kein Datum)

### 2.2.3 ntopng von ntop

Neben bekannten kommerziellen Anbietern von Monitoring-Software gibt es noch viele andere Software-Firmen und Projekte, welche sich mit der Analyse von NetFlow-Daten auseinandergesetzt haben. Wir haben uns hier ntopng von ntop genauer angeschaut, da sowohl Cisco[6] als auch pcwdld.com[7] ntop erwähnt haben. pcwdld.com sagt Folgendes über ntop: «*Probably the most well-known open source traffic analyzers, Ntop, […]*»[8]. ntopng ist wie auch die beiden anderen erwähnten Produkte eine volle Netzwerküberwachungslösung. Für ntopng existiert der Zusatz nProbe, welches die Sammlung und Weiterleitung von NetFlow-Datensätzen an das Monitoringsystem übernimmt.

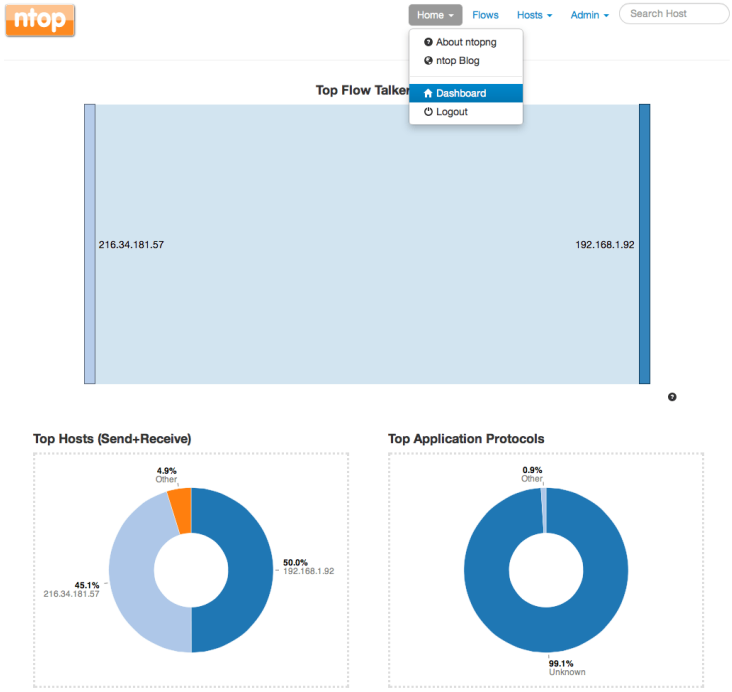


Abbildung 2-5 ntop Übersicht[9]

Im sogenannten Dashboard von ntopng ist eine Übersicht über das Netzwerk aufgeführt. Unter anderem sieht man darin, welche Hosts am meisten kommuniziert haben und welche Protokolle benutzt werden.

[6] (Cisco, kein Datum)

[7] (pcwdld.com, kein Datum)

[8] (pcwdld.com, kein Datum)

[9] (ntop, kein Datum)

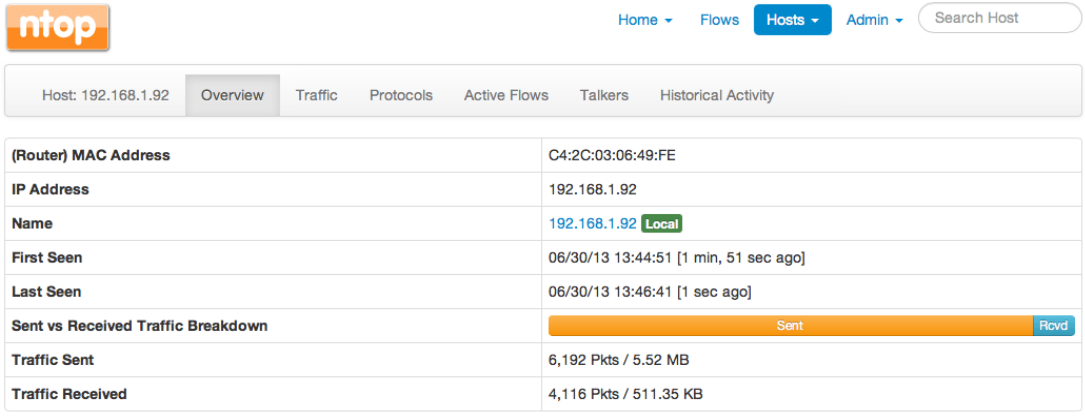


Abbildung 2-6 ntop Detailansicht[10]

In der sehr detaillierten Ansicht eines bestimmten Hosts werden alle Informationen dargestellt, welche diesen betreffen. Unter anderem wird angezeigt, wieviel Netzwerkverkehr produziert wurde.

## 2.3 Defizite

Obwohl uns alle diese Produkte einen guten Überblick über das Netzwerk geben hat uns bei allen angeschauten Produkten eine Visualisierung als Graph von Netzwerkknoten gefehlt. Wir vermuten, dass der Benutzer in einem Graphen ganz andere Schlussfolgerungen ziehen kann als mit Hilfe der Darstellung als klassisches Diagramm.

[10] (ntop, kein Datum)

# 3. Evaluation Visualisierungslibrary

Um die Visualisierungen im ToffiAnalyser vorzunehmen benötigen wir eine clientseitige Visualisierungslibrary, welche uns bei dieser Aufgabe unterstützt. Da es hierfür verschiedenste Kandidaten gibt, haben wir uns für eine kleine Gegenüberstellung entschieden. Dafür haben wir die vielversprechendsten Graph-Visualisierungslösungen angeschaut und eine Evaluationsmatrix erstellt.

## 3.1 Bewertungspunkte

Die Bewertungspunkte haben wir so gewählt, dass diese einen Einfluss auf unser Projekt haben und möglichst gut beurteilbar sind. Dabei haben wir vier Hauptkategorien gebildet und diese unterschiedlich gewichtet.

### 3.1.1 Community

In dieser Hauptkategorie bewerten wir verschiedene Aspekte der Community, welche sich um die Visualisierungslösung gebildet hat. Dabei war uns vor allem die Aktivität der Maintainer wichtig, weil dies viel über die Pflege des Produktes sowie dessen Zukunftstauglichkeit aussagt. Ebenfalls haben wir uns dafür entschieden die Grösse der Community in die Evaluationsmatrix aufzunehmen, obwohl dies relativ schwer zu beurteilen ist.

### 3.1.2 Unterlagen

In der Kategorie «Unterlagen» gehen wir auf die gebotene Unterstützung ein, sei es beim Einstieg in das Produkt als auch in weiterführenden Unterlagen. Dazu zählen sowohl Informationen, die normalerweise von der Community stammen, als auch solche, die vom Herausgeber der Lösung angeboten werden. Der Umfang der Hilfestellungen ist relativ schwer abschätzbar; wir haben uns hier neben den verlinkten Unterlagen des Herstellers an Stackoverflow gehalten, weil hier für fast alle Projekte Hilfestellungen angeboten werden und dies unsere erste Anlaufstelle für technische Probleme jeglicher Art ist.

### 3.1.3 Komplexität

Unter der Komplexität der Software verstehen wir sowohl die vorhandenen Features und die Erweiterbarkeit der Lösung als auch die normalerweise damit einhergehende Implementierungskomplexität für den anzuwendenden Entwickler

### 3.1.4 Endbenutzer

Als letzte Hauptkategorie haben wir den Endbenutzer gewählt, dabei haben wir uns aber auf das «Look-and-feel» beschränkt. Wir hätten gerne noch die Performance als einen Bewertungspunkt

aufgenommen, haben nach kurzer Überlegung aber davon abgesehen, da für eine sinnvolle Aussage selbst Performancemessungen durchgeführt werden müssten.

## 3.2 Sigmajs

Bei Sigmajs handelt es sich um eine JavaScript-Library, welche ausschliesslich dazu dient, Graphen zu visualisieren. Sie benutzt dazu standardmässig entweder WebGL oder Canvas.

Der Code ist Opensource und steht auf Github unter der MIT Lizenz zur Verfügung.

Das Projekt besitzt eine relativ grosse Community und hat momentan fasst 50 Contributors. Der Code sieht relativ aktuell aus, der letzte Commit war im August, das Projekt besitzt aber verhältnismässig viele offene Issues.[11]

Es gibt für den Einstieg ein Tutorial, wie die einfachste Art von Graph ohne Modifikation erstellt werden kann. Leider haben wir auf der Webseite keine weiterführenden Tutorials gefunden. [12]

Die Dokumentation sieht auf den ersten Blick sehr sauber strukturiert und umfangreich aus, was vor allem im weiteren Verlauf des Projektes wichtig ist. Mit dabei sind auch Beispiele für einige Usecases der Entwickler selbst verlinkt.

Auf Stackoverflow findet man einige Hilfestellungen zu Problemen, jedoch hält sich die Anzahl in Grenzen.

## 3.3 D3.js

D3 steht für Data-Driven Documents, es ist eine JavaScript-Library, welche jegliche Art von Visualisierung von Daten unterstützt. Unter Anderem können damit auch Graphen dargestellt werden. Sie benutzt standardmässig SVG und ist deshalb bei grossen Graphen tendenziell weniger performant als Sigmajs, bietet aber einfachere Interaktionsmöglichkeiten.

Das D3 Projekt besitzt eine riesige Community und hat momentan über 100 Contributors. Die Maintainer arbeiten sehr aktiv am Code, der letzte Commit war vor wenigen Tagen. Die Issues scheinen, wenn möglich, gut bearbeitet zu werden, es sind viel weniger offene vorhanden und bei allen ist mindestens ein Kommentar vorhanden. [13]

Da die Library nicht auf Graphen spezialisiert ist, ist auf der Webseite selbst kein graph-spezifisches Tutorial zu finden. Es ist jedoch eine ganze Sammlung von Tutorials aus der Community verlinkt, in welcher auch mehrere Graph-Spezifische enthalten sind. Auch bei den Beispielen wird voll auf die Zusammenarbeit mit der Community gesetzt; so sind diese Projekte lediglich sortiert

[11] (jacomyal, 2016)

[12] (sigmajs.org, kein Datum)

[13] (mbostock, d3/d3: Bring data to life with SVG, Canvas and HTML., 2016)

und verlinkt. Deshalb ist hier die Auswahl an implementierten Beispielen viel grösser.[14] Die Dokumentation ist wiederum sehr sauber, und ist sicher viel grösser, was aber auch auf den grösseren Umfang des Projektes zurück zu führen ist.

Auf Stackoverflow findet man relativ viele Hilfestellungen zu Problemen.

## 3.4 Alchemy.js

Als erste Library haben wir uns Alchemy.js angeschaut, welche auf D3.js aufbaut und wiederum auf Graphen spezialisiert ist. Die Community für dieses Projekt ist leider so gut wie inexistent und der Haupt-Maintainer sucht einen Nachfolger. Der letzte Commit ist schon über ein Jahr alt.[15] Ein klassisches Tutorial ist auf der Seite nicht zu finden, jedoch gibt es mehrere sehr einfach gehaltene Beispiele, an welchen man sich orientieren kann. Zudem ist bei den wichtigsten Funktionen in der Dokumentation noch ein kleines Beispiel aufgeführt, was den Einstieg erleichtert, die Dokumentation aber im Vergleich unübersichtlich erscheinen lässt. Da das Projekt keine sehr grosse Community besitzt, findet sich auf Stackoverflow fast keine Hilfestellung.

[14] (pratapvardhan, 2016)

[15] (GraphAlchemist, 2016)

## 3.5 Schlussfolgerung

**Evaluation Matrix für die Visualisierung von ToffiAnalyser**

| Category Weight | Relative Weight | Category | sigmajs | D3.js | alchemy.js |
|---|---|---|---|---|---|
| **20** | | **Community** | | | |
| | *35* | Grösse der Community | 7 | 9 | 3 |
| | *65* | Aktivität der Maintainer | 7 | 10 | 1 |
| | *100* | *Subtotal* | 7.0 | 9.7 | 1.7 |
| | | | | | |
| **35** | | **Unterlagen** | | | |
| | *15* | Umfang der Tutorials | 5 | 5 | 5 |
| | *25* | Sauberkeit der Dokumentation | 8 | 8 | 6 |
| | *25* | Umfang der Dokumentation | 9 | 9 | 8 |
| | *15* | Umfang von Beispielen | 8 | 8 | 5 |
| | *15* | Umfang an Hilfestellungen zB stackoverflow | 7 | 9 | 4 |
| | *95* | *Subtotal* | 7.6 | 7.9 | 5.9 |
| | | | | | |
| **35** | | **Komplexität** | | | |
| | *25* | Einfachheit für den Entwickler | 9 | 6 | 9 |
| | *25* | verfügbare Features (graph) out of the box | 9 | 5 | 7 |
| | *20* | Erweiterbarkeit für Graph | 9 | 9 | 3 |
| | *10* | Erweiterbarkeit sonst | 0 | 10 | 0 |
| | *80* | *Subtotal* | 7.9 | 6.9 | 5.8 |
| | | | | | |
| **10** | | **Endbenutzer** | | | |
| | *100* | Look and Feel | 8 | 9 | 5 |
| | *100* | *Subtotal* | 8.0 | 9.0 | 5.0 |
| | | | | | |
| **100** | | **Overall Scores** | **76.3** | **80.4** | **49.2** |

Abbildung 3-1 Evaluationsmatrix der Visualisierungslibrary

Aufgrund der Evaluationsmatrix Abbildung 3-1 Evaluationsmatrix der Visualisierungslibrary haben wir uns für D3.js entschieden. Grundsätzlich haben sich Sigmajs und D3.js klar von alchemy.js abgehoben. Den Ausschlag zugunsten D3.js hat wohl die Grösse und Aktivität der Community gegeben. Zu beachten ist hier auch die Tatsache, dass mit Sigmajs lediglich Graphen dargestellt werden können, im Gegensatz zum D3.js mit welchem Visualisierungen jeglicher Art gemacht werden können.

Alchemy.js hat am Anfang aufgrund seiner kleineren Komplexität einen guten ersten Eindruck hinterlassen, bei genauerem Hinschauen wurde jedoch relativ schnell klar, dass auf eine andere Library zurückgegriffen werden muss.

# 4. Umsetzungskonzept

In diesem Kapitel wollen wir eine grobe und allgemein verständliche Übersicht über das Umsetzungskonzept von ToffiAnalyser geben. Weiterführende Informationen sind im Teil «II. SW-Projektdokumentation» nachzulesen. Die Funktionalität von ToffiAnalyser kann grundsätzlich in zwei logisch unterschiedliche Tätigkeiten unterteilt werden. Zum einen die Informationsbeschaffung mittels eines Imports und zum anderen die Visualisierung der gesammelten NetFlow-Daten. Wir wollen diese Unterscheidung im Umsetzungskonzept festhalten, obwohl in der Realisation sowohl der Import als auch die Visualisierung über die gleiche Webapplikation gemacht werden. Dies hat den Grund, dass grundsätzlich mehrere Importmöglichkeiten angedacht sind.

## 4.1 Grundsätzlicher Aufbau

Grundsätzlich besteht das ToffiAnalyser-Projekt aus drei verschiedenen Komponenten:

- ToffiAnalyser-Webapplikation
- ToffiAnalyser-Server
- Datenbank

Die ToffiAnalyser-Webapplikation ist die Schnittstelle zum Benutzer und wurde mit dem Angular2-Framework umgesetzt.

Der ToffiAnalyser-Server basiert auf Spring Boot und liefert die Webapplikation aus. Zudem bietet er eine Programmierschnittstelle an, um die relevanten Daten abzufragen oder zu manipulieren.

Die Datenbank ist in unserem Fall eine Graphdatenbank, namentlich Neo4j, weil sich darin unsere Daten gut abbilden lassen.

## 4.2 ToffiAnalyser-Import

Beim Teil ToffiAnalyser-Import geht es darum, die erhaltenen Daten in die Datenbank abzuspeichern. Momentan wird lediglich der manuelle Import von NetFlow-Daten über die Webapplikation angeboten. Es ist jedoch auch eine Programmierschnittstelle angedacht, mit welcher der automatische Import ermöglicht werden soll.

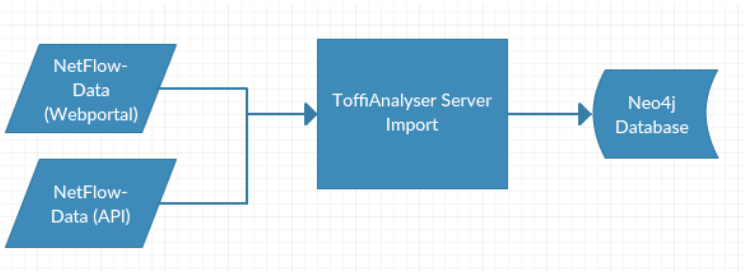


Abbildung 4-1 Umsetzungskonzept Import

Die NetFlow-Daten werden unabhängig der Quelle vom Server in die Datenbank gespeichert.

## 4.3 ToffiAnalyser-Visualisierung

In der ToffiAnalyser-Webapplication werden die in der Datenbank gespeicherten NetFlow-Datensätze visualisiert. Es gibt dafür mehrere verschiedene Ansichten auf die Daten, welche je einen Detailierungsgrad darstellen. Es gibt dabei momentan folgende Ansichten:

- Overview
- NetFlow-Overview
- NetFlow

Es gibt dabei folgende Grundfunktionalitäten, welche in jeder Ansicht verfügbar sind:

- Zoom
- Verschieben des kompletten Graphen
- Verschieben und fixieren eines einzelnen Nodes

Durch das Betätigen des Mausrades in der gesamten Graph-Ansicht ist es möglich tiefer in den Graphen zu zoomen oder einen besseren Überblick zu erhalten, indem in die Ferne gezoomt wird. Dem Zoom sind dabei bewusst in beide Seiten eine Grenze gesetzt, um den Benutzer nicht zu verwirren.

Durch «Drag-and-Drop» auf der weissen Fläche um den Graphen kann dieser komplett in eine beliebige Richtung verschoben werden.

Durch Klicken auf einen bestimmten Node wird dieser fixiert und kann beliebig auf der Fläche verschoben werden. Der Node wird zu Erkennungszwecken rot umrandet und kann durch einen Doppelklick wieder gelöst werden.

In der Ansicht «Overview» ist eine Übersicht über das Netzwerk möglich. Es werden darin jegliche Flows zwischen zwei Netzwerkkomponenten zusammengefasst als eine Verbindung dargestellt. Diese Ansicht dient vor allem dazu, einen Überblick über das gesamte Netzwerk zu erhalten. Die Filtermöglichkeiten sind aus diesem Grund auch auf einige wenige beschränkt.

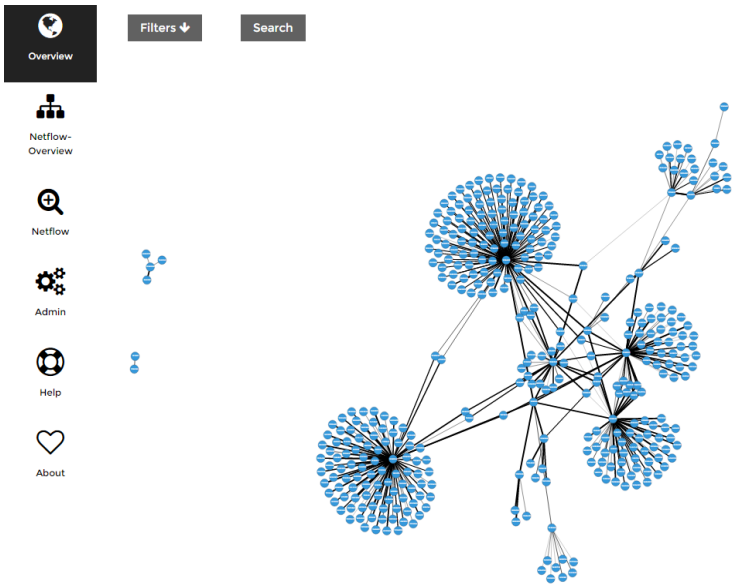


Abbildung 4-2 Userinterface «Overview»

In der Ansicht «NetFlow-Overview» werden die Flows einer Netzwerkkomponente nach Protocol-typ kategorisiert dargestellt. Hier kann die Kommunikation einer bestimmten Netzwerkkomponente genauer untersucht werden. Es werden dafür zusätzliche Filter zur Verfügung gestellt, welche es erlauben, die darzustellenden Daten zu minimieren.

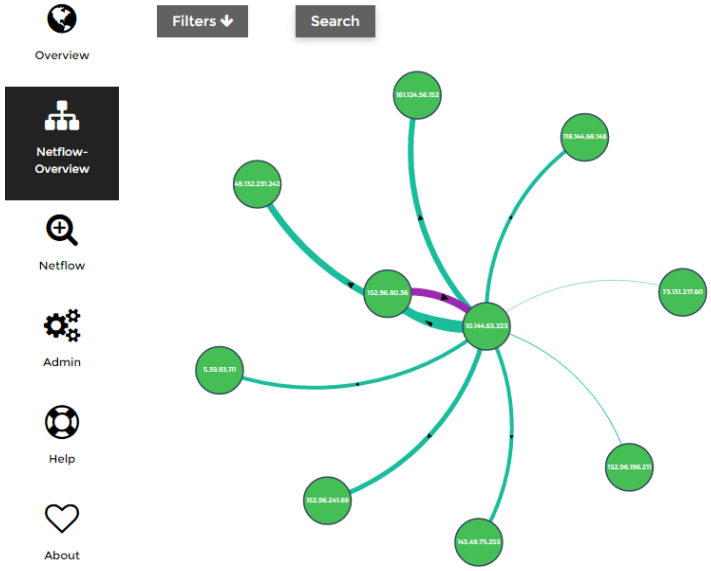


Abbildung 4-3 Userinterface «NetFlow-Overview»

In der Ansicht «NetFlow», welche den höchsten Detailierungsgrad aufweist, wird nun jeder NetFlow einzeln dargestellt. Mit dieser Ansicht sind somit weitere Informationen über die einzelnen Flows für den Benutzer einsehbar.

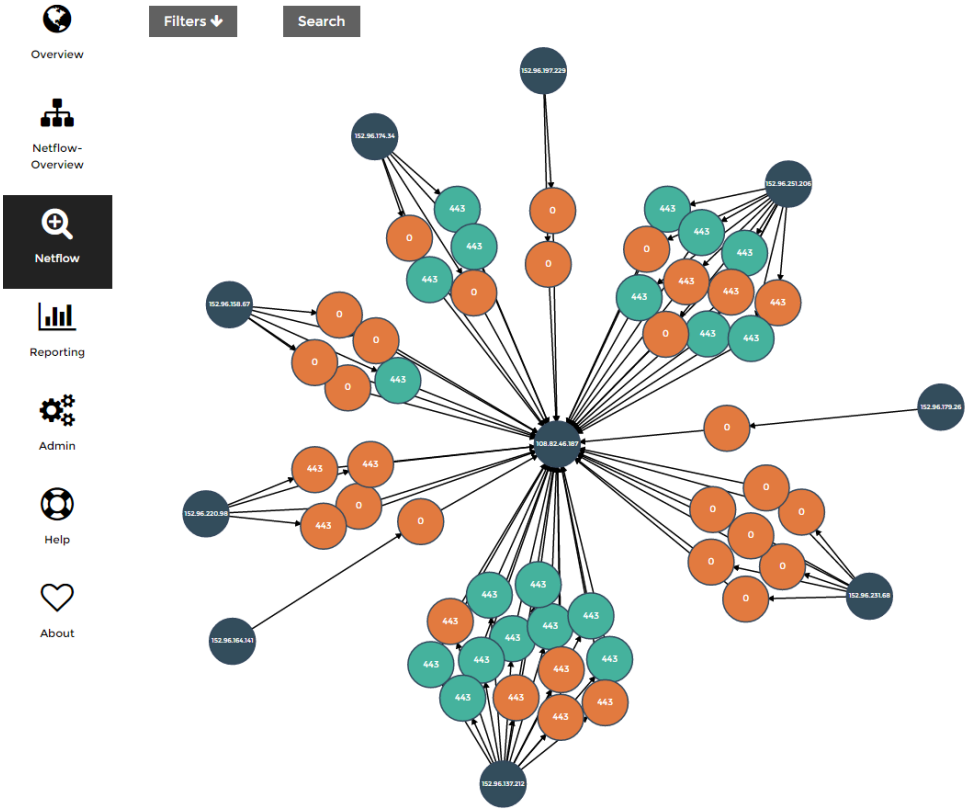


Abbildung 4-4 Userinterface «NetFlow»

Da sich mit dem Ansteigen des Detailierungsgrades auch die dargestellten Datenmengen vergrössern ist vorgesehen, dass der Benutzer die angebotenen Filter nutzt. Durch diese sollte die darzustellenden Datenmenge so verkleinert werden, dass sie übersichtlich dargestellt werden kann.

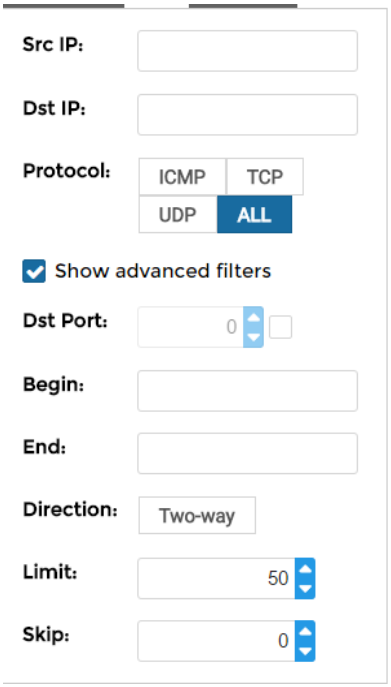


Abbildung 4-5 Userinterface-Ausschnitt «Filtermöglichkeiten»

| Filteroption | Beschreibung |
|---|---|
| Src IP | IP-Adresse des Endpoint-Device, welches der Sender des NetFlows ist. |
| Dst IP | IP-Adresse des Endpoint-Device, welches der Empfänger des NetFlows ist. |
| Protocol | Netzwerkprotokoll, von welchem der Verkehr dargestellt werden soll. |
| Dst Port | Der Destination Port des NetFlows. Diese Filtermöglichkeit kann vom Benutzer deaktiviert werden um nicht danach zu filtern. |
| Begin | Die genaue Zeit, ab welcher die NetFlows angezeigt werden sollen |
| End | Die genaue Zeit, bis zu welcher die NetFlows angezeigt werden sollen |
| Direction | Mit der Option «Two-way» ist es möglich die NetFlows unabhängig der Richtung darzustellen. |
| Limit | Die maximale Anzahl der Einträge, welche angezeigt werden sollen. |
| Skip | Die Anzahl der Einträge, welche übersprungen werden sollen. Diese Option funktioniert nur gemeinsam mit der Funktion Limit. |

Tabelle 4-1 Beschreibung der Filtermöglichkeiten

# 5. Resultate

## 5.1 Zielerreichung

Mit dem ToffiAnalyser kann ein Produkt präsentiert werden, welches die gestellten Basisanforderungen erfüllt. Wir sehen jedoch noch viele Erweiterungsmöglichkeiten. Zu einem Teil wurden solche schon von Anfang an als optional festgelegt, zu einem anderen Teil haben sich sinnvolle Erweiterungen im Verlauf der Arbeit herauskristallisiert. Leider konnten wir aufgrund der begrenzten Zeit nicht alle der eingebrachten Ideen realisieren.

Im Rahmen des ToffiAnalyser-Projekts wurden folgende Resultate erbracht:

- ToffiAnalyser-Webapplication
- ToffiAnalyser-Server
- Die gesamte Applikationsinfrastruktur ist dockerfähig
- Beispielapplikation mit anonymisierten Daten

## 5.2 Ausblick

Wie bereits im Kapitel I 5.1 Zielerreichung erwähnt sehen wir für den ToffiAnalyser noch ein grosses Potential für Erweiterungen.

- Da der gesamte ToffiAnalyser bereits dockerfähig ist, könnte vom INS Institute for Networked Solution der HSR ein ToffiAnalyser-Service angeboten werden. Bei diesem Service könnten externe Firmen via ein entsprechendes Webportal eine ToffiAnalyser-Docker-Instanz beziehen und dort ihre selbst gesammelten NetFlow-Daten hochladen und auswerten lassen. Dieser Service würde sich insbesondere für einmalige Auswertungen oder zu Demonstrationszwecken mit eigenen Daten eignen.
- Es könnte eine Möglichkeit eingebaut werden, welche es erlaubt NetFlow-Daten direkt oder via Umweg über ein Produkt wie z.B. LogStash[16] im ToffiAnalyser zu speichern und danach entsprechend auszuwerten. Eine solche Möglichkeit würde es gestatten, NetFlow-Daten automatisiert und insbesondere auch zeitnah im ToffiAnalyser zur Verfügung zu stellen.
- Die Daten könnten mit der gewählten und flexiblen Visualisierungslibrary, D3.js, auf diverse andere Arten visualisiert werden, da für jeden Anwendungszweck eine andere Darstellung geeignet ist.
- Es könnte die Funktionalität implementiert werden, um gewisse Netzwerke abstrahiert als eine «Wolke» darzustellen. Dies fördert den Überblick und erlaubt eine bessere Fokussierung auf die wesentlichen Informationen.

[16] (Elasticsearch BV, kein Datum)

- Das Reporting könnte um diverse Reporting-Fragestellungen erweitert werden. Dazu müssten vom Netzwerk Team des INS Institute for Networked Solutions der HSR die für sie denkbaren und vor allem relevanten Auswertungen eruiert werden. Der ToffiAnalyser kann für diese Auswertungen um einen neuen Reporttyp ergänzt werden.
- Ein weiterer möglicher Ausbau des ToffiAnalysers könnte darin bestehen, eine vollständige Benutzerverwaltung einzubauen. Mittels einer solchen Verwaltung wäre der ToffiAnalyser in der Lage, benutzerspezifische Informationen wie favorisierte Filtereinstellungen oder Hostnamen zu speichern.

## 5.3 Persönliche Berichte

Wir sind mit dem Resultat des Projektes zufrieden, auch wenn wir uns anfänglich erhofft hatten noch einige optionale Inhalte realisieren zu können. Wir haben beide durch den Einsatz von vielen neuen Technologien sehr viel neue Erfahrungen gemacht. Während der Arbeit konnten wir uns immer aufeinander verlassen und gegenseitig von bestehenden Erfahrungen lernen.

Insbesondere möchten wir folgende für uns neue Technologien hervorheben:

- Spring Boot
- Angular
- Neo4j
- D3.js
- Docker

## 5.4 Danksagung

Wir möchten uns bei Herr Prof. Stefan Keller für seine umfassende Unterstützung während des gesamten Projektes bedanken. Er hat uns mit seinem grossen Erfahrungsschatz während dem Projekt auf wichtige Punkte aufmerksam gemacht und ist uns bei Probleme jeglicher Art zur Seite gestanden.

Wir möchten uns auch bei Silvan Gehrig und Michael Gfeller bedanken, welche uns bei technischen Aspekten mit Tipps Hilfestellung geleistet haben.

Als letztes möchten wir uns bei den Mitarbeitern des INS Institute for Networked Solutions der HSR für die zur Verfügung gestellte Unterstützung bedanken. Insbesondere ist hier die Datenbeschaffung hervorzuheben, ohne welche unser Projekt nicht realisierbar gewesen wäre.

# II. SW-Projektdokumentation

# 1. Vision

Unsere Vision haben wir im Kapitel I 1.1 Problemstellung und Vision auf der Seite 19 beschrieben.

# 2. Anforderungsspezifikation

Dieses Kapitel beschreibt die funktionalen und nicht-funktionalen Anforderungen für die Applikation ToffiAnalyser.

## 2.1 Anforderungen an die Arbeit

Die Arbeit ToffiAnalyser soll eine Applikation als Resultat haben, welche in der Lage ist, Netzwerkverkehr visualisiert darzustellen. Damit wird ein Tool zur Verfügung gestellt, welches den Überblick und die Analyse des Netzwerkverkehrs ermöglicht.

## 2.2 Actors

Für die Applikation ToffiAnalyser existieren folgende Actors, welche die beschriebenen Kenntnisse und Interessen haben:

- *IT-Mitarbeiter:* Ein Mitarbeiter, welcher in der IT-Abteilung arbeitet und selbst IT-Bereich betreut. Der IT-Mitarbeiter hat ein Basiswissen aus dem Netzwerk-Bereich und ist somit in der Lage, die visualisierten Daten zu interpretieren. Beispiele für einen IT-Mitarbeiter sind: IT-Security-Officer, IT-Compliance-Manager, Firewall-Administrator, SDN-Netzwerk-Entwickler und System-Administrator.
- *Netzwerk-Administrator:* Kennt die Netzwerk-Materie ausgezeichnet und will der Toffi-Analyser-Applikation Netzwerkverkehrs-Informationen zur Verfügung stellen.

## 2.3 User-Stories

- *Netzwerkverkehr-Informationen bereitstellen:* Als Netzwerk-Administrator bin ich daran interessiert, von meinen zentralen Netzwerk-Devices die Netzwerkverkehr-Informationen dem ToffiAnalyser zuzustellen. Hierzu kann ich die Daten als Datei liefern, oder meine Netzwerk-Devices so konfigurieren, dass diese die Netzwerkverkehr-Informationen automatisch einer ToffiAnalyser-Schnittstelle übermitteln
- *Übersicht des Netzwerkverkehrs:* Als IT-Mitarbeiter möchte ich darüber informiert sein, welche Art von Netzwerkverkehr in meinem Firmennetzwerk existiert. Dies kann mir helfen Unregelmässigkeiten zu erkennen und Gegenmassnahmen zu ergreifen
- *Netzwerkverkehr analysieren:* Als IT-Mitarbeiter möchte ich Informationen über die Netzwerkkommunikation eines bestimmten Netzwerkknotens analysieren. Dies erlaubt es mir gewisse Komponenten des Netzwerkes bei einem Verdacht oder als Routinetask genauer zu untersuchen.
- *Route des Netzwerkflows analysieren:* Als Netzwerk-Administrator möchte ich genau sehen, welche Komponente mir einen Flow gemeldet hat, sowie über welche Komponente dieser als nächstes geht. Damit kann ich Informationen über den Fluss der NetFlows gewinnen, welche ich anderweitig einsetzen kann.

- *Reports generieren:* Als IT-Mitarbeiter möchte ich über vordefinierte Reports erfahren, was in meinem Firmennetzwerk passiert und wo potentielle Probleme vorhanden sind. Die Reports eigenen sich sehr gut als Routinetask, womit mit kleinstmöglichem Aufwand eine aktuelle Übersicht verschafft wird.

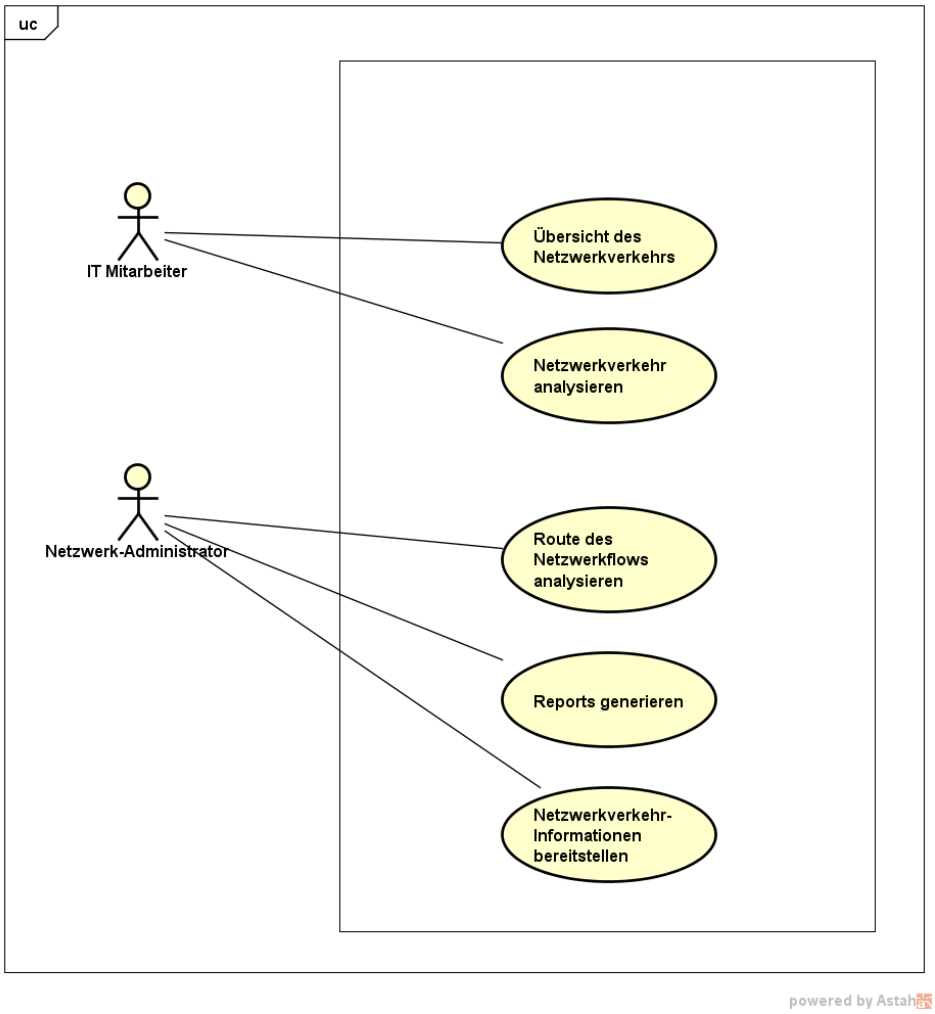


Abbildung 2-1 User-Story-Diagramm

## 2.4 System-Sequenzdiagramme

Die Benutzerinteraktionen mit dem ToffiAnalyser sind sich in den Grundsätzen sehr ähnlich. Aus diesem Grund haben wir uns dafür entschieden lediglich ein System-Sequenzdiagramm zu entwerfen. Die Punkte 1-3 bleiben dabei in jeder User-Story bestehen. Lediglich in den Punkten 4 und 5 unterscheiden sie sich. Eine detailliertere Analyse der Interaktionen ist im Kapitel II 4.3 Sequenzdiagramme zu finden.

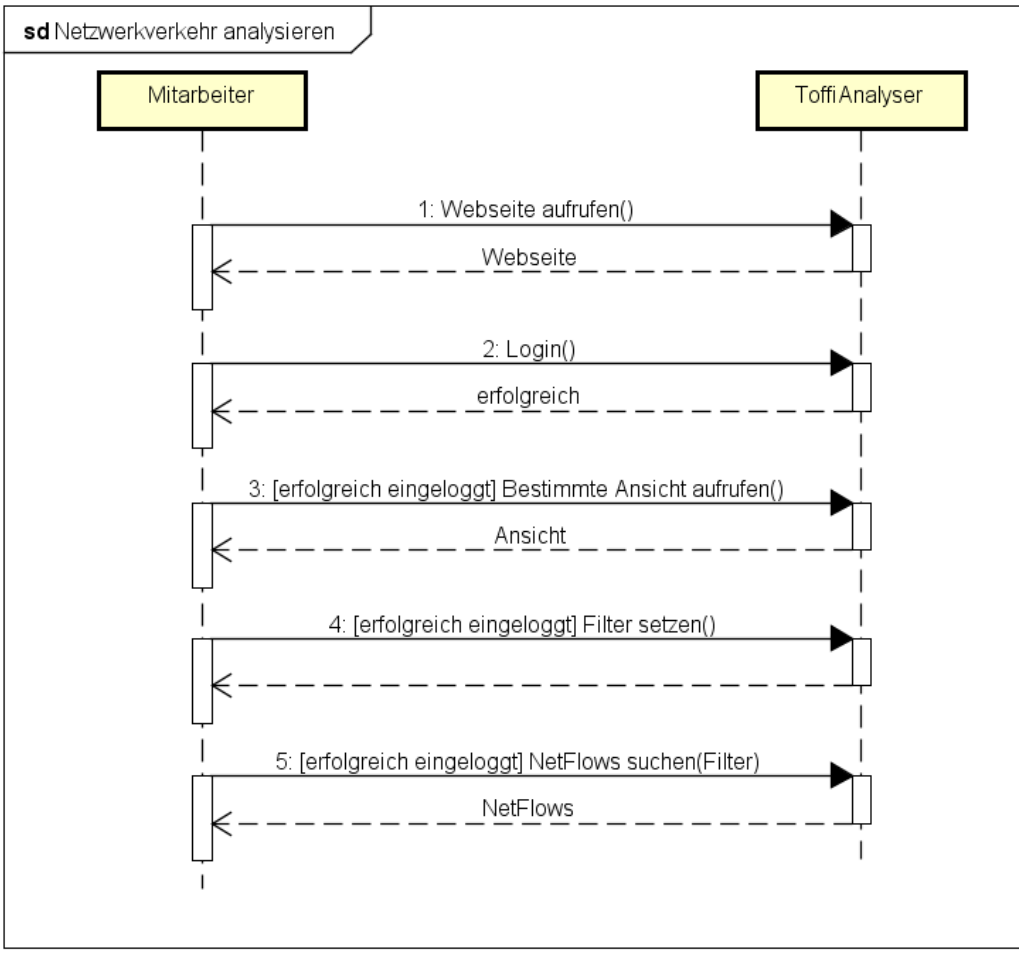


Abbildung 2-2 System-Sequenzdiagramm Netzwerkverkehr analysieren

## 2.5 Nicht-funktionale Anforderungen

### 2.5.1 Performance

Da diese Software nur IT-intern und nur von einer kleineren Benutzer-Gruppe verwendet wird, gehen wir von höchstens 5 gleichzeitigen Benutzern aus. Für diese Benutzer sollte die Reaktionszeit der ToffiAnalyser-Applikation in einem, nach gesundem Menschenverstand beurteilten, vernünftigen Zeitrahmen liegen.

### 2.5.2 Sicherheit

- Die ToffiAnalyser-Webapplikation darf ausschliesslich über eine HTTPS-Verbindung zugreifbar sein.
- Sämtliche ToffiAnalyser-APIs müssen über eine Benutzer- oder Zertifikat-Authentisierung abgesichert sein.

### 2.5.3 Kompatibilität

Die ToffiAnalyser-Webapplikation, muss mindestens 2 der 3 am meisten verbreiteten Webbrowser (Google Chrome, Internet Explorer und Firefox) der aktuellsten Version ohne Einbussen unterstützen.

# 3. Analyse

## 3.1 Domainmodell

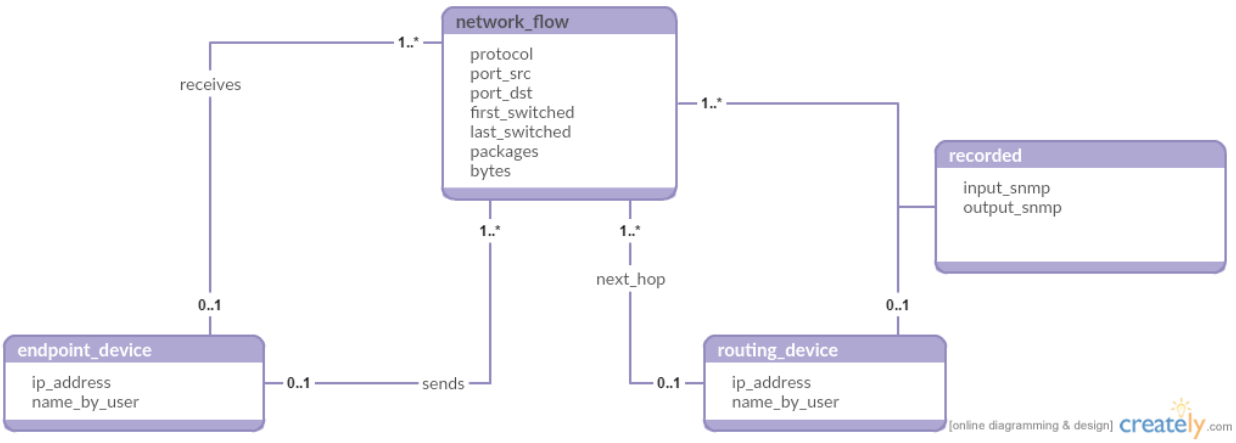


Abbildung 3-1 Domainmodell

Das Domainmodell des ToffiAnalyser besteht aus den folgenden Entitäten

| | |
|---|---|
| endpoint_device | Ein Endpoint-Device repräsentiert einen Netzwerkteilnehmer, welcher mit anderem Teilnehmern kommunizieren kann. Er kann neben einer einzigartigen IP-Adresse auch einen Namen besitzen. |
| network_flow | Ein Network-Flow repräsentiert eine Kommunkationsbeziehung zwischen zwei Endpoint-Devices. Die Network-Flows sind neben den Endpoint-Devices auch an eine bestimmte Zeit gebunden. |
| routing_device | Ein Routing-Device ist eine Netzwerkkomponente, welche zwei unterschiedliche Beziehungen mit dem Network-Flow haben kann. Zum einen kann von NetFlow der nächste Router des Network-Flows angegeben werden. Zum anderen wird der Network-Flow auf einer bestimmten Netzwerkkomponente aufgezeichnet. |
| recorded | Recorded ist eine Entität, welche zusätzliche Daten für die Beziehung zwischen der aufzeichnenden Komponente und einem bestimmten Network-Flows bestimmt. Es sind darin sowohl das eingehende Interface als auch das ausgehende Interface festgehalten. |

## 3.2 Datenmodell

Neo4j kennt seit der Version 2.0 ein optionales Schema für den Graphen, welches auf dem Konzept von Labeln aufbaut. Sie dienen dazu Indices und Constraints für die Datenbank zu definieren. Wir haben das Schema aus dem Domainmodell folgendermassen modelliert.

Es gibt die Node-Labels Routing_Device, Network_Flow und Endpoint_Device für die entsprechenden Entitäten. Zudem gibt es die gerichteten Relationship-Labels SENDS, RECEIVES sowie NEXT_HOP, welche keine zusätzliche Properties besitzen. Als Letztes gibt es das Relationship-Label RECORDED, welches neben der Richtung auch noch gewisse Eigenschaften besitzt, welche direkt auf der Beziehung zu definieren sind.

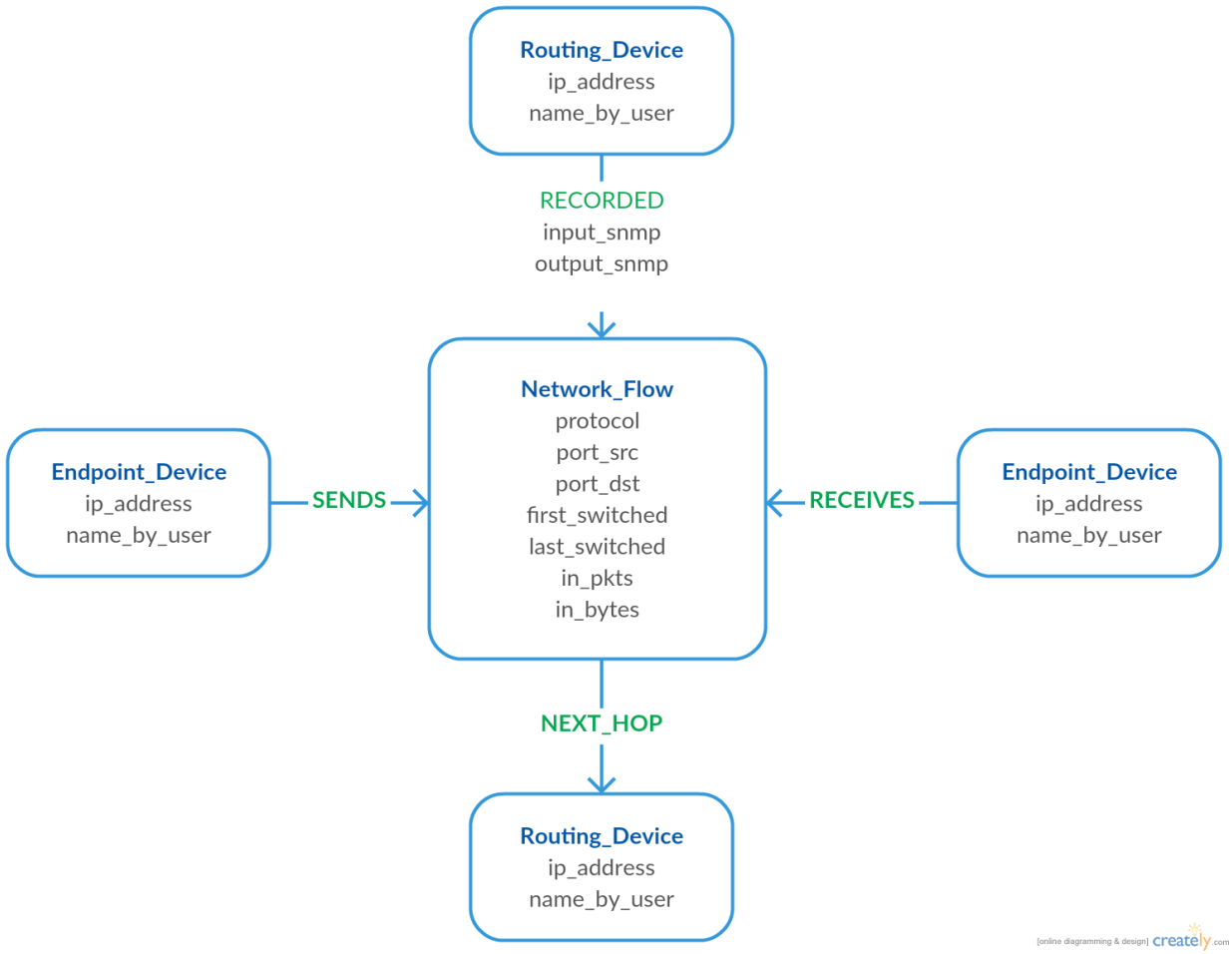


Abbildung 3-2 Neo4j Datenmodell

Zusätzlich dazu definieren wir folgende zwei Constraints, welche dafür sorgen, dass die verwendeten IP-Adressen nur einmalig vorkommen. Zudem wird dadurch automatisch ein Index auf dem entsprechenden Attribut angelegt, was uns sehr entgegen kommt, da dieses bei unseren Abfragen oft benutzt wird.

> *Note that adding a unique property constraint on a property will also add an index on that property, so you cannot add such an index separately.*[17]

```
CREATE CONSTRAINT on (endpoint:Endpoint_Device) ASSERT endpoint.ip_address IS UNIQUE
```

Abbildung 3-3 Endpoint-Device Unique-Constraint

```
CREATE CONSTRAINT on (routingDevice:Routing_Device) ASSERT routingDevice.ip_address IS UNIQUE
```

Abbildung 3-4 Routing-Device Unique-Constraint

[17] (Neo Technology, Inc, kein Datum)

# 4. Design

## 4.1 Architektur

Die grundsätzliche Architektur des ToffiAnalysers besteht aus den folgenden drei Komponenten

- ToffiAnalyser-Client
- ToffiAnalyser-Server
- Neo4j-DB

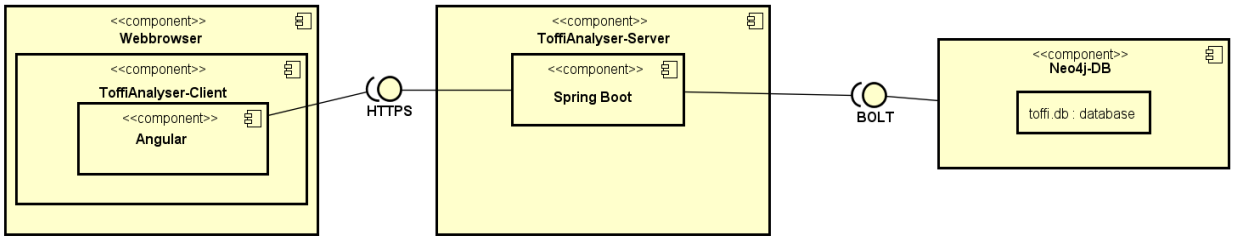


Abbildung 4-1 Architekturübersicht

Der ToffiAnalyser-Client ist eine Angular-Single-Page-Application und stellt die Schnittstelle zum Benutzer dar.

Der ToffiAnalyser-Server ist eine Spring Boot-Applikation, welche zwei Funktionen besitzt. Einerseits stellt sie den ToffiAnalyser-Client zur Verfügung[18], andererseits ist auch die Programmierschnittstelle über den ToffiAnalyser-Server erreichbar[19]. Der ToffiAnalyser-Server ist aus Sicherheitsgründen nur über HTTPS erreichbar.

Die Persistierung der Daten erfolgt in einer Neo4j-Datenbank. Für die Verbindung mit der Datenbank wird dabei aus Performancegründen die Bolt-Schnittstelle verwendet. Die Kommunikation mit der Datenbank über Bolt ist dabei mit SSL geschützt.

> *«The Neo4j 3.0 Milestone 1 release introduces Bolt, a new network protocol designed for high-performance access to graph databases.»*[20]

[18] Detaillierte Informationen dazu im Kapitel 5.1.1.2 Auslieferung des ToffiAnalyser-Clients

[19] Detaillierte Informationen dazu im Kapitel 5.1.1.4 API Dokumentation

[20] (Kollegger, 2015)

Der ToffiAnalyser-Server sowie die Neo4j-Datenbank sind als Docker-Image verfügbar, der Toffi-Analyser-Client ist dabei in der Serverkomponente enthalten. Dadurch kann die Installation und Konfiguration der benutzten Komponenten umgangen werden und die gesamte ToffiAnalyser-Umgebung kann mit einigen wenigen Command-Line-Befehlen gestartet werden.

## 4.2 Package- und Klassendiagramme

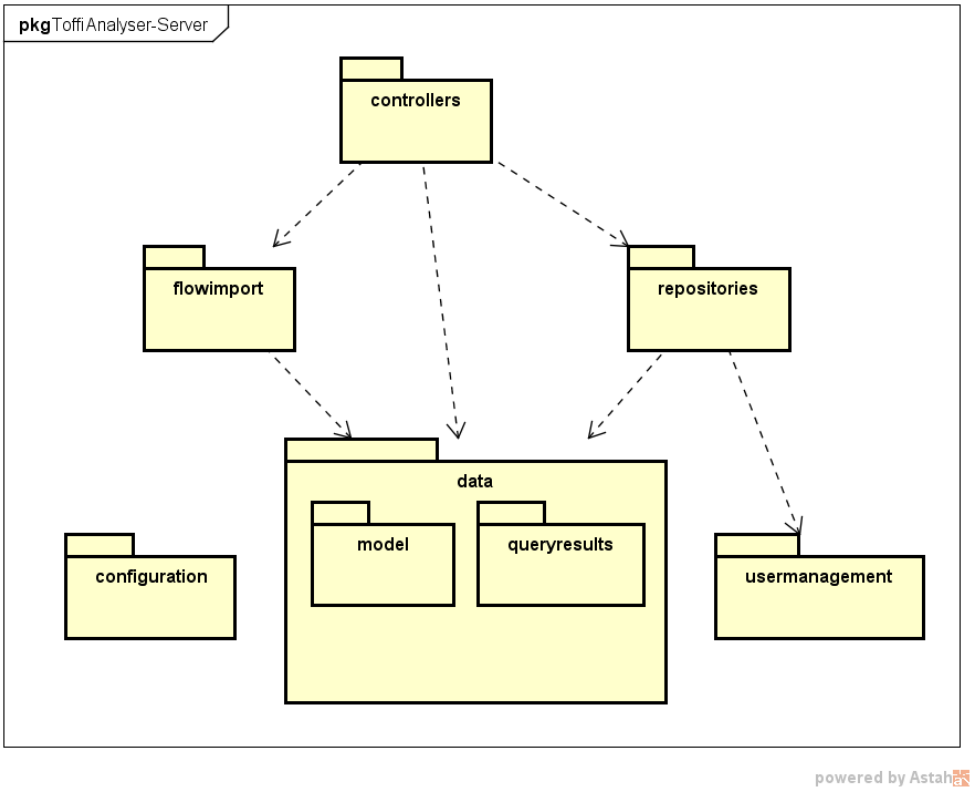


Abbildung 4-2 Package Diagramm ToffiAnalyser-Server

Die Serverkomponente besteht aus den folgenden Packages. Die Pfeile stehen dabei für eine Abhängigkeit.

| | |
|---|---|
| Configuration | Alle Spring Boot spezifischen Konfigurationsklassen, wie z.B. die Neo4j-Datenbankverbindung. |
| Controllers | Alle Spring Boot «RestController» welche die zur Verfügung gestellte Schnittstelle definieren. |
| Repositories | Alle Repositories, welche die Daten für die Controller zur Verfügung stellen. |
| data.model | Das ToffiAnalyser-Datenmodell. Siehe auch II 3.1 Domainmodel |

| | |
|---|---|
| data.queryresults | Enthält alle Klassen, welche von einem Datenbank-Query eines Repositories zurückgegeben werden, damit Spring Data ein Mapping herstellen kann. |
| flowimport | Alle Klassen, welche für den Import von NetFlow-Daten zuständig sind. |
| usermanagement | Die stark vereinfachte Benutzerverwaltung des ToffiAnalysers |

Tabelle 4-1 Package Übersicht ToffiAnalyser-Server

## 4.3 Sequenzdiagramme

Der Ablauf der User-Stories «Übersicht des Netzwerkverkehrs», «Netzwerkverkehr analysieren» sowie «Route des Netzwerkflows analysieren» sind sich sehr ähnlich. Es wurde für diese User-Stories deshalb lediglich für ein gemeinsames System-Sequenzdiagramm erstellt. Der Kontext der Benutzerinteraktion ist im Kapitel II 0

System-Sequenzdiagramme festgehalten.

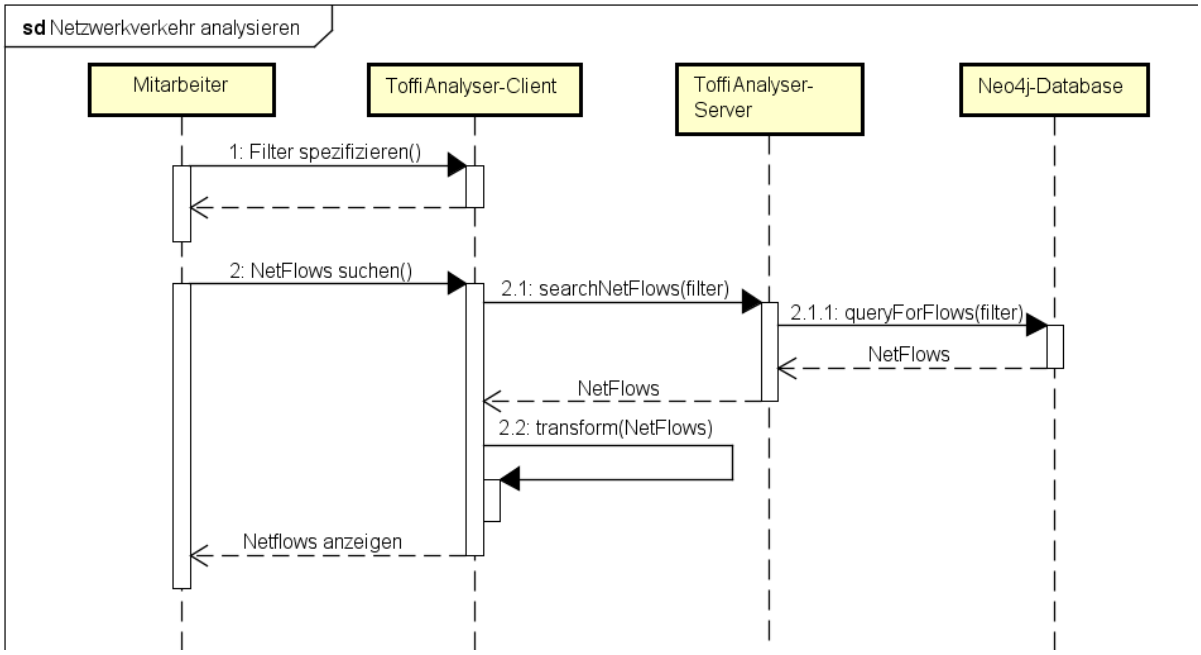


Abbildung 4-3 Sequenzdiagramm «Netzwerkverkehr analysieren»

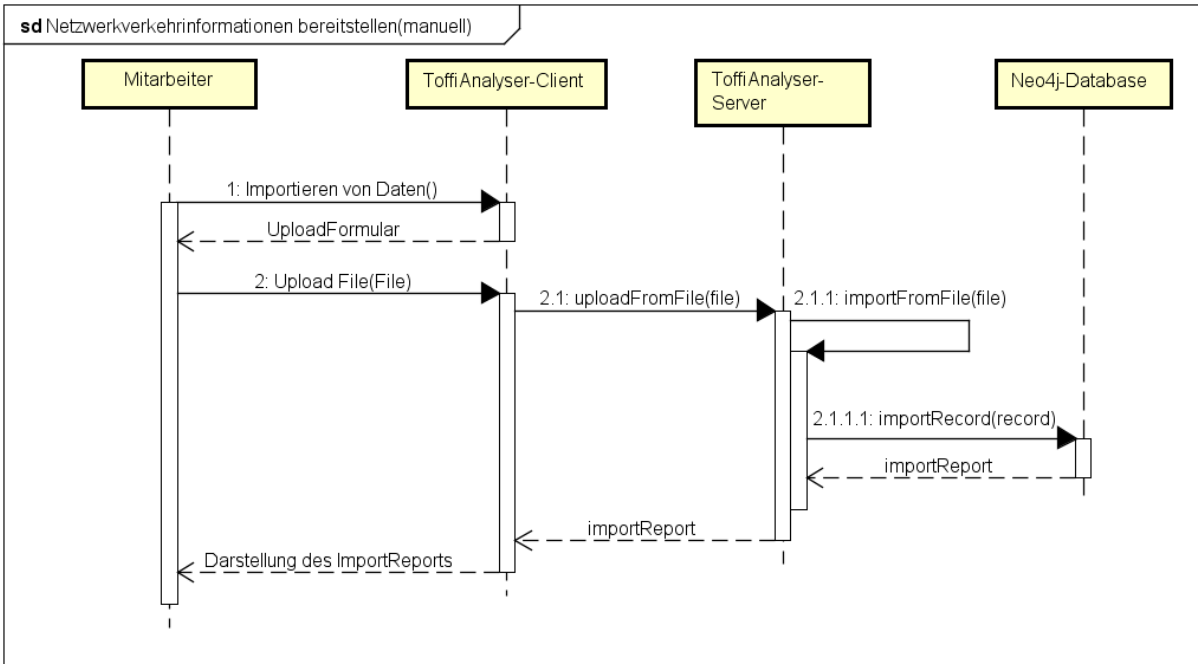


Abbildung 4-4 Sequenzdiagramm «Netzwerkverkehrsinformationen bereitstellen»

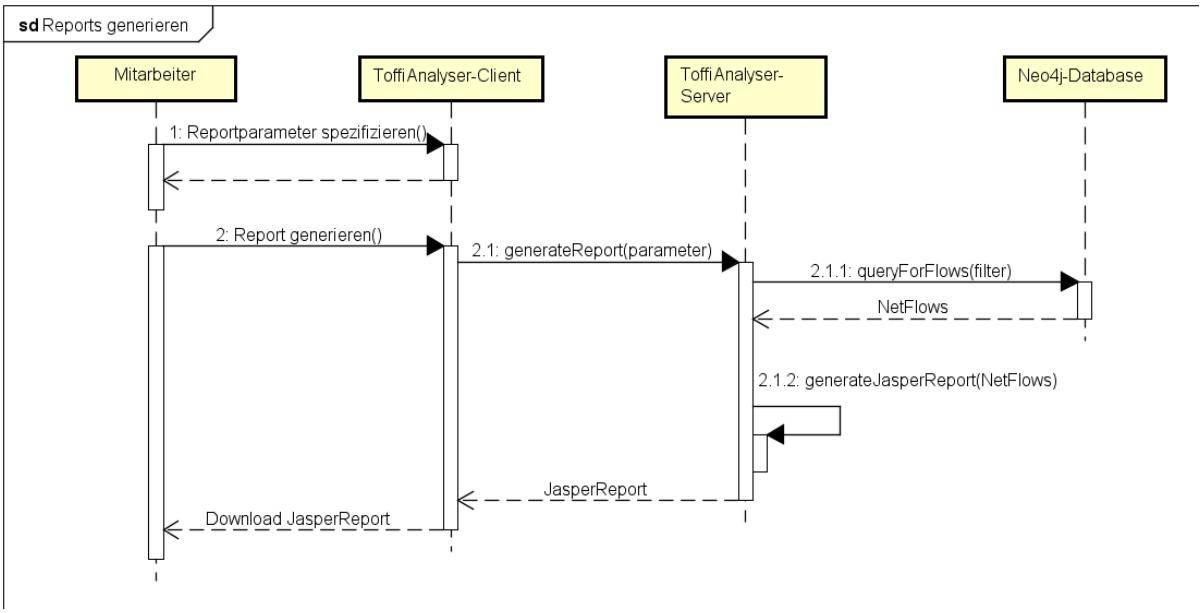


Abbildung 4-5 Sequenzdiagramm «Reports generieren»

## 4.4 UI-Design

Am Anfang des Projekts haben wir einige Mockups mittels balsamiq für den ToffiAnalyser-Client erstellt. Die wichtigste Ansicht ist dabei die «Overview», welche den Einstiegspunkt in unsere Applikation bietet. Wir haben uns deshalb dazu entschieden, die Mockups der Visualisierungen an dieser View zu orientieren. Zusätzlich haben wir noch ein grobes Mockup der «Admin»-Ansicht erstellt. Wichtig dabei war uns vor allem die Anordnung der einzelnen Basis-Elemente, wie der Navigation und den Filtern.

In einer ersten Version haben wir uns die Visualisierungsansichten der Applikation folgendermassen vorgestellt. Auf der linken Seite ist die Navigation, welche schmal gehalten ist, um möglichst viel Platz für die Visualisierung zur Verfügung zu stellen. Am Anfang der Seite ist zudem die Möglichkeit, die Daten zu filtern. Die Seite ist möglichst einfach gehalten und auf das wesentliche fokussiert.

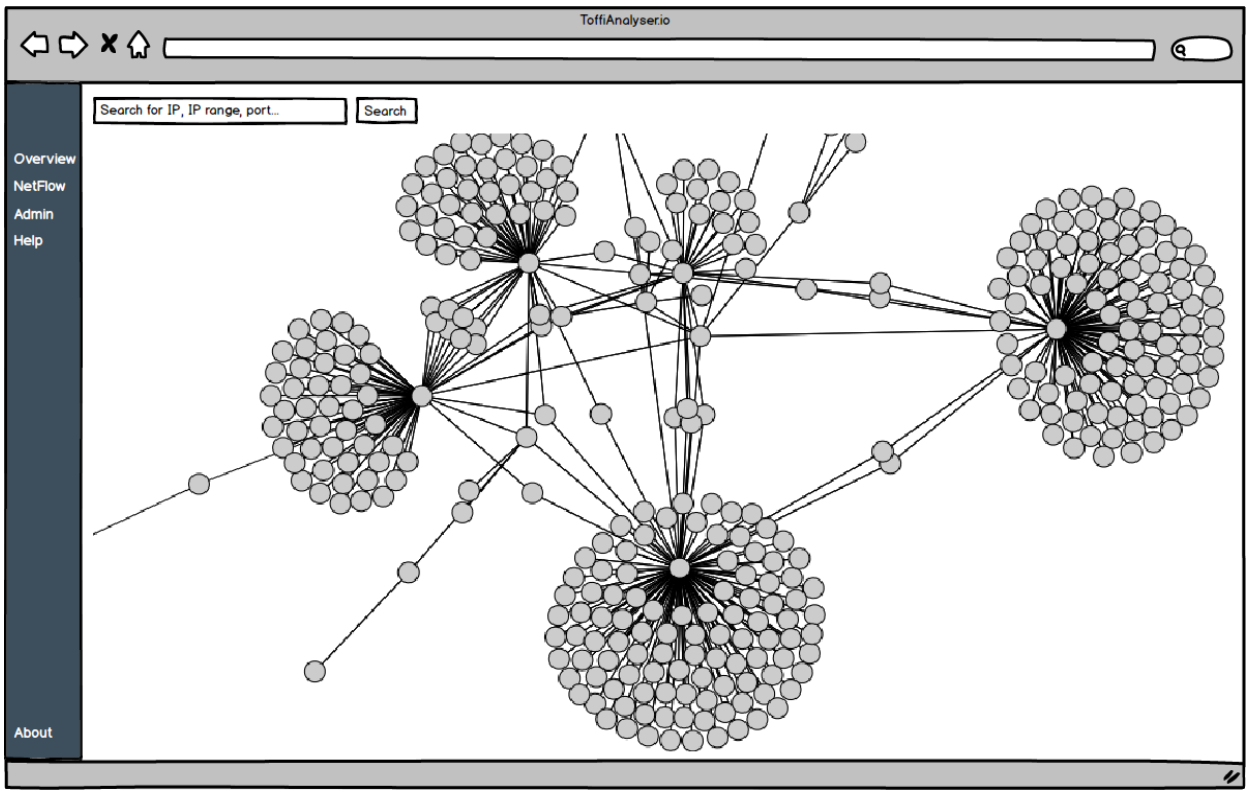


Abbildung 4-6 Mockup «Overview»

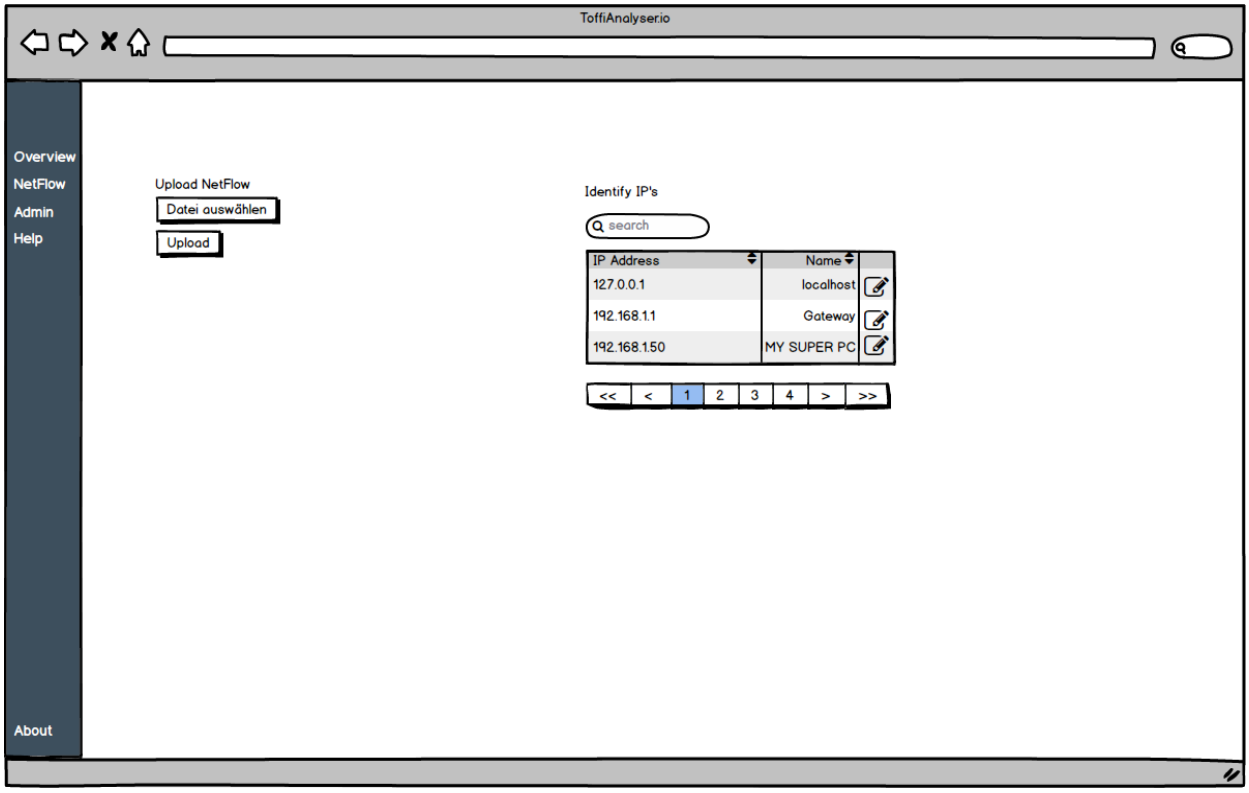


Abbildung 4-7 Mockup «Admin»

Das Admin-Userinterface sieht eine einfache Möglichkeit vor, eine Datei auszuwählen und hochzuladen; dies ist auf der linken Seite der Ansicht realisiert. Auf der rechten Seite ist noch genug Platz vorhanden um die verschiedenen Netzwerkkomponenten als Liste darzustellen und dem Benutzer die Möglichkeit zur Verfügung zu stellen, diesen einen Namen zu geben.

Im Verlauf des Projektes haben sich diese Ansichten zwar noch leicht verändert, die Grundsätze sind allerdings gleichgeblieben.

# 5. Implementation und Testing

## 5.1 Implementation

Das Kapitel Implementation enthält genauere Informationen über die Implementation des Toffi-Analysers. Unter anderem unsere wichtigsten Überlegungen sowie einige Herausforderungen mit welchen wir uns beschäftigt haben.

Das Kapitel ist entsprechend den Komponenten in verschiedene Unterkapitel eingeteilt.

### 5.1.1 ToffiAnalyser-Server

Bereits bei der Planung des Projektes haben wir uns bei der Serverkomponente für eine Spring Boot-Applikation entschieden. Die Vorteile von Spring Boot liegen darin, dass im Gegensatz zu Spring selbst keine XML-Konfigurationen angelegt werden müssen, aber trotzdem die gesamte Funktionalität des Spring Frameworks verfügbar bleibt. Uns hat auch der folgende Leitsatz von Spring Boot gefallen:

> *«Spring Boot favors convention over configuration and is designed to get you up and running as quickly as possible.»*[21]

Wir versprachen uns davon, dass wir mit möglichst wenig Aufwand eine vollumfängliche Serverkomponente zur Verfügung haben. Ein weiterer Vorteil von Spring Boot gegenüber anderen Konkurrenten wie z.B. dem Play Framework ist für uns das Community-Projekt «Spring Data Neo4j», welches die Kommunikation mit der Neo4j-Datenbank stark vereinfacht.

#### 5.1.1.1 Spring Data Neo4j

Spring Data Neo4j übernimmt für uns die Kommunikation mit der Datenbank sowie das Erstellen der entsprechenden Java-Objekte basierend auf den Resultaten.

Das Mapping der Attribute entspricht dabei standardmässig wiederum dem Prinzip «convention over configuration». Eine Änderung der Konfiguration kann über Java-Annotationen gemacht werden. Es ist ebenfalls möglich, Beziehungen im Modell zu annotieren.

Als Beispiel haben wir einen Ausschnitt unserer Modell-Klasse «EndpointDevice» eingefügt, welches unsere verwendeten Annotationen beinhaltet.

[21] (Spring.io, kein Datum)

```
@NodeEntity(label="Endpoint_Device")
public class EndpointDevice {

    @GraphId
    private Long id;
    @Property(name="ip_address")
    private String ipAddress;
    private String name;

    @Relationship(type="SENDS", direction= Relationship.OUTGOING)
    private Set<NetworkFlow> outgoingFlows;

    @Relationship(type="RECEIVES", direction= Relationship.OUTGOING)
    private Set<NetworkFlow> incomingFlows;
```

Abbildung 5-1 «Spring Data Neo4j» Modell-Annotationen

Neben *@NodeEntity*, welches einen entsprechenden Node in der Datenbank voraussetzt, gibt es für Abfrageresultate die Annotation *@QueryResult*. Klassen welche wir damit annotiert haben und somit nicht zu unserem Domainmodell gehören, sind im Package data.queryresults enthalten.

Für die Abfragen der Datenbank existieren die Repositories. Ein Repository ist in Spring Data lediglich ein Interface, welches die gewünschten Methoden enthält. Die Implementation des Interface übernimmt das Framework. Einfachere Abfragen können wiederum über den Namen der Methode gesteuert werden, kompliziertere Abfragen, wie die meisten von uns benötigten, werden mittels Annotationen definiert.

```
@Query("MATCH (endpoint:Endpoint_Device) - [] - (flow:Network_Flow) "
        + "WHERE endpoint.ip_address =~ {ipAddress} "
        + "AND flow.first_switched >= {firstSwitched} "
        + "AND flow.last_switched <= {lastSwitched} "
        + "WITH endpoint.ip_address as ipAddress, "
        + "endpoint.name as name, "
        + "sum(flow.bytes) as totalBytes "
        + "RETURN ipAddress, name, "
        + "totalBytes "
        + "ORDER BY totalBytes DESC")
Iterable<ServerTraffic> getMonthlyServerTraffic(
        @Param("ipAddress") String ip_address,
        @Param("firstSwitched") Long first_switched,
        @Param("lastSwitched") Long last_switched);
```

Abbildung 5-2 Spring Data Neo4j Repository Annotations

Das Query wird dann von «Spring Data Neo4j» zusammen mit den Parametern an die Datenbank geschickt und das Resultat in Objekte der Klasse «ServerTraffic» umgewandelt. Es ist von Vorteil die Parameter separat zu schicken, damit Injections verhindert werden können.

Neben den vielen Vorteilen, uns der Einsatz von «Spring Data Neo4j» bot, hatten wir jedoch auch einige Herausforderungen zu lösen.

Die grösste Herausforderung war, dass viele unserer Schnittstellen mehrere optionale Parameter zur Verfügung stellen. Da wir jedoch für Neo4j keine Möglichkeit gefunden haben, optionale Parameter zu übergeben, blieben uns hier zwei mögliche Auswege. In einer ersten Variante wäre es bei den meisten Abfragen möglich, die Filterung erst serverseitig durchzuführen. In einer zweiten Variante definiert man mehrere Abfragen um mit den optionalen Parametern klar zu kommen.

Wir haben uns auf folgenden Gründen für die zweite Variante entschieden:

- Eine einheitliche Definition der Queries, da die serverseitige Filterung nicht mit Aggregationen kombinierbar ist.
- Bei einer serverseitigen Filterung ist mit Performance-Einbussen zu rechnen

Wir haben dabei die Problematik in Kauf genommen, mehrere Funktionen im Repository zu definieren und mehrere @Query-Annotationen zu schreiben. Um die zu definierenden Funktionen in einem möglich kleinen Umfang zu halten, setzten wir folgende Massnahmen um:

- Default-Werte, wo es als sinnvoll erachtet werden kann
- Grosse Teile der Abfrage sind separat definiert und werden lediglich um die Filter ergänzt.

```
String findFlow1 = " MATCH (endpoint:Endpoint_Device) "
        + "- [:SENDS] - (n:Network_Flow ) - "
        + "[:RECEIVES] - (partner:Endpoint_Device) ";
String findFlow2 = " AND n.first_switched >= {startDate} "
        + "AND n.last_switched <= {endDate} "
        + "RETURN endpoint as src, "
        + "COUNT(n) as totalFlows, "
        + "SUM(n.packages) as totalPackages, "
        + "SUM(n.bytes) as totalBytes, "
        + "n.dst_port as dst_port, "
        + "MIN(n.first_switched) as totalFirstSwitched, "
        + "MAX(n.last_switched) as totalLastSwitched, "
        + "partner as dst "
        + "SKIP {skip} "
        + "LIMIT {limit} ";

@Query(findFlow1
        + " WHERE endpoint.ip_address={src_ip}"
        + findFlow2 )
Iterable<NetflowOverview> findFlow(
        @Param("src_ip") String src_ip,
        @Param(value = "limit") Integer limit,
        @Param(value = "skip") Integer skip,
        @Param(value="startDate") Long startDate,
        @Param(value="endDate") Long endDate);
@Query(findFlow1
        + " WHERE endpoint.ip_address={src_ip} "
        + "AND n.protocol={protocol} "
        + findFlow2 )
Iterable<NetflowOverview> findFlow(
        @Param("src_ip") String src_ip,
        @Param(value = "limit") Integer limit,
        @Param(value = "skip") Integer skip,
        @Param(value="startDate") Long startDate,
        @Param(value="endDate") Long endDate,
        @Param(value="protocol") Integer protocol);
```

Abbildung 5-3 «Spring Data Neo4j»-Repository-Annotationen mit geteilter Abfrage

### 5.1.1.2 Auslieferung des ToffiAnalyser-Clients

Neben der API stellt der ToffiAnalyser-Server für den Benutzer auch den ToffiAnalyser-Client-Code zur Verfügung.

Spring Boot stellt standardmässig alle Ressourcen zur Verfügung, welche im Ressourcen-Unterordner «public» oder «static» abgelegt sind. Dadurch ist es uns möglich den fertigen Clientcode auf dem Server zu hinterlegen und den Clients zur Verfügung zu stellen.

Da neben der Serverkomponente auch der Client über eine Routingfunktionalität verfügt (nachzulesen im Kapitel II 5.1.2 ToffiAnalyser-Client), reicht dies noch nicht aus um die gewünschte Funktionalität zu realisieren.

Wenn der Benutzer auf eine bestimmte Ansicht navigiert hat, passt sich die URL an diese Ansicht an. Bei einem erneuten Laden derselben URL wird ein Request an den Server geschickt, welcher diese Ressource nicht kennt. Dieses Problem haben wir dadurch behoben, dass der Server alle Requests mit dem Status `HttpStatus.NOT_FOUND` an index.html weiterleitet. Die damit ausgelieferte Angular SPA routet danach wieder clientseitig zur richtigen Ansicht weiter.

### 5.1.1.3 Externe Konfiguration

Die Konfiguration diverser Parameter erfolgt über den Spring-eigenen Mechanismus. In den Ressourcen des ToffiAnalyser-Clients ist eine Standardkonfiguration vorhanden, welche unter anderem auf folgende Arten überschrieben werden kann:

- Kommandozeilen Argumente beim Aufruf
- Andere «application.properties»-Datei, welche im Directory des Jar liegt
- Andere properties Datei, welche beim Aufruf verlinkt wird

```
C:\Toffi>java -jar toffianalyser-server-0.1.0.jar --toffi.db.uri=https://localhost:7000
```

Abbildung 5-4 ToffiAnalyser-Konfiguration über Kommandozeilenargumente

Wenn nur einzelne Parameter überschrieben werden sollen, können diese bequem über die Kommandozeile übergeben werden. Dies eignet sich speziell dann, wenn diese Parameter nur temporär angepasst werden sollen.

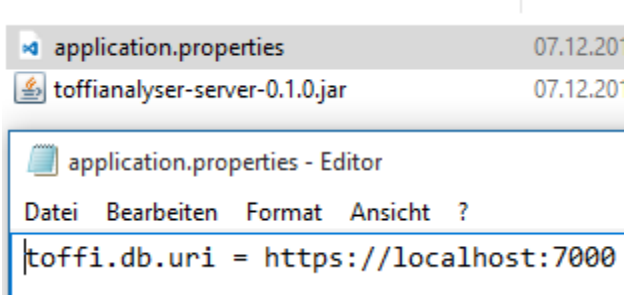


Abbildung 5-5 Konfiguration über application.properties

Wenn die ToffiAnalyser-Konfiguration standardmässig überschrieben werden soll, sollte eine «application.properties»-Datei im Ordner des ToffiAnalyser-JARs gelegt werden. Alternativ kann dies auch im Unterordner config abgelegt werden.


Abbildung 5-6 Konfiguration mit Dateipfad

Wenn die Properties-Datei an einem anderen Ort oder mit einem anderen Namen abgelegt werden muss, kann dies über die spring.config.location Variable beim Starten des ToffiAnalysers definiert werden. Mehrere Konfigurationsdateien sind dabei mit einem Komma zu trennen.

Neben den hier aufgezählten Möglichkeiten die Konfiguration zu überschreiben gibt es noch viele andere[22], welche unserer Meinung nach beim ToffiAnalyser keine primäre Rolle spielen.

Wir haben uns dafür entschieden applikationsspezifische Konfigurationsvariablen mit dem Präfix toffi zu versehen um sie von den Spring-internen Variablen abzuheben.

Die Einbindung der Konfigurationswerte im Code ist abhängig vom Einsatzort.

Wenn der Wert in einer Annotation verwendet wird kann der folgende Syntax verwendet werden *«${the.name.of.the.variable}»*

```
@GetMapping(path = "${toffi.path.api.admin.getNodes}")
```

Abbildung 5-7 Konfiguration in Annotation

Wenn der Wert in einer Variablen gespeichert werden soll, kann diese mit @Value annotiert werden, um Zugriff auf den Wert der Konfiguration zu erhalten.

```
@Value("${toffi.db.uri}")
private String NEO4J_URI;
```

Abbildung 5-8 Konfiguration in Variable

#### 5.1.1.4 API Dokumentation

Um immer eine mit dem Code übereinstimmende Dokumentation der API zu besitzen haben wir Springfox[23] eingesetzt. Springfox ist eine Implementation von Swagger spezifisch für Spring.

[22] (Spring.io, kein Datum)

[23] (springfox, kein Datum)

Springfox ermöglicht es mit kleinem Konfigurationsaufwand eine übersichtliche API-Dokumentation zu erstellen, welche auf einer generierten Webseite unter `/swagger-ui.html` einsehbar ist.

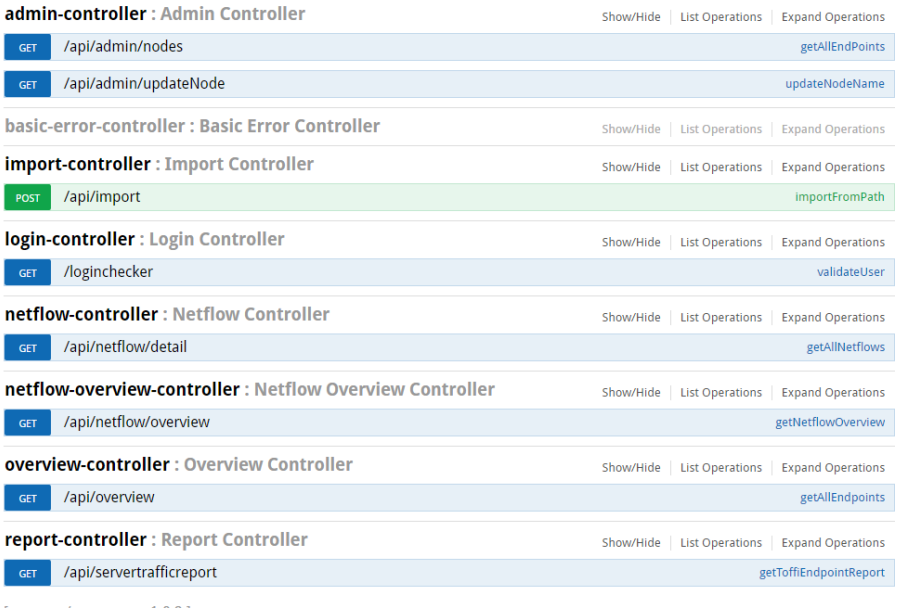


Abbildung 5-9 Swagger UI Übersicht über die verfügbaren API-Aufrufe

In einer ersten Ansicht ist eine Übersicht über alle API Aufrufe aufgelistet. Es ist zudem möglich, sich für jeden Aufruf eine detaillierte Liste der verfügbaren Parameter und des Return Values anzeigen zu lassen.

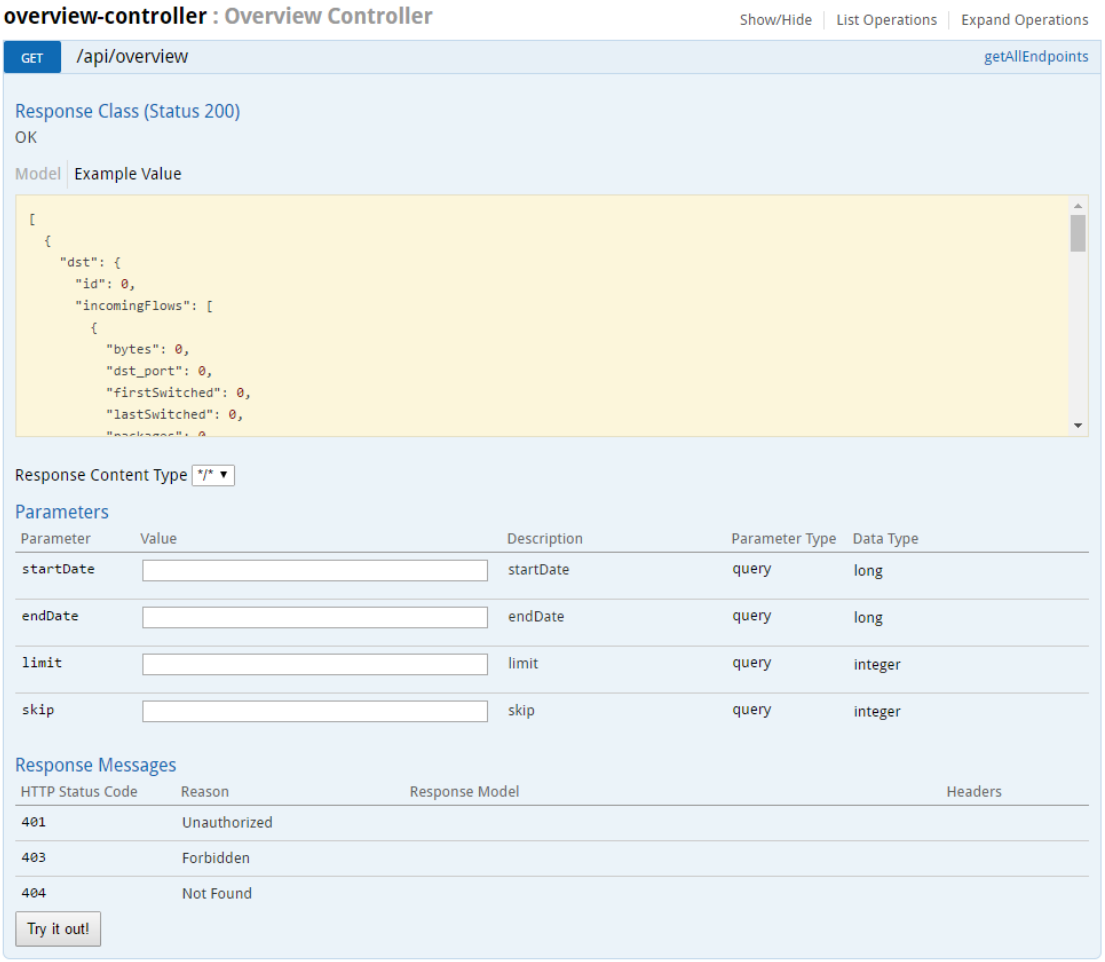


Abbildung 5-10 Swagger UI Detailansicht des Overview Controllers

Es existiert momentan leider ein Problem mit der Darstellung des erwarteten Return-Types, wenn dieser verschachtelte Typen beinhaltet.[24] Dies hat zur Folge, dass die Darstellung des Return-Types anstelle des richtigen Typen lediglich «Inline Model» benannt wird. Dies kann zu einiger Verwirrung bei der Benutzung von Swagger UI führen, wir haben uns jedoch entschieden Swagger UI trotzdem einzusetzen, da die Vorteile überwiegen.

[24] (pgodzin, 2016)

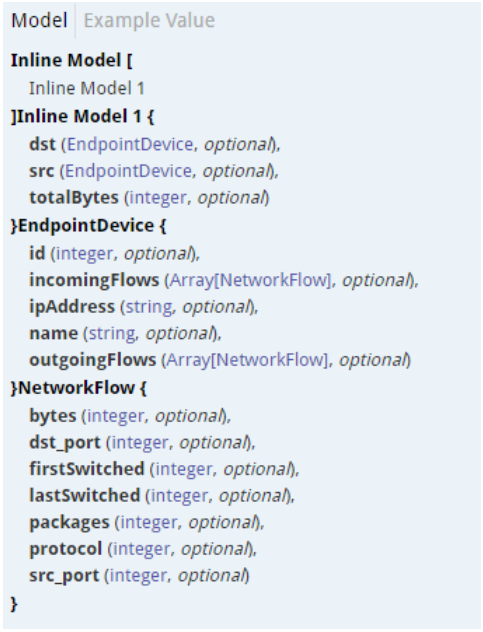


Abbildung 5-11 Swagger UI Return-Type

### 5.1.2 ToffiAnalyser-Client

Wie die Serverkomponente, haben wir auch das clientseitige Framework bereits bei der Planung des Projektes festgelegt. Wir haben dazu das Angular Framework ausgewählt, welches im Sommer 2016 neu in der Version 2.0 veröffentlicht wurde. Wir haben Angular aus den folgenden Gründen gewählt:

- Typescript
- Laut Dokumentation ein einfacher Einstieg
- Bereits Erfahrungen mit dem Vorgänger Angular.js
- Zukunftstauglichkeit
- Möglichkeit einer Single-Page-Application

Wir haben für die Einrichtung von Angular das vom Projekt zur Verfügung gestellte Angular CLI verwendet, welches das Erstellen von neuen Teilen stark vereinfacht.

Der grösste Vorteil von Angular hat sich für uns aus der Wiederverwendbarkeit der einzelnen Components ergeben. Components sind ein Konzept von Angular, welches es ermöglicht die Templatesprache mit eigenen Components zu erweitern. Dies erlaubt einerseits, wie bereits erwähnt, die Components mehrfach zu verwenden, andererseits kann die Webseite dadurch relativ modular zusammengestellt werden. Neben den Hauptkomponenten der Angular Webpage, in welchen jeweils die einzelnen Seiten enthalten sind, haben wir noch diverse Hilfskomponenten erstellt und in den Hauptkomponenten eingebunden.

Das clientseitige Routing, d.h. die Zuteilung, welche Komponente beim Aufruf einer bestimmten URL aufgerufen wird, ist in einer einzelnen Datei, app-routing.module.ts, definiert.

Die folgende Tabelle enthält alle Routen, sowie die aufgerufene Komponente. Als letzte Spalte ist definiert, ob die Route mit einem Login geschützt ist. Der Benutzer kann die Route nur aufrufen, wenn er sich im Voraus angemeldet hat.

| URL | Komponente | Beschreibung | Geschützt |
|---|---|---|---|
| /help | HelpComponent | Bietet Hilfestellung zum ToffiAnalyser und seiner Bedienung | ✘ |
| /login | LoginComponent | Ist für das Login zuständig | ✘ |
| /about | AboutComponent | Allgemeine Informationen über den ToffiAnalyser, seine Urheber, sowie das Copyright | ✘ |
| /error | ErrorComponent | Darstellung eines Fehlers, wenn es zur Problemen mit der Applikation kommt | ✘ |
| /overview | OverviewComponent | Darstellung der ToffiAnalyser-Overview, welche eine Übersicht über das Netzwerk gibt | ✓ |
| /netflow | NetFlowComponent | Darstellung der ToffiAnalyser-NetFlow Ansicht, welches die detaillierteste Ansicht ist | ✓ |
| /netflow-overview | NetFlowOverviewComponent | Darstellung der ToffiAnalyser-NetFlow-Overview, welche eine genauere Analyse erlaubt. | ✓ |
| /admin | AdminComponent | Funktionalitäten, welche nicht von allen ToffiAnalyser-Benutzern benötigt werden, wie das Benennen von Netzwerkkomponenten | ✓ |
| / | OverviewComponent | Wenn der ToffiAnalyser ohne weitere URL aufgerufen wird, stellen wir die Overviewkomponente dar. | ✓ |
| /** | ErrorNotFoundComponent | Die Default-Route, welche dann in Kraft tritt, wenn keine andere Route definiert ist. Es wird dann eine Fehlerseite angezeigt. | ✘ |

Tabelle 5-1 Clientseitiges Routing

#### 5.1.2.1 Visualisierungen

Die Visualisierungen im ToffiAnalyser-Client sind mittels der Visualisierungslibrary D3.js realisiert. Weshalb wir uns für D3.js entschieden haben ist im Kapitel I 3 Evaluation Visualisierungslibrary ausführlich beschrieben.

Im Juni 2016 wurde eine neue Version von D3.js fertiggestellt, welche nicht rückwärtskompatibel ist. Wir mussten allerdings vom Einsatz dieser neuen Version für den ToffiAnalyser absehen, da bis zum Start der Arbeit keine offiziellen Typefiles für diese Version existieren. Es existieren schon aktuelle Typefiles, es ist jedoch noch nicht abzusehen, wann diese in das @Types-Repository aufgenommen werden.[25] Die Typefiles werden für die Integration der Javascript-Library in Typescript benötigt.

D3.js baut auf den Technologien HTML, SVG und CSS auf um die Visualisierungen durchzuführen. SVG besitzt gegenüber anderen Technologien den Vorteil, dass für das Styling der Komponenten normal auf CSS gesetzt werden kann und dadurch stark vereinfacht wird.

Insgesamt beinhaltet der ToffiAnalyser-Client drei Ansichten, in denen eine ähnliche aber nicht identische Visualisierung benötigt wird. Um möglichst wenig redundanten Code zu besitzen und trotzdem flexibel genug zu bleiben haben wir eine generische Klasse, den D3GraphHelper, für das Aufbauen des Graphen geschrieben. Diese generische Klasse arbeitet mit mehreren Callbacks, um genügend Flexibilität zu gewährleisten. Zum Beispiel lässt sich damit der Inhalt der Tooltips definieren, da dieser stark von der entsprechenden Ansicht abhängt.

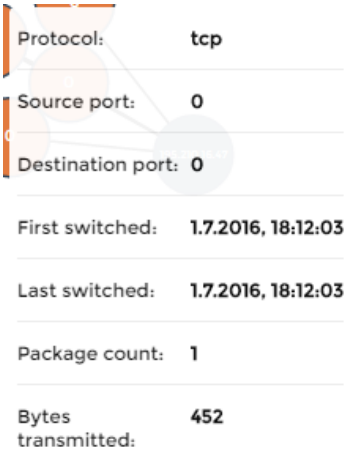


Abbildung 5-12 Tooltip NetFlow

Für die NetFlow-Overview-Ansicht, in der zwei Nodes mehrere Verbindungen miteinander haben können, gingen die Anpassungsfähigkeiten über die Callbacks nicht weit genug. Wir haben deshalb eine zweite Klasse D3GraphHelperMultipleEdges geschrieben, welche einige Methoden der Basisklasse überschreibt, somit bleibt der Grundaufbau bei allen Ansichten gleich.

Alle Darstellungen setzten auf dem Force-Layout von D3 auf, welches eine automatische Anordnung der Nodes vornimmt[26]. Die Anordnung ist dabei nicht deterministisch und kann deshalb von Aufruf zu Aufruf ändern.

[25] (tomwanzek, 2016)

[26] (mbostock, d3-3.x-api-reference/Force-Layout.md, 2016)

#### 5.1.2.2 Services

Die in den Ansichten benötigten Daten, bezieht die Komponente nicht direkt vom API-Endpoint, sondern von einem Service. Der Service ist für die Kommunikation mit der ToffiAnalyser-Serverkomponente zuständig und stellt dem Aufrufer so eine einfache Möglichkeit zum Datenbezug zur Verfügung. Zusätzlich zur Verbindung mit der Serverkomponente ist der Service auch für die Aufbereitung der Daten, wie dem Umformen von JSON in Typescript-Objekte, zuständig.

Vom Service werden allerdings nicht direkt die Daten zurückgegeben, sondern ein Observable, auf welches dann EventHandler gebunden werden können. Es ist somit sowohl möglich, das Verhalten beim Erhalt der Daten zu spezifizieren, als auch direkt eine Fehlerbehandlung zu bestimmen.

#### 5.1.2.3 Konfiguration der API

Um den ToffiAnalyser-Client möglichst flexibel zu halten, haben wir gemäss den Empfehlungen des Angular Development Teams[27] die Konfiguration der API-Endpoints in eine separate Datei ausgelagert. Die Konfiguration wird dann, wenn benötigt, von Angular geladen, und der Wert kann ausgelesen werden. Bei einer Anpassung des API-Endpoints kann dadurch die Anpassung clientseitig schnell und unkompliziert durchgeführt werden.

### 5.1.3 Reporting

Ein Reporting war am Anfang der Arbeit als optionales Ziel definiert worden. Während der Realisierung der Arbeit hat sich gezeigt, dass ein Reporting in unserem Fall sehr sinnvoll sein kann. Deshalb haben wir uns dazu entschieden für einen einfachen Fall ein Reporting zu implementieren und damit aufzuzeigen, wie ein solches Reporting aussehen kann. Die Realisierung des Reportings ist mittels JasperReports Library implementiert, welche in Java geschrieben ist. Die Integration in den ToffiAnalyser hat deshalb sehr einfach funktioniert. Die Reporting-Funktionalität besteht aus den folgenden Teilen:

- Eingabemaske für den Benutzer im ToffiAnalyser-Client
- JasperReport-Vorlage
- Controller im ToffiAnalyser-Server

[27] (Android Development Team, kein Datum)

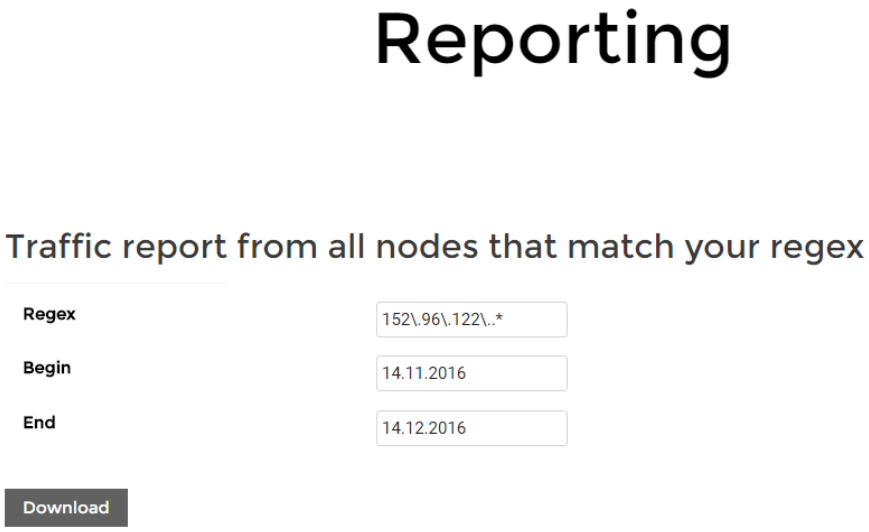


Abbildung 5-13 Userinterface Reporting

Wir haben mit dem JasperSoftStudio eine einfache JasperReport-Vorlage definiert und diese auf dem ToffiAnalyser-Server hinterlegt. Der Benutzer kann nun über die Eingabemaske die fehlenden Parameter für den Report definieren und diesen direkt als PDF herunterladen. Zum einen lässt sich eine RegularExpression eingeben, welche gegen die IP-Adresse aller Endpoint-Devices geprüft wird. Zum anderen ist der Zeitraum konfigurierbar über welchen die Daten im Report ausgewertet werden sollen.

> *Cypher supports filtering using regular expressions. The regular expression syntax is inherited from the Java regular expressions.*[28]

Die RegularExpression entspricht dabei der normalen Java Regular Expression Syntax. Der Report gibt eine Übersicht über den Umfang des Netzwerkverkehrs, an welchem der Endpoint beteiligt war. Ein Beispiel eines solchen Reports ist im Anhang zu finden.

### 5.1.4 Sicherheitsüberlegungen

Neben den bereits erwähnten Sicherheitsüberlegungen zur Sicherung der Kommunikation im Kapitel II 4.1 Architektur, wurden noch weitere Aspekte betrachtet.

Um die ToffiAnalyser-Serverkomponente gegen unbefugte Zugriff abzusichern, wurde Basic-Authentication eingesetzt. Als Konfigurationsbasis wird beim Starten des ToffiAnalysers ein Systemadministrator-Benutzer angelegt. Es wurde, wie im Semesterarbeit-Scope geplant, auf eine komplette Benutzerverwaltung verzichtet. Aktuell wird dieser Benutzer lediglich in der «applica-

[28] (Neo Technology, Inc., kein Datum)

tion.properties»-Datei konfiguriert und befindet sich während der ToffiAnalyser-Laufzeit im Arbeitsspeicher zwischengespeichert. Es ist uns bewusst, dass diese Methode zur Benutzerverwaltung für produktive Systeme nicht adäquat ist. Für Hinweise, wie die Sicherheit erhöht werden könnte, verweisen wir gerne auf das Kapitel II 6.2.6 Benutzerverwaltung.

### 5.1.5 Randomisation-Script

Um den ToffiAnalyser zu Demonstrationszwecken laufen zu lassen, benötigen wir gültige, aber auf keinen Fall produktive Daten. Wir haben deshalb ein Python-Script geschrieben, welches uns die IP-Adressen im Script austauscht. Damit die Adressen nicht komplett zufällig und somit zur Auswertung nutzlos werden, befolgt unterscheidet das Programm folgende IP-Netzwerke:

- Private Class A (10.0.0.0/8)
- Private Class B (172.16.0.0/12)
- Private Class C (192.168.0.0/16)
- HSR-Netzwerk (152.96.0.0/16)

Zudem ist es möglich einzelne spezielle IP-Adressen zu hinterlegen, die nicht ausgetauscht werden sollen.

Jeder Source- und Destination-IP-Adresse in den NetFlow-Daten teilt das Programm einmalig eine zufällige IP-Adresse aus dem jeweiligen Netzwerk zu. Dadurch ist sichergestellt, dass die gleichen Beziehungen zwischen den Netzwerkkomponenten bestehen aber trotzdem kein Rückschluss auf einen bestimmten Benutzer gemacht werden kann.

### 5.1.6 Bekannte Programmfehler

Neben diversen Problemen, welche während der Projektrealisation erfolgreich behoben werden konnten, existiert zum Abgabezeitpunkt noch ein nicht behobener Fehler im ToffiAnalyser.

Das Login der ToffiAnalyser-Single-Page-Application besitzt den Fehler, dass bei der Eingabe von falschen Login-Informationen das standardmässige Basic-Authentication-Eingabefeld des Browsers erscheint. Dieses muss dann zuerst vom Benutzer weggeklickt werden, bevor er einen weiteren Login-Versuch starten kann.

## 5.2 Deployment

Das Ziel beim Deployment des ToffiAnalysers war, möglichst viele Hostsysteme zu unterstützen und zugleich das Setup so einfach die möglich zu halten. Aus diesem Grund wurde uns relativ schnell klar, dass Docker eine gute Grundlage für die Erfüllung dieser Ziele liefern würde.

Grundsätzlich besteht das gesamte ToffiAnalyser-Setup aus zwei Docker-Container, welche beide zusammen arbeiten um den ToffiAnalyser-Service anbieten zu können. Zum einen gibt es den Spring Boot ToffiAnalyser-Container, welcher die eigentliche Spring Boot Serverkomponente hostet und die Angular SPA ausliefert. Zum anderen verlangt es ein Neo4j Container, welcher dem

ToffiAnalyser-Container eine Neo4j Instanz zur Speicherung der NetFlow-Daten zur Verfügung stellt.

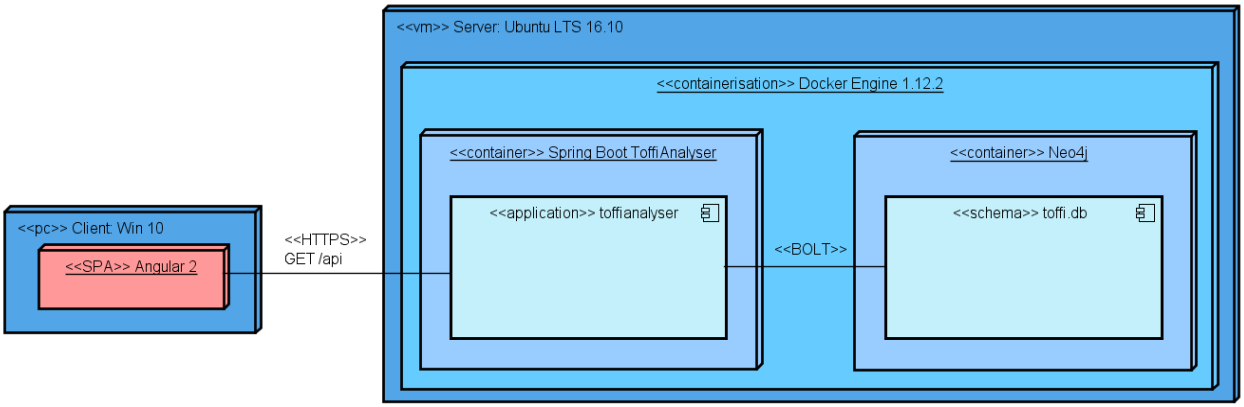


Abbildung 5-14 Deployment mittels Docker

## 5.3 Automatische Testverfahren (CI)

Wir haben für unser Projektsetup bereits anfangs entschieden, auf Atlassian Produkte wie Jira und Bitbucket zu setzten. Somit kam uns gerade gelegen, dass Atlassian im Sommer 2016 mit Bitbucket Pipelines[29] ihr eigenes Cloud CI vorgestellt hat. Wir entschieden uns, Bitbucket Pipelines eine Chance zu geben, da es direkt in Bitbucket integriert ist. Ausserdem ist es möglich, beliebige Docker-Images vom Docker Hub[30] als Build Environment zu verwenden.

Die Konfiguration für das CI wird über eine simple «.yml»-Datei im Root Directory des Repositories gesteuert.

[29] (Atlassian, kein Datum)

[30] (Docker Inc., 2016)

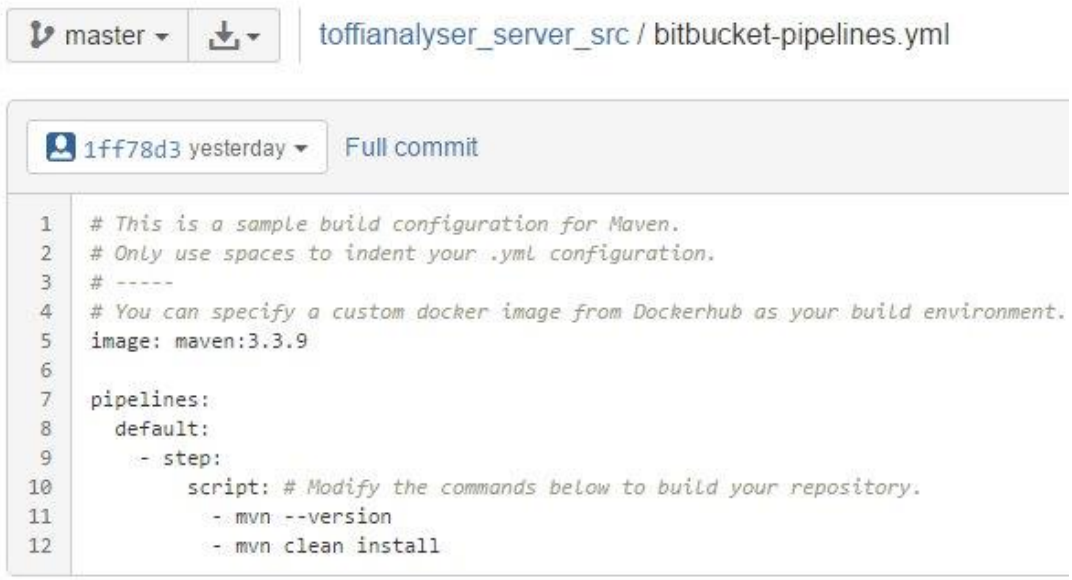


Abbildung 5-15 Bitbucket Pipelines Konfiguration

Wie das Beispiel unseres Server Repositories zeigt, genügen wenige Zeilen Konfiguration um den gesamten Build und die Ausführung der Unit-Tests zu testen. Die wirklich relevanten Zeilen sind hier die Zeile 5 und Zeile 12. Mit «image: maven:3.3.9» auf Zeile 5 wird definiert, dass das auf dem Docker Hub Verfügbare Maven Docker-Image der Version 3.3.9 für den Build verwendet werden soll. Das Command «mvn clean install» auf Zeile 12 führt alle Unit-Tests aus und buildet die Software.

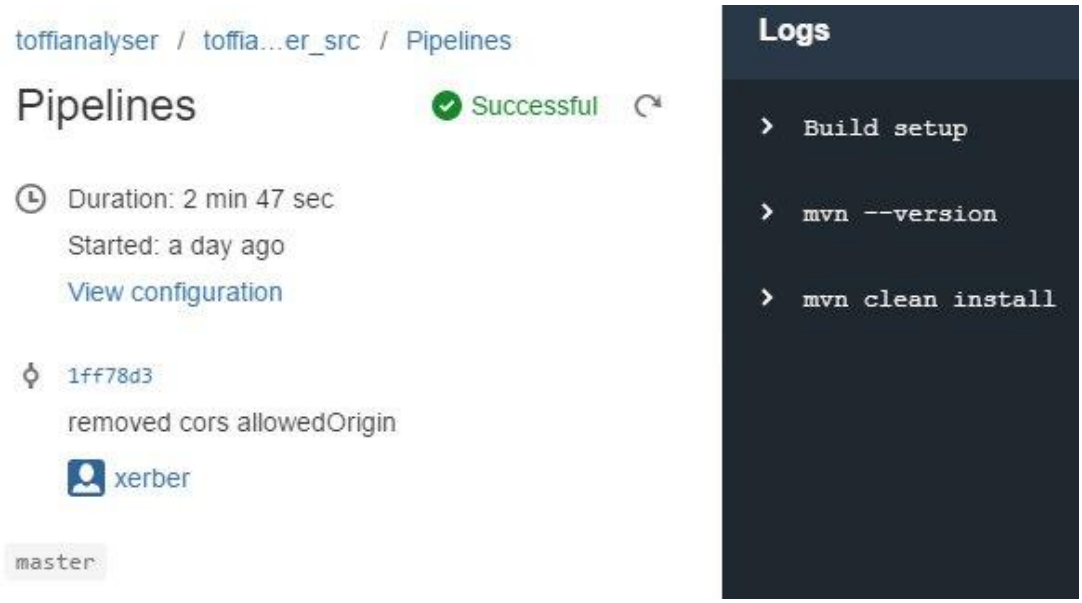


Abbildung 5-16 Bitbucket Pipelines Build Status

## 5.4 Manuelle Testverfahren (Systemtests etc.)

Neben den automatischen Testverfahren, haben wir für den ToffiAnalyser auf Systemtests gesetzt. Die komplette Systemtest-Spezifikation ist im Anhang aufzufinden.

# 6. Resultat und Weiterentwicklung

## 6.1 Resultate

Alle Resultate wurden bereits im Kapitel I 5.1 Zielerreichung sowie im Kapitel II 5.1 Implementation beschrieben.

## 6.2 Möglichkeiten der Weiterentwicklung

In diesem Kapitel wollen wir gewisse bereits erwähnte und mögliche Weiterentwicklungen aus Kapitel I 5.2 Ausblick aus technischer Sicht ausführlicher aufzeigen.

### 6.2.1 ToffiAnalyser-«as a Service»

Um die ToffiAnalyser-Software als einen Service im Internet zur Verfügung zu stellen, wären mehr Arbeiten um die eigentliche Software herum nötig, als an der Software selbst.

Als Erstes bräuchte es eine Möglichkeit, eine ToffiAnalyser-Instanz als externe Firma über das Internet zu beziehen. Hier wäre eine relativ simple Webseite denkbar, über welche man sich mittels OAuth[31] authentifizieren könnte.

Beim jeweiligen Starten eines ToffiAnalyser-Docker-Containers müsste sichergestellt werden, dass je eine neue «application.properties»-Datei generiert würde. Für diese Dateien müssten jeweils zufällige und starke Passwörter generiert und darin eingefügt werden. Mehr Informationen zu dieser spezifischen Konfigurationsdatei können unter Kapitel II 5.1.1.3 Externe Konfiguration nachgelesen werden.

Zu guter Letzt müsste noch überlegt werden, auf welcher Umgebung der Service genau angeboten werden kann. Einerseits käme hier eine Instituts-interne Container Technologie Lösung wie z.B. OpenShift Origin[32] in Frage und andererseits ein Hosting auf einer Public Container Cloud wie z.B. DigitalOcean[33].

### 6.2.2 Datenimport

Der Datenimport könnte um die Funktionalität erweitert werden, dass automatisiert und zeitnah NetFlow-Daten dem ToffiAnalyser hinzugefügt werden.

[31] (Auth0® Inc, kein Datum)

[32] (OpenShift Origin, kein Datum)

[33] (DIGITALOCEAN ™ INC., kein Datum)

Um diese Funktionalität zu erreichen, braucht es auf dem ToffiAnalyser einen weiteren API Eintrittspunkt, welcher die Daten z.B. über einen HTTP POST entgegennimmt, parst und anschliessend in der Neo4j Datenbank abspeichert. Der Einfachheit halber würde es Sinn machen, diese Daten als JSON entgegen zu nehmen, da dann das binäre und zugleich proprietäre Cisco NetFlow Protokoll nicht implementiert werden müsste. Zugleich wäre so die einfache Erweiterbarkeit auf NetFlow ähnliche Protokolle von anderen Herstellern gewährleistet.

### 6.2.3 Visualisierungsarten ergänzen

Um die Daten möglicherweise noch in einer anderen Art darzustellen, wären Anpassungen in der Angular2 SPA selbst nötig. Denkbare Visualisierungsmöglichkeiten können in der D3.js Galerie[34] betrachtet werden.

Unter anderem bräuchte die SPA eine weitere Ansicht, in welcher die neue Visualisierungsmöglichkeit dargestellt würde. Um die Visualisierung selbst zu erstellen, müsste eine weitere Typescript Klasse erstellt werden, welche die D3 Library importiert und entsprechend auf von D3.js zur Verfügung gestellte Klassen zugreift.

### 6.2.4 Netzwerkabstrahierung

Um die Übersichtlichkeit des ToffiAnalysers weiter zu steigern, könnte eine Netzwerkabstrahierung helfen. Wir stellen uns hierfür eine Funktionalität vor, mit welcher - auf Benutzerwunsch hin - gewisse Netzwerke zusammengefasst dargestellt werden können. Welche Netzwerke abstrahiert werden sollen, ist dabei durch den Benutzer zu steuern. Wir können uns hierfür zwei verschiedene Konzepte vorstellen, welche sich zum Teil auch kombinieren lassen.

Als Eingabeformat stellen wir uns eine Regular-Expression vor, da damit sehr mächtige Definitionen möglich sind. Die Eingabe könnte direkt in der jeweiligen Ansicht durch ein weiteres Eingabefeld getätigt werden. Alternativ erachten wir, in Kombination mit einer ausgereiften Benutzerverwaltung, die Abspeicherung von standardmässigen Abstrahierungen pro Benutzer als sehr sinnvoll.

### 6.2.5 Reporting Erweiterungen

Das eingebaute Reporting Framework Jasper bietet grundsätzlich eine enorme Flexibilität. Aktuell ist lediglich eine mögliche Reporting-Fragestellungen implementiert (siehe IV 5 Beispiel-Report). Um weitere Arten von Reports zu stellen, sind folgende Anpassungen durchzuführen.

Als Erstes kann ein Neo4j Cypher Query geschrieben werden, welches die gewünschten Daten der Fragestellung liefert. Zugleich braucht es eine zusätzliche Jasper-Vorlage, welche definiert, wie der neue Report anschliessend auszusehen hat.

[34] (d3.js, 2016)

Anschliessend kann im Report Controller des ToffiAnalysers alles zusammengeführt werden.

Zum Schluss müssen in der SPA nur noch die benötigten Eingabefelder hinzugefügt und die Anbindung auf den vorhin erstellten API Punkt muss realisiert werden.

### 6.2.6 Benutzerverwaltung

Für einen langfristigen und produktiven Einsatz benötigt der ToffiAnalyser eine vollumfängliche Benutzerverwaltung sowie eine Anbindung an ein LDAP/Kerberos. Durch eine Anbindung an ein externes System kann dem Benutzer ein eigenes Login erspart und gewisse Informationen zum Benutzer direkt abgefragt werden. Zugleich wäre somit die Grundlage gegeben, ein mögliches Single-Sign-On zu implementieren.

Um eine Benutzerverwaltung zu implementieren müsste als erstes ein geeignetes Datenbanksystem zur Benutzerdatenpersistierung evaluiert werden sofern keine direkte Anbindung an ein LDAP/Kerberos gewünscht ist oder zusätzliche Daten zu jeweiligen Benutzer gespeichert werden sollen. Nachdem dies gemacht wurde, müssen die serverseitig benötigten Veränderungen in der ToffiAnalyser-Software selbst ausfindig gemacht werden. Da Spring Boot eine grosse Anzahl an möglichen Anbindungen an die verschiedensten Datenbanksysteme unterstützt, dürfte die Anbindung selbst nicht allzu komplex werden. Wichtig zu beachten ist serverseitig insbesondere, dass die Benutzer-Passwörter in der gewählten Datenbank ausschliesslich als salted Hash[35] abgespeichert werden.

Eine vollumfängliche Benutzerverwaltung eröffnet dem ToffiAnalyser noch viele weitere Funktionen, welche vor allem einen Einfluss auf die User-Experience haben. Unter anderem könnte der ToffiAnalyser um folgende Features erweitert werden, welche benutzerspezifische Daten speichern.

- Favorisierte Filtermöglichkeiten
- Benennung von Geräten
- Konfiguration von automatischen Reports
- Speicherung von Netzwerkabstrahierungen

Clientseitig müsste primär das Usermodell um die gewünschten Werte erweitert und die neue Funktionalität implementiert werden.

[35] (Wikipedia, 2016)

# 7. Projektmanagement

## 7.1 Projektmeetings

Wie bei Semesterarbeiten üblich, haben wir uns zu Beginn der Arbeit mit dem Betreuer darauf geeinigt, wöchentlich ein Projektmeeting abzuhalten. Sofern inmitten einer Projektphase keine nennenswerten Hindernisse aufgetreten sind, konnte ein solches wöchentliches Projektmeeting auch gewollt ausgelassen werden. Zu jedem abgehaltenen Projektmeeting mit dem Betreuer wurde ein Protokoll erstellt.

Neben den offiziellen Projektmeetings mit dem Betreuer wurden diverse team-interne kleinere Meetings für geringfügigere Absprachen gehalten. Diese wurden meist sehr spontan geplant und oft per Skype abgehalten.

## 7.2 Vorgehensmodell

Da wir bereits im Software Engineering Projekt RUP kennenlernten und damit gute Erfahrungen machten, entschieden wir uns auch bei dieser Arbeit für RUP als Vorgehensmodell. Innerhalb der einzelnen Projekt-Phasen wurde mit agilen 2 Wochen Sprints gearbeitet. Dies erlaubte uns sofort auf Veränderungen zu reagieren und somit das Projekt in die richtige Richtung zu steuern.

## 7.3 Software-Entwicklungsprozess

Wir wollten den Software-Entwicklungsprozess so einfach wie möglich gestalten. Zugleich wollten wir so wenig Zeit wie möglich für die Systembetreuung aufwenden. Aus diesem Grund haben wir uns entschieden, diverse Services von Atlassian zu beziehen.

Grundsätzlich entwickelten wir die Benchmarksoftware und das ToffiAnalyser-Spring Boot Backend mit der Eclipse IDE. Für die Angular2 SPA setzten wir auf Visual Studio Code als Editor, da dieser eine sehr gute Typescript-Unterstützung bietet. Als SCM kam für beide Teammitglieder seit Anfang an nichts Anderes in Frage als Git. Um den Code nun im Git komfortabel verwalten zu können, setzten wir auf SourceTree von Atlassian.

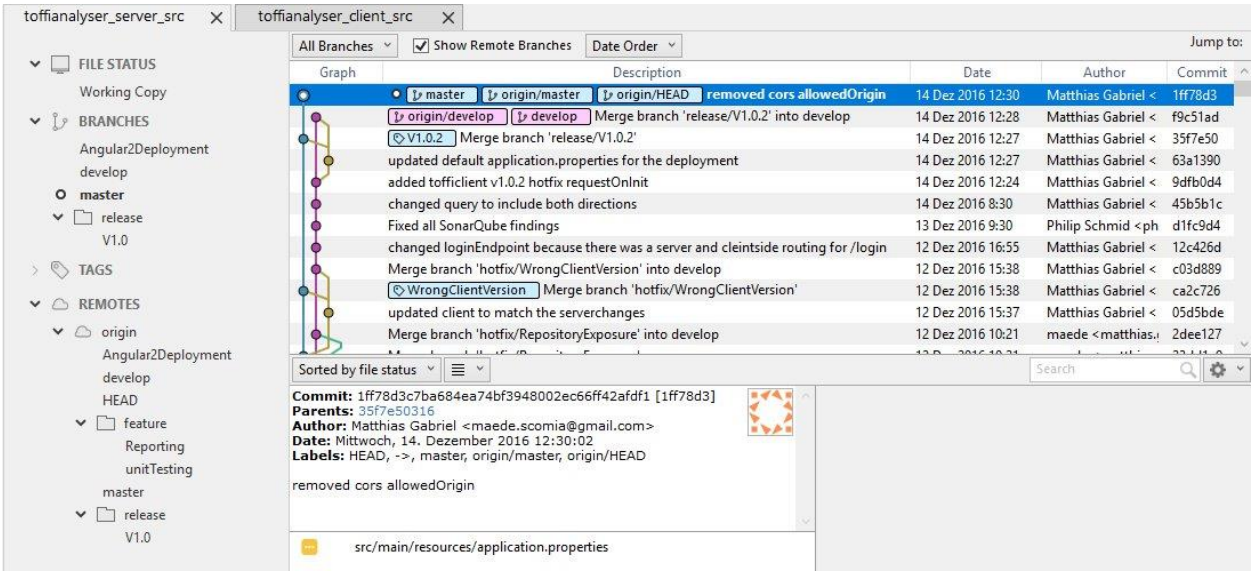


Abbildung 7-1 Komfortables Verwalten von Git Repositories mittels SourceTree

Der Sourcecode wurde auf Bitbucket in der Cloud gehostet, was uns ersparte, einen eigenen Git Service oder gar Server zu betreiben. Wiederum gleich an die jeweiligen Bitbucket Repositories angebunden, verwendeten wir Bitbucket Pipelines für das CI. Mehr Details zum Thema CI können dem Kapitel II 5.3 Automatische Testverfahren (CI) entnommen werden.

Um alle Projektinformationen, den Status und die gesamte Kommunikation an ein und demselben Ort zu finden, entschieden wir uns, Slack[36] als Team Collaboration Tool einzusetzen. Gegliedert nach Channels konnten wir uns somit mittels Integrationen für Jira, Pipelines oder Bitbucket automatisiert über neue Statusupdates im jeweiligen Produkt informieren lassen. Wurde zum Beispiel automatisch nach einem Git Push ein neuer CI Build gestartet, wurden wir entsprechend informiert.

[36] (Slack Technologies, kein Datum)

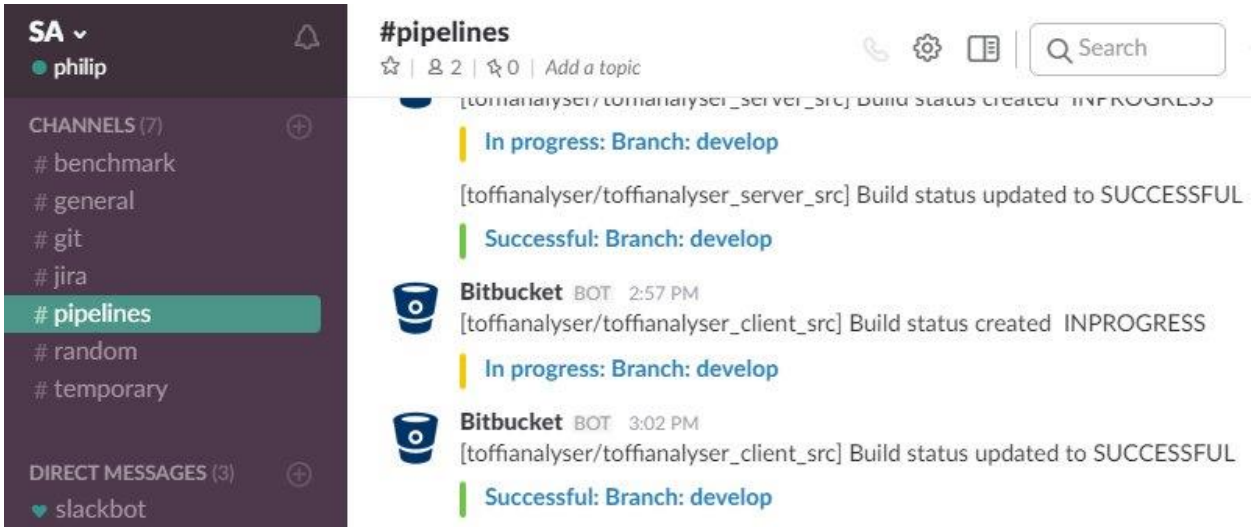


Abbildung 7-2 Slack mit den verschiedenen Kommunikations-Channels

Um die Software-Qualität zu überprüfen, haben wir eine von der HSR bereitgestellten Ubuntu 16.04 VM SonarQube[37] als Docker-Container eingerichtet. Dies erlaubte es uns, auf ein manuelles Kommando hin („mvn sonar:sonar“) den Sourcecode direkt aus dem Eclipse heraus an SonarQube zu schicken und analysieren zu lassen.

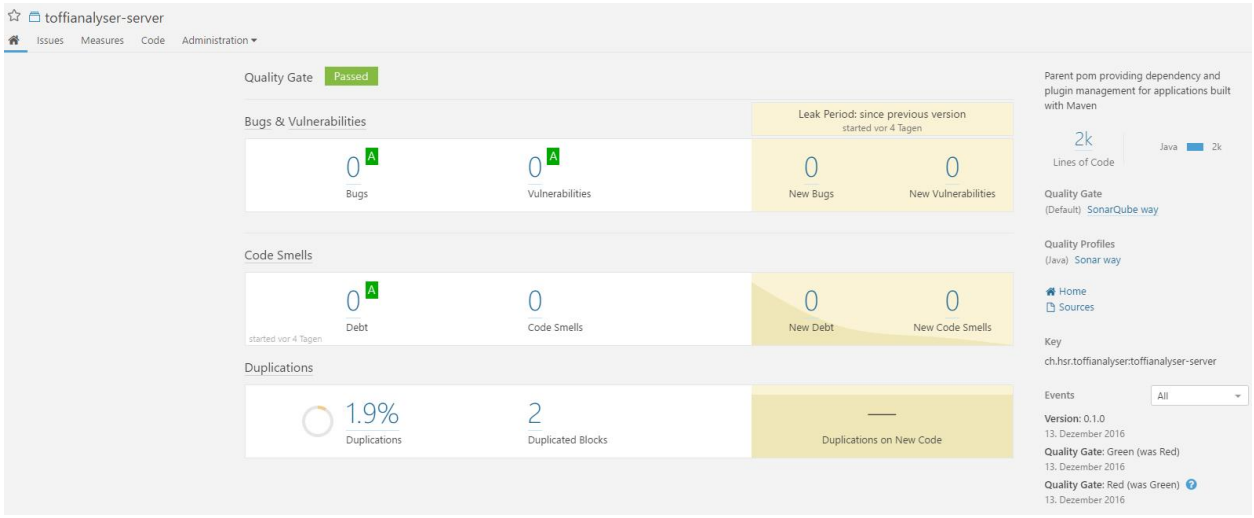


Abbildung 7-3 ToffiAnalyser auf SonarQube

Mehr zum Thema SonarQube kann dem Kapitel II 8.2 Codestatistik entnommen werden.

[37] (SonarSource SA, kein Datum)

## 7.4 Releases

Für die ToffiAnalyser-Software waren vier Releases vorgesehen.

| Nr. | Name | Erreicht am |
|---|---|---|
| RL1 | Prototyp | 26.10.2016 |
| RL2 | Alpha | 22.10.2016 |
| RL3 | Beta | 09.12.2016 |
| RL4 | Final Release (1.0) | 14.12.2016 |

Tabelle 7-1 ToffiAnalyser-Releases

## 7.5 Meilensteine

Für diese Semesterarbeit wurden 5 Meilensteine vorgesehen.

| Nr. | Name | Beschreibung | Datum |
|---|---|---|---|
| MS1 | End of Inception | Die Requirements und Risiken wurden erfasst und die Toolchain wurde definiert. | 30.09.2016 |
| MS2 | End of Elaboration | Der Benchmark sollte lauffähig sein und ein Toffi-Analyser-Prototyp, welcher durch alle Architektur-Layers geht, soll bestehen. | 28.10.2016 |
| MS3 | Feature Freeze | Ab diesem Zeitpunkt dürfen keine Features mehr in die Software eingebaut werden. Die Software muss in einem guten Zustand vorliegen. | 09.12.2016 |
| MS4 | End of Construction | Alle restlichen Bugs sind gefixt und die Software ist bereit für die Transition Phase. | 16.12.2016 |
| MS5 | Projektabschluss | Alle Dokumente wurden auf einen abgabewürdigen Stand gebracht und das Projekt wird abgeschlossen. | 23.12.2016 |

Tabelle 7-2 Meilensteine

## 7.6 Team

| Name | Rolle | Zuständigkeiten |
|---|---|---|
| Prof. Stefan Keller | Studienarbeit-Betreuer | Verantwortlich für die Semesterarbeit und die Betreuung der Entwickler |
| Matthias Gabriel | Entwickler | Zuständig für die Software-Architektur und das Backend (Serverkomponente & Logik in der SPA) |
| Philip Schmid | Entwickler | Zuständig für das GUI Design, das Frontend und die Infrastruktur inkl. Deployment |

Tabelle 7-3 Zuständigkeiten der Projektbeteiligten

## 7.7 Projektplan

| | Startdatum | KW | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 19.09.2016 | Datum | 19. Sep | 26. Sep | 03. Okt | 10. Okt | 17. Okt | 24. Okt | 31. Okt | 07. Nov | 14. Nov | 21. Nov | 28. Nov | 05. Dez | 12. Dez | 19. Dez |
| **Aktivität** | **Datum** | **Verantwortlich** | | | | | | | | | | | | | | |
| **Inception** | | | | | | | | | | | | | | | | |
| Setup Toolchain | 20.09.2016 | Team | | | | | | | | | | | | | | |
| Project initialization | 23.09.2016 | Team | | | | | | | | | | | | | | |
| Risikoanalyse | 27.09.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| Requirements grob / Scopedefinition | 30.09.2016 | Philip Schmid | | | | | | | | | | | | | | |
| **MS 1 End of Inception** | **30.09.2016** | **Team** | | ◆ | | | | | | | | | | | | |
| **Elaboration** | | | | | | | | | | | | | | | | |
| Einarbeiten | 07.10.2016 | Team | | | | | | | | | | | | | | |
| Benchmark | 21.10.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| Domain Model | 14.10.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| Requirements fein / Use Cases / NFR | 24.10.2016 | Philip Schmid | | | | | | | | | | | | | | |
| GUI Entwurf (Wireframe) | 19.10.2016 | Philip Schmid | | | | | | | | | | | | | | |
| Deployment Diagramm | 27.10.2016 | Philip Schmid | | | | | | | | | | | | | | |
| *Architcture Prototype* | 28.10.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| **MS 2 End of Elaboration** | **28.10.2016** | **Team** | | | | | | ◆ | | | | | | | | |
| **Construction** | | | | | | | | | | | | | | | | |
| Alpha | 18.11.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| Reporting | 25.11.2016 | Philip Schmid | | | | | | | | | | | | | | |
| Beta | 09.12.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| **MS 3 Feature Freeze** | **09.12.2016** | **Team** | | | | | | | | | | | | ◆ | | |
| Testphase / Bug Fixing | 14.12.2016 | Matthias Gabriel | | | | | | | | | | | | | | |
| **MS 4 End of Construction** | **16.12.2016** | **Team** | | | | | | | | | | | | | ◆ | |
| **Transition** | | | | | | | | | | | | | | | | |
| Dokumentation fertigstellen | 21.12.2016 | Philip Schmid | | | | | | | | | | | | | | |
| **MS 5 Projektabschluss** | **23.12.2016** | **Team** | | | | | | | | | | | | | | ◆ |

Legende:
◆ Meilensteine

Abbildung 7-4 Projekt Grobplanung

## 7.8 Risiken

| Nr | Titel | Beschreibung | max. Schaden [h] | Eintrittswahr-scheinlichkeit | Gewichteter Schaden | Vorbeugung | Verhalten beim Eintreten | Risio eliminiert | Risiko eliminiert durch / Status |
|---|---|---|---|---|---|---|---|---|---|
| R1 | JIRA nicht verfügbar | Der JIRA online Service steht wegen eines Problems nicht zur Verfügung. | 4 | 5% | 0.2 | Keine Möglichkeit | Normal weiterarbeiten, alle Projektmanagement-Dinge manuell in einem Excel dokumentieren | OK | NICHT EINGETRETEN |
| R2 | Software Versionen | Gewisse Software-Versionen werden sich während der Projektlaufzeit verändern. | 0 | 99% | 0 | Die verwendete Software wird zu Projektbeginn mit der Version festgelegt und während der Projektdauer nicht angepasst. | - | OK | Software Versionen wurden festgelegt. |
| R3 | Software Architektur Fehler | Die Software Architektur könnte Fehler enthalten und muss korrigiert werden. | 16 | 25% | 4 | Bis End. of Elab. muss ein Prototyp bestehen, welcher alle wichtigen Teile der Software umfasst, um Designfehler frühstmöglich erkennen zu können. | Allfällige Designfehler müssen im Team besprochen und eine Gegenmassnahme bestimmt werden. Gegebenfalls wird externe Hilfe hinzugezogen. | OK | Es wurde ein Prototyp erstellt, welcher durch alle Tiers/Layers geht und somit die Machbarkeit beweist. |
| R4 | Git Repository | Git Repository wird aufgrund eines Fehlers korrupt. | 15 | 10% | 1.5 | Einmal im Tag sollte das gesamte Git Repository durch einen automatischen Job | Backup zurückspielen | OK | NICHT EINGETRETEN |
| R5 | Bitbucket nicht verfügbar | Bitbucket ist nicht erreichbar, dadurch kann nicht auf das Repository zugegriffen werden | 8 | 1% | 0.04 | Keine Möglichkeit | In der Zwischenzeit sollte mit lokalen Commits gearbeitet werden bis der Service wieder verfügbar ist. Sollte auch nach längerer Zeit BitBucket noch nicht wieder online sein, wird bei den lokalen Git Repositories auf eine neue Remote gewechselt (z.B. Github). | OK | NICHT EINGETRETEN |
| R6 | AlchemyJS | AlchemyJS wird nicht mehr aktiv weiterentwickelt. Dadurch könnte es zu Problemen kommen | 15 | 30% | 4.5 | Einsetzen von AlchemyJS im Prototypen | Es wird direkt D3 eingesetzt, auf welchem AlchemyJS basiert. | OK | Es wurde ein Evaluation von Visualierungslibraries durchgeführt wobei Alchemy.js ausgeschieden ist. |
| R7 | Pipelines | Pipelines ist erst im Beta-Stadium, dadurch kann es sein, dass zu wenig Dokumentation vorhanden ist | 8 | 10% | 0.8 | Keine Möglichkeit | Kontaktieren des Supports von Pipelines | OK | NICHT EINGETRETEN |
| R8 | NetFlow Testdaten unbrauchbar | Die gesammelten Testdaten sind unbrauchbar und müssen ersetzt werden | 16 | 5% | 0.8 | Frühzeitige Überprüfung der Brauchbarkeit der Testdaten | Es müssen neue Testdaten besorgt werden, eventuell andere Konfiguration beim Collector der Daten | OK | Die Testdaten konnten sowohl für den Benchmark als auch den ToffiAnalyser Prototypen verwendet werden und stellen somit kein Problem dar. |
| R9 | Zuwenig Testdaten | Es sind für den Benchmark zu wenig Testdaten verfügbar | 6 | 25% | 1.5 | - | Es müssen weitere Testdaten besorgt werden | OK | Risiko eingetreten, es mussten für den Benchmark zusätzliche Daten besorgt werden. Dies hat die Ausführung der definitiven Benchmarks im Zeitplan des Projektes verschoben |
| R10 | Dokument korrupt | Ein Dokument auf OneDrive wird korrupt und kann nicht mehr geöffnet werden. | 4 | 10% | 0.4 | Der gesammte OneDrive Ordner wird einmal pro Woche (Abend vor dem Meeting) manuell auf das Dokumenten Backup Repository kopiert und somit gesichert. | Dokument, sofern als möglich, aus der OneDrive History restoren oder im schlimmsten Fall aus dem Dokumenten Backup Repo holen. | OK | NICHT EINGETRETEN |
| **Summe** | | | **92** | | **13.74** | | | | |

Abbildung 7-5 Technische Risiken

# 8. Projektmonitoring

## 8.1 Projektauswertung

In den folgenden Unterkapiteln sind diverse Auswertungen zum Verlauf des Projektes als Chart aufgeführt.

### 8.1.1 Arbeitszeit

In den folgenden Diagrammen wird aufgeschlüsselt, welches Teammitglied in welcher Phase wie viel Zeit aufgewendet hat.

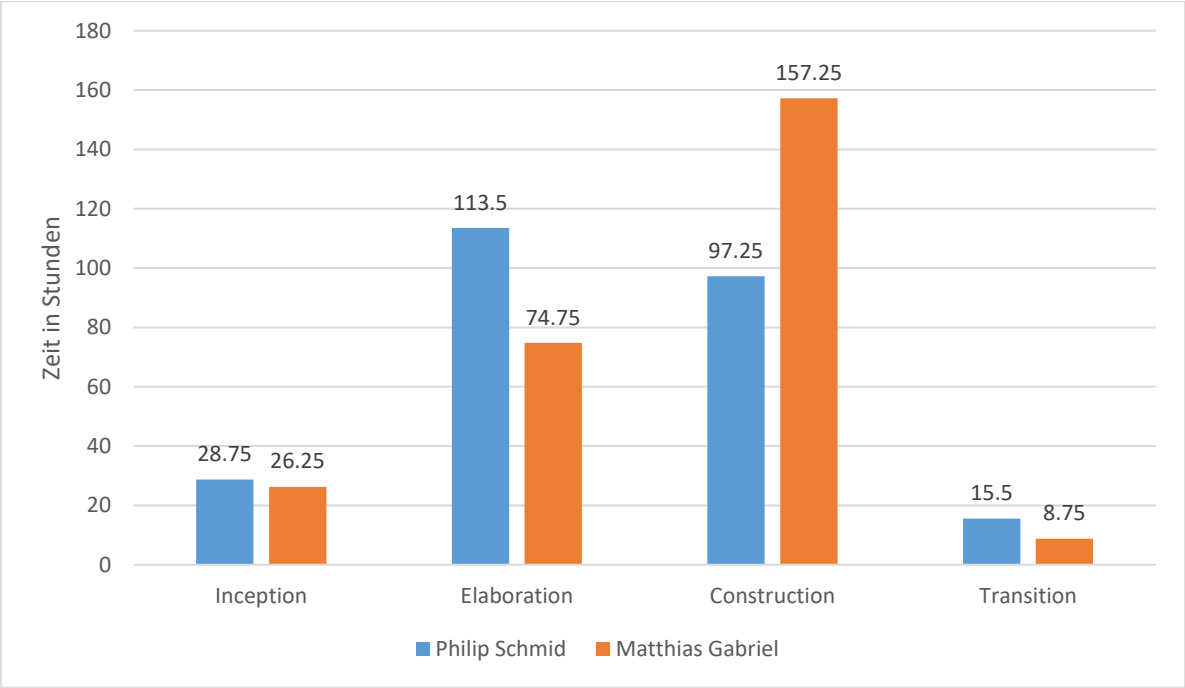


Abbildung 8-1 Ist-Zeit [h] pro Teammitglied und Projektphase

Für die Semesterarbeit ist pro Student eine Soll-Arbeitszeit von rund 240 Stunden als Vorgabe gegeben. Im Projekt existierten insgesamt 13 Epics.

| **Epic** | **Philip Schmid** | **Matthias Gabriel** |
|---|---|---|
| Alpha | 50.75 | 55.75 |
| Architektur Prototype | 21.5 | 17 |
| Benchmark | 80 | 56.25 |
| Beta | 15 | 28.5 |
| Dokumentation | 35.25 | 56 |
| Einarbeiten | 6 | 22.5 |
| GUI Entwurf | 1.5 | |
| Project initialization | 3.75 | 1 |
| Reporting | | 8.5 |
| Requirements grob | 5 | |
| Risikoanalyse | 0.5 | 1.25 |
| Setup Toolchain | 11.5 | 2 |
| Weekly Meeting | 24.25 | 18.25 |
| **Total [h]:** | **255** | **267** |

Tabelle 8-1Ist-Zeit [h] pro Teammitglied und Epic

### 8.1.2 Projektphase

In dem folgenden Diagramm wird aufgezeigt, wie viele Issues es pro Projektphase gegeben hat. Es werden ausschliesslich abgeschlossene Issues mitgezählt.

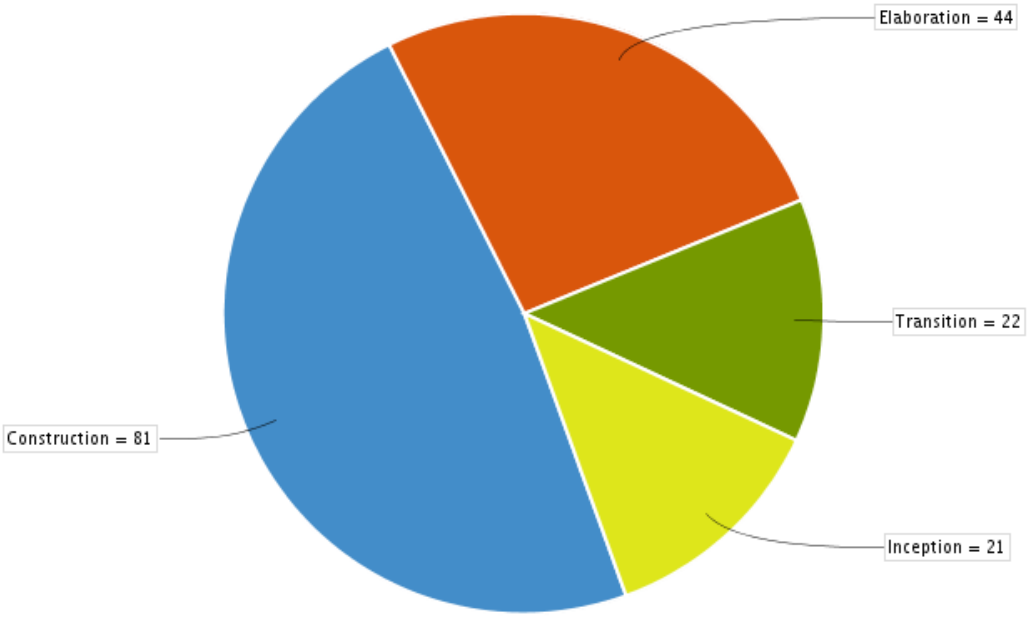


Abbildung 8-2 Anzahl Issues pro Projektphase

Nachstehend aufgezeigt die aufgewandte Zeit pro Projektphase.

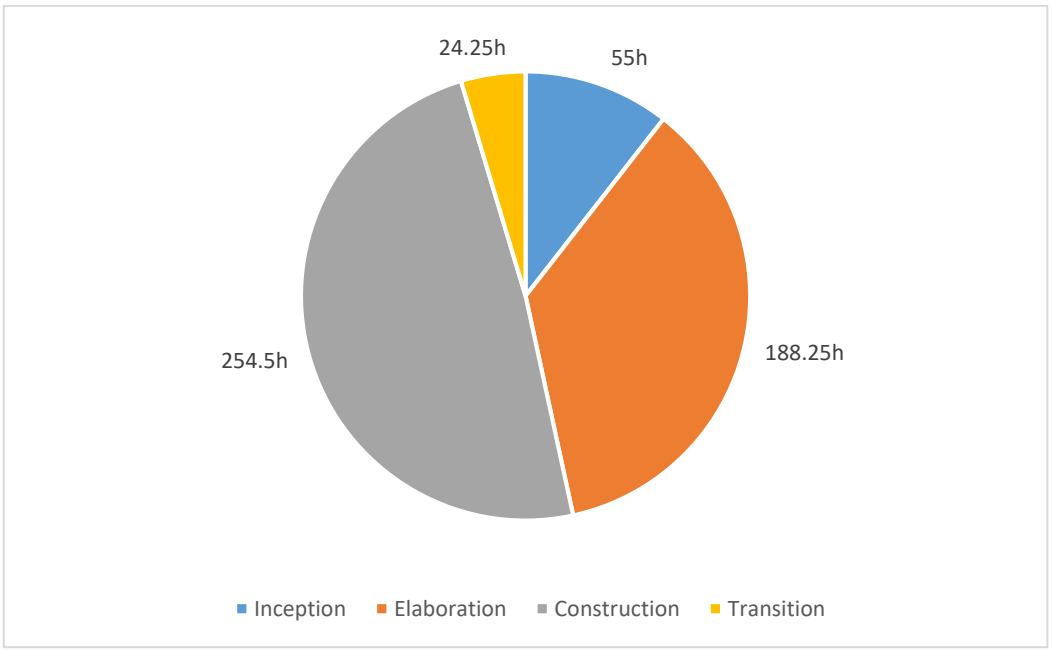


Abbildung 8-3 Aufgewandte Zeit pro Projektphase

### 8.1.3 Issuetyp

Die folgende Grafik zeigt auf, wie viele Issues es pro Issuetyp gegeben hat.

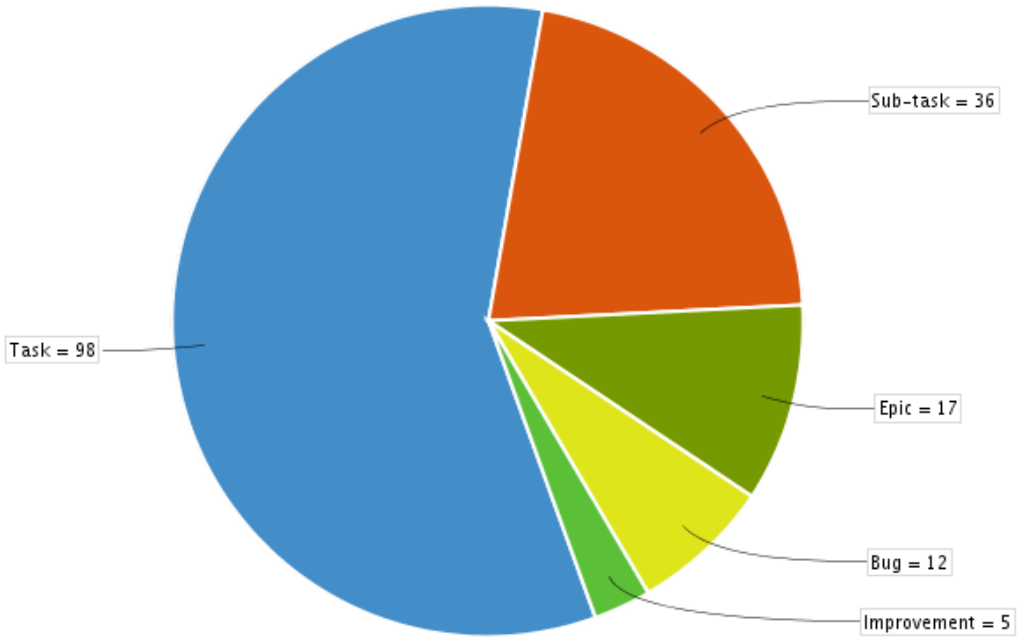


Abbildung 8-4 Anzahl Issues nach Issuetyp gruppiert

## 8.2 Codestatistik

### 8.2.1 Lines of Code (LOC)

Für die ToffiAnalyser-Software wurden mehrere Programmier- und Abfragesprachen verwendet.

| Software | Sprache | LOC |
|---|---|---|
| SPA | Typescript (JavaScript) | 2693 |
| SPA | HTML | 465 |
| SPA | CSS | 90 |
| Server | Java | 1975 |
| Server | Cypher | 105 |
| Randomisation Script | Python | 92 |
| Deployment | Bash | 73 |
| Deployment | Dockerfile (inkl. Docker-Compose) | 55 |
| **Total** | | **5548** |

Tabelle 8-2 ToffiAnalyser-Software Lines of Code

LOC für die Benchmark Software betrachtet, ergibt die folgende Anzahl an Lines.

| Software | Sprache | LOC |
|---|---|---|
| Benchmark | Java | 2080 |
| Benchmark | SQL | 225 |
| Benchmark | Cypher | 41 |
| **Total** | | **2346** |

Tabelle 8-3 Benchmark Software Lines of Code

Total auf die gesamte Semesterarbeit ergibt sich die folgende Lines of Code Anzahl.

| Software | LOC |
|---|---|
| ToffiAnalyser | 5548 |
| Benchmark | 2346 |
| **Total** | **7894** |

Tabelle 8-4 Totale Anzahl Lines of Code

# III. Benchmark

# 1. Vorwort

Im Rahmen dieser Semesterarbeit sollte neben der eigentlichen Software ToffiAnalyser auch ein Benchmark durchgeführt werden. Besonders brisant am Benchmark ist, dass ein relationales Datenbanksystem mit NoSQL Graphdatenbanksystemen verglichen wird – und das Ganze erst noch mit speziell gut geeigneten Testdaten für Graphdatenbanksysteme. Im Benchmark selbst wurde das relationale Datenbanksystem PostgreSQL mit den NoSQL Graphdatenbanksystemen Neo4j und OrientDB verglichen.

Die Schwierigkeit dieses Benchmarks besteht hauptsächlich darin, eine Möglichkeit zu finden, um relationale Datenbanksysteme überhaupt mit NoSQL Graphdatenbanksysteme in einem sinnigen Rahmen vergleichen zu können. Des Weiteren war es das Ziel, dass der Benchmark auf einem einzelnen System lauffähig ist und das kein DBMS Clustersetup getestet wird.

# 2. Machbarkeitsprüfung & Recherche

Vor dem Benchmark galt es als Erstes zu prüfen, wie genau vorgegangen werden kann und ob es schon entsprechende Datenbank-Benchmark Lösungen auf dem Markt gibt.

Um den Ablauf des Benchmarks strukturiert und sinnig zu gestalten, wurde eine Anlehnung an bereits erfolgreiche Benchmarks in Betracht gezogen. Einerseits wurde das Vorgehen durch den HSR Texas Geo Database Benchmark[38] inspiriert und andererseits durch ein HSR Master Seminar[39], in welchem es im das Thema Benchmarking und Performance von (R)DBMS ging.

Eine zusätzliche entsprechende Recherche nach bestehenden Datenbank-Benchmarksoftware verdeutlichte, dass keine vergleichbaren Datenbank-Benchmarks existieren, welche die Schwierigkeiten dieses Benchmarks meistern. Einerseits war unter anderem ein NoSQL Benchmark von ArangoDB[40] in der näheren Auswahl und andererseits ein GraphDB Benchmark auf Github von «socialsensor»[41]. Die Benchmarksoftware von Socialsensor schied leider aus, da sie keinerlei Unterstützung für relationale Datenbanksysteme eingebaut hatte. Der Benchmark von ArangoDB wiederum schien nicht schlecht geeignet zu sein. Er setzt auf Node.js als Backend für die Benchmark-Logik. Da der eigene Benchmark jedoch insbesondere auch als Vorbereitung für die Toffi-Analyser-Software diente, haben wir einen Mehrwert darin gesehen, den Benchmark gleich mit NetFlow Testdaten und mit einer Java-Anbindung zu realisieren.

Aus diesen Gründen wurde entschieden, für den Benchmark in Eigenregie ein Java-Programm zu entwickeln. Dieses Programm soll die verschiedenen Testqueries beliebig oft auf den verschiedenen Datenbanksystemen ausführen und zugleich je die Ausführungszeit messen. Die Resultate sollen beim Beenden des Testlaufs in ein CSV exportiert werden, damit anschliessend manuell Rückschlüsse gezogen und Interpretationen gemacht werden können. Die Queries sollten möglichst so gewählt werden, dass die wichtigsten Querytypen vorhanden sind.

[38] (Keller, HSR Texas Geo Database Benchmark, 2014)

[39] (Keller, Datenbanksysteme im HS11 für Master (MSE), 2012)

[40] (Weinberger, 2015)

[41] (SocialSensor, 2016)

# 3. Vorbereitungen

## 3.1 Hardware & Betriebssystem

Bei den Benchmark-Systemen handelt es sich um folgende Hardware:

| | |
|---|---|
| **System Typ** | Workstation |
| **Model** | Fujitsu Celsius W530 |
| **CPU** | Quad-Core mit Hyper-Threading aktiviert (entspricht 8 Cores)<br>(Intel(R) Xeon(R) CPU E3-1245 v3 @ 3.40GHz) |
| **RAM** | 16GB @ DDR3 1600 MHz |
| **HDD** | 256GB SAMSUNG MZ7TD256 |
| **Betriebssystem** | Ubuntu Server 16.04.1 LTS |
| **Kernel** | 4.4.0-47-generic |

Tabelle 3-1 Hardwarespezifikationen

Namentlich wurden folgende 2 HSR SA Zimmer PCs (Raum 1.258) dafür verwendet:

Workstation 1: PIN1258007

Workstation 2: PIN1258008

Auf diesen beiden PCs wurde je ein Ubuntu Server 16.04.1 LTS installiert. Die Installationen haben folgende, von der Standard-Installation abweichende, Konfigurationen:

- Security Updates werden automatisch installiert
- Zusätzlich installierte Software:
    - OpenSSH Server (wird zur vereinfachten Administration gebraucht)

## 3.2 Software

Auf beiden Systemen wurden je die folgende Software installiert:

- **PostgreSQL (9.5+173)**: PostgreSQL wurde über die offiziellen Ubuntu Repositories bezogen und installiert.
- **OrientDB (2.2.11)**: Für die OrientDB Installation wurde nach der Anleitung von digitalocean.com[42] vorgegangen.
- **Neo4j (3.0.6)**: Neo4j gemäss offizieller Anleitung[43] vom Hersteller installiert.
- **Oracle Java (8u101)**: Java wurde gemäss der Anleitung von DigitalOcean[44] installiert.

## 3.3 Testdaten

Für den Benchmark stehen genau 1 Million Flow Entries, im JSON Format vorliegende Testdaten zur Verfügung. Dies entspricht ungefähr 500 MB. Die Testdaten wurden uns vom INS Institute for Networked Solutions der HSR zur Verfügung gestellt.

Wir haben das selbstgeschrieben Randomisation-Script genutzt (siehe Kapitel II 5.1.5 Randomisation-Script), um eine für einige Abfragetypen 2 Millionen Datensätze zur Verfügung zu haben

[42] (Fox, 2016)

[43] (Neo Technology, Inc, kein Datum)

[44] (Vlaswinkel, 2014)

Ein einzelner Entry hat dabei folgende Struktur:


```
{
   "@timestamp":"2016-07-01T23:52:00.000Z",
   "netflow":{
      "version":9,
      "flow_seq_num":1881,
      "flowset_id":257,
      "ipv4_src_addr":"152.96.122.161",
      "ipv4_dst_addr":"152.96.120.200",
      "protocol":17,
      "src_tos":0,
      "l4_src_port":54999,
      "l4_dst_port":53,
      "input_snmp":151060602,
      "output_snmp":371965952,
      "direction":0,
      "src_as":0,
      "dst_as":0,
      "ipv4_next_hop":"10.90.0.1",
      "tcp_flags":0,
      "in_bytes":133,
      "in_pkts":1,
      "first_switched":"2016-07-01T23:51:55.000Z",
      "last_switched":"2016-07-01T23:51:55.000Z"
   },
   "@version":"1",
   "host":"10.20.0.1"
}
```


Abbildung 3-1 NetFlow Beispieldatensatz

Bei dem aufgeführten JSON Datensatz handelt es sich um ein NetFlow Packet, welches über LogStash[45] empfangen und verarbeitet wurde.

[45] (Elasticsearch BV, kein Datum)

| Attribut | Beschreibung |
|---|---|
| **@timestamp** | Ein von LogStash eingefügter Zeitstempel, welcher beim Empfang des NetFlow Packets gesetzt wird. |
| **netflow**[46] | Ein proprietäres Protokoll von Cisco Systems Inc, welches in der Lage ist, auf zentralen Netzwerkgeräten Informationen zum Netzwerkverkehr zu sammeln (genannt Exporter) und diese an einen Collector weiterzuleiten. |
| **version** | Version des NetFlow Packets. |
| **flow_seq_num** | Sequenznummer, welche vom NetFlow Exporter pro versendetem NetFlow Packet inkrementiert wird. |
| **flowset_id** | Template ID, welche auf ein zuvor empfangenes Template verweist. Diese Templates werden vom Exporter periodisch an den Collector verschickt und enthalten Informationen, wie die anschliessend eintreffenden binären NetFlow Packets geparst werden müssen. |
| **ipv4_src_addr** | IPv4 Adresse des Senders der Packets. |
| **ipv4_dst_addr** | IPv4 Adresse des Empfängers der Packets. |
| **protocol** | ID für das IP Protokoll (z.B. 6 = TCP, 17 = UDP). |
| **src_tos** | Type of Service, welcher beim Einritt der Packets auf dem ankommenden Interface vergeben wird. |
| **l4_src_port** | TCP/UDP Port, über welchen die Packets verschickt wurden. |
| **l4_dst_port** | TCP/UDP Port, an welchen die Packets geschickt wurden. |
| **input_snmp** | SNMP ID es Exporter Interfaces, auf welchem die Packets ankamen. |
| **output_snmp** | SNMP ID es Exporter Interfaces, auf welchem die Packets hinausgingen. |
| **direction** | Richtung, welche der Flow hatte (0 = ankommend, 1 = ausgehend). |
| **src_as** | BGP AS Nummer des AS, von welchem die Packets herkamen. |
| **dst_as** | BGP AS Nummer des AS, in welches die Packets weitergeleitet wurden. |
| **ipv4_next_hop** | IP-Adresse des nächsten Hops der Packets. |
| **tcp_flags** | Kumulatives OR alles TCP Flags, welche auf dem Flow gesetzt waren. |
| **in_bytes** | Grösse des Flow in Anzahl Bytes. |
| **in_pkts** | Anzahl Packets, welche der Flow hatte. |
| **first_switched** | Zeitstempel, an welchem das erste Packet des Flows beim Exporter eintraf. |

[46] (Cisco Systems Inc, 2011)

| | |
|---|---|
| **last_switched** | Zeitstempel, an welchem das letzte Packet des Flows beim Exporter eintraf. |
| **@version** | LogStash Version, welche ausschliesslich für LogStash interne Angelegenheiten gebraucht wird. |
| **host** | IP-Adresse, woher das NetFlow Packet dem Collector geschickt wurde. |

Tabelle 3-2 NetFlow Protokoll Attribute und Beschreibung

Die oben beschriebenen Attribute stellen nur einen Teil, der vom Protokoll NetFlow zur Verfügung gestellten Attribute dar. Welche Attribute exportiert werden sollen, kann beliebig selbst auf dem Exporter Device konfiguriert werden. Die für diese Testdaten betätigte Konfiguration kann dem Anhang unter Punkt IV 4 NetFlow Konfiguration entnommen werden.

## 3.4 Geplantes Vorgehen

Für den Benchmark wurde das folgende Vorgehen mit den dazugehörigen Rahmenbedingungen festgelegt.

- Jede Benchmark-Query wird dreimal hintereinander laufen gelassen. Für die Auswertung können die einzelnen Durchläufe, aber auch der Median dieser drei verwendet werden.
- Der gesamte Benchmark wird durchgängig nur auf einem System laufen gelassen. Es wird auch kein DBMS Cluster oder ein Load-Balancing über mehrere Systeme konfiguriert.
- An den verschiedenen DBMS wird kein Tuning vorgenommen. Es wird die von der Installation standardmässig vorgegebene Konfiguration verwendet.
- Um volle Transparenz zu schaffen, sollen für alle DBMS die jeweilig verwendeten Benchmark-Queries als Prepared-Statement aufgeführt werden.
- Bei jedem DBMS wird die vom Hersteller empfohlene Schnittstelle verwendet. Bei Neo4j Bolt[47], bei PostgreSQL JDBC und bei OrientDB die Blueprints API[48].
- Währendem eine Query eines gewissen DBMS läuft, sollen vorgängig die anderen DBMS Services gestoppt werden, damit für die jeweilige Benchmark-Query die vollen Systemressourcen zur Verfügung stehen.
- Eine Benchmark-Query wird in verschiedenen Durchläufen mit unterschiedlich vielen Testdaten getestet. Es wird mit 100'000 Testdatensätzen gestartet und bis 1 Million wird die Testdatenmenge in 100’000-er Schritten pro Durchlauf erhöht. Nach dem 1 Million-Testdatensätzen-Durchlauf wird es noch einen Durchlauf mit 2 Millionen Testdatensätzen geben.
- Für den gesamten Benchmark werden grundsätzlich die gleichen Testdaten verwendet.

Für eine zusammengefasste Auswertung einzelner gleicher Durchläufe wurde auf den Median gesetzt, da der Median gegenüber dem Durchschnitt weniger anfällig für Extremwerte ist. Ein Durchschnitt über die drei Benchmark-Durchläufe pro Query und DBMS wäre im Falle eines grösseren Ausreissers stärker verzerrt, als dies mit dem Median der Fall wäre.

[47] (Neo Technology, Inc., kein Datum)

[48] (OrientDB LTD, kein Datum)

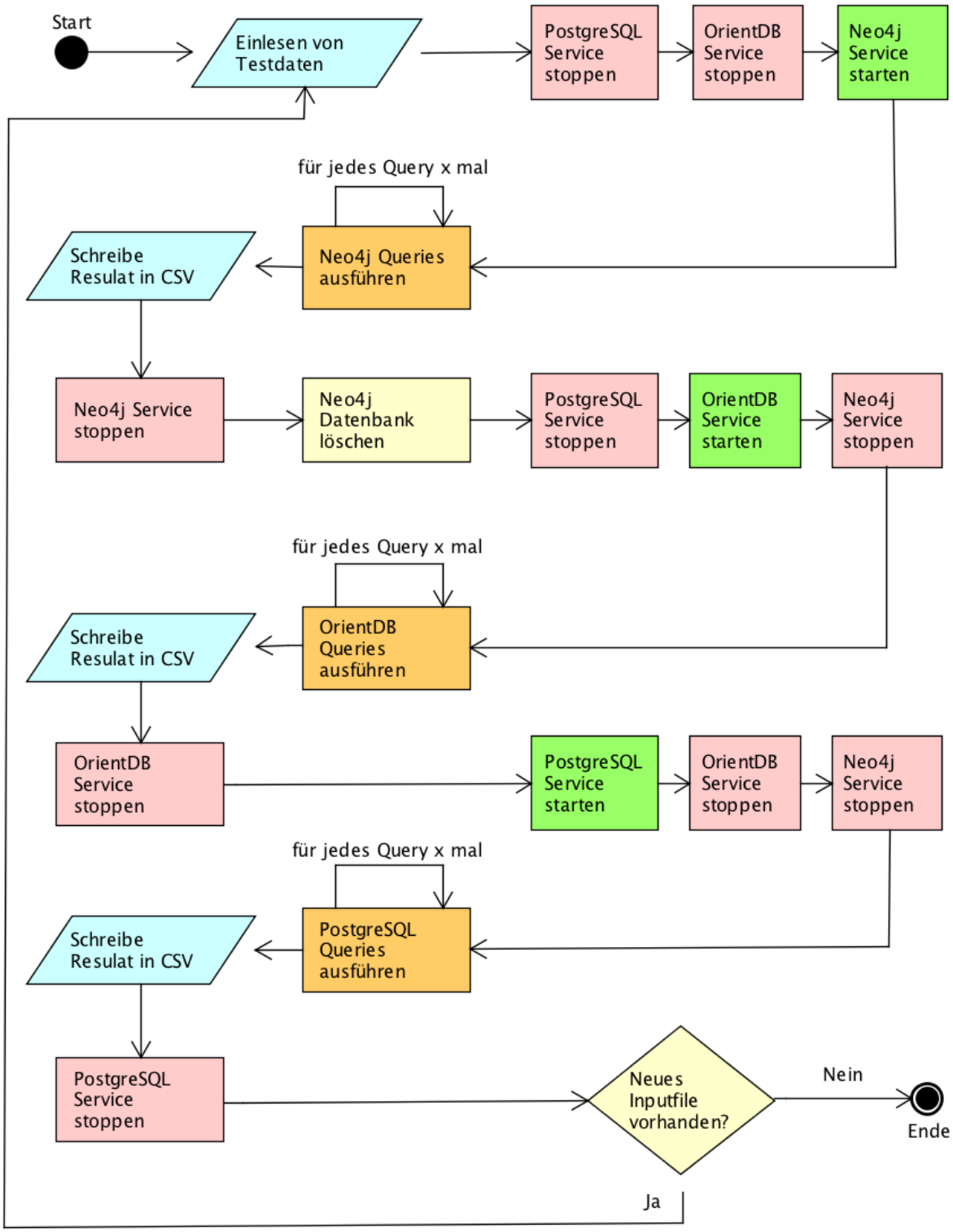


Abbildung 3-2 Benchmarkablauf

## 3.5 Datenmodell

Für den Benchmark wurde für die Graphdatenbanksysteme das gleiche Datenmodell verwendet, wie es auch in der ToffiAnalyser-Software eingesetzt wurde. Genauere Details können unter II 3 Analyse eingesehen werden.

Da dieses jedoch nicht ohne Weiteres auf ein RDBMS wie PostgreSQL adaptiert werden kann, wurde dafür ein separates relationales Datenmodell erstellt.

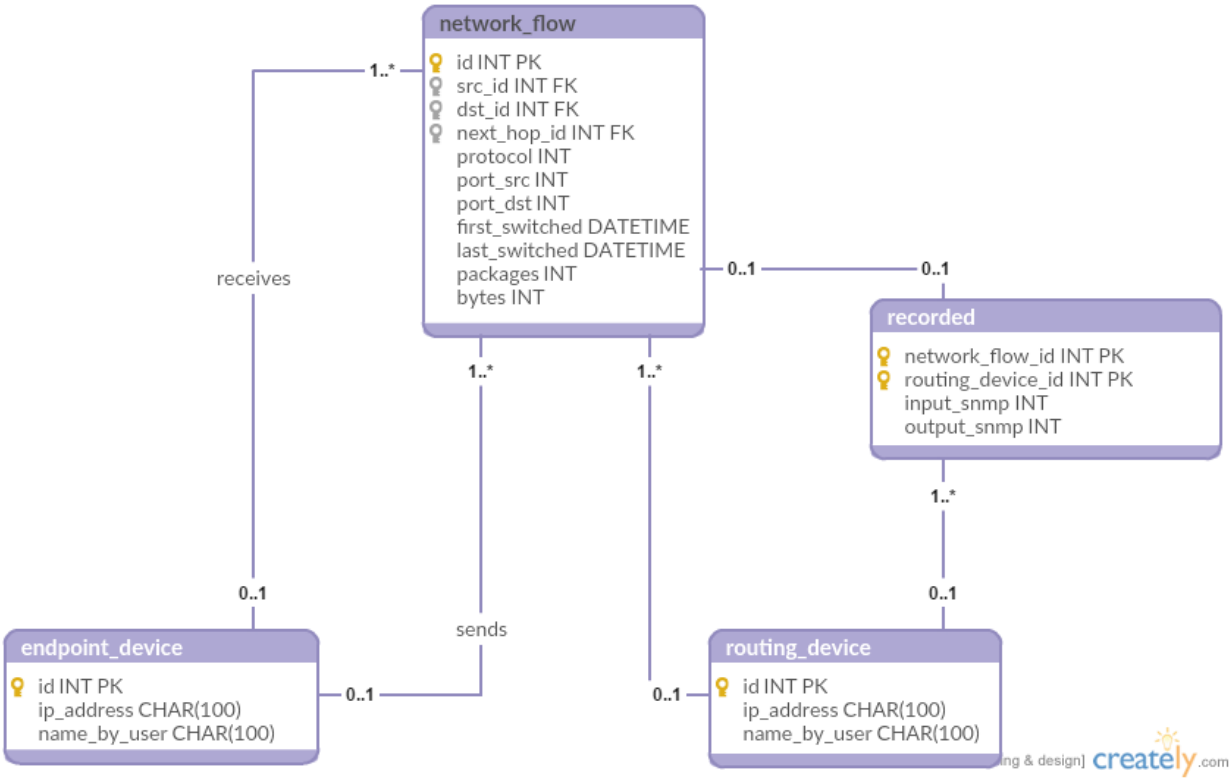


Abbildung 3-3 PostgreSQL Datenmodell

# 4. Queries

Die folgenden Queries wurden so gewählt, dass ein möglichst breites Spektrum an Querytypen gemessen werden kann.

## 4.1 Massive Insertion

Bei dieser Datenbankoperation geht es darum, viele Einträge auf einmal in eine leere Datenbank einzufügen.

### 4.1.1 Query

#### 4.1.1.1 Neo4j

```
//Insert routing_device
MERGE (rd:Network_Entity {ip_address:{ip_address}})
SET rd:Routing_Device
//Insert endpoint_device (Needed twice for source and destination)
MERGE (endpoint:Network_Entity {ip_address:{ip_address}})
SET endpoint:EndPoint_Device
//Insert Flow
MATCH (nextHop:Network_Entity {ip_address:{next_hop}}),
(src:Network_Entity {ip_address:{src_ip}}),
(dst:Network_Entity {ip_address:{dst_ip}})
CREATE (src) - [:SENDS] ->
(nf:Network_Flow {protocol:{protocol},port_src:{port_src},
  port_dst:{port_dst},first_switched:{first_switched},
  last_switched:{last_switched},packages:{packages},bytes:{bytes}})
<- [:RECEIVE] - (dst),
(nf) - [:NEXT_HOP] -> (nextHop)
```

#### 4.1.1.2 OrientDB

```
OrientVertex src = insertNetworkDevice(graph,
  "class:EndPoint_Device", networkFlowImport.getSrc());
OrientVertex dst = insertNetworkDevice(graph,
  "class:EndPoint_Device", networkFlowImport.getDst());
OrientVertex router = insertNetworkDevice(graph,
  "class:Routing_Device", networkFlowImport.getNextHop());
OrientVertex flow = graph.addVertex("class:Network_Flow",
  "protocol", networkFlowImport.getProtocol(),
  "port_src", networkFlowImport.getPort_src(),
  "port_dst", networkFlowImport.getPort_dst(),
  "first_switched", networkFlowImport.getFirstSwitched(),
  "last_switched", networkFlowImport.getLastSwitched(),
  "packages", networkFlowImport.getPackages(),
  "bytes", networkFlowImport.getBytes());
graph.addEdge("class:SENDS", src, flow, null);
graph.addEdge("class:RECEIVE", dst, flow, null);
graph.addEdge("class:NEXT_HOP", flow, router, null);
```

#### 4.1.1.3 PostgreSQL

```
-- Insert routing_device
INSERT INTO routing_device(ip_address)
SELECT ?
WHERE NOT EXISTS (
  SELECT ip_address
  FROM routing_device
  WHERE ip_address=?
);
-- Insert endpoint_device (twice needed, source and destination)
INSERT INTO endpoint_device(ip_address)
SELECT ?
WHERE NOT EXISTS (
  SELECT ip_address
  FROM endpoint_device
  WHERE ip_address=?
);
-- Insert network_flow
INSERT INTO network_flow(
  src_id, dst_id, next_hop_id, protocol,
  port_src, port_dst, first_switched,
  last_switched, packages, bytes)
  VALUES(
    (SELECT id FROM endpoint_device WHERE ip_address=?),
    (SELECT id FROM endpoint_device WHERE ip_address=?),
    (SELECT id FROM routing_device WHERE ip_address=?),
    ?,?,?,?,?,?,?
  );
```

### 4.1.2 Analyse der Ergebnisse

Beim «Massive Insertion» Benchmark-Teil kann festgestellt werden, dass sich die Einfüge-Zeiten linear zur Steigerung der Daten verhalten.

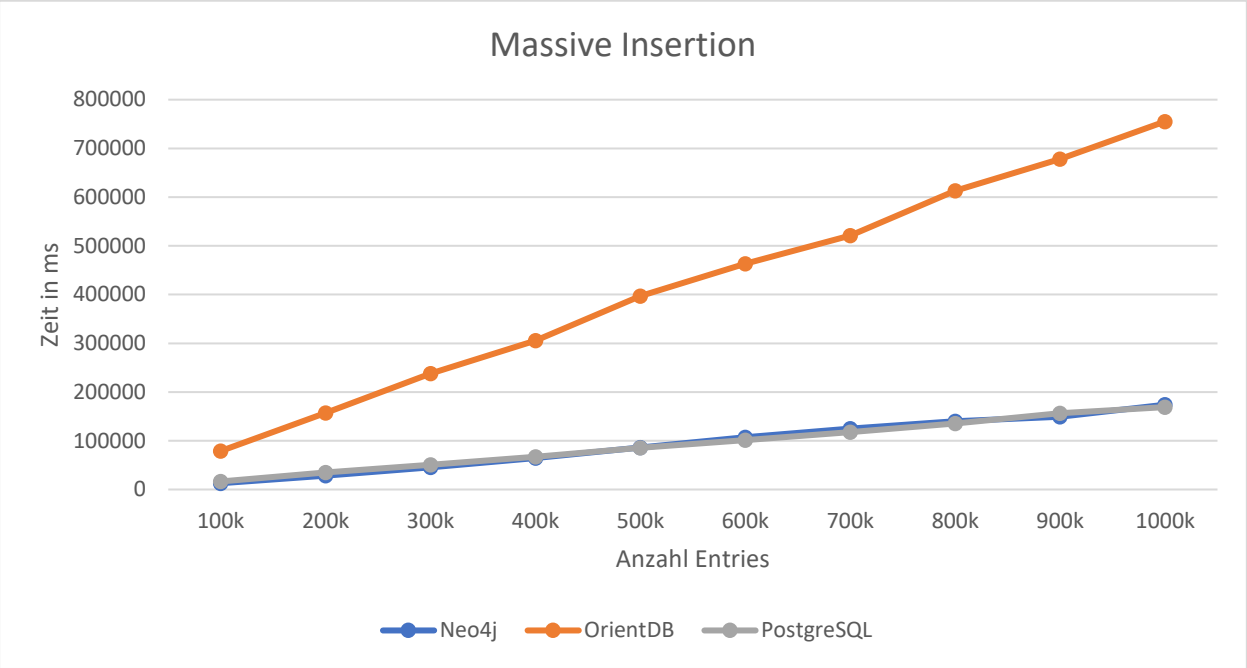


Abbildung 4-1 Massive Insertion

Wie aus dem Diagramm herausgelesen werden kann, braucht OrientDB um den Faktor länger für das Einfügen der gleichen Datenmenge, als dies bei Neo4j oder PostgreSQL der Fall ist. PostgreSQL und Neo4j liefern sich hingegen ein Kopf-an-Kopf-Rennen und haben jeweils ein Resultat mit einem Unterschied von maximal Faktor ±0.28 verglichen zueinander (Abbildung 4-2 Massive Insertion - Neo4j vs. PostgreSQL).

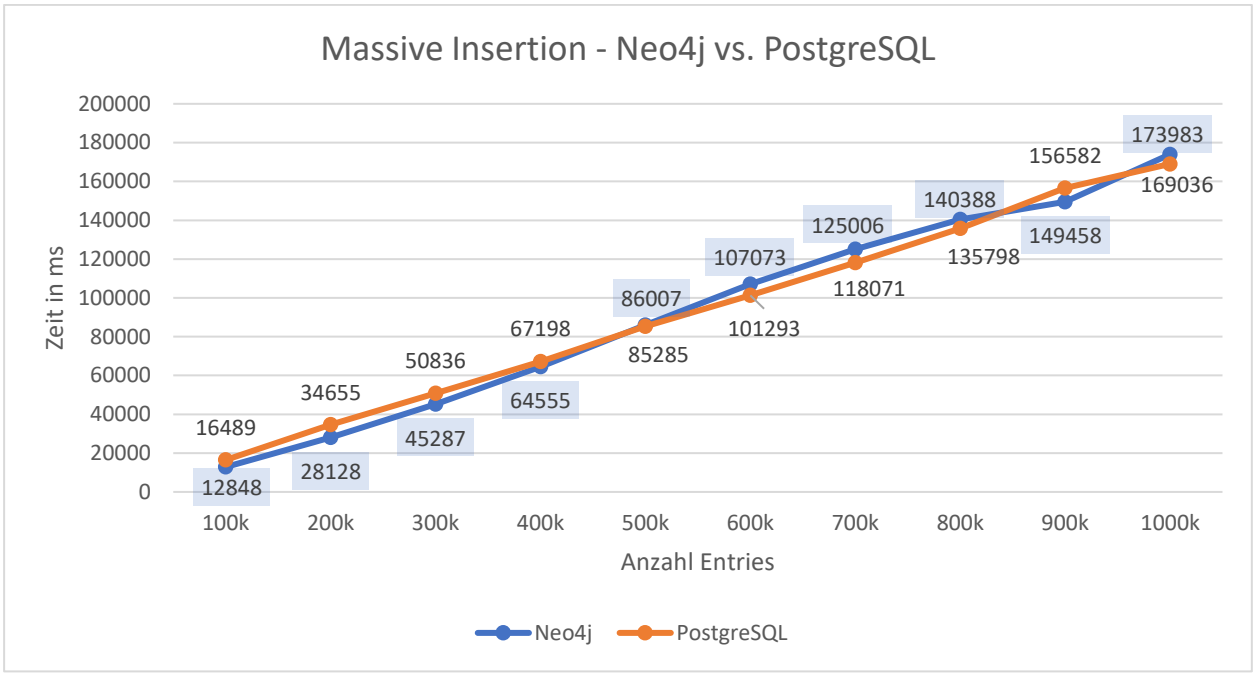


Abbildung 4-2 Massive Insertion - Neo4j vs. PostgreSQL

## 4.2 Equal Search

Diese Query dient dazu herauszufinden, wie sich die Datenbanksysteme beim Abfragen eines einzelnen Entries/Tupels bewähren. Gesucht wird mittels IP-Adresse eines bestimmten End-points-Devices.

### 4.2.1 Query

Bei dieser Query existiert ein Unique Index über das IP-Adressenfeld.

#### 4.2.1.1 Neo4j

```
MATCH (endpoint:EndPoint_Device {ip_address:{ip_address}})
RETURN endpoint.ip_address
```

#### 4.2.1.2 OrientDB

```
SELECT ip_address
FROM endpoint_device
WHERE ip_address = :node
```

#### 4.2.1.3 PostgreSQL

```
SELECT ip_address
FROM endpoint_device
WHERE ip_address=?
```

### 4.2.2 Analyse der Ergebnisse

Anhand dieser Benchmark-Query kann sehr schön gezeigt werden, wie ein Zugriff mit O(1) aussehen kann. PostgreSQL zeigt sich hier von der besten Seite mit einer Zugriffzeit von lediglich 3-4 ms.

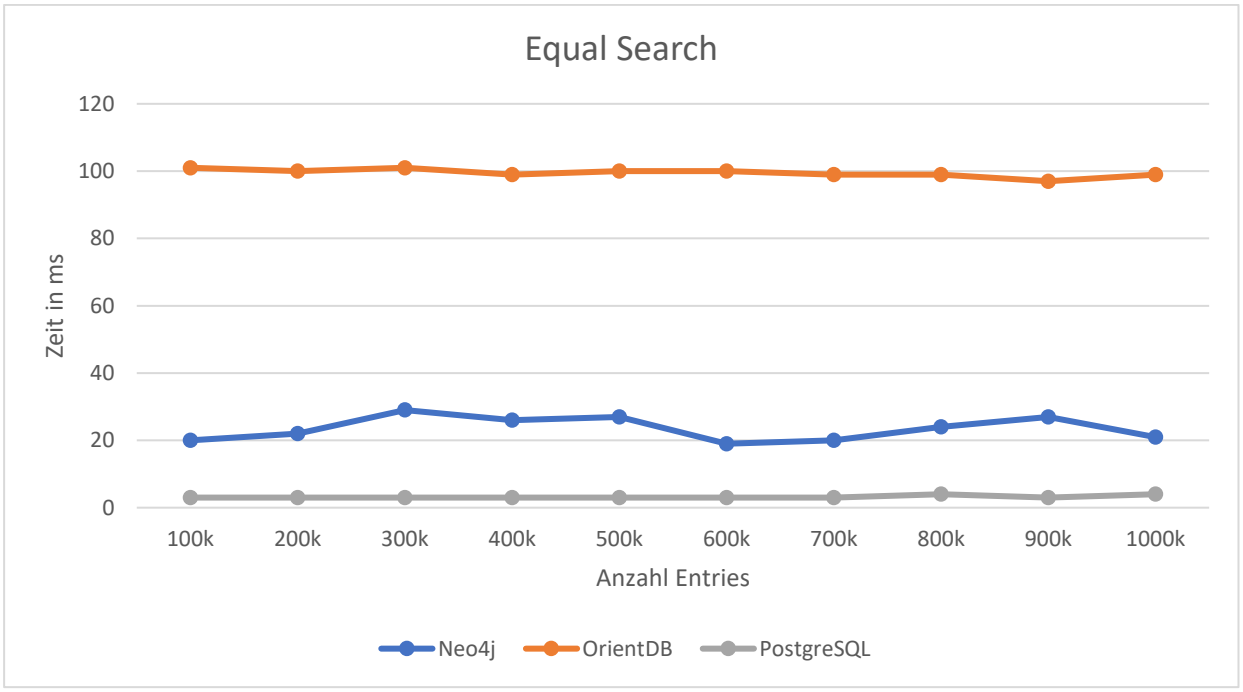


Abbildung 4-3 Equal Search Vergleich

OrientDB wiederum liefert stabile Zugriffszeiten von ± 100 ms und Neo4j pendelt ein wenig um die 20 ms Marke herum. In Abbildung 4-4 Neo4j Equal Search erkennt man, dass Neo4j Mühe hat, eine stabile Antwortzeit zu liefern. Für die Antwort braucht Neo4j zwischen 19 ms bis 29 ms. Dies entspricht einer Abweichung mit dem Faktor 1.5 oder 0.65.

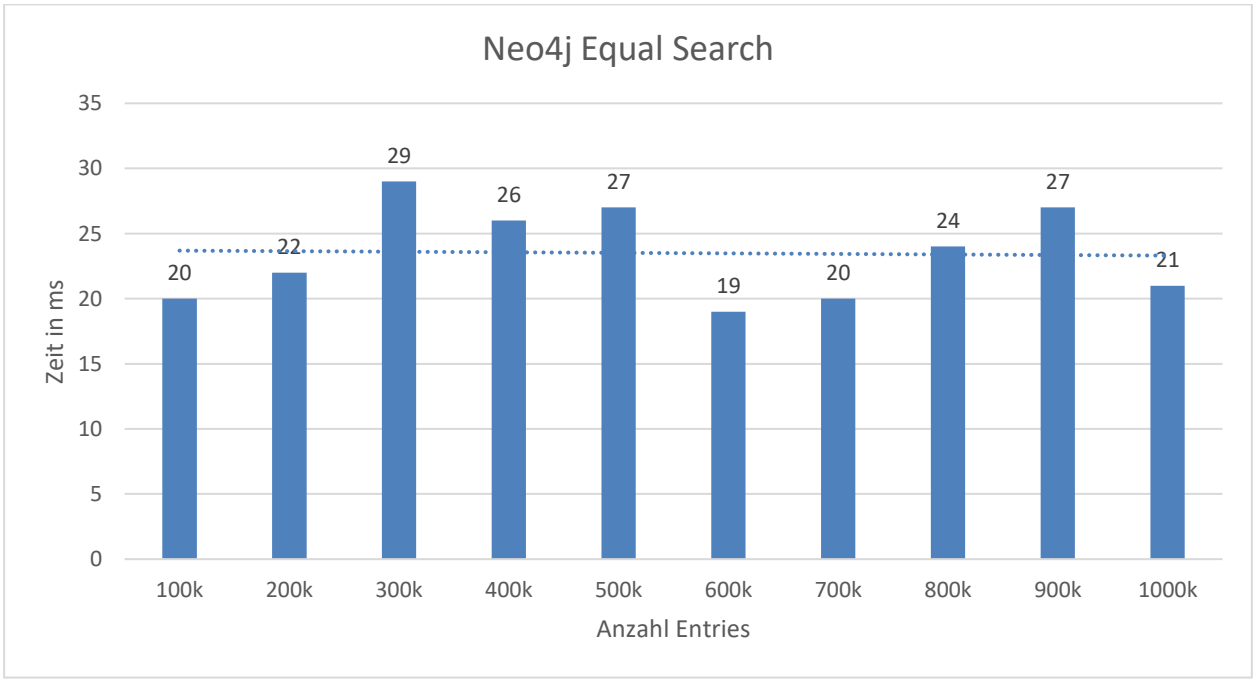


Abbildung 4-4 Neo4j Equal Search

## 4.3 Range Search

Mittels der Range Search Query werden Network-Flows abgefragt, welche eine bestimmte Grösse überschreiten. Bei dieser Query werden alle Flows gesucht, welche mehr als 500 Bytes an Daten übertragen haben. Die Zahl 500 ist des Weiteren nicht von grosser Relevanz. Sie wurde nur so gewählt, da dadurch kleine Flows ausgeschlossen sind, welche nur aus zwei bis drei Pings bestehen. Aus diesem Grund können wir bei den zurückgelieferten Flows davon ausgehen, dass sie effektive Nutzdaten übertragen haben.

### 4.3.1 Query

Über das von dieser Query verwendete Propertyfeld «bytes» wurde kein Index angelegt.

#### 4.3.1.1 Neo4j

```
1 MATCH (flow:Network_Flow)
2 WHERE flow.bytes >= 500
3 RETURN flow
```

#### 4.3.1.2 OrientDB

```
1  MATCH {class: Network_Flow, as: network_flow,
2  where:(bytes >= 500)}
3  RETURN network_flow
```

#### 4.3.1.3 PostgreSQL

```
1  SELECT *
2  FROM network_flow
3  WHERE bytes >= 500
```

### 4.3.2 Analyse der Ergebnisse

Wie anhand des Diagrammes Abbildung 4-5 Range Search schön gesehen werden kann, verlaufen die Query-Ausführungszeiten linear mit der steigenden Anzahl Entries.

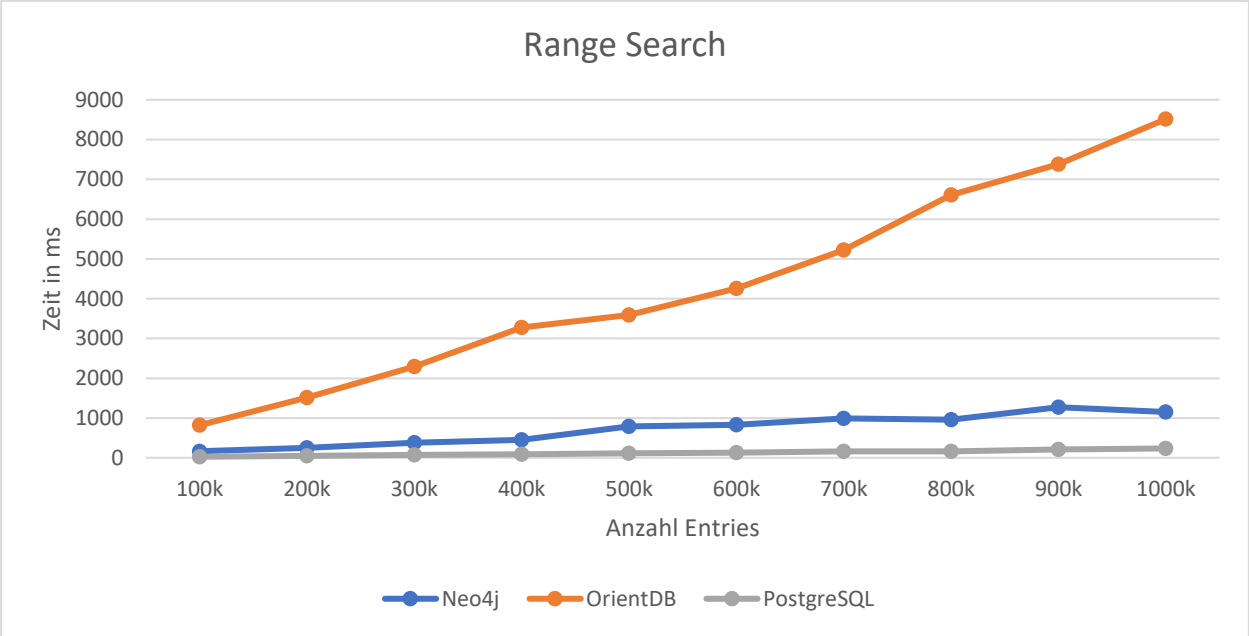


Abbildung 4-5 Range Search

Auch bei dieser Query spielt OrientDB in einer anderen Liga als dies Neo4j und PostgreSQL tun. PostgreSQL ist Neo4j um ca. Faktor 5.8 überlegen (Abbildung 4-6 Range Search - Neo4j vs. PostgreSQL) während OrientDB nochmals ungefähr um Faktor 6 schlechter ist als Neo4j. Zwischen dem Sieger PostgreSQL und dem Verlierer OrientDB besteht somit ein Unterschied von ca. Faktor 34.

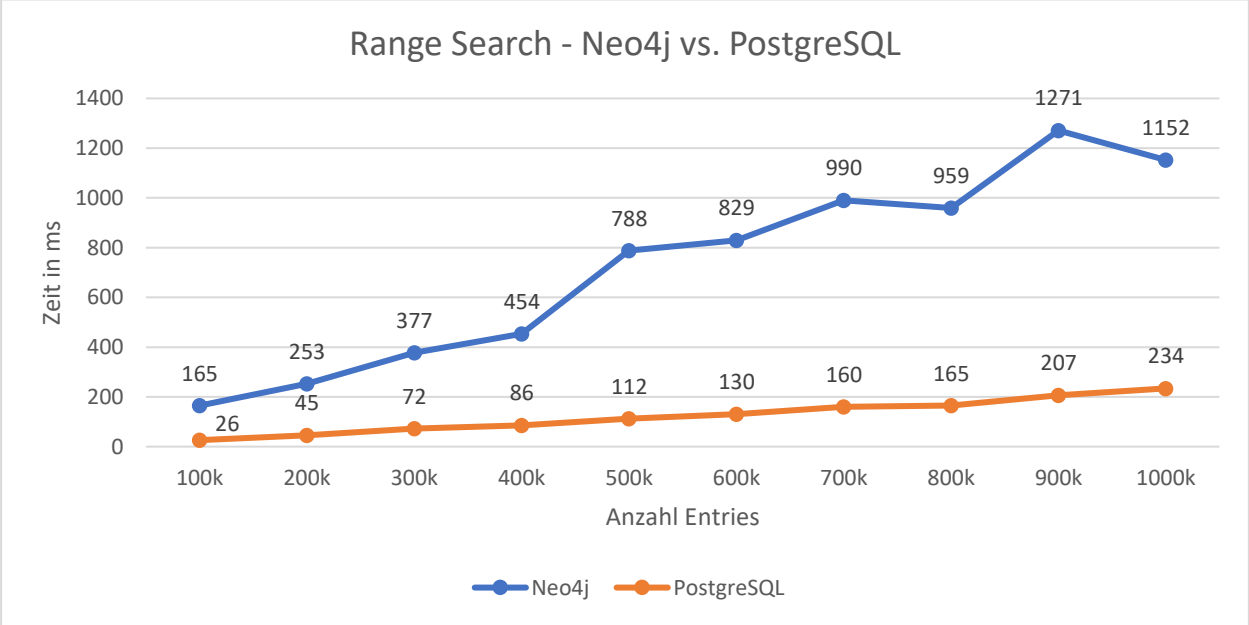


Abbildung 4-6 Range Search - Neo4j vs. PostgreSQL

## 4.4 Big Traffic Range Search

Bei der Bit Traffic Range Search ging es darum, Nodes zu finden, welche bereits mindestens 5000 Bytes zusammengezählt über beliebig viele Flows gebraucht haben. Es zählt dabei sowohl ankommender als auch ausgehender Traffic.

### 4.4.1 Query

Über das von dieser Query verwendete Property «bytes» wurde kein Index angelegt. Für das Property «ip_address» existierte ein Unique-Index.

#### 4.4.1.1 Neo4j

```
MATCH (endpoint:EndPoint_Device) - [] - (flow:Network_Flow)
WITH endpoint.ip_address as ip, sum(flow.bytes) as totalBytes
WHERE totalBytes > 5000
return ip,totalBytes
```

#### 4.4.1.2 OrientDB

```
SELECT FROM (
  SELECT SUM(out('RECEIVE','SENDS').bytes) as totalBytes, ip_address
  FROM EndPoint_Device
  GROUP BY ip_address
)
WHERE totalBytes > 5000
ORDER BY totalBytes
```

#### 4.4.1.3 PostgreSQL

```
SELECT traffic.ip, SUM(bytes) as totalBytes FROM
(
      SELECT endpoint_device.ip_address as ip,bytes
      FROM network_flow
      JOIN endpoint_device ON endpoint_device.id=network_flow.dst_id
      UNION ALL
      SELECT endpoint_device.ip_address as ip,bytes
      FROM network_flow
      JOIN endpoint_device ON endpoint_device.id=network_flow.src_id
  ) traffic
GROUP BY traffic.ip
HAVING SUM(bytes) > 5000
ORDER BY totalBytes
```

### 4.4.2 Analyse der Ergebnisse

Anhand der Abbildung 4-7 Big Traffic Range Search kann erkannt werden, dass bei der Big Traffic Range Search Query pro DBMS ein lineares Verhalten zum Vorschein kam. Erneut ist OrientDB weit unterlegen, während sich Neo4j und PostgreSQL energisch konkurrieren.

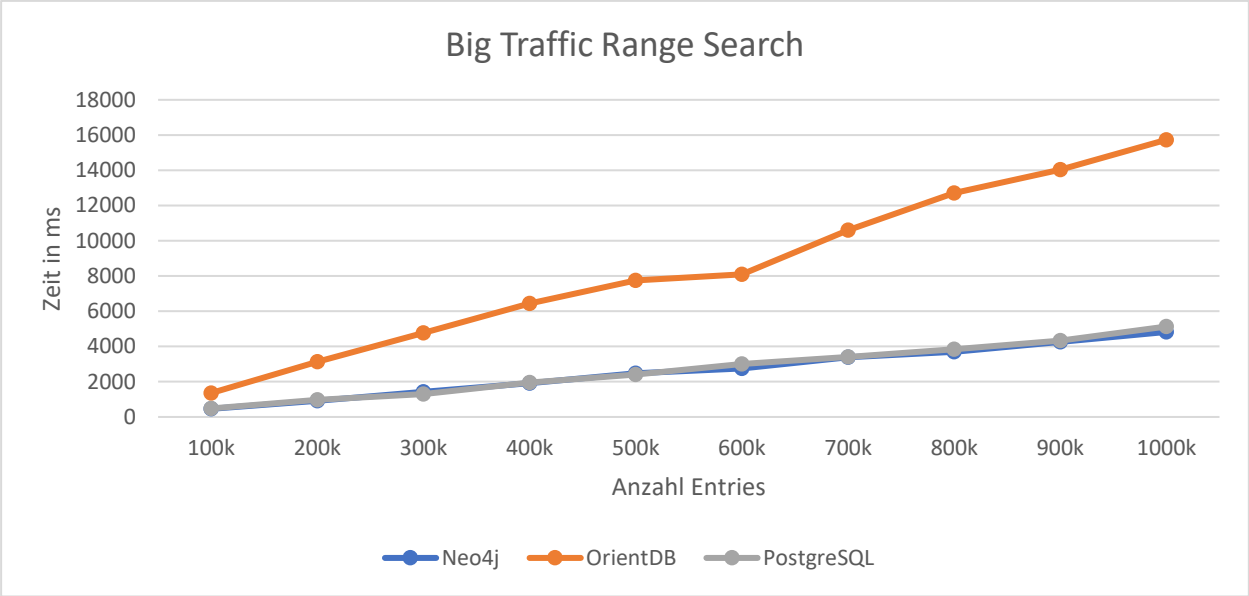


Abbildung 4-7 Big Traffic Range Search

OrientDB ist im Vergleich zu Neo4j und PostgreSQL rund 3.2-mal langsamer.

In einer genaueren Betrachtung der Abbildung 4-8 Big Traffic Range Search - Neo4j vs. PostgreSQL kann sehr schön erkannt werden, dass Neo4j und PostgreSQL ziemlich gleichauf sind. Bei dieser Query scheint es, zumindest für die getestete Datenmenge, keine Rolle zu spielen, ob man nun Neo4j oder PostgreSQL als DBMS eingesetzt würde.

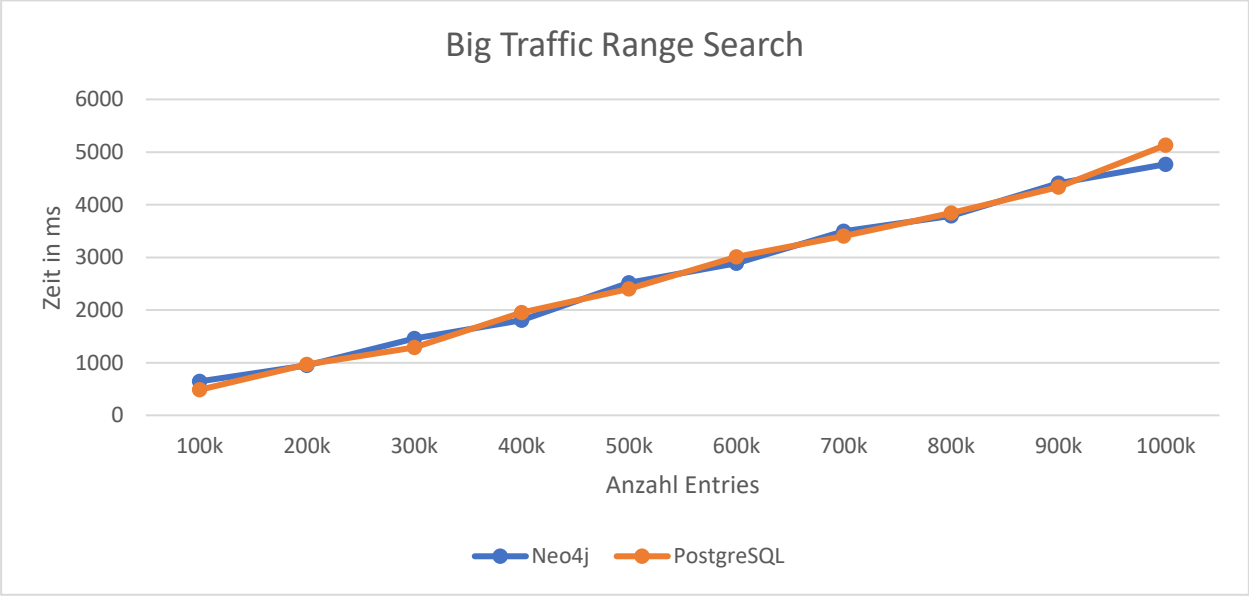


Abbildung 4-8 Big Traffic Range Search - Neo4j vs. PostgreSQL

Da sich bei dieser Query kein wirkliches Sieger-DBMS zeigte, wurde die Testdatenmenge auf 2 Millionen Testdatensätze erhöht. Wie in der Abbildung 4-9 Big Traffic Range Search - 2000k Entries (nicht lineare Skala) gesehen werden kann, zeichnet sich nun mit gesteigerter Datenmenge

doch ein noch Sieger ab. Neo4j scheint mit 2 Millionen Testdatensätzen besser zu skalieren als PostgreSQL.

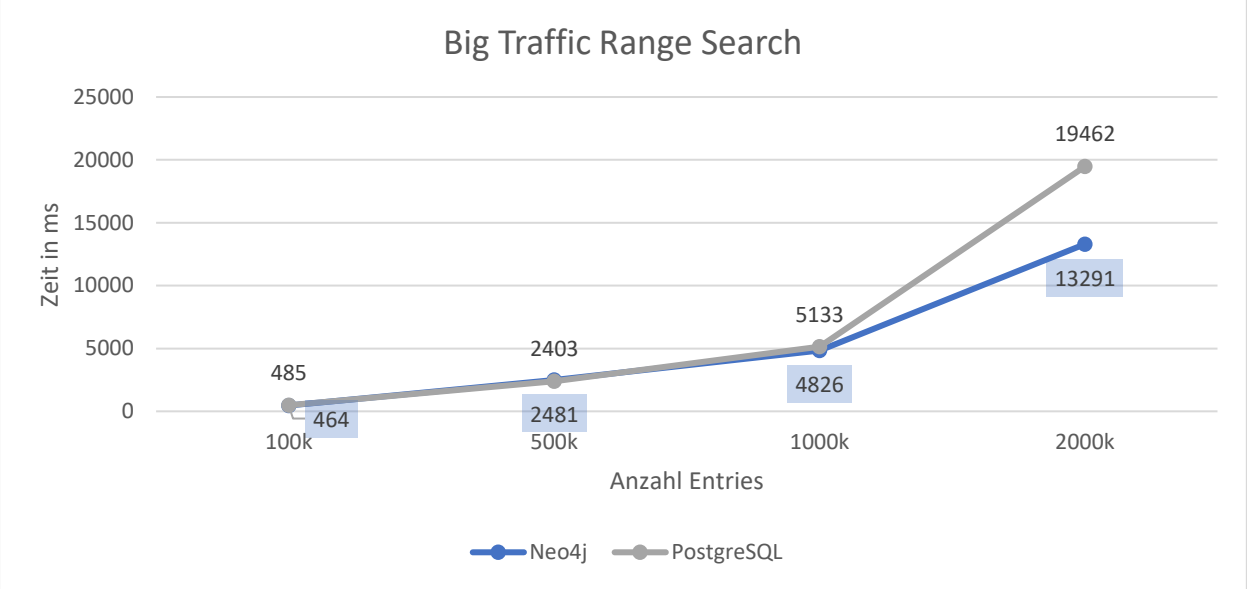


Abbildung 4-9 Big Traffic Range Search - 2000k Entries (nicht lineare Skala)

## 4.5 Pattern Search

Bei der Pattern Search Query sollen alle Endpoints Devices aus einem bestimmten Netzwerksegment (z.B. 152.96.122.0/24) gefunden werden. Es wurde mit Regex gearbeitet.

Der Regex wurde aufgrund der Verständlichkeit so kurz wie möglich gehalten (Regex: 152\.96\.122\..*). Theoretisch sind für den oben erwähnten Regex alle Werte gültig, welche mit “152.96.122.” starten und beliebig enden. Da die Testdaten allerdings zwingend eine gültige IP beinhalten, wurde dies vernachlässigt. Würde ein Packet an eine ungültige IP geschickt (Format falsch), würde dieses Packet bereits schon im Netzwerkstack des Absenders oder spätestens beim NetFlow Exporter verworfen. Somit kann es nicht sein, dass überhaupt ein Flow für das Packet erstellt würde.

### 4.5.1 Query

Für diese Query wurde ein Unique Index über das IP-Adressenfeld angelegt.

#### 4.5.1.1 Neo4j

```
1 MATCH (endpoint:EndPoint_Device)
2 WHERE endpoint.ip_address =~ {regex}
3 RETURN endpoint.ip_address
```

#### 4.5.1.2 OrientDB

```
1  SELECT ip_address
2  FROM endpoint_device
3  WHERE ip_address MATCHES :regex
```

#### 4.5.1.3 PostgreSQL

```
1  SELECT ip_address
2  FROM endpoint_device
3  WHERE ip_address ~ ?
```

### 4.5.2 Analyse der Ergebnisse

Allgemein kann anhand der Abbildung 4-10 Pattern Search gesagt werden, dass sich die Antwortzeiten linear verhalten.

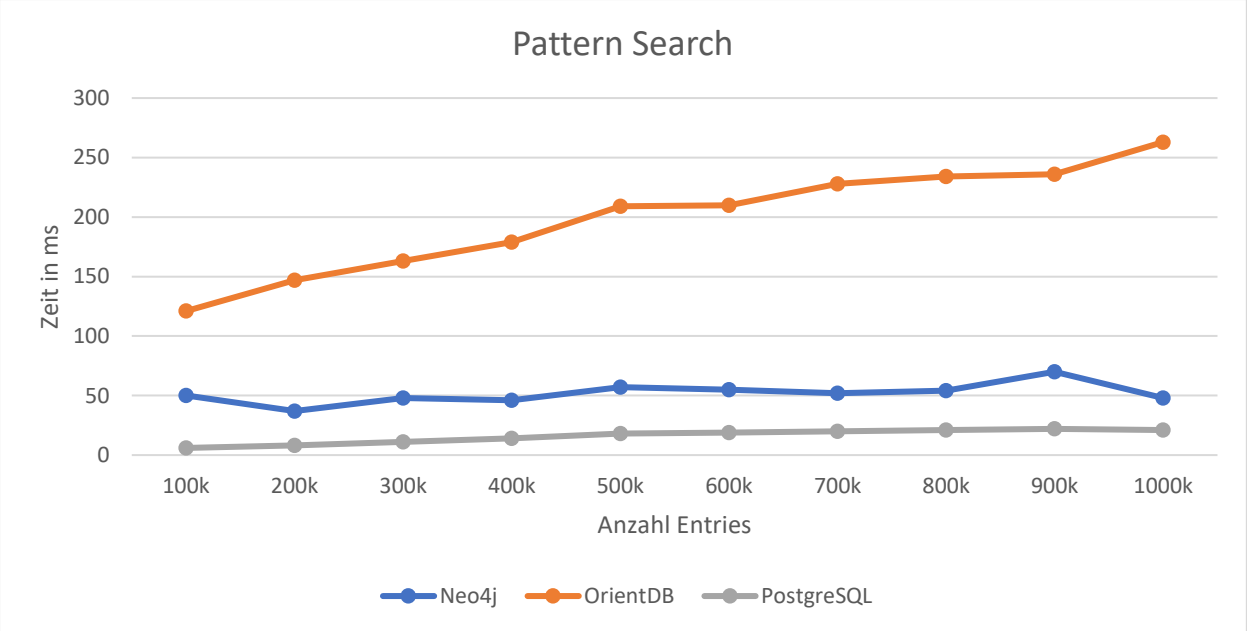


Abbildung 4-10 Pattern Search

Bei Neo4j und insbesondere auch bei PostgreSQL kann erkannt werden, dass sie mit steigender Datenmenge für das Resultat nur marginal länger brauchen. Dies kann hauptsächlich mit dem Inhalt der Benchmark-Testdaten begründet werden. Wird von einem Entry gesprochen, entspricht dies immer 1 NetFlow. Das heisst mit anderen Worten, dass in einem Entry immer zwei Node IP-Adressen enthalten sind. Da sich nun aber in einem LAN nur eine begrenzte Anzahl Nodes befinden kann und das Surfverhalten der Benutzer zum grössten Teil immer in etwa gleich ist, stagniert die Anzahl neuer Node IP-Adressen im Vergleich zur Anzahl neuer Flows relativ schnell.

OrientDB ist erneut in der Spitze - im negativen Sinne. Auch bei dieser Query konnte PostgreSQL den Sieg für sich in Anspruch nehmen. PostgreSQL siegt gegen Neo4j mit einem Vorsprung von

rund Faktor 3.5 (Abbildung 4-11 Pattern Search - Neo4j vs. PostgreSQL). Zwischen PostgreSQL und OrientDB liegt gar ein Faktor von ca. 12.

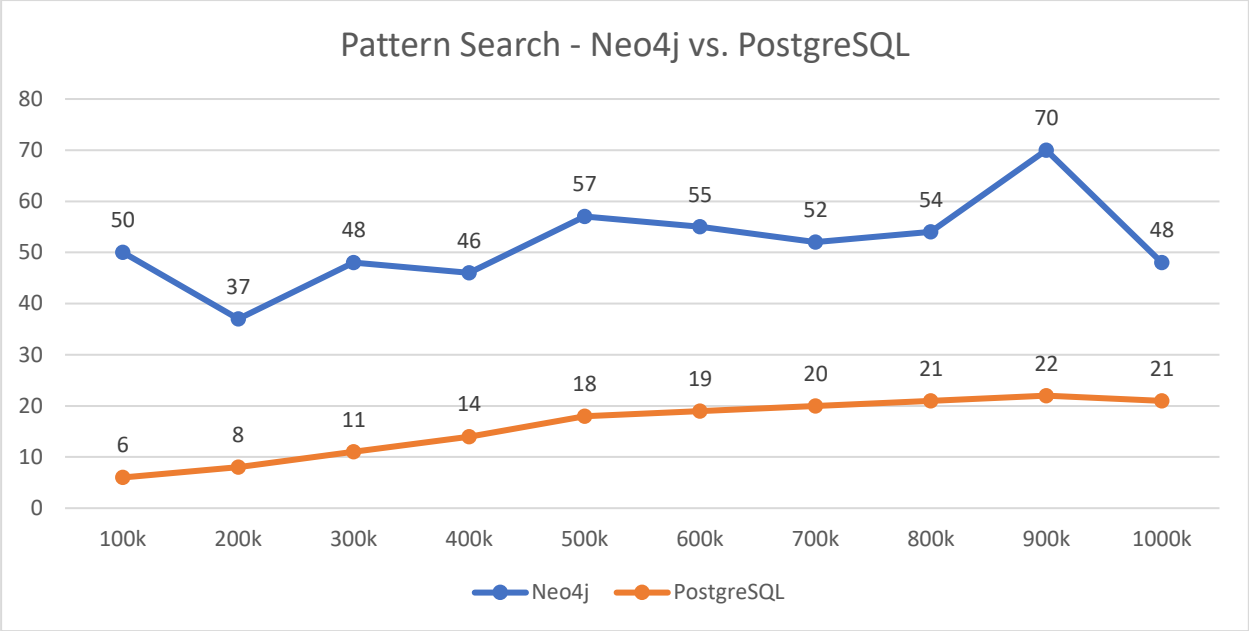


Abbildung 4-11 Pattern Search - Neo4j vs. PostgreSQL

## 4.6 Neighbours Search

Beim Neighbours Search Query geht es darum, alle Nachbar-Nodes eines spezifischen Nodes zu finden. Der eine spezifische Node wird anhand seiner IP-Adresse identifiziert. Bei den Nachbar-Nodes handelt es sich aus Schemasicht um NetFlows, da diese als Nodes und nicht als Edges modelliert sind.

Die Challenge bei dieser Query besteht darin, dass es in beide Richtungen Beziehungen geben kann (Sending und Receiving). Dies führt bei der PostgreSQL Query dazu, dass mit einem Union und zwei einzelnen Queries gearbeitet werden muss.

### 4.6.1 Query

#### 4.6.1.1 Neo4j

```
1 MATCH (endpoint:EndPoint_Device {ip_address:{ip_address}})
2 - [relation:RECEIVE|:SENDS] ->
3 (flow:Network_Flow)
4 RETURN endpoint.ip_address, flow.first_switched
```

#### 4.6.1.2 OrientDB

```
MATCH {class: EndPoint_Device, as: endpoint,
where:(ip_address = :ip_address)}
.both('RECEIVE','SENDS') {as: flow}
RETURN endpoint.ip_address, flow.first_switched
```

#### 4.6.1.3 PostgreSQL

```
SELECT src.ip_address, outgoingFlow.first_switched
FROM endpoint_device AS src
JOIN network_flow AS outgoingFlow
ON src.id = outgoingFlow.src_id
WHERE src.ip_address=?
UNION ALL
SELECT dst.ip_address, incomingFlow.first_switched
FROM endpoint_device AS dst
JOIN network_flow AS incomingFlow
ON dst.id = incomingFlow.dst_id
WHERE dst.ip_address=?;
```

### 4.6.2 Analyse der Ergebnisse

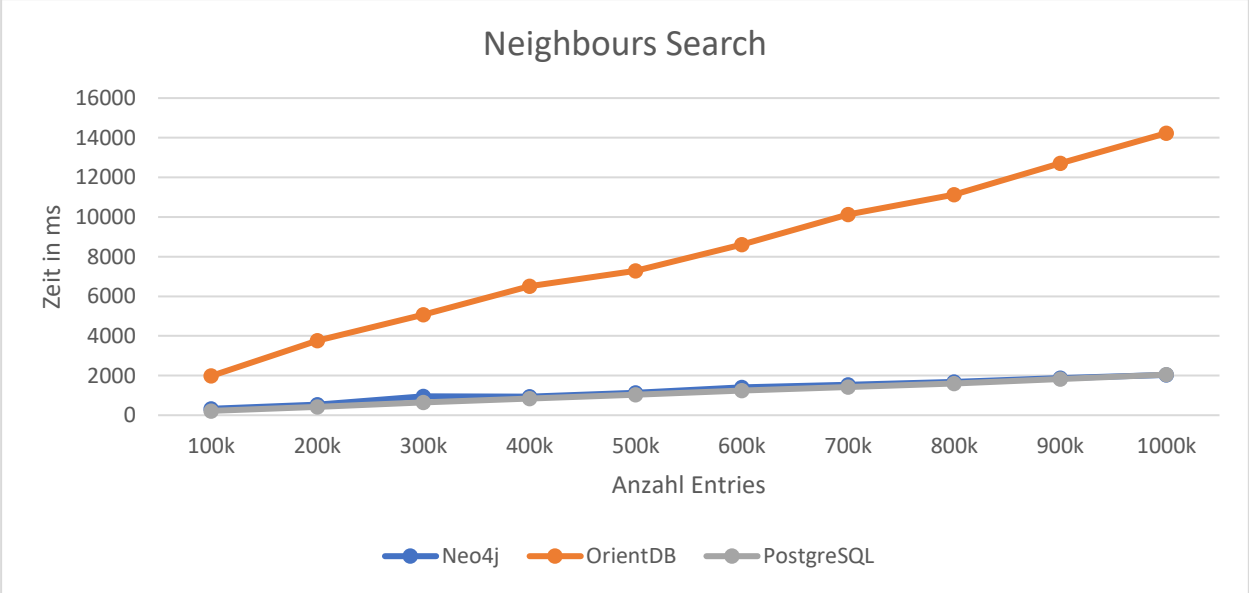


Abbildung 4-12 Neighbours Search

Bei dieser Query ist markant, dass OrientDB erneut negativ auffällt. Im Gegensatz dazu liegen Neo4j und PostgreSQL relativ nahe beieinander und PostgreSQL liegt im Durchschnitt um Faktor 1.4 vorne.

Wird das Rennen um den Sieg genauer betrachtet, zeigt sich eine strikt linearwirkende Steigerung seitens PostgreSQL während Neo4j kleinere Schwankungen aufweist.

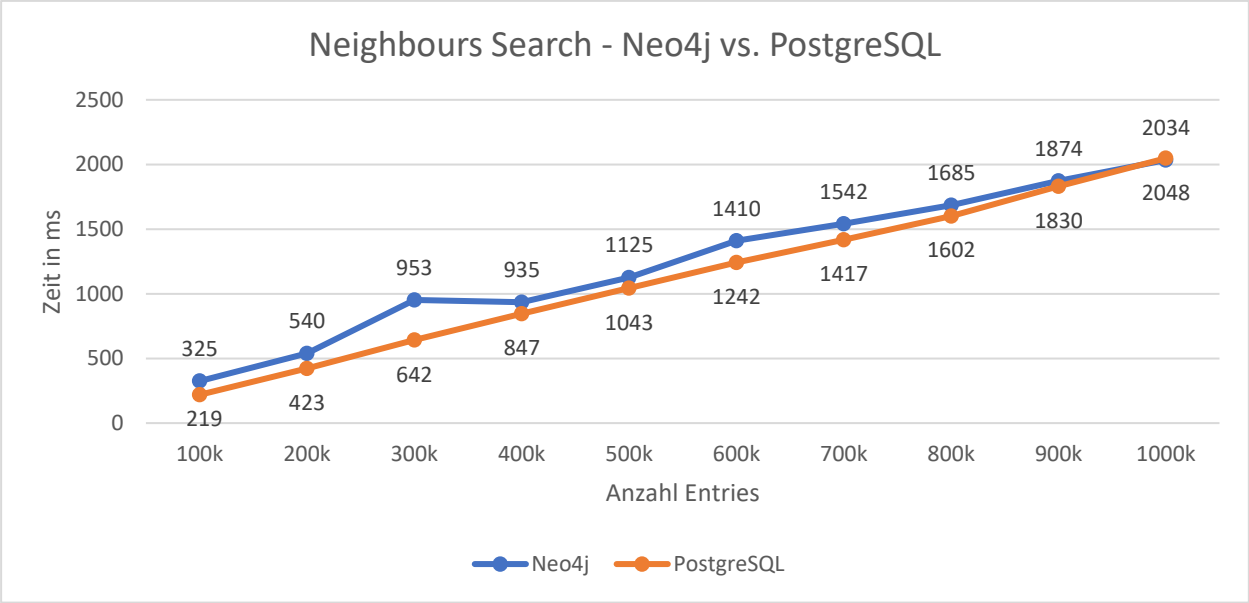


Abbildung 4-13 Neighbours Search - Neo4j vs. PostgreSQL

## 4.7 Neighbours of Neighbours

Bei dieser Query müssen Nachbarn von Nachbarn gefunden werden. In diesem konkreten Fall bedeutet dies, das Auffinden aller Network-Flows zu einem bestimmten Endpoint-Device und alle direkt damit verknüpften Endpoint-Devices.

```
(EndPoint) - [relation] -> (Flow) <- [relation] - (EndPoint)
```

Abbildung 4-14 Neighbours of Neighbours Schema

Die runden Klammern stehen für Nodes und die Eckigen für Edges.

### 4.7.1 Query

Auch für diese Query wurde ein Unique Index über das IP-Adressenfeld angelegt.

#### 4.7.1.1 Neo4j

```
1 MATCH (endpoint:EndPoint_Device {ip_address:{ip_address}})
2 - [relation:RECEIVE|:SENDS] -> (flow:Network_Flow)
3 <- [relation2:RECEIVE|:SENDS] - (partner:EndPoint_Device)
4 RETURN DISTINCT endpoint.ip_address, partner.ip_address
5 ORDER BY partner.ip_address
```

#### 4.7.1.2 OrientDB

```
SELECT DISTINCT(ip_address) as partner_ip
FROM (TRAVERSE BOTH('SENDS','RECEIVE')
FROM (
  SELECT
  FROM EndPoint_Device
  WHERE ip_address = :ip_address) MAXDEPTH 2
)
WHERE ip_address <> :ip_address AND $depth >= 2
ORDER by partner_ip
```

#### 4.7.1.3 PostgreSQL

```
SELECT DISTINCT * FROM (
    SELECT endpoint.ip_address, partner.ip_address as partner_ip
    FROM endpoint_device AS endpoint
    JOIN network_flow AS netflow
    ON endpoint.id =  netflow.src_id
    JOIN endpoint_device AS partner
    ON partner.id = netflow.dst_id
    WHERE endpoint.ip_address=?
  UNION ALL
    SELECT endpoint.ip_address, partner.ip_address as partner_ip
    FROM endpoint_device AS endpoint
    JOIN network_flow AS netflow
    ON endpoint.id =  netflow.dst_id
    JOIN endpoint_device AS partner
    ON partner.id = netflow.src_id
    WHERE endpoint.ip_address=?
) AS traffic
ORDER BY traffic.partner_ip;
```

### 4.7.2 Analyse der Ergebnisse

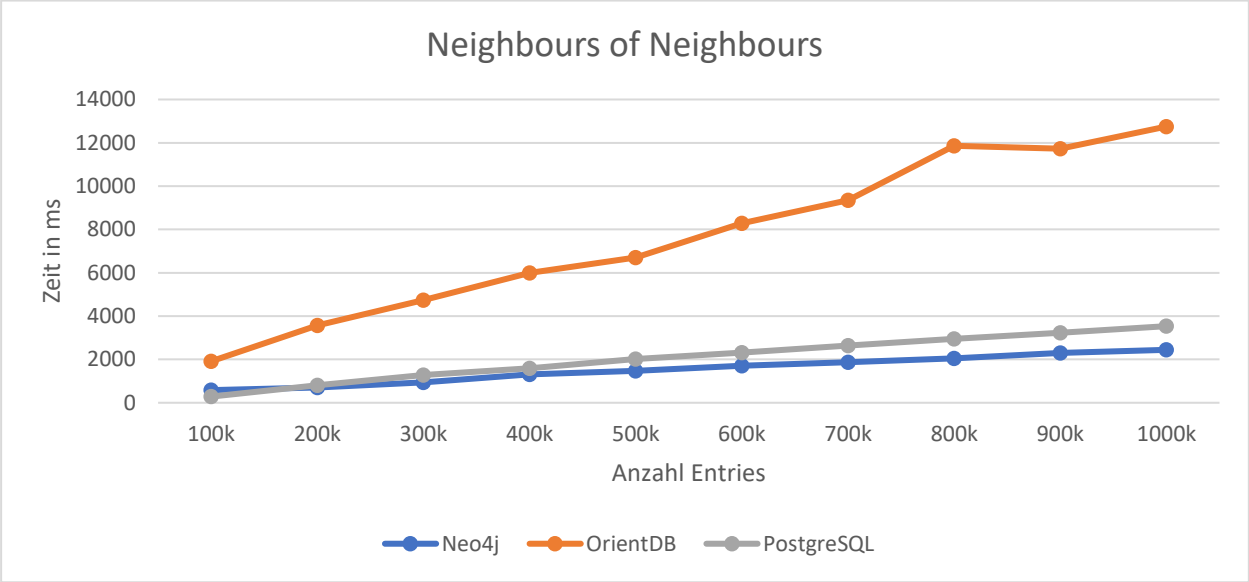


Abbildung 4-15 Neighbours of Neighbours

Wie anhand von Abbildung 4-15 Neighbours of Neighbours gesehen werden kann, schwingt OrientDB erneut massiv oben auf, während Neo4j und PostgreSQL das Rennen für sich ausmachen können.

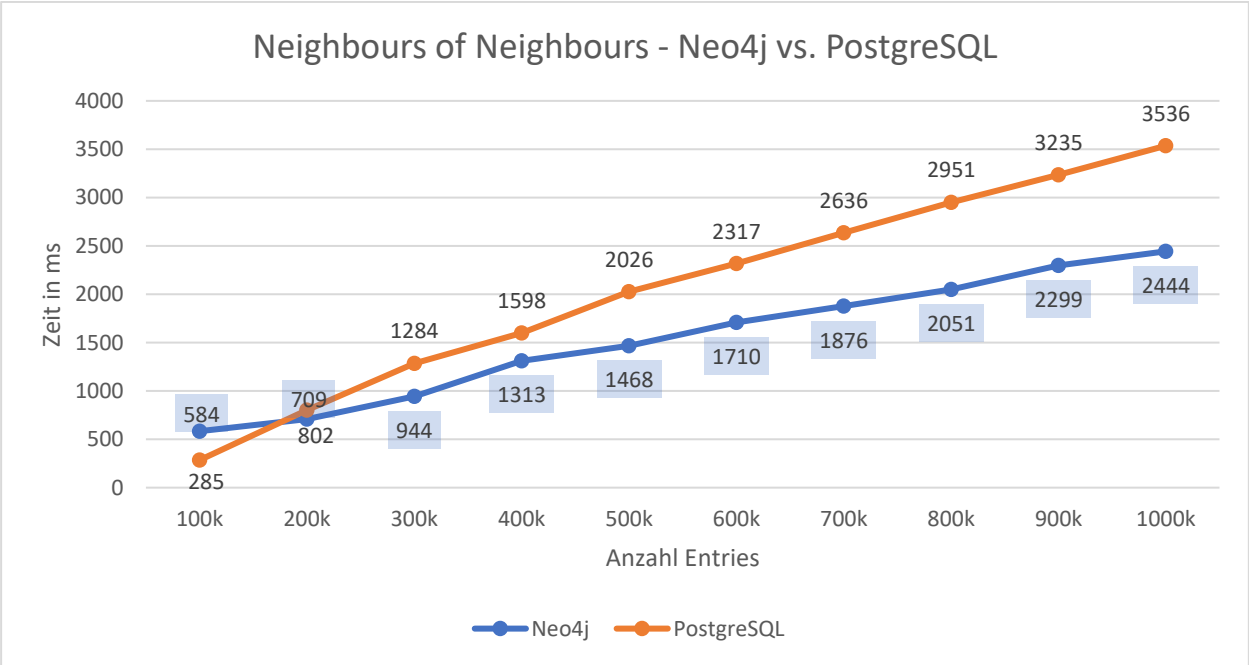


Abbildung 4-16 Neighbours of Neighbours - Neo4j vs. PostgreSQL

Bei einer genaueren Betrachtung in Abbildung 4-16 Neighbours of Neighbours - Neo4j vs. PostgreSQL kann erkannt werden, dass zu Beginn PostgreSQL bis rund 180k Entries die Nase vorn hat.

Anschliessend kann Neo4j mit der immer grösser werdenden Datenmenge besser umgehen als PostgreSQL und übernimmt somit die Spitze. Ab diesem Zeitpunkt fallen diese beiden DBMS immer weiter auseinander und Neo4j kann seinen Vorsprung kontinuierlich ausbauen. In Zahlen ausgedrückt, ist Neo4j über diese 10 Datenmengenstufen hinweggesehen um rund Faktor 1.26 schneller als PostgreSQL.

Wie anhand dieser Query schön gesehen werden kann, kommt nun langsam die Art von Query, auf welche Graphdatenbanken im Speziellen abzielen.

## 4.8 Neighbours Advanced

Die anspruchsvollste Query des Benchmarks sucht nach Endpoints Devices, welche auch mit den Endpoint-Devices kommuniziert haben, mit welchen ein spezifisches Endpoint-Device kommuniziert hat. Schematisch dargestellt sucht diese Query nachfolgender Konstellation:

```
(EndPoint) -> (Flow) <- (EndPoint) -> (Flow) <- (EndPoint)
```

Abbildung 4-17 Neighbours Advanced Schema

Einfachheitshalber wurden bei dieser Darstellung die Edges weggelassen und lediglich als Pfeil ausgedrückt.

### 4.8.1 Query

#### 4.8.1.1 Neo4j

```
MATCH (endpoint:EndPoint_Device {ip_address:{ip_address}})
- [:RECEIVE|:SENDS*2] - (middle:EndPoint_Device)
WITH DISTINCT endpoint,middle
MATCH (middle) - [:RECEIVE|:SENDS*2] - (partner:EndPoint_Device)
WHERE partner.ip_address <> endpoint.ip_address
RETURN DISTINCT endpoint.ip_address, middle.ip_address, partner.ip_address
ORDER BY middle.ip_address
```

#### 4.8.1.2 OrientDB

```
SELECT DISTINCT(ip_address) as partner_ip
FROM (
  TRAVERSE BOTH('SENDS','RECEIVE')
  FROM (
    SELECT
    FROM EndPoint_Device
    WHERE ip_address = :ip_address
  ) MAXDEPTH 4
)
WHERE ip_address <> :ip_address AND $depth >= 4
ORDER by partner_ip
```

#### 4.8.1.3 PostgreSQL

```
SELECT DISTINCT
  traffic2.endpoint_ip,
  traffic2.middle_ip,
  partner.ip_address
FROM (
  SELECT DISTINCT
    traffic.endpoint_ip as endpoint_ip,
    netflow2.src_id as netflowSrc,
    netflow2.dst_id as netflowDst,
    traffic.middle_ip as middle_ip,
    traffic.endpoint_id as endpoint_id,
    traffic.middle_id as middle_id
  FROM (
    SELECT DISTINCT
      endpoint.ip_address as endpoint_ip,
      partner.ip_address as middle_ip,
      endpoint.id as endpoint_id,
      partner.id as middle_id
    FROM endpoint_device AS endpoint
    JOIN network_flow AS netflow
    ON (endpoint.id = netflow.src_id
      OR endpoint.id = netflow.dst_id)
    JOIN endpoint_device AS partner
    ON ((partner.id = netflow.dst_id
      OR partner.id = netflow.src_id)
      AND endpoint.id <> partner.id)
    WHERE endpoint.ip_address=?
  ) AS traffic
  JOIN network_flow AS netflow2
  ON(traffic.middle_id = netflow2.src_id
    OR traffic.middle_id = netflow2.dst_id)
) AS traffic2
JOIN endpoint_device AS partner
ON ((partner.id = traffic2.netflowDst
  OR partner.id = traffic2.netflowSrc)
  AND partner.id <> traffic2.middle_id
  AND traffic2.endpoint_id <> partner.id)
ORDER BY traffic2.middle_ip
```

### 4.8.2 Analyse der Ergebnisse

Bei dieser Query hat sich deutlich gezeigt, wo genau die Stärken von Graphdatenbanksystemen liegen. Oben weg verläuft dieses Mal PostgreSQL. Neo4j und OrientDB scheinen das Rennen für unter sich zu entscheiden.

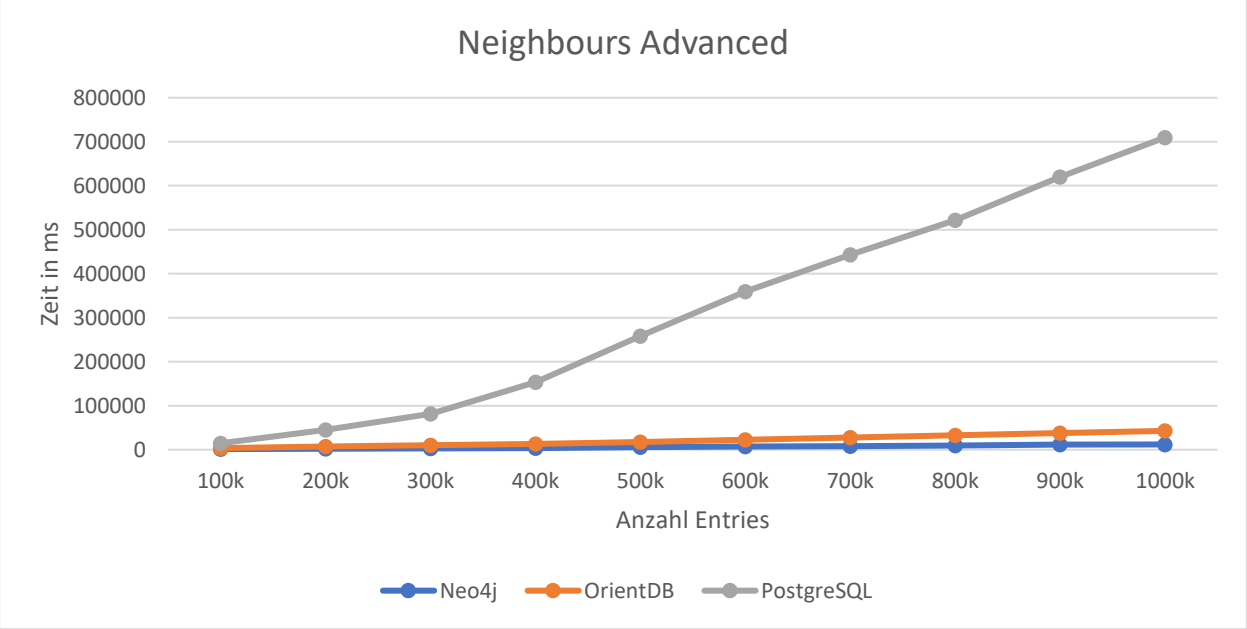


Abbildung 4-18 Neighbours Advanced

In genauer Betrachtung kann in der Abbildung 4-19 Neighbours Advanced - Neo4j vs. OrientDB erkannt werden, dass Neo4j gegenüber OrientDB die Nase doch noch deutlich vorne hat. Neo4j ist ca. um Faktor 3.2 schneller als OrientDB.

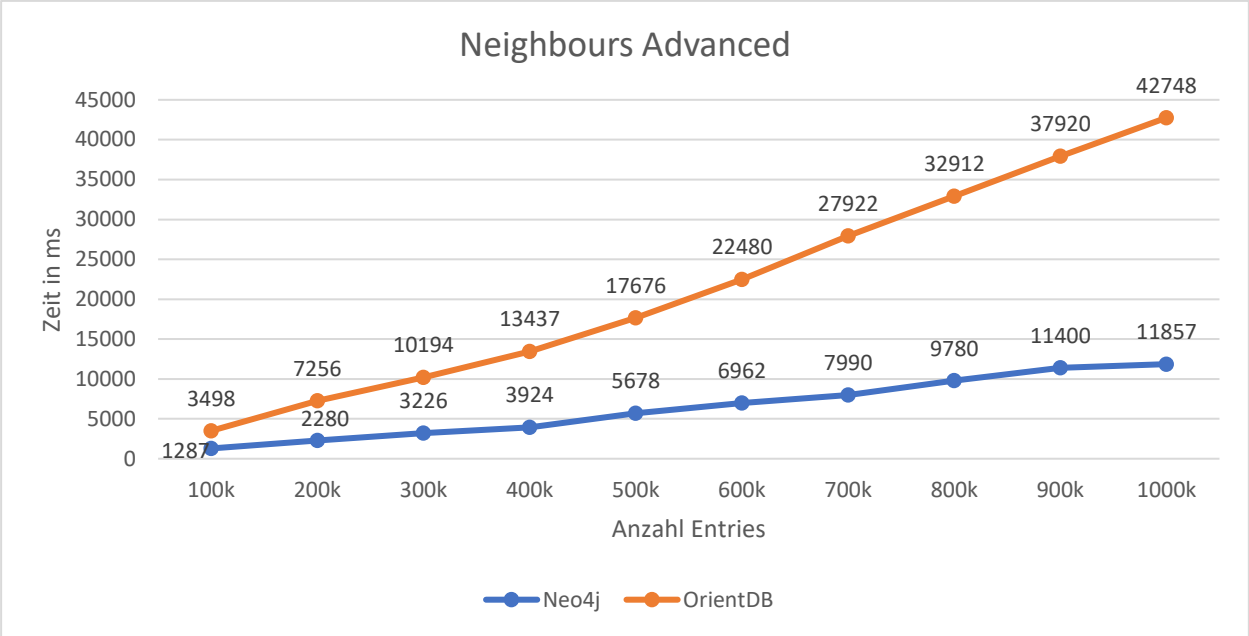


Abbildung 4-19 Neighbours Advanced - Neo4j vs. OrientDB

# 5. Aufgetretene Probleme

Während des Benchmarks sind wir auf zwei erwähnenswerte Probleme gestossen.

## 5.1 OrientDB: Crash bei grösseren Datenmenge

OrientDB verursachte das Problem, dass es im Benchmark-Run mit 2 Millionen Entries ständig zu einem Absturz kam. Bis und mit 1 Million Entries war noch kein Problem bemerkbar und der Service funktionierte wie gewünscht. Bei 2 Millionen Entries kam es dann allerdings bereits bei der Massive Insertion zu einer Heap Space Exception.

```
1   Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
```

Abbildung 5-1 OrientDB Heap Space Exception

Zu vermerken gilt, dass während diese Exception auftragt, OrientDB in der Standardkonfiguration lief.

Um dem Problem nun aber auf die Spur zu kommen, versuchten wir die Performance Tuning Tipps[49] vom OrientDB Hersteller anzuwenden. Was wir alles angewandt hatten:

- Die Datenbank wurde nicht via «remote»- sondern via «plocal»-Methode angesprochen.
- Auf den Edges wurden keine Properties gespeichert.
- Wir verwendeten die «Massive Insertion»-Methode, welche von OrientDB speziell für grosse Inserts zur Verfügung gestellt wird.
- OrientGraph wurde auf OrientGraphNoTx umgestellt, damit automatische Transaktionen unterbunden werden. Stattdessen schickten wir jeweils nach 50'000 Entries manuell eine Transaktion ab.
- Wir legten einen Unique-Index für das IP-Adressen-Property an.
- Wie des Weiteren auch vorgeschlagen wurde, verwendeten wir ein eigenes Schema statt nur die Standard Entities (V)ertex und (E)dge.

Leider halfen all diese Schritte nicht, das ursprüngliche «Heap Space Exception»-Problem zu beheben. Aus zeitlichen Gründen beschlossen wir, dass wir dem Problem nicht weiter nachgehen und somit auf die OrientDB 2000k Benchmark-Resultate zu verzichten.

## 5.2 Neo4j Diskspace

Bei Neo4j standen wir vor dem Problem, dass gegen Ende der Neo4j-Queries der Benchmark plötzlich und nicht deterministisch abstürzte. Als Fehler wurde «Java.io.IOException: There is not enough space on the disk» ausgegeben. Wie sich bereits vermuten lässt, lag der Ursprung des

[49] (OrientDB LTD, kein Datum)

Problems an einer vollen Partition. Die Neo4j-Transaction-Logs verbrauchten sämtlichen zur Verfügung stehenden Speicher der /var-Partition und dies führte dazu, dass Neo4j selbst nicht mehr in seine Datenbank unter „/var/lib/neo4j/data/databases/graph.db“ schreiben konnte.

```
root@pin1258008:/opt/benchmark# ls -lath /var/lib/neo4j/data/databases/graph.db/neostore.transaction.*
-rw-r--r-- 1 neo4j nogroup 166M Dec  8 11:10 /var/lib/neo4j/data/databases/graph.db/neostore.transaction.db.5
-rw-r--r-- 1 neo4j nogroup 269M Dec  8 10:55 /var/lib/neo4j/data/databases/graph.db/neostore.transaction.db.4
-rw-r--r-- 1 neo4j nogroup 272M Dec  8 10:39 /var/lib/neo4j/data/databases/graph.db/neostore.transaction.db.3
-rw-r--r-- 1 neo4j nogroup 253M Dec  8 10:10 /var/lib/neo4j/data/databases/graph.db/neostore.transaction.db.2
-rw-r--r-- 1 neo4j nogroup 269M Dec  8 09:49 /var/lib/neo4j/data/databases/graph.db/neostore.transaction.db.1
-rw-r--r-- 1 neo4j nogroup 256M Dec  8 09:31 /var/lib/neo4j/data/databases/graph.db/neostore.transaction.db.0
```

Abbildung 5-2 Neo4j Transaction Logs

Neo4j bietet grundsätzlich Konfigurationsmöglichkeiten[50], um die Transaction-Logs zu steuern. Wir haben uns entschieden, ein Transaction-Logging grundsätzlich aktiviert zu lassen aber aufgrund der vielen Transaktionen innerhalb des Benchmarks, die Logs auf eine Anzahl von 500’000 Transactions zu begrenzen.

```
# Will keep logical logs which contains any of the most recent 500 000 transactions
dbms.tx_log.rotation.retention_policy=500k txs
```

Abbildung 5-3 Neo4j Retention Policy

Wie sich herausstellte, bewirkte diese Konfiguration alleine nichts. Um nicht noch mehr Zeit mit diesem Problem zu verbrauchen, entschieden wir uns schliesslich für eine Neo4j externe Methode als Workaround. In unserem Benchmark-Bash-Script führen wir nach jedem Neo4j-Durchlauf (d.h. pro verschiedene Anzahl Entries) ein Löschbefehl auf dem Neo4j-Directory aus.

```
rm -rf /var/lib/neo4j/data/databases/graph.db
```

Abbildung 5-4 Neo4j Diskspace Problem Workaround

Mit diesem Workaround haben wir für den Benchmark eine Lösung gefunden, welche sicherstellt, dass auch grosse Neo4j-Benchmark-Queries ohne Probleme durchlaufen. Der Workaround mag auf den ersten Blick relativ brachial wirken. Dies relativiert sich jedoch, da der Hersteller von Neo4j selbst diese Methode[51] empfiehlt, falls man eine grosse Datenmenge löschen will.

[50] (Neo Technology, Inc., kein Datum)

[51] (Gordon, 2016)

# 6. Vergleich zu anderen Benchmarks

Um die Tendenz unserer Benchmark-Daten besser beurteilen zu können, haben wir diese mit anderen Benchmark-Resultaten vergleichen, welche im Internet publiziert wurden.

Wir verglichen unsere Resultate mit denen des ArangoDB Benchmarks[52] und des Benchmarks von Socialsensor[53]. Wie sich herausstellte, lassen sich drei unserer Benchmark-Queries mit den Resultaten der anderen Benchmarks vergleichen.

## 6.1 Massive Insertion

Die III 4.1 Massive Insertion Query unseres Benchmarks lässt sich mit der «Massive Insertion Workload»-Query des Benchmarks von Socialsensor vergleichen.

| Dataset | Workload | Titan | OrientDB | Neo4j |
|---|---|---|---|---|
| EN | MIW | 9.36 | 62.77 | **6.77** |
| AM | MIW | 34.00 | 97.00 | **10.61** |
| YT | MIW | 104.27 | 252.15 | **24.69** |
| LJ | MIW | 663.03 | 9416.74 | **349.55** |

Abbildung 6-1 Socialsensor Benchmark Massive Insertion Workload

Bei unserer III 4.1 Massive Insertion Query war Neo4j gegenüber OrientDB teilweise bis zu Faktor 5 schneller. Die Tendenz der Resultate vom Benchmark von Socialsensor geht in die genau gleiche Richtung, wobei sie mit bis zu Faktor 10 Unterschied noch mehr ins Extreme gehen.

## 6.2 Neighbours Search

Unsere Query III 4.6 Neighbours Search lässt sich ansatzweise mit der Workload-Query «Find-Neighbours» des Benchmarks von Socialsensor vergleichen. Beide Queries suchen nach allen direkten Nachbarn. Unsere Query sucht nach allen Nachbarn von einem Node mit einer bestimmten IP-Adresse und in der «FindNeighbours»-Query wird nach allen Nachbarn von allen Nodes gesucht. Diese beiden Queries lassen sich unserer Meinung nach deshalb nur unter Schwierigkeiten vergleichen, da es bei der «FindNeighbours»-Query vermehrt zu Caching kommen könnte während bei unserer Query noch ein Equal Search auf den IP-Adressen-Index gemacht werden muss.

[52] (Weinberger, 2015)

[53] (SocialSensor, 2016)

| Dataset | Workload | Titan | OrientDB | Neo4j |
|---|---|---|---|---|
| EN | QW-FN | 1.87 | **0.56** | 0.95 |
| AM | QW-FN | 6.47 | 3.50 | **1.85** |
| YT | QW-FN | 20.71 | 9.34 | **4.51** |
| LJ | QW-FN | 213.41 | 303.09 | **47.07** |

Abbildung 6-2 Socialsensor Benchmark FindNeighbours[54]

Hier lässt sich sagen, dass Neo4j gegenüber OrientDB im Durchschnitt schneller ist. Zugleich entfernt sich OrientDB mit grösser werdendem Daten-Set immer weiter von Neo4j weg – genau gleich wie bei unserem Benchmark-Resultat.

[54] (SocialSensor, 2016)

## 6.3 Neighbours of Neighbours

Mit unserer III 4.7 Neighbours of Neighbours Query lassen sich die Resultate vom ArangoDB-Benchmark «Neighbors Search Test» vergleichen.

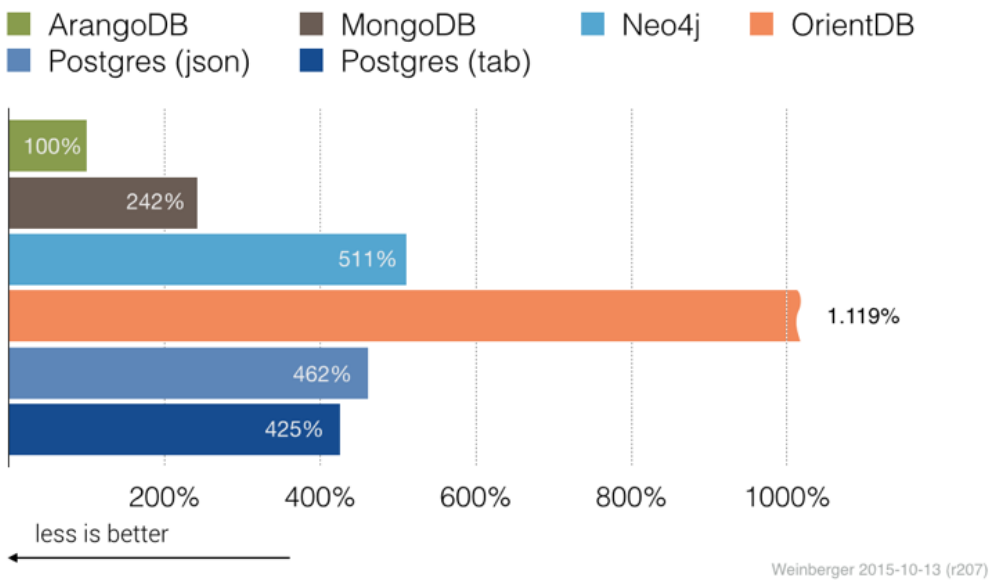


Abbildung 6-3 ArangoDB Benchmark Neighbors Search Test[55]

Wie sich zeigt, liegt auch hier OrientDB massiv oben auf und PostgreSQL und Neo4j liegen relativ nahe beieinander. Beim Benchmark von ArangoDB hat PostgreSQL gegenüber Neo4j die Nase vorn während dies bei unseren Resultaten genau umgekehrt ist. Zur Erinnerung, zu Beginn liegt bei uns PostgreSQL in Führung und ab rund 180k Entries gewinnt Neo4j die Oberhand.

[55] (Weinberger, 2015)

# 7. Schlussfolgerung

Alles in allem wurden wir durch die Benchmark-Resultate doch recht überrascht. Es stellte sich heraus, dass man wirklich genau abwägen sollte, welches DBMS man nun einsetzen will. Während sich PostgreSQL bei nahezu allen Queries behaupten konnte, zeigten die Graphdatenbanksysteme vor allem bei den Neighbours Queries ihre Stärken. Beim Equal Search oder Range Search stach besonders hervor, dass Graphdatenbanksysteme, welche auf Java laufen, einen gewissen Overhead besitzen. Es zeigt sich auch, dass Neo4j gegenüber OrientDB das ausgereiftere Graphdatenbanksystem zu sein scheint. Es kam zu keinem Benchmark-Query-Resultat, in welchem OrientDB schneller war als Neo4j. Im Gegenteil, OrientDB war oft um einen recht beachtlichen Faktor langsamer als Neo4j.

Ein weiterer grosser Pluspunkt von Neo4j ist die DBMS spezifische Abfragesprache Cypher[56]. Zu Beginn war die Cypher Syntax ein wenig gewöhnungsbedürftig zu erlernen – besonders als langjährige SQL Syntax Anwender. Diese Fremdheit legte sich jedoch schnell und das volle Potential der Syntax kam zum Vorschein. Für die Graph-spezifischen Queries ist Cypher Gold wert (man vergleiche die Queries der verschiedenen DBMS der III 4.8 Neighbours AdvancedNeighbours Advanced Query).

Für unsere Benchmark-Queries können wir sagen, dass insbesondere bei Auswertungen in entgegengesetzte Richtungen (hier Traffic Sending und Receiving) und/oder über mehr als 2 Joins hinweg, die Graphdatenbanksysteme diesen Workload schneller abarbeiten können. Dies haben die Queries Neighbours of Neighbours und Neighbours Advanced gezeigt. Braucht man selten Queries, welche solche Bedingungen erfordern, ist man unserer Meinung nach mit einem RDBMS wie PostgreSQL performancemässig besser bedient.

Die Resultate zeigen, dass es sich lohnt auf Polyglot Persistence zu setzen und somit für verschiedene Anforderungen in derselben Applikation das jeweils passende DBMS zu verwenden. Es sollte jedoch nicht bloss auf den Datentyp geschaut werden, sondern viel mehr noch auf die Art der üblichen Queries, welche man auf dem DBMS ausführen möchte.[57]

[56] (Neo Technology, Inc., kein Datum)

[57] (Fowler, 2011)

# IV. Anhang

# 1. Abkürzungen und wichtige Begriffe

| Begriff / Abkürzung | Bedeutung |
|---|---|
| Bolt | Bolt ist ein neues proprietäres Protokoll, welches für high-Performance Zugriffe auf Graphdatenbanken entworfen wurde[58] |
| CI | CI steht für **C**ontinuous **I**ntegration[59]. Unter CI versteht man einen fortlaufenden Prozess, welcher es erlaubt, Softwarekomponenten zusammenzufügen. Häufig wird der CI Prozess auf einem eigenen Server laufen gelassen und wird dazu verwendet, eine Software zu builden und die Unit- oder Integrationstests laufen zu lassen. |
| Cloud | Wenn man in der IT von Cloud spricht, ist damit meist das Cloud-Computing[60] gemeint. Unter Cloud-Computing versteht man den Ansatz, Rechenleistung hochverfügbar und transparent als Service anzubieten. Eine Cloud kann private, public oder hybrid sein. In einer hybriden Cloud kann z.B. zusätzliche Rechenleistung über eine public Cloud bezogen werden, falls die private Cloud nicht mehr genügend Ressourcen frei hat. |
| Cypher | Cypher ist die Graphdatenbank-Abfragesprache von Neo4j[61]. Der Cypher Syntax ist im Gegensatz zum Standard SQL speziell gut geeignet, um Graph-Traversierungen effizient und einfach abfragen zu können. |
| DBMS | Database-Management-System |
| Endpoint-Device | Siehe II 3.1 Domainmodel |
| LogStash | LogStash erlaubt es, Logs zu sammeln, sie zu parsen und zu transformieren[62]. Häufig wird LogStash im Zusammenhang mit Elasticsearch eingesetzt. |
| NetFlow | NetFlow ist eine Funktionalität des Herstellers Cisco, welche es erlaubt verbindungs-spezifische Daten direkt auf dem Netzwerkgerät zu sammeln. Einen solchen einzelnen Datensatz unterteilen wir in verschiedene Entitäten (siehe II 3.1 Domainmodel).[63] |
| Network Flow | Siehe II 3.1 Domainmodel |
| recorded | Siehe II 3.1 Domainmodel |
| Routing-Device | Siehe II 3.1 Domainmodel |

[58] (Kollegger, 2015)

[59] (CI Continuous integration, 2016)

[60] (Cloud computing, 2016)

[61] (Neo Technology, Inc., kein Datum)

[62] (Elasticsearch BV, kein Datum)

[63] (Cisco, kein Datum)

| | |
|---|---|
| RUP | RUP steht für Rational Unified Process und bezeichnet einen iterativen Software Entwicklungsprozess[64]. RUP ist in 4 Phasen aufgeteilt: Inception, Elaboration, Construction und Transition. |
| SDN | SDN steht für **Software-Defined-Networking** und bezeichnet den Ansatz, ein Computernetzwerk mittels Software zu steuern[65]. Mittels SDN müssen die Netzwerkkomponenten nicht mehr manuell konfiguriert werden. Die Konfiguration wird bei SDN durch Software übernommen. |
| SPA | SPA steht für **S**ingle-**P**age-**A**pplication[66]. SPAs sind im wesentlichen Web Apps, welche als eine einzige HTML Page geladen werden können und ihren Content anschliessend dynamisch updaten, basierend auf den betätigten Benutzerinteraktionen. Ein klassisches Beispiel für eine SPA ist z.B. Gmail. |

Tabelle 1-1 Abkürzungen und wichtige Begriffe

[64] (Rational Unified Process, 2016)

[65] (Software-defined networking, 2016)

[66] (Takada, kein Datum)

# 2. Portable ToffiAnalyser - Bedienungsanleitung

Die hier aufgeführte Anleitung ist dafür vorgesehen, den ToffiAnalyser auf einem Windows Rechner schnellstmöglich zu starten. Zugleich sollte diese Anleitung eine Hilfestellung für Personen sein, welche den ToffiAnalyser für Entwicklungs- oder für Demozwecke lokal bei sich auf dem Rechner laufen lassen wollen.

Der ToffiAnalyser sollte in dem unten beschriebenen Setup nicht produktiv eingesetzt werden! Will man den ToffiAnalyser produktiv eingesetzten, sollte dazu für das Setup die Anleitung unter dem Kapitel IV 3 Installationsanleitung beachtet werden.

## 2.1 Vorbereitungen

### 2.1.1 Dateien & Ordnerstruktur

Um den ToffiAnalyser als portable Instanz starten zu können, müssen entsprechende Konfigurationdateien und das ToffiAnalyser-JAR selbst, lokal auf dem gewünschten Rechner gespeichert sein.

Der ToffiAnalyser braucht in der minimalen Konfiguration mindestens zwei Unterordner, um Dateien abzulegen. Einerseits einen Ordner für die Logs («logs») und anderseits einen Ordner, um die NetFlow-Uploads («upload») zu speichern. Diese Unterordner-Pfade können in der mitgelieferten «application.properties»-Datei angepasst werden. Standardmässig werden mit dem «Portable ToffiAnalyser» diese Ordner bereits mitgeliefert und es muss nichts unternommen werden.

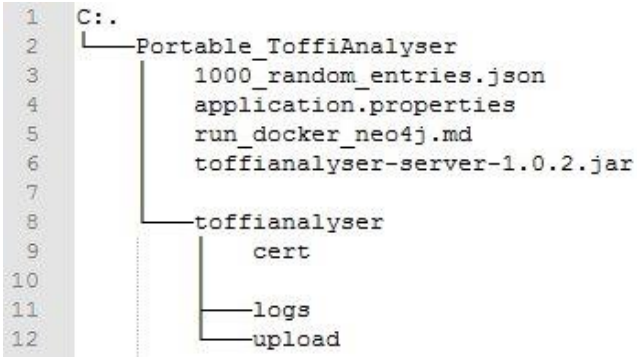


Abbildung 2-1 Benötigte Ordnerstruktur für den portable ToffiAnalyser

### 2.1.2 Neo4j Installieren und konfigurieren

Da der ToffiAnalyser die Daten in einer Neo4j Datenbank speichert, muss auf dem Rechner eine Instanz einer Neo4j Datenbank im Hintergrund laufen. Dazu kann am einfachsten die Neo4j Installationsdatei von https://neo4j.com[67] runtergeladen und anschliessend installiert werden.

Als Erstes muss nun die Neo4j Bolt Schnittstelle aktiviert werden. Dazu kann mittels des Neo4j GUIs über den Button «Options…» und den «Edit…»-Button beim Punkt «Database Configuration» komfortabel die entsprechende Konfigurationdatei geöffnet werden.

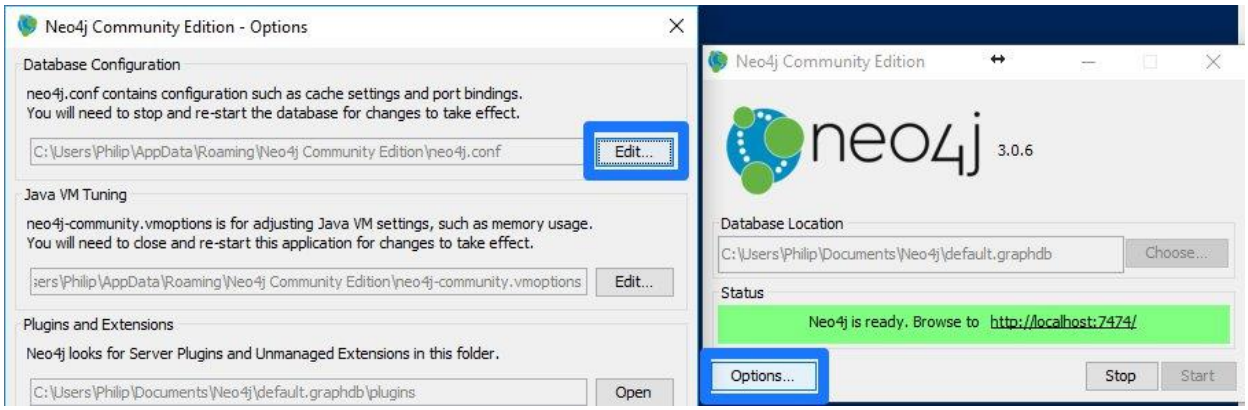


Abbildung 2-2 Öffnen der Neo4j Konfigurationsdatei

In der Konfigurationsdatei selbst, muss die Zeile «dbms.connector.bolt.address=0.0.0.0:7687» ein-kommentiert werden.

```
# To have Bolt accept non-local connections, uncomment this line:
dbms.connector.bolt.address=0.0.0.0:7687
```

Abbildung 2-3 Bolt Schnittstelle aktivieren

Damit die geänderte Konfiguration nun angezogen wird, muss über das Klicken vom «Stop»- und anschliessendem «Start»-Button Neo4j neu gestartet werden.

Nach dem Neustart von Neo4j, kann über http://localhost:7474 auf das Neo4j Webinterface zugegriffen werden. Hier muss standardmässig als Erstes das Default Passwort geändert werden. Es ist am einfachsten, das Passwort gleich auf «**toffiDB-4tw**» zu setzten, da dies das Standardpasswort von ToffiAnalyser für den Zugriff auf eine Neo4j Instanz ist. Sollte hier ein anderes Passwort gewählt werden, muss dies entsprechend in der mitgelieferten «application.properties»-Datei unter dem Wert «toffi.db.password» angepasst werden.

[67] (Neo Technology, Inc., kein Datum)

### 2.1.3 Java

Um das ToffiAnalyser-JAR ausführen zu können, muss Java auf dem Rechner installiert sein. Eine entsprechende Installationsanleitung kann auf https://java.com[68] gefunden werden. Damit in der PowerShell oder in der Eingabeaufforderung (CMD) «java» als Befehl vorhanden ist, muss sichergestellt sein, dass sich Java im Path befindet. Falls dies nicht der Fall sein sollte, kann dies gemäss offizieller Anleitung konfiguriert werden[69].

## 2.2 ToffiAnalyser starten

Sofern alle erwähnten Punkte im Kapitel IV 2.1 Vorbereitungen durchlaufen worden sind, kann nun der ToffiAnalyser mittels eines einzigen Commands gestartet werden. Am einfachsten wird hier mittels der Windows PowerShell gearbeitet – alternativ funktioniert das Command auch mit der Standard Eingabeaufforderung (CMD).

**Wichtig** ist, dass man sich mit der PowerShell oder dem CMD im «Portable_ToffiAnalyser»-Pfad befindet!

```
java -jar .\toffianalyser-server-1.0.2.jar --spring.con-
fig.location=application.properties
```

Wurde der Befehl oben abgesetzt, kann kurze Zeit später über https://localhost:8080 auf den ToffiAnalyser zugegriffen werden. Da der ToffiAnalyser standardmässig kein offiziell signiertes Zertifikat besitzt, muss im Browser entsprechend der Zugriff auf den ToffiAnalyser explizit erlaubt werden.

Nun kann mittels des Standard-Logins mit dem **Benutzernamen «admin»** und dem **Passwort «toffi@hsr»** eingeloggt werden. Sollte zuvor mindestens einer der Werte vom Property «toffi.app.admin.username» oder «toffi.app.admin.password» in der mitgelieferten «application.properties»-Datei verändert worden sein, müssen hier nun die angepassten Werte verwendet werden.

[68] (Oracle, kein Datum)

[69] (Orcale, kein Datum)

## 2.3 Daten importieren

Da nach dem initialen Start des ToffiAnalysers die Neo4j Datenbank noch leer ist, sollten nun als erstes auf https://localhost:8080/admin über die Funktion «Upload NetFlow data» NetFlow-Demodaten importiert werden. Dazu wurde eine entsprechende Datei namens «1000_random_entries.json» bereitgestellt.

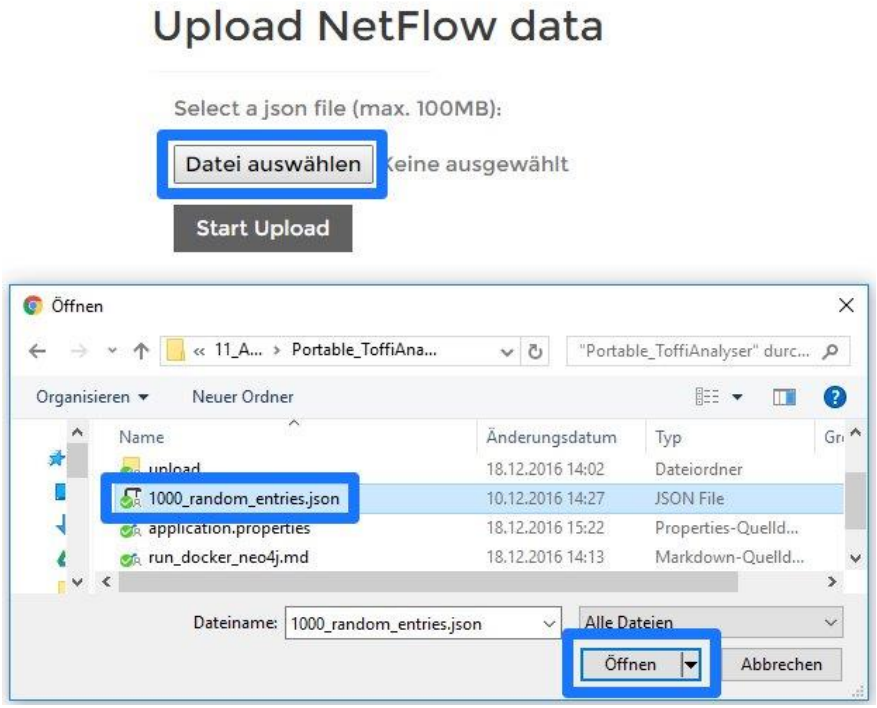


Abbildung 2-4 Demodaten Upload und Import

Wurden die Demo-Daten ausgewählt, kann der Upload über «Start Upload» gestartet werden. Nachdem dies gemacht wurde, kann der ToffiAnalyser vollumfänglich benutzt werden.

# 3. Installationsanleitung

Diese Installationsanleitung zeigt auf, wie der ToffiAnalyser produktiv in Betrieb genommen werden kann. Zugleich werden die Schritte beschrieben, mit welchen der ToffiAnalyser wieder deinstalliert werden kann.

Grundsätzlich wurde das Deployment so eingerichtet, dass möglichst wenig vom Hostsystem verändert werden muss und zugleich wurde darauf abgezielt, dass der ToffiAnalyser auf einer beliebigen Hostsystemplattform läuft. Der ToffiAnalyser und die dazu benötigte Neo4j Datenbank laufen als Docker-Container und werden somit in einer vom Hostsystem isolierten Umgebung ausgeführt.

Als Installationsdateien stehen für den ToffiAnalyser unter anderem ein Dockerfile, ein Docker-Compose File und ein Bashscript zur Verfügung.

## 3.1 Systemanforderungen

Für den ToffiAnalyser existieren die folgenden Systemanforderungen.

### 3.1.1 Betriebssystem

Als Hostsystem kann ein beliebiges Betriebssystem verwendet werden. Die einzigen Bedingungen, die das verwendete Betriebssystem erfüllen muss, sind unter Punkt IV 3.1.2 festgehalten.

Die ToffiAnalyser-Entwickler empfehlen, ein Linux System (x86-64) zu verwenden. Zugleich geht die Basis-Konfiguration der Installation von einem Linux System aus. Will der Benutzer ein Windows System als Hostsystem verwenden, müssen in den Konfigurationsdateien diverse Einstellungen verändert werden. Dies betrifft vor allem Systempfade. Der Einfachheit halber und aufgrund der grossen Verbreitung von Linux Systemen als Server, wird in dieser Installationsanleitung lediglich die Installation unter Linux behandelt.

### 3.1.2 Docker-Container & Building-Tools

Für das Builden des ToffiAnalyser-Docker-Containers, wird die folgende Software installiert auf dem Hostsystem benötigt. Besonders wichtig sind die fett markierten Versionen.

- `maven` (>=3.3.9)
- `openjdk-8-jdk` (>=1.8.0_111)
- `docker-engine` **(>=1.12.2)**[70]
- `docker-compose` (**>=1.9.0**)
- `git` (>=2.11.0)

[70] (Docker Inc., kein Datum)

### 3.1.3 Sonstige Systemanforderungen

Für die Installation und das anschliessende Betreiben des ToffiAnalysers sind zusätzlich die folgenden Anforderungen zu erfüllen:

- Es werden Root Berechtigungen gebraucht. Die Installation sollte als Benutzer Root ausgeführt werden.
- Das Hostsystem braucht eine aktive Internet Verbindung, welche mindestens Port 22 (SSH), 80 (HTTP) und 443 (HTTPS) für ausgehenden Traffic erlaubt.

## 3.2 Vorbereitungen

Für die Installation sind die folgenden Vorbereitungen zu treffen, damit die ToffiAnalyser-Software anschliessend einfach per bereitgestelltem Bash Script installiert werden kann.

- Alle unter dem Kapitel IV 3.1 Systemanforderungen aufgeführten Bedingungen müssen erfüllt sein.
- Damit die Installation-Dateien und die ToffiAnalyser-Software ohne Probleme installiert werden können, muss der SSH Public Key des Root Systembenutzers als Deployment Key bei den Git Repositories „toffianalyser_deployment“ und „toffianalyser_server_src“ hinterlegt sein. Eine Anleitung dazu kann auf https://confluence.atlassian.com[71] gefunden werden.
- In der Neo4j Docker-Container-Instanz werden die Files der eigentlichen Datenbank direkt per Docker Mount[72] auf dem Host-Filesystem abgespeichert. Dies ist standardmässig der Pfad „/opt/neo4j“. Aus diesem Grund muss vor der Installation sichergestellt werden, dass dieser Pfad existiert und der Ordner leer ist. Will der Benutzer diesen Hostsystempfad anpassen, muss dies in der „docker-compose.yml“-Datei des „toffianalyser_deployment“ Repositories gemacht werden („services“ -> „toffineo4j“ -> „volumes“).
- Vor der Installation muss sichergestellt werden, dass keine Überbleibsel einer alten ToffiAnalyser-Installation auf dem Hostsystem vorhanden sind. Es sei auf das Kapitel IV 3.4 Deinstallation verwiesen, um dies sicherzustellen.

[71] (Atlassian, kein Datum)

[72] (Docker Inc., kein Datum)

## 3.3 Konfiguration & Installation

Um den ToffiAnalyser erfolgreich auf dem Hostsystem zu installieren sind die folgenden Schritte zu durchlaufen.

1. Zu Beginn sollte in der Shell in ein Verzeichnis gewechselt werden, welches für die Installation von externen Softwarekomponenten geeignet ist. Hier wird „/opt“ empfohlen.

   ```
   cd /opt
   ```

2. Anschliessend kann mittels „git clone“ das vorbereitete „toffianalyser_deployment“ Repository geklont werden, worin sich anschliessend das Bash Installation-Script befindet.

   ```
   git clone git@bitbucket.org:toffianalyser/toffianalyser_deployment.git && \

   cd toffianalyser_deployment
   ```

3. Da mit dem vorherigen Befehl zusätzliche Dateien lokal angelegt wurden, in welchen sich Plaintext Passwörter befinden, sollten diesen restriktive Berechtigungen gesetzt werden.

   ```
   chmod 600 neo4j-credentials.env && chmod 600 toffiAnalyser/config/application.properties
   ```

4. Um die Installation abzusichern, sollte das Passwort vom Neo4j Benutzer geändert werden. Dazu müssen zwei Dateien angepasst werden. Zum einen muss das Passwort in der „neo4j-credentials.env“-Datei auf ein möglichst starkes Passwort geändert werden. Bevorzugt sollten keine anderen Sonderzeichen, als die folgend Aufgelisteten verwendet werden:

   ```
   +, -, _, = , !, *
   ```

   Zu beachten ist das Format des Inhalts:

   ```
   NEO4J_AUTH=<Neo4j_Benutzer>/<Neo4j_Passwort>
   ```

   Zum anderen muss nun dem ToffiAnalyser das geänderte Neo4j Passwort mitgeteilt werden. Dazu kann das Property „toffi.db.password“ in der Datei „toffiAnalyser/config/application.properties“ entsprechend verändert werden.
   Die ToffiAnalyser-Software besitzt standardmässig einen lokalen Administrator Account. Dieser verwendet als Username „admin“ und als Passwort standardmässig „toffi@hsr“. Nun sollte auch dieses Passwort noch auf ein selbstgewähltes und zugleich starkes Passwort geändert werden. Dieses kann mittels des Property „toffi.app.admin.password“ in der Datei „toffiAnalyser/config/application.properties“ geändert werden.

5. Zu guter Letzt muss nur noch das Bash Installation-Script ausgeführt werden damit die ToffiAnalyser-Docker-Container erstellt und anschliessend hochgefahren werden.

   ```
   chmod +x runToffiAnalyser.sh && ./runToffiAnalyser.sh
   ```

6. Der ToffiAnalyser ist rund eine Minute nach erfolgreichem Start der Docker-Container unter https://<server-ip-or-fqdn>:8080 erreichbar. Diese Zeit wird benötigt, damit im Hintergrund der ToffiAnalyser richtig gestartet werden kann.

### 3.3.1 Weitere Schritte für den produktiven Einsatz

Um den ToffiAnalyser produktiv einsetzen zu können, sollten neben dem oben erwähnten Installationsablauf auch noch die hier aufgeführten Konfigurationen betätigt werden. Alle erwähnten Properties können innerhalb der „toffiAnalyser/config/application.properties“-Datei entsprechend konfiguriert werden:

- Es sollte über die „server.ssl.*“ Properties ein offiziell signiertes Zertifikat eingebunden werden.
- Der Serverport über welchen der ToffiAnalyser-Service zur Verfügung gestellt wird, sollte auf einen üblichen Port geändert werden. Dies kann über das Property „server.port“ bewerkstelligt werden. Es empfiehlt sich TCP/443.
- Das ToffiAnalyser-Log wird aktuell lokal innerhalb des Docker-Containers unter „/var/log/toffianalyser“ gespeichert. Dieses Log sollte für den produktiven Einsatz über das Property „logging.file“ an einen Ort ausserhalb des Docker-Containers gespeichert werden. Hier wäre anschliessend auch eine Anbindung vom Hostsystem an ein Log-Collector-System wie zum Beispiel Elasticsearch[73] denkbar.
- Über das Property „logging.level.root“ sollte die Log-Intensität auf ein sinnvolles Level gesetzt werden. Hier wäre „WARN“ oder „ERROR“ denkbar. Mögliche Werte sind: DEBUG, INFO, WARN, ERROR.

[73] (Elasticsearch, kein Datum)

## 3.4 Deinstallation

Wurde nach der unter Punkt IV 3.3 Konfiguration & Installation beschrieben Installationsanleitung vorgegangen, kann die Installation durch die folgenden Schritte einfach wieder rückgängig gemacht werden. Die passenden Bash Commands dazu sind weiter unten aufgeführt und können per Copy-&-Paste in einer Shell als Root ausgeführt werden. Wurde dies gemacht, ist der Toffi-Analyser vollständig vom Hostsystem deinstalliert und alle dazugehörigen Daten wurden entfernt.

1. Die noch laufenden ToffiAnalyser-Docker-Container werden heruntergefahren.
2. Anschliessend werden die Docker-Container und die Docker-Images gelöscht.
3. Zuletzt werden die sich noch auf der Disk befindenden Dateien endgültig gelöscht.

**Achtung:** Mit dem letzten Command (Nr. 7) wird die gesamte Neo4j Datenbank gelöscht. Somit sind sämtliche darin gespeicherten NetFlow-Daten endgültig gelöscht!

Bash Commands für die Deinstallation:

```
cd /opt/toffianalyser_deployment \
&& docker-compose down \
&& docker rmi toffianalyserdeployment_toffianalyser \
&& docker rmi neo4j:3.1 \
&& docker rmi openjdk:8u111-jre \
&& rm -rf toffiAnalyser/ \
&& rm -rf toffianalyser_deployment/ \
&& rm -rf /opt/neo4j
```

# 4. NetFlow Konfiguration

Die Testdaten für diese Semesterarbeit wurden uns freundlicherweise vom INS Institute for Networked Solutions der HSR zur Verfügung gestellt. Dieses Kapitel wurde aus Gründen der Vollständigkeit und Verständlichkeit hier im Anhang hinzugefügt.

Die unten aufgeführte Konfiguration wurde uns als Textdokument bereitgestellt. Es gilt folge dessen klar festzuhalten, dass während der Zeit für die Semesterarbeit keinerlei Exporter/Collector Konfiguration vorgenommen wurde und es sich somit nicht um unsere eigene Arbeit handelt.

## 4.1 Exporter (Core Switch)

Als Erstes muss auf dem Switch ein Flow Record definiert werden. Dieser dient dazu zu definieren, welche Packet Attribute als Identifizierung für einen Flow gebraucht werden und welche Attribute zusätzlich mitgesammelt werden sollen.

Da auf dem Core Switch (Cisco Nexus 6000) vom INS bereits ein NetFlow Standard Flow Record definiert ist, wird gleich dieser verwendet. Falls ein solcher Standard Flow Record noch nicht existieren würde (je nach Device), müsste dieser erst erstellt werden. Der Standard Flow Record trackt folgende Informationen:

```
flow record netflow-original

  description Traditional IPv4 input NetFlow with origin
ASs

  match ipv4 source address

  match ipv4 destination address

  match ip protocol

  match ip tos

  match transport source-port

  match transport destination-port

  match input interface

  match output interface

  match flow direction

  collect routing source as

  collect routing destination as
```

```
  collect routing next-hop address ipv4
  collect transport tcp flags
  collect counter bytes
  collect counter packets
  collect timestamp sys-uptime first
  collect timestamp sys-uptime last
```

Damit nun NetFlow-Daten an den Collector geschickt werden können, muss ein Exporter konfiguriert werden:

```
flow exporter netflow-sa-logstash
  description export netflow data to the netflow-sa
logstash docker container
  destination 10.20.0.10 use-vrf student
  transport udp 9995
  source Vlan319
  dscp 0
  version 9
    template data timeout 180
```

`source vlan 319` besagt, auf welchem VLAN Interface (SVI) die NetFlow-Daten verschickt werden sollen. Im vorliegenden Fall ist der NetFlow Collector im VLAN 319 am Core Switch angeschlossen, weshalb das SVI dieses VLAN's gewählt wurde.

Nun wird ein Monitor erstellt, welcher den Flow Record auf den Exporter bindet:

```
flow monitor MonitorNetFlowSA
  record netflow-original
  exporter netflow-sa-logstash
```

Weil mit konfiguriertem NetFlow jedes einzelne Packet neben dem Routing Processing auch noch durch das NetFlow Processing muss, macht es Sinn, mittels eines Sampling die CPU Last auf dem Switch zu reduzieren. Ein NetFlow Sampler erlaubt es, nur jedes x-te Packet durch das NetFlow Processing zu schicken. Gewisse Switches und Router können gar nur NetFlow in Kombination mit Sampling. Beim INS Cisco Nexus 6000 Core Switch ist dies zum Beispiel auch der Fall.

```
sampler SampleStudentVRF
  mode 1 out-of 64
```

Zuletzt, muss noch auf den gewünschten VLAN Interfaces (SVIs) der Monitor definiert werden, sodass von diesem VLANs Netzwerkpakete mittels NetFlow analysiert werden und an den Collector geschickt werden:

```
interface Vlan122
  ip flow monitor MonitorNetFlowSA input sampler SampleStu-
dentVRF


interface Vlan319
  ip flow monitor MonitorNetFlowSA input sampler SampleStu-
dentVRF
```

## 4.2 Collector (LogStash Server)

Um die NetFlow-Daten zu sammeln, wird ein Collector benötigt, welcher die NetFlow Packets entgegennimmt, parst und ein einem brauchbaren Format abspeichert. Im INS wurde hierfür extra ein Docker-Container[74] erstellt. Dieser Docker-Container wird über ein auf Github gehostetes Dockerfile[75] automatisch gebuilded. Der relevanteste Teil des Docker-Containers, steckt in der LogStash Konfiguration.

```
input {
  udp {
    port => 9995
    codec => netflow {
      netflow_definitions => "/srv/netflow.yaml"
      versions => [5,9,10]
    }
```

[74] (INS, 2016)

[75] (Schmid, 2016)

```
    }
  }
  filter {
    json {
      source => "message"
    }
  }
  output {
    stdout { codec => rubydebug }
    file {
      path => "/data/netflow_data.json"
      create_if_deleted => true
    }
  }
```

Mittels dieser Konfiguration wird der Input auf dem UDP Port 9995 akzeptiert, welcher im NetFlow-Format encoded daherkommt. Anschliessend wird der NetFlow-Packet-Inhalt als JSON abgespeichert. Dies sind dann die Testdaten, welche uns das INS für unsere Semesterarbeit zur Verfügung gestellt hat.

# 5. Beispiel-Report

Der aufgeführte Beispielreport enthält anonymisierte Daten, welche alle zusammen summiert an einem Tag verschickt oder empfangen wurden. Er umfasst das gesamte HSR-Netz mit der Regular-Expression «`152\.96\..*`».

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

# ToffiReport - 01.07.2016 00:00 to 01.07.2016 23:59

| IP address | Hostname | Total bytes |
|---|---|---|
| 152.96.79.133 | null | 604.75 kB |
| 152.96.174.199 | null | 604.75 kB |
| 152.96.116.223 | null | 198.21 kB |
| 152.96.176.67 | null | 198.21 kB |
| 152.96.218.244 | null | 157.89 kB |
| 152.96.150.199 | null | 157.89 kB |
| 152.96.223.70 | null | 99.42 kB |
| 152.96.62.240 | null | 99.42 kB |
| 152.96.35.253 | null | 98.24 kB |
| 152.96.75.72 | null | 98.24 kB |
| 152.96.11.210 | null | 48.33 kB |
| 152.96.144.198 | null | 48.33 kB |
| 152.96.206.105 | null | 43.66 kB |
| 152.96.231.68 | null | 43.66 kB |
| 152.96.226.0 | null | 18.84 kB |
| 152.96.73.10 | null | 18.84 kB |
| 152.96.255.110 | null | 14.58 kB |
| 152.96.195.142 | null | 14.58 kB |
| 152.96.148.148 | null | 10.59 kB |

Dezember 14, 2016 Page 1 of 3

# ToffiReport - 01.07.2016 00:00 to 01.07.2016 23:59

| IP address | Hostname | Total bytes |
|---|---|---|
| 152.96.117.194 | null | 10.59 kB |
| 152.96.92.4 | null | 6.27 kB |
| 152.96.124.210 | null | 6.27 kB |
| 152.96.4.113 | null | 2.84 kB |
| 152.96.95.226 | null | 2.84 kB |
| 152.96.4.169 | null | 1.63 kB |
| 152.96.231.140 | null | 1.63 kB |
| 152.96.84.231 | null | 1.28 kB |
| 152.96.102.54 | null | 1.28 kB |
| 152.96.223.185 | null | 256 B |
| 152.96.199.221 | null | 256 B |
| 152.96.132.76 | null | 252 B |
| 152.96.73.48 | null | 252 B |
| 152.96.14.169 | null | 240 B |
| 152.96.187.252 | null | 240 B |
| 152.96.183.213 | null | 232 B |
| 152.96.139.129 | null | 232 B |
| 152.96.8.53 | null | 220 B |
| 152.96.114.31 | null | 220 B |
| 152.96.30.61 | null | 200 B |
| 152.96.129.50 | null | 200 B |
| 152.96.238.247 | null | 165 B |

Dezember 14, 2016 Page 2 of 3

# ToffiReport - 01.07.2016 00:00 to 01.07.2016 23:59

| IP address | Hostname | Total bytes |
|---|---|---|
| 152.96.101.199 | null | 165 B |
| 152.96.56.227 | null | 158 B |
| 152.96.22.42 | null | 158 B |
| 152.96.240.2 | null | 134 B |
| 152.96.17.111 | null | 134 B |
| 152.96.115.10 | null | 104 B |
| 152.96.206.178 | null | 104 B |
| 152.96.46.202 | null | 104 B |
| 152.96.138.209 | null | 104 B |
| 152.96.234.43 | null | 76 B |
| 152.96.13.16 | null | 76 B |

Dezember 14, 2016 Page 3 of 3

# 6. Systemtest-Spezifikation



**ToffiAnalyser – Systemtestspezifikation**

**Version 1.0**

***Autor:***
*Matthias Gabriel und Philip Schmid*

Revision 1, Dezember 2016
Bezugsquelle: https://eprints.hsr.ch

HSR Hochschule für Technik Rapperswil
Oberseestrasse 10
8640 Rapperswil
Schweiz



[1] https://creativecommons.org/licenses/by-nc-sa/3.0/ch/legalcode.de

# Inhaltsverzeichnis

# 1. Einführung

## 1.1 Zweck

Dieses Dokument erläutert die Systemtests des Projektes ToffiAnalyser.

## 1.2 Gültigkeitsbereich

Das Dokument ist gültig während der Entwicklung der Semesterarbeit ToffiAnalyser und allfälligen Nachfolgeprojekten



# 2. Systemtests

Die Systemtests sollen die Richtigkeit des ToffiAnalysers in Bezug auf die definierten Funktionalitäten sicherstellen, welche in der Software-Projektdokumentation definiert sind.

## 2.1 Systemtests Netzwerkverkehr-Informationen bereitstellen

### 2.1.1 Voraussetzungen

Diese Voraussetzungen müssen erfüllt sein, damit die Tests durchgeführt werden können.

- Der ToffiAnalyser muss laufen und eine gültige Neo4j-Instanz muss verwiesen sein
- Der Benutzer muss über ein gültiges JSON-File mit NetFlow Daten verfügen
- Der Benutzer ist im ToffiAnalyser eingeloggt

### 2.1.2 Testablauf Import von JSON Daten

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Admin Bereich navigieren | 1. Menupunkt Admin auswählen | Das Admin Interface mit der Möglichkeit zum Upload von Daten erscheint |
| 2 | Daten importieren | 1. Klick auf den Button Datei auswählen | Die Möglichkeit eine Datei auszuwählen erscheint |
| | | 2. Gewünschte JSON-Datei auswählen | Der Dateidialog schliesst sich und die Applikation zeigt den Namen der ausgewählten Datei an |
| | | 3. Start Upload Button drücken | Der Upload wird durchgeführt und ein “Spinner” eingeblendet. Nach dem erfolgreichen Upload erscheint eine grüne Bestätigung, welche aufzeigt wieviele NetFlows importiert wurden |
| 3 | Datenimport verifizieren | a) Verifizierung über eine Ansicht des ToffiAnalysers | Die importierten Daten werden im ToffiAnalyser dargestellt |
| | | b) Alternativ: Verifizierung über die Neo4j-Datenbank | Die importierten Datensätze sind in der Datenbank vorhanden |

### 2.1.3 Testablauf Netzwerkgerät manuell benennen

Voraussetzung: Bereits importierte Daten sind verfügbar.

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Admin Bereich navigieren | 1. Menupunkt Admin auswählen | Das Admin Interface mit der Möglichkeit zum Upload von Daten erscheint |
| 2a | Benennung eines Netzwerkgeräts | 1. IP-Adresse des gewünschten Netzwerkgerätes im Filterfeld eingeben | Nur noch das zutreffende Netzwerkgerät wird eingezeigt |
| | | 2. Änderung des Namens durch Eingabe des neuen Namens.<br>Es können auch mehrere Netzwerkgerätenamen auf einmal gespeichert werden | Der neue Name wird angezeigt. |
| | | 3. Speicherung des Namens durch Klick auf den Button Save changes | Bei der erfolgreichen Speicherung erscheint eine grüne Bestätigung mit der Information wieviele Namen gespeichert wurden |
| 3 | Verifizierung der geänderten Daten | a) Verifizierung über eine Ansicht des ToffiAnalysers | Die geänderten Namen werden in den Ansichten des ToffiAnalysers angezeigt |
| | | b) Alternativ: Verifizierung über die Neo4j-Datenbank | Die Netzwerkgerätenamen sind auch in der Datenbank angepasst |

## 2.2 Systemtests Generelle Graph-Features

Alle drei verschiedenen Graph-Ansichten haben in gewissen Teilbereichen die gleichen Features. Um diese nicht mehrmals beschreien zu müssen, sind diese in diesem separaten Kapitel erfasst.

### 2.2.1 Voraussetzungen

Diese Voraussetzungen müssen erfüllt sein, damit die Tests durchgeführt werden können.

- Der ToffiAnalyser muss laufen und eine gültige Neo4j-Instanz muss verwiesen sein
- Es müssen bereits NetFlow Daten im ToffiAnalyser importiert sein
- Der Benutzer ist im ToffiAnalyser eingeloggt

### 2.2.2 Testablauf Zoom

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Bereich navigieren | 1. gewünschter Menupunkt auswählen | Das gewünschte Interface erscheint |
| 2 | Zoomen im Graphbereich | 1. Betätigen des Mausrades, wenn diese über der Graphen-Fläche liegt. | Der Graph wird kleiner oder grösser je nach Richtung der Mausrad-Drehung. |
| | | 2. Betätigen des Mausrades, wenn diese ausserhalb des Graphen (z.B. Menu) liegt. | Es tritt keine Änderung ein. |

### 2.2.3 Testablauf Verschiebung des Graphen

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Bereich navigieren | 1. gewünschter Menupunkt auswählen | Das gewünschte Interface erscheint |
| 2 | Graph verschieben | 1. Klicken und halten der linken Maustaste auf die leere Graphenfläche | Verschiebung des Graphen ist möglich |

### 2.2.4 Testablauf Verschiebung und Fixierung eines Nodes

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Bereich navigieren | 1. gewünschter Menupunkt auswählen | Das gewünschte Interface erscheint |
| 2 | Node verschieben | 1. Klicken und halten der linken Maustaste auf einen Node | Verschiebung des Nodes ist möglich.<br>Der Node wird mit einem roten Rand versehen<br>Der gesammte Graph verschiebt sich mit dem Node |
| | | 2. Klicken und halten der linken Maustaste auf einen zweiten Node | Verschiebung des Nodes ist möglich.<br>Der Node wird mit einem roten Rand versehen<br>Der erste Node bleibt an Ort und Stelle<br>Die anderen Nodes ordnen sich normal an |
| 3 | Node freigeben | 1. Doppelklicken mit der linken Maustaste auf einen fixierten Node (rot umrandet) | Der fixierte Node wird freigeben<br>Der Node fügt sich wieder im Graphen ein<br>Die rote Umrandung verschwindet |

### 2.2.5 Testablauf Tooltip über Node

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Bereich navigieren | 1. gewünschter Menupunkt auswählen | Das gewünschte Interface erscheint |
| 2 | Tooltip anzeigen | 1. Maus über einem Node platzieren | Einblendung eines Tooltips unterhalb des Nodes<br>Der Tooltip enthält ansichtsspezifsiche Daten über den Node |
| 3 | Tooltip ausblenden | 1. Maus vom Node entfernen | Ausblendung des Tooltips |

### 2.2.6 Testablauf Tooltip über Verbindung

Dieses Feature ist in der Netflow-Ansicht nicht verfügbar.

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Bereich navigieren | 1. gewünschter Menupunkt auswählen | Das gewünschte Interface erscheint |
| 2 | Tooltip anzeigen | 1. Maus über einer Verbindung platzieren | Einblendung eines Tooltips unterhalb der Verbindung<br>Der Tooltip enthält ansichtsspezifsiche Daten über die Verbindung |
| 3 | Tooltip ausblenden | 1. Maus von der Verbindung entfernen | Ausblendung des Tooltips |

## 2.3 Systemtests Filter

Alle drei mometan verfügbaren Graphansichten verfügen über verschiedene Filter. Diese sind aber abhängig von der aktiven Ansicht. In diesem Kapitel wird jeder Filter separat beschrieben um eine Redundanz zu verhindern. Gewisse Filter sind in einigen Ansichten erst nach der Aktivierung der Option «Show advanced filters» verfügbar. Dies ist jeweils im Kapitel vermerkt. Wenn ein Filter zu stark einschränkt kommt eine gelbe Mitteilung, welche besagt, dass keine Daten gefunden werden können.

### 2.3.1 Voraussetzungen

Diese Voraussetzungen müssen erfüllt sein, damit die Tests durchgeführt werden können.

- Der ToffiAnalyser muss laufen und eine gültige Neo4j-Instanz muss verwiesen sein
- Es müssen bereits NetFlow Daten im ToffiAnalyser importiert sein
- Der Benutzer ist im ToffiAnalyser eingeloggt
- Der Benutzer hat bereits eine der Ansichten geöffnet, für welche der Filter zur Verfügung steht

### 2.3.2 Testablauf Datums-Filter

Die Datums-Filter sind in den folgenden Ansichtern verfügbar:

- Overview
- Netflow-Overview (Advanced)
- Netflow (Advanced)

| | | | |
|---|---|---|---|
| 1 | Datums-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Datums-Filter ausfüllen | Beim Ausfüllen der Datums-Filter erscheint ein Datumsdialog, welcher die Auswahl erleichtert. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden nur Daten zwischen Begin- und End-Datum angezeigt. |

### 2.3.3 Testablauf Limit-Filter

Die Limit-Filter ist in den folgenden Ansichtern verfügbar:

- Overview
- Netflow-Overview (Advanced)
- Netflow (Advanced)

| | | | |
|---|---|---|---|
| 1 | Limit-FFilter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Limit-Filter ausfüllen | Der Limit-Filter kann entweder über eine Tastatureingabe oder über die Pfeile gesetzt werden. Eine negative Eingabe ist nicht möglich.<br>Wenn ein Limit gesetzt wird erscheint die zusätzliche Filtermöglichkeit Skip. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden maximal die Anzahl an NetFlow-Verbindungen angezeigt, welche in Limit festgelegt sind. |

### 2.3.4 Testablauf Skip-Filter

Der Skip-Filter ist in den folgenden Ansichtern verfügbar:

- Overview
- Netflow-Overview (Advanced)
- Netflow (Advanced)

| | | | |
|---|---|---|---|
| 1 | Skip-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Limit-Filter ausfüllen | Der Limit-Filter kann entweder über eine Tastatureingabe oder über die Pfeile gesetzt werden. Eine negative Eingabe ist nicht möglich.<br>Wenn ein Limit gesetzt wird erscheint die zusätzliche Filtermöglichkeit Skip. |
| | | 2. Skip-Filter ausfüllen | Der Skip-Filter kann entweder über eine Tastatureingabe oder über die Pfeile gesetzt werden. Eine negative |

| | | | |
|---|---|---|---|
| | | | Eingabe ist nicht möglich. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden maximal die Anzahl an NetFlow-Verbindungen angezeigt, welche in Limit festgelegt sind.<br>Durch den Skip-Filter können eine gewisse Anzahl von Resultaten übersprungen werden. |

### 2.3.5 Testablauf Src IP-Filter

Der Src-IP-Filter ist in den folgenden Ansichtern verfügbar:

- Netflow-Overview
- Netflow

| | | | |
|---|---|---|---|
| 1 | Src IP-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Src IP-Filter ausfüllen | Im Src IP Filter kann eine IP-Adresse eingeben werden. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden nur NetFlows angezeigt, welche als Source IP-Adresse die eingebene Adresse besitzen. |

### 2.3.6 Testablauf Dst IP-Filter

Der Dst IP-Filter ist in den folgenden Ansichtern verfügbar:

- Netflow-Overview
- Netflow

| | | | |
|---|---|---|---|
| 1 | Dst IP-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Dst IP-Filter ausfüllen | Im Dst IP Filter kann eine IP-Adresse eingeben werden. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden nur NetFlows angezeigt, welche als Destination IP-Adresse die eingebene Adresse besitzen. |

### 2.3.7 Testablauf Protocol-Filter

Der Protocol-Filter ist in den folgenden Ansichtern verfügbar:

- Netflow-Overview
- Netflow

| | | | |
|---|---|---|---|
| 1 | Protocol-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. gewünschter Protocol Button anklicken | Der gewünschte Protocol Filter wird blau markiert.<br>Es kann nur ein Protocol Filter gleichzeitig ausgewählt werden. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden nur NetFlows angezeigt, welche dem gewählten Protocoltypen entsprechen. |

### 2.3.8 Testablauf Dst Port Filter

Der Dst Port-Filter ist in den folgenden Ansichtern verfügbar:

- Netflow-Overview (Advanced)
- Netflow (Advanced)

| | | | |
|---|---|---|---|
| 1 | Dst Port-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Dst-Port-Filter durch klicken auf die Checkbox aktivieren | Der Dst Port-Filter wird aktiviert |
| | | 2. Dst Port-Filter ausfüllen | Der Dst Port-Filter kann entweder über eine Tastatureingabe oder über die Pfeile gesetzt werden. Eine negative Eingabe ist nicht möglich.<br>Die Maximale Eingabe ist 65535 |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Es werden nur NetFlows angezeigt, welche den eingebenen Dst Port besitzen. |

### 2.3.9 Testablauf Direction Filter

Der Dst Port-Filter ist in den folgenden Ansichtern verfügbar:

- Netflow-Overview (Advanced)
- Netflow (Advanced)

| | | | |
|---|---|---|---|
| 1 | Direction-Filter setzten | 1. Klick auf den Button Filters | Der Dialog mit den Filterungsmöglichkeiten erscheint |
| | | 2. Two-way Button anklicken | Der Two-way Button wird blau markiert. |
| | | 3. Search Button drücken | Es wird eine neue Anfrage an den ToffiAnalyser mit dem neuen Filter geschickt und die erhaltenen Daten dargestellt.<br>Der Direction-Filter beeinflusst die Filter Src IP und Dst IP, indem er auch den entgegengesetzten Verkehr darstellt. |

## 2.4 Systemtests Übersicht des Netzwerkverkehrs

### 2.4.1 Voraussetzungen

Diese Voraussetzungen müssen erfüllt sein, damit die Tests durchgeführt werden können.

- Der ToffiAnalyser muss laufen und eine gültige Neo4j-Instanz muss verwiesen sein
- Es müssen bereits NetFlow Daten im ToffiAnalyser importiert sein
- Der Benutzer ist im ToffiAnalyser eingeloggt

### 2.4.2 Testablauf Overview darstellen

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Übersichts Bereich navigieren | 1. Menupunkt Overview auswählen | Das Overview Interface erscheint und lädt die NetFlows.<br>Zwischen zwei Nodes existiert jeweils lediglich eine Verbindung.<br>Die Dicke der Verbindung ist abhängig von der Anzahl der übertragenen Daten im Vergleich zu den anderen dargestellten Verbindungen. |

HSR Hochschule für Technik Rapperswil
**Systemtestspezifikation**

## 2.5 Systemtests Netzwerkverkehr analysieren

### 2.5.1 Voraussetzungen

Diese Voraussetzungen müssen erfüllt sein, damit die Tests durchgeführt werden können.

- Der ToffiAnalyser muss laufen und eine gültige Neo4j-Instanz muss verwiesen sein
- Es müssen bereits NetFlow Daten im ToffiAnalyser importiert sein
- Der Benutzer ist im ToffiAnalyser eingeloggt

### 2.5.2 Testablauf Netflow-Overview

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Übersichts Bereich navigieren | 1. Menupunkt Netflow-Overview auswählen | Das Netflow-Overview Interface erscheint und lädt die NetFlows.<br>Es wird pro Destination Port und Richtung zwischen zwei Nodes eine Verbindung angezeigt.<br>Die Dicke der Verbindung ist abhängig von der Anzahl der übertragenen Daten im Vergleich zu den anderen dargestellten Verbindungen.<br>Die Richtung des Flows wird über Pfeile innerhalb der Verbindung angezeigt.<br>Die Verbindungen sind je nach Eigenschaften unterschiedlich eingefärbt. |

### 2.5.3 Testablauf Netflow

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Übersichts Bereich navigieren | 1. Menupunkt Netflow auswählen | Das Netflow Interface erscheint und lädt die NetFlows.<br>Es wird jeder Flow einzeln als Node angezeigt.<br>Die Richtung des Flows wird über Pfeile angegeben.<br>Die Netflow Nodes sind je nach Eigenschaften unterschiedlich eingefärbt.<br>Die Netflow Nodes enthalten als Beschriftung den Destination Port. |

## 2.6 Systemtests Route des Netzwerkflows analysieren

Auf die Beschreibung von Systemtests für die Userstory Route des Netzwerkflows analysieren wurde verzichtet, da diese im Verlauf der Semesterarbeit nicht umgesetzt wurde.

## 2.7 Systemtests Reports genererien

### 2.7.1 Voraussetzungen

Diese Voraussetzungen müssen erfüllt sein, damit die Tests durchgeführt werden können.

- Der ToffiAnalyser muss laufen und eine gültige Neo4j-Instanz muss verwiesen sein
- Es müssen bereits NetFlow Daten im ToffiAnalyser importiert sein
- Der Benutzer ist im ToffiAnalyser eingeloggt

### 2.7.2 Testablauf Netflow-Overview

| Nr. | Titel | Handlung | Reaktion |
|---|---|---|---|
| 1 | Zum Reporting Bereich navigieren | 1. Menupunkt Reporting auswählen | Das Reporting Interface erscheint<br>Standardmässig wird der letzte Monat als Reportingzeitraum ausgewählt |
| 2 | Reporting regex | 1. Eintragen der gewünschten Regular-Expression auf welche die Netzwerkgeräte zutreffen müssen um im Report aufgenommen zu werden | |
| 3 | Datum eintragen | 1. Die Zeitspanne über welche der Report ausgewertet werden soll auswählen | Beim Ausfüllen der Datums-Filter erscheint ein Datumsdialog, welcher die Auswahl erleichtert. |
| 4 | Report herunterladen | 1. Betätigen des Download Buttons | Der Report wird mit den entsprechenden Daten generiert und als PDF-Datei heruntergeladen<br>Die PDF-Datei ist im standardmässig im Download-Ordner des Browsers zu finden und heisst toffiEndpointReport.pdf |

# 7. Bibliographie

Android Development Team. (kein Datum). *Dependency Injection - ts- GUIDE*. Abgerufen am November 2016 von https://angular.io/docs/ts/latest/guide/dependency-injection.html#!#dependency-injection-tokens

Atlassian. (kein Datum). *Atlassian Bitbucket Pipelines*. Abgerufen am 15. Dezember 2016 von https://confluence.atlassian.com/bitbucket/bitbucket-pipelines-792496469.html

Atlassian. (kein Datum). *BitBucket: Add deployment Key*. Abgerufen am Oktober 2016 von https://confluence.atlassian.com/bitbucket/use-deployment-keys-294486051.html

Auth0® Inc. (kein Datum). *Single Sign on & Token Based Authentication*. Abgerufen am 17. Dezember 2016 von https://oauth.net/2/

*CI Continuous integration*. (17. Dezember 2016). Abgerufen am 17. Dezember 2016 von https://en.wikipedia.org/wiki/Continuous_integration

Cisco. (kein Datum). *Freeware NetFlow Software*. Abgerufen am November 2016 von http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/networking_solutions_products_genericcontent0900aecd805ff72b.html

*Cloud computing*. (5. Dezember 2016). Abgerufen am 17. Dezember 2016 von https://en.wikipedia.org/wiki/Cloud_computing

d3.js. (17. Dezember 2016). *D3.js Gallery*. Von https://github.com/d3/d3/wiki/Gallery abgerufen

DIGITALOCEAN ™ INC. (kein Datum). *DigitalOcean Product Overview*. Abgerufen am 17. Dezember 2016 von https://www.digitalocean.com/products/

Docker Inc. (15. Dezember 2016). *Docker Hub*. Von https://hub.docker.com/ abgerufen

Docker Inc. (kein Datum). *Docker Installationsanleitung für Linux*. Abgerufen am 18. Dezember 2016 von https://docs.docker.com/engine/installation/linux/

Docker Inc. (kein Datum). *Docker Mount*. Abgerufen am 18. Dezember 2016 von https://docs.docker.com/engine/tutorials/dockervolumes/

Elasticsearch BV. (kein Datum). *elastic.coelastic.co*. Abgerufen am 16. Oktober 2016 von elastic.co: https://www.elastic.co/products/logstash

Elasticsearch BV. (kein Datum). *LogStash: Centralize, Transform & Stash Your Data*. Abgerufen am 17. Dezember 2016 von https://www.elastic.co/products/logstash

Elasticsearch. (kein Datum). *Elasticsearch*. Abgerufen am 18. Dezember 2016 von https://www.elastic.co/

Fowler, M. (16. November 2011). *PolyglotPersistance*. Abgerufen am Dezember 2016 von http://martinfowler.com/bliki/PolyglotPersistence.html

Fox, T. (16. Oktober 2016). *digitalocean*. Von digitalocean: https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-orientdb-on-ubuntu-14-04 abgerufen

Gordon, D. (15. Dezember 2016). *Neo4j Transaction Delete Best Practices*. Von https://neo4j.com/developer/kb/large-delete-transaction-best-practices-in-neo4j/ abgerufen

GraphAlchemist. (September 2016). *GraphAlchemist/Alchemy*. Von github.com: https://github.com/GraphAlchemist/Alchemy abgerufen

INS. (12. Oktober 2016). *Docker Hub*. Abgerufen am 12. Oktober 2016 von Docker Hub: https://hub.docker.com/r/pschmid/docker-logstash-netflow/

jacomyal. (3. Nomveber 2016). *jacomyal/sigma.js: A JavaScript library dedicated to graph drawing*. Abgerufen am November 2016 von github.com: https://github.com/jacomyal/sigma.js

Keller, P. S. (8. Oktober 2012). *Datenbanksysteme im HS11 für Master (MSE)*. Von Benchmarking und Performance von (R)DBMS: http://wiki.hsr.ch/Datenbanken/wiki.cgi?SeminarDatenbanksystemeHS1112 abgerufen

Keller, P. S. (9. Februar 2014). *HSR Texas Geo Database Benchmark*. Von https://giswiki.hsr.ch/HSR_Texas_Geo_Database_Benchmark abgerufen

Kollegger, A. (4. Dezember 2015). *Bolting Forward: The Neo4j 3.0 Milestone 1 Release Is Here*. Abgerufen am Dezember 2016 von Neo4j Blog: https://neo4j.com/blog/neo4j-3-0-milestone-1-release/

mbostock. (17. Dezember 2016). *D3.js*. Von https://d3js.org/ abgerufen

mbostock. (September 2016). *d3/d3: Bring data to life with SVG, Canvas and HTML.* Von github.com: https://github.com/d3/d3 abgerufen

mbostock. (Dezember 2016). *d3-3.x-api-reference/Force-Layout.md*. Von https://github.com/d3/d3-3.x-api-reference/blob/master/Force-Layout.md abgerufen

Nagios. (kein Datum). *Nagios Network Analyzer. Netflow Analysis and Monitoring*. Abgerufen am September 2016 von https://www.nagios.com/products/nagios-network-analyzer/#faqs

Neo Technology, Inc. (kein Datum). *Neo4j Ubuntu Installationsanleitung*. Abgerufen am 16. Oktober 2016 von http://debian.neo4j.org/

Neo Technology, Inc. (kein Datum). *Chapter 3. Cypher - The Neo4j Developer Manual v3.0*. Abgerufen am Oktober 2016 von https://neo4j.com/docs/developer-manual/current/cypher/#query-where-regex

Neo Technology, Inc. (kein Datum). *Intro to Cypher*. Abgerufen am September 2016 von Intro to Cypher: https://neo4j.com/developer/cypher-query-language/

Neo Technology, Inc. (kein Datum). *Language Guides*. Von neo4j.com: https://neo4j.com/developer/language-guides/ abgerufen

Neo Technology, Inc. (kein Datum). *Neo4j Installation on Windows*. Abgerufen am 18. Dezember 2016 von https://neo4j.com/download/community-edition/

Neo Technology, Inc. (kein Datum). *Neo4j Transaction Logs Handling*. Abgerufen am 15. Dezember 2016 von https://neo4j.com/docs/operations-manual/current/performance/transaction-logs/

ntop. (kein Datum). *ntopng - ntop*. Abgerufen am September 2016 von http://www.ntop.org/products/traffic-analysis/ntop/

OpenShift Origin. (kein Datum). *The Open Source Container Application Platform*. Abgerufen am 17. Dezember 2016 von https://www.openshift.org/

Oracle. (kein Datum). *How to install Java*. Abgerufen am 18. Dezember 2016 von https://java.com/en/download/help/download_options.xml

Orcale. (kein Datum). *How to add Java to Path*. Abgerufen am 18. Dezember 2016 von https://www.java.com/en/download/help/path.xml

OrientDB LTD. (kein Datum). *OrientDB Blueprints API*. Abgerufen am 09. Dezember 2016 von OrientDB Blueprints API: http://orientdb.com/docs/2.2/Tutorial-Java.html

OrientDB LTD. (kein Datum). *OrientDB Performance Tuning*. Abgerufen am 09. 12 2016 von OrientDB Performance Tuning: http://orientdb.com/docs/master/Performance-Tuning-Graph.html

Paessler. (kein Datum). *NetFlow Analyzer & Monitoring Tool - PRTG*. Abgerufen am September 2016 von https://www.paessler.com/netflow_monitoring

pcwdld.com. (kein Datum). *13 Free Open Source NetFlow Analyzers for Windows and Linux/Unix*. Abgerufen am November 2016 von http://www.pcwdld.com/free-open-source-netflow-analyzers

pgodzin. (Dezember 2016). *Model of all methods annotated with responseContainer = "List" incorrectly begins with Inline Model*. Von https://github.com/swagger-api/swagger-ui/issues/2325 abgerufen

pratapvardhan. (14. September 2016). *Gallery d3/d3 Wiki*. Abgerufen am September 2016 von https://github.com/d3/d3/wiki/Gallery

*Rational Unified Process*. (20. Oktober 2016). Abgerufen am 17. Dezember 2016 von https://en.wikipedia.org/wiki/Rational_Unified_Process

Schmid, P. (1. Juli 2016). *docker-logstash-netflow*. Abgerufen am 12. Oktober 2016 von docker-logstash-netflow: https://github.com/PhilipSchmid/docker-logstash-netflow

sigmajs.org. (kein Datum). *Sigma.js*. Abgerufen am September 2016 von http://sigmajs.org/#tutorial

Slack Technologies. (kein Datum). *Slack*. Abgerufen am 15. Dezember 2016 von https://slack.com/is

SocialSensor. (12. Januar 2016). Abgerufen am September 2016 von https://github.com/socialsensor/graphdb-benchmarks

*Software-defined networking*. (16. Juni 2016). Abgerufen am 17. Dezember 2016 von https://de.wikipedia.org/wiki/Software-defined_networking

SonarSource SA. (kein Datum). *SonarQube*. Abgerufen am 17. Dezember 2016 von SonarQube: http://www.sonarqube.org/

Spring.io. (kein Datum). *24. Externalized Configuration*. Abgerufen am Dezember 2016 von http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

Spring.io. (kein Datum). *Spring Boot*. Abgerufen am Dezember 2016 von https://projects.spring.io/spring-boot/

springfox. (kein Datum). *SpringFox by springfox*. Abgerufen am November 2016 von http://springfox.github.io/springfox/

stackoverflow. (kein Datum). *Posts containing 'sigmajs' - Stackoverflow*. Abgerufen am September 2016 von http://stackoverflow.com/search?q=Sigmajs

Takada, M. (kein Datum). *Single page apps in depth*. Abgerufen am 17. Dezember 2016 von http://singlepageappbook.com/goal.html

tomwanzek. (11. Juli 2016). *Request for d3.js API update from v3. to v4.1.0*. Abgerufen am September 2016 von https://github.com/DefinitelyTyped/DefinitelyTyped/issues/9936

Vlaswinkel, K. (13. Februar 2014). *digitalocean*. Abgerufen am 16. Oktober 2016 von digitalocean: https://www.digitalocean.com/community/tutorials/how-to-install-java-on-ubuntu-with-apt-get

Weinberger, C. (13. Oktober 2015). *Benchmark: PostgreSQL, MongoDB, Neo4j, OrientDB and ArangoDB*. Abgerufen am Oktober 2016 von https://www.arangodb.com/wp-content/uploads/2015/09/neighbor_v207.png

Wikipedia. (15. Dezember 2016). *Salt (cryptography)*. Von Wikipedia.org: https://en.wikipedia.org/wiki/Salt_(cryptography) abgerufen

Winkler, C. (23. April 2013). *Office-Helden: Genial Kombination von Fussnote und Literaturverzeichnis in Word - Blog Microsoft Deutschland*. Abgerufen am 28. Oktober 2016 von https://blogs.technet.microsoft.com/microsoft_presse/office-helden-geniale-kombination-von-funote-und-literaturverzeichnis-in-word/